Lyrics Generation with Deep Learning Models

Naihuan Jing, Kefan Yu, Jiaqin Wu

Abstract

This article employs a dual approach, integrating traditional Recurrent Neural Network (RNN) models and pre-trained models to delve into the realm of lyric generation. Our methodology incorporates three distinct input options, encompassing initial sentences, artist specifications, and titles. This multifaceted input strategy aims to assess the performance variations among deep learning models. Through exhaustive experimentation, we have discerned that pre-trained models, exemplified by GPT-2, outshine RNN models. Noteworthy attributes contributing to this superiority include the expeditious generation speed and enhanced performance in crafting generated lyrics. This comparative analysis underscores the efficacy of pre-trained models in the domain of lyric generation, emphasizing their distinct advantages over traditional RNN counterparts.

Keywords: Deep Learning, Lyric Generation, Pre-trained Models, RNN Model

1 Background

In the contemporary era, music serves as a ubiquitous source of relaxation and enjoyment, offering a diverse array of popular songs that resonate with audiences globally. Beyond mere auditory pleasure, the intricate interplay of lyrics, rhythms, and stylistic nuances significantly contributes to the widespread acclaim and popularity of a musical composition(Barradas & Sakka, 2022). In this context, artists wield their distinctive styles, and the very nomenclature of a song can wield considerable influence over its reception.

In the realm of lyric generation, prior research has delved into harnessing the capabilities of Recurrent Neural Network (RNN) models, exploring diverse applications and methodologies. One notable study focused on concealing secret information within Chinese pop music lyrics. The research employed the RNN Encoder-Decoder model, specifically utilizing the LSTM (Long Short-Term Memory) model to generate subsequent Chinese characters or words based on an initial lyric line. This innovative approach aimed at transforming a given seed lyric into a new song by iteratively predicting and extending the lyrical content (Tong, Liu, Wang, & Xin, 2019).

Similarly, another literature expanded on the application of LSTM networks in lyric generation, this time tailoring the output to a specific genre or artist. The study investigated the use of modern deep learning techniques to enhance the songwriting process, drawing inspiration from existing artists and their notable works. By utilizing LSTM networks, the research sought to capture the nuances of distinct genres or artists, contributing to the development of more refined and artistically resonant lyric generation models (Dhandapani, Ilakiyaselvan, Mandal, Bhadra, & Viswanathan, 2023).

In a different vein, the concept of an AI-Lyricist was introduced, a system designed to produce novel yet meaningful lyrics given a predetermined vocabulary and a MIDI file as inputs. This multifaceted task involved overcoming challenges such as automatic melody identification, syllable template extraction from multi-channel music, creative lyric generation in alignment with the music's style, and adherence to specific vocabulary constraints. The research not only showcased advancements in AI-driven lyric composition but also highlighted the complexities involved in crafting meaningful and musically coherent lyrics (Ma, Wang, Kan, & Lee, 2021).

Collectively, these studies underscore the diverse applications of RNN models in lyric generation, ranging from concealing information to tailoring output to specific genres or artists, and even extending to intricate tasks involving multi-channel music analysis. The exploration of deep learning techniques continues to shape the landscape of AI-driven lyric generation, offering insights into both creative and technical aspects of this evolving field.

2 Research Objectives

In our research, we harnessed the power of deep learning models including Recurrent Neural Network (RNN) and pre-trained models to delve into the intricate art of lyrical composition. Our exploration unfolded across three distinct dimensions. Firstly, we delved into the realm of predictive creativity by training our model on existing lyrics, empowering it to craft seamlessly connected and contextually relevant follow-up lyrics. Secondly, we ventured into the personalized domain of musical expression by inputting the name of a singer, allowing the model to generate songs tailored to the unique stylistic attributes of the specified artist. Lastly, we engaged with

the thematic essence of music by inputting song titles, enabling the model to compose songs harmoniously aligned with the overarching themes encapsulated in the titles.

This multi-faceted approach not only showcases the versatility of deep learning models in the realm of creative arts but also opens new avenues for exploring the intricate dynamics between artist identity, song titles, and the lyrical tapestry that forms the heart of musical compositions. As we navigate the intersections of technology and creativity, our study seeks to contribute valuable insights to the ever-evolving landscape of music generation and artistic expression.

3 Exploratory Data Analysis

We obtained our dataset from Kaggle (Kapoor, 2021) and augmented it by manually adding additional records to enhance its size. Initially encompassing information on artists, titles, and lyrics, we further enriched the dataset by introducing new variables for a more comprehensive understanding. These additions included: 1) the character count in the lyrics; 2) the word count in the lyrics; and 3) the line count in the lyrics.

Our exploratory data analysis (EDA) commenced with a focus on fundamental dataset characteristics. From a singer-centric perspective, Figure 1 illustrates the distribution of songs across different artists. Notably, the majority of artists boast song numbers exceeding 20, with the dataset's highest count reaching 50—a testament to the robustness of our dataset.

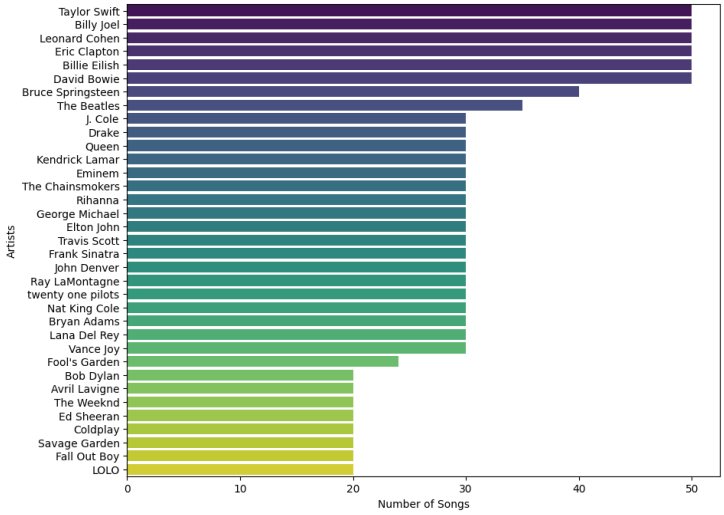

Figure 1: Number of Songs per Artist

Further delving into the lyrical landscape, Figure 2 provides insights into the distribution of word counts across songs. The majority of songs exhibit word counts ranging between 200 and 400, reflecting a well-balanced distribution of word lengths.

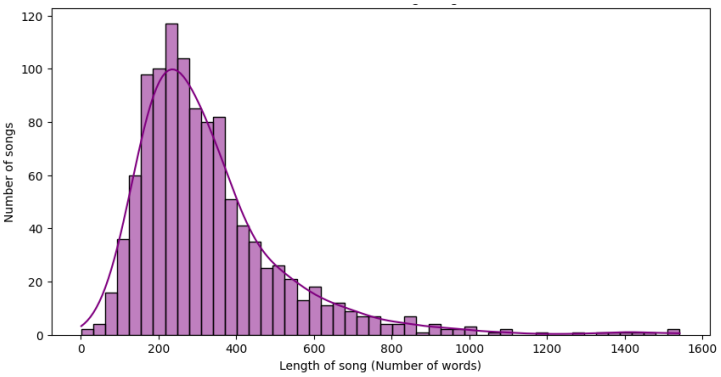

Figure 2: Distribution of Song Lengths

In a bid to unravel the linguistic tapestry within the lyrics, we extracted the most frequently occurring words and visually presented them in a word cloud (Figure 3). Prominent words such as "know," "love," "now," and

"time" emerged as recurrent themes in the lyrical corpus, offering valuable insights into the thematic essence of the dataset.

Figure 3: Common Words in Songs

4 Lyrics Generator from Starting Sentences

4.1 RNN Models

In the initial phases of our lyric generation process, we conducted meticulous preprocessing on the existing lyric records to ensure a refined and standardized dataset. Recognizing the presence of irrelevant characters and symbols, we curated a comprehensive lyric corpus by eliminating extraneous elements such as foreign language text, symbols, and numbers. Additionally, we converted all English words to lowercase to maintain uniformity and facilitate subsequent analysis.

To facilitate the encoding and decoding processes within the RNN models, we created a dictionary for character mapping. The corpus was then segmented into smaller, equal-length sentences. Encoding involves generating sequences of features and their corresponding targets. Subsequently, we resized and normalized the labels, preparing them for effective RNN training.

Within our model, we implemented a specific approach for starting sentences, setting them to a standardized length of 40 characters. A function was defined to enable the model to generate the subsequent n characters after inputting sentences. In experimenting with the RNN model, we explored LSTM models with various hyperparameters to identify the configuration that yielded optimal performance. This involved evaluating models with one hidden layer and a dropout rate of 0.2, one hidden layer with a dropout rate of 0.3, and two hidden layers with a dropout rate of 0.2. In order to address the risk of overfitting, we implemented early stopping mechanisms throughout the training phase. Additionally, we employed Adammax optimization during the training process to fine-tune the parameters in the RNN models.

4.1.1 RNN Model with One Hidden Layer and a Dropout Rate of 0.2

In our initial exploration, we implemented the RNN model with a single hidden layer and a dropout rate of 0.2, as depicted in Figure 4. To assess the model's performance, we employed the loss function as the evaluation metric, meticulously tracking its behavior throughout the training process. With efficiency in mind, we set the epoch number to 40 and the batch size to 256, significantly reducing the training time to approximately 80 seconds per epoch. The model demonstrated favorable performance, as illustrated in Figure 5, showcasing a consistent decrease in the loss function during most epochs. By the end of the training, the model achieved a commendable training loss of around 1.8.

To further scrutinize the generation capabilities of this RNN model, we subjected it to two examples, as displayed in Figures 6 and 7. However, the outcomes revealed some limitations, as the generated lyrics often contained nonsensical English words such as "farher" instead of "farther" and exhibited repetitions. These shortcomings prompted us to consider fine-tuning certain hyperparameters of the RNN model to enhance its overall performance.

4.1.2 RNN Model with One Hidden Layer and a Dropout Rate of 0.3

We adjusted the dropout rate of our preceding model to 0.3, aiming to increase the regularization effect by dropping a larger proportion of connections. The RNN model with a dropout rate of 0.3 introduces greater randomness, compelling it to rely on a broader array of features, potentially enhancing its resilience to overfitting compared to the model with a dropout rate of 0.2. The model specifications remain consistent with the previous

```
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)
_________________________________________________________________
```

Figure 4: Lyrics Generator from Starting Sentences - RNN Model Architecture

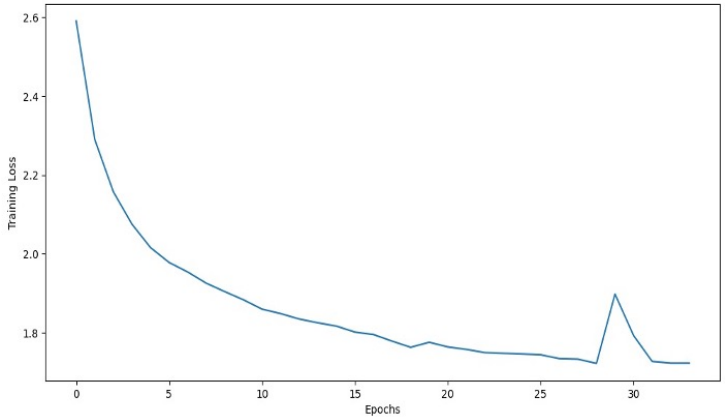

Figure 5: Lyrics Generator from Starting Sentences - Model Loss

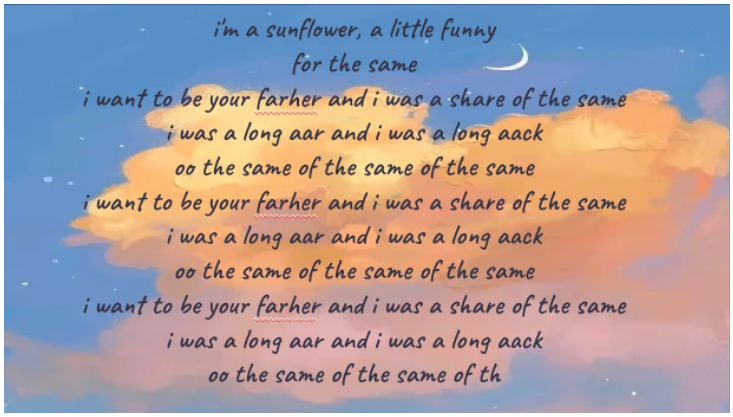

Figure 6: Lyrics Generator from Starting Sentences - Output

iteration. We set the batch size to 256 and the epoch to 40, with each epoch requiring approximately 80 seconds during training.

In terms of performance, the RNN model with a dropout rate of 0.3 exhibits increased instability, as depicted in Figure 8. The training loss experiences an initial decline, but after 15 epochs, it becomes erratic, culminating in the cessation of backpropagation after 35 epochs. However, the lowest training loss is comparable to the previous model, hovering around 1.8. Examining the generated lyrics in Figures 9 and 10 reveals similarities to the previous model, featuring nonsensical English words and repetitive patterns.

Considering these observations, we contemplate adopting a more intricate model, such as incorporating two hidden layers in the LSTM architecture, to assess whether it yields improved outcomes.

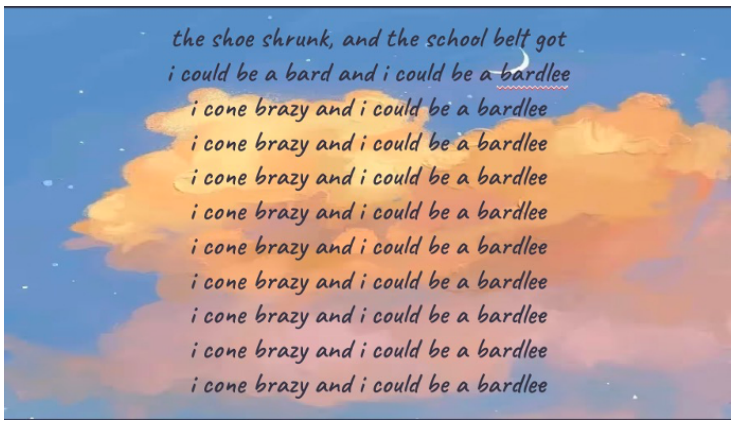

Figure 7: Lyrics Generator from Starting Sentences - Output

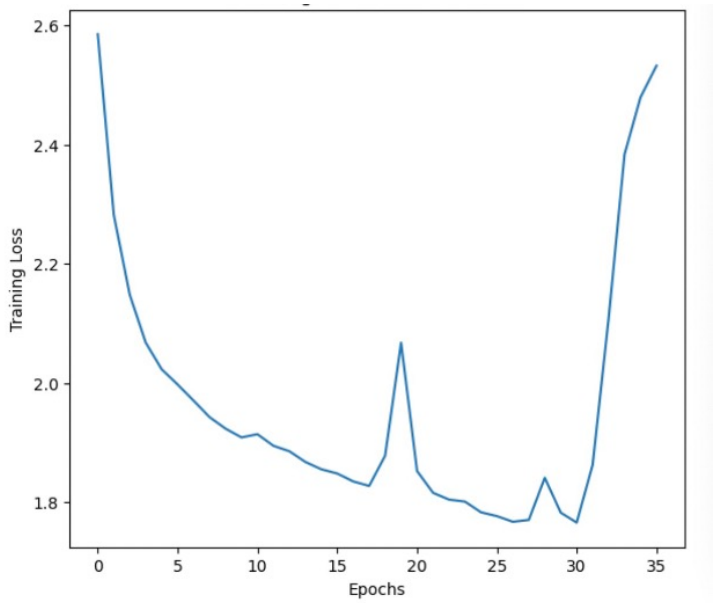

Figure 8: Lyrics Generator from Starting Sentences - Model Loss

Figure 9: Lyrics Generator from Starting Sentences - Output

4.1.3 RNN Model with Two Hidden Layers and a Dropout Rate of 0.2

In our latest experimentation, we introduced two hidden layers into the RNN model, incorporating a dropout rate of 0.2. The detailed model description is illustrated in Figure 11. Notably, the model with dual hidden layers

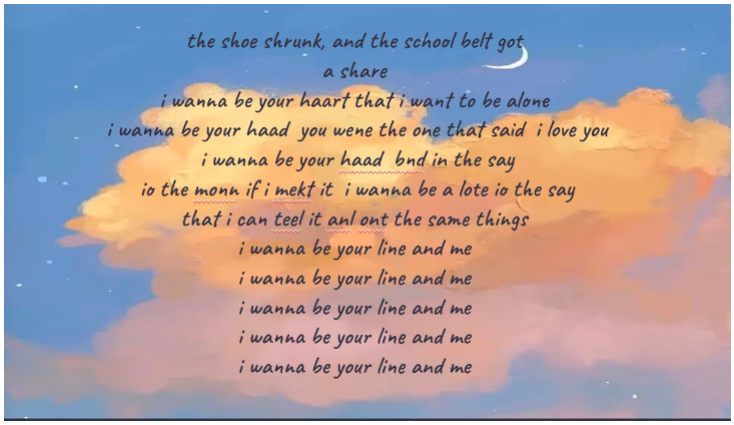

Figure 10: Lyrics Generator from Starting Sentences - Output

possesses a greater number of parameters compared to its single-layer counterpart, resulting in increased memory consumption. While maintaining a batch size of 256, we adjusted the epoch to 15 due to the computationally intensive nature of the process. Each epoch in this training regimen takes approximately 30 minutes to complete.

```
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)
_________________________________________________________________
```

Figure 11: Lyrics Generator from Starting Sentences - RNN Model Architecture

The training loss plot, presented in Figure 12, reveals that the performance of the two hidden layers model is suboptimal when contrasted with the single hidden layer model. The training loss exhibits significant fluctuations, hovering around 2.98, and fails to diminish rapidly in the initial stages of training. These observations raise concerns about the actual efficacy of the two hidden layer RNN model. To further evaluate its performance, we inspected the generated lyrics, depicted in Figure 13, and observed a notable deterioration. The generated texts consist predominantly of nonsensical English words and repetitive patterns, lacking the coherent elements essential for crafting meaningful lyrics. This outcome underscores the challenges associated with the complexity introduced by the additional hidden layer.

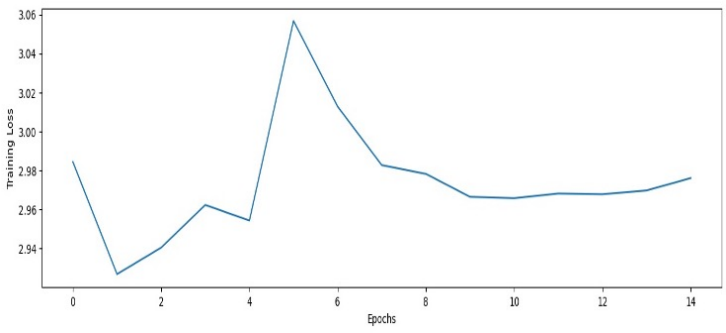

Figure 12: Lyrics Generator from Starting Sentences - Model Losss

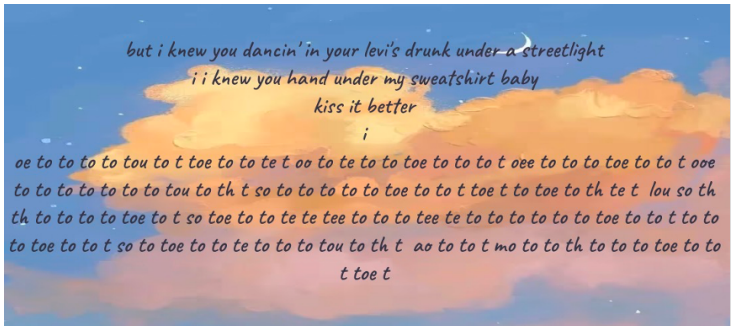

Figure 13: Lyrics Generator from Starting Sentences - Output

4.2 Pre-trained Models

Given the less-than-ideal outcomes obtained from our various RNN models in generating lyrics based on input sentences, we are exploring the use of pre-trained models to gauge their performance. Specifically, we have chosen to assess the capabilities of GPT-2 and Google's T5 models through a comparative analysis. While acknowledging the effectiveness of GPT-3, its non-free nature led us to exclude it from our current investigation. Despite its impressive capabilities, the availability of free alternatives such as GPT-2 and T5 aligns more closely with our current project constraints. By leveraging these pre-trained models, we aim to ascertain their suitability for the lyrical generation task and identify potential advantages they may offer over our custom RNN implementations.

4.2.1 GPT-2

GPT-2, an innovation from OpenAI, emerged as a pivotal component in our exploration (*Better language models and their implications*, n.d.). In Figure 14, we introduced a dedicated function for lyric generation, configuring key parameters to optimize the output. Specifically, we set the maximum length to 400, temperature to 0.7, the number of beams to 5, non-repetitive n-gram size to 2, top k value to 50, and top p value to 0.95. The efficiency of the GPT-2 model is striking, as the generation process completes in a mere 10 seconds, in stark contrast to the more time-consuming traditional RNN models.

```python
# 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
```

Figure 14: Lyrics Generator from Starting Sentences - Lyrics Generation Function

The results, showcased in Figures 15 and 16, underscore the superior performance of the GPT-2 model. Notably absent are nonsensical English words, and the prevalence of repetitive patterns has markedly diminished. The generated lyrics exhibit a coherent narrative quality, aligning more closely with the thematic essence of our initial input. This compelling evidence leads us to a resounding conclusion: GPT-2 outperforms traditional RNN models in the realm of lyric generation, offering both speed and enhanced output quality.

4.2.2 Google's T5

In addition to the GPT-2 model, we explored the text generation capabilities of Google's T5, a prominent pre-trained model. Google's T5 (Text-To-Text Transfer Transformer) is a state-of-the-art natural language processing model developed by Google Research. Introduced in 2019, T5 is based on the Transformer architecture and is designed for a versatile range of natural language understanding and generation tasks (*Weights & Biases*, 2023). Figure 17 details the function employed for lyric generation based on our input sentences. Aligning with our

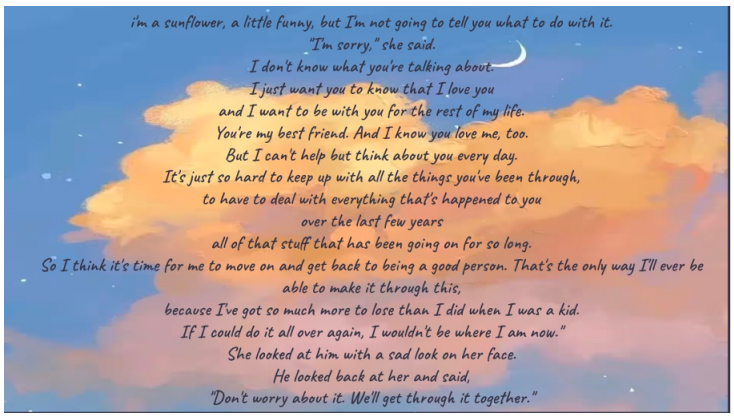

Figure 15: Lyrics Generator from Starting Sentences - Output

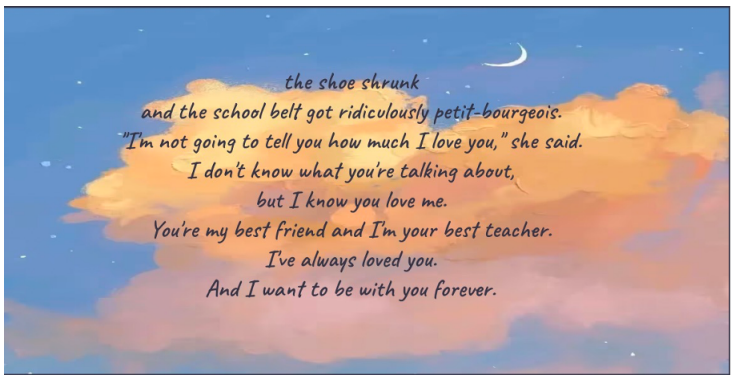

Figure 16: Lyrics Generator from Starting Sentences - Output

approach for GPT-2, we fine-tuned the hyperparameters of Google's T5 model, setting the maximum length to 400, the number of beams to 2, the non-repeated n-gram size to 3, top k value to 50, top p value to 0.95, and length penalty value to 0.2.

```python
# 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
```

Figure 17: Lyrics Generator from Starting Sentences - Lyrics Generation Function

While the T5 model exhibits a performance superior to that of traditional RNN models, it falls slightly short when compared to the impressive results achieved by GPT-2, as illustrated in Figures 18 and 19. Notably, the T5 model encounters challenges in generating sufficiently lengthy lyrics based on our input. In some instances, it struggles to produce any output. However, in the first example, T5 demonstrates an ability to generate coherent and meaningful lyrics, albeit with some limitations. These findings underscore the nuanced strengths and weaknesses of different pre-trained models in the context of our lyric generation task. In general, pre-trained models outperform traditional RNN models in this context.

4.3 Comparison

In summary, our empirical findings consistently affirm the superiority of pre-trained models when juxtaposed with their traditional RNN counterparts in the realm of our study. This distinction is evident not only in their

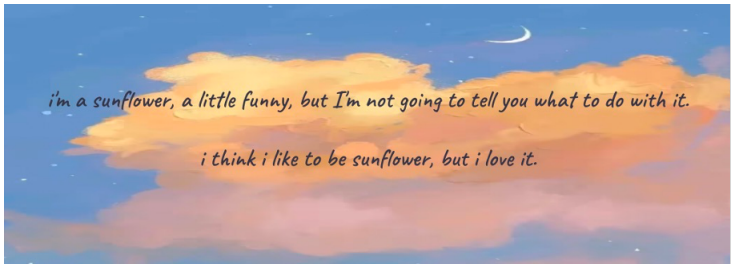

Figure 18: Lyrics Generator from Starting Sentences - Output

Figure 19: Lyrics Generator from Starting Sentences - Output

effectiveness but also in their significantly reduced execution time compared to RNN models. The pre-trained models showcase an elevated proficiency in nuanced pattern recognition, resulting in superior performance across diverse facets of text generation. This marked advancement underscores the efficacy of harnessing pre-existing linguistic knowledge inherent in these models, affording a substantial advantage over conventional RNN models within the specific context of our research.

5 Lyrics Generator from Artists

5.1 RNN Models

We utilized Gated Recurrent Unit (GRU) in our RNN model. GRU is a relatively new mechanism designed by Cho et al in 2014. It is similar to LSTM in the aspect of fixing short term memory issues, but differs in some details, as shown in Figure 20. Specifically, instead of having cell states to discard and add information, it has update gate and reset gate to help to transfer information. Update gate is similar to the input and forget gate in LSTM, which decides which information to discard and what new information to add. The reset gate decides what past information to forget (Phi, 2018).

In the model structure, we employed two GRU layers with following drop-out layers to prevent over-fitting. Both GRU layers have 128 units. Large number of units can better learn more complex patterns, but also has issues of computational cost.

5.1.1 Model Performance

With the number of epochs set to 70, the accuracy reaches to around 0.65 and loss drops below 2, as shown in Figure 21. We can see that the performance is decent but definitely can be improved. With a larger number of epochs, the accuracy can definitely be higher and loss can drop even more. The first 20 epochs have a larger increase in accuracy and larger decrease in loss. Specifically, we can see that the accuracy starts below 0.1 and increases to approximately 0.5 after 20 epochs. Similarly for loss, it starts close to 7, and drops to around 3 after 20 epochs. After 20 epochs, the increase and decrease rate become more stable.

5.1.2 Output

We chose "Taylor Swift" as the seed text, and from the generated lyric, there are some phrases like "I Smile" (pink) and "A Call" (green) occur several times, which is like Taylor Swift's style, as shown in Figure 22. However, there are also some repetitions that do not make sense such as "I Said On On On On..." and "We Can't Can't

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)

Figure 20: Lyrics Generator from Artists - RNN Model Architecture

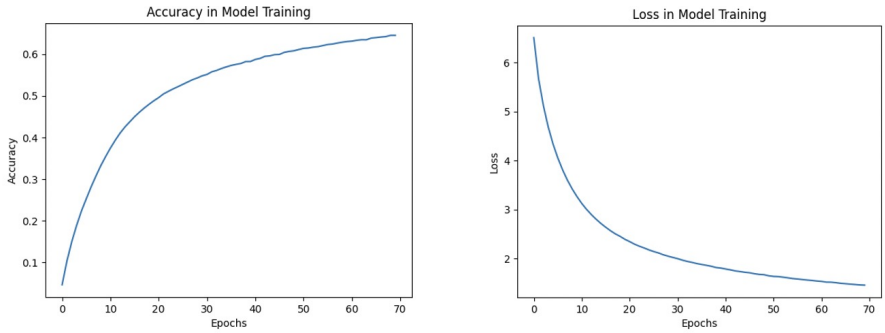

Figure 21: Lyrics Generator from Artists - Model Accuracy and Loss

Can't. . . ". Thus, we utilized another pre-trained model, expecting to generate lyrics that are more Talor Swift style.

Figure 22: Lyrics Generator from Artists - Output

5.2 Pre-trained Model

Conditional Transformer Language Model (CTRL) is a pre-trained model training on a very large corpus of around 140 GB of text data with the first token reserved as a control code (*CTRL*, n.d.). This model has a built-in dictionary in which control codes are the keys in that dictionary. If the control code is not detected in the input string, it will return errors. For example, if we want to generate text for "love" in CTRL model, we can input "Wikipedia love" or "Books love" in which "Wikipedia" and "Books" are control codes.

From the code snippet shown in Figure 23, we can see that instead of passing "Taylor Swift" into the CTRL model, we added "Horror" at the beginning of the text, which is the source code.

```
tokenizer = AutoTokenizer.from_pretrained("Salesforce/ctrl")
model = CTRLLMHeadModel.from_pretrained("Salesforce/ctrl")

# CTRL was trained with control codes as the first token
inputs = tokenizer("Horror Tylor Swift", return_tensors="pt")
```

Figure 23: Lyrics Generator from Artists - CTRL Model Code

5.2.1 Output

From the result as shown in Figure 24, we can see that instead of generating Taylor Swift's style of lyrics, it outputs a horror story in which Taylor Swift is the protagonist. Compared to lyrics generated from the RNN model, this piece of text is well-written without any repetition, and it is perfect for literature like horror novels and movie scripts. However, CTRL does not follow the research objectives of writing lyrics.

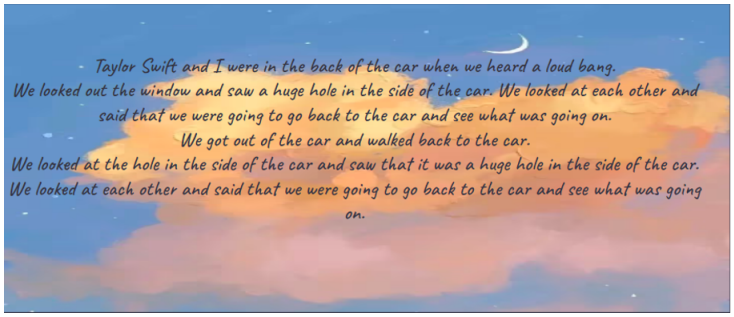

Figure 24: Lyrics Generator from Artists - Output

6 Lyrics Generator from Titles

6.1 RNN Models

The architecture of this RNN model is designed with the consideration of both performance and system limitations. It comprises several layers, each chosen for their role in the process of sequence learning, shown in Figure 25. An Embedding layer, which translates our tokenized indices into dense vectors of fixed size. Here, we chose a dimensionality of 100 to capture sufficient semantic information while balancing memory constraints. Two Bidirectional LSTM layers. They are chosen for their ability to remember information over long sequences, which is crucial for generating coherent lyrics. The bidirectionality allows the network to access both past (backward) and future (forward) contexts at every point in the sequence, providing a richer understanding of the text. The first LSTM layer has 150 units, and the second one has 100 units. This decreasing unit strategy is a design choice to condense the information flow for the final prediction. A Dense layer, with a number of neurons equal to the size of the vocabulary, which serves as the output layer. It uses a softmax activation function to generate a probability distribution over all possible next words in the sequence. We have carefully chosen the size of the network to stay within the boundaries of 83.5GB of system RAM and 40GB of GPU RAM on Google Colab, ensuring that we maximize learning capability without exceeding our computational resources.

6.1.1 Model Performance

The training performance of our RNN model over 20 epochs demonstrates a consistent improvement in both loss and accuracy metrics, shown in Figure 26. Initially, the model started with a loss of 6.0379 and an accuracy of 8.50% in the first epoch. As training progressed, both the loss steadily decreased and the accuracy increased, which is indicative of the model learning from the training data. By the 20th epoch, the model achieved a loss of 2.8500 and an accuracy of 42.12%.

This consistent decrease in loss from over 6.03 to below 2.85 suggests that the model is effectively optimizing its internal parameters to better predict the next word in the sequence of song lyrics. Similarly, the improvement in accuracy from approximately 8.5% to over 42% indicates that the percentage of the model's predictions that are correct has significantly increased, which shows the model's growing proficiency in lyric generation. Moreover, the training times remained relatively stable, averaging around 205 milliseconds per step, which reflects efficient utilization of computational resources throughout the process. The overall trend is positive and suggests that

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)

Figure 25: Lyrics Generator from Title - RNN Model Architecture

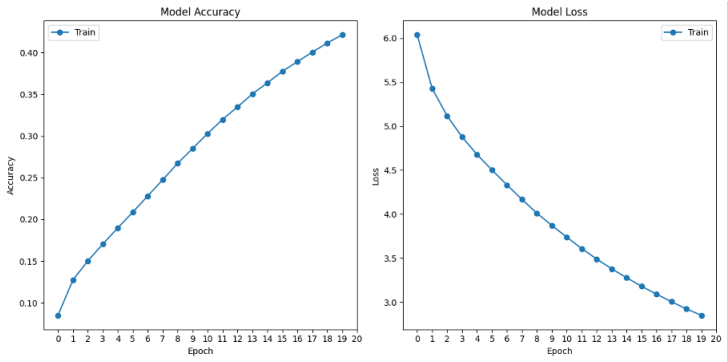

Figure 26: Lyrics Generator from Title - Model Accuracy and Loss

with further training, regularization, or architectural adjustments, the model could continue to improve its ability to generate coherent and contextually relevant lyrics.

6.1.2 Lyrics Generation

The generate_text function is the culmination of our efforts, where we harness the trained model to produce novel lyrics. This function operates by taking a seed text as input, which is the title of a song that prompts the model to begin the generation process, as shown in Figure 27. It then predicts the next word in the sequence, appends it to the seed text, and repeats this process for a specified number of iterations, which is defined by the next_words parameter. The core functionality of the function is predicated upon the model's ability to predict the probability distribution of the next word given a sequence of words. To convert this distribution into a specific word choice, we employ a technique known as temperature sampling, a method that introduces an element of randomness or diversity into the selection process. The temperature parameter plays a pivotal role in this function. A temperature of 1 maintains the model's original distribution, whereas a temperature below 1 makes the model more conservative, making it more likely to choose the words with the highest probabilities, thus reducing diversity. Conversely, a temperature greater than 1 increases diversity and randomness in the word selection process.

6.1.3 Output

The lyrics generated by the RNN model exhibit a certain degree of coherence and a flow that resembles song lyrics, with emotional expressions and a semblance of narrative, as shown in Figure 28. Like "i need all my plans ooh i'd follow you in my head "; "i'm not your rider when i'm so sad, when it's so sorry well that's alright". However, there is a noticeable repetition of phrases and a somewhat abrupt transition of ideas, which is common in simpler models like RNNs that have limited context awareness.

6.2 Pre-trained Model

GPT-2 is chosen because it's one of the most advanced language models available that can be run locally without the need for an API (*Better language models and their implications*, n.d.). It understands context and

```python
# 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))
```

Figure 27: Lyrics Generator from Title - Lyrics Generation Function

Figure 28: Lyrics Generator from Title - Output

can generate coherent and contextually relevant text based on prompts. Figure 29 shows how we import GPT-2 model to generate lyrics by inputting the song's title.

```python
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
```

Figure 29: Lyrics Generator from Title - GPT-2

6.2.1 Architecture

GPT-2 uses a transformer architecture, which allows it to consider the entire context of the input text it generates from. As shown in Figure 30, it has multiple layers of attention mechanisms that let it focus on different parts of the input text when predicting the next word. The model outputs probabilities over the vocabulary for the next word, which is then sampled to produce the output text.

6.2.2 Output

For a pre-trained model in inference mode, performance visualization like loss graphs is not applicable because there's no training process involved. Instead, the quality of the generated text is the primary indicator of perfor-

```
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)
)
```

Figure 30: Lyrics Generator from Title - GPT-2 Pre-trained Architecture

mance. We would review the generated lyrics shown in Figure 31 and assess how well they align with the style, structure, and content expected from a song given its title.

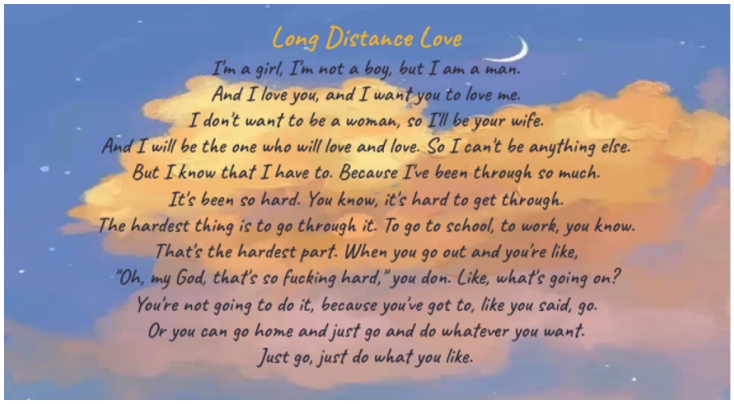

Figure 31: Lyrics Generator from Title - Output

The GPT-2 output is more structurally diverse, reflecting its advanced understanding of language and context due to its transformer architecture. It does not show repetitive phrases and demonstrates a wider variety of sentence structures. Nonetheless, the content seems to diverge from the intended theme of "Long Distance Love," and some passages may lack coherence or a clear relation to each other, which can be a byproduct of generating text without fine-tuning on a domain-specific dataset.

6.3 Comparison

The RNN model is able to generate lyrics that have a repetitive but clear theme, which can be beneficial for certain types of songs where repetition is a stylistic choice. The GPT-2 model, on the other hand, generates more complex text with less repetition, but may require more guidance or fine-tuning to produce lyrics that are thematically consistent and coherent throughout. To improve the lyric generation further, one might consider fine-tuning the GPT-2 model on a dataset of lyrics to better capture the style and thematic consistency expected in song lyrics. For the RNN model, more sophisticated architectures such as LSTM or GRU could be explored, along with techniques to prevent repetition and improve the diversity of the text. Additionally, post-processing steps such as manually editing or employing rule-based methods to ensure thematic consistency and lyrical quality could enhance the final output. As AI language models continue to improve, the integration of more advanced models and fine-tuning techniques will likely result in even more convincing and artistically valuable generated content.

7 Limitations

Despite the contributions of our research, it is essential to acknowledge its inherent limitations. These constraints include:

- **Limited Training Epochs for RNN Models:** The number of training epochs for RNN models was constrained due to the time-intensive nature of the training process. This limitation hindered the exploration of extended training periods, potentially impacting the models' convergence and the overall optimization of their parameters.
- **Hyperparameter Fine-Tuning:** The crucial process of fine-tuning hyperparameters faced constraints in our study due to the considerable time required for the execution of each RNN model. Optimal parameter adjustments play a pivotal role in model performance, and our ability to thoroughly explore this space was restricted, potentially impacting the models' effectiveness.
- **Limited Diversity in Training Data:** The scope and diversity of our training dataset were restricted, which could influence the models' ability to generalize across a broader range of input sentences. A more extensive and diverse dataset would contribute to a more robust and adaptable model.
- **Pre-trained Models' Limitation:** While pre-trained models such as GPT-2 exhibited impressive performance, our research acknowledges their inherent limitations. The adaptability of these models to specific tasks and the potential biases embedded in their pre-training data pose challenges that should be considered in the interpretation of results.
- **Dependency on Dataset Quality:** The quality and representativeness of the dataset used for training and evaluation significantly impact the robustness of our models. Limitations in the dataset quality may affect the models' ability to handle diverse input sentences, potentially leading to biased or incomplete outcomes. A more comprehensive and high-quality dataset could mitigate these concerns and enhance the reliability of our research findings.

8 Conclusion

In conclusion, our exploration into text generation using RNN models and pre-trained models has provided valuable insights into the dynamics of creative text generation. RNN models exhibited commendable performance, but their efficacy was challenged by constraints in training epochs and hyperparameter tuning. The resource-intensive nature of these models underscored the delicate balance required between achieving optimal performance and the associated computational costs.

In contrast, pre-trained models, particularly GPT-2, emerged as formidable contenders, showcasing superior results with enhanced efficiency and speed compared to their RNN counterparts. The success of GPT-2 in generating coherent and meaningful lyrics highlighted the potential advantages of leveraging pre-trained models for text generation tasks.

Our findings underscore the intricate interplay between model complexity, training efficiency, and the quality of generated output in the field of text generation. Achieving a delicate equilibrium between these factors is essential for developing models that not only perform well but also do so in a resource-efficient manner.

Moreover, our study reflects the continuous evolution of natural language processing, where researchers and practitioners constantly strive to find innovative approaches to enhance the creativity and quality of text generation. As we delve deeper into this field, the pursuit of novel methodologies and the incorporation of advanced models will undoubtedly contribute to further advancements in the realm of creative text generation.

References

Barradas, G. T., & Sakka, L. S. (2022, March). When words matter: A cross-cultural perspective on lyrics and their relationship to musical emotions. *Psychol. Music*, *50*(2), 650–669.

Better language models and their implications. (n.d.). https://openai.com/research/better-language-models. (Accessed: 2023-12-4)

CTRL. (n.d.). https://huggingface.co/docs/transformers/model_doc/ctrl. (Accessed: 2023-12-4)

Dhandapani, A., Ilakiyaselvan, N., Mandal, S., Bhadra, S., & Viswanathan, V. (2023). Lyrics generation using LSTM and RNN. In *Lecture notes in electrical engineering* (pp. 371–388). Singapore: Springer Nature Singapore.

Kapoor, K. (2021, August). *Lyrics.*

Ma, X., Wang, Y., Kan, M.-Y., & Lee, W. S. (2021, October). AI-lyricist. In *Proceedings of the 29th ACM international conference on multimedia.* New York, NY, USA: ACM.

Phi, M. (2018, September). *Illustrated guide to LSTM's and GRU's: A step by step explanation.* https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. (Accessed: 2023-12-4)

Tong, Y. J., Liu, Y. L., Wang, J., & Xin, G. J. (2019, June). Text steganography on RNN-Generated lyrics. *Math. Biosci. Eng.*, *16*(5), 5451–5463.

Weights & biases. (2023, December). https://wandb.ai/mukilan/T5_transformer/reports/Exploring-Google-s-T5-Text-To-Text-Transformer-Model--VmlldzoyNjkzOTE2. Weights & Biases, Inc. (Accessed: 2023-12-4)