A Market-Based Recommender System

Luc Moreau[1], Norliza Zaini[1], Jing Zhou[1], Nicholas R. Jennings[1], Yan Zheng Wei[1], Wendy Hall[1], David De Roure[1], Ian Gilchrist[2], Mark O'Dell[2], Sigi Reich[3], Tobias Berka[3], Claudia Di Napoli[4]

[1] Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ UK
[2] QinetiQ, St Andrews Road Malvern WR14 3PS, UK
[3] Salzburg Research, SunTREC, Salzburg, Austria
[4] Istituto di Cibernetica, "E. Caianiello", Napoli, Italy

Abstract. We have designed, implemented, deployed and evaluated a large-scale agent-oriented information system that recommends relevant documents to users. Our recommender system is now being used across several European institutions. Its two key features are a modular design capable of accomodating multiple recommendation methods, and the use of a marketplace to select and rank the best recommendations for the user. As part of our evaluation, we have extensively simulated this marketplace in order to understand its dynamics and validate its suitability for a recommender system.

1 Introduction

The task of developing large-scale agent-oriented information systems that can deliver the right information, to the right people, at the right time is a major research challenge. This problem has been exacerbated by the massive proliferation of information sources, owned by different stakeholders, available over the Web (with their concomitant degrees of dynamism and openness) and by the increased use of multimedia content in these sources. To this end, this paper reports on our experiences of developing and deploying a large-scale multi-agent *recommender* system [15] that is able to assist users in finding documents relevant to their current context. In designing such systems, we believe there are two key challenges that need to be addressed:

(i) Many methods for suggesting relevant information already exist and more will be developed in the future. Some may be specialised to specific media such as audio or video; others may be particularly suited to process information in text documents; while others are more efficient in specific application domains, e.g. because they rely on detailed ontological domain models. From an architectural point of view, this means the challenge is to design a modular system that has the ability to accommodate multiple recommending methods, and to integrate them in a seamless and dynamic fashion as they appear on line.

(ii) Given multiple recommendation methods, it is comparatively easy to provide the user with a multitude of recommendations. However, the challenge

is to filter them and to present them to the user in a decreasing order of relevance, given that the different methods have different mechanisms and different semantics for relevance. Moreover, another difficulty is that the notion of relevance is not absolute but user-defined and varies over time, typically according to the user's interests and activities.

For the following reasons, we chose to adopt an agent-based approach to design and deploy a recommender system that addresses these challenges. *(i)* Users can naturally be represented in the system by user agents [9] that act autonomously on their behalf, finding information relevant to them, but also observing their activities, so that the system can tailor its answers to their needs. *(ii)* Information sources can naturally be represented as active agents whose objective is to ensure their content is widely disseminated to appropriate users. *(iii)* The agent based approach is well suited as a software engineering paradigm for distributed applications, with multiple stakeholders, in cases where the system grows in an organic fashion, as new users and information sources become active and new recommendation methods become available [6]. *(iv)* The loosely coupled and open nature of the recommender system means the software components need to interact in flexible ways and that such interactions cannot be hand-crafted at design time.

Against this background, we have designed a recommender system, and have used SoFAR [10], the Southampton Framework for Agent Research, to implement and deploy it across the authors' institutions in Europe. In this paper, we report on the design, the implementation and our practical experience with the system. Our contributions are:

1. Design of a modular large-scale recommender system;
2. Design of a marketplace approach to coordinate recommendations presented to the user;
3. Practical deployment of the system across multiple European sites;
4. Empirical evaluation of the system.

This paper is organised as follows. We describe the structure of the recommender system, and in particular we discuss its modular nature (Section 2). We then show how a marketplace can be used to rank multiple recommendations from heterogeneous methods (Section 3); our investigation includes an analytical study of the key parameters of the marketplace, and a simulation of its behaviour. We then describe the deployment of our recommender system across several European institutions and discuss the results of an evaluation process we undertook (Section 4). This is followed by some related work and a conclusion.

2 A Multi-Agent Recommender System

For many users, the "point and click" paradigm has become the preferred method for accessing information. Browsers on desktops or handheld devices are frequently used for transparently accessing Web pages, databases and knowledge

bases [18]. However, it has become apparent that there is a need for tools that can guide users in their information finding and navigation activities [5].

To this end, in this Section, we present a modular multi-agent recommender able to assist users in finding documents relevant to their current context; these documents may be owned by themselves or by other users. The recommender is composed of a collection of agents operating together to recommend documents to users, while they access information using their browser. Recommended documents are displayed in a browser sidebar, while the user navigates information in the main browser window. A *User Agent* is responsible for all interactions with the user, and for displaying recommended documents in the browser sidebar. Additionally, the User Agent acts as the user's representative in the recommender system.

The different agent functionalities have been categorised in three groups, as illustrated by Figure 1: *(i)* Agents managing information, *(ii)* Agents acting as the community memory, and *(iii)* Agents computing recommendations. Each of these will now be dealt with in turn.

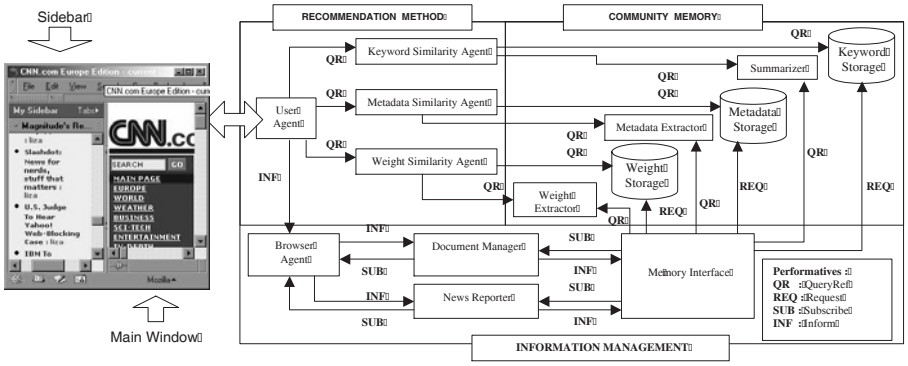


Fig. 1. Recommender Architecture

2.1 Information management

Recommended documents are computed from the *community memory*, which is a store of information created and updated by users. Some agents are responsible for managing information so that the community memory can be maintained and expanded over time. The more comprehensive the information is, the higher the likelihood of more documents or higher quality documents being recommended.

The system allows users to bookmark documents; references to these, expressed as URLs, are stored into the community memory. We define an *Information Manager* as an agent able to manage the information pertaining to a

document referred to by a URL. There exist two instances of Information Managers in the current version of the system. The *Document Manager* Agent is responsible for handling static documents, whereas the *News Reporter* Agent deals with dynamic documents, by periodically downloading the documents associated with the URL. An example of dynamic document is a news site such as www.cnn.com.

Other categories of documents can also be added to the system such as images or audio files. Being modular and extensible, adding a new Information Manager to manage a new type of document is simple. An Information Manager has to subscribe to the *Browser* Agent, declaring the type of document it is interested in. In return, the Browser Agent receives from the User Agent requests for adding and removing documents from the community memory, and sends them to all agents that have subscribed to such notifications. The Information Managers interact with the *Memory Interface* that calls information extraction methods in the community memory on new documents to be added, and that stores and deletes such information upon request.

2.2 Community Memory

The community memory is the source of information used to compute recommendations. The memory consists of agents acting as stores of information about documents (*Storage Agents*) and of agents able to extract information (*Information Extractor Agents*). A pair of Storage and Information Extractor Agents supports a specific recommendation method; for example, the *Keyword Storage* and *Summarizer* are agents acting as the memory for keyword-based recommendations. If a new recommendation method is added to the system, a new pair of Storage and Information Extractor Agents needs to be added to the community memory.

Three functions are supported by a pair of Storage and Information Extractor Agents. *(i)* Each pair of agents serves the need of the associated recommender method (implemented by a similarity agent to be explained) by returning the required information upon requests. *(ii)* A Storage Agent acts as a repository for information about documents bookmarked by users in the community. At the time of creating a bookmark, the user decides whether the information should be public or private; public information is shared by all users and can be created anonymously or with the user's identity. Private information is only visible by its creator. *(iii)* A Storage Agent must also support requests to delete bookmarks from the memory.

2.3 Recommendations

Recommended documents are characterised by their degree of similarity with the document currently viewed by the user. As there is not a universal method of deciding the similarity of two documents, we have introduced the idea of an abstract relationship "`SimilarDocs`" that represents the similarity of two documents referred to by URLs. When the user navigates to a new document,

the User Agent issues a query requesting to find out all the documents that bear a similarity with the current document; agents that are able to recommend similar documents upon receiving such a query are called *Similarity* Agents. Currently, we have defined three Similarity Agents based on different recommendation methods.

Note that we are *not* developing new information extraction and retrieval algorithms in this work; rather, we are working with standard methods (such as keyword and metadata similarity discussed in the following paragraphs). Our contribution is in integrating the various methods to produce high quality recommendations that are consistent with the user's context.

The *Keyword Similarity* and *Metadata Similarity* Agents recommend documents based on similar keywords and metadata. For the *Weight Similarity* Agent, the similarity of two documents is determined by using the cosine similarity function, over a vector representation computed by the "term frequency, inverse document frequency" method, as used in the recommender described in [5].

There are many other measures according to which a document can be regarded as related to the currently viewed document, such as similarity measures based on the document's context or on colour histogram. Envisioning more recommendation methods to be developed, the system allows designers to subclass the `SimilarDocs` relationship into specialised definitions. New Similarity Agents implementing different comparison algorithms can then simply be plugged into the system. They would need to advertise their presence to the User Agent, which would at once be able to query them, using the relation `SimilarDocs`. The `SimilarDocs` abstract concept has therefore become the agreed interface between UserAgents and Similarity Agents.

3 The Marketplace

From our practical experience with the system (detailed in Section 4), we observed that the user becomes very quickly swamped by recommendations suggested by agents. Such a phenomenon is further amplified by the modular nature of the system, which allows us to bring new agents in the system in a seamless manner.

To combat this, we decided it was necessary to introduce a mechanism that was able to sort and select the recommendations according to their relevance to the user[1]. In this context, we decided to adopt a market-based approach since such systems are an efficient means of allocating scarce resources in dynamic and open systems [2]. The market operates according to the following metaphor. A user and the user agent acting on their behalf are selling browser space where

[1] Some feedback from the user is required in order to design a system that is able to tailor information to that user. In this paper, we assume that such a feedback exists; our focus is not on how we obtain it and we may use techniques such as user ratings [5], bookmark information, or printing logs.

information may be displayed (the left-hand side of the browser window in Figure 1). Information providers are keen to get their recommendations advertised in the user's browser, and compete in a marketplace, ready to pay for such advertisements. This first flow of currency, from information suppliers to users, needs to be counter-balanced by another flow, so that the system can reach an equilibrium. Our second flow of currency is derived from the user's feedback and takes the form of a reward to the agents that provided useful recommendations to the user. The intent of such a marketplace is that suppliers of useful recommendations will be rewarded and will increase their profit, while suppliers of information not deemed relevant will receive no reward, and so will be less able to trouble users. In the rest of the section, we describe the design of this market place and its evaluation.

3.1 Rationale

The fundamental observation is that in a modular recommender system like ours, recommendation methods will be designed and implemented by different people who will use different measures of relevance. It is therefore not possible to sort recommendations according to their relevance only.

Our idea is to introduce a price[2] that recommendation suppliers are ready to pay to have their information displayed. The bid price is seen as an adjustive multiplicative factor to the relevance, because an auctioneer ranks recommendations according to the product of their relevance and price. The price will vary according to the dynamics of the marketplace, reaching an equilibrium for each agent, determined by its spending and income. The intent is that an agent uses the feedback that it gets from the user in the form of a reward to adjust its bid price in order to be able to compete with other bidders, which may use different measures of relevance. The resulting equilibrium is such that the product of the relevance and price represents an absolute comparable value across recommendation methods.

3.2 Marketplace Overview

Figure 2 shows how the marketplace is introduced into the recommender system; due to the lack of space, we did not represent all the other agents which remain the same as in Figure 1. In the market-based recommender, the User Agent sends a request for N recommendations to the *Auctioneer*. Upon receiving this request, the Auctioneer posts a call-for-bids, which consists of a currently displayed document, an initial price, and the number N of requested recommendations. The Auctioneer opens a new auction to allow bidders to submit their bids. Each *Bidder* Agent is associated with a Similarity Agent and acquires from it recommendations in the form of URLs with associated relevances. Bidders then form their bids, each consisting of a URL, a bid price, and a relevance, and send

[2] This is only an abstraction that we introduce in order to make the algorithm work rather than a real currency.

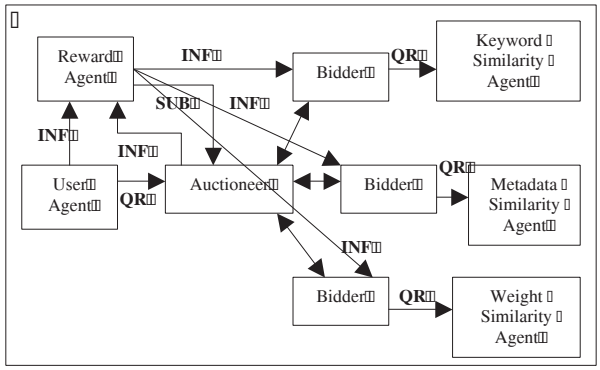


Fig. 2. Market-Based Recommender

them to the Auctioneer. The Auctioneer sorts these bids in descending order according to the product of bid price and relevance (as discussed in Section 3.1). Then, it selects the N best bids — a process called *short-listing* — and for each of them, it obtains a payment from the corresponding bidder, and forwards the bids to the User Agent.

The user's feedback is based on the recommendations that the user elects to follow; the user's selection is observed by the User Agent, which in turn notifies the Reward Agent. The *Reward* Agent is responsible for sending a reward back to the Bidder that generated the selected URL.

3.3 Simulation

In order to design the rules of the marketplace and to understand their dynamics, we have used our implementation to perform simulations of the auctions described above.

In the simulated environment, bidders generate the relevance of documents randomly from a standard uniform distribution; similarly, the bidder to be rewarded is the first in the ranking returned by the auctioneer. Therefore, bidders have an equal probability of being short-listed and even rewarded. Knowing that relevances are generated randomly and that the bid price is the variable factor that gets adjusted by the market mechanism according to the relevance of the bid, it makes little sense to use a single bid price for all possible relevances. Therefore, we split the range of possible relevances $[0, 100]$, expressed as a percentage, into $\mathcal{C}$ equal categories. When a bidder determines what price to give a bid of a relevance R, it will use data from the most recent auction during which a bid with a relevance of the same category was produced by the bidder.

The auctioneer's behaviour is shown in Figure 3, while a bidder's behaviour can be found in Figure 4. When forming a bid for a given relevance, a bidder refers to the most recent auction A_k with a relevance in the same category as the

current bid. The bidder adjusts its bid price by a multiplication factor X only if it was rewarded in auction A_k. If the bid in auction A_k was not short-listed, its price was too low, and therefore the bidder increases its bid price by Y. If the bid in auction A_k was short-listed but not rewarded, the bidder lost some credit and attempts to minimise its loss for the next round by decreasing its bid price by Z; note that the bidder has no way of obtaining the winner's price and therefore cannot try to compete with it by increasing its bid price in a rational manner.

Number of auctions in the simulation:	W
Initial price:	P_i
Sequence of auctions:	$A_1, A_2, \ldots, A_W$
Number of recommendations requested by the user:	N

Initial Configuration:

- *The Auctioneer gets a request for N recommendations from the User Agent.*

For the auction $A_i, (0 < i \leq W)$, the Auctioneer

- *Starts auction A_i by sending a call-for-bids containing (url, N, P_i, δ) to all bidders and a timeout message that specifies the time period during which all bidders are allowed to submit their bids;*
- *Gets bids from the bidders;*
- *Sorts all the bids according the product of the bid price and the relevance;*
- *Gets payment for N best bids, and notifies unsuccessful bids;*
- *Forwards the N best bids to the User Agent;*
- *Signals the end of auction A_i.*

Fig. 3. *Auctioneer's Behaviour*

3.4 Market Mechanisms

In this Section, we discuss the market mechanisms that we have put in place. The rewarding mechanism is the key element that needs to be defined. The Auctioneer sorts all bids and short-lists the N best ones. Bids numbered according to their ranking satisfy the following inequalities:

$$B_1 R_1 \geq B_2 R_2 \geq \ldots \geq B_N R_N, \tag{1}$$

where B_i and R_i denote the bid price and relevance of bid i in the ranking with $(1 \leq i \leq N)$.

When defining the reward, our goal is to ensure that the marketplace remains fair and that bid prices do not spiral up in an uncontrollable manner, which would inevitably bankrupt some of the agents, or prevent them from competing.

Number of auctions in the simulation:	W
Number of the selected URLs *requested by the user:*	N
Sequence of auctions:	$A_1, A_2, \ldots, A_W$
The reward for the winning bidder in auction i for bid j:	$\sigma_{i,j}$
Sequence of categories of relevance:	$C_1, C_2, \ldots, C_{\mathcal{C}}$
The cost paid by a bidder in auction i for bid j:	$\rho_{i,j}$
Percentages by which a bidder adjusts its price:	X, Y, Z

Initial Configuration:

- *Allocate an amount of credit to each bidder*

For the auction $A_i (1 \leq i \leq W)$, the bidder:

- *Generates N random relevances and calculates a bid price for each of them. Forming bid $j (1 \leq j \leq N)$, for relevance R_j:*
  - *Let us assume that relevance R_j comes from category $C^c (1 \leq c \leq \mathcal{C})$. Let k denote the most recent auction $A_k (1 \leq k < i)$, where the bidder made a bid with a relevance from C^c; let ℓ be the number of that bid then; let $B^c_{k,\ell}$ be its bid price. The bid price $B^c_{i,j}$ for information with relevance R_j in the current auction should be:*
    If $\sigma_{k,\ell} > 0$, then $B^c_{i,j} = XB^c_{k,\ell}$ *(Rewarded)*
    If $\sigma_{k,\ell} = 0$ and $\rho_{k,\ell} > 0$, then $B^c_{i,j} = B^c_{k,\ell}(1 - Z)$ *(¬ Rewarded and short-listed)*
    If $\sigma_{k,\ell} = 0$ and $\rho_{k,\ell} = 0$, then $B^c_{i,j} = B^c_{k,\ell}(1 + Y)$ *(¬ Rewarded and ¬ short-listed)*
  - *If A_i is the first auction for a relevance in category C^c that the bidder produced, then $B^c_{i,j} = P_i$.*
- *If it has enough credit, informs the Auctioneer of the bids:*
  - *For each bid j accepted by the Auctioneer, it has to pay the bid price $\rho_{i,j} = B^c_{i,j}$ for it (otherwise $\rho_{i,j} = 0$);*
  - *For each bid j which leads to a reward, this bidder gets $\sigma_{i,j}$ from the reward agent; (otherwise $\sigma_{i,j} = 0$).*
- *Waits for the next auction to be signalled after getting a message indicating the end of the auction from the Auctioneer.*

Fig. 4. *Bidder's Behaviour*

In our initial design, we chose a reward that was directly related to the bid price and relevance of the winner. Our simulations indicated that this gave a decisive advantage to the first winner, which could artificially increase its bid price in the second auction; if then successful, it would keep increasing its bid price, outbiddings all other agents. To combat this, we define the reward for the winning bidder (with bid price B_1 and relevance R_1) as a function of the *second best bid*:

$$\sigma = \frac{B_2 R_2 + \delta}{R_1} \tag{2}$$

with $\delta(\delta > 0)$ a public parameter, which we also include in the call-for-bids sent by the Auctioneer. For similar reasons, the parameter δ must be chosen with a reasonable value to maintain the fairness of the market.

3.5 Rational Bidding

Having defined the auction parameters, we discuss the values of the parameters X, Y, Z that a rational agent may adopt. Referring to the notation of Figure 4, a bidder needs to decide what its bid price $B^c_{i,j}$ is in auction A_i, for a bid of relevance R_j of category C^c, knowing that the most recent of its bids in that category took place in auction A_k, with bid price $B^c_{k,\ell}$, relevance R_ℓ, reward $\sigma_{k,\ell}$ and cost $\rho_{k,\ell}$.

Rational behaviour for a bidding agent was explained above. So now, let us study what the optimum value of X is, if the agent was rewarded in auction A_k. We assume that the second best bid in the next round of the auction will be characterised by price B^n_2 and relevance R^n_2. The bidder will attempt to win, which implies that:

$$B^n_2 R^n_2 \leq B^c_{i,j} R_j \tag{3}$$

Given its current credit K_i, it can estimate what its new credit K_{i+1} will be.

$$K_{i+1} = K_i + \frac{B^n_2 R^n_2 + \delta}{R_j} - B^c_{i,j} \tag{4}$$

The bidder will increase its credit if the reward is greater than the cost of publishing the recommendation:

$$\frac{B^n_2 R^n_2 + \delta}{R_j} - B^c_{i,j} \geq 0, \tag{5}$$

or

$$B^c_{i,j} R_j \leq B^n_2 R^n_2 + \delta. \tag{6}$$

So in order to win and make a profit, the bid price must satisfy equations (3) and (6), summarised in the following constraints.

$$B^n_2 R^n_2 \leq B^c_{i,j} R_j \leq B^n_2 R^n_2 + \delta \tag{7}$$

We assume that the bidder who submits a bid price B^n_2 with relevance R^n_2 submitted a price B^p_2 with relevance R^p_2 in the previous auction, and that the relevance of the bid is preserved $R^n_2 = R^p_2$. Assuming that all bidders are rational and follow the algorithm of Figure 4, we can therefore rewrite B^n_2, $B_{i,j}$ in equation (7), and we obtain:

$$(1 - Z) B^p_2 R^p_2 \leq B^c_{k,\ell} X R_j \leq (1 - Z) B^p_2 R^p_2 + \delta \tag{8}$$

Since bidders are independent of each other and bids are private, they ignore the behaviour of other bidders. However, the reward as defined in (2) can be rewritten for auction A_k.

$$\sigma_{k,\ell} = \frac{B^p_2 R^p_2 + \delta}{R_{k,\ell}} \tag{9}$$

After substituting (9) in (8), we obtain constraints on X:

$$\frac{(1-Z)(\sigma_{k,\ell}R_{k,\ell}-\delta)}{B^c_{k,\ell}R_j} \leq X \leq \frac{(1-Z)(\sigma_{k,\ell}R_{k,\ell}-\delta)+\delta}{B^c_{k,\ell}R_j} \tag{10}$$

There is a further constraint that X must remain positive to avoid generating negative prices, which would be meaningless: $X > 0$. So, the value of X is constrained by $LB < X \leq UB$, with:

$$LB = \max(0, \frac{(1-Z)(\sigma_{k,\ell}R_{k,\ell}-\delta)}{B^c_{k,\ell}R_j}) \tag{11}$$

$$UB = \frac{(1-Z)(\sigma_{k,\ell}R_{k,\ell}-\delta)+\delta}{B^c_{k,\ell}R_j} \tag{12}$$

For convenience, we introduce $x \in]0,1]$ as a percentage dictating the value of X as follows: $X = LB + x(UB - LB)$.

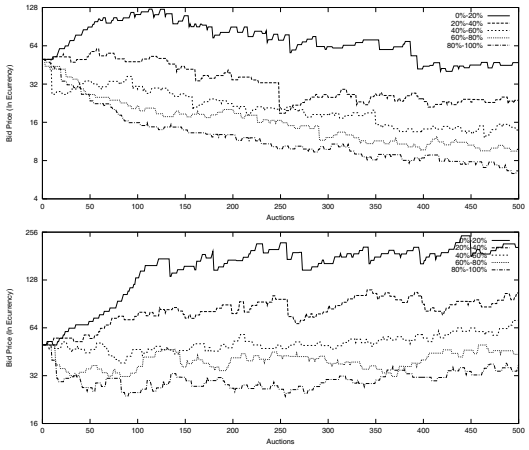
In the following section, we will show the effects of Y and Z. In summary, we have defined a sealed-bid one-shot auction system, where the reward allocated to the winner is a function of the price of the best bid coming from another bidder. A new auction is initiated every time the user visits a new URL.

3.6 Analysis

The main auction parameters are the initial price P_i and the reward variable δ defined as a proportion of P_i. From a bidder's viewpoint, the parameters X, Y, Z can be changed. In this section, we show the effect of these parameters through various simulations. Figure 5 shows the change of a bidder's bid price in 500 auctions for two different values of δ. In these simulations, we are using 6 bidders, 3 recommendations are requested by the user and only one agent gets rewarded; agents use five categories of relevance ($\mathcal{C} = 5$).

For all six bidders, the patterns of their bid prices are similar. Here, we only show the evolution of prices for one bidder taking part in the auction; prices are plotted for each of the 5 relevance categories adopted. The bids with the smallest relevance [0%-20%] have a higher price than other bids most of the time: since their relevance is low, the bid price has to be increased in order to be able to compete with bids with higher relevance. The time to reach such a level depends on the percentage by which the bid price is increased. Then, the bid price oscillates when a suitable level has been reached.

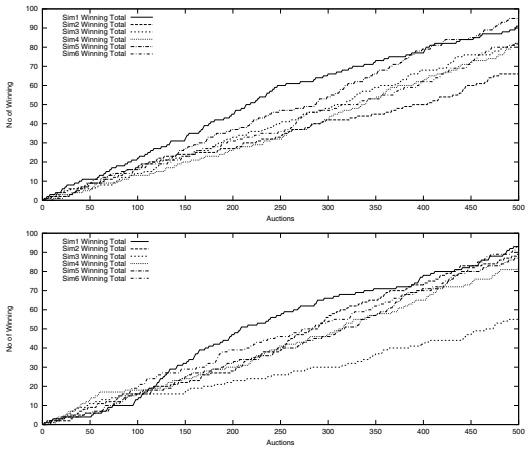
Once equilibrium has been reached, a difference between the two graphs is that, in a given category, the bid price in the top graph is higher than that in the bottom graph. This indicates that the bid price is decided by δ to some extent, the higher the reward (related to δ), the higher the bid price is, if all the other parameters remain the same. A low value of δ maintains equal opportunities amongst bidders. If δ has a high value, the range of possible values for X is wider. Therefore, the probability of making profit in the next round is higher. This leads to unequal opportunities of winning, which is shown in the Figure 6, where some bidders win less over the series of auctions.



$\delta = 10\%P_i$

$\delta = 90\%P_i$

Fig. 5. Bid Price ($x = 10\%, Y = 5\%, Z = 5\%$)

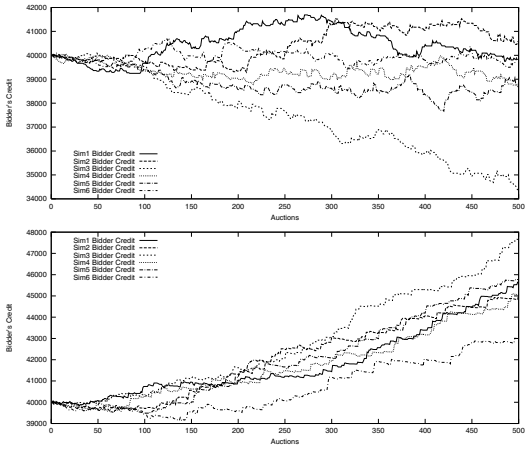


$\delta = 10\%P_i$

$\delta = 90\%P_i$

Fig. 6. Number of Wins ($x = 10\%, Y = 5\%, Z = 5\%$)

So far, we have obtained the simulation results that account for the effect of δ on the bid price, number of wins and credit. However, the unsatisfactory aspect is that at the end of 500 auctions, the bidders have a relatively low credit. As we discussed earlier, if we keep on increasing the value of δ, this may help keep the credit, but in some extreme cases, one of the six bidders ends up winning most of the time, which is in contradiction with our design objectives. As far as a bidder's credit is concerned, spendings cover short-listed bids, whereas incomes come from the reward obtained by the bidder. Given a fixed value of δ, bidders



$x = 10\%$

$x = 1\%$

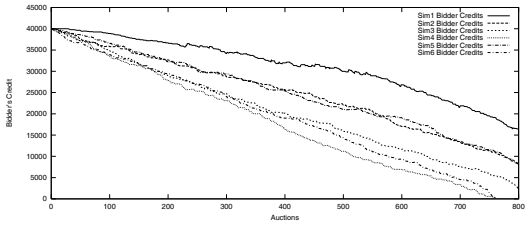
Fig. 7. Bidder's Credit ($Y = 5\%, Z = 5\%, \delta = 90\% P_i$)

can still maintain their credit by keeping the price of winning bids as low as possible.

The parameter X can take any value in the interval $]LB, UB]$, cf. equations (11) and (12), according to the percentage x. Taking a small value of x, the bid price is maintained close to the estimated bid price of the second bidder, which results in a maximised profit. The top graph in Figure 7 displays the credit during an auction, where x is set to 10%, whereas the bottom graph shows the credit for a smaller value of x (1%). We see that the credit increases over time in the latter case.

Being restricted by space, we only describe our observations about the roles of Y and Z in the auction without showing any simulation graphs. Higher values of Y and Z widen the range of bid prices. If the value of Z remains the same, then the higher the value of Y is, the higher the bid price increases. On the other hand, for the same value of Y, the higher the value of Z is, the lower the bid price becomes. Additionally, higher values of Z tend to preserve the agent's credit. Indeed, from equations (11,12), a higher value for Z implies a lower value for X.

Finally, we have compared our rational agents against bidders choosing their price randomly, and bidders continuously choosing the same price (the initial price). Figure 8 shows the behaviour of one rational agent against 5 bidders choosing random prices; a similar graph is obtained for bidders with a fixed price. The top curve is the credit of the rational agent, which shows that our strategy consisting of reducing the cost of publishing wins in the long term. The rational agent is more able to maintain its credit, and can therefore outlive agents not applying such a strategy.



$x = 10\%, Y = 5\%,$
$Z = 5\%, \delta = 90\% P_i$

Fig. 8. Rational Bidder against Random Bidders

4 Deployment and Evaluation

The recommender system has been implemented using the SoFAR agent system [10], and we have deployed it across several sites in Europe. The recommender system and SoFAR total about 120,000 lines of code. In this section, we report on our deployment experience, and our evaluation of the system.

Distributed Deployment. At first, our prototyping took place on a single machine, where we were running all the agents in a single JVM. When the system reached a satisfactory degree of stability, we separated the User Agents from the rest of the recommender system. Typically, each user would run a browser and a user agent on their machine; the user agent will discover through a lookup service the location of the recommender system and would interact with it over the network. At this stage, all agents were still running in a single laboratory in Southampton. The next stage has been to deploy User Agents in the different European sites. In particular, this required us to introduce some support for firewalls. This was our targeted deployment, which would enable us to evaluate the system.

Currently the community memory is a centralised component, but we want it to be distributed, created dynamically, and shared across the network. We are confident that no single solution will be suitable for all imaginable configurations. Therefore, we are aiming again at defining a modular architecture that could be configured in multiple ways. Our solution is to introduce a community proxy agent that could act as a recommender method in another community. This would allow communities to be linked to other communities through such proxies, hereby supporting a distributed organisation of communities.

Evaluation. We undertook a trial in order to evaluate the recommender system, without the marketplace, and then with the marketplace. We adopted three topics for the purpose of evaluation: "auctions", "mobile agents" and "games". In the first stage, we populated the community memory, by bookmarking a series of documents that we found relevant to each topic. In the second stage, we made use of the recommendation methods based on keywords, weights, and metadata. Each user involved in the trial visited a series of documents of their choice in these topics, and was recording recommendations from each method. In the third stage, we analysed the recommendations and drew the following conclusions.

In the evaluation, all the recommendations provided by the weight similarity method are from the same topic as the document being viewed; we observed a rate of success of over 99%. In general, many recommendations supplied by the keywords method belong to the same topic as the document currently viewed (77%); however a few recommendations are from different topics. Finally, we did not get any significant result about the metadata similarity, because only a small percentage of the pages visited contained metadata. In general, most of the recommendations for a document are from the desired topic. Recommendations from the weight-based method are more precise than those from the other two methods.

Another experiment was set up to assess the recommender system with a marketplace. We started with a given document. After following 15 recommendations from the same agent, we returned to the initial document. We observed that the ranking of recommendations had changed. The recommendation supplied by the agent was now top of the recommended list. This experiment shows that the marketplace allows a recommendation method to score highly if rewarded regularly for its recommendations.

Extensibility. We also evaluated the extensibility of our architecture, by integrating new recommendation methods into the framework. *Without inside knowledge* of the system, two new recommendation methods were successfully integrated in a single day — the main work was to wrap the existing functionality with the `SimilarDocs` relation. Both of the recommendation methods are based on the notion of trails [14], which, unlike guided tours, constitute a specific path a user has chosen in exploring a set of documents. Two types of trail-based recommendations were provided. In the first type, documents are recommended based on people's browsing behaviour. The agent's knowledge has been primed with about half a year of proxy logs from 50 people working in an IT related research organisation. The agent suggests related URLs based on the order of traversal (as recorded in the proxy logs) and ranks them according to their frequency. The second type of trail-based recommendation agent uses an Artificial Neural Network (ANN) and the same proxy data as a training set for the network. The agent actually suggests related domains (rather than complete URLs). Domains are considered related if users traverse often between them.

Discussion. Our practical experimentation shows that introducing a market mechanism in a recommender system appears to be a useful mechanism to control the ranking of recommendations made by multiple heterogeneous recommender methods. Some practical but challenging aspects still need to be investigated to make it a routinely useable system. *(i)* Obtaining user's feedback that really reflects how a user found a document useful. *(ii)* How to handle bidders that go bankrupt: their inability to perform well during a browsing session may not reflect on their capacity to deliver useful recommendations in different contexts. *(iii)* The system should be able to react promptly to support users' changes of context.

5 Related Work

There have been several other attempts to develop large-scale, agent-based distributed information management systems. The computational market of SIGMA, System of Information Gathering Market-based Agents [7], is a model of decentralised decision making for the task of information filtering in multi-dimensional spaces such as the Usenet netnews. Most related to this work, is the University of Michigan Digital Library [4] which uses a variety of marketplaces to moderate a range of digital library services, the InfoSleuth system [13] which exploits user, middleware and resource agents to deliver complex information services, and Retsina which provides a rich infrastructure of user, task and middleware agents for information management activities [17]. Mullen and Wellman's Simple Computational Market model [12] aims to tackle the problem of when and where to establish mirror sites for the more popular information services. Competitive agents choose to set up mirrors based on going prices for network bandwidth, computational resources, and the information service. In contrast to this work, however, here we report on the application of our technology to a particular information management task (that of recommending relevant documents). This means we need to specify and implement specific marketplace structures and endow agents with specific decision making capabilities.

Some authors have also studied user feedback. For instance, Baclace [1] describes a personal, adaptive recommendation system that uses an exit-question to rate documents. Ratings collected in this way have the property of being minimally articulated, which is an advantage when preferences are difficult to describe. The goal of this work is to direct readers' attention in a self-correcting way with as little user effort as possible, and to utilise ratings automatically to connect readers whose interests concur.

Filtering systems have also been the subject of investigation. From a user's viewpoint, Sheth [16] demonstrates a learning approach to personalised information filtering using relevance feedback and genetic algorithm. The system is a collection of information filtering interface agents which has learning capabilities of specializing users' interests and exploring new potential domains. On a larger scale, the Stanford Information Filtering Tool (SIFT) [19] is a tool to provide information dissemination service, which supports full-text filtering using well-known information retrieval methods and is capable of processing large volumes of information against a large number of profiles.

The system described in the current paper could be applied to other forms of recommendations, including the recommendation of expertise as in its agent-based predecessor, MEMOIR [3]. Letizia [8] was an early agent-based recommender system which is also coupled to a Web browser; it has the additional feature of scouting ahead of the user's current position on the Web. Amalthea [11] also uses a notion of "credit" in the context of an evolutionary approach to information filtering and discovery but does not have the full flexibility of a marketplace.

6 Conclusion

We have designed an agent-based modular distributed recommender system and have deployed it over multiple institutions in Europe. In order to be able to rank recommendations provided by multiple methods and to select the most relevant ones, we have conceived a new marketplace, based on repeated single-shot sealed-bid auctions, with a reward based on the second best bid price. We have performed an analytical study of the marketplace, and using our implementation, we have extensively simulated it, showing that it achieves fairness and that it rewards agents that make useful recommendations. Using our deployed system, we have undertaken a first evaluation, which validated the key properties of the recommender.

We will use this deployed architecture in our future work. First, we wish to refine the notion of community memory, making it distributed across multiple sites, and allowing users to dynamically select communities they want to extract information from. Second, we want to investigate the means by which the market mechanism can adapt to the behaviour of a user who works on simultaneous multiple tasks. Third, users' feedback is crucial in this context and needs to be further addressed.

7 Acknowledgement

This research is funded in part by QinetiQ and EPSRC Magnitude project (reference GR/N35816).

References

1. Paul E. Baclace. Personal information intake filtering. In *Bellcore Information Filtering Workshop*, November 1991.
2. Scott H. Clearwater, editor. *Market-Based Control. A Paradigm for Distributed Resource Allocation.* World Scientifid Publishing, 1996.
3. David C. DeRoure, Wendy Hall, Siegfried Reich, Aggelos Pikrakis, Gary J. Hill, and Mark Stairmand. Memoir – an open framework for enhanced navigation of distributed information. *Information Processing & Management. An International Journal*, 37:53–74, 2001.
4. E. H. Durfee, D. L. Kiskis, and W. P. Birmingham. The agent architecture of the University of Michigan Digital LIbrary. *IEE Proc on Software*, 144(1):61–71, 1997.
5. Samhaa El-Beltagy, Wendy Hall, David De Roure, and Leslie Carr. Linking in context. In *Proc The Twelfth ACM Conference on Hypertext and Hypermedia (Hypertext '01)*, pages 151–160. ACM, ACM Press, August 2001.
6. N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 2000.
7. G. Karakoulas and I. Ferguson. A computational market for information filtering in multi-dimensional spaces, 1995.
8. Henry Lieberman. Letizia: An agent that assists web browsing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 1995.

9. Pattie Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7):31–40, July 1994.
10. Luc Moreau, Nick Gibbins, David DeRoure, Samhaa El-Beltagy, Wendy Hall, Gareth Hughes, Dan Joyce, Sanghee Kim, Danius Michaelides, Dave Millard, Sigi Reich, Robert Tansley, and Mark Weal. SoFAR with DIM Agents: An Agent Framework for Distributed Information Management. In *The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pages 369–388, Manchester, UK, April 2000.
11. A. Moukas. Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem. In *Practical Application of Intelligent Agents & Multi-Agent Technology, London*, 1996.
12. Tracy Mullen and Michael Wellman. A simple computational market for network information services. In *Proceedings of the First International Conference on Multiagent Systems.*, pages 283–189, Menlo park, California, 1995. AAAI Press / MIT Press.
13. M. H. Nodine, J. Fowler, T. Ksiezyk, B. Perry, M. Taylor, and A. Unruh. Active Information Gathering in InfoSleuth. *Cooperative Information Systems*, 9(1-2), 2000.
14. Siegfried Reich, Leslie A. Carr, David C. DeRoure, and Wendy Hall. Where have you been from here? Trails in hypertext systems. *ACM Computing Surveys — Symposium on Hypertext (published as electronic supplement)*, 31(4es), December 1999.
15. Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
16. Beerud D. Sheth. A learning approach to personalized information filtering. Master's thesis, 1994.
17. K. Sycara and D. Zeng. Coordination of multiple intelligent software agents. *Cooperative Information Systems*, 5(2-3), 1996.
18. Mark J. Weal, Gareth V. Hughes, David E. Millard, and Luc Moreau. Open hypermedia as a navigational interface to ontological information spaces. In *Proceedings of the Twelveth ACM Conference on Hypertext and Hypermedia HT'01*, pages 227–236, Aarhus, Denmark, August 2001.
19. T. Yan and H. Garcia-Molina. SIFT—A tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–186, New Orleans, 1995.