Soundalike

Technology: Recommender System

Group 14: Robert, Ajay, Maria

Simply put, Soundalike recommends songs

01 Using the Million Song Challenge Dataset, we created a recommender system to create a curated playlist for any user.

02 Our final product is a React web app with a Flask server that allows users to input a song and get a list of recommended songs.

01

Motivation & Goal

Motivation

Everyone’s journey with music is different and unique -- and in 2022, technology should be able to curate music for you and you only.

Our Goal

We aren't trying to reinvent the wheel.

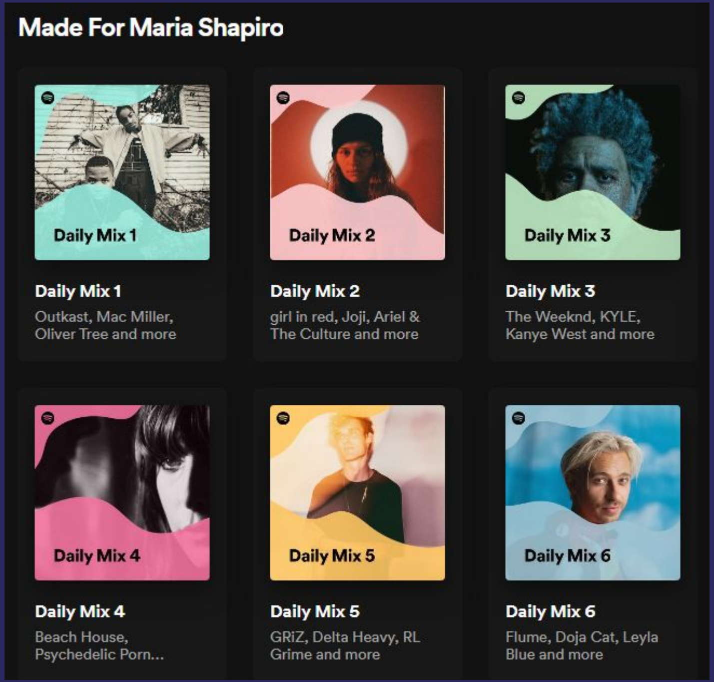
- Spotify, Apple Music, and Youtube Music already have massive user bases.
- These streaming giants have much more time, funding, and data than we do.
- We offer a good idea to fill in the gaps that these streaming platforms have.

02

Inspirations

01 Spotify

Spotify has 180 million paying subscribers, making it the most-subscribed music service. They have a suite of curated playlists called Daily Mix.

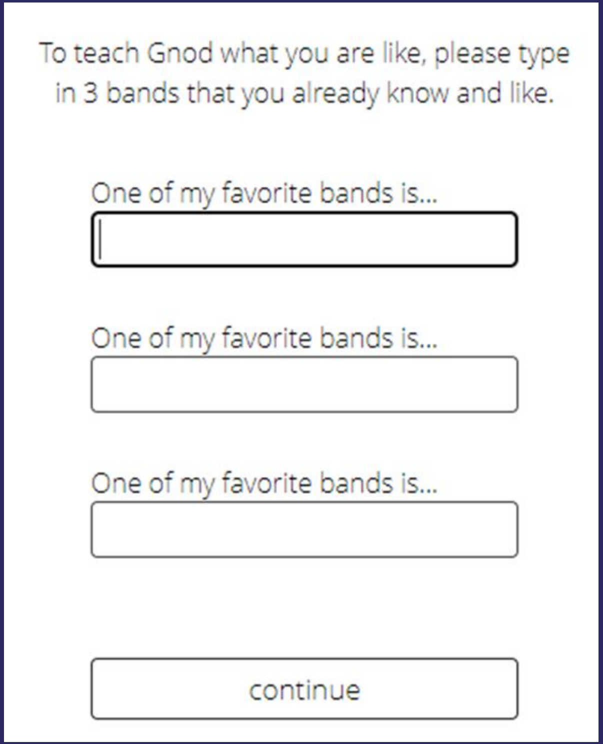


Weaknesses:

- Little transparency on how songs are chosen
- Doesn't often show new music and artists
- Doesn't vary much from day to day

02 Gnoosic

Gnoosic has about 200,000 visitors/month. As a smaller service, it does not include a music player.



Weaknesses:

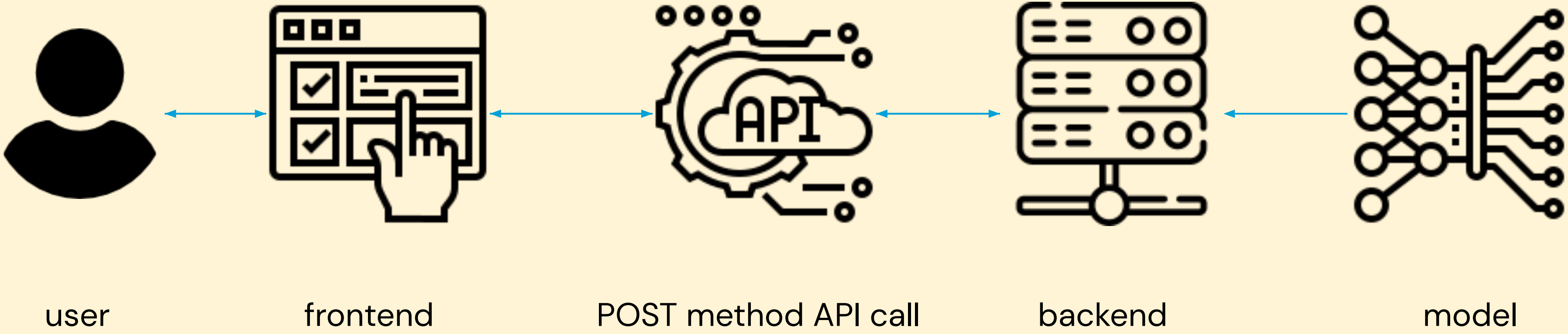
- Recommends artists, not songs
- No integration with any music services

03

App Functionality

Dataset -- Million Song Dataset Challenge

Open-access with listening history for 1 million+ users


user
frontend
POST method API call
backend
model

Frontend

For the front end, we are utilizing React and deploying with Vercel

We chose this for a variety of positives:

- Mix of HTML, CSS, and JavaScript in JSX which makes React incredibly flexible
- Virtual DOM (Document Object Model) is responsible for React’s performance and speed
- Library’s strong community support - can be attributed to the fact that React is open-source and includes many packages to aid development

Backend

We are using Python Flask as our backend server because it is quick to connect and has very minimal boilerplate code. Our entire team is very familiar with Python.

- Service layer handles POST request call (axios) from React frontend
- Handler class validates inputted song title with Million Song dataset
- Sends song title to ML model

Models

- Item-Item Nearest Neighbor (with Cosine Similarity)
- Matrix Factorization
  - Logistic Matrix Factorization (LMF)
  - Alternating Least Squares (ALS)

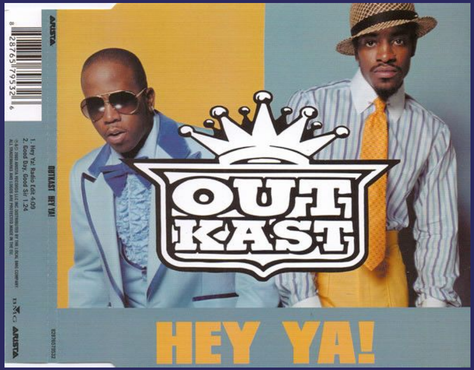
Recommendation Quality

- Precision-at-*k* metric:

$$P_k(u, y) = \frac{1}{k}\sum_{j=1}^{k} Mu, y(j)$$

- Proportion of top-*k* predicted rankings a user listened to in testing dataset

Song Recommendation Examples



Hey Ya! – Outkast

1. Please Please Please – Shout Out Louds
2. Tiny Explosions – The Presidents of the United States of America
3. Already Gone – Kelly Clarkson
4. The Way You Lived – CKY
5. Greenback Dollar – The Kingston Trio

04

Evaluation

Metrics for Success

1. User satisfaction
2. Recommendation speed
3. Recommendation quality

05

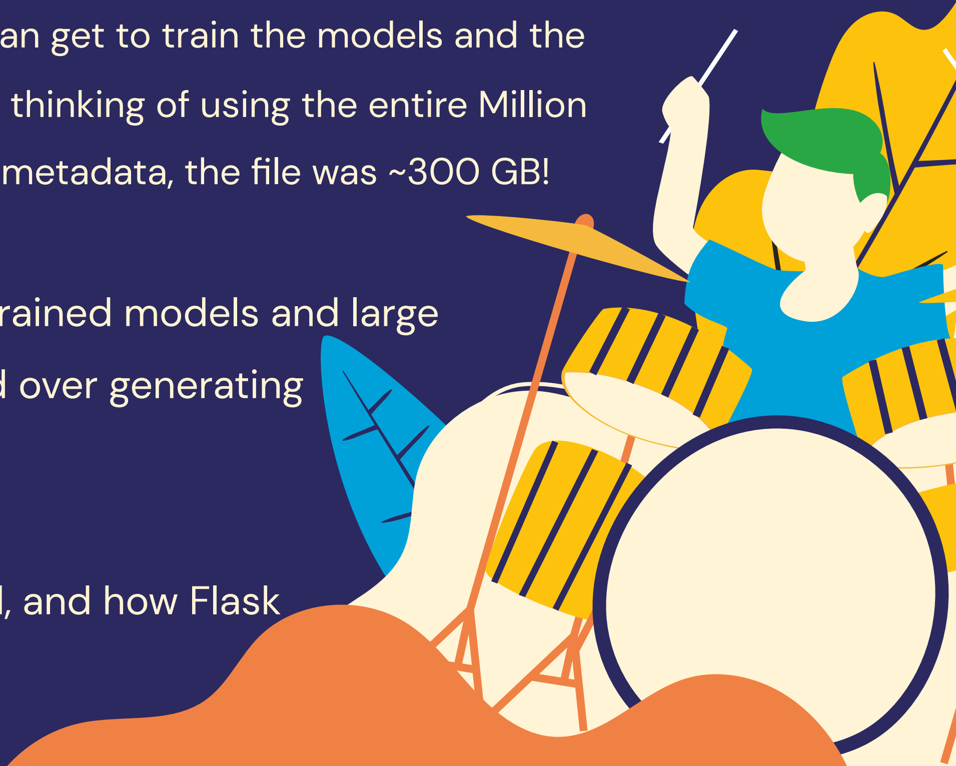
Lessons Learned

Lessons Learned

Maria: I didn't realize how big some files can get to train the models and the required storage resources. We we initially thinking of using the entire Million Song Dataset, and even though it just had metadata, the file was ~300 GB!

Robert: Using Pickle in Python to save trained models and large arrays/matrices greatly improves speed over generating them from scratch.

Ajay: I learned more about the backend, and how Flask React can complement each other.



Thanks!

Do you have any questions?

mas@gatech.edu
rmorgan61@gatech.edu
ajayvijayakumar2018@gmail.com