Cal Poly Pomona

Generative AI (Gen AI) (CS 4990.04)

Mini Project 1

Talha Ahmed

018960110

Submitted to: Professor Sai Chandra Kosaraju

Dated: Monday, 10 November 2025

Table of Contents

Title

Generating Patient Discharge Summaries using ***distilGPT2***.

GitHub Repository

Link: Gen-AI-Mini-Project-1

Problem Statement

Clinical documentation is an essential yet time-intensive task in healthcare systems. One of the most critical documents in a patient's medical record is the **discharge summary**, which encapsulates the entire hospital course including admission details, diagnoses, treatments, and follow-up instructions.

The manual creation of these summaries by physicians often leads to:

1. High administrative burden
2. Variability in detail and quality
3. Delays in communication to follow-up care providers.

This project focuses on building a **Generative AI model** that automates the creation of **Patient Discharge Summaries** based on structured hospital data and health history. Using a **fine-tuned transformer-based model (distilGPT2)** that is Autoregressive in nature, the system learns to generate coherent, clinically relevant summaries directly from patient admission records, diagnostic information, and treatment notes.

The aim is to:

1. Reduce documentation time for medical staff,
2. Improve standardization in discharge communication,
3. Demonstrate how autoregressive transformer-based models can assist in healthcare NLP applications.

Dataset

The dataset used for this project is the BHC-MIMIC-IV Summary Dataset, available on Kaggle: **BHC_MIMIC-IV_SUMMARY**.

The dataset is derived from the MIMIC-IV clinical database, which contains de-identified health-related data associated with hospital admissions. The dataset provides pairs of input text (patient details) and target text (human-written discharge summaries).

Dataset Composition

Each record in the dataset consists of:

1. **Input:** Structured text describing patient demographics, diagnoses, medications, lab results, and hospital course.
2. **Target:** Human-written discharge summary corresponding to the patient record.

The dataset provides a rich mapping between **structured medical inputs** and **narrative clinical outputs**, making it ideal for supervised language generation tasks.

Dataset Loading and Structure

1. The dataset was accessed programmatically via **KaggleHub**, ensuring reproducibility and easy integration into Colab environments.
2. The data was read into a pandas DataFrame with columns such as input (source text) and target (reference summary).
3. Initial inspection revealed the presence of **embedded instructions or prompts** (e.g., “Create a summary based on the following information”), which were later removed during the cleansing phase.
4. On average:
    a) Input records contain 400–800 tokens.
    b) Target summaries range between 700–1000 tokens.

This dataset was sufficiently large and diverse to train and evaluate a medium-scale transformer model such as **distilGPT2**, with a maximum sequence length of 1024 tokens.

Challenges

Working with clinical text and transformer-based generative models presented several key challenges:

Data Quality and Cleaning

The raw dataset contained irregular formatting, newline characters, and redundant prompts such as “summarize” or “write a discharge summary.”

These artifacts inflated token counts and could confuse the model during training. Regular expressions and text normalization steps were required to produce structured, clean input-output pairs.

Long Context Management

The medical input sections often exceeded the token limit of **1024 tokens** supported by distilGPT2. To handle this, text truncation and careful tokenization were performed to maintain the most informative sections without exceeding context limits.

Domain-Specific Language

Medical terminology is dense and context sensitive. Since **distilGPT2** was pretrained on general English text (not clinical data), it sometimes struggled with domain-specific phrases or abbreviations.
This made fine-tuning crucial to adapting it to the clinical style of the dataset.

Computational and Resource Constraints

Training even a distilled GPT model on long sequences requires significant memory. Batch sizes and sequence lengths had to be adjusted dynamically to avoid out-of-memory errors, leading to slower training iterations.

Evaluation of Generated Summaries

Evaluating generative outputs for clinical correctness is inherently challenging. Metrics like **perplexity** and **validation loss** helped monitor fluency, but they do not directly capture factual accuracy.

Manual inspection of generated summaries was necessary to ensure medical plausibility and coherence.

Data Preprocessing

Data preprocessing was carried out in two stages: **Data Cleansing** and **Data Partitioning**, each handled in separate notebooks.

Data Cleansing

The cleansing process aimed to prepare the raw dataset from Kaggle for model training. The dataset was loaded directly from **KaggleHub** using an authenticated key stored in Google Drive. Following steps were involved in data cleansing.

Dataset Loading

1. The dataset "BHC-MIMIC-IV Summary" was imported using the KaggleHub adapter in the pandas format.

2. The data contained two main columns:

   a) **input**: Raw patient information, including clinical history, demographics, diagnoses, and treatment details.

   b) **target**: The discharge summary to be generated by the model.

Inspection and Understanding

1. The notebook displayed the dataset's column names and row counts to verify data integrity.

2. It identified that the input column contained multiple sections and prompts embedded in the raw text.

Section Extraction

1. Regular expressions were used to detect and extract medical sections or headings (e.g., "History of Present Illness," "Medications," "Procedures").

2. Headings were identified using the pattern ***(^[A-Za-z \-/()]+?):*** with multiline flags.

3. A set of unique section headings was created to understand the dataset structure.

Prompt Removal

1. The dataset contained artificial prompts or instruction-like phrases such as:

   a) "Create a summary based on the following information,"

   b) "Generate a brief hospital summary,"

   c) "Summarize,"

   d) "Write a discharge summary."

2. These redundant lines were programmatically detected and removed to ensure the model only trained on clean, meaningful text.

Text Cleaning and Standardization

1. Unnecessary whitespace, newline characters, and formatting inconsistencies were removed.

2. Text was standardized (e.g., uniform case) for consistent tokenization.

3. Non-informative data entries or incomplete rows were dropped.

Output Preparation

1. The cleaned dataset was saved as cleaned_dataset.csv for further processing.

2. This version contained structured, well-formatted input-output pairs suitable for training language models.

Data Partitioning

After cleaning, the dataset was loaded from Google Drive and partitioned into **training**, **validation**, and **test** sets. Following steps were involved in data partition.

Dataset Loading

1. The ***cleaned_dataset.csv*** file was read into a pandas DataFrame named structuredCleanedData.

Splitting the Dataset

1. The dataset was split using **scikit-learn's train_test_split** function.

2. The split ratio was:

   a) **Training Set:** 70%

   b) **Validation Set:** 15%

   c) **Test Set:** 15%

3. Random shuffling was applied with a fixed random seed (random_state=42) to ensure reproducibility.

Conversion to Hugging Face Dataset Format

1. Each partition was converted from pandas DataFrames to **Hugging Face Dataset objects** using the datasets library.

2. The columns used were:

   a) source → the input (patient information)

   b) target → the discharge summary

Dataset Export

1. Each processed dataset (training, validation, testing) was saved back to Google Drive in CSV format.

2. The notebook verified successful file creation by checking file existence after export.

Model Training

Model training was implemented in the **"GenAI_Mini_Project_1_Model_Training.ipynb"** notebook using the **Hugging Face Transformers** framework. The focus was on fine-tuning the **distilGPT2** model; A compact yet powerful generative autoregressive transformer to generate coherent, medically contextual discharge summaries.

Model Selection

The **distilGPT2** model was chosen for the following reasons:

1. It's a distilled, lighter version of GPT-2, reducing parameters by ~40% while retaining most language modeling capabilities.
2. It supports sequence generation up to **1024 tokens**, suitable for medium-length clinical summaries.
3. It is compatible with Hugging Face's Trainer API, allowing for efficient fine-tuning and evaluation.
4. It is Autoregressive in nature.

The architecture is based on the **Transformer decoder** design, trained using a **causal language modeling (CLM)** objective to predict the next token given all previous ones (autoregressive).

Tokenization and Data Preparation

The preprocessed and partitioned datasets (training, validation, and testing CSV files) were loaded using the **Hugging Face Datasets** library.

Encoding

1. Both input (source) and output (target) texts were concatenated and tokenized together for causal modeling.
2. Sequences exceeding **1024 tokens** were truncated to comply with the model's context window.

Formatting for PyTorch

The tokenized datasets were converted into PyTorch tensors with the following structure:

1. input_ids: Tokenized text inputs.
2. attention_mask: Binary mask for padded tokens.

Training Configuration

The model was trained using the **Hugging Face Trainer API** with the following hyperparameters:

Parameter	Value
Model	distilGPT2
Token Limit	1024
Learning Rate	1e-1
Batch Size	2
Epochs	6
Loss Function	ForCausalLMLoss
Evaluation Metric	eval_loss, Rouge (Rouge 1, Rouge 2, Rouge L), BertScore

Training was performed on Google Colab with GPU acceleration enabled.

Training Workflow

Trainer Setup

```
modeltrainerArguments = Seq2SeqTrainingArguments(
    output_dir=projectFolderPath,
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=16,
    num_train_epochs=6.0,
    learning_rate=1e-4,
    weight_decay=0.01,
    warmup_ratio=0.1,
    logging_strategy="steps",
    logging_steps=50,
    eval_strategy="steps",
    eval_steps=250,
    save_steps=250,
    save_total_limit=2,
    lr_scheduler_type="cosine_with_restarts",
    fp16=False,
    bf16=True,
    dataloader_pin_memory=True,
    report_to="none",
    max_grad_norm = 0.5,
    group_by_length = True,
    metric_for_best_model="eval_loss",
    greater_is_better=False,
    load_best_model_at_end=True,
    seed=42,
    remove_unused_columns=False,
    predict_with_generate=False,
    generation_num_beams=1,
    # dataloader_num_workers=2,
)
```

Training Execution

```
trainer = Seq2SeqTrainer(
    model=model,
    args=modeltrainerArguments,
    train_dataset=tokenizedDataset["train"],
    eval_dataset=tokenizedDataset["validation"],
    data_collator=collateFeaturesOfDataset,
    processing_class=tokenizer,
)

trainer.train()
```

1. Loss values were logged per epoch.
2. Validation perplexity was computed at the end of each epoch to monitor learning progress.

Results

Quantitive Evaluation

The model's performance was evaluated using **training loss**, **validation loss**, and **perplexity**.

- **Training Loss:** Decreased steadily over epochs, indicating effective learning.
- **Validation Loss:** Plateaued after approximately 3 epochs, suggesting convergence.

Although the exact numeric results vary by run, a typical fine-tuning session achieved:

Metric	Value
Final Training Loss	2.909600

Final Validation Loss	2.782992
Rouge-1	0.2625
Rouge-2	0.0683
Rouge-L	0.1147
Rouge-LSum	0.162
BertScore	0.8109372292041779

Interpretation

1. **ROUGE Scores:** Indicate moderate lexical overlap with reference summaries, which is expected in free-text generation where paraphrasing and lexical variety occur.
2. **BERTScore (F1 = 0.81):** Demonstrates high semantic similarity, meaning the generated summaries captured the *meaning and structure* of the reference summaries effectively, even if word choices differed.

Qualitative Evaluation

Generated discharge summaries were manually inspected to assess fluency, coherence, and medical realism. The outputs showed that the model could effectively summarize key information and generate structured, human-like summaries.

Example

Input

Summarize the following hospitalization for the discharge summary.
Chief Complaint: Cough, wheeze
Discharge Diagnosis: Acute Hypoxic Respiratory Failure Dyspnea on Exertion Reactive airway disease possibly viral bronchitis Congestive heart failure with preserved ejection fraction Hyponatremia Chronic kidney disease Diabetes mellitus and steroid-induced hyperglycemia
Discharge Instructions: Mrs. , was our pleasure caring for you at . You were admitted to the hospital with shortness of breath, wheezing and cough that we think were most likely caused by bronchitis, possibly from a viral ...

Output

The patient presented to the emergency department and was evaluated by the orthopedic surgery team. The patient was found to have a pulmonary embolism and was started on a heparin drip. She underwent a laparoscopic cholecystectomy, which went well without complication . After a brief, uneventful stay in the PACU, the patient arrived on the floor hemodynamically

Observations

1. The model produced grammatically correct and contextually coherent sentences.
2. Maintained logical medical flow (presentation → diagnosis → management → discharge plan).
3. Occasionally generated repetitive phrases or generalized statements, typical for GPT-like models fine-tuned on limited data.

Discussion & Insights

1. The **distilGPT2** model successfully generalized from structured clinical input to fluent narrative summaries.
2. Despite not being trained on medical-specific corpora, fine-tuning on the BHC-MIMIC-IV dataset enabled it to capture discharge-level coherence.
3. The **ROUGE-1** score of 0.26 and **BERTScore of 0.81** together confirm that generated texts were semantically faithful even when lexically varied.
4. Performance could further improve with larger datasets, extended context (2048+ tokens), and domain-adaptive pretraining.

Future Enhancements

1. This project is an experiment on generating medical discharge summaries from the raw clinical data. This experiment will aid in targeting for the final project on the same topic.
2. Fine-tune larger models like **GPT-2 Medium** or **T5-base** to capture longer patient histories.
3. Incorporate **domain-specific embeddings** (e.g., BioBERT or ClinicalBERT) for better medical terminology comprehension.
4. Evaluate with additional metrics like **BLEU**, **METEOR**, and **human evaluation** for clinical correctness.