PROJECT REPORT

GENERATING MOOD-SPECIFIC SONG LYRICS USING NLP, LSTM AND RNN

SREENIVAS KASULANATI
MS IN DATA SCIENCE
skasu5@unh.newhaven.edu

SUPRAJA KODIGANTI
MS IN DATA SCIENCE
skodi7@unh.newhaven.edu

***ABSTRACT*:** The generation of mood-specific song lyrics presents a unique challenge in the field of natural language processing (NLP) and artificial intelligence (AI). This project explores the application of Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNN) to create song lyrics that correspond to specified emotional tones. Utilizing a dataset of song lyrics annotated with mood classifications, we develop a model capable of understanding and generating text that not only maintains lyrical coherence but also aligns with the desired emotional context. The LSTM and RNN architectures are employed due to their proficiency in handling sequences and their ability to remember long-term dependencies, making them ideal for text generation tasks. Our approach includes preprocessing the data with tokenization and encoding techniques, followed by training the neural networks to predict subsequent words based on previous sequences conditioned on the mood. The results demonstrate that our model can effectively generate lyrics that are not only creatively plausible but also relevant to the specified moods. This capability could be instrumental in assisting songwriters and composers in the creative process, providing them with inspiration or complete verses that reflect the intended emotional tone of their compositions.

KEYWORDS: LSTM, RNN

I. INTRODUCTION:

The intersection of natural language processing (NLP) and creative arts such as music offers fascinating opportunities for artificial intelligence (AI) applications. This project delves into the realm of automated song lyric generation, aiming to produce text that is not only creatively engaging but also emotionally aligned with the desired moods of the listeners. The ability to generate mood-specific lyrics using AI can significantly enhance the creative process for songwriters and music producers, providing them with novel content that resonates with audiences on an emotional level.

Traditionally, the creation of song lyrics has been an intensely personal and human-centric task, relying on the songwriter’s ability to evoke emotions through words. However, the advent of advanced NLP techniques, such as Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNN), has opened new avenues for automating and enhancing this creative process. These models are particularly suited for text generation due to their capability to process sequences of data and maintain context over longer stretches of text, which is crucial for maintaining coherence in lyrics.

This project leverages the OpenAI API to further refine the generation process by integrating cutting-edge sentiment analysis, ensuring that the generated lyrics accurately reflect the intended emotional tones. By automating sentiment analysis, the project not only streamlines the lyric creation process but also enhances the lyrical quality, making the final output more contextually relevant and emotionally engaging.

Moreover, this initiative contributes to NLP research by showcasing the practical application of LSTM and RNN models in a creative domain. By generating contextually relevant and mood-aligned lyrics, the project highlights the potential of AI to extend beyond conventional applications, influencing artistic domains traditionally dominated by human creativity.

II. DATA SET OVERVIEW:

Dataset is downloaded from Kaggle. This dataset contains the details of more than 25,000 songs with their Lyrics. It also contains details of Albums released by singers. This contain lyrics of 150 singers whose list is attached below: Taylor Swift Lyrics, Ariana Grande Lyrics, Nelly Furtado Lyrics, Shawn Mendes Lyrics, Westlife Lyrics, Frank Sinatra Lyrics, Britney Spears Lyrics, Justin Timberlake Lyrics, Eminem Lyrics, Linkin Park Lyrics, Jhonny Cash Lyrics, Bob Marley (Bob Marley & The Wailers) Lyrics, John Legend Lyrics etc.,

III. METHODOLOGY:

1. Data Pre-Processing:

- Data Acquisition: The data is sourced from Kaggle, featuring detailed song lyrics and album information from various artists.
- Data Cleaning: Techniques to clean the data might include removing duplicates, filling missing values, and standardizing text data.
- Data Exploration: Analyzing the dataset to understand the distribution of data, the prevalence of various musical themes, or mood indications within lyrics.
- Feature Extraction: Identifying and extracting features that are relevant for training models, such as tokenizing text and using techniques like TF-IDF for text representation.

2. Analysis & Generation:

- Model Selection: Choosing LSTM and RNN for their strengths in handling sequential data like text, crucial for maintaining the flow and thematic consistency of lyrics.
- Model Training: Implementing a training regime that involves dividing the data into training and validation sets, using the cleaned and preprocessed lyrics.
- Hyperparameter Tuning: Adjusting parameters such as the number of layers, hidden units, and learning rate to optimize performance.
- Sentiment Analysis: Applying sentiment analysis, through the OpenAI API, to ensure the generated lyrics reflect the desired mood.
- Lyrics Generation: Developing a system to generate new lyrics based on input mood, utilizing the trained models to predict and stitch together sequences of lyrics.

IV. BACKGROUND DETAILS:

1. Recurrent Neural Networks

RNNs are a form of Neural Network that allows for previous outputs as inputs. They employ the abstract idea of Sequential Memory, which states that a sequence of items provides greater information and efficiency in obtaining a model's output (think the Alphabet). Sequences have an intrinsic property of structure and information. RNNs can recognize patterns and sequences and utilize them to create predictions. They do this by establishing a feedback loop in the network that uses prior data to guide the next iteration of inputs. The hidden state is a vector representation of prior inputs in this feedback loop. This permits information to remain across the model, which is impossible with standard feed-forward networks.

RNNs have several advantages, including the ability to analyze any length of input sequence since the model's size does not scale with the amount of the input, and the ability to consider previous historical data while operating. RNNs help identify patterns in data sequences such as text, audio, or numerical time series data because of their benefits. Our method has a narrower domain using RNNs, however, the Vanishing Gradient Problem might impair text (lyric) production.

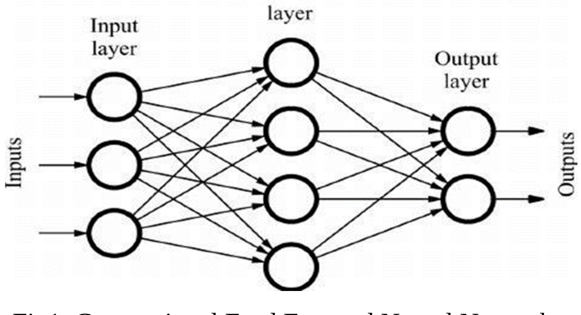


Fig1. Conventional Feed Forward Neural Network

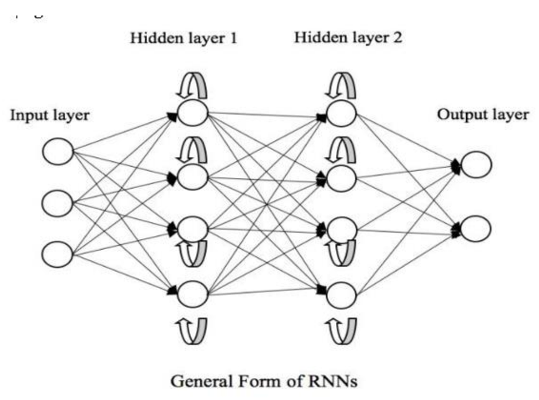
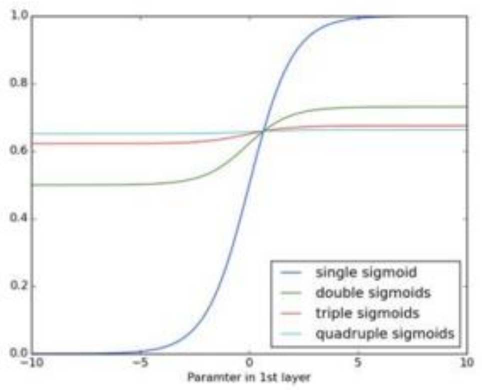


Fig2. Recurrent Neural Network

2. Long Short-Term Memory (LSTM):

As previously stated, RNNs have difficulties learning long-term dependencies across time steps. RNNs have a short-term memory and cannot access information from the past. The Vanishing Gradient Problem occurs when the gradient of the loss function (values used to update weights) diminishes rapidly during back-propagation and eventually disappears. A gradient that gets too thin (and finally zero) does not help you learn much. Because of the microscopic adjustments of the weights by exceedingly small gradients, the neural network's earlier layers are unable to learn.



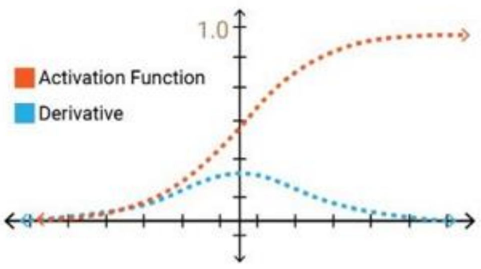


Fig3 & 4: Vanishing Gradient Problem

One can utilize LSTMs to fix this issue. An LSTM is a form of RNN that can learn long-term dependencies. LSTMs can preserve errors that can be transmitted backward in time and layers. They improve RNNs by allowing them to learn across multiple time steps by keeping the error value constant. LSTMs can do this by employing gates, which are tensor operations that can figure out what information to add or remove from the hidden state. The LSTM network has an input gate, a "forget" gate, and an output gate in each unit. The input gate can evaluate the worth of provided data. The "forget gate" can select whether information should be discarded or saved for later use. The output gate is responsible for determining if the information is relevant at a given phase. The gates take inputs as sigmoid function parameters throughout each phase. The sigmoid returns a number between 0 (allow nothing through the gate) and 1 (permit everything through the gate) (let everything through the gate). This idea is used in backpropagation (updating layer values).

LSTMs may pick which information to take forward and which information to drop by employing these gates. These gates assist govern information flow inside the network and allow the error value to remain throughout the network, giving them a significant benefit over RNNs. We concluded that adopting LSTMs would be the best line of action for our lyric generation model. LSTM units were used to form a network because we are trying to produce lyrics based on an initial input sequence.

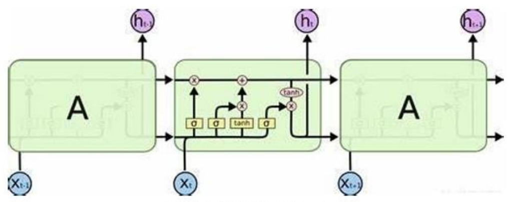


Fig5. LSTM Architecture

V. COMPARISION :

1. Basic Architecture:

- RNNs: These are designed to handle sequence prediction problems by processing sequences one element at a time and maintaining a state that encodes information about the sequence seen so far. The primary structure of an RNN includes a loop that allows information to persist from one step of the network to the next.
- LSTMs: An LSTM is a special kind of RNN specifically designed to avoid the long-term dependency problem. It contains

special units called memory cells which replace traditional neurons in the hidden layer of the network. LSTMs can maintain a long-term state alongside the short-term state, with three gates (input, forget, and output) that regulate the flow of information.

2. **Problem with Vanishing and Exploding Gradients**

- RNNs: They are prone to vanishing and exploding gradients—a problem where gradients, during backpropagation, either shrink to zero (vanish) or become too large (explode), making the network hard to train, particularly for long sequences.
- LSTMs: They effectively combat these issues with their gate-based architecture. The gates in LSTMs can learn which data in a sequence is important to keep or throw away, thus mitigating the impact of vanishing gradients and more capably handling long-range dependencies.

3. **Training and Performance**

- RNNs: Simpler and faster to train than LSTMs due to their simpler architecture. However, they may struggle with longer sequences, making them less effective for complex sequence modeling tasks that involve long-range dependencies.
- LSTMs: While more computationally intensive to train, LSTMs perform significantly better on tasks that require understanding context over longer sequences. This makes them more suitable for sophisticated applications like speech recognition, language modeling, and text generation.

4. **Use Cases**

- RNNs: Effective for shorter sequence prediction tasks and where model complexity and training time are major concerns. They are used in models where quick predictions are more valuable than precise accuracy over long sequences.
- LSTMs: Preferred for tasks where the sequence length is considerable and the context is crucial, such as in machine translation, sequential image captioning, and complex time series forecasting.

5. **Flexibility and Variants**

- RNNs: Have simpler variants and can be modified to create bidirectional RNNs or integrate with other types of neural network layers.
- LSTMs: Also have several variants like GRU (Gated Recurrent Unit), which simplifies the LSTM architecture while maintaining a similar level of efficacy. These variants offer flexibility in dealing with distinct types of sequence data and computational constraints.

VI. MODEL TRAINING :

1. **Epoch Loop**

- Purpose: An epoch represents one complete pass through the entire training dataset. Looping through multiple epochs is essential because it allows the model to learn progressively from the entire dataset multiple times, which is crucial for complex data patterns.
- Implementation: for epoch in range(max_epochs) iterates from 0 to max_epochs - 1, ensuring the model trains for the specified number of epochs.

2. **Time Tracking**

- Purpose: Tracking the time for each epoch helps in monitoring the training efficiency. It's useful for debugging performance issues and for optimizing the training time, especially when scaling up the model or data.
- Implementation: The time.time() function is used at the start and end of each epoch. The difference gives the duration of one epoch.

3. **Setting Training Mode**

- Purpose: Certain layers like dropout or batch normalization behave differently during training and inference. Setting the model to training mode (model.train()) ensures these layers behave correctly, e.g., dropout layers randomly disable neurons only during training to prevent overfitting.
- Implementation: The model.train() method explicitly sets these layers to training mode.

4. **Cross Entropy Loss**

- Purpose: Cross entropy loss is a standard loss function for classification problems, which measures the performance of a classification model whose output is a probability value between 0 and 1. It penalizes the probability of divergence from the actual label.
- Implementation: torch.nn.CrossEntropyLoss() initializes the loss function, and L = crossentropyloss(pred_y, y) computes the loss between the predictions (pred_y) and the true labels (y).

5. **Zeroing Gradients**

- Purpose: In neural networks, gradients accumulate by default. Clearing the gradient buffers before each weight update prevents gradients from previous forward passes from influencing the current pass.
- Implementation: opt.zero_grad() resets the gradient of all model parameters to zero.

6. **Backpropagation**

- Purpose: Backpropagation is used to calculate the gradient of the loss function with respect to each parameter (weight and bias). This tells us how to adjust the parameters to minimize the loss.
- Implementation: L.backward() computes the derivative of the loss with respect to the parameters.

7. **Optimization Step**

- Purpose: This step updates the model parameters based on the gradients calculated during backpropagation and the learning rate. It is crucial for the learning process as it iteratively reduces the loss.
- Implementation: opt.step() adjusts the weights based on the computed gradients, using the chosen optimization algorithm (like SGD, Adam, etc.).

8. **Loss Tracking and Printing**

- Purpose: Tracking the loss provides insight into how well the model is learning and converging towards a solution. Printing these metrics provides a live update during training to monitor progress and diagnose issues early.
- Implementation: temploss += L.item() / len(dataloader) calculates the average loss per epoch, and print(...) outputs the epoch number, loss, and time taken for the epoch.

VII. COMPARISION :

1. **Function Overview**

- Purpose: To generate song lyrics based on given seed phrases using a trained neural network model.
- Inputs:

- seeds: A list of seed phrases to initiate the generation process.
- token: A tokenizer that maps words to indices and vice versa.
- model: The trained model that predicts the next word based on the current sequence of words.
- maxiter: The maximum number of iterations, or words to generate, following the seed.
- dims: The dimensions of the input tensor, typically the maximum sequence length the model can handle.

2. **Generation Process**

- Initialization: Prepares a list to store the generated songs and a counter for tracking purposes.
- Iterative Generation: Each seed phrase is used to start generating a new song. For each seed, the function generates words up to maxiter times, updating the input sequence for the model at each step.
- Model Inference:Converts words in the current sequence to indices using the tokenizer.Feeds the sequence to the model and applies a SoftMax function to predict the probability distribution of the next word.Samples a word from this distribution using a custom sampling function (prob_sample).
- Post-processing:Handles special tokens like newlines correctly.Updates the seed sequence by shifting the words and adding the novel word.
- Compilation of Results: Combines the initial seed and all generated words into a single string.Appends the completed song to the list of generated songs.

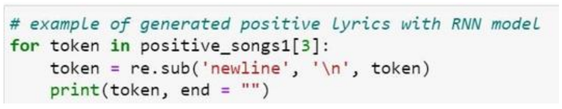
3. Key Features

- Dynamic Seed Update: Maintains the sequence length by updating the seed with newly generated words, ensuring the context is carried forward.
- Handling of Special Characters: Translates newline characters to and from a placeholder to accommodate model vocabulary constraints.
- Probabilistic Word Selection: Employs a probability-based selection method to choose each subsequent word, enhancing the naturalness and variability of the generated lyrics.

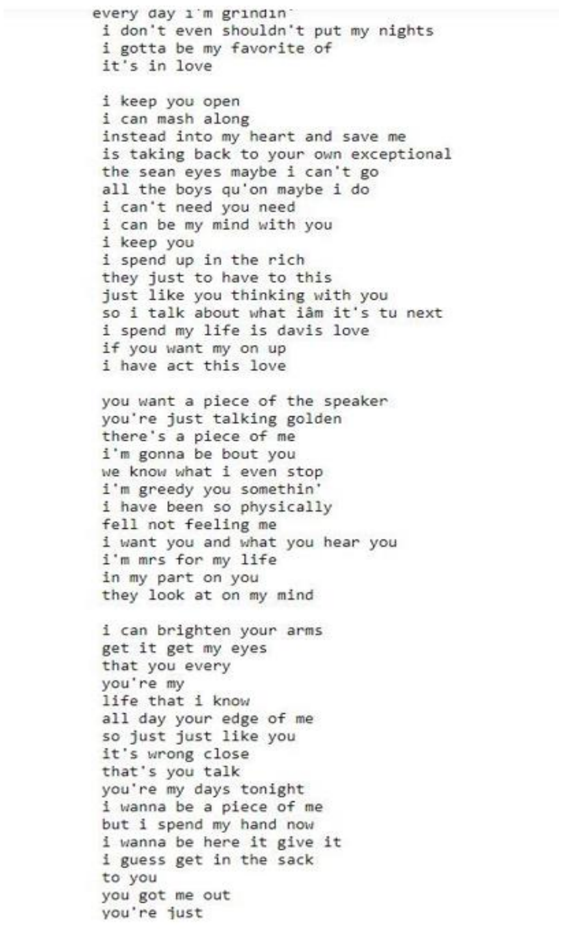
VIII. RESULTS AND ANALYSIS:

1. RESULTS WITH RNN

Positive Lyrics with RNN model

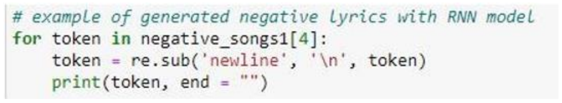


```
# example of generated positive lyrics with RNN model
for token in positive_songs1[3]:
    token = re.sub('newline', '\n', token)
    print(token, end = "")
```

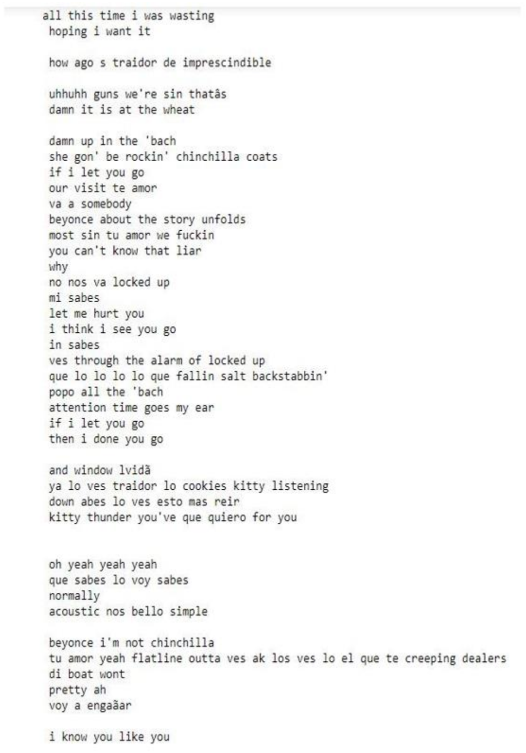


```
every day i'm grindin'
 i don't even shouldn't put my nights
 i gotta be my favorite of
 it's in love

 i keep you open
 i can mash along
 instead into my heart and save me
 is taking back to your own exceptional
 the sean eyes maybe i can't go
 all the boys qu'on maybe i do
 i can't need you need
 i can be my mind with you
 i keep you
 i spend up in the rich
 they just to have to this
 just like you thinking with you
 so i talk about what iâm it's tu next
 i spend my life is davis love
 if you want my on up
 i have act this love

 you want a piece of the speaker
 you're just talking golden
 there's a piece of me
 i'm gonna be bout you
 we know what i even stop
 i'm greedy you somethin'
 i have been so physically
 fell not feeling me
 i want you and what you hear you
 i'm mrs for my life
 in my part on you
 they look at on my mind

 i can brighten your arms
 get it get my eyes
 that you every
 you're my
 life that i know
 all day your edge of me
 so just just like you
 it's wrong close
 that's you talk
 you're my days tonight
 i wanna be a piece of me
 but i spend my hand now
 i wanna be here it give it
 i guess get in the sack
 to you
 you got me out
 you're just
```

Negative Lyrics with RNN



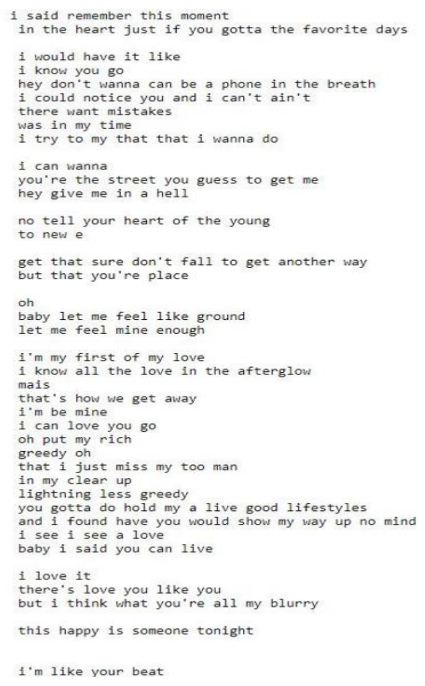
```
# example of generated negative lyrics with RNN model
for token in negative_songs1[4]:
    token = re.sub('newline', '\n', token)
    print(token, end = "")
```



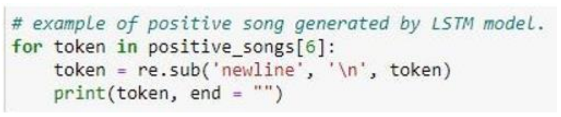
```
all this time i was wasting
 hoping i want it

 how ago s traidor de imprescindible

 uhhuhh guns we're sin thatâs
 damn it is at the wheat

 damn up in the 'bach
 she gon' be rockin' chinchilla coats
 if i let you go
 our visit te amor
 va a somebody
 beyonce about the story unfolds
 most sin tu amor we fuckin
 you can't know that liar
 why
 no nos va locked up
 mi sabes
 let me hurt you
 i think i see you go
 in sabes
 ves through the alarm of locked up
 que lo lo lo lo que fallin salt backstabbin'
 popo all the 'bach
 attention time goes my ear
 if i let you go
 then i done you go

 and window lvidã
 ya lo ves traidor lo cookies kitty listening
 down abes lo ves esto mas reir
 kitty thunder you've que quiero for you


 oh yeah yeah yeah
 que sabes lo voy sabes
 normally
 acoustic nos bello simple

 beyonce i'm not chinchilla
 tu amor yeah flatline outta ves ak los ves lo el que te creeping dealers
 di boat wont
 pretty ah
 voy a engaãar

 i know you like you
```

2. RESULTS WITH LSTM

Positive Lyrics with LSTM model

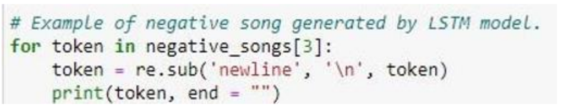


```
i said remember this moment
 in the heart just if you gotta the favorite days

 i would have it like
 i know you go
 hey don't wanna can be a phone in the breath
 i could notice you and i can't ain't
 there want mistakes
 was in my time
 i try to my that that i wanna do

 i can wanna
 you're the street you guess to get me
 hey give me in a hell

 no tell your heart of the young
 to new e

 get that sure don't fall to get another way
 but that you're place

 oh
 baby let me feel like ground
 let me feel mine enough

 i'm my first of my love
 i know all the love in the afterglow
 mais
 that's how we get away
 i'm be mine
 i can love you go
 oh put my rich
 greedy oh
 that i just miss my too man
 in my clear up
 lightning less greedy
 you gotta do hold my a live good lifestyles
 and i found have you would show my way up no mind
 i see i see a love
 baby i said you can live

 i love it
 there's love you like you
 but i think what you're all my blurry

 this happy is someone tonight


 i'm like your beat
```

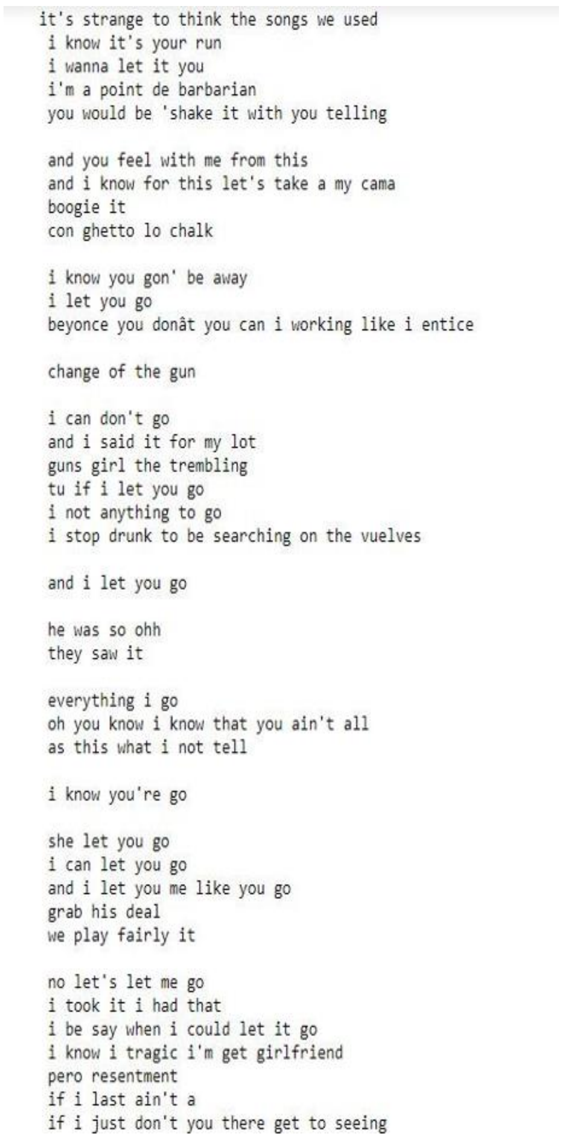


```python
# example of positive song generated by LSTM model.
for token in positive_songs[6]:
    token = re.sub('newline', '\n', token)
    print(token, end = "")
```

Negative Lyrics by LSTM model



```python
# Example of negative song generated by LSTM model.
for token in negative_songs[3]:
    token = re.sub('newline', '\n', token)
    print(token, end = "")
```



```
 it's strange to think the songs we used
  i know it's your run
  i wanna let it you
  i'm a point de barbarian
  you would be 'shake it with you telling

  and you feel with me from this
  and i know for this let's take a my cama
  boogie it
  con ghetto lo chalk

  i know you gon' be away
  i let you go
  beyonce you donât you can i working like i entice

  change of the gun

  i can don't go
  and i said it for my lot
  guns girl the trembling
  tu if i let you go
  i not anything to go
  i stop drunk to be searching on the vuelves

  and i let you go

  he was so ohh
  they saw it

  everything i go
  oh you know i know that you ain't all
  as this what i not tell

  i know you're go

  she let you go
  i can let you go
  and i let you me like you go
  grab his deal
  we play fairly it

  no let's let me go
  i took it i had that
  i be say when i could let it go
  i know i tragic i'm get girlfriend
  pero resentment
  if i last ain't a
  if i just don't you there get to seeing
```

ANALYSIS OF RESULTS:

```
distinct_words :  average number of tokens :  avg_line_length :  avg_word_len
296 2209.37 426.59 5.195445416613766
300 2007.7 385.00666666666666 5.223250874305751
```

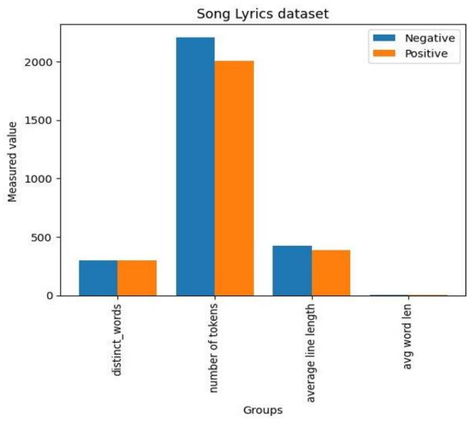


Fig6. Song Lyrics Dataset Analysis

Function named metrics that calculates various textual statistics from datasets of song lyrics. It specifically computes the number of distinct words, average number of tokens per lyric, average line length, and average word length. These metrics help analyze the vocabulary diversity, lyric length, line complexity, and word verbosity within the lyrics.

The function is applied to two datasets labeled as 'negative' and 'positive', reflecting different themes or sentiments in the lyrics. After calculating these metrics for both datasets, the results are visualized using a bar chart that compares the two datasets across the four calculated metrics.
This visualization effectively demonstrates any linguistic differences between the two themes, providing insights into how different sentiments might influence the style and complexity of song lyrics.

```
distinct_words :  average number of tokens :  avg_word_length :   avg_word_length:
10 1230.8 6.284018127156896 2.7679090188514417
10 1218.7 6.7305984510444405 2.7811732937757028
```

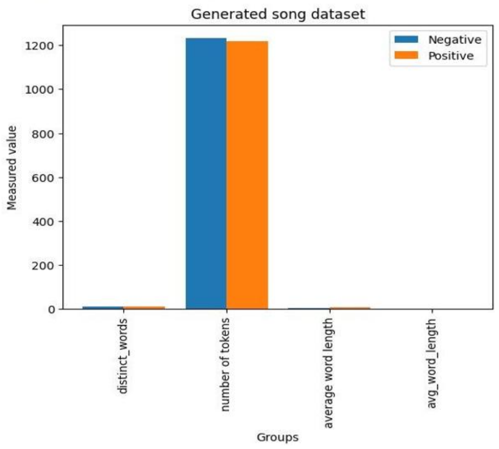


Fig7. Generated Song Dataset Analysis

The gen_metrics function calculates key textual me
trics for song lyrics datasets, specifically designed t
o analyze the distinct words, average number of tok
ens (words) per lyric, average line length, and avera
ge word length. The function is applied to two sepa
rate datasets characterized by their negative and po
sitive themes, helping assess and compare the lexic
al diversity, lyric verbosity, line complexity, and lin
guistic intricacy between the two.

The calculated metrics for both datasets are then vis
ually represented in a bar chart, which plots distinct
words, number of tokens, average line length, and a
verage word length for both negative and positive t
hemed lyrics. This visualization allows for a straigh
tforward comparison, highlighting how thematic el
ements like sentiment may influence various aspect
s of lyric composition, such as vocabulary richness
and stylistic complexity.

This approach provides a comprehensive understan
ding of the textual characteristics in different sets o
f song lyrics, highlighting the influence of sentimen
t on lyrical content and structure effectively.

METRICS:

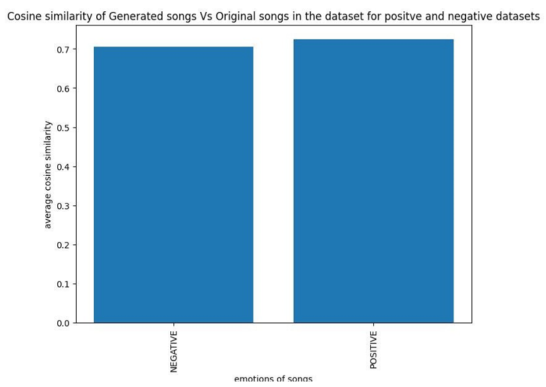


Fig8. Cosine Similarity Graph

By comparing the average cosine similarities, one can evaluate how well the models generating these lyrics are performing, especially in terms of maintaining thematic consistency with the original songs.

Cosine similarity here is a critical metric because it quantifies the likeness in semantic space between the texts, indicating how closely the generated lyrics match the style and substance of the target (original) lyrics. This kind of analysis is crucial in applications like automated songwriting or other creative text generation tasks where maintaining a certain thematic or emotional tone is necessary.

IX. CONCLUSION:

- Model Efficacy: The project demonstrates the model's ability to generate coherent and contextually relevant lyrics using seed phrases. The use of advanced sampling techniques ensures that the lyrics are not only diverse but also maintain a logical flow.
- Textual Analysis Insight: The analysis of distinct words, average tokens, line length, and word length provides valuable insights into the lyrical content's complexity and diversity. This shows that the model can manipulate linguistic elements effectively to produce lyrics that mirror human-like quality and variety.
- Thematic Accuracy: Cosine similarity scores between generated and original lyrics indicate that the model can capture the emotional and thematic essence of the source material. However, there is room for improvement, especially in enhancing the semantic alignment with the original lyrics to ensure the generated content consistently reflects the intended sentiment.
- Visualization Utility: The use of visual aids in comparing datasets has proven effective for presenting complex data in an accessible manner, aiding in quicker decision-making and analysis.

Overall, the project successfully integrates NLP techniques to innovate automated lyric generation, providing a foundation for further enhancements in creative text generation. Future work could focus on refining the model to improve thematic alignment and exploring other dimensions of lyric analysis to enrich the understanding and generation of song lyrics.

X. REFERENCES:

[1] Pyrovolakis et al. demonstrated mood detection in songs using separate analysis of audio signals and lyrics[3].

[2] Mahey discussed sentiment and emotion analysis in lyrics using NLP, citing a case study on Biggie Smalls' "Suicidal Thoughts"[6].

GITHUB LINK:

https://github.com/Supraja27/DSCI6004--Natural-Language-Processing