**DePaul University: College of Computing and Digital Media**

dittAudio: A Spotify music recommender system

**ABSTRACT**

Theoretical recommendation systems have been proposed in countless scientific papers. This paper seeks to simplify the bridge between complex recommendation algorithms and the general population. Utilizing the Spotify web API, dittAudio provides an interactive GUI and the ability to latch to a user’s premium Spotify account to build playlists, play songs, and generate recommendations based on Spotify audio features. Custom recommendation algorithms can easily be implemented as a class with a common dictionary-based format of sending and receiving data/recommendations. Designed as a building block, this paper brings together theory and implementation to help guide future recommendation system designers.

KEYWORDS: Recommendation Systems, Music, Spotify, Graphical User Interface (GUI), and

INTRODUCTION

The rate of digital content production has exponentially increased since the birth of the internet, and as a result users will be overcome with options for content to enjoy. Given the broad diversity of content and each person having their own preference for certain types of content, recommendation systems are now a part of our every day lives to ensure we can find the content we enjoy more easily. For music specifically, there are hundreds of services that provide music streaming and recommending; Spotify, YouTube Music, Pandora, radio.com, Sirius Satellite Radio, AM / FM radio, and many more. Each of these collect user profile data to help improve recommendations to their users, and thus keep their business. The issue we have, is that the general population typically does not have access to other user’s data, so this creates the problem of “the rich get richer”. To explain this issue further consider the fact it becomes very difficult to best a recommender system that uses both collaborative and content-based recommendation, while using solely content-based recommendation. It is because of this issue, that much research has been done on how to generate good recommendations using strictly the data open to the public.

Creating a successful recommender system can be incredibly difficult and involve a lot of complex math and computer programming that most users have no knowledge of. To simplify this for the general populous, programmers typically separate out a front-end graphical user interface (GUI) and a back-end subsystem. The front-end focuses on making the application easy to use, think of the recommended videos on YouTube while watching one you are interested in. This would be the front-end, in that the user does not see all the code and math that goes into finding the videos suggested in that window. The back end holds all the code that is not relevant to the user. This may sound simple to design a GUI that only shows the user what they need to know, but much research shows that there are certain methods a designer should take to make the users feel like the recommendations are better, even if they are not.

The logical next step in designing a successful recommendation system that can compete with larger streaming services is to put both the back end and front end together. From our research, there is little done to bridge this gap with respect to open-source code that can help developers start building their own end-to-end solution.

LITERATURE REVIEW

Many works regarding the implementation and usability of recommendation systems have been published providing some good background knowledge of “what works” and “what does not”. This section will compile findings from other scientists relevant to designing and implementing a music recommendation system. Design decisions and implementations based on these papers guided the logic for developing dittAudio, and drove the desire to provide the building blocks for successful app development.

Topic 1: Personal Characteristics on Music Recommender UI’s

This paper talked about creating an effective UI that made users feel like they discovered new music and that it matched the taste/style of the inputs they provided. Focusing more on the user interface rather than the recommender algorithm itself, I learned that the radar graph made people feel like they were given better songs than a slider, so I will use that technique in my user interface for the recommender system. I also found the inputs to be interesting, rather than songs the authors utilized Artists, which makes sense because you can only determine genre from artist, not from track. Ideally, the user interface for our design will utilize either the current listener’s artists, or it will allow them to manually enter them. As for what attributes they found most important, they focused on energy, acousticness, danceability, instrumentalness, tempo, and valence.

Takeaway:Use of a radar chart for user selection of energy, acousticness, danceability, instrumentalness, tempo, and valence makes users feel like the recommendations presented to them are better.

Topic 2: Factors Affecting Music Recommender Success

This paper discusses various factors that can influence a person’s musical taste. One of the first factors discussed is geolocation, which was also mentioned in the first paper. It seems that it may be useful for our recommender to access the user’s country to help find songs that match their preferences. The paper also discusses people of similar ages often share the same music taste because there is a higher chance they heard the same music at the same time during their development years prior to turning 24. If we are able to utilize other user’s music preferences, we could use age as a way to tie similar users together. Tempo also showed to be a large factor in determining if people like recommended songs. We should pay attention to tempo and potentially give it a greater weight in the recommender algorithm. One warning that this paper discusses is accurate genre classification, since there are so many. We will be using Spotify as a sole source of genre classification, so we should investigate how this is done and check for consistency.

Takeaway: Use geolocation, give higher weight to tempo in recommender, investigate consistency of genre classification using Spotify.

Topic 3: Efficient Hybrid Music Recommender System

The authors in this paper sought to solve a similar problem to the one we are trying to solve – providing recommendations of songs that are not just the most popular for each genre, but also match user preferences. They discuss how collaborative filtering tends to keep recommending the same popular artists for each genre, since they have the most listeners. By going beyond this, users can discover new artists and new music that may not be popular, but match their music tastes. Their model unifies collaborative and content-based data utilizing probability theory. Their data sets correlated signal processing theory to get track / artist features with the sales and popularity of the track / artist on e-commerce sites – truly an interesting approach basing popularity off of actual sales. All of this is done incrementally, so the model builds itself dynamically. The advantage of this approach, is that it took 10 minutes to train the base model, and only 5 seconds to update the model. So if we can train a model similar to this, and dynamically update latent variables, we could have a recommender system that requires very little computational cost, but maintain high accuracy and processing speed.

Takeaway: Consider using probability theory to train a base model, so that incremental updates require little processing power and could be done on something as lightweight as a mobile device. Content based filtering may be the better approach to take if just using available Spotify data.

Topic 4: Music Recommendation using Internet Radio Streams

This paper seeks to solve the “rich get richer” problem with recommender systems that typically rely on the usage of the service to improve recommendations. For example, millions of people use YouTube and Spotify to listen to music, so how can a new streaming app compete with the billions of track plays and ratings that these services have access to? To answer that question, they used internet streaming playlists on radio stations to jump start their recommender system. An issue I see off this initially is that internet radio is heavily disproportionate in the variety of music it offers. For instance, there could be hundreds of stations that play popular genres of music like pop, but only a handful of less popular genres like metal and classical. Handling new users with no data, is done by utilizing popular music found on the internet – not a bad approach.

Takeaway: For users new to Spotify, or users that have below a predetermined quantity of music listened to, we could default recommendations to some of the most popular playlists on Spotify at the time the user accesses our app.

Topic 5: Different Users Require Different Interaction Methods

This paper focusses on the relationship between five interaction methods (how we can use the recommender system) and three user characteristics (the types of people using the system). They suggest that the best way to interact with a UI is based on the type of person using it, and their experience in the subject that is being recommended / how it is being recommended. What they found is that the five interaction methods really make no difference on how people feel about the recommendations. Rather, they find that certain types of people prefer certain interaction methods. The authors offer great recommendations, which I will copy and paste below because I find them THAT useful:

Designers seem to have to find a way to combine the simplest method (TopN) and the most complex method (Hybrid), while avoiding their respective downsides. Combining these methods in a single recommender system is a difficult design challenge. One option is to spatially separate the two interaction methods in different sections of the system. Another option is to temporally separate them: start with the TopN, carefully introduce implicit recommendations, and then introduce explicit controls as well. A final option is assign the correct method to each user: try to discover before (or during) the interaction what the user’s characteristics are, and then tailor the interface to [their] specific needs.

More generally, we show that designers and researchers alike should investigate the impact of their system on the user’s satisfaction in terms of both process and outcome. Being satisfied with the system itself and the outcomes of using it are two separate concerns, which may at times even be in conflict with one another.

Takeaway: Allow flexibility in the user interface and recommendation system, so that the user can not only tailor their recommendations, but also their interaction.

**Topic 6:**

Sakthi Lit review

**Topic 7:**

Sakthi Lit review

**Topic 8:**

Sakthi Lit review

**Topic 9:**

Sakthi Lit review

**Topic 10:**

Sakthi Lit review

DATA

This section will discuss the data used for our recommendation model, as well as some findings from utilizing the Spotify API to generate a data set. We were unsuccessful in generating our own data set, but the lessons learned would make this possible for future users. Efficient calls to the API are required, due to data query limits – we will discuss ways to generate a more efficient query utilizing the python package Spotipy.

Data Origin

Chart, bar chart, histogram

Description automatically generatedThe dataset used for this project was sourced from Kaggle and contains approximately 170,000 songs from the year 1920-2020. The track release date is skewed left, with the majority of years releasing 2000 tracks each year. This provided a good diversity of tracks by year from 1950-2020. For that reason, our data recommendation accuracy is best for recommended tracks released after 1950. See Figure 1 for this distribution.

Feature selection was done automatically using a signal processing algorithm developed by Spotify. This algorithm will take a track and analyze its associated audio wave forms and extract numerical values that represent certain track features. Spotify has named these Energy, Liveness

Danceability, Acousticness, Speechiness, Tempo, and Instrumentalness.

Based on correlation and applicability to our selected date range of song recommendations, instrumentalness was not included in our recommender. It was found that diversity in instrumentalness declined after 1960, which made sense because high values of this field typically indicated classical style music like Opera. Given the vast majority of new music being produced and listened to is not in this field, it was omitted to help give better recommendations for the music genres we were interested in.

Additional Data

Provided with the dittAudio recommender app is a data set chosen for advanced users. This data set lists over 3000 genres with corresponding values for all input parameters needed for recommendations using the dittAudio algorithm. Given more time, it was originally desired to incorporate this as a drop-down feature in Step 1 of the GUI, but we were not able to meet this functionality prior to the end of life of the project. Future work should aim to include something like this, so that recommendations can be provided based on genre as well as features from the existing user. This could also provide a way for non-premium Spotify users to utilize this app.

Spotify API Dataset Generation Findings

An original goal of the dittAudio project, was to dynamically build a training set of data upon first run of the application. This would require a Spotify premium account and would query data from a handful of genres and extract the audio features into a file format similar to the existing data.csv file in the repo. The difference would be that it would allow more dynamic recommendations, in that the seed track library would not be a static list of songs. As time progressed, there would be no need to supply new tracks. Due to inefficiencies in the original attempts, the Spotify API would lock the application due to excessive query requests. The following lessons learned should be helpful to those looking to accomplish this task using the Spotipy package, or directly querying the Spotify API.

1. Pack as many URIs as you can into each query.
   1. Our downfall was using a unique audio\_features request for each track. You can pass multiple URI’s into an audio\_features request, and get all the audio features in the query response in the same order you requested them.
2. Utilize a tab delineated output file, xml, or json. Tabs are not found in any track names or artist names, so this is a safe delimeter.
3. Searches are limited to 50 responses. Be sure to utilize the “offset” parameter to get the next page of results.
4. Limit scope to meet your application needs. It is important because users will be prompted with what your app needs access to based on your selection of scope at the creation of the OAuth object.
5. Genres are not listed by track, but by artist.
6. The URI is universally required for artists, playlists, tracks.. everything. Be sure to keep this value in all query truncations.

THEORETICAL DEVELOPMENT/MODEL

Sakthi’s write up on development of the model and hyper parameter tuning / justification for selected method.

Model Selection

Sakthi model selection – why did we pick mean vector / kmeans

Hyper Parameter Tuning

Sakthi explanation for hyper parameter selection

Model Workflow

Optional section, but I think we should include this graphic or something similar!

Diagram

Description automatically generated

Model Evaluation

Sakthi section on model evaluation, how good are the recommendations that come out of the model?

GUI DEVELOPMENT

Graphical user interface, text, application, chat or text message

Description automatically generated

The GUI was developed using the built-in python 3.9 package tkinter. Considerations were made for both simplicity to the user and making modified recommendation requests. This took the shape of a three-step recommendation workflow. First the user can select how to provide seed information to the recommendation algorithm.

Step 1: Seeds

The first seed selection option is using followed artists. This will take the first 10 followed artists by a user and query each artists most popular 10 tracks. It will extract the audio features for all the tracks that are found, average all of the found features together, and write the results into the boxes in step 2.

The second seed selection option is a user’s most listened to tracks. This will query the user’s profile for the most listened to songs, extract the audio features, and write them to Step 2.

The third seed option is a user’s top artists. This will perform similarly to the first option, just with a different list of artists.

Intent is to make the seed selection simple for all users. Advanced user’s have the option in step 2 to modify factors for more refined recommendations.

Step 2: Modifications and Recommender Selection

The optional modification of audio features is designed for advanced users. This should be expanded upon to allow a graphical usage. According to research, a radar chart should be used in place. Some scaling may need to be applied to the audio features. For this application, it is left as text boxes, which can be modified as needed. The additional genre data set could be used to fill in these boxes without selecting a seed, however seeds are required when using the Spotify recommendation method.

The user can select which recommendation algorithm to use to generate recommended tracks. The hipster toggle can be selected to return only songs with a popularity less than 10. The threshold can be changed in the GUI code.

Step 3: Review Recommendations

This is where the user can play a track on one of their active Spotify enabled devices. They can also request more recommendations or add the track to a playlist so that it can be listened to later. Expansions could be done here to use playlists as a seed for step 1.

This GUI As A Building Block

This simple GUI is meant to provide a baseline for testing and launching a new recommendation algorithm. The GUI backend utilizes a simple means for communicating with recommendation algorithms. It will send audio features in the form of a python dictionary. This would be the required input for the recommendation model. From there, the recommendation model should be built as a python class that can be imported to the GUI and should return a dictionary of recommended songs, where the keys are track URIs and the values are the track name and artist separated by a dash.

The GUI has some built in functions that can return track URI lists, artist URI lists, and more. These could also be used as seeds for a recommendation model, not just a dictionary of audio features. It is this versatility that makes the GUI such a good start for those building new recommender algorithms.

CONCLUSION

STILL NEED THIS

NEXT STEPS / FUTURE WORKS

This section will list some ideas for how to improve and build upon the existing open source GUI and overall recommender system interface. Rather than go into detail, it will be a list that includes some insights from current industry research that was not able to be applied to this project, but should be considered if looking to develop this further.

1. A radar chart for step 2 instead of text based entry
2. More customizable seed options
   1. Playlist based
   2. List of artists / tracks / playlists
   3. Genre – utilizing a drop down and existing data set for genres
3. A like / dislike button on step 3, that will influence future recommendations
   1. Also link this to the like feature on Spotify
4. Clean the GUI.py code – separate out GUI functions from the GUI builder code
5. Continue considering common data formats for sharing information across recommendation algorithms – be forward thinking.
6. Utilize json / xml packages in python to clean up field selection. Currently using it as a dictionary which gets messy and requires a lot of inefficient loops
7. Extend beyond audio and spotify – Netflix, YouTube, Steam, Discord … allow more than one API connection, allow users to select what usage they want. Tie different account information to each recommendation algorithm. Think big!

REFERENCES

Alphabetical Order

Aizenberg, N., Koren, Y., & Somekh, O. (2012, April). Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st international conference on World Wide Web* (pp. 1-10).

[Build your own music recommender by modeling internet radio streams (acm.org)](https://dl.acm.org/doi/pdf/10.1145/2187836.2187838?casa_token=eXLDE7i7Y0kAAAAA:QBQKmYLk4uXHJr36uYcVnFfJYgCarDWR2NpaUpXUf8jGUB4teSt0sLdnTOYqQdidfGUdVpywEA2Flg)

Knijnenburg, B. P., Reijmer, N. J., & Willemsen, M. C. (2011, October). Each to his own: how different users call for different interaction methods in recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 141-148).

[Each to his own: how different users call for different interaction methods in recommender systems (acm.org)](https://dl.acm.org/doi/pdf/10.1145/2043932.2043960)

Millecamp, M., Htun, N. N., Jin, Y., & Verbert, K. (2018, July). Controlling spotify recommendations: effects of personal characteristics on music recommender user interfaces. In *Proceedings of the 26th Conference on user modeling, adaptation and personalization* (pp. 101-109).

[Controlling Spotify Recommendations: Effects of Personal Characteristics on Music Recommender User Interfaces (acm.org)](https://dl.acm.org/doi/pdf/10.1145/3209219.3209223)

Uitdenbogerd, A., & Schyndel, R. (2002). A review of factors affecting music recommender success. In *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Proceedings* (pp. 204-208). IRCAM-Centre Pompidou.

[A review of factors affecting music recommender success (exlibrisgroup.com)](https://ap-st01.ext.exlibrisgroup.com/61RMIT_INST/upload/1666274280638/n2002001461.pdf?Expires=1666274402&Signature=ju819oowokU31zCj3Q-H~u-tjr-A3OA64kER0UGUxGLRPJhfh5vrmSNfoNchXM5fMml3dt~iOVHQ69gNFYs1jo79tkxuMDMex26fled3kaXXNoQXgwRNPbbXZlXrR7pp6U0aaji1eF4wsXTMCe2n0YJk5cfmUEhkAZF-Q1pyHgIoRct-o6Kee3BcuUFfLcuGR0MgsEXtPRrYTQHeHhQnNwr6WQzEg4FGmW0ynQoaEiAjShH6FK34mWjIQtWwljKAX15FshYbYiWQczy4bVxV~QnWruV430JkVgQOpyl4Pl5Rncv4qneK7TDw4n6nlpaKpyU2ohNfanmq-jcA~Ia6nw__&Key-Pair-Id=APKAJ72OZCZ36VGVASIA)

Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2008). An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. IEEE Transactions on Audio, Speech, and Language Processing, 16(2), 435-447.

[04432655.pdf (kyoto-u.ac.jp)](https://repository.kulib.kyoto-u.ac.jp/dspace/bitstream/2433/50284/1/04432655.pdf)

**GUIDELINES TO FOLLOW**

Here are more guidelines regarding Full Paper submissions. Please follow these carefully. Failure to do so may prevent your paper from appearing in the conference proceedings.

**File Size for Uploading**

The size of a PDF file to upload should not exceed 1MB. Large PDF file sizes are created when your file has many pictures and graphics, so be carefully with the inclusion of pictures and graphics into your Full Paper submission.

**Default Style in Word**

The default style in Word is not compatible with the guidelines in this document, so be sure to change the font type = Arial, font size = 11, line spacing = 1.0 with no spacing before or after paragraph, 1 in margins through, paper size = letter (8.5x11 inches), justify = left.

**Length**

Normal page length for articles submitted to conferences is about 6 to 35 pages, inclusive of all figures, tables, and references. Please try to maintain this standard. Excessively long papers are likely not to be carefully reviewed and, in the worst case, to be rejected.

**Page Numbers**

Do not create any page numbers. Page numbers will be added during the production of the program booklet.

**Language**

Your PDF file may contain traditional English or Greek characters. It should NOT contain any other language characters, such as Korean or Chinese (we mention these two languages because they have caused problems in the past).

**Copyright**

Material printed in its entirety in the conference proceedings is considered published. The copyrights for all forms of presentation at the annual meeting remain with the authors.

Do NOT submit anything to the proceedings that is already copyrighted. Authors assume full responsibility for what is submitted and for any and all copyright violations.

**Author Information**

Do NOT include author information. Once your Full Paper has been accepted, you will get a chance to include author information for resubmission.

**Figures and Tables**

Figures and tables should be placed as close as possible to where they are cited. If you are using symbols, please make sure that they are still in correct forms after you convert your file to Acrobat PDF. Please feel free to include figures, graphs, pictures as long as the total size of your file does not go beyond 500K.

**Calling References**

Use parenthesis around the last names of the referenced author along with the published year of the referred publication separated by commas. Examples: (Davenport & Harris, 2007; Hopkins et al, 2010).

**Footnotes**

Do NOT use footnotes. Instead use [text text text] where you may need to use a footnote.

**Equations**

All equations should be placed on separate lines and numbered consecutively, with the equation numbers placed within parentheses and aligned against the right margin.

|  |  |
| --- | --- |
| R1=α (X1) +β (X2) | (1) |

**Appendices**

Appendices should immediately follow the body of the paper (using the first-level heading "APPENDIX") and precede the references. If there is more than one appendix, number each one consecutively.

**References (Bibliography)**

Since the bibliography should include only those references cited in the text of the paper, it should be referred to as "references." References should be listed at the end of the paper. Entries should appear in alphabetical order. Examples of different types of entries:

Davenport, T. H., & Harris, J. G. (2007). *Competing on analytics.* Boston: Harvard Business School Press, 46.

Elliott, T. (2012). 2012: The year analytics means business. Retrieved from smartdatacollective.com, February 10.

Evans, J. R. (forthcoming). *Business analytics: Methods, models, and decisions.* Prentice-Hall.

Liberatore, M. J., & Luo, W. (2010). The analytics movement: Implications for operations research. *Interfaces, 40*(4), July-August, 313-324.

Hopkins, M. S., LaValle, S., Balboni, F., Kruschwitz, N., & Shockley, R. (2010). 10 data points: Information and analytics at work. *MIT Sloan Management Review, 52*(1), Fall, 27-31.

All Full Papers should contain a list of references. In rare cases, this may not be possible (e.g., due to the file size limitation). When this rarity arises, please insert the phrase "References available upon request” after References.

Making Managerial Decisions in Your Firm in 2018 as a Sample of a Full Paper Submission to Help Authors Understand How to Format Their Full Paper Submission

**ABSTRACT**

We present research examining how managerial decisions are being made in your firm in the year 2017. These decisions are difficult ones. Sometimes, these decisions have to be driven from the top instead of letting them organically form. We analyze survey data from 500 mid-level managers in our part-time MBA program using regression analysis to present new insights. Please keep to 100 words.

KEYWORDS: Managerial decision making, Firm decisions, Decision theory, Survey research, Regression

INTRODUCTION

Managerial decisions in your firm in 2018 will be even more important than in the past. We need to do more research. Research to date is incomplete. This paper has the following sections . . .

LITERATURE REVIEW

Much work has been done in decision making by managers (Smith & Smith, 2010). This work can be reviewed from two streams: the individual stream and the group-consensus stream. Below, we provide a synthesis of the research in each stream as it relates to our research questions. Table 1 provides a summary.

|  |  |  |
| --- | --- | --- |
| Table 1: Summary of Literature Review | | |
| YEAR | REFERENCES | JOURNAL |
| 2010 | Smith & Smith | Decision Sciences |
| 2000 | Johnson et al. | Decision Theory |

The Individual Stream

This is research looking at how individual managers make decisions (Johnson et al., 2000). This understanding helps us to decipher how managers in your firm will be making decisions in 2018 . . .

The Group-Consensus Stream

This stream assumes managers make decisions as efforts to appease many people who he or she sees as being part of the group. The insights from this stream focus not on how individuals process information to make information but more on how individuals interact and respond to outside pressures and how these interactions and responses shape their decisions [There are exceptions but we do not discuss them here]. Interactions between managers and their group members and responses by managers to their groups are therefore a further refinement of how the research to date in this stream can be analyzed.

Managerial Interactions with Groups

More text about this . . .

Managerial Response to Groups

More text about this . . . Figure 1 is a tabular summary of the frequency of work in this stream.

Figure 1: Frequency of publications by year

* *Repeat formatting of sections and section titles until Full Paper is compete*

HYPOTHESES/MODEL

More text about this . . . use subheadings as appropriate.

METHODS

More text about this . . . use subheadings as appropriate.

RESULTS

More text about this . . . use subheadings as appropriate.

DISCUSSION AND CONCLUSIONS

More text about this . . . use subheadings as appropriate.

REFERENCES

References available upon request.