AI-based Generative QA System

Group No: 17

Chandrasekhar B, K Kasi Viswanath

Abstract

This project, titled "AI-based Generative QA System," focuses on fine-tuning generative language models for two specific tasks: email subject line generation and answering questions related to artificial intelligence and machine learning (AIML). As a participant-driven initiative, the project emphasizes hands-on experience with GPT-based models, allowing participants to explore the intricacies of generative text systems.

The first task involves training a GPT variant to generate concise email subject lines from the content of email bodies. This task simulates real-world applications where large amounts of text need to be condensed into impactful summaries. The second task requires participants to create a domain-specific QA system, where a GPT model is fine-tuned on a custom-curated AIML question-answer dataset. The goal is to improve the model's ability to generate accurate, relevant answers for technical questions within the AIML domain.

Throughout this project, participants will work through the complete machine learning lifecycle—data curation, model training, fine-tuning, evaluation, and deployment—using industry-standard tools such as Hugging Face, PyTorch, TensorFlow, and cloud-based deployment platforms. The project aims to offer participants a comprehensive understanding of how to build, fine-tune, and deploy state-of-the-art generative models in specialized domains.

Introduction

The development of AI-based generative models has transformed numerous fields, especially in natural language processing (NLP). This project, "AI-based Generative QA System," seeks to introduce participants to two distinct yet interrelated tasks in generative modeling. The project provides an opportunity to apply advanced NLP techniques using GPT-based models, focusing on text summarization and question-answering systems.

In the first task, participants will explore the challenge of email subject line generation, a task closely related to text summarization. Unlike traditional headline generation or summarization tasks, generating concise email subjects involves extracting the most relevant information from a body of text and expressing it in just a few words. This task will help participants understand how generative models work in extracting key information from large amounts of data.

The second task transitions from summarization to domain-specific question-answering (QA). Participants will collaboratively curate an AIML-specific dataset by compiling a collection of questions and short answers derived from their course material. This dataset will then be used to fine-tune a GPT model, enabling it to generate precise answers to technical questions in the field of AIML. The project introduces the concept of domain adaptation, where pre-trained models are specialized to perform well in specific fields through fine-tuning.

By participating in this project, individuals will gain a practical understanding of how generative models can be fine-tuned to achieve high performance on both generic and domain-specific tasks. In addition, the project fosters teamwork in data collection and emphasizes the importance of proper evaluation using established metrics.

Objective

The primary objective of this project is to fine-tune a generative language model (specifically, a GPT variant) for two distinct tasks:

1. **Email Subject Line Generation:** To develop a model capable of analyzing the body of an email and generating a succinct, relevant subject line that captures the essence of the email content. This task aims to improve efficiency in email communication by creating impactful summaries.
2. **Question Answering on AIML Queries:** To create a domain-specific question-answering system that generates accurate responses to questions related to artificial intelligence and machine learning. This involves fine-tuning a GPT model on a curated dataset of AIML-related questions and answers, thereby enhancing its ability to handle technical inquiries effectively.

Through these objectives, the project aims to provide participants with hands-on experience in fine-tuning models, understanding the nuances of generative text systems, and applying machine learning techniques to real-world challenges.

Project Overview

The project aims to provide hands-on experience with generative text systems through two main tasks:

1. **Email Subject Generation:**
    - Fine-tune a GPT model using a prepared dataset.
    - Generate succinct email subjects from email bodies.
    - **Additional Learning:** Explore and compare other LLM models such as BERT, T5, and BART to understand their performance in email subject generation. Learn from different architectures to enhance the effectiveness of the GPT model.

2. **AIML Question Answering:**
    - Create a new dataset for AIML questions.
    - Fine-tune a GPT model to generate answers to these questions.
    - **Additional Learning:** Investigate other LLM models like T5, BERT, and BART for question answering. Compare their results to gain insights into their strengths and weaknesses in handling AIML queries.

1. Email Subject Line Generation

This task involves creating extremely short and concise email subjects, which is unique compared to tasks like news summarization or headline generation. The goal is to extract the most important sentences from the email body and condense them into a brief subject line.

Key Aspects:

- **Uniqueness:** The challenge lies in generating very brief summaries by focusing on the most salient information from the email body.
- **Implementation:** Utilize any GPT-2 variant to address the task, providing hands-on experience with generative models in NLP.
- **Evaluation:** Analyze and apply various metrics to evaluate the quality and effectiveness of the generated text.

This task offers a chance to delve into generative models, learn about their application in creating concise email subject lines, and understand different evaluation methods for text generation.

2. Question Answering on AIML Queries

Building on the experience gained from the first task, this task focuses on developing a domain-specific GPT variant model to answer questions related to the AIML course. Pretrained models often perform well on general tasks but may struggle with domain-specific queries. To address this, the model needs to be fine-tuned on a dataset specifically tailored to AIML topics.

Key Aspects:

- **Domain-Specific Modeling:** Fine-tune a GPT variant model to improve its ability to handle AIML-related questions.
- **Dataset Creation:** Collaborate to build a relevant dataset specifically for AIML questions.
- **Performance Evaluation:** Assess the model's effectiveness by testing it on new, unseen AIML queries.

This task provides an opportunity to refine model finetuning techniques, develop domain-specific datasets, and evaluate the model's performance in answering specialized questions.

Datasets

1. Email Subject Line Generation

- **Dataset:** Annotated Enron Subject Line Corpus
- **Description:**
    - **Content:** This dataset includes a subset of cleaned, filtered, and deduplicated emails from the Enron Email Corpus, featuring employee inboxes from the Enron Corporation.
    - **Evaluation Split:** The development and test sets each have 3 annotated subject lines per email by human annotators. This approach provides multiple reference points, as there isn't always one unique, appropriate subject line per email.
    - **Statistics:**
        - **Train/Dev/Test Splits:** 14,436 / 1,960 / 1,906
        - **Average Email Length:** 75 words
        - **Average Subject Length:** 4 words

2. Question Answering on AIML Queries

- **Dataset:** AIML QA Corpus
- **Description:**
    - **Creation:** This dataset will be curated collaboratively by all teams participating in the AIML course's NLP projects.
    - **Content:** Each team will contribute by answering a question bank of 250 questions, providing short 1-2 line answers which will be compiled into a CSV file.
    - **Source:** Questions will be drawn from the AIML course material covered in lectures.
    - **Deadline:** The dataset needs to be completed within 1 month of the project start date to allow enough time for QA model fine-tuning.
    - **Post-Creation:** After dataset completion, a common train/dev/test split will be provided for further experimentation with the QA model.

These datasets are essential for fine-tuning and evaluating the models for their respective tasks, providing a solid foundation for the project.

Preprocessing Steps

Preprocessing Steps for Task 1: Email Subject Line Generation

1. **Dataset Cloning**
   The dataset for this task, **The Annotated Enron Subject Line Corpus**, is sourced from a public GitHub repository. To begin, the repository is cloned, and separate paths are defined for training, validation, and testing data.
2. **Training Data Preprocessing**
    - Each file in the training dataset contains an email body and its corresponding subject line.
    - The files are read, and the content is split into two parts: the **Email Body** and the **Subject Line**.

- A **DataFrame** is created to store these pairs for further use in model training.

3. **Validation Data Preprocessing**
   - The validation dataset contains additional **annotations** for each email, apart from the email body and subject.
   - Files are processed by splitting the content into the **Email Body**, **Subject Line**, and three human-labeled annotations (**Ann0**, **Ann1**, **Ann2**).
   - A separate **DataFrame** is created for this, ensuring the annotations are mapped correctly for evaluation purposes.
4. **Test Data Preprocessing**
   - The test dataset is similar to the validation dataset in structure, with each file containing the **Email Body**, **Subject Line**, and three annotations (**Ann0**, **Ann1**, **Ann2**).
   - The data is processed similarly, and a final **DataFrame** is created to hold the test data.
5. **Final Output**
   Once preprocessing is completed, we have three DataFrames:
   - **Train DataFrame** containing emails and their corresponding subjects.
   - **Validation DataFrame** with emails, subjects, and three additional annotations for model evaluation.
   - **Test DataFrame** structured like the validation set, used for final performance testing.

These steps ensure the dataset is clean, structured, and ready for training the GPT model to generate concise email subject lines.

For the full preprocessing code, refer to the Preprocessing Notebook.

Preprocessing Steps for Task 2: Question Answering on AIML Queries

1. **Dataset Preparation**
   The dataset for the second task is created collectively by teams as part of the AIML course. The dataset consists of questions related to AIML concepts with short, concise answers. The dataset is split into **train**, **dev**,
   and **test** sets.

2. **Loading Training Data**
   - The **training data** is loaded from a CSV file (train.csv).
   - This file contains AIML-related questions and their corresponding short answers.

3. **Loading Validation Data**
   - The **validation data** is loaded from a separate CSV file (dev.csv).
   - This file follows the same structure as the training data and is used for model validation during fine-tuning.

4. **Loading Test Data**
   - The **test data** is similarly loaded from another CSV file (test.csv).
   - This data is used to evaluate the final performance of the model on unseen AIML questions.

5. **Final Output**
   After loading, the datasets are organized into **train**, **dev**,
   and **test** DataFrames, which are then used to fine-tune and evaluate the GPT model for AIML question answering.

These preprocessing steps prepare the data for modeling and evaluation in the task of AIML question answering.

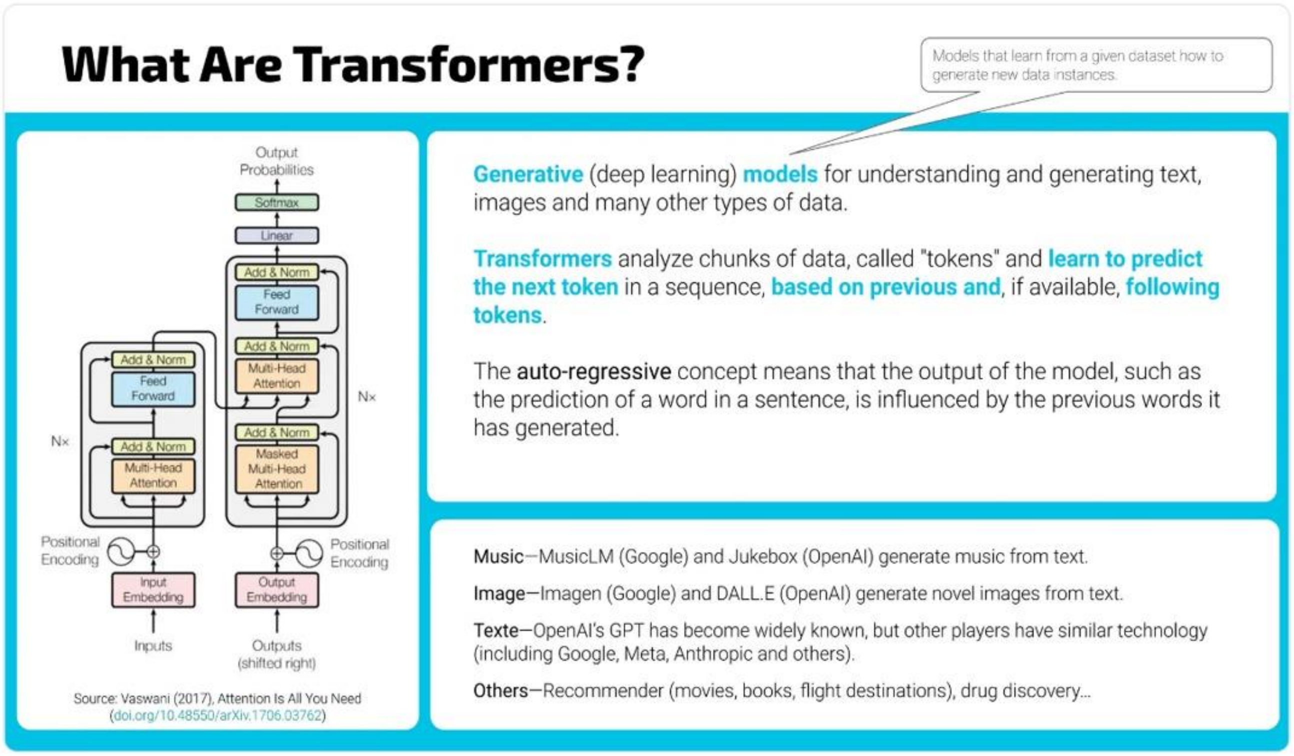
For the full preprocessing code, refer to the Preprocessing Notebook.

LLM (Large Language Model)

Large Language Models (LLMs) are a type of artificial intelligence designed to understand and generate human-like text. They are trained on vast amounts of text data to learn patterns in language, enabling them to perform various tasks such as:

- **Text Generation**: Creating coherent and contextually relevant text based on prompts.
- **Text Completion**: Predicting the next word or sentence in a given context.
- **Question Answering**: Providing answers to user queries based on the information they've learned.
- **Summarization**: Condensing long texts into shorter summaries while preserving key information.

LLMs use architectures like transformers to achieve these tasks effectively.



What Are Transformers?

Models that learn from a given dataset how to generate new data instances.

Source: Vaswani (2017), Attention Is All You Need (doi.org/10.48550/arXiv.1706.03762)

Generative (deep learning) **models** for understanding and generating text, images and many other types of data.

Transformers analyze chunks of data, called "tokens" and **learn to predict the next token** in a sequence, **based on previous and**, if available, **following tokens**.

The **auto-regressive** concept means that the output of the model, such as the prediction of a word in a sentence, is influenced by the previous words it has generated.

Music—MusicLM (Google) and Jukebox (OpenAI) generate music from text.

Image—Imagen (Google) and DALL.E (OpenAI) generate novel images from text.

Texte—OpenAI's GPT has become widely known, but other players have similar technology (including Google, Meta, Anthropic and others).

Others—Recommender (movies, books, flight destinations), drug discovery...

LLM Models Information

Here's the information formatted as a single table for easy pasting into MS Word:

Model	Model Name	Parameters	Architecture	Architecture Type	Main Use Case	Open-source
GPT-1	gpt	117M	Transformer	Decoder	Text generation, language modeling	No
GPT-2 Small	gpt2	117M	Transformer	Decoder	Text generation, simpler tasks	Yes
GPT-2 Medium	gpt2-medium	345M	Transformer	Decoder	Text generation, dialogue	Yes
GPT-2 Large	gpt2-large	774M	Transformer	Decoder	Long-form text, story generation	Yes
BERT Base	bert-base-uncased	110M	Transformer	Encoder	Text classification, token classification, Q&A	Yes
T5 Small	t5-small	60M	Text-to-Text Transformer	Encoder-Decoder	Text generation, translation, summarization	Yes
T5 Base	t5-base	220M	Text-to-Text Transformer	Encoder-Decoder	Text-to-text tasks, summarization, Q&A	Yes
T5 Large	t5-large	770M	Text-to-Text Transformer	Encoder-Decoder	High-quality text generation, complex NLP tasks	Yes
RoBERTa Base	roberta-base	125M	Transformer	Encoder	Text classification, token classification, Q&A	Yes
BART Base	facebook/bart-base	140M	Transformer	Encoder-Decoder	Text generation, summarization, translation	Yes
LLaMA 7B	llama-7b	7B	Transformer	Decoder	General-purpose text generation, language modeling	Yes
LLaMA 13B	llama-13b	13B	Transformer	Decoder	More complex text generation, advanced NLP tasks	Yes

Training Steps Summary

Task 1: Email Subject Line Generation

1. **Dataset Loading:**
    - Loaded train.csv, val.csv, and test.csv into pandas DataFrames.
2. **Model Selection and Tokenizer Initialization:**
    - Explored models: GPT-2, DistilGPT2, GPT2-Medium, BART-base, BART-large-CNN, T5-base, and T5-small.
    - Initialized corresponding tokenizers with special tokens.
3. **Dataset Preparation:**
    - Implemented a custom dataset class for tokenizing emails and subjects, generating input IDs and attention masks with a max length of 250 tokens.
4. **Model Training Setup:**
    - Defined training parameters (15 epochs, batch sizes of 8 and 16, evaluation strategy, gradient accumulation, checkpointing) using Hugging Face’s TrainingArguments.
    - Best model selected based on **ROUGE-L**.
5. **Metrics Computation:**
    - Used **ROUGE** to evaluate model performance:
        - ROUGE-1: unigram overlap
        - ROUGE-2: bigram overlap
        - ROUGE-L: longest common subsequence
6. **Model Training:**
    - Trained model using Hugging Face's Trainer API, generating and evaluating candidate subject lines.
7. **Evaluation and Testing:**
    - Tested trained model on unseen data and evaluated using **ROUGE** metrics, saving predictions to a CSV file.
8. **Multiple Model Testing:**
    - Models tested: GPT-2, DistilGPT2, GPT2-Medium, BART-base, BART-large-CNN, T5-base, T5-small.
    - Achieved ROUGE scores summarized in a table.

Task 2: Question Answering on AIML Queries

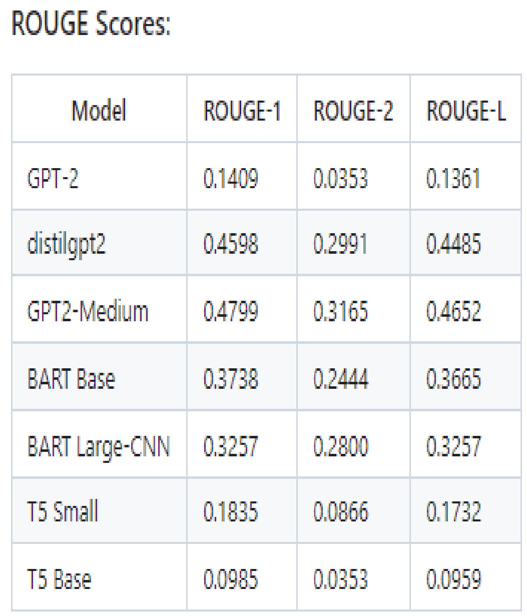
1. **Dataset Loading:**
   - Loaded train.csv, dev.csv, and test.csv into pandas DataFrames.
2. **Model Selection and Tokenizer Initialization:**
   - Explored models: GPT-2, DistilGPT2, GPT-2 Medium, BART-large, T5-base, Flan-T5-base, Flan-T5-small, RoBERTa-base.
   - Initialized tokenizers for each model.
3. **Dataset Preparation:**
   - Implemented a custom dataset class for tokenizing questions and answers, generating input IDs and attention masks with a max length of 512 tokens.
4. **Model Training Setup:**
   - Defined training parameters with Hugging Face's TrainingArguments, focusing on optimal validation performance metrics.
5. **Metrics Computation:**
   - Computed BLEU, ROUGE, and METEOR metrics to evaluate generated answers against reference answers.
6. **Model Training:**
   - Trained model with Hugging Face's Trainer API, generating and evaluating answers.
7. **Evaluation and Testing:**
   - Tested trained model on unseen data, evaluating predictions with BLEU, ROUGE, and METEOR metrics, saving results to a CSV file.
8. **Multiple Model Testing:**
   - Models tested included GPT-2, DistilGPT2, GPT-2 Medium, BART-large, T5-base, Flan-T5-base, Flan-T5-small, and RoBERTa-base.
   - ROUGE scores achieved summarized in a table.

Additional Testing and Deployment

- **Zero-Shot Inferencing:** Conducted zero-shot inference using various models for text generation.
- **Model Deployment:** Fine-tuned models pushed to Hugging Face model hub for sharing and deployment.

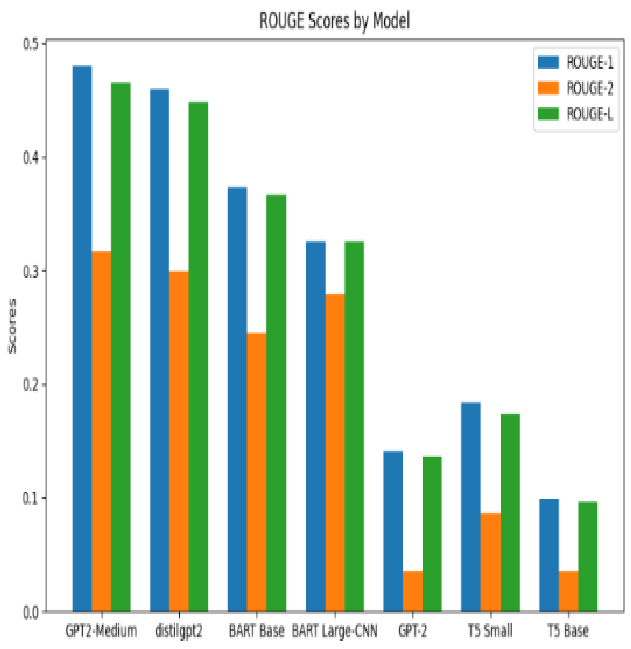
Model Evaluation

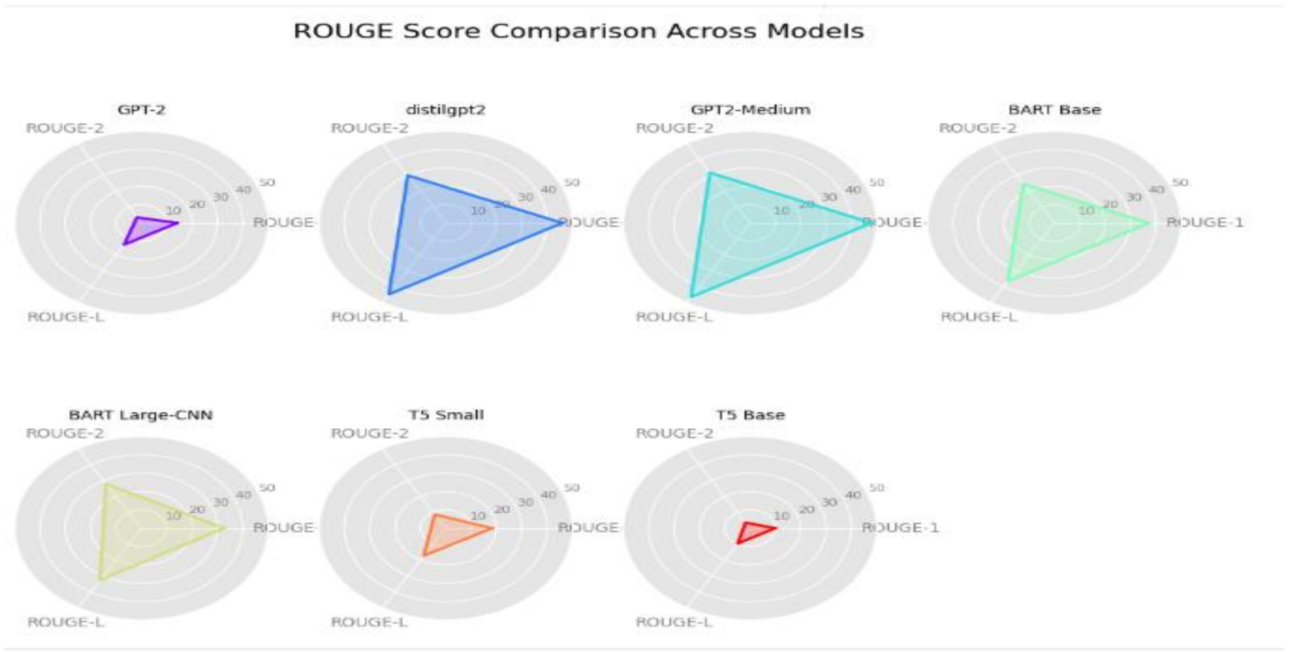
ROUGE Scores for Task 1: Email Subject Line Generation:



ROUGE Scores:

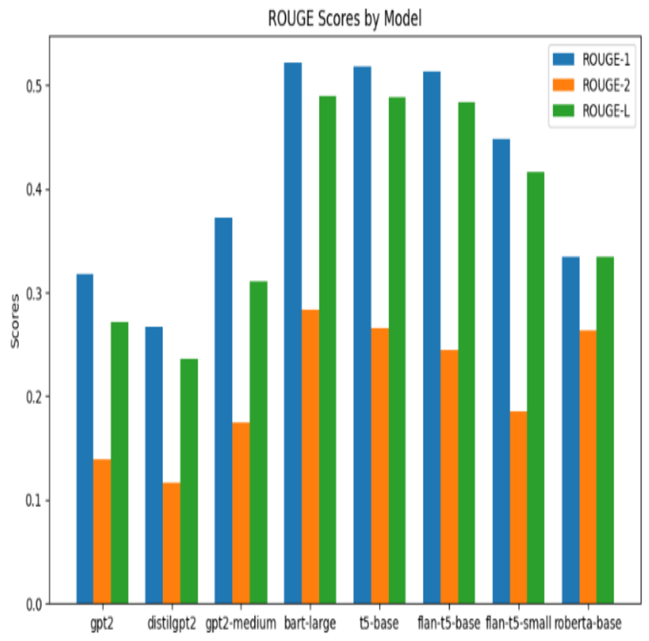
Model	ROUGE-1	ROUGE-2	ROUGE-L
GPT-2	0.1409	0.0353	0.1361
distilgpt2	0.4598	0.2991	0.4485
GPT2-Medium	0.4799	0.3165	0.4652
BART Base	0.3738	0.2444	0.3665
BART Large-CNN	0.3257	0.2800	0.3257
T5 Small	0.1835	0.0866	0.1732
T5 Base	0.0985	0.0353	0.0959

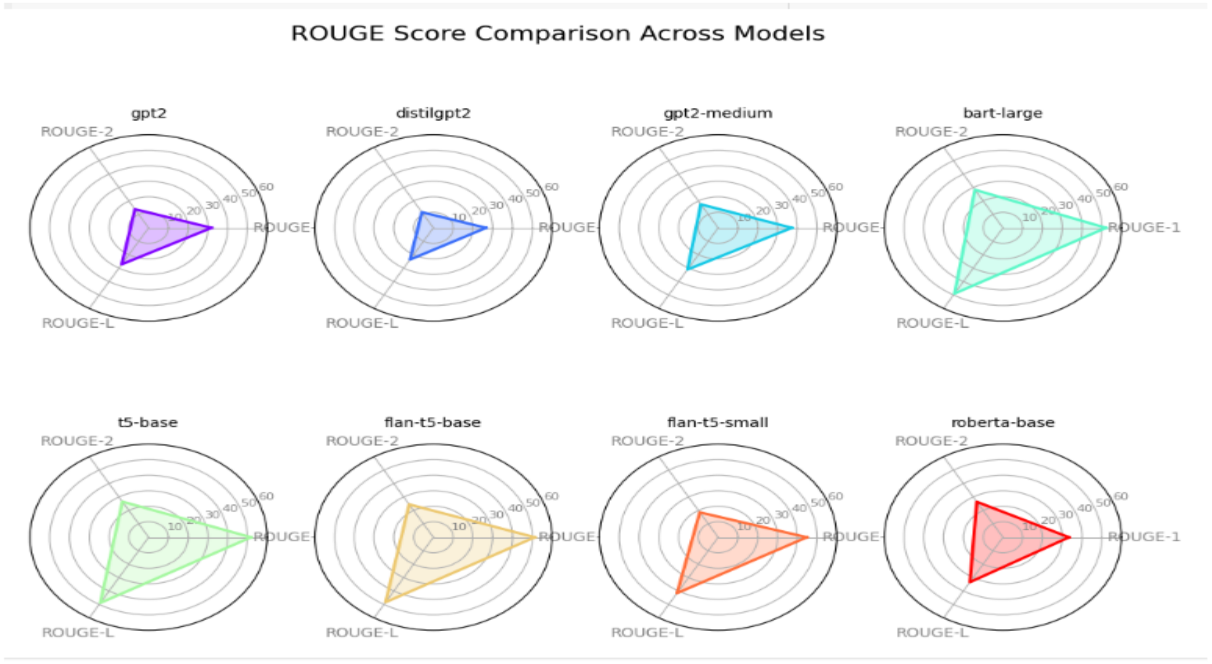




ROUGE Scores for Task 2: Question Answering on AIML Queries

Model	ROUGE-1	ROUGE-2	ROUGE-L
gpt2	0.3183	0.1386	0.2713
distilgpt2	0.2665	0.1163	0.2363
gpt2-medium	0.3728	0.1749	0.3103
bart-large	0.5213	0.2832	0.4892
t5-base	0.5180	0.2653	0.4888
flan-t5-base	0.5128	0.2448	0.4832
flan-t5-small	0.4484	0.1855	0.4157
roberta-base	0.3340	0.2638	0.3340





Output for few Emails for Task 1: Email Subject Line Generation:

1. **Email**: The following reports have been waiting for your approval for more than 4 days.Please review.Owner: James W Reitmeyer Report Name: JReitmeyer 10/24/01 Days In Mgr.Queue: 5

   Generated Subject: Following report waiting approval

2. **Email**: Late on October 25th, we received information about a nonspecific threat to the Enron Center.We communicated with law enforcement officials who found the threat unsubstantiated and without merit.Nonetheless we take all threats seriously and have increased the security presence at the Enron Center still further.Once again, if you observe suspicious behavior, please call security at 3-6200.

   Generated Subject: Late october 25th received

Answers for few Questions for Task 2: Question Answering on AIML Queries:

1. **Question**: Can we utilize an autoencoder to perform dimensionality reduction on numerical datasets?

   Generated Answer: Yes, autoencoders can be used for dimensionality reduction by utilizing their inherent feature representations to reduce the dimensionality of numerical datasets.

2. **Question**: What is NLP's current biggest challenge that is being tried to overcome ?

   Generated Answer: ANLP challenges are often difficult to solve because the language itself is not well-structured.

Tools and Frameworks

List the tools and frameworks used for this project:

- **Hugging Face Transformers**: For accessing pre-trained models like GPT-2.
- **PyTorch and TensorFlow**: For model training.
- **Keras**: Optional for high-level modeling.
- **NLTK**: For tokenization and text preprocessing.
- **Gradio, FastAPI and NextJS**: For deployment.
- **Github Code spaces, Vercel**: For cloud computing and hosting the model.

Deployment Strategy

Model Deployment to Hugging Face:

- Fine-tuned model was pushed to the Hugging Face Model Hub.
- Deployed the fine-tuned model using a Gradio app on Hugging Face Spaces, enabling an interactive interface for users to test the model's capabilities.

API Development with FastAPI:

- Created a FastAPI backend to serve the fine-tuned model using Hugging Face's transformers pipeline for inference.
- Integrated the FastAPI API to handle model predictions through simple API calls, offering a robust and scalable solution for handling requests.

UI Development with Next.js:

- Built the front-end interface using Next.js, creating a user-friendly interface for interacting with the model via the FastAPI backend.
- Ensured seamless integration of the FastAPI API for model predictions within the Next.js interface, enabling users to input data and view responses in real-time.
-
    - **FastAPI** backend was deployed in **GitHub Codespaces**, offering a cloud-based development environment and scalable backend deployment.
    - **Next.js** front-end was deployed on **Vercel**, taking advantage of its seamless integration with Next.js for hosting and continuous deployment.

This setup provides a robust and scalable architecture, leveraging Hugging Face, Gradio, FastAPI, and Next.js for model deployment, API interaction, and user interface.

References and Research Papers

A Comprehensive Overview of Large Language Models

This paper provides a detailed survey on the evolution and development of large language models (LLMs). It covers essential aspects such as architectures, pre-training techniques, fine-tuning strategies, multi-modal models, and evaluation metrics. The authors also highlight key design decisions and training optimizations, such as tokenization, attention mechanisms, and distributed training approaches like data parallelism and model parallelism. The paper offers insights into the challenges of scaling LLMs and future research directions, making it a useful guide for both researchers and practitioners looking to leverage LLM technology(ar5iv) .

Reference: https://ar5iv.labs.arxiv.org/html/2307.06435

The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs:

This paper presents an extensive guide to fine-tuning large language models, from foundational concepts to advanced techniques. It explores different fine-tuning methods, such as prompt-based, instruction-based, and multi-task fine-tuning, addressing practical challenges like data scarcity, model overfitting, and generalization issues. The guide includes use cases where fine-tuning has shown breakthroughs, especially in domain-specific tasks. It also offers best practices for choosing datasets, hyperparameter optimization, and applying transfer learning techniques to maximize performance.

Reference: https://ar5iv.labs.arxiv.org/html/2408.13296

When Scaling Meets LLM Finetuning:

This research discusses the impact of scaling LLMs during the fine-tuning process. It examines how scaling up the number of parameters and training data affects model adaptability to downstream tasks. The paper highlights challenges such as catastrophic forgetting and the computational costs of scaling. It also compares various finetuning techniques, showing that larger models benefit more from data-efficient methods, which reduce the need for vast amounts of labeled data. The authors present empirical results on how scaling can enhance performance but stress that it requires careful balancing of resources(ar5iv) .

Reference: https://ar5iv.labs.arxiv.org/html/2402.17193