TEXT SUMMARIZATION

Mentor: Mr. Narendra Kumar

Intern: Mr. Mohan Krishna G R,

III – M.Tech., CSE (Integrated 5 Years),

Sri Ramakrishna Engineering College,

Coimbatore, Tamil Nadu.

Author Note

E-Mail id: mohankrishna.2253035@srec.ac.in, mohankrishnagr08@gmail.com

GitHub Repo: Text Summarization ; LinkedIn: Profile ; Contact Number: +91 908069 6129

ACKNOWLEDGEMENT

I ***Mohan Krishna G R***, would like express my sincere gratitude to ***Infosys*** that has given me this wonderful opportunity to work as an Artificial Intelligence (AI/ML) Intern, Summer 2024 at ***Infosys Springboard***. This internship has been very rewarding for me in terms of the knowledge that I have gained from it as well as its contribution towards my professional development.

I would also like to express my deep gratitude and I am thankful to ***Mr. Narendra Kumar***, my mentor at Infosys for his guidance which is priceless, feedbacks that are insightful and encouragement which never stops. His expertise and assistance have been crucial in shaping this project.

I would like to thank all those who supported me throughout my internship period and helped me with completing this project. Moreover, I am grateful to the members of my team in ***group 4*** and ***colleagues*** for their cooperation and support that made this experience truly enriching.

I would like to acknowledge ***Sri Ramakrishna Engineering College*** for their help and encouragement.

Finally, I would like to take this opportunity to sincerely thank ***my family*** for continuously standing by me throughout the journey, irrespective of my ups-and-downs.

Contents

Problem Statement

- Developing an automated text summarization system that can accurately and efficiently condense large bodies of text into concise summaries is essential for enhancing business operations.
- This project aims to deploy NLP techniques to create a robust text summarization tool capable of handling various types of documents across different domains.
- The system should deliver high-quality summaries that retain the core information and contextual meaning of the original text.

Project Statement

- Text Summarization focuses on converting large bodies of text into a few sentences summing up the gist of the larger text.
- There is a wide variety of applications for text summarization including News Summary, Customer Reviews, Research Papers, etc.
- This project aims to understand the importance of text summarization and apply different techniques to fulfill the purpose.

Approach to Solution

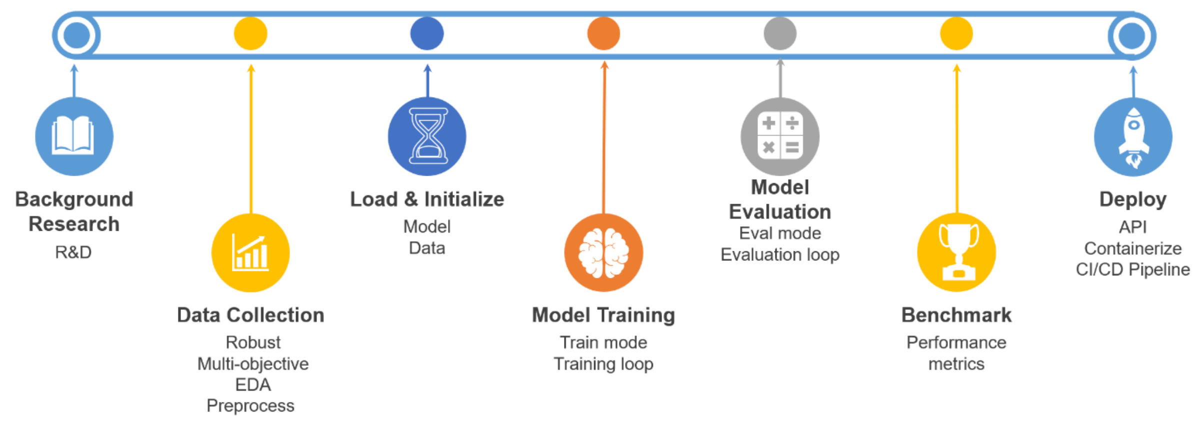


Fig.: Transformer architecture

Background Research

Literature Review

S. No	Use-Case	Paper Title	Year	Method	Dataset	Results	Limitations
1	General text summarization	Text Summarization Using Deep Learning Techniques: A Review	2023	Deep Learning (Seq2Seq, Attention, Transformers)	CNN/Daily Mail, XSum	Improved performance in capturing semantic relationships, better coherence	Computationally expensive, requires large datasets
2	Implementation of the Transformer architecture	Attention is all you need	2023	Transformer	WMT 2014 English-German, WMT 2014 English-French	Introduced the Transformer architecture, significantly improving the performance of text summarization tasks.	Requires large datasets and computational resources for training.
3	Multi-document summarization	Surveying the Landscape of Text Summarization with Deep Learning	2023	Deep learning methods. Various techniques like RBMs and fuzzy logic employed for summarization.	CNN/DailyMail	Incorporating transfer learning enhances summary quality and reduces data demand.	Complex models, high computational resources
4	Abstractive summarization	Pegasus: Pre-training with gap-sentences for abstractive summarization	2020	Transformer (Pegasus)	XSum, CNN/DailyMail, and Reddit TIFU	Significant improvements in abstractive summarization quality	Resource-intensive
5	Extractive summarization	Text Summarization with Pretrained Encoders	2019	Intersentence Transformer layers for summarization	CNN/Daily Mail, NYT, Xsum, DailyMail	BERT-based models outperformed other approaches in abstractive summarization.	High computational resources required

SOLUTION

Selected Deep Learning Architecture

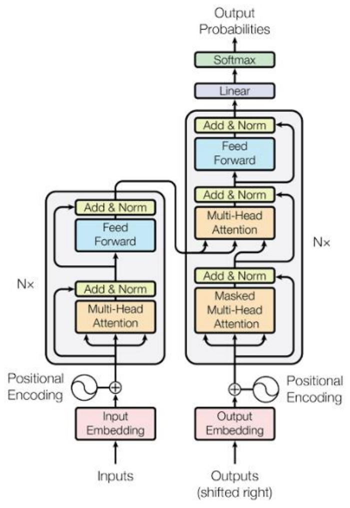


Fig.: Transformer architecture.

- **Implementation methods:**
  - *From Scratch*
    - Build Model
      - NN
    - Initialize normalized W&B
    - Train model with extensive data
    - Hence,
      - Computationally Intensive
      - Sub-Optimal usage of resources
      - Out-of-scope
  - *Using Pre-trained model*
    - Load Model & its parameters
    - Re-Train with specific dataset
    - Evaluate
    - Hence,
      - Innovation can be done at intended tasks
      - Optimal utilization of resources

WORKFLOW

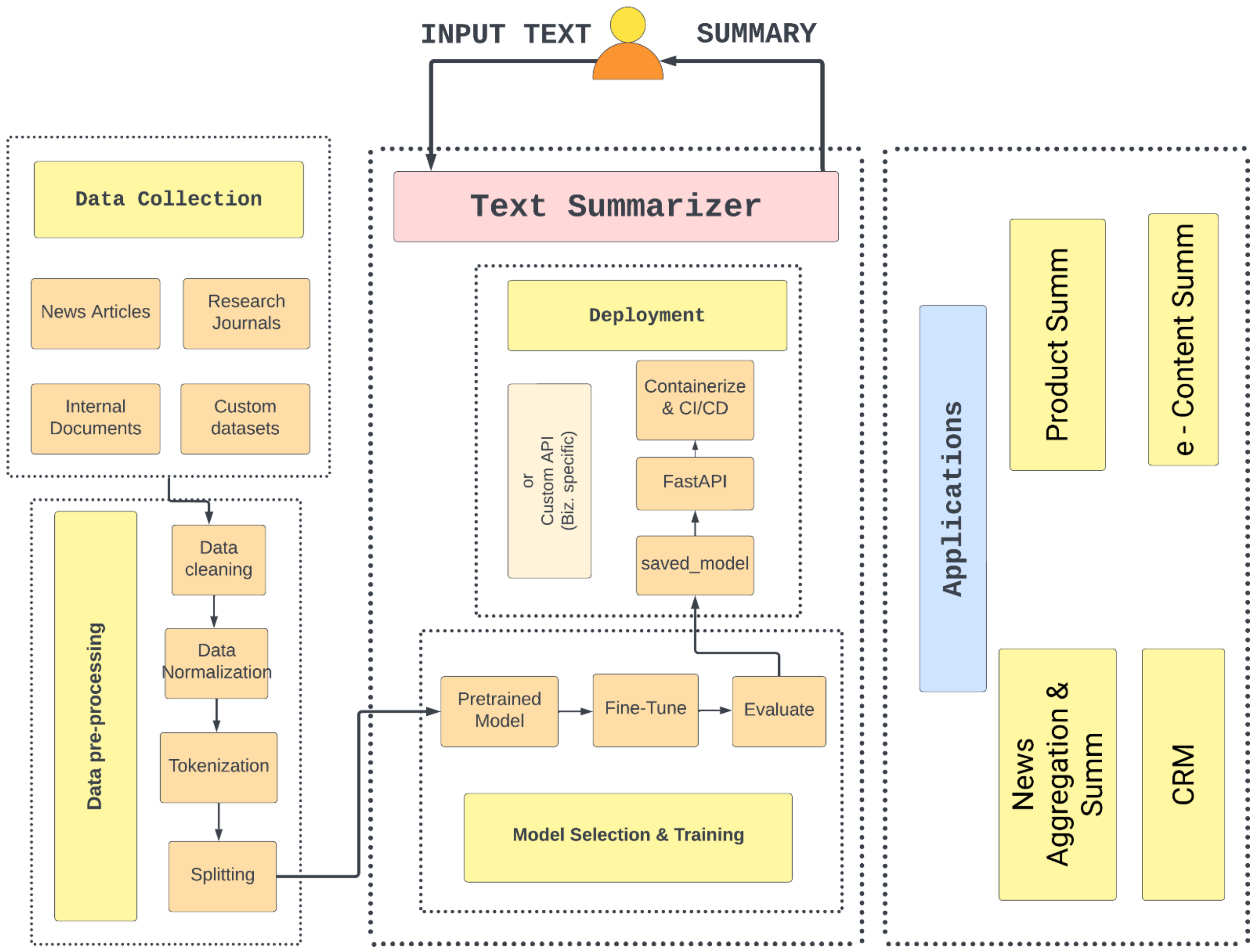


Fig.: Workflow for Text Summarizer.

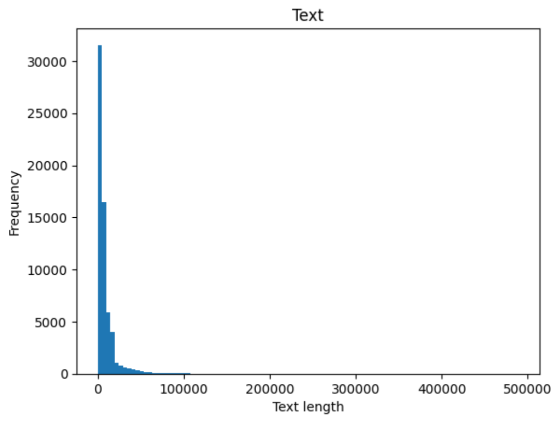
DATA COLLECTION

- Data Preprocessing & Pre-processing Implemented in src/data_preprocessing.
- Data collection from different source
    - CNN, Daily Mail: News,
    - BillSum: Legal,
    - ArXiv : Scientific,
    - Dialoguesum : Conversations.
- Data were integrated from different sources, to ensure robust and multi-objective data. The objective of the data spans including News articles, Legal Documents – Acts, Judgements, Scientific papers, Conversations between persons.
- The quality and consistency of the raw data is very vital for the model training, to achieve benchmark performance metrics - ROUGE.
- Validated the data through Data Statistics and Exploratory Data Analysis (EDA) through Frequency Plotting, for every data source.
- Saved each data in separate files (preprocessed - 1, 2, 3, 4) in csv format.
- Data integration of selective data from each source.
    - selected from splits (train, test, val).
    - Factors considered:
        - Number of records
        - Quality of data
        - Representation of different data groups (Multi-Objective)
        - To enable the model training in domestic GPU.
            - Limits the number of records.

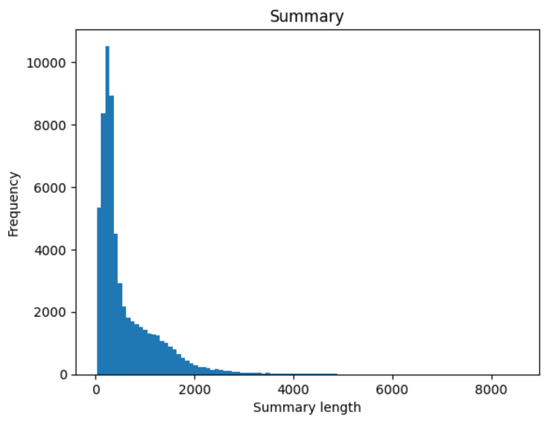
- Selected data:
  - CNN, Daily Mail - test, validation
  - BillSum - train
  - ArXiv - test
  - Dialoguesum - train
- renamed the attributes to general form (text, summary).
- Validated the data using statistics and frequency plot.

Dataset before cleansing

```
count     62707.000000
mean       8209.765879
std       11786.346696
min         190.000000
25%        1892.000000
50%        5051.000000
75%        9439.500000
max      489287.000000
Name: text, dtype: float64
```



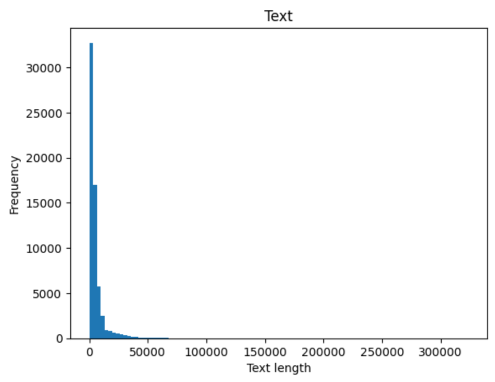
```
count    62707.000000
mean       612.585150
std        620.618153
min         31.000000
25%        213.000000
50%        353.000000
75%        839.000000
max       8541.000000
Name: summary, dtype: float64
```



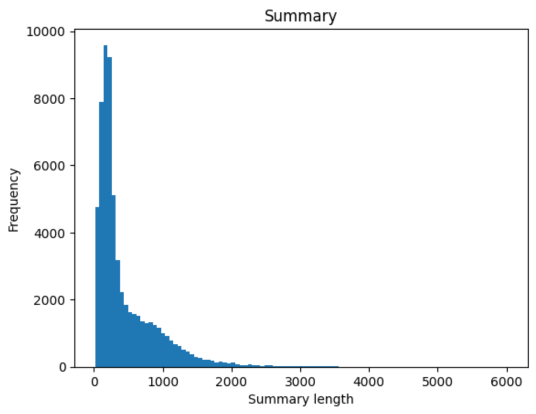
- Performed data cleansing optimized for NLP tasks.
  - Removed null records.
  - Lowercasing.
  - Punctuation removal.
  - Stop words removal
  - Lemmatization.

Dataset after cleansing

```
count     62707.000000
mean       5211.270975
std        7794.860686
min          83.000000
25%        1275.000000
50%        3176.000000
75%        5684.500000
max      323742.000000
Name: text, dtype: float64
```



```
count    62707.000000
mean       448.081937
std        459.087443
min         16.000000
25%        154.000000
50%        255.000000
75%        618.000000
max       6014.000000
Name: summary, dtype: float64
```



- Performed data splitting from the integrated data- train, test, validation - for training, testing and validating the model respectively, using sci-kit learn.
  - saved the data in csv format.
  - Dataset – for reference.

ABSTRACTIVE TEXT SUMMARIZATION

MODEL TRAINING & EVALUATION

TRAINING

- The selected transformer architecture, for ABSTRACTIVE SUMMARIZATION, can be implemented in a couple of ways. Either developing the neural network from scratch and initializing it with normalized weight and biases, then training the model with massive datasets for NLP tasks, or retraining the foundational model with custom dataset i.e. fine-tuning the pre-trained model.
- The former way, will pave the way to excessive unoptimized resource utilization, in terms of computation (GPU). Also, this would not out-perform the fine-tuning of the larger foundational model.

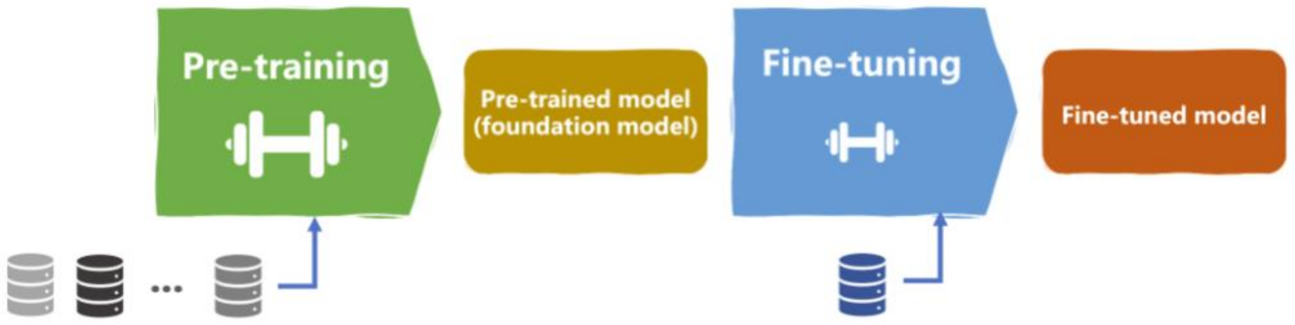


Fig.: Fine-Tunning Overview

- The choice of foundational model vests in considering lots of factors including its performance metrics and efficient trainable parameters in the model.
- With due analysis, *Facebook's Bart Large* – was chosen as the foundational model for abstractive text summarization.
  - `406,291,456 total parameters.`
  - `406,291,456 training parameters.`
- Model Training can be achieved in two (2) methods:
  1. Native PyTorch Implementation
  2. Trainer API implementation.

EVALUATION

- Performance metrics – ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

  - ROUGE – Measures the overlap between generated summary and reference summary.
  - Best suited: evaluating 'Text Summarization' tasks.
  - Other options : **BLEU**.
  - Available metrics in ROUGE: ROUGE-N = 1,2,3; ROUGE LSUM.

METHOD 1 - Native PyTorch

- Native PyTorch implementation, source code: src/model.ipynb
- Implemented works as follows:
- Loaded pre-trained transformer
    - Facebook's Bart Large
- Developed OOP implementation of Dataset
    - Feature, Target Loading
    - Tokenized the dataset
    - Padding, Truncate w.r.t maximum length
    - Converted to Tensor
    - Passed on to DataLoader – with batch size
- Developed manual PyTorch training Loop
    - Set the model in 'Train mode'.
    - Utilized 'Adam' optimizer.
    - Forward pass & compute loss
    - Backward pass
    - Updated params – compute gradient
    - Updated Learning Rate
    - Zeroed the gradients
    - Updated total loss
    - [Average Training Loss: 1.3280]
- Saved the fine-tuned transformer model -> saved model
- Developed manual evaluation loop:
    - Set the model to 'Eval mode'.
    - No gradient calculation for evaluation.

  - Forward pass & compute loss.
  - Accumulate batch loss.
  - Printed batch information.
  - Calculated average validation loss.
  - Printed final evaluation results (loss and time).
  - [Validation Loss: 2.4502].

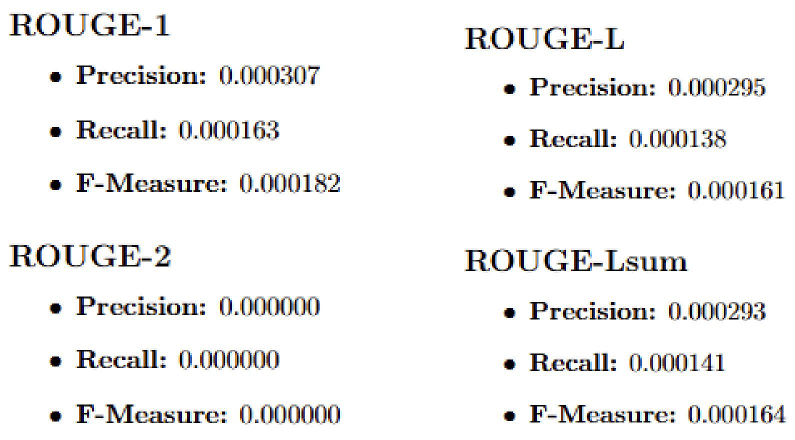
METHOD 1 - MODEL EVALUATION

- Model Evaluation, source code: src/evaluation.ipynb
- Implemented works:
- Load fine-trained transformer
  - From saved model
- OOP implementation of Dataset
  - Feature, Target
  - Tokenize
  - Padding, Truncate
  - Convert to Tensor
  - Pass to: DataLoader – with batch size
- Evaluate Model
  - Set model to evaluation mode
  - Load ROUGE metric
  - Loop through batches in DataLoader
  - Move data to device
  - Generate summaries
  - Decode predictions and labels
  - Add to ROUGE metric

  - Compute ROUGE scores
- Evaluate model on validation dataset
  - Print results (ROUGE mid).
- Saved fine-tuned BART model and tokenizer. (Saved model)

Results:

- Performance metrics – After fine-tuning.



ROUGE-1
- **Precision:** 0.000307
- **Recall:** 0.000163
- **F-Measure:** 0.000182

ROUGE-2
- **Precision:** 0.000000
- **Recall:** 0.000000
- **F-Measure:** 0.000000

ROUGE-L
- **Precision:** 0.000295
- **Recall:** 0.000138
- **F-Measure:** 0.000161

ROUGE-Lsum
- **Precision:** 0.000293
- **Recall:** 0.000141
- **F-Measure:** 0.000164

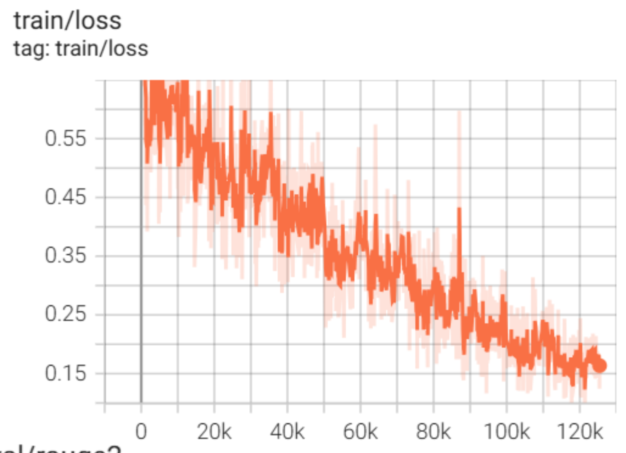
Observations:

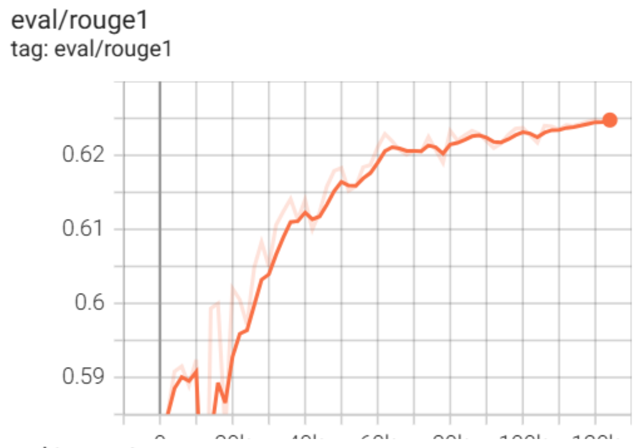
- The trained model from method 1 was not used for deployment.
  - (Trained model from method 2 was used for deployment)
- Reason:
  - Even though the model has very minimal training loss but, the model performed inconsistently in validation & testing phase.
  - There's a suspected tensor error while training using method 1, which could be attributed to the inconsistency of the model's output.
- *Dire need to implement alternative approach* optimized for transformer model, to produce benchmark performance.

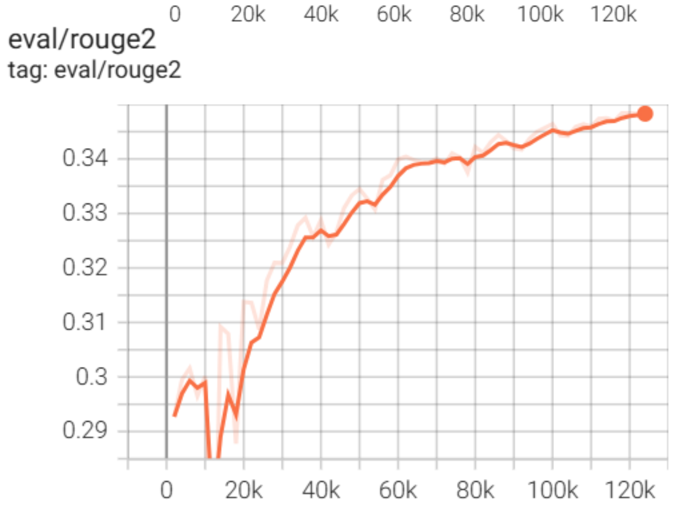
METHOD 2 – Trainer Class Implementation

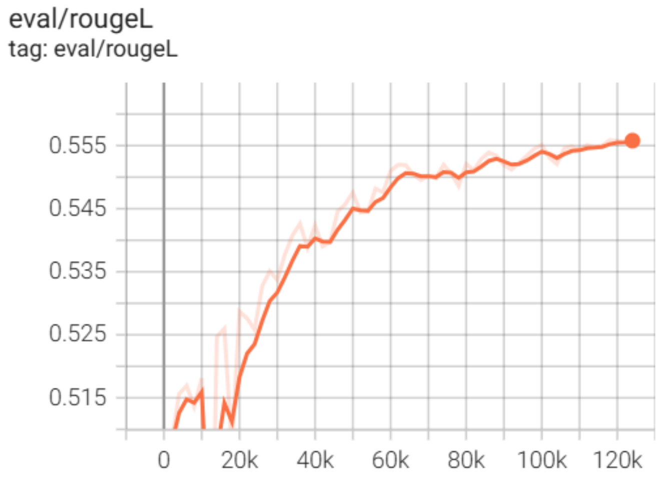
- Trainer class implementation, source code: src/bart.ipynb.
- Utilized Trainer API from Hugging face, Trainer class is optimized to train the transformer models and Pre-Trained models.
- Enables to have feature-complete training and eval loop for PyTorch, optimized for transformer models.
- Implemented works in this module, as follows:
- Data Preparation:
    - Loaded and converted datasets to Hugging Face Dataset format -PyArrow
- Data Preprocessing:
    - text-to-input conversion.
    - mapped preprocessing to datasets with parallel processing.
- Model Training:
    - Load pre-trained model
        - Facebook - Bart large
    - defined evaluation metrics
    - Configured training arguments
    - Initialized Trainer
    - Trained model and obtained training history.
- Model Saving:
    - Saved fine-tuned BART model and tokenizer (Saved model)

TensorBoard Training Monitor









Results

- The model was trained with whole dataset for 10 epochs for 26:24:22 (HH:MM:SS) in 125420 steps.
    - Train loss = 17.28 (final)
    - ROUGE1 score = 62.52 (Last checkpoint)
    - Transformer model for abstractive text summarization was successfully trained with the integrated custom data.

Observations

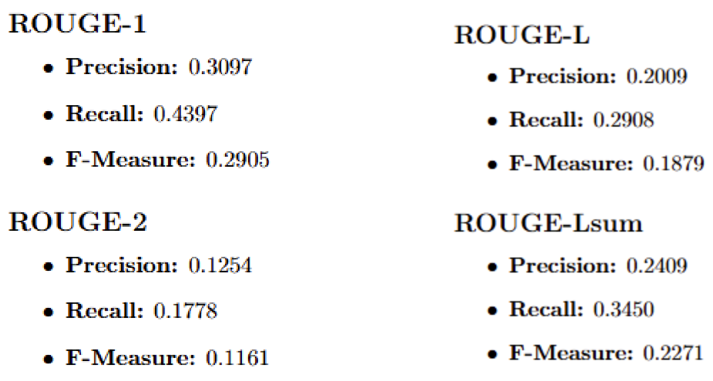
- On testing the model, from method 2, consistent performance was observed.
- Considered the performance metrics of the models trained by the forementioned methods.
- After the due analysis, the model trained using 'Method 2' was selected for deployment.

METHOD 2 – MODEL EVALUATION

- Evaluation for the model trained using Trainer class implementation, source code: src/rouge.ipynb.
- Performance metrics – ROUGE (Recall-Oriented Understudy for Gisting Evaluation).
- Implemented works:
  - Load the validation data.
  - feature - load & tokenize & convert to tensor
  - Generated summary IDs with specified parameters
  - Decoded summary IDs to text and skip special tokens
  - Generated summaries for the validation set
  - computed rouge metrics based on generated summary and original summary (target)

Results:

- Performance Metrics - Before fine-tuning:



ROUGE-1

- **Precision:** 0.3097
- **Recall:** 0.4397
- **F-Measure:** 0.2905

ROUGE-2

- **Precision:** 0.1254
- **Recall:** 0.1778
- **F-Measure:** 0.1161

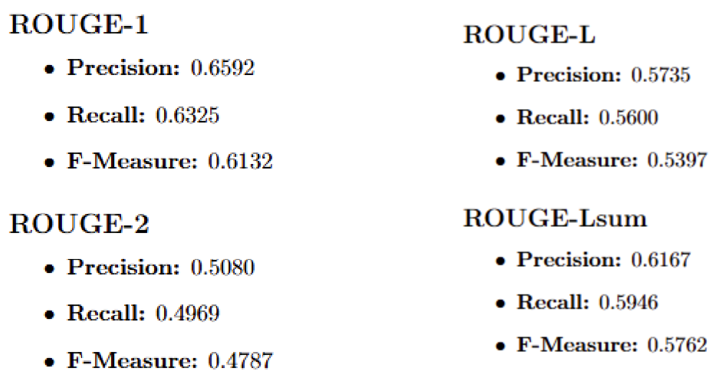
ROUGE-L

- **Precision:** 0.2009
- **Recall:** 0.2908
- **F-Measure:** 0.1879

ROUGE-Lsum

- **Precision:** 0.2409
- **Recall:** 0.3450
- **F-Measure:** 0.2271

- Performance Metrics - After fine-tuning:



ROUGE-1

- **Precision:** 0.6592
- **Recall:** 0.6325
- **F-Measure:** 0.6132

ROUGE-2

- **Precision:** 0.5080
- **Recall:** 0.4969
- **F-Measure:** 0.4787

ROUGE-L

- **Precision:** 0.5735
- **Recall:** 0.5600
- **F-Measure:** 0.5397

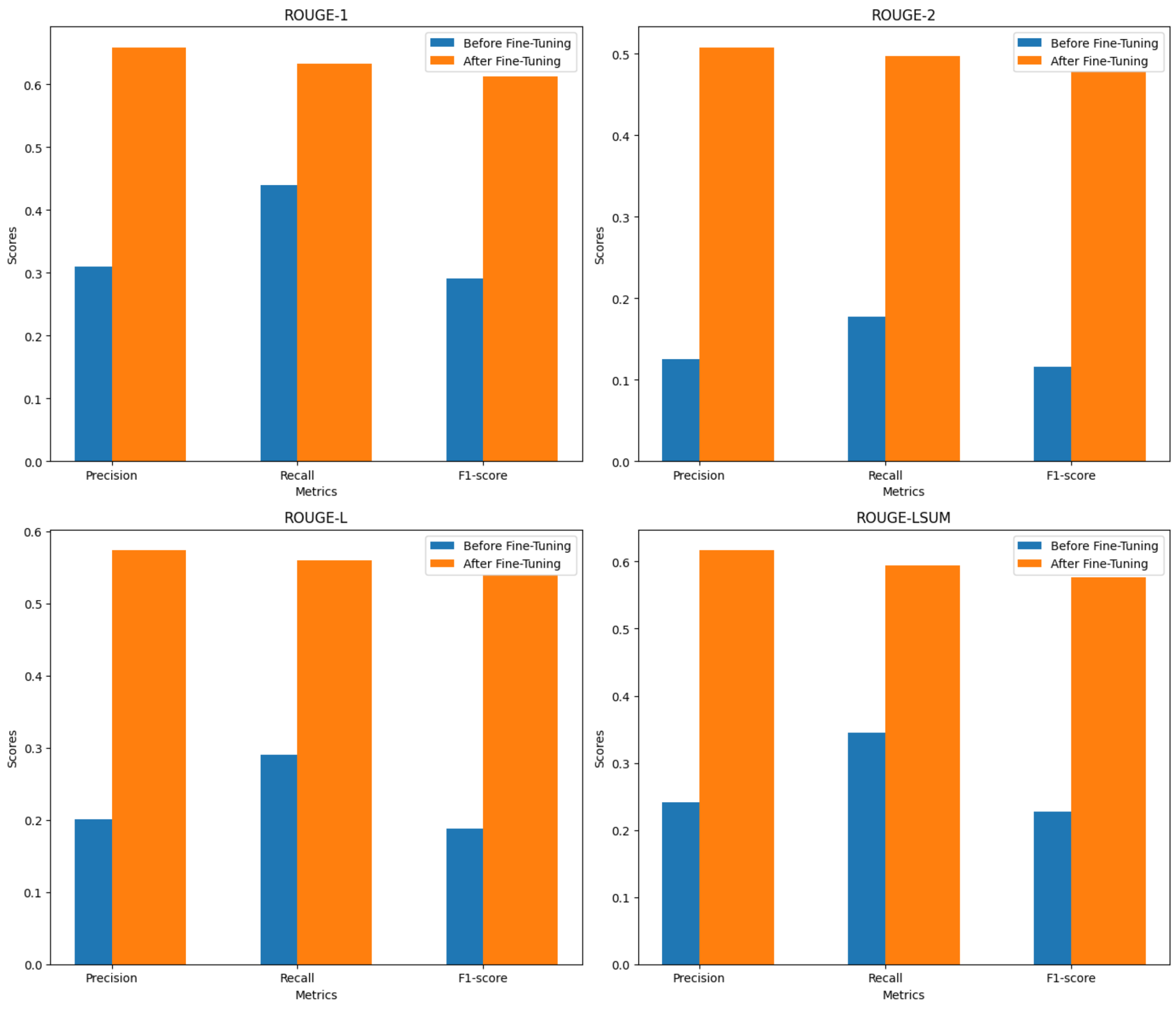
ROUGE-Lsum

- **Precision:** 0.6167
- **Recall:** 0.5946
- **F-Measure:** 0.5762

Observations

- The trained model from method 1 was not used for deployment:
- Trained model from method 2 was used for deployment
- Reason:
    - Even though the model has very minimal training loss but, the model performed inconsistently in validation & testing phase.
    - There's a suspected tensor error while training using method 1, which could be attributed to the inconsistency of the model's output.
    - Model 2 - results outperformed that of method 1.
        - ROUGE1 (F-Measure) = 61.32 -> Benchmark grade
        - GPT4 performance for text summarization - ROUGE1 (F-Measure) is 63.22

COMPARATIVE ANALYSIS



- Performance comparison of the fine-tuned transformer model, source code: src/compare.ipynb.
- Implemented works:
  - Before fine tuning vs After fine tuning
    - Basis: ROUGE scores - mid values

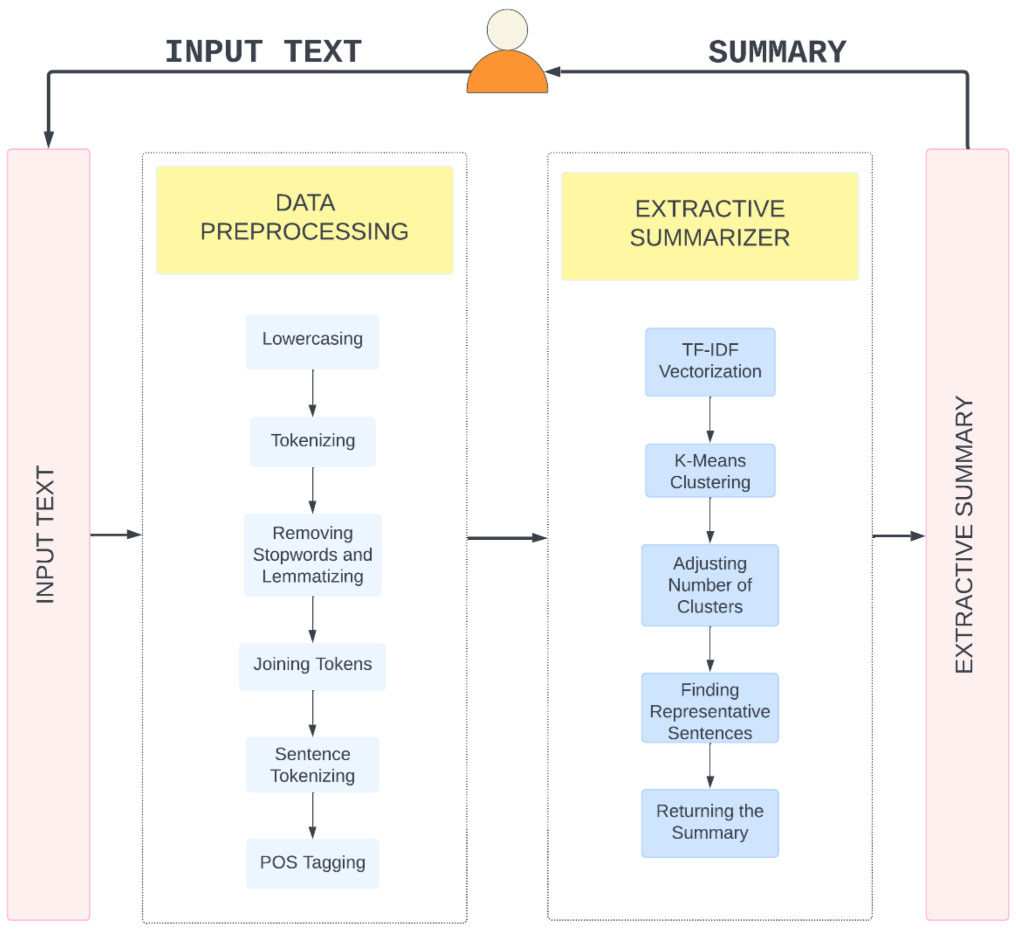
Observations

- The fine-tuning process significantly improved the transformer's performance across all ROUGE metrics:
    - ROUGE-1 and ROUGE-LSUM show over 100% improvement in F1-score.
    - ROUGE-2 demonstrates the highest improvement with over 300% increase in F1-score
    - ROUGE-L scores also saw substantial improvements, particularly in Precision and F1-score.
- These results indicate that fine-tuning has greatly enhanced the model's ability to generate accurate and relevant summaries.
- Hence, the *fine-tuning of the transformer model is successful.*

EXTRACTIVE TEXT SUMMARIZATION

MODEL

- Utilized a rule based approach, source code: src/Extractive_Summarization.ipynb.

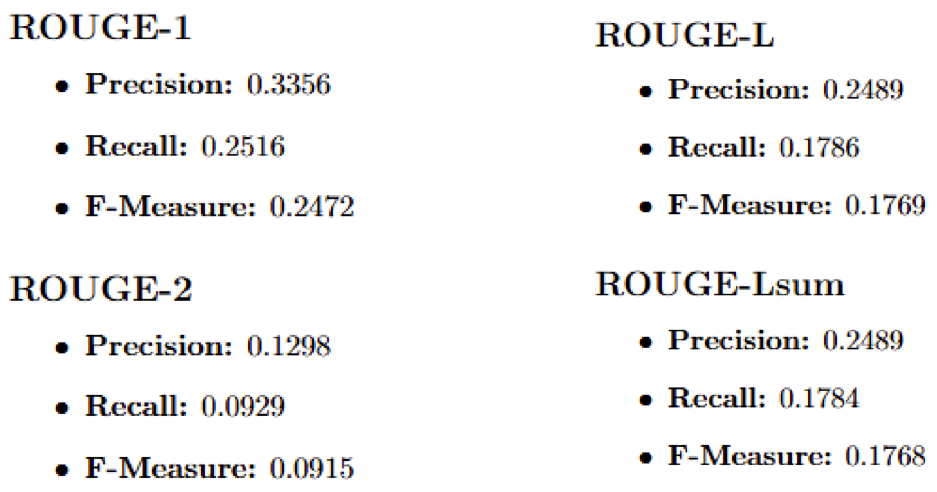


- The existing ways for extractive summarization: Text Rank, TF-IDF, ML, DL – models.
- Rather than choosing computationally intensive deep-learning models, utilizing a rule based approach will result in optimal solution. Utilized a new-and-novel approach of combining the matrix obtained from TF-IDF and KMeans Clustering methodology.
- It is the expanded topic modeling specifically to be applied to multiple lower-level specialized entities (i.e., groups) embedded in a single document. It operates at the individual document and cluster level.
- The sentence closest to the centroid (based on Euclidean distance) is selected as the representative sentence for that cluster.

- Implemented works:
    - Preprocesses the input text to get POS-tagged sentences.
    - Data Preprocessing:
        - Lowercasing
        - Stop Words Removal.
        - Lemmatization.
        - Tokenization.
        - POS Tagging.
    - Adjusts the number of clusters if there are fewer sentences than clusters.
    - Initializes a TF-IDF vectorizer with stop words removed.
    - Transforms the sentences into a TF-IDF matrix.
        - Matrix where each sentence is represented as a vector of TF-IDF scores
    - TF-IDF matrix is fed into the KMeans clustering algorithm.
        - Each sentence is assigned a cluster label.
    - Identifies the cluster labels for each sentence.
    - For each cluster:
        - Finds the indices of sentences in the cluster.
        - Calculates the centroid of the cluster.
        - Identifies the sentence closest to the centroid (representative sentence).
    - Collects the representative sentences from each cluster.
    - Joins and returns the representative sentences as a extractive summary.
    - Evaluating Summaries
        - ROUGE scores as performance metrics

Results

- Rule-based approach for extractive summarization was implemented and evaluated successfully.
- ROUGE1 (F-Measure) = 24.72.



ROUGE-1

- **Precision:** 0.3356
- **Recall:** 0.2516
- **F-Measure:** 0.2472

ROUGE-2

- **Precision:** 0.1298
- **Recall:** 0.0929
- **F-Measure:** 0.0915

ROUGE-L

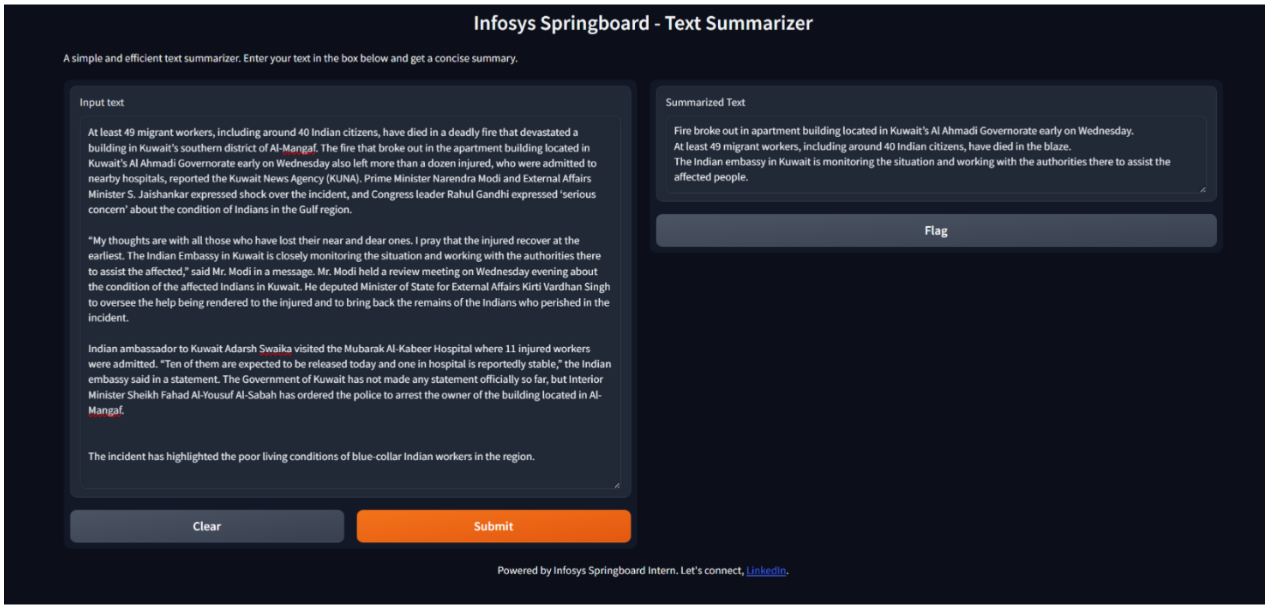
- **Precision:** 0.2489
- **Recall:** 0.1786
- **F-Measure:** 0.1769

ROUGE-Lsum

- **Precision:** 0.2489
- **Recall:** 0.1784
- **F-Measure:** 0.1768

TESTING

- Implemented works, as follows, source code: src/interface.ipynb.
    - text summarization application (Abstractive text summarization).
        - using a fine-tuned transformer model (from method 2)
        - Gradio library - for web-based interface



Infosys Springboard - Text Summarizer

A simple and efficient text summarizer. Enter your text in the box below and get a concise summary.

Input text

At least 49 migrant workers, including around 40 Indian citizens, have died in a deadly fire that devastated a building in Kuwait's southern district of Al-Mangaf. The fire that broke out in the apartment building located in Kuwait's Al Ahmadi Governorate early on Wednesday also left more than a dozen injured, who were admitted to nearby hospitals, reported the Kuwait News Agency (KUNA). Prime Minister Narendra Modi and External Affairs Minister S. Jaishankar expressed shock over the incident, and Congress leader Rahul Gandhi expressed 'serious concern' about the condition of Indians in the Gulf region.

"My thoughts are with all those who have lost their near and dear ones. I pray that the injured recover at the earliest. The Indian Embassy in Kuwait is closely monitoring the situation and working with the authorities there to assist the affected," said Mr. Modi in a message. Mr. Modi held a review meeting on Wednesday evening about the condition of the affected Indians in Kuwait. He deputed Minister of State for External Affairs Kirti Vardhan Singh to oversee the help being rendered to the injured and to bring back the remains of the Indians who perished in the incident.

Indian ambassador to Kuwait Adarsh Swaika visited the Mubarak Al-Kabeer Hospital where 11 injured workers were admitted. "Ten of them are expected to be released today and one in hospital is reportedly stable," the Indian embassy said in a statement. The Government of Kuwait has not made any statement officially so far, but Interior Minister Sheikh Fahad Al-Yousuf Al-Sabah has ordered the police to arrest the owner of the building located in Al-Mangaf.

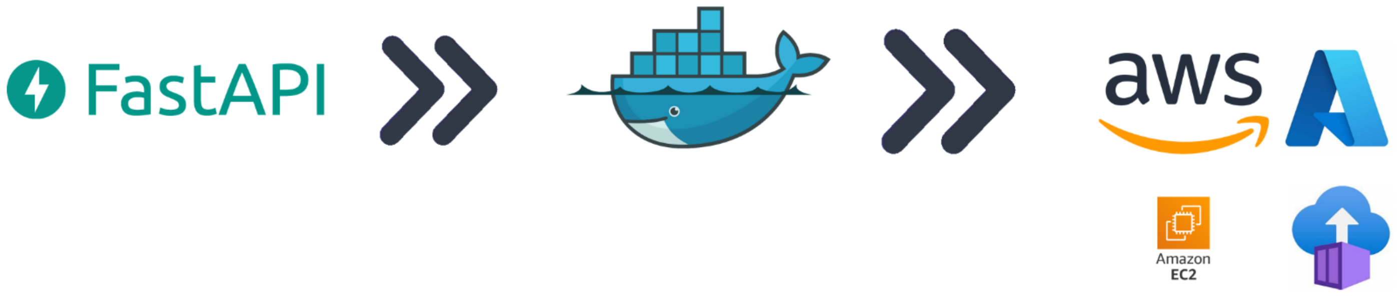
The incident has highlighted the poor living conditions of blue-collar Indian workers in the region.

Summarized Text

Fire broke out in apartment building located in Kuwait's Al Ahmadi Governorate early on Wednesday.
At least 49 migrant workers, including around 40 Indian citizens, have died in the blaze.
The Indian embassy in Kuwait is monitoring the situation and working with the authorities there to assist the affected people.

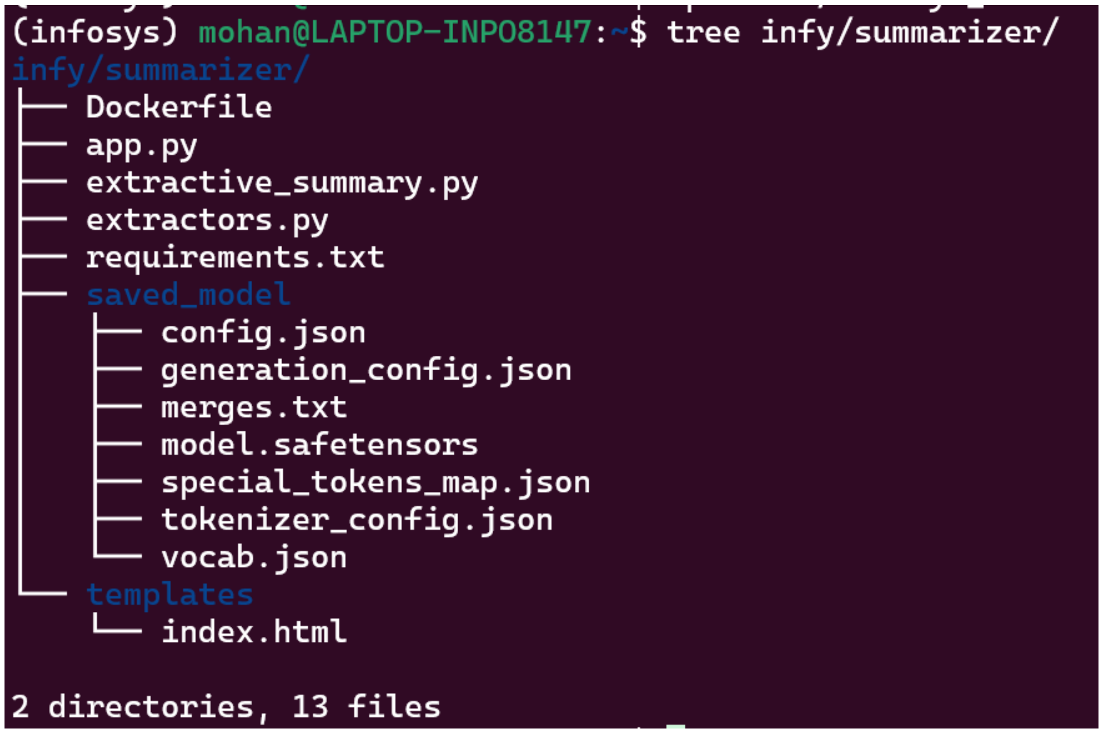
Powered by Infosys Springboard Intern. Let's connect, LinkedIn.

DEPLOYMENT



APPLICATION

- File structure:



```
(infosys) mohan@LAPTOP-INPO8147:~$ tree infy/summarizer/
infy/summarizer/
├── Dockerfile
├── app.py
├── extractive_summary.py
├── extractors.py
├── requirements.txt
├── saved_model
│   ├── config.json
│   ├── generation_config.json
│   ├── merges.txt
│   ├── model.safetensors
│   ├── special_tokens_map.json
│   ├── tokenizer_config.json
│   └── vocab.json
└── templates
    └── index.html

2 directories, 13 files
```

API ENDPOINTS

- Utilized the FastAPI framework to create a web application for text summarization.
  - source code: [summarizer/app.py]
- Developed to handles three main types of input: URLs, files, and direct text input.
- Created API endpoints using FastAPI, for inferencing the summarized model.
- Defined a function to summarize text using the saved model for abstractive text summarization.
  - Encodes the text, generates a summary, and decodes the summary back to text.
- API Endpoints:
  - Root Endpoint:
    - Serves an HTML template for the root URL.
  - Summarize URL:
    - Accepts a URL, extracts text from the URL.
    - generates both abstractive and extractive summaries.
    - Returns them as a JSON response.
  - Summarize File:
    - Accepts file uploads (PDF or DOCX), extracts text from the file,
    - generates summaries.
    - Returns them as a JSON response.
    - Unsupported file types return an error response.
  - Summarize Text:
    - Accepts direct text input from a form.
    - Generates summaries.
    - Returns them as a JSON response.
- Developed to run the FastAPI application using Uvicorn, listening on all available IP addresses on port 8000.

EXTRACTOR MODULES

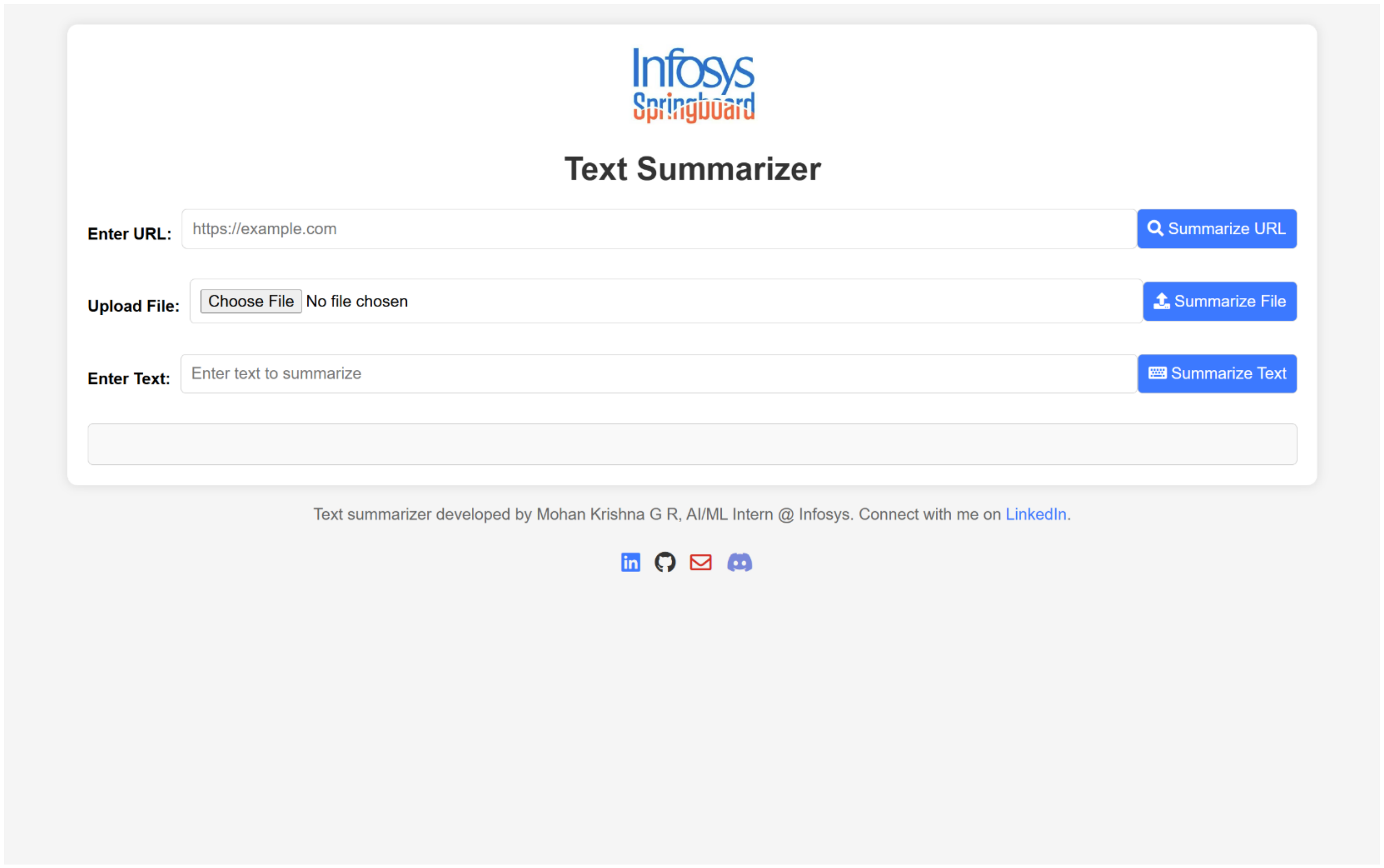
- Developed to extract text from various sources. Source code: summarizer/extractors.py
    - URLs, PDF files, and DOCX files.
- Text from URL:
    - Utilized BeatifulSoup for web crawling the web pages.
        - Extracts text from all paragraph with ‘<p>’ tags.
        - Joins the text of these paragraphs into a single string.
- Text from PDF:
    - Utilized fitz to open to open and extract text from pdf.
        - Iterates through each page of the PDF.
        - Extracts text from each page and concatenates it into a single string.
- Extract Text from DOCX:
    - Opens the DOCX file using Document.
        - Iterates through all paragraphs in the document.
        - Extracts text from each paragraph and joins them into a single string.
- These modular functions can be used independently or integrated into a larger application for processing different types of text sources.

EXTRACTIVE SUMMARY SCRIPT

- Implemented an extractive summarizer module as python script for application.
  - Source code: summarizer/extractive_summary.py .
- Developed to utilize the same programmatic approach of src/Extractive_Summarization.ipynb.
- This script is designed to:
  - Preprocess text.
  - Extract important features using TF-IDF.
  - Summarize the text by selecting representative sentences from different clusters.

USER INTERFACE

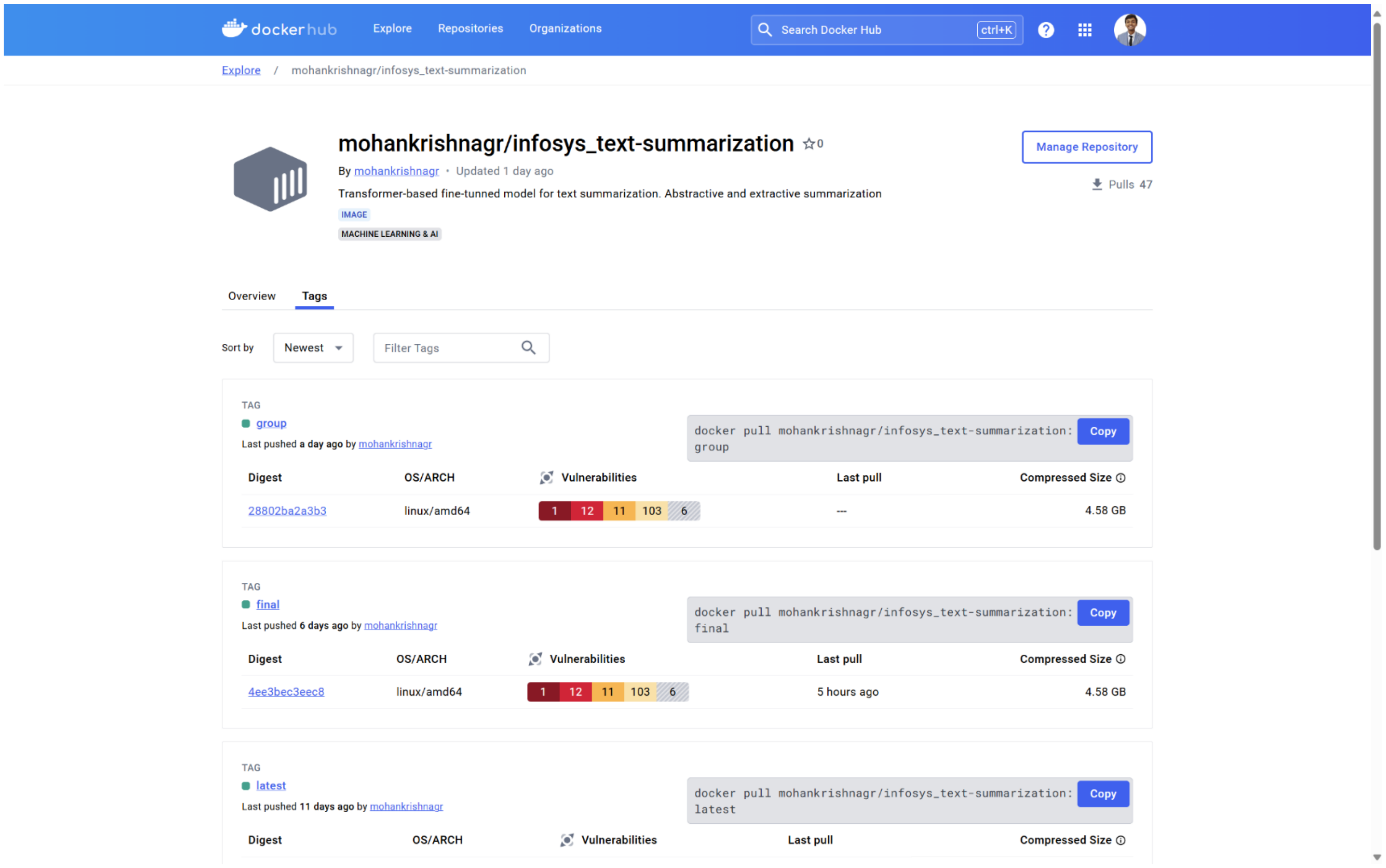
- Developed to provide a user-friendly interface for text summarization application.
    - Source code: summarizer/templates/index.html
- Designed forms for different input types (URL, file, and text).
- Utilized jQuery for event handling and AJAX.
    - Utilized AJAX to communicate with a backend server to perform the summarization.
    - Fetch API to POST server with request.
    - Implemented a loading spinner - during the request processing.
    - Displays results or errors based on the response.
- The results are displayed on the same page without reloading, providing a dynamic webpage.
- The design is enhanced with CSS for aesthetics and JavaScript for functionality.
- Added a footer with links to my LinkedIn, GitHub, email, and Discord profiles.



Text summarizer developed by Mohan Krishna G R, AI/ML Intern @ Infosys. Connect with me on LinkedIn.

CONTAINERIZATION

- Developed a Dockerfile to build the docker image for the FastAPI application.
    - Source code: summarizer/Dockerfile .
- The containerized docker image packages the entire application and its dependencies along with the saved model. Making the application easy to run consistently across different environments, hence removes the bottlenecks in production environments.
- Built the image & pushed into docker hub.
    - Docker image



mohankrishnagr/infosys_text-summarization ☆0

By mohankrishnagr • Updated 1 day ago

Transformer-based fine-tunned model for text summarization. Abstractive and extractive summarization

IMAGE

MACHINE LEARNING & AI

Pulls 47

TAG
group
Last pushed a day ago by mohankrishnagr

```
docker pull mohankrishnagr/infosys_text-summarization:group
```

Digest	OS/ARCH	Vulnerabilities	Last pull	Compressed Size
28802ba2a3b3	linux/amd64	1 12 11 103 6	---	4.58 GB

TAG
final
Last pushed 6 days ago by mohankrishnagr

```
docker pull mohankrishnagr/infosys_text-summarization:final
```

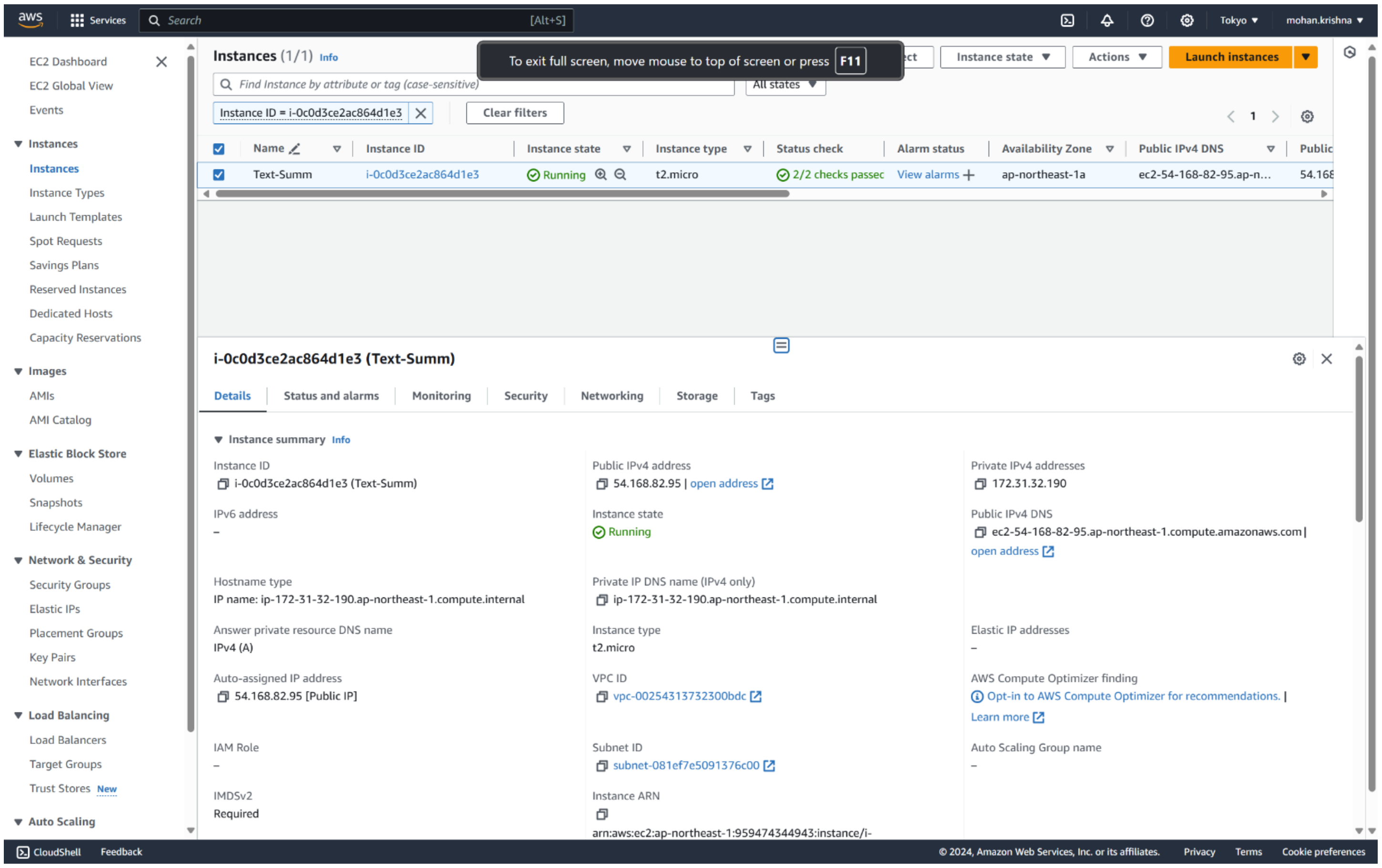
Digest	OS/ARCH	Vulnerabilities	Last pull	Compressed Size
4ee3bec3eec8	linux/amd64	1 12 11 103 6	5 hours ago	4.58 GB

TAG
latest
Last pushed 11 days ago by mohankrishnagr

```
docker pull mohankrishnagr/infosys_text-summarization:latest
```

Digest	OS/ARCH	Vulnerabilities	Last pull	Compressed Size

AWS EC2



To exit full screen, move mouse to top of screen or press F11

Instances (1/1) Info

Find Instance by attribute or tag (case-sensitive)

Instance ID = i-0c0d3ce2ac864d1e3

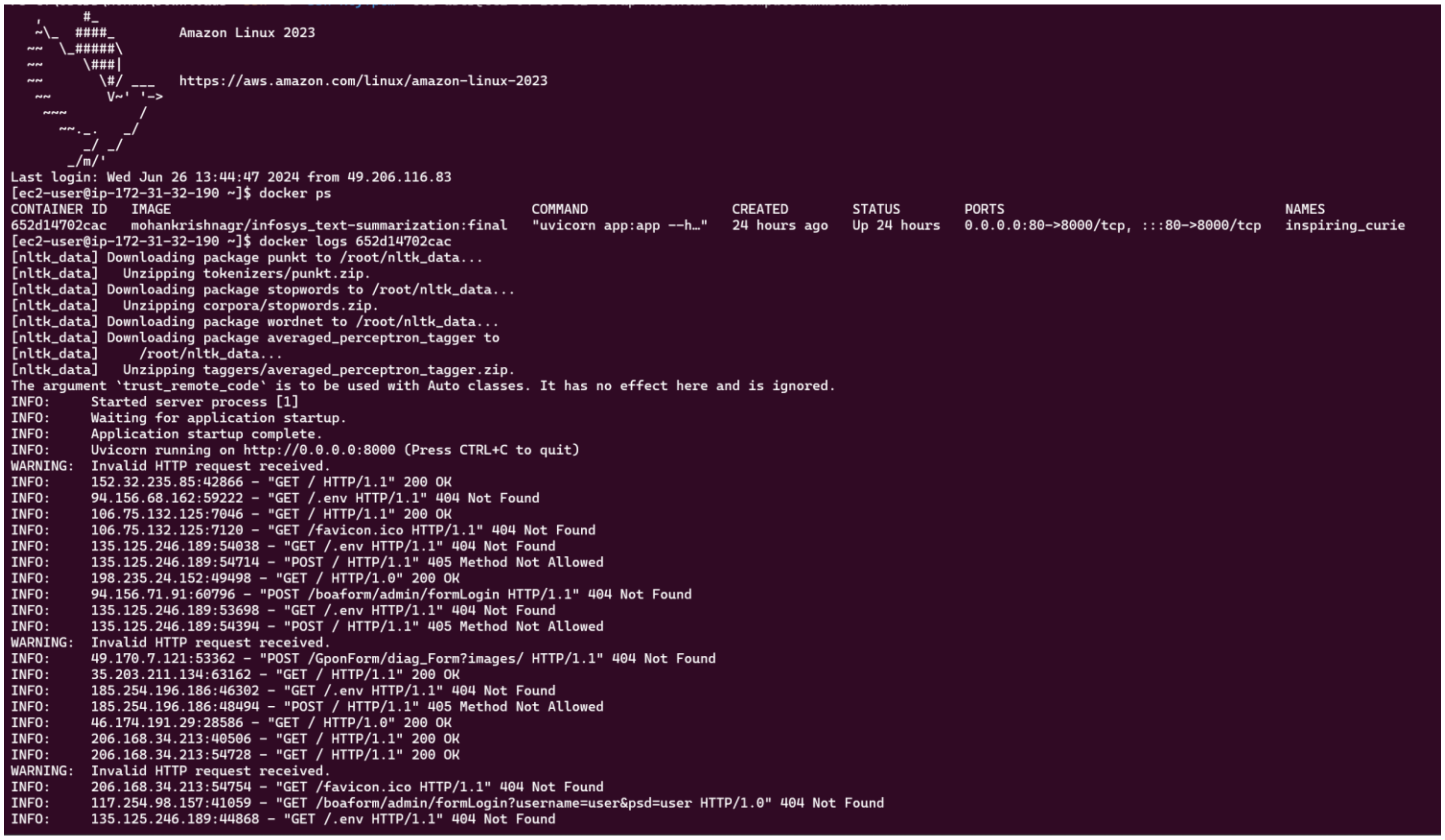
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
Text-Summ	i-0c0d3ce2ac864d1e3	Running	t2.micro	2/2 checks passec	View alarms +	ap-northeast-1a	ec2-54-168-82-95.ap-n...	54.168

i-0c0d3ce2ac864d1e3 (Text-Summ)

Instance summary Info

Instance ID i-0c0d3ce2ac864d1e3 (Text-Summ)	Public IPv4 address 54.168.82.95 | open address	Private IPv4 addresses 172.31.32.190
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-54-168-82-95.ap-northeast-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-32-190.ap-northeast-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-32-190.ap-northeast-1.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 54.168.82.95 [Public IP]	VPC ID vpc-00254313732300bdc	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-081ef7e5091376c00	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:ap-northeast-1:959474344943:instance/i-	

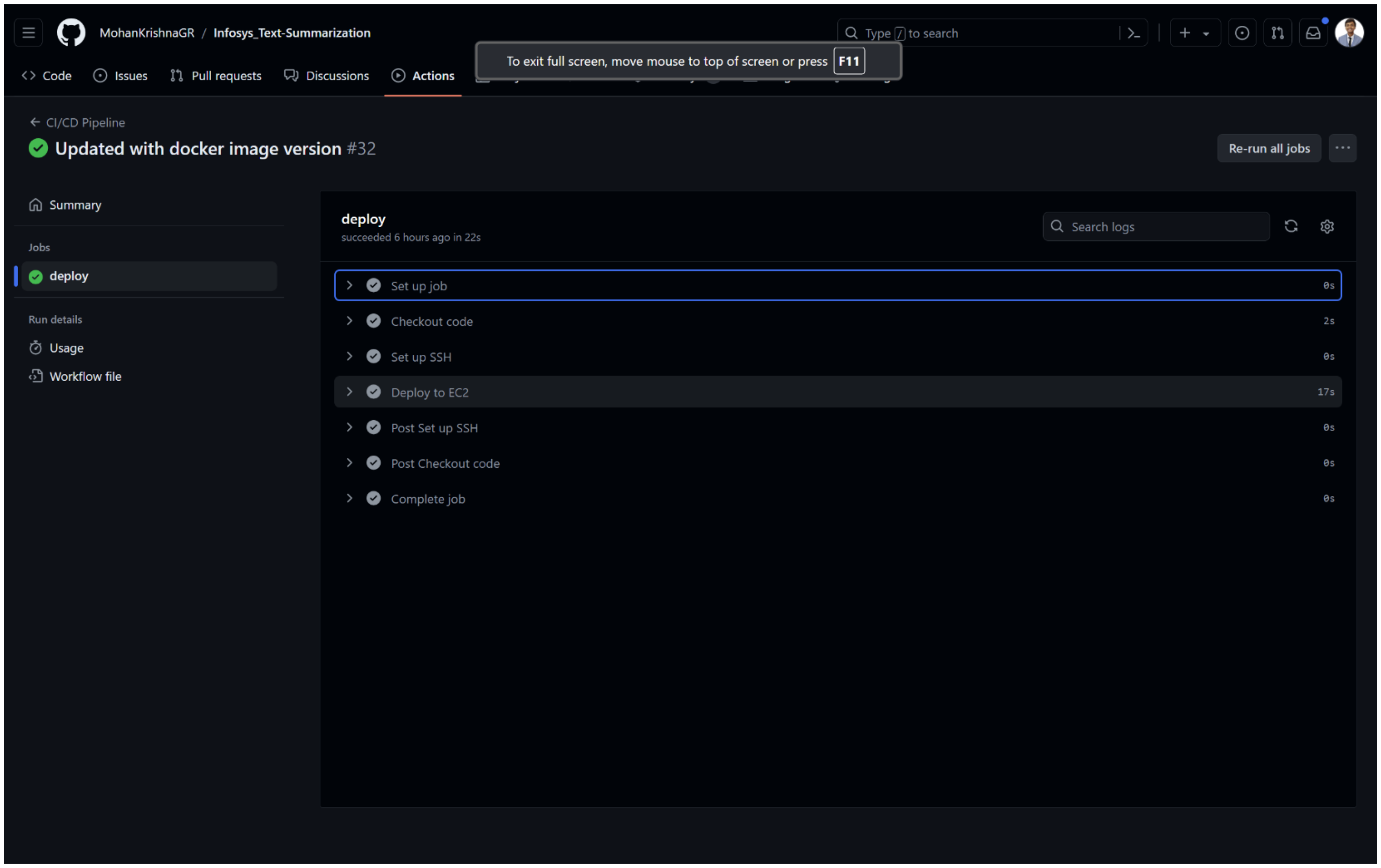
- AWS provides secure and resizable compute capacity for virtually any computational workload, under the service named Elastic Compute Cloud (EC2). But, limits the computation resources for free-tier plan.
- Launched a new EC2 instance named Text-Summ under the free-tier plan with *t2.micro* instance which has 1vCPU and 1 GiB Memory & 16 GiB of gp3 Root Volume. Amazon Linus was chosen as AMI. The instance was enabled with HTTP & HTTPS traffic from the internet.
- Further adding computation power like increased vCPU or GPU costs heavily and it is not covered under Free-Tier plan.



```
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Wed Jun 26 13:44:47 2024 from 49.206.116.83
[ec2-user@ip-172-31-32-190 ~]$ docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED        STATUS        PORTS                                   NAMES
652d14702cac   mohankrishnagr/infosys_text-summarization:final   "uvicorn app:app --h…"   24 hours ago   Up 24 hours   0.0.0.0:80->8000/tcp, :::80->8000/tcp   inspiring_curie
[ec2-user@ip-172-31-32-190 ~]$ docker logs 652d14702cac
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
The argument `trust_remote_code` is to be used with Auto classes. It has no effect here and is ignored.
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
WARNING:  Invalid HTTP request received.
INFO:     152.32.235.85:42866 - "GET / HTTP/1.1" 200 OK
INFO:     94.156.68.162:59222 - "GET /.env HTTP/1.1" 404 Not Found
INFO:     106.75.132.125:7046 - "GET / HTTP/1.1" 200 OK
INFO:     106.75.132.125:7120 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     135.125.246.189:54038 - "GET /.env HTTP/1.1" 404 Not Found
INFO:     135.125.246.189:54714 - "POST / HTTP/1.1" 405 Method Not Allowed
INFO:     198.235.24.152:49498 - "GET / HTTP/1.0" 200 OK
INFO:     94.156.71.91:60796 - "POST /boaform/admin/formLogin HTTP/1.1" 404 Not Found
INFO:     135.125.246.189:53698 - "GET /.env HTTP/1.1" 404 Not Found
INFO:     135.125.246.189:54394 - "POST / HTTP/1.1" 405 Method Not Allowed
WARNING:  Invalid HTTP request received.
INFO:     49.170.7.121:53362 - "POST /GponForm/diag_Form?images/ HTTP/1.1" 404 Not Found
INFO:     35.203.211.134:63162 - "GET / HTTP/1.1" 200 OK
INFO:     185.254.196.186:46302 - "GET /.env HTTP/1.1" 404 Not Found
INFO:     185.254.196.186:48494 - "POST / HTTP/1.1" 405 Method Not Allowed
INFO:     46.174.191.29:28586 - "GET / HTTP/1.0" 200 OK
INFO:     206.168.34.213:40506 - "GET / HTTP/1.1" 200 OK
INFO:     206.168.34.213:54728 - "GET / HTTP/1.1" 200 OK
WARNING:  Invalid HTTP request received.
INFO:     206.168.34.213:54754 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     117.254.98.157:41059 - "GET /boaform/admin/formLogin?username=user&psd=user HTTP/1.0" 404 Not Found
INFO:     135.125.246.189:44868 - "GET /.env HTTP/1.1" 404 Not Found
```

- Connected to the EC2 instance using SSH. The screenshot of the terminal of instance is attached above.
- Public IPv4 address:
  - Text Summarizer (http://54.168.82.95/)
- Drawbacks:
  - With the less computation power, the inferencing time for the deep learning models is very high.
  - Results in very high response time to server the API requests from the clients in production.
- Remedies:
  - Any attempt to optimize the API endpoints or API servicing methods doesn’t result in any easing of response time in production.
  - Only approach to enhance the performance, is to increase the computational power in the instance, with 1 vCPU, it is not expected to optimal inference deep learning models.

CI/CD PIPELINE – AWS


MohanKrishnaGR / Infosys_Text-Summarization

To exit full screen, move mouse to top of screen or press F11

← CI/CD Pipeline

Updated with docker image version #32

Re-run all jobs

Summary

Jobs

deploy

Run details

Usage

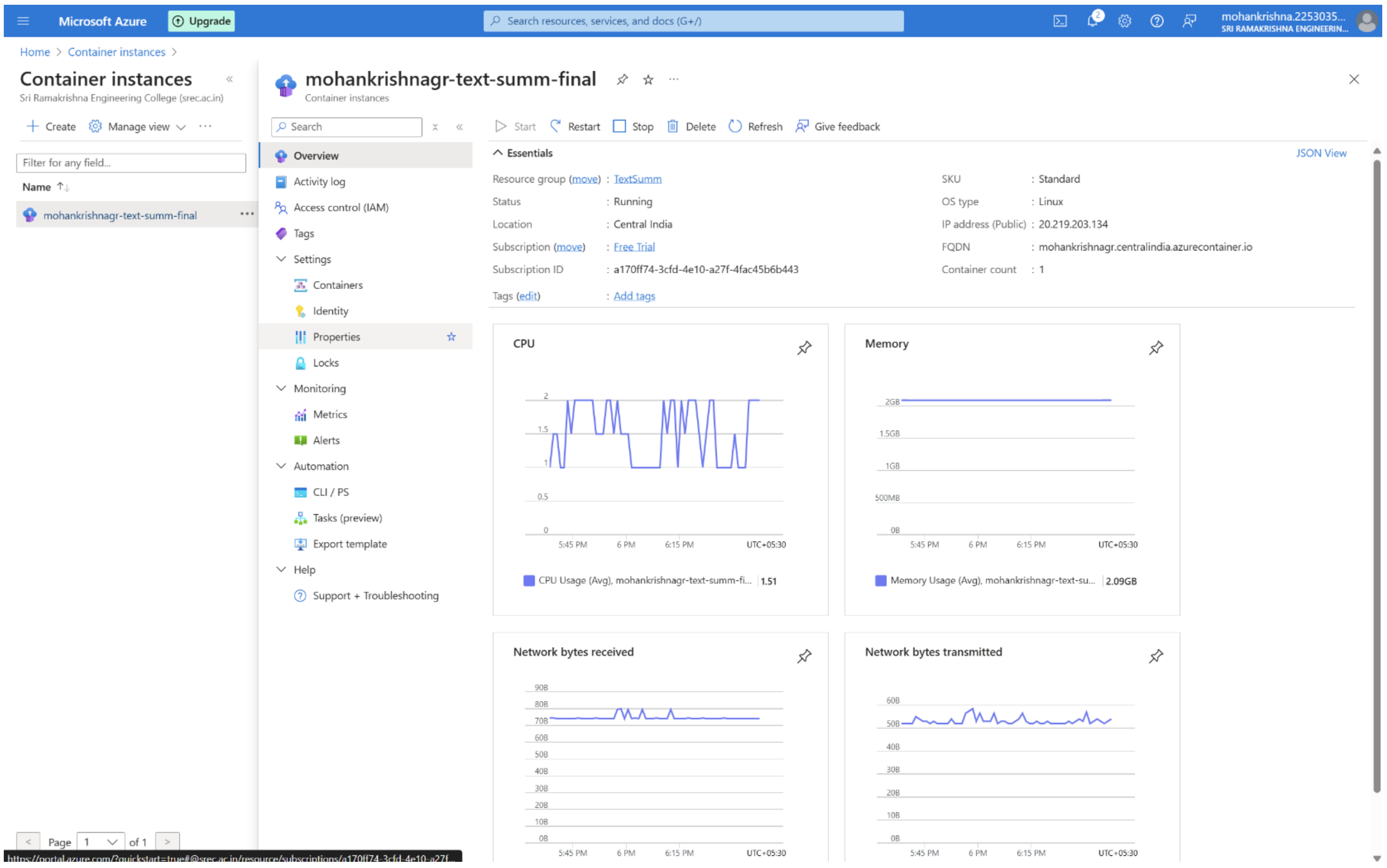
Workflow file

deploy
succeeded 6 hours ago in 22s

```
Set up job            0s
Checkout code         2s
Set up SSH            0s
Deploy to EC2         17s
Post Set up SSH       0s
Post Checkout code    0s
Complete job          0s
```

- Created a Continuous Integration / Continuous Deployment (CI/CD) pipeline for this project using GitHub Actions.
    - Source code: .github/workflows/deploy.yml
- This pipeline automates the process of deploying the Dockerized application to the Text-Summ EC2 instance whenever there are changes pushed to the 'main' branch.
- Implemented works, as follows:
    - Checks out the code from the main repository using actions.
    - Sets up the SSH with AWS EC2 instances, for job submission.
    - Runs a series of commands on a remote EC2 instance.
    - Connects and logs in to Docker Hub to pull the image: mohankrishnagr/Infosys_text-summarization:final
    - Stops and removes all running docker containers, to run the newly pulled container.

AZURE CONTAINER INSTANCE

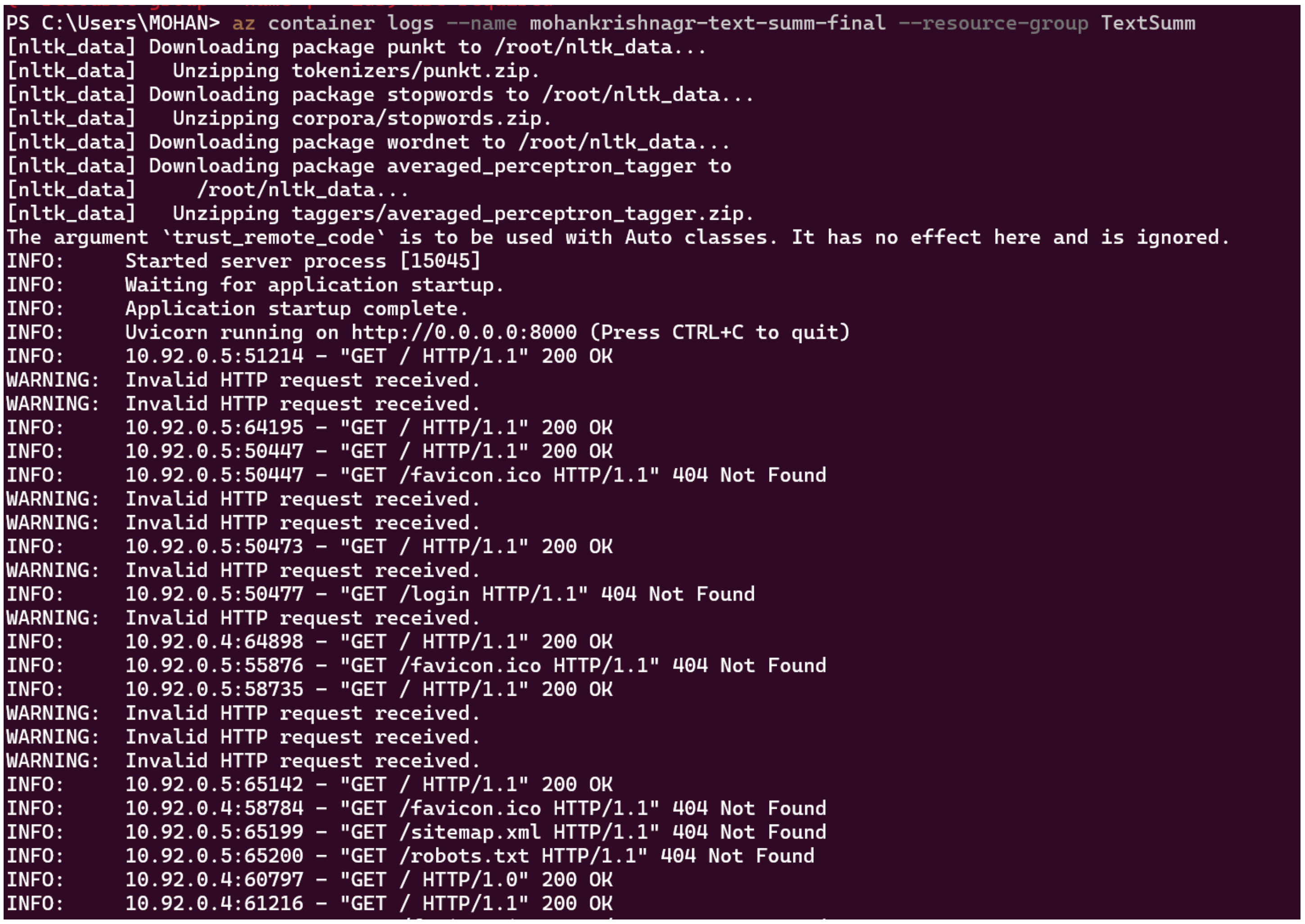


mohankrishnagr-text-summ-final

Resource group (move)	: TextSumm
Status	: Running
Location	: Central India
Subscription (move)	: Free Trial
Subscription ID	: a170ff74-3cfd-4e10-a27f-4fac45b6b443
Tags (edit)	: Add tags
SKU	: Standard
OS type	: Linux
IP address (Public)	: 20.219.203.134
FQDN	: mohankrishnagr.centralindia.azurecontainer.io
Container count	: 1

- Azure Container Instances (**ACI**) is a managed service that allows you to run containers directly on the Microsoft Azure public cloud, without requiring the use of virtual machines (VMs).
- In Azure, created a new public instance named 'TextSumm' with container name 'mohankrishnagr-text-summ-final', under 'Free Trail' subscription. With free trail, a student user will have INR.16,700 as free credit.
- Hence, a comparatively large compute instance can be created in Azure. Created instance with 4 vCPU and 16 GiB Memory.
- Selected 'Other registry' for Image source (Docker Hub) and specified the image as: mohankrishnagr/Infosys_text-summarization:final
- Modified the networking so as to allow port 8000 as TCP connection, along with DNS label to avail a Fully Qualified Domain Name (FQDN).
- Created Role Based Access Control (RBAC), in order to integrate this with the GitHub Actions Workflow.

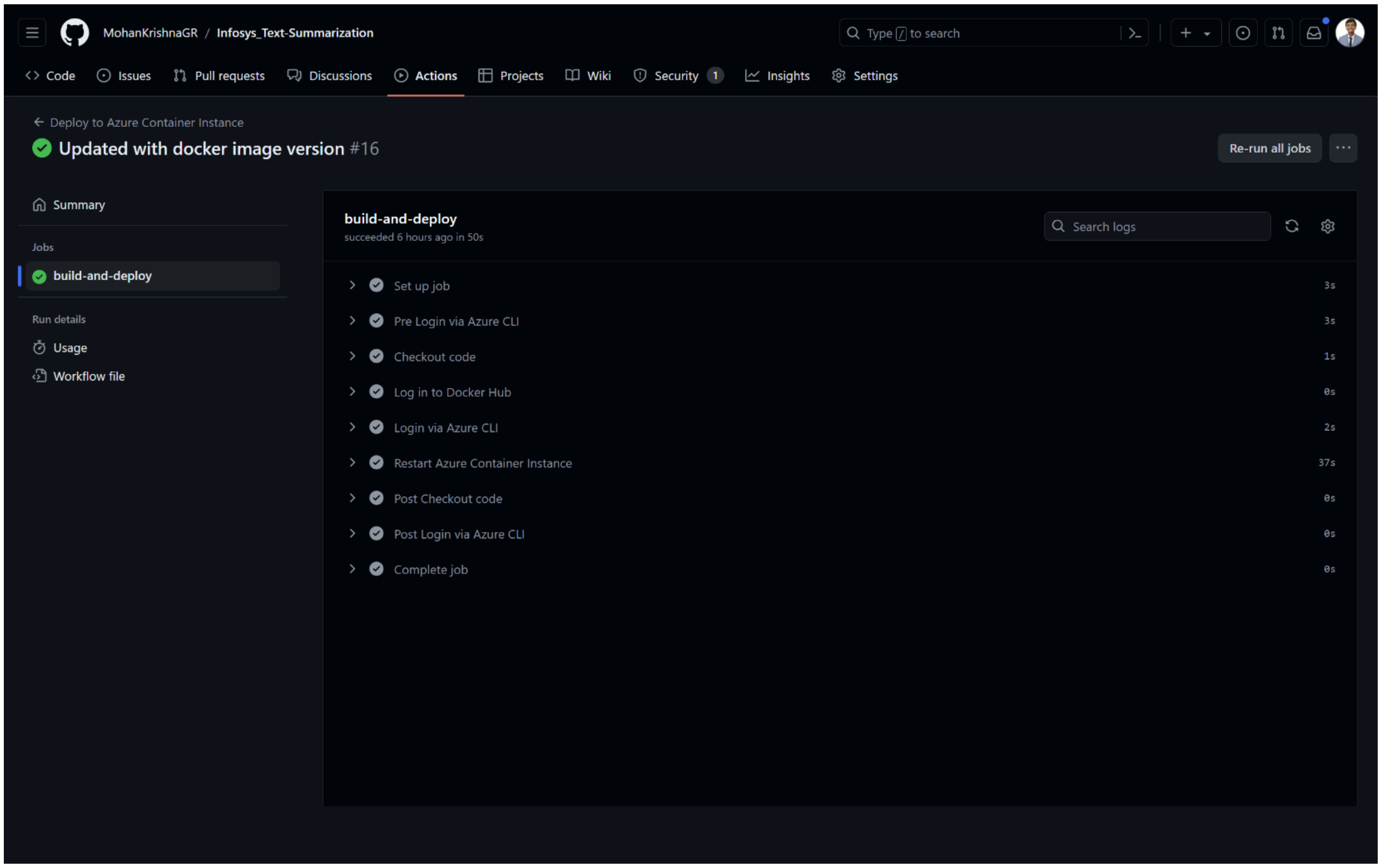
- Utilized Azure CLI to access the container instance – RBAC details & logs.



```
PS C:\Users\MOHAN> az container logs --name mohankrishnagr-text-summ-final --resource-group TextSumm
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
The argument `trust_remote_code` is to be used with Auto classes. It has no effect here and is ignored.
INFO:     Started server process [15045]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO:     10.92.0.5:51214 - "GET / HTTP/1.1" 200 OK
WARNING:  Invalid HTTP request received.
WARNING:  Invalid HTTP request received.
INFO:     10.92.0.5:64195 - "GET / HTTP/1.1" 200 OK
INFO:     10.92.0.5:50447 - "GET / HTTP/1.1" 200 OK
INFO:     10.92.0.5:50447 - "GET /favicon.ico HTTP/1.1" 404 Not Found
WARNING:  Invalid HTTP request received.
WARNING:  Invalid HTTP request received.
INFO:     10.92.0.5:50473 - "GET / HTTP/1.1" 200 OK
WARNING:  Invalid HTTP request received.
INFO:     10.92.0.5:50477 - "GET /login HTTP/1.1" 404 Not Found
WARNING:  Invalid HTTP request received.
INFO:     10.92.0.4:64898 - "GET / HTTP/1.1" 200 OK
INFO:     10.92.0.5:55876 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     10.92.0.5:58735 - "GET / HTTP/1.1" 200 OK
WARNING:  Invalid HTTP request received.
WARNING:  Invalid HTTP request received.
WARNING:  Invalid HTTP request received.
INFO:     10.92.0.5:65142 - "GET / HTTP/1.1" 200 OK
INFO:     10.92.0.4:58784 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     10.92.0.5:65199 - "GET /sitemap.xml HTTP/1.1" 404 Not Found
INFO:     10.92.0.5:65200 - "GET /robots.txt HTTP/1.1" 404 Not Found
INFO:     10.92.0.4:60797 - "GET / HTTP/1.0" 200 OK
INFO:     10.92.0.4:61216 - "GET / HTTP/1.1" 200 OK
```

- Attached a screenshot for the Azure Container Instance logs.
- Public IPv4 address & FQDN:
    - Text Summarizer (http://20.219.203.134:8000/)
    - Text Summarizer (http://mohankrishnagr.centralindia.azurecontainer.io:8000/)

CI/CD PIPELINE - AZURE



Deploy to Azure Container Instance

Updated with docker image version #16

build-and-deploy

succeeded 6 hours ago in 50s

```
Set up job                        3s
Pre Login via Azure CLI           3s
Checkout code                     1s
Log in to Docker Hub              0s
Login via Azure CLI               2s
Restart Azure Container Instance  37s
Post Checkout code                0s
Post Login via Azure CLI          0s
Complete job                      0s
```

- Created a GitHub Action Workflow to automate the process of deploying the containerized application using Azure Container Instance (ACI).
  - Source code: .github/workflows/azure.yml
- Even push event on the main branch triggers this workflow.
  - Checks out the repository code to the runner using GitHub Actions
  - Logs in to the Docker Hub.
  - Logins in to the Azure CLI using RBAC credentials, generated using Azure PowerShell commands.
  - Restarts the container instances with the docker image.

- If model.safetensors file could be pushed into repo, then the building of image can also be automated in GitHub actions, for individual pro users in GitHub, it limits the maximum size of a file that can be uploaded.
    - Git LFS also has the maximum limit of 1 GiB of storage only.
    - Whereas, model.safetensors file size is 1.51 GB.

OBSERVATIONS

- The application deployed in Azure has proven to be the optimal choice comparatively to AWS.
    - The inferencing speed in Azure ACI is 10x higher that of AWS, given its computation instance.
    - Thus, resulted in reduced response time by the server to the clients' requests in production.
    - ACI has also a feature to get FQDN.
- Further enhancement in CI/CD pipeline could be considered in order to be more robust.

RESULTS (Ongoing)

- All the approved modules have been implemented successfully.

CONCLUSION AND FUTURE SCOPE (Ongoing)