# Melody Mapper: Utilizing Machine Learning to Analyze Song Themes

Nikki D'Costa, Joy Fan, Shanmuga Ganesh Thiruppathy, Neha Panduri, Maisoon Rahman, Kethan Srinivasan

# Introduction and background

- ML models find patterns in various forms of data
- Text Mining
    - identifies textual patterns and trends within unstructured data through the use of machine learning, statistics, and linguistics.
- Our project: MelodyMapper
    - Understanding the linguistics of songs
- What we use
    - Python libraries
    - Transformers
    - Dataset from Kaggle
    - Implementation of ML model

# Literature review

## What is text mining?

- Extracting valuable insights and information from unstructured textual data.
- This data can be anything from articles, reviews, posts, and other textual evidence.

## Why is natural language processing important?

- Automates analysis and pattern recognition

**Natural language processing enabled computers to understand and process human language**

# Content of Project

- Overall goal: to analyze the theme of a song using machine learning and natural language processing (NLP) via text-mining techniques
- Main objective: determining meanings of lyrics to further advance song analysis
- What we used:
  - Lyrics in question retrieved from an online dataset
    - Spanning from the 1950s up to 2019
    - Used the ML model to learn what songs belong to which one of the seven general categories
    - Sorted into seven general categories:
      - Sadness
      - Violence
      - world/life
      - Obscene
      - Feelings
      - Romantic
      - music

# Dev tools

- Python
  - Widely-used and highly readable programming language with support for object-oriented and functional programming
- NumPy (Numerical Python)
  - A library in Python primarily used for numerical computing and data manipulation
- Pandas
  - An open-source data manipulation and analysis library, written in Python
- Transformers
  - A type of deep learning architecture with a significant impact on NLP
- Tensorflow
  - Open-source machine learning library, used mainly for building/training machine learning models
- Scikit-learn
  - Python-based machine learning library with tools for data mining and analysis

# Project aim/core theory

- Use quantitative analysis to map out patterns, correlations, and thematic associations within songs by identifying the sub topics covered in the song
- We use parameters such as romanticism and violence to acoustic and instrumental elements to  categorize the nuanced themes in the song dataset we have

**Main mission: enhance human capacity for understanding song themes by employing machine learning techniques.**

# Data Preprocessing

- Loading in the data
- Dropping duplicates
- Dropping NA values
- Dropping columns
- Tokenizing lyrics using BERT tokenizer
  - Xid and XMask
- Label
  - One-hot encoded

# Neural Network

- Architecture Overview
  - Input Layer: Accepts sequences of tokenized text with specified sequence length.
  - BERT Model: Utilizes the 'bert-base-cased' pre-trained model from the Transformers library to extract contextualized embeddings.
  - Output Layers: Consists of a dense layer with 512 neurons and ReLU activation, followed by a softmax layer for topic classification.
- Model Summary
  - Input Layers: 'input_ids' for tokenized input sequences, and 'attention_mask' to handle variable sequence lengths.
  - BERT Embeddings: Extracted using the pre-trained BERT model, capturing contextual information.
  - Dense Layers: A fully connected layer with 512 neurons and ReLU activation to capture complex patterns.
  - Output Layer: Softmax activation for multiclass classification based on the unique topics in the dataset.

- Training Configuration
  - Learning Rate Schedule: Exponential decay with an initial learning rate of 1e-5, decayed every 10,000 steps with a decay rate of 1e-6.
  - Optimizer: Adam optimizer with the specified learning rate schedule.
  - Loss Function: Categorical Crossentropy, suitable for multiclass classification tasks.
  - Evaluation Metric: Categorical Accuracy to measure the model's performance.

- Model Completion
  - The model is compiled with the defined optimizer, loss function, and evaluation metric.

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_ids (InputLayer) | [(None, 128)] | 0 | [] |
| attention_mask (InputLayer) | [(None, 128)] | 0 | [] |
| bert (TFBertMainLayer) | TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 128, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None) | 108310272 | ['input_ids[0][0]', 'attention_mask[0][0]'] |
| dense (Dense) | (None, 512) | 393728 | ['bert[0][1]'] |
| outputs (Dense) | (None, 8) | 4104 | ['dense[0][0]'] |

Total params: 108708104 (414.69 MB)
Trainable params: 108708104 (414.69 MB)
Non-trainable params: 0 (0.00 Byte)

# Distributed GPU Training

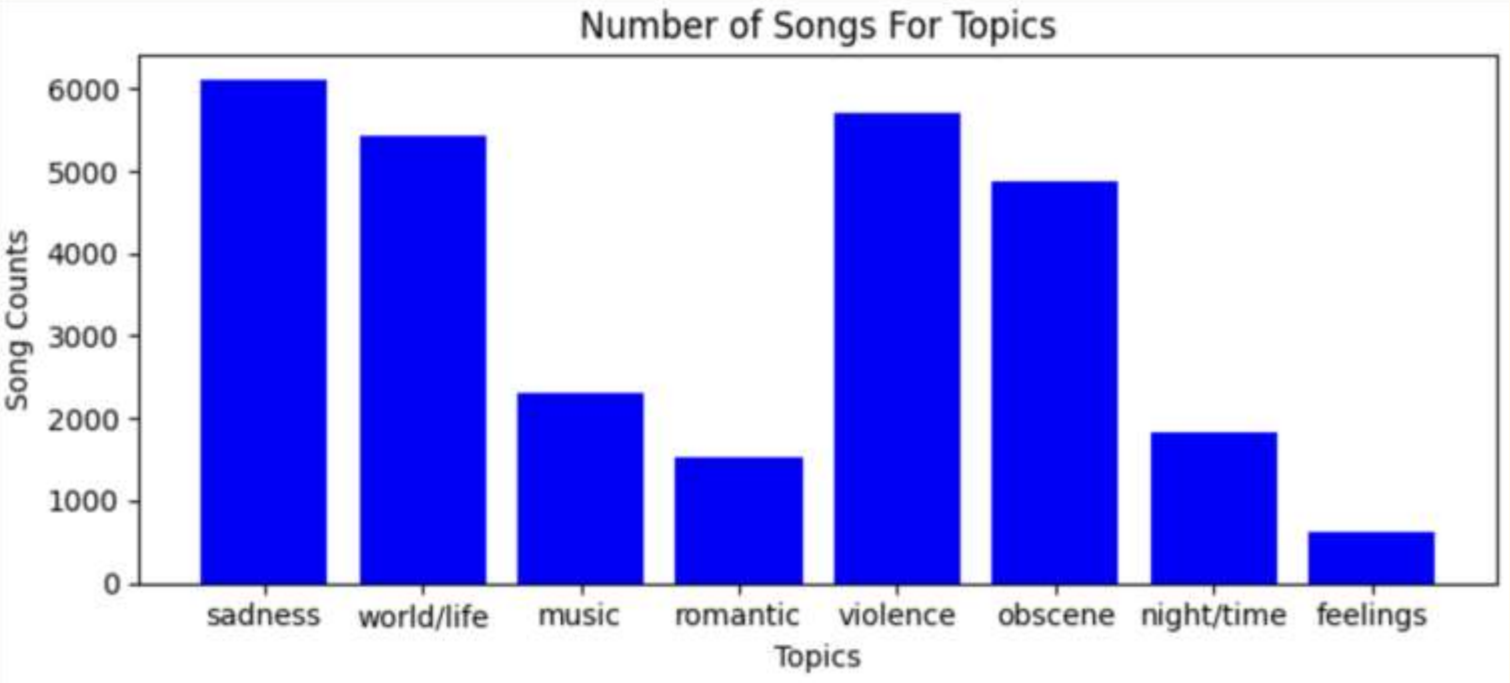
- Tensorflow's MultiWorkerMirroredStrategy
- Creation of neural network in a singular function
- Create an instance of MultiWorkerMirroredStrategy

```
strategy = tf.distribute.experimental.MultiWorkerMirroredStrategy()
```

```
with strategy.scope():
    model = giveOurModel()
```

- Define model in strategy scope
- Fit the model

# Data visualization



# Results - Accuracy of Model

- It can be seen that the model is very reliable
  - Accuracy increases from 0.6754 to 0.9121 during the course of the three epochs (entire dataset) which is good as the model is getting more and more accurate
  - Loss decreases from 0.9602 to 0.2678, which means that the model is performing well as there are fewer and fewer errors on the training data
  - Val_loss explains how well the model is going to perform on new data it is given. Decreases from 0.3815 to 0.2210 across the three epochs which is good as the model is having fewer and few errors on new data
  - Val_accuracy increases (from 0.8809 → 0.9299) which is good as the model is being able to perform more and more accurately when given new data

```
[33] history = model.fit(train_ds, validation_data=val_ds, epochs=3)

Epoch 1/3
398/398 [==============================] - 692s 2s/step - loss: 0.9602 - accuracy: 0.6754 - val_loss: 0.3815 - val_accuracy: 0.8809
Epoch 2/3
398/398 [==============================] - 656s 2s/step - loss: 0.3827 - accuracy: 0.8748 - val_loss: 0.2690 - val_accuracy: 0.9132
Epoch 3/3
398/398 [==============================] - 656s 2s/step - loss: 0.2678 - accuracy: 0.9121 - val_loss: 0.2210 - val_accuracy: 0.9299
```

**Song Track:** Ee Jagamantha Natika

**Lyrics:** “know baby hang touch weak strong know cope down stay need roses right arm sweet thorns know touch weak strong know cope down stay need touch weak strong know cope down stay need touch weak strong know cope down stay need”

```
probs_pop = loaded_model.predict(pop)
print("probs_pop", topic[np.argmax(probs_pop[0])])

1/1 [==============================] - 5s 5s/step
probs_pop romantic
```

**Song Track:** “Falling In And Out Of Love”

**Lyrics:** “fall feel touch real mind reelin round cause feel today fall fall know gonna fall try thinkin ease mind mean fall fall know gonna fall think stay maybe longer”

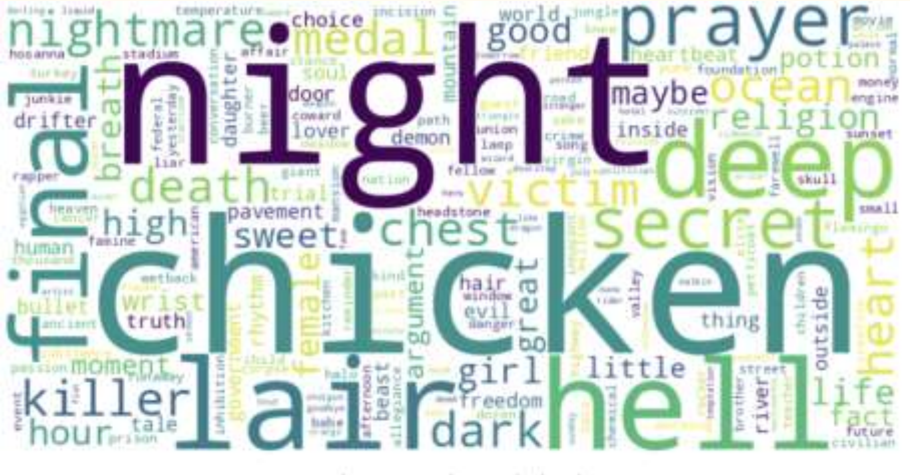
```
probs_country = loaded_model.predict(country)
print("probs_country", topic[np.argmax(probs_country[0])])
```

```
1/1 [==============================] - 0s 81ms/step
probs_country sadness
```

**Song Track:** “Cold Hands”

**Lyrics:** “splay canker brain bone decay little remain heart beat feel restrain live go gonna shoot cold hand misfortune pass long time render senseless gold hear stick limbo perch atop throne live go better fast cold hand splay canker brain boones decay little remain heart beat feel restrain live go gonna shoot cold hand live go better fast cold hand cold hand cold hand. ”

```
probs_blues = loaded_model.predict(blues)
print("probs_blues", topic[np.argmax(probs_blues[0])])

1/1 [==============================] - 0s 72ms/step
probs_blues violence
```



**Song Track:** “Rough Rider”

**Lyrics:** “rough rider cool stroker strong whiner hard night hard night hard night night night night feel break today feel break today lord feel mash today strong whiner rough rider whiney whiney night know yesterday know today strong whiner rough rider whiney whiney night chop night wear brush tonight say wear brush tonight. ”

```
probs_reggae = loaded_model.predict(reggae)
print("probs_reggae", topic[np.argmax(probs_reggae[0])])

1/1 [==============================] - 0s 56ms/step
probs_reggae night/time
```

**Song Track:** “Theme from "La Fayette”

**Lyrics:** “spit minute watch team money hurt competition start clinic repertoire live long haul stop ballin like chamique holdsclaw single black female outlaw want hardcore brother want pedestal garbage residue rybody talk righteous responsible busy judge judge know scriptures play doubt gettin bout wait right crazy catch send hollars people lafayette woods booker thousand strong start name baby test go baby phase baby test go baby baby truth recognize fight fact track better guy hella tight bedstuy come comin head high fresh zone night rip mics lonely alright stop livin life fullest thinkin positive killers sound stupid ready wasn henny cause prove fakers thinkin twice cause moment truth right raisin rything stake eyebrows want gotta come crowd pity read story take shoot dumb sharpie spot like catch future dismal sound miserable clown world cause pitifu ”


```
probs_hip_hop = loaded_model.predict(hip_hop)
print("probs_hip_hop", topic[np.argmax(probs_hip_hop[0])])

1/1 [==============================] - 0s 54ms/step
probs_hip_hop obscene
```

**Song Track:** “Always”

**Lyrics:** “lovin true things plan need help hand understand days fair hour year year lovin true things plan need help hand understand days fair hour year”

```
probs_pop_1 = loaded_model.predict(pop_1)
print("probs_pop_1", topic[np.argmax(probs_pop_1[0])])
```

```
1/1 [==============================] - 0s 104ms/step
probs_pop_1 world/life
```

**Song Track:** “Jam”

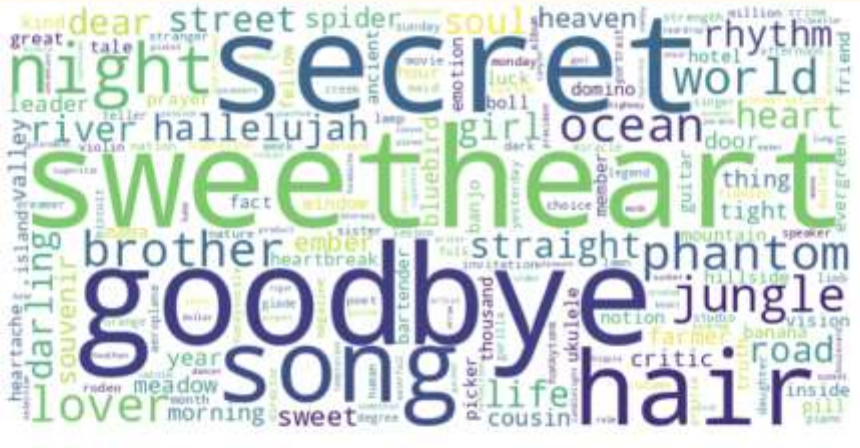
**Lyrics:** “baby lord feel touch summertime love fall home ball feel right feel right feel right feel right feel right feel right feel baby feel feel baby feel wanna clap hand wanna clap hand feel feel feel right feel right feel right feel right”

```
probs_pop_2 = loaded_model.predict(pop_2)
print("probs_pop_2", topic[np.argmax(probs_pop_2[0])])

1/1 [==============================] - 0s 96ms/step
probs_pop_2 feelings
```

**Song Track:** “An Old Fashioned Love Song”

**Lyrics:** “oldfashioned song playin radio wrap music sound promise swear hear slowly ramble need bringin cause go oldfashioned song sure write oldfashioned song comin threepart weave dream listen even light underscore affair tenderness feel come know swear hear slowly ramble need bringin cause go oldfashioned song comin threepart oldfashioned song sure write oldfashioned song comin threepart oldfashioned song sure write weave dream listen song song comin song sure write song comin song oldfashioned song sure write oldfashioned song song comin comin threepart oldfashioned song song sure write oldfashioned song song comin comin threepart oldfashioned song song sure write”

```
probs_pop_3 = loaded_model.predict(pop_3)
print("probs_pop_3", topic[np.argmax(probs_pop_3[0])])

1/1 [==============================] - 0s 91ms/step
probs_pop_3 music
```



# Conclusion

- ➢ Familiarize ourselves with the HPC and training running models in parallel
- ➢ Training and testing a model
- ➢ Utilizing Pandas, NumPy, Scikit-learn, Transformers, and Tensorflow
- ➢ Implementing model successfully
  - ○ Model has high accuracy based on the tests we have done
- ➢ Overall high success and accuracy in creating a model
- ➢ Very useful to predict topic of songs without any biases

# Future Works

- ➢ Predict more song topics, use different song categorization system
- ➢ Positive or negative sentiments → topic of the song
- ➢ Finding more training data

# Sources


[1] “What is text mining?,” IBM, https://www.ibm.com/topics/text-mining (accessed Dec. 6, 2023).

[2] S. A. Metwalli, “NLP basics: Data Mining vs. text mining,” Medium, https://towardsdatascience.com/nlp-basics-data-mining-vs-text-mining-9e21389986b4 (accessed Dec. 1, 2023).

[3] A. Kao and S. R. Poteet, Natural Language Processing and Text Mining. Scholars Portal.

[4] R. Talib, M. K. Hanif, and S. Ayesha, “Text Mining: Techniques, Applications and Issues,” the SAI, https://thesai.org/Downloads/Volume7No11/Paper_53-Text_Mining_Techniques_Applications_and_Issues.pdf (accessed Dec. 1, 2023).

[5] S. Shahane, “Music dataset : 1950 to 2019,” Kaggle, https://www.kaggle.com/datasets/saurabhshahane/music-dataset-1950-to-2019/ (accessed Dec. 1, 2023).

[6] Apache Hadoop, https://hadoop.apache.org/ (accessed Dec. 1, 2023).

[7] “What is python? executive summary,” Python.org, https://www.python.org/doc/essays/blurb/ (accessed Dec. 1, 2023).

[8] “NumPy documentation#,” NumPy documentation - NumPy v1.26 Manual, https://numpy.org/doc/stable/ (accessed Dec. 1, 2023).

[9] “Pandas documentation#,” pandas documentation - pandas 2.1.3 documentation, https://pandas.pydata.org/docs/ (accessed Dec. 1, 2023).

[10] Z. Keita, “An introduction to using transformers and hugging face,” DataCamp, https://www.datacamp.com/tutorial/an-introduction-to-using-transformers-and-hugging-face (accessed Dec. 1, 2023).

[11] “Why tensorflow,” TensorFlow, https://www.tensorflow.org/about (accessed Dec. 2, 2023).

[12] “Learn,” scikit, https://scikit-learn.org/stable/ (accessed Dec. 3, 2023).

[13] L. Adeofe, "Artificial intelligence and subjective experience," Proceedings of Southcon '95, Fort Lauderdale, FL, USA, 1995, pp. 403-408, doi: 10.1109/SOUTHC.1995.516138.

# Thank you for listening!!

If you have any questions about our project, please reach out anyone of us.