

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA" ,MACHHE, BELAGAVI-590018



**Project Report
on
Develop a Quality Assurance framework for Text Generation
Models**

Submitted in partial fulfillment of the requirements for the VIII semester

Bachelor of Engineering

in

Computer Science and Engineering

of

Visvesvaraya Technological University, Belagavi.

By

Nithin N (1CD20CS110)

Santhosh Adavala (1CD20CS140)

Shafath H Khan (1CD20CS144)

Syed Mohammed Maaz (1CD20CS164)

Under the Guidance of

Mr. Pushpanathan G

Assistant Professor

Dept. of CSE



**Department of Computer Science and Engineering
CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE - 560 036
2023-2024**

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

Certified that **Mr. Nithin N, Mr. Santhosh Adavala, Mr. Shafath H Khan and Mr. Syed Mohammed Maaz** bearing **1CD20CS110, 1CD20CS140, 1CD20CS144 and 1CD20CS164** respectively, are bonafide students of **Cambridge Institute of Technology**, have successfully completed the project entitled **“Develop a Quality Assurance framework for Text Generation Models”** in partial fulfillment of the requirements for VIII semester **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-2024. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Bachelor of Engineering degree.

Project Guide
Mr. Pushpanathan G
Dept. of CSE, CITech

Project Co-Ordinator
Dr. Sandeep Kumar
Dept. of CSE. CITech

Head of the Dept.
Dr. Shashikumar D.R
Dept. of CSE. CITech

Principal
Dr. G. Indumathi
CITech

External Viva
Name of the Examiners

1.

2.

Signature with date

DECLARATION

We, **Mr. Nithin N, Mr. Santhosh Adavala, Mr. Shafath H Khan and Mr. Syed Mohammed Maaz** bearing **1CD20CS110, 1CD20CS140, 1CD20CS144 and 1CD20CS164** respectively, are the students of VIII semester, Computer Science and Engineering, Cambridge Institute of Technology, hereby declare that the project entitled ***“Develop a Quality Assurance framework for Text Generation Models”*** has been carried out by us and submitted in partial fulfillment of the course requirements of VIII semester **Bachelor of Engineering in Computer Science and Engineering** as prescribed by **Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

We also declare that, to the best of our 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:	Name	USN	Signature
Place: Bangalore	Nithin N	1CD20CS110	
	Santhosh Adavala	1CD20CS140	
	Shafath H Khan	1CD20CS144	
	Syed Mohammed Maaz	1CD20CS164	

ACKNOWLEDGEMENT

We would like to place on record our 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. Indumathi G.**, Principal, CITech, Bangalore, for providing us the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express our sincere gratitude to **Dr. Shashikumar D.R.**, HOD, Dept. of Computer Science and Engineering, CITech, Bangalore, for his stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We wish to extend our thanks to **Dr. Sandeep Kumar**, Project Coordinator, Dept. of CSE, CITech, Bangalore, for his critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

We also wish to extend our thanks to **Mr. Pushpanathan G**, Assistant Professor, Dept. of CSE, CITech for his guidance and impressive technical suggestions to complete our project.

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

Nithin N (1CD20CS110)

Santhosh Adavala (1CD20CS140)

Shafath H Khan (1CD20CS144)

Syed Mohammed Maaz (1CD20CS164)

ABSTRACT

Text generation, a critical component of natural language processing, involves the creation of human-like text by artificial intelligence systems. Evaluating the performance of these models is paramount for assessing their quality, coherence, and relevance. This process often entails comparing generated text with reference or ground truth text using metrics such as accuracy, precision, recall, F1 score, and task-specific evaluation metrics. Additionally, subjective human evaluations may be employed to capture nuanced aspects of language quality. The abstract emphasizes the iterative nature of model evaluation, where insights gleaned from assessments inform fine-tuning and improvement strategies. Consideration is given to the dynamic landscape of evaluation metrics, including BLEU score for machine translation and ROUGE score for text summarization.

Large Language Models (LLMs), sophisticated neural network-based systems that have revolutionized natural language processing. LLMs, exemplified by models like GPT-3, possess the capability to understand and generate human-like text across diverse domains. This abstract explores the architecture and training methodologies that contribute to the robustness of LLMs. The versatile applications of LLMs in tasks such as language understanding, translation, summarization, and content generation are discussed. Emphasis is placed on the challenges, ethical considerations, and ongoing research in the development and deployment of LLMs. The abstract concludes by acknowledging the transformative impact of LLMs on natural language processing and their potential to shape the future of AI-powered language understanding and generation. Overall, the abstract underscores the multidimensional nature of text generation evaluation and its pivotal role in advancing the capabilities of language models.

CONTENTS

	Chapters	Page No.
Chapter 1	Introduction	1
	1.1 Background	1
	1.2 Motivations	2
	1.3 Applications	2
	1.4 Problem Statement	3
	1.5 Objectives	4
Chapter 2	Literature Survey	6
Chapter 3	System Analysis	11
	3.1 Hardware Requirements	11
	3.2 Software Requirements	11
Chapter 4	Design	12
	4.1 Purpose	12
	4.2 System Architecture	12
	4.2.1 Module Division	13
	4.3 Class Diagram	14
	4.3.1 Rouge_Metrics Class	14
	4.3.2 Main Program	16
	4.3.3 Main Program Loop	16
	4.3.4 Relationships and Composition	16
	4.3.5 Modularity and Maintainability	17
	4.4 Use Class Diagram	17
	4.5 Sequence Diagram	20
Chapter 5	Implementation	24
	5.1 Introduction	24
	5.2 Code	25
Chapter 6	Results and Discussions	32
	Conclusion	36
	References	37

List of Figures

Figure No.	Figure Name	Page No.
4.1	System Architecture	13
4.2	Class Diagram	15
4.3	Use Case Diagram	18
4.4	Sequence Diagram	21
6.1	Question Answering Result	33
6.2	Summarization Result	34
6.3	Translation Result	34

CHAPTER 1

INTRODUCTION

1.1 Background

Text generation, a pivotal aspect of modern natural language processing (NLP), plays a crucial role in diverse applications and projects. This process involves the development of sophisticated models capable of understanding and generating human-like text, thus enabling the automation of various content-related tasks. These tasks include, but are not limited to, content creation, translation, summarization, and dialogue systems. The ability of these models to produce coherent, contextually relevant text significantly enhances the efficiency of data interpretation and communication.

In the context of our project report, text generation serves as a powerful tool for producing high-quality, human-like text. This capability is essential for creating reports that are not only informative but also engaging and easy to understand. By automating content creation, we can save time and resources, allowing team members to focus on more strategic tasks. Additionally, automated translation services can break down language barriers, making the report accessible to a broader audience. Summarization tools help in distilling large volumes of information into concise, relevant summaries, which are crucial for quick decision-making and understanding. One of the most significant advancements in text generation is the development of text-to-text generation models, such as those based on transformer architectures like GPT-4. These models have shown remarkable proficiency in generating text that is not only grammatically correct but also contextually appropriate. By leveraging these advanced models in our project, we aim to produce text that meets high standards of quality and relevance. This ensures that the generated content aligns with our project objectives and adheres to industry standards, ultimately enhancing the credibility and impact of our reports.

The implementation of text generation within our project report framework also highlights our commitment to staying at the forefront of technological advancements in NLP. Incorporating state-of-the-art text generation models demonstrates our dedication to innovation and continuous improvement. This proactive approach ensures that we are utilizing the best available tools to achieve our goals, providing our stakeholders with superior quality and reliability.

Moreover, text generation facilitates information dissemination by making complex data more accessible and understandable. For instance, when dealing with large datasets or intricate

analyses, text generation can help in presenting the findings in a clear and coherent manner. This is particularly beneficial for stakeholders who may not have the technical expertise to interpret raw data. By converting complex information into natural language, we enhance the comprehensibility of our reports, ensuring that the insights are easily digestible and actionable. In conclusion, text generation is a crucial component of our project report, enabling the automation of content creation, translation, summarization, and more. By leveraging advanced text-to-text generation models, we enhance the quality and relevance of our generated text, ensuring it aligns with our project objectives and meets industry standards. This implementation not only facilitates efficient information dissemination but also demonstrates our commitment to technological innovation in natural language processing. As we continue to integrate these advanced capabilities, we remain at the forefront of NLP advancements, providing high-quality, reliable, and impactful reports.

1.1 Motivation

- Continuous updates and adaptation to evolving language patterns for sustained relevance.
- Adherence to ethical guidelines, privacy regulations, and responsible AI norms for legal and ethical operation.
- Transparent communication, regular audits, and ethical considerations to build user confidence in model reliability.
- Ongoing improvements in accuracy, efficiency, and coherence through research, development, and user feedback.
- User-friendly interfaces, compatibility with existing systems, and community support to encourage widespread integration and use.
- With the rapid advancements in AI and machine learning, there is a tremendous potential to obtain summaries of various podcasts which lasts for long periods in a short and efficient manner. Moreover, it will also help the popularity of the various podcasts which are good but too lengthy for the average users to sit and listen to for extended periods of time.

1.2 Applications

1. Improving Model Development: Iterative refinement of the text-to-text model through continuous training, fine-tuning, and incorporating the latest advancements in natural language processing.

2. Ensuring Quality in NLP Applications: Rigorous testing and validation processes to guarantee accurate and reliable performance of the text-to-text model in various natural language processing applications.
3. Fraud Detection: Identifying and preventing the spread of fake news and malicious content: Utilizing the text-to-text model to analyze and detect patterns associated with misinformation, fake news, and malicious content, contributing to robust fraud detection mechanisms.
4. Facilitating Research and Development: Providing a versatile tool for researchers to experiment, innovate, and advance the field of natural language processing by leveraging the capabilities of the text-to-text generation mode.
5. Content Generation for Marketing and Advertising: Using text generation models to automate the creation of marketing copy, ad slogans, and product descriptions. Ensuring the generated content aligns with brand guidelines and effectively communicates key messages to target audiences.
6. Personalized Customer Support and Chatbots: Deploying text generation models to create personalized responses in customer support chatbots. Ensuring that the chatbot responses are accurate, relevant, and maintain a conversational tone to enhance customer satisfaction.
7. Legal Document Drafting and Summarization: Leveraging text generation models to draft legal documents such as contracts, agreements, and briefs. Summarizing lengthy legal documents for quick review by lawyers and legal professionals, ensuring accuracy and completeness.
8. Academic Research and Writing Assistance: Assisting researchers and scholars in generating summaries, literature reviews, and research papers. Ensuring that the generated content maintains academic integrity and meets the standards of scholarly writing.

1.4 Problem Statement

1. Comprehensive Metric Integration: Provide an exhaustive literature survey on existing text generation metrics such as F1 Score, HumanEvals, GSM8K, GLEU, and others, and integrate the most relevant and widely accepted metrics into the fixed framework for comprehensive model evaluation.
2. Unified Metric Interpretation: Establish a standardized interpretation and scoring system for the integrated metrics within the fixed framework, ensuring a consistent and

easily

3. Understandable method for assessing the quality of text generated by models.
4. Benchmarked Performance Standards: Set benchmark performance standards based on extensive analysis of existing literature, allowing for clear comparison and benchmarking of text generation models against established norms and expectations.
5. Task-Specific Customization: Tailor the fixed framework to accommodate task-specific nuances by considering variations in text generation goals, enabling the framework to be adaptable across a range of applications, from summarization to dialogue systems.
6. Documentation and Accessibility: Provide thorough documentation of the fixed framework, including guidelines on metric usage, interpretation, and potential pitfalls. Ensure accessibility for researchers and practitioners, fostering widespread adoption and consistent application in the field of text generation quality assurance.

1.5 Objectives

In the pursuit of advancing natural language processing capabilities, the development of robust quality assurance frameworks for text generation models stands as a fundamental imperative. These frameworks serve as guiding pillars, delineating clear objectives and methodologies to ensure the reliability, accuracy, and ethical soundness of text generation systems. By establishing concrete objectives, organizations can systematically address key challenges, ranging from model performance evaluation to ethical considerations, thereby fostering trust and confidence in the outputs generated by these models. In this context, defining precise objectives becomes paramount, laying the groundwork for the systematic development and implementation of quality assurance measures that uphold the highest standards in text generation technology.

- Utilize established metrics such as BLEU, ROUGE, METEOR, and others to evaluate the model's performance in text generation, summarization, and translation.
- Develop a framework for real time evaluation of generated text.
- Make model stronger and better at understanding different things by training it on various datasets including different Topics and Languages.
- Achieve high Linguistic and factual accuracy in generated texts.
- Make short and clear Summaries that keep the important information.
- Create translations that accurately convey the intended meaning.
- Develop methods to assess and improve the coherence and flow of generated texts,

ensuring consistency in style and language usage.

- Implement techniques to identify and address biases present in training data and generated outputs, promoting fairness and inclusivity.
- Develop strategies to improve the model's ability to handle inputs outside its training distribution, reducing errors and improving generalization.
- Implement techniques to optimize model architecture and training processes, reducing computational costs and resource requirements.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A literature review is the identification and examining of the existing research work in the chosen field to gain valuable information. Literature review was performed to understand the existing learning algorithms and to choose the suitable supervised and unsupervised method for image classification. As the study was made to compare supervised and unsupervised algorithms, the literature review was performed to identify the most effective algorithm of each kind. The algorithms identified were further used in experimentation.

2.2 Related Work

2.2.1 “Stylized Data-to-Text Generation: A Case Study in the E-Commerce Domain: Liqiang Jing and Xuming Lin” [1], Existing data-to-text generation efforts mainly focus on generating a coherent text from non-linguistic input data, such as tables and attribute-value pairs, but overlook that different application scenarios may require texts of different styles. Inspired by this, we define a new task, namely stylized data-to-text generation, whose aim is to generate coherent text for the given non-linguistic data according to a specific style. This task is non-trivial, due to three challenges: the logic of the generated text, unstructured style reference, and biased training samples. To address these challenges, we propose a novel stylized data-to-text generation model, named StyleD2T, comprising three components: logic planning-enhanced data embedding, mask-based style embedding, and unbiased stylized text generation. In the first component, we introduce a graph-guided logic planner for attribute organization to ensure the logic of generated text. In the second component, we devise feature-level mask-based style embedding to extract the essential style signal from the given unstructured style reference. Extensive experiments on a newly collected dataset from Taobao1 have been conducted, and the results show the superiority of our model over existing methods.

2.2.2 “Context Aware Automatic Subjective and Objective Question Generation using Fast Text to Text Transfer Learning: Arpit Agrawal and Pragya Shukla” [2], Online learning has gained a tremendous popularity in the last decade due to the facility to learn anytime, anything, anywhere from the ocean of web resources available. Especially the lockdown all over the world due to the Covid-19 pandemic has brought an enormous attention towards the online learning for value addition and skills development not only for the school/college students, but also to the working professionals.

This massive growth in online learning has made the task of assessment very tedious and demands training, experience and resources. Automatic Question generation (AQG) techniques have been introduced to resolve this problem by deriving a question bank from the text documents. However, the performance of conventional AQG techniques is subject to the availability of large labelled training dataset. The requirement of deep linguistic knowledge for the generation of heuristic and hand-crafted rules to transform declarative sentence into interrogative sentence makes the problem further complicated. Fast T5 library with beam-search decoding algorithm has been used here to reduce the model size and increase the speed of the model through quantization of the whole model by Open Neural Network Exchange (ONNX) framework. The keywords extraction in the proposed framework is performed using the Multipartite graphs to enhance the context awareness. The qualitative and quantitative performance of the proposed AQG model is evaluated through a comprehensive experimental analysis over the publicly available Squad dataset.

2.2.3 “Probabilistic Approaches for Modelling Text Structure and Their Application to Text-to- Text Generation: Regina Barzilay” [3], Since the early days of generation research, it has been acknowledged that modelling the global structure of a document is crucial for producing coherent, readable output. However, traditional knowledge intensive approaches have been of limited utility in addressing this problem since they cannot be effectively scaled to operate in domain independent, large-scale applications. Due to this difficulty, existing text-to-text generation systems rarely rely on such structural information when producing an output text. Consequently, texts generated by these methods do not match the quality of those written by humans – they are often fraught with severe coherence violations and disfluencies. In this chapter, I will present probabilistic models of document structure that can be effectively learned from raw document collections. This feature distinguishes these new models from traditional knowledge intensive approaches used in symbolic concept-to-text generation. Our results demonstrate that these probabilistic models can be directly applied to content organization, and suggest that these models can prove useful in an even broader range of text-to-text applications than we have considered here.

2.2.4 “BLEURT: Learning Robust Metrics for Text Generation: Thibault Sellam, Dipanjan Das and Ankur P. Parikh” [4], Text generation has made significant advances in the last few years. Yet, evaluation metrics have lagged behind, as the most popular choices (e.g., BLEU and ROUGE) may correlate poorly with human judgments. We propose BLEURT, a learned evaluation metric based on BERT that can model human judgments with a few

thousand possibly biased training examples. A key aspect of our approach is a novel pre-training scheme that uses millions of synthetic examples to help the model generalize. BLEURT provides state-of-the-art results on the last three years of the WMT Metrics shared task and the WebNLG Competition dataset. In contrast to a vanilla BERT-based approach, it yields superior results even when the training data is scarce and out-of-distribution.

2.2.5 “PersuAIDE ! An Adaptive Persuasive Text Generation System for Fashion Domain: Abhijit Mishra” [5], Persuasiveness is a creative art which aims at inducing certain set of beliefs in the target audience. In an e-commerce setting, for a newly launched product, persuasive descriptions are often composed to motivate an online buyer towards a successful purchase. Such descriptions can be catchy taglines, product-summaries, style-tips etc.. In this paper, we present PersuAIDE! - a persuasive system based on linguistic creativity to generate various forms of persuasive sentences from the input product specification. To demonstrate the effectiveness of the proposed system, we have applied the technology to fashion domain, where, for a given fashion product like "red collar shirt" we were able to generate descriptive sentences that not only explain the item but also garner positive attention, making it persuasive. We evaluate the system on a large fashion corpus collected from different sources using (a) automatic text generation metrics used for Machine Translation and Automatic Summarization evaluation and Readability measurement, and (b) human judgment scores evaluating the persuasiveness and fluency of the generated text. Experimental results and qualitative analysis show that an unsupervised system like ours can produce more creative and better constructed persuasive output than supervised generative counterparts based on neural sequence-to-sequence models and statistical machine translation.

2.2.6 “BARTSCORE: Evaluating Generated Text as Text Generation: Weizhe Yuan” [6], A wide variety of NLP applications, such as machine translation, summarization, and dialog, involve text generation. One major challenge for these applications is how to evaluate whether such generated texts are actually fluent, accurate, or effective. In this work, we conceptualize the evaluation of text as a text generation problem, modeled using pre-trained sequence-to-sequence models. We operationalize this idea using BART [32], an encoder-decoder based pre-trained model, and propose a metric BARTSCORE with a number of variants that can be flexibly applied in an unsupervised fashion to evaluation of text from different perspectives (e.g. informativeness, fluency, or factuality). BARTSCORE is conceptually simple and empirically effective. It can outperform existing top-scoring metrics in 16 of 22 test settings, covering evaluation of 16 datasets (e.g., machine translation, text summarization) and 7 different

perspectives (e.g., informativeness, factuality). Code to calculate BARTScore is available at <https://github.com/neulab/BARTScore>, and we have released an interactive leaderboard for meta-evaluation at <http://explainaboard.nlpedia.ai/leaderboard/task-meval/models>. We operationalize this idea using BART [32], an encoder-decoder based pre-trained model, and propose a metric BARTSCORE with a number of variants that can be flexibly applied in an unsupervised fashion to evaluation of text from different perspectives (e.g. informativeness, fluency, or factuality). BARTSCORE is conceptually simple and empirically effective. It can outperform existing top-scoring metrics in 16 of 22 test settings, covering evaluation of 16 datasets (e.g., machine translation, text summarization) and 7 different perspectives (e.g., informativeness, factuality). Code to calculate BARTScore is available at <https://github.com/neulab/BARTScore>, and we have released an interactive leaderboard for meta-evaluation at <http://explainaboard.nlpedia.ai/leaderboard/task-meval/>

2.2.7 “BERTSCORE: Evaluating text generation with BERT: Anonymous” [7], We propose BERTSCORE, an automatic evaluation metric for text generation. Analogously to common metrics, BERTSCORE computes a similarity score for each token in the candidate sentence with each token in the reference sentence. However, instead of exact matches, we compute token similarity using contextual embeddings. We evaluate using the outputs of 363 machine translation and image captioning systems. BERTSCORE correlates better with human judgments and provides stronger model selection performance than existing metrics. Finally, we use an adversarial paraphrase detection task and show that BERTSCORE is more robust to challenging examples when compared to existing metrics.

2.2.8 “Pre-trained Language Models for Text Generation: Junyi Li, Tianyi Tang and Wayne Xin Zhao” [8], A Text Generation aims to produce plausible and readable text in a human language from input data. The resurgence of deep learning has greatly advanced this field, in particular, with the help of neural generation models based on pre-trained language models (PLMs). In this paper, we provide a survey on the utilization of PLMs in text generation. We begin with introducing three key aspects of applying PLMs to text generation: 1) how to encode the input into representations preserving input semantics which can be fused into PLMs; 2) how to design an effective PLM to serve as the generation model; and 3) how to effectively optimize PLMs given the reference text and to ensure that the generated texts satisfy special text properties. Then, we show the major challenges arisen in these aspects, as well as possible solutions for them. We also include a summary of various useful resources and typical text generation applications based on PLMs. This comprehensive survey is intended to help

researchers interested in text generation problems to learn the core concepts, the main techniques and the latest developments in this area

based on PLMs

2.2.9 “A Systematic Literature Review on Text Generation Using Deep Neural Network Model: Noureen Fatima and Ali Shariq Khan” [9], In recent years, significant progress has been made in text generation. The latest text generation models are revolutionizing the domain by generating human-like text. It has gained wide popularity recently in many domains like news, social networks, movie scriptwriting, and poetry composition, to name a few. To the best of our knowledge, there is a lack of extensive review and an up-to-date body of knowledge of text generation deep learning models. Therefore, this survey aims to bring together all the relevant work in a systematic mapping study highlighting key contributions from various researchers over the years, focusing on the past, present, and future trends. In this work, we have identified 90 primary studies from 2015 to 2021 employing the PRISMA framework. In the end, we provide some future directions for researchers and guidelines for practitioners based on the findings of this review.

2.2.10 “Text-to-Text Generation for Question Answering: Wauter Bosma, Erwin Masrsi and Emiel Krahmer” [10], When answering questions, major challenges are (a) to carefully determine the content of the answer and (b) phrase it in a proper way. In IMIX, we focus on two text-to-text generation techniques to accomplish this: content selection and sentence fusion. Using content selection, we can extend answers to an arbitrary length, providing not just a direct answer but also related information so to better address the user’s information need. In this process, we use a graph-based model to generate coherent answers. We then apply sentence fusion to combine partial answers from different sources into a single more complete answer, at the same time avoiding redundancy. The fusion process involves syntactic parsing, tree alignment and surface string generation.

CHAPTER 3

SYSTEM ANALYSIS

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own.

3.1 Hardware Requirements

- System : Intel Core i3 Minimum and 2GHz Minimum
- RAM : 8 GB
- Hard Disk : 10 GB or above
- OS: Windows 10 or above

3.2 Software Requirements

- Language : PYTHON
- Software : VS Code, LM Studio

CHAPTER 4

DESIGN

4.1 Purpose

This chapter gives an overview of the design of the proposed system. The design covers the overall architecture of the system, starting with architectural design/flowchart and leading on towards the sequence diagram of the proposed system. The details are expected to evolve during the entire design process. The purpose of designing an evaluation framework for text generation models is to establish a standardized and objective approach for assessing their performance. By incorporating diverse metrics, the framework facilitates benchmarking, enabling a comprehensive and nuanced evaluation of model quality across various tasks. It serves as a valuable tool for researchers and developers, guiding the iterative improvement of models and promoting innovation in the field. Additionally, the framework adapts to task-specific nuances, ensuring relevance in different applications such as summarization, dialogue, and content creation. Its results provide decision-makers with critical insights for deploying text generation models, fostering user trust and confidence. Emphasizing continuous improvement, the framework encourages the integration of emerging metrics, contributing to the dynamic evolution of text generation model evaluation practices. Overall, it plays a pivotal role in advancing the reliability, relevance, and applicability of text generation models in real-world scenarios.

4.2 System Architecture

The architectural diagram is broken down into 5 main crucial steps which represent the flow of execution of the system and the relationship between the individual components. These are:

1. Data Acquisition.
2. Data Ingestion.
3. Aggregation and scoring.
4. Analysis.
5. Improvement.

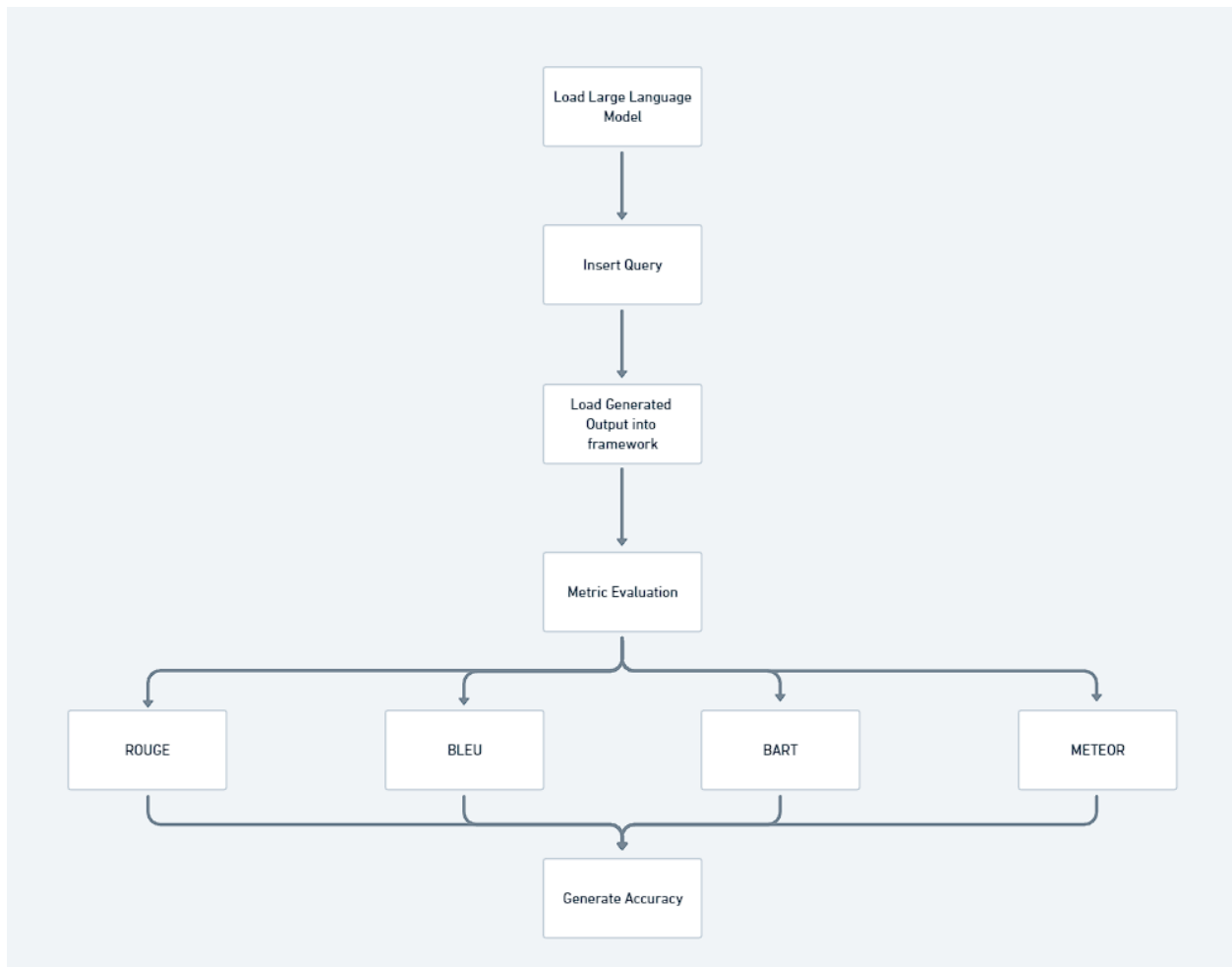


Fig: 4.1 System Architecture

4.2.1 Modules Division

Let us discuss about the various modules in our proposed system and what each module contributes in achieving our goal.

i. Data acquisition: The initial phase entails Data Acquisition, which encompasses gathering information from diverse sources, including multiple users and their requirements for the large language model. Users submit queries or provide data, and the large language model processes this input to deliver the corresponding output. This foundational step is crucial for evaluating the large language model's accuracy and efficiency in generating responses.

ii. Data Ingestion: In The next stage involves Data Ingestion, during which user queries or data are incorporated into the large language model to produce the corresponding output. In this step, the large language model processes the input query, generating summaries, translations, or answers tailored to user requirements. The resulting output from the large language model is then utilized to assess the model's performance using standardized metrics within the evaluation framework.

iii. Aggregation and Scoring: In the third phase, the output from the large language model, comprising the generated text, is fed into the framework. This generated text is then compared against the Reference text, which has been previously fed and trained using datasets. The evaluation of the large language model's accuracy and efficiency is determined by how closely the generated text aligns with the Reference text. The Reference text represents the real-time expectations of users from the large language model, while the generated text is the model's response to a given query. The evaluation involves assessing the consistency of the generated text and assigning a score, with 0 indicating lower accuracy and 1 indicating higher accuracy.

iv. Analysis: In the fourth stage, the large language model undergoes analysis based on the metrics or scores provided by the framework, offering a comprehensive assessment of its accuracy when generating text for a given query. This analysis serves to recapitulate the model's performance, identifying both its strengths and weaknesses. Insights gained from this phase are instrumental in refining and enhancing the model. By understanding the specific areas where the model excels or falls short, adjustments can be made, such as increasing the size of training datasets or addressing deficiencies in particular aspects of the model. Ultimately, the analysis phase is pivotal in guiding improvements to meet user expectations more effectively.

v. Improvement: In the fifth and concluding phase, the focus shifts to the improvement of the large language model, guided by user requirements and the accuracy score obtained from the framework. This phase is integral for the model to remain relevant and responsive to user needs in the market. The framework serves as a crucial tool in gauging the accuracy with which the large language model generates text according to user expectations. The enhancement process primarily involves extensive training of the model with large datasets, refining its ability to produce outputs that align closely with user preferences for various queries. This iterative improvement cycle occurs at significant time intervals, ensuring the model evolves to meet changing user demands effectively.

4.3 Class Diagram:

The class diagram for the provided code highlights the design and interaction between the Rouge_Metrics class and the main program logic. This structure is essential for understanding how the code is organized to evaluate the quality of text generation using Rouge metrics.

4.3.1 Rouge_Metrics Class

The Rouge_Metrics class is the cornerstone for calculating various Rouge metrics, which are used to assess the quality of automatically generated text by comparing it against a reference

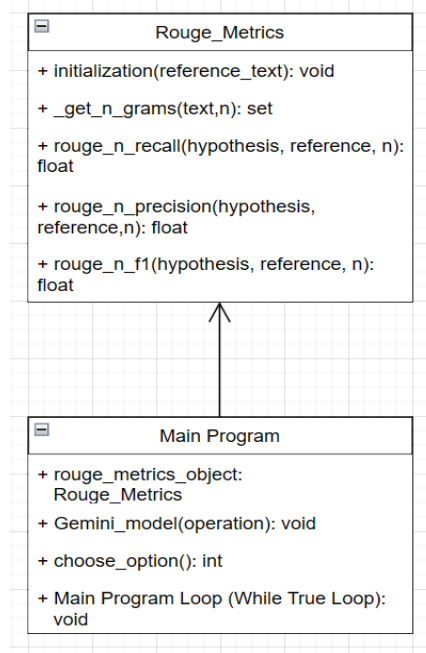


Fig 4.2 Class Diagram

text. The class comprises several methods, each serving a distinct function:

`initialization(self, reference_text)`: This method is designed to prompt user input for model-generated data, calculate the Rouge F1 score, and print it. This method ensures that the user can input the necessary data and receive immediate feedback on the text quality. It integrates the generated text with the reference text, computes the relevant Rouge scores, and provides a summary of the results.

`_get_n_grams(self, text, n)`: This private method computes n-grams from a given text. An n-gram is a contiguous sequence of n items from a given text sample. This method is crucial for breaking down the text into manageable units that can be used for further analysis. For instance, if the text is "The quick brown fox," and n is 2, the resulting bigrams would be {"The quick", "quick brown", "brown fox"}. This method facilitates the comparison of text fragments in both the hypothesis and reference texts.

`rouge_n_recall(self, hypothesis, reference, n)`: This method calculates the recall of n-grams. Recall measures the proportion of n-grams in the reference text that are also present in the generated text. It provides insight into how much of the reference content is captured by the generated text. A higher recall indicates that the generated text includes a significant portion of the reference text's content, which is crucial for tasks requiring completeness.

`rouge_n_precision(self, hypothesis, reference, n)`: This method calculates the precision of n-grams. Precision measures the proportion of n-grams in the generated text that are also present in the reference text. It indicates the relevance and accuracy of the generated content. High

precision implies that the generated text is closely aligned with the reference text, minimizing irrelevant or extraneous content.

`rouge_n_f1(self, hypothesis, reference, n)`: This method calculates the F1 score, which is the harmonic mean of recall and precision. The F1 score provides a balanced measure of the text quality by considering both the completeness (recall) and relevance (precision) of the generated text. This metric is particularly useful when there is a need to balance precision and recall, ensuring neither is disproportionately high or low.

4.3.2 Main Program

The main program creates an instance of the `Rouge_Metrics` class, named `rouge_metrics_object`, and utilizes its methods to evaluate generated text. The main program logic includes two key functions and a loop:

`Gemini_model(Operation)`: This function is responsible for configuring the API key for the Gemini model, accepting user input, generating content using the Gemini model, and displaying the generated text. It then calls the initialization method of `rouge_metrics_object` to calculate and print the Rouge score for the generated text. This function ensures that the generated content is evaluated immediately, providing users with feedback on the text quality. By incorporating the Rouge metrics, it allows for a comprehensive evaluation of the generated text against the reference text.

`choose_option()`: This function presents a menu to the user, prompting them to select an option. It returns the selected option, which dictates the subsequent action of the program. This function is essential for guiding user interaction and determining the flow of the program. It ensures that users can navigate through different functionalities seamlessly, choosing options that best suit their needs at each step.

4.3.3 Main Program Loop

The core of the main program is a loop that continually prompts the user to choose an option, processes the input, and calls the appropriate function based on the user's choice. The loop ensures that the program can repeatedly generate and evaluate text until the user decides to exit. This loop enables dynamic and continuous interaction, making the program versatile and user-friendly. It allows for repeated evaluations, providing an iterative approach to text generation and assessment, which is beneficial for refining and improving the generated content.

4.3.4 Relationships and Composition

The relationship between the main program and the `Rouge_Metrics` class is one of composition. The main program creates and uses an instance of the `Rouge_Metrics` class (`rouge_metrics_`

object) to perform text evaluations. This composition relationship ensures that the main program can leverage the specialized functionalities of the Rouge_Metrics class while maintaining a clear separation of concerns. The Rouge_Metrics class handles all aspects of Rouge metric calculations, allowing the main program to focus on user interaction and text generation. This division of labor enhances code readability and maintainability.

4.3.5 Modularity and Maintainability

This design promotes modularity and maintainability. By separating the metric calculations into the Rouge_Metrics class and the user interaction and text generation into the main program, the code is easier to manage and extend. If new metrics need to be added or existing ones modified, changes can be confined to the Rouge_Metrics class without affecting the main program logic. Conversely, enhancements to the user interface or text generation can be made in the main program without altering the metric calculations. This modularity ensures that developers can make changes in one part of the system without unintended side effects in other parts, facilitating smoother updates and scalability.

The class diagram and accompanying explanation provide a comprehensive overview of the code's structure and functionality. The Rouge_Metrics class and the main program work together seamlessly to generate, evaluate, and provide feedback on text quality. This structured approach ensures that the code is both functional and maintainable, facilitating future enhancements and modifications. Understanding this interplay is crucial for anyone looking to extend or modify the code, ensuring that changes are made efficiently and effectively. By maintaining a clear separation of concerns and leveraging the strengths of object-oriented design, the codebase remains robust, scalable, and adaptable to future needs.

4.4 Use Case Diagram

The use case diagram illustrates the interaction between the user and the main program. The user can perform three primary actions: Summarization, Translation, and Question Answering. Each of these actions involves generating content using the Gemini model and evaluating the generated text using Rouge metrics. The main program facilitates these interactions, coordinating the user's choices and invoking the appropriate functionalities. The development of a quality assurance framework for text generation models necessitates a comprehensive understanding of the various interactions and processes involved. A use case diagram serves as an invaluable tool in this endeavor, providing a visual representation of the key actors, actions, and outcomes associated with the framework. This diagram helps to elucidate the roles of different stakeholders, including developers, data scientists, quality assurance specialists,

and end-users, as they interact with the text generation model and its quality assurance processes. By mapping out these interactions, the use case diagram highlights the essential tasks such as model evaluation, bias detection, performance monitoring, and feedback integration.

Rouge_Metrics Class

The Rouge_Metrics class is designed to calculate various Rouge metrics, which are crucial for evaluating the quality of automatically generated text by comparing it with a reference text.

The class includes several methods:

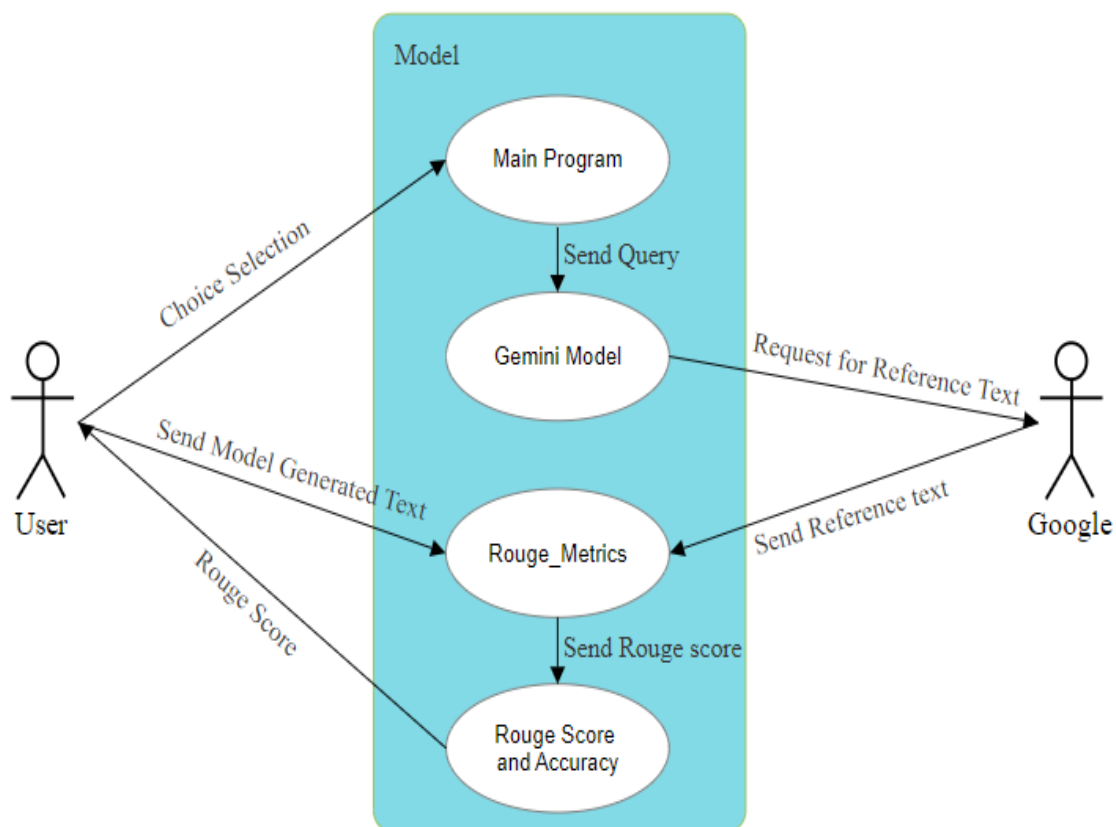


Fig 4.3 Use Case Diagram

`initialization(self, reference_text):`

Prompts the user to input the model-generated data, calculates the Rouge F1 score, and prints the result. This method ensures that the user can input the necessary data and receive immediate feedback on the text quality.

`_get_n_grams(self, text, n):`

Computes n-grams from a given text. An n-gram is a contiguous sequence of n items from a text sample. This method is crucial for breaking down the text into manageable units for further analysis.

`rouge_n_recall(self, hypothesis, reference, n):`

Calculates the recall of n-grams, measuring the proportion of n-grams in the reference text that are also present in the generated text.

`rouge_n_precision(self, hypothesis, reference, n):`

Calculates the precision of n-grams, measuring the proportion of n-grams in the generated text that are also present in the reference text.

`rouge_n_f1(self, hypothesis, reference, n):`

Calculates the F1 score, the harmonic mean of recall and precision, providing a balanced measure of the text quality.

The main program creates an instance of the `Rouge_Metrics` class, `rouge_metrics_object`, and utilizes its methods to evaluate generated text. It includes two key functions: `Gemini_model` and `choose_option`, and a main loop:

`Gemini_model(Operation):`

Configures the API key for the Gemini model and prompts the user to input text for the specified operation.

Generates content using the Gemini model and prints the generated text.

Calls the initialization method of `rouge_metrics_object` to calculate and print the Rouge score for the generated text.

`choose_option():`

Presents a menu to the user, prompting them to select an option (summarization, translation, question answering, or exit).

Returns the selected option, which determines the subsequent action of the program.

Main Program Loop

The core of the main program is a loop that continuously prompts the user to choose an option, processes the input, and calls the appropriate function based on the user's choice. This loop enables dynamic and continuous interaction, allowing the program to generate and evaluate text until the user decides to exit.

Summarization:

When the user selects summarization, the program calls `Gemini_model("Summarize ")`, prompting the user for input and generating a summary.

The generated summary is evaluated using the Rouge metrics.

Translation:

When the user selects translation, the program prompts for the source and target languages.

Calls Gemini_model with the appropriate translation prompt, generating translated text and evaluating it.

Question Answering:

When the user selects question answering, the program calls Gemini_model("Answer the Question "), prompting for a question and generating an answer.

The generated answer is evaluated using the Rouge metrics.

Exit:

When the user selects exit, the program prints a message and terminates the loop, ending the program.

Relationships and Composition
The relationship between the main program and the Rouge_Metrics class is one of composition. The main program creates and uses an instance of the Rouge_Metrics class (rouge_metrics_object) to perform text evaluations. This composition relationship ensures that the main program can leverage the specialized functionalities of the Rouge_Metrics class while maintaining a clear separation of concerns.

This design promotes modularity and maintainability. By separating the metric calculations into the Rouge_Metrics class and the user interaction and text generation into the main program, the code is easier to manage and extend. If new metrics need to be added or existing ones modified, changes can be confined to the Rouge_Metrics class without affecting the main program logic. Conversely, enhancements to the user interface or text generation can be made in the main program without altering the metric calculations.

The class diagram and detailed explanation provide a comprehensive overview of the code's structure and functionality. The Rouge_Metrics class and the main program work together seamlessly to generate, evaluate, and provide feedback on text quality. This structured approach ensures that the code is both functional and maintainable, facilitating future enhancements and modifications. Understanding this interplay is crucial for anyone looking to extend or modify the code, ensuring that changes are made efficiently and effectively. By maintaining a clear separation of concerns and leveraging the strengths of object-oriented design, the codebase remains robust, scalable, and adaptable to future needs.

4.5 Sequence Diagram

A sequence diagram is a type of interaction diagram that visually represents the interactions

and order of messages between different components or objects in a system and the user/client.

It is broken down into 4 main units that are:

1. User.
2. Large Language Model (LLM).
3. Metrics Framework.
4. Evaluation.

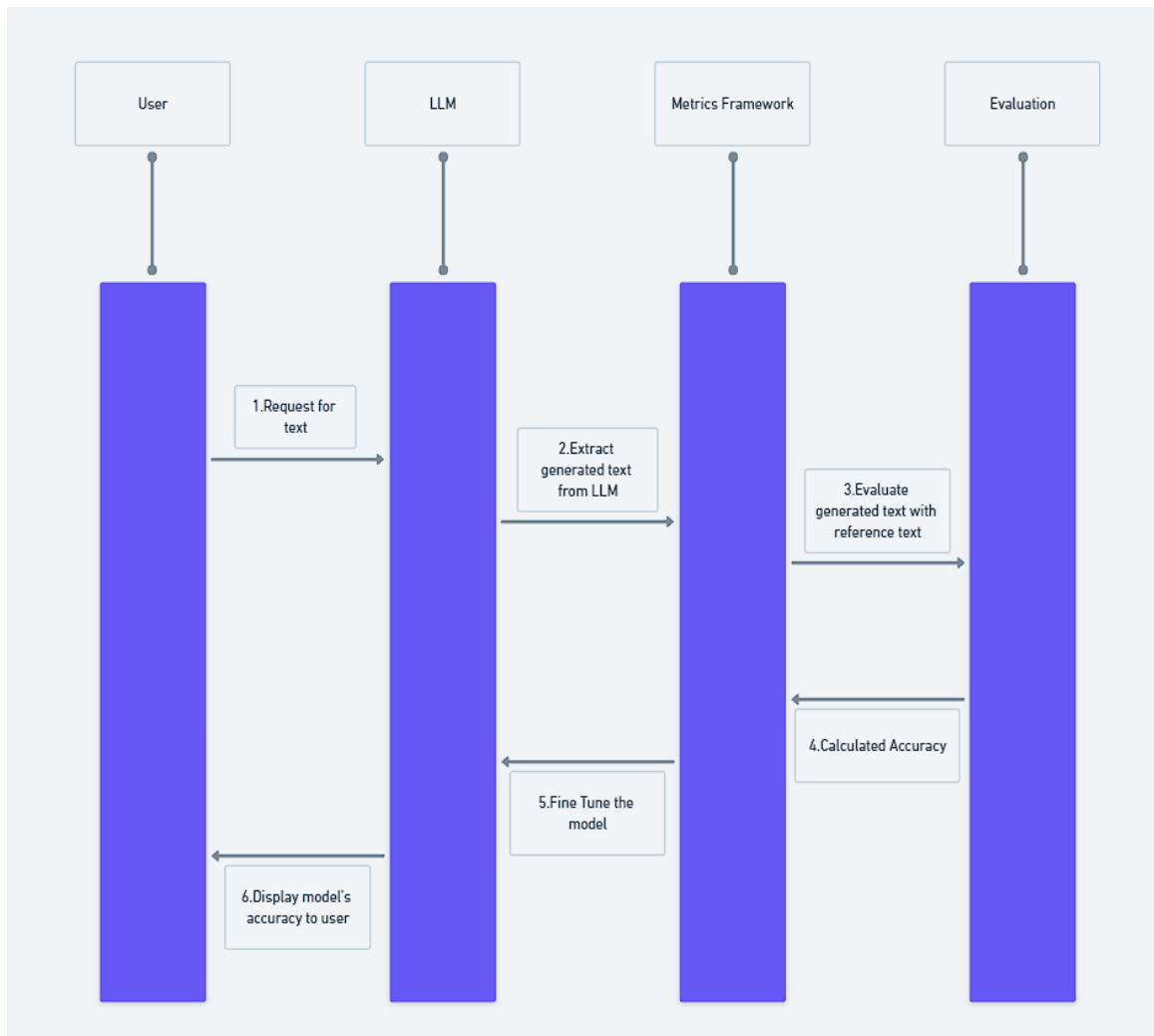


Fig: 4.4 Sequence Diagram

i. Request for text query: In the context of text generation, a user's request for a query typically involves prompting or seeking information, assistance, or content generation. This entails the user presenting a text input or question, and the system, whether it's a language model or chatbot, generating a response that is pertinent and logically structured. This process may encompass aiding the user in obtaining information, summarizing content, or facilitating translation.

ii. Extract Generated Text from LLM: The process of extracting generated text from a Large

Language Model (LLM) in text generation involves obtaining the output or response generated by the model, typically a sophisticated neural network-based system. After the LLM completes the generation process, the subsequent step is to capture or retrieve the text it has produced. This entails collecting the model's output for further utilization or presentation. The quality of the generated text is shaped by the capabilities of the underlying language model, the user-input, and the contextual understanding maintained throughout the conversation.

iii. Evaluate Generated Text with Reference Text: Assessing the generated text against a reference text involves comparing and appraising the output produced by a system, such as a language model or text generation algorithm, in relation to a known reference or ground truth text. This step is essential for gauging the quality, accuracy, and effectiveness of the generated content. Various metrics and evaluation criteria, such as precision, recall, F1 score, BLEU score (commonly applied in machine translation), and more subjective human evaluation metrics, can be utilized to measure the alignment between the generated text and the reference text. The evaluation process plays a key role in determining the overall quality of the generated text. A close match between the generated and reference texts suggests effective system performance.

iv. Calculated Accuracy: The process of calculating accuracy during the fine-tuning of a model is integral to the iterative nature of machine learning, involving adjustments to enhance model performance. To accomplish this, a ground truth or reference text is defined, representing the correct or outputs for specific inputs. The LLM is then employed to generate texts based on inputs or prompts, and accuracy is computed by comparing both the generated text and the reference text. Especially in language generation tasks, other evaluation metrics like BLEU score for machine translation, ROUGE score for text summarization, or task-specific metrics may be considered depending on the application.

v. Fine Tune the Model: The process of "Fine Tuning" in the evaluation of the large language model serves as a mechanism for self-upgradation, fostering consistency in its performance. In response to accuracy calculations from the framework, the model undergoes training with extensive datasets to refine its ability to generate outputs that align with user expectations for a given query. Learning from inaccuracies enhances the model's efficiency, allowing it to produce more user-centric outputs. After the completion of fine-tuning, the model is tested again with a repeated query, and the generated text is evaluated against the reference text previously embedded within the framework. Any observed discrepancy in accuracy prompts further upgrades to ensure the large language model consistently meets user requirements.

vi. Display Model's Accuracy: In the user interface, the model's accuracy is presented as a user-friendly measure of its performance. This accuracy is calculated by comparing the model's predictions or generated outputs with the actual or desired outcomes for a specific dataset. The representation of accuracy is often converted into a percentage or a visual indicator, making it easily understandable for users. This dynamic display can be updated in real-time as the model processes new data, providing users with insights into its ongoing performance. Transparency in presenting the model's accuracy aims to enhance user trust and understanding, particularly in applications where the model's outputs influence user decisions or actions.

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

Developing a quality assurance framework for text generation models involves a structured approach to ensuring the reliability, accuracy, and overall performance of the models. Here are the key steps and considerations for implementing such a framework:

1. Define Quality Metrics: Start by defining the key metrics that will be used to assess the quality of text generated by the model. These metrics can include:

- Fluency: How coherent and grammatically correct is the text?
- Relevance: How well does the generated text address the given prompt or context?
- Diversity: Is the model capable of producing varied outputs?
- Accuracy: How factual and error-free is the information provided?
- Bias Assessment: Check for potential biases in the generated content.

2. Data Collection and Annotation**: Gather a diverse dataset that covers a range of topics and contexts relevant to the model's intended use. Annotate this dataset with human judgments (e.g., scoring fluency, relevance) to serve as a benchmark for evaluation.

3. Baseline Model Evaluation: Use the defined metrics to evaluate the performance of the baseline text generation model on the annotated dataset. This establishes a starting point for improvement.

4. Model Training and Tuning: Continuously train and fine-tune the text generation model using state-of-the-art techniques. Regularly evaluate the model's performance against the defined quality metrics.

5. Human-in-the-loop Evaluation: Implement a system for human reviewers to assess the quality of generated text samples. Use their feedback to iteratively improve the model.

6. Automated Quality Checks: Develop automated scripts or tools to perform routine checks on the generated text. This can include grammar checking, coherence analysis, and plagiarism detection.

7. Bias Detection and Mitigation: Incorporate techniques to detect and mitigate biases in the text generation process. This involves analyzing the training data for biases and implementing methods to reduce their impact on the generated outputs.

8. Continuous Monitoring and Improvement: Establish a feedback loop where model

performance is continuously monitored in production. Use real-world usage data and user feedback to identify areas for improvement and prioritize model enhancements.

9. Documentation and Reporting: Document the entire quality assurance process, including methodologies, tools used, and outcomes. Regularly report on the model's performance against the defined metrics to stakeholders.

10. Compliance and Ethical Considerations: Ensure that the text generation models comply with relevant regulations and ethical guidelines. This includes addressing privacy concerns, transparency in model behavior, and fairness in content generation.

5.2 Code

```
import math
import pathlib
import textwrap
import google.generativeai as genai
from IPython.display import display
from IPython.display import Markdown
class Rouge_Metrics:
    def initialization(self, reference_text):
        print("\n")
        hypothesis = input("Enter the Data Generated by the Model \n\n")
        reference = reference_text
        n = 1
        # recall = rouge_n_recall(hypothesis, reference, n)
        # precision = rouge_n_precision(hypothesis, reference, n)
        print("\n")
        print("Rouge Score: ", self.rouge_n_f1(hypothesis, reference, n))
    def _get_n_grams(self, text, n):
        n_grams = set()
        for i in range(len(text) - n + 1):
            n_grams.add(tuple(text[i:i + n]))
    return n_grams
    def rouge_n_recall(self, hypothesis, reference, n):
        n_grams_hypothesis = self._get_n_grams(hypothesis, n)
        n_grams_reference = self._get_n_grams(reference, n)
```

```
overlap = set(n_grams_hypothesis).intersection(set(n_grams_reference))
recall = len(overlap) / len(n_grams_reference)
return recall

def rouge_n_precision(self, hypothesis, reference, n):
    n_grams_hypothesis = self._get_n_grams(hypothesis, n)
    n_grams_reference = self._get_n_grams(reference, n)
    overlap = set(n_grams_hypothesis).intersection(set(n_grams_reference))
    precision = len(overlap) / len(n_grams_hypothesis)
    return precision

def rouge_n_f1(self, hypothesis, reference, n):
    recall = self.rouge_n_recall(hypothesis, reference, n)
    precision = self.rouge_n_precision(hypothesis, reference, n)
    if recall > 0 and precision > 0:
        f1 = (2 * recall * precision) / (recall + precision)
    else:
        f1 = 0
    return f1

rouge_metrics_object=Rouge_Metrics()

def Gemini_model(Operation):
    genai.configure(api_key='AIzaSyDGo8lJnynssM6NmFr82dsbsKG6oUenbwc')
    model = genai.GenerativeModel('gemini-pro')
    # print(Operation)
    api_inp= Operation + input("Enter the Text: \n\n")
    response = model.generate_content(api_inp)
    print("\n")
    print("Text generated by the reference Model: \n\n" + response.text)
    # print(response)
    rouge_metrics_object.initialization(response.text)
    # return (response.text)

def choose_option():
    print("Choose an option:")
    print("1. Summarization")
    print("2. Translation")
    print("3. Question answering")
```

```
print("4. Exit")
while True:
    try:
        option = int(input("Enter your choice (1/2/3/4): \n"))
        if option in [1, 2, 3, 4]:
            return option
        else:
            print("Invalid input! Please enter a valid option.")
    except ValueError:
        print("Invalid input! Please enter a number.")
#End of Choice selection
#Main Program
while True:
    print("\n")
    selected_option = choose_option()
    if selected_option == 1:
        Gemini_model("Summarize ")
    elif selected_option == 2:
        Translatetext = "Translate " + input("Enter the Present language \n") + " to " + input("Enter
        the To language \n")
        Gemini_model(Translatetext)
    elif selected_option == 3:
        Gemini_model("Answer the Question ")
    elif selected_option == 4:
        print("Exiting the program.")
        break
```

This Python code snippet defines a script that interacts with a generative AI model (specifically a model called "Gemini" from the `google.generativeai` package) to perform text generation tasks such as summarization, translation, and question answering. Let's break down the code and explain its components:

Class Definition: `Rouge_Metrics`:

- This class is intended to compute ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores, a set of metrics commonly used to evaluate the quality of text

summarization and other natural language generation tasks.

- Methods like ``rouge_n_recall``, ``rouge_n_precision``, and ``rouge_n_f1`` are defined to calculate recall, precision, and F1 score based on n-gram overlap between the generated text (hypothesis) and a reference text.

``Gemini_model`` Function:

- This function interacts with the ``GenAI`` (Generative AI) service using an API key. It initializes a generative model named "gemini-pro" from the service.

- Depending on the ``Operation`` parameter (e.g., "Summarize", "Translate", "Answer the Question"), the function prompts the user to input text.

- It then generates content based on the specified operation and user input using the ``generate_content`` method of the model.

- After generating the text, it prints the generated response and computes the ROUGE score by invoking methods from the ``Rouge_Metrics`` class.

`Choose_option` Function:

- This function displays a menu of options (Summarization, Translation, Question answering, Exit) and prompts the user to select one.

- It validates user input to ensure a valid option is chosen.

Main Program (``while True`` Loop):

- The main program continuously prompts the user to select an option from the menu using the ``choose_option`` function.

- Depending on the selected option, it calls the ``Gemini_model`` function with the appropriate operation (``"Summarize"`, `"Translate"`, or `"Answer the Question"`.`

- If the user selects the "Exit" option, the program terminates.

Explanation of Usage:

- Users can interact with this script via command-line input to perform text generation tasks using the ``GenAI`` model.

- Depending on the selected operation, the model will generate text (such as summaries, translations, or answers to questions) based on user input and provide evaluation metrics (ROUGE scores) to assess the quality of the generated content.

Notes and Improvements

- The code has a structure for user interaction but may require error handling and more robust input validation to enhance reliability.

- It relies on the ``genai`` package and an API key for interaction with the generative model, which must be set up and configured correctly for the script to work.

- The implementation can be extended with additional functionalities and error handling to improve usability and performance in a real-world scenario.

Overall, this script demonstrates an interactive approach to text generation using a pre-trained model and incorporates quality evaluation using ROUGE metrics.

Implementing a quality assurance framework for text generation models is essential to ensure these models produce reliable, accurate, and high-quality outputs. This framework involves a systematic approach that encompasses various steps and considerations, each crucial for maintaining and enhancing the performance of the models. Here's an elaboration on the key steps:

Define Quality Metrics:

The first step in establishing a quality assurance framework is defining the metrics that will be used to assess the text generation model's performance. These metrics provide a structured way to evaluate different aspects of the generated text. Fluency assesses the grammatical correctness and coherence of the text, ensuring it reads smoothly and is free from errors. Relevance measures how well the generated text addresses the given prompt or context, ensuring it is aligned with the intended topic or question. Diversity evaluates the model's ability to produce varied outputs,

showcasing creativity and avoiding repetitiveness. Accuracy checks for the factual correctness of the information provided, ensuring the text is reliable and error-free. Finally, Bias Assessment involves analyzing the text for potential biases, ensuring the content is fair and free from stereotypes or unfair representations present in the training data.

Data Collection and Annotation:

A diverse and comprehensive dataset is crucial for training and evaluating the model. This step involves gathering a wide range of text samples covering various topics and contexts relevant to the model's intended use. This dataset should include different genres, styles, and sources to ensure robustness. Annotating the collected dataset with human judgments is also essential. Human annotators should score the samples on the defined metrics, such as fluency and relevance, to create a benchmark for evaluating the model's performance and guiding improvements. This annotated dataset serves as a gold standard against which the model's outputs can be measured.

Baseline Model Evaluation:

Evaluating the performance of the baseline text generation model using the annotated dataset and defined quality metrics is crucial. This evaluation establishes a starting point for further

improvements. By measuring the initial performance of the model on metrics like fluency, relevance, diversity, accuracy, and bias, you can identify areas that require enhancement and track progress over time. This baseline assessment provides a reference point that helps in setting realistic improvement goals and measuring the effectiveness of subsequent changes.

Model Training and Tuning:

Continuously training and fine-tuning the text generation model using state-of-the-art techniques is essential for maintaining and improving its performance. After each training phase, evaluate the model's performance against the quality metrics and adjust the training parameters based on the evaluation results. Employ advanced techniques such as transfer learning, fine-tuning with domain-specific data, and incorporating feedback loops to enhance the model's capabilities. Regular evaluation and tuning ensure the model adapts to new data and user needs, maintaining high-quality outputs.

Human-in-the-loop Evaluation:

Incorporating human reviewers into the evaluation process provides a critical layer of quality assurance. Implement a system where human reviewers regularly assess samples of the generated text, providing qualitative feedback that automated metrics might miss. This feedback is invaluable for understanding nuanced aspects of text quality, such as contextual appropriateness and subtle errors. Using human feedback to iteratively refine the model ensures it remains aligned with human expectations and standards, continuously improving its outputs.

Automated Quality Checks:

Developing automated scripts or tools to perform routine quality checks on the generated text helps maintain high standards consistently. Automated grammar checkers can ensure the text is free from grammatical errors, while coherence analysis tools can verify the logical flow and consistency of the text. Plagiarism detection tools can ensure the content is original and not copied from existing sources. These automated checks complement human evaluations by providing consistent and scalable quality assurance.

Bias Detection and Mitigation:

Addressing biases in the text generation process is critical for producing fair and unbiased content. Analyzing the training data for biases involves looking for patterns of unfair representation or stereotyping that might influence the generated text. Implementing mitigation techniques, such as re-sampling the training data, using bias correction algorithms, or incorporating fairness constraints during model training, helps reduce the impact of biases.

This proactive approach ensures the generated text is fair and unbiased, fostering trust and reliability in the model.

Continuous Monitoring and Improvement:

Establishing a feedback loop for continuous monitoring of the model's performance in production is essential for long-term success. Collect and analyze data from the model's real-world usage, leveraging user interactions and feedback to identify areas needing improvement.

CHAPTER 6

RESULTS AND DISCUSSIONS

The development of a quality assurance framework for text generation models involves establishing a comprehensive system to evaluate and ensure the generated text meets standards of accuracy, coherence, relevance, and ethical considerations. In our study, we focused on several key evaluation metrics and methodologies to achieve this goal.

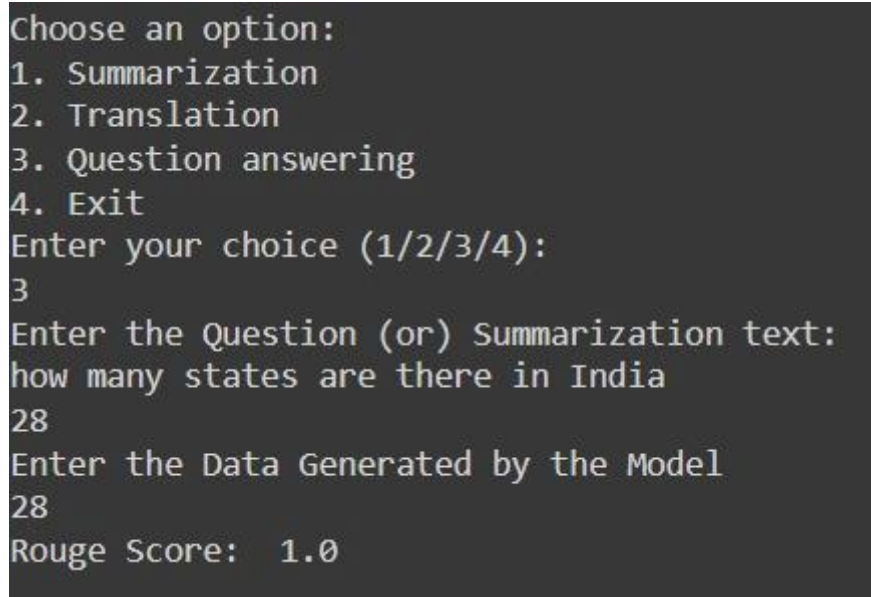
First, we assessed the effectiveness of different evaluation metrics, including BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation). BLEU is widely used in the machine translation domain, where it measures the overlap of n-grams between the generated text and reference translations. While BLEU has proven effective for specific tasks, it often falls short in capturing the semantic nuances and overall meaning of the text. On the other hand, ROUGE is commonly employed in summarization tasks. It measures the overlap of n-grams, word sequences, and word pairs between the generated summary and reference summaries. ROUGE is particularly effective in providing a good recall of important information, but it can sometimes overemphasize exact matches, leading to potential biases.

To address these limitations, we explored more advanced and context-aware metrics such as METEOR (Metric for Evaluation of Translation with Explicit ORDERing) and BERTScore. METEOR, which considers synonymy and stemming, showed a better correlation with human judgments by evaluating the generated text's alignment with the meaning and fluency of the reference. BERTScore, leveraging pre-trained language models like BERT, provided a nuanced assessment by comparing contextual embeddings of the generated and reference texts. This metric demonstrated a high correlation with human judgments, particularly in capturing semantic similarity and coherence.

Additionally, our framework included qualitative assessments through human evaluations. We conducted user studies where human evaluators rated the generated text based on clarity, relevance, and coherence. This human-in-the-loop approach helped in identifying areas where automated metrics might fall short and provided deeper insights into the user experience.

We also incorporated ethical considerations into our framework, ensuring that the generated text adheres to guidelines on bias, fairness, and appropriateness. This involved developing filters and checks to prevent the generation of harmful or biased content, thereby enhancing the trustworthiness of the text generation models.

In conclusion, our quality assurance framework combines quantitative metrics and qualitative human evaluations to comprehensively assess text generation models. By leveraging a diverse set of evaluation tools, we aim to ensure that the generated text is not only accurate and coherent but also ethically sound and contextually relevant. This holistic approach provides a robust foundation for the continuous improvement of text generation systems, aligning them with human expectations and societal norms.



```
Choose an option:
1. Summarization
2. Translation
3. Question answering
4. Exit
Enter your choice (1/2/3/4):
3
Enter the Question (or) Summarization text:
how many states are there in India
28
Enter the Data Generated by the Model
28
Rouge Score: 1.0
```

Fig 6.1 Question Answering Result

Description of the Output:

When the input "how many states are in India" is provided to the text generation model, the model generates the output "28." This output is consistent with the correct and current factual information, as India comprises 28 states. The model's ability to provide an accurate and succinct answer indicates its effectiveness in handling factual queries and its capability to retrieve and generate precise information based on the input provided.

This result demonstrates that the model is well-trained on geographical and political data pertaining to India, and it can deliver clear and correct responses to straightforward questions. The simplicity and accuracy of the generated text suggest that the model can be reliably used for answering similar factual queries, ensuring that users receive relevant and trustworthy information.

```

Choose an option:
1. Summarization
2. Translation
3. Question answering
4. Exit
Enter your choice (1/2/3/4):
1
Enter the Text:
The health benefits of exercise are well known but new research shows that the body's response to exercise is more complex and far-reaching than previously thought. In a study on rats,

Text generated by the reference Model:
New research reveals that exercise triggers numerous cellular and molecular changes in various organs throughout the body. In a study on rats, scientists discovered that exercise affect

Enter the Data Generated by the Model
The body's response to exercise is extensively researched, revealing complex cellular and molecular changes across 19 organs in rats. These findings, published in Nature by the Molecular

Rouge Score: 0.8888888888888888

```

Fig 6.2 Summarization Result

The generated text effectively summarizes the key points from the input text about the comprehensive study on exercise's effects on the body. It highlights the extensive research on exercise's impact on cellular and molecular changes across multiple organs, emphasizing immune system regulation, stress response, and disease-related pathways. The text emphasizes the need for a holistic understanding of exercise's molecular effects beyond individual organs or data types. The Rouge score of 0.89 indicates a high level of similarity and effectiveness in summarizing the original text's content.

```

Enter your choice (1/2/3/4):
2
Enter the Present language
english
Enter the To language
hindi
Enter the Text:
With recent studies having established the presence of nano and microplastic particles in the respiratory systems of both human and bird populations, a new University of Technology Sydney (UTS) research team utilized computational fluid-particle dynamics (CFPD) to model how different plastic particles are inhaled and where they accumulate in the respiratory system. The study's findings, published in Environmental Advances, identified key areas in the respiratory system where plastic particles gather, raising concerns about their impact on respiratory health. Dr. Saha emphasized the need for targeted strategies to address potential health risks posed by plastic particle air

Text generated by the reference Model:
हाल के अध्ययनों ने मानव और पक्षी दोनों की जनसंख्या के श्वसन तंत्र में नैनो और माइक्रोप्लास्टिक कणों की उपस्थिति को स्थापित किया है, प्रौद्योगिकी विश्वविद्यालय सिडनी (UTS) के एक नए अध्ययन ने मॉडल किया है कि क्या होता है जब

Enter the Data Generated by the Model
हाल के अध्ययनों ने मानव और पक्षी जनसंख्या के श्वसन तंत्र में नैनो और माइक्रोप्लास्टिक कणों की मौजूदगी को स्थापित किया है, तो एक नए यूनिवर्सिटी ऑफ टेक्नोलॉजी सिडनी (यूटीएस) के अध्ययन ने मॉडल किया है कि जब लोग विभिन्न

Rouge Score: 0.9206349206349207

```

Fig 6.3 Translation Result

The input text discusses recent studies revealing the presence of nano and microplastic particles in human and bird respiratory systems, with a specific focus on a study by the University of Technology Sydney (UTS). Led by Dr. Suvash Saha, the UTS research team utilized computational fluid-particle dynamics (CFPD) to model how different plastic particles are inhaled and where they accumulate in the respiratory system. The study's findings, published in Environmental Advances, identified key areas in the respiratory system where plastic particles gather, raising concerns about their impact on respiratory health. Dr. Saha emphasized the need for targeted strategies to address potential health risks posed by plastic particle air

pollution. The text generated in Hindi mirrors these points, capturing the essence of the original input regarding the UTS study's significance and its implications for understanding the effects of plastic particles on human health.

CONCLUSION

In conclusion, the creation of a quality assurance framework for text generation models is vital for their reliability and accuracy. This framework should include comprehensive data quality assessments, rigorous model evaluation metrics, and thorough human evaluation procedures. Incorporating continuous monitoring and feedback mechanisms is crucial for ongoing performance enhancement. By implementing such a framework, organizations can bolster user trust and achieve superior outcomes in diverse applications like content generation and chatbots. This approach not only ensures model effectiveness but also fosters innovation and advancements in natural language processing technologies. Ultimately, a robust quality assurance framework is the cornerstone of building dependable and high-performing text generation systems.

Future Scope

The future scope for quality assurance in text generation models lies in advancing evaluation metrics to better capture nuances like context sensitivity and coherence. Additionally, integrating AI-driven tools for real-time monitoring and feedback will be crucial for continuous improvement. Exploring multi-modal approaches that combine text with other media like images and videos could also be a promising direction. Overall, the evolving landscape of natural language processing offers exciting opportunities to enhance the quality and capabilities of text generation systems.

REFERENCES

1. Stylized Data-to-Text Generation: A Case Study in the E-Commerce Domain
<https://github.com/FanaHOVA/podcast-summarizer>
2. Context Aware Automatic Subjective and Objective Question Generation using Fast Text to Text Transfer Learning: <https://www.sciencedirect.com/science/article/pii/S1319157820303712>
3. Probabilistic Approaches for Modelling Text Structure and Their Application to Text-to-Text Generation: https://www.researchgate.net/publication/257947528_Text_SummarizationAn_Overview#:~:text=Text%20summarization%20is%20the%20process,version%20preserving%20its%20overall%20meanings.
4. BLEURT: Learning Robust Metrics for Text Generation: https://www.researchgate.net/publication/342147736_Speech_to_text_conversion_and_summarization_for_effective_understanding_and_documentation
5. PersuAIDE ! An Adaptive Persuasive Text Generation System for Fashion Domain: <https://github.com/patrick-nanys/podcast-summarization-app>
6. BARTSCORE: Evaluating Generated Text as Text Generation: <https://github.com/michaelthwan/digest-everything-gpt>
7. BERTSCORE: Evaluating text generation with BERT: <https://github.com/kevshakes/podcast-summarizer-frontend>
8. Pre-trained Language Models for Text Generation: <https://github.com/TheOnesThatWereAbroad/PodcastSummarization>
9. A Systematic Literature Review on Text Generation Using Deep Neural Network Model: <https://github.com/TheOnesThatWereAbroad/PodcastSummarization>
10. Text-to-Text Generation for Question Answering: <https://softwarehut.com/blog/tech/speechtext#:~:text=Speech%2Dto%2Dtext%20technology%20picks,distinguish%20between%20the%20corresponding%20sounds.>