GenAI Assignment

Name: Yashmitha Shailesh

SRN: PES2UG23CS715

Section: K

EmailID: [yashmithashailesh@gmail.com](mailto:yashmithashailesh@gmail.com)

Mobile number: 8618764905

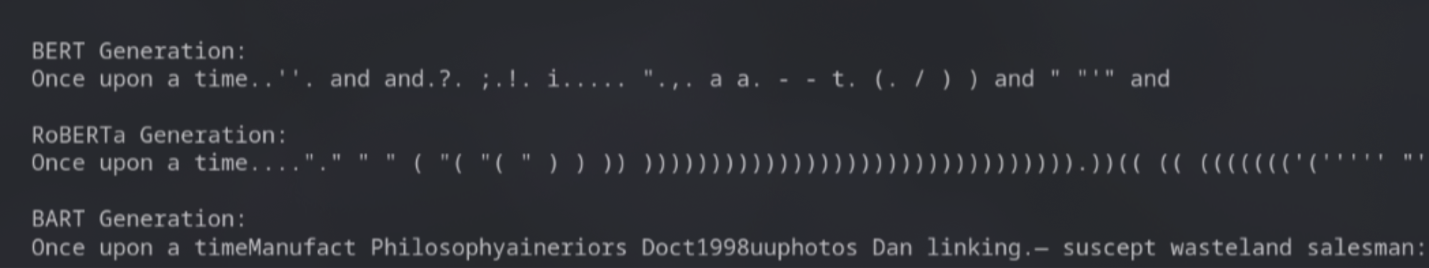
Github link: <https://github.com/yashmithaa/Genai>

# Assignment 1

**Experiment 1:** Text Generation

Task: Try to generate text using the prompt: "Once upon a time"

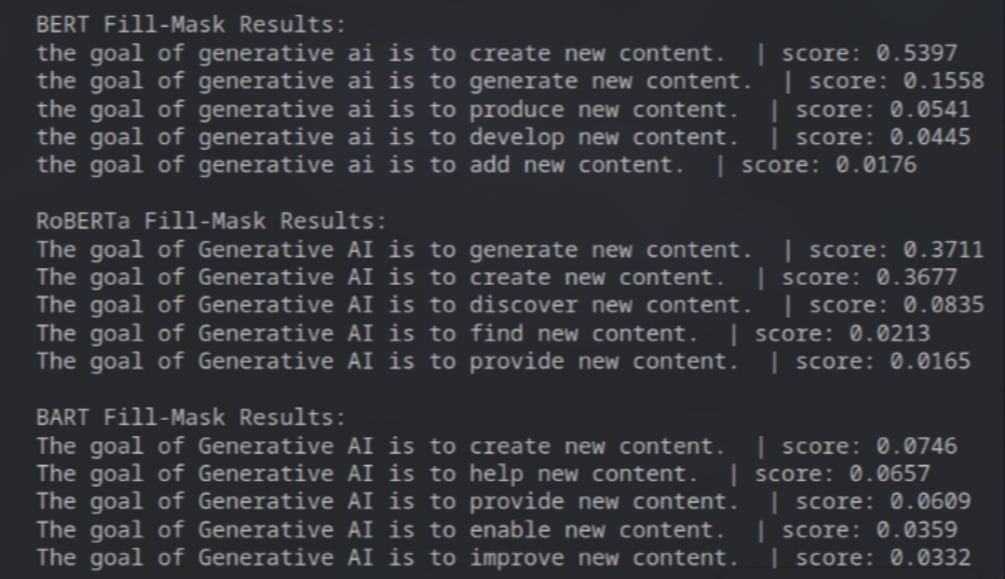
Observations:



**Experiment 2:** Masked Language Modeling (Missing Word)

Task: Predict the missing word in: "The goal of Generative AI is to [MASK] new content."

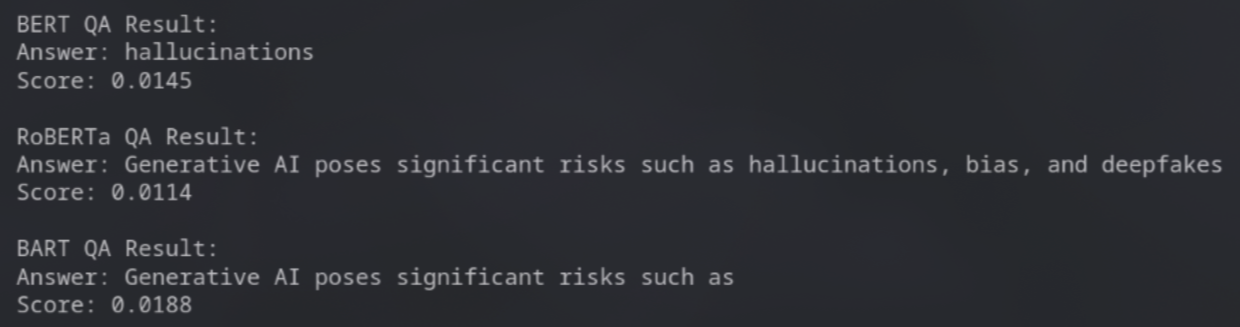
Observations:

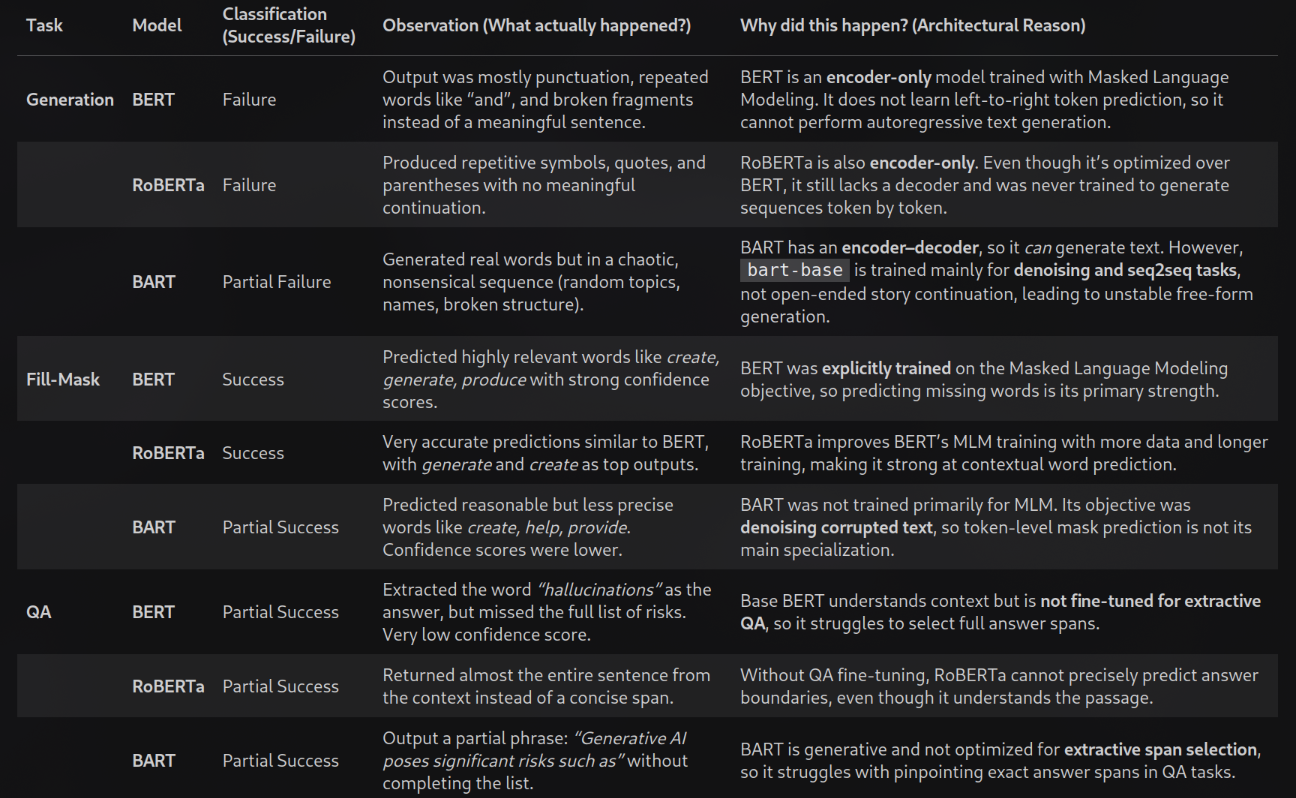


**Experiment 3:** Question Answering

Task: Answer the question "What are the risks?" based on the context: "Generative AI poses significant risks such as hallucinations, bias, and deepfakes."

Observations:





**Key concepts understood**

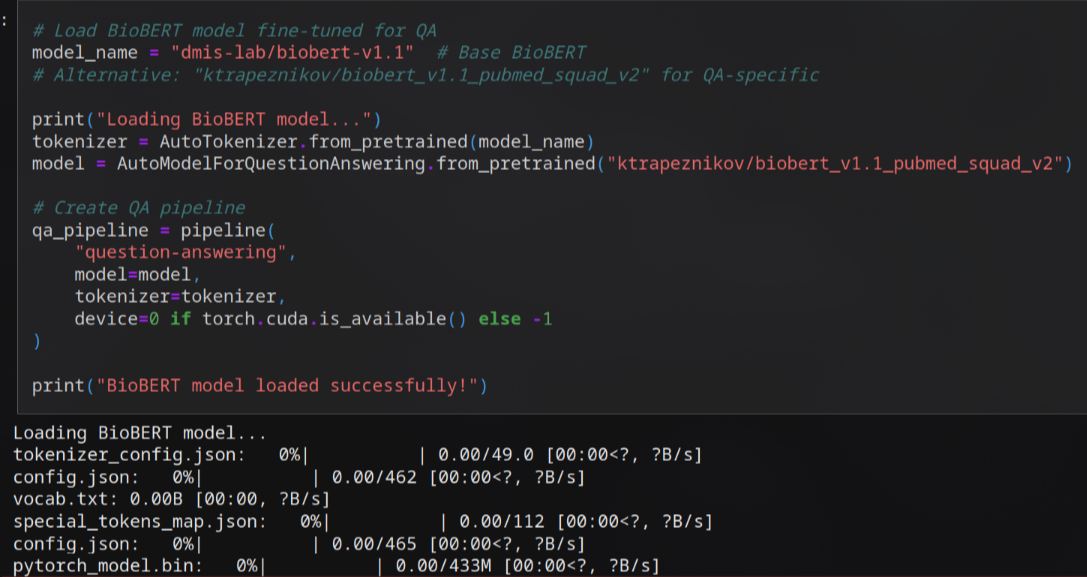
* Transformer families and capabilities:
  + Encoder-only (BERT, RoBERTa): strong at masked token prediction, poor for autoregressive generation.
  + Encoder–decoder (BART): suited for seq2seq/denoising tasks and can generate, but not necessarily optimal for open-ended autoregressive continuation without proper fine-tuning.
  + Hugging Face pipelines: quick way to run tasks (text-generation, fill-mask, question-answering) with minimal boilerplate.
* Task-model alignment: models must match task objectives (e.g., MLM models for fill-mask; decoder/generative models for text generation).
* Evaluation: qualitative checks (success/failure), inspect outputs and confidence scores, and reason about architectural causes for failures.

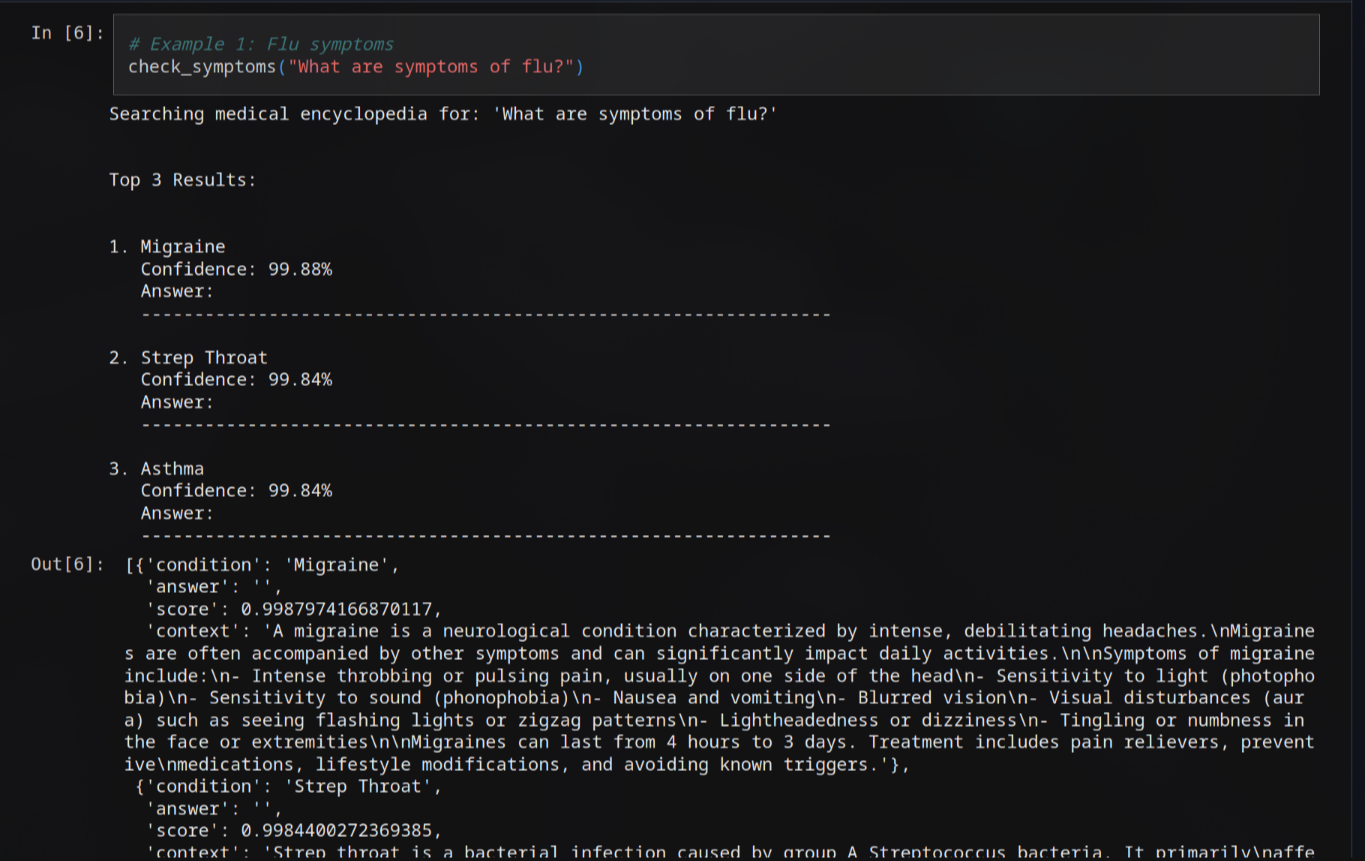
**The solution implemented**

1. bert-base-uncased, roberta-base, facebook/bart-base used to compare behaviors across tasks.
2. Pipelines initialized
   1. Text generation pipelines for each model (attempted autoregressive generation).
   2. Fill-mask pipelines for MLM-style prediction.
   3. Question-answering pipelines for extractive QA.
3. Experiments performed
   1. Text generation: same prompt ("Once upon a time") used to observe each model’s ability to continue text; used sampling, temperature, top-p and penalties.
   2. Fill-mask: masked sentences formatted per model conventions and top predictions inspected.
   3. QA: simple context+question to test extractive span selection; answer and score displayed.
4. Observations / outcomes (brief)
   1. BERT / RoBERTa: fail at open-ended generation; successful at fill-mask.
   2. BART: partial generation ability but unstable for free-form continuation; moderate fill-mask performance.
   3. QA with base checkpoints: partial/low-quality extractive answers because base models were not fine-tuned for QA.

# Assignment 2

**Medical symptom checker**





**Key concepts understood**

1. Pretrained domain models: BioBERT (biomedical pretraining) used to improve medical-text understanding.
2. Extractive QA behavior: model returns answer span, score (confidence), and handles impossible/no-answer cases.
3. Result filtering and ranking: apply minimum score threshold and sort by confidence to produce top-K candidates.
4. Safety & limitations: model outputs are probabilistic and not medical advice; include a disclaimer and encourage professional consultation.

**The solution implemented**

1. Tokenizer loaded from a base BioBERT checkpoint.
   1. QA model loaded from a BioBERT checkpoint fine-tuned for SQuAD-style QA ("ktrapeznikov/biobert\_v1.1\_pubmed\_squad\_v2").
   2. Creates a QA pipeline with device selected automatically.
2. Medical knowledge source
   1. Builds MEDICAL\_ENCYCLOPEDIA: mapping of condition name → description/context.
3. Symptom-checking workflow (check\_symptoms)
   1. Accepts a natural-language question, top\_k, and min\_score.
   2. Iterates all conditions, runs the QA pipeline using each condition's context.
   3. Keeps answers above min\_score and records condition, answer, score, and context.
   4. Sorts results by confidence and prints the top-K formatted results.
   5. Returns the top-K results programmatically.
   6. Detail extraction (get\_condition\_details)