Jan Dlabal
Silviana Ciurea-Ilcus
Norman Yu

An artist recommendation system based on artists' collaboration patterns

Abstract:
Despite the plethora of readily available songs and music recommendation engines, it remains difficult to discover new music that one likes. Current recommendation software consists of either content-based approaches or collaborative filtering, which often get stuck in simplistic classifications of music or cannot be applied to recommend new items that have not been previously consumed or artists that are not in the musical mainstream. In our project, we show that an approach based on the structure of the graph of artist collaborations can provide relevant, non-obvious recommendations in a variety of scenarios. We experiment with a flow-based algorithm and test it against user-generated playlists. We also examine the spread in the time-periods of the artists our algorithm suggests and visualize part of the artist collaboration network, based on frequently-occurring recommendations.

1. Introduction

There are literally hundreds of thousands of artists whose songs are readily available to anyone with an internet connection on online services such as iTunes, Spotify, or Last.fm. However, it remains difficult to discover new artists that one might like. Even though many of the above services feature various recommendation engines, they often get stuck in classifications of music that are too simplistic to yield anything interesting. Spotify, for example, has many playlists, but too often those playlists are based on simple genre tags, which results in a rather dull listening experience after a while.
In our project, we try to find a better way to search through music and find artists that a user will like, based on cooperation between artists (i.e. a song featuring another artist, and similar). We also hope that analyzing the connections artists choose to make between themselves will provide us with some insight into the music landscape as a whole.
The current state of the field of music recommendation is characterized by two approaches common to nevertheless, collaborative filtering depends on usage data being available and thus cannot be applied to recommend new items that have not been previously consumed.

2. Related work

Our project aims to alleviate some of the problems in music recommendation discussed in "From hits to niches: or how popular artists can bias music recommendation and discovery"[1]. The paper seeks to answer whether current recommendation systems are well equipped to recommend music in the "long tail" of the music available. In particular, the paper examines the effects of popularity bias in music recommendation. The paper finds that common techniques such as collaborative filtering are prone to popularity bias - the more popular tracks are overly represented in collaborative filtering systems, while

[1] O. Celma, and P. Cano. "From hits to niches?: or how popular artists can bias music recommendation and discovery." Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition. ACM, 2008.

tracks in the long tail are not recommended often enough. The paper performs several experiments to evaluate the effect of popularity on the following recommendation systems: collaborative filtering, content–based audio similarity, and human expert–based resemblance. Their results show that collaborative filtering used by last.fm tended to reinforce popular artists and thus yields a poor discovery ratio.

While there do not appear to have been any attempts to recommend artists based on their collaboration network, the authors of "Analysis and Exploitation of Musician Social Networks for Recommendation and Discovery" attempted to use the social network information, in particular, myspace social network information, to aid music recommendation and discovery[2]. To gain an understanding of whether there exists valuable information in the social networks, the authors consider a sub-network that only includes the top connections between artists. Then the authors analyze both the structure of this network as well as metrics involving acoustic distance between songs. Using this information, the authors were able to identify neighborhoods of artists that closely resemble music genres, as measured by genre entropy. The paper suggests future work of using max-flow on a graph that incorporates both the acoustic similarity as well as social-network data into the edge weights to make recommendations. The algorithm would take as input two artists and model an artist as a source and the other as a sink for the max flow problem. This paper is relevant to our framework as we also use external information such as collaboration as input into our recommendation algorithm. In particular, we believe that collaboration is a better signal than social network connection because collaboration is an authentic signal and carries more value than a social network connection.

3. Data

3.1 Graph data

We were initially planning to the million song dataset, available at http://labrosa.ee.columbia.edu/millionsong/ and extract specific music metadata from it, such as the song name, the artist(s) that were involved in the production of the song, and the album name. The million song dataset is a very large data set, with various features available for each song, such as timbre, pitches, tempo, etc.

However, after some more investigation, we decided to not use the million song dataset as previously planned. We made this decision because of two main issues with the million song dataset:

- for our purposes, the dataset contained a lot of information that is not relevant to what we wanted to do (timbre, tempo, etc.). This was especially an issue because the entire dataset weighs at about 280 GB, which is not very practical.
- the dataset is labeled with only one artist per song, which is not ideal for our purpose (we would ideally want to make links between different artists, which will not work well if every song we have only features one artist). We have previously thought that this could be solved through analyzing the titles of songs by looking for patterns such as "X feat Y", but this proved very error prone when testing on a small subset of the million song dataset.

Instead, we have found a dataset that suited our needs much better, in the form of the discogs releases dataset, of which monthly dumps are available at http://data.discogs.com/. The great advantage of this

[2] B.Fields, et al. "Analysis and exploitation of musician social networks for recommendation and discovery." Multimedia, IEEE Transactions on 13.4 (2011): 674-686.

dataset for our purposes is that it contains a set of releases, where each release is tagged with a list of artists that participated in creating it. This makes it much simpler for us to count the number of collaborations between artists, and creating the graph we wanted.
One caveat of this dataset is that it stores different stylizations/spellings of a band's name as different entries, instead of recognizing them as different ways to write the same band's name and collapsing them into one entry. This is a problem for bands whose
The dataset is also considerably more practical, because the size is a much more manageable 24 GB. This still proved to be a challenge to parse on the machines that we had available (i.e. normal laptops) due to its size. To make development simpler, we first parsed the 24 GB XML using the python library lxml.etree. We did this using the iterparse() functionality, which calls a callback function every time a new element is discovered. This has the advantage of being able to delete elements from the XML tree immediately after they are parsed, which is necessary because otherwise the library would try to keep the whole 24GB tree in memory which is not workable on our hardware.
Our parser discards most of the information from the XML that we do not use, and builds three data structures:

- artistMap: this is a map going from artist name to the id it corresponds to in the dataset
- edgeMap: this is a map going from an edge (denoted by two artist ids) to the weight of the edge, which is calculated as the count of releases on which the two artists collaborated
- artistNameMap: this is just the reverse of artistMap, i.e. going from artist id to a list of names this artist appears under

From this information we then build our main graph G, which consists of a node for each artist ID, and edges between artists that have collaborated.
We also cache artistMap, edgeMap, and artistNameMap using pickle so that we do not need to reparse the XML (which takes several hours otherwise).

4. Approach

4.1 Baseline: Flow algorithm for artist similarity

This is one approach we came up with for determining similar artists. The inputs are:

- starting list of artists the user likes
- graph G, along with weights of edges

The intuition of the algorithm is that we want to report artists that have collaborated a lot with the artists that the user gave us in the initial preference set. The algorithm works in two phases:

1) build neighbor list for each artist in the list of artists the user likes. This generates a list of neighbors at distance = 1 from the artist, list of neighbors at distance = 2, etc., up to a preset maximum. It also associates the weight from the edge weights with each artist. once we have this data, we assign importance to all the artists from the neighbor lists according to the formula: *(fraction of weight) * 0.95distance*, where 0.95 is just a constant that we can experiment with, distance is how far this artist was from a given artist from the user's starting list, and fraction of weight is the fraction of the edge weight that went to this artist. This is repeated for all the neighbor lists we have, so the end result will be a map going from artist id to the sum of weights that we computed for it.

2) sort the given map by the weight, and we report the top artists as artists that are deemed similar given the starting list the user gave us.

We initially did not normalize the weight. This resulted in quite interesting results, where a lot of the suggested artists were symphony orchestras. We assume that this is because there are relatively few symphony orchestras compared to the number of artists in general, but the orchestras tend to collaborate with a lot of people. Unfortunately, this is not really very useful data to us, since it doesn't really provide very relevant recommendations. Normalizing the weights helps account for this because the orchestras will generally get a low fraction of the weight, even though they appear a lot of times.

4.2 Flow algorithm with weight and iTunes support

According to the Phoenix 2 UK Project from 2006, music listeners can be grouped into four types: **savants**, whose music knowledge is very extensive and listen to music very frequently, **enthusiasts**, for whom music is a key—yet not the only—hobby, **casuals**, who listen to music occasionally, and indifferents, for whom music plays no important role[3].

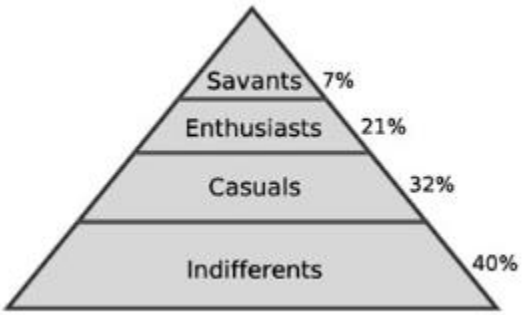

Figure 1.The four types of music listeners. Reproduced from O. Celma, Music Recommendation. Springer, 2010: 47

Our baseline algorithm does well when recommending new artists for casuals and indifferents at the very least; however, we wanted to make our algorithm suitable and valuable for all types of users. In order to

[3] D. Jennings,Net, *Blogs and Rock 'n' Roll: How Digital Discovery Works and What it Means for Consumers*. Boston, MA: Nicholas Brealey Publishing, 2007

make more clever and niche recommendations for savants and enthusiasts, who are probably uninterested in popular recommendations, we introduced the optional feature of an exploration radius. This feature aims to make the results less predictable for users who feel that the recommendations given by the original flow algorithm are too obvious, since the original flow algorithm gives highest weight to immediate neighbors of the user's preference artists, next highest weight to neighbors one hop away, and so on. Depending on the value (which needs to be greater than 0) that the user enters for the radius, the algorithm will skip artists that are too close to the original artists. . For example, if exploration radius = 2, this means that immediate neighbors will not count, and neighbors one hop away will not count; counting only starts at neighbors at least 2 hops away. Because of the subjectivity of this measure, we only conducted preliminary testing within our team. Time constraints made thorough testing of this feature with subjects that would rate the received recommendations unachievable. We choose to focus our efforts on understanding the performance of our basic algorithm, but we plan to conduct further tests of the expansion radius feature in the future.

We have used the exploration radius feature, by setting the radius to 1, to solve the initial issue of receiving artists from the user preference sets in the list of suggested artists. Due to the nature of our data set, which stores different stylizations of an artist's name as distinct entries, artists from the user preference set still occasionally entered the recommendations list.

In addition, our improved algorithm now supports the user giving weights to certain artists entered as queries. Our music recommendation program can now also integrate a user's iTunes library; we read the library from the library xml, whose path is set by the user in itunes.py (example: ITUNES_LIBRARY_XML_PATH = "/Users/jan/Music/iTunes/IML_BACKUP_JAN.xml")
The top 50 artists in the user's iTunes library are selected as the user preference set, they are weighed by play count, and matched with their corresponding IDs.

5. Tests

To test the flow algorithm, we used user-specific methodologies. We downloaded the top 10 rock and top 10 pop user-generated playlists on Spotify using their API.
We have noticed that the algorithm performs best when the user inputs 4-5 artists, or over 20 artists. For this reason, we have chosen to conduct our tests inputting a user preference set of size 4. For example, when these eight artists were inputted into the program as sets of four, the median overlap was 5/10, but when entered as a single set of eight, the overlap was 2/10.

INPUT: Nirvana, Metallica, AC/DC, Rammstein, Children of Bodom, Deep Purple, Rolling Stones, Chris Cornell, Led Zeppelin
OUTPUT:
The Royal Philharmonic Orchestra - 0.37
The San Francisco Symphony Orchestra - 0.36
Michael Kamen - 0.32
Malcolm Arnold - 0.29
Melvins - 0.24
King Buzzo - 0.24

t.A.T.u. - 0.21
Martin Scorsese - 0.19
Anthrax - 0.19

5.1. Measuring overlap with the downloaded playlists

For our first test, we chose 4 well-known bands which occurred in one of the top rock user-created playlists on Spotify, and examined the overlap between our algorithm's suggestions and the rest of the artists on the playlist. We then compared our suggestions with the contents of the other nine playlists. We ran the test both with and without weighing the user-preferred artists. We observed that when inputting the names of bands whose members have also pursued extensive careers, at least one of these band members would top the outputted list. (ex: Black Sabbath and Tony Iommi; interestingly, in all 5 tests where we included Black Sabbath, Ozzy was not listed as a similar artist) In the following results, we have bolded the artist names which occur in the original playlist, and italicized the names which only occur in the rest of the top playlists.

5.1.1 Weights or no weights?

We tested whether weighing the artists by the number of songs included in the initial playlist resulted in a great overlap with the original playlist. We have not been able to identify any significant increase or decrease in overlap when inputting multiple user preference sets of size 4, of which we give an example below. Based on a small set of tests using our iTunes libraries with and without weighing, we hypothesize that weighing only makes a difference for large user preference sets.

INPUT: Nirvana (1.0), Metallica (1.0), AC/DC (0.5), Rammstein (0.33)
OUTPUT: 2 results in the original playlist, 2 distinct results in the other playlists = 4/10 total overlap
Melvins - 0.24
The San Francisco Symphony Orchestra - 0.24
King Buzzo - 0.24
Michael Kamen - 0.22
Lou Reed - 0.12
The Jesus Lizard - 0.08
The Black Crowes - 0.07
Pantera - 0.07
t.A.T.u. - 0.07
Adam Freeland - 0.06

INPUT: Nirvana, Metallica, AC/DC, Rammstein
OUTPUT: (we have excluded Metallica with a differently stylized name from these results)
3 results in the original playlist, 2 distinct results in the other playlists= 5/10 total overlap
The San Francisco Symphony Orchestra - 0.25
Melvins - 0.24
King Buzzo - 0.24
Michael Kamen - 0.23
t.A.T.u. - 0.21
Lou Reed - 0.12
The Black Crowes - 0.11
Pantera - 0.11
Marilyn Manson - 0.10

5.1.2 Total overlap with the downloaded playlists

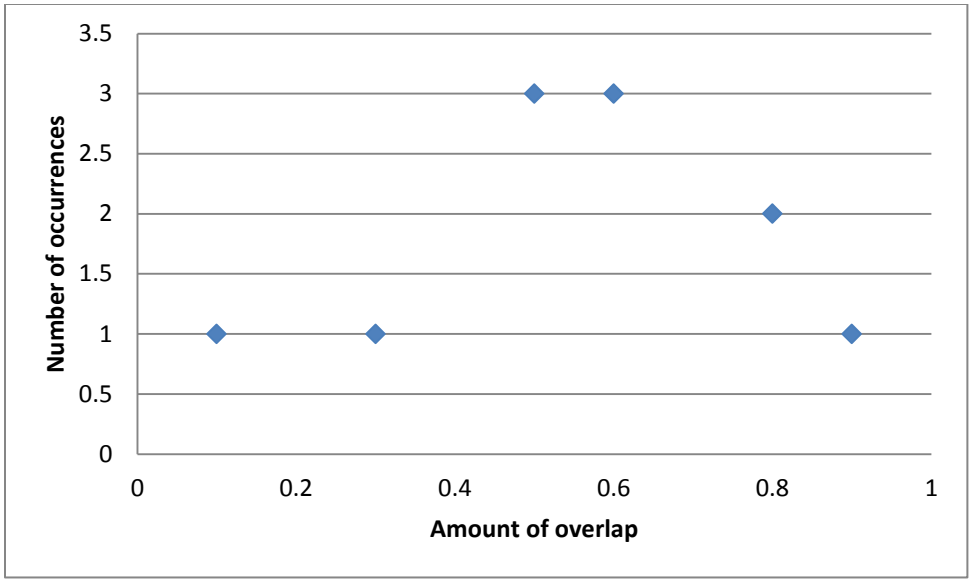

Figure 2. The total overlap of the suggested artists with the downloaded rock and pop playlists.

We have observed that more than 0.5 of the suggested artists overlapped with the artists featured in the downloaded playlists in more than 80% of the tested cases. We tested using sets of four rock artists in 5/11 cases and pop artists in 6/11 cases. We observed an average of 0.2 increase in overlap for pop artists from rock artists. We illustrate below the most successful query:

INPUT: Britney Spears, Ariana Grande, Madonna, Mariah Carey
OUTPUT: 9/10 overlap; the only non-overlapping artists is Ariana Grande with a differently stylized name, which we excluded
Nicki Minaj - 0.33
Madonna - 0.33
Iggy Azalea - 0.31
B.o.B - 0.24
Will I Am - 0.19
Big Sean - 0.19
Jessie J - 0.18
David Guetta - 0.15
Zedd - 0.11

We also attempted to test our recommendation program against electronic artists producing "chill" music which can usually be found in user-generated playlists which contain focus- and work-friendly music. We undertook five tests using artists from a popular "Chillwork" playlist. While the results we obtained certainly belonged to the targeted genre, we only obtained a median of 1/10 overlap with other top focus-friendly playlists. We believe that is because the artists making such music are smaller and more niche, and because the genre does not contain a lot of collaborations.

5.1.2.1 Why do certain artists appear in most sets of recommendations for rock user preference sets?

We noticed that t.A.T.u, Melvins, Michael Kamen, and King Buzzo appeared in most sets of recommendations when the input were rock artists. In order to understand why they were featured as recommended so often, we ran the algorithm with these four artists as inputs. We obtained 30% of the queries which led to the four artists as recommendations listed as suggestions, along with the San Francisco Symphony Orchestra with which these four artists were frequently co-occurring in the recommendations lists.

INPUT: t.A.T.u, Melvins, Michael Kamen, and King Buzzo
OUTPUT:
Nirvana - 0.61
Rammstein - 0.50
The San Francisco Symphony Orchestra - 0.33
METALLICA - 0.32
The Jon Spencer Blues Explosion - 0.13
Patton Oswalt - 0.10
Sharam Tayebi - 0.10
Jello Biafra - 0.09

In order to understand our results better in terms of the structure of the network, we selected the subgraphs containing t.A.T.u, which do not have a high total number of collaborations, and all the nodes which were 2 hops away from them. We then visualized the graphs using Gephi, using the Yifan Hu Proportional layout and then the label adjust, coloring the nodes increasingly dark according to their distance from the origin (i.e. t.A.T.u. or Melvins) (Fig.2). We also sized the nodes according to their degree in the artist graph, and sized the node labels proportional to the node's degree (Fig.3) We observed that t.A.T.u. is directly connected to artists from multiple genres, such as electronic-rock (Depeche Mode), hard rock (Rammstein), and pop (Fergie). From Fig.3 we can see differences in the structure of the network by genre. The rock section of the graph (3—inset) connected to t.A.T.u. mostly through Rammstein, consists of artists which do not have high total numbers of collaborators, but which are very densely connected amongst each other. Contrastingly, the pop section of the graph consists of artists which are less densely connected with elements of this graph, but which have a higher number of total collaborators.

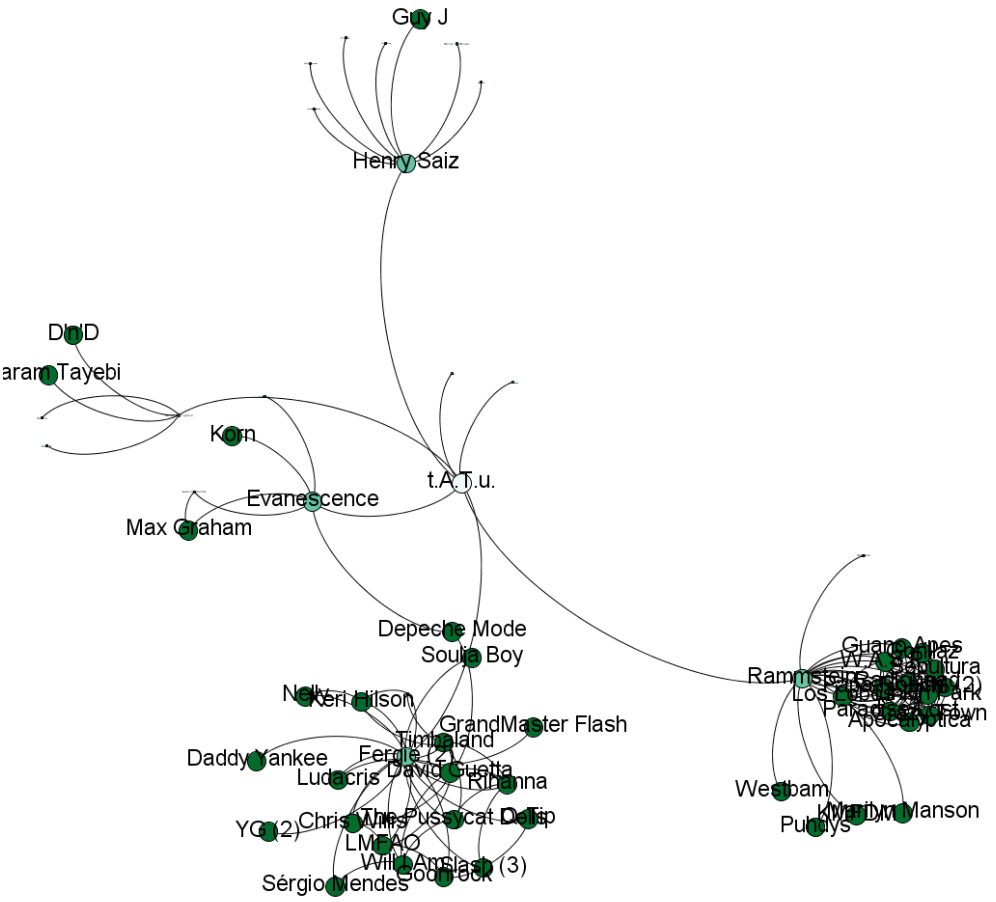

Figure 3.Visualizing the neighbors of t.A.T.u, without any node sizing.

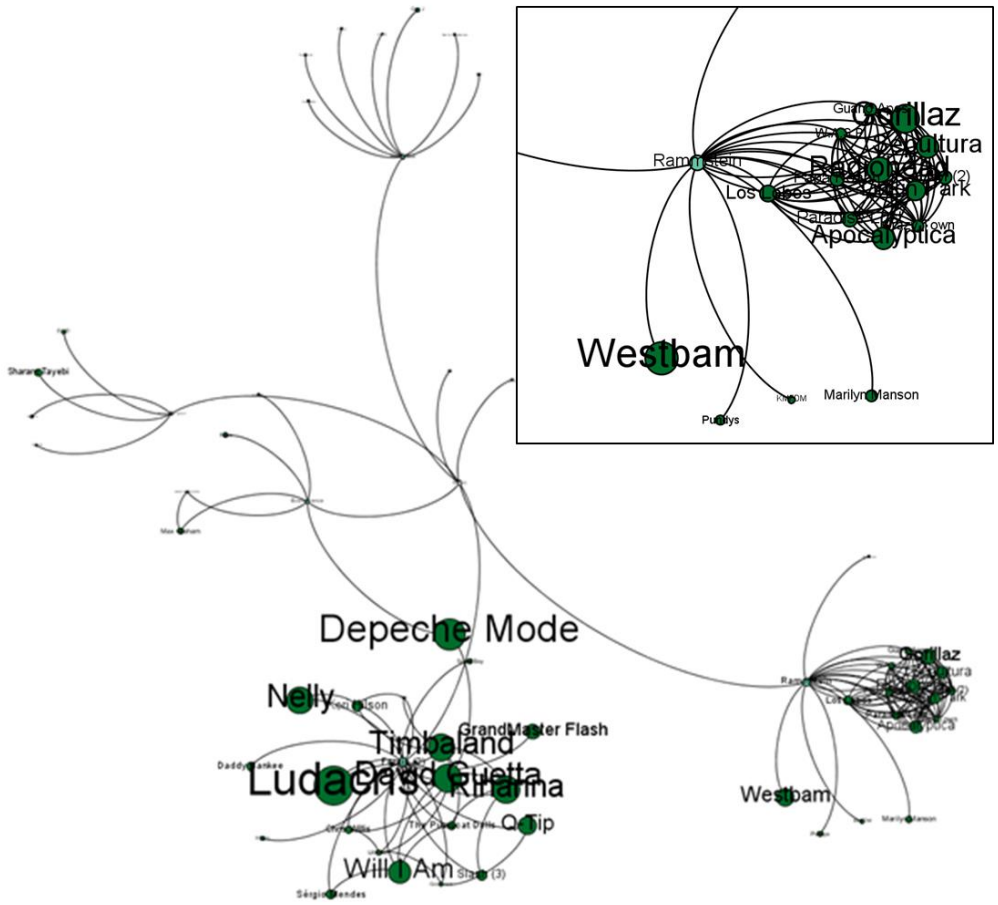

Figure 4.Visualizing the neighbors of t.A.T.u, with the node sizes proportional to their degrees.

5.2 Investigating the time periods of the results by inputting artists from one decade

For the second test, we wanted to test investigate the time periods spanned by the generated results, and compare them with those of the inputs. These results also begin to yield some insight into collaborations across decades within different genres of music. For example, the collaborators around artists active in the rock/folk scene in the late 1960s and early 1970s worked in diverse genres; for example, for this time period, our algorithm listed The L.A. Express—a jazz band Joni Mitchell collaborated with, Janet Jackson—who did not collaborate directly with any of the artists in the user preference set and is active on the pop scene, and Q-Tip—who again did not collaborate directly with the inputted artists and is active as a rapper. In contrast, the suggested artists for the 90's belong rather strictly to the rock genre. Similarly, given an input of contemporary pop/hip-hop artists, our recommendation algorithm suggests more pop and hip-hop artists. Moreover, 50% of the artists suggested for the 90's user preference set were either active or most successful in that time period, as opposed to the artists suggested for the other two user preference sets.

For this test, we chose sets of four rock and four pop artists that were active in one particular decade. We did not limit our user preference set to artists which were *exclusively* active during that decade. In the following results, we bolded the names of the suggested artists which were active/predominantly successful only in the given time period, and italicized the names of the artists who were active across a wider time period.

Late 1960s, early 1970s
INPUT: Led Zeppelin, Deep Purple, Janis Joplin, Joni Mitchell
OUTPUT:
Big Brother & The Holding Company - 0.62 (1965-1984)
The L.A. Express - 0.48 (1974-1980)
The Royal Philharmonic Orchestra - 0.35 (n/a)
Malcolm Arnold - 0.28 (wrote film scores; 1947-1969)
The Band - 0.17 (1964–1977, 1983–1999)
Full Tilt Boogie Band - 0.15 (1969-1970)
James Taylor (2) - 0.13 (1966-present)
janet jackson - 0.12 (1973-present)
Q-Tip - 0.12 (1988-present)
Curtis Knight - 0.09 (1967-1999)

90's artists
INPUT: Nirvana, Smashing Pumpkins, Soundgarden, Pearl Jam
OUTPUT (excluding the differently stylized names of Nirvana and Pearl Jam):
Melvins - 0.34 (1985-present, predominantly active in the 90's)
King Buzzo - 0.33 (1998-present)
Neil Young - 0.20 (1965-present)
The Plastic Population - 0.14 (1988-2001)
Alice in Chains - 0.13 (1985-present, predominantly successful in the 90's)
The Jesus Lizard - 0.10 (1987-1999)
David Bowie - 0.10 (1962-present)
Paul Weller - 0.10 (1990-present)

Contemporary pop/hip-hop artists:
INPUT: Ariana Grande, Iggy Azalea, Nicki Minaj, Jessie J
OUTPUT: (excluding the differently stylized names of the original artists)
Britney Spears - 0.35 (1992-present)
Madonna - 0.28 (1979-present)
David Guetta - 0.28 (1985-present)
B.o.B - 0.24 (2006-present)
Big Sean - 0.19 (2007-present)
M.I.A. (2) - 0.17 (2000-present)

6. Conclusions

Our algorithm performs well in tests measuring the overlap between suggested artists and artists in user-generated playlists, achieving on average 20% more overlap for user preference sets consisting of pop artists than of rock artists. While the algorithm made genre-appropriate suggestions when inputted electronic focus-friendly music, its suggestions did not overlap with the artists in the top "Chillwork" playlists, because this genre consists many small, niche bands who do not collaborate frequently. Certain relevant artists, who did not have a high total number of collaborations, appeared frequently in the suggestions; upon visualizing their neighborhood in the graph, we understood this was because they were connected to some nodes with very high degrees and to some very dense communities.

Our algorithm also makes time period-appropriate suggestions, as tested against user preference sets with artists from three different time periods.

We have hence shown that artist collaboration patterns can be useful in recommending new artists to users, providing relevant contents while also including non-obvious results, which can be further "diversified" by using an expansion radius. In our future work we will further test the expansion radius feature.