Final Report - Spotifynd

Introduction

Streaming now accounts for a large portion of revenue in the music industry. The rise of paid subscriptions and mobile applications means that more people have access to the music of their choice. Companies like Spotify have spent significant amounts of capital to retain and attract subscribers through music selection, analytics, and overall application experience. As this trend continues, companies that to continue to invest in their product and innovate will be able to reap the benefits of a lucrative business model.

Problem Definition

In this project, we present Spotifynd. The goal of the Spotifynd web application is to provide an interactive music exploration environment leveraging data from Spotify's API, advanced visualization techniques, and diverse similarity algorithms. The result is an environment which helps users navigate music based on their attributes while satisfying their acoustic palette. Please visit www.spotifynd.ml to try out the application for yourself.

Background / Survey

Music Recommender Systems

Music recommender systems usually evaluate the similarity of songs using two approaches: content analysis and collaborative filtering. Content analysis can come in variety of forms. For example, there have been many attempts to use Mel-Frequency Cepstrum Coefficients (MFCCs) *[R-1, R-2, R-3]* to be able to recommend songs based on a song's inherent properties. Other approaches have used lyrics *[R-14]* and names of playlists *[R-5]* to find associated keywords that then drive recommendations.

Collaborative filtering (CF) assesses the similarity of songs by observing the strength of a song's connection between networks of users *[R-18]*. Some approaches for CF involve scraping data from social networks to build user profiles and preferences *[R-5, R-8]*.

Spotify

Spotify uses advanced recommender systems to build playlists. In *[R-21]*, Johnson provides some insights on how Spotify uses CF to model the interactions between the users and songs they select. One example is Spotify's "Discover Weekly" *[W-1]*, which finds songs from coinciding playlists based on the user's profile. Through the Spotify Analyzer *[D-1],* they can convert attributes like timbre, rhythm, and pitch to characteristics like danceability, liveness, and acousticness that can be used to compare between songs *[R-7]*.

Spotify has found great success through this process which has led to mass adoption. However, that does not mean the method has been perfected. One drawback is that the process is proprietary, making it difficult to analyze *[R-12]*. Additionally, it can take time and significant user interaction before results are satisfactory *[R-16]*. We believe all of these aspects can be improved on.

Proposed Method

Our unique offerings in this project are three-fold:

- Use of k-d trees to efficiently return similar artists/songs with fewer resources and improved benchmarks.
- Use of data visualization to enhance the music exploration and recommendation experience over existing tabular applications.
- Improved user interface provides users with a sense of discovery and creation as they interact with the app to create their playlists.

The process of developing this application can be divided into three parts: gathering data, building a music recommendation system, and developing a visual exploration framework.

Gathering Data

We scraped the artist information and artist similarity networks for over 100,000 artists. We also identified these artists' most popular songs, totalling over 1,000,000 songs, and scraped audio features for each of those songs provided in the Spotify Web API.

The following are steps we used to request data using a python wrapper for Spotify's Web API:

First, we began by requesting artists' names and artists' unique identifiers in all playlists made by Spotify along with the main music trends (i.e. country, hip hop, etc.) and their associated playlists. This gave us an initial 30,000 artists to work with.

Next, we requested up to 20 similar artists for each of these 30,000 artists. To narrow down the list for relevant artists, we only kept artists that have in-degree greater than or equal to 3. We repeated this step for newly added artists until we reached 117,961 artists.

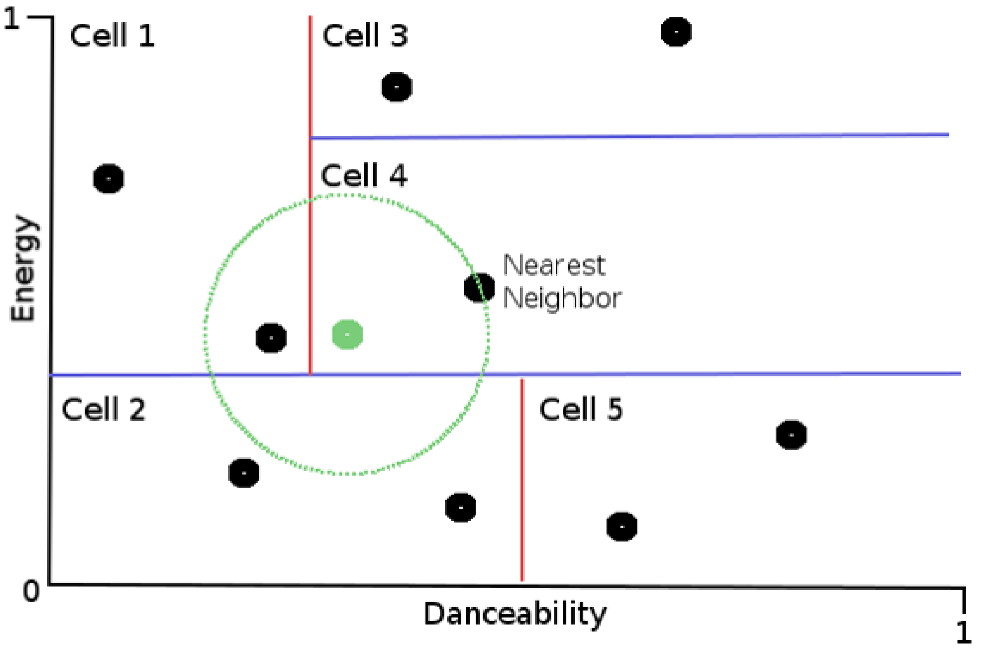
Finally, we requested each artist's top tracks and their associated features. The Spotify Web API responds with up to 10 top tracks per artist.

Music Recommendation System

Through the Spotify Web API, we have access to a variety of precomputed track-level features. Some of these are acousticness, danceability, energy, instrumentalness, loudness, speechiness, tempo, and valence. All these features except for tempo (measured in beats per minute) and loudness (measured in decibels) are continuous valued between 0 and 1. We plan to treat these seven features as a music track feature-vector in $\mathbb{R}^8$. To measure similarity between two tracks, we will primarily rely on Euclidean distance.

To avoid a computational bottleneck when calculating similarities between songs, we will use k-d trees *[R-17, R-22]* which offer a O(log n) lookup time. K-d trees expand the idea of binary search trees into the k-th dimension. Binary search trees are used to search for points in $\mathbb{R}$, whereas K-d trees can be used to search for points in $\mathbb{R}^n$. The algorithm begins by dividing the data according to the first dimension, the second split is determined by the second dimension, the third split is determined by the third dimension, and so on. K-d trees are commonly used as the underlying data structure in Nearest Neighbor search algorithms since they effectively and efficiently segment the dataset into groups with similar metrics across all dimensions.

Figure 1: K-d Tree 2D Example



In addition to making recommendations based on track features, we also make recommendations based on genre. For each artist in our dataset, we have a list of genres associated with that artist. This data, coupled with the track feature data, enables us to make recommendations in two ways. On the one hand, with the k-d tree of music similarity we can recommend similar songs independent of the genre or artist. On the other hand, with lists of genres and similar artists, we can recommend similar songs independent of the specific song features.

With these independent approaches, we ensured that our different methods of recommending songs play well with each other so the strengths of each can be leveraged at the same time. For example, our application can answer queries such as:

- What songs in the indie-folk genre has valence most similar to We Are the Champions by Queen?
- Independent of genre, what songs have song features closest to the song features of Don’t Stop Believin’ by Journey?
- What are some of the most popular songs by artists similar to Taylor Swift?

User Interface / Visual Exploration

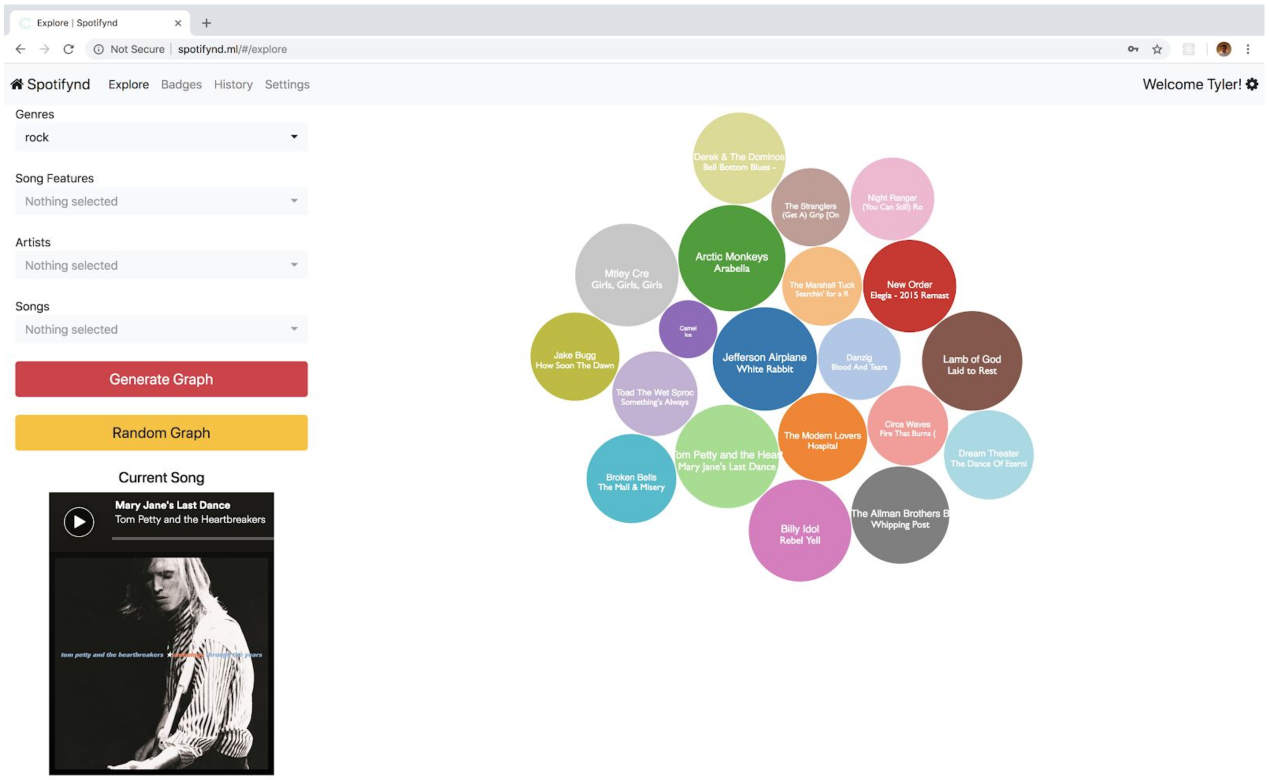
The responsive user interface is built using Aurelia framework, bootstrap 4, and utilizes a restful Express API. This responsive user interface works well on tablets and mobile devices. D3 Data visualizations combined with a well designed visual feature taxonomy *[R-13]* provide users with an interactive and enjoyable exploration environment. We implemented the following interface workflow to facilitate this experience.

User Experience Workflow

1. The user can login if they are signed up. If they are not signed up then they have the ability to sign up for a user account on the Spotifynd application.
2. The user is presented with a random collection of songs. At any time, the user may hit "Random Graph" to reset all selections and start fresh with a new graph.
3. The user has options to filter based on genre, similar artists, and/or similar songs based on song features.
4. The user clicks "Generate Graph" to send all seeds and preferences to our Music Recommendation Engine.
5. The application uses the recommendation results to generate a song cloud visualization.
6. Clicking on song nodes allows user to listen to 30 second sample of songs

7. The artists and songs selection menus have now been populated with the data from the current cloud visualization. Once a user has found a song he/she is interested in, they can use the application to then find songs similar to the new song they just discovered.

Figure 2: Application User Interface


Explore | Spotifynd
Not Secure | spotifynd.ml/#/explore
Spotifynd
Explore
Badges
History
Settings
Welcome Tyler!
Generate Graph
Random Graph
Current Song
Derek & The Dominos Bell Bottom Blues -
The Stranglers (Get A) Grip (On
Night Ranger (You Can Still) Ro
Arctic Monkeys Arabella
Mtley Cre Girls, Girls, Girls
The Marshall Tuck Searchin' for a R
New Order Elegia - 2015 Remast
Camel Ice
Jake Bugg How Soon The Dawn
Jefferson Airplane White Rabbit
Danzig Blood And Tears
Lamb of God Laid to Rest
Toad The Wet Sproc Something's Always
Circa Waves Fire That Burns (
Dream Theater The Dance Of Eterni
The Modern Lovers Hospital
Tom Petty and the Hear Mary Jane's Last Dance
Broken Bells The Mall & Misery
Billy Idol Rebel Yell
The Allman Brothers B Whipping Post

Genres: rock

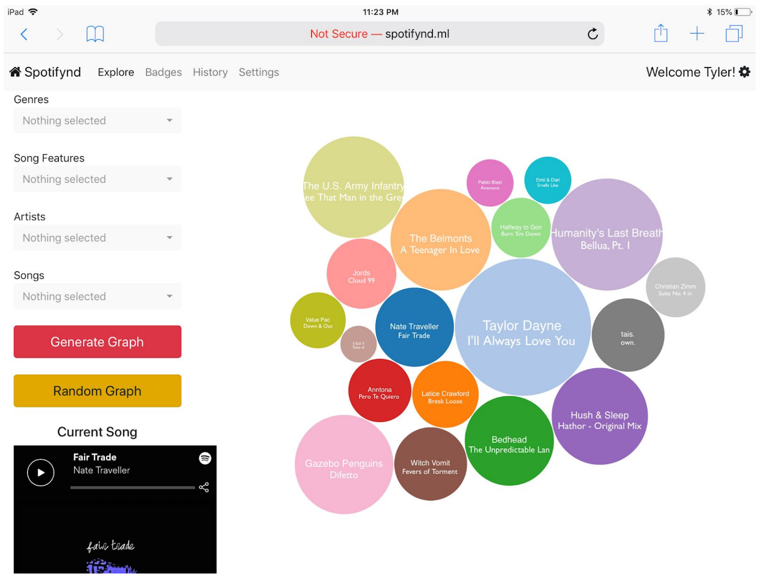
Song Features: Nothing selected

Artists: Nothing selected

Songs: Nothing selected

Current Song: Mary Jane's Last Dance — Tom Petty and the Heartbreakers

Responsive View (Tablet and Mobile)


iPad
11:23 PM
15%
Not Secure — spotifynd.ml
Spotifynd
Explore
Badges
History
Settings
Welcome Tyler!
Generate Graph
Random Graph
Current Song
The U.S. Army Infantry ee That Man in the Gre
The Belmonts A Teenager In Love
Humanity's Last Breath Bellua, Pt. I
Jordis Cloud 99
Taylor Dayne I'll Always Love You
Nate Traveller Fair Trade
tais. own.
Antoona Pero Te Quiero
Lattice Crawford Break Loose
Bedhead The Unpredictable Lan
Hush & Sleep Hathor - Original Mix
Gazebo Penguins Difetto
Witch Vomit Fevers of Torment
fair trade

Genres: Nothing selected

Song Features: Nothing selected

Artists: Nothing selected

Songs: Nothing selected

Current Song: Fair Trade — Nate Traveller

Spotify Integration

Spotify generously extends their connect player for use through their web playback SDK. We embedded the player in the Spotifynd application. As the user clicks a song node, the song information is passed to the Spotify connect player to play a sample of the song. This feature was essential in adding an audio component to the exploration experience.

Evaluation

The main areas for evaluation include:

- Accuracy of predictions
- Quality of user experience
- Meaningful contribution to music exploration

User Surveys

Since the goal of the application is to improve the user's experience in music exploration, we decided to get user feedback through structured surveys to help evaluate and guide application development. Because music recommendation is an unsupervised machine learning problem, we rely on these survey questions to help us determine how effective our predictions are. The main objective of the questions was to compare the application with Spotify or Apple Music which are two of the largest music applications available.

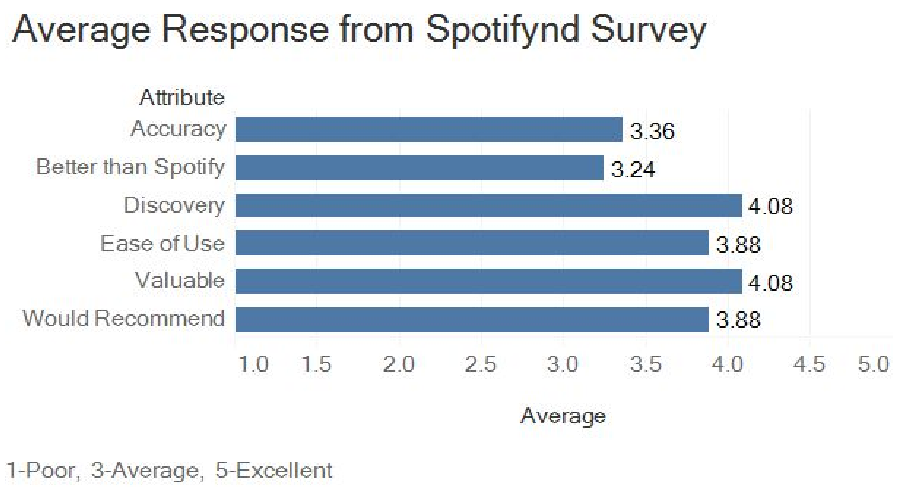


Figure 3: Average responses evaluating attributes of Spotifynd.

The list of survey questions are presented below and have a numerical response from 1 to 5.

- Does this application help you make relevant playlists according to your preferences?
- Does this application help you discover music?
- Is this application easy and intuitive to use?
- Are the similar songs recommended by the application accurate and helpful?
- Is the application experience enjoyable and worthwhile?
- With respect to discovering new music, how does this application compare to Spotify?
- Would you recommend this application to your friends?

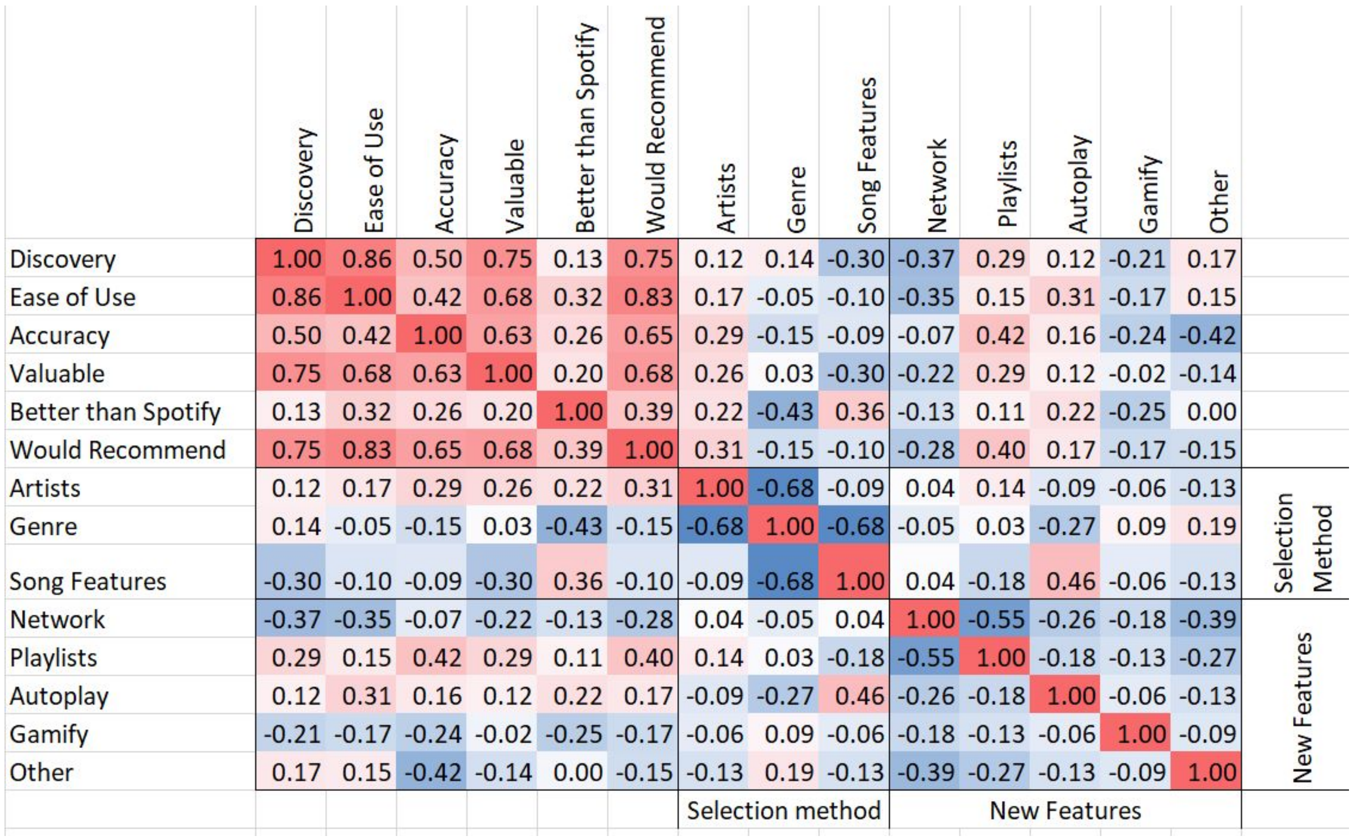
In addition, we had questions evaluating the search options and future areas of improvement.

- What method of filtering did you prefer using? [Genre, Song Features, Artists]
- What feature do you want the next version to have? [Under future works]

Discussion

Correlation Observations

Figure 4 - Survey Response Correlation Matrix



	Discovery	Ease of Use	Accuracy	Valuable	Better than Spotify	Would Recommend	Artists	Genre	Song Features	Network	Playlists	Autoplay	Gamify	Other	
Discovery	1.00	0.86	0.50	0.75	0.13	0.75	0.12	0.14	-0.30	-0.37	0.29	0.12	-0.21	0.17	
Ease of Use	0.86	1.00	0.42	0.68	0.32	0.83	0.17	-0.05	-0.10	-0.35	0.15	0.31	-0.17	0.15	
Accuracy	0.50	0.42	1.00	0.63	0.26	0.65	0.29	-0.15	-0.09	-0.07	0.42	0.16	-0.24	-0.42	
Valuable	0.75	0.68	0.63	1.00	0.20	0.68	0.26	0.03	-0.30	-0.22	0.29	0.12	-0.02	-0.14	
Better than Spotify	0.13	0.32	0.26	0.20	1.00	0.39	0.22	-0.43	0.36	-0.13	0.11	0.22	-0.25	0.00	
Would Recommend	0.75	0.83	0.65	0.68	0.39	1.00	0.31	-0.15	-0.10	-0.28	0.40	0.17	-0.17	-0.15	
Artists	0.12	0.17	0.29	0.26	0.22	0.31	1.00	-0.68	-0.09	0.04	0.14	-0.09	-0.06	-0.13	Selection Method
Genre	0.14	-0.05	-0.15	0.03	-0.43	-0.15	-0.68	1.00	-0.68	-0.05	0.03	-0.27	0.09	0.19	
Song Features	-0.30	-0.10	-0.09	-0.30	0.36	-0.10	-0.09	-0.68	1.00	0.04	-0.18	0.46	-0.06	-0.13	
Network	-0.37	-0.35	-0.07	-0.22	-0.13	-0.28	0.04	-0.05	0.04	1.00	-0.55	-0.26	-0.18	-0.39	New Features
Playlists	0.29	0.15	0.42	0.29	0.11	0.40	0.14	0.03	-0.18	-0.55	1.00	-0.18	-0.13	-0.27	
Autoplay	0.12	0.31	0.16	0.12	0.22	0.17	-0.09	-0.27	0.46	-0.26	-0.18	1.00	-0.06	-0.13	
Gamify	-0.21	-0.17	-0.24	-0.02	-0.25	-0.17	-0.06	0.09	-0.06	-0.18	-0.13	-0.06	1.00	-0.09	
Other	0.17	0.15	-0.42	-0.14	0.00	-0.15	-0.13	0.19	-0.13	-0.39	-0.27	-0.13	-0.09	1.00	
							Selection method			New Features					

In *Figure 4*, by comparing the responses between questions we observed associations on qualities that we would like to build on the product. We used 0.8 as a marker for high correlation for the 1-5 response questions and 0.4 for categorical questions. This is due to the pre-existing strong correlations between the 1-5 response questions. One possible reason for this pre-existing correlation is that the people have a tendency input the same score.

We can see that the ability for users to visualize the song selections can enable them to better discover new songs and from their experience, would be more likely to share the app with their friends. We also found that people who have found the application accurate and provided a good experience, were more likely to recommend a playlist as a new feature.

Conclusions

Users indicated they enjoyed the visual exploration experience which was one of the main goals of the project. User's were able to explore songs by artists, genre, and song features efficiently. The application response time for user's requests well which can be attributed to the effective use of k-d trees to reduce latency.

Some users indicated that the recommendations were not as helpful which may be attributed to our lack of using collaborative filtering. This was especially true if a user did not select a genre for recommendations in which case some of the songs returned by the similarity algorithm could be of a completely different style or language. This may be why our users reported using the genre filter the most in the survey (*See Figure 5 in Appendix*). Spotify leverages all of their user's activity to segment users in addition to using song features and genres for improved recommendations. As more users use Spotifynd, we may be able to similarly incorporate user activity.

We received suggestions to include song release dates in the song title or in the visualization itself. User's expressed interest in being able to filter for similar songs by year especially for those that are looking for recently released songs they likely have not heard yet. We have received the most responses in implementing features to allow users to compare music network with other as well as creating their own playlists (*See Figure 6 in Appendix*).

Overall, we feel that the application methodology was successful. On average, the user scores in each characteristic were positive and many claimed to even prefer discovering music through our application over Spotify itself. Given that there are still features to be incorporated, we feel there is a great amount of potential in the methodology that is worth pursuing further.

Future Work

The following is a list of features and activities that we put together which may help improve the experience and direction of the platform:

Incorporate Additional Application Functionality

Maintain Playlists

With a user provided Spotify API token, the application can add or remove songs for a playlist in the user's Spotify account. This will help the user quickly save songs for later use.

User Profile Tuning

Well developed user profiles will be important since music exploration preferences can vary greatly among listeners. Methodologies in user profile tuning *[R-14]* will be valuable in adjusting the guided visual exploration and recommendation systems.

Autoplay

With user provided preferences, a handsfree music navigation feature for passive streaming can be used when the user is engaged in another activity like driving or exercising.

Share Activity Socially

If users can share their activity with each other, they can compare songs explored by friends or celebrities. It would also be easy to find other users with similar music interests.

Gamify the Application

Graph metrics can be used to give out badges or scores based on user activity.

- Graph Diameter - User music exploration score.
- Average Path Length - User music diversity score.

Additional Visualization Controls

- A node can be dragged to a seed group to refine the recommendation results.
- Double-clicking a node can extend the graph network further from the node so that the user can explore songs similar to it.
- With spotify integration for maintaining playlists, right-clicking on a node can provide options to add the song to a playlist.

Perform Additional Experiments

Network Overlap Experiments

A desirable outcome from this application is to create node networks that accurately model music preferences for the user. As such, a possible evaluation technique could include extracting existing playlists developed by Spotify or Apple Music and comparing groups of songs co-occurring together in both our playlists and their playlists. However, the amount of data necessary for an analysis like this may require having proprietary access to their data.

Sample Size Trade Offs

The amount of songs sampled to generate the song's selection provides a trade off between the speed of computation and the selection's collective similarity scores. By increasing the number of songs in the sample, we would expect the length of processing time to increase by O(log(n)), based on the k-d data structure [R-17]. The trend of similarity scores is expected be modeled by a logarithmic curve representing the amount of information [R-23] gained from each song. We may extend this experiment to consider the effect of the type of search (genre, artist, features) on these trends.

Feature Selection

There may be some attributes of a songs that are redundant to include in our application. To assess this, we first search for results from different attributes of the same song. We would evaluate the overlap generated by these attributes using the Jaccard Index with a random sampling of 1000 songs. This would provide us a distribution of these coefficients to find which attributes are closely related. If we take the average similarity score across these same songs and generate a distribution from these, we could also assess which attributes would provide more accurate and consistent results.

Project Task Distribution

All team members contributed a similar amount of effort. The task breakdown is displayed in the table below.

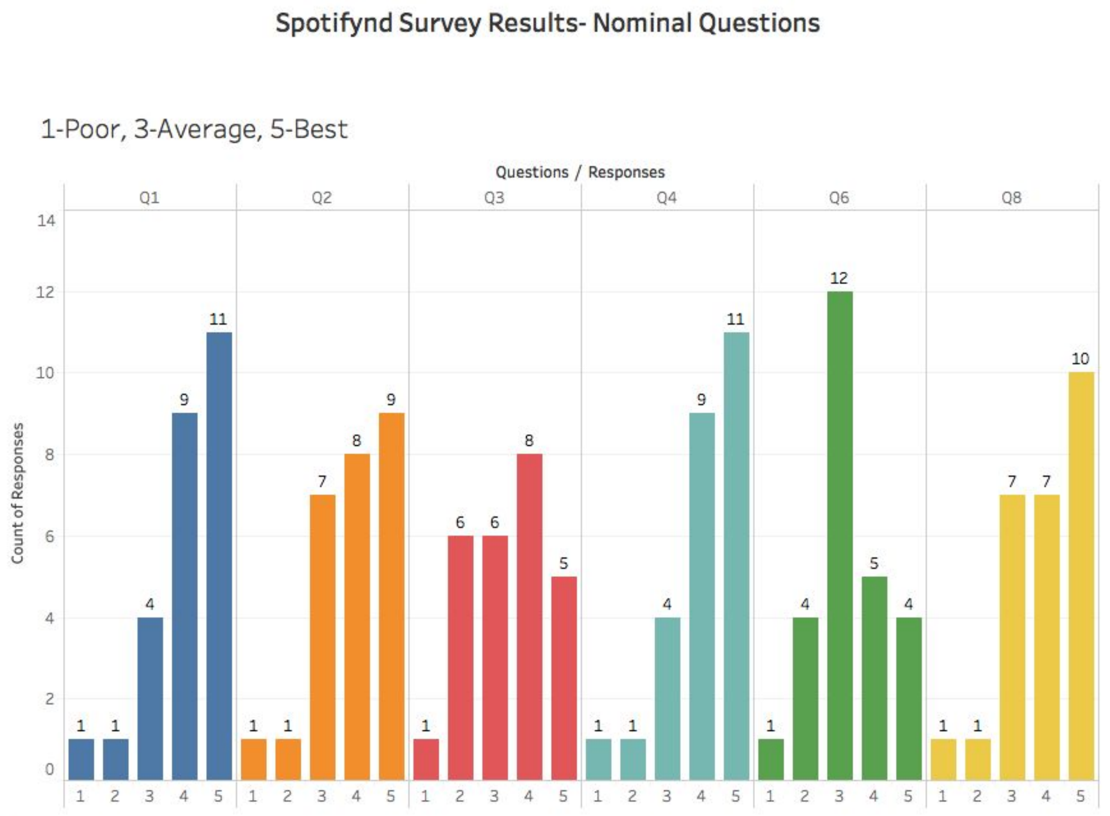
				Task Distribution				
Task	**Subtask**	**Description** (*Tools*)	**Days**	**TD**	**RR**	**NM**	**TC**	**KW**
Proposal	Report	Write-Up / Presentation	8	🟩	🟩	🟩	🟩	🟩
Setup	Environment	Server, Logins (*Github / AWS*)	1	🟩	🟩			
Data Processing	API Mapping	Retrieval Methods (*Python, Express Restful API*)	8	🟩			🟩	🟩
	DB Design	SQL Schema (*MySQL*)	2	🟩	🟩	🟩		
	DB Population	Data Loading, Cleaning, Maintenance (*Python / MySQL*)	2	🟩	🟩	🟩		
Progress	Report	Write-Up	6	🟩	🟩	🟩	🟩	🟩
Application Development	Algorithms	Similarity, Profiling (*Python*)	4		🟩		🟩	🟩
	Functionality	User Workflow (*Python, JS*)	4	🟩	🟩		🟩	
	User Interface	(*Aurelia, Webpack, Bootstrap 4 , d3*)	6	🟩				
	Production	Compiling Components (*AWS*)	2	🟩	🟩	🟩	🟩	🟩
Evaluation	Feedback	Surveys (*Google Forms*)	5	🟩	🟩	🟩	🟩	🟩
Final Report	Report	Write-Up / Poster	6	🟩	🟩	🟩	🟩	🟩
	Software	Packaging, Documentation	2	🟩	🟩	🟩	🟩	🟩

*TD - Tyler Dotson, RR - Rishi Rebello, NM - Nathan Miller, TC -Tanner Christensen, KW - Weiyang Wang

Appendix

Survey Results

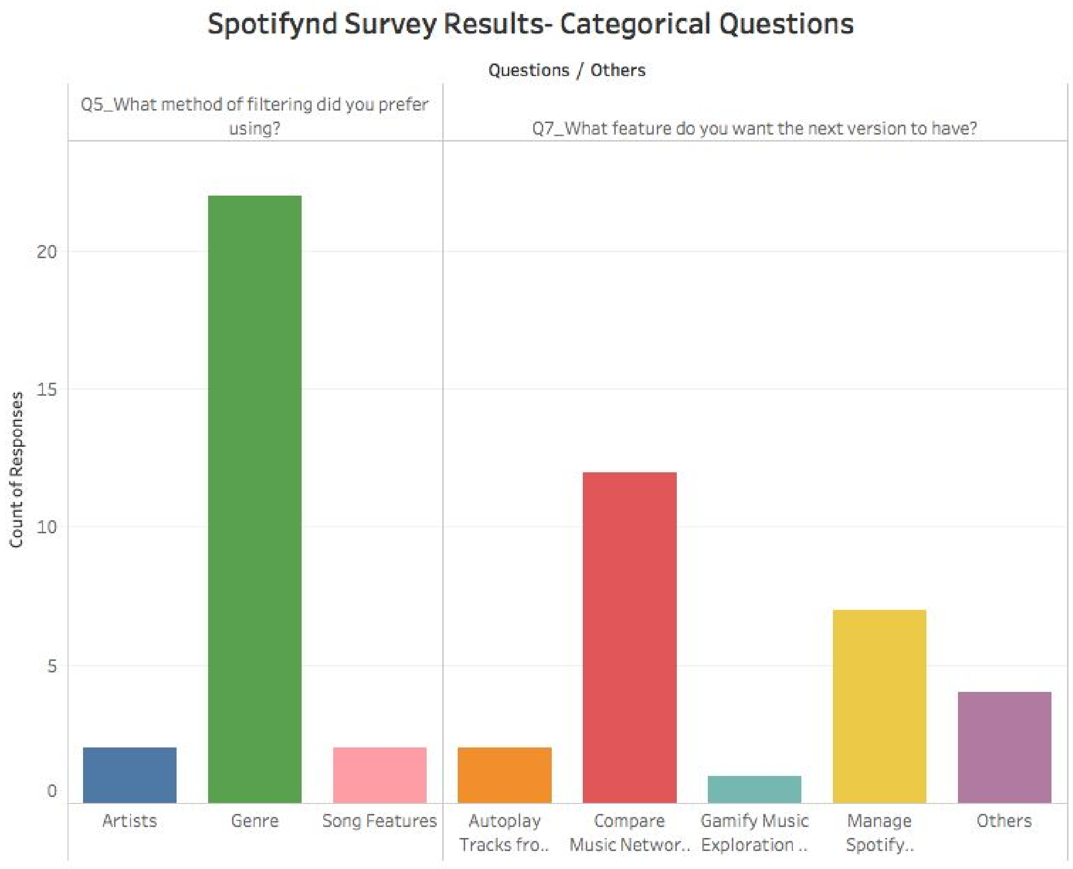
Figure 5



Questions Legend

- 🟦 Q1_Does this application help you discover music?
- 🟧 Q2_Is this application easy and intuitive to use?
- 🟥 Q3_Are the similar songs recommended by the application accurate and helpful?
- 🟦 Q4_Is the application experience enjoyable and worthwhile?
- 🟩 Q6_With respect to discovering new music, how does this application compare to Spotify?
- 🟨 Q8_Would you recommend this application to your friends?

Figure 6



Responses Legend

- 🟦 Artists
- 🟩 Genre
- 🟥 Song Features
- 🟧 Autoplay Tracks from your Music Network
- 🟥 Compare Music Network Across Friends and Celebrities
- 🟦 Gamify Music Exploration with Badges and Ratings
- 🟨 Manage Spotify Playlists and Library
- 🟪 Others

References

D. Documentation
1. Jahan T. (2014) Spotify Analyzer Documentation. Version 3.2. Echonest. http://docs.echonest.com.s3-website-us-east-1.amazonaws.com/_static/AnalyzeDocumentation.pdf

P. Programs
1. Lamere P. Boil the Frog. https://github.com/plamere/BoilTheFrog
2. Litven J. (2016) Explorify. https://jlitven.shinyapps.io/music_explorer/
3. Rieder B. Spotify Artist Network. http://labs.polsys.net/playground/spotify/

W. Web Articles
1. Le J. (2018) Spotify's "This Is" playlists: the ultimate song analysis for 50 mainstream artists. Towards Data Science. Medium. https://towardsdatascience.com/spotifys-this-is-playlists-the-ultimate-song-analysis-for-50-mainstream-artists-c569e41f8118
2. Pasick A. (2015) The magic that makes Spotify's Discover Weekly playlists so damn good. Quartz. https://qz.com/571007/the-magic-that-makes-spotifys-discover-weekly-playlists-so-damn-good/

R. Research Articles
1. Kim D., Kim K., Park K., Lee J. and Lee K. (2007) A music recommendation system with a dynamic k-means clustering algorithm. Sixth International Conference on Machine Learning and Applications. IEEE Computer Society. https://ieeexplore.ieee.org/document/4457263/
2. Bagci U. and Erzin E. (2007) Automatic Classification of Musical Genres Using Inter-Genre Similarity. IEEE Signal Processing Letters, Vol. 14, No. 8, August 2007. https://ieeexplore.ieee.org/document/1659788/
3. Celma O. and Cano P. (2008) From hits to niches? or how popular artists can bias music recommendation and discovery. 2nd Netflix-KDD Workshop. ACM. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.5009&rep=rep1&type=pdf
4. Lin N., Tsai P., Chen Y., and Chen H. (2014) Music recommendation based on artist novelty and similarity. 2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP). IEEE. https://ieeexplore.ieee.org/document/6958801/
5. Pichl M., Zangerle E., and Specht G. (2015) Towards a Context-Aware Music Recommendation Approach: What is Hidden in the Playlist Name? 2015 IEEE 15th International Conference on Data Mining Workshops. IEEE Computer Society. https://ieeexplore.ieee.org/document/6958801/
6. Aucouturier J. and Pachet F. (2003) Representing Musical Genre: A State of the Art. Journal of New Music Research 2003, Vol. 32, No. 1. Swets & Zeitlinger. https://www.tandfonline.com/doi/abs/10.1076/jnmr.32.1.83.16801
7. Kennelly T. and Berger P. (2018) How Loudness, Song Negativity and Playlist Personalization Can Increase Spotify's Customer Retention. EPH - International Journal of Business & Management Science. Volume 4, Issue 1. 2018 https://ephjournal.com/index.php/bms/article/view/447/356
8. Germain A. and Chakareski J. (2013) Spotify Me: Facebook-Assisted Automatic Playlist Generation. 2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP). IEEE. https://ieeexplore.ieee.org/document/6659258/
9. Stumpf S. and Muscroft S. (2011) When users generate music playlists: when words leave off, music Begins? 2011 IEEE International Conference on Multimedia and Expo. IEEE. https://ieeexplore.ieee.org/document/6012152/

10. Pachet F. and Roy P. (1999) Automatic Generation of Music Programs. In: Jaffar J. (ed). Principles and Practice of Constraint Programming - CP'99. Springer. USA https://link-springer-com.prx.library.gatech.edu/content/pdf/10.1007%2Fb72297.pdf
11. Tan S., Bu J., Chen C., He X. (2011) Using Rich Social Media Information for Music Recommendation via Hypergraph Model. In: Hoi S., Luo J., Boll S., Xu D., Jin R., King I. (eds) Social Media Modeling and Computing. Springer, London https://link-springer-com.prx.library.gatech.edu/book/10.1007%2F978-0-85729-436-4
12. Schindler A. and Rauber A. (2012) Capturing the Temporal Domain in Echonest Features for Improved Classification Effectiveness. 10th international workshop on Adaptive Multimedia Retrieval. LNCS, Vol 8382. Springer. http://www.ifs.tuwien.ac.at/~schindler/pubs/AMR2012.pdf
13. Nazemi K., Breyer M., Kuijper A. (2011) User-Oriented Graph Visualization Taxonomy: A Data-Oriented Examination of Visual Features. In: Kurosu M. (eds) Human Centered Design. HCD 2011. Lecture Notes in Computer Science, vol 6776. Springer, Berlin, Heidelberg https://link-springer-com.prx.library.gatech.edu/chapter/10.1007/978-3-642-21753-1_64
14. Mianowska B., Nguyen N.T. (2012) A Method for Tuning User Profiles Based on Analysis of User Preference Dynamics in Document Retrieval Systems. In: Rutkowski L., Korytkowski M., Scherer R., Tadeusiewicz R., Zadeh L.A., Zurada J.M. (eds) Artificial Intelligence and Soft Computing. ICAISC 2012. Lecture Notes in Computer Science, vol 7267. Springer, Berlin, Heidelberg https://link-springer-com.prx.library.gatech.edu/chapter/10.1007/978-3-642-29347-4_78
15. Cacheda F., Carneiro V., Fernandez D. (2011) Comparison of Collaborative Filtering Algorithms: Limitations of Current Techniques and Proposals for Scalable, High-Performance Recommender Systems. ACM Transactions on the Web, Vol. 5, No. 1, Article 2 https://www.researchgate.net/publication/220593852_Comparison_of_collaborative_filtering_algorithms_Limitations_of_current_techniques_and_proposals_for_scalable_high-performance_recommender_systems
16. Schein A., Popescul A., Ungar L., and Pennock D. (2002) Methods and Metrics for Cold-Start Recommendations. SIGIR 2002. Pages 253-260. ACM. https://repository.upenn.edu/cgi/viewcontent.cgi?article=1141&context=cis_papers
17. Bentley J.L. (1975) Multidimensional Binary Search Trees Used for Associative Searching. Communications of the ACM. Vol. 18, No. 9. ACM. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.335&rep=rep1&type=pdf
18. Herlocker J., Konstan J., Terveen L., and Riedl J. (2004) Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004, Pages 5–53. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.5270&rep=rep1&type=pdf
19. Su X. and Khoshgoftaar T. (2009) A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence. Vol 2009. Hindawi Publishing Corporation. https://www.hindawi.com/journals/aai/2009/421425/abs/
20. Brusilovsky P., Kobsa A., and Nejdl W. (2007) The Adaptive Web, LNCS 4321, pp. 291-324 Collaborative Filtering Recommender Systems https://link.springer.com/content/pdf/10.1007%2F978-3-540-72079-9_9.pdf
21. Johnson C. (2014) Logistic Matrix Factorization for Implicit Feedback Data. Advances in Neural Information Processing Systems. Vol 27. http://web.stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf
22. Maneewongvatana S., Mount D.M. (1999) It's Okay To Be Skinny If Your Friends Are Fat. 4th Annual CGC Workshop on Computational Geometry. https://www.cs.umd.edu/~mount/Papers/cgc99-smpack.pdf
23. Shannon, C.E. (1948) A Mathematical Theory of Communication. The Bell System Technical Journal,Vol. pp. 379–423, 623–656 http://math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf