Interactive Visualization of Transcript Based Podcast Clustering

V. Balaji, K. Jackson, Y. H. Sun, A. Vadan, D. Wang, and Y. Wang

HIGHLIGHTS OF THE PROJECT

- 100K podcast episode data provided by Spotify was used in the project
- A novel transparent user personalized recommendation system for podcasts using clustering based on the large transcripts dataset was built
- Different clustering methods with various hyperparameters were tested and evaluated to optimize the clustering algorithm
- The recommendation is based on the episode instead of whole podcast to make the recommendation more precise to the user
- An interactive user interface with recommendation and clustering visualization was set up to encourage user engagement (https://anonyknight.github.io/podcast-visualization/index.html)

BACKGROUND AND MOTIVATION

- 51% of the United States population has listened to at least one podcast in 2019[1].

- Apple, Spotify, and Amazon are the top three podcast platforms available today, however their recommendations are NOT customizable [2]. Therefore, a user interactive recommendation system can improve the user experience [3].

- A better recommendation system should be able to:
  - categorize and recommend based on episodes instead of the whole podcast which can provide more precise recommendations based on user's interest
  - weight old podcasts lower than the recent stream of podcasts listened to
  - allow users to explore the podcast graph directly rather than just being served the top recommendation, leading to higher user engagement.

METHODS AND EXPERIMENTS

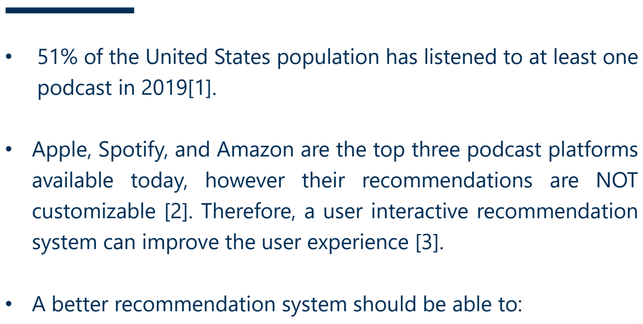

Fig 1. Architecture Diagram

- Data Cleaning & Preprocessing:
  - 105,360 JSON files which was 87.88 GB of uncompressed data.
  - Each JSON file represented one episode transcript
  - All text in each transcript was set to lowercase and typical stop words removed
  - Term Frequency-Inverse Document Frequency (TF-IDF) was also performed in order to transform each transcript's list of words into a vector of numbers representing its relative importance to the transcript
  - A lemmatized version of the corpus files was also made in order to determine whether clustering performance would benefit from possible noise reduction
  - Google Cloud's DataProc was used with a standard cluster configuration and a PySpark runtime environment during TF-IDF

- Clustering:
  - K-Means, HDBSCAN, and Agglomerative clustering methods were tested.
  - With this computing configuration and corpus subset, only K-Means and HDBSCAN successfully completed training within a 6-hour period.
  - HDBSCAN's hyperparameters minimum cluster size and minimum sample size were tested with multiple values.
  - Hyperparameter testing for k-values was performed.

Fig 2. Cluster Visualization

- Recommendation:
  - We implemented an iteratively improving dynamic recommendation engine that derives inspiration from Bayesian update methods.
  - *Clustering prior* – Where we run clustering algorithms based on transcript extracted episode feature vectors, this is used to address cold start scenarios.
  - Combining the observed data and prior, we update the recommendations model to yield the improved *posterior model*. This iterative approach helps us continuously improve recommendations.

- User Interfaces:

  Concept:
  - Focused on specialized visualization for the user to interact with podcast episodes and get better recommendations.
  - Similar to IMdB, focus on building a web browser client where users can browse Spotify podcasts.
  - HTML5, CSS, and JavaScript libraries such as d3 are extensively used to power the lightweight user interface. We hosted the application on Github Pages.

  Main functionalities:
  - Browse through the podcast clustering graph.
  - Show the expanded podcast detailed information.
  - Update rating of episodes visited as preferences change over time.
  - Rate the quality of recommendations on a scale of 0-5

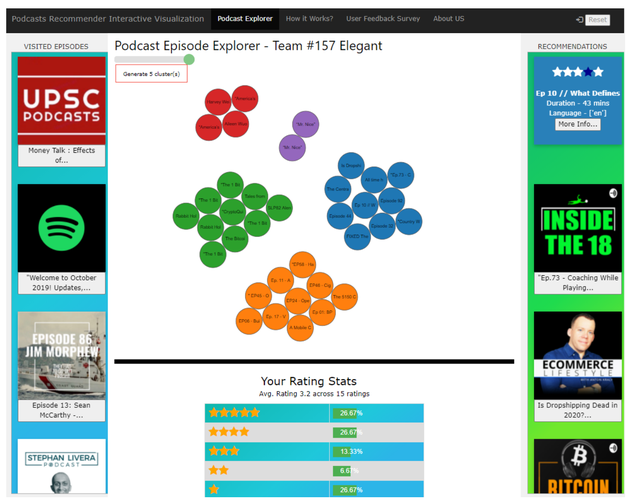

Fig 3. Overview of the user interface

The user interface is shown in Fig 3. At the top the UI, we added functionality to provide user guidance, and accumulate user feedback. In the main pane, there is a central cluster graph that lets the user explore some of the available episodes. If the user hovers over an episode and double clicks on it, a like action button is shown as in Fig 4. The user can click this, and the episode will show on the left pane (visited episodes), where there is a scrollable view of the user's liked episodes.

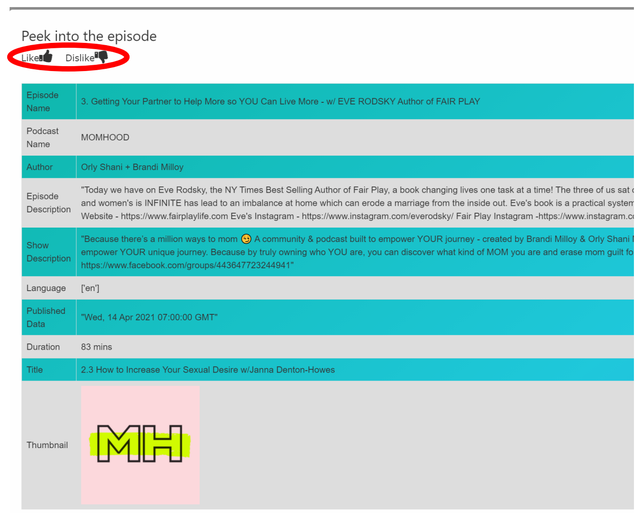

Fig 4. Like and dislike buttons on the episode details modal

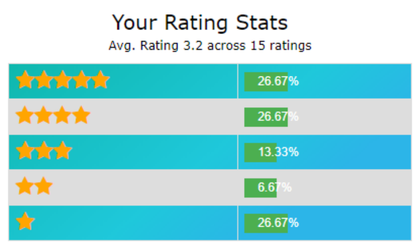

Fig 5. Interactive scoring mechanism for recommendation

User can also click on the recommendation pane on the right side and this input will trigger the backend calculation to change the rating score of the clusters (shown in Fig 5) and generate new recommendation results showing up on the recommendation pane. This allows a user interactive functionality to provide more precise recommendation results by using the user input and clustering results.

EVALUATION

- Clustering performance:

Our project has four main components to evaluate: clustering, recommendation quality, visualization, and user interface. Only the HDBSCAN and K-Means clustering methods finished computation on the small corpus subset. The optimal hyperparameters for HDBSCAN were min_cluster_size = 35 and min_samples = 10. However, this still left approximately 27% of episodes unassigned to a cluster. K-Means was then trained with the entire corpus and the entire lemmatized corpus. Hyperparameter testing was performed with the following k-values: 2, 5, 10, 15, 25, 50, 75, 100. K-Means performance was determined using PySpark's silhouette value with squared Euclidean distance**. The optimal k-value was 5 with a silhouette value of 0.574.** The range for this metric is from -1, being very poor clusters, to positive 1 with very good clusters. K-Means was the chosen algorithm for clustering.

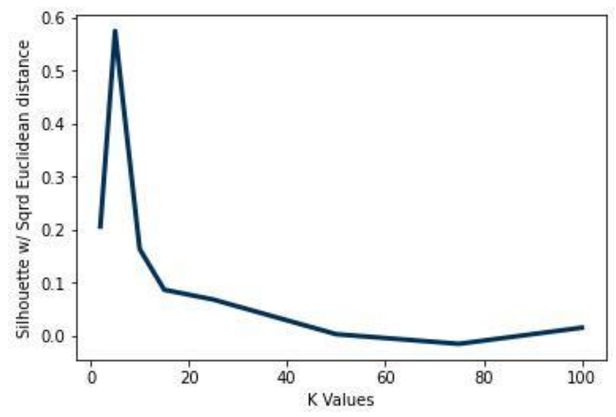

Fig. 6 K-Means hyperparameter testing

- Recommendation system and user interface

We are using a google survey collected in the UI to evaluate our recommendation and visualization success and user experience. Some user feedback is shown in Fig. 7. So far total 17 responses were obtained from the user and more than 60% users rating is equal or higher than 4 out of 5 for recommendation while higher than 70% user are satisfy with visualization .

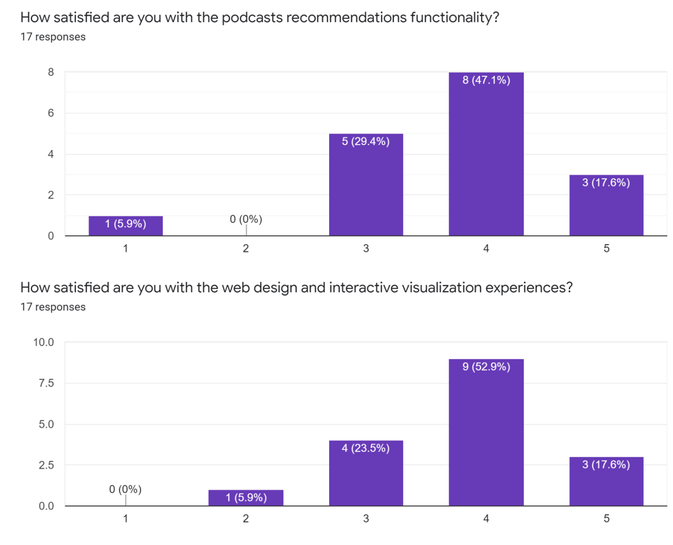

Fig. 7 Google survey results for recommendation and visualization functions

CONCLUSION AND DISCUSSION

Our project attempted to provide an innovative improvement of podcast episode recommendations by training a clustering algorithm on podcast episode transcripts which provides a more objective categorization than current practices. The visualization of the clusters and the iteratively improving recommendation algorithm provides a new way for users to discover podcast episodes they may find interesting. However, the practicality of our app may be limited by the fact that the optimal cluster size for the K-Means algorithm was 5 which may be too small for visualizing more than 100,000 episodes in a browser. The recommendation algorithm was also partially based upon this small cluster size which may reduce its performance. Furthermore, the optimal k-value with a silhouette score 0.574 could be higher to improve confidence in the actual clusters. Manual reviews would be necessary to conclusively evaluate clusters.

REFERENCES

[1] Z. Nazari et al., "*Recommending Podcasts for Cold-Start Users Based on Music Listening and Taste*," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1041–1050, doi: 10.1145/3397271.3401101.
[2] "*Spotify for Podcasters.*" https://podcasters.spotify.com/blog/podcast-show-pages.
[3] C.-W. Chen, L. Yang, H. Wen, R. Jones, V. Radosavljevic, and H. Bouchard, "*PodRecs: Workshop on Podcast Recommendations*," in Fourteenth ACM Conference on Recommender Systems, 2020, pp. 621–622, doi: 10.1145/3383313.3411444.