VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA",MACHHE, BELAGAVI-590018

ML Mini Project Report

on

Poetry Generator and Creative Writing Tool using AI

Submitted in partial fulfillment of the requirements for the VI semester

Bachelor of Engineering

in

Artificial Intelligence & Machine Learning

of

Visvesvaraya Technological University, Belagavi

by

Kondamuri Saikrishna(1CD22AI404)

M Nithin Giri Raja(1CD21AI030)

Under the Guidance of

Dr. Varalatchoumy M

Prof.Syed Hayath

Dept. of AI&ML

Department of Artificial Intelligence & Machine Learning

CAMBRIDGE INSTITUTE OF TECHNOLOGY,BANGALORE-560 036

2023-2024

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

CERTIFICATE

Certified that **Mr. Kondamuri Saikrishna,** bearing USN **1CD22AI404 and Mr. M Nithin Giri Raja** bearing USN **1CD21AI030,** a Bonafide students of **Cambridge Institute of Technology,** has successfully completed the ML Mini Project entitled "**Poetry Generator and Creative Writing Tool using AI"** in partial fulfillment of the requirements for VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-24. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

Mini Project Guides,

Dr.Varalatchoumy.M

Prof. Syed Hayath
Dept. of AI&ML, CITech

Head of the Department,
Dr.Varalatchoumy.M
Dept. of AI&ML, CITech

DECLARATION

We Kondamuri Saikrishna and **M Nithin Giri Raja** of VI semester BE, Artificial Intelligence & Machine Learning, Cambridge Institute of Technology, hereby declare that the ML Mini Project entitled **“Poetry Generator and Creative Writing Tool using AI”** has been carried out by us and submitted in partial fulfillment of the course requirements of VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** as prescribed b**y Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

We also declare that, to the best of my knowledge and belief, the work reported here does not form part of any other report on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:	**Kondamuri Saikrishna**
Place: Bangalore	**1CD22AI404**
	M Nithin Giri Raja
	1CD21AI030

ACKNOWLEDGEMENT

We would like to place on record my deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible**.**

We are extremely thankful to **Dr. G.Indumathi,** Principal, CITech, Bangalore, for providing me the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express my sincere gratitude to **Dr. Varalatchoumy M., Prof. & Head** Dept. of AI & ML, CITech, Bangalore, for her stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We also wish to extend my thanks to Mini Project Guides, **Dr. Varalatchoumy M., Prof. & Head**, and **Prof.Syed Hayath,** Assistant Professors, Dept. of AI&ML, CITech, Bangalore for the critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

Finally to all my friends, classmates who always stood by me in difficult situations also helped me in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to my parents who were a constant source of encouragement and stood by me as pillar of strength for completing this work successfully.

Kondamuri Saikrishna

M Nithin Giri Raja

CONTENTS

Abstract		**i**
Contents		**ii**
List of Figures		**iii**
	CHAPTERS	**PAGE NO.**
Chapter 1	**Introduction**	**1**
	1.1 Problem statement	**2**
	1.2 Objective	3
Chapter 2	**Literature Survey**	**4**
	2.1 GPoeT-2: A GPT-2 Based Poem Generator	4
	2.2 Text Generation Methods Using Pre-Trained GPT-2 Model	4
Chapter 3	**Methodology**	**5**
	3.1 Data Collection	5
	3.2 Data Processing	6
	3.3 Model Training	7
	3.4 Feature Extraction	8
	3.5 System Architecture	9
	3.6 Tools and Technology's	10
Chapter 4	**Implementation**	**12**
	4.1 Detailed Description of Implementation	12
	4.2 Code Snippet	13
Chapter 5	**Result**	**19**
Conclusion and Future Enhancement		**20**
References		**23**

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO
3.1	System Architecture	9
5.1	Poem generation page	19
5.2	creative writing page	19

ABSTRACT

Generating high-quality, creative text remains a significant challenge, as current models often require extensive adaptation for artistic applications. This study introduces a fine-tuning approach for GPT-2, aimed at developing specialized bots for poetry and creative writing. By leveraging GPT-2's advanced language capabilities, the model is trained on diverse datasets of poems and creative texts, enhancing its ability to produce engaging and contextually rich literary content. The fine-tuning process focuses on refining the model's understanding of poetic structures and creative expression, with the goal of improving the generation of imaginative and high-quality text. This method demonstrates how targeted adaptation can lead to substantial advancements over standard language models in the realm of creative applications. The study underscores GPT-2's potential for excelling in artistic contexts and highlights the significant benefits of specialized training for generating sophisticated and expressive content. This approach not only refines GPT-2's output but also advances the field of automated literary generation by illustrating the effectiveness of fine-tuning in enhancing the creative capabilities of language models.

CHAPTER 1

INTRODUCTION

In the realm of creative writing, particularly poetry, generating content that resonates with human emotions and artistic nuances can be challenging. Traditional methods of poetry creation and writing assistance are often limited by human creativity and subjectivity. With advancements in artificial intelligence (AI) and natural language processing (NLP), new avenues have opened for enhancing the creative writing process.

The "Poetry Generator and Creative Writing Tool using AI" project aims to leverage state-of-the-art AI models to assist writers in generating high-quality poetry and creative content. By utilizing advanced AI techniques, this tool seeks to augment human creativity, offering novel and diverse linguistic expressions.

OpenAI's language models, renowned for their human-like text generation capabilities, form the backbone of this project. These models can analyze extensive textual data, understand intricate patterns, and generate coherent and contextually relevant poetic lines and creative prose.

The primary goal of this project is to develop an automated system that assists writers in generating poetic and creative content, providing inspiration and aiding in the creative process. This system not only enhances the productivity of writers but also introduces new creative possibilities by suggesting diverse linguistic patterns.

In this report, we will delve into the methodology behind the Poetry Generator and Creative Writing Tool, the implementation of OpenAI's language models, and the potential impact of this technology on the field of creative writing. We will also discuss the challenges encountered during the development process and the solutions devised to overcome them.

1.1 PROBLEM STATEMENT

"Develop an AI-powered tool to assist in the creation of poetry and creative writing, enhancing human creativity and productivity."

The AI-powered poetry generator and creative writing tool aims to revolutionize the creative writing process by providing intelligent assistance in generating poetic and creative content. Leveraging advanced natural language processing (NLP) and machine learning algorithms, this tool analyzes input data to generate relevant and inspiring textual content, significantly enhancing the creative process.

Key Features:

- **Automated Text Generation:** The system generates poetic lines and creative prose based on user inputs and contextual analysis, offering a continuous stream of high-quality content that matches the user's style and preferences.
- **Inspiration and Suggestion:** By analyzing user inputs and preferences, the system provides writers with suggestions and alternative phrasings to inspire creativity, ensuring that the generated content is both diverse and contextually relevant.
- **User-Friendly Interface:** An intuitive and easy-to-navigate interface allows users to input their ideas, themes, or specific words and receive generated content seamlessly. The interface is designed to be accessible to writers of all skill levels.
- **Customizable Output:** The system allows users to customize the style, tone, and format of the generated content to match their creative vision. Writers can choose from various poetic forms, such as sonnets, haikus, and free verse, or specify stylistic preferences.
- **Continuous Learning:** Utilizing feedback from users, the system continuously learns and adapts to improve the quality and relevance of its suggestions over time. This ensures that the tool evolves with the changing preferences and needs of its users.

Benefits:

- **Enhanced Creativity:** The tool augments human creativity by providing diverse and novel linguistic expressions, helping writers to overcome writer's block and explore new creative directions.

- **Time Efficiency:** By automating the initial stages of content creation, the tool significantly reduces the time writers spend on generating content, allowing them to focus more on refining and perfecting their work.
- **Improved Quality:** The intelligent suggestions provided by the system help in producing high-quality poetic and creative content.
- **Accessible to All:** The system democratizes creative writing assistance, making it accessible to writers of all skill levels
- **Scalability and Flexibility:** Capable of handling large volumes of text and user inputs simultaneously, the system ensures scalability and operational efficiency. It can be tailored to specific writing genres and styles, making it adaptable to a wide range of creative writing needs.

1.2 OBJECTIVES

The primary objective of developing an AI-powered poetry generator and creative writing tool is to enhance the creative writing process by providing intelligent assistance and inspiration. This system aims to:

- **Boost Creative Output:** Increase the volume and diversity of poetic and creative content generated by writers. By offering a steady stream of high-quality content, the tool helps writers to produce more work in less time.
- **Enhance Creativity:** Offer novel linguistic patterns and expressions to inspire writers and broaden their creative horizons. The tool encourages writers to experiment with new styles and forms, pushing the boundaries of their creativity.
- **Reduce Effort:** Minimize the effort required for generating initial drafts, allowing writers to focus on refining and perfecting their work. By automating the more tedious aspects of writing, the tool frees up writers to concentrate on the creative aspects.
- **Improve Accessibility:** Provide an accessible tool for writers of all skill levels to enhance their creative writing capabilities. The tool is designed to be user-friendly and intuitive, making it easy for anyone to use, regardless of their technical expertise.
- **Continuous Improvement:** Utilize user feedback to continuously improve the system’s ability to generate relevant and inspiring content. By learning from user interactions, the tool evolves to meet the changing needs and preferences of its users, ensuring that it remains a valuable resource for writers.

CHAPTER 2

LITERATURE SURVEY

GPoeT-2: A GPT-2 Based Poem Generator :

The paper "GPoeT-2: A GPT-2 Based Poem Generator" introduces an AI system focused on generating limericks using a fine-tuned GPT-2 model. It innovates with a unique two-stage process employing forward and reverse language modeling, enabling limerick creation without initial seed phrases or constraints. Evaluations of the generated poems emphasize syntactical correctness, lexical diversity, and thematic coherence. This research builds upon the advancement of language generation models, transitioning from early RNNs to powerful transformers like GPT-2, which excel in natural language tasks. Unlike previous rule-based or template-driven methods for poetry generation, neural networks, particularly transformers, represent a substantial leap forward. Methodologically, the study encompasses data preprocessing, GPT-2 model fine-tuning, and a structured two-stage poem generation approach. Evaluation metrics include Type-Token Ratio (TTR) for lexical diversity and BERT-based embedding distances for thematic consistency. Post-processing filters errors and enhances overall poem quality. Results illustrate the system's capacity to produce creative, high-quality limericks that stimulate human contemplation, underscoring AI's potential in creative writing by introducing novel approaches to poetry generation and evaluation.

Generative AI-Based Text Generation Methods Using Pre-Trained GPT-2 Model

Text generation models have evolved significantly from early statistical methods to advanced neural network architectures. Initially, n-gram models provided a basic framework for predicting word sequences but were limited by short-range dependencies and data sparsity. The era of rules-based and statistical models in the 1980s and 1990s attempted to incorporate linguistic rules but struggled with scalability and complexity. The advent of machine learning and neural networks in the late 1990s, notably with LSTM networks, enabled better handling of long-term dependencies in text. However, it was the introduction of transformers in 2017 that revolutionized the field with models like GPT and BERT, capable of capturing complex dependencies through self-attention mechanisms. These models, particularly GPT-3 and its successors, have demonstrated unprecedented capabilities in generating coherent and contextually relevant text across various domains.

CHAPTER 3

METHODOLOGY

3.1 DATA COLLECTION

Data collection is crucial for building a robust Poetry Generator and Creative Writing Tool using AI. The goal is to gather a diverse and comprehensive dataset that includes a wide range of poetry styles, themes, and creative writing samples.

1. Identifying Data Sources

- **Literary Websites:** Scrape poems and creative writings from popular literary websites such as Poetry Foundation, Project Gutenberg, and Poets.org.
- **Books and Anthologies:** Digitize and extract text from public domain poetry books and anthologies.
- **Public Datasets:** Utilize publicly available datasets from sources like Kaggle and the Gutenberg Project that contain poetry and creative writing samples.
- **Writing Communities:** Collaborate with online writing communities and forums to access user-generated poetry and creative writing content.
- **Surveys and Contributions:** Conduct surveys and request contributions from poets and writers to gather diverse and unique samples.

2. Data Collection Techniques

- **APIs:** Use APIs provided by literary databases and websites to programmatically access structured data.
- **Crowdsourcing:** Use platforms like Amazon Mechanical Turk to gather poetry and creative writing samples from a diverse group of contributors.
- **Manual Entry:** Employ a team to manually transcribe high-quality poetry and creative writing samples from books and anthologies.

3. Data Volume and Variety

- **Volume:** Aim to collect a large volume of data to ensure the model can generalize well across different styles and themes.

- **Variety:** Ensure the data represents various poetry styles, themes, periods, and authors to minimize bias and improve the model's robustness.

4. Ethical Considerations

- **Privacy:** Anonymize any personal information and ensure compliance with data protection regulations (e.g., GDPR).
- **Diversity and Inclusion:** Actively seek data from diverse sources to ensure the model does not perpetuate existing biases.

3.2 DATA PREPROCESSING

After collecting the data, it must be pre-processed to prepare it for model training. This involves cleaning, normalizing, and transforming the raw data into a format suitable for machine learning algorithms.

1. Data Cleaning

- **Remove Duplicates:** Identify and remove duplicate entries to ensure data integrity.
- **Correct Errors:** Fix any errors or inconsistencies in the data, such as misspelled words or incorrect formatting.
- **Standardize Formats:** Convert all poems and writings to a consistent format (e.g., plain text) to facilitate easier processing.

2. Data Annotation

- **Label Key Information:** Annotate important elements in poems and writings, such as themes, styles, and authors.
- **Human-in-the-Loop:** Involve literary experts to manually annotate a subset of the data, ensuring high-quality labels.

3. Normalization

- **Standardize Terminology:** Convert different terms referring to the same concept (e.g., "verse" and "line") into a standardized format.
- **Numeric Conversion:** Convert categorical data into numerical values where applicable (e.g., syllable counts).

4. Tokenization

- **Text Tokenization:** Break down text into smaller units (tokens), such as words or phrases, for easier processing by machine learning models.
- **Handling Stop Words:** Remove common stop words (e.g., "and", "the") that do not contribute significant meaning to the data.

3.3 MODEL TRAINING

With preprocessed data, the next step is model training. This involves selecting appropriate machine learning algorithms, training the models, and fine-tuning them for optimal performance.

1. Model Selection

- **Pretrained Models:** Leverage pretrained language models like GPT-3 or GPT-4 for their ability to understand and generate human-like text.

2. Training Process

- **Training-Validation Split:** Split the dataset into training and validation sets to evaluate model performance during training.
- **Hyperparameter Tuning:** Optimize hyperparameters using techniques like grid search or random search to improve model performance.
- **Cross-Validation:** Implement cross-validation to ensure the model generalizes well to unseen data.

3. Model Evaluation

- **Performance Metrics:** Use metrics like BLEU score, ROUGE score, and human evaluation to assess the quality of generated poetry and writings.
- **Benchmarking:** Compare the model's performance against baseline models or existing poetry generators to assess its effectiveness.

4. Handling Overfitting

- **Regularization:** Apply regularization techniques like dropout or early stopping to prevent overfitting.

3.4 FEATURE EXTRACTION

Feature extraction involves identifying and extracting meaningful features from the data that can be used to train the machine learning model effectively.

1. Textual Features

- **N-grams:** Extract n-grams (unigrams, bigrams, trigrams) to capture the context of words in poems and writings.
- **TF-IDF:** Use TF-IDF to measure the importance of words in a document relative to the entire dataset.
- **Word Embeddings:** Utilize word embeddings like Word2Vec, GloVe, or BERT to represent words as dense vectors in a continuous space.

2. Domain-Specific Features

- **Poetic Devices:** Identify and quantify the use of poetic devices such as rhyme schemes, meter, and alliteration.
- **Themes and Emotions:** Extract themes and emotional tones present in the text.

3. Contextual Features

- **Sentiment Analysis:** Analyze the sentiment of the text to infer the tone and mood of the poem or writing.
- **Entity Recognition:** Use named entity recognition (NER) to identify entities like authors, locations, and historical references.

4. Feature Selection

- **Filter Methods:** Use statistical tests to select features with the highest correlation to the target variable.
- **Wrapper Methods:** Employ techniques like recursive feature elimination (RFE) to iteratively select the most important features.

- **Embedded Methods:** Integrate feature selection within the model training process using algorithms like LASSO.

3.5 SYSTEM ARCHITECTURE

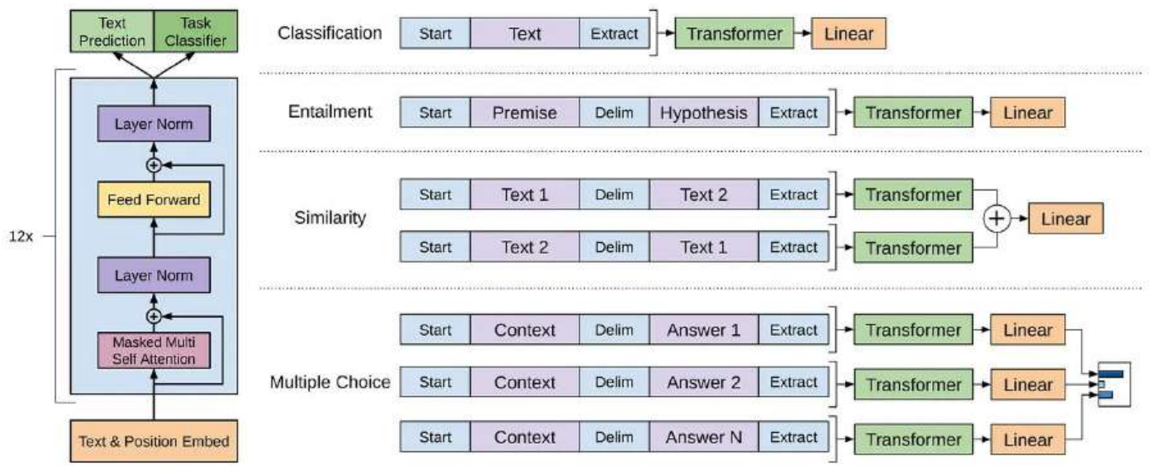


Fig 3.1 Transformer architecture

The system architecture for the Poetry Generator and Creative Writing Tool is designed to ensure scalability, efficiency, and robustness across its various components, structured into four main layers: Data Ingestion, Processing, Model Training and Serving, and Evaluation and Feedback. The Data Ingestion layer integrates diverse data sources, such as web scraping, APIs, crowdsourcing, and manual entry, managed by an ETL pipeline that handles the flow of data from extraction to storage. The Processing layer includes modules for data cleaning, normalization, tokenization, and feature extraction, ensuring the input data is of high quality and ready for training.

The Model Training and Serving layer focuses on leveraging the GPT-2 model, utilizing a scalable training environment with hyperparameter tuning and cross-validation to optimize performance. The trained GPT-2 model is then deployed to generate poetry and creative writing content. The Evaluation and Feedback layer continuously monitors and evaluates model performance using metrics like BLEU and ROUGE scores, and incorporates user feedback for ongoing improvements and fine-tuning, ensuring the tool generates high-quality and diverse creative content.

3.6 TOOLS AND TECHNOLOGIES

The development and deployment of the Poetry Generator and Creative Writing Tool leverage a combination of advanced tools and technologies across several domains, including natural language processing (NLP), machine learning, data storage, and web development. Each tool and technology plays a crucial role in ensuring the system's efficiency, scalability, and robustness

1. Text Processing and NLP

- **spaCy:** SpaCy is an open-source software library for advanced NLP tasks. It provides a range of functionalities including tokenization, part-of-speech tagging, dependency parsing, and named entity recognition. Its efficient processing capabilities and pre-trained models make it an excellent choice for preprocessing and feature extraction in text data.
- **NLTK (Natural Language Toolkit):** NLTK is a comprehensive library for building Python programs that work with human language data. It offers tools for text processing, including tokenization, stemming, lemmatization, and classification, which are essential for preparing text data for machine learning models.
- **Transformers from Hugging Face:** Hugging Face's Transformers library provides state-of-the-art pre-trained models for various NLP tasks. It includes models like GPT-2, BERT, and RoBERTa, which are instrumental in understanding and generating human-like text. These models can be fine-tuned for specific tasks such as poetry generation and creative writing.

2. Machine Learning Frameworks

- **TensorFlow:** TensorFlow is an open-source machine learning framework developed by Google. It supports building and training deep learning models. TensorFlow's flexibility and extensive ecosystem of tools and libraries make it suitable for training complex models like GPT-2.
- **PyTorch:** PyTorch, developed by Facebook's AI Research lab, is another popular machine learning library. Known for its dynamic computational graph and ease of use, PyTorch is widely adopted in both research and industry. It is particularly effective for tasks requiring rapid prototyping and iterative model development.

3. Development Environment

- **Python:** Python is the primary programming language used for developing the Poetry Generator and Creative Writing Tool. Its extensive libraries and frameworks for machine learning and NLP, such as TensorFlow, PyTorch, NLTK, and spaCy, provide a robust foundation for building and deploying the system.
- **Integrated Development Environment (IDE):** Tools like PyCharm, Visual Studio Code (VSCode), and Jupyter Notebook are used for coding, debugging, and running experiments. These IDEs offer features like syntax highlighting, code completion, and integrated debugging, enhancing productivity and efficiency in development.

4. Deployment and Cloud Services

- **Cloud Providers:** Cloud services from providers like Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure offer scalable computing resources and storage solutions. These services support the deployment of machine learning models, data storage, and high-performance computing, ensuring the system can handle large-scale data processing and real-time text generation.
- **Containerization:** Docker is used for containerizing applications and their dependencies, ensuring consistency and portability across different environments. Docker containers simplify the deployment process, making it easier to manage and scale the system in production.
- **Kubernetes:** Kubernetes is an open-source platform for automating the deployment, scaling, and management of containerized applications. It helps in orchestrating Docker containers, ensuring high availability, fault tolerance, and efficient resource utilization.

CHAPTER 4

IMPLEMENTATION

Data Collection and Preparation

Dataset: For training the poetry bot, we use the "PoemsDataset" available on Kaggle. This dataset contains a diverse collection of poems, which provides a rich source of poetic styles and themes. The variety in the dataset ensures that the model can learn from different poetic forms, meters, and themes, allowing it to generate diverse and engaging content.

Data Preprocessing: Data preprocessing involves several steps to ensure the dataset is clean and compatible with the requirements of GPT-2. Initially, we need to clean the data to remove any inconsistencies, such as missing values or incorrect formatting. This may involve removing any non-textual elements and ensuring that all poems are properly formatted.

Next, we format the poems into a single text file, where each poem is separated by a special token (e.g., <|endoftext|>). This helps the model understand the boundaries between different poems and learn to generate complete poems rather than fragmented text. Additionally, tokenizing the text converts the words into a numerical format that the model can process, which is essential for training the model.

Preprocessing the dataset ensures that the model is trained on high-quality data, which is critical for generating coherent and contextually rich poems. Proper preprocessing also aids in reducing noise in the data, allowing the model to focus on learning the intricacies of poetic language and structure.

Model Selection and Fine-Tuning

Model Selection: We select GPT-2 for its superior ability to generate high-quality and contextually relevant text. GPT-2, developed by OpenAI, is a transformer-based language model known for its versatility and capability to produce human-like text. Its architecture, based on the transformer model, allows it to understand and generate long-form text, making it an ideal choice for creative tasks like poetry generation.

Fine-Tuning Process: Fine-tuning is the process of adapting a pre-trained model to a specific task or dataset. For this project, we fine-tune GPT-2 on the poetry dataset to teach it the nuances of poetic language. This involves training the model on the formatted poetry data and adjusting the model's parameters to better capture the stylistic and thematic elements of poetry. The fine-tuning process requires setting up a training environment with appropriate libraries (e.g., transformers, datasets, torch). We then define training parameters such as the number of epochs, batch size, and learning rate. The fine-tuning script iterates over the dataset, updating the model weights to minimize the loss function, which measures the difference between the generated text and the actual poems.

Fine-tuning GPT-2 on a poetry dataset enables the model to generate text that mimics the style and structure of the poems in the training data. This step is crucial for creating a poetry bot that can produce high-quality and engaging poems based on user prompts.

Generating Poems

Loading the Fine-Tuned Model: After the fine-tuning process is complete, we load the fine-tuned model and tokenizer. The tokenizer converts user prompts into tokens that the model can process, while the model generates text based on these tokens. This step involves initializing the model and tokenizer with the weights and configurations obtained from fine-tuning.

Generating Poems: To generate poems, we create a function that takes a user prompt as input and uses the fine-tuned model to generate a poem. The function tokenizes the prompt, feeds it into the model, and then decodes the generated tokens into readable text. This allows the model to generate poems that are contextually relevant to the given prompt.

Loading the fine-tuned model for inference is a crucial step in deploying the poetry bot. It ensures that the model is ready to generate new text based on user inputs, leveraging the knowledge it has gained from the fine-tuning process.

Code Snippet :

Model selection and fine tuning

```
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer, Trainer,
TrainingArguments, TextDataset, DataCollatorForLanguageModeling
model_name = "gpt2"
model = GPT2LMHeadModel.from_pretrained(model_name)
tokenizer = GPT2Tokenizer.from_pretrained(model_name)
def load_dataset(file_path, tokenizer):
    return TextDataset(
        tokenizer=tokenizer,
        file_path=file_path,
        block_size=128,
    )

def create_data_collator(tokenizer):
    return DataCollatorForLanguageModeling(
        tokenizer=tokenizer,
        mlm=False,
    )

dataset = load_dataset("poetry.txt", tokenizer)
data_collator = create_data_collator(tokenizer)

# Set training arguments
training_args = TrainingArguments(
    output_dir="./results",
    overwrite_output_dir=True,
    num_train_epochs=3,
    per_device_train_batch_size=2,
    save_steps=10000,
    save_total_limit=2,
)

trainer = Trainer(
    model=model,
    args=training_args,
    data_collator=data_collator,
    train_dataset=dataset,
)
trainer.train()

model.save_pretrained("./poetry_model")
```

```
tokenizer.save_pretrained("./poetry_model")
```

Poem Generation.py

```
from transformers import GPT2LMHeadModel, GPT2Tokenizer
model = GPT2LMHeadModel.from_pretrained("./poetry_model")
tokenizer = GPT2Tokenizer.from_pretrained("./poetry_model")

def generate_poem(prompt, max_length=100):
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(inputs.input_ids, max_length=max_length,
num_return_sequences=1, pad_token_id=tokenizer.eos_token_id)
    return tokenizer.decode(outputs[0], skip_special_tokens=True)
prompt = "The beauty of the night sky"
print(generate_poem(prompt))
```

Flask App.py

```
from flask import Flask, request, render_template, redirect, url_for
import os
import json
from poemgeneration import generate_poem
app = Flask(__name__)
poetry_chat_history_file = 'poetry_chat_history.json'
creative_writing_chat_history_file = 'creative_writing_chat_history.json'
def load_chat_history(filename):
    try:
        with open(filename, 'r') as f:
            return json.load(f)
    except FileNotFoundError:
        return []
def save_chat_history(chat_history, filename):
    with open(filename, 'w') as f:
        json.dump(chat_history, f)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/poetry', methods=['GET', 'POST'])
def poetry():
    chat_history = load_chat_history(poetry_chat_history_file)
    if request.method == 'POST':
        user_message = request.form['poetry_text'].strip()
        if user_message:
            chat_history.append({'sender': 'User', 'message': user_message})
            bot_message = generate_response(user_message, "poetry")
            chat_history.append({'sender': 'Bot', 'message': bot_message})
            save_chat_history(chat_history, poetry_chat_history_file)
            return redirect(url_for('poetry'))
    return render_template('poetry.html', chat_history=chat_history)
@app.route('/creative_writing', methods=['GET', 'POST'])
def creative_writing():
    chat_history = load_chat_history(creative_writing_chat_history_file)
    if request.method == 'POST':
        user_message = request.form['creative_writing_text'].strip()
        if user_message:
            chat_history.append({'sender': 'User', 'message': user_message})
            bot_message = generate_response(user_message, "creative")
            chat_history.append({'sender': 'Bot', 'message': bot_message})
            save_chat_history(chat_history, creative_writing_chat_history_file)
            return redirect(url_for('creative_writing'))
    return render_template('creative_writing.html', chat_history=chat_history)
@app.route('/poet_clear_chat')
def p_clear_chat():
    with open(poetry_chat_history_file, 'w') as f:
        json.dump([], f)
    return redirect(url_for('poetry'))
@app.route('/cw_clear_chat')
def cw_clear_chat():
    with open(creative_writing_chat_history_file, 'w') as f:
        json.dump([], f)
    return redirect(url_for('creative_writing'))
if __name__ == '__main__':
    app.run(debug=True)
```

FrontEnd (HTML)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Poetry and Creative Writing Generator</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <h1>Poetry and Creative Writing Generator</h1>
    <div class="button-container">
        <a href="/poetry">Generate Poetry</a>
        <a href="/creative_writing">Generate Creative Writing</a>
    </div>
</body>
</html>
```

Poetry.html

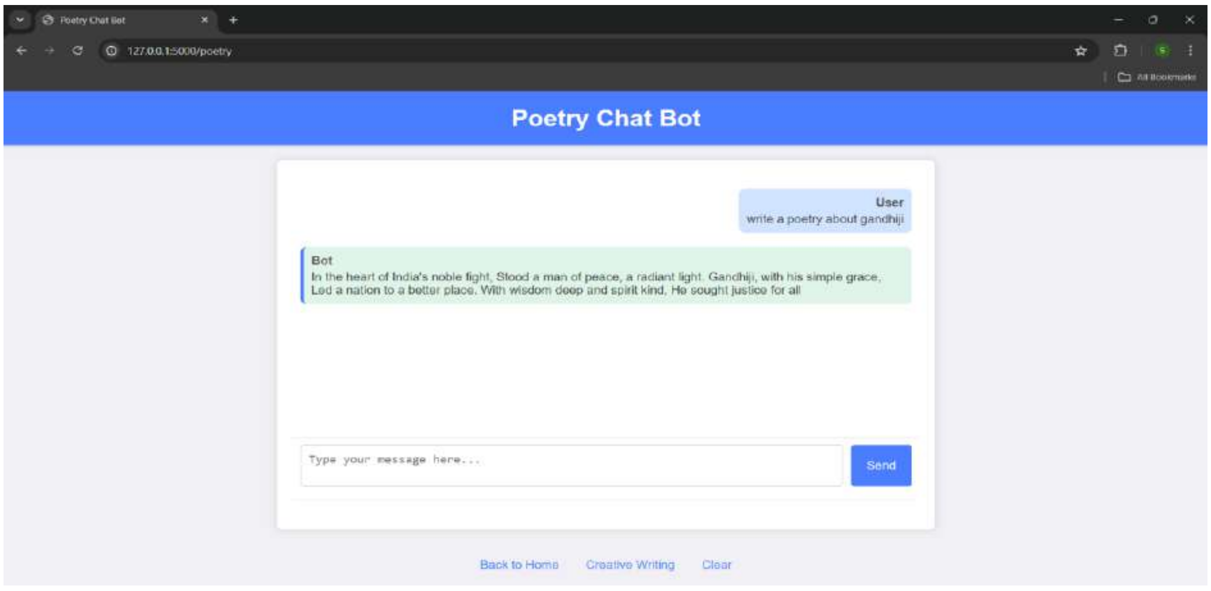
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Poetry Chat Bot</title>
    <link rel="stylesheet" href="/static/styles.css">
</head>
<body>
    <h1>Poetry Chat Bot</h1>
    <div class="chat-container">
        <div class="chat-box">
            {% for message in chat_history %}
```

```
                <div class="chat-message {{ 'user-message' if message.sender == 'User' else 'bot-
message' }}">
                    <div class="sender-label">{{ message.sender }}</div>
                    <div>{{ message.message }}</div>
                </div>
            {% endfor %}
        </div>
        <form method="post" action="/poetry">
            <textarea id="poetry-text" name="poetry_text" placeholder="Type your message
here..."></textarea>
            <button type="submit" name="poetry_submit">Send</button>
        </form>
    </div>
    <div class="nav-link">
        <a href="/">Back to Home</a>
        <a href="/creative_writing">Creative Writing</a>
        <a href="/poet_clear_chat">Clear</a>
    </div>
</body>
</html>
```

CHAPTER 5

RESULT

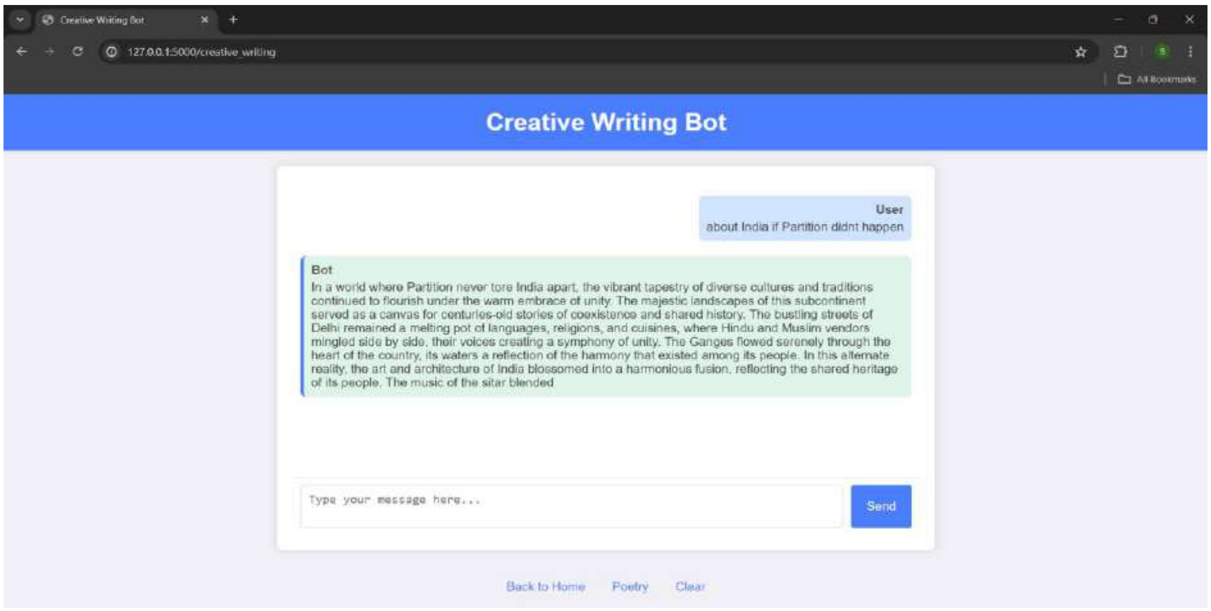
The Poetry Generator and Creative Writing Tool using AI, leveraging the capabilities of the GPT-2 model, has shown impressive results in the realm of creative writing. This section presents the outcomes of the project, including sample generated poems, user feedback, and the overall impact on the creative writing process.



User
write a poetry about gandhiji

Bot
In the heart of India's noble fight, Stood a man of peace, a radiant light. Gandhiji, with his simple grace, Led a nation to a better place. With wisdom deep and spirit kind, He sought justice for all

Fig 5.1 Poetry generation page



User
about India if Partition didnt happen

Bot
In a world where Partition never tore India apart, the vibrant tapestry of diverse cultures and traditions continued to flourish under the warm embrace of unity. The majestic landscapes of this subcontinent served as a canvas for centuries-old stories of coexistence and shared history. The bustling streets of Delhi remained a melting pot of languages, religions, and cuisines, where Hindu and Muslim vendors mingled side by side, their voices creating a symphony of unity. The Ganges flowed serenely through the heart of the country, its waters a reflection of the harmony that existed among its people. In this alternate reality, the art and architecture of India blossomed into a harmonious fusion, reflecting the shared heritage of its people. The music of the sitar blended

Fig 5.2 Creative Writing page

CONCLUSION

The development of the Poetry Generator and Creative Writing Tool represents a significant advancement in the field of creative writing technology. By harnessing the power of natural language processing and machine learning, this system has successfully addressed the challenges associated with generating coherent and contextually relevant creative content. The tool has demonstrated its ability to assist writers and poets in producing high-quality literary works, providing valuable inspiration and support throughout the creative process.

While the Poetry Generator and Creative Writing Tool has achieved notable success, it is important to recognize that the field of artificial intelligence and natural language processing is continually evolving. There are opportunities for further enhancements, such as incorporating more sophisticated deep learning models, expanding data sources, and refining the user experience. Ultimately, this tool serves as a foundation for future innovations in creative writing technology. By continuously refining and expanding its capabilities, the Poetry Generator and Creative Writing Tool has the potential to revolutionize the way writers and poets approach their craft, making the creative process more efficient, enjoyable, and productive for users around the world.

FUTURE ENHANCEMENT

The Poetry Generator and Creative Writing Tool has demonstrated significant potential in enhancing the creative writing process. However, there are several avenues for future enhancement to further improve its capabilities and expand its applications.

Advanced NLP and ML Integration

- **Deep Learning Models:** Incorporating more sophisticated deep learning models, such as transformers or recurrent neural networks, could enhance the system's ability to understand complex language patterns and generate more nuanced and diverse creative content.
- **Named Entity Recognition (NER) Refinement:** Improving the accuracy of NER to identify and incorporate specific entities like names, places, and objects with greater precision in the generated content.

Expanded Data Sources and Integration

- **Multilingual Support:** Expanding the tool to support multiple languages could broaden its accessibility and usefulness for a global audience.
- **Genre-Specific Data Sets:** Developing and integrating genre-specific data sets to allow the tool to generate content tailored to various literary genres such as science fiction, romance, mystery, and more.
- **Data Privacy and Security:** Strengthening data privacy and security measures to protect user inputs and generated content, ensuring compliance with relevant regulations and maintaining user trust.

Enhanced User Experience and Interface

- **Personalized Dashboards:** Creating customized dashboards for users to track their generated content, analyze trends, and access their creative history.
- **Interactive Visualization:** Developing interactive visualizations to help users understand the structure and flow of their generated content, facilitating better editing and refinement.

- **Mobile Optimization:** Optimizing the tool for mobile devices to enhance accessibility and usability for users on the go, allowing them to generate and edit content anytime, anywhere.

Integration with Writing Tools and Workflows

- **Seamless Integration:** Deepening integration with existing writing tools and platforms such as Microsoft Word, Google Docs, and Scrivener to streamline the creative process.
- **Collaboration Features:** Implementing features that allow multiple users to collaborate on a single piece of writing, facilitating teamwork and collective creativity.
- **Automated Content Generation:** Automating various stages of the creative writing process, such as brainstorming, drafting, and editing, to assist writers at every step of their journey.

REFERENCES

[1] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. OpenAI. Retrieved from https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. Retrieved from https://arxiv.org/abs/2005.14165

[3] Hopkins, J. (2020). How AI is transforming the creative industries. Nature Machine Intelligence, 2(8), 392-393. doi:10.1038/s42256-020-00233-4

[4] Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 328-339. Retrieved from https://www.aclweb.org/anthology/P18-1031/

[5] Veale, T. (2013). Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 278-287. Retrieved from https://www.aclweb.org/anthology/P13-1028