AI-based Generative QA System

Project Report - Group 16

Paramasivan Dorai, Naga Maruthi Vangara, Jagadeesh Vana, Arun Ganireddi

September 28, 2024

Mentors:

- Mr. Nayan Anand
- Mr. Lokesh Madasu

Contents

1 Introduction

In this project, we explored the capabilities of generative models, specifically GPT-variant models, to solve two distinct text generation tasks:

- **Task 1**: Generating succinct email subject lines based on the email body.
- **Task 2**: Modeling a system to answer AIML-related questions based on a fine-tuned GPT model.

These tasks allowed for hands-on experience in dataset curation, model finetuning, and deployment of generative models. The project was completed over two phases, each focusing on one of the tasks.

2 Task 1: Email Subject Line Generation

2.1 Problem Description

The first task focused on generating concise subject lines from the body of an email. This problem requires the model to identify the most salient sentences within the email body and summarize them into a few words. The task is challenging because the generated subject must be informative and brief, typically only containing a few words while encapsulating the essence of the email's content.

2.2 Dataset

We used the **AESLC (Annotated Enron Subject Line Corpus)** dataset, available on GitHub AESLC. The dataset includes cleaned, filtered, and deduplicated emails from the Enron Email Corpus and is split into training, development, and test sets:

- **Training set**: 14,436 emails
- **Development set**: 1,960 emails
- **Test set**: 1,906 emails
- **Average email length**: 75 words
- **Average subject length**: 4 words

Each email in the development and test datasets contains multiple annotations for the subject line, denoted as `@subject`, `@ann0`, `@ann1`, and `@ann2`. The training dataset only contains a single subject line (`@subject`).

2.3 Data Preprocessing

We performed the following preprocessing steps on the data:

- Removed unnecessary spaces and replaced multiple spaces with a single space.
- Filtered out non-word and non-space characters.
- Extracted the content before `@subject` as the `email_body`.
- Extracted `@subject`, `@ann0`, `@ann1`, and `@ann2` as potential subject lines.
- Combined the datasets into single files for training, development, and testing, with the following columns: `email_body`, `subject_line`, `subject_line1`, `subject_line2`, and `subject_line3`.

Since the average email body contains 2500 characters, we limited the email body to the first 2500 characters for training the models.

2.4 Models Used

We fine-tuned and evaluated several models for the task. The following table provides an overview of the models used:

Model	Training Notebook
BART	Capstone_Group_16_BART_WandB.ipynb
T5 Small	email_sub_gen_T5_small.ipynb
DistilGPT2	Capstone_Group_16_DistilGPT2_WandB.ipynb
GPT2	email_sub_gen_gpt2_latest.ipynb

Table 1: Models used for training and testing

2.5 Zero-Shot Testing

We conducted zero-shot testing on each of the models. The common issue across the models was the generation of overly long subject lines. To address this, we experimented with various hyperparameters, including:

- `length_penalty`
- `max_new_tokens`
- `max_length` and `min_length`
- `num_beams`

For example, given the following email body:
Hi Judy How are you I sent Ingrid an email and she told me that you are still at Enron That's good I tried to call you at work but the girl that answered the phone said that you were on vacation today How are the kids and Rob I think about you all of the time Ingrid told me that she told you that I got married It isn't Tod it is James Johnson that I dated in high school ...

The expected subject line was *Latest Marketing List*, but the generated summary was *Laurie writes to her ex-boyfriend from high school. She wants to know if he is still at Enron. She also wants to meet up for lunch.*

2.6 Model Fine-Tuning and Evaluation

We fine-tuned the four models and observed that the decoder-only models (e.g., GPT2 and DistilGPT2) generated creative but often irrelevant subject lines. In contrast, the encoder-decoder models (e.g., BART and T5) performed significantly better in generating concise and relevant subject lines.

2.7 Performance Metrics

We evaluated the performance of each model using the ROUGE score, which measures the overlap between the generated and reference subject lines. The following table provides a comparison of the ROUGE scores across different models:

Model	ROUGE 1	ROUGE 2	ROUGE L
BART	0.3007	0.1517	0.2885
T5 Small	0.2756	0.1322	0.2671
DistilGPT2	0.1066	0.0418	0.1023
GPT2	0.0151	0.0058	0.0132

Table 2: ROUGE score comparison for different models

2.8 Model Weights

The fine-tuned weights for each of the models were saved on Hugging Face Spaces for further use:

- **BART**: BART on Hugging Face
- **T5 Small**: T5 Small on Hugging Face
- **DistilGPT2**: DistilGPT2 on Hugging Face
- **GPT2**: GPT2 on Hugging Face

2.9 Observations and Further Reading

We observed that BART and T5 models outperformed GPT2 and DistilGPT2 models due to their encoder-decoder architectures, which allowed them to better understand and generate concise, relevant subject lines. Below is a comparison of the models:

	BART	**T5**	**GPT2-small**	**DistilGPT2**
Developed By	Facebook	Google	OpenAI	OpenAI
Architecture	Encoder-decoder	Encoder-decoder	Decoder-only	Distilled decoder-only
Training Objective	Denoising autoencoder	Text-to-text transfer	Autoregressive	Distilled GPT
Common Use Cases	Summarization, QA	Text generation, QA	Text generation	Faster inference

Table 3: Model comparison

2.10 Gradio App Integration

Initially used a simple Colab notebook Gradio for review. Final version of the application published to Hugging Face Gradio Spaces.

- **BART**: BART Gradio App
- **T5:** T5 Gradio App

3 Task 2: AIML Question Answering

3.1 Problem Description

The second task focuses on fine-tuning a GPT model to answer questions specific to the AIML course. The objective is to generate accurate and relevant answers based on domain-specific knowledge related to AI and Machine Learning.

3.2 Dataset

We used the dataset provided by the Talentspirit team, which can be found at the following links:

- Dataset-1
- Dataset-2

3.3 Data Preprocessing

The preprocessing for this task involved formatting the training dataset into an appropriate prompt format. We experimented with two prompting styles:

- **GPT-style prompting**
- **Alpaca-style prompting**

The following figures show the visualization of the training and test datasets.

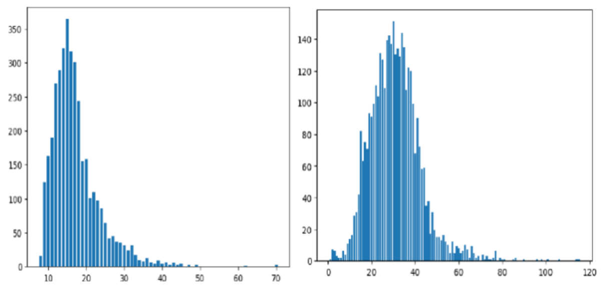


Figure 1: Visualization of the training dataset

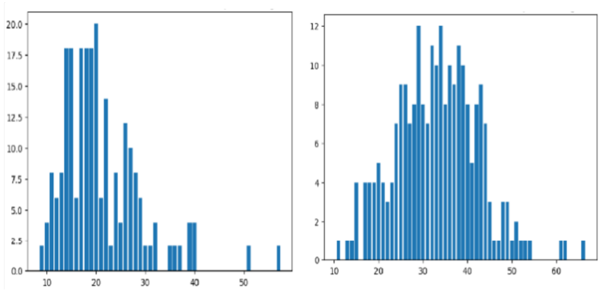


Figure 2: Visualization of the test dataset

3.4 Models

We fine-tuned the following models for answering AIML-related questions:

Model	Training Notebook
GPT2	QnA_finetuning_gpt2_V4.ipynb
Llama 2	Capstone_Group16_QnA_finetuning_LLAMA2_V1.ipynb
Gemma 2	QnA_gemma_2_2b_v5.ipynb
Llama 3	Capstone_Group16_QnA_finetuning_LLAMA3_V1.ipynb

Table 4: Models used for training and testing

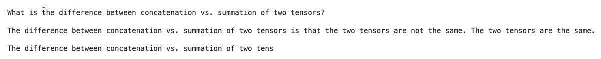
3.5 Zero Shot Testing

We performed zero-shot testing on each of the models before fine-tuning, and several challenges were identified:

- **Hallucinations**: Models generated irrelevant and repetitive information, such as the following:

  Question: What is AlexNet?
  Generated Answer: AlexNet is a web application that allows you to create and manage your own websites. It is a web application that allows you to create and manage your own websites.

- **Repetitive Answers**: The models often repeated parts of the answer. See the visualization below:



```
What is the difference between concatenation vs. summation of two tensors?

The difference between concatenation vs. summation of two tensors is that the two tensors are not the same. The two tensors are the same.

The difference between concatenation vs. summation of two tens
```

Figure 3: Repetitive answers in zero-shot testing

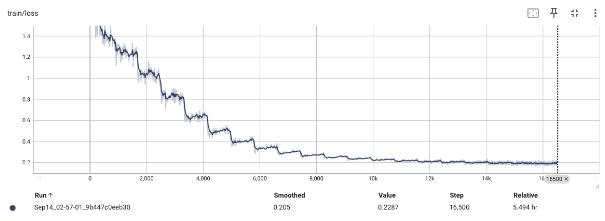
- We focused on hyperparameters such as:
  - `Prompt Structure`

– `Max New Tokens`

– `Max Length` and `Min Length`

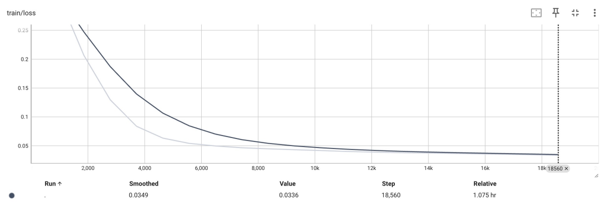
3.6 Model Fine-Tuning and Training

We fine-tuned and tested four models: GPT2, Llama 2, Gemma 2, and Llama 3. Initially, we tried encoder-decoder models, but we shifted to decoder-only models to improve performance on unseen topics.



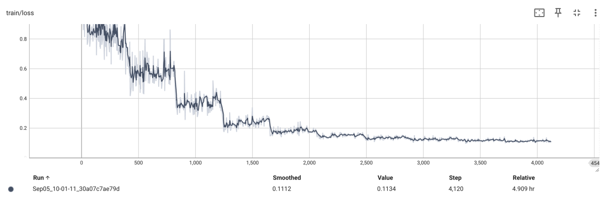
	Run ↑	Smoothed	Value	Step	Relative
●	Sep14_02-57-01_9b447c0eeb30	0.205	0.2287	16,500	5.494 hr

Figure 4: Llama 2 - 20 Epochs Training Loss



	Run ↑	Smoothed	Value	Step	Relative
●	.	0.0349	0.0336	18,560	1.075 hr

Figure 5: Gemma 2 - 20 Epochs Training Loss



	Run ↑	Smoothed	Value	Step	Relative
●	Sep05_10-01-11_30a07c7ae79d	0.1112	0.1134	4,120	4.909 hr

Figure 6: Llama 3 - 20 Epochs Training Loss

3.7 Model Weights

We saved all the fine-tuned model weights in Hugging Face Spaces:

- **GPT2**: GPT2 for Q&A
- **Llama 2**: Llama 2 for Q&A
- **Gemma 2**: Gemma 2 for Q&A
- **Llama 3**: Llama 3 for Q&A

3.8 ROUGE Score Comparison

The following table presents the ROUGE score comparison for each model:

3.9 ROUGE Score Inference

1. Model Size and Capacity:

Model	ROUGE 1	ROUGE 2	ROUGE L
GPT2	0.3696	0.1639	0.3128
Llama 2	0.3912	0.1807	0.3158
Gemma 2	0.4551	0.2290	0.3845
Llama 3	0.4834	0.2632	0.4149

Table 5: ROUGE score comparison for the Q&A task

- Larger models, such as Llama 3 (8 billion parameters) and Gemma 2 (2 billion parameters), performed better due to their ability to capture complex language patterns and generalize to topics not covered in the training set.
- GPT2 Small (124 million parameters) struggled with generalization, particularly in multi-word sequences (ROUGE-2) and sentence-level coherence (ROUGE-L).

2. Architecture Advancements:

- Llama 3, with its advanced architecture and optimizations, generated more coherent responses.
- Gemma 2 showed improvement over GPT2 due to its more modern architecture, but Llama 3 outperformed in most metrics.
- Llama 2 also performed well, but it did not reach the fine-tuning efficiency of Llama 3.

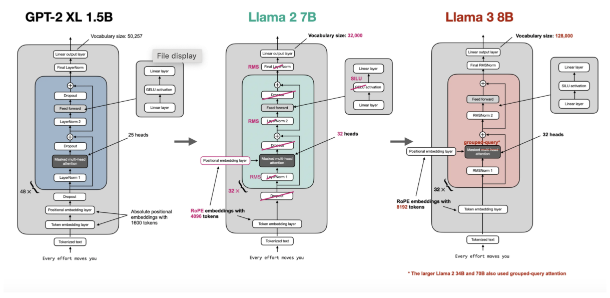
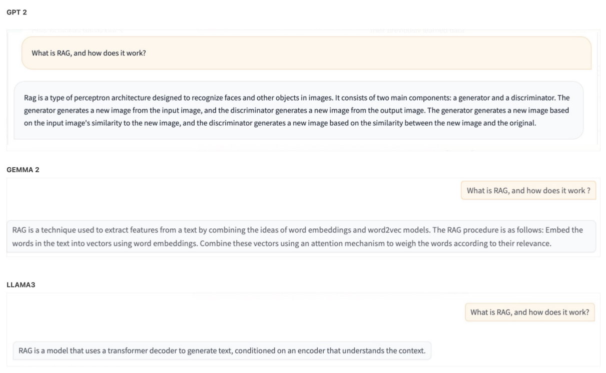


Figure 7: Architecture Advancement

3. Generalization on unseen topics:

- The ROUGE score differences can also be attributed to how well each model generalizes to topics not covered in the training data.
- Llama 3 shows the best ability to generalize beyond the AI/ML domain, likely due to its large size, architectural sophistication, and fine-tuning efficiency.
- Gemma 2 and Llama 2 also generalize well but not as effectively as Llama 3.
- GPT-2 Small struggles the most with out-of-domain generalization, leading to significantly lower scores.
- The following sample question is an interesting example, the training data does not contain anything related to RAG



GPT 2

What is RAG, and how does it work?

Rag is a type of perceptron architecture designed to recognize faces and other objects in images. It consists of two main components: a generator and a discriminator. The generator generates a new image from the input image, and the discriminator generates a new image from the output image. The generator generates a new image based on the input image's similarity to the new image, and the discriminator generates a new image based on the similarity between the new image and the original.

GEMMA 2

What is RAG, and how does it work ?

RAG is a technique used to extract features from a text by combining the ideas of word embeddings and word2vec models. The RAG procedure is as follows: Embed the words in the text into vectors using word embeddings. Combine these vectors using an attention mechanism to weigh the words according to their relevance.

LLAMA3

What is RAG, and how does it work?

RAG is a model that uses a transformer decoder to generate text, conditioned on an encoder that understands the context.

Figure 8: Performance on Unseen topics

4 Conclusion

It was a great learning experience fine tuning these transformer models to perform a specific task. The nuances of encoder-decoder vs decoder-only models and how it could influence the performance was a great learning experience.

There were engineering issues that these large models caused and we had to figure out ways to deal with them along the way. This task has increased our understanding multifold.