DEVELOPING A PROFESSIONAL-GRADE PERSONALIZED RADIO APP IN AN ACADEMIC SETTING

Douglas Turnbull[1] **Chris Perez**[2] **David Dorsey**[1] **Jon Burger**[1] **Joe Menduni**[1]
Daniel Akimchuk[1] **Elena Piech**[1] **Alex Python**[1] **Thorsten Joachims**[2]
[1] Department of Computer Science, Ithaca College, USA
[2] Department of Computer Science, Cornell University, USA
Contact: `dturnbull@ithaca.edu`

1. INTRODUCTION

MegsRadio is a personalized Internet radio service that allows users to create customizable streams of music based on seed artists, tracks, and tags. However, our radio player is different from commercial systems, like Pandora and Slacker, in four significant ways:

Local Focus: MegsRadio focuses on promoting music by local artists [1]. Our playlist algorithm is specifically designed to select a mixture of relevant songs by both well-known artists as well as artists who are located in and around Ithaca, NY, USA.

Hybrid Recommendation: We have developed powerful new playlist algorithms that combine information from a variety of data sources (e.g., web scraping, audio analysis, user listening habits). In addition, our playlist algorithm uses online learning to update our recommendation model in real-time based on user feedback (e.g, like/dislike, new seeds.)

Interactive Controls: MegsRadio gives users the power to control customizable streams of music with a number of novel interactive features. For example, a user can create and modify a station with multiple seeds where a *seed* might be an artist, a track, an event, a venue, a genre, an emotion, an instrument, or some other common text-based descriptor. Users can also control the mix of music by localness, popularity, and various acoustic properties (e.g., tempo, positiveness).

Real-time Playlist Research: We are also developing MegsRadio as a shared academic testbed so that other researchers can submit and evaluate their own playlist algorithms.

MegsRadio was officially launched in September 2016 with a jQuery-based web app and an iOS mobile app. In September 2017, we launched an Android mobile app as well as a new React/Redux web app having incorporated a year's worth of user feedback.

2. SYSTEM ARCHITECTURE

As is shown in Figure 1, we use standard model-view-control (MVC) architecture and maintain a RESTful API to support communication between our client apps and our main backend server. The overall system has a number of distinct pieces:

MegsRadio Apps - iOS app (Objective C), Android (Java) app, and web app (React/Redux).

MegsAdmin - dashboard and administration tool for cleaning metadata, setting featured stations, running backend scripts, testing playlist algorithms, and uploading of new music.

Controller - the main web service in Play/Java that communicates with client apps, handles all user interaction, generates playlists, and imports new music.

Scrapers - scrapes information from various APIs (e.g., Spotify, Last.fm) and websites (e.g., Local Events Websites).

Database - a relational database is used to store all user and music information.

File Storage - stores all static files (album cover artwork, audio files) that are used by the client apps and MegsAnalysis.

MegsAnalysis - a stand-alone web service in Flask/Python that uses signal processing (LibROSA) and machine learning (scikit-learn) for content-based audio features (e.g., tempo, energy), automatic tag generation, and music-similarity calculations.

Much of our software development work has focused on making this complex system robust and automatic. For example, when a new track is added, we resolve and clean the track's metadata, scrape information about the artist and

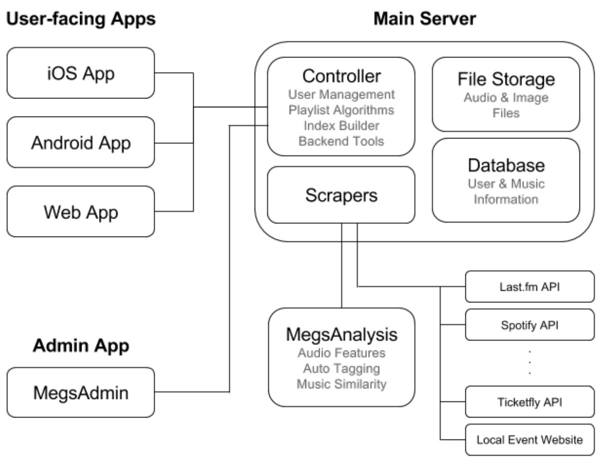

Figure 1. MegsRadio System Architecture

track from various data sources (Spotify, Last.fm, TicketyFly, MegsAnalysis), and set them to be *active* if they are connected to enough other artists and tracks already in our database. This final step is important so as to prevent artists, tracks, and tags from being used as seeds to start new stations which would not have enough relevant music to make the station sufficiently diverse or legal [1] .

3. UNDERGRADUATE DEVELOPMENT

What makes MegsRadio unique is that it has been built almost entirely by undergraduate students over the past seven years. This includes UI/UX design, software development, user testing and outreach efforts. Most of our progress on software development is made over the summer when a team of 6-8 students works on a full-time basis. During the academic year, we tend to focus on outreach and user feedback. Specifically, our outreach team maintains a local music blog, works with venue owners and festival organizers on cross-promotion (e.g., ticket giveaways, tabeling at events), and meets with artists and active users to elicit feedback and generate ideas for the following summer. Throughout the year, the tech team works to maintain our staging and production environments as well as addressing technical issues that arise (e.g., bugs, feature requests, user management).

4. CHALLENGES

While we have made good progress towards our goal of building a successful personalized radio player, we face the challenge of competing with commercial stream music services like Pandora, Spotify, Apple Music, and Google Music. These companies typically operate with multi-million dollar budgets and employ thousands of individuals. This creates many challenges (and a few advantages) for our small academic project:

Music Corpus: Over time, we have built up a corpus of about 50,000 music tracks by 15,000 artists. This is small when compared with commercial music databases which contain tens of millions of tracks. However, our corpus consists of a relatively large amount of (donated) music from local artists.

Multiple Apps: In order to match the offerings of a commercial service, it is important to maintain multiple client apps (iOS, Android, Web). This means developing and updating three different code repository in three different programming languages. To address this, we recently started to develop our client apps using React and React Native.

Tech Infrastructure: In order to stream music to multiple simultaneous users, we must have a robust backend that includes servers, databases, and static files. While this seems straightforward, academic projects such as ours typically do not have dedicated tech professionals to monitor and maintain the system. We have attempted to adopt best practices (unit testing, continuous integration, automated monitoring), but this continues to be difficult for us. It also comes at a cost of several thousand dollars per year in cloud computing expenses.

Promotion: With only a small advertising budget, we rely on word-of-mouth recommendation, social media, artist outreach, and creative exchanges (e.g., volunteering at festivals in return for free advertising) to help get the word out about MegsRadio. This takes a surprisingly large amount of time and energy, but many of the undergraduate students seem to enjoy working with members of the local music community.

Licensing: One advantage of being a not-for-profit academic project is that licensing costs are relatively inexpensive when compared to commercial rates. Our academic institutions maintains performance rights licenses and we pay about $600 per year for a statutory sound recording license.

At present, we have built up a small base of 10-20 active daily users. However, we have a lot of work to do in terms of improving our user experience before we can expect to grow to a level (100+ active daily users). Once we reach this target, we will be able to better use our system for real-time playlist algorithm research.

5. ACKNOWLEDGMENTS

This work is supported by NSF Grants IIS-1615679, IIS-1615706, and IIS-1513692.

6. REFERENCES

[1] D. R. Turnbull, J. A. Zupnick, K. B. Stensland, A. R. Horwitz, A. J. Wolf, A. E. Spirgel, S. P. Meyerhofer, and T. Joachims. Using personalized radio to enhance local music discovery. In *Extended Abstracts on Human Factors in Computing Systems (CHI)*, pages 2023–2028, 2014.

[1] The United States' Digital Millennium Copyright Act (DMCA) requires that we do not repeat an artist or track too many times within a fixed period of time.