Introduction

Our goal for the project was to create a podcast recommendation engine built off user generated content. We created a tool, GuidePod, that allows users to find podcasts they are most likely to enjoy based on the podcast they select, regardless of popularity. Engines such as iTunes and Spotify rely on collaborative filtering (CF), resulting in the most popular podcasts being recommended to most users, even if they are not the best fit. We also wanted to provide users with a visual experience and also give them insight into why each podcast was recommended, which is lacking from existing recommendation engines.

The Problem

In our research, we found that users have limited ways to find new podcasts: they typically need to rely on platform recommendations, along with manually searching and word of mouth. Other platforms' use of collaborative filtering leads to poor coverage of ratings (Chen et al., 2015) and neglecting smaller, but relevant podcasts. User generated content (such as reviews and ratings) is becoming increasingly popular in recommendation systems (Lops et al., 2019). As the podcast audience is enormous, this creates a sufficient amount of user generated content to tap into, as well as a large community to benefit from our tool. There are over 1 million podcasts and listeners spend an average of 6 hours and 39 minutes per week listening to an average of 7 different shows (Whitner, 2021). About half of Americans over age 12 have listened to a podcast (Chan-Olmsted & Wang, 2020).

Approach & Innovations

The main difference between our approach and the current state is that we rely on user generated content and episode details to build our model. In our research, we found that user feedback and reviews produce robust results in recommendation engines (Malik, 2020). We found that reviews have a U-shaped effect, where positive reviews are associated with higher enjoyment and negative reviews are associated with being more useful for users (Park & Nicolau, 2015). Our research revealed that a reviewer's profile characteristics can help fine tune a recommendation system, as the reviewer's recency of reviews, frequency of reviews, and monetary value of the product can help improve the accuracy of recommendations (Malk & Hussain, 2020).

In addition to our unique modeling approach, we also wanted to offer users a more transparent and visually appealing user experience, something most recommendation applications are lacking. Users can search for a podcast in a search box, which populates suggestions from a curated list of 17,500+ podcasts as they type its name. Once a podcast is selected, we present the users with an interactive network graph showing a network of most similar podcasts. The user can drag the nodes to the desired position, and there are tool tips to provide additional insights such as average rating, total review count, number of episodes, and similarity score (see Figure 1). The colored nodes are the top recommendations, and the size of the nodes represent how similar they are to the original podcast selection. The bigger the node, the more similar it is. The width of the links between nodes represent their own similarity score. The thicker the link, the more similar they are to each other. Network graphs can be very powerful in showing relationships in a large community of nodes, which other visualizations might have trouble presenting (Ifenthaler et al., 2018).

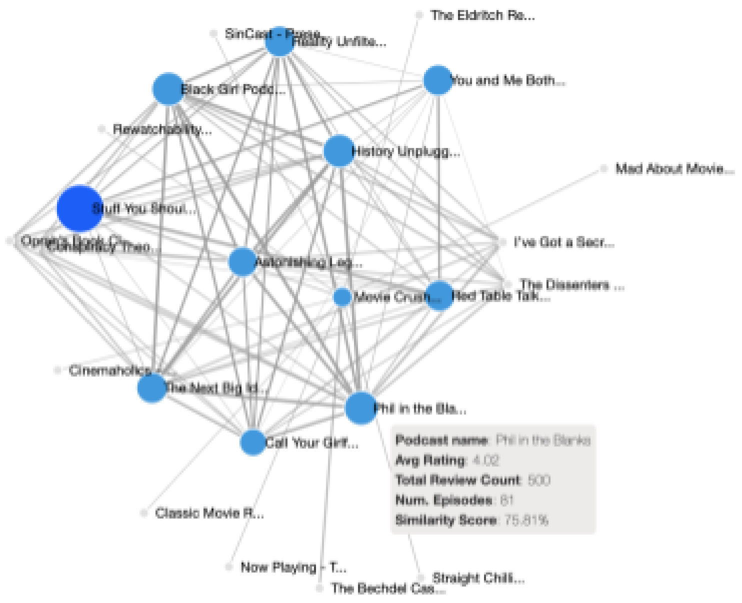

Figure 1. Network Graph representing top recommendations for podcast "Stuff You Should Know".

Unlike other podcast recommendation applications, we showcase the logic and the data behind why a certain podcast was recommended. When a user double clicks on any nodes on the network graph, we populate three additional visualizations (Figure 2). A word cloud with varying sizes provides a quick and easy to understand visualization on which words our model considered to be most relevant. The bigger the word, the more relevant it was to the podcast recommendation. A scatterplot to provide some additional context to the top recommendations, such as total duration (hours) of the show, the average rating, and the number of user reviews. And finally, a diverging stacked bar graph to help users visualize the breakdown of the user ratings.

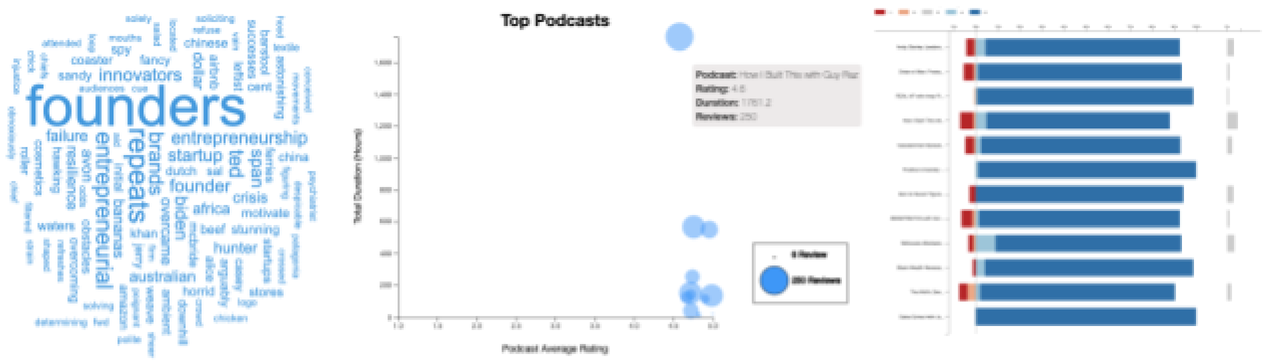

Figure 2. Additional visualizations providing context for podcast "How I Built This With Guy Raz": Word cloud (left), Scatterplot (middle), Diverging Stacked Barchart (right).

We used SQLite and Python to scrape, clean and store the data (Albrecht et al., 2021). The recommendation modeling was built with Python, and the front-end visualizations were created with HTML, CSS, and JavaScript D3 (Cui et al., 2016). To actually compile this data, we used Apple's API and RSS feeds to scrape podcast data, episode data, and listener reviews. We initially scraped Apple's website to get a list of 22,000 popular podcast IDs. Then we called Apple's API to retrieve general show data on those IDs: The data pulled included genre, country, reviews, and RSS feeds. Once we created a final list of podcasts, we scraped over 1 million user reviews from Apple's API; we then scraped episode details (e.g. guests, descriptions, episode duration, etc.) from the show's individual RSS feeds. We complete these steps using Beautiful Soup and ElementTree XML packages to process the structured data within the RSS feeds. Our final dataset, once expanded to countries outside the US, but still limited to English-only podcasts, included over 3.6 million unique data points.

Once we had all the data, we began to clean it. We encountered many issues typical of user generated content: typos, unique characters, irrelevant words, etc. We used Regex and the Python String package to clean and process the data. To fix typos, we used SymSpell as the primary autocorrect function. We tested other packages, but found SymSpell to have the largest word corpus to check against and was the most efficient in speed. To avoid tense variation and to reduce the amount of features, we used NLTK and Wordnet packages to lemmatize verbs into their base form. We then removed stop words (i.e. commonly used words in English) to further reduce the features in the model. In addition to these, we built a custom set of common podcast terms (e.g. "episode", "subscribe") to add to our stop words list. We also implemented a TF-IDF function to evaluate how relevant a word is to each bag-of-words in the context of our entire corpus. TF-IDF is a statistical method used as a weighting factor in modeling. To complete our feature selection, we removed words with the TF-IDF score less than 6 and removed words that appear less than 5 times in the entire corpus of text. This level of deep cleaning was necessary as bag-of-words models disregard sentence semantics (Takase et al., 2016; Wu & Hoi, 2011). These cleaning and feature reduction techniques helped reduce the overall feature size (number of unique words) by approximately 26%, thus greatly increased the model efficiency.

Using this data, we created a bag-of-words for each podcast. Bag-of-words methods are popular for representing text documents as numerical vectors, where each vector element represents its normalized number of occurrences (Zhao, 2018).

Finally, we utilized nearest-neighbor modeling to find the most relevant podcasts. This model is popular amongst pattern recognition algorithms using distance or similarity between samples as a means of classification (Keller et al., 1985) and typically outperforms other machine learning methods, such as Naive Bayes and Term-Graphs, when applied to recommendation tasks from bag-of-word vectors (Bijalwan et al., 2014). Our main visualization is a k-nearest-neighbor graph (i.e. network graph) to show the user the podcasts most similar to the one they select (Paredes & Chavez, 2005). We are using Cosine similarity as our distance metrics, as our research revealed that it is very efficient for spare vectors (Li & Han, 2013). We also found that Cosine similarity typically out-performs Euclidean distance at clustering large volumes of vectorized text data (Muflikhah & Baharudin, 2009). There is typically a trade off between similarity and diversity in recommendation models (Smyth & McClavel, 2001), we want to find a balance between the two with our model. As our model is unsupervised, we could not use automated or even simple ways to validate our results and thus needed to rely on manual surveying. From the results of our experiments, our approach seems to produce diverse results that are still similar to the user selection.

Experiments & Surveys

As a private school project and as an unsupervised ML model, it is hard for us to measure results or strength of a recommendation (as it is subjective to each person), so we decided to collect user feedback as a measure of how relevant our recommendations were to people in the aggregate. We randomly selected 11 podcasts and pulled their top recommendations for a user survey. Users were presented with 10 of our recommendations per podcast and asked which ones they considered a “good recommendation”, based on the podcast descriptions. They were given a binary Yes or No option for the 10 recommendations, using this question: *“Based on the below descriptions, which of these podcasts are good recommendations for "[Podcast]" by [Author]?”*

This survey was sent to over 200 people, with the goal of collecting at least 20 responses. The survey consisted of 10 recommendations for 11 podcasts, for a total of 110 recommendation observations per survey. We collected a total of 30 responses – for a total of 3,300 data points. We then calculated the percentage of user-generated good recommendations, for a total of 2,242 Yes & 1,058 No responses, and an average of 67.9% good recommendations.

Our best podcast “Stuff You Missed in History Class” had 81% Good ratings. Our worst podcast “Stuff You Should Know” had 61.3% Good ratings. The former podcast is a very specific type of podcast for history buffs, and the latter was a very general podcast which covered any and every topic the author discussed. Given its broad coverage of topics and therefore broad user reviews as a result, 61.3% does not surprise us. Yet, even at its worst, more than three out of five people would consider them good recommendations.

Insights

Based on our evaluation, we learned some important things about our model and the recommendations that it produces.

One thing that stood out in using our tool is that recommendations for political podcasts ignore party lean. For example, left-leaning podcasts are recommended for conservative podcasts (and vice versa). This recommendation makes sense, given that they share a topic, politics, and oftentimes share a format, as well. However, they may not make ideal recommendations, as liberals and conservatives tend to avoid exposure to one another’s opinions (Frimer, Skitka, Motyl, 2017). This could also be seen as a positive – removing the echo-chambers created by most social media platforms that only amplify content that someone already enjoys, by providing a more diverse range of podcasts.

We also found that podcasts can be recommended for reasons that may appeal to some, but not others. For example, we found that the model tends to recommend other podcasts by the host of their selected podcast, but with a topic that differs from that of the selected. To some users, this appears to be a relevant recommendation (e.g. they like the host and want more content by them), but to others, it is not a match (e.g. they were looking for more podcasts of the same topic/genre as their selected podcast). Another insight is that some podcasts tend to use the same keywords every episode, possibly for SEO purposes, and produce poor recommendations.

While we feel confident in our measurement methodology, measuring the effectiveness of an unsupervised model proved to be challenging. We had difficulty recruiting users to complete our survey. We had to carefully balance the number of datapoints in the survey with the amount of time we asked of users. If the survey took too much time (or if it had too many questions), users would be more reluctant to complete it.

Conclusion

Overall, we are pleased with the results of our project. We have achieved what we set out to do in building a tool that reliably provides good recommendations to users in most cases. With some further fine-tuning, we believe we could resolve the edge cases noted above. Additionally, we learned some interesting things about effective ways to build a recommendation engine and uncovered a few limitations of a simple bag-of-words methodology. In building our model, we overcame a number of challenges and were agile in our adjustments. In the end, we have produced a tool that works, does what it is meant to do, and has room for improvements over time.

To further build out this tool, we could begin to scan transcripts of episodes to incorporate into the bag-of-words. Furthermore, we could improve contextual awareness of our model by looking at sentences and sentiment instead of solely relying on words. Additionally, we could create a larger stop word list (e.g. add pronouns to the list) and incorporate collaborative filtering. We could have users rate recommendations, which adds weighting, allowing the model to improve over time. We could also incorporate other models, such as Word2vec or AWS Comprehend.

Final Distribution of Effort

Proposal / Slide / Presentation	**Scraping / Storage**	**Data Cleaning**	**Modeling / Tuning**	**Progress Report**	**Visualization**	**Final Poster**	**Final Presentation**
30 Hours	**50 Hours**	**60 Hours**	**40 Hours**	**10 Hours**	**90 Hours**	**15 Hours**	**30 Hours**
Russell	Anne	Michael	Harjot	Dustin	Harjot	Anne	Russell

Zhuo (Gary)	Michael	Gary	Gary	Russell	Gary	Gary	Michael
Harjot	Gary	Harjot			Anne		Dustin

References

Albrecht, Jens, Ramachandran, Sidharth, & Winkler, Christian. (2021). *Blueprints for Text Analytics Using Python*. O'Reilly Media, Incorporated.

Bijalwan, V., Kumar, V., Kumari, P., & Pascual, J. (2014). KNN based machine learning approach for text and document mining. International Journal of Database Theory and Application, 7(1), 61-70.

Chan-Olmsted, S., & Wang, R. (2020). Understanding podcast users: Consumption motives and behaviors. *New Media & Society*. https://doi.org/10.1177/1461444820963776

Chen, Li, Chen, Guanliang, & Wang, Feng. (2015). Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction, 25(2),* 99–154. https://doi.org/10.1007/s11257-015-9155-5

Cui, Weiwei, Cao, Nan, & Yu-Ru, Lin. (2016). *Introduction to Text Visualization (Vol. 1).* Atlantis Press (Zeger Karssen).

Ifenthaler, D., Gibson, D., & Dobozy, E. (2018). Informing learning design through analytics: Applying network graph analysis. Australasian Journal of Educational Technology, 34(2).

Keller, J. M., Gray, M. R., & Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. IEEE transactions on systems, man, and cybernetics, (4), 580-585. https://doi.org/10.1109/TSMC.1985.6313426

Li, B., & Han, L. (2013, October). Distance weighted cosine similarity measure for text classification. In International conference on intelligent data engineering and automated learning (pp. 611-618). Springer, Berlin, Heidelberg.

Lops, P., Jannach, D., Musto, C. et al. (2019). Trends in content-based recommendation. *User Model User-Adap Inter 29,* 239–249. https://doi.org/10.1007/s11257-019-09231-w.

Malik, M. S. I, & Hussain, Ayyaz. (2020). Exploring the influential reviewer, review and product determinants for review helpfulness. *The Artificial Intelligence Review, 53(1)*, 407–427. https://doi.org/10.1007/s10462-018-9662-y.

Malik, Muhammad Shahid Iqbal. (2020). Predicting users' review helpfulness: the role of significant review and reviewer characteristics. *Soft Computing* (Berlin, Germany), *24(18)*, 13913–13928. https://doi.org/10.1007/s00500-020-04767-1.

Muflikhah, L., & Baharudin, B. (2009, November). Document clustering using concept space and cosine similarity measurement. In 2009 International Conference on Computer Technology and Development (Vol. 1, pp. 58-62). IEEE. https://doi.org/10.1109/ICCTD.2009.206.

Paredes R., Chávez E. (2005) Using the k-Nearest Neighbor Graph for Proximity Searching in Metric Spaces. In: Consens M., Navarro G. (eds) String Processing and Information Retrieval. SPIRE 2005. *Lecture Notes in Computer Science, vol 3772.* Springer, Berlin, Heidelberg. https://doi.org/10.1007/11575832_14.

Park, Sangwon, & Nicolau, Juan L. (2015). Asymmetric effects of online consumer reviews. *Annals of Tourism Research*, *50*, 67–83. https://doi.org/10.1016/j.annals.2014.10.007.

Smyth, Barry, & McClavel, Paul. (2001). Similarity vs. diversity. *Lecture Notes in Computer Science*, 347–361.

Takase, Sho, Okazaki, Naoaki, & Inui, Kentaro. (2016). Modeling semantic compositionality of relational patterns. *Engineering Applications of Artificial Intelligence, 50*, 256–264. https://doi.org/10.1016/j.engappai.2016.01.027.

Whitner, Gavin. *Podcast Statistics (2021) – [Infographic]*. Music Oomph, Retrieved March 4, 2021 from www.musicoomph.com/podcast-statistics/.

Wu, L., & Hoi, S. C. (2011). Enhancing bag-of-words models with semantics-preserving metric learning. IEEE MultiMedia, 18(1), 24-37. https://doi.org/10.1109/MMUL.2011.7.

Zhao, R., & Mao, K. (2017). Fuzzy bag-of-words model for document representation. IEEE transactions on fuzzy systems, 26(2), 794-804. https://doi.org/10.1109/TFUZZ.2017.2690222.

Frimer, Skitka, Motyl (2017). Liberals and conservatives are similarly motivated to avoid exposure to one another's opinions. https://doi.org/10.1016/j.jesp.2017.04.003