Hitachi 2 User's Manual

About our Product

Our team has created a pipeline that takes a list of songs and their respective artists, and outputs a series of visualizations based around song metadata, pulled from Spotify's API, and song lyrics, pulled from Genius's API and interpreted through Pinecone. All of the code, data, and API pulling needed to produce our final visualizations can be achieved by following a few simple steps.

Table of Contents:

Requirements

- Python IDE or Interpreter
  - We used Microsoft Visual Code, as it has compatibility with .ipynb files as well as standard .py files, but other IDEs will also work.
- lyricsgenius
  - lyricsgenius is a Python library that has versatile tools for using the Genius API, and it can be directly pip installed to the terminal, as seen below:

- spotipy
  - Spotipy is a Python library that helps harness spotify metadata. If your environment does not have spotipy pre-installed, you can also pip install it directly into the terminal:

- Acceptable Format for Songs:
  - CSV containing columns for song title and artist name
  - Data from tsort.info, a website that charts top 100 billboard songs from every year.
- Code from GitHub repository
  - Our code can be accessed from this link: https://github.com/darthjuju1/data_capstone_399

Data Extraction and Cleaning Pipeline

Extraction from CSV:

- In order to fetch the metadata for songs in a given CSV file, open run_df_fetcher.py.
- Replace variable given_df with your respective CSV file.
- Make sure that the generate_csv function has all of the variables set how you want them, though the default settings will output the product as desired. Below is the documentation for this function:

```
def generate_csv(top_songs, file_path, start_index=0, create_copy=False, artist_name = 'Artist Name(s)', track_name = 'Track Name'):
    """
    Parameters
    top_songs: CSV containing the top songs
    file_path: Name of the file to write to
    start_index: Index to start from in the DataFrame (default=0)
    create_copy: If True, creates a copy of the CSV instead of overwriting (default=False)
    artist_name: Column name for artist names (default='Artist Name(s)')
    track_name: Column name for track names (default='Track Name')
    ---------------
    Description
    This function will take in a CSV containing the top songs and write the song titles and artist names to a CSV file.
    Needs LyricsGenius to be installed, as well as a Genius API key and a specifically formatted CSV file.
    ---------------
    Returns
    If create_copy=False:
        index = The index of the last song written to the CSV file
    If create_copy=True:
        (index, copy_path) = The index of the last song written and the path to the copied CSV file
    """
```

- Run the file.

> **Additional Explanation**
>
> This function will not create a copy of the csv by default, but if you set “create_copy” to True it will, in case you would like to inspect the data before merging. If you set this, be sure to use function merge_csv in oder to merge your copy back into the main dataframe.
>
> Regular Expressions are also being run for all added lyrics and artists that clean the data for later usage in Pinecone.

- Genius’s API will expire after around an hour. As a result, the function will only search for 55 minutes before stopping. This will typically gather data for 800-1000 songs, and the function can be run again after an hour.

Extraction from Website

- Open run_scraper.py
- Replace url with your address to website, and file_path with the desired output csv.

> **Notice**
>
> We currently only have the web scraper built around the website tsort.info, which has songs from the top 100 billboard charts on an annual basis dating back to the mid-20th century.

- Run the file.
- Like the previous function, Genius's access token will expire after around an hour, so this file will break the loop after 55 minutes. This will typically cycle through around 8-10 years, and the function can be run again after an hour.

End Result

You will now have a csv file with cleaned lyrics and a total of 9 fields called song_lyrics.csv. This will be used in the next part of the process: Pinecone.

Pinecone Vectorization

Index Creation:

- Used Pinecone's default code snippet for creating an index with integrated text embeddings through the llama-text-embed-v2 model.

Upserting:

Using a try/except block to skip over songs that failed to upsert, we were able to push our initial dataset of 8000 songs to our index's namespace over the course of 20 minutes. However due to some total of 7,752 upserted songs. We were able to develop a more efficient batch upsert function, for use with our final 12,500 song data frame.

Fetching:

Used Pinecone's query() to pull back vectors for semantic lookups—whether for search or similarity scoring. They were then saved to a CSV for analysis and visualization.

Data Visualization

Visualizations: What to Know

All of our visualizations can be generated by running a single file, but we will break down the details of each one below, as well as how to access them.

How to Access

- Open data_capstone_399/src/visualization/run_vis.py
- All visualizations rely on only two variables: df (DataFrame), and filepath. Fill in those two variables at the top of the file.

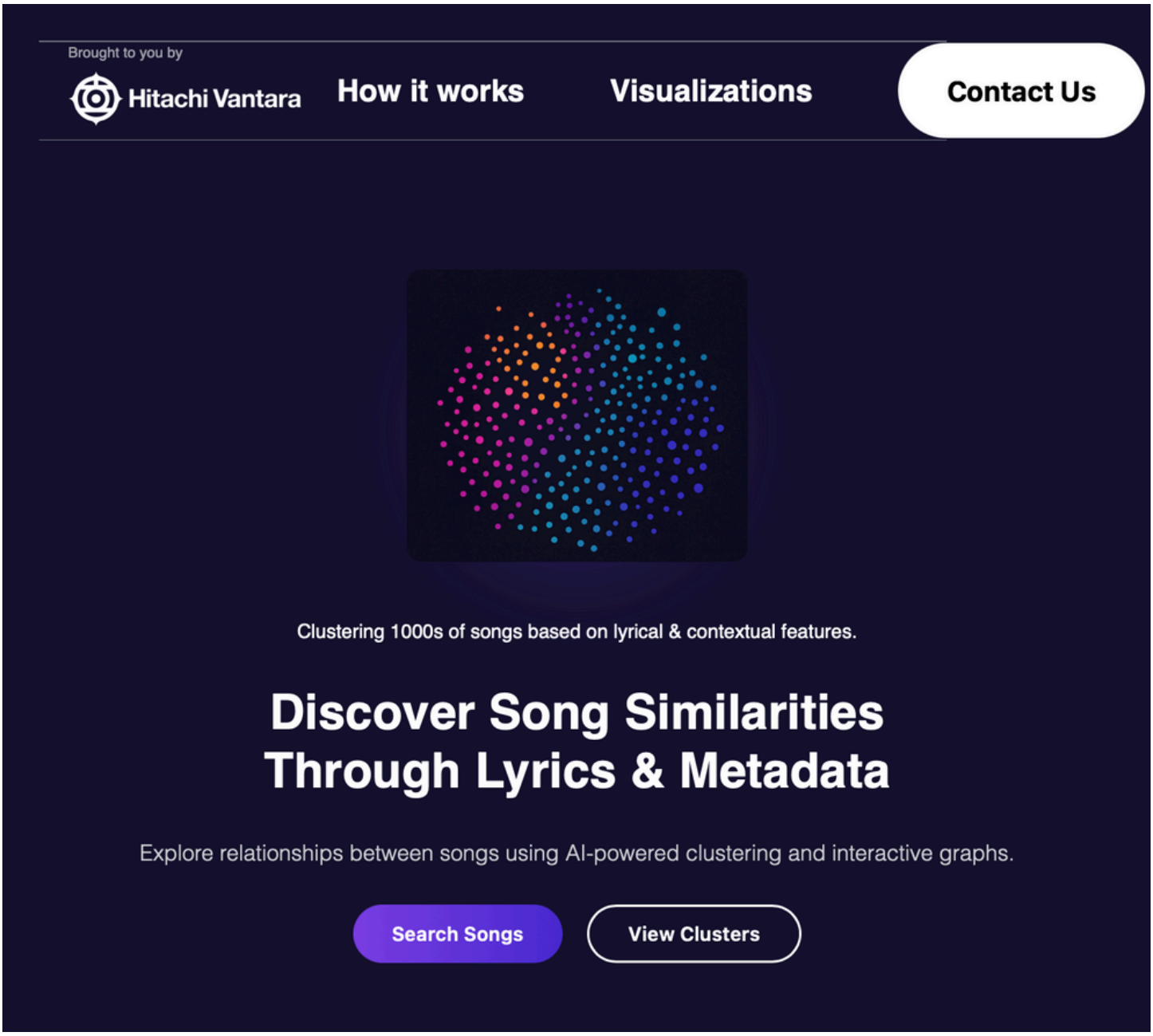
```
import vis_functions as vis
import pandas as pd

df = pd.read_csv("C:/Users/Daniel Day/Downloads/School/Data-399/data_capstone_399/data/Improved_Bucket_Accuracy_balanced_60s_up.csv")
filepath = "C:/Users/Daniel Day/Downloads/School/Data-399/data_capstone_399/src/Visualizations"

decades_list = vis.get_decades(df)
vis.decades_scatter(decades_list, filepath=filepath)
vis.decades_radial(decades_list, 'C:/Users/Daniel Day/Downloads/School/Data-399/data_capstone_399/Visualizations')
vis.plotly_scatter_explicit(df)
vis.plot_avg_intra_genre_cosine_by_decade(df, filepath=filepath)
vis.plot_cos_sim_bubble_chart(df, filepath=filepath)
vis.plot_decade_bar_chart(df, filepath=filepath)
vis.decades_genre_similarity(decades_list, filepath=filepath)
```

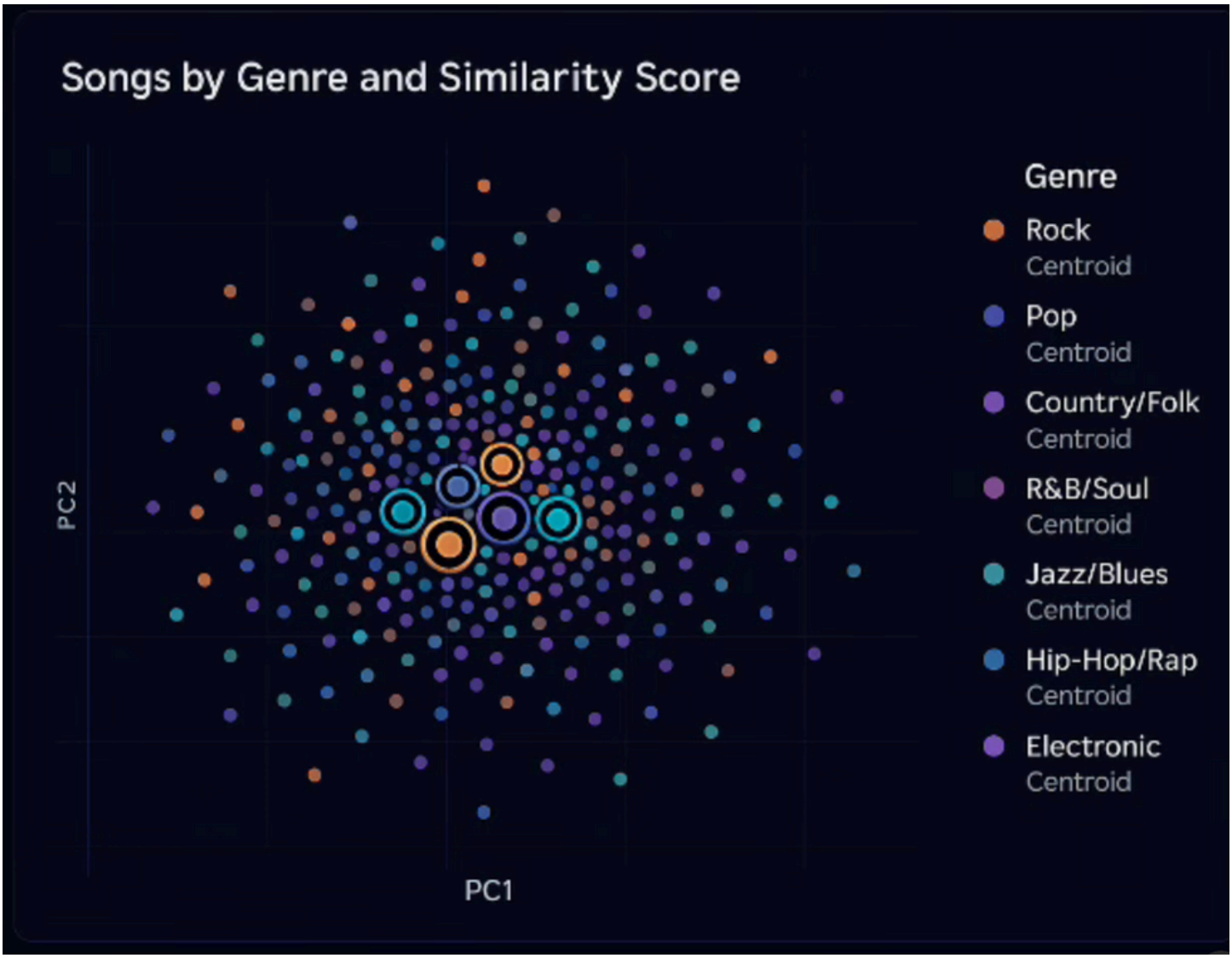
- Run the file.

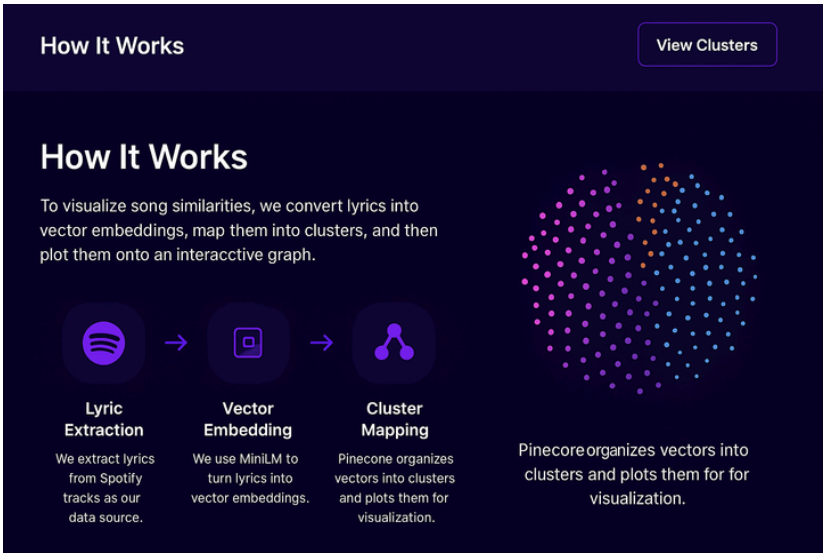
Your visualizations will now be saved as HTML files within your chosen directory! You are now free to explore your interactive graphs and charts. In the case of our team, we chose to make an interactive application and website:

🌐 Music Analysis

User Interface and Website




Songs by Genre and Similarity Score
PC2
PC1
Genre
Rock
Centroid
Pop
Centroid
Country/Folk
Centroid
R&B/Soul
Centroid
Jazz/Blues
Centroid
Hip-Hop/Rap
Centroid
Electronic
Centroid


How It Works
View Clusters
How It Works
To visualize song similarities, we convert lyrics into vector embeddings, map them into clusters, and then plot them onto an interacctive graph.
Lyric Extraction
We extract lyrics from Spotify tracks as our data source.
Vector Embedding
We use MiniLM to turn lyrics into vector embeddings.
Cluster Mapping
Pinecone organizes vectors into clusters and plots them for visualization.
Pinecore organizes vectors into clusters and plots them for for visualization.

Contact & Feedback

What features would you like to see?

Enter your feedback here...

Submit

GitHub

LinkedIn

Research Docs