MEUSE: RECOMMENDING INTERNET RADIO STATIONS

Maurice Grant
Ithaca College
mgrant1@ithaca.edu

Adeesha Ekanayake
Ithaca College
aekanay1@ithaca.edu

Douglas Turnbull
Ithaca College
dturnbull@ithaca.edu

ABSTRACT

In this paper, we describe a novel Internet radio recommendation system called MeUse. We use the Shoutcast API to collect historical data about the artists that are played on a large set of Internet radio stations. This data is used to populate an *artist-station* index that is similar to the term-document matrix of a traditional text-based information retrieval system. When a user wants to find stations for a given seed artist, we check the index to determine a set of stations that are either currently playing or have recently played that artist. These stations are grouped into three clusters and one representative station is selected from each cluster. This promotes diversity among the stations that are returned to the user. In addition, we provide additional information such as relevant tags (e.g., genres, emotions) and similar artists to give the user more contextual information about the recommended stations. Finally, we describe a web-based user interface that provides an interactive experience that is more like a personalized Internet radio player (e.g., Pandora) and less like a search engine for Internet radio stations (e.g., Shoutcast). A small-scale user study suggests that the majority of users enjoyed using MeUse but that providing additional contextual information may be needed to help with recommendation transparency.

1. INTRODUCTION

Prior to the advent of personal computers and the Internet, there were two primary music recommendation technologies: the jukebox and the AM/FM radio. A Jukebox was a common feature of many social spaces such as bars and diners. Using a jukebox, an individual could chose from a small set of *on-demand* songs to play for the rest of the people in the nearby vicinity. AM/FM radios were found in more personal spaces such as the home or car. However, listeners were connected to one another through a common stream of music that was *broadcast* over the air waves. This gave popular, trend-setting DJs the opportunity to be heard by millions of listeners at the same time.

Today, we see a number of new music recommendation technologies emerging as a result of the availability of personal computers, mobile devices and the Internet. We can generally classify them as being similar to a jukebox, AM/FM radio or a hybrid of the two (see table 1). Celestial jukeboxes [2] such as Apple iTunes and Spotify provide users with instant on-demand access to millions of songs at the click of a button. Internet radio allows computer users to listen to broadcasts of songs or programs from traditional radio stations like NPR and BBC or through radio aggregators such as Shoutcast and AccuRadio.

The hybrid of these two digital technologies is *personalized* Internet radio which allows users to listen to a personalized stream of music based seed artists or semantic tags (e.g., genres, emotions). Popular examples include Pandora and Slacker. These systems are like jukeboxes in that a user has some control over which songs are selected, but like radio in that there is some element of serendipity in that the user cannot predict exactly what will be played ahead of time. In this paper, we describe a system called MeUse which attempts to harness the strengths of Internet radio but functions more like personalized Internet radio.

The core of our system relies on data collected using the Shoutcast API [1] . Shoutcast is a music service that aggregates 50,000 Internet radio stations, many of which are curated by human DJs. They provide a simple user interface that allows users to search by artist or genre. This search returns a list of stations that are currently playing the artist or genre of music that a user is seeking. They also provide metadata for each station which includes the station's web address, the number of current listeners, bit rate and stream format. Using the web address, users can launch a 3rd party media player (e.g., Apple iTunes, Winamp, Windows Media Player, VLC) to connect to the radio stream.

One problem with Shoutcast is that they may not recommend any stations for a given seed artist if no station is currently playing that artist. They may also return too many stations if the seed artist is currently very popular (e.g., a search for Rihanna might return over 100 stations.) This causes the *paradox of choice* in which increasing consumer choices can also increase the chance of user dissatisfaction [8]. A user may also not know which station to choose based on the limited metadata that is provided for each radio station. The ability to search by genre is also limited because radio stations are only allowed to have one genre (even though stations often play music from a diverse set of genres) and the genre label may be vague, rarely updated, or simply inaccurate.

[1] http://wiki.winamp.com/wiki/SHOUTcast_Radio_Directory_API

Technology	Discovery Mode	Delivery Medium	Examples	Advantages
Internet Radio (analogous to AM/FM radio)	Passive	Broadcast	NPR, BBC, Shoutcast, AccuRadio	Often Curated by Human DJs, Ease of Use, Serendipity
Personalized Internet Radio	Passive	Automatic Playlist	Pandora, Slacker, Jamendo	Personalization, Ease of use, Serendipity
Celestial Jukebox (analogous to a physical jukebox)	Active	Metadata Search	iTunes, Spotify, Tomahawk	Customized Playlists, Immediate Gratification, Personalization

Table 1. Comparison of Music Recommendation Technologies

From an interactive standpoint, another problem with Shoutcast is that it forces the user to use a web browser for recommendation and a 3rd party audio player for listening. That is, it takes a relatively long time before a user can start listening to music. By comparison, most personalized Internet radio players embed the audio player directly in the webpage and require only a small of amount of interaction with the user before the music starts playing. In the case of Pandora, music will start playing immediately when a user (who has previously logged in) returns to the site without the single press of a button. Also, using Shoutcast makes it difficult to change stations since a user has to do another search using the website and then load the newly selected station into the 3rd party media player.

In this paper, we are interested in improving Internet radio station recommendation both in terms of backend recommendation algorithms as well as developing a better frontend user experience. Specifically, we use the Shoutcast API to collect historical data about the artists that each station plays over time. We use this data to populate an *artist-station* index that is similar to the term-document matrix of a traditional text-based information retrieval system but where the documents are stations and the terms are artists. When a user wants to find stations for a given seed artist, we check the music index to determine a set of stations that are either currently playing or have recently played that artist. These stations are grouped into three clusters and one representative station is selected from each cluster. Clustering is intended to promote diversity among the stations that are returned to the user. In addition, we provide additional information such as relevant tags and commonly played artists to give the user more contextual information about the recommended stations. Finally, we incorporate an embedded audio player directly into our website so that users do not need to use an external 3rd party media player.

2. RELATED WORK

While we have been unable to find related work on the specific task of Internet radio station recommendation, our work is functionally equivalent to a generic text-based search engine in that the main data structure is a inverted index and we rely on a vector space model to access relevance and cluster our data [4]. That is, each radio station is represented as a vector over a vocabulary of artists. When given a seed artist, we can rank stations by the dimension corresponding to that artist. We are also able to cluster stations once they are each represented as a vector.

There has been considerably more work on using historical playlists from Internet radio stations as data for studying music similarity and automatic playlist algorithms [1, 3, 5, 6]. Although it is not the focus of this work, our artist-station index can also be used to calculate artist similarity if we instead think each vector corresponding to an artist over a vocabulary of stations.

3. SYSTEM OVERVIEW

When designing MeUse for Internet radio recommendation, we focus on three information retrieval concepts: *Relevance, Diversity and Transparency.* In this sections we describe both the backend recommendation algorithm and frontend user interface in terms of these important design concepts.

3.1 Backend Recommendation Architecture

In this subsection, we discuss how we collect data, use this data to find a set of relevant stations, clustering these stations, and then select stations that are recommended to the user. An overview of our system architecture is shown in figure 1.

3.1.1 Data Collection

The set of artists in the database was initialized by downloading the top 100 tags on Last.fm, and then downloading the top 100 artists for each tag. This provided us with a diverse set of 4460 artists. The set of tags was initialized by downloading the set of tags for all of these artists. Once downloaded, the set of tags was pruned by first removing weak song-tag associations (e.g., a Last.fm tag score < 5) and then making sure that each tag was associated with a minimum of 5 artists.

We then grow a set of Internet radio stations by querying the Shoutcast API multiple times for each of our artists. That is, Shoutcast returns a set of stations that are currently playing a given seed artist. For each of these stations, we increment a counter (e.g., a cell in our artist-station index

matrix) each time we see that the stations is playing the artist. If we have never observed the station in the past, it is added to our set of stations.

In addition, whenever a user searches for an artist using our frontend user interface, we use the given artist as a query to the Shoutcast API. The results are used to further grow the set of artists, the set of stations, and increment the values in the artist-station index. Finally, we apply a daily decay to the values in the artist-station index so that newer observation have more weight than older observations.

3.1.2 Relevance

Once we had gathered the above data, we used the following algorithm to recommend stations for a given seed artist. To begin, we selected a set of between 10 and 30 *candidate* stations. These stations have played the seed artist the highest number of times in the past or are currently playing the seed artist according to Shoutcast. If we have too few candidate stations, we find the top ranked similar artist to the seed artist according to Last.fm and use that artist to find additional candidate stations. If we have too many candidate stations, we rank order stations by the number of times they have played the seed artist but also taking into account observation decay as described above.

3.1.3 Diversity

Next, we use the information that is stored in our artist-station index to represent each candidate stations as a vector over our vocabulary of artists (e.g., columns of the artist-station matrix). These vectors are grouped into three clusters using the k-mean algorithm [7]. We then selected a representative station from each cluster by selecting the station with the highest listen count while giving stations currently playing the seed artist priority. This ensures that the station is not only relevant but also important.

3.1.4 Transparency

To make our station recommendation more transparent, we provide three representative artists and three representative tags for each of the recommended station. To select representative artists for a given station, we ranked artists according to the difference between the play count on that station and the sum of the play counts for that artist on the other two stations. By this method, we select representative artists that *differentiate* the recommended stations from one another.

The three representative tags for a recommended station are found by first finding all of the tags for all of the artists that are played on that station and then removing the tags that are also associated with artists who have been played on the other two recommended stations. The remaining tags are then rank ordered by the average Last.fm artist-tag score for all artists associated with the station. Again, this algorithm has been designed to pick tags that differentiate the three recommended stations rather than select the most representative tags for the station.

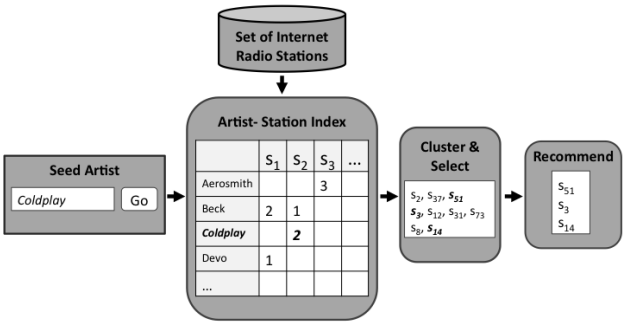

Figure 1. Backend System Architecture. A *artist-station index* is created by counting the number of times an artist is played on a station. When a user provides seed artist (e.g., Coldplay), the index is used to find a set of relevant station. These stations are clustered and a representative station is selected from each cluster.

3.2 Frontend User Interface

The web-based user interface for MeUse is shown in figure 2. We wanted the interactive experience to be more like a personalized Internet radio player (e.g., Pandora) and less like a search engine for Internet radio stations (e.g., Shoutcast). This is accomplished in a few ways.

First, after a user has entered a seed artist in the search bar, only three stations are recommended to the users based on clustering and station selection as is described in the preceding subsection. We provide a clear and concise snippet of information for each station. This includes the name of the station and other important metadata that is provided directly from Shoutcast (e.g., current number of listeners, bit rate, audio format). In addition, we provide the lists of representative artists and tags that differentiate the recommend stations.

We also provide an embedded VLC audio player that is hidden from the user but can be manipulated by the user through various control mechanisms found on the page (e.g., play/pause, volume). In addition, the user can easily switch between the recommended stations, request new recommendations, or change the seed artists. While the VLC plugin is useful for removing the need for an external 3rd party player (e.g., iTunes, Winamp), it does require the user to have the (free) pluggin be installed for their browser. In the future, we expect that web technologies (e.g., HTML5) will allow for more seamless streaming of Internet radio directly through the web browser.

4. EVALUATION

To evaluate MeUse, we first explore how well our algorithm is able to recommend a diverse set of Internet radio stations. We then describe a small-scale user study that was primarily directed at evaluating our user interfaces but also allows us to ask questions about our backend recommendation system.

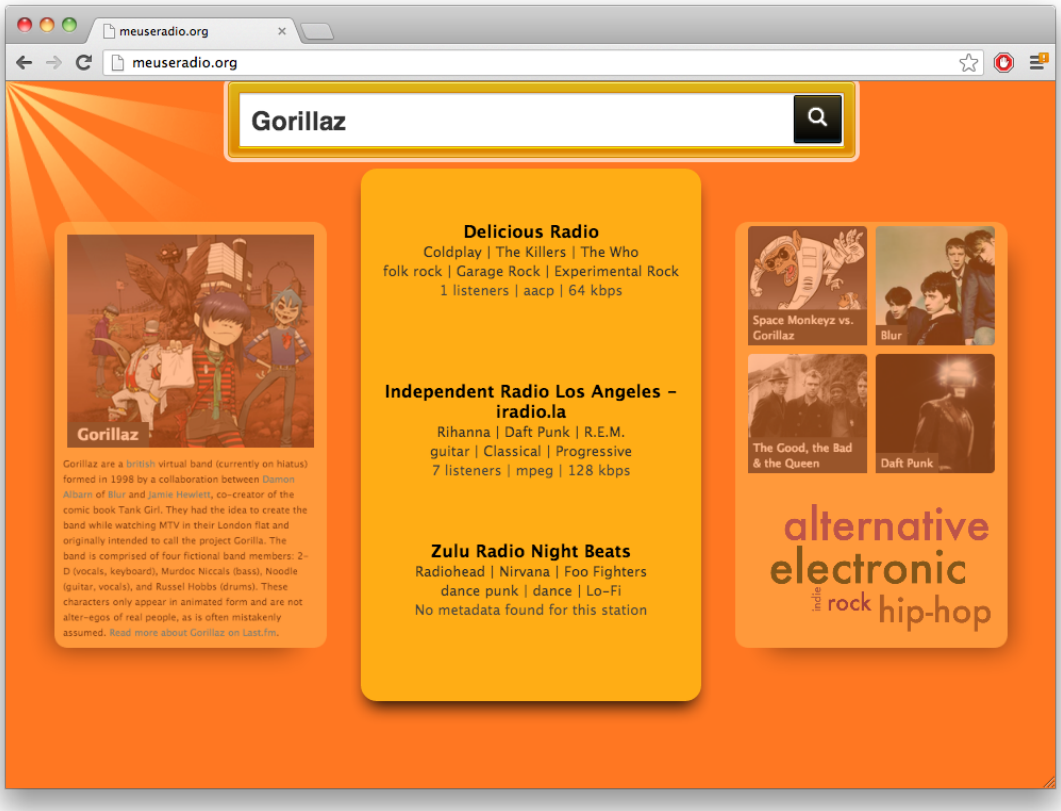

Figure 2. MeUse User Interface

4.1 Exploring Diversity

One of the primary goals of MeUse is to dramatically limit the number of recommended stations while providing a diverse set of relevant stations to match the user's interests. We also want to provide users with contextual information, such as representative tags and artists, so that the user can make an informed decision when choosing between the recommended stations. To evaluate this, we designed an experiment that compares how often a representative artist for a station is played on that station verses how often the representative artist is played on one of the other recommended stations.

For the experiment, we randomly selected 100 artists from the set of 500 most popular artists (according to Last.fm) in our database. For each artist, we obtained three recommended stations using MeUse. We then *listened* to each of these three stations for the next two hours by recording the currently playing artist every 10 minutes. Finally, we counted how often the seed artist was played, how often one of the three representative artists for the station was played, and how often one of the six representative artists from the other two recommended stations was played.

The results for our experiment are shown in table 2. While we should have collected on the order of 3600 song-play observations (e.g., 100 artists, 3 stations, 12 observations), we found data collection to be a more noisy process than expected. That is, some stations did not appear to update their "recently playing" information and other stations' "recently playing" information contained information about the station and not about the music currently being played. In both these cases, we ignored the duplicated "recently playing" information. We also found a few cases where a station stopped broadcasting during our two hour observation window which prevented us from collecting some additional song-play observation. In the end, we were able to collect 2627 observations.

	# of Artists	Play Count	$\frac{\text{Play Count}}{\text{\# of Artists}}$
Seed Artist	1	147	147
Representative Artists for Station	3	25	8.3
Representative Artists for Other Recommended Stations	6	30	5

Table 2. Results from diversity experiment after recording 2627 song-play observations.

The results show that stations play the seed artists 5.6% of the time. We also observe that the stations play each of our representative artists 0.32% of the time. While this appears to be rather low, our goal in picking representative artists is to pick artists that differentiate the station from the other two recommended stations rather than simply pick-

ing popular artists that have been played on the station in the past. We also note that this is higher than the 0.19% of times that each of the representative artists from the other two recommended stations are played on the station. However, this is not a statistically significant improvement ($\alpha = 0.18$, one-tailed two-proportion pooled z-test). We suspect that the ability to find better representative artists will improve as we are able to collect more data to populate the artist-song index. [2]

4.2 User Study

To evaluate the usability of MeUse we conducted a small-scale user study of our interface. The study involved 20 college-aged individuals who were asked to play around with the interface and then fill out a short survey about their experience. About half of the test subjects were observed in our lab while using the system. The other half were asked to use MeUse in their own environment. Of the ones that we observed, all users seemed to find MeUse easy-to-use, were quickly able to listen to music through the embedded player, and seem to enjoy switching between the recommended stations.

No	Not Really	Sort of	Mostly	Definitely
0	3	3	9	4

Table 3. Relevance: Were the recommend stations that were relevant to you?

No	Not Really	Sort of	Mostly	Definitely
2	3	5	7	2

Table 4. Transparency: Did we give you enough information to make a clear choice between the 3 stations we recommended?

In terms of our ability to recommend Internet radio stations, 70% of the test subjects stated that the recommended stations were mostly or definitely relevant (see table 3) but only 45% of users felt that they were given enough information to make an informed decision on which of the three stations to choose (see table 4). This suggest that we need to think about additional ways to provide the user with contextual information about the stations. For example, one test subject suggested adding a point-rating system for stations. The test subjects did indicate that the stations that were recommended were diverse in nature (see table 5) and that, in general, they enjoyed using MeUse to listen to Internet radio (see table 6).

5. DISCUSSION

In this paper we described MeUse as a complete Internet radio recommendation system. The results of our small-scale user study suggests that the system shows promise but additional user testing is required. In particular we planned to do extensive A/B testing to isolate specific aspects of our system (e.g. UI design, recommendation algorithm). This will include both observing a small number of users in our lab as well as large-scale and long-term user studies in natural user environments.

We also would like to further develop MeUse by exploring additional ways in which we can make MeUse more like a personalized Internet radio player. This will include allowing users to be able to rate stations and using collaborative filtering to improve our station recommendation algorithm. Finally, we plan to explore modifying our artist-tag index to benefit from common text retrieval techniques (e.g. tf-idf) to further improve recommendations.

[2] At the time of submission, we have only been able to search the Shoutcast API approximately 5 times for each of the ∼4,500 artists in our database. This is because Shoutcast limits the number of query's one can make on a daily basis.

No	Not Really	Sort of	Mostly	Definitely
1	1	3	11	3

Table 5. Diversity: Were the three stations we recommended different enough from each other to make selecting a station meaningful?

No	Not Really	Sort of	Mostly	Definitely
2	1	2	6	8

Table 6. Overall: Did you enjoy using MeUse to listen to Internet radio?

6. ACKNOWLDEGEMENTS

Steven Lam help implement MeUse. This research was supported in part by NSF Awards IIS-1217485.

7. REFERENCES

[1] N. Aizenberg, Y. Koren, and O. Somekh. Build your own music recommender by modeling internet radio streams. In *WWW*. WWW, 2012.

[2] P. Lamere and J. Donaldson. Tutorial on using visualization for music discovery. In *ISMIR*. ISMIR, 2009.

[3] F. Maillet, D. Eck, G. Desjardins, and P. Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR*. ISMIR, 2009.

[4] C.D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[5] B. McFee and G. R. G. Lanckriet. The natural language of playlists. In *ISMIR*. ISMIR, 2011.

[6] J. Moore, S. Chen, T. Joachims, and D. Turnbull. Learning to embed songs and tags for playlist prediction. In *ISMIR*, 2012.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] B. Schwartz. *The paradox of choice*. HarperCollins e-books, 2009.