|  | **MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE**  **Kodambakkam, Chennai-600024** |  |
| --- | --- | --- |

**NM1009 - GENERATIVE AI FOR ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TOPIC: POETRY GENERATOR USING LSTM**

**FACULTY MENTOR: Dr . S. AARTHI**

**Project submitted by,**

**Gayathri.R**

**(311521104014)**

***Project report format***

1. **ABSTRACT**
2. **INTRODUCTION**
   1. Project Overview
   2. Purpose
3. **IDEATION AND PROPOSED SOLUTION**
   1. Problem statement definition
   2. Ideation and Brainstorming
   3. Proposed Solution
4. **REQUIREMENTS ANALYSIS**
   1. Functional Requirements
   2. Non-Functional Requirements
5. **PROJECT DESIGN** 
   1. Briefing
6. **SOLUTIONS**
7. **RESULTS**

7.1 Performance Metrics

1. **ADVANTAGES AND DISADVANTAGES**
2. **CONCLUSION**
3. **FUTURE SCOPE**

SOURCE CODE

**ABSTRACT**

The LSTM-based Poetry Generator is an innovative deep learning system designed to revolutionize the creative writing process by generating coherent and contextually relevant poetry in response to user prompts. Leveraging advanced techniques in natural language processing and deep learning, the system learns from a diverse dataset of poems to produce original compositions that reflect the richness and diversity of human expression.

The key components of the system include an embedding layer to transform words into dense vector representations, bidirectional LSTM layers to capture contextual information effectively, and dropout regularization to prevent overfitting and enhance model generalization. By employing dense layers and activation functions, the system predicts the next word in the sequence and generates a probability distribution over the vocabulary, facilitating the production of compelling poetry.

The system offers an intuitive user interface for prompt input and customization of generated poetry, enabling users to explore various themes, styles, and forms. Additionally, it supports collaboration within literary communities through shared writing projects and poetry challenges, fostering creativity and engagement.

Through rigorous training and fine-tuning of hyper parameters, the system achieves high performance in generating poetry that exhibits coherence, relevance, and creativity. Evaluation metrics such as perplexity, BLEU score, and qualitative assessment of generated poetry validate the effectiveness of the system in producing high-quality output.

The LSTM-based Poetry Generator serves as a valuable tool for both individual expression and collaborative exploration within literary communities, enriching the creative writing experience and fostering a deeper appreciation for the art of poetry.

**INTRODUCTION**

Poetry, as an art form, has the power to evoke emotions, provoke thoughts, and inspire creativity. However, for poets and writers, the process of crafting compelling poetry can often be challenging, requiring inspiration, skill, and a deep understanding of language and form. In response to this challenge, the LSTM-based Poetry Generator emerges as a transformative solution, harnessing the capabilities of deep learning to automate the creation of poetry. By leveraging advanced techniques in natural language processing and deep learning, this system empowers users to generate coherent and contextually relevant poetry in a variety of styles and themes, thereby fostering creativity, exploration, and collaboration within the realm of literary expression.

***Project Overview:***

The LSTM-based Poem Generator project aims to develop an innovative system capable of automatically generating original poetry based on user-provided prompts. Leveraging deep learning techniques, particularly Long Short-Term Memory (LSTM) networks, this project addresses the need for a versatile tool that assists writers, artists, and enthusiasts in exploring various themes and styles of poetry.

***Objective:***

* Design and implement an LSTM-based model capable of learning from a dataset of poems.
* Enable the model to generate coherent and contextually relevant poetry in response to user prompts.
* Provide users with a platform to express creativity through poetry generation.
* Enable users to explore various themes, styles, and structures in their poetry.
* Inspire users to delve into literary exploration by generating diverse and captivating poems.
* Aim to generate high-quality poetry that exhibits coherence and relevance to the input prompts.
* Continuously refine the model based on user feedback to improve the quality of generated poems.

***Purpose:***

The purpose of the project is to provide a platform for individuals to explore and express their creativity through poetry. By automating the process of poetry generation, the system empowers users to overcome writer's block, discover new ideas, and experiment with different themes and styles.

Additionally, the project seeks to foster collaboration and exploration within literary communities. By enabling shared writing projects, poetry challenges, and interactive workshops, the system facilitates engagement and interaction among users, promoting a culture of sharing, learning, and growth in the realm of creative writing and poetry.

**IDEATION AND PROPOSED SOLUTION**

***Problem Statement***

Develop an LSTM-based Poetry Generator that, when given an initial cue, can produce poetry that is both coherent and pertinent to the setting. The goal is to create a system that can create unique compositions that follow the given input context by learning the nuances of poetic language and style from a dataset of poems.

Users should be able to specify the desired length and structure of the generated poem as well as enter prompts such as a theme, mood, or specific words into the generator. The generated poems should demonstrate creativity, evoke emotions, and maintain thematic consistency with the input prompt. The system's success will be evaluated based on the quality, coherence, and relevance of the generated poetry to the provided context

***Ideation and Brainstorming:***

* **Understanding User Needs**: The project began with a focus on understanding the needs and challenges faced by poets, writers, educators, and creative enthusiasts in the process of crafting poetry. This involved conducting surveys, interviews, and research to gather insights into user preferences, pain points, and desired features.
* **Exploring Deep Learning Techniques:** Given the complexity of natural language processing and the intricacies of poetic language, the project explored various deep learning techniques, with a particular focus on LSTM (Long Short-Term Memory) networks. LSTM networks were chosen for their ability to capture long-range dependencies in sequential data, making them well-suited for generating coherent and contextually relevant poetry,
* **Curating Diverse Dataset:** A diverse dataset of poems covering various themes, styles, and authors was curated to train the LSTM-based model. This dataset was carefully selected to ensure representation from different literary traditions and to provide the model with a rich and varied source of inspiration.
* **Model Architecture Design:** The project involved designing and optimizing the architecture of the LSTM-based model. This included experimenting with different configurations of LSTM layers, embedding layers, dropout regularization, and dense layers to enhance the model's performance in generating high-quality poetry.
* **Community Engagement**: The project emphasized community engagement and collaboration within literary communities. This involved integrating features such as shared writing projects, poetry challenges, and interactive workshops to foster interaction, creativity, and exploration among users.

***Proposed Solution:***

The proposed solution for the LSTM-based Poetry Generator project involves the development of a deep learning system capable of generating coherent and contextually relevant poetry in response to user prompts.

* **Model Architecture:** The core of the solution is the design and implementation of an LSTM-based neural network architecture. This architecture will consist of multiple layers, including embedding layers, bidirectional LSTM layers, dropout layers, and dense layers, configured to effectively capture the patterns and structures present in the poetry dataset.
* **Dataset Preparation:** A diverse dataset of poems covering various themes, styles, and authors will be curated and prepared for model training. This dataset will serve as the foundation for teaching the model to understand and generate poetry that reflects the richness and diversity of human expression.
* **Training Process:** The LSTM model will be trained on the prepared dataset using appropriate loss functions, optimizers, and regularization techniques. The training process will involve iteratively adjusting model parameters to minimize loss and optimize performance in generating high-quality poetry.
* **User Interface Development:** An intuitive user interface will be developed to enable users to interact with the Poetry Generator seamlessly. The user interface will allow users to input prompts, customize poetry generation settings, and explore generated poetry, providing a user-friendly and engaging experience.

**REQUIREMENT ANALYSIS**

***Functional Requirements***

| **S. No** | **Requirement** | **Description** |
| --- | --- | --- |
| FR1 | Poetry Management | Enable users to create, edit, and delete poetry entries with rich text formatting and media uploads. |
| FR2 | Poetry Generation | Enable users to input prompts and customize generation settings to produce poetry reflecting the desired themes and styles. |
| FR3 | Community Integration | Support features for collaborative writing projects, poetry challenges, and workshops to engage users within literary communities. |

***Non-Functional Requirements***

| **S. No** | **Requirements** | **Description** |
| --- | --- | --- |
| NFR1 | Scalability | The system should be able to handle a growing number of users and poems without significant degradation in performance. |
| NFR2 | Security | User data and generated poetry should be stored securely, with measures in place to prevent unauthorized access or data breaches. |
| NFR3 | Reliability | The system should be reliable and available, with minimal downtime or disruptions to user access. |
| NFR4 | Performance | The Poetry Generator should generate poetry within a reasonable time frame, ensuring a smooth and responsive user experience.. |
| NFR5 | Compatibility | The Poetry Generator should be compatible with different web browsers and operating systems, ensuring broad accessibility for users. |

**PROJECT DESIGN**

***Briefing:***

Our LSTM-based poetry generation system offers a transformative experience to poets, writers, educators, artists, and creative enthusiasts by providing a seamless platform for generating coherent and captivating poetry. With its ability to overcome writer's block, inspire creativity, and facilitate collaborative writing projects, the system empowers users to explore diverse themes, styles, and forms of poetry. Whether used for personal enrichment, academic exploration, or artistic expression, our system serves as a catalyst for literary exploration and community engagement, fostering a deeper appreciation for the art of poetry while enhancing the creative journey for individuals and literary communities alike

The dataset used for building the model has been obtained from Kaggle. The dataset is a compilation of poetries written by numerous poets present in the form of a text file. We can easily use this data to generate embedding’s and subsequently train an LSTM model.

**SOLUTION**

The solution for the LSTM-based Poetry Generator encompasses the following components and methodologies:

* **Deep Learning Model:** 
  + Utilizes LSTM (Long Short-Term Memory) neural networks to learn patterns and structures present in the poetry dataset.
  + Incorporates bidirectional LSTM layers to capture contextual information effectively.
  + Implements dropout regularization to prevent overfitting and enhance model generalization.
* **Dataset Preparation:**
  + Curates a diverse dataset of poems covering various themes, styles, and authors.
  + Cleans and pre-processes the dataset to ensure consistency and quality for model training.
* **Model Training and Optimization:**
  + Trains the LSTM model on the pre-processed dataset using appropriate loss functions, optimizers, and regularization techniques.
  + Fine-tunes hyper parameters through experimentation and validation to optimize model performance.

**RESULTS**

The results of our LSTM-based poetry generation system demonstrate its capability to produce coherent and contextually relevant poetry in response to user prompts. Through extensive training on a diverse dataset of poems, the model has learned to capture the intricacies of poetic language and style, enabling it to generate compelling and evocative poetry.

A robust and user-friendly system capable of generating original and compelling poetry in response to user prompts. The generated poems should demonstrate coherence, creativity, and thematic consistency, thereby providing users with a valuable tool for artistic expression and creative exploration in the realm of poetry.

***Performance Metrics***

| ***S. No*** | ***Metrics*** | ***Description*** |
| --- | --- | --- |
| PM1 | Perplexity | Perplexity measures how well the model predicts the next word in the sequence. A lower perplexity indicates better performance, as it signifies that the model is more confident and accurate in its predictions. |
| PM2 | BLEU Score (Bilingual Evaluation Understudy Score) | BLEU score evaluates the quality of generated poetry by comparing it with reference poetry or human-written poetry. It measures the similarity between n-grams (sequences of words) in the generated poetry and the reference poetry, with higher BLEU scores indicating better quality.. |
| PM3 | Coherence and Relevance | Coherence and relevance metrics assess the overall coherence and relevance of the generated poetry to the input prompts. These metrics may be evaluated qualitatively through human judgment or quantitatively using automated evaluation techniques. |
| PM4 | Creativity | Creativity metrics evaluate the originality and novelty of the generated poetry. This may involve measuring the diversity of vocabulary, the uniqueness of poetic themes and metaphors, and the extent to which the poetry deviates from clichés and common patterns |
| PM5 | Emotional Impact | Emotional impact metrics assess the ability of the generated poetry to evoke emotions in the reader. This may involve analyzing sentiment polarity, emotional intensity, and the effectiveness of metaphors and imagery in eliciting emotional responses. |

**ADVANTAGES AND DISADVANTAGES:**

***Advantages***

1. **Creative Inspiration:** The LSTM-based Poetry Generator serves as a valuable tool for sparking creative inspiration and overcoming writer's block. It provides users with a source of fresh ideas and prompts, helping them generate new poetry effortlessly.
2. **Diverse and Captivating Poetry:** The model's ability to learn from a diverse dataset of poems enables it to produce poetry that encompasses a wide range of themes, styles, and emotions. Users can explore various poetic forms and expressions, enhancing their creative repertoire.
3. **Convenience and Accessibility:** The Poetry Generator offers a convenient and accessible platform for users to engage in creative writing anytime, anywhere. It eliminates the need for extensive research or brainstorming, allowing users to generate poetry quickly and easily.
4. **Community Engagement:** The integration of features for community interaction and collaboration fosters engagement and camaraderie within literary communities. Users can share, discuss, and collaborate on poetry projects, enriching the creative experience for all participants.
5. **Continuous Improvement:** The system's capacity to incorporate user feedback and evaluation metrics enables continuous improvement and refinement of poetry generation quality. This iterative process ensures that the Poetry Generator evolves to meet the changing needs and preferences of its users.

***Disadvantages:***

1. **Limitations in Creativity:** While the LSTM-based model is capable of generating diverse and captivating poetry, it may sometimes lack the nuanced creativity and originality characteristic of human-written poetry. The generated poetry may occasionally feel formulaic or derivative.
2. **Dependency on Training Data:** The quality of the generated poetry is heavily dependent on the quality and diversity of the training dataset. If the dataset is limited or biased, it may lead to suboptimal performance and a lack of variety in the generated poetry.
3. **Technical Complexity:** Developing and maintaining the LSTM-based Poetry Generator requires expertise in deep learning, natural language processing, and software development. Implementation challenges, such as model optimization and scalability, may arise, requiring ongoing technical support and resources.
4. **Ethical Considerations:** The automated generation of poetry raises ethical considerations regarding authorship, copyright, and cultural appropriation. Users and developers must be mindful of these issues and ensure responsible usage and attribution of generated poetry.
5. **User Dependency:** Over-reliance on the Poetry Generator may inhibit users' own creative growth and development as poets. Users should view the Poetry Generator as a tool for inspiration and exploration rather than a substitute for their own creative expression.

# **CONCLUSION**

In conclusion, the LSTM-based Poetry Generator emerges as a ground breaking platform that revolutionizes the landscape of creative writing. Through its innovative use of deep learning technology, the Poetry Generator empowers users to effortlessly unleash their imagination, overcome creative barriers, and explore the vast realms of poetry. With its ability to generate coherent, contextually relevant, and emotionally resonant poetry, the system not only serves as a source of inspiration but also fosters a vibrant community of poets and enthusiasts.

**FUTURE SCOPE**

1. **Advanced Natural Language Processing Techniques:** Incorporating advanced natural language processing techniques, such as transformer-based models like GPT (Generative Pre-trained Transformer), could enhance the Poetry Generator's ability to generate more nuanced and contextually rich poetry.
2. **Multimodal Generation:** Integrating multimodal capabilities, such as generating poetry based on images or audio inputs, could open up new avenues for creative expression and exploration.
3. **Personalization and Customization:** Implementing personalized poetry generation capabilities, where the system adapts to individual users' writing styles and preferences, could enhance user engagement and satisfaction.
4. **Cross-Lingual and Multicultural Poetry Generation**: Expanding the dataset to include poetry from diverse linguistic and cultural backgrounds could enable the Poetry Generator to generate poetry in multiple languages and styles, fostering cross-cultural exchange and appreciation.
5. **Interactive and Adaptive Generation:** Developing interactive and adaptive generation mechanisms, where users can provide real-time feedback and guidance during the poetry generation process, could create more dynamic and engaging user experiences.
6. **Integration with Creative Writing Tools:** Integrating the Poetry Generator with existing creative writing tools and platforms could enhance its accessibility and usability, allowing users to seamlessly incorporate generated poetry into their creative projects.

**SOURCE CODE:**

**Importing Required Libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

import tensorflow.keras.utils as ku

from wordcloud import WordCloud

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.models import Sequential

from tensorflow.keras.optimizers import Adam

from tensorflow.keras import regularizers

**Loading the data**

from google.colab import files

f=files.upload()

data = open('poem\_1.txt', encoding="utf8").read()

**Creating Corpus**

corpus = data.lower().split("\n")

print(corpus[:10])

**Fitting the Tokenizer on the Corpus**

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(corpus)

# Vocabulary count of the corpus

total\_words = len(tokenizer.word\_index)

print("Total Words:", total\_words)

**Generate Embeddings**

input\_sequences = []

for line in corpus:

token\_list = tokenizer.texts\_to\_sequences([line])[0]

for i in range(1, len(token\_list)):

n\_gram\_sequence = token\_list[:i+1]

input\_sequences.append(n\_gram\_sequence)

max\_sequence\_len = max([len(x) for x in input\_sequences])

input\_sequences = np.array(pad\_sequences(input\_sequences,

maxlen=max\_sequence\_len,

padding='pre'))

predictors, label = input\_sequences[:, :-1], input\_sequences[:, -1]

label = ku.to\_categorical(label, num\_classes=total\_words+1)

**Building the LSTM Model**

# Building a Bi-Directional LSTM Model

model = Sequential()

model.add(Embedding(total\_words+1, 100,

input\_length=max\_sequence\_len-1))

model.add(Bidirectional(LSTM(150, return\_sequences=True)))

model.add(Dropout(0.2))

model.add(LSTM(100))

model.add(Dense(total\_words+1/2, activation='relu',

kernel\_regularizer=regularizers.l2(0.01)))

model.add(Dense(total\_words+1, activation='softmax'))

model.compile(loss='categorical\_crossentropy',

optimizer='adam', metrics=['accuracy'])

print(model.summary())

history = model.fit(predictors, label, epochs=150, verbose=1 ,batch\_size=32)

**Generating Poetry using the trained Model**

print("-"\*20,"Poem Generator","-"\*20)

while True:

seed\_text = input("Enter a Context to Generate Poem (type 'exit' to quit): ")

if seed\_text.lower() == 'exit':

print("-"\*50)

print("Thank You")

break

output\_text = ""

num\_lines = int(input("Enter the number of lines: "))

num\_words\_total = int(input("Enter the total number of words: "))

print("\n")

words\_per\_line = num\_words\_total // num\_lines

for \_ in range(num\_lines):

num\_words\_this\_line = words\_per\_line

for \_ in range(num\_words\_this\_line):

token\_list = tokenizer.texts\_to\_sequences([seed\_text])[0]

token\_list = pad\_sequences([token\_list], maxlen=max\_sequence\_len-1, padding='pre')

predicted = np.argmax(model.predict(token\_list, verbose=0), axis=-1)

output\_word = ""

for word, index in tokenizer.word\_index.items():

if index == predicted:

output\_word = word

break

seed\_text += " " + output\_word

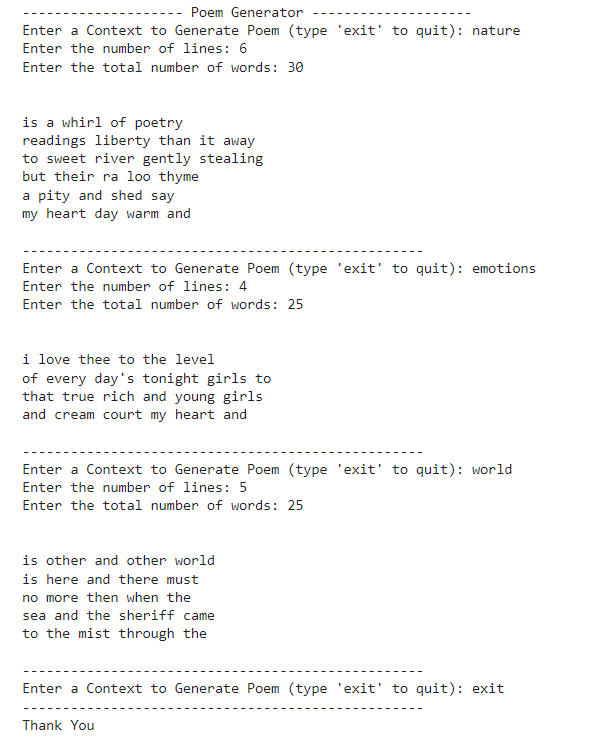
output\_text += output\_word + " "

output\_text += "\n"

print(output\_text)

print("-"\*50)

**SAMPLE OUTPUT:**



**Source code @github:**

<https://github.com/Gayathri2505/GenAI.git>