NLP SONG GENERATION

By: Andrew Shehata, Malak El-Khashab

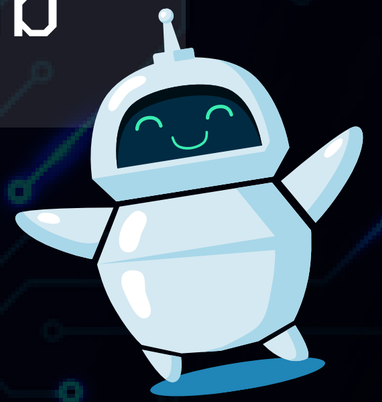
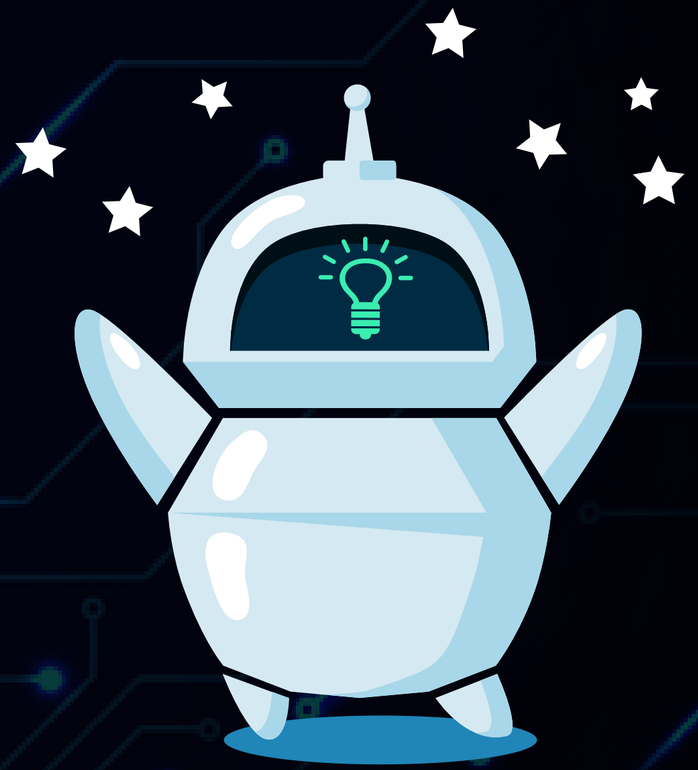


TABLE OF CONTENT



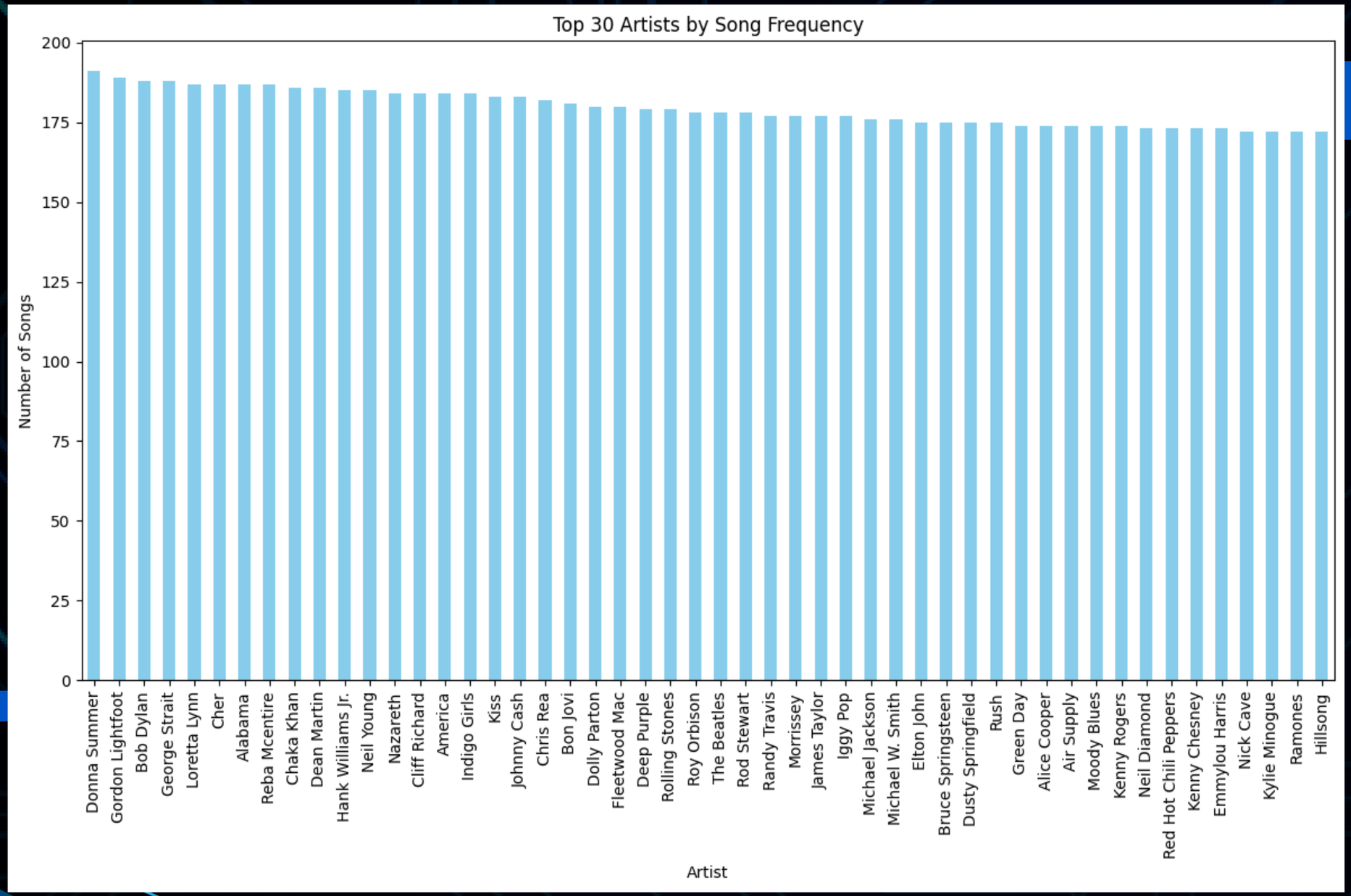
INTRODUCTION

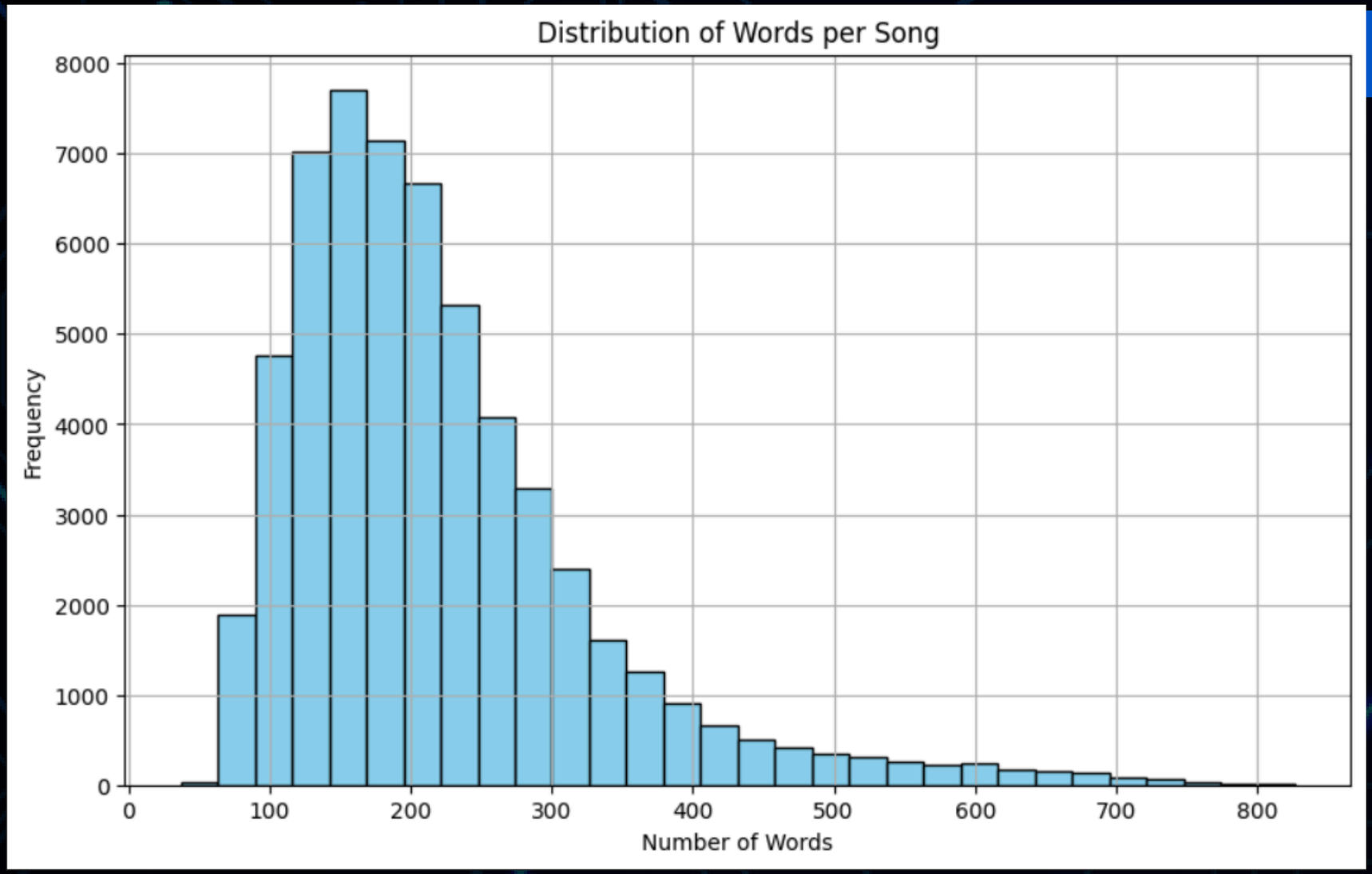
- **Project Objective:** Generate song lyrics based on a specific artist's style and given lyrics.
- **Technology Used:** Advanced NLP techniques and a pretrained T5 model.
- **Goals:**
  - Emulate the unique lyrical style of various artists.
  - Generate coherent and creative lyrics matching the input and artist's style.
- **Key Steps:**
  - Data pre-processing.
  - Model training.
  - Fine-tuning for stylistic accuracy.
- **Outcome:** Achieved relevant and stylistically accurate generated lyrics.

DATA PRE-PROCESSING

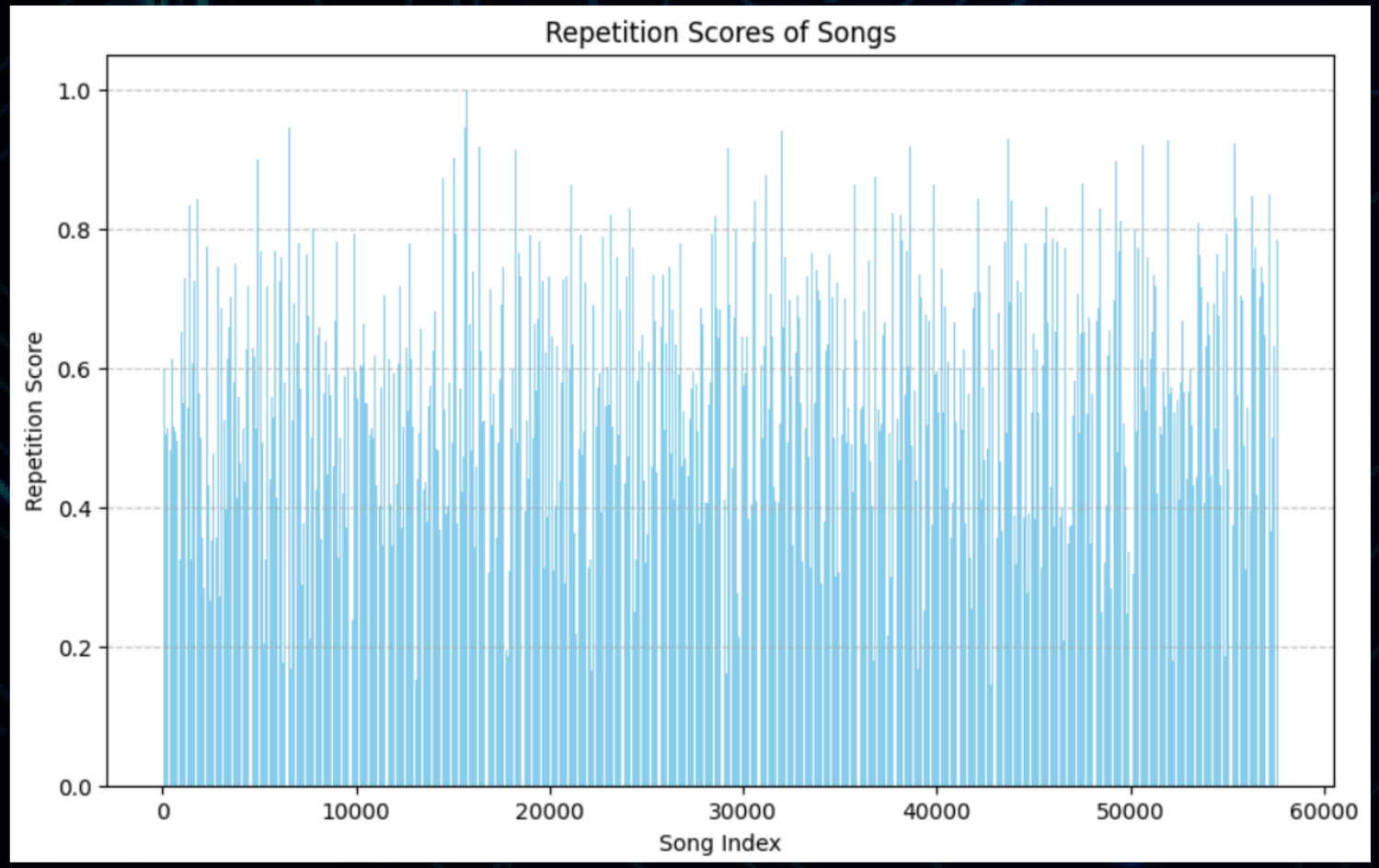
DATA PRE-PROCESSING

- Duplicates check
- Tokenization
- Stop word removal
- TF-IDF
- Word Cloud
- Some Graphs

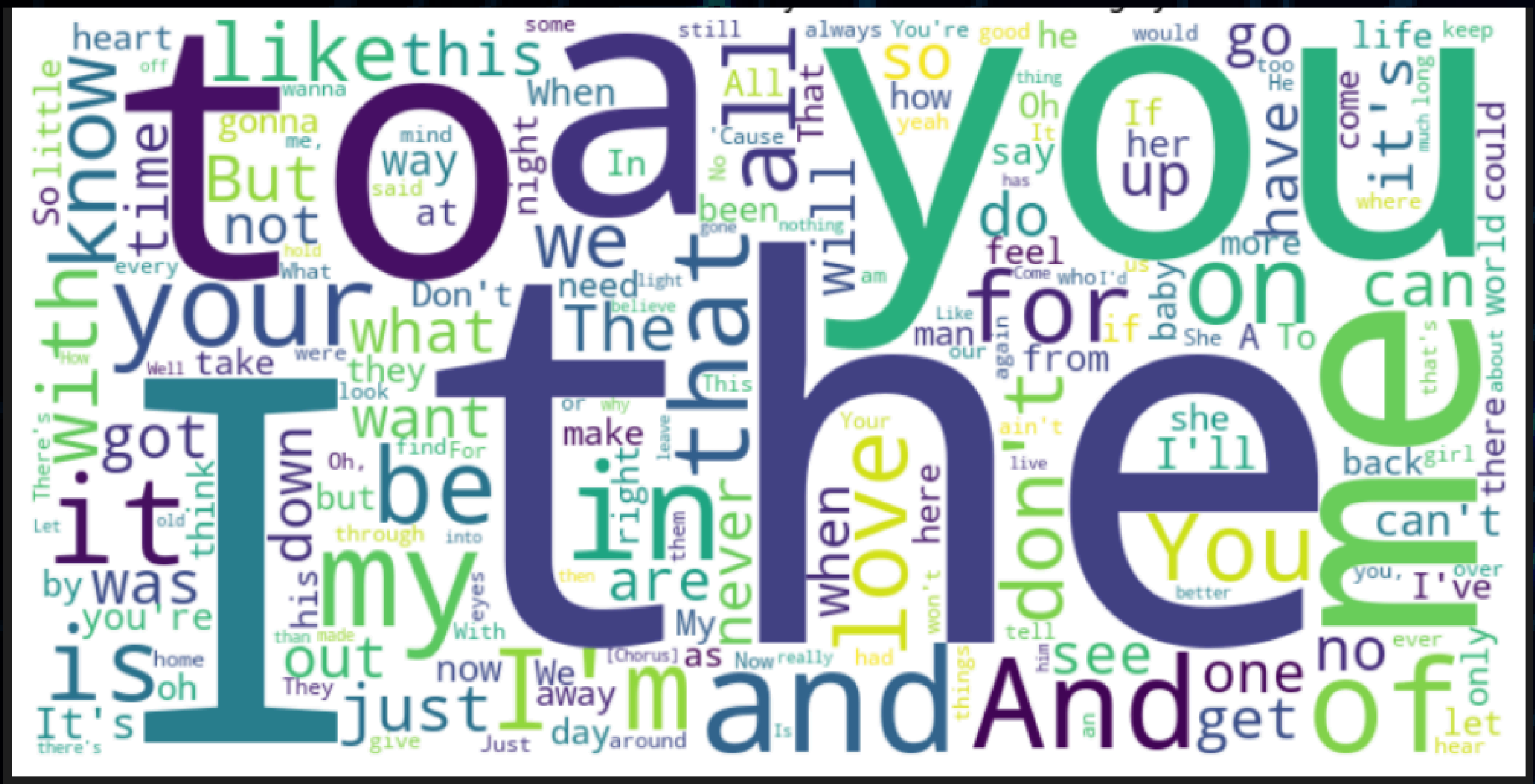

Top 30 Artists by Song Frequency
Number of Songs
0
25
50
75
100
125
150
175
200
Donna Summer
Gordon Lightfoot
Bob Dylan
George Strait
Loretta Lynn
Cher
Alabama
Reba Mcentire
Chaka Khan
Dean Martin
Hank Williams Jr.
Neil Young
Nazareth
Cliff Richard
America
Indigo Girls
Kiss
Johnny Cash
Chris Rea
Bon Jovi
Dolly Parton
Fleetwood Mac
Deep Purple
Rolling Stones
Roy Orbison
The Beatles
Rod Stewart
Randy Travis
Morrissey
James Taylor
Iggy Pop
Michael Jackson
Michael W. Smith
Elton John
Bruce Springsteen
Dusty Springfield
Rush
Green Day
Alice Cooper
Air Supply
Moody Blues
Kenny Rogers
Neil Diamond
Red Hot Chili Peppers
Kenny Chesney
Emmylou Harris
Nick Cave
Kylie Minogue
Ramones
Hillsong
Artist


Distribution of Words per Song
Frequency
Number of Words
0
1000
2000
3000
4000
5000
6000
7000
8000
0
100
200
300
400
500
600
700
800

$$\text{Repetition Score} = \frac{\text{Number of Repeated Words}}{\text{Total Number of Words}}$$



BEFORE DATA CLEANING



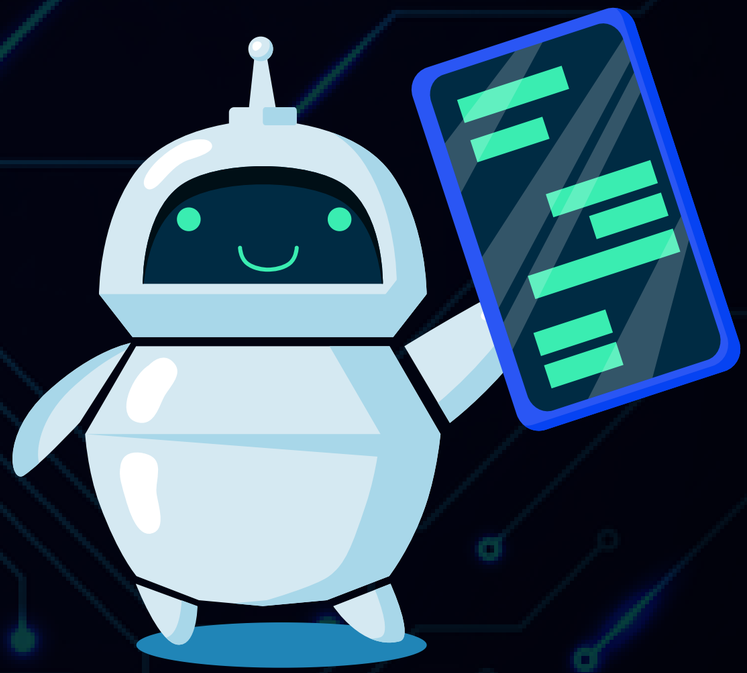
AFTER DATA CLEANING

ARRAYCACHES

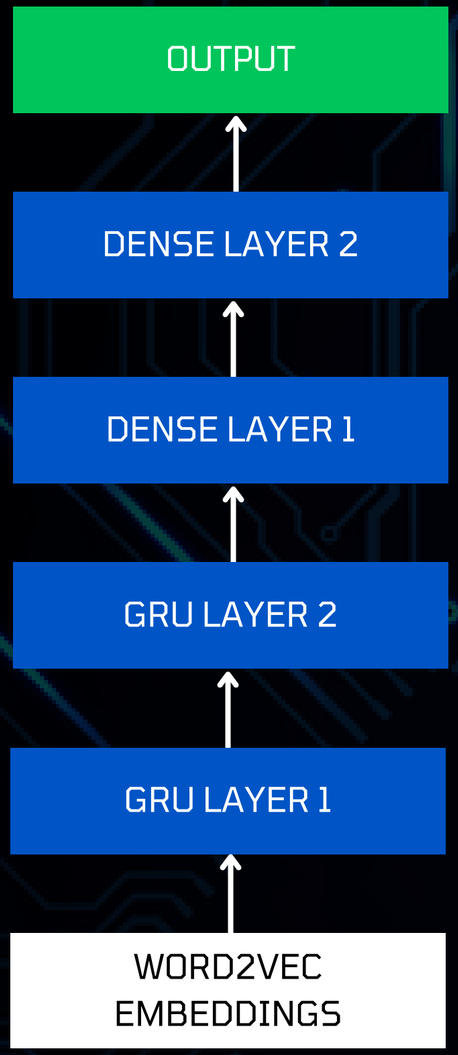
APPROACHES

Our created model/pipeline

Pre-trained Model

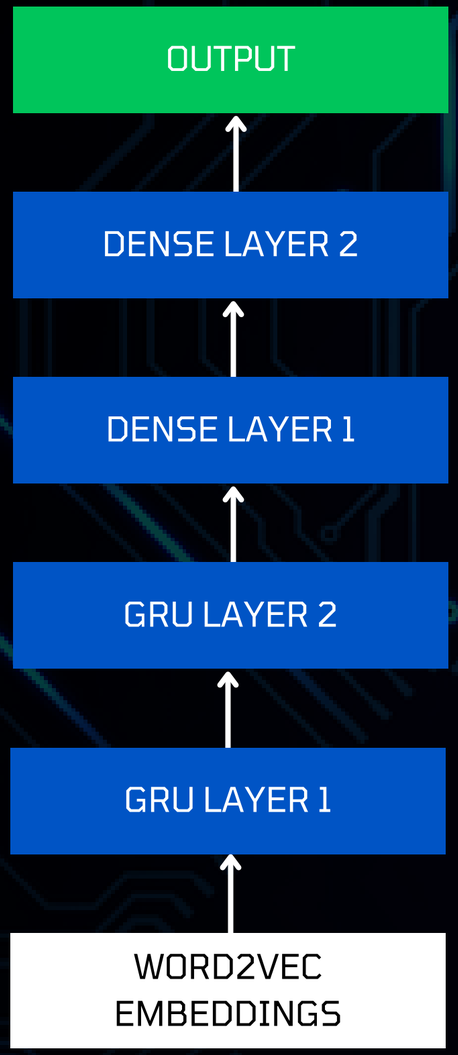


CREATED MODEL ARCHITECTURE



- **Gated Recurrent Units (GRUs):**
  - Address vanishing gradient problem in RNNs.
  - Two GRU layers capture short-term & long-term dependencies:
    - First GRU (return_sequences=True) outputs a sequence.
    - Second GRU (return_sequences=False) summarizes the sequence.
- **Hidden Layer for Feature Extraction:**
  - Dense layer with 64 units & ReLU activation extracts features from GRU outputs.
- **Output Layer and Vocabulary:**
  - Final Dense layer with softmax activation predicts next word.
  - Number of units matches vocabulary size (all unique words).
  - Softmax activation gives probability distribution over vocabulary.

CREATED MODEL ARCHITECTURE




```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_14 (GRU)                (None, 224, 128)          53760

 gru_15 (GRU)                (None, 128)               99072

 dense_14 (Dense)            (None, 64)                8256

 dense_15 (Dense)            (None, 109)               7085

=================================================================
Total params: 168173 (656.93 KB)
Trainable params: 168173 (656.93 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

RESULTS AND FINDINGS

Including 1 artist
“Donna Summer”

```
Epoch 1/10
5/5 [==============================] - 5s 123ms/step - loss: 0.6901 - accuracy: 0.0072
Epoch 2/10
5/5 [==============================] - 1s 125ms/step - loss: 0.6670 - accuracy: 0.0072
Epoch 3/10
5/5 [==============================] - 1s 122ms/step - loss: 0.5388 - accuracy: 0.0072
Epoch 4/10
5/5 [==============================] - 1s 119ms/step - loss: 0.2464 - accuracy: 0.0072
Epoch 5/10
5/5 [==============================] - 1s 119ms/step - loss: 0.1530 - accuracy: 0.0072
Epoch 6/10
5/5 [==============================] - 1s 118ms/step - loss: 0.0973 - accuracy: 0.0072
Epoch 7/10
5/5 [==============================] - 1s 120ms/step - loss: 0.0674 - accuracy: 0.0072
Epoch 8/10
5/5 [==============================] - 1s 121ms/step - loss: 0.0559 - accuracy: 0.0290
Epoch 9/10
5/5 [==============================] - 1s 127ms/step - loss: 0.0523 - accuracy: 0.1087
Epoch 10/10
5/5 [==============================] - 1s 120ms/step - loss: 0.0509 - accuracy: 0.1087
```

RESULTS AND FINDINGS

```
Epoch 1/10
80/80 [==============================] - 13s 110ms/step - loss: 0.1655 - accuracy: 0.0208
Epoch 2/10
80/80 [==============================] - 9s 107ms/step - loss: 0.0073 - accuracy: 0.0513
Epoch 3/10
80/80 [==============================] - 9s 109ms/step - loss: 0.0072 - accuracy: 0.0431
Epoch 4/10
80/80 [==============================] - 9s 113ms/step - loss: 0.0072 - accuracy: 0.0466
Epoch 5/10
80/80 [==============================] - 9s 110ms/step - loss: 0.0072 - accuracy: 0.0478
Epoch 6/10
80/80 [==============================] - 9s 110ms/step - loss: 0.0072 - accuracy: 0.0462
Epoch 7/10
80/80 [==============================] - 9s 112ms/step - loss: 0.0072 - accuracy: 0.0517
Epoch 8/10
80/80 [==============================] - 9s 111ms/step - loss: 0.0072 - accuracy: 0.0493
Epoch 9/10
80/80 [==============================] - 9s 113ms/step - loss: 0.0072 - accuracy: 0.0501
Epoch 10/10
80/80 [==============================] - 9s 113ms/step - loss: 0.0072 - accuracy: 0.0525
```

Including more artists

TESTING OUR MODEL

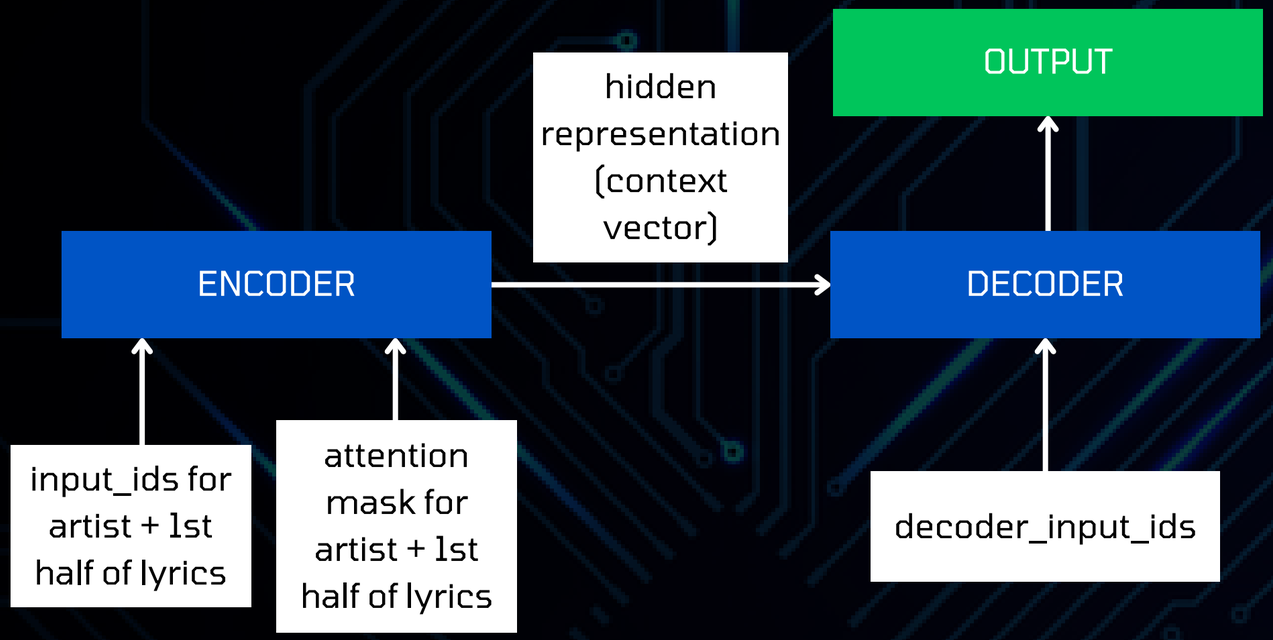
- Identical words due to the low accuracy.
- It's important to note that the quality of the generated text is directly proportional to the model's accuracy.

```
[['queen', 'day', 'queen', 'night', 'dressed', 'head', 'toe']]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
queen day queen night dressed head toe oh oh oh oh oh oh oh oh oh oh
```

ADVANTAGES/DISADVANTAGES

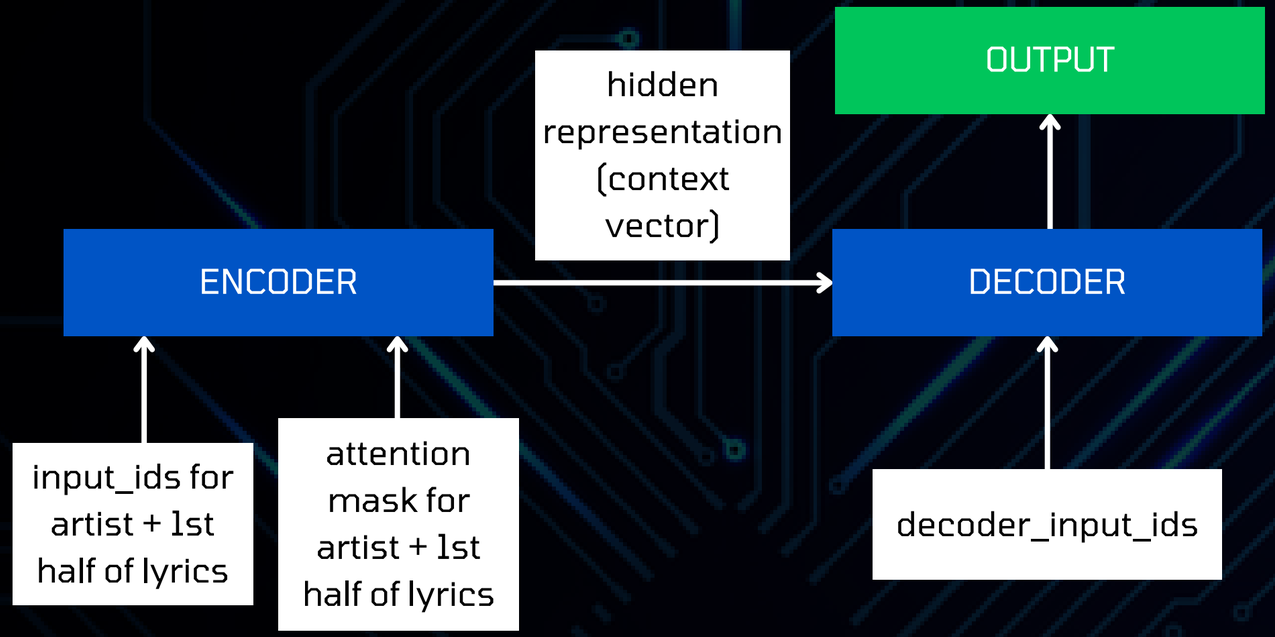
- *Advantages:*
  - Easier to understand and analyze how the model processes sequences due to its simpler structure.
  - Having the full control over the model architecture and can experiment with hyperparameters for optimization.
  - Training a custom GRU model might require less computational power compared to pre-trained Transformers.
- *Disadvantages and Limitations:*
  - May not achieve the same level of performance as pre-trained Transformers in complex tasks like lyric generation.

T5 PRE-TRAINED MODEL



- **T5:**
  - Handles various text tasks, including lyric generation.
  - Takes additional instructions to tailor outputs.
- **Conditional T5:**
  - Provide instructions alongside the main input sequence.
- **Encoder-Decoder Architecture:**
  - Encoder: Processes input & conditions, captures meaning.
  - Decoder: Generates target text (lyrics) influenced by conditions.
- **Fine-tuning on Lyric Data:**
  - Train a pre-trained T5 model on lyrics with conditions.
  - Improves understanding of lyric patterns and vocabulary.

T5 PRE-TRAINED MODEL




```
Model: "tft5_for_conditional_generation"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 shared (Embedding)          multiple                  16449536

 encoder (TFT5MainLayer)     multiple                  35330816

 decoder (TFT5MainLayer)     multiple                  41625344

=================================================================
Total params: 60506624 (230.81 MB)
Trainable params: 60506624 (230.81 MB)
Non-trainable params: 0 (0.00 Byte)
```

T5 PRE-TRAINED MODEL

- Approach A:
  - **model.fit()**
    - Converted data to NumPy format for efficiency in machine learning.
    - Defined optimizer, loss function, and accuracy metrics.
    - Compiled the model with defined elements for training.
    - Trained on provided data and monitored performance on unseen test data.

T5 PRE-TRAINED MODEL

- Approach B:
  - **training a sequence-to-sequence model with Hugging Face Transformers (for loop)**
    - Converted dataset to TensorFlow format for efficient processing and shuffling.
    - Defined train step: *process batch data, calculate loss, update gradients, update weights, update accuracy.*
    - Defined evaluate step: *process batch data, calculate loss, update accuracy (for unseen data).*
    - Trained in epochs: *iteratively train on batches, evaluate on unseen data after each epoch.*

RESULTS AND FINDINGS (APPROACH 1)

On Top 5 artists

+ Code + Markdown

[68]:
```
model.fit(train_dataset, validation_data=test_dataset,batch_size=16, epochs=3)
```

```
Epoch 1/3
WARNING: AutoGraph could not transform <function infer_framework at 0x7dae71a17400> and will run it as-is.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
48/48 [==============================] - 164s 1s/step - loss: 4.4051 - sparse_categorical_accuracy: 0.4947 - val_loss: 1.2440 - val_sparse_categorical_accuracy: 0.7962
Epoch 2/3
48/48 [==============================] - 35s 728ms/step - loss: 1.6087 - sparse_categorical_accuracy: 0.7422 - val_loss: 1.1684 - val_sparse_categorical_accuracy: 0.8048
Epoch 3/3
48/48 [==============================] - 35s 729ms/step - loss: 1.2830 - sparse_categorical_accuracy: 0.7863 - val_loss: 1.1071 - val_sparse_categorical_accuracy: 0.8108
```

[68]: `<tf_keras.src.callbacks.History at 0x7dae2410c820>`

RESULTS AND FINDINGS (APPROACH 1)

Full Dataset on kaggle with GPU T4X2

RESULTS AND FINDINGS (APPROACH 1)

Full Dataset on kaggle with GPU P100

[33]:
```
model.fit(train_dataset, validation_data=test_dataset,batch_size=16, epochs=3)
```

```
Epoch 1/3
WARNING: AutoGraph could not transform <function infer_framework at 0x7b0f774a2dd0> and will run it as-is.
Cause: for/else statement not yet supported
To silence this warning, decorate the function with @tf.autograph.experimental.do_not_convert
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1716047933.435422      89 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
2883/2883 [==============================] - 2310s 764ms/step - loss: 1.1300 - sparse_categorical_accuracy: 0.8108 - val_loss: 0.9427 - val_sparse_categorical_accuracy: 0.8343
Epoch 2/3
2883/2883 [==============================] - 2183s 757ms/step - loss: 0.9780 - sparse_categorical_accuracy: 0.8289 - val_loss: 0.9069 - val_sparse_categorical_accuracy: 0.8387
Epoch 3/3
2883/2883 [==============================] - 2186s 758ms/step - loss: 0.9477 - sparse_categorical_accuracy: 0.8329 - val_loss: 0.8882 - val_sparse_categorical_accuracy: 0.8412
```

[33]:
```
<tf_keras.src.callbacks.History at 0x7b0ea44c1fc0>
```

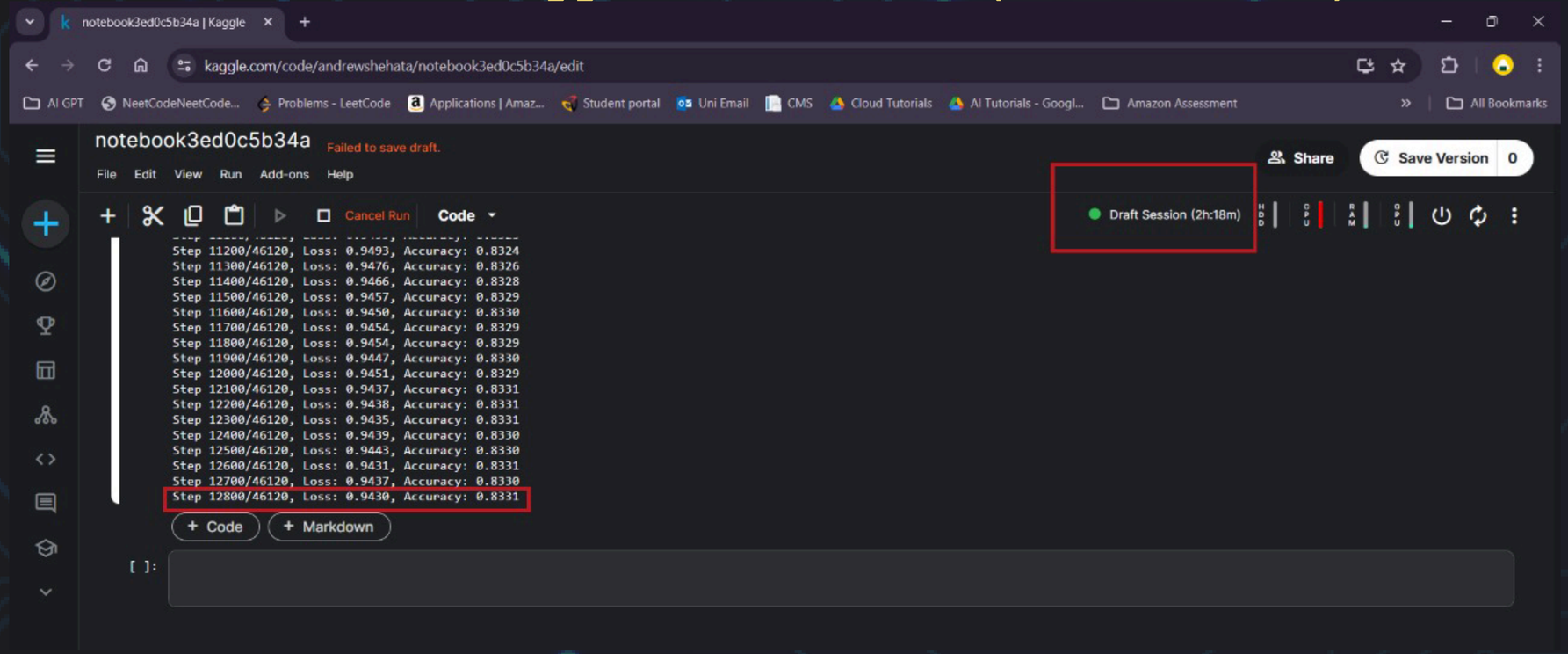
RESULTS AND FINDINGS (APPROACH 2)

On Top 5 artists

```
TRAINING
Epoch 1/3
Step 100/754, Loss: 2.9951, Accuracy: 0.6227
Step 200/754, Loss: 2.0151, Accuracy: 0.7241
Step 300/754, Loss: 1.6826, Accuracy: 0.7564
Step 400/754, Loss: 1.5234, Accuracy: 0.7707
Step 500/754, Loss: 1.4276, Accuracy: 0.7794
Step 600/754, Loss: 1.3637, Accuracy: 0.7860
Step 700/754, Loss: 1.3252, Accuracy: 0.7892
EVALUATION
Validation Loss: 0.9789, Validation Accuracy: 0.8265
TRAINING
Epoch 2/3
Step 100/754, Loss: 0.9598, Accuracy: 0.8302
Step 200/754, Loss: 0.9571, Accuracy: 0.8305
Step 300/754, Loss: 0.9486, Accuracy: 0.8316
Step 400/754, Loss: 0.9558, Accuracy: 0.8298
Step 500/754, Loss: 0.9559, Accuracy: 0.8300
Step 600/754, Loss: 0.9609, Accuracy: 0.8292
Step 700/754, Loss: 0.9592, Accuracy: 0.8291
EVALUATION
Validation Loss: 0.9464, Validation Accuracy: 0.8313
TRAINING
Epoch 3/3
Step 100/754, Loss: 0.9786, Accuracy: 0.8276
...
Step 600/754, Loss: 0.9045, Accuracy: 0.8372
Step 700/754, Loss: 0.9134, Accuracy: 0.8352
EVALUATION
Validation Loss: 0.9322, Validation Accuracy: 0.8326
```

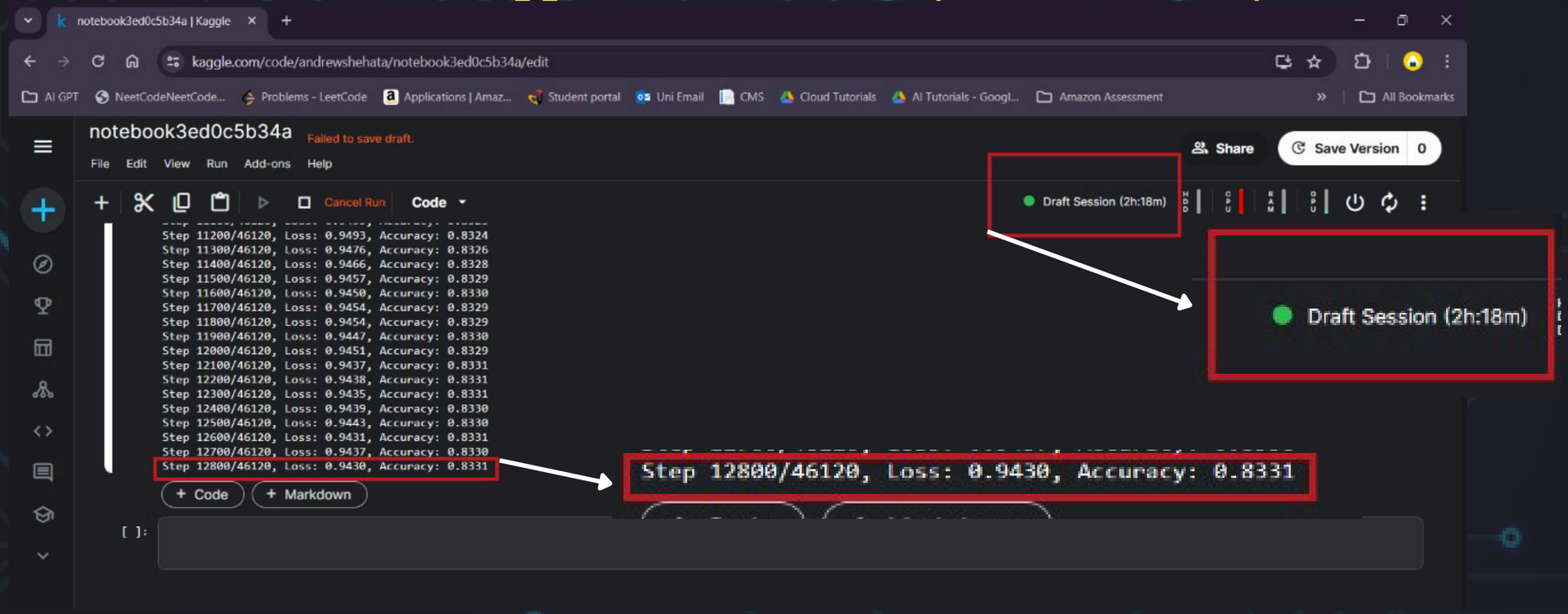
RESULTS AND FINDINGS [APPROACH 2]

Full Dataset on kaggle with GPU P100 (***unsuccessful***)



RESULTS AND FINDINGS [APPROACH 2]

Full Dataset on kaggle with GPU P100 (***unsuccessful***)



TESTING OUR MODEL

- Feed lyric snippet to tokenizer (text to numbers).
- Use model to generate continuation with attributes:
  - *max_new_tokens:* limit generated text length (e.g., 40 words).
  - *do_sample:* inject randomness for diverse outputs.
  - *top_k:* consider only top 30 probable tokens for next word.
  - *top_p:* balance high-probability with some exploration.
- Decode generated sequence back to human-readable text (numbers to text), excluding special tokens.

ADVANTAGES/DISADVANTAGES

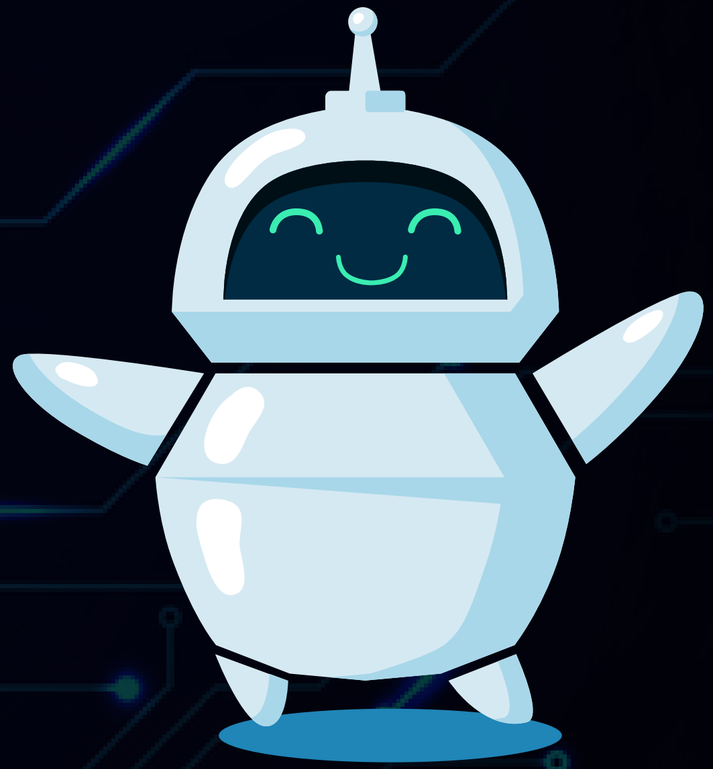
- *Advantages:*
  - Leverages the pre-trained knowledge of TFT5, potentially leading to better lyric generation quality.
- *Disadvantages and Limitations:*
  - Understanding how a pre-trained Transformer arrives at its predictions can be challenging.
  - Have less control over the overall model architecture as you're primarily fine-tuning an existing model.
  - Training even with fine-tuning might require more computational resources compared to a custom GRU model.

CHALLENGES AND LIMITATIONS

- **Small Dataset (56,000 entries):**
  - Training takes too long, especially with small batches for better understanding.
  - Lowers accuracy when not using a pre-trained model.
- **Hardware Constraints (laptops):**
  - Insufficient RAM and CPU lead to crashes during training .
- **Spelling Check Limitation:**
  - Takes too long (4 hours) for the entire dataset.
  - Song lyrics may not require strict spelling (e.g., "oh" vs "ohhhhh").
- **Sparse Feature Set:**
  - Lacks information like genre, duration, etc.
  - Needs web scraping or external data integration for better analysis.

CONCLUSION

- **Interpretability & Customization:** Choose a custom GRU model (but expect lower performance and longer training).
- **Speed & Performance:** Use model.fit with a pre-trained TFT5 (balances ease of use with results).
- **Maximum Control & Expertise:** Implement a custom training loop with TFT5 (requires expertise but offers most control).



THANK YOU

ANY QUESTIONS?