Optimal Radio Channel Recommendations with Explicit and Implicit Feedback

Omar Moling
Free University of Bozen-Bolzano
Piazza Domenicani 3
Bolzano, Italy
omoling@gmail.com

Linas Baltrunas
Telefonica Research
Plaza se E. Lluchi Martin 5
Barcelona, Spain
linas@tid.es

Francesco Ricci
Free University of Bozen-Bolzano
Piazza Domenicani 3
Bolzano, Italy
fricci@unibz.it

ABSTRACT

The very large majority of recommender systems are running as server-side applications, and they are controlled by the content provider, i.e., who provides the recommended items. This paper focuses on a different scenario: the user is supposed to be able to access content from multiple providers, in our application they offer radio channels, and it is up to a personal recommender installed on the clients' side to decide which channel to select and recommend to the user. We exploit the implicit feedback derived from the user's listening behavior, and we model channel recommendation as a sequential decision making problem. We have implemented a personal RS that integrates reinforcement learning techniques to decide what channel to play every time the user asks for a new music track or the current track finishes playing. In a live user study we show that the proposed system can sequentially select the next channel to play such that the users listen to the streamed tracks for a larger fraction, and for more time, compared to a baseline system not exploiting implicit feedback.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*

General Terms

Design, Experimentation, Human Factors

Keywords

Sequential music recommendations, implicit feedback, reinforcement learning

1. INTRODUCTION

Recommender Systems (RS) target the "information overload" problem, i.e., the difficulty to deal with the increasing amount of available information, e.g., in Internet, when facing an information search problem or a decision task. RSs suggest items that are estimated to be of interest and appropriate for the users in a particular usage context [15]. RSs have become important tools in many application domains, but especially for music information retrieval, since the today popular and large repositories of digital music (Last.fm, Pandora, iTunes) could not be fully exploited without such kind of support. In the last years, several recommendation techniques have been applied to the music domain, such as, collaborative filtering, content-based, social-based approaches and their combination as hybrid systems [6, 5].

1.1 Addressed Limitations of RSs

The large majority of RSs are running as server-side applications, and, more importantly, they are controlled by the content provider, i.e., who provides the recommended items. When a user accesses the system for retrieving new content, e.g., music tracks, the recommender computes a personalized selection of items and delivers them to the user. In [2], and more recently in [11], the authors advocate for new recommendation technologies that can work client-side, are controlled by the user, and can exploit items offered by several independent content providers. In their proposed scenario the recommender selects, among the items offered by the content providers, the most relevant ones for the user in a particular context [1]. Client-side recommenders can still be networked, and can benefit from interaction data collected by observing a community of users, but are designed to optimize performance metrics that are important for the users, and that could be different from those typically considered by the content providers (e.g., conversion rate).

Other two severe limitations of more traditional RSs are: the lack to support sequential consumption of items, and the difficulty to exploit implicit evaluation feedback on items. Regarding the first point, one can observe that users often listen to sequences of tracks during a listening session (playlists), and their preferences at a certain point in time are influenced by the tracks previously listened to. Therefore recommending a sequence of items defines a different problem than recommending individual items. Sequential items selection problems are not found in music applications only. For instance, after having read certain books, one decides what books to read next. These decisions are influenced by what has been read before, the background knowledge of a person, and what is available and preferred at the time the decision is made.

Regarding the second point, i.e., the difficulty to exploit implicit feedback on consumed items, consider the situation where a user is recommended a track, but after listening to it for a few seconds, she requests a new recommendation. That action should be interpreted as a sign that, at that point in time, i.e., after having listened to some particular tracks before, and given the user's preferences, that suggestion was not optimal. In the book recommender scenario we may observe a similar situation; a reader may stop reading a book for a while, or may even start reading a new one. These signs could be interpreted by the recommender as indications that the recommended book was not a convenient choice at that point in time.

Some recent research works have tried to address these two issues: recommending a sequence of items and exploiting implicit feedback in building the user profile and generating recommendations. We will review them in Section 6.

1.2 Recommendation Scenario and Results

This paper focuses on a specific scenario: the user is supposed to be able to access content from multiple providers, in our application they offer music channels, but it is up to a personal recommender to decide what channel to select and play. We show how we have addressed the specific application scenario and the above-mentioned issues, namely, sequential item recommendation, and the usage of implicit feedback in a music RS named RLradio. RLradio plays music streamed by different channels, each one featuring music of a particular genre. Moreover, it learns whether to stay tuned on the current channel or to switch to a different one. This decision is taken, for a target user, when the currently played track finishes, or when the user, using a *next* button of the system GUI, explicitly requests a new track from one of the available channels. We stress that the system has no control on the precise music tracks that are streamed by the available channels, and can only automatize the channel selection action (tune in) that is normally executed manually by a user. The goals are: a) to avoid the need to manually change channel in situations where this could result in a usability problem, e.g., in a car driving scenario; and b) to adapt in real time to the changed preferences of the user, or the group of users listening to the radio system.

In order to implement the above mentioned functionality and requirements, RLradio leverages both explicitly revealed music channel preferences, and implicit feedback produced by the users: the fraction of the track played in the recommended channel that is actually listened to by a user. It exploits a particular type of reinforcement learning (RL) techniques, i.e., R-Learning that is suited for continuous tasks learning [17]. The goal is to learn the optimal policy for selecting the music channel to suggest next.

We first acquired a model of the listening behavior of the users by observing their interaction with RLradio while the system was using a baseline channel selection policy that exploits only the explicitly revealed channel preferences of the users. We call this system variant RLradio P. Then, we computed the optimal policy for the acquired state transition model off-line, and forced RLradio to use it. This new system variant is called RLradio RL. Finally, during the successive interactions of the users with RLradio RL, the system adapted its behavior on-line using R-Learning.

We evaluated RLradio in a live user study. We conducted a within-subjects experiment where the users tested two RLradio variants with identical GUI. The first variant (RLradio P) uses only the user's channel preferences elicited at the beginning of the listening session. The second one (RLradio RL) exploits also the implicit feedback derived from the listening behaviors of all the users. The evaluation showed that the learning process is effective: users prefer RLradio RL, and they listen to both a larger fraction of the suggested tracks and a higher daily time span while using RLradio RL. Hence, in conclusion RLradio offers an effective solution for the considered sequential item recommendations problem. It minimizes user effort, and is capable to track the evolving preferences of the user (or users) receiving the recommendations.

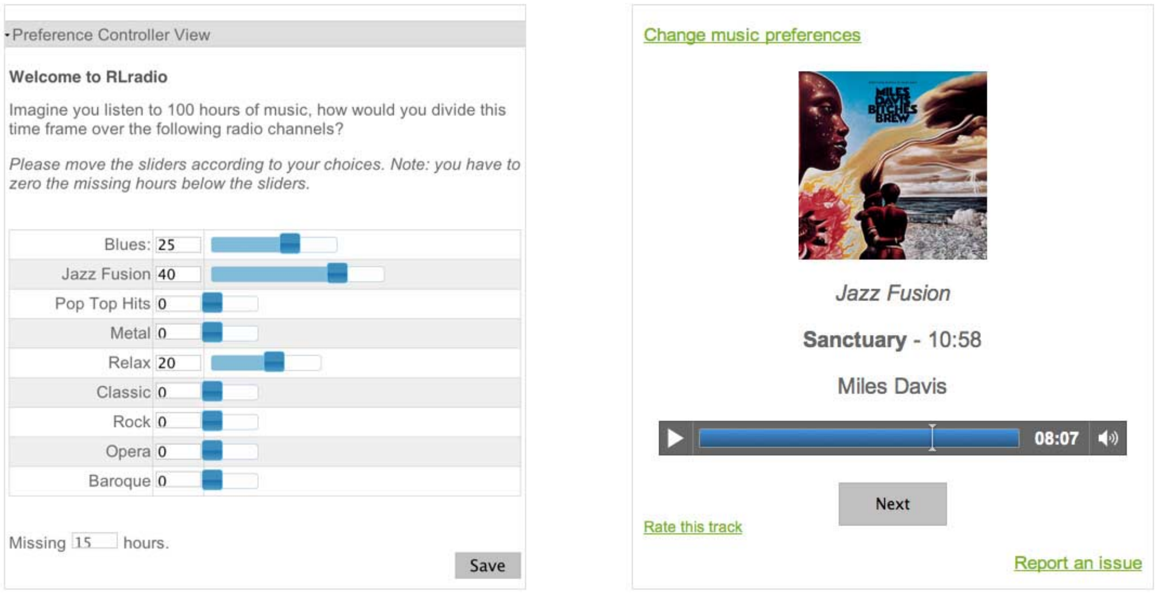
The rest of the paper is structured as follows: sections 2 and 3 describe the recommender system functionality and techniques. The evaluation approach and the results are described in sections 4 and 5. Then in section 6 we survey some related work, and finally, conclusions are drawn and future work defined in section 7.

2. RLRADIO RECOMMENDER SYSTEM

RLradio is a music player that offers music channels, each one featuring music of a rather homogeneous genre, such as in a thematic channel. The user can listen to the played music track either fully (when he probably likes it), or can request a new track (presumably when he does not like what is currently played). At that point RLradio decides whether to stay tuned on that channel, or to switch to another one offering music of a different genre. In a normal radio, changing the channel is up to the user, who would search for one, among a set of available channels, which is broadcasting music that he likes. This takes time and effort, and may not be convenient in some situations (e.g., when the user is driving or biking). RLradio automatizes and simplifies this process by providing a *single* button to request a new track (the *next* button in Figure 1(b)). At that point, it is up to the system to decide whether to stay tuned on the current channel, or to switch to another one, with the ultimate goal of providing music that the user, or the users, will like at that moment.

In our system prototype we have considered nine popular Internet radios. These are presented to a user indicating only the music genre played in the channel (Figure 1(a)). At the beginning of each listening session, RLradio collects the user's preferences for the available channels. These preferences are modeled as the percentage of music of that channel that the user would like to listen to (Figure 1(a)). In fact, these explicit preferences have been used by a baseline system, here called RLradio P, that at each channel recommendation step, i.e., each time the currently played track finishes, or when the user clicks on the next button: a) selects a channel with a probability proportional to the user's preference for that channel, and b) plays the next track in the queue that is offered by that channel. The sequence of the channels is therefore (pseudo) random, it exploits the user's explicitly entered preferences, but RLradio P does not take into account neither what channels were previously played nor any knowledge about when the user has used the next button.

We consider this selection policy as a *baseline*. Note that this is not a bad policy, and in fact, in the experimental evaluation, users exposed to this policy reported rather high satisfaction scores (see Section 5). Nevertheless, we hypothesize that *the performance of RLradio, measured as the percent-*



Welcome to RLradio

Imagine you listen to 100 hours of music, how would you divide this time frame over the following radio channels?

Please move the sliders according to your choices. Note: you have to zero the missing hours below the sliders.

Blues:	25
Jazz Fusion	40
Pop Top Hits	0
Metal	0
Relax	20
Classic	0
Rock	0
Opera	0
Baroque	0

Missing 15 hours.

Jazz Fusion

Sanctuary - 10:58

Miles Davis

(a) Music channels preferences collection (b) RLradio main view

Figure 1: RLradio user interface

age of each proposed track that is actually listened to, and the amount of time, per day, spent by the user listening to the recommended music, can be improved by using also implicit feedback, i.e., the knowledge of when the user clicks next. In other words, we conjectured that the system performance can be improved by observing what channel is played when the user stops listening to a music track in that channel, and requests a new track. We hypothesized that these observations can be exploited to learn a better channel selection policy (an optimal policy).

This policy, for instance, can override explicit preferences made by the user, and adapt to the real listening behavior. This is useful when the user is not entirely sure about his preferences, makes misjudgment, or his preferences change during the session, e.g., in a in-car player when new passengers enters into the car and they contribute to the decisions of requesting new tracks. We also conjectured that the policy can also model the optimal *sequence* of music genres preferred by the user, i.e., how music from different channels should be combined. For instance, the user may not like to switch continuously from one channel that he likes to another that he also likes, but could prefer to listen to more music tracks from the same channel before switching to another.

In our experiments, the nine Internet radio channels that have been selected offer music of a particular genre: blues, jazz fusion, pop, metal, ambient, classic, rock, opera and baroque. For each of the nine selected channels, 24 consecutive hours of music were ripped from the selected Internet radios and the music tracks were split automatically using the track metadata provided by the channels. This resulted in a total of 3,637 music tracks. The number of tracks per channel varies, as the average length of tracks belonging to different music genres is different. We decided to rip the music and then take the music from a local repository to make the experiments replicable and reduce the possible connection problems to external music servers.

3. THE RECOMMENDATION AGENT

In order to implement the channel recommendation agent used in RLradio, we modeled the recommendation task as a Markov Decision Problem (MDP) [17]. In general, a MDP is defined by:

- A set of states S, modeling the possible different situations that the decision maker can face.
- A set of actions A, listing the possible alternative actions, or decisions, that the decision maker can take.
- A real-valued function $T(s, a, s')$ that defines for each state s and action a combinations, the probability to make a transition to the new state s'. This is called the "environment model" because it describes the reaction of the environment to the decisions taken.
- A real-valued reward function $R(s, a, s')$ that measures the payoff obtained by the decision maker when making a particular state transition after the application of an action.

The solution to a MDP is an optimal policy, i.e., a rule for selecting at each state the best action, i.e., the actions that, transition after transition, will obtain the largest accumulated reward. We used reinforcement learning (RL) to implement the recommendation agent, i.e., the channel selection decision policy at each step. The actions taken by the recommendation agent in RLradio correspond to the channels to play next, and the environment is the "place" where the actions are executed, which in our case is the user listening to the tracks in the suggested channels. The user decides whether to listen, and to what extent, to the played tracks. The agent must learn from the user's (implicit) feedback what action to execute in each possible state in order to maximize the accumulated reward, i.e., as we will show, the percentage of the played tracks that is actually listened to. In the following we will instantiate the model to our particular application.

State Model S**.** The state models the current recommendation situation, i.e., it represents the current channel preferences of the user, and the previous listening behavior. In principle, the full knowledge of the user's listening behavior, i.e., the sequence of all the previously listened channels, and how much the user listened to them should be considered. The main drawback of this solution is the exponential growth of the size of the state space, and the difficulty to estimate the state transition probabilities, and compute the optimal policy. Moreover, one can hypothesize that the most recent listening history plays a stronger role in determining the preferences of a user at a certain point in time.

For these reasons, RLradio records only the two previously recommended channels and the percentages of the music tracks in these two channels that were actually listened to before the user clicked on the next button (if he clicked). These percentages were discretized in three levels that are described later when the Reward function is presented.

Moreover, in order to keep the state model small, the music channel preferences entered by the user were also discretized in four levels: low preference for a channel ($< 15\%$), fair (between 15% and 40%), high (between 40% and 60%) and very high (above 60%). These thresholds were defined heuristically, and we thought that four levels were adequate to score the user preference for a music channel, like in a five star Likert rating schema.

With such a discretization there are 4^9 possible configurations of the user explicit preferences. In fact, we discovered that only 77 configurations were actually entered by the users while using the baseline system. Hence, we constrained the system to use only these configurations or profiles, and when a user entered different preferences we associated this user to the closest profile among these 77 configurations. With these assumptions we produced a state space of $56,133$ states, i.e., $9 * 3 * 9 * 3 * 77$. Where, 9 is the number of possible channels, and 3 is the number of possible (discretized) listening percentages, for the two tracks previously listened by the user. A summary of the state variables is shown in Figure 1.

Actions A**.** The actions correspond to the nine possible channel recommendations that can be selected by the system to pick up the new track to be played. We observe that the recommendation agent cannot recommend a particular track, but only the channel of the next track. The tracks in a channel have a predefined order and are played one after the other. This decision is motivated by the application scenario, i.e., we assume that the recommender can only decide the channel to tune in, and the compilation is defined for all the listeners of that channel by the channel's manager. All the nine actions are available in each state of the state space. Hence, by considering the size of the state space and the available actions, there were $56,133 * 9 = 505,197$ state-action combinations.

Reward R**.** The reward is computed observing the user's implicit feedback, i.e., the fraction of the played track's length in the suggested channel that was actually listened

Table 1: State variables

Variable	Values
The last proposed channel	9 channels
Percentage of the track in the last proposed channel that was actually listened to	3 discretized levels
The second-last proposed channel	9 channels
Percentage of the track in the second-last proposed channel that was actually listened to	3 discretized levels
Explicitly entered channel preferences	77 music profiles

to by the user, before he clicked *next*, if he did. This percentage is mapped to a three-levels discrete reward score that measures how successful the action (channel selected) was in that particular state (user preferences and previous listening behavior). 0 reward is assigned to the state transition if the music track is listened to less than 15%, i.e., the user requests another track before the playback reaches 15% of the length of the track. We considered this a strong sign of not liking the currently played track. If the user requests another track while the playback is between 15% and 60%, the reward equals 1, while for more than 60% up to the full length, i.e., the user does not request another track, we decided to assign a double reward of 2. These decisions are somewhat arbitrary, but necessary for defining the MDP problem. Slightly different choices are possible, and they can result in a different optimal policy.

Moreover, we note that here a feedback on a track is interpreted as an implicit evaluation of the channel. In fact, the user is listening to a particular track, and the skipping feedback is related to that specific music and only indirectly to the channel of that music. The rationale of this choice is again related to the application scenario: even though we could learn from the implicit feedback if a user likes or dislikes a track (as in [13]) this is not very useful since the recommendation agent has not full control on the next track that is going to be played, it can only select the channel.

Transition Probabilities. We collected raw data on the users' listening behavior by letting a group of 29 users to use the previously mentioned baseline system (RLradio P) for approximately two weeks. This is a common practice in RL. The baseline system adopts a policy that considers only the explicit music channel preferences entered by the user at the beginning of each listening session, and provides music from a channel with a probability equal to the value of the user preference for the channel. For example, setting 20% preference to channel *classic* results in having, at each step, a probability of 20% that classical music is selected. In addition, 10% of the proposals were chosen randomly in order to better explore the state space, a common practice in RL [17].

From the collected data, several values were computed, in particular we estimated the transition probabilities: i.e., the probability that a user will listen to a certain (discretized) percentage of a track (current state), if it belongs to the suggested channel A (action), given a particular channel preferences profile and the percentage of the two previously played tracks (in channels B and C) that were listened to.

We also estimated the probability that a user will listen to a certain percentage of a track in a channel given his channels' preferences alone. These prior values were used to provide a default state transition model for states that were poorly explored by the users. We combined these priors with the estimated state transition probabilities with Laplace smoothing [17]. The smoothing was such as to assign more weight to the prior values when a given state was visited poorly or not at all, while more weight was set on the experienced transition probabilities for good or fairly good explored states.

Optimal Policy. We computed the optimal policy using Policy Iteration with $\gamma = 0.8$ [17]. γ is a parameter that measures how the future reward is discounted compared with the immediate reward. We tried some other values, but we did not find significant differences; 0.8 is a rather standard value for this parameter. All other variables were set analyzing the data collected during the usage of the baseline system (RLradio P).

The obtained optimal policy served then as the starting policy for the learning-capable agent system variant (RLradio RL), which implemented the *R-Learning for Undiscounted Tasks* [17] technique, since music recommendation is considered a continuous task, as there is no identifiable terminal state. The algorithm is illustrated in Figure 2.

```
Initialize $p$ and $Q(s,a)$, for all $s, a$, arbitrarily
**repeat**
    $s \leftarrow$ current state
    Choose action $a$ in $s$ using $\epsilon$-greedy behavior policy
    Take action $a$, observe $r, s'$
    $Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r - p + \max_{a'} Q(s',a') - Q(s,a)\Big]$
    **if** $Q(s,a) = \max_a Q(s,a)$ **then**
        $p \leftarrow p + \beta\Big[r - p + \max_{a'} Q(s,a) - \max_a Q(s,a)\Big]$
    **end if**
**until** forever
```

Figure 2: R-Learning

As explained in more details in [17], R-Learning maintains two policies: a behavior policy, which serves to generate experience, and an estimation policy, which allows a generalized policy iteration process. The behavior policy is the ϵ-greedy action selection with $\epsilon = 0.1$ [17]. The estimation policy only maximizes the action according to action-value function $Q(s,a)$. In addition to the two policies, also the action-value function $Q(s,a)$ and the average reward p are maintained. In Figure 2, r denotes the immediate reward of an action, while the average reward p is set initially for each user to the overall average reward, and then is updated individually for each user. The values of the step-size parameters α and β were set to $\alpha = 0.3$, $\beta = 0.4$, which are common values for these parameters, such that p reacts faster than the state-action value $Q(s,a)$.

4. EXPERIMENTAL STUDY

As we mentioned above, we hypothesized that the proposed approach, which is implemented in RLradio RL, can improve the performance of the baseline system (RLradio P). We measure the system performance as the average percentage of the proposed music tracks that is actually listened to by each user, and as the average number of minutes of music, per day, that the users listen to. We focus on these measures because we aim at building a system that automatically changes the channel such that the played tracks are actually listened to by the users as much as possible, and the system is overall largely used.

We note that RLradio P faithfully uses the channel preferences of the user, while RLradio RL discretizes them, but in addition exploits the dynamically generated implicit feedback provided by the users while interacting with the system. Hence, the expected improvements must be due to the exploitation of the implicit feedback, i.e., the knowledge of when the user requests a new music track while listening to a track in the suggested channel, and having listened previously to other tracks from possibly different channels.

We designed a within group evaluation study. We split the users in two groups: (Group #1) first tested the baseline system (System P), and then the system equipped with the optimal policy and also using R-Learning (System RL); while (Group #2) executed the test in the opposite order. This decision is aimed at balancing a possible order effect. For each group, at least one week of pause was inserted between the evaluations of the two systems. The two groups were formed by 70 users in total and all users were initially informed that the evaluation consisted of two phases, but they had absolutely no knowledge about the systems' differences. The graphical user interfaces of the two systems and the music tracks played in the channels were identical. Hence the two systems differed only in the way the recommender selected the channel to use for playing the next music track.

At the beginning of each phase, all involved users were contacted via email. The initial steps of entering a nickname and defining the channel preferences were explained on the home page. Then, the user interface of the system was deemed as simple, and thus no special explanations were provided to evaluate it. Each evaluation phase lasted for 12 days on average. All data generated by the system usage, i.e. listening sessions, explicit preferences and implicit feedback, was stored on the server running the RS.

In addition to the collection of implicit feedback throughout all the test phases, all users were asked to take a survey at the end of each evaluation phase, i.e., after having used the Systems P and RL (according to their evaluation order). The aim of the survey was to measure the level of satisfaction of the user with the proposed system, and the effectiveness of the music recommendations. The survey included statements, for which the users were asked to express the level of agreement. The statements had possible answers on a five-level Likert scale: *Strongly Disagree*, *Disagree*, *Undecided*, *Agree* and *Strongly Agree*.

The most relevant statements were the following:

S1: I liked the music played by the system.
S2: The system switched the played channel at the right time.
S3: I would use such a player if available.
S4: The system is easy to use.

Statement S1 is related to the degree of satisfaction for the channels and tracks proposed and played by the player. S2 and S3 refer to the behavior of the system, and whether the

Table 2: Number of implicit feedback events collected

System	Feedback items
RLradio P	3423
RLradio RL	4385
Total	7807

user would like a system that behaves like the proposed one. Finally, S4 refers to the evaluation of the user-interface.

5. EXPERIMENT RESULTS

We compared RL and P system with paired and unpaired data. We considered paired data comparing the average percentage of played tracks listened by each user. In this case we could consider only the users that tested both systems: 17 for Group #1 and 13 for Group #2. We also considered not paired data while comparing the daily system usage, in minutes, by a user. In this case we could consider all the daily recorded sessions and we compared the length of the sessions observed for the two systems. Moreover, in order to increase the reliability of the observations we discarded the data generated by the sessions where the user did not interact with the system during a sequence of five tracks. That could be a sign that the user was not paying attention to the played music.

We note that RLradio RL is not a technology that is supposed to provide a large benefit just after the user has entered the channel preferences. This technology is supposed to adapt to the user while the user is interacting with the system. So in order to detect some differences the users must interact with the application for some time. We also observe that the system P was stable during the evaluation, i.e., its channel selection policy was not changing as new implicit feedback was accumulated. Conversely, the RL system was continuously updating its behavior as new implicit feedback was acquired. Hence, when a user tried the RL system, the implicit feedback provided by all the users that tried the system before was used by the R-Learning algorithm to improve the system's behavior. This is the major benefit of using reinforcement learning: the system a) adapts to the users that are actually using the system, and b) benefits from the feedback produced by previous usage, even by different users.

Table 2 illustrates the overall number of implicit feedback events collected during the evaluation phases, divided per system. We note that 1723 feedback events were collected during the first evaluation phase of RLradio P, and served as training data for the computation of the optimal policy for RLradio RL. A feedback event records: the channel of the track currently listened to, the channel of the two tracks that were listened before, the channels' preferences of the user (his profile configuration), and the time, during the track playback, when the user clicked on the *next* button (if he did it).

Figure 3 represents the frequencies of the observed listening sessions where the user gave a not null preference to $n = 1, \ldots, 9$ channels (e.g., in more than 80 sessions the user gave a positive preference to 7 different channels). This illustrates that typically the users entered a not null preference for many channels, hence switching channel makes sense for satisfying these users.

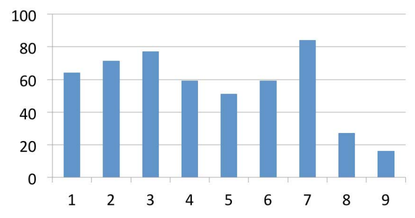


Figure 3: Frequency of sessions with a given number of channels with not null preference

We then analyzed the implicit feedback data. Regarding the within-group evaluation, i.e., considering the listening sessions generated by the users that tested both systems. On average each user listened to 64.35% of the length of the proposed music when using RLradio P, with a standard deviation (SD) of 0.209. Instead, when using RLradio RL, the average percentage of track length listened to by each user was 67.41%, with a SD of 0.182. Hence, the improvement in the percentage of the listened track length of RLradio RL over RLradio P is 4.76%. This difference is significant (paired-samples t-test $p = 0.028$).

We also measured that 63.33% *of the users, while using RLradio RL, listened, on average, to a larger percentage of the track length, compared to RLradio P.* In practice the large majority of the users intervened less frequently when using the RL system compared to when they used the baseline P system and listened longer to the proposed tracks.

Considering the daily usage of the system, we observed that the listening sessions with system RL were significantly longer than the sessions performed with P. The average daily usage when using RLradio P was 62.68 minutes per day. That increased to 75.58 minutes per day when using RLradio RL, i.e., the users listened to music while using RL 20% more than while using P. Again this difference is significant to a t-test (unpaired samples p=0.043).

The optimal policy, which was computed by the policy iteration algorithm on the log data of the System P, and served as the starting policy for System RL, was improved online by the R-Learning algorithm during the evaluation. The users visited collectively 1606 states. Actually, this is a small percentage of all the possible states. So, in practice the learning that we could perform with the collected data is still minimal compared to that achievable if the system had been used for a longer period.

We analyzed the changes made by R-Learning to the optimal policy by looking at the state-action values of the best actions, i.e., the actions with the highest value, for each of the possible initial states. The best action, computed by R-Learning, changed for 29.8% of the initial states. This clearly indicates that RLradio RL had a different channel selection strategy. This is confirmed by observing the log files and the state-to-action map of the learned policy. Several different patterns regarding the sequence of the suggested music channels could be detected. For instance, there is a clear preference of the system to keep proposing a channel if the user does not use the *next* button. We also observed the tendency to switch to another channel, which received

Table 3: Mean and standard deviation (SD) values of the responses to the usability survey

Stat.	P (SD)	RL (SD)	p-value
S1	4.13 (0.43)	4.03 (0.56)	0.317
S2	3.33 (0.92)	3.50 (0.82)	0.519
S3	3.70 (0.92)	4.13 (0.82)	**0.012**
S4	4.63 (0.49)	4.70 (0.47)	0.527

Table 4: Evaluation results summary

Result	P	RL	Imprv.
Avg. track's listening time percentage	64.35%	67.41%	+4.76%
Avg. daily listening time	62.6	75.5	+12.9 min
Percentage of users that listened to the suggested tracks longer while using the system vs. the other system	36.67%	63.33%	n.a.
Intention to use the system	3.70	4.13	p-value 0.012

by the user some initial explicit preference, when the user starts using the *next* button.

Finally, we analyzed the answers of the surveys by performing the related-samples Wilcoxon signed-rank test on the results of the (within-group) users' evaluations, as shown in Table 3.

As the user interfaces are absolutely identical for the two compared systems, we did not expect notable changes regarding their usability. Overall, RLradio (P and RL) received a very good evaluation, especially with respect to the quality of the played music (S1) and the ease of use (S4), and, as expected, with no significant difference between the two systems. Regarding the channel switching time (S2), we did measure an improvement, although it is not significant. We measured a statistically significant improvement of RLradio RL for statement S3 ("I would use such a player if available."). This result is very important as it represents the users' likeliness of using the player, and shows that the users had generally a higher satisfaction level when using System RL.

In conclusion, the evaluation results and the improvement of RLradio RL over RLradio P are summarized in Table 4.

6. RELATED WORK

The importance of implicit feedback has been recognized by some researchers who noted that the acquisition of user preferences, or explicit feedback (e.g., ratings), which is the most popular type of input for a RS, requires a remarkable user effort [12, 9]. Moreover, explicit feedback data is not always available, thus implicit feedback can come up as a necessary alternative to build the user profile.

In [9] the authors try to characterize explicit and implicit feedback on Last.fm. The dataset consists of explicit positive feedback (loved tracks) and implicit positive feedback (the number of times a track is played). Their analysis shows that explicit feedback is very scarce. Moreover, they found that the rate at which a user provides explicit feedback decreases with time, and that overall, leaving explicit feedback has a negative effect on the user's behavior. [14] presents an analysis of the functional dependency between implicit and explicit feedback, coming to the conclusion that the prediction of a user's explicit feedback (ratings), starting from implicit feedback, can be made with an acceptable level of accuracy. [8] presents an extension of matrix factorization to incorporate implicit feedback (whether the user bought or not an item) into the optimization process. They show that a significant improvement of the RS accuracy can be achieved.

Regarding the second main issue tackled in our paper, i.e., the generation of a meaningful sequence of recommendations, [16] was the first to model sequential recommendation as a Markov Decision Process (MDP). They introduced a state model that represents previous system recommendations and user selections. Then, they used a model-based reinforcement learning algorithm to learn an optimal policy, showing the benefit of such approach in a book RS.

[10] applies MDP and reinforcement learning for selecting the next conversational act that the system must follow. The goal is to adapt the recommendation process, and it is shown that in this way the effectiveness of the interaction is improved, as measured by the ratio of successful sessions (when the user actually bought something) over the full set of recommendation sessions. In particular, it is shown that by leveraging user actions (implicit feedback), the system can reduce the usage of explicit feedback, e.g., in that application, the amount of travel specific preferences.

The most similar approach to generate a sequence of music track recommendations is probably presented in [13]. Here, an approach to automatically generate a playlist given a song to start with (seed song), and the user's immediate feedback to the system proposals is presented and evaluated. The authors use an audio-based similarity measure to generate recommendations. The user gives feedback by pressing a skip button if the user dislikes the current song. Songs similar to the skipped songs are removed, while songs similar to the accepted ones are added to the playlist. It is shown that by using audio similarity and simple heuristics it is possible to drastically reduce the number of necessary skips. We stress again that in our scenario the recommender cannot select the next track but only the next channel, hence it cannot choose any track among a set of available ones, as in this system.

RL approaches have been applied to the music domain in the past by [7], but not with the aim of proposing a sequence of music tracks to be listened to. The music sequencing problem was explored in [4], where the authors present a mobile music recommender for a group of users. Finally, in [3] the authors model sequential music recommendation as a case-based reasoning problem. To select each song in the sequence, they search for songs musically associated with the last song of the sequence. Then, the preferences of the audience, expressed as cases, are reused to customize the selection.

In conclusion, although RL techniques have been explored in the past in RSs research, there is no known approach that exploits them to deal with sequential music recommendation and that integrates both explicit and implicit feedback as we propose.

7. CONCLUSIONS

In this paper we have presented RLradio, a novel type of music recommender system, that learns how to automatically switch channel from a collection of radio channels, and hence offering more useful music to the user, i.e., music that the user listen more. RLradio is a system sitting on the user side, not the provider side, as it is more common in currently available recommender systems [11]. RLradio exploits and combines explicit channel preferences and implicit feedback in a MDP that is solved with reinforcement learning. Implicit feedback is generated by the user when he requests a new track, from the current or a different channel, while listening to music in the current channel.

The users tested both a baseline system that chooses the next channel to be played next by considering only the user's channels' preferences, and a system extending the baseline with reinforcement learning that is exploiting implicit feedback. The proposed RL-based approach has shown a significant performance improvement with respect to the baseline. Overall, the users listened to a larger part of the recommended tracks and their daily listening sessions were longer. Hence, the proposed approach can minimize the user intervention, i.e., requesting a new track from the available channels, while the radio is playing music that the user does not like. The subjective evaluation of the two compared system variants also showed a significant improvement of the users' intention to use the reinforcement learning based system over the baseline.

The proposed techniques are not domain specific. In this paper we focussed on a radio player, but we are developing a new application of the same techniques in the domain of adaptive document presentation. The goal is to find a meaningful and effective way to present chunks of a document as the user reads the presented chunks and requests a new one. We are developing a system that adaptively presents information about the patient disease in a hospital information system.

As we have mentioned already, RLradio is not currently designed to choose a channel based on information about the track that is going to be played. But this information could be available in some Internet radios or in a different application scenario. Therefore, in the future we plan to investigate ways to combine reinforcement learning and collaborative filtering techniques (CF) to extend channel selection with actual track selection. Such hybridization would allow to generate accurate user preference estimation using CF, and would effectively solve the exploration/exploitation tradeoff using RL.

8. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] G. Adomavicius and A. Tuzhilin. An architecture of e-butler: A consumer-centric online personalization system. *International Journal of Computational Intelligence and Applications*, 2(3):313–327, 2002.

[3] C. Baccigalupo and E. Plaza. A case-based song scheduler for group customised radio. In *7th International Conference on Case-Based Reasoning, ICCBR 2007, Belfast, Northern Ireland, UK, August 13-16, 2007, Proceedings*, pages 433–448, 2007.

[4] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger. Incarmusic: Context-aware music recommendations in a car. In *E-Commerce and Web Technologies - 12th International Conference, EC-Web 2011, Toulouse, France, August 30 - September 1, 2011. Proceedings*, pages 89–100. Springer, 2011.

[5] R. Burke. Hybrid web recommender systems. In *The Adaptive Web*, pages 377–408. Springer Berlin / Heidelberg, 2007.

[6] Ò. Celma and P. Lamere. If you like radiohead, you might like this article. *AI Magazine*, 32(3):57–66, 2011.

[7] N. Collins. The potential of reinforcement learning for live musical agents. In *Proceedings of ICMC2008*, Belfast, 2008.

[8] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. IEEE Computer Society, December 2008.

[9] G. Jawaheer, M. Szomszor, and P. Kostkova. Characterisation of explicit feedback in an online music recommendation service. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 317–320, 2010.

[10] T. Mahmood, F. Ricci, and A. Venturini. Improving recommendation effectiveness by adapting the dialogue strategy in online travel planning. *Journal of Information Technology and Tourism*, 11:285–302, 2010.

[11] F. J. Martin, J. Donaldson, A. Ashenfelter, M. Torrens, and R. Hangartner. The big promise of recommender systems. *AI Magazine*, 32(3):19–27, 2011.

[12] D. Oard and J. Kim. Implicit feedback for recommender systems. In *Proceedings of the AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

[13] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR 2005, 6th International Conference on Music Information Retrieval, London, UK, 11-15 September 2005, Proceedings*, pages 634–637, 2005.

[14] D. Parra and X. Amatriain. Walk the talk - analyzing the relation between implicit and explicit feedback for preference elicitation. In *User Modeling, Adaption and Personalization - 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings*, pages 255–268, 2011.

[15] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer Verlag, 2011.

[16] G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.

[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.