Healthcare documentation has grown increasingly complex and time-consuming. Medical professionals now

spend significant portions of their workday reviewing lengthy clinical reports to extract essential patient

information. This application uses generative AI technology to create concise summaries of doctor reports.

The system identifies and extracts key clinical findings, diagnoses, treatment plans, and follow-up

recommendations from comprehensive medical documentation. These AI-generated summaries maintain

medical accuracy while presenting information in a structured, accessible format. The technology serves as a

practical tool for healthcare providers who need to quickly understand patient cases without sacrificing critical

clinical details.

**Disclaimer – This application is not a substitute for a licensed medical professional. Please consult**

**your physician before taking any medical decisions**Context

When deploying Large Language Models (LLMs) on Infrastructure as a Service (IaaS) platforms, several

critical considerations must be addressed. The computational requirements are substantial, demanding careful

hardware selection—typically high-performance GPUs or specialized AI accelerators with sufficient VRAM to

accommodate model parameters. Memory requirements extend beyond just the model size, as additional

overhead is needed for tokenization, attention mechanisms, and intermediate computations. Cost optimization

becomes essential, requiring strategies like batch processing, model quantization, or distillation to balance

performance against infrastructure expenses. Scaling considerations also play a vital role, as organizations

must decide whether to implement horizontal scaling with multiple smaller instances or vertical scaling with

fewer, more powerful machines, each approach offering different benefits for throughput, redundancy, and

cost management.  
  
Google FLAN

Google Flan is a family of fine-tuned large language models developed by Google Research. The name "Flan"

stands for "Fine-tuned LAnguage Net," and it represents a major effort by Google to improve the performance

of language models on a wide range of natural language processing tasks. The key innovation behind Flan is

instruction tuning—fine-tuning models using a large and diverse set of tasks framed as instructions (e.g.,

"Translate this sentence to French" or "Summarize this paragraph"). This approach enables the models to

generalize better to unseen tasks simply by following natural language prompts, making them more effective

and accessible for end users.

One of the most well-known iterations is **Flan-T5**, which builds upon the T5 (Text-To-Text Transfer

Transformer) architecture. Flan-T5 has been trained with a large mixture of datasets and instructions, resulting

in significantly improved zero-shot and few-shot capabilities compared to its base models. These models are

open-source and come in various sizes, from small versions suitable for edge devices to large-scale models

capable of competing with other state-of-the-art LLMs. Google's Flan models have been widely adopted for

research and practical applications and have helped push the boundaries of what instruction-tuned models

can achieve in the broader AI ecosystem.

Architecture design  
What are you expected to do?

Steps to perform – What are you expected to do

1. 2. Create an EC2 instance using the below specifications

AMI – Ubuntu 22.04 LTS

Size – M7i-flex.large

Root Storage – 30GB

Download the application Python scripts to the EC2 instance

3. Setup the environment to execute the LLM

4. Execute the Python program

5. Observe the output generated by the LLM

Setting up the environment

SSH into the EC2 instance and run the below commands one by one

• sudo apt update

• sudo apt install unzip

• wget https://bootstrap.pypa.io/get-pip.py

• sudo python3 get-pip.py

• mkdir summarizer

• cd summarizer

• #This command downloads the application code

wget https://d6opu47qoi4ee.cloudfront.net/genAI/ccaws/flan/summarizer.zip

• unzip summarizer.zip

• sudo pip3 install virtualenv

• virtualenv venv

• source venv/bin/activate  
  
Setting up the environment – Contd.

• pip3 install transformers torch accelerate

• #This command will execute the Python program

python3 app\_tester.py

The above command will take ~2 and a half minutes to complete execution on the first run since it involves

downloading of the LLM as well. Once the LLM is downloaded, subsequent executions should take ~1

minute .

If you have a paid AWS account, you can also attempt this on a c4.4xlarge instance. In that case, the initial

execution of the command will take ~30 seconds while later executions will take ~10-15 seconds due to

the larger size of the instance.

The execution of the program will result in the creation of a folder called summaries containing the output

of the program which is a JSON file. This file can be opened to observe the generated output by using the

below commands

• cd summaries

• cat summary <Press the tab key on your keyboard at this point to autocomplete the file name which is

randomly generated>