Build Your Own Music Recommender by Modeling Internet Radio Streams

Natalie Aizenberg
Yahoo! Labs, Haifa, Israel
anatalia@yahoo-inc.com

Yehuda Koren
Yahoo! Labs, Haifa, Israel
yehuda@yahoo-inc.com

Oren Somekh
Yahoo! Labs, Haifa, Israel
orens@yahoo-inc.com

ABSTRACT

In the Internet music scene, where recommendation technology is key for navigating huge collections, large market players enjoy a considerable advantage. Accessing a wider pool of user feedback leads to an increasingly more accurate analysis of user tastes, effectively creating a "rich get richer" effect. This work aims at significantly lowering the entry barrier for creating music recommenders, through a paradigm coupling a public data source and a new collaborative filtering (CF) model. We claim that Internet radio stations form a readily available resource of abundant fresh human signals on music through their playlists, which are essentially cohesive sets of related tracks.

In a way, our models rely on the knowledge of a diverse group of experts in lieu of the commonly used wisdom of crowds. Over several weeks, we aggregated publicly available playlists of thousands of Internet radio stations, resulting in a dataset encompassing millions of plays, and hundreds of thousands of tracks and artists. This provides the large scale ground data necessary to mitigate the cold start problem of new items at both mature and emerging services.

Furthermore, we developed a new probabilistic CF model, tailored to the Internet radio resource. The success of the model was empirically validated on the collected dataset. Moreover, we tested the model at a cross-source transfer learning manner – the same model trained on the Internet radio data was used to predict behavior of Yahoo! Music users. This demonstrates the ability to tap the Internet radio signals in other music recommendation setups. Based on encouraging empirical results, our hope is that the proposed paradigm will make quality music recommendation accessible to all interested parties in the community.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords

collaborative filtering, internet radio, music recommendation

1. INTRODUCTION

The convergence of the media industry with the Internet and the appearance of mobile devices with rich media playing capabilities have created a variety of new services. In particular, nowadays users can consume, buy, share, and review media items in many new exciting ways. Yet, the variety and volume of media items is overwhelming. As a result, users may find it difficult to navigate through the vast variety of new media items and get what they like. To overcome this explosive content volume, helping users find items they like, many new recommendation services have emerged in recent years. These services are usually based on users' feedback and stated preferences, and use editors, content analysis or rely on the wisdom of crowds to tag their database items.

Music recommender systems often employ *Collaborative Filtering* (CF) [22], which relies only on past user behavior—e.g., their previous transactions or feedback—and does not require the creation of explicit profiles. Notably, CF techniques require no domain knowledge or content analysis. CF excels at exploiting popularity trends, which drive much of the observed user interaction, and typically would be completely missed by content based approaches. In addition, relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes. Consequently, CF attracted much attention in the past decade, resulting in significant progress and adoption by successful commercial systems.

Notwithstanding its great success, CF poses some restrictions on system designers. First and foremost, as a paradigm relying on the wisdom of crowds, its success strongly depends on having a sufficient amount of user feedback signal on record. Therefore, a new service with a limited user base will have to resort to alternative recommendation approaches, at least until gathering a critical mass of users. Given the vital role recommendation engines play in music consumption, this may put new market players at a competitive disadvantage. Furthermore, even successful CF systems sometimes fail to model new items, which have a dearth of user signal, leading to the item cold start problem.

The main goal of this work is to lessen these restrictions. It so happens that there is a huge repository of "free music" that is available to users today and is not widely exploited, namely, online media streams such as Internet radio stations (referred to hereafter as "stations"). We show that when properly analyzed, stations form an invaluable collection of human generated musical signals. They allow modeling the nature of played tracks and artists, relating tracks, as well as monitoring popularity trends. All these merely requires collecting the publicly available metadata (namely, track and artist names) of stations' playlists along time.

We exploit the wealth of information provided by stations using a new probabilistic recommendation model. For each station the model predicts the corresponding probability distribution of items to be played. The model offers the flexibility to represent stations either directly or through the collection of previously played tracks. Moreover, the temporal dynamics in the system, such as changes of music type by time of day, or similarity of adjacently played tracks, are naturally captured by the model. The model also shares information between all tracks belonging to the same artist, which is important to learning less frequent tracks. Furthermore, the model caters for the kind of feedback prevalent at music recommendation which is mostly unary positive feedback in the form of playing

WWW, '2012 Lyon, France
ACM 978-1-4503-1229-5/12/04.

events, and occasionally binary feedback including also negative reactions (e.g., play/skip, like/dislike, thumbs-up/down).

By applying the model to a playlists' dataset crawled from thousands of stations, we show its ability to model genuine musical patterns and establish similarity patterns across tracks, artists, and stations. We also test the model in a *cross-source learning* setup, where we transfer the learned model to a different musical domain consisting of Yahoo! Music users. We find that the model learned from stations data is valuable for building a recommender system to Yahoo! Music users.

To summarize, we present a paradigm consisting of a new data source together with a tailored model, which enables bootstrapping music recommendation system. This enables young market players (or other interested parties, like researchers) to provide a quality recommendation system for their items. At the same time, mature systems would be able to exploit the proposed paradigm for mitigating the cold start problem typical to new items.

2. BACKGROUND AND RELATED WORK

There are different approaches to music recommendation, including: (1) Expert-based approaches that are based on human annotation of music data such as the *Music Genome Project* [4], which is used by the Internet music service Pandora.com. (2) Social-based approaches that characterize items based on textual attributes, cultural information, social tags and other kinds of web-based annotations [13, 15, 25]. (3) Content-based approaches that analyze the audio content for characterizing tracks and establishing item-to-item similarities [2]. (4) Collaborative Filtering (CF) methods that analyze listening patterns (or other sorts of feedback) by many users, in order to establish similarities across users and items.

Each of the above approaches comes with distinct disadvantages. An expert-based approach is extremely time consuming and hard to scale. Social tagging is unlikely to provide a complete characterization of music. Content based approaches, even at their best, miss important relations and distinctions humans make between songs; the reader is referred to the interesting piece by Slaney [26] on the topic. The ability to directly model user desires makes CF among the more accurate and serendipitous approaches to recommendation. In addition CF naturally learns popularity trends by observing actual user behavior, which the other approaches overlook. However, its availability is severely hindered by the need to have a substantial amount of relevant user behavior on record. The approach described in this paper aims at alleviating this central issue of CF.

Latent factor models constitute one of the leading CF techniques. They characterize both items and users as vectors in a space automatically inferred from observed data patterns. The latent space representation strives to capture the semantics of items and users (or, *stations* in our case), which drive their observed interactions. In music, factors might measure obvious dimensions such as genres, tempo or pitch or less well defined dimensions such as the emotions provoked by the song or the social target audience. One of the most successful realizations of latent factor models is based on *matrix factorization* (MF); see, e.g., [12]. These methods have become very popular in recent years by combining good scalability with predictive accuracy. In addition, MF provides a substantial expressive power that allows modeling specific data characteristics such as temporal effects [11], item taxonomy [9] and attributes [1], social relations [8], and 3-way interactions [21]. This paper adopts some practices common at MF methods, while using a likelihood maximization principle. This allows us the needed flexibility for modeling temporal effects and tying tracks with their artists.

Literature provides several probabilistic CF models, most predict a probability distribution among rating values for a single item [23, 24], or induce order among item pairs [20]. The probabilistic model we suggest predicts a probability distribution of all items for a fixed station. In this sense, it bears more resemblance to a few other works producing such distributions [6, 28], with an important distinction that we follow an MF-styled model with gradient descent optimization and an ability to account also for negative feedback.

Among MF methods our work is mostly related to those dealing with implicit user feedback (as opposed to the explicit feedback more common in MF literature), such as [7, 16, 20]. Indeed such methods could provide a reasonable alternative to the model we suggest. We would argue that the unique appeal of our model is in combining together: (1) An ability to produce a single probability distribution over all items for a given user or station. (2) Borrowing the rich expressive power of MF that offers a considerable flexibility in modeling diverse real life situations. (3) While most implicit feedback-oriented models use sampling for drawing items a user did not interact with, our method employs a more principled sampling approach (in Section 4.5). (4) The model we describe includes several accommodations tailored to music recommendation and in particular to Internet radio signals. (5) Accounting for both situations of unary (positive-only) user feedback and situations of binary (positive and negative) user feedback.

Our work is related to the topic of automatic playlist generation, though that subject has several aspects that are beyond the scope of this paper (e.g., ensuring the overall coherence and diversity of a full sequence). We would refer the reader to some recent works on the subject [5, 19, 28]. In particular, Ragno et al. [19] modeled playlist generation by initiating a random walk over expertly authored streams, such as Internet radio playlists. We note that this work also identifies Internet radio streams as a valuable resource.

A few recent works [14, 17, 27] introduced transfer learning methods for collaborative filtering. They allow jointly modeling user ratings made in different domains, in order to lessen the data sparseness problem. This is related to our goal of inferring generic models of user preferences in music from Internet radio playlists. Our music-oriented work takes a different perspective, by focusing on the identification of a viable source dataset and devising models suitable for Internet radio streams.

3. INTERNET RADIO AS A DATA SOURCE

Our data source is Internet radio streams metadata, rather than the more commonly used users' playlists and feedback data. Hence, instead of encompassing signals generated by many music fans, it holds signals created by groups of professional music experts and amateur music savvies.

3.1 Advantages

We argue that a dataset based on Internet radio streams metadata provides several advantages:

- **Freshness** - The music arena is highly dynamic and music trends are constantly changing. Existing datasets, whether public or created by a single permission crawl, provide only a snapshot of time. Therefore, they fail to follow the ever changing music scene and cannot capture new trends. In contrast, an Internet radio based dataset can provide a continuous cover where "fresh" metadata is constantly arriving.
- **Completeness** - Existing public music datasets usually provide only partial and/or implicit metadata of their playlists. For example, the Yahoo! music dataset [9] does not expose the artist and track names and uses numeric id's instead. Another example is the Last.FM 360K dataset[1] where only artist

[1] `http://mtg.upf.edu/node/1671`.

Rank	Artist	Title	Album	#Plays	#Stations
1 (8)	Katy Perry	Last Friday Night (T.G.I.F.)	Teenage Dream	4265	312
2 (1)	Adele	Rolling In The Deep	21	4160	407
3 (151)	Bruno Mars	Lazy Song	Doo-Wops & Hooligans	3568	303
4 (23)	Lady Gaga	The Edge Of Glory	Born This Way	3099	262
5 (59)	Coldplay	Every Teardrop is a Waterfall	Every Teardrop is a Waterfall	3080	229
6 (2)	Adele	Someone Like You	21	2903	243
7 (>420)	Alexandra Stan	Mr. Saxobeat	(Single)	2871	324
8 (3)	Foster the People	Pumped Up Kicks	Foster the People–EP	2793	187
9 (43)	LMFAO	Party Rock Anthem	(Single)	2733	194
10 (63)	Britney Spears	I Wanna Go	Femme Fatale	2708	206

Table 1: Dataset top 10 tracks (Last.FM ranks are added in brackets).

information is provided. This trend is not likely to change in the future, where the high valued data will continue to be only partly exposed. In contrast to data keepers who tend to protect their data assets, Internet radio stations are promoting themselves by publicly exposing their content. Accordingly, the lion's share of Internet radio metadata is explicit and usually complete (although it is noisy and ambiguous in part).

- **Robustness** - Collectors of Internet radio based datasets are not dependent on a single data source that may shut information seekers off or may cease to exist. On the contrary, a major portion of the radio stations are operated by a stable set of playlist editors with known identities. In addition, with the growing trend of personalized music streams being shared publicly, new stations are joining the scene constantly.
- **Scale** - The number of Internet radio stations is increasing constantly and already measured worldwide in hundreds of thousands. Since most stations are constantly streaming music, Internet radio based dataset size is theoretically unbounded.
- **Diversity** - Playlists generated by a specific service provider (such as Yahoo! and Last.FM) are expected to be biased in nature according to the taste of the specific provider's editors and/or users' collective profile. In contrast, a dataset based on metadata stemming from many thousands of independent Internet radio stations is expected to be more diverse and eclectic. We believe that some of the findings presented in the sequel indeed indicate this phenomenon.
- **Accessibility** - Internet radio streams and metadata are constantly accessible to every information seeker with Internet access. Moreover, as we shall see in the sequel, the barriers for monitoring and aggregating the metadata are quite low.

We dare to conclude that the above advantages make Internet radio streams a valuable highly accessible resource available for both-researchers and service providers that wish to "feel" the pulse of the lively music scene.

3.2 Collection procedure

The results reported in this paper are based on a snapshot of data collected during a period of 15 days between September 22^{nd} and October 6^{th}, 2011. The data was collected across 4147 stations (some of which were later filtered out). For each sampled track play, we extracted the metadata, the station local time of play, and its corresponding system time. The monitored stations are all associated with the ShoutCast directory[2] and therefore tend to provide metadata in a similar manner. This fact helps in retrieving and parsing the track metadata. It is noted that the dataset can be arbitrary large by sampling more stations over a longer period.

The collection of the playlists metadata consisted of three phases: station list composition, playlist collection, and meta data parsing. We first composed the list of stations, sampling all major genres in the ShoutCast directory. Among these, were stations not necessarily dedicated to music, for example stations under the sports and news subdirectories. According to our examination of the data, the stations also exhibit a global diversity, as we encountered many European stations in our collection. We then monitored the listed stations, while collecting and registering their playlist metadata, as well as associated time of play information.

We applied some general parsing heuristics to each playlist, in order to extract artist and track titles. Extraction was mainly based on "–" (dash) separation, where a play is represented by a string formatted as <artist name> – <track title>. Extractions were also normalized in several ways: lower casing, preceding enumeration elimination, etc. Normalized titles were then subjected to frequency based filtering, to ensure they are not wrongly parsed.

3.3 Statistics

Rank	Name	#Plays	#Unique tracks	#Stations
1 (4)	The Beatles	84587	438	754
2 (49)	Michael Jackson	22850	235	1033
3 (47)	AC/DC	20964	223	688
4 (12)	Rihanna	19308	89	798
5 (90)	Madonna	18671	191	890
6 (7)	Lady Gaga	18157	63	735
7 (33)	The Rolling Stones	17961	371	796
8 (48)	Black Eyed Peas	16151	105	758
9 (18)	Queen	15920	260	934
10 (3)	Coldplay	15364	86	718

Table 2: Dataset top 10 artists (Last.FM ranks are added in brackets).

In this subsection we demonstrate the richness of our dataset, which is essential to supporting applications such as music recommending systems. The statistics presented here include 563,417 unique tracks that were played a total of 6,727,692 times, by 96,681 different artists, over 3,541 stations. We have restricted our records to include only stations that were playing at least 10 plays of more than 5 different tracks. In addition, we have included only artists associated with more than 10 plays.

Top tracks and artists.

The top 10 most played tracks are listed in Table 1 along with their number of plays and number of stations. Corresponding ranks

[2]An Internet radio stations directory `http://www.shoutcast.com/`. The number of listed stations may vary from tens of thousands to more than one hundred thousand stations, depending on their availability.

Rank	Stream Title	Genre	#Unique tracks	#Artists
1	RMNgoodtimes Wir Lieben Oldies	Oldies, Goldies, 60er 70er 80er	4273	1767
2	KX Classics	Oldies 60s 70s 80s	4132	1369
3	AFTC Radio	Free Form, Eclectic, Non-Commercial	3992	1359
4	Always Country	Country	3864	782
5	AVRO 60ies Steenen Tijdperk	60s	3761	930
6	Humboldt 101	Easy Listening	3600	1276
7	HardRockin80s.com	80s, metal, rock	3472	578
8	Public Domain Jazz	Big Band Jazz Swing	3452	139
9	3WK Classic Alternative Radio	Classic Alternative, 80s, 90s	3446	660
10	SwissGroove	Jazz, Funk, World, Soul, Brazil, Nu Grooves	3359	1805

Table 3: Dataset most diverse stations. Stations can be found on the ShoutCast directory and played using WinAmp.

reported by Last.FM appear in brackets[3]. Although the two rankings are quite different, four tracks appear in both top-10 lists. Examining the list reveals that it includes 9 different artists compared to the 5 artists included in the corresponding Last.FM list.

Similarly, the top 10 artists are listed in Table 2 along with their number of plays, number of tracks, number of stations, and Last.FM rank (in brackets). Here, 3 artists are also found among the top-10 of Last.FM, where the rest lie within the top-100. The top two artists are *The Beatles* and *Michael Jackson*, where the former showing a staggering 3.8 fold increase in number of plays compared to the latter or any other artist.

General statistics.

Next, we present a few general statistics of the dataset: (1) Number of plays per track/artist frequencies – Figure 1.a; (2) Number of stations per track/artist frequencies – Figure 1.b; and (3) Number of unique tracks per artist frequency – Figure 1.c.

It is observed that all distributions follow a power law or double Pareto law with "healthy" long heavy tails.

Diversity.

The next two figures demonstrate the diversity of the monitored stations. In Figure 1.d we plot a histogram of the number of unique tracks per station (using bins of width 200; bin centers are shown). It is observed that more than two thirds of the stations played at least 200 unique tracks during the monitored period. Similarly, in Figure 1.e the histogram of the number of unique artists per station is plotted. It is observed that more than half of the stations hosted at least 200 unique artists during the monitored period.

Top 10 most diverse stations are listed in Table 3, along with their genres, numbers of tracks, and number of artists. These stations (and many others as is observed in Figure 1.d) have hardly played tracks more than once over the data collection period. Also visible from the table is the diversity of genres declared by these stations.

The next figure provides an educated guess to the question whether an aggregation performed on a larger scale would provide more diversity and richness. To produce the curve, we ordered the stations randomly and monitored the increase in number of unique tracks as a function of the number of stations. The results are plotted in Figure 1.f, which averages multiple random orderings of the stations. Examining the figure it is observed that the curve is obviously sub linear but has not reached its asymptote. Hence, with more than 3000 monitored stations we have not exhausted the diversity potential of the Internet radio streams resource, and adding more stations is likely to materially increase the number of tracks.

4. PLAYLIST MODELING

We model playlists using a latent factor model, where each playlist induces a probability distribution over the items. Model parameters are learned in order to maximize the probability of playing the listened items. The model directly captures the relations between tracks and their corresponding artists, thereby indirectly tying all tracks belonging to the same artist. In addition, the model utilizes temporal effects underlying the data.

4.1 Basic notions

We observe musical *items* that are arranged within *playlists* associated with *stations*. Here items refer to both *tracks* and *artists* unless otherwise stated. Ultimately, we are interested in environments where both users and stations coexist. In particular, we will be interested in modeling new users based on the station editors wisdom. Hence, based on the context, we may exchange the notion stations with *users*, which are indexed by u, and are technically interchangeable with stations. We denote timestamps by t, where a timestamp corresponds to the start play of a track. The artist playing track i is denoted by $a(i)$. The sequence P_s denotes a list of tracks played by station s along with their associated timestamps. These playlists are arranged within a train set $\mathcal{S}$.

4.2 Modeling outline

We propose a method for modeling playlists. More specifically, given a playlist, the model predicts the probability distribution of the next played item. Our model maps both items and stations to latent factor vectors, a representation proven successful in many recommendation systems. Formally, the latent factor representation is defined as follows:

- Each item i is mapped into a vector $p_i \in \mathbb{R}^\ell$.
- Each station s is mapped into a vector $v_s \in \mathbb{R}^\ell$.

Typically, we use a latent space dimensionality of $20 \leqslant \ell \leqslant 50$. In addition, we introduce biases for items, such that the bias of item i is denoted by $c_i \in \mathbb{R}$. Such biases reflect the varying popularity levels of items. Henceforth, we denote all model parameters by Θ.

Musical artists often have a distinct style that can be recognized in all their songs. Therefore, we share parameters to reflect the affinity of tracks by the same artist. Parameters associated with a track i sum both its own specific representation, together with the one associated with its artist ($a(i)$). This helps in modeling sparsely observed tracks. Each track i is associated with an artist-enhanced latent factor vector

$$q_i \stackrel{\text{def}}{=} p_i + p_{a(i)} \tag{1}$$

Similarly, the artist-enhanced bias of track i is

$$b_i \stackrel{\text{def}}{=} c_i + c_{a(i)} \tag{2}$$

We denote by $r_{si;t}$ the affinity of item i to station s at time t; exact definition will follow shortly. Accordingly, given a target station at a certain time, we can rank all items by their decreasing affinity values. We model the likelihood of observing i as the item

[3]Last.FM charts are available at `http:/www.last.fm/charts`

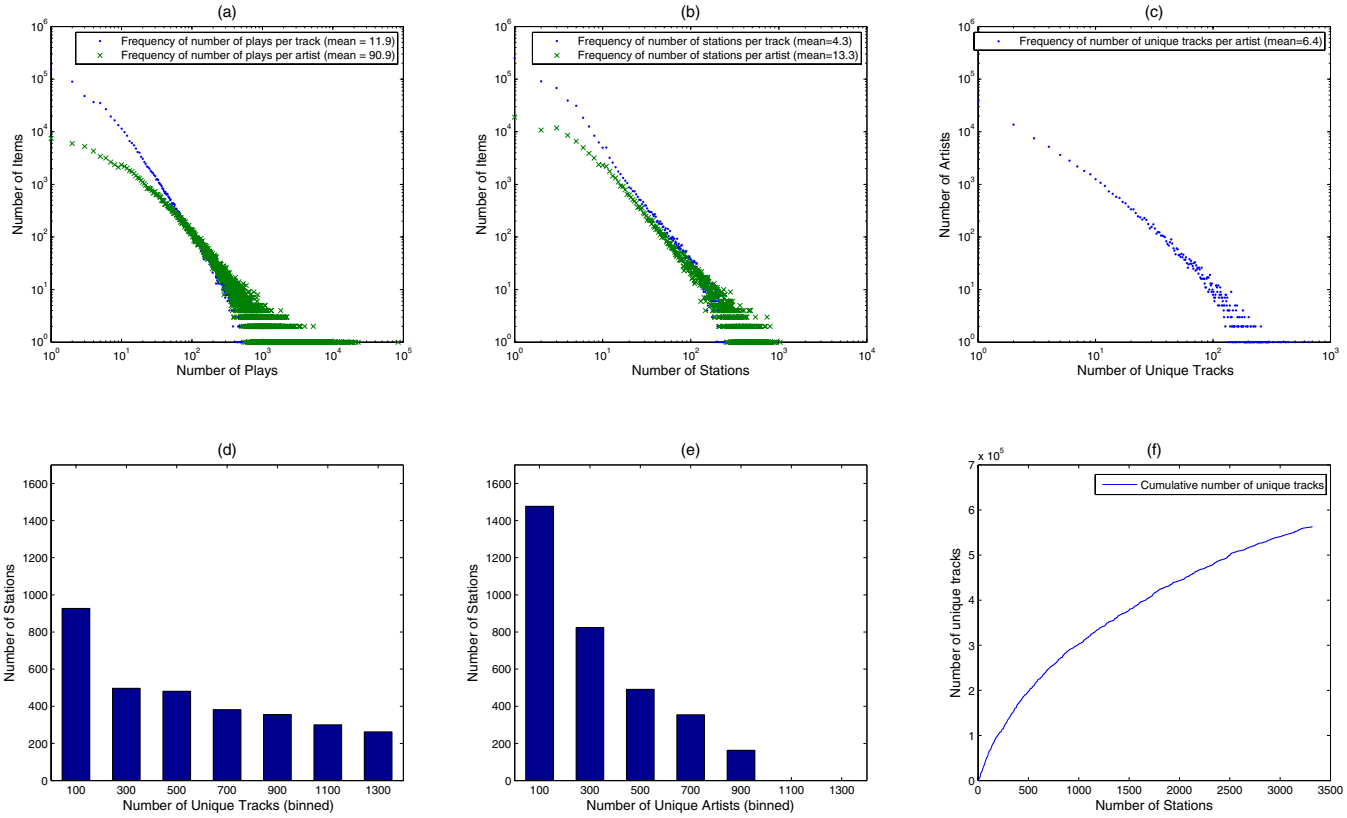

Figure 1: Dataset statistics: (a) Number of plays per track/artist frequencies; (b) Number of stations per track/artist frequencies; (c) Number of unique trackd per artist frequency; (d) Number of unique tracks per station; (e) Number of unique artists per station; and (f) Cumulative number of unique tracks (the stations are randomly ordered).

played in station s at time t by the multinomial distribution

$$P(i|s;t,\Theta) = \frac{\exp(r_{si;t})}{\sum_j \exp(r_{sj;t})} \tag{3}$$

In the followings we often omit Θ from the probability definition, using the notation $P(i|s;t)$. The learning task is finding the model parameters that maximize the log-likelihood of observing the train set, which in turn dictate all station-item affinities.

4.3 Station-item affinities

A first approximation of the affinity function would be

$$r_{si;t} \stackrel{\text{def}}{=} b_i + q_i^T v_s \tag{4}$$

Hence, affinity depends on the inner product between the respective factor vectors, and on the item bias. This follows a common practice in matrix-factorization (MF) recommenders [12]. However, the reader should note the difference between the setups. While MF deals with recovering provided ratings, usually by minimizing a squared error loss, we will aim at a log-likelihood maximization involving a unary (or sometimes binary) observed signal.

Radio stations are likely to play different kinds of music during different times of the day, in order to cater to the time-dependent moods and contexts of their listeners. Hence, we split the day into different time slots. In our implementation we use 8 equal slots (12am–3am, 3am-6am, 6am-9am,...); alternative choices could be considered as well. Accordingly, we introduce the notation $1 \leqslant \text{slot}(t) \leqslant 8$ as the integer denoting the slot corresponding to time t. The station profile during time slot k is denoted by latent factor vector $v_s^{(k)} \in \mathbb{R}^\ell$, which is added to the model parameters set Θ.

In addition, it is expected that adjacently played items will be more tightly related. This stems from the increased chance that both were played at the same program, and also from a tendency of playlist editors to impose smoothness between closely played items. Hence, for a station s, a timestamp t, and a time window w, we denote by $P_s^{(t,w)}$ the set of tracks played in station s during time $[t-w,t)$. Utilizing the item vectors, this set of items will be characterized by the vector $|P_s^{(t,w)}|^{-0.5} \sum_{j \in P_s^{(t,w)}} q_j$. In our implementation a time window of 30 minutes was used.

Thus, we extend (4) to account for the aforementioned temporal effects by evolving it into

$$r_{si;t} \stackrel{\text{def}}{=} b_i + q_i^T \left(v_s + v_s^{(\text{slot}(t))} + \frac{1}{\sqrt{|P_s^{(t,w)}|}} \sum_{j \in P_s^{(t,w)}} q_j \right) \tag{5}$$

4.4 Log-likelihood maximization

We seek model parameters (Θ) in a way that maximizes the log-likelihood of the training set

$$L(\mathcal{S};\Theta) \stackrel{\text{def}}{=} \sum_{P_s \in \mathcal{S}} \sum_{(i,t) \in P_s} \log P(i|s;t,\Theta) \tag{6}$$

Note that playlist P_s is not a set, so the same item can appear in it multiple times.

Let us consider the case where negative feedback is also to be found, e.g., when we observe users skipping tracks or voting them down. We would like to account also for such feedback by maximizing the likelihood of not playing those negatively voted items. We denote by N_s the list of items with negative feedback associated with s. Then, the maximized log-likelihood becomes

$$L(\mathcal{S};\Theta) \stackrel{\text{def}}{=} \sum_{P_s\in\mathcal{S}}\sum_{(i,t)\in P_s} \log P(i|s;t,\Theta) + \sum_{N_s\in\mathcal{S}}\sum_{(i,t)\in N_s} \log(1-P(i|s;t,\Theta)) \quad (7)$$

Within the scope of this paper, we do not encounter negative feedback, and hence concentrate on optimizing (6). However, in other conceivable scenarios, when models are applied to users, negative feedback is expected to emerge. In such scenarios, pursuing optimization of (7) would follow an optimization process analogous to the one to be described shortly.

4.5 Optimization process

Learning proceeds by stochastic gradient ascent. Given a training example (s,i,t) we update each parameter $\theta\in\Theta$ by

$$\Delta\theta = \eta\frac{\partial P(i|s;t)}{\partial\theta} = \eta\left(\frac{\partial r_{si;t}}{\partial\theta} - \sum_j P(j|s;t)\frac{\partial r_{sj;t}}{\partial\theta}\right) \quad (8)$$

where η is the learning rate.

However, such a training scheme would be too slow to be practical, as each update rule requires summing over all items. Thus, we resort to sampling the weighted sum in (8), which is based on the importance sampling trick proposed by Bengio and Senécal [3].

With importance sampling we draw items according to a *proposal distribution*. In our case we assign each item a probability proportional to its empirical frequency in the train set, and denote this proposal distribution by $P(i|\mathcal{S})$. Consequently, items are sampled with replacement from $P(i|\mathcal{S})$ into a list $\mathcal{J}$. An efficient way to do so is to uniformly sample an (s,i,t) tuple from the training set, and add i to $\mathcal{J}$. Thus, the expensive-to-compute $P(i|s;t)$ probabilities are approximated with the weighting scheme

$$w(i|s) = \frac{\exp(r_{si;t})/P(i|\mathcal{S})}{\sum_{j\in\mathcal{J}}\exp(r_{sj;t})/P(j|\mathcal{S})} \quad (9)$$

Consequently, the approximated gradient ascent step given training example (s,i,t) will be

$$\Delta\theta = \eta\left(\frac{\partial r_{si;t}}{\partial\theta} - \sum_{j\in\mathcal{J}} w(j|s)\frac{\partial r_{sj;t}}{\partial\theta}\right) \quad (10)$$

As mentioned in [3], it is desirable that the size of set $\mathcal{J}$ grows as the training process proceeds, because at later training phases more delicate parameter adjustments are needed. We employ a simple rule for controlling the sample size ($|\mathcal{J}|$) based on fitness of current estimate. Given a training example (s,i,t), we keep sampling items into $\mathcal{J}$ till satisfying:

$$\sum_{j\in\mathcal{J}} P(j|s;t) > P(i|s;t) \Leftrightarrow \sum_{j\in\mathcal{J}}\exp(r_{sj;t}) > \exp(r_{si;t}) \quad (11)$$

The adaptive sampling automatically lets the sample size grow when parameters are nearing final values and the correct item is getting a relatively high probability. For efficiency we also limit the maximal sample size to 1000. Henceforth, we dub the method described in this section as a *station-based model*.

We provide a few finer details on tuning the learning algorithm. In our implementation we ran the process for 20 sweeps over the training data. During the k-th sweep we set the learning rate to $\eta = 0.005/k$, based on validation data performance. Factorization-based recommenders usually employ regularization in order to avoid overfitting. The most common regularization is weight decay, which penalizes the squared magnitude of each learned parameter; see, e.g., [12]. This amounts to assuming a normal prior centered at zero for each parameter. In our experiments we observed that such a weight decay has a very minor positive impact on generalization performance. Thus, we have set the corresponding weight decay constant to 10^{-4}. Finally, in order to ensure numerical stability (important due to exponentiation operations) we clip all parameters to the $[-1,1]$ range throughout the whole learning procedure.

4.6 Modeling new users

A central goal of our work is leveraging the learned parameters in order to model new users or stations. Another related case is modeling evolving users as they add more tracks into their playlist. These scenarios may require a real time derivation of the user representation, hence re-training the model would not be desirable. In the followings we will discuss ways to fold new users into trained models, without requiring re-training.

When encountering a new user u, or a new station s, affinity functions (4) or (5) cannot be employed because they rely on precomputed parameters associated with the users/stations, namely, their latent factor vector v_s (or v_u).

We modify the original affinity definition (4-5) to allow a direct incorporation of new stations. The employed principle is that we can profile a station s through its playlist P_s, replacing the station factor vector v_s with $|P_s|^{-0.5}\sum_{j\in P_s} q_j$. This principle was successfully practiced in various matrix factorization techniques [10, 18]. More specifically, now we map each item i into three distinct factor vectors $p_i^{(1)}, p_i^{(2)}, p_i^{(3)} \in \mathbb{R}^\ell$, corresponding to the three distinct roles an item assumes: being recommended, annotating a station, and annotating a time window. Additionally, we tie the track and artist representation as before, by using the three summed vectors for track i: $q_i^{(1)} \stackrel{\text{def}}{=} p_i^{(1)} + p_{a(i)}^{(1)}$, $q_i^{(2)} \stackrel{\text{def}}{=} p_i^{(2)} + p_{a(i)}^{(2)}$, $q_i^{(3)} \stackrel{\text{def}}{=} p_i^{(3)} + p_{a(i)}^{(3)}$, in parallel to (1). Then, the affinity function between station s and item i at time t is defined as

$$r_{si;t} \stackrel{\text{def}}{=} b_i + \left(q_i^{(1)}\right)^T\left(\frac{1}{\sqrt{|P_s|}}\sum_{j\in P_s} q_j^{(2)} + \frac{1}{\sqrt{|P_s^{(t,w)}|}}\sum_{j\in P_s^{(t,w)}} q_j^{(3)}\right) \quad (12)$$

As usual, we learn the model parameters so as to maximize the log-likelihood of the train data. Importantly, since none of the model parameters describe stations, handling a new station becomes trivial. Namely, both existing stations, rapidly updated stations, and new users can all equally use the same affinity function (12), which allows ranking items without needing to retrain the model. In the following we will call such a model a *station-less model*.

As we will see in Section 5, the station-less model produces results of accuracy comparable or a bit better than those achieved by station-based model of Subsection 4.3. However, its training time is slower because of the indirect representation of stations. Hence, as an alternative, we can still train the model according to the original station-based representation. Then, when facing a new station s we profile items affinity to s at time t by

$$r_{si;t} = b_i + q_i^T\left(\frac{1}{\sqrt{|P_s|}}\sum_{j\in P_s} q_j + \frac{1}{\sqrt{|P_s^{(t,w)}|}}\sum_{j\in P_s^{(t,w)}} q_j\right) \quad (13)$$

We will compare the two alternatives ((12) and (13)) in Section 5.

5. EMPIRICAL STUDIES

5.1 Quantitative analysis

Evaluation cases.

We have based our empirical study on the Internet radio dataset described in Section 3; hereafter denoted as IRDB. For training and evaluating the models, we filtered out less reliable items, i.e., those appearing in less than three stations or played less than 10 times. Three major evaluation cases were considered.

First, we experimented with a "weak generalization" test. Here, each station went through a 75:25 split, with its earlier 75% plays inserted into the train set, and the latest 25% plays inserted into the test set. This represents a scenario of modeling existing stations.

We also experimented with a more difficult "strong generalization" test. Here we initially made a 75:25 split of whole stations, creating two station-disjoint subsets known as StrongGen1 and StrongGen2. The smaller subset, StrongGen2, is further split into (pseudo) train and test subsets, with the early 75% plays of each station residing in the train subset, and the last 25% residing in the test set. The models are fully trained only on StrongGen1. Then a trained model will observe some, or all, of the ratings in StrongGen2's train subset in order to predict its test subset. This represents a scenario of exploiting an existing model for predicting new stations or users.

Finally, we experimented with an even more challenging "cross-source learning" test. Here, a model fully trained on IRDB was used to predict the behavior of Yahoo! Music users. We relied on a dataset providing Yahoo! Music ratings, which was publicized during the KDD-Cup'11 contest, henceforth referred to as YMDB; see [9]. Matching the two datasets became all the more challenging, since most activity in YMDB was performed around the year 2005 or earlier. Given the shifts in musical trends throughout the 6+ year period separating the datasets, we are facing a truly out of sample test. In order to make the two datasets comparable, we filtered out of YMDB all items we could not match with IRDB items. Additionally, since YMDB is based on explicit user feedback (0-to-100 ratings), we took the $\geqslant 80$ ratings as positive feedback and discarded all other ratings.

Statistics of the resulting datasets are brought in Table 4. For each of the aforementioned evaluation cases we provide the number of playlists (number of stations at IRDB, or number of users at YMDB), the number of distinct items, the number of play (or positive feedback) events in the train set, and the number of play events in the test set. Note that at IRDB many items are repeating within the same station. Hence, we also devise an arguably purer test set without repetitions of items already played at the station (within either train or test period). This considerably reduces the number of plays in the test set, as shown in the last column of the table.

Evaluation metrics.

In order to measure the accuracy of the model we adopt a metric that was promoted in the KDD-Cup'11 challenge[4]. We pair each played item in the test set with a randomly drawn item not played by the user. We use the model to rank the two items for the relevant station (or user). The fraction of successfully ranked pairs is the metric value.

We picked this metric for several reasons. First it is a reasonable proxy of the relevant task we are after, which is suggesting the "right" item to the user. Notably, compared with other appropriate metrics that would require evaluating many competing items, the employed metric is much cheaper to compute. Finally, it can elegantly diminish effects attributed to the tendency of favoring universally popular items, as we will shortly explain.

We have experimented with three flavors of the metric, capturing different performance aspects, by changing the criteria for selecting the played item or for selecting the paired non-played item, as follows:

1. *NonRepeat-vs-Uni.* We account only for test items that have not previously appeared in the train or test set for the same station, corresponding to the right-most column of Table 4. Then, we compare each such played item against a random item uniformly picked from the items set.
2. *NonRepeat-vs-Pop.* As demonstrated in Figure 1.a, the number of plays per items follows a power law distribution. By the nature of the power-law distribution, we expect a played item to be more popular than a randomly picked item. Hence, having a played item being ranked higher than a randomly chosen item may merely reflect the ability to separate popular items from less popular ones. Accordingly, we rectify NonRepeat-vs-Uni in a way that better measures the personalization power of the algorithm. For each of the items in the test set we pick a competing item with probability proportional to the item's popularity (popularity is precisely defined as $P(i|\mathcal{T})$; see Section 4.5). This way, the played and the randomly picked items are coming from the same distribution. Other details of this metric equal those of NonRepeat-vs-Uni. Clearly, results under this metric are expected to be lower than those of NonRepeat-vs-Uni.
3. *Played-vs-Pop.* In the music domain, it is expected that over an extended period playlists will repeat already played songs. After all, listeners are expected to enjoy their favorite tracks more than once. Hence we offer an alternative to NonRepeat-vs-Pop, where we consider all played items in the test period, regardless of whether it is their first play in the station or not. This corresponds to the fifth column in Table 4. We expect prediction of repetitive plays to be easier than predicting first time plays. Hence, results under Played-vs-Pop are expected to be higher than those of NonRepeat-vs-Pop.

Baseline methods.

We compared our model against three baselines, as follows:

- *Popularity.* This baseline always favors the more popular item (i.e., the item with more plays overall).
- *Genre.* We used an internal Yahoo! Music database for associating tracks with genres. This way we could associate 45,713 tracks with one to six genres each. Overall, there are 130 editorial genres with the most popular being: Rock, Pop, Classic Rock, Country, Electronic/Dance, Metal, Alternative Rock, Hard Rock, Soft Pop, and Classic Soul. For each station we compute the genre distribution during the train period. Then, we score an item based on the mean probability its genres receive on the respective station. Since not all tracks could get associated with genres (mostly due to incomplete match between the databases), when applying this baseline we considered only a the subset of items that were assigned at least one genre.
- *k-Means* Since not all tracks could be associated with genres, we offer an alternative based on "made-up genres". Here we ran k-Means clustering on the items, where an item vector corresponds to its probability distribution across stations. We treat clusters as genres, and score items just like in the Genre baseline. It was observed that refined clustering improves results, so we have partitioned the data into 1,000 clusters when using this baseline.

[4] http://kddcup.yahoo.com

Dataset	#playlists	#items	#plays(train)	#plays(test)	#non-repeat-plays(test)
IRDB	3,535	167,695	5,955,208	1,983,049	247,109
StrongGen1	2,649	167,185	5,967,980	N/A	N/A
StrongGen2	886	140,971	1,478,073	492,204	59,069
Yahoo! Music	203,956	41,727	18,783,943	4,596,710	4,596,710

Table 4: Summary statistics for Internet radio weak- and strong-generalization data, and for Yahoo! Music data.

Results.

We begin with reporting the weak generalization results. We experimented with the station-based and station-less models, described in Section 4. For both models we picked the dimensionality of the latent factor space to be 20. Changing the dimensionality to 10 or 50 had a very minor impact on the metrics reported here (skipped for brevity). The two models were compared against the three aforementioned baselines, as shown in Table 5.

Method	NonRepeat-vs-Uni	NonRepeat-vs-Pop	Played-vs-Pop
Popularity	65.12%	39.91%	50.26%
Genre	78.41%	73.89%	79.33%
k-Means	74.61%	72.83%	83.80%
station-based model	90.91%	87.38%	95.72%
station-less model	**91.41%**	**88.66%**	**95.95%**

Table 5: Comparative model accuracy under a weak-generalization test.

Among our two models, it seems that the station-less model offers a slightly better accuracy than the station-based model. However, recall that its training is slower. A single iteration of the station-based model takes around 12 minutes, whereas a station-less model iteration takes about 20 minutes; all measured on an Intel Xeon X5650 2.67GHz CPU machine. As expected, comparing newly played items against popular items (NonRepeat-vs-Pop) is the most difficult task, where our models could achieve about 88% accuracy. Under all three metrics the models compare very favorably against the baselines. The NonRepeat-vs-Uni metric demonstrates one advantage a CF method has in comparison with popularity-oblivious methods such as Genre and k-Means reported in the table. While the CF models reach about 91% accuracy on such a metric, Genre and k-Means cannot utilize this source of information achieving significantly lower accuracies (75-78%). Considering the simplistic popularity-based method, as expected it behaves no better than random on NonRepeat-vs-Pop and Played-vs-Pop metrics, which remove popularity effects. On the other hand, even when popularity effects are in-place, namely under the NonRepeat-vs-Uni metric, Popularity merely achieves a 65.12% accuracy, indicating that adequate playlist modeling has to go much beyond offering popular items.

Moving on to the strong generalization tests described in Table 6, results pretty much remain the same. We are encouraged by the generalization ability of the models that enables modeling unseen stations with about same accuracy achieved on pre-learned stations.

Method	NonRepeat-vs-Uni	NonRepeat-vs-Pop	Played-vs-Pop
Popularity	65.24%	34.59%	49.94%
Genre	78.70%	74.86%	80.21%
k-Means	74.37%	72.83%	83.99%
station-based model	**90.38%**	86.96%	91.90%
station-less model	90.23%	**87.98%**	**92.56%**

Table 6: Comparative model accuracy under a strong-generalization test.

We also explored the way prediction accuracy increases with the number of observed plays. Here we take the model trained on the StrongGen1 stations, and gradually apply it to more and more plays of the StrongGen2 stations. We let the model observe just the last K train plays of each station, with K ranging between 10 and 250. We are tracking the moving performance of our accuracy metrics on the StrongGen2 test set; see Figure 2. We provide the results of the station-based and the station-less models, along with those of the better baseline, Genre. As expected accuracy of all metrics improves with number of observations, stabilizing at between 100-150 observed plays. Note that NonRepeat-vs-Uni is high (except for Genre) even with very few observations, as this metric benefits from popularity effects which do not require personalization.

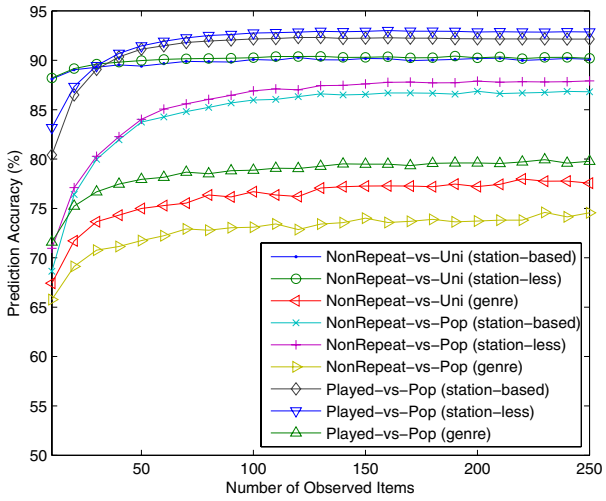

Figure 2: Strong generalization accuracy as a function of the number of observed plays per station.

Finally, we report the results of the cross-source learning test, where we predict behavior of Yahoo! Music users based on a model learned on IRDB. In order to establish a strong yardstick, we trained the station-based model directly on YMDB itself. The reader should note that at YMDB there are no repeating items for the same user, hence we dropped the Played-vs-Pop metric which became identical to the NonRepeat-vs-Pop metric. The results are described in Table 7. It seems that popularity effects are very strong at YMDB, as the naive Popularity method, which was allowed to observe YMDB's popularity distribution, achieved a 93.18% accuracy on the NonRepeat-vs-Uni metric. This also shows the disadvantage of content-based approaches like Genre, which completely miss the big accuracy gains that could be made by differentiating popular from less popular tracks. On the other hand, when removing the popularity effects, by using the NonRepeat-vs-Pop metric, YMDB is found to be inherently more difficult than IRDB, as reflected by the worsened performance of the station-based model even when trained on YMDB itself.

As expected, IRDB-trained models could not match the accuracy of a YMDB-trained model on the YMDB test set. Yet, our station-based model could achieve, on both metrics, accuracies that are within 5-6% of the YMDB-trained model. We argue that considering the significant disparity between the two datasets (over 6 years mean time distance, and rating-orientation vs. playing-orientation) achieving such a small accuracy difference is quite impressive.

Method	NonRepeat-vs-Uni	NonRepeat-vs-Pop
Trained-on-YMDB	*96.33%*	*82.85%*
Popularity-on-YMDB	*93.18%*	*46.95%*
Genre	78.22%	75.19%
k-Means	63.94%	59.99%
station-based model	91.25%	77.14%
station-less model	90.79%	75.13%

Table 7: Cross-source learning: evaluating IRDB-trained models on YMDB. The Trained-on-YMDB and Popularity results were learned on YMDB and are brought for comparison.

Artist and genre prediction.

We also evaluated the model success in linking tracks belonging to the same artist or genre. IRDB contains 113,103 tracks, each associated with an artist. For each track we compute the most similar track based on cosine similarity of their respective factor vectors, thereby generating 113,103 closest track pairs. Then we report the fraction of track pairs that share the same artist. We repeat the same experiment with genres. Here we consider only the aforementioned 45,713 tracks which we could associate with genres. We report the fraction of pairs that share a genre. Besides using a 20-D model we also report results of 50-D and 100-D models, as dimensionality somewhat matters here. As a reference, we also report results by a random matching between items. All results are reported in Table 8.

Before we discuss the results we emphasize the following. Since our model explicitly accounts for the given track-artist relations, we have to re-train the model without exploiting this information, in order to justify the tests reported here. Hence, we re-trained the station-based model while replacing Equations (1)-(2) with $q_i \stackrel{\text{def}}{=} p_i$ and $b_i \stackrel{\text{def}}{=} c_i$.

Method	%same genre	%same artist
random	12.48%	0.02%
station-based 20-D	63.42%	8.16%
station-based 50-D	**63.92%**	8.81%
station-based 100-D	63.79%	**8.89%**

Table 8: Fraction of nearest item pairs that share the same genre or the same artist. Results of random pairing are brought as a reference.

First, we analyze the artist-linking results. When randomly forming track-pairs, 0.02% of the pairs contain two tracks of the same artist. A quite negligible fraction, as expected. However, when picking pairs based on cosine similarity of the model's factor vector, the fraction of shared artist pairs jumps over 400 times, reaching best result with a 100-D model where 8.89% of the pairs share an artist. This demonstrates the ability of the model to uncover true musical properties.

We also analyze the genre-linking results. Interestingly, even with a random match, 12.48% of track pairs share at least one genre. This is attributed to the fact that the leading genres are extremely popular, with each covering a significant fraction of the tracks. Still, when picking neighboring tracks based on our model, the amount of pairs sharing a genre jumps more than 5-fold up to 63.92%. Note that it would be difficult to reach significantly better results, as some human based genre assignments are quite subtle and not well defined (e.g., compare the genre "Film Scores" with "Shows & Movies", or "Indie Rock" with "Alternative Rock"). In this sense, artist linking is a more natural task than genre linking.

5.2 Qualitative analysis

When examining our model, we have found that very often it captures the true nature of music. We would like to give a taste of our qualitative experience with the model, by providing a few results we found to exhibit particularly interesting intuitions.

The model was able to identify, using cosine similarity, individual band members as related to the band itself. In Table 9, we show that artists found most similar to "The Beatles" are the band members themselves. Notably, all similarity values are above 0.9, corresponding to the top 0.2%-quantile of artist pairs.

Rank	Artist	Similarity Score
1	Paul McCartney	0.927
2	John Lennon	0.925
3	George Harrison	0.920
4	Ringo Starr	0.916

Table 9: Artists most similar to "The Beatles".

Another good example is "Crosby, Stills and Nash". Although the band changed its name, upon addition of a new member (Young), to "Crosby, Stills, Nash and Young" the model finds similarity at as high as 0.906 between the new name and the former one.

An especially interesting observation surfaced by the model was finding Kylie Minogue as the one most similar to Madonna. Indeed, it is a known fact that these two artists were in great competition for a very similar audience in the past. However, looking only at user feedback/playlists from recent years, it will be a very hard insight to gain, as in this period Kylie Minogue was not very active in her musical career and therefore may not receive a strong enough signal. Radio stations however, have the world knowledge which can bind music across time and genres, and by incorporating this into our model we are able to offer recommendations with the human touch otherwise hard to gain.

To investigate the intra-artist understanding of the model, for each artist we have calculated the standard deviation of factor vectors for corresponding tracks. For this exercise we used an aforementioned version of the model where artist information is omitted from the tracks vector. We filtered out artists with less than 15 tracks and less than 1000 plays. As shown in Table 10 "The Beatles" are associated with the least standard deviation. From a pure content analysis viewpoint, it may be considerably hard to find the proper association between the different styles within the discography of this band. However, our model captures the fact that in a sense "The Beatles" define a style of their own, which transcends the mere combination of instruments and harmony. At the other end of the spectrum we find "Celine Dion". This may be explained by the fact that as a bilingual singer she has many tracks that are mostly appreciated only by her French speaking audience and are totally obscure to her other fans. Another reason that may account for her "diversity" are her several "soundtrack super hits" like the theme song of Titanic attracting a much wider crowd than the rest of her English repertoire.

Artist	Standard Deviation
The Beatles	1.942595597
Tina Turner	1.979493839
Pet Shop Boys	2.019336532
...	
Celine Dion	11.19272216

Table 10: Artists sorted by the standard deviation of their songs factor vectors (including only artists with at least 15 tracks and 1000 plays).

6. CONCLUDING REMARKS

By leveraging the wisdom of crowds, collaborative filtering (CF) brings important benefits to recommendation technology. However, the flip side of getting this wisdom is the need to get a reasonably sized crowd providing it. A major limitation of CF is its

need of elaborate user activity on each considered item. This poses a challenge to new service providers, which do not have enough information in order to facilitate building an appealing collaborative filtering system. In fact, we observe an effect of "rich get richer", where large media providers could build well tuned CF systems leveraging their vast number of users, and by this way maintain a quality advantage.

We try to break this "rich get richer" cycle in music recommendations, by tapping into the publicly available wisdom of the Internet radio editors. This would enable interested parties such as small ventures and research groups build quality CF systems. In a similar manner this concept benefits more mature providers facing the cold start problem related to items new to the system. By tracking a huge number of human crafted playlists, we model the ingredients of a successful playlist. The proposed probabilistic model is carefully designed for the needs of music recommendation, by accounting for the typically unary user feedback such systems observe, together with temporal effects specific to music playing and artist-track linkage. The model enables synthesizing new playlists that will appeal to the unique tastes of individual users. This allows young market players create collaborative filtering music recommendation at low initial cost.

Another direct application of this work is a station recommendation system, which will suggest the Internet radio station most suitable to the user's taste. Additional applications are building general music database, and music trend detection systems along the lines of Tables 1-2.

Our model is based on the wisdom of (many) experts, instead of the wisdom of crowds typical to CF systems. This raises several interesting questions that deserve further exploration: can a diverse group of experts match (or exceed) the wisdom of the crowd? What distinguishes experts-generated data from crowd-sourced data? (e.g., can we expect a longer memory and less bias towards commercially promoted items?) How reliance on identified experts can be exploited for better explaining the reasoning behind specific recommendations to the end-user?

7. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[2] X. Amatriain, J. Bonada, Àlex Loscos, J. L. Arcos, and V. Verfaille. Content-based transformations. *Journal of New Music Research*, 32:2003, 2003.

[3] Y. Bengio and J.-S. Senécal. Quick training of probabilistic neural nets by sampling. In *Proc. 9th International Workshop on Artificial Intelligence and Statistics (AISTATS'03)*, 2003.

[4] M. Castelluccio. The music genome project. *Strategic Finance*, 88:57–58, 2006.

[5] B. Fields, C. Rhodes, and M. d'Inverno. Using song social tags and topic models to describe and compare playlists. Barcelona, September 2010. ACM, ACM.

[6] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[8] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142, 2010.

[9] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[10] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[11] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[12] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[13] P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[14] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *ICML*, page 78, 2009.

[15] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *IEEE Trans. on Audio, Speech and Language Processing*, 18(2):407–412, 2010.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[17] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, 2010.

[18] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[19] R. Ragno, C. J. C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *Proc. 7th ACM SIGMM int. workshop on Multimedia information retrieval*, pages 73–80, 2005.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, 2009.

[21] S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736, 2009.

[22] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[23] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.

[24] R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

[25] M. Schedl and P. Knees. Context-based Music Similarity Estimation. In *Proc. 3rd International Workshop on Learning the Semantics of Audio Signals (LSAS 2009)*, 2009.

[26] M. Slaney. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, 18(2):12–15, 2011.

[27] Y. Zhang, B. Cao, and D.-Y. Yeung. Multi-domain collaborative filtering. In *Proc26th Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 725–732, 2010.

[28] E. Zheleva, J. Guiver, E. Mendes Rodrigues, and N. Milić-Frayling. Statistical models of music-listening sessions in social media. In *Proc. 19th int. conference on World wide web*, WWW '10, pages 1019–1028, 2010.