Smulemates: A Karaoke Singing Style Recommendation System

Catherine Magsino

Project #5 – Passion Project

I've grown up singing karaoke for as long as I remember – at home, at family parties, and at get-togethers with friends. It is not only one of my favorite hobbies, but also a big part of my own culture. So naturally, I decided to focus my final project on this great pastime.

Background

The aspect of karaoke that I love the most is that it brings people together. In fact, there is a whole social app around this concept, called Smule. As of 2018, this popular karaoke app had over 50 million monthly active users. It unites people from all over the world to either sing a solo, duet, or group performance. Users may also view other users' performances, and see what's popular and trending. However, what I did notice is that the app doesn't recommend specific users based on the user's particular singing style. For myself, I know that I am unable to belt out Mariah Carey songs from the 90's, so I prefer to stay within my own vocal range. So, with millions of other users on the app, how can we make it easier for users to choose who to sing with? With this project, I propose that Smule add a feature called Smulemates, a recommendation system that suggests other users with similar singing styles.

Data

As I was looking for data for my project, I came across the Stanford Digital Archive of Mobile Performances (DAMP) Dataset, consisting of several datasets of actual Smule user performances. Since the access to the files was restricted, I sent an e-mail to the alias that was specified on the website. However, after two days of not hearing back, I took it upon myself to expedite the process by trying to contact the actual owner of the dataset. First, I looked at the parent site where the dataset was being hosted – the homepage for the Stanford Center for Computer Research in Music and Acoustics. My initial attempt was a call to the phone number listed, but did not receive an answer. I then looked through the department's Facebook page and reached out through Facebook messenger. Although the administrator was willing to help me figure out the contact, I was redirected back to the alias I had originally e-mailed. Since I was back at square one, I decided to take a bolder step and reach out to an individual on LinkedIn who had written a research paper using the DAMP dataset. She was a former intern at Smule, and had worked on creating one of the datasets. A few hours later, I received a very kind response that provided an e-mail address to the owner of the dataset. The next morning, I e-mailed the owner directly and received access within the hour.

The dataset that I chose consisted of 24,874 performances across 14 songs and 5,429 singers. The data was also balanced between male and female singers.

Feature Extraction

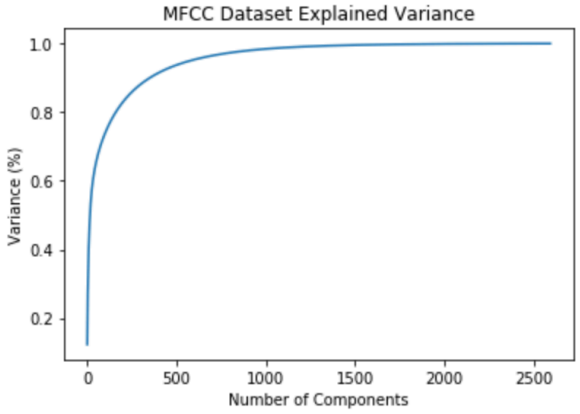
Once I collected all my data, I loaded the audio into Librosa to extract mel-frequency cepstral coefficient (MFCC) features. MFCC simulates the human auditory system and is often used when examining sound similarities. Initially, I was running into some errors, and noticed that there were about 250 files that did not contain any singing or were cut short. I worked around this issue by adding a "try-except" to my code. If the file gave an error, the file name was added to a "bad_files" list, passed, and continued with my for-loop. The file names that were not giving me errors were added to a "good_files" list. I set my number of MFCC features to 12, and once the code was run for all the files, I had a matrix of 12 features by 216 frames for each performance. I then flatted this matrix to a 1 x 2592, so that I only had one row for each performance.

```
mfcc_all = []
good_files = []
bad_files = []
i=0
for file in files:
    file_path = 'Data/Vocal_Balanced/audio/' + file
    try:
        y, sr = librosa.load(file_path,offset=67,duration=5)
        mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=12)
        mfcc_r = mfccs.reshape(1,-1)
        mfcc_r = mfcc_r[0].tolist()
        mfcc_all.append(mfcc_r)
        good_files.append(file)
    except:
        print('Failed at '+file)
        bad_files.append(file)
        pass
    i+=1
    print(i)
```

After creating 3 separate dataframes (mfcc_all, good_files, and bad_files), I had to merge the mfcc_all and good_files dataframes in order to add a "perf_key" column to the end of my row of MFCC features. Lastly, I removed null values from this dataframe and re-indexed.

Dimensionality Reduction

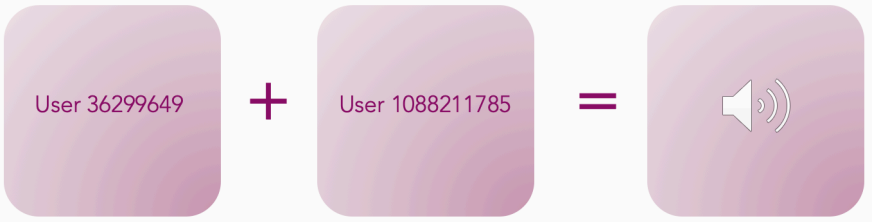
Since the mfcc dataframe had 2,592 features, I had to perform dimensionality reduction to reduce the number of features. I first normalized my data using Standard Scaler, so that all the data was between 0 and 1. I then used principal component analysis to reduce my dimensions. After plotting the number of components to variance, I chose to reduce it to 1000 components.



Content-Based Recommendation System

Lastly, I created my content-based recommendation system by comparing each performance against all the other 25K performances. I chose the distance metric of cosine similarity to recommend the most similar singing style. The recommendation system then outputs the top 3 performances with the highest cosine similarity. Here is an example of a top recommendation based on a particular user's performance:

The sound of the recommended user's voice sounds very close to the user, as they both have a raspy and mellow style of singing. Some may even think it is the same person! Now, if the two users decide to have a duet, their voices go pretty well together:



Conclusion

With singer recommendations a user may:

- Explore singers with similar singing styles.
- Choose to collaborate with the singer.
- View other performances by the singer and get ideas of songs they may be interested in performing as well.

Going forward, I would like to explore how different singing styles work well together based on popularity. Since not everyone wants to only perform with singers like themselves, I would like to create an additional recommendation system for other singing styles that would work well with the user.