Yet Another Spotify Recommender (YASR)

Jesse Tao

OVERVIEW

INTRODUCTION 01

Why is there a need for another recommender?

DATA COLLECTION 02

Where we are getting our data

INITIAL MODEL 03

Analyzing Audio Files

IMPROVED MODEL 04

Using Spotify API Audio Features

LIVE DEMO 05

Test out YASR for Yourself!

CONCLUSION 06

Wrap up and further considerations

01 INTRODUCTION

Why Do We Need Another Recommender?

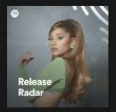
Finding new music on Spotify is difficult:

- Only top artists are easily discoverable
- Large library that gets updated daily
- Foreign music is hard to search for

Makes it very difficult for new artists to get recognized

Spotify's New Releases

New Music Friday
New music from Ariana Grande, 24kGoldn, Conan Gray, and more!
3,707,128 FOLLOWERS



Release Radar
Catch all the latest music from artists you follow, plus new singles picked for you. Updates every Friday.
MADE FOR JESSE.P.TAO

New albums & singles

3, 2, 1
24kGoldn

Pegasus: Neon Shark vs Pegasus Presented By Travi...
Trippie Redd

Go Crazy (Remix) (feat. Young Thug, Future, Lil Dur...
Chris Brown , Future , Mulatto

Loyalty Over Royalty
CJ

Positions (Deluxe)
Ariana Grande

Núcleo
vf7

Only The Generals Part II

29

Way Less Sad

Terra Firma

times

Rocky

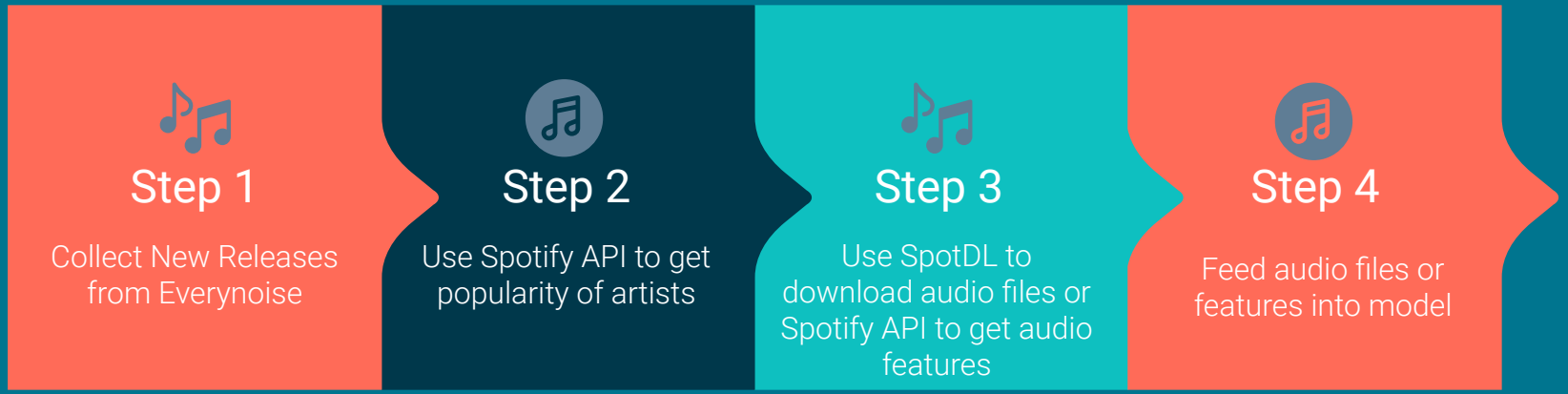
Everynoise New Releases

17162 USA Releases

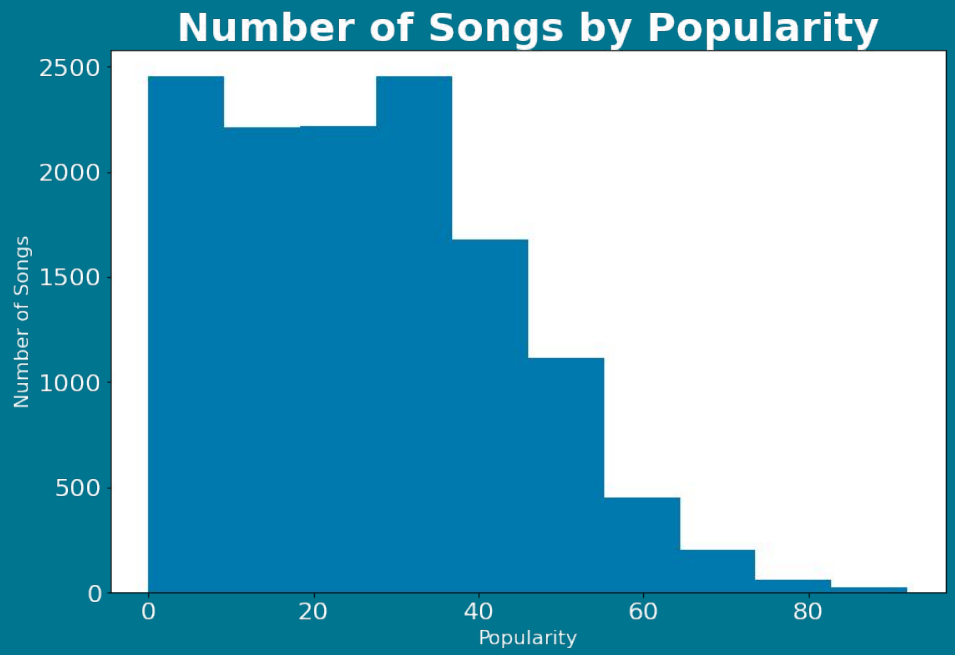
- **Ariana Grande** *Positions (Deluxe)* 19
- **YoungBoy Never Broke Again** It Ain't Over 2 ⟲
- **YoungBoy Never Broke Again** Toxic Punk 3 ⟲
- **Trippie Redd** *Pegasus: Neon Shark vs Pegasus Presented By Travis Barker (Deluxe)* 40
- **Chris Brown** Go Crazy (Remix) (feat. Future, Lil Durk & Mulatto)
- **Marshmello** Lavandia
- **The Beatles** *The Beatles For Kids - Animals* 6 ⟲
- **NF** CLOUDS
- **NF** CLOUDS (Edit)
- **Florida Georgia Line** *Life Rolls On (Deluxe)* 17
- **Brent Faiyaz** Eden (From "Black History Always / Music For the Movement Vol. 2")
- **Tory Lanez** F.E.E.L.S. (feat. Chris Brown)
- **Lil Yachty** Hit Bout It
- **Kevin Gates** *Only The Generals Part II* 12
- **24kGoldn** 3, 2, 1
- **Rauw Alejandro** 2/Catorce
- **Russ** MISUNDERSTOOD
- **Jason Derulo** Lifestyle (feat. Adam Levine) (David Guetta Slap House Mix)
- **AJR** Way Less Sad 4
- **Conan Gray** Overdrive
- **David Guetta** Let's Love (feat. Sia) (Aazar Remix)
- **ILLENIUM** Hearts on Fire (The Remixes) 4
- **Mariah Carey** *We Belong Together (Mimi's Late Night Valentine's Mix)* 2 ⟲
- **CJ** *Loyalty Over Royalty* 8
- **Young Dolph** Case Closed
- **Powfu** the way that you see me (feat. Ayleen Valentine)
- **Wallows** Quarterback
- **Wallows** *Remote (Deluxe)* 11
- **Denzel Curry** So.Incredible.pkg [Robert Glasper Version Feat. Smino]
- **Nelly** Lil Bit (FGL Remix)
- **Nelly** *Country Grammar (Live)* 14
- **David Bowie** That's Entertainment (2021 Version) / Cosmic Dancer (Live) 2
- **El Fantasma** *Carteles* 10
- **Johann Sebastian Bach** Bach: Toccata in G Major, BWV 916
- **Johann Sebastian Bach** *Hooked on MIDI, Vol. 1* 7
- **Johann Sebastian Bach** *I Love Bach* 200 ⟲
- **Johann Sebastian Bach** *Johann Sebastian Bach: Sonate e partite per il flauto traversiere* 16
- **Johann Sebastian Bach** *Prisme- Bach* 20
- **Alan Walker** Fake A Smile
- **John Williams** The Adventures of Tintin: The Duel

02 Data Collection

Process


Step 1
Collect New Releases from Everynoise
Step 2
Use Spotify API to get popularity of artists
Step 3
Use SpotDL to download audio files or Spotify API to get audio features
Step 4
Feed audio files or features into model

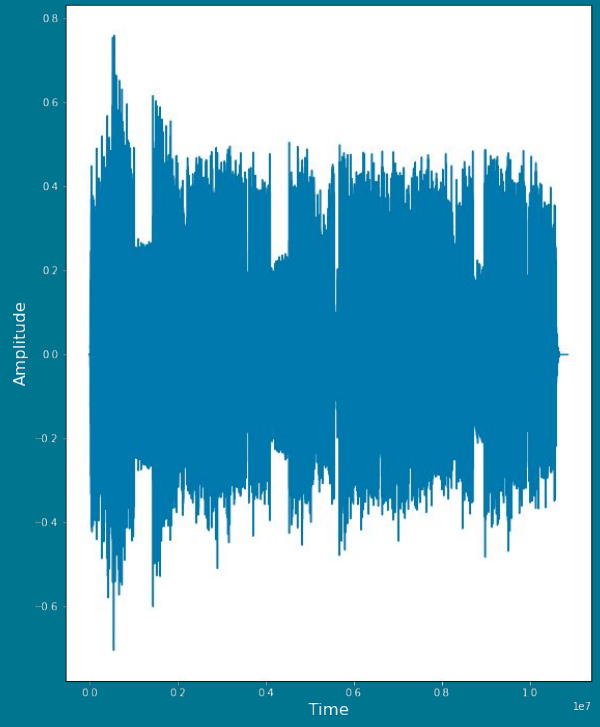
It’s Hard to Get Popular

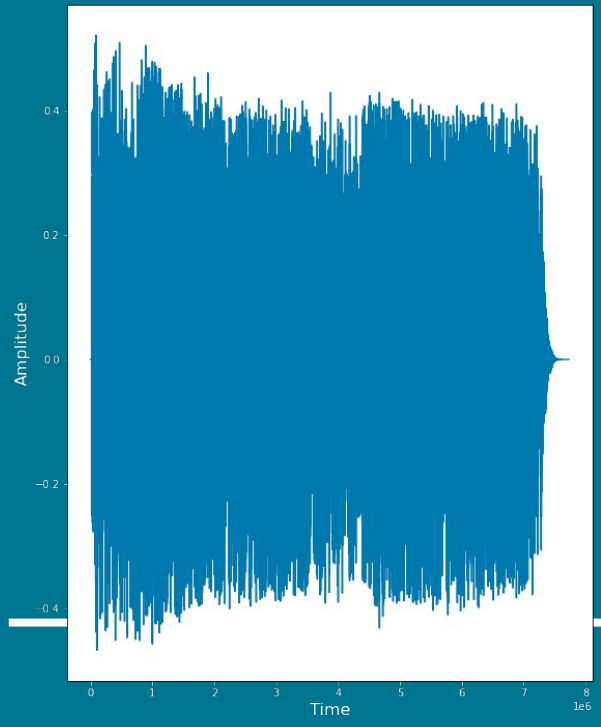


03 Initial Model

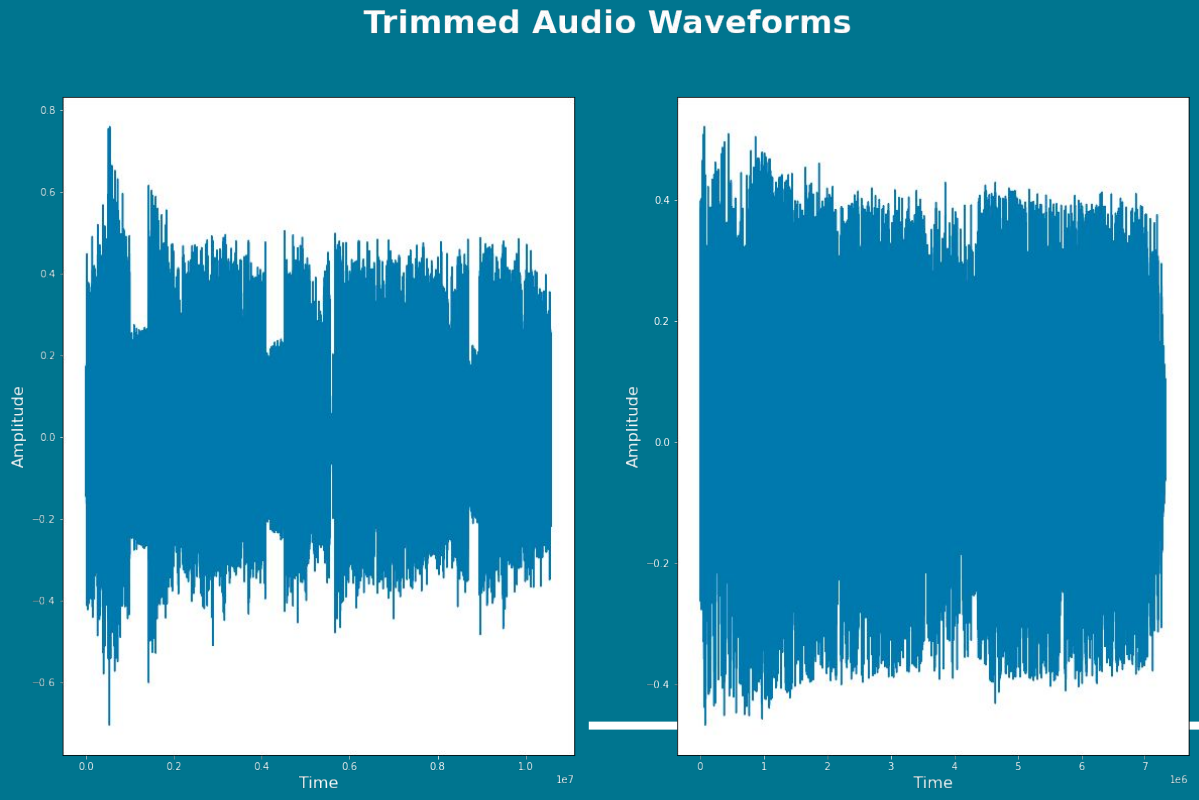
Initial Look at Audio Files

Audio Waveforms

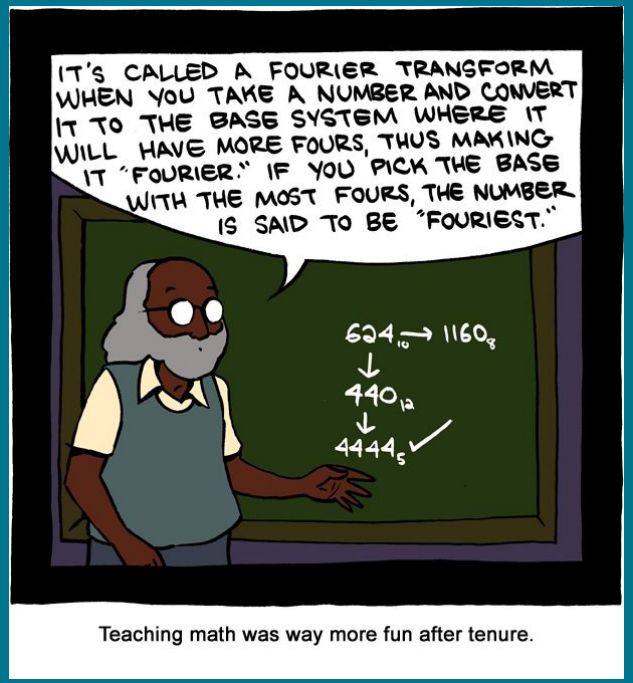




Trimming Audio



More Fourier Transforms



More Fourier Transforms

Equation for Fourier transform:

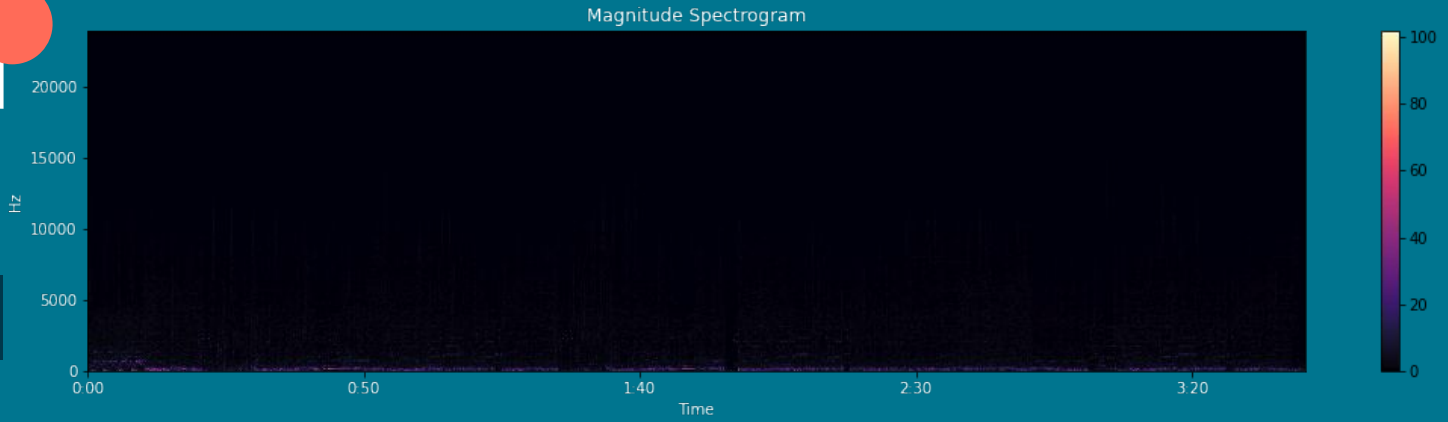
$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\,dx,$$

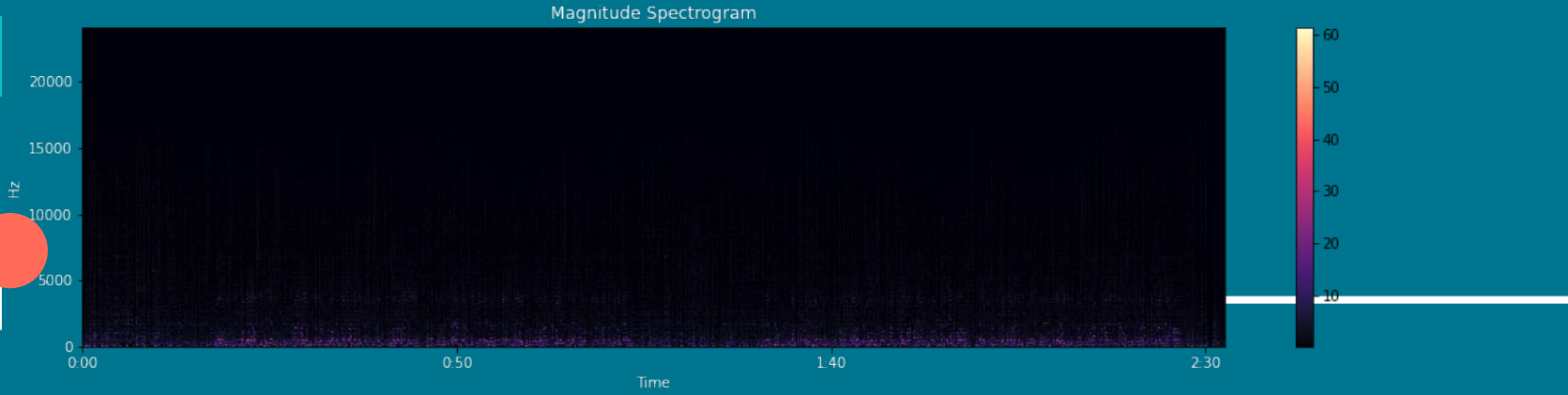
- Allows signals to be converted from time to frequencies
- Main disadvantage is we lose all time component of signal
- Luckily, we have the Short-Time Fourier Transform (STFT):

$$X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t-\tau) e^{-i\omega t}\,dt$$

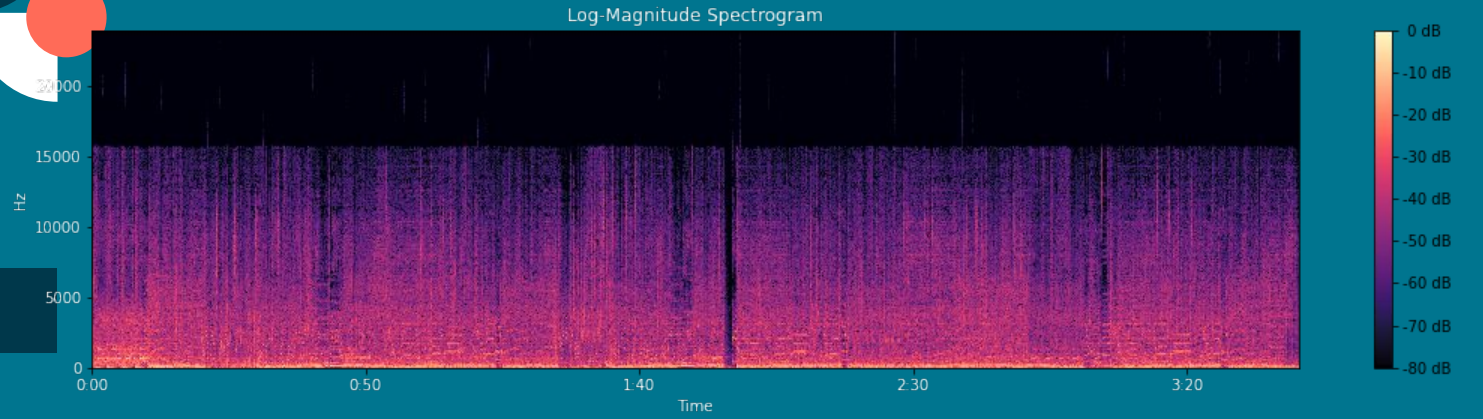
- Keeps some of our time component of signal using windowing

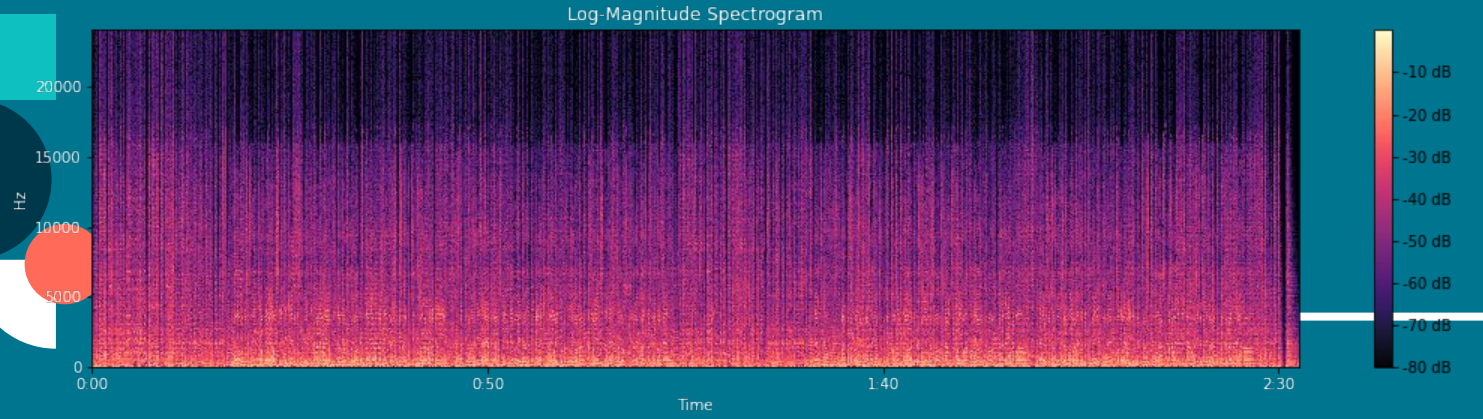
Visualizing STFT



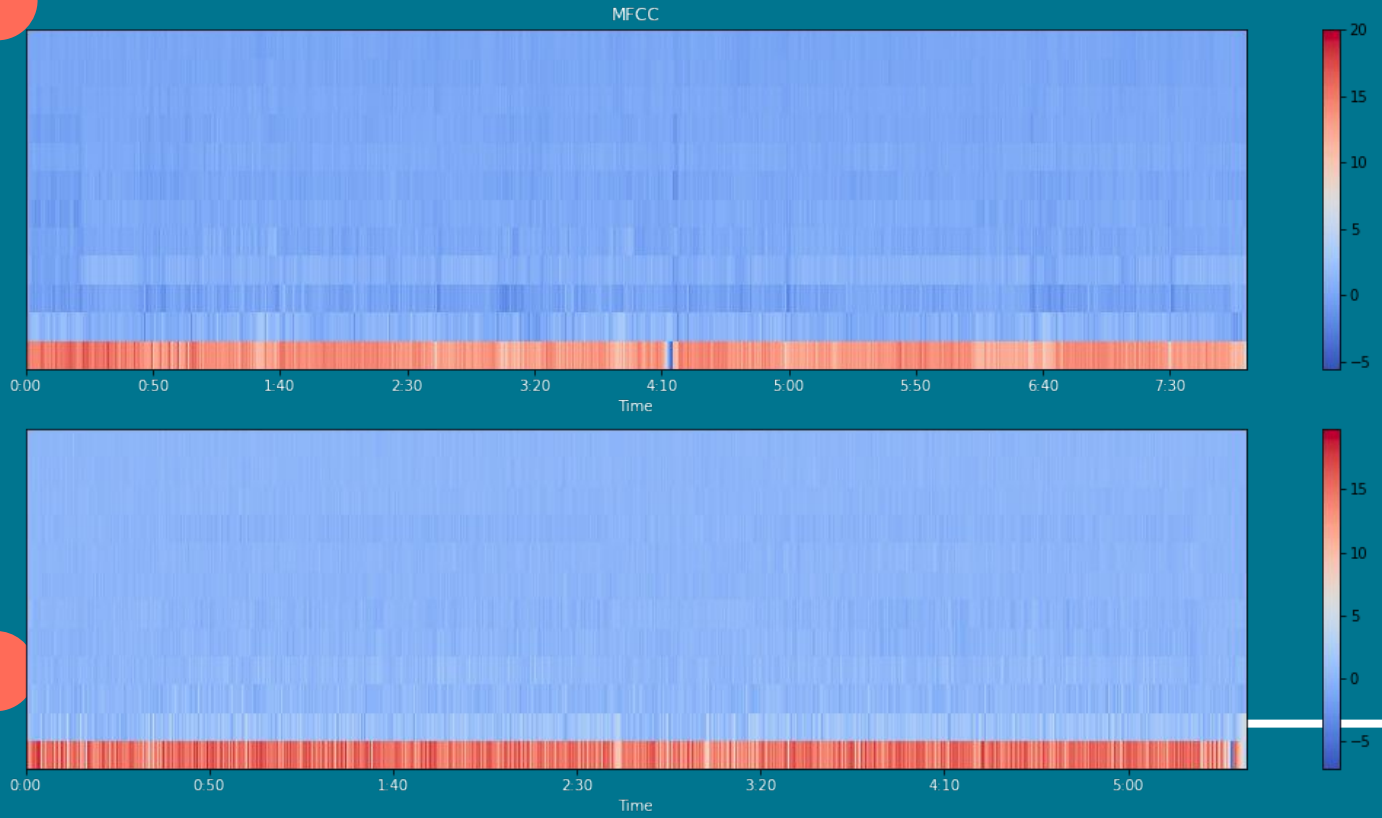


Scaling STFT



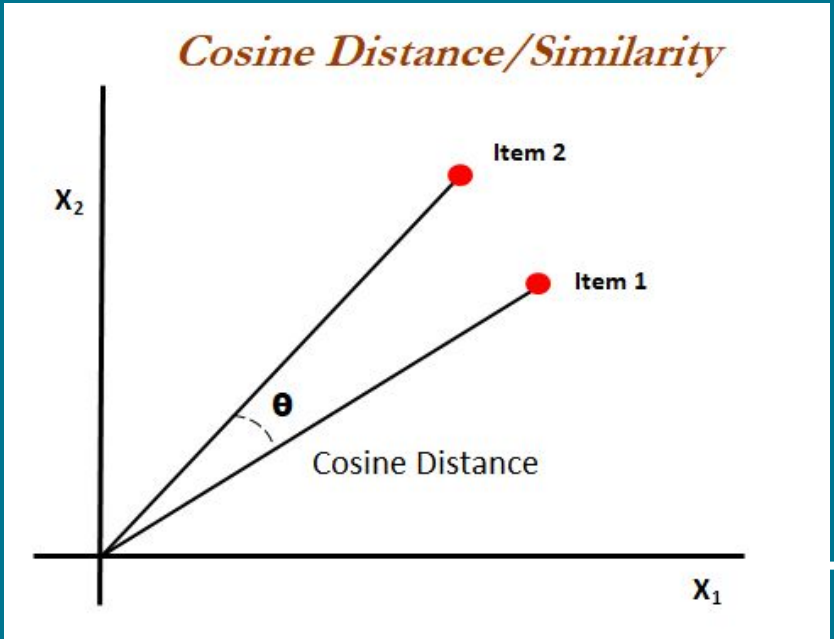


MFCCs


MFCC
20
15
10
5
0
−5
0:00
0:50
1:40
2:30
3:20
4:10
5:00
5:50
6:40
7:30
Time
15
10
5
0
−5
0:00
0:50
1:40
2:30
3:20
4:10
5:00
Time

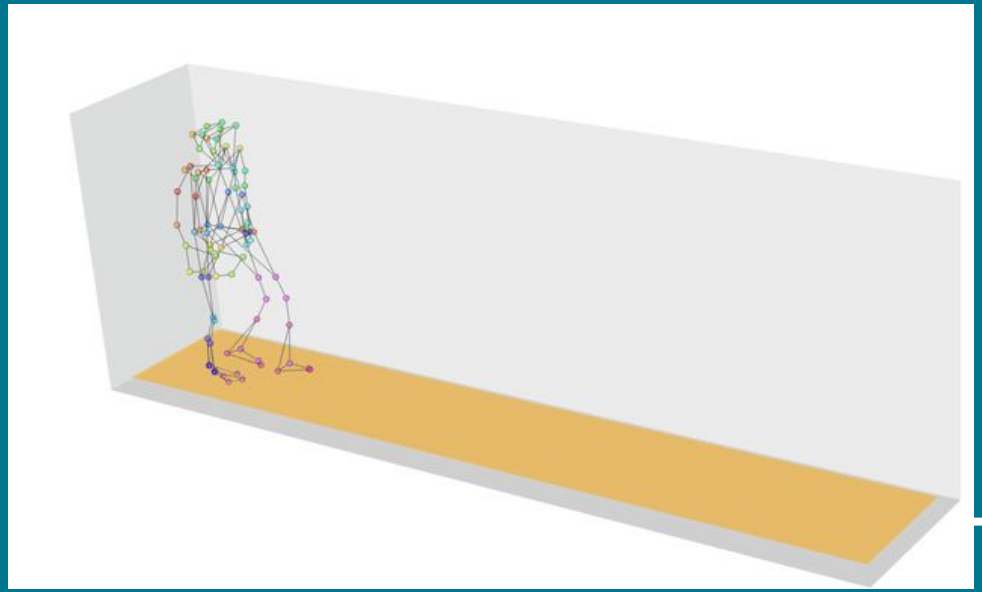
Using Cosine Similarity

- Make sure signals are changed to be equal length
- Using the cosine similarity metric
- Takes nearly 2 hours just to calculate all cosine similarities



Using DTW

- Signals do not have to be changed to be equal length
- Much fairer comparison between 2 audio tracks of different length
- Regular DTW algorithm takes about 5 seconds per comparision
- FastDTW algorithm was used for performance



DTW is Fast, but Still Slow

```
100%|██████████| 4806/4806 [1:07:17<00:00,  1.19it/s]
100%|██████████| 4806/4806 [44:09<00:00,  1.81it/s]
100%|██████████| 4806/4806 [55:10<00:00,  1.45it/s]
100%|██████████| 4806/4806 [50:36<00:00,  1.58it/s]
100%|██████████| 4806/4806 [48:44<00:00,  1.64it/s]
100%|██████████| 4806/4806 [58:31<00:00,  1.37it/s]
100%|██████████| 4806/4806 [43:48<00:00,  1.83it/s]
100%|██████████| 4806/4806 [59:17<00:00,  1.35it/s]
100%|██████████| 4806/4806 [53:39<00:00,  1.49it/s]
100%|██████████| 4806/4806 [1:32:25<00:00,  1.15s/it]
100%|██████████| 4806/4806 [40:51<00:00,  1.96it/s]
100%|██████████| 4806/4806 [1:07:43<00:00,  1.18it/s]
100%|██████████| 4806/4806 [57:22<00:00,  1.40it/s]
100%|██████████| 4806/4806 [48:58<00:00,  1.64it/s]
100%|██████████| 4806/4806 [59:20<00:00,  1.35it/s]
100%|██████████| 4806/4806 [37:21<00:00,  2.14it/s]
100%|██████████| 4806/4806 [55:22<00:00,  1.45it/s]
100%|██████████| 4806/4806 [54:28<00:00,  1.47it/s]
100%|██████████| 4806/4806 [52:55<00:00,  1.51it/s]
100%|██████████| 4806/4806 [55:16<00:00,  1.45it/s]
100%|██████████| 4806/4806 [53:20<00:00,  1.50it/s]
100%|██████████| 4806/4806 [57:03<00:00,  1.40it/s]
100%|██████████| 4806/4806 [48:15<00:00,  1.66it/s]
100%|██████████| 4806/4806 [53:52<00:00,  1.49it/s]
100%|██████████| 4806/4806 [1:05:02<00:00,  1.23it/s]
100%|██████████| 4806/4806 [51:50<00:00,  1.55it/s]
100%|██████████| 4806/4806 [40:48<00:00,  1.96it/s]
100%|██████████| 4806/4806 [58:21<00:00,  1.37it/s]
100%|██████████| 4806/4806 [45:10<00:00,  1.77it/s]
100%|██████████| 4806/4806 [1:04:51<00:00,  1.23it/s]
100%|██████████| 4806/4806 [57:31<00:00,  1.39it/s]
100%|██████████| 4806/4806 [56:26<00:00,  1.42it/s]
100%|██████████| 4806/4806 [54:41<00:00,  1.46it/s]
100%|██████████| 4806/4806 [59:50<00:00,  1.34it/s]
100%|██████████| 4806/4806 [46:43<00:00,  1.71it/s]
100%|██████████| 4806/4806 [57:56<00:00,  1.38it/s]
100%|██████████| 4806/4806 [54:00<00:00,  1.48it/s]
```

04 Improved Model

Spotify API Audio Features

- Danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration, time signature
- Removed songs by artists with over 70 popularity
- Only takes 2 seconds to compare all tracks

Model Training Times Comparison

	Data Retrieval	Preprocessing	Training
Audio File Cosine Similarity	1 Day	1 Hour	80+ Hours
Audio File DTW	1 Day	1 Hour	40+ Hours
Spotify API Features	20 Minutes	10 Seconds	2 Seconds

Pros and Cons of Audio File Model

Pros

- Can actually see what we are comparing
- Uses interesting machine learning algorithms

Cons

- Long preprocessing and training times
- Only songs that are available on Youtube Music can be compared

Pros and Cons of Spotify API Model

Pros

- Really quick preprocessing and training times
- All new releases on Spotify can be recommended

Cons

- No knowledge of how numbers for audio features are defined
- Relies purely on the difference between these arbitrary features

05 Live Demo!

The Part You've Been Waiting For!

https://yasr.jesseptao.com

06 Conclusions

What did we learn?

- Comparing audio files is an arduous process that even GPUs struggle to do quickly
- We ended up having to just use simple numbers to build a recommender due to these challenges
- Building a robust web app that accommodates different user's is difficult

Further Steps

- Create interaction terms between Spotify API Audio Features to see if our recommender improves
- Use speech recognition with our Audio File modeling to see if we can use NLP to recommend songs
- Build our own features from our Audio File analysis so we don't have to rely on Spotify API's audio features
- Classify audio into genres as Spotify's API does not include the genre of a song
- Create another page where users can see more recommendations
- Use user's ratings on recommendations to recommend songs to similar users
- Write code that takes better advantage of the GPU using TensorRT or CUDA in C/C++
- Expand this model to Apple Music, Google Play Music, other music streaming services

THANKS

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.

Resources Used

- https://en.wikipedia.org/wiki/Fourier_transform
- https://en.wikipedia.org/wiki/Short-time_Fourier_transform
- https://en.wikipedia.org/wiki/Dynamic_time_warping
- https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/eb9cd609-e44a-40a2-9c3a-f16fc4f5289a.xhtml
- https://everynoise.com/new_releases_by_genre.cgi?genre=anygenre&region=US
- https://www.tensorflow.org/io/tutorials/audio
- https://github.com/ernestk-git/data-scientist-ish/blob/master/find_nearby_coords.ipynb
- https://www.tensorflow.org/guide
- https://github.com/titu1994/dtw-numba
- https://developer.spotify.com/documentation/web-api/reference/
- https://www.w3schools.com/
- https://github.com/mari-linhares/spotify-flask
- https://github.com/plamere/spotipy
- https://stackoverflow.com/questions/14525029/display-a-loading-message-while-a-time-consuming-function-is-executed-in-flask
- https://github.com/ctralie/linmdtw
- https://realpython.com/flask-by-example-implementing-a-redis-task-queue/