qM-AI-L

Email Subject Generation

&

Question Answering

PGCP Capstone Project

AIML-B22-Group-15

Lohith Reddy Manchireddy

Ravi Kanth Jami

Shilpa Sirikonda

Sudheendra Gopinath

[Introduction 4](#_Toc177752703)

[Email Subject Generation 4](#_Toc177752704)

[Dataset 4](#_Toc177752705)

[Data Loading and Pre-Processing 4](#_Toc177752706)

[Methodology 4](#_Toc177752707)

[Zero Shot Inferencing 5](#_Toc177752708)

[Finetuning Models 5](#_Toc177752709)

[Model Prompts – Illustrations 7](#_Toc177752710)

[Human Evaluation of Fine-tuned Models 8](#_Toc177752711)

[Quantitative Evaluation of Fine-tuned Models 9](#_Toc177752712)

[Granularity 9](#_Toc177752713)

[Context 9](#_Toc177752714)

[Summarization 9](#_Toc177752715)

[Evaluation against given subject lines 10](#_Toc177752716)

[Evaluation of Fine-tuning the pre-trained model 11](#_Toc177752717)

[Tuning the hyperparameters 13](#_Toc177752718)

[Observations & Key Learnings 13](#_Toc177752719)

[Deployment 14](#_Toc177752720)

[Artefacts 14](#_Toc177752721)

[Question Answering 15](#_Toc177752722)

[Model Approaches 15](#_Toc177752723)

[Dataset and Data Preparation 15](#_Toc177752724)

[Models' Selection 16](#_Toc177752725)

[Prompt Structure and Formatting 16](#_Toc177752726)

[Fine Tuning 16](#_Toc177752727)

[Environment Setup 16](#_Toc177752728)

[Model Loading 16](#_Toc177752729)

[Fine Tuning Configuration 17](#_Toc177752730)

[Training 17](#_Toc177752731)

[Evaluation and performance 17](#_Toc177752732)

[Dataset-1 17](#_Toc177752733)

[ROUGE Metrics 18](#_Toc177752734)

[Dataset-2 19](#_Toc177752735)

[ROUGE Metrics 20](#_Toc177752736)

[Fine Tuning Performance 20](#_Toc177752737)

[Save and Deploy 21](#_Toc177752738)

[Artefacts 21](#_Toc177752739)

# Introduction

qM-AI-L is a project to evaluate and compare the performance of various pre-trained generative AI models in NLP on two distinct tasks:

* given an email body, generate a succinct subject line for it
* answer technical questions on AI / ML.

# Email Subject Generation

qM-AI-L identifies the most salient words, phrases and sentences from the given email body and abstracts the message contained in that set into a very short, impactful subject line.

## Dataset

The pre-trained models are fine-tuned using the "[Annotated Enron Subject Line Corpus](https://github.com/ryanzhumich/AESLC)" dataset.

* The dataset consists of a subset of cleaned, filtered and deduplicated emails from the Enron Email Corpus which consists of employee email inboxes from the Enron Corporation.

## Data Loading and Pre-Processing

* LangChain\_community.DirectoryLoaders are used to load the email files and then converted to Pandas DataFrame.(LangChain document\_loader was found to be organized, scalable, easy to use)
* Evaluation (dev, test) split of the data contains 3 annotated subject lines by human annotators. Multiple possible references facilitate a better evaluation of the generated subject, since it is difficult to have only one unique, appropriate subject per email
* Some dataset statistics:
  + Sizes of train / dev / test splits: 14,436 / 1,960 / 1,906
  + An email contains an average of 75 words
  + A subject contains an average of 4 words
* A subset of train dataset is created for finetuning language models, although full train data set is also used a couple of times.

## Methodology

* On high level different Open Source language models are researched and assessed that suits the problem statement of extracting most important words / context / concise summarization. Transformer models and Bart models were found to be most apt for the given task other than the ChatGPT models.
* Couple of pretrained models were selected to test with zero-shot inferencing and further finetuning

## Zero Shot Inferencing

* Several models were loaded directly from hugging face and random records were inferenced to see how the models were behaving.
* [facebook/bart-base](https://huggingface.co/facebook/bart-base), [FLAN-T5](https://huggingface.co/docs/transformers/en/model_doc/flan-t5#overview) [Gemma-7b](https://huggingface.co/unsloth/gemma-7b-bnb-4bit), [unsloth](https://huggingface.co/unsloth)/[mistral-7b](https://huggingface.co/unsloth/mistral-7b), Phi-3 models were tried.
* By testing with various models with the Zero Shot Inferencing, we could see that the model struggles to extract the same subject line compared to the human baseline subject, but it does pull out some important information from the email which indicates the models can be fine-tuned to the task at hand.

Below is an example with Flan-T5 Base model.

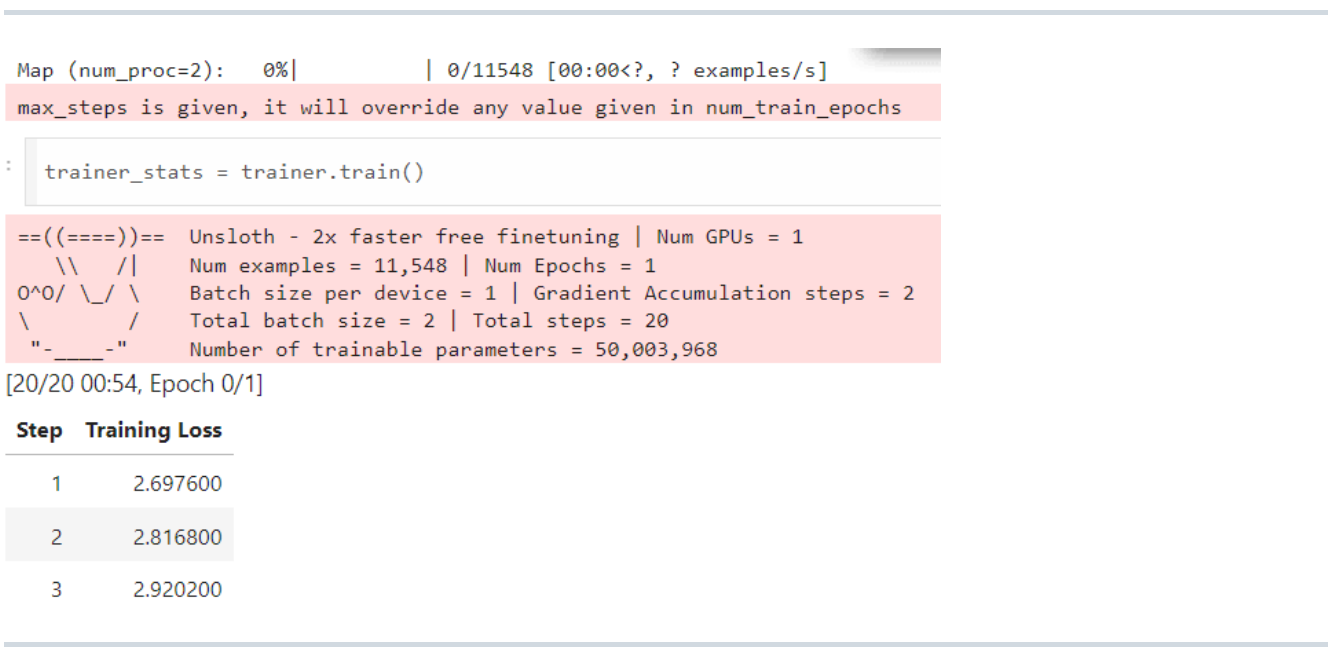
A screenshot of a computer program

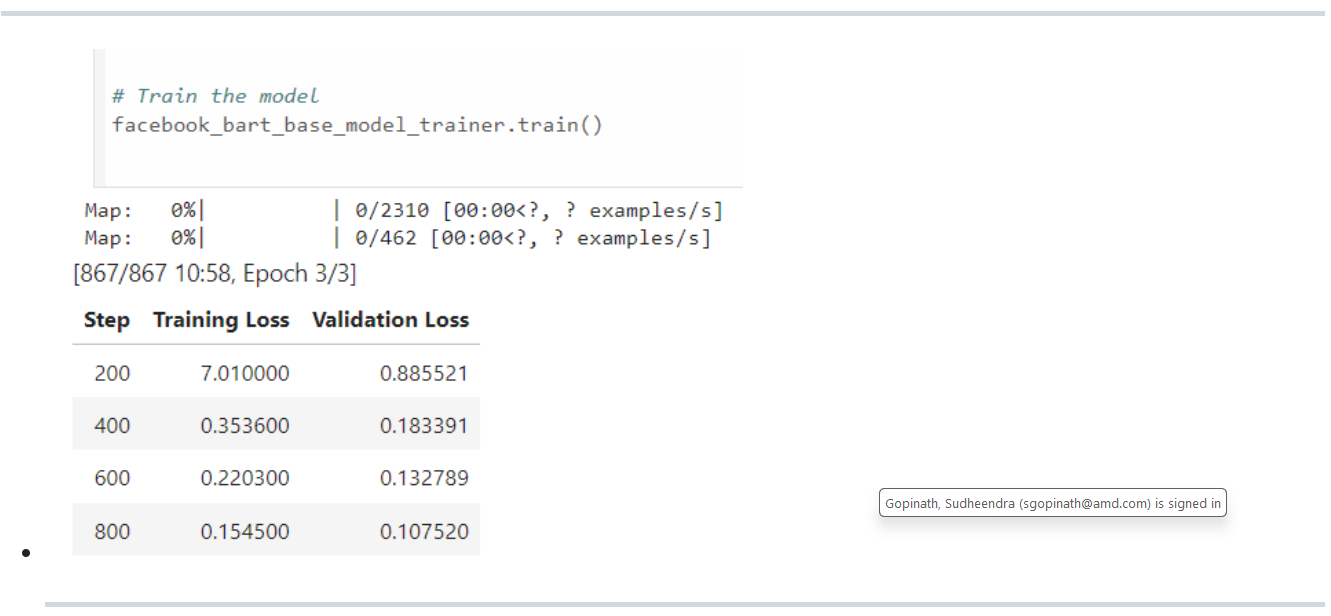
Description automatically generated

## Finetuning Models

* Utilize the built-in Hugging Face/ SFTTrainer Trainer class. Pass the preprocessed dataset with reference to the original pretrained model. Several training parameters are tweaked and explored experimentally.
* Training a fully fine-tuned version of the basic/ small model takes a few hours on a GPU. To save time, several checkpoints were created, and the fully fine-tuned model were then initialized to use in the rest of experiments.

Some snapshots of the training process with the given data after pre-processing are below.





A screenshot of a computer

Description automatically generated

A screenshot of a computer

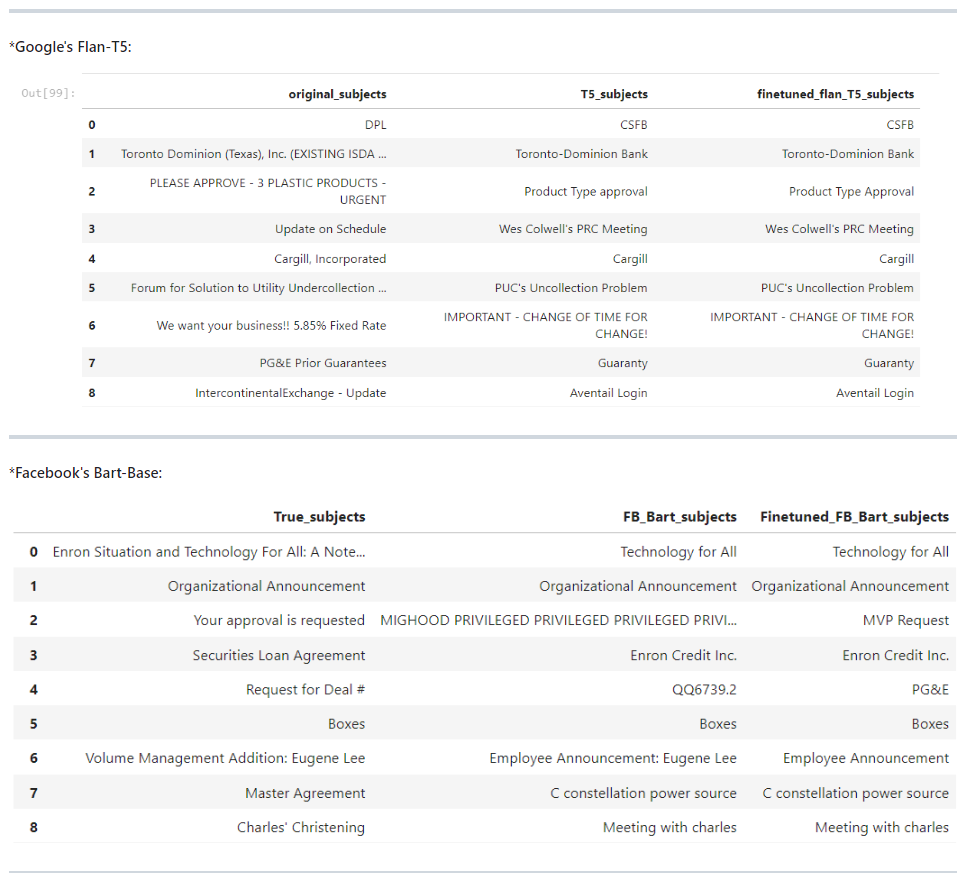
Description automatically generated

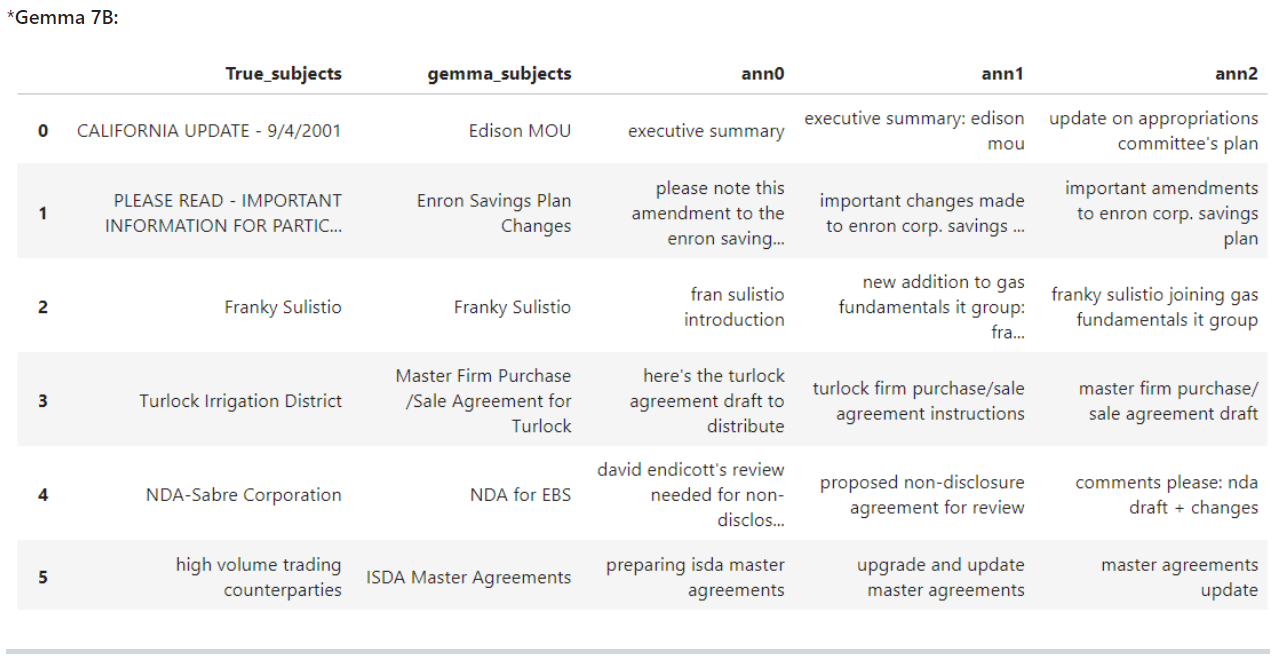
## Model Prompts – Illustrations

|  |  |  |
| --- | --- | --- |
| **Model** | **Prompt** | |
| [FLAN-T5](https://huggingface.co/docs/transformers/en/model_doc/flan-t5) | Email-Subject (prompt-input-response) format is created as explicit  instructions for the LLM. Prepend a prompt instruction to the start of  email body and generate the subject with Subject as follows:  Training prompt (email):  prompt = f""" Generate a subject line for the following email.  Email: {email}  Subject:  """ | |
| [google-gemma-with-unsloth](https://www.analyticsvidhya.com/blog/2024/04/fine-tuning-google-gemma-with-unsloth/) | instruction = "Generate a subject line for the following email."  if x['body']:  formatted\_text = f"""Below is an instruction that describes a task. \  Write a response that appropriately completes the request.  ### Instruction:  {instruction}  ### Input:  {x['body']}  ### Response:  {x['subject']}""" | Instruction = “Generate a subject line for the following email.”  If x[‘body’]:  formatted\_text = f”””Below is an instruction that describes a task. \  Write a response that appropriately completes the request.  ### Instruction:  {instruction}  ### Input:  {x[‘body’]}  ### Response:  “”” |

## Human Evaluation of Fine-tuned Models

The performance of the fine-tuned models was qualitatively by comparing their ability to generate a reasonable subject line against its original subject, to asses if they are behaving the way they are supposed to, and are able to understand the input. This approach confirmed that the fine-tuned models behave as expected.





## Quantitative Evaluation of Fine-tuned Models

The ROUGE metric helps quantify the validity of subject lines produced by models. It compares subjects to an "annotated baseline" subject which is usually created by a human. While not perfect, it does indicate the overall increase in subject line generation effectiveness that we have accomplished by fine-tuning. The quantification is done on several dimensions such as below.

### Granularity

* ROUGE-1 focuses on individual words
* ROUGE-2 on word pairs
* ROUGE-L on the longest sequence of words

### Context

ROUGE-2 captures context better than ROUGE-1 due to its consideration of word pairs, while ROUGE-L and ROUGE-Lsum capture the overall sentence structure.

### Summarization

ROUGE-Lsum is specifically designed for summarization, making it more relevant for evaluating the quality of summaries compared to ROUGE-L, which can be applied more generally.

Bleu measures the precision: how much the words (and/or n-grams) in the machine generated summaries appeared in the human reference summaries.

Rouge measures the recall: how much the words (and/or n-grams) in the human reference summaries appeared in the machine generated summaries.

Rouge scores were evaluated for individual records and also as averages across all records.

Below table is evaluation for individual records.

A screenshot of a computer program

Description automatically generated

Below are the average-wise scores with Facebook/Bart model.

A screen shot of a computer code

Description automatically generated

### Evaluation against given subject lines

The ROUGE scores were calculated w.r.t the original subjects and the three given human annotations.

A screenshot of a table

Description automatically generated

### Evaluation of Fine-tuning the pre-trained model

The absolute percentage improvement of finetuned model over the pretrained model was also calculated.

A computer screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

### Tuning the hyperparameters

Below are the key arguments used for training Gemma-7b.

* **per\_device\_train\_batch\_size**: It is set to 1, meaning 1 examples will be processed per device in each step.
* **gradient\_accumulation\_steps:** Grad Accumulation steps before performing a parameter update. Increases the batch size by accumulating gradients over multiple steps. Here, it is set to 2, meaning gradients will be accumulated over 2 steps before updating the model parameters.
* **warmup\_steps**: This sets the number of warm-up steps during training, gradually increasing the Learning Rate from 0 to the provided value. We set to 5, so the Learning Rate will linearly increase over the first 5 steps.
* **max\_steps:** Total number of training steps to perform. Here, it is set to 50, meaning the training will stop after 50 steps.
* **learning\_rate**: First Learning Rate used for training. We set it to 2e-4
* **fp16 and bf16**: Control the precision used for training. fp16 is for half-precision (16-bit) training, while bf16 is for bfloat16 training if GPU supported.
* **logging\_steps**: Sets the interval at which training metrics and losses are logged. We set it to 1, so logs are printed after every training step.
* **optim:** Optimizer to use for training. We set it to ‘paged\_adamw\_8bit’, a specialized optimizer for memory-efficient training.
* **weight\_decay**: Weight Decay Rate that we need for regularization. Set to 0.01.
* **lr\_scheduler\_type:** Learning Rate Scheduler to use during training, "linear“.

## Observations & Key Learnings

* Fine-tuned models show performance improvement. They are effectively capturing key points and overall essence, with improved ROUGE-1 scores showing alignment with essential topics.
* The models demonstrated potential for **understanding nuanced details**, as indicated by ROUGE-2 scores, though there is room for improvement.
* Higher ROUGE-L and ROUGE-Lsum scores reflect good maintenance of subject length and relevance.
* Specific prompts, such as "generate a subject line" yield better results compared to combined prompts like "summarize the text“.
* Repetitive responses in pre-trained models (e.g., Mistral) are managed by applying a repetition\_penalty of 1.5, but excessive penalties cause unusual outputs.
* Phi3 excels in text completion and GPT-style conversations but may produce hallucinations and less accurate results.

## Deployment

* **Build the Gradio App**: Designed Gradio interface, defining how the user will interact with the model and ensuring the input and output specifications are clear.
* **Save the App and Dependencies**: Prepared our app script and ensure all necessary dependencies are listed in a requirements file, ready for deployment.
* **Publish on Hugging Face Spaces**: Created an account on Hugging Face, set up a new Space for our app, and push our code to this Space, making our app publicly accessible.

## Artefacts

|  |  |
| --- | --- |
| **Description** | **Link** |
| Github | <https://github.com/nutworker/qM-AI-L> |
| Deployment | <https://huggingface.co/ssirikon/Gemma7b-bnb-Unsloth>  <https://huggingface.co/Lohith9459/gemma7b> |
| Gradio | <https://huggingface.co/spaces/ssirikon/Gradio2-SubjectGen> |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Question Answering The QnA task is to generate free-form responses that require not only finding relevant information from its training knowledge but also synthesizing this information into multiple accurate answer sentences. Model Approaches Both **extractive** and **sequence-to-sequence** approaches were explored for the given problem statement and fine-tuning. After careful evaluation, the Sequence-to-Sequence (Seq2Seq) approach was selected for the task due to following reasons:   * AI/ML knowledge corpus usage was not recommended to use * Flexible Output Generation: Seq2Seq models generate new sequences, unlike extractive models, which are restricted to selecting text spans, making them ideal for tasks like summarization or translation.  Dataset and Data Preparation  * Input-Output Pairs: For Seq2Seq fine-tuning the model is provided with structured input-output pairs, where:   + Input: The question without a context   + Output: The target answer. * A total of 462 question-answer pairs were collaboratively prepared from the AIML course. The dataset was coalited in the prescribed format in the CSV file. * A consolidated train/dev/test set was provided for further fine-tuning with the GPT variant model. * Dataset-1 has a question-and-answer pair for train set and, a question and two human annotated answers for the test and dev sets.   + Train set -(1316, 2)   + Test set (120, 3)   + Dev set (80, 3) * Dataset-2 has a question-and-answer pair   + Train set -(1985, 2)   + Test set -(249, 2)   + Dev set - (248, 2))  Models' Selection GPT-2 medium, Gemma 7B and Llama 3 8B were used for finetuning |  |  |  |  |
|  |  |  |  |  |
| Prompt Structure and Formatting  * Added Clear Instructions: Specified format, tone, or length to guide the model’s response. * Provideed Context: Includeed relevant details to anchor the model's answer closer to the reference. * Useed Examples: Provided sample responses to show the desired structure and style. |  |  |  |  |
| Fine Tuning |  |  |  |  |

### Environment Setup

* Necessary libraries (e.g., PyTorch, Hugging Face Transformers) were set up
* Necessary GPU/TPU resources were made available for handling large models efficiently.

### Model Loading

* Load the pretrained model (Gemma 7B or LLaMA 8B) from Huggingface using FastLanguageModel/ SFTTrainer class
* Create a PEFT model with the given parameters and load adapters - LoRA (Low-Rank Adaptation) for parameter-efficient fine-tuning with following parameters
  + r=16, # LoRa Rank
  + target\_modules=["q\_proj", "k\_proj", "v\_proj", "o\_proj",
  + "gate\_proj", "up\_proj", "down\_proj",],
  + lora\_alpha=16,
  + lora\_dropout=0,
  + bias="none",
  + use\_gradient\_checkpointing=True

### Fine Tuning Configuration

Hyperparameters are defined such as learning rate, batch size, and max sequence length.

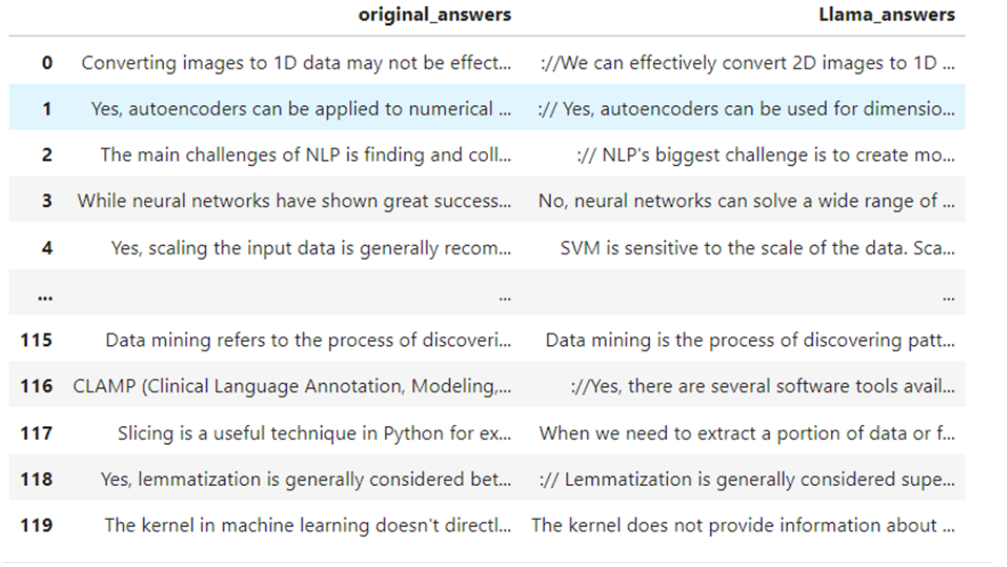
## Training

* Perform backpropagation to adjust model weights based on the task-specific loss function.
* Monitor model performance (validation loss, accuracy) during training to prevent overfitting.
* TrainingArguments used as below
  + per\_device\_train\_batch\_size = 1,
  + gradient\_accumulation\_steps = 2,
  + warmup\_steps = 5,
  + max\_steps = 30,
  + learning\_rate = 2e-4,
  + fp16 = not torch.cuda.is\_bf16\_supported(),
  + bf16 = torch.cuda.is\_bf16\_supported(),
  + logging\_steps = 1,
  + optim = "paged\_adamw\_8bit",
  + weight\_decay = 0.01,
  + lr\_scheduler\_type = "linear",
  + seed = 3407,
  + output\_dir = "outputs",

## Evaluation and performance

* Validate the model on a test set to check its performance and generalization ability.
* Hyperparameters are adjusted and retrained until desired performance is reached.

## Dataset-1



A screenshot of a computer

Description automatically generated

### ROUGE Metrics

A table with numbers and text

Description automatically generated

## Dataset-2

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### ROUGE Metrics

A table of data tables

Description automatically generated

## Fine Tuning Performance

A graph of performance

Description automatically generated

## Save and Deploy

* Fine-tuned model and tokenizer are saved for evaluation, inferencing and deployment
* Deployed the model to production or use it for inference for generating answers

## Artefacts