Learning Users' Interests by Quality Classification in Market-Based Recommender Systems

Yan Zheng Wei, Luc Moreau, and Nicholas R. Jennings

Abstract—Recommender systems are widely used to cope with the problem of information overload and, to date, many recommendation methods have been developed. However, no one technique is best for all users in all situations. To combat this, we have previously developed a market-based recommender system that allows multiple agents (each representing a different recommendation method or system) to compete with one another to present their best recommendations to the user. In our system, the marketplace encourages good recommendations by rewarding the corresponding agents who supplied them according to the users' ratings of their suggestions. Moreover, we have theoretically shown how our system incites the agents to bid in a manner that ensures only the best recommendations are presented. To do this effectively in practice, however, each agent needs to be able to classify its recommendations into different internal quality levels, learn the users' interests for these different levels, and then adapt its bidding behavior for the various levels accordingly. To this end, in this paper, we develop a reinforcement learning and Boltzmann exploration strategy that the recommending agents can exploit for these tasks. We then demonstrate that this strategy does indeed help the agents to effectively obtain information about the users' interests which, in turn, speeds up the market convergence and enables the system to rapidly highlight the best recommendations.

Index Terms—Information filtering, machine learning, recommender systems, markets.

1 Introduction

Recommender systems have been widely advocated as a way of coping with the problem of information overload. Such systems help make choices among recommendations from all kinds of sources for users who do not have sufficient personal experience of all these alternatives [1]. Many recommender systems have been developed, but they are primarily based on two main kinds of information filtering techniques: *content-based filtering* and *collaborative filtering*. The former makes recommendations by analyzing the similarity between the objective properties of the items (such as textual contents of documents) that are ready to be recommended and those that have previously been marked as liked by the user. The latter makes recommendations by analyzing the subjective properties of items (such as people's opinion of food taste or music style) putting forward items that have been deemed appropriate by people who have similar interests to the present user. However, both kinds of techniques have their weaknesses. Content-based filtering techniques cannot easily recommend nonmachine parsable items (such as audio and video items). Additionally, content-based techniques do not have an inherent method for generating serendipitous finds because they tend to recommend more of what the user has already seen [2], [3]. In contrast, collaborative filtering techniques do not have such weaknesses, but they fail to accurately predict the present user's interests when there is an insufficient number of peer users (sharing similar interests). Additionally, all collaborative recommender systems share the cold start problem: When new users start off with empty profiles of interests and must train a profile from scratch, they share little interest with others and, therefore, receive poor recommendations [1], [4]. Against this background, it has been argued that there is no universally best method for all users in all situations [5].

Given this, the available recommendation methods are often unable to make suggestions that satisfy the user. This is because these different methods use different metrics and different algorithms to evaluate different properties (either objective or subjective) of the items they may recommend. Thus, the rating of the quality of the same recommendation can vary dramatically from one method to another (e.g., one method may think the item is very relevant for the user, another may think it moderately relevant, while yet another may believe it is completely irrelevant). Here, we term this evaluation the method's *internal quality* (INQ) to reflect the fact that the rating is as perceived by the internal algorithm/processes of the particular method. However, a high INQ recommendation from one method does not necessarily mean the recommendation is any more likely to better satisfy a user than a low INQ item suggested by another. Ultimately, whether a recommendation satisfies a user can only be decided by that user (Fig. 1 exemplifies this phenomenon). Therefore, we term the user's evaluation of a

- *Y.Z. Wei is with the British Telecommuniations, MLB1/pp12, Orion Building, Adastral Park, Martlesham Heath, Ipswich IP5 3RE, United Kingdom. E-mail: yanzheng.wei@bt.com.*
- *L. Moreau and N.R. Jennings are with the Intelligence Agents Multimedia Group, School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, United Kingdom. E-mail: {L.Moreau, nrj}@ecs.soton.ac.uk.*

Manuscript received 6 Jan. 2005; revised 29 Mar. 2005; accepted 1 Apr. 2005; published online 19 Oct. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDESI-0009-0105.

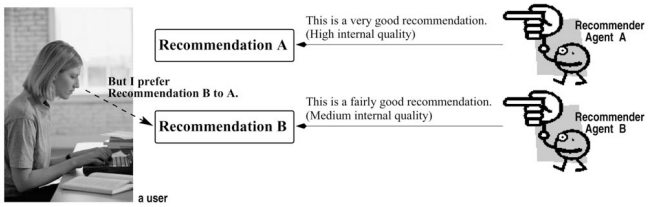
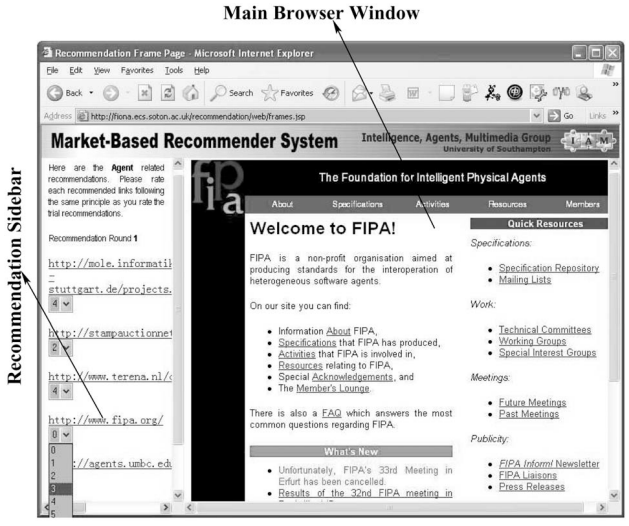


Fig. 1. Different views of quality of recommendations.

recommendation the *user's perceived quality* (UPQ). Now, an efficient recommender system is one that achieves a good correlation between the INQ values it generates and the corresponding UPQ values. However, to date, there is little in the way of work that can efficiently correlate the recommendation methods' INQs to the users' UPQs.

To overcome this problem, we believe that the way to move forward in this area is to develop an overarching system that incorporates multiple different recommendation methods, that lets recommendations from whatever methods that are available or get developed compete with each other, and that then automatically selects only the best items (whose INQs consistently reflect the top UPQs) to pass through. In order to validate and verify the feasibility of our approach, we have previously designed a market-based recommender system to deal with the "where to go next" problem by presenting recommendations (represented as URLs) that are relevant to the users' current browsing context (Fig. 2 shows an example of our system). To date, we have shown that a well-structured information marketplace with the appropriate incentives can function effectively in making decisions among the multiple different recommendation methods [6], [7]. Specifically, in our system, the various recommendation methods, represented as recommender agents, compete to advertise their recommendations to the user. Through this competition, only the



By convention, recommendations in this kind of system are usually displayed in a separate window without interrupting the user's current navigation context. Thus the main window displays the user's current context (the page being viewed).

Fig. 2. Browser with recommendations.

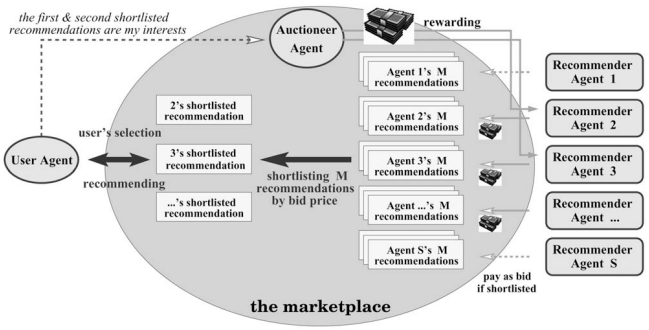


Fig. 3. The market protocol.

best recommendations (from whatever source) are presented to the user. Essentially, our system uses a particular type of auction (generalized first price sealed bid) and a corresponding reward regime to incite the agents to align their bids with the user's preferences. To this end, Fig. 3 depicts the auction protocol and the reward mechanism (see [6] for more details of the market protocol and its Pareto efficiency and social welfare maximization properties). In short, the system functions by ensuring that its recommendations that the user considers good are encouraged by receiving a reward, whereas poor ones are deterred (because they have to pay to advertise their recommendations in the sidebar but they receive no reward). Thus, the market acts as a feedback mechanism that helps agents to correlate the INQs of their recommendations to the desires (UPQ) of the user.

Now, while the market protocol works effectively as a coordinator for the overarching system, an open problem from the point of view of the individual recommender agents remains: *Given a set of recommendations with different INQ levels, in what order should an agent try to advertise them so that it can learn the user's interests as quickly as possible, while still maximizing its revenue?* Thus, for example, the agent could bid the items that have never been advertised to the user, which would allow it to learn the user's interests quickly, but could also result in its losing money. Conversely, the agent could always bid those items that have been highly rewarded, so ensuring a good return, but it would take a very long time to learn the extent of the user's interests. While this problem is couched in the context of our specific system, this is a general problem that all recommender and information filtering systems face. Thus, even though they may not have a currency or an explicit reward, they still need to determine the user's preferences as quickly as possible, while still making good suggestions, in order to make effective recommendations. From a data engineering perspective, the marketplace needs high quality recommendation inputs with as little irrelevant information as possible. Thus, from the point of view of the marketplace, learning user's interests by the constituent recommender agents can be seen as a data preparation (or information enhancement) process for the marketplace [8], [9]. In this way, the agents filter out the noisy and redundant items from their recommendation pools.

To overcome this problem, we have developed a *quality classification* mechanism and a reinforcement learning strategy for the agents to learn the user's interests [10].

Intuitively, to make good suggestions, an agent needs to classify its recommendations into different categories based on some specific features of the recommendations and then suggest the right categories of items to the user according to his (or her) interests. In our context, each agent classifies its recommendations into different INQ levels (e.g., very good, good, bad, etc.) based on its internal belief about their relevance to the user's context. Then, to assist an agent to direct the right categories of recommendations to the user, we developed a concomitant reinforcement learning strategy. This strategy enables an agent to relate the user's feedback about its recommendations to its INQ measure and then to put forward those recommendations that are consistent with this. This is important because the more effectively an agent relates its recommendations to the user's interests, the better it serves the user and the more rewards it receives.

Against this background, this paper advances the state of the art in the following ways: First, a novel reinforcement learning strategy is developed to enable the agents to effectively and quickly learn the user's interests, while still making good recommendations. Second, and perhaps more important, we demonstrate how our learning strategy, coordinated through the marketplace, can be viewed as a quality classification problem and how the marketplace assists the classification and aligns the right recommendations to the right people. Third, from an individual agent's point of view, we show that the learning strategy enables an agent to maximize its revenue. Finally, we show that when all agents adopt our strategy, the market rapidly converges and makes good recommendations quickly and frequently.

The remainder of this paper is structured in the following manner: Section 2 briefly recaps the basics of our multiagent recommender system and highlights the problem an individual agent faces in it. Section 3 details the design of our learning strategy. Section 4 empirically evaluates this design. Section 5 outlines related work in terms of reinforcement learning and market-based recommendations. Section 6 concludes and points to future work.

2 The Quality Classification Problem for Market-Based Recommendations

This section briefly describes how our marketplace coordinates bids with recommendations and rewards from the point of view of individual recommender agents and, then, goes into more detail on the quality classification problem that each agent faces.

We outline our market-based recommender in terms of the sequencing of the main market processes (see the circled numbers in Fig. 4). First, when the market calls the agents for a number (M) of recommendations, each agent submits M items and bids a price for each of them. Second, the market ranks all recommendations from all the agents in decreasing order of their prices and displays the M items with the highest bid prices to the user. Consequently, each agent with displayed items pays an amount of credit (equal to the amount it bids) for each of the corresponding displayed items it advertises. Third, the user then visits a number (between 0 and M) of the displayed items and gives

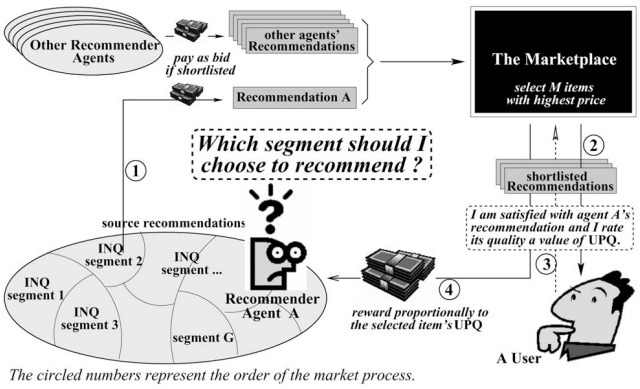


Fig. 4. An individual agent's quality classification problem.

a rating (i.e., a UPQ value) to each visited item based on his satisfaction. Fourth, the market rewards the agents that supplied the recommendations that were assigned a positive UPQ. The amount of credit they receive is proportional to their UPQ values (see [6] for the details of the mechanism and the proof that this mechanism is Pareto optimal with respect to the group of rewarded agents and that it maximizes their social welfare). Thus, the system completes one round of operation and proceeds with another following the above four steps.

In this context, the role of the reward mechanism is to provide the agents with incentives to align their bidding behavior with the interests of the user. From the point of view of an individual agent, however, it needs to learn which recommendations the user prefers. To do this, agents classify their recommendations into a predetermined number (G) of categories (or segments) based on their INQs (e.g., in the simplest case, where $G = 2$, an agent could classify "bad" recommendations as those with an INQ of less than 0.5 and those with an INQ between 0.5 and 1.0 as "good") and then they relate these INQs to the UPQs. Intuitively, the more the user is satisfied with a recommendation, the more reward the corresponding agent receives. Thus, an agent that has sufficient experience of the user's feedback can learn the user's interests by correlating its recommendations (and their corresponding INQ segments) to the rewards (that reflect their UPQs) they receive [7]. This, in turn, enables a self-interested agent to consciously make recommendations from those INQ segments that correspond to high UPQs so that it can best satisfy the user and, thus, gain maximal revenue. To effectively compute the agents' revenue, we define an agent's *immediate reward* (made from a recommendation displayed to the user in one auction round) as the reward it received minus the price it has paid for the advertisement.[1] With this, what an agent needs to do is to learn how much immediate rewards, on average, it can expect for items in each category (i.e., each INQ segment). We term this average immediate reward for

1. Agents pay nothing for items they put forward that are not displayed to the user (this occurs when other agents are willing to pay more to advertise their recommendations). By definition, an immediate reward may either be positive or negative. If a displayed recommendation is not selected by the user or if it has paid too much to display an item, the corresponding agent's immediate reward is negative since it has paid for the display and received less reward.

each INQ segment an agent's *expected revenue*. Thus, a self-interested agent can maximize its revenue by frequently bidding recommendations from the segments with high expected revenue. Therefore, an agent's recommending task can be seen as a quality classification problem and it needs to align the user's preferences with its INQ segments (reflected by expected revenue) and meanwhile make maximal revenue.

However, when an agent starts bidding in the marketplace, it has no information about how much revenue it can expect for each segment. Therefore, the agent needs to interact in the marketplace by taking actions over its G segments to learn this information (as per Fig. 4). In this way, an agent can produce a profile of such information from which it can form an optimal strategy to maximize its overall revenue. In this context, the agent's learning behavior is on a "trial-and-error" basis. The agent bids its recommendations and receives the corresponding feedback in a manner that good recommendations gain rewards, whereas bad ones attract a loss. This kind of trial-and-error learning behavior is exactly what happens in *Reinforcement Learning* [11]. Thus, to be more concrete, an agent needs an algorithm to learn the expected revenue over each segment. In addition, it also needs an exploration strategy to make trials on its G segments such that it strikes a balance between learning as quickly as possible, while still maximizing revenue.

3 The Learning Strategy

This section details the design of an agent's learning algorithm and exploration strategy in Sections 3.1 and 3.2, respectively. The overall strategy is then pulled together in Section 3.3.

3.1 The Q-Learning Algorithm

In previous work, we have proved (theoretically and empirically) that our marketplace enables an agent to relate the rewards it received to its G INQ segments [7]. Building on this basis, the contribution of this paper is in how to effectively learn the expected revenue that is likely to accrue over its G segments. Such a strategy is desirable because high expected revenue on a specific segment implies that more rewards can be expected if it repeats bidding on that segment in future. Therefore, this section aims to address the problem of producing the expected revenue profile over an agent's G segments, while still trading profitably in the marketplace.

In detail, an agent needs to execute a set of *actions* (bidding on its G segments, $a_1, a_2, \cdots, a_G$) to learn the expected revenue of each segment ($R(a_i)$, $i \in [1..G]$). Specifically, an action a_i that results in its recommendation being displayed to the user must pay some amount of credit. Then, it may or may not receive an amount of reward (depending on whether its recommendation satisfies the user). We record the tth immediate reward that a_i has received as $r_{i,t}$ ($t = 1, 2, \cdots$). From a statistical perspective, the expected revenue can be obtained from the mean value of the series of discrete immediate reward values:

$$E[R(a_i)] = \lim_{t\to\infty}\left(\frac{1}{t}\sum_{t} r_{i,t}\right). \quad (1)$$

In this context, the Q-learning technique provides a well-established way of estimating the optimality [11]. In particular, we use a standard Q-learning algorithm to estimate $R(a_i)$ by learning the mean value of the immediate rewards:

$$\hat{Q}_i := \left(1 - \frac{1}{t}\right)\cdot \hat{Q}_i + \frac{1}{t}\cdot r_{i,t}, \quad (2)$$

where $\hat{Q}_i$ is the current estimation of $R(a_i)$ and $\frac{1}{t}$ is the learning rate that controls how much weight is given to the immediate reward (as opposed to the old estimation). As $\frac{1}{t}$ decreases, $\hat{Q}_i$ builds up an average of all experiences, and the odd new unusual experience, $r_{i,t}$, does not significantly affect the established $\hat{Q}_i$. As t approaches infinity, the learning rate tends to zero which means that learning is no longer taking place. This, in turn, makes $\hat{Q}_i$ converge to a unique set of values that define the expected revenue of each segment.

Proposition 1. *As $t \longrightarrow \infty$, $\hat{Q}_i$ converges to $E[R(a_i)]$.*

Proof. We use $Q_{i,0}$ to represent the initial value of $\hat{Q}_i$, and $\hat{Q}_{i,t}$ to represent the local estimation to $R(a_i)$ when a_i has been experienced t times. $\hat{Q}_i$'s updates go:

$$\begin{aligned}
&\hat{Q}_{i,1} = 0\cdot\hat{Q}_{i,0} + 1\cdot r_{i,1} = r_{i,1}\\
&\hat{Q}_{i,2} = \tfrac{1}{2}\cdot r_{i,1} + \tfrac{1}{2}\cdot r_{i,2} = \tfrac{1}{2}(r_{i,1} + r_{i,2})\\
&\hat{Q}_{i,3} = \tfrac{2}{3}\cdot\tfrac{1}{2}(r_{i,1} + r_{i,2}) + \tfrac{1}{3}\cdot r_{i,3} = \tfrac{1}{3}(r_{i,1} + r_{i,2} + r_{i,3})\\
&\vdots\\
&\hat{Q}_{i,t} = \tfrac{1}{t}(r_{i,1} + r_{i,2} + \cdots + r_{i,t}) = \tfrac{1}{t}\textstyle\sum_{j=1}^{t} r_{i,j}
\end{aligned}$$

As $t \to \infty$, $\lim_{t\to\infty}(\frac{1}{t}\sum_{j=1}^{t} r_{i,j})$ statistically defines $E[R(a_i)]$. □

This proof exemplifies how newly experienced immediate rewards, combined with the learning rate, produce convergence. With the Q-learning algorithm in place, an agent needs an exploration strategy to execute actions to build up its $\hat{Q}$ profile.

3.2 The Exploration Strategy

We assume all agents are self-interested and want to gain maximal revenue as they bid. However, before $\hat{Q}_i$ converges, it is difficult for an agent to know how much can be expected through each action and, therefore, which action it should choose. It is faced with the classic dilemma of choosing actions that have a well-known reward or choosing new ones that have uncertain rewards (which may be higher or lower than the well-known actions). To this end, the agent needs an exploration strategy over its G segments to build up its $\hat{Q}_i$ in an effective way so that it can know how much return can be expected from each segment.

In general, there is a fairly well-developed formal theory for exploration strategies for problems similar to that faced by our agents [12]. However, the standard methods require very specific conditions (detailed in Section 5) that do not hold in our context.[2] Specifically, the number of times that an agent can interact with the marketplace is not limited.

2. In fact, it is hard to find the absolutely best strategy for most complex problems. In the reinforcement learning practice, therefore, approaches tend to be developed for specific contexts. They solve the problems in question in a reasonable and computationally tractable manner, although they are often not the absolutely optimal choice [12].

Thus, the agent can gather as much information as it wants in order to form its expected revenue profile. Knowing how much can be expected through each action, an agent can use a probabilistic approach to select actions based on the law of effect [13]: *Choices that have led to good outcomes in the past are more likely to be repeated in the future*. To this end, a *Boltzmann exploration* strategy fits our context well; it ensures the agent exploits higher $\hat{Q}$ value actions with higher probability, whereas it explores lower $\hat{Q}$ value actions with lower probability [12]. The probability of taking action a_i is formally defined as:

$$P_{a_i} = \frac{\mathbf{e}^{\hat{Q}_i/T}}{\sum_{j=1}^{G} \mathbf{e}^{\hat{Q}_j/T}} \quad (T > 0), \tag{3}$$

where T is a system variable that controls the priority of action selection. In practice, as the agent's experience increases and all $\hat{Q}_i$s tend to converge, the agent's knowledge approaches optimality. Thus, T can be decreased such that the agent chooses fewer actions with small $\hat{Q}_i$ values (meaning trying not to lose credits) and chooses more actions with large $\hat{Q}_i$ values (meaning trying to gain credits).

In general, however, we have observed that the learning algorithm of (2) accompanied with the exploration strategy of (3) has a problem of producing bias from the optimal and very little work has been done to address this. This problem occurs when an agent obtains a very small negative $\hat{Q}_i$ value for a particular action in its first few trials.[3] If this happens, a bias from the true expected revenue of this action may occur (since the action may, in general, produce positive $R(a_i)$) and the agent will seldom choose it. This kind of bias is a particular problem in our system. This is because a user may not always visit all displayed items in the sidebar and, thus, some good recommendations may be skipped and, therefore, be deemed bad ones. To avoid such bias, T needs to be assigned a very large value in the beginning of learning to limit the exploration priority given to those actions with very large $\hat{Q}$ values. However, controlling T in terms of producing the unbiased optimal strategy is hard to achieve since different actions' $\hat{Q}$s converge with different speeds and their convergence is difficult to detect. Even with other exploration strategies, such biases still exist since no exploration can avoid such unlucky trials at the beginning of learning. To this end, we developed an algorithm that takes positive initial $\hat{Q}_i$ values into account to overcome this problem. We detail this in the next section.

3.3 The Overall Strategy

To overcome the impact of bias in the beginning of learning, we use positive initial $\hat{Q}$ values (i.e., $\hat{Q}_{i,0}$) and make them affect the learning. Thus, instead of algorithm (2), we use the following learning algorithm:

$$\hat{Q}_i := \left(1 - \frac{1}{t_0 + t}\right) \cdot \hat{Q}_i + \frac{1}{t_0 + t} \cdot r_{i,t}. \tag{4}$$

The difference between (2) and (4) is that the former does not take $\hat{Q}_{i,0}$ into account, whereas the latter does. Specifically, algorithm (4) assumes that each action has been experienced t_0 (t_0 is positive and finite) times and each time with a feedback of $\hat{Q}_{i,0}$ ($\hat{Q}_{i,0} \gg 0$) before the agent starts learning. This, in turn, removes the problem discussed in Section 3.2. Indeed, if an action causes a negative immediate reward in the beginning, it does not force its $\hat{Q}_i$ to become negative. In this way, all actions will still be allocated a relatively equal opportunity of being explored as an agent begins learning. As the agent continues to interact with the marketplace, its $\hat{Q}_i$s update gradually to different levels and these levels still make its exploration follow the law of effect. Thus, the agent's exploitation tends to optimality with its $\hat{Q}$ values tending to converge. Additionally, by initializing $\hat{Q}$ with positive values, the exploration does not need a sophisticated control on T since a relatively small positive value is sufficient and is easier to control. Moreover, the change from (2) to (4) does not affect the convergence (as proved below).[4]

Proposition 2. *Given $\hat{Q}_i$'s definition by algorithm (4), its convergence to $E[R(a_i)]$ is independent of its initial value $\hat{Q}_{i,0}$ and initial time t_0.*

Proof. $\hat{Q}_i$'s updates go:

$$\hat{Q}_{i,1} = \tfrac{t_0}{t_0+1} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+1} \cdot r_{i,1}$$

$$\hat{Q}_{i,2} = \left(1 - \tfrac{1}{t_0+2}\right)\left(\tfrac{t_0}{t_0+1} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+1} \cdot r_{i,1}\right) + \tfrac{1}{t_0+2} \cdot r_{i,2}$$

$$= \tfrac{t_0}{t_0+2} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+2} \cdot (r_{i,1} + r_{i,2})$$

$$\hat{Q}_{i,3} = \left(1 - \tfrac{1}{t_0+3}\right)\left(\tfrac{t_0}{t_0+2} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+2} \cdot \left(r_{i,1} + r_{i,2}\right)\right) + \tfrac{1}{t_0+3} \cdot r_{i,3}$$

$$= \tfrac{t_0}{t_0+3} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+3} \cdot (r_{i,1} + r_{i,2} + r_{i,3})$$

$$\vdots$$

$$\hat{Q}_{i,t} = \tfrac{t_0}{t_0+t} \cdot \hat{Q}_{i,0} + \tfrac{t}{t_0+t} \cdot \tfrac{1}{t} \cdot \textstyle\sum_{j=1}^{t} r_{i,j}$$

Since t_0 is finite, $\lim_{t\to\infty} \frac{t_0}{t_0+t} \longrightarrow 0$ and $\lim_{t\to\infty} \frac{t}{t_0+t} \longrightarrow 1$. Thus, $\lim_{t\to\infty} \hat{Q}_{i,t} \longrightarrow \lim_{t\to\infty} (\frac{1}{t}\sum_{j=1}^{t} r_{i,j}) = E[R(a_i)]$. □

This proof shows that algorithm (4) also produces unbiased learning. Thus, we will use (4) and (3) for our agents and the overall strategy is detailed in Fig. 5.

4 Evaluation

This section reports on the experiments to evaluate the learning strategy we have developed. The experimental settings are discussed in Section 4.2, before the evaluations are presented in Section 4.3. First, however, we discuss the criteria with which we can evaluate our design.

4.1 Evaluation Metrics

To evaluate the learning strategy we use the following evaluation metrics (the first two are concerned with an individual learner's performance and the second two with the performance of the collective of learners):

- *Convergence to Optimality:* Many learning algorithms come with a provable guarantee of asymptotic

3. A negative immediate reward means punishment and an erroneous action. A reward of zero means that the action has received no feedback. Thus, actions with negative, zero, and positive feedback are differentiated and exploration priority should be given to the latter two.

4. However, the time it takes to converge is extended slightly depending on the values of $\hat{Q}_0$ and t_0 (the larger their values are, the longer it takes to converge).

THE MAIN STRATEGY:

```
for i = 1 to G do {
    // initialize Q̂_i and Q_init ≫ 0
    Q̂_{i,0} = Q_init;
    // initialize t_i
    t_i = 0;
}
do {
    // compute probability according to Equation (3)
    for i = 1 to G do
        P_{a_i} = ExploreProbability( i, Q̂_1, Q̂_2, ···, Q̂_G );
    // k ∈ [1..G]
    a_k = ActionSelection( P_{a_1}, P_{a_2}, ···, P_{a_G} ) ★;
    // increase the times that a_k has been experienced
    t_k = t_k + 1;
    // compute immediate reward
    r_{k,t_k} = ImmediateReward( a_k );
    // update Q-value according to Equation (4)
    Q̂_k = UpdateQ( Q̂_k, t_k, r_{k,t_k} );
} while (true)
```

★ METHOD ACTIONSELECTION:

```
ActionSelection( P_{a_1}, P_{a_2}, ···, P_{a_G}){
    // declare probability boundaries of the G segments
    double boundary[0..G];
    for i = 0 to G do
        boundary[i] = 0;
    // compute the G actions' probability boundary
    for i = 1 to G do
        for j = 1 to i do
            boundary[i] = boundary[i] + P_{a_j};
    // generate a probability
    double Rand = UniformRandom0to1() ♠;
    // select a random action based on its probability
    for k = 1 to G do
        if ( boundary[k − 1] ⩽ Rand < boundary[k] )
            return a_k;
}
```

♠ *UniformRandom0to1() returns a random value that follows a uniform distribution within the range [0, 1.0).*

Fig. 5. The Learning Strategy for an Individual Agent.

convergence to optimal behavior [11]. This criterion is included here to evaluate the quality of learning itself; it is important because if an algorithm does not converge, the agent will have no incentive to follow its behavior.

- *Individual Rationality:* All component recommenders in our system are self-interested agents that aim to maximize their revenue by bidding their recommendations [6]. Thus, if an agent can make a profit by participating in a particular auction it will do so. This criterion is included here because without such individually rational mechanisms, there is no motivation for the agents to participate in the system.
- *Quick Market Convergence:* If the prices of the displayed recommendations reach a steady state after a number of consecutive auctions, the market is convergent. In the analysis of our recommender system, we proved that convergence is necessary to ensure only the best items are displayed and that they are shortlisted in decreasing order of UPQ [7]. Therefore, a market that converges quickly means that it starts satisfying the user quickly. This is clearly important since a user will stop using a recommender if it takes too long to produce good suggestions. Thus, this criterion is used to ensure recommender agents clear incentives.
- *Best Recommendation's Identification:* A good recommender system should be able to identify the best recommendation (the one with the highest UPQ) quickly and suggest it frequently [14]. This is important because, otherwise, if the best recommendations cannot be identified and displayed, the user will stop using the system. Therefore, this criterion is used to ensure users high quality recommendations.

4.2 Experimental Settings

Having previously shown that our marketplace is capable of effectively incentivizing good recommendation methods to relate their INQs to the UPQ [7], we will not discuss how the agents do this. Rather, here, we simply assume that there are four good recommendation methods (able to correlate their INQs to the UPQ) and four poor ones (unable to do so). Given a specific recommendation (Rec), the correlations of its UPQ to a good method's INQ (INQ_g) and to a poor one's (INQ_p) are described in (5) and (6), respectively ("$\ncong$" means "has no relation to"):

$$UPQ(Rec) = INQ_g(Rec) \pm 0.1 \cdot random() \tag{5}$$

$$UPQ(Rec) \ncong INQ_p(Rec), \tag{6}$$

where $random()$ returns a random value that follows a uniform distribution within the range [0, 1.0). This random value can be seen as the noise between the INQ and the UPQ. All UPQ and INQ values are fixed within [0, 1.0). In each auction round, the marketplace calls for 10 bids. We use an independent-selection user model to decide which recommendations displayed to the user will be rewarded [15], [7]. In this model, selecting one item is independent of selecting another and all recommendations with a UPQ higher than a particular threshold will be rewarded. Here, we set this threshold to 0.75. To correlate their INQs to the UPQs, all agents divide their INQ range into $G = 20$ equal segments. We assume that all agents share the same set of recommendations and each agent has at least 10 items in each segment. Before starting to bid, Q_{init} is set to 250, $T = 200$ and $t_0 = 1$ for all agents. All agents are initially endowed with same amount of credit (65,536). At the beginning, each agent will bid the same (128) for items from any segment since it does not know which segments are more valuable than others.

4.3 Learning Strategy Effectiveness

Having outlined the configuration of the agents, this section details the evaluations. Among all the properties that we want the learning strategy to exhibit, $\hat{Q}$ convergence is the most important. Indeed, in its absence, an agent loses its

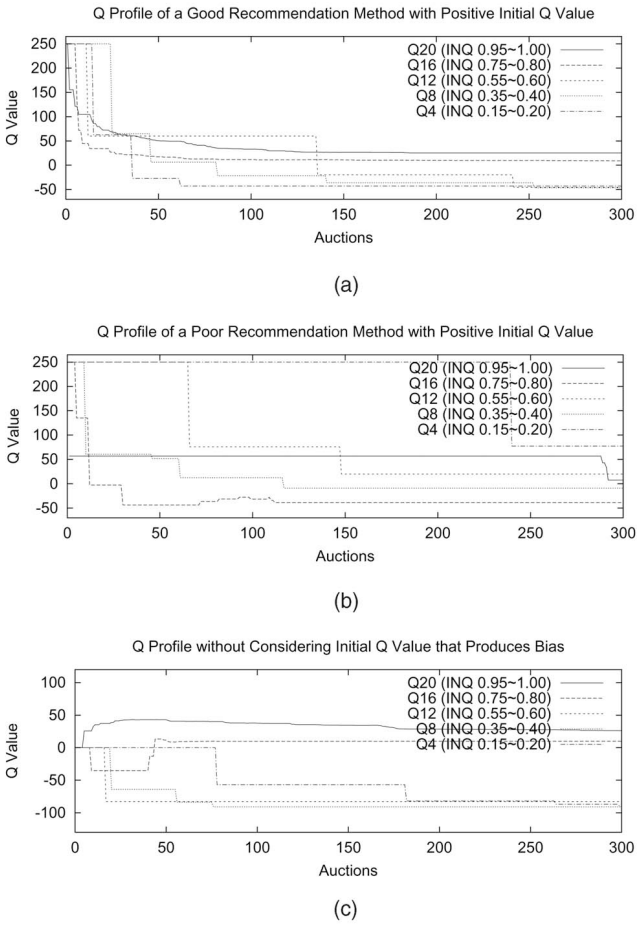


Fig. 6. Q-Learning Convergence.

basis to reason. Thus, we will start with experiments on the convergence of $\hat{Q}$ values.[5]

4.3.1 Convergence to Optimality

To evaluate an agent's $\hat{Q}$ value convergence, we arranged 300 consecutive auctions. Among the eight agents, the first four employ the good recommendation method and the last four employ the poor one. We find that, with a good method, an agent's $\hat{Q}$ values always converge such that high INQ segments' $\hat{Q}$s (corresponding to high UPQ because of (5)) converge to high values and low INQ segments' $\hat{Q}$s converge to low values (see Fig. 6a). Specifically, the $\hat{Q}$ values of those INQ segments corresponding to the UPQs above the user's satisfaction threshold (0.75) converge proportionally to their corresponding UPQs. The higher the corresponding UPQ, the higher the $\hat{Q}_i$'s convergence value, because the recommendations from a segment corresponding to higher UPQs receive more immediate reward than those corresponding to lower UPQs. The $\hat{Q}$ values of those segments that correspond to the UPQs below 0.75 converge to negative values since they do not receive rewards if their recommendations are displayed. Moreover, the convergence is independent of the specific form of (5). Specifically, once there is a unique UPQ level corresponding to each INQ level (even high INQ corresponding to low UPQ), the $\hat{Q}$ value of an INQ segment corresponding to a high UPQ will always converge to a high level (since it induces high immediate rewards). However, with a poor method, an agent's $\hat{Q}$ values cannot converge such that high INQ segments' $\hat{Q}$s converge to high values (see Fig. 6b). This is because a specific INQ corresponds to very different UPQs (and very different immediate rewards) at different times because of (6).

To exemplify that our learning algorithm (4) overcomes the bias problem that may occur in (2), we organized another set of experiments with all agents taking zero initial $\hat{Q}_i$ values and all other settings remained unchanged (see Fig. 6c). From Fig. 6c, we can see that $\hat{Q}_{12}$ is updated only once and with a large negative value of -82 (this gives the corresponding action virtually no chance of being selected in future). $\hat{Q}_{16}$ also produces a bias in the beginning. In even worse cases, $\hat{Q}_{16}$ can never update itself like $\hat{Q}_{12}$ (however, it should actually have a positive expected revenue). However, with positive initial $\hat{Q}_i$ values, such biases do not occur (see Fig. 6a).

4.3.2 Individual Rationality

The agents with good methods are able to know what recommendations better satisfy the user. Therefore, they can achieve more immediate rewards. Thus, good recommendations are raised more frequently by a learning agent than by a nonlearning one. This, in turn, means learning agents can maximize their revenue by selecting good recommendations. In particular, Fig. 7 shows that good recommendation methods with learning capability (the first four agents in Fig. 7a) make, on average, significantly greater amounts (about 43 percent) of credit than those without (the first four agents in Fig. 7b). With a poor method, the agents cannot relate their bids to the user's interest and therefore bid randomly. Thus, they cannot consistently achieve positive immediate rewards and their revenue is low (the last four agents in Figs. 7a and 7b).

4.3.3 Quick Market Convergence

We have shown that market convergence enables the agents to know what prices to bid for recommendations relating to certain UPQs so as to gain maximal revenue [6], [7]. Thus, quick market convergence lets agents reach this state quickly. To evaluate this, we organized two sets of experiments (using the same settings as the experiments assessing the convergence). The first one contains all learning agents and the other contains none. We find that a marketplace with learning agents always converges quicker than the one without. From Fig. 8, we can see that a marketplace with learning agents (Fig. 8a) converges after about 40 auctions, whereas one without (Fig. 8b) converges after about 120 auctions. Indeed, as the learning agents' $\hat{Q}$ profiles converge, more high-quality recommendations are consistently suggested (since their high $\hat{Q}$ values induce

5. The results shown are for a single simulation run. However, to ensure these results are typical for our system, we repeated the experiments for 200 simulation trials. Thus, the results we will show and discuss are representative of the outcomes. Specifically, over the 200 simulations, we found that: In 78.1 percent of the trials, the good recommendation methods' $\hat{Q}$s converge; all good recommendation methods with converged $\hat{Q}$ profiles make a significantly greater amount of credit (38.3 percent on average); a marketplace with learning agents takes 59.4 percent less time to converge than one without; and the number of best recommendations that a learning market is able to identify is, on average, 2.73 times that of a market without learning capability.

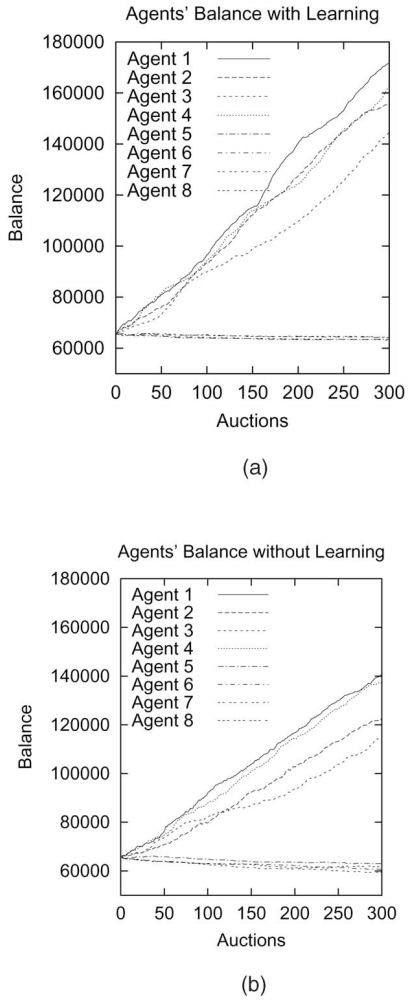


Fig. 7. Recommenders' balance.

high probability for the agent to bid these items because of (3)) and low-quality ones are deterred. This, in turn, accelerates effective price iterations to chase the market equilibrium. It takes approximately one third of the time for a market with learning agents to chase the equilibrium compared to one without.

4.3.4 Best Recommendation's Identification

To evaluate the learning strategy's ability to identify the best recommendation (from the viewpoint of the user, i.e., the top UPQ item) quickly and bid it consistently, we use the same set of experiments that were used to assess market convergence. We then trace the top UPQ item highlighted by a randomly selected learning agent with a good recommendation method and a corresponding one from a nonlearning agent in Figs. 8a and 8b, respectively. We do this by plotting this top UPQ items' bidding prices with circle points in the figures. To clearly display the points of the trace and not to damage the quality of lines (representing the three displayed bids), we do not display the points when this item is raised by other agents. From Fig. 8a, we can see that this item's bidding price keeps increasing till it converges to the first bid price of the displayed items. This means that as long as the randomly selected agent chooses

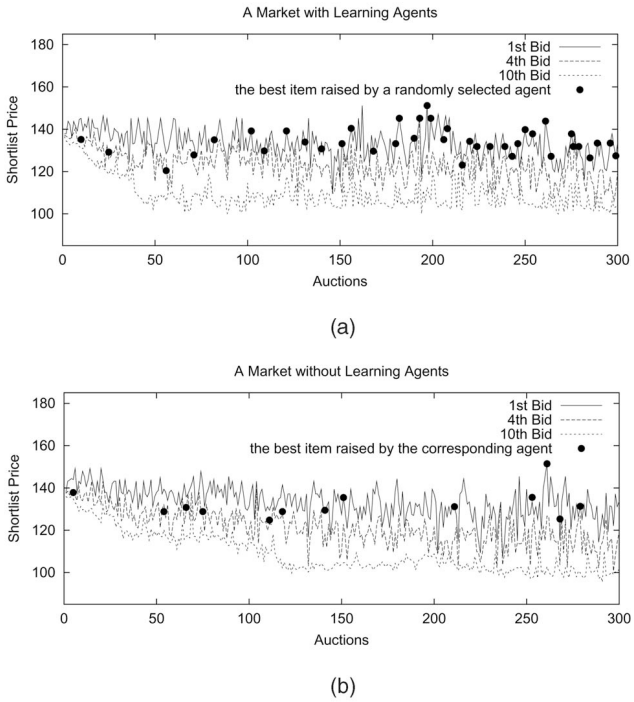


Fig. 8. Market convergence.

this particular item to bid in an auction (after the market converges), it is always displayed in the top position displayed to the user. However, in contrast, this phenomenon in a market without learning agents proceeds slowly (see Fig. 8b). This means that a learning market can satisfy the user quicker than a nonlearning one. Additionally, a learning market raises the best recommendation more frequently (39 times by the selected learning agent, see Fig. 8a) than a market without learning capability (13 times by the corresponding nonlearning agent, see Fig. 8b).

5 Related Work

The learning strategy presented in this paper significantly improves our previously reported market-based recommender system [6], [7] by speeding up the market's ability to make good recommendations. Previously, the strategy we developed for selecting which recommendations to bid was random (i.e., an agent randomly selects an item from any one of the G INQ segments in one auction round) [7]. While this strategy performed sufficiently to enable the viability of the market-based recommender to be evaluated, it often presented poor recommendations for too long and learned the user's interests too slowly. In contrast, by learning the expected revenue of each INQ segment and consistently bidding on those items that have high expected revenue (since they satisfy the user), an agent quickly identifies the best recommendation and maximizes its revenue (typically making 43 percent more credit than our previous method). With all agents employing the learning strategy, the market converges quickly (in about one third of the time of the previous method) and satisfies the user more consistently (making high-quality recommendations about three times as often as the previous method).

In terms of learning users' interests, most existing recommender systems use techniques that are based on two kinds of features of recommendations: objective features and subjective features. For example, LIBRA is a book recommender system that extracts textual information from books that a user has previously indicated a liking for and learns his interests through the extracted contents [16]. GroupLens is a Usenet news recommender that predicts the INQ of a specific recommendation based on other users' ratings on it [14]. However, many researchers have shown that learning techniques based on either objective or subjective features of recommendations cannot successfully make high-quality recommendations to users in all situations [2], [17], [5]. Thus, no one learning technique is universally best for all users in all situations. The fundamental reason for this is that these existing learning algorithms are built *inside* the recommenders and, thus, the recommendation features that they employ to predict the user's preferences are fixed and cannot be changed. Therefore, if a learning algorithm is computing its recommendations based on the features that are relevant to a user's context, the recommender is able to successfully predict the user's preferences (e.g., a customer wants to buy a "blue" cup online and the recommendation method's learning algorithm is just measuring the "color" but not the "size" or the "price" of cups). Otherwise, if the user's context related features do not overlap any of those that the learning algorithm is computing on, the recommender will fail (e.g., the user considers "color" and the learning algorithm measures "size").

To overcome this problem and successfully align the features that a learning technique measures with a user's context in all possible situations, we seek to integrate multiple recommendation methods (each with a different learning algorithm) into one single system and use an overarching marketplace to coordinate them. Essentially, our market-based system's learning technique encapsulates multiple learners and each one computes its recommendations based on some specific features. Thus, our approach has a larger probability of relating its features to the user's context and, so, correspondingly, has a larger opportunity to offer high-quality recommendations.

In terms of the integration of multiple recommendation methods, our approach is significantly different from the conventional hybrid filtering systems. Such systems integrate both content-based and collaborative filtering techniques and try to use the strength of one kind of technique to compensate for the weakness of the other [17], [18], [19]. For example, collaborative via content is a typical hybrid approach to address the sparse-data problem in pure collaborative filtering systems [20]. Such techniques use secondary data (e.g., document contents) to predict users' preferences in collaborative recommendations when collaborative user profiles lack data [21]. While hybrid systems can sometimes overcome the shortcomings of pure content-based and pure collaborative systems, this conventional integration of multiple filtering techniques is typically performed in a rigid and predetermined manner. Thus, there is no automated way of determining in what circumstances which kind of filtering technique is more relevant to a particular user in his current context. In contrast, by using the market to reward effective recommenders, our approach dynamically tunes the relative importance of the methods according to the feedback received from the users.

In terms of general work on market-based recommendations, the most related work to our own is that of [15]. This work uses a market to competitively allocate consumers' attention space in the domain of retailing online products (such as PC peripherals). However, this is less concerned with the information retrieval and filtering. Specifically, they use a market to competitively allocate consumers' attention space in the domain of retailing. Here, the scarce resource is the consumer's ability to focus on a set of banners or products. However, this work and our own use the market mechanisms in different ways to help recommendations. The market in [15] is used only to coordinate agents' bidding, whereas ours is used not only for this purpose, but also to correlate the INQ to the UPQ of recommendations (i.e., also for quality classification and alignment).

In terms of specific work in the area of reinforcement learning, the most relevant work to ours is that of the *k*-armed gambling problem [22]. In this, an agent faces k gambling machines, each of which has a probability of payoff of zero or one. The similarity with our problem is that, in both cases, the agent needs to learn the average payoffs that can be obtained from each gambling machine (or from each INQ segment in our case) as quickly as possible, while still maximizing its revenue. In this context, the solution to the *k*-armed gambling problem also suits our problem. Specifically, Berry and Fristedt developed a recursive algorithm to find the optimal strategy to gain the maximal payoffs in the case that the agent is permitted a fixed number of pulls [22]. However, in contrast, our recommender agents do not have a limit on the number of interactions they can have with the marketplace. Thus, our agents can gain sufficient experience to build up an unbiased optimal strategy. Gittins also tackles the *k*-armed gambling problem [23]. His "allocation index" method indexes all the actions that an agent experienced with a combined value of the expected payoff of each action and the value of the information that can be obtained by choosing it. The agent then chooses the action with the largest index value and this is shown to guarantee the optimal balance between exploration and exploitation. However, this technique only applies if the expected future rewards are discounted, which is inappropriate in our context because future immediate rewards are equally important as the current ones in our system (thus, we do not discount rewards). Thrun also develops an exploration strategy for the *k*-armed gambling problem [24]. Specifically, his strategy always chooses the action with the highest payoff. However, this strategy may produce a biased estimation from the true expectation since the actions with negative signals received in the beginning may have insufficient experience to produce biased expectations (as we discussed in Section 3.2). In contrast, by using the positive $\hat{Q}$ initial value and large T value to give equal opportunity to different segments to update their expected revenue, our strategy can produce an unbiased expected revenue profile. Kaelbling's interval-based technique can be seen as an extension of Thrun's

greedy strategy [25]. Her approach computes the upper bound of the confidence interval on the success probabilities of all actions and chooses the one with the highest upper bound. However, this approach relies on an a priori analysis of the payoff distribution of each action. But, in our context, actions from the same segment correspond to a relatively stable amount of reward. Thus, we do not have to perform this analysis (which would be difficult if not impossible) in our case because insufficient experience in the beginning of learning may produce biased confidence intervals and this, in turn, also induces a biased expected revenue profile.

Generally speaking, in reinforcement learning, an agent is assumed to be situated in a multistate environment where the agent's actions determine both its immediate reward and the next state of the environment. Moreover, this state transition affects the agent's future actions and, consequently, its future expected rewards. Thus, the expected future reward needs to be factored into the agent's current decision making. Therefore, in this kind of problem, an agent needs to learn which actions are desirable based on rewards that can be obtained arbitrarily far in the future. In this context, Markov Decision Processes (MDPs) are a typical model of this kind of reinforcement learning [11]. An MDP consists of a set of states (s), a set of actions (a), a state transition function ($\delta(s_t, a_t) = s_{t+1}$), and a reward function with respect to the transitions ($r = (s_t, a_t)$). The MDP model makes the complicated decision making processes intuitively and simplified by estimating the overall payoff (current reward plus the discounted delayed rewards in future) of all the possible state transitions (all combinations of the state-action pair (s_i, a_j)). After this estimation profile converges, an agent's optimal action selection strategy is choosing the action with the maximal overall payoff at any given state. However, in our context, taking one action is independent of taking another (because future rewards are only based on future recommendations' UPQs and have no relationship to the current recommendation). Therefore, our learning strategy is on a single-state basis.

6 Conclusions and Future Work

To be effective in a multiagent recommender system (such as our market-based system), an individual agent needs to adapt its behavior to reflect the user's interests. However, in general, an agent initially has no knowledge about these preferences and it needs to obtain such information. But, in so doing, it needs to ensure that it continues to maximize its revenue. In more detail, therefore, the contributions of this paper are as follows:

1. We have developed a quality classification mechanism and a reinforcement learning strategy that achieve the balance between quickly learning users' interests and maximizing recommender agents' revenue. The quality classification mechanism together with the reinforcement learning strategy enables a recommender agent to quickly learn the users' interests, while still maximizing its revenue.
2. Our market-based recommender system automates the correlation between individual recommender agents' internal evaluation of the properties (and features) of recommendations and the users' evaluations. Essentially, the marketplace integrates multiple different recommendation methods and, thus, evaluates a wide variety of properties of recommendation items. Through bidding and receiving rewards, those methods that evaluate properties that reflect the users' concerns are able to learn users' interests consistently and thrive in the marketplace. Thus, these methods' INQs are effectively correlated to the UPQs and recommendations from these methods are consistently shortlisted in the recommender sidebar.
3. From an individual agent's point of view, the learning strategy we have developed enables it to quickly maximize its revenue. With our learning strategy, a recommender agent builds up a profile of a user's interests with respect to its INQ classification. Armed with this profile, a learning agent can bid those items that satisfy the user's interests (and, so, make good profits) not bid those items that do not satisfy the users' interests (and, so, avoid losing credits).
4. A market in which all agents employ our learning strategy converges rapidly and identifies the best recommendations quickly and frequently. The more effective component recommenders that are integrated in our marketplace, the greater the likelihood that the system can correlate its measures of recommendations consistently to users and the more high-quality recommendations are suggested to the right users.

Having demonstrated the viability and the effectiveness of the learning approach to our market-based recommender system, we now need to carry out more extensive field trials with real users to ascertain that the theoretical properties of our strategy do actually hold in practice.

Acknowledgments

This research is funded in part by QinetiQ and the EPSRC Magnitude project (reference GR/N35816). This paper is a significantly revised and extended version of [10].

References

[1] P. Resnick and H.R. Varian, "Recommender Systems," *Comm. ACM,* vol. 40, no. 3, pp. 56-58, 1997.
[2] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating 'Word of Mouth,'" *Proc. Conf. Human Factors in Computing Systems, CHI'95,* pp. 210-217, 1995.
[3] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, "Recommending and Evaluating Choices in a Virtual Community of Use," *Proc. Conf. Human Factors in Computing Systems, CHI'95,* pp. 194-201, 1995.
[4] L. Terveen and W. Hill, "Human-Computer Collaboration in Recommended Systems," *Human-Computer Interaction in the New Millennium,* chapter 22, J. Carroll, ed. Addison Wesley, 2001.
[5] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.
[6] Y.Z. Wei, L. Moreau, and N.R. Jennings, "Recommender Systems: A Market-Based Design," *Proc. Int'l Conf. Autonomous Agents and Multi Agent Systems (AAMAS03),* pp. 600-607, 2003.
[7] Y.Z. Wei, L. Moreau, and N.R. Jennings, "Market-Based Recommendations: Design, Simulation, and Evaluation," *Agent-Oriented Information Systems, LNAI 3030,* P. Giorgini, B. Henderson-Sellers, and M. Winikoff, eds., pp. 63-78, Springer-Verlag, 2004.
[8] S. Zhang, Q. Yang, and C. Zhang, "Data Preparation for Data Mining," *Applied Artificial Intelligence,* vol. 17, pp. 375-381, 2003.
[9] S. Zhang, C. Zhang, and Q. Yang, "Information Enhancement for Data Mining," *IEEE Intelligent Systems,* pp. 12-13, Mar./Apr. 2004.

[10] Y.Z. Wei, L. Moreau, and N.R. Jennings, "Market-Based Recommender Systems: Learning Users' Interests by Quality Classification," *Proc. Sixth Int'l Bi-Conf. Workshop Agent-Oriented Information Systems (AOIS-2004)*, pp. 119-133, 2004.
[11] T. Mitchell, *Machine Learning*. McGraw Hill, 1997.
[12] L.P. Kaelbling, M.L. Littman, and A.W. Moore, "Reinforcement Learning: A Survey," *J. Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.
[13] E.L. Thorndike, "Animal Intelligence: An Experimental Study of the Associative Processes in Animals," *Psychological Rev., Monograph Supplements*, no. 8, New York: MacMillan, 1898.
[14] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl, "Grouplens: Applying Collaborative Filtering to Usenet News," *Comm. ACM*, vol. 40, no. 3, pp. 77-87, 1997.
[15] S.M. Bohte, E. Gerding, and H. La Poutre, "Market-Based Recommendation: Agents that Compete for Consumer Attention," *ACM Trans. Internet Technology*, vol. 4, no. 4, pp. 420-448, 2004.
[16] R.J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization," *Proc. Fifth ACM Conf. Digital Libraries*, pp. 195-204, 2000.
[17] M. Montaner, B. Lopez, and J.L. Dela, "A Taxonomy of Recommender Agents on the Internet," *Artificial Intelligence Rev.*, vol. 19, pp. 285-330, 2003.
[18] B.M. Sarwar, J.A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using Filtering Agents to Improve Prediction Quality in the Grouplens Research Collaborative Filtering System," *Proc. 1998 ACM Conf. Computer Supported Cooperative Work*, pp. 345-354, 1998.
[19] J.L. Herlocker, J.A. Konstan, and J. Riedl, "Explaining Collaborative Filtering Recommendations," *Proc. ACM Conf. Computer Supported Cooperative Work*, pp. 241-250, 2000.
[20] M. Pazzani, "A Framework for Collaborative, Content-Based and Demographic Filtering," *Artificial Intelligence Rev.*, vol. 13, nos. 5-6, pp. 393-408, 1999.
[21] A. Popescul, L.H. Ungar, D.M. Pennock, and S. Lawrence, "Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments," *Proc. 17th Conf. Uncertainty in Artificial Intelligence (UAI-2001)*, pp. 437-444, 2001.
[22] D.A. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments*. London: Chapman and Hall, 1985.
[23] J.C. Gittins, *Multi-Armed Bandit Allocation Indices*. Wiley, 1989.
[24] S.B. Thrun, "The Role of Exploration in Learning Control," *Handbook of Intelligent Control*, D.A. White and D.A. Sofge, eds., New York: Van Nostrand, 1992.
[25] L.P. Kaelbling, *Learning in Embedded Systems*. Cambridge, Mass.: MIT Press, 1993.

Yan Zheng Wei received a double major degree in computer science and economics from Peking University, Beijing, China, and the PhD degree in computer science from the University of Southampton, Southampton, United Kingdom. He is currently a research professional at the British Telecom Intelligent Systems Research Centre. His research interests include artificial intelligence, multiagent systems, auctions, market-based control, machine learning, recommender systems, information retrieval, and filtering.

Luc Moreau is a professor of computer science in the School of Electronics and Computer Science at the University of Southampton. His research is concerned with large-scale open distributed systems not subject to centralized control; examples of these include the Internet, the World Wide Web, the Grid, the pervasive computing environment, and their successors likely to be a combination of these. He has published more than 80 articles in the domain of Grid computing, distributed systems, agent-based systems, and distributed information management. His investigation covers the spectrum of software engineering: design, specification, proof of correctness, implementation, performance evaluation, and application. He is investigator of a number of UK and European e-Science and agent-based projects. Dr. Moreau spent time as a visiting researcher at the University of Chicago and Argonne National Laboratory, at the Institute for Human and Machine Interaction, Pensacola, Florida, at École Normale Supérieure, Lyon, France, at the University of Linz, Austria, at the University of Edinburgh, and at INRIA-Rocquencourt. He serves on the program committees of multiple international conferences and workshops on grid and distributed computing, multiagent systems, and peer-to-peer computing.

Nicholas R. Jennings received the BSc (Hons.) degree in computer science from Exeter University, Exeter, United Kingdom, and the PhD degree in artificial intelligence from the Queen Mary College, University of London, London, United Kingdom. He is a professor of computer science in the School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom, where he carries out basic and applied research in agent-based computing. He is the deputy head of school (research), head of the Intelligence, Agents, Multimedia Group, and is also the chief scientific officer for Lost Wax. He helped pioneer the use of agent-based techniques for real-world applications and has also made theoretical contributions in the areas of automated negotiation and auctions, cooperative problem solving, and agent-oriented software engineering. He has published some 200 articles on various facets of agent-based computing, and is in the top 100 most cited computer scientists (according to the citeseer digital library). Dr. Jennings is a fellow of the British Computer Society, the Institute of Electrical Engineers (IEE), and the European Artificial Intelligence Association. He is the recipient of several awards for his research, including the Computers and Thought Award (1999), an IEE Achievement Medal (2000), and the ACM Autonomous Agents Research Award (2003).