Radialize: A Tool for Social Listening Experience on the Web Based on Radio Station Programs

Álvaro R. Pereira Jr. Diego Dutra Milton Stiilpen Jr. Alex Amorim Dutra
Felipe Martins Melo Paulo H. C. Mendonça Ângelo Magno de Jesus
Kledilson Ferreira

Idealize Lab, Department of Computer Science
Federal University of Ouro Preto, Minas Gerais, Brazil
{alvaro, ddutra, mstiilpenj, alexamorim, felipe.melo, paulohcm, angelo, kledilson}@iceb.ufop.br

ABSTRACT

Radialize represents a service for listening to music and radio programs through the Web. The service allows the discovery of the content being played by radio stations on the Web, either by managing explicit information made available by those stations or by means of our technology for automatic recognition of audio content in a stream. Radialize then offers a service in which the user can search, be recommended, and provide feedback on artists and songs being played in traditional radio stations, either explicitly or implicitly, in order to compose an individual profile. The recommender system utilizes every user interaction as a data source, as well as the similarity abstraction extracted out of the radios' musical programs, making use of the wisdom of crowds implicitly present in the radio programs.

Categories and Subject Descriptors

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

Keywords

Radio; music; search; recommendation; signal processing; crawling; software architecture

1. INTRODUCTION

Radio is the traditional tool for social listening experience which has historically played a very important role in the society as a mass media. The first radio program was broadcast in 1920 in Detroit, Michigan, USA[1]. Radio stations soon became the main way to disseminate information at that age. Nowadays there are tens of thousands of radio stations around the world, being maintained mainly by advertising, playing an important role not only on people's lives but also on the economy.

Broadcasting services such as radio and television represent a limited class of social interaction called mass media on which a few individuals communicate unidirectionally to the mass. Receptor individuals are generally passive players, making it hard for a radio or TV station to analyze the profile of its audience. Another historical limitation of radio and television is the geographical barrier. Given that the transmission system is not able to propagate the signal throughout neighbor transmission points (antennas), receptors are limited to listen or watch stations that broadcast within their geographical range location. Nevertheless, although living in the same location, people have different preferences, so that they might like different artists, songs, talk shows, news, and advertising contents.

In this paper we present Radialize, a tool with the general purpose of allowing radio stations to, both, interact closely to their listeners, and to transmit to listeners independently of their locations. Thus, Radialize aims to be the "Google of the radios", as it is able to discover and index the content being played by radio stations transmitting through the Web, either by managing explicit information made available by radio stations or by means of our technology for automatic recognition of audio content.

Under the perspective of listeners, Radialize comes to redefine the way people listen to radio. It represents a disruptive technology which will deliver the second generation of radio listening experience. With Radialize users can rate or provide feedback on artists and songs being played in traditional radio stations, either explicitly or implicitly, in order to compose an individual profile. The service offers new items that might be of the user's interest, where the items may be radio stations, artists, songs, or other users with similar tastes. To deliver the best experience for users, Radialize makes use of the wisdom of the crowds by taking into account for its search and recommendation services the programs of the radio stations, which are actually the result of specialist disc-jockeys' action when creating playlists.

With Radialize being the tool, radio communities will no longer be geographically restricted. Though, there will be ways to facilitate radio-to-individual listeners interaction, and listener-to-listener interaction, if they are part of the same community (i.e., if they share tastes or if they are connected). Mass media has lost space for social networks, which is how people want to interact in the information age. People want to share with friends what they like. They want to play a role in the community they participate, being active. Furthermore, they want exclusivity. Radialize is the tool to make the transition of the radio from mass media to social network. Other audio streaming services like Pan-

[1] `http://en.wikipedia.org/wiki/Radio_stations`

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

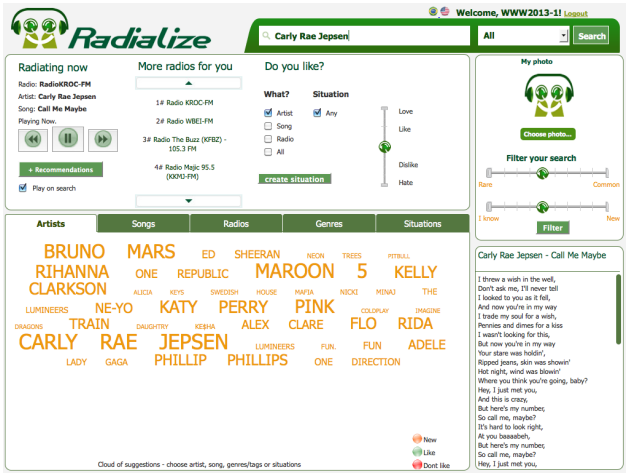

Figure 1: A snapshot of the Radialize front-end.

dora (`www.pandora.com`), Spotify (`www.spotify.com`) and Last.fm (`www.last.fm`) cannot play this role, because they are not adapting the almost 100-years-old media for social listening experience to our age's needs, rather, they built their services from scratch without relying on radio stations. Pandora, Spotify and Last.fm are about music playlist, whereas Radialize is about radio.

Radialize is implemented as a real-world tool that will be available soon for free use. The first version, which is the object of study in this paper, does not include any module to facilitate radio stations to communicate with their users. Rather, the first release is intended to gain the attention of users. In this demonstration paper we present the Radialize tool's description and architecture along with its main components and a demonstration of use.

2. DESCRIPTION OF THE TOOL

Radialize is a service for listening experience based on radio station programs. When compared to other solutions like Pandora and Last.fm, Radialize's main difference is on the possibility of returning third-party services like radio stations, web radios, and playlists. Figure 1 presents a snapshot of the Radialize's front-end.

Below we present the main functional requisites for the Radialize tool. The service must:

1. Allow users to listen to radio stations, connecting directly to the radio station server.
2. Allow users to search for artists, songs, radio stations, genres, and user names (friends).
3. Return as result of a query a list of radio stations associated with the query.
4. Provide recommendation of artists, songs, radio stations, genres, and users.
5. Generate clickable clouds of words when users are interacting, being the words artists, songs, radio stations, genres, or user names, all of them associated with the query or the recommendation (based on user profile).
6. When users click on an item of any cloud of words, the system responds on the same way as for a search for that item.
7. Allow users to explicitly rate artists and songs being played on a radio station. Allow users to rate radio stations.
8. Allow users to filter results for both recommendation and search results. Examples of filters are local/non-local, rare/popular, new-for-me/not-new.
9. Allow users to create situations, and to choose the situation(s) at any time. The concept of situation exists to let users make sub-profiles out of their general profiles.

Below we present the main non-functional requisites for the Radialize tool.

1. The system must provide recommendations based on the user's profile and based on the radios programs.
2. The Radialize's graph similarity is based on how items co-occur in radio stations. The more two songs co-occur in a given radio station program, and the more those two songs co-occur in different radio stations, the more similar the songs are.
3. The user profile consists of explicit and implicit ratings, where implicit rating means a set of actions allowed for users. For instance, a skip is a negative implicit feedback, whereas a volume increase is a positive implicit feedback.
4. Any radio station transmitting through the Web must be able to be included in the list of radios monitored by Radialize. It means that the service needs to detect the items being played in radio stations automatically, since a relevant fraction of radio stations do not inform what they are playing by metadata.
5. The result of the search must return in the top positions radio stations playing the item of the query, if there is any.
6. Radios that stop their transmission due to any reason (e.g. server break or maintenance) must also stop being returned as a result of users' interactions.

According to the requisites, we summarize the service offered by Radialize in the following way: Radialize maintains at real time information on what is being played in the set of radio stations it monitors. There are two ways of identifying the content played by a radio station: by means of the metadata informed by the radio station or by means of signal processing on the audio stream, which is very important to be performed because the number of radio stations that do not inform metadata is quite high[2]. The service then allows users to search and to be recommended of items, which may be artists, songs, radio stations, genres, and users. Every search or recommendation request returns not only a music experience (radio stations or playlist that start playing at query time) but also a list of items to be displayed on the associated cloud of words. Users can refine their search or recommendation requests by using filters.

[2]For instance in Brazil we have found only 30 radio stations that informed metadata, although 84% of the Brazilian radios transmit through the Web (http://goo.gl/DoOhc).

3. TECHNOLOGY BEHIND RADIALIZE

Radialize operates by relying upon a set of different technologies that allow for multimedia data crawling, indexing, searching, and recommendation.

Before adding a new radio to the set of radio stations covered by Radialize, it is necessary to discover the sources of audio and metadata streaming (when it exists) for that radio. This process is most of the times done automatically, though it may require manual intervention. As soon as those sources are found they are sent to one of the Radialize's crawlers to start being crawled.

Music content broadcast by a radio station is named a *media* in Radialize architecture vocabulary. A media is composed by the artist and song names. When a media is crawled it is necessary to find out if it is already known by Radialize database, so that its execution history can be promptly updated, otherwise the media is joined to the database.

In order to process queries Radialize needs to, both, use an Information Retrieval system [1] to cope with approximate queries, and keep track of real-time data about what is being played by each monitored radio station. When Radialize users search for a given subject they are not only interested in finding that subject, but also in being introduced to content related to that subject which they do not know yet but are likely to appreciate. Radialize relies upon a recommendation module to provide users with those recommendations. The recommendation module uses the radios' broadcast history to calculate the similarity between items (artists, songs, etc) and then produces recommendations by matching users' profiles to those items.

Figure 2 presents the Radialize back-end architecture. This architecture was designed to be easily scalable with hardware in order to support an increasing number of radios to be covered, while relying on commodity hardware. The database layer is accessed through a Facade [2] which hides the database manipulations implementation. It allows for changes in the storage layer going unnoticed by any other components of the back-end architecture.

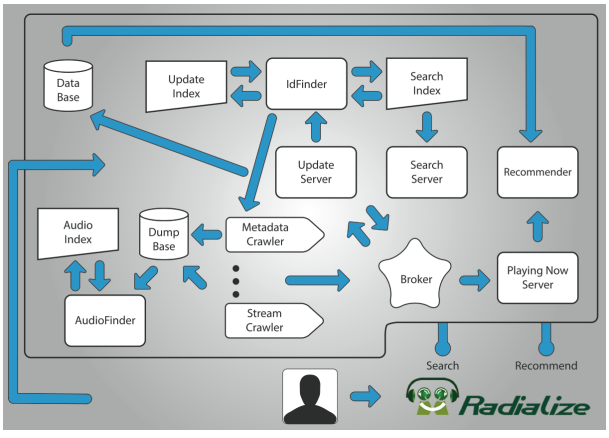

Figure 2: Radialize high-level back-end architecture.

The back-end operation can be summarized as follows:

1. There are two kinds of *Crawlers*: textitMetadata and *Stream* Crawlers. *Metadata Crawlers* extract radios' metadata from the Web, check in the IdFinder if a crawled media (artist/song) is known by the Radialize's database, and send the media to the *Broker* along with its respective identifier, in case the media is known.
2. *Stream Crawlers* extract radios' audio content from the Web, submit segments of the audio to the *AudioFinder* in order to check by signal processing if a set of segments represents a song occurrence. In case an occurrence is found, the Stream Crawler sends the media to the *Broker* along with its respective identifier.
3. The *Broker* forwards the data received from the *Crawlers* to the *PlayingNowServer* and to the *UpdateServer*.
4. The *PlayingNowServer* is responsible for holding data such as which artists and songs are being played at each radio station.
5. The *UpdateServer* is responsible for updating the indexes and the database with the new incoming data, may it be a new artist, a new song, or simply one more entry to be stored into the execution history. It uses the *IdFinder* and the *AudioFinder* for doing its job.
6. Radialize maintains two text indexes to support *IdFinder* requests: update index and search index. The update index is queried by the *UpdateServer* in order to check if an incoming media is unknown. The search index is responsible for serving users' queries. The decision of keeping two indexes aims at eliminating the competition between users' queries and the discovery module, since the latter needs to access the index to evaluate each incoming media. Both indexes are guaranteed to be always in the same state.
7. In case a media is found to be new, it is inserted into the database and the associated primary keys are retrieved and inserted into both indexes. The new media is then re-sent to the *Broker* in order to make the *PlayingNowServer* aware of it. Otherwise, the incoming media is simply written into the execution history. The MusicBrainz search server [4] is also used as source for finding medias, apart from the Radialize's database.
8. Radialize also maintains an audio index to support the job of the *AudioFinder* module. The audio index contains all the songs known by the Radialize database. New songs are indexed as soon as they are identified by metadata crawlers, and then receive identifiers from the UpdateServer. Note that labeled items (found by *Metadata Crawlers*) are used by the system to learn a content and then to allow content discovery in other radio stations by means of signal processing via *Stream Crawlers*.
9. When users perform a search or a recommendation request, their request gets at the back-end, which asks the *SearchServer* to identify the requested item. Next, the *Recommender* module is requested to provide recommendations related to that item. We used the IRF framework [3] to create our recommender system. After the recommendations are calculated, the *Recommender* sorts them according to which items are playing at the moment by using the services provided by the *PlayingNowServer*. If no recommended item is playing at the query time, the order initially set by the *Recommender* is kept.

Figure 3: Cloud of artists for the query "Psy".

Figure 4: Cloud of artists for the query "Gotye".

10. Recommended radios are played by connecting the Radialize player directly to the audio streaming server used by the radio.

Observe that the IdFinder uses the update index to find out if an incoming media is new, whereas the AudioFinder uses the audio index to accomplish the same task. When a media is found to be new, it arrives at the UpdateServer which inserts the media into the database and notifies the IdFinder, which indexes the new song into both, update and search indexes. Next the UpdateServer notifies the AudioFinder informing the physical location of the recorded stream, as well as the beginning and ending times of the media within the record. The AudioFinder then accesses the audio record and retrieves the segment contained within the informed time frame, indexing it as a song into the audio index.

4. USE CASE

The Radialize tool is available for beta users, which are allowed to create their own account at `www.radialize.com.br/register`. We have also created an account with the goal of presenting the use case in this section (`www.radialize.com.br` – username 'WWW2013-1', password 'WWW'). Readers may feel free to use the WWW account or to create their own accounts to experiment the tool.

In order to demonstrate an important feature of Radialize, we have performed the following case study: we searched in Google for "top ten songs of 2012" and chose two of the main results, which were a top ten list by MTV (`http://goo.gl/p5Vyw`) and a top ten list by Spotify (`http://goo.gl/Px8Fs`). We observed the artists/bands present in the lists, and chose three of then to be searched in Radialize. Then we compared the results returned by Radialize with the artists in the two lists. The artists/bands present in the top 10 lists are: Gotye, Fun, Carly Rae Jepsen, Maroon 5, One Direction, Flo Rida, M83, The Wanted, Nicki Minaj, Taylor Swift, Kanye West, and Psy.

Figure 1 in Section 2 presented the Radialize interface for the query "Carly Rae Jepsen", where the cloud of artists can be observed. Figures 3 and 4 present the cloud of artists respectively for the queries "Psy" and "Gotye". We observe that most of the artists present in the list above are considered by Radialize as similar with respect to each of the searched artist, which is an expected result since Radialize is capturing well the concept of similar artists based on the radio station programs. Actually, Radialize can be used to generate its own list of top ten artists and songs for a given period of time.

5. CONCLUSIONS AND FUTURE WORK

For almost 100 years radio has been a tool for mass communication. People from a given community listen to the same songs, talk shows, sport events, and advertising. The problem is that in the information age we live today the communities are no longer defined because of people's geographical locations like it had been for a relevant part of those almost 100 years of the presence of radio stations. Radialize aims at being the tool for radio stations to cross the barrier of the information age. In the future, Radialize will deliver to each user the best content, according with the user's taste. A given user may receive songs from a radio station in another country, talk shows from a radio station distant a thousand kilometers, and specific, directed, and local advertising.

This demonstration paper has presented the first release of Radialize, which incorporates a substantial set of functionalities which will start enabling Radialize as the tool for social listening experience. Radialize has been properly designed to be the disruptive innovation that will be responsible for the transition of radio stations as a kind of mass media to a kind of social network.

As future work there is a set of features to be included in Radialize. For instance, given that the taste of users is already being modeled, it is easy to offer a service for music playlists like Pandora and Last.fm. An example of a more sophisticated feature to be included soon is a component for speech recognition in radio programs. The idea is to allow users to search for words, and then return the radio whose words are being said on that radio. Because Radialize is about radio, which is used to listen to news and talk shows as well.

6. ACKNOWLEDGEMENTS

This work was funded by the AudiInfo Project–grant CNPq PDI 560285/2010-8, with all the support of DECOM/ UFOP and its R&D lab Idealize (`www.idealizelab.com.br`).

7. REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval – The Concepts and Technology behind Search*. Pearson, 2^{nd} edition, 2011.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[3] Felipe Martins Melo and Álvaro Pereira, Jr. A component-based open-source framework for general-purpose recommender systems. In *Proceedings of the 14th international ACM Sigsoft symposium on Component based software engineering*, CBSE'11, pages 67–72, New York, NY, USA, 2011. ACM.

[4] A. Swartz. MusicBrainz: A semantic web service. In *Intelligent Systems, IEEE*, pages 76–77, USA, 2002.