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

Yan Zheng Wei, Luc Moreau and Nicholas R. Jennings

Intelligence, Agents, Multimedia Group
School of Electronics and Computer Science
University of Southampton, UK.
{yzw01r,L.Moreau,nrj}@ecs.soton.ac.uk

Abstract. Recommender systems are widely used to cope with the problem of information overload and, consequently, 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 according to the users' ratings of their suggestions. Moreover, we have shown this incentivises the agents to bid in a manner that ensures only the best recommendations are presented. To do this effectively, however, each agent needs to classify its recommendations into different internal quality levels, learn the users' interests and adapt its bidding behaviour for the various internal quality 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 helps 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.

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 filtering techniques: *(i) content-based filtering* recommends items based on their objective features (such as the text content of a Web document), whereas *(ii) collaborative filtering* recommends items based on their subjective features (e.g., the fact that a user with similar tastes likes them). However, both kinds of techniques have their weaknesses. The former cannot easily recommend non-machine parsable items (such as audio and video items), whereas the latter fail when there are an insufficient number of peers to accurately predict a user's interests.

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

Given this, it has been argued that there is no universally best method for all users in all situations [2].

In previous work, we have shown that an information marketplace can function effectively as an overarching coordinator for a multi-agent recommender system [3, 4]. In our system, the various recommendation methods, represented as recommender agents, compete to advertise their recommendations to the user. Through this competition, only the 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 incentivise the agents to align their bids with the user's preferences. Thus, recommendations that the user considers good are encouraged by receiving a reward, whereas poor ones are deterred (by paying to advertise their recommendations but by receiving no reward). In short, the market acts as a feedback mechanism that helps agents to correlate their own *internal ratings* of recommendations (i.e. the relevance rating computed by whatever recommendation algorithm they use) to the desires of the user.

While our system works effectively most of the time, an open problem from the point of view of the individual recommender agents remains: *given a set of recommendations with different internal rating 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 would also result in it losing money. Conversely, the agent could always bid those 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 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.

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. 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 interests. In our context, each agent classifies its recommendations into different quality 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 internal quality 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 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 multi-agent 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

Different recommendation methods use different metrics and different algorithms to evaluate the items they may recommend. Thus, the internal rating of the quality of a recommendation can vary dramatically from one method to another (e.g. some may think it is very relevant for the user, others may think it moderately relevant, while yet others may believe it is irrelevant). Here, we term this internal evaluation the method's *internal quality* (INQ). 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. Therefore, we term the user's evaluation of a recommendation the *user's perceived quality* (UPQ).

With these concepts in place, we now briefly outline our market-based recommender in the order of the market processes (see the circled numbers in Fig. 1) as follows. First, when the market calls the agents for a number (S) of recommendations, each agent submits S items and bids a price for each of them. Second, the market ranks all recommendations in decreasing order of their prices and displays the S items with the highest bid prices to the user. Consequently, each agent with displayed items pays an amount of credits (equal to how much it bids) for each of the corresponding displayed items for the advertisement. Third, the user then visits a number of the displayed items and gives a rating (i.e. UPQ) to each visited item based on his satisfaction. Fourth, the market rewards the agents with positive UPQ recommendations an amount of credit that is proportional to their UPQ values (see [3] for the details and the proof that this mechanism is Pareto optimal with respect to the group of rewarded agents and 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 behaviour 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

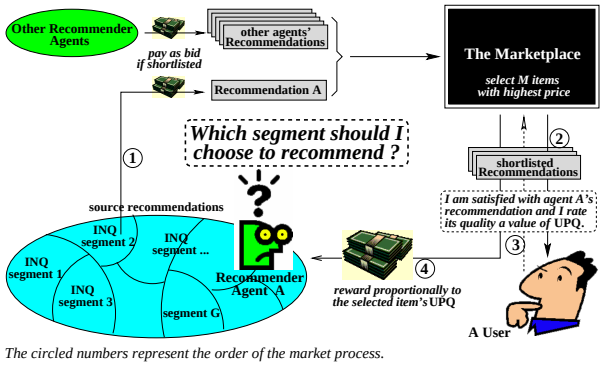

Fig. 1. An Agent's Learning Problem

$G = 2$, 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 [4]. 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 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. 1). 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 behaviour is on a "trial-and-error" basis. The agent bids its recommendations and receives the corresponding feedback in a manner that good

[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 be either 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.

recommendations gain rewards, whereas bad ones attract a loss. This kind of trial-and-error learning behaviour is exactly what happens in Reinforcement Learning [5]. 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 [4]. Building on this basis, the contribution of this paper is in how to 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 subsection aims to address the problem of producing the expected revenue profile over an agent's G segments.

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 t^{th} 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:

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

In this context, the Q-learning technique provides a well established way of estimating the optimality [5]. In particular, we use a standard Q-learning algorithm to estimate $R(a_i)$ by learning the mean value of the immediate rewards:

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

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 no learning is 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: *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:
$\hat{Q}_{i,1} = 0 \cdot \hat{Q}_{i,0} + 1 \cdot r_{i,1} = r_{i,1}$
$\hat{Q}_{i,2} = \frac{1}{2} \cdot r_{i,1} + \frac{1}{2} \cdot r_{i,2} = \frac{1}{2}(r_{i,1} + r_{i,2})$
$\hat{Q}_{i,3} = \frac{2}{3} \cdot \frac{1}{2}(r_{i,1} + r_{i,2}) + \frac{1}{3} \cdot r_{i,3} = \frac{1}{3}(r_{i,1} + r_{i,2} + r_{i,3})$
$\vdots$
$\hat{Q}_{i,t} = \frac{1}{t}(r_{i,1} + r_{i,2} + \cdots + r_{i,t}) = \frac{1}{t}\sum_{j=1}^{t} r_{i,j}$
As $t \rightarrow \infty$, $\lim_{t\rightarrow\infty}(\frac{1}{t}\sum_{j=1}^{t} r_{i,j})$ 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 [6]. 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. 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 [7]: *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 [6]. The probability of taking action a_i is formally defined as:

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

[2] In fact, it is hard to find the absolutely best strategy for most complex problems. In 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 [6].

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 equation (2) accompanied with the exploration strategy of equation (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. Because a user may not always visit all displayed items and, thus, some good recommendations may be skipped and, therefore, be deemed as 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:

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

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.

[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.

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.

PROPOSITION: *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:

$$\begin{aligned}
\hat{Q}_{i,1} &= \tfrac{t_0}{t_0+1} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+1} \cdot r_{i,1} \\
\hat{Q}_{i,2} &= (1 - \tfrac{1}{t_0+2})(\tfrac{t_0}{t_0+1} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+1} \cdot r_{i,1}) + \tfrac{1}{t_0+2} \cdot r_{i,2} \\
&= \tfrac{t_0}{t_0+2} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+2} \cdot (r_{i,1} + r_{i,2}) \\
\hat{Q}_{i,3} &= (1 - \tfrac{1}{t_0+3})(\tfrac{t_0}{t_0+2} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+2} \cdot (r_{i,1} + r_{i,2})) + \tfrac{1}{t_0+3} \cdot r_{i,3} \\
&= \tfrac{t_0}{t_0+3} \cdot \hat{Q}_{i,0} + \tfrac{1}{t_0+3} \cdot (r_{i,1} + r_{i,2} + r_{i,3}) \\
&\vdots \\
\hat{Q}_{i,t} &= \tfrac{t_0}{t_0+t} \cdot \hat{Q}_{i,0} + \tfrac{t}{t_0+t} \cdot \tfrac{1}{t} \cdot \textstyle\sum_{j=1}^{t} r_{i,j}
\end{aligned}$$

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. 2.

THE MAIN STRATEGY:

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

★ **METHOD ACTIONSELECTION:**

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

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

Fig. 2. The Learning Strategy

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 are concerned with the performance of the collective of learners):

Convergence to Optimality: Many learning algorithms come with a provable guarantee of asymptotic convergence to optimal behaviour [5]. 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 behaviour.

Individual Rationality: All component recommenders in our system are self-interested agents that aim to maximise their revenue by bidding their recommendations [3]. Thus, if an agent can make a profit by participate in a particular encounter it will do so. Thus, such individually rational mechanisms are important because without them, 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 [4]. 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.

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 [8]. This is important because, otherwise, if the best recommendation cannot be identified and displayed, the user will stop using the system.

4.2 Experimental Settings

Having previously shown that our marketplace is capable of effectively incentivising good recommendation methods to relate their INQs to the UPQ [4], 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 equations (5) and (6) respectively ("$\not\cong$" means "has no relation to"):

$$UPQ(Rec) = INQ_g(Rec) \pm 0.1 \cdot random() \quad (5)$$

$$UPQ(Rec) \not\cong INQ_p(Rec) \quad (6)$$

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 (or bias) 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 ten bids. We use an independent-selection user model to decide which recommendations displayed to the user will be rewarded [9, 4]. 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 ten items in each segment. Before starting to bid, Q_{init} is set to 250, $T = 20$ and $t_0 = 1$ for all agents. All agents are initially endowed with same amount of credit (65536). 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, convergence is the most important. Indeed, in its absence, an agent loses its basis to reason. Thus, we will start with experiments on the convergence of $\hat{Q}$ values.

• **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 equation (5)) converge to high values and low INQ segments' $\hat{Q}$s converge to low values (see Fig. 3(a)). 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 equation (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. 3(b)). This is because a specific INQ corresponds to very different UPQs (and very different immediate rewards) at different times because of equation (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 (all other settings remained unchanged (see Fig. 3(c))). From Fig. 3(c), we can see that $\hat{Q}_{12}$ is updated only once and with a very small value of -82 (this gives the

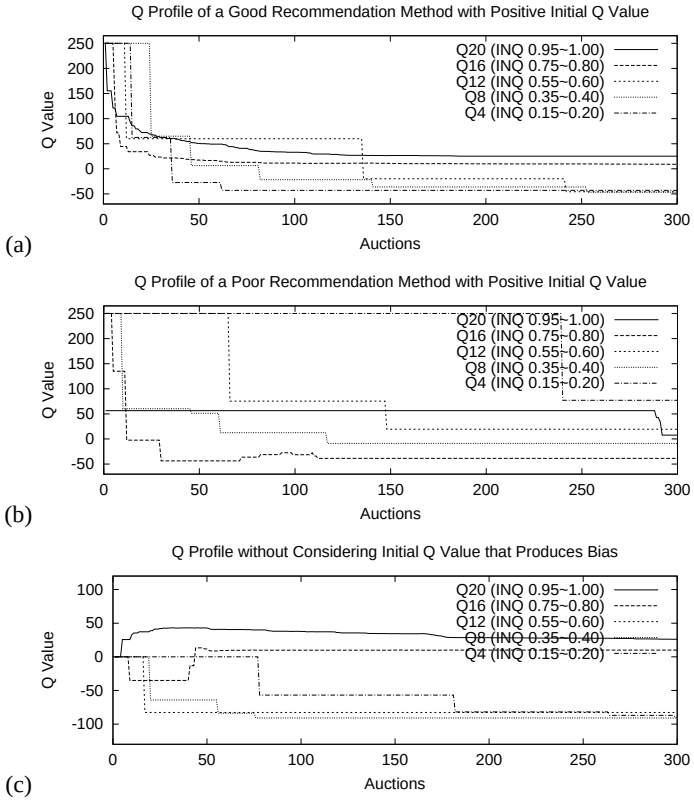

Fig. 3. Q-Learning Convergence

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. 3(a)).

• **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 non-learning one. This, in turn, means learning agents can maximize their revenue by selecting good recommendations. In particular, Fig. 4 shows that good recommendation methods with learning capability (the first four agents in Fig. 4(a)) make, on average, significantly greater amounts (about 43%) of credit than those without (the first four agents in Fig. 4(b)). 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 Fig. 4 (a) and (b)).

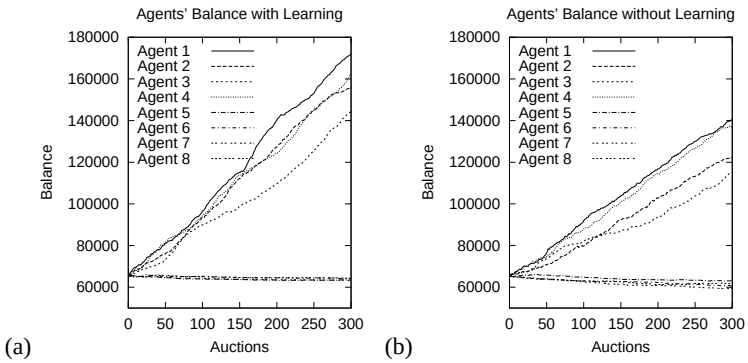

Fig. 4. Recommenders' Balance

• **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 [3, 4]. Thus, quick market convergence let 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. 5, we can see that a marketplace with learning agents (Fig. 5(a)) converges after about 40 auctions, whereas one without (Fig. 5(b)) 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 high probability for the agent to bid these items because of equation (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.

• **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 the 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 non-learning agent in Fig. 5 (a) and (b) 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. 5(a), 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 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. 5(b)). This means that a learning market can satisfy the user quicker than a

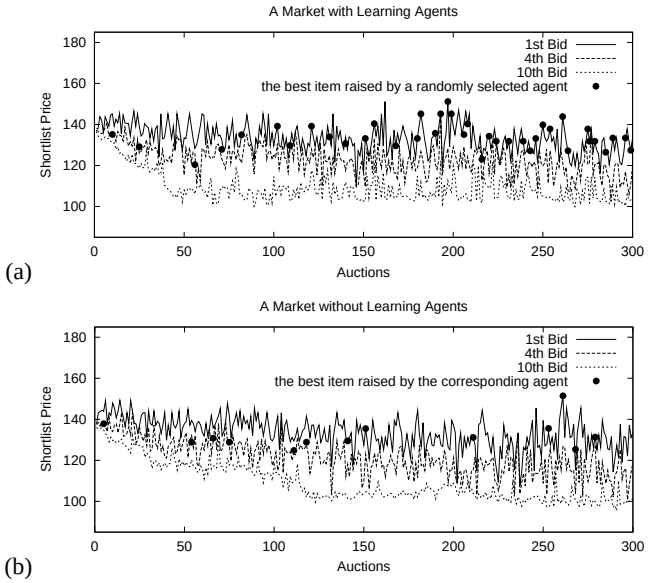

Fig. 5. Market Convergence

non-learning one. Additionally, a learning market raises the best recommendation more frequently (39 times by the selected learning agent, see Fig. 5(a)) than a market without learning capability (13 times by the corresponding non-learning agent, see Fig. 5(b)).

5 Related Work

The learning strategy presented in this paper significantly improves our previously reported market-based recommender system [3, 4] 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) [4]. While this strategy performed sufficiently to enable the viability of the market-based recommender to be evaluated, it sometimes 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 (making 43% more credits 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

(such as textual content in content-based recommenders) and subjective features (such as user ratings in collaborative recommenders). 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 [10]. GroupLens is a Usenet news recommender that predicts the INQ of a specific recommendation based on other users' ratings on it [8]. 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 [11, 12, 2]. 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 "colour" 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 "colour" 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 more learners and each learner 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 general work on market-based recommendations, the most related work to our own is that of [9]. This work uses a market to competitively allocate consumers' attention space in the domain of retailing online products (such as PC peripherals). 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 [9] 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. the quality classification and alignment).

6 Conclusions and Future Work

To be effective in a multi-agent recommender system (such as our market-based system), an individual agent needs to adapt its behaviour 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. To this end, we have developed a quality classification mechanism and a reinforcement learning strategy that achieve this balance. Essentially, our approach enables an agent to classify its recommendations into different categories

(based on its own quality measure) and then direct the right categories of items to the right users (by learning their interests by bidding and by receiving rewards). Specifically, through empirical evaluation, we have shown that our strategy works effectively at this task. In particular, a good recommendation method equipped with our learning strategy is capable of rapidly producing a profile of the user's interests and maximizing its revenue. Moreover, a market in which all agents employ our learning strategy converges rapidly and identifies the best recommendations quickly. Finally, we showed that our Q-learning strategy with positive initial $\hat{Q}$ values avoids bias. For the future, however, we need to carry out more extensive field trials with real users to determine whether the theoretical properties of the strategy do actually hold in practice.

References

1. Resnick, P., Varian, H.R.: Recommender Systems. Communications of the ACM **40** (1997) 56–58
2. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems **22** (2004) 5–53
3. Wei, Y.Z., Moreau, L., Jennings, N.R.: Recommender systems: A market-based design. In: Proceedings of International Conference on Autonomous Agents and Multi Agent Systems (AAMAS03), Melbourne (2003) 600–607
4. Wei, Y.Z., Moreau, L., Jennings, N.R.: Market-based recommendations: Design, simulation and evaluation. In: Proceedings of International Workshop on Agent-Oriented Information Systems (AOIS-2003), Melbourne (2003) 22–29
5. Mitchell, T.: Machine Learning. McGraw Hill (1997)
6. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285
7. Thorndike, E.L.: Animal intelligence: An experimental study of the associative processes in animals. Psychological Monographs **2** (1898)
8. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: Grouplens: Applying collaborative filtering to usenet news. Communications of the ACM **40** (1997) 77–87
9. Bohte, S., Gerding, E., Poutré, H.L.: Market-based recommendation: Agents that compete for consumer attention. ACM Transactions on Internet Technology (2004)
10. Mooney, R.J., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the 5th ACM Conference on Digital Libraries, TX, US (2000) 195–204
11. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Proceedings of Conference on human factors in computing systems. (1995) 210–217
12. Montaner, M., Lopez, B., Dela, J.L.: A taxonomy of recommender agents on the internet. Artificial Intelligence Review **19** (2003) 285–330