

# Trust-aware Decentralized Recommender Systems: PhD research proposal

Paolo Massa

Department of Information and Communication Technology - University of Trento  
Via Sommarive 14 - I-38050 Povo (TN) - Italy  
E-mail: massa@itc.it

29th May 2003

## Abstract

This PhD thesis addresses the following problem: exploiting of trust information in order to enhance the accuracy and the user acceptance of current Recommender Systems (RS). RSs suggest to users items they will probably like. Up to now, current RSs mainly generate recommendations based on users' opinions on items. Nowadays, with the growth of online communities, e-marketplaces, weblogs and peer-to-peer networks, a new kind of information is available: rating expressed by an user on another user (trust). We analyze current RS weaknesses and show how use of trust can overcome them. We proposed a solution about exploiting of trust into RSs and underline what experiments we will run in order to test our solution.

## 1 Introduction

*“Although an application designer’s first instinct is to reduce a noble human being to a mere account number for the computer’s convenience, at the root of that account number is always a human identity”* [24].

We are in the Information society. The quantity of new information available every day (news, movies, scientific papers, songs, websites, ...) goes over our limited processing capabilities. For this reason, we need something able to suggest us only the worthwhile information. Recommender Systems (RS) [34, 39] have this aim. In particular, RSs based on Collaborative Filtering (CF) [16, 7] try to automate the “word of mouth” process. The intuition is the following: when we have to decide about going to see a new movie for example, we often ask to some friends with similar movies tastes and then we act based on their recommendations. CF tries to automate this process to a world scale: there is no more need the users asks to known people but it is the system (that knows the judgement of everyone) that finds users similar to her and recommends to her the items they like.

However, RSs based on CF have some weaknesses: cold start problem [40] is related to the situation when an user enters the system and has expressed no ratings so that the RS cannot find the like-minded users; any-

way in general the quantity of ratings an user gives is very low compared to the quantity of available items (for example, Eachmovie [29] dataset is 97.4% sparse), this results in a great sparseness of the data and low confidence of the automated strategies. Another concern is related to the existence of users that want to maliciously influence the system: if they know how the recommendations are built they can express fake ratings in order to achieve a certain goal (such as get their book recommended to everyone). There are some recent study of possible attacks in distributed systems [26, 22] but we are not aware of research lines taking into account this as a problem for RSs. Moreover, with current RSs it is very hard (or impossible) for the user to control the recommendation process so that if the RS starts giving bad quality recommendations, usually the user just stops in using it [46, 17].

Nowadays, online communities (slashdot.org, afro.org), e-marketplaces (ebay.com, amazon.com, epinions.com, ...), peer-to-peer networks (edonkey, gnutella) and weblogs use and make available trust information (judgement expressed by users on users). They use it with slightly different specific goals but what they share is the fact this introduce a sort of social control over the all system.

We claim taking into account explicit trust information provided by users into Recommender Systems can overcome their inherent, previously cited weaknesses.

We also claim this environment reclaims a decentralized approach: the only possible way to be in control of the recommendation process is being able to do it by yourself; in this sense, it does not make sense having one centralized server as the only entity able to access the information expressed by users and to generate recommendations. In order to have a real control over the recommendation process, every single entity (we call it peer referring to the peer-to-peer architecture) must self-publish the data (ratings and trusts) so that it is possible for every peer to fetch all the information and self-computes recommendation on behalf of its user. Self-publishing is made very easy by recent tools such as weblogs. We think decentralized data publishing is important for research too, in fact we claim research in RS was slow and not successful as possible also because of

lack of available datasets and real testbeds for testing innovative solutions.

We propose a strategy to exploit trust information in order to enhance current RSs. Our proposed solution is a multi step one where different algorithms, caring for different types of input data, can be combined with a special emphasis to the confidence they have in the correctness of predicted information.

We also present some experiments we will run in order to test our hypothesis. Some experiments will be on synthesized data, while some experiments will be on real online communities. With this regard, we are adding weblog functionalities to a classical music recommend system: CoCoA<sup>1</sup> [5]. In this case, our goal is to have a system running with real users (about 1400 daily now) that can express what are the bloggers they trust and what are the classical tracks they like in which we will be able to test our proposed solution against a community of real users.

The structure of the proposal is the following.

In section 2, we present the past work relevant for our research line. The chosen subject is quite interdisciplinary and so we refer to different topics such as Recommender Systems (section 2.1), Reputation-aware systems and Trust metrics (section 2.2), Peer-to-Peer and distributed architecture (section 2.3), and Weblogs and Semantic Web (section 2.4)

Section 3 provides the motivating context, i.e. the reasons we claim make sense and innovative to exploit trust information with the goal to create better Recommender Systems.

In section 4 we outline what are the research problems arisen in the new context while in section 5 we clearly indicate the research problems we address along with our proposed solution.

We present the experiments we plan to run in order to evaluate our proposed solution in section 6. Finally in section 7 we present the work already done and a detailed road map for what must be done, while in section 8 we indicate what we leave as future work.

## 2 State of the Art

In this section, we present the past work relevant for our research line. The chosen subject is quite interdisciplinary and so relevant papers come from many different fields. First, we analyze Recommender Systems, automated tools that promise to give a solution to information overload. Then we concentrate on systems that take explicitly into account trust and reputation, analyzing how this information is currently used. We then present peer-to-peer and distributed architectures because it is in this environment that trust information can really unveil its power being every peer free to behave as it prefers. Last, we analyze weblogs that are a recent and very interesting Internet trend. They can be seen as the first, real instantiation of the Semantic

Web, a web understandable by the machines. We think a great amount of machine readable data created daily in a distributed manner reclaims some mechanisms such as trust metrics and Recommender Systems in order to bring a little bit of order.

### 2.1 Recommender Systems

Acting upon recommendations from other people is a normal part of life. We do it when we eat at restaurant on the advice of a friend, or we see a movie having read the review in the newspaper of our choice. In each case our decision to act upon a recommendation is based on essentially three premises: first, we trust the recommender; second, we assume that the recommender has sufficient knowledge of our tastes or of the tastes of people like us; third, we assume that the recommender has knowledge of the alternatives available. By using recommendations we can take a shortcut to the things we like without having to try many things we dislike or without having to acquire all the knowledge to make an informed decision. Unsurprisingly, systems that automate this facility have become popular on the Internet.

Recommender systems (RS) [34, 39], in fact, have been used to suggest movies, books, songs, jokes, etc. They have been an important research line because they promise to fulfill the e-commerce dream: a different and personalized store for every single (potential) customer.

Two main algorithmic techniques have been used to compute recommendations: content-based and Collaborative Filtering (CF). The first one tries to suggest to the user items similar to the ones she liked in the past. To do this, it needs a representation of the items in term of features: they can be some automatically extracted features such as words frequency for text items or some human edited features such as genre for movies. The CF [16, 7] approach is more interesting and innovative. The recommender asks users to rate items so that it knows who likes what. Then, when asked for a recommendation for the current user (recommendee), it identifies users similar to her (her neighbours) and it suggests her the items the neighbours have liked in past. The interesting point is that the algorithm doesn't need a representation of the items in term of features but it is based on the tastes of its users' community. The best systems are hybrid in the sense they combine the two approaches [40]. Strengths and weaknesses of content-based and Collaborative Filtering RSs will be explored in section 3.

A RS based on CF can acquire users' opinions about items in two ways: explicit, it asks the user to rate some items or implicit, it infers users' tastes from their usage activity, for example, it assumes the user likes a book if she buys it.

One problem the research on RS must face is the difficulty in evaluating algorithms, due to lack of really meaningful metrics and of precise statement of the problem faced. In [20], we proposed an on-line evaluation framework. Another related problem is the scarcity of public datasets of real ratings on items for offline experiments and the total absence of publicly usable running

---

<sup>1</sup>The running RS can be found at: <http://cocoa.itc.it>

Recommender Systems for online tests.

An interesting application for RS is in the music domain: we developed a RS for suggesting classical music compilations [5]. The running application can be found at <http://cocoa.itec.it>. We also proposed recommendation strategies to adapt Internet radio programs on the fly [6].

Interesting possibilities still unexploited for recommender algorithms come from social network analysis by which it is possible to spot out what are the neighbours of an user by her actions. Location-aware computing also offer new frontiers for RS [9], in fact, information about the current location of the user opens new interesting possibilities for recommendation algorithms.

Lately, there have been some attempts to move CF in a distributed environment. John Canny in “Collaborative Filtering with privacy” [10] criticizes the centralized approach in which all the user data resides on a central server and proposes an alternative model in which users control all their log data. He also describes an algorithm whereby every single user of the community can compute a public “aggregate” of their data that does not expose individual user’s data. Then every single peer can use this aggregate to compute recommendations. The proposed algorithm uses factor analysis (a technique of dimensionality reduction similar to singular value decomposition but more efficient) and cryptographic techniques (asymmetric encryption). The key idea is homomorphic encryption to allows sums of encrypted vectors to be computed and decrypted without exposing individual data.

With this regard, another interesting papers is “Collaborative Filtering In A Distributed Environment: An Agent-Based Approach” [19]. The authors have proposed a system called iOwl to exchange metadata about web surfing activity between peers. They have also created a usable application downloadable at <http://www.iowl.net>. Essentially, a modified browser records user’s clickstreams and data mining techniques extract profile data, such as usual navigation patterns. This metadata are exchanged with other peers and are used to self compute recommendations of possibly interesting URLs.

## 2.2 Reputation-aware systems and trust metrics

Nowadays, with the emergence of online communities, e-marketplaces, weblogs and peer-to-peer communities, a new kind of information is available: rating expressed by an user on another user. We call this information trust.

Moreover, there are examples of communities that are no more only virtual and online but have started to produce important effects in the real world: for example, ebay.com is the cause of a big movement of goods and money in the real world even if it’s nothing but a virtual community. Other examples of no-more-so-virtual communities are weblogs (the blogosphere), auction sites (ebay.com, yahoo auction, amazon auction and many more), slashdot.org, affero.org, epinions.com, peer-to-peer networks (edonkey network,

reputation-aware gnutella network). In the future with the spread of mobile, pervasive and ubiquitous computing this information will become even more available and useful and concrete.

We can ask ourselves “Is this information useful?”. The answer is a great yes. For instance, ebay.com [36] a site devoted to online auctions, allows traders to rate their partners after a commercial transaction; in this way, it can compute and show to users the reputation as a seller and buyer of every member and this simple number near the usernames is what enables big economic movements to happen in the real world between people geographically distant, who never met before and who will never meet in future. This is the main reason for the great (economic) success of this dotcom.

Other examples are slashdot.org, the “news for nerds” site where everyone can post news story and rate other users depending on posts they submit; here reputation is used to keep noise-to-signal ratio low and to give special emphasis to interesting posts; affero.org is a system that exploits trust elicitation in order to democratically and distributedly decide which open source projects are the more promising for the community and worth funding.

There are some attempts to build reputation-aware systems also on top of current P2P networks: on eDonkey network with the open source client eMule (<http://emule.sourceforge.net>) and in reputation-aware Gnutella servents [11]. Trust has been introduced with the goal of blocking and spotting out leeches (peers who only download without sharing) and misrupters (peers who pollute the network with faked items).

[35] provides a complete analysis of most of the current existing reputation systems.

Trust is not a strange or artificial concept. If you are member of a mailing list, surely you have experienced the following: the topic of the mailing list is of interested for you but while you are eager of reading mails from some members (you value/trust them a lot!), you don’t want to be bothered by mails of some other members/spammers/disrupters (you totally distrust them!). Tools that can help you in saying how much you trust any member will enable services personalizing for you the access to relevant information minimizing the quantity of bad quality content you have to process. The same can be said for newspapers, journalists, politicians, musicians, and people in general.

There are many different definitions of trust and related concepts such as reputation and reciprocity. This shouldn’t surprise since trust is a very human and social concept and it is in the speculation of men since the first philosophers. Moreover these concepts have been studied by researchers and thinkers in different fields other than computer science such as economics (game theory), scientometrics (or bibliometrics), evolutionary biology, anthropology, philosophy and sociology [31].

We will clearly state our definition of trust and related concepts in section 5.2.

For now, we just point out how, in computer science literature, trust is intended mainly as a diadic quantity

involving two peers: “a subjective expectation an agent has about another’s future behaviour” [31] while reputation is mainly seen as property of a peer assigned to her by the embedded social network and computed from the many trust relationships: “reputation is the memory and summary of behaviour from past transactions” [32] (chapter 17). Anyway they can be seen as two sides of the same concept and are often used as synonyms.

Many persons believe the world of future will be based on reputation: reputation will become the only “currency” of our life [37]. Also one of the sci-fi 2003 bestsellers explores the topic: in “Down and out in the magic kingdom” [12], Cory Doctorow envisions a near-future realistic world that’s seen “the death of scarcity”, where nanotechnology takes care of everyone’s basic needs and there is no more need for money. Instead, what the population aspire to is “Whuffie”, a sort of reputation capital representing the approval and respect of your peers.

Marsh [28] was the first to introduce with his PhD thesis a computational model for trust in the distributed artificial intelligence community and there are some attempts to scientifically understand what trust and reputation concepts can represent for computer science [1, 45, 23, 22, 2, 31] but it is worth noting how the research is very recent and how most of the proposed reputation systems for online communities have been constructed with an intuitive approach and without a formal model and a deep learning from approaches of the social sciences. Often these papers just provide some possible, reasonable intuitions but without solid ground or motivations; sometimes there aren’t even tests on simulated data and in general no evaluation with real communities. Researching in this direction is indeed very needed in order to discover potentialities and possible problems and to design systems that can get full advantage of this information.

It is of course not possible to build a direct reputation relationship with every other peer, so it is important to share judgements about other peers. Sen et al. in [41] demonstrate that cooperating agents who share their opinions about other agents perform better (i.e. maximize individual utility) than selfish ones who don’t collaborate.

Some trust metrics have been proposed: among others, Advogato [26] and Fionna [25].

It is important to underline how all these metrics are simple and intuitive and that more advanced ones can be built only evaluating them in real, used systems. With this regard, the more interesting projects are: NewsMonster and BlogNet<sup>2</sup>, two OpenPrivacy projects working with weblogs and news channels.

Many of these systems compute a global reputation value for every single peer, with this regard they are very similar to PageRank [33], the algorithm used in Google.com for deciding the importance of an Internet page. We will see some weaknesses of this approach in section 3.

<sup>2</sup><http://newsmonster.org> and <http://peerfear.org/blognet>

## 2.3 Peer-to-Peer and distributed architecture

The topic is very broad. In this research proposal, we only briefly mention why it is a so hot subject nowadays, which are the most interesting trends and how this is relevant for our research.

From an historical point of view [42] up until 1994, Internet’s architecture was peer-to-peer (P2P), in fact machines were assumed to be always on, always connected, and with a permanent IP address. With the invention of Mosaic, another model began to spread. To run a web browser, a PC needed to be connected to the Internet over a modem, with its own IP address. But PCs were not always connected and so didn’t have a fixed IP address. This prevented PC users from hosting any data or net-facing application locally. This was the client-server model: servers were powerful machines always connected and with a fixed IP address while PCs were used only as dumb client for occasional web browsing. Over time, as hardware and software improved, treating PCs only as clients starts to look as a waste and a new paradigm started becoming the solution: P2P.

P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet [42].

Three primary classes of P2P applications have emerged: distributed computing (SETI@Home), content sharing and collaboration (Groove Networks). There are many examples of P2P networks for content sharing: Napster, Gnutella, Kazaa, Freenet, Chord, Neurogrid, Alpine, Mojonation, Mnet, Overnet, Semplesh, Kademia, PlanetP, Pastry, Tapestry, CAN, just to name a few of them<sup>3</sup>. There are also examples of frameworks for P2P communication: Jxta and Jabber for instance<sup>4</sup>.

Decentralized architectures offer some great opportunities but present also some problems especially because of their open and autonomous nature. Every peer is in fact free to behave (i.e. following the protocol) as it prefers. If the source code is available than everyone can just change the standard behaviour modifying the code; for example, this happened for Gnutella [27]. If the protocol and the source code is secret, everyone can just reverse-engineer the protocol and write his own peer with a different (maybe better, maybe worst) behaviour [44].

In general, in a P2P network it is not possible to rely also on availability of other peers, on file authenticity and there is no central control of the network.

There is a big debate about P2P: it is positive or it is negative? Two opposite approaches exist and can be summarized as the “tragedy of the commons” [18] and the “cornucopia of the commons” [8]. The commons are some goods owned by the whole community and not belonging to everyone in particular. Who supports the first approach states that commons suffers of free riding

<sup>3</sup>Decentralized Meta-Data Strategies at [http://neurogrid.net/Decentralized\\_Meta-Data\\_Strategies\\_neat.html](http://neurogrid.net/Decentralized_Meta-Data_Strategies_neat.html)

<sup>4</sup><http://jxta.org> and <http://jabber.org>

phenomena<sup>5</sup>: everyone tries to increase her utility from them, so overuses them even without a real need and this leads shortly to total consumption and unavailability for everyone. The other approach instead says that if the effort in contributing to create a resource (an up to date repository of informations, for example) is very low, than new resources can be created by the spontaneous work of all the users of a system and made available to everyone. Moreover if the resource is not a consumable one (as it is the case with bits) there is only increase of available resources.

In the following we will use the term P2P in a more relaxed ethimological sense: we will consider a peer every logical independent entity that can be identified in a unique way and is able to expose some information. To be clear, we'll consider a weblog (see subsection 2.4) as a logical peer.

How P2P architectures are relevant for our research? We have seen that in a decentralized environment every peer is free to behave as she prefers. Anyway one important way to incentive good behaviours is the possibility for peers to be rewarded or punished, essentially to enable the explicitation of trust (see section 2.2).

In systems like ebay.com, every peer is free to behave badly, for example to not send the item she has already received the payment for. Instead peers almost always behave correctly because the opinions of other peers contribute to create their reputation in the system and reputation is a value because it allows to make trade and to get better prices. The same can be seen on slashdot.org where having a good reputation is equal to be recognized as a guru by the community or in affero.org where receiving many trust statements means you are valuable for the open source community (i.e. a skilled hacker) and this means you could find a very interesting job easily. We can say that in a decentralized environment that takes into account trust, there are incentives to well behave. This incentives allows to "harness the power of disruptive technologies" [32] such as peer-to-peer systems lowering down the inherent risks of these distributed environment.

## 2.4 Weblogs and Semantic Web

A very interesting phenomena of the last years in Internet are weblogs (often contracted in blogs).

They are a sort of online diary, a frequently updated web site arranged chronologically, very easy to create and maintain that doesn't require knowing HTML or programming. It provides a very low barrier entry for personal web publishing and so many millions of people in the world maintain their own blog and post on it daily thoughts.

Their relevance is confirmed by the following facts: in February 2003, Google bought Pyra Labs, a company that created some of the earliest technology for writing weblogs and its website, Blogger.com; Stanford and Harvard are promoting their use among their students as a valuable mean of publishing of research ideas and results.

<sup>5</sup>Studies have claimed Gnutella is affected by free riding [3]

The technology is incredibly simple but has some disruptive characteristics. Weblogging tools create the standard HTML file for human browsing but also some semantically well defined XML files that have the great advantage of being machine understandable. For example, together with the standard *index.htm* file there is *index.rss* (and often *index.xml*); these files are expressed in RSS (Rich Site Summary) format<sup>6</sup>.

By providing a summary of articles recently posted to a website, this format allows receipt of categorized information, the collection of which is automated and can be read, searched and followed up at any time, also with programs different from a web browser. All the weblogs publish information in RSS format but also big traditional media do it, CNN.com for instance.

Every weblog can so be parsed by machines and in fact can be syndicated and aggregated via centralized services (<http://daypop.com>, <http://weblogs.com>, <http://blo.gs>, etc.).

Moreover, links between blogs and items (so called blog rolling) support the decentralized construction of a rich information network (called blogosphere).

To give an idea of the disruptive potential of this simple technique and its wide adoption let us look at <http://www.allconsuming.net>, a site where you can know what the blogging community is reading at the moment. The functioning of the system is simple: a crawler gets the blogs list from <http://weblogs.com> and then starts parsing all of them extracting every URL containing an ISBN (and a pointer to Amazon.com or other online book repository). Then it aggregates this data and show the books mentioned in the last hour or week or month.

This is an example of what Jon Udell calls "Manufactured Serendipity". "Serendipity is all about making fortunate discoveries by accident. You can't automate accidental discoveries, but you can manufacture the conditions in which such events are more likely to occur. Blogs are all but that." (from <http://www.intertwingly.net/stories/2002/03/13/manufacturedSerendipity.html>).

Some bloggers have started expressing other kinds of information with XML files. There is foaf.xml (Friend-Of-A-Friend<sup>7</sup>) in which you can state who are your friends, people you trust. There is smbmeta.xml<sup>8</sup> by which small and medium enterprises can express their physical location, the area they serve, their type of business, etc. There is XFML<sup>9</sup>, a simple XML format for exchanging metadata in the form of taxonomies and many more.

In a way, weblogs are the real instantiation of semantic web. Semantic web envisions a web of machine processable information: "The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users... The Semantic Web is not a separate Web

<sup>6</sup><http://backend.userland.com/rss>

<sup>7</sup><http://xmlns.com/foaf/0.1/>

<sup>8</sup><http://www.trellixtech.com/smbmetaintro.html>

<sup>9</sup><http://xfml.org>

but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation” (from <http://w3.org/2001/sw/>).

It is everything about converging on some XML standard formats and succeeding in getting them widely used. Nowadays, weblogs seems the only possible walkable way in this direction.

Weblogs are important for our research line because they can be the empowering tool fostering the availability of an always up to date distributed datasets of correlations among peers (trust) and with items (ratings). We will see in section 3 how one factor that slowed down research in RS was the unavailability of public datasets and testbeds. Weblogs promise to fill this gap.

### 3 Motivations and impact

We have already seen the great potential behind Recommender Systems (RS). In particular we have examined the simple but very effective intuition at the base of Collaborative Filtering (CF) that is automating the “word of mouth” process. Anyway RSs (especially based on CF) have some weaknesses we will examine in a short.

Essentially, we can say that CF automates the process too much forgetting the social dimension of the environment. It doesn’t take into account what are the opinions of people about other people but it simply tries to predict them based on similarity about how they rate items. We claim that taking into account direct judgement of users on users (trust) can enhance RSs performances.

An analysis of the concept of trust and its use has already been done in section 2.2. We have outlined how many current online communities make use of the concept of trust in order to have a decentralized control over the system. It is worth noting again how many of these systems compute a global value of trust for every peer (for instance, ebay.com shows shows near every username a star of different colors where a color means a certain level of reputation in the community computed on the basis of every positive and negative feedback given by users after every transaction). We claim this is a nonsense because an user can be valuable for one peer and not for another peer. We claim only personalized and subjective computation of reputation can be really useful and not attackable. We use the term “indirect trust” to indicate a predicted trust to emphasize how this value has only meaning as a relation among two peers and not as a global value equal for everyone.

In the following we will continue using the term “peer” intending a logical independent entity that can be identified in a unique way and is able to expose some information fetchable by other peers.

Let come back to our starting point: Recommender Systems weaknesses; in the following we will examine them one by one explaining also how trust-awareness can help in overcoming them.

*Content-based RSs require human editors to tag and*

*classify items.* Content-based RSs tries to suggest to the user items similar to the ones she liked in the past. To do this, they need a representation of the items in term of features. This representation can be extracted automatically from items whenever it is possible: for example, for read news, it is possible to extract the words an user has interested in simply by parsing the text. Even if this methods often doesn’t capture the essence of the tastes of an user, they are used because very easy to implement. Content-based RSs have more problems when items cannot be parsed by machines: for example, it is impossible or very difficult nowadays for a machine to extract meaningful features such as genre, authors from a song or a movie file. In this case we need human beings to tag and classify items. This introduces a lot of problems: first, it is not easy to decide which are the right features we have to tag (genre, instruments, year, ...) and tagging is an expensive, boring, error-prone, subjective activity. Moreover for some items such as jokes is almost impossible to find what the right features are.

Our approach is to use only information provided by the user: ratings on items and trusts on users. In this way our resulting system is totally independent of the specific items features that are not taken into account and can be applied to whatever domain.

*Confidence in computed User Similarity is often very low.* The information expressed by user about who and how much she trusts (“PeerA trusts PeerB 0.9”) is considered to be correct information, with total confidence. Instead the information “peerA is similar to peerB 0.8” is computed by the system. The confidence in the correctness of this value depends on the quantity of information that is available and the algorithm that is used. We will define better confidence in section 5.2, for now we use it for indicate the reliability of a computed value, the degree of certainty the system has in it.

If we consider every user as a vector of ratings on items and we put them in a matrix, the resulting matrix users  $\times$  items, the traditional input of Collaborative Filtering techniques, is very very sparse (for example, Eachmovie [29] dataset is 97.4% sparse). This is totally normal, infact, if we need something to filter out bad items or to suggest interesting ones, this means there are many items we are not going to try and, consequently, to rate. This results in our “profile” consisting of ratings of few items over a huge set. This is an inherent weakness of systems relying only on users rating items and this is even esacerbated in the early phases of RS life cycle when really few ratings are available.

In general, sparseness means that overlapping of profiles of 2 users (i.e. the number of items they both rated) is very low and their user similarity often not computable or anyway computable with low confidence. Singular Value Decomposition has been proposed as a solution for dimensionality reduction and consequent reduction of sparseness [38] but it is not clear if when you start with a so sparse matrix, it is really the amount of information available that is too less. Another proposed solution are hybrid systems [40, 17], in which we have to

rely also on features based on content, especially in an initial phase.

Another problem resulting in low confidence in computed User Similarity is the so called cold start problem. It arises when a new user enters in a new domain where, clearly, she has expressed no ratings. In this case, CF cannot compute similarity and find neighbours and compute recommendations. It is even possible she is not able or doesn't want to rates items in this domain (let's take for example a computer scientist wanting to start reading "black holes" physics books and knowing nothing about them). However, in a social setting (as Internet at broad is), the "new" user probably has (or can easily find) some trustworthy, valuable peers; then these explicitly stated valued neighbours will guide the recommendation process that she has always under control with the possibility of expressing trust on new discovered peers, to rate recommended items, to calibrate her own tastes and to change every previously expressed statement.

Summarizing, we claim that expressing a trusted peer is easier and more natural for users and allows an easy bootstrap of the system; direct trust information has total confidence and is more reliable than computed user similarity. Moreover trust has some inherent transitivity properties (if A trusts B and B trusts C, A will probably trust C). Considering the "Six Degrees of Separation" folklore axiom<sup>10</sup>, we can claim that a small number of trust statements can easily cover a big portion of the social network if we exploit trust transitivity and propagate trust. In the case of an undirect trustable peer, the confidence is no more total but it is anyway a very useful information in order to reduce significantly sparseness.

There can be the case when user similarity and trust contradict each other; we will try to understand better what this means in the following, anyway taking into account also trust should be better than just relying on user similarity.

*CF techniques are attackable by malicious users.*

CF takes into account every peer in the same way. In this sense it has no means at all to discover malicious peers. For this reason, malicious peers that knows how the algorithm works can easily exploit it in order to influence the created recommendations. For example, let suppose a malicious user wants the RS to recommend ItemSpam to PeerA: she can create FakePeer, copy the profile of PeerA and add a good rating to ItemSpam. In this way the RS will find FakePeer as the most similar to PeerA and recommend ItemSpam to it.

We think it is very important to take into account attacks in a distributed system where every peer is free to behave as it prefers. There is some research about this in P2P systems [26, 22] but we know of no papers pointing out this problem for RSs.

Moreover in a distributed virtual systems, the fact that it is easy to create infinite new identities means that you cannot trust a peer unless you have grown a positive (direct or indirect) trust in her [14].

---

<sup>10</sup>In 1967, the Harvard Social Psychologist Stanley Milgram tried to establish mail connections among random people. The average number of steps in a successful chain was around 6. [30]

It is important to underline how also systems that compute a global value of trust can be easily faked: for example, this can be done in ebay.com when a person creates a lot of fake peers (ebay identities) that rate her as a very good trading partner. This allows the malicious user to gain a global high reputation and then to use it in a single, rich fraud transaction in which she don't fulfill her obligations stealing the money without sending the agreed good. This attack have been studied in [22].

In our vision, exploiting of trust information allows to be influenced only (or mainly) by "trustable" peers, either direct peers or indirect ones (friends of friends). This can reduce the user base used to find neighbours but surely keeps out malicious and fake peers. The sharing of opinions about peers is also a good way for detecting or spotting out misbehaving peers.

*CF techniques are difficult to control by the user.*

As long as RSs give good results, everything is fine, but when they start recommending badly, it is very difficult for the user to understand why and to fix the problem; at the end the user usually quits using the RS [46, 17].

Up to now, RSs are black boxes, i.e. the user receive the recommendations but doesn't know how it was generated and has no control in the recommendation process. For example, in [21], the authors conducted a survey with real users and found that users want to see how recommendations are generated, how their neighbours are computed and how their neighbours rate items. Also [43] that analyzes RSs from a Human Computer Interaction perspective finds that RSs are effective if, among other, "the system logic is at least somewhat transparent".

Even if the RS exposes what it thinks of you (explicit or implicit past ratings on items) and allows the user to modify them, this is a very complicated, hard task (imagine yourself re-examining 1000 ratings about books and correcting the wrong ones!) [46]. It has been claimed that "few of the amazon users revise their profiles" when the recs start becoming obviously wrong [17].

Moreover, in order to allow the user to control and correct the recommendation process, RSs should expose to the user the internal recommendation process and let the user correct possible errors. For example, presenting the predicted trust value of some users (the neighbours) and giving the user the possibility to correct them. In order to this be possible, RSs should expose as well info about the peers (her blog, her ratings on items, her photo, etc.) that the user can use to decide what the real trust value can be.

Infact, usually RSs don't recommend peers but just items, they jump a step going directly to the items space: doing so, they fail in helping in finding the right person (and this is often what a user wants). CF automates the process of recommending items but doesn't help in putting in touch with like minded people for community forming.

Indeed, trust-aware RSs can easily also recommend persons (possible trustable peers) and show the reasons behind such a recommendation (the social network) and, in doing so, allow community forming. By seeing peers,

users are encouraged to judge them and provide a trust value in them allowing their social network to grow and to become more precise and reliable.

Moreover, acting directly modifying explicit trust on users (and the following social network) is easier and gives more control to user than acting on explicit rating of items. Anyway this is a claim and needs to be demonstrated (see experiments in section 6).

*RS up to now are centralized servers.*

While trust-awareness can be done inside a single centralized server, centralized approaches of data collection in general suffer of some huge disadvantages. Expressing information (what you like, who you like) in a centralized RS server means that only that server will be able to use it. We know that a private company has no advantage in sharing it and making it public because it is part of its enterprise value.

This results in users profiles being scattered in many different, not cooperating servers (for example, users preferences about books are stored in amazon.com, barnesandnobles.com, gorilla.it, . . . ); every single server suffers even more of sparseness and has problems in giving good quality recommendations. Moreover, this means the user cannot move from one RS to another without losing her profile (and, with it, the possibility to receive good recommendations and to save time). In essence, this situation is against competition and can easily lead to global monopoly because it is almost impossible for new RSs to enter the market while, for consolidated RS owning much user information, it is even possible to enter new correlated markets. Moreover, with a centralized approach, it is the server that control your personal data and decide about privacy policies; you cannot control your data.

This is true also for trust information.

Clearly, companies prefer to not allow interoperability or publicity of this information because it is their company value. Ebay.com for instance doesn't allow other auctions websites (not even big companies like amazon.com) to copy "their" reputation values and it's fighting law battle against competitors that have started downloading them and including them in their own sites [32]. Anyway, until this useful information will remain confined in narrow marketplaces, it will not inveil all its disruptive potential power. For example, if you specify what are your "trusted peers" about books in amazon.com and another online books seller (such as barnesandnobles.com) could have made an interesting for you use of this information, you are in a sense slowing down progress.

It make no sense to introduce all this trust concerns and then let some commercial RS to make the computation with not even a possible concurrency. We know that a commercial RS has some contradicting goals: for example, it wants to provide good recommendations but it also wants to recommend you items that are available at its stores or with a greate revenue margin.

We claim that a recommendation computation can be under user control only if the user has the possibility of making self-recommendations (disposing of all the data).

The solution is already visible on the web and it is straightforwardly simple: every user publishes this valuable information on her peer, on her site, under her control and then whichever service wants to use it in order to provide clever services could just fetch it. This is happening in weblogs (see section 2.4) where many bloggers publish on their site their blogosphere friends (the blogs they find interesting and read regularly), the favourite books, movies, songs lists, the currently listening lists, the currently reading lists, their physical location, etc.

Now these new technologies (along with bookmarklets<sup>11</sup> for example) allow decentralized and easy publishing of semantic information and the Internet can be seen (or can start to be seen) as a unique, huge, always updated database of relationships.

We want also to point out how research in RSs has long suffered this overwhelming problem: lack of datasets and real testbeds on which to apply and test new hypothesis (imagine the difference if all the world researchers could have had access to data of amazon.com, barnesandnoble.com, etc. and could have tested their new hypothesis on them). In fact there were only two freely available datasets of ratings on items, Eachmovie [29] and Movielens (available at <http://www.cs.umn.edu/Research/GroupLens/>) and they were used for offline testing but, in order to run online experiments, researchers had to invest a lot of time into creating their own RS and gathering enough users. This is not an easy task: Grouplens working group of University of Minnesota<sup>12</sup> is a notable exception in this.

Summarizing, we claim that users' judgements on items and users, the important information used by Recommender Systems, shouldn't be locked in the hard disk of some server so that competition is not possible but they should be self-published by every peer.

In this distributed environment, every peer becomes the recommender of its user. In fact, peers self-publish their "profiles" and can actively fetch other peers' profiles so that every single peer has enough relevant information to give recommendations to its user. In general, we assume that a peer has enough memory (to record peers' profiles) and enough computational power (to calculate recommendations); if it is not the case, a peer can also ask for recommendations to a different peer, possibly a peer specialized in processing information and giving specially clever recommendations.

## 4 Research Problems: Open issues

In the environment envisioned at the end of section 3 a good number of interesting issues arise. We will summarize briefly them here. Then, in the next session we will state which are the focuses of our work and which are left for future work.

<sup>11</sup>They are javascript links that can be saved on bookmark lists and works on webpages. See <http://bookmarklets.com>

<sup>12</sup><http://www.cs.umn.edu/Research/GroupLens>

Trust is inherently transitive (if A trusts B and B trusts C, there is some probability A trusts C). It is interesting to study how it is possible to propagate trust in a social network in order to predict some level of indirect trust on unknown peers. An open issue is related to a deep understanding of what trust represents in an online community, especially considering the negative values that don't seem to be very used in real communities [36].

Another open problem is to understand what are the relationships between the concept of trust (either direct or indirect) and user similarity in the context of a distributed Recommender System and to study how this information can be exploited in order to increase performances. In particular it is interesting to analyze how the information can be combined considering the different levels of confidence the system has in it.

Another interesting topic is related to explanation of recommendations to the user that can be achieved by showing what were the items influencing the recommendation, who were the peers taken into account and how they rated items and peers [21]. Moreover, studying ways to show her social network to the peer are an open problem too. An interesting and used approach is one based on an open source graph visualization tool (<http://www.touchgraph.com>).

A big concern with RS in general is the lack of objective and effective ways to evaluate the performances [20]. Offline measures are in a way artificial and don't always capture user acceptance while having a running recommender used by many real users in order to test working hypothesis is always very difficult and uncommon in the research community.

Concerning the architecture, there are a number of open questions: we have just indicated as a condition the possibility for every peer to fetch all the information of the other peers. This can be done through a standard HTTP protocol or through one of the many different P2P networks (Gnutella or Freenet for instance). Issues such as replication and caching of the data also arise.

The distributed setting and, especially, the information self-publishing create some issues related to interoperability of different formats. This arises both at the syntactic level and at the semantic one.

In general, there is a big debate about the incentives to well-behave in a distributed environment (see the cornucopia/tragedy of the commons in section 2.3).

A huge concern in online communities is privacy. This is especially true when these communities are related to expressing opinions about items of the world (let's suppose you are reading many books about how to cure AIDS) [10]. Related to this there is the problem of having a way to uniquely identify a peer in order to be able to keep trust (or distrust) in her. The approaches can go from using pseudonyms based on public key cryptography and not relatable to real identities to normal IP addresses. It is also interesting to analyse the possible attacks that can be created against a distributed system and the way in which trust can help in detecting malicious peers and to spot them out.

Last it is surely a concern to deeply analyze the complexity of proposed algorithms and to optimize them. However, the strategy we follow is similar to what Google.com does: the algorithms run offline in order to precompute all the useful data (trust, user similarity, recommendations for every user), then when asked, the system just outputs the precomputed data. This creates a time delay between when data are available and when they are taken into account but we believe that in such an environment, this is a reasonable trade-off.

## 5 Proposed solution

In this section we define the problems we address, the environment we assume and the solution we propose. We also introduce precise definitions for the key concepts.

We claim current Recommender systems use different techniques in order to suggest to users which items they will probably like. But, up to now, RSs don't consider explicitly stated direct trust information among users to guide the recommendation process.

Our goal is to exploit this information in order to create better RSs that can enhance current accuracy and user acceptance.

In order to reach this goal, we address two of the specific problems introduced in section 4.

The first is related to trust propagation: we are interested in understanding how much we can predict trust values for peers not known by our peer (i.e. on which she didn't provide a direct trust value) but that are reachable through chains of trust. For instance, if A trusts B at a certain degree and B trusts C at another degree, what we can predict about the trust of A in C?

The second problem we address is about combining the information related to trust (either direct or indirect) with the user similarity about items ratings that is the information at the base of current RSs based on Collaborative Filtering. We will take into special account the concept of confidence in every kind of information.

We propose a solution to the problem and we define some tests we will run in order to verify our solution (see section 6). But first let us define the environment we work on.

### 5.1 The environment

We assume the environment is composed by uniquely identifiable peers that express trust values on other peers and rating values on uniquely identifiable items. All the information are public to every peer and every peer is able to fetch them whenever it prefers.

Users express trust values in peers based on their perceived quality as source of advice about items, i.e. a peer should trust a peer if it rates items in a consistent way w.r.t. herself and also if it especially good in providing ratings on unknown but relevant items or if it rates very rapidly new items.

More formally, we have:

$$P = \{\text{set of } n \text{ uniquely identifiable peers}\}$$

$$I = \{\text{set of } m \text{ uniquely identifiable items}\}$$

Every item has a creation date and a creator; there are 2 possible environments: one where creators are peers and the other where creator are just external identities (ex. “Kubrick”). For now, we consider the second one, because the first one (that arise for example when blog entries are the items to recommend) introduces more problematic issues.

Every peer can express one or more *trust statements* that embed her opinions about the trustworthiness of another peer. Trust statements have the following form:

- `trust(SourcePeer,TargetPeer).value=0.8 ∈ [0, 1]`
- `trust(SourcePeer,TargetPeer).date=22dec2002`

This represents the fact SourcePeer has explicitied, on December 22, 2002, the fact she trusts TargetPeer as 0.8.

We can assume every trust statement of SourcePeer is in a file `Trusts.xml` (expressed with some XML-based format) that is publicly available on her peer. Every peer is free to delete or change her trust statements but also every peer is free to retain previously fetched trust statements and to reason about them.

Moreover, every peer can also express one or more *rating statements* in which she explicit her degree of interest in a certain item. Rating statements have the following form:

- `rating(SourcePeer,Item).value=0.1 ∈ [0, 1]`
- `trust(SourcePeer,Item).date=12sep2001`

This represents the fact SourcePeer has explicitied on September 12, 2001 she is not very interested in Item, infact the rating is 0.1 where 0 is the minimum and 1 is the maximum.

We can assume every rating statement of SourcePeer is in a file `Ratings.xml` (expressed with some XML-based format) that is publicly available on her peer. Every peer is free to delete or change a rating statement but also every peer is free to retain previously fetched rating statements and to reason about them.

The information about the date is not considered in this work, for example, if two trust statements of SourcePeer on TargetPeer are available, we just consider the last one (see section 8).

Every peer is also able to express information about herself such as homepage, photos, description, blog entries, etc. This information is not necessarily in machine readable format and it is used by humans in order to decide, along with what items likes and which peers trusts, about trust values on peers. The idea is that if the peer writes many clever and interesting entries about the books you like in her blog, you should decide to trust her when the domain is books. This information is really dependent on the domains in which the RS is in (music, books, auctions, opensource projects, etc.) but anyway this information has to be consumed only by the human users in deciding about trust values. The algorithm will then use the provided trust statements that embed such a judgement.

In fixing the environment, we don't consider different ways of data publishing such as the fact data are cached or replicated in some way or the fact that a peer could like to show some statements to a peer and different ones to another peer. This is an interesting problem because it would be possible to take into account trust also as provider of reliable information but this adds a good level of complexity and we don't address the issue.

In the following we just assume that all the peers publish the complete information, that peers can fetch complete information about a peer, that it is not possible to fool peers by showing to some peers some data and to different peers different data and that all the peers information are always available (either because they are always up or because there is a clever replication mechanism).

## 5.2 Definitions

Now that we have defined in detail the environment we assume, let us give precise definitions of the key concepts.

### *Trust*

The definition by Gambetta [15] is often cited: “... trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both *before* he can monitor such action (or independently of his capacity ever to be able to monitor it) *and* in a context in which it affects *his own* action”.

In our environment the particular action is “to rate items with the same value as the peer would have rated”

We assume that every trust statement has a value in  $[0, 1]$ , so that in this sense, `trust(SourcePeer,TargetPeer).value = 1` means that SourcePeer believe that TargetPeer will rate the next items in the exact same way as he would have done. Conversely, `trust(SourcePeer,TargetPeer).value = 0` means that SourcePeer believe that TargetPeer will rate the next items exactly in the opposite way.

In reality the way an user decide about trust values in a real setting can depend on many factors, for example, if she writes interesting posts (on her weblog, on slashdot.org or on a mailing list), if she fulfill her obligations as a vendor or as a seller (on ebay.com), if you download many and high quality files from her (on edonkey P2P file sharing network or on gnutella reputation aware clients), if she contributes greatly to and carries on interesting open source projects (on affero.org), if you think she is any good in writing reviews of books (on amazon.com), and in general, if you trust her as a good source of advice, possibly in the real world (she likes items similar the ones you like, has many rated items (good coverage), is very fast in rating new items, is able to discover unknown, unrated but interesting items, ...).

There are even different definitions of trust: for example, in evolutionary biology or in repeated games where it is often defined as “perceived willingness to reciprocate”; here we will consider trust on peer as rating on peer expressing her subjective perceived quality as source of advice.

A special attention should deserve the negative trust statement (trust.value = 0), this can embed two semantically different judgements: the meaning could be “it is totally different from me (without regard to its maliciousness)” or “it is a malicious user (without regard to its similarity)”. We have to understand better if the two meanings have to be separated into two judgements. Anyway, for now, we are not considering this issue.

Moreover, we consider that the user express by her trust value in PeerA her degree of agreement on PeerA’s ratings on items and ratings on peers; in a way we assume that if a peer is good (for me) in expressing ratings to items, she is also good in finding trustable peers.

Trust usually is not symmetric: if A trusts B this doesn’t mean B trusts A.

#### *Reputation*

Webster defines “reputation” as: 1. The general estimation in which a person is held by the public.

In general, reputation is seen more as a property of a peer depending on the social network. Usually reputation means the global value computed weighting all the available trust values. Many current online reputation systems [35] allows you to see the global value of reputation of every user and this value is computed from every trust statements expressed by users; for example, ebay.com shows near every username a star of different colors where a color means a certain level of reputation in the community computed on the basis of positive and negative feedback given by users after every transaction.

Reputation in called karma on slashdot.org and whuffie on [12].

We claim it is nonsense to compute a global value for every peer when it is possible to compute a personalized, subjective value. It make no sense to show to everyone that user “Richard M. Stallman” is a reputable peer. His reputation (from one peer’s subjective point of view) should depend on the fact this peer expressed, for example, she values “Bill Gates” or not. Computing a global value is also what PageRank [33] does when trying to guess the “reputation” of a web page based on the incoming and outgoing links. These approaches have some inherent weaknesses: in particular they don’t take into account as starting points the very personal view of the world every peer has and they try to compute a global value that should fit everyone. For this reason they are also an easy target for a very basic attack (see section 3) that allows to influence the computed global value of reputation by the creation of many fake peers stating “false” rating and trust statements.

For these reasons, we don’t use the term “reputation”. We make a difference between the term “direct trust” (value provided by the peer) and “indirect trust” (value computed by an algorithm on the base of some chains of direct trust but always on a subjective basis).

Every trust value is totally subjective and different for every peer. In our model, a global reputation value for a peer is not used and so it is not computed; actually we also claim it is useless because easily fakable.

#### *Rating on item*

A value expressing the subjective interest the peer has in the item, i.e. how much she like it. Typical items considered in research have been movies, songs, restaurants, jokes, etc. We consider it as a real number in  $[0, 1]$ . This is the standard input data for the Collaborative Filtering framework.

#### *Peer*

A uniquely identifiable autonomous entity able to expose some information. In a sense, it is the avatar of the user, i.e. her virtual ego. It can be her blog, her Gnutella running application, her instant messaging client, etc. We use it often as a synonym of user, a user in a society can be seen as a peer among peers.

#### *User similarity*

It is a value in  $[0, 1]$  that it’s computed by the system; it represents the similarity of two peers. It can be based on whatever values; we compute it based on ratings on items (if we like same items, we are similar) but we should not forget that it can be computed based on trusts values (if we trust same people, we are similar) or based on geographical coordinates (if we are physically near, we are similar) or on any peer’s characteristic.

It is not required that User similarity was symmetric [ $\text{Similarity}(A,B) = \text{Similarity}(B,A)$ ] even if it often the case.

#### *Confidence*

It is a value in  $[0,1]$  that represents how much we can consider reliable a data. We will use it on information computed by algorithms such as User Similarity or Indirect Trust but also on starting data such as rating and trust statements. Also the outputs of the algorithms will exit with confidence values along.

For example, confidence in user similarity can depend on the number of items both the users have rated.

#### *Accuracy*

A metric used to represent the quality of generated recommendations. It is computable offline. For standard CF approaches, it is usually computed on standard public datasets of ratings expressed by users on movies such as Eachmovie [29] or Movielens.

The metrics most used are the following: Mean Absolute Error and Ranking Error. With the first, we suppose we don’t know the values we know, we try to predict them and then compute some weighted sum of difference among predicted and real values. With the second one, we suppose we have the correct ordered list of most recommendable items and we compute the difference between the generated list and the correct one. It is also possible to use standard metrics of information retrieval such as Precision and Recall.

It has been argued that for RSs online evaluation are more meaningful (but also more difficult to create) [20].

#### *User acceptance*

A metric used to represent the overall perceived quality of the Recommender System. Human computer interaction factors go in it as well [43]. It is only computable with an online survey with real users. In social

applied computer science application, it is much more meaningful than offline metrics.

### Context

A semantic category [31, 1]. Some examples are: classical music, movies, romantic movies, jokes, research papers, research papers on artificial intelligence, and so on ...

Every statement (trust and rating) is referred only to a certain context. If you trust PeerB as a mechanic, this does not imply that you trust PeerB as well as baby sitter. In real world situations, there are surely dependencies among different contexts: for example, trust as violinist is probably very related to trust as musician and it is possible and interesting to consider trust values in one context to predict trust values in another context. However, existing reputation systems (ebay.com, amazon.com, affero.org, ...) consider only a single global context. In this PhD work, relations of trust among different contexts is not considered because it add a huge amount of complexity (see section 8)

## 5.3 Proposed solution: details

As previously stated, our goal is to exploit explicitly provided direct trust information in order to enhance the accuracy and user acceptance of current Recommender Systems.

Let us before state what is the basic assumption that has guided our choices. This assumption will be anyway verified by some experiments (see section 6).

The basic idea is that increase of trust of SourcePeer in TargetPeer should also increase the weight given to TargetPeer's opinions when forming recommendations for SourcePeer. The same point holds for User Similarity: recommendations to SourcePeer should be based on opinions of peers similar to SourcePeer.

And what are the expected relationships between Trust and User Similarity? Intuitively, the most common and expected situation is that, when  $\text{Trust}(A,B)$  is available and  $\text{UserSimilarity}(A,B)$  is computable with a sufficient level of Confidence,  $\text{Trust}(A,B)$  and  $\text{UserSimilarity}(A,B)$  have similar values; this represents that in general a peer trusts similar peers.

This is so by definition because the user expresses trust values in peers depending on how much they are good as source of advice for them and in order to be a good source of advice a peer should, in general, be at least similar to the target peer.

But we are not saying that one information is the exact copy of the other or that one is created using the other: for example,  $\text{trust}(\text{PeerA},\text{PeerB})$  is not computed using similarity of ratings of PeerA and PeerB, or viceversa. Infact Trust is explicitly provided by the user and User Similarity is computed by the algorithm taking as input the rating statements explicitly provided by users.

We claim that Trust and US can complement each other and also that it is totally possible one contradicts the other, i.e. "I'm similar to a peer I don't trust" or "I trust a peer totally different from me". In this cases it is

important to understand the deep meaning of such a social situation and to exploit in the best possible way: for example, it can be the case a malicious peer has copied the ratings of another peer in order to influence the recommendations she receives. Many of the possible situations will be analyzed in some experiments in section 6.

Moreover, in general, User Similarity (US) is computed as a symmetric quantity, i.e.  $\text{US}(A,B)=\text{US}(B,A)$ ; while Trust in general is not symmetric, i.e. the fact I trust Richard doesn't imply that Richard trusts me.

We are going to test our basic assumptions about relationships between trust and user similarity against a database of real users, allconsuming.net (see section 6). Then, we keep a special attention when we synthetize data for simulations in making the generating models clearly explicit. We will of course simulate different communities, ranging from the most "natural" to the most "unexpected" ones. In this sense, the logical process is the following: if there is a community where rating and trust statements obey to these rules, then the algorithm has those performance.

A trust-aware Recommender System has the following methods:

```
Rating getRating(Peer,Item)
Item[] getAppreciatedItems(Peer,n)
Rating predictRating(Peer,Item)
Item[] predictAppreciatedItems(Peer,n)
Trust getTrust(Peer,Peer)
Peers[] getTrustedPeers(Peer,n)
Trust predictTrust(Peer,Peer)
Peers[] predictTrustedPeers(Peer,n)
```

This means it is able to predict how much the peer will rate an item and how much the peer will trust another peer. As a consequence, it is able to output the list of most appreciated items and of most trustable peers.

We are now ready to expose the solution we propose in order to achieve our previously stated goal. We give a special emphasis to the concept of Confidence through all our strategy. In fact we have claimed Direct Trust has a total confidence because it is provided by the user while User Similarity and Undirect Trust has a minor confidence since they are computed by an algorithm. We think that taking into account confidence in every step (from data collection to peers trustability prediction to recommendations computation) allows to better keep the process under control and to let easily present it to the user in every single step. This can solve the black box problem as pointed out in [21]. Moreover the combining strategy is able to integrate every algorithm in the different steps as long as they provide also self-confidence in the predicted values.

The message the user should grasp is the following: if the confidence is high, this means the algorithm was able to produce a reliable enough recommendation and this is probably a conservative one; if the confidence is low, this means the available data don't allow to predict with a safe level of certainty and also that the recommendation can be a serendipitous, unexpected one.

We propose a 4 steps strategy to combine trust and rating statements in order to create recommendations. The steps are: trust propagation, user similarity computation, trust and user similarity combination and ratings prediction.

The basic idea is that a direct trust statements should be taken into account as certain while for all the other computed quantities the algorithm should reason also about their confidence.

Let us remind that all the computation are local to a peer (in the following called ME) and computed on behalf of her user, i.e. the user is the center of the computation and everything is seen from her point of view.

The first step is *trust propagation* (figure 1). Here the algorithm takes as input all the available Trust statements of every peer. The goal of this step is to predict how much ME should trust the peers she doesn't know (i.e. she has not expressed a trust statement on). We call this data indirect trust.

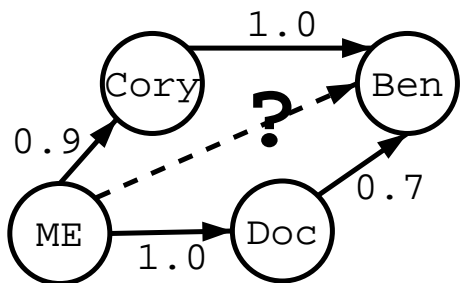


Figure 1: **Trust propagation step:** Given the social network centred in ME and represented by trust statements (solid lines), this step predicts indirect trust values, with confidence, on unknown peers (dotted line).

This is an open problem in research, there are some proposed solutions (see section 2.2) but no clear winner. We would also like to underline how all the proposed metrics are simple because what was really missing was a real testbed in which to test the reasonability of them.

We plan to take some algorithms from the literature and to test them and to improve and adapt them. We model the problem as a Max Flow problem on a graph [13].

The Max Flow problem consist of a directed graph in which edges are labeled with capacities. There are 2 special nodes: the source and the sink. A flow is an assignment of flows to each edge, such that the flow does not exceed the capacity for each edge and the sum of the incoming flows equals the sum of the outgoing flows for each edge, but the source or the sink. The goal is to find a flow which maximizes the incoming flow at the sink.

We take figure 1 as an example of trust graph. In this, edges are trust statements and capacities are the trust values. For example the edge from Cory to Ben means that ME has fetched in Cory's peer a trust statement saying that Cory trusts Ben as 1.0 out of 1.0. The Max Flow problem is about finding the maximum flow of trust from ME (source) to Ben (sink). Essentially the trust

begins at ME and flows through peers depending on the capacity of each edge (i.e. the amount of trustworthiness given). The trustworthiness of peer sink is the quantity of trust flow that reaches it.

We have to modify the Max Flow algorithm in order to take into account the fact that every step in a trust chain reduces the confidence we have in the computed trust. To do this, we enter 2 parameters: propagation horizon and step confidence decay. Propagation horizon is the maximum number of steps we let flow the trust while step confidence decay is a factor that reduce the confidence in the computed indirect trust on a peer at every step. Both of them have to be decided based on experiments with some heuristics. We can say from now that surely propagation horizon will be a number in the range of 3 to 6, also considering the "Six Degrees of Separation" folklore axiom [30]. The best algorithm to solve the max flow problem is Ford and Fulkerson [13] created in 1956.

Let us remember that computing a subjective value of indirect trust for every peer enables us to not be influenced by malicious peers as long as there is no trust chain from ME to them. A similar approach based on max flow algorithm is taken in advogato trust metric [26] that is also proven to be attack resistant.

In order to test alternatives, we could take into consideration algorithms based on bayesian networks (considering trust as a probability) and Collaborative Filtering strategies applied directly on the User  $\times$  User trust matrix.

The second step is *User Similarity computation*. It takes every rating statements as input and produce as output the similarity of the current peer (ME) against all the other peers. The similarity value is complemented by its confidence. For this step, we use the Pearson Correlation coefficient that has proven to be the most effective way to compute user similarity in CF settings [7]. Another less effective possibility is Vector Cosine similarity. Both the techniques

We are referring here to standard CF techniques that consider an user as a vector of ratings on items and intend user similarity as the distance between two vectors but we should note that alternatives have been proposed, for example the use of Singular Value Decomposition in order to reduce dimensionality [38]. The confidence in user similarity can depend for example on how many items where rated in common by the considered peers, it is clear that if 2 peers rated only one movie and it happens that the ratings are equal, this does not mean they are necessarily two very similar peers. In this case the algorithm should return 1 as user similarity but with a confidence very low.

The third step is *trust and user similarity combination*. This take as input the trust values (either direct and indirect) and the user similarity. At this moment the algorithm must decide what should be the weight given to every peer's opinions when forming recommendations for peer ME. We call this quantity User Influence. As a baseline solution, we will compute User Influence in

one peer as a weighted sum of the two quantities coming from previous steps (trust and user similarity).

We still consider the very important information related to confidence of input and also we provide confidence in the computed User Influence.

In first approximation, we will use heuristics for determining weights. At this stage we also think that direct trust should always overwrite user similarity, in a sense we are saying that if the user said she wants her recommendations based on some peer she trusts, we cannot disattend this expectation.

The heuristics for calibrating the weights will be defined thanks to the experiments (see section 6). The experiments will also allow to better understand the relationships between the two concepts of trust and user similarity. This is a contribution all the research community can benefit.

In experiments, it is important to analyze results with varying weights on similarity versus indirect trust, considering also the two extremes: only indirect trust and only user similarity. For example, in an environment with many malicious users it can be safer to just rely on (direct and indirect) trust while on systems where there are few incentives to directly express trust in other peers or this information is very noisy it can be safer to rely mainly in user similarity. The confidence in User Influence about every single peer will also be computed.

In experiments we will also examine what are the best approaches: for instance, if it's better to simply don't consider a peer whose trust in is 0 or if a peer with ratings exactly the opposite can be considered as a negative predictor (I like what she doesn't and viceversa).

It is even possible to learn the optimal weights for the different information by reinforcement learning or adjust them on the fly by taking into account feedback from users, but this is something we will think about depending on time.

The last step is *ratings prediction*. Its goal is to predict for every item not rated by peer ME how much the user will like it. Then this information can be used for example to output the top ten list of unrated interesting items to the user. This step is a standard step in most Collaborative Filtering approaches and we will rely on standard solutions. The most common is the following: in order to predict the rating of peer ME for an item, let use the ratings given by other peers to this item and consider them accordingly to the computed User Influence of every single peer. The confidence in final predictions about one item can be computed depending on the number of peers that rated that item and also depending on the confidence the system carried on about their user influence.

Let us underline the importance of confidence tracking through all the recommendation process. Essentially, this would allow us to easily integrate every possible algorithm in the different steps as long as they provide also self-confidence in the predicted values, without having to retune or redesigning the all strategy. We can say that it is not very important that the algorithm chosen

for one step would be the best one. It is indeed very important that it can output a correct value for the confidence in the computed data. For example, we could game our recommendation process using a random algorithm in one step, only requiring that it correctly states the confidence in computed, that in this case is 0. In the case of a step with a random algorithm, the final outputs must be with confidence 0 but still everything should work without ad hoc adjustments. Moreover, it is also true that we can predict the mean confidence an algorithm can have, for example, by making it run against a test set. In this sense, our goal is to find the heuristics for confidence combination that work with whatever algorithms. If tomorrow a better algorithm for network propagation arise, we should be able to simply insert it for step 1 with no changes.

Confidence can be taken into account also before the first step, directly on starting data: let suppose for a moment (violating our assumptions) that the starting data are not 100% certain but that there is some noise over some channels or that some peer can provide false data. In this case being able to understand and remember what is the confidence of starting statements could be also of help. Another case happens if we suppose that starting

It is important to underline that this recommendation process should be transparent and known to the user, i.e. the RS is not a black box. In this way, the user can be aware of the way the system works and every computed information at every step can be outputted to her so that she can see it, understand it and, possibly, directly modify it, providing feedback on the algorithm as well. Tests will say which solutions are the most effective.

We are not going to conduct in this work a deep study about complexity of the algorithms or optimization. However, as already pointed out in section 4, we follow the strategy of precomputing all the useful data offline (trust, user similarity, recommendations for every user) and then when asked the system just outputs the precomputed data. This is similar to what many online systems do, for instance Google.com.

There are some issues we haven't decided about yet: for instance, if the algorithm should also try to adjust trust keeping track of what is changing in the society or at least warning the user, if the algorithm should warn the user when there are big changes in statement expressed by one trusted user and also if the system should warn the user when there is a big change in trust statement for some trusted peer (for example, many of your friends change their trust value for HiddenPeer from 1 to 0 in a short time). These and other similar issues will be clarified on a real system (see section 6.4).

## 6 Evaluation

In this section, we explain the tests we will run in order to understand strengths and weaknesses of the proposed solution. Some test will also be devoted to understand

relationships between trust and user similarity in real world systems.

We will concentrate on Recommender System accuracy but we will also create the conditions in order to be able to run our solutions in an real system. In the latter we will be able to conduct surveys with real users in order to test also user perceived usefulness and controllability of the system.

### 6.1 Experiment to verify relationships among trust and user similarity on a real system: allconsuming.net

In section 5.3, we have defined what is the expected relationship among trust statements and rating statements. In this first experiment, our aim is to understand what the situation is in a real system with data collected from real users.

Allconsuming.net is a web site where an user can create her book lists (favourite, currently reading, finished, etc.) and her list of friends in the literature domain, i.e. users whose opinions about books the user cares about; there are also some powerful and easy to use tools to manage this information, to show it on personal blog and to easily cite books in your daily posts. What is fantastically good of Allconsuming.net is the fact that all the collected data are public (they are published in XML and are also fetchable through a SOAP interface). At May 2003, in Allconsuming.net, there are almost 1000 users, the rated, cited or inserted books are many hundreds of thousands and every user has expressed a list of friends (users they trust).

We plan to use this information to validate our assumptions. The mapping of Allconsuming.net data into our setting (see section 5.1) is the following: personal books lists and book citations are the rating statements while friend lists are the trust statements; in this sense we only have a boolean information:  $trust = 1$  (when peer is in friend list) or  $trust = undef$  (otherwise).

Similar experiments could be conducted on epinions.com but we have to verify if crawling this web site in order to collect its data does not infringe some copyrights.

### 6.2 Experiment to test trust propagation strategies

We will conduct some experiments in order to understand which of the trust metrics available in literature (see section 2.2) works better under many different conditions. In order to do this, we will synthesize data corresponding to many different types of communities; for example, we will create communities that are cluster consistent (peers of one cluster trust each other and distrust peers of different clusters) or where there are some leaders trusted by many and gregarious users who have no trust statements on them and bad users who are distrusted by many and also some communities where there are no clear patterns of trust relationships.

According to the hypothesised models, we will generate complete trust statements, this means we will know the subjective trust expressed by every peer on every other peer. However, we will suppose peers will express only a little percentage of these trust statements (with varying percentages), as in a realistic setting usually is.

We will use different trust metrics in order to predict indirect trust values and we will compare predicted data with real data computing a global error value for every single trust metric.

We will do the same also for data representing a real community (allconsuming.net). In this case we will use "leave one out" techniques: we will exclude one trust statement and we will try to predict it.

This experiment is similar to what is done in [31]. With this experiment we will give a contribution to better understand which algorithms works under which conditions.

### 6.3 Experiments to test accuracy of proposed solution on simulated communities.

After having verified what are relationships among rating and trust statements in a real system, we will synthesize data in order to analyze the behaviour of our proposed strategy in many different simulated situations, from the most expected to the least expected.

We need to synthesize data in order to be able to test our system performances under a controllable, replicable, laboratory situation. In every case, we will point out clearly what are the generating hypothesis; the statement is the following: "if the population of a system behave like this, then the algorithm performs in this way".

This approach is used also by most of the research in trust, reputation, evolution of cooperation [45, 22, 31, 26].

Let now see some of the communities generating hypothesis. In the following, for sake of clarity, we consider the movie domain, so that the items are movies. The first synthesized community obeys to our assumptions that usually it is normal to trust an user if this is similar in rating movies and distrust her if she is dissimilar.

We assume there are some hidden features related to every user: for example, the mean value they give to movies of a certain category (romance, thriller, etc.) which represents their interest in this category. For example, peer A likes romance as 0.6, while peer B likes them as 0.7. We also assume that the distribution of ratings of one peer in one category obeys a certain gaussian with a certain variance.

Then we define the trust value of A in B as the mean distance in their interests in categories. For example,  $trust(A, B) = 1 - |0.6 - 0.7| = 0.9$

In this way we have all the information about every user: all the correct rating statements on movies and all the correct trust statement on peers. We can imagine every peer rated only a fraction of items and users. We can then use our strategy to predict the missing values and compute the error between predicted and correct

values. In particular, we can predict the items ratings and compute the mean square error or we can predict the list of 10 preferred items and compute the ranking distance with the correct list. We can do the same for the predicted indirect trust on peers. In this way, we can verify the accuracy of the different possible strategies (such as considering only user similarity or only trust propagation or any combination).

There are many parameters we could play with in generating the simulated communities, in particular the mean and variance of the distributions, the number of movies, the number of peers, the number of movies rated by every peer and the number of trust expressed by every peer. Modifying them generates many different communities, for example, one where there are few ratings (sparseness problem) or few trust values (a not very connected society).

We will also create communities where user similarity and trust does not always confirm each other. Let analyze the main cases that can arise.

- A peer expresses a high trust value on a peer with low similarity: this tells the system “even if for now we are not similar, I know her in real life, I trust her and I want my recommendations to be based on what she likes). Probably the system should launch a warning to the user.
- A peer expresses a low trust value on a peer with high similarity: this tells the system “even if we are similar, I don’t want my recommendations to be influenced by her opinions”. She is probably a misbehaving peer, a thief or a spammer.
- A peer didn’t express a trust values for a very similar peer: this means the algorithm should probably suggest her as a possible trusted peer

Moreover we design specific experiments to verify the claims made in section 3. Let analyze them here:

*Trust-awareness solve CF sparseness and cold start.* In this case we are interested in communities where the ratings expressed on items are really a few, i.e. the matrix of rating users  $\times$  items is very sparse.

When there are few ratings on items, traditional CF approaches have difficulties in computing user similarities and for this reason the list of neighbours is a very unreliable information; this usually results in low quality recommendations. This is a huge problem for recommender systems, especially when they start operating.

We are interested in understanding how few explicitly provided trust statements can improve the accuracy of RS. We can assume that the user cognitive effort is the same for producing a rating statement or a trust statement and we can assume an user has expressed  $n$  statements (either rating or trust). In this case, we want to verify if trust statements can improve the performances w.r.t. rating statements.

Another problem of traditional CF is cold start: this happens when a new user enter in the system and has

expressed no statements. For traditional CF, it is impossible to compute her neighbours and so to give recommendations. We claim that usually an user entering in a system has (or can easily find) a trusted user already in the system. This can also be the case of an user desiring recommendations in one domain that is unknown to her, for example, let imagine a computer scientist wanting to receive recommendations about physics theorems; even if she doesn’t venture herself into rating physics theorems, she probably has some physician friend she trusts, in a sense she is delegating judgement on items to this trusted friend; this has been labelled the Einstein problem (see section 5.4.2 of [31]).

In this case, we want to understand if just one (or few) trust statement allows the new user to easily jump in and what is the accuracy of the resulting recommendations. It is also interesting to verify how many indirect trust values can be predicted with a certain level of confidence, in a sense, we want to understand what is the coverage of a social network given only few initial trust statements.

*Trust-awareness helps managing malicious peers.* Our approach allows to not be influenced by malicious peer when they are detected, simply because having them with  $trust = 0$  (either by you or by your trusted peers) means they are not taken into account. But we need to understand better if the proposed strategy allows also to spot out malicious peers. Essentially, a malicious peer tries to create false statements in order to influence the recommendations generated by another peer; the simplest attack is made by copying a peer profile (all her statements) in order to be considered similar and by then rating high one item. This could result in this item being recommended to that peer. We can easily show how this simple attack have disruptive effects on a traditional RS based on CF, for example simulating the attack on a dataset of real ratings such as Each-movie [29].

Interesting experiments to demonstrate the attack-resistance of proposed trust metrics are carried on in [26, 22]. We need to think about similar experiments.

*Trust-awareness increases user control.*

It has been claimed that RSs are difficult to control by the user [46, 17]. Typically when the RSs start giving bad quality recommendations, the user simply stops using it because it is very hard to understand why a recommendation was made and how to influence the recommending process. Even when possible, for example in Amazon.com with the editing feature which allows you to change past ratings in books, “most people never take the time to use it” [17].

It has also been claimed that often users don’t know or understand the process behind a RS [21, 43]. In general, this black box model reduces the efficiency of RSs.

Moreover, we claim that showing to the user as well the social network that is at the basis of the generated recommendations allows her to better understand the reasons behind a recommendation and to control it easily when wrong.

Anyway, this has a lot to do with Human Computer

Interface issues and the increase in user control is something we will verify better by surveys with real users (see next sections).

## 6.4 Online test: [cocoa.itc.it/blog](http://cocoa.itc.it/blog)

We think that for social, applied computer science research, more meaningful results derive from testing the proposed solutions with real systems and users. In order to have a viable testbed, we are going to add weblog capabilities (see section 2.4) to an online recommender system we developed: CoCoA (Compilation Compiler Advisor) [5]. The system is usable at: <http://cocoa.itc.it>.

CoCoA is a RS suggesting classical musical compilations on the basis of past user created compilations. It is based on the Karadar archive and this means 11.000 MP3 of classical music without copyright problems, Composers Biographies, 400 Opera's Librettos, 2.000 photos of composers, rare scores and theatres, 5.000 texts of classical songs in original languages and 1.100 Midi files. But what is really important is the fact there are about 1400 users connecting every day willing to make compilations and to contribute to the system with their passion and knowledge for classical music.

A CoCoA user will be able to keep her blog on [cocoa.itc.it/blog](http://cocoa.itc.it/blog) where they will be able to post daily entries about classical music. They will also be able to express ratings about other cocoaBloggers (trust values) regarding their knowledge and utility for them on classical music. They will be able to express their ratings about classical tracks and about classical composers. They will create compilations (assembling some tracks according to some idea) and rate compilations as well. We will have a lively audience of users really interested in classical music willing to contribute to the system (we already receive many mails with suggestions and corrections). We will be able to test many recommendation strategies with real users and to tune and improve our choices.

In homepage there will be a link to a controlled usage of the system; the user who will agree to use this version will be prompted with questions about their experience with the Recommender System. We will ask to them to provide lively feedback about every single recommendation, for example a thumb up/down one click feedback.

We will also ask to participate in surveys; some of the questions could be "Were recommendation useful?", "Do you think you have understood the recommendation process the system use? Can you describe it?", "Was useful the possibility to see your neighbours?", "Why did you modify your previously expressed trust statements? (if they did)", "Why did you modify your previously expressed rating statements? (if they did)", "Was the RS starting giving bad quality recommendations? If yes, what did you do? If not, what would you do in this case?", "Was the system easy to use?", "which other RSs do you know (amazon, epinions, ebay, ...)?", "which ones you have used?", "did you find any differences? which ones?".

Anyway, we will also be able to run experiments offline with collected data.

An important side effect of blog on CoCoA is the fact that all the data will remain publicly available (in XML format) to everyone. This will benefit the all research community that will be able to use an always up to date, large, distributed database of useful data. This will also provide a solution to the traditional lack of dataset for RSs experimentation.

We are also thinking about making a competition of Recommender Systems by providing a standard API that every RS willing to participate has to implement in order to participate in the challenge; we will design a script that, when asked for a recommendation, will ask to a randomly chosen RS and provide the result to the user [20].

## 6.5 Other online tests

It is possible to think other test with real users based on other running systems. We could address the blogosphere (i.e. the set of all the blogs) or [allconsuming.net](http://allconsuming.net), a collector of books preferences for bloggers (see description in section 6.1).

In the first case, we could use the data about every blog registered on [weblogs.com](http://weblogs.com). In particular, almost every blog has a blogrolling list or a FOAF file of trusted blog and we could use this information as trust statements. We can then consider as items the words the blogger writes in her entries or the categories she classifies her posts in ("politics", "open source software", "mac", "my cat", etc.).

We could run our algorithm in order to discover new trustable peers and to "recommend" new unexplored categories (or words). In order to receive feedback about the generated recommendations there are 2 possible ways: either we send a polite mail to the blogger saying this mail was automatically generated by a research driven RS and asking her to go to a web site to write down some feedback about the recommendations or we can just create a web site that is able to create recommendations for every blog and then make it enough visited by bloggers (considered what the blogosphere is, this is not an impossible situation).

Similar points hold for [allconsuming.net](http://allconsuming.net) but in this case we could recommend books (and similar users). This is precisely in the intentions of [Allconsuming.net](http://Allconsuming.net)'s developer who invites everyone to use this data in order to create new, interesting services.

# 7 State of the project

## 7.1 Progress up to date

Relevant to this proposal, we have been surveying the literature related to Recommender Systems and their always problematic evaluation, to trust and reputation metrics and to peer-to-peer. This activity has produced as well some published papers [4, 5, 20].

We have developed an online Recommender System: CoCoA (running at [cocoa.itc.it/blog](http://cocoa.itc.it/blog)). We have investigated into Weblogs and have installed on an IRST machine what we repute the most promising tool, Movable Type (downloadable at <http://www.movabletype.org>). Weblogs will be available on CoCoA website soon.

## 7.2 Road map

In order to achieve our goals before the end of the third PhD year, we have set up a precise road map.

In the month of June 2003, we will completely define our strategies and write or adapt the used algorithms (maxflow, Collaborative Filtering and rating combiner). In the mean time, we will deploy blogs on CoCoA web site.

In the month of July 2003, we will fetch the data from allconsuming.net and run the first experiments on them, in particular in order to understand how our assumptions map on a real world case.

In the month of August, based on the insight the previous experiment will be able to give us, we will create the community generator in order to be able to synthesize many simulated communities based on many different parameters.

September, October and November will be spent at Stanford University as a visiting PhD. This time will be devoted to get feedback from professors and students of this University and to, possibly, tune and calibrate our solutions. In the mean time, while more experiments are run, we will keep on refining the proposed strategies and the evaluation techniques as well.

During November 2003, we think we will be able to collect enough meaningful data from blogs on CoCoA and will start running our online experiments. This will sure need some revising and retuning.

In November, we will as well start writing one paper for an important conference whose deadlines are usually in January or February, for example ECAI-2004 (European Conference on Artificial Intelligence) whose deadline for paper submission is in February 2004.

The first months of 2004 will be devoted to present the work in conferences and to write a paper for an international journal.

In the following months, we will concentrate on writing the PhD thesis, in order to finish for September 2004.

## 8 Future work

In this section we recall briefly what are the topics we are not going to work on in this PhD thesis but that are anyway worth thinking about. Most of them were explained in section 4.

We will not take into account privacy concerns that are however very relevant. In peer-to-peer networks, this is a very studied topic and has many implications in online communities and decentralized recommender systems as well. Authentication and identity persistence is also very important because without these it's impossible to keep trust in every single peer and reason about it. In

our work, we assume there are no privacy problems (every peer agrees on showing every information to others) and that it is possible to keep the same identity while it is not possible for a peer to impersonate another one (for example, we could use as ID the IP address).

Another very relevant issue in distributed systems and in trust-aware systems are possible attacks. Being every entity (we called them peers) totally autonomous, it is free to behave as it prefers. It is still a big challenge to design systems that are resistant to every possible attack.

Related to the architecture, we just made the assumptions that every peer self-publish some information and that every other peer is able to fetch it. In a more realistic environment, there could be issues such as replication and caching of data, data forwarding, secure communication through public key cryptography, different fetch policies, etc.

Another great concern in a distributed environment is syntactic and semantic interoperability. The first arise when two different peer publish information with different formats, the second when the same format is used in a semantically different way. In this work we assumed that all the peers share a common syntax and semantic. Related to this there is the issue about statements being context-dependent (a peer can trust RudePeer as a mechanic but not as a baby sitter). It is true that some kind of inference from one context to another can be done, moreover contexts could be organized in ontologies (context "violinist classical music" is child of context "classical music"). Such cross context reasoning is not taken into account, every single context will have its own statements.

A deep study of complexity of the algorithm is out of the scope of this work, especially concerning optimization. We will test runtimes in experiments, of course. Anyway we plan to have the algorithms running offline pre-computing recommendations and, when asked, they will simply output the precomputed data. This inserts a delay into taking into account recent data but we believe it is a reasonable tradeoff.

Moreover, we will not use the time information provided in every statement. We believe taking into account time in systems usually increase the amount of complexity and effects are usually hard to keep under control and to evaluate. In our work, when we have more than one statement made by a peer about a target, we consider only the last one.

## References

- [1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *HICSS*, 2000.
- [2] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *CIKM*, pages 310-317, 2001.

- [3] E. Adar and B. Huberman. Free riding on gnutella. Technical report, Xerox PARC, 2000.
- [4] S. Aguzzoli, P. Avesani, and P. Massa. Compositional cbr via collaborative filtering. In *ICCBR '01 Workshop on CBR in Electronic Commerce*, Vancouver - Canada, August 2001.
- [5] S. Aguzzoli, P. Avesani, and P. Massa. Collaborative case-based recommendation systems. *Lecture Notes in Computer Science*, 2416, 2002.
- [6] P. Avesani, P. Massa, M. Nori, and A. Susi. Collaborative radio community. *Lecture Notes in Computer Science*, 2347, 2002.
- [7] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998. Morgan Kaufmann.
- [8] Dan Bricklin. The Cornucopia of the Commons: How to get volunteer labor. <http://bricklin.com/cornucopia.htm>, 2001.
- [9] M. Brunato and R. Battiti. PILGRIM: A location broker and mobility-aware recommendation system. Technical report, DIT - University of Trento - Italy, 2002.
- [10] J. Canny. Collaborative filtering with privacy. In *IEEE Conference on Security and Privacy*, Oakland, CA, USA, May 2002.
- [11] F. Cornelli, E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. Implementing a reputation-aware gnutella server. In *International Workshop on Peer-to-Peer Computing*, May 2002.
- [12] C. Doctorow. *Down and Out in the Magic Kingdom*. Tor Books, January 2003.
- [13] L. R. Jr. Ford and D. R. Fulkerson. Maximal Flow Through a Network. *Canadian Journal of Mathematics*, pages 99–404, 1956.
- [14] E. J. Friedman and P. Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, 2001.
- [15] Diego Gambetta. *Can We Trust Trust? (in Making and Breaking Cooperative Relations)*, chapter 13, pages 213–237. 2000.
- [16] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [17] Lisa Guernsey. Making Intelligence a Bit Less Artificial. *New York Times*, 5 January 2003.
- [18] Garrett Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.
- [19] A. Harth, M. Bauer, and B. Breutmann. Iowl collaborative filtering in a distributed environment: An agent-based approach. Technical report, University of Applied Sciences - Wurzburg - Germany, 2000.
- [20] C. Hayes, P. Massa, P. Avesani, and P. Cunningham. An on-line evaluation framework for recommender systems. In *Workshop on Personalization and Recommendation in E-Commerce*, Malaga, 2002. Springer.
- [21] J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining Collaborative Filtering Recommendations. In *Proc. of CSCW 2000.*, 2000.
- [22] S. Kamvar, M. Schlosser, and H. Garcia-molina. Eigenrep: reputation management in p2p network. In *Proc. of WWW*, 2003.
- [23] S. Ketchpel and H. Garcia-Molina. A sound and complete algorithm for distributed commerce transactions. *Distributed Computing*, 12(1), 1999.
- [24] R Khare and A Rifkin. Weaving a Web of Trust. *World Wide Web Journal*, 2(3):77–112, 1997.
- [25] F. Labalme and K. Burton. Enhancing the internet with reputations: an openprivacy white paper. Web page, March 2001.
- [26] R. Levien. Advogato Trust Metric. <http://www.advogato.org/trust-metric.html>, 2000.
- [27] Farhad Manjoo. Gnutella bandwidth bandits. [salon.com/tech/feature/2002/08/08/gnutella\\_developers/](http://salon.com/tech/feature/2002/08/08/gnutella_developers/), August 2002.
- [28] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Univ. of Stirling, Scotland, 1994.
- [29] P. McJones. Eachmovie collaborative filtering data set. <http://research.compaq.com/SRC/eachmovie/>, 1997.
- [30] Stanley Milgram. The Small World Problem. *Psychology Today*, 61, 1967.
- [31] L. Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 20 December 2002.
- [32] Andy Oram, editor. *Peer-to-peer: harnessing the power of disruptive technologies*. O'Reilly and Associates, March 2001.
- [33] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

- [34] P. Resnick and H.R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
- [35] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation Systems. *Communication of the ACM*, 43(12), December 2000.
- [36] Paul Resnick and Richard Zeckhauser. Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay’s Reputation System. The Economics of the Internet and E-Commerce. *Advances in Applied Microeconomics*, 11, 2002.
- [37] Rishab Aiyer Ghosh. Cooking pot markets: an economic model for the trade in free goods and services on the Internet. First Monday, a Peer-reviewed Journal on the Internet.
- [38] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems—a case study, 2000.
- [39] J.B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceeding of the ACM Conference on Electronic Commerce*, Pittsburgh, PA, USA, November 1999.
- [40] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations, 2002.
- [41] S. Sen, A. Biswas, and S. Debnath. Believing others: pros and cons. *Artificial Intelligence*, 142(2):179–203, December 2002.
- [42] Clay Shirky. What is p2p ... and what isn’t? <http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html>, November 2000.
- [43] K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems, 2001.
- [44] Bryce Wilcox-O’Hearn. Experiences Deploying A Large-Scale Emergent Network. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS ’02)*, 2002.
- [45] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS*, 1999.
- [46] Jeffrey Zaslow. If TiVo Thinks You Are Gay, Here’s How to Set It Straight. *The Wall Street Journal*, 26 November 2002.