



Search

Home

Library

Lyrics Generation with Deep Learning Models

Group 14: Jiaqin Wu, Kefan Yu, Naihuan Jing





Table of contents



01

Background &
Research Objectives

02

Data Gathering &
Preprocessing

03

EDA

04

Lyrics Generator with
Starting Sentences

05

Lyrics Generator
with Artists

06

Lyrics Generator
with Titles

07

Limitations &
Discussions

08

Conclusion



Lyrics Generation with Deep Learning Models

Search

Home

Library

01



Background & Research Objectives



Background

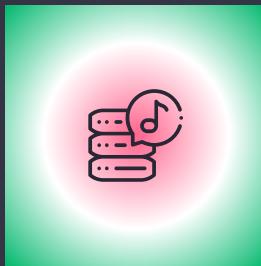


Today, music is a universal source of joy and relaxation, featuring a rich variety of popular songs that connect with audiences worldwide. More than just a pleasant sound, the blend of lyrics, rhythms, and stylistic elements plays a crucial role in a song's popularity. Artists use their unique styles, and even the song's title can greatly impact how it's received.

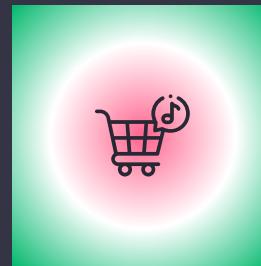




Research Objectives



RNN



Pre-trained
Models





Research Objectives



- Lyrics Generator with Starting Sentences
- Lyrics Generator with Artists
- Lyrics Generator with Titles





Lyrics Generation with Deep Learning Models



Search



Home



Library

02



Data Gathering & Preprocessing



Data Gathering



Source: Kaggle

Features: artist, title, lyrics

Methodology: Download + manually input more records

Final Shape: 1099 records





Data Preprocessing



- Created a Corpus of Lyrics text
- Removed the unrequired characters
- Created a dictionary to map characters and their indices
- Splitted the corpus into smaller sentences of equal length





Preprocessed Data



- The preprocessed corpus would only contain lowercase English alphabets, numbers, and a few punctuations
- Encoding the corpus and creating sequences of equal lengths for features and their corresponding targets
- Each feature and target will represent the mapped index from the dictionary of unique characters





Lyrics Generation with Deep Learning Models



Search



Home



Library

03 •



EDA



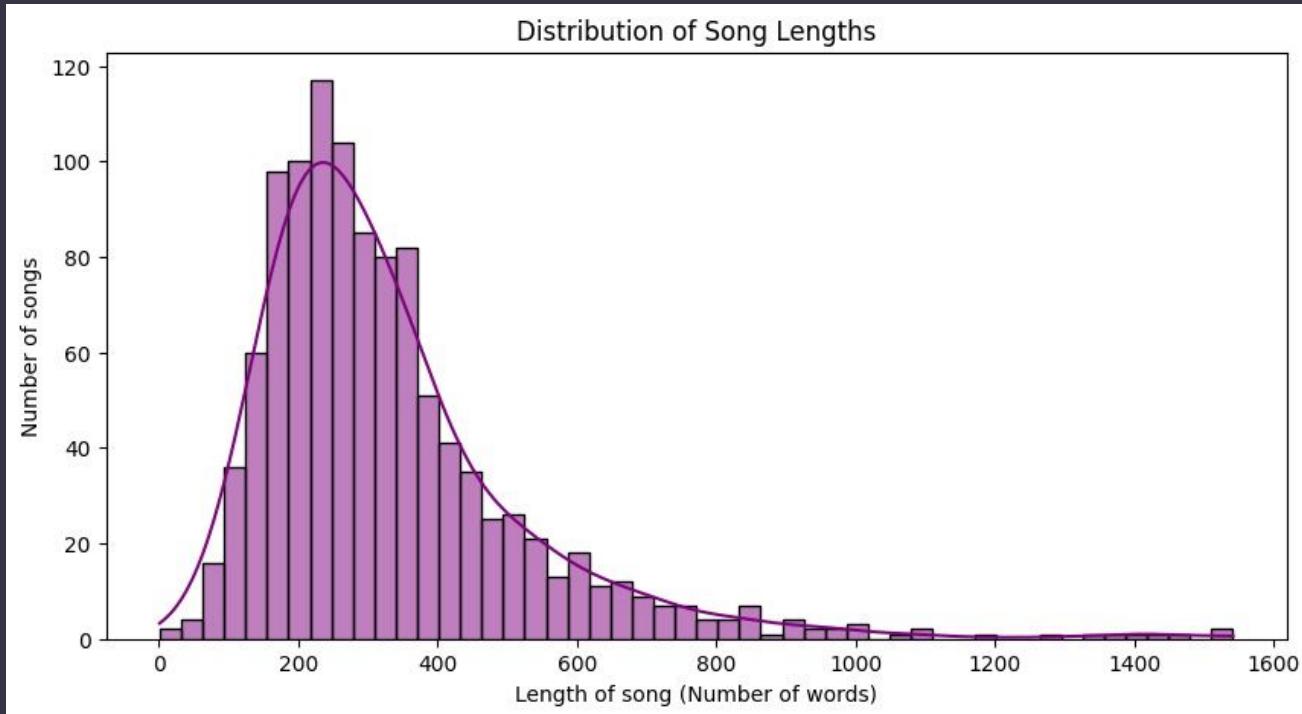
Number of characters/words/lines



	Num_char	Num_word	Num_line
count	1099.000000	1099.000000	1099.000000
mean	1675.098271	390.432211	52.649682
std	1007.412870	244.696049	25.612499
min	1.000000	1.000000	1.000000
25%	1028.000000	231.000000	35.000000
50%	1422.000000	328.000000	48.000000
75%	2014.000000	473.000000	64.000000
max	7846.000000	1828.000000	224.000000

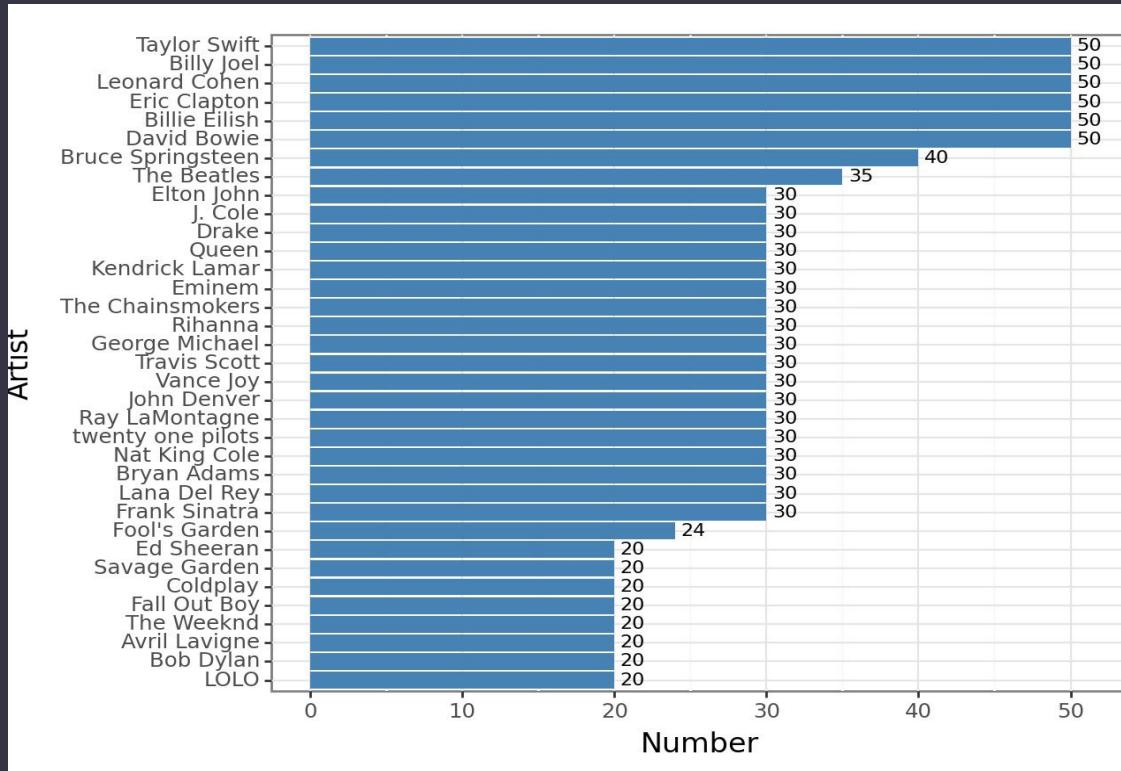


Distribution of Song Lengths





Distribution of Artists





Word Cloud





Lyrics Generation with Deep Learning Models



Search



Home



Library

04



Lyrics Generator
with Starting Sentences



RNN Models



- RNN-1: One Hidden Layer + Dropout Rate 0.2
- RNN-2: One Hidden Layer + Dropout Rate 0.3
- RNN-3: Two Hidden Layers + Dropout Rate 0.2





Lyrics Generation with Deep Learning Models



Search



Home



Library

RNN-1



One Hidden Layer + Dropout Rate 0.2



Model Description



Model: "sequential_3"

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 256)	264192
dropout_3 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 47)	12079

Total params: 276271 (1.05 MB)

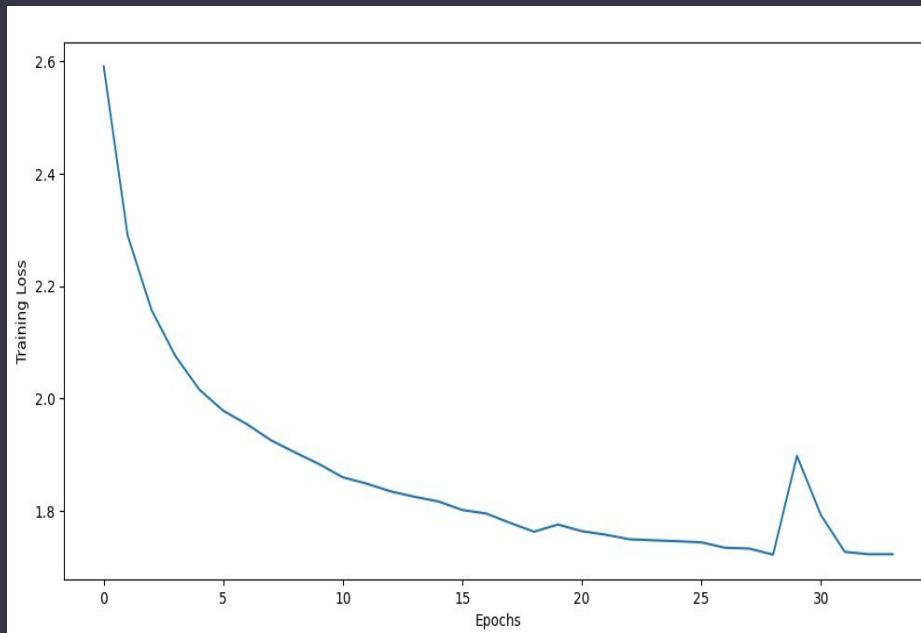
Trainable params: 276271 (1.05 MB)

Non-trainable params: 0 (0.00 Byte)





Model Performance





Lyrics Generation Function



```
# Define a function to generate the text
def generate_text(model, starter, length, temperature=1.0):
    model.eval()
    generated = starter

    with torch.no_grad():
        for _ in range(length):
            seed = [mapping[char] for char in starter]
            x_pred = torch.tensor(seed, dtype=torch.float32).view(1, len(seed), 1).to('cuda')
            x_pred /= float(L_symb)

            prediction = model(x_pred)

            # Ensure prediction is a 2D tensor
            prediction = prediction.squeeze()

            # Applying temperature to control randomness
            prediction = prediction / temperature
            prediction = F.softmax(prediction, dim=0)

            # Sample from the predicted probabilities
            index = torch.multinomial(prediction, 1).item()
            next_char = reverse_mapping[index]

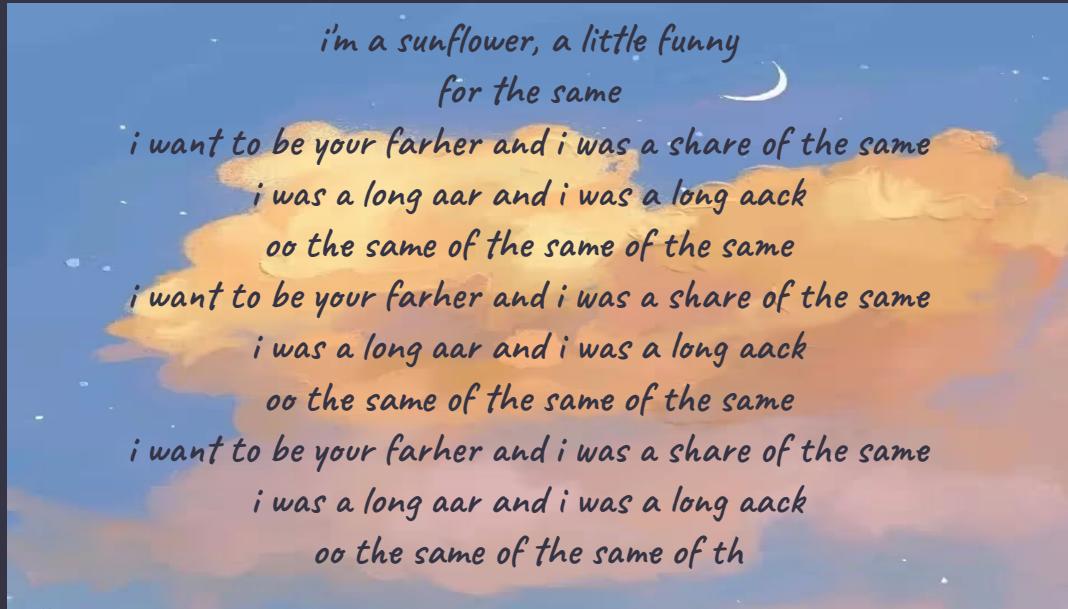
            # Generating new text
            generated += next_char
            starter = starter[1:] + next_char

    return generated
```



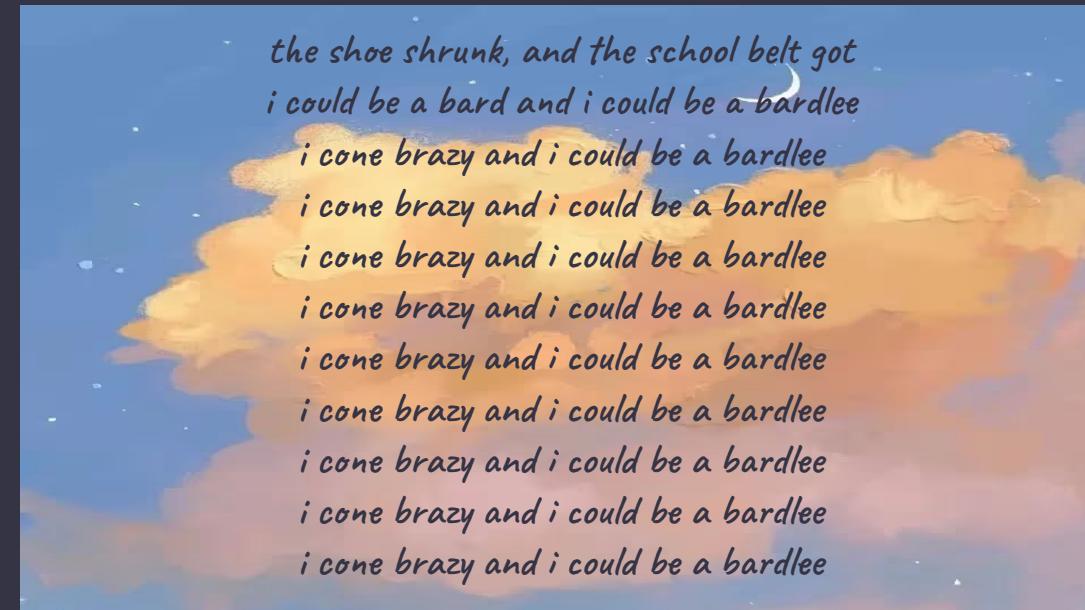


Generated Lyrics





Generated Lyrics





Lyrics Generation with Deep Learning Models



Search



Home



Library

RNN-2



One Hidden Layer + Dropout Rate 0.3



Model Description



Model: "sequential_4"

Layer (type)	Output Shape	Param #
lstm_5 (LSTM)	(None, 256)	264192
dropout_4 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 47)	12079

=====
Total params: 276271 (1.05 MB)

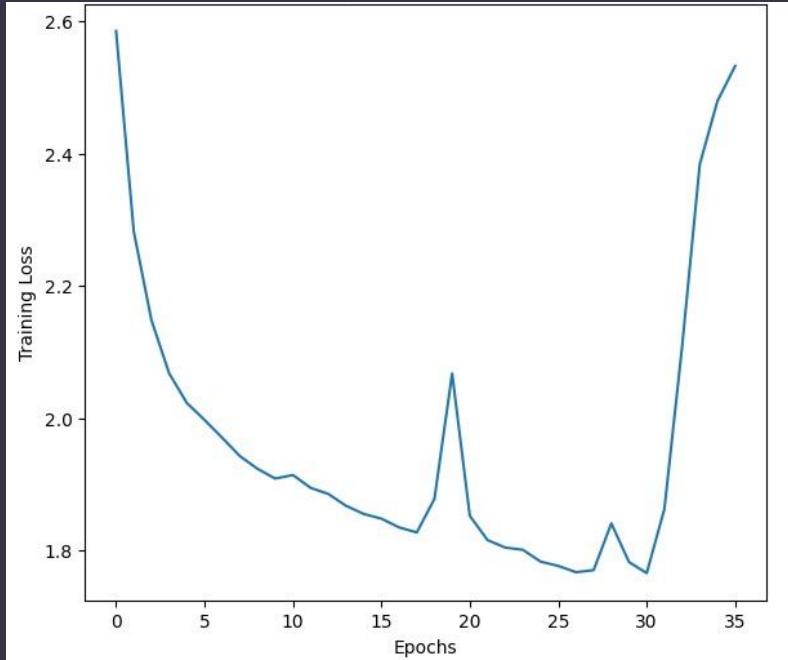
Trainable params: 276271 (1.05 MB)

Non-trainable params: 0 (0.00 Byte)





Model Performance





Generated Lyrics

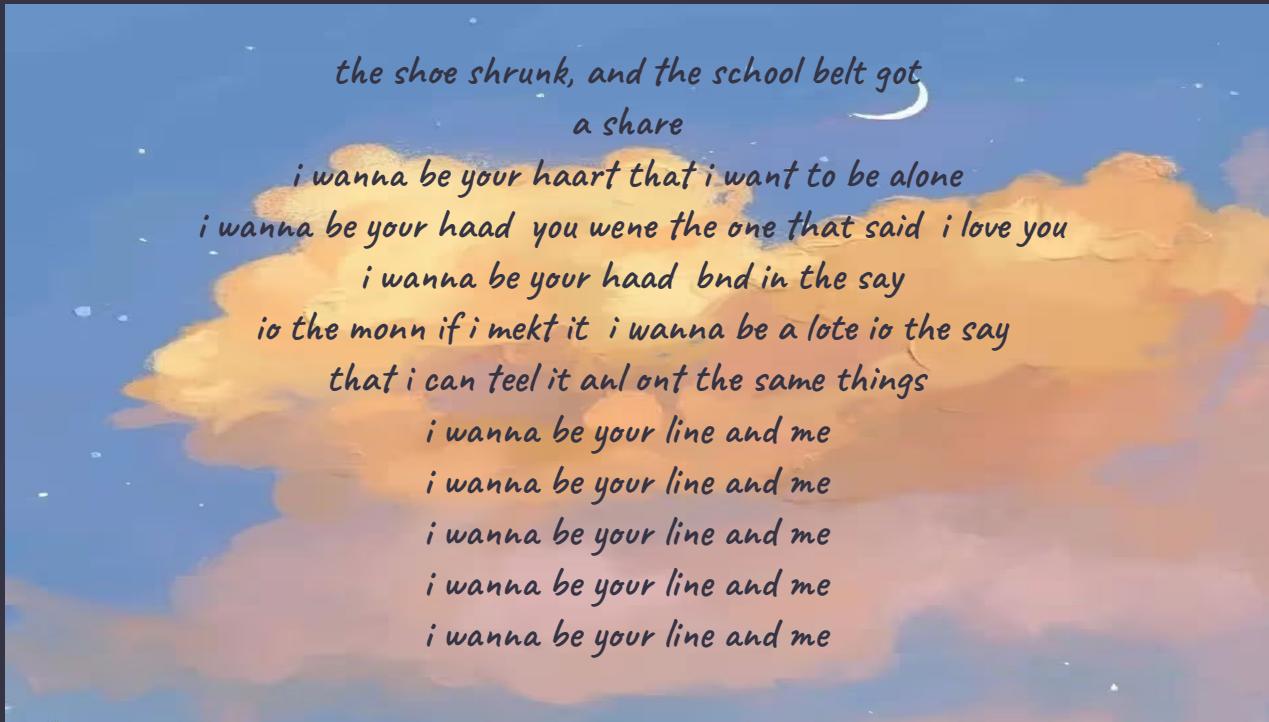




Generated Lyrics









Lyrics Generation with Deep Learning Models



Search



Home



Library

RNN-3



Two Hidden Layers + Dropout Rate 0.2



Model Description



Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 100, 256)	264192
dropout (Dropout)	(None, 100, 256)	0
lstm_1 (LSTM)	(None, 128)	197120
dropout_1 (Dropout)	(None, 128)	0
dense (Dense)	(None, 47)	6063

Total params: 467375 (1.78 MB)

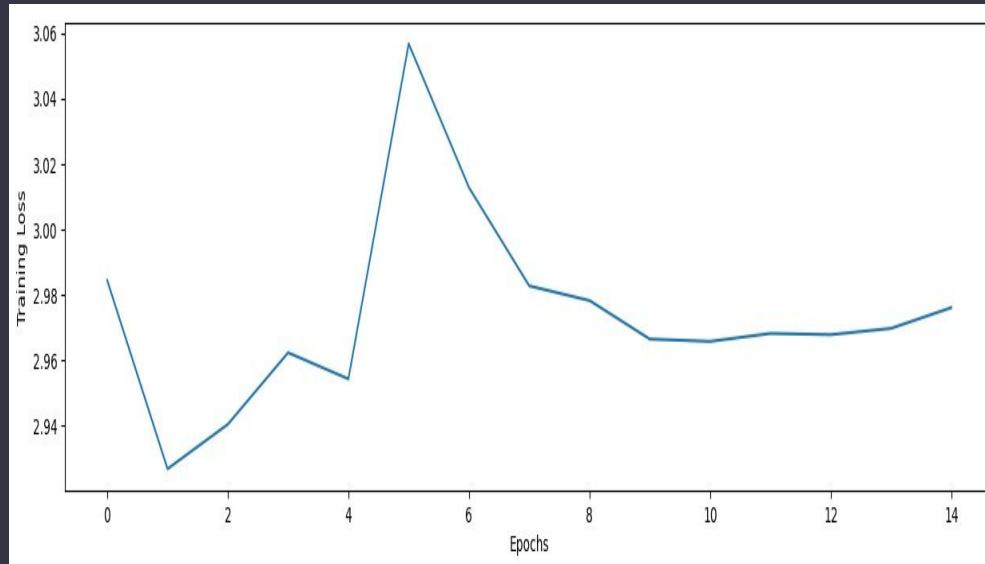
Trainable params: 467375 (1.78 MB)

Non-trainable params: 0 (0.00 Byte)



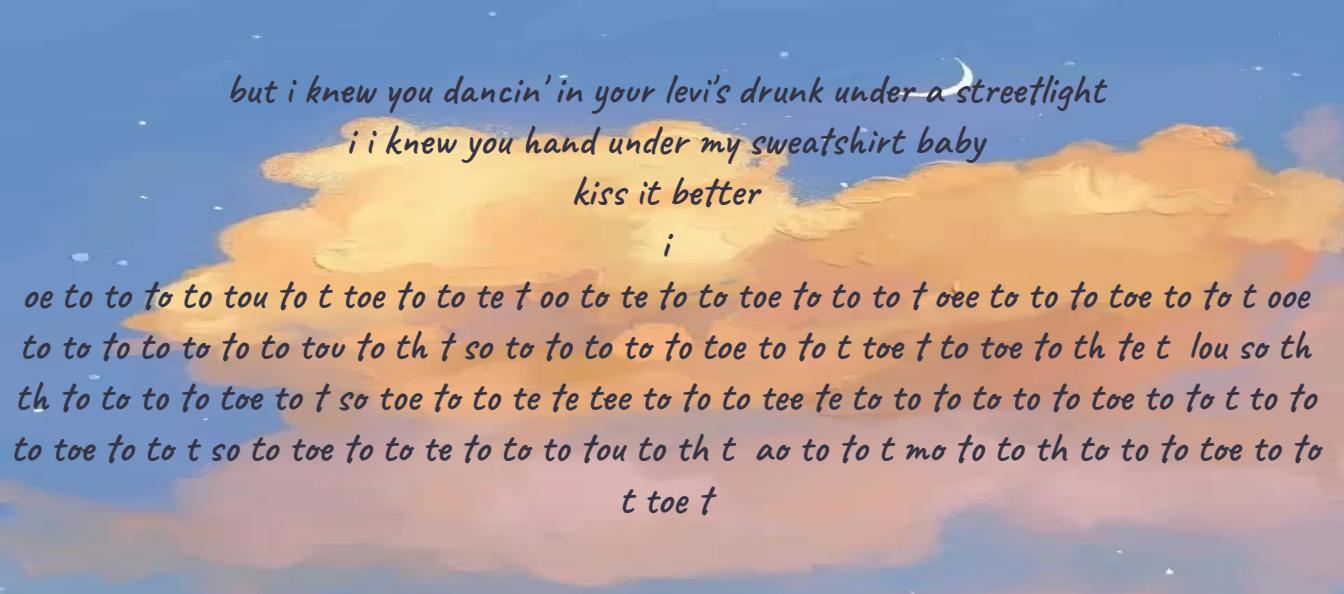


Model Performance





Generated Lyrics



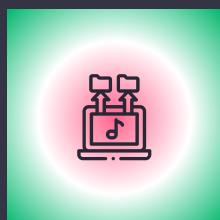
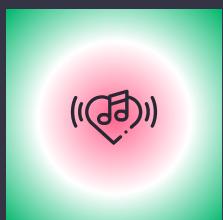
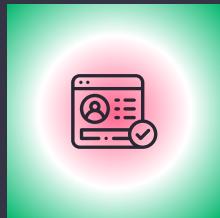
but i knew you dancin' in your levi's drunk under a streetlight
i i knew you hand under my sweatshirt baby
kiss it better
i

oe to to to to tou to t toe to to te f oo to te to to toe to to to t oee to to to toe to to t ooe
to to to to to to tou to th t so to to to to toe to to t toe t to toe to th te t lou so th
th to to to to toe to t so toe to to te te tee to to to tee te to to to to toe to to t to to
to toe to to t so to toe to to te to to tou to th t ao to to t mo to to th to to to toe to to
t toe t





Pre-trained Models



GPT-2
Google's T5





Lyrics Generation with Deep Learning Models



Search



Home



Library

GPT-2



Pretrained Model from transformers library



Introduce GPT-2 Model



```
# Install the transformers library
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load pre-trained GPT-2 model and tokenizer
model_name = "gpt2"
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
model = GPT2LMHeadModel.from_pretrained(model_name)
# Set the model to evaluation mode
model.eval()
```





GPT-2: Lyrics Generation



```
# Function to generate lyrics
def generate_lyrics(prompt, max_length=400, temperature=0.7):
    input_ids = tokenizer.encode(prompt, return_tensors="pt")

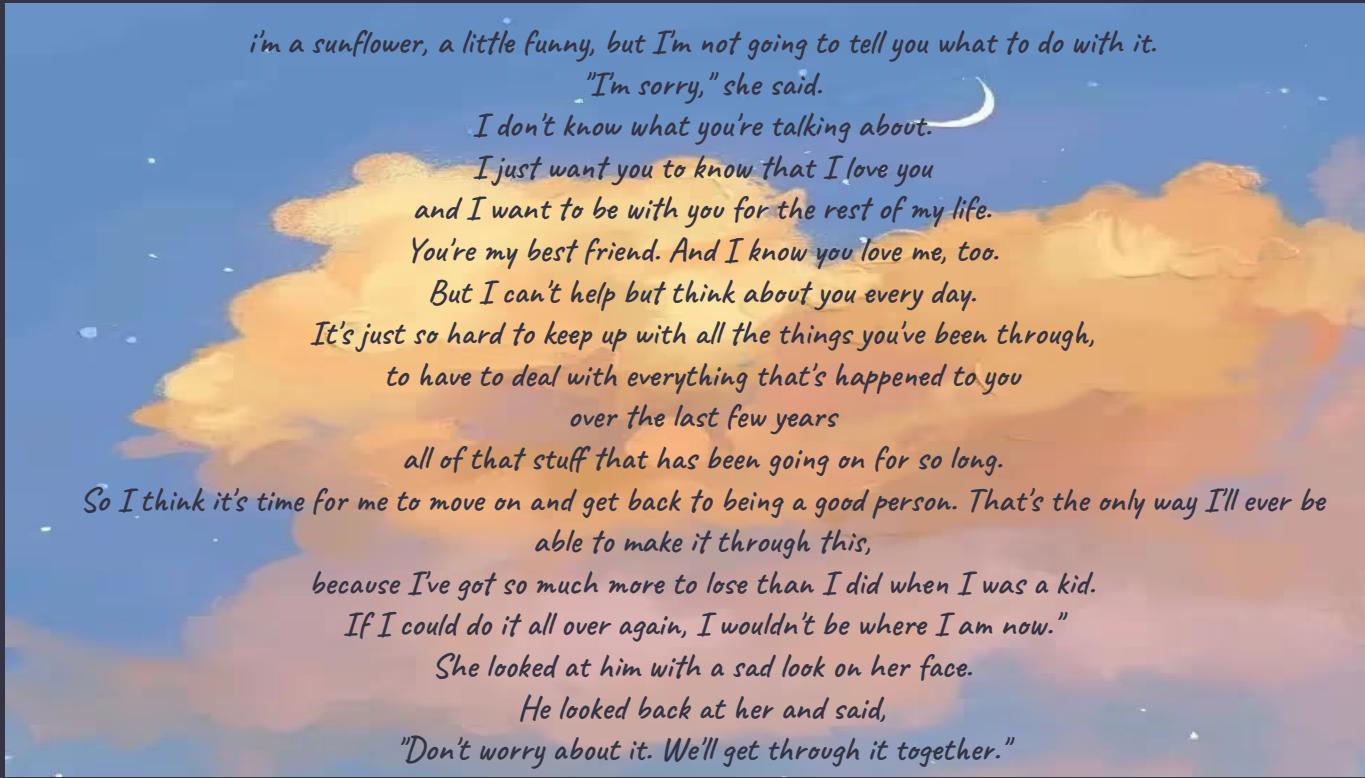
    # Generate text
    output = model.generate(
        input_ids,
        max_length=max_length,
        temperature=temperature,
        num_beams=5,
        no_repeat_ngram_size=2,
        top_k=50,
        top_p=0.95,
        pad_token_id=tokenizer.eos_token_id,
    )

    generated_lyrics = tokenizer.decode(output[0], skip_special_tokens=True)

    return generated_lyrics
```

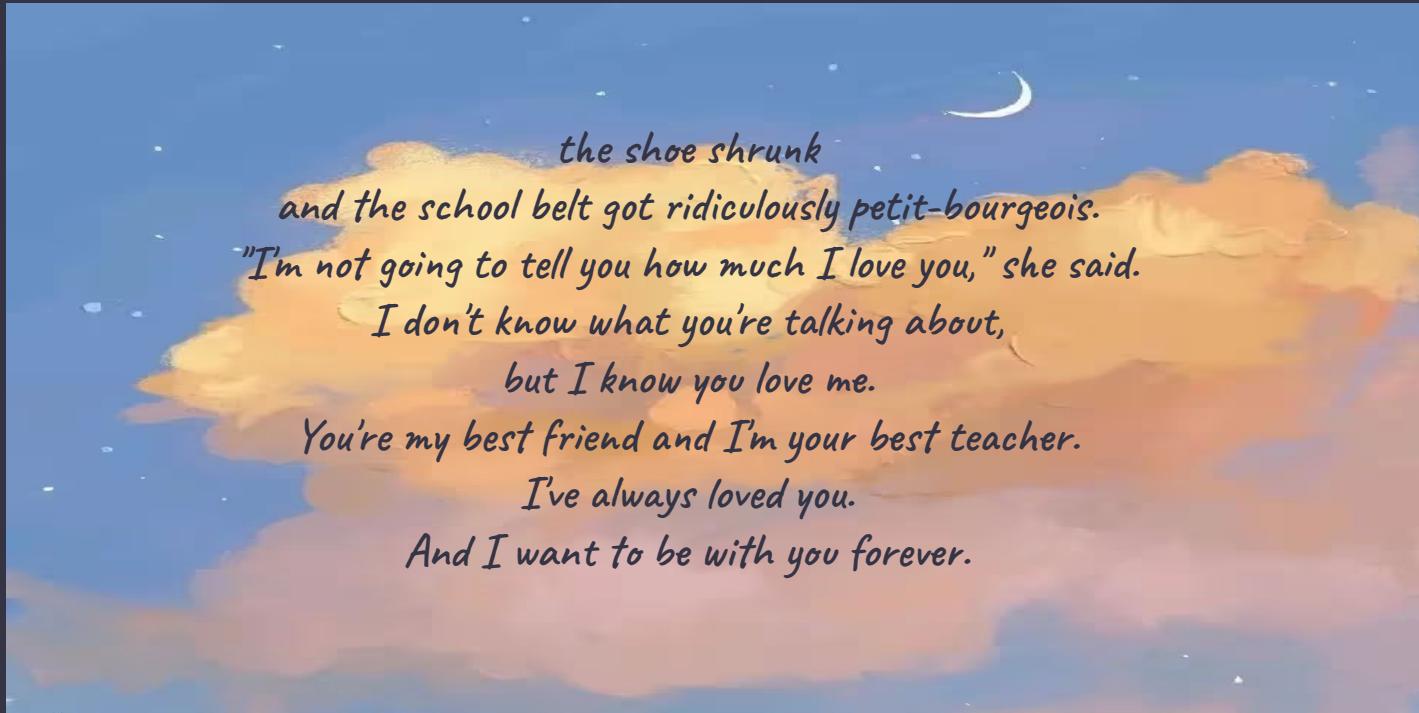


GPT-2: Model Performance





GPT-2: Model Performance





Lyrics Generation with Deep Learning Models



Search



Home



Library

Google T5



Pretrained Model from transformers library



Introduce T5



```
from transformers import T5ForConditionalGeneration, T5Tokenizer

# Load pre-trained T5 model and tokenizer
model = T5ForConditionalGeneration.from_pretrained('t5-small')
tokenizer = T5Tokenizer.from_pretrained('t5-small')
```





T5: Lyrics Generation



```
# Function to generate lyrics
def generate_lyrics(prompt, return_tensors="pt", max_length=512, truncation=True):
    input_ids = tokenizer.encode(prompt, return_tensors="pt", max_length=512, truncation=True)

    # Generate text
    output = model.generate(
        input_ids,
        max_length=400,
        num_beams=2,
        no_repeat_ngram_size=3,
        top_k=50,
        top_p=0.95,
        length_penalty=0.2)

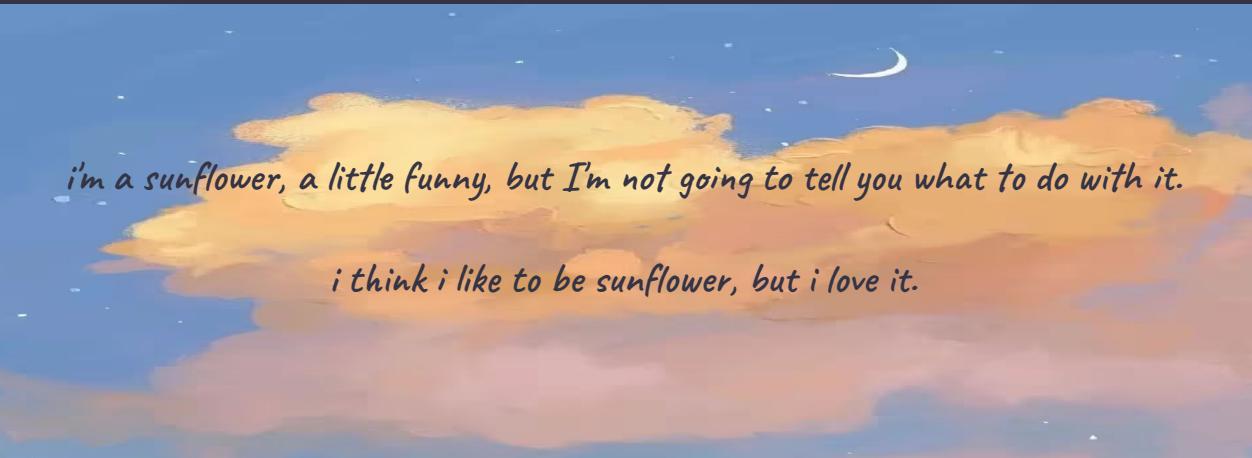
    generated_lyrics = tokenizer.decode(output[0], skip_special_tokens=True)

    return generated_lyrics
```





T5: Model Performance





T5: Model Performance





Lyrics Generation with Deep Learning Models



Search



Home



Library

05 •

Lyrics Generator with Artists



Seed Text: Artist Names



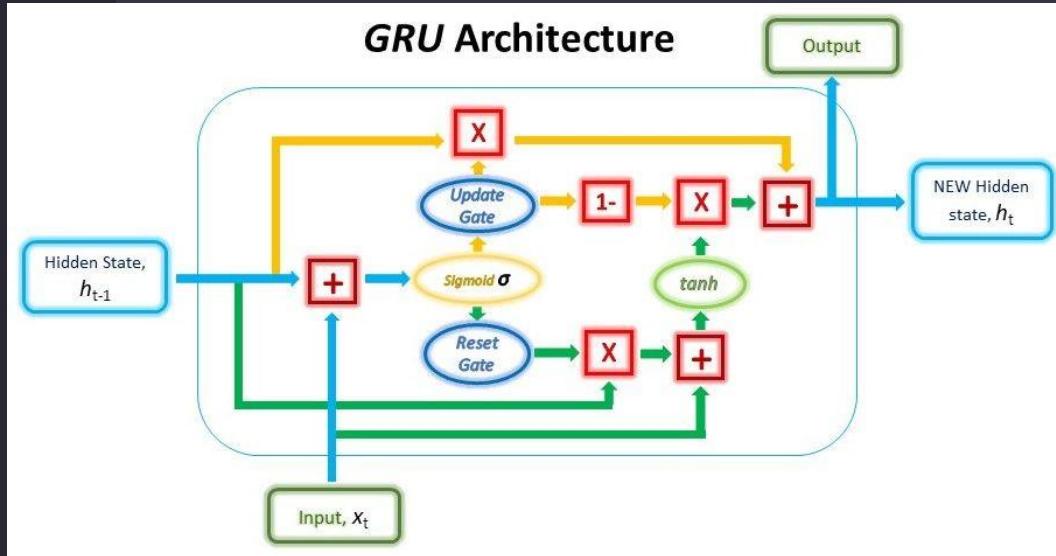
Different artists have their own styles of songs.
Therefore, we passed the artist names as seed text to
see if the model can generate text with their styles.





Gated Recurrent Unit

Gated Recurrent Unit (GRU) is a relatively new mechanism designed by Cho et al. at 2014.



GRU gets rid of the cell states in LSTM, and it has a reset gate and an update gate.

Reset gate decides how much pass information to forget. Update gate decides what information to discard and add.

Reference:

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>



Search

Home

Library

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 100, 256)	2574848
gru_2 (GRU)	(None, 100, 128)	148224
dropout_1 (Dropout)	(None, 100, 128)	0
gru_3 (GRU)	(None, 128)	99072
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10058)	1297482

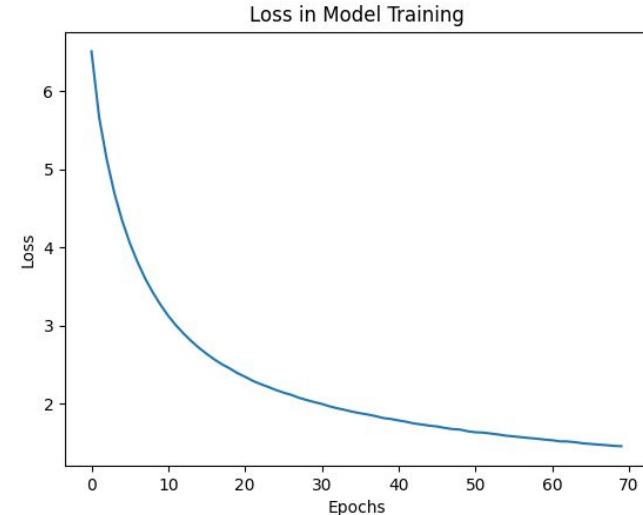
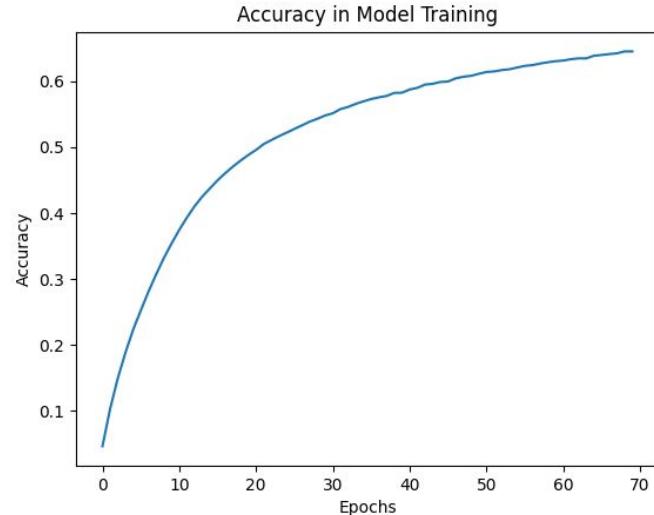
Total params: 4119626 (15.72 MB)

Trainable params: 4119626 (15.72 MB)

Non-trainable params: 0 (0.00 Byte)



Metrics





Pre-trained Model: CTRL



Conditional Transformer Language Model (CTRL) is a pre-trained model training on a very large corpus of ~140 GB of text data with the first token reserved as a control code (Wikipedia, Books, etc.)

To utilize this model, we have to add a control code token at the beginning of the seed text, which might influence the results

Reference: https://huggingface.co/docs/transformers/model_doc/ctrl#ctrl



Results: Taylor Swift (RNN)





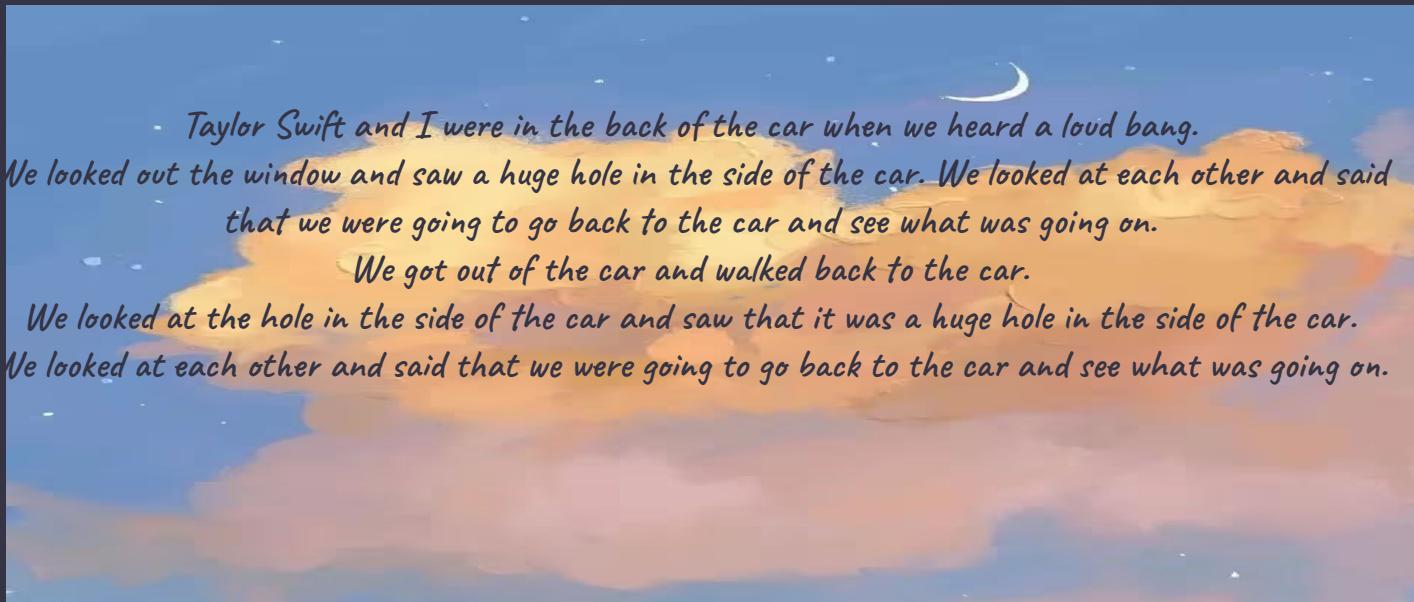
Results: Horror Taylor Swift (CTRL)



Taylor Swift and I were in the back of the car when we heard a loud bang.
We looked out the window and saw a huge hole in the side of the car. We looked at each other and said
that we were going to go back to the car and see what was going on.

We got out of the car and walked back to the car.

We looked at the hole in the side of the car and saw that it was a huge hole in the side of the car.
We looked at each other and said that we were going to go back to the car and see what was going on.





Lyrics Generation with Deep Learning Models

Search

Home

Library

06



Lyrics Generator with Titles



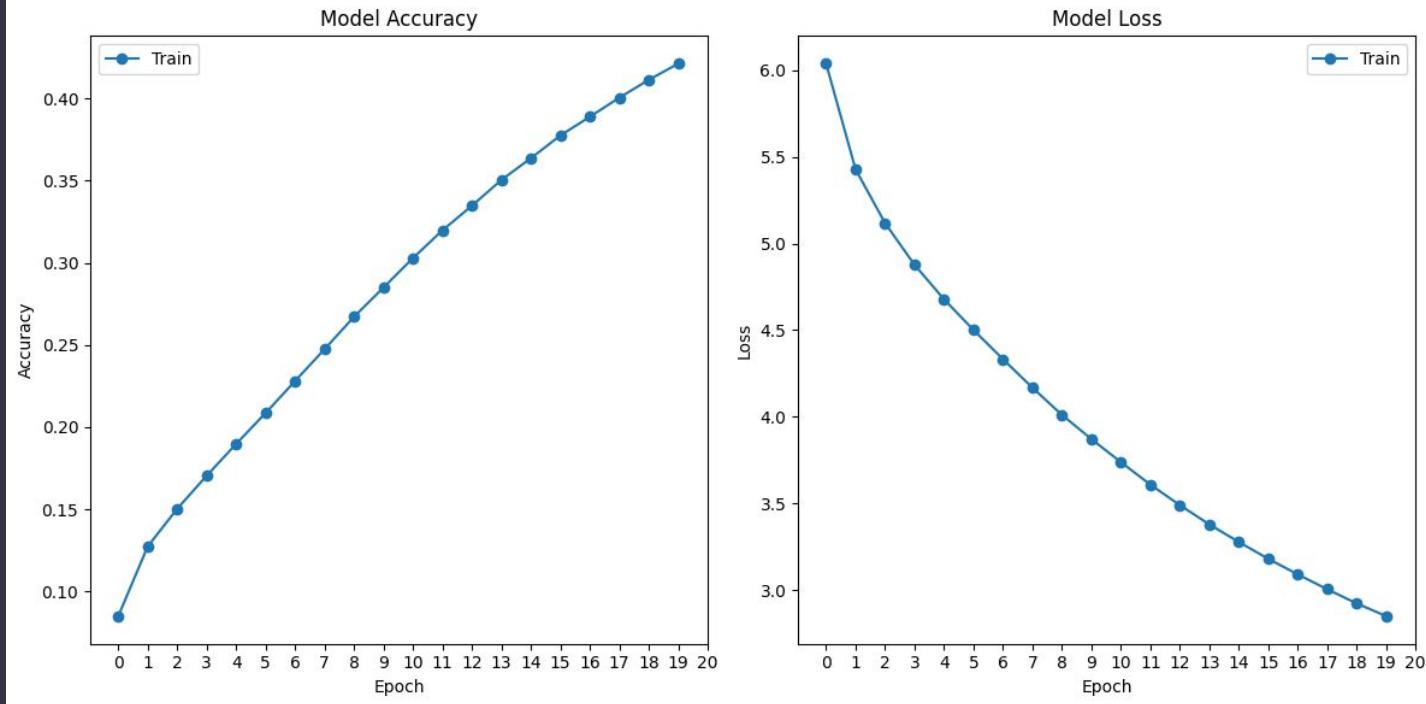
RNN Model - Summary



Layer (type)	Output Shape	Param #
=====		
embedding (Embedding)	(None, 1569, 100)	1496500
bidirectional (Bidirectional)	(None, 1569, 300)	301200
bidirectional_1 (Bidirectional)	(None, 200)	320800
dense (Dense)	(None, 14965)	3007965
=====		
Total params: 5126465 (19.56 MB)		
Trainable params: 5126465 (19.56 MB)		
Non-trainable params: 0 (0.00 Byte)		



RNN Model - Fit





RNN Model - Lyrics Generation



```
# Function to generate text

def generate_text(seed_text, next_words, model, max_sequence_len, temperature=1.0):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predictions = model.predict(token_list, verbose=0)[0]

        # Apply temperature to the predictions
        predictions = np.log(predictions + 1e-7) / temperature # Smoothing and avoid log(0)
        exp_predictions = np.exp(predictions)
        predictions = exp_predictions / np.sum(exp_predictions)

        # Choose the next word with weighted probability
        next_word_index = np.random.choice(len(predictions), p=predictions)

        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == next_word_index:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text

# Generate new lyrics
print(generate_text("Love", 200, model, max_sequence_len, temperature=1.003))
```





RNN Model - Lyrics Generation



Love

Love is been true and i'm still trying
when i feel here at home

and i know you're gone long nights but i think i have to laugh
baby i know that you all have guns
and she just wanna be alone

i'm giving you to talk what we once want to wake all that one broke oh way out
i need all my plans ooh i'd follow you in my head
the things you could know how i know

ow ow ow

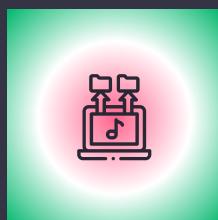
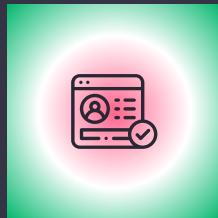
i'm on guard

when i see you again when i see you again
i'm not your rider when i'm so sad when it's so sorry well that's alright
cause you make me feel so young i look inside





Pre-trained Models



- Model Choice
- How the Model Works
- Model Performance

GPT-2





GPT-2: How the Model Works



```
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# Load pre-trained model tokenizer (vocabulary)
tokenizer = GPT2Tokenizer.from_pretrained('gpt2')

# Load pre-trained model (weights)
model = GPT2LMHeadModel.from_pretrained('gpt2')
model.eval() # Set the model to evaluation mode
```





GPT-2: How the Model Works

```
GPT2LMHeadModel(  
    (transformer): GPT2Model(  
        (wte): Embedding(50257, 768)  
        (wpe): Embedding(1024, 768)  
        (drop): Dropout(p=0.1, inplace=False)  
        (h): ModuleList(  
            (0-11): 12 x GPT2Block(  
                (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)  
                (attn): GPT2Attention(  
                    (c_attn): Conv1D()  
                    (c_proj): Conv1D()  
                    (attn_dropout): Dropout(p=0.1, inplace=False)  
                    (resid_dropout): Dropout(p=0.1, inplace=False)  
                )  
                (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True)  
                (mlp): GPT2MLP(  
                    (c_fc): Conv1D()  
                    (c_proj): Conv1D()  
                    (act): NewGELUActivation()  
                    (dropout): Dropout(p=0.1, inplace=False)  
                )  
            )  
        )  
        (ln_f): LayerNorm((768,), eps=1e-05, elementwise_affine=True)  
    )  
    (lm_head): Linear(in_features=768, out_features=50257, bias=False)  
)
```





GPT-2: Lyrics Generation



```
# Define the prompt with the song title and an indicator to start the lyrics
prompt = "Title: Long Distance Love\nLyrics:\n"
# Encode the prompt text to token ids
input_ids = tokenizer.encode(prompt, return_tensors='pt')
# Generate the lyrics with modified parameters
lyrics_output = model.generate(
    input_ids,
    max_length=300,
    temperature=1.,
    top_p=0.9,
    no_repeat_ngram_size=2,
    num_return_sequences=1,
    pad_token_id=tokenizer.eos_token_id
)
# Decode and print the generated lyrics
lyrics = tokenizer.decode(lyrics_output[0], skip_special_tokens=True)
print(lyrics)
```





GPT-2: Model Performance



Long Distance Love

I'm a girl, I'm not a boy, but I am a man.

And I love you, and I want you to love me.

I don't want to be a woman, so I'll be your wife.

And I will be the one who will love and love. So I can't be anything else.

But I know that I have to. Because I've been through so much.

It's been so hard. You know, it's hard to get through.

The hardest thing is to go through it. To go to school, to work, you know.

That's the hardest part. When you go out and you're like,

"Oh, my God, that's so fucking hard," you don. Like, what's going on?

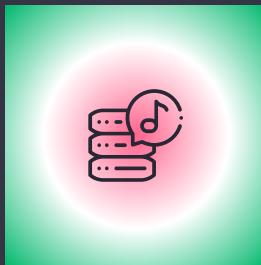
You're not going to do it, because you've got to, like you said, go.

Or you can go home and just go and do whatever you want.

Just go, just do what you like.



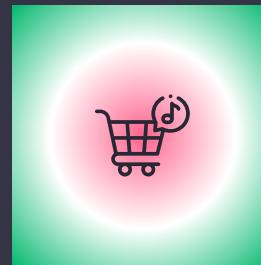
Compare Performances



RNN

More than 20 hours of training

Cost me more than \$150 on Google Colab



GPT-2

Generate in 5 seconds

Cost Nothing





Lyrics Generation with Deep Learning Models



Search



Home



Library

07



Limitations & Discussions



Limitation & Discussion



- Limited Training Epochs for RNN Models
- Hyperparameter Fine-Tuning
- Limited Diversity in Training Data
- Pre-trained Models' Limitation
- Dependency on Dataset Quality





Lyrics Generation with Deep Learning Models



Search



Home



Library

08



Conclusion



Conclusion



- RNN models demonstrated good performance, but faced challenges due to limitations in training epochs and hyperparameter tuning.
- Pre-trained models, especially GPT-2, showed better results with higher efficiency and speed compared to RNN models.
- The field of text generation involves balancing model complexity, training efficiency, and the quality of the output.
- The ongoing evolution of natural language processing is driven by efforts to find innovative ways to improve creative text generation.





Reference



- Github Repository:
https://github.com/JiaqinWu/ds6600_Final_Group14/tree/main
- Kaggle:
<https://www.kaggle.com/code/karnikakapo/or/lyrics-generator-rnn>
- GRU:
<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- CTRL:
https://huggingface.co/docs/transformers/model_doc/ctrl#ctrl
- GPT pretrained Model:
<https://openai.com/research/better-language-models>
- Google T5 Model:
https://wandb.ai/mukilan/T5_transformer/reports/Exploring-Google-s-T5-Text-To-Text-Transformer-Model--VmIldzoyNjkzOTE2



Search



Home



Library

Thank You!

Do you have any questions?

Naihuan Jing: nj417@georgetown.edu

Jiaquin Wu: jw2104@georgetown.edu

Kefan Yu: ky285@georgetown.edu

