Shao-Min Yeh

Project Proposal

CS 410: Text Information Systems

Spotify Feature Analysis App

1. Team: Shao-Min Yeh (snyeh2@illinois.edu) - Spotify Feature App.
    a. I'm working alone due to taking this last semester remotely. Furthermore, I'm recruiting extensively and will not have much outside time, so I'm doing this project individually to maximize my time usage while making a worthwhile application.
2. Track: Free Topics. I will be doing analysis among songs from a Spotify Dataset. More specifically, I will make a Flask web application that consists of three different models a user can choose from:
    a. TF-IDF Search Engine: This model takes a user-submitted query and outputs the 50 most similar songs based on the TF-IDF weighting of lyrics, title, and artist name. Within this, the user can edit different weightings for the three categories (such as to focus on one category specifically) to further optimize their results. This will be implemented using Metapy in a similar fashion to MP2, as we can implement a search engine utilizing BM25 here. The purpose of this model is to aid the searching process of the user for many of their needs (searching for a song, looking at an artists' catalog, finding songs with certain sayings, etc.), which is a major pain point in having many songs on Spotify. One thing to note is that evaluation for this step is hard since we don't have relevance scores; however, we will manually tune the model by outputting the results for various queries and

comparing them together. To conclude, we will have a search engine that utilizes various features to output the most similar songs.

b. Song Similarity: This model takes a user-chosen song and outputs the most similar songs based on cosine similarity. This utilizes TF-IDF of the lyrics and more specific song features (key, energy, modeness, tempo, etc.) found in the dataset. Both TF-IDF and feature models will be implemented utilizing various Scikit-Learn functions, as we can individually calculate TF-IDF and compute the cosine similarity with that model and the other features. With this, the user can also separate these two weightings or combine them. The purpose of this model is to allow users to find similar songs based on a multitude of different features, which should allow users to find more desired songs. As with the search engine, evaluation for this is also hard due to the lack of relevance scores, so we will also manually compare results for different songs. To conclude, we will have a system that lets users find similar songs to a chosen song by utilizing various features.

c. Lyrics Sentiment Analysis: This model takes a user-chosen song and outputs the sentiment analysis (valence) based on that song. More specifically, this outputs the emotion level based on the TF-IDF of lyrics and musical features. This is done by randomly assigning train and test sets and training the model based on the "valence" feature, which measures the positivity of a song from 0.0 to 1.0 (and can be manipulated as our relevance rating). This will be implemented by first utilizing Metapy for tokenizing song lyrics. Then, we will use Scikit-Learn functions to calculate the sentiment analysis. The purpose of this model is to understand important features of the mood of a song while aiding the user in

searching for a song's valence. One thing of note is that evaluation for this is also easy, as we have built in relevance scores with valence. To conclude, we will have a sentiment analysis system for a chosen song by training the model on valence, a feature in the dataset.

3. Programming Languages:
    a. Backend: I am using Python
        i. Analysis Libraries: Metapy and Scikit-Learn
        ii. Utility Libraries: Pandas and Numpy
        iii. Application Library: Flask
    b. Web pages: I am using HTML files for the website
4. Work Allocation:
    a. Preprocessing Data (config, csv, dat, etc files): 2 hours
    b. Search Engine Methods: 6 hours
    c. Song Similarity Methods: 5 hours
    d. Sentiment Analysis Methods: 6 hours
    e. Application Development and Incorporation: 3 hours
    f. HTML Web Pages: 2 hours
    g. Tuning Parameters: 1 hour
    h. Internal Testing: 1 hour
    i. Cleaning up Code: 2 hours
    j. README Documentation: 2 hours
    k. Total: 30 hours