Crowdsourcing-Based Musical Predictions

Nicholas Boyd
Department of Computer Science
Saint Joseph's University
Philadelphia, PA, USA 19131
Email: nb619430@sju.edu

Wei Chang
Department of Computer Science
Saint Joseph's University
Philadelphia, PA, USA 19131
Email: wchang@sju.edu

Jie Wu
Department of Computer
& Information Sciences
Temple University
Philadelphia, PA, USA 19121
Email: jiewu@temple.edu

***Abstract*—Data is becoming more and more valuable as technology advances. Through crowdsourcing organizations are able to collect large amounts of data at an effective rate with little cost. This paper proposes a crowdsourcing-based music playing system, where the next song to play is determined by the listening preferences of realtime online users. Unlike conventional radio play system, where songs are randomly selected, our system selects songs which satisfies the listening preferences of a majority of users, who are currently online. Our system consists of two important components: a predictor, which estimates a user's listening preferences based on his features, and a decision maker, which selects the next song to play based on a majority vote. In order to find out the relationship between user features and listening preferences, K-means algorithm is adopted. A large amount of real data is collected via online surveys, and extension simulations show that our system can effectively selects a proper song for its online audience.**

***Index Terms*—Crowdsourcing, Clustering, Mobile Networks**

I. Introduction

Crowdsourcing offers a new way of getting large amounts of data for next to nothing, by employing people from all over the world. Jeff Howe describes the process of crowdsourcing as a "new pool of cheap labor: everyday people using their spare cycles to create content or solve problems"[1]. Figure 1 shows how the basic framework of crowdsourcing allows the labor force of everyday people to complete a series of tasks for a wide range of companies and purposes.

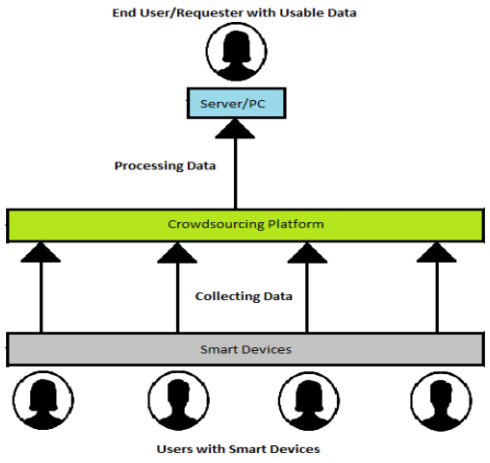

Fig. 1. Simple Crowdsourcing Framework

Users with smart devices- Through the use of internet-capable smart devices, users or workers are able to participate in crowdsourcing tasks, by entering their results in a crowdsourcing platform like Amazons Mechanical Turk[2][3] or TopCoder[4].

Collecting Data- Platforms use a wide variety of question types to get different types of data from a users. For example, if a company is looking to get an answer in a specific range, they may prompt the worker with a multiple-choice survey.

Processing Data- After the data is collected, the platform may need to sort or group data to make the dataset more usable and organized. Popular sorting and clustering techniques include K-Means clustering and Hierarchical Clustering[5].

End User/Requester with Usable Data- After the data has been sorted, the End User/Requester has access to a dataset that can be used for a wide range of things, such as marketing, feedback to make changes to a game or program, or to study for research.

With the huge increase in wireless network, a framework that is able to reach many people on a range of devices is necessary in the modern world. Using this framework, many companies have been able to find effective solutions to a wide range of problems for little or no cost. This paper uses a modified version of this framework that will make musical predictions based on preferences of listeners who are currently tuned in to a radio station. After these musical predictions are made, crowdvoting techniques will be utilized to build a dynamic queue of songs. This process will be completed using clustering techniques that are applied to the set of user data that is collected.

Our main contributions for this paper are summarized as follows:

1) We propose a crowdsource-based musical platform, where songs are selected based on active user/listener feedback and preference;
2) We apply the K-Mean clustering algorithm to better learn the relationship between user/listener preference and song choice;
3) We collect a real dataset through popular platforms like Spotify and Google Forms;
4) We conduct extensive real data-driven experiments that test our proposed solutions.

This paper will be divided into multiple sections: Section II, *Related Work*, will outline papers, platforms, and examples

that integrate similar methods and techniques, implementing a wide range of data collection methods. Section III, *Crowdsourcing With Music*, will describe the modified crowdsourcing framework that is used. Through this framework we will then describe how we collected and processed the data using popular music platforms, including problems encountered while collecting the data, an overview of clustering techniques that were applied, and the results of the simulations that shows which songs a radio station should play. Section IV, *Simulations*, will outline the data processing methods used, as well as the results from the simulations. Finally, section V, *Future Work*, will outline work that still needs to be done to take this framework from a test setting to real world implementation. This section ends by discussing common security risks with the overall process of crowdsourcing, and how they need to be kept in mind as this project moves forward.

II. Related Work

A. Popular Platforms

Crowdsourcing platforms have been around for quite some time, offering users a range of monetary prizes and rewards for completing tasks as easy as answering a survey, to something as difficult as writing code for government agencies. Working with these platforms allows us to see similarities and differences with how these sites manage and facilitate problem solving from the requester/end user to the workers. Some of the most popular crowdsourcing platforms are Amazon's Mechanical Turk, TopCoder, InnoCentive, iStockPhoto, Threadless and Crowdflower. Although these platforms may ask questions that appear to be simple, with little application to a real world problem, there are many examples of large companies using crowdsourcing to solve real-world issues. One example of this is using user reviews to make adjustments to their product that can greatly improve performance. PLAYERUNKNOWN'S BATTLEGROUNDS[6] is one of the most popular games with over 300,000 daily players, and is one of the first games in the battle-royale genre, relying heavily on user feedback[7] for new content and overall development. The process of how this game makes frequent adjustments directly based on user preference. Our project takes a similar overall process, but instead of making weekly updates, our framework will change in real time as new users tune-in.

B. Similar Approaches To Processing and Collecting Data

Similar problems have been tackled by other papers, for example, "Making Many People Happy: Greedy Solutions For Content Distribution"[8] by Jie Wu, *et al.* outlines how greedy algorithms can be used in maximum coverage problems. Comparing the results from local greedy algorithms, simple local greedy algorithms and complex local greedy algorithms this paper concludes that the use of a complex local greedy algorithm performs best, not limiting its cluster centers to a point that is in the dataset. Similar to our clustering technique, the centers can be anywhere in the plot for more accurate clusters.

Since our musical selection system will be available to a wide range of devices on various platforms, data collection will used be achieved in part through what Layla Pournajaf, *et al.*[9] refer to as Mobile Crowd Sensing (MCS). A Mobile Crowd Sensing network is a system of various smart devices that interact through multiple signals and sensors (i.e. Wifi, Bluetooth, or Cellular). This paper also describes "Opportunistic Sensing", which is the process of crowdsourcing data with little or no user involvement. Our system uses this form of crowdsourcing because the musical data is collected automatically, a large portion of this framework requires little user involvement, only requiring the users to participate in crowdvoting [10]. This little user involvement is important when crowdsourcing because tasks with a large amount of user involvement is sometimes not as accurate [11].

Another example can be found in "Progressive or Conservative: Rationally Allocate Cooperative Work in Mobile Social Networks" by Wei Chang, *et al.*[12] that outlines how to assign workload in a distributed crowdsourcing system. Similarly, the framework in this paper is a centralized crowdsourcing system that deals with similar distribution tasks.

III. Crowdsourcing With Music

The music industry is another place that can greatly benefit from crowdsourcing, specifically, a radio station ability to crowdsource music history from users/listeners to effectively select a new song or make a recommendation that is similar to the music taste of that same users/listeners. Using a modified version of the simple crowdsourcing framework, we have designed a bidirectional framework that would allow radio stations and users to send their musical data to a platform or sever that will be able to make recommendations that are specific to the users who are tuned in at a given time. This new framework will be focused around a server that is separate from the suers and radio station that will take data from both sides, process it, then send a song back to the pool of listeners with the goal of making the audience as happy as possible at any given moment.

A. Overview

Trying to find an effective process that focuses on the relationship between users/listeners and a radio station, our system is a centralized crowdsourcing system that is designed to make musical predictions by collecting and grouping songs that users have frequently played (See Figure 2). These songs are stored and grouped in a server, or 'predictor', that determines the centers of a set number of clusters. This predictor then is able to collect data on the side of the radio station, specifically, previous songs the radio station has played. After the data on both the users and the radio station is collected, they can then be compared to see where the centers of the users clusters fall in relation to the data from the radio stations in another service called the 'decision maker'. The 'decision maker' compares the cluster centers from the users data, to the data from the radio station. This comparison will make a list of possible

songs that could be played. Finally, the 'decision maker' will send a song from this list out to the users.

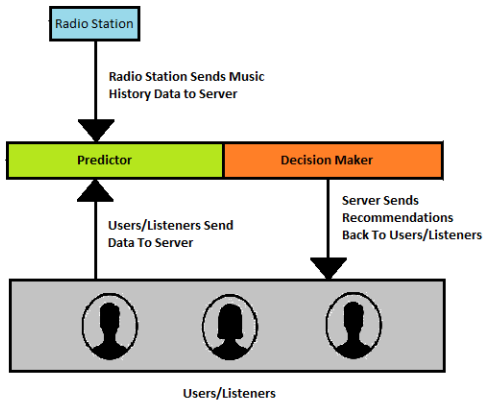

Fig. 2. Modified Crowdsourcing Framework

B. Collecting The Data

In an ideal situation, users will have their music streaming devices linked to the radio station through a server that will be able to track what a user listens to, examine the songs and break them into groups that will be compared to the songs a radio station has played/will play to determine how closely the new songs is to the group of songs the user has already played. For our testing, we used the Last.fm[13] 'scrobbling' feature paired with Spotify[14] to collect the data of a single user. Last.fm's 'scrobbling' is a 3rd party application that can run in Spotify, recording every song that a user played and sends it to their Last.fm account. After accessing the Last.fm account, you are able to view the users top played songs, genres, artists and albums. We broke down the users 50 most-listened to songs through a Spotify's "Sort Your Music" service[14] that decompiles songs into a chart of various numerical values (See Table 1 for example).

TABLE I
USER DATA COLLECTED FROM SORT YOUR MUSIC SERVICE

Title	Artist	BPM	Energy	Dance
Your Graduation	Modern Baseball	185	90	40
Constant Headache	Joyce Manor	99	89	41
Disappeared	Sorority Noise	97	94	42
True Believers	The Bouncing Souls	98	98	38
Nutshell	Alice In Chains	136	56	38

For our simulations, we decided to limit the data to song title, artist, Beats Per Minute (BPM), Energy, and Dance. Spotify Define BPM, Energy, and Dance as: *BPM*- Beats Per Minute how fast the song is. *Energy*- The energy of a song - the higher the value, the more energetic the song. *Dance*- The higher the value, the easier it is to dance to this song.

C. Clustering The Data

To process the large amount of data that our framework will be collecting from the users and radio station, we are using the K-Means Clustering algorithm to cluster the data into K clusters. Having many advantages, the K-Means clustering algorithm was chosen based on its ease of implementation and its ability to consistently clusters large amounts of data in a reliable way. K-Means is a partition-based algorithm [5] that partitions a set of data into a specified number of clusters. These clusters are calculated by comparing the distance of surrounding points to K center points, and the closest center will determine what cluster the point will be put under. After the point is classified in a cluster, the center is recalculated based on all points in a cluster, and the process is repeated recursively until all points are classified.

While K-Means produces quality results, we faced several issues while trying to apply K-Means to our musical data. We originally conducted surveys that asked the user about favorite genre of music, but found that since musical genres are discrete rather than continuous, applying K-Means to this set of data yielded meaningless results. To solve this, we decided to decompile the songs using Spotify's "Sort Your Music" service several, numerical categories. Since these were now numerical values, K-Means was able to cluster songs into meaningful groups.

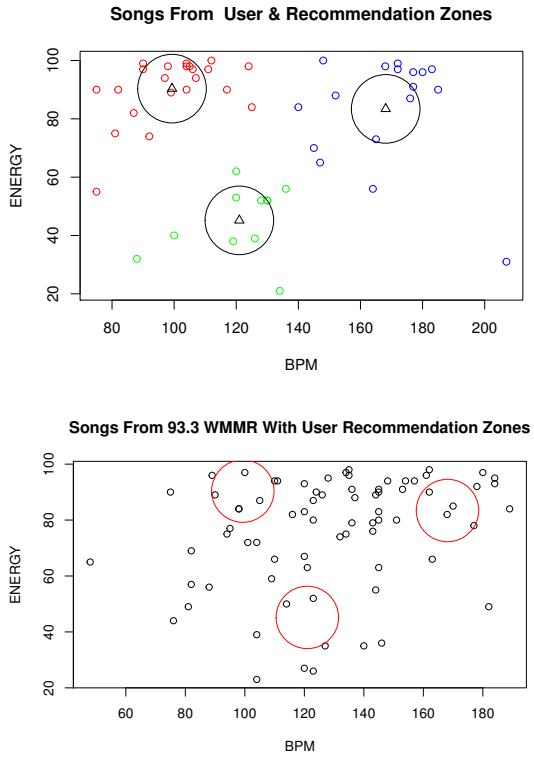

Fig. 3. K-Means Simulation Results

D. Song Selection Process

The "Decision Maker" is an essential part of this framework, that handles selecting an appropriate song after the data has been collected. During the decision making phase of the framework, the recommendation zones are compared with the

song data from the radio station. Another list of songs is created based on songs that fall inside the preference zone (refer back to Figure 3). From here, the "Decision Maker" will select the song based on a preference score that is computed from user/listener input. This preference score is based on a majority-voting system that users are able to interact with, to build a queue of songs that will be played based on user feedback. This system is designed to operate as a dynamic radio station that will be able to select songs based on what users historically have listened to, and real-time feedback through user/listener crowd-voting. The "Decision Maker" part of the song selection process will be how the entire framework comes together to deliver quality results and make the most users/listeners happy.

IV. Simulations

After collecting the musical data that was needed from the user, we graphed the data on two separate 2-D graphs. The first graph, we plotted the data by (BPM,Energy) and the second graph by (BPM,Dance). BPM was always a constant, because that is one of the easiest and universal ways to get a numerical value for an aspect of the song that isn't the length of the song. Next, we used 93.3 WMMR's music archive [15] of previously played songs as the radio station for our testing environment. For our test, 93.3WMMR is heavily based around the "Rock" genre. Passing an entire day's worth of music through the same Spotify sorting service, we had a group of music to compare the users data to.

Similar to crowdsourcing examples and platforms, we needed to apply some sort of clustering to the dataset. Clustering techniques take a group of data and try to find relation between the points into groups that all share similarities. Two of the most popular methods for clustering are K-means clustering and Hierarchical clustering[5]. K-means clustering partitions the dataset into K clusters where all of the data points inside a partition are related to one another. Hierarchical clustering can be subdivided into two categories, Agglomerative and Divisive. Agglomerative is a bottom-up approach that starts by assuming all individual points are a cluster, then merges closest cluster together until there is either one giant cluster (All data points together-same as not running any clustering technique) or a desired number of clusters is reached. Divisive is a top-down approach assuming all points make up one giant cluster, then recursively breaks into smaller clusters until a desired number of clusters is reached. For its simplicity and ease, we used K-means clustering to find three clusters in the users musical history. The K-means simulations and graphs were made using the R statistical language in RStudio.

Applying the K-means algorithm to the users data gave us three clusters, with three centers that can be used as a starting point to make the recommendation. After the cluster center was found, we expanded the area around the center point into what we refer to as the "recommendation zone". In Figure 3, the lower graph these zones are displayed as red circles, representing the processes of comparing these zones to the list of songs from the radio station. This zone represents the range of values that a recommendation song should have to be similar to a group of songs that the user has already listened to. These recommendation zones can then be compared to the set of songs played by the radio station (In our case, 93.3WMMR's dataset)to find which songs could be played next that would be related to the users musical history. Numerically, each part of the song that is examined (BPM, Energy, and Dance) is generally between 20 and 200. This range allowed us to make the recommendation zone a circle with the cluster center as the center, with a diameter of about 20, to get a small range for songs that should be played.

After getting the recommendation zones for the desired characteristics of a song, they can then be compared to the graphs from the radio station's songs (See figure 3 for examples).

A. Additional Data Collection

In addition to platforms like Spotify and Last.fm, to collect real world data on musical preference, we sent out surveys on social media websites that were created in Google Forms. To get an idea of what this process will look like with a large pool of users outside our testing environment, we collected data that asked the users to select what their favorite genre of music was. This simple question was effective because we were able to get a good idea of the general interest of a group of people, while not wasting the time of individuals with a lengthy survey. After sending this survey, we received over 140 responses in a week, with a wide range of genres (See Figure 4 for results). This data is important, because this shows that there are a wide range of musical genres, so the need for our system to change dynamically as new users/listeners tuned in, is extremely important.

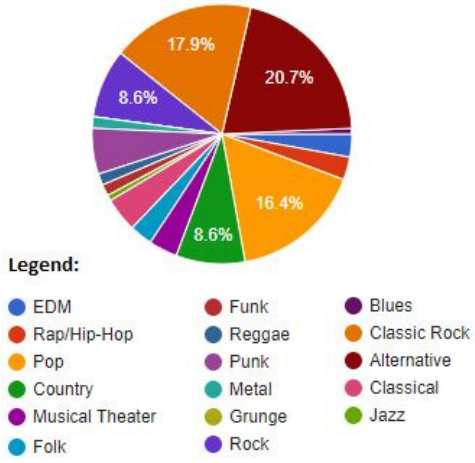

Fig. 4. Results From Survey

B. Results

After running several simulations with data from both a user and 93.3WMMR, we were able to successfully make recommendations based on a groups of songs by breaking them down by Beats Per Minute, Energy, and Dance numerical

values. While our simulation only examined a dataset that came from a single user, this process can easily be scaled to a group of users by plotting all the data on a single graph. Also, while we used a 2-D graph to find the clusters and centers, this framework can also easily handle higher multi-dimensional datasets. For example, instead of plotting the songs as (BPM, Energy), you could plot the songs in a 3-D space, (BPM, Energy, Dance), and still successfully run the K-Means clustering algorithm. This process can also be applied to different radio stations to get different types of recommendations.

V. Future Work

While we were able to successfully make musical recommendations based on crowdsourced musical data, this framework still only exists and operates in a limited test setting. Future work includes development of the framework and implementation of this system in the real world, collecting data from multiple people. Also, since this is based around crowdsourcing, there are security risks that still exist in the larger process, such as privacy and malicious users[16][17]. Specifically, how can you protect the identity of the users/listeners.

VI. Conclusion

In this paper we have described a modified framework that uses crowdsourcing to collect musical data from a set of users and a radio station, as well as related crowdsourcing examples that show similar approaches to data collection. We also outlined how we used the K-Means clustering algorithm to process the data in real-time in our theoretical framework. We outlined the importance of a system that can change as new users/listeners become active, through the user of survey platforms like Google Forms. This dynamic system will be using a series of methods to allow user feedback to decide what musical direction the system will shift toward, through the use of crowd voting. Finally, we showed how you can combine the previously stated methods to make a musical recommendation with the goal of satisfying all active listeners. Overall, we have described how crowdsourcing can be effectively intergrated into the music industry to create a unique listening experience that changes with the users/listeners.

References

[1] Howe, Jeff. *"The rise of crowdsourcing."* Wired magazine 14.6 (2006): 1-4.

[2] *https://www.mturk.com/mturk/welcome*

[3] Difallah, Djellel Eddine, et al. *"The dynamics of micro-task crowdsourcing: The case of amazon mturk"* Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015.

[4] Mao, Ke, et al. *"A survey of the use of crowdsourcing in software engineering."* Journal of Systems and Software 126 (2017): 57-84

[5] Munir, Mussawer, et al. *"Comparitive Study of Clustering Techniques in Data Mining"*

[6] *https://www.playbattlegrounds.com/overview.pu*

[7] Villarroel, Lorenzo, et al. *"Release planning of mobile apps based on user reviews."* Proceedings of the 38th International Conference on Software Engineering. ACM, 2016.

[8] Wang, Yunsheng, Yuhong Guo, and Jie Wu. *"Making many people happy: Greedy solutions for content distribution."* Parallel Processing (ICPP), 2011 International Conference on. IEEE, 2011.

[9] Pournajaf, Layla, et al. *"Participant privacy in mobile crowd sensing task management: a survey of methods and challenges."* ACM SIGMOD Record 44.4 (2016): 23-34.

[10] Prpic, John, et al. *"How to work a crowd: Developing crowd capital through crowdsourcing."* Business Horizons 58.1 (2015): 77-85.

[11] McDuff, Daniel, Rana El Kaliouby, and Rosalind W. Picard. *"Crowdsourcing facial responses to online videos."* Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on. IEEE, 2015.

[12] W. Chang and J. Wu, *"Progressive or Conservative: Rationally Allocate Cooperative Work in Mobile Social Networks,"* IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 7, 2015, 2020-2035.

[13] *https://www.last.fm/home*

[14] *http://organizeyourmusic.playlistmachinery.com/*

[15] *http://wmmr.com/stream/WMMRFM/*

[16] Gadiraju, Ujwal, et al. *"Understanding malicious behavior in crowdsourcing platforms: The case of online surveys."* Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, 2015.

[17] Yang, Kan, et al. *"Security and privacy in mobile crowdsourcing networks: challenges and opportunities."* IEEE Communications Magazine 53.8 (2015): 75-81.