GENAI LAB 1

SRN- pes2ug23cs129

Section – B

Name – bhuvan hebbar

Inference: Learning Generative AI & NLP Using Google Colab

Hand – typed :

Working through this Google Colab notebook was a helpful introduction to the fundamentals of Generative AI and Natural Language Processing. Instead of focusing only on theory, the notebook allowed hands-on interaction with real models, which made the concepts much easier to understand and relate to real-world applications.

The initial setup introduced the Hugging Face ecosystem and the transformers library, which clearly showed how modern NLP development is done today. Hugging Face felt like a central hub for AI models, where powerful pre-trained models can be used directly without needing to train them from scratch. The use of the pipeline() function was especially useful, as it simplified complex tasks like text generation, summarization, and question answering into a few lines of code.

One of the most interesting parts of the notebook was comparing different text generation models. Using the same prompt with a smaller model (distilgpt2) and a slightly larger one (gpt2) clearly showed how model size affects output quality. The smaller model was fast but produced shallow and repetitive text, while the larger model generated more coherent and meaningful responses. This comparison helped in understanding the trade-off between efficiency and quality when selecting models.

The notebook also explained what happens behind the scenes in NLP. Concepts like tokenization, part-of-speech tagging, and named entity recognition were demonstrated using simple examples. Seeing how sentences are broken into subwords, how grammatical roles are assigned, and how important entities are extracted from text gave a better understanding of how language models process human language.

More advanced tasks such as summarization, question answering, and masked language modeling further strengthened this understanding. Different summarization models showed noticeable differences in output quality, again reinforcing the idea that better results often require more computational resources. The question-answering task highlighted how models extract answers strictly from the given context, rather than reasoning beyond it.

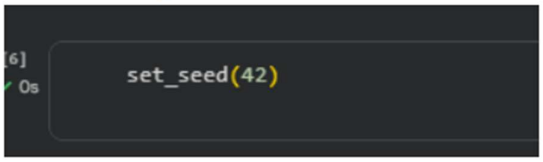
Overall, this notebook provided a clear and practical learning experience. It helped connect theoretical NLP concepts with real implementations and emphasized the importance of

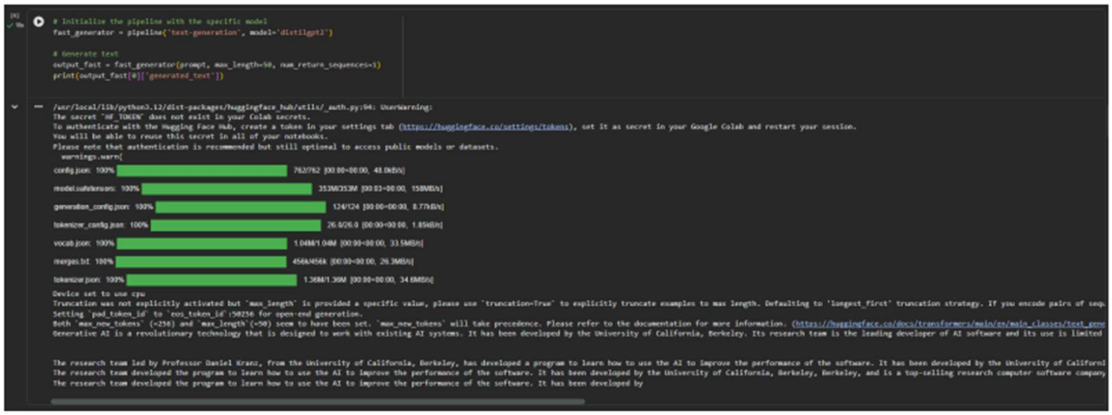
choosing the right model based on the task. The hands-on approach made Generative AI feel less abstract and more applicable, making it a valuable starting point for anyone learning NLP.

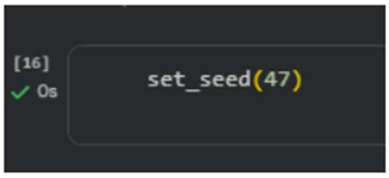
With the help of little AI as mentioned by TA :

Seed Values

- Experimented by setting different random seed values.
- Observed how changing the seed affects the generated output.
- Different seeds produce different text even for the same prompt.







Difference Between Distilled Model and Main Model Outputs

Distillation mainly reduces the **depth of the network**, while keeping the vocabulary and embeddings mostly unchanged.

distilgpt2

- Loads faster
- Generates text quickly
- Produces shorter responses
- Slight repetition
- Limited long-range coherence
- Simpler sentence construction

gpt2

- Slower generation
- Better quality output
- Smoother sentence flow
- Stronger overall coherence
- Less repetition

- More human-like text continuation

Main Model (Teacher Model)

The main model is the original large-scale model trained directly on huge datasets.

Characteristics:

- Large number of parameters
- High accuracy and better generalization
- Requires more computation and memory
- Commonly used for research or cloud-based inference

Analogy:
Like an experienced expert — very knowledgeable but slower and expensive to consult.

Distilled Model (Student Model)

A distilled model is a compressed version of the main model trained using knowledge distillation to mimic its behavior.

Characteristics:

- Smaller and lightweight
- Faster inference
- Low memory and power usage
- Slightly reduced accuracy but still close to the original
- Suitable for mobile devices and real-time systems

Analogy:
Like an assistant trained by the expert — quicker and more efficient.

Transformer Architecture

Transformers consist of:

- **Encoder** → understands input
- **Decoder** → generates output text

GPT-2 (Generative Pre-trained Transformer-2)

GPT-2 is a **decoder-only transformer model** designed specifically for text generation.

Key features:

- Uses masked self-attention
- Autoregressive (predicts one token at a time)
- Focused mainly on generation tasks
- Not ideal for tasks like summarization

What GPT-2 is Trained For

GPT-2 is trained using **next-token prediction**, enabling it to generate fluent text such as:

- Stories
- Paragraphs
- Code-like content
- Conversations

How GPT-2 Predicts the Next Word

1. Softmax Layer

- Converts raw logits into probabilities for each token.

2. Sampling Techniques

- **Temperature sampling:** controls randomness (lower temperature → safer outputs)
- **Top-p sampling:** selects smallest group of tokens whose cumulative probability ≥ p
- **Top-k sampling:** selects only the top k probable tokens

3. Context Encoding

- Text → tokens → embeddings → transformer layers → prediction

---

Task	Model	Classification	Observation (What happened)	Why this happened
Text Generation	BERT	Failure	The model couldn't really generate text properly and either failed or gave meaningless output.	BERT is an encoder-only model and isn't meant to generate the next word in a sentence.

Task	Model	Classification	Observation (What happened)	Why this happened
	RoBERTa	Failure	Similar to BERT, text generation didn't work as expected.	RoBERTa is also encoder-only and is designed for understanding text, not generating it.
	BART	Success	BART was able to continue the sentence and generate readable text.	BART has a decoder and is trained for sequence-to-sequence generation.
Fill-Mask	BERT	Success	It predicted sensible words like "create" or "generate" for the masked token.	BERT is trained using masked language modeling, so this task fits it well.
	RoBERTa	Success	RoBERTa gave accurate and fluent predictions for the missing word.	RoBERTa is an improved version of BERT and performs very well on MLM tasks.
	BART	Partial Success	It gave predictions, but they were not as clean or consistent as BERT or RoBERTa.	BART supports masking but is mainly optimized for generation tasks.
Question Answering	BERT	Poor / Partial	The answers were incomplete or sometimes incorrect.	The base BERT model is not fine-tuned for question answering.
	RoBERTa	Poor / Partial	Similar behavior to BERT, with weak or unreliable answers.	RoBERTa-base is also not trained specifically for QA tasks.
	BART	Poor / Failure	The output was mostly irrelevant or incorrect.	BART-base is not designed for extractive QA without fine-tuning.