GenAI HandsOn-1

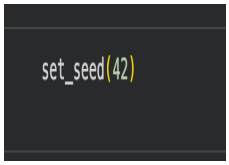
Name : Anushka Mandal	Section : B	SRN : PES2UG23CS083

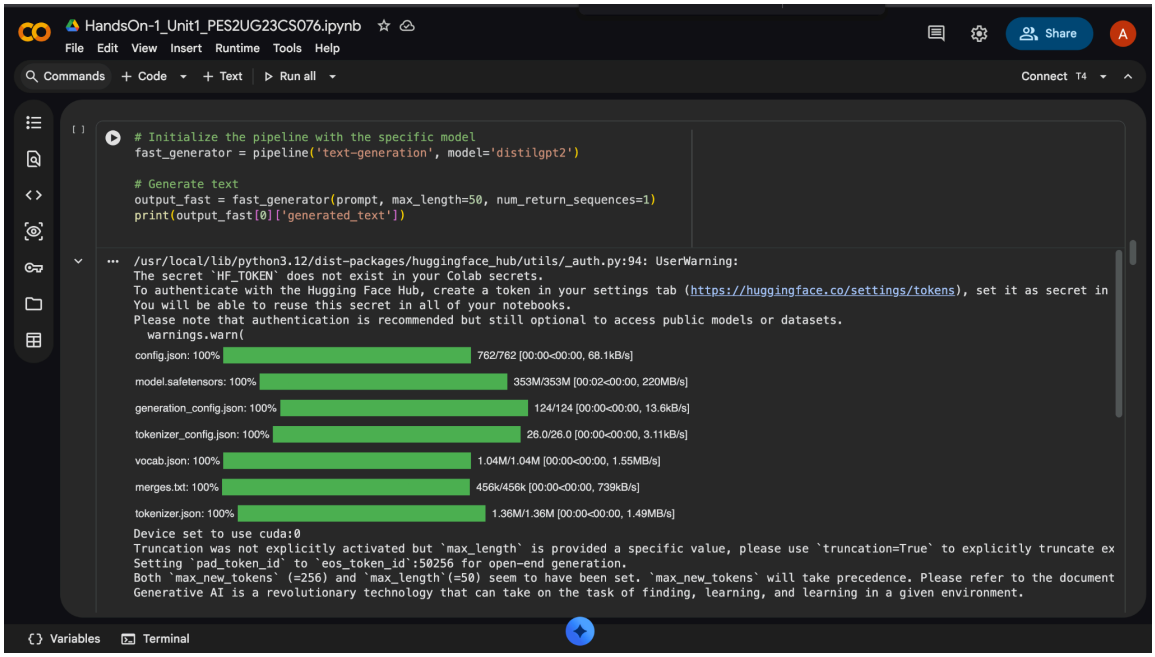
Model Comparisons: Standard vs. Distilled

Standard Model (e.g., GPT-2): These models have a larger architecture, enabling better language understanding and more consistent text generation. They rely on probability-based next-token prediction using the Softmax function.
• **Distilled Model (e.g., DistilGPT-2):** These models compress the knowledge of the larger model into a lighter version, making them faster and more memory-efficient. This efficiency gain, however, may come at the cost of reduced precision and slightly weaker coherence in outputs.

SEED Value

The set_seed() function controls how randomness begins in a program by fixing an initial reference point for the random number generator. It works like locking in the starting position before a shuffled process begins. When the same seed value (such as set_seed(42)) is reused, the sequence of random numbers generated remains identical, resulting in the same outputs for any randomness-dependent tasks like text generation. This consistency is essential for reproducing experimental results. Changing the seed value alters the starting point, producing a new sequence of random values and therefore different outputs. The seed's numeric value does not affect quality; it only guarantees a distinct but repeatable pattern of randomness.






```
set_seed(68)
```

```
smart_generator = pipeline('text-generation', model='gpt2')

output_smart = smart_generator(prompt, max_length=50, num_return_sequences=1)
print(output_smart[0]['generated_text'])
```

```
config.json: 100% 665/665 [00:00<00:00, 20.2kB/s]
model.safetensors: 100% 548M/548M [00:04<00:00, 248MB/s]
generation_config.json: 100% 124/124 [00:00<00:00, 6.86kB/s]
tokenizer_config.json: 100% 26.0/26.0 [00:00<00:00, 1.23kB/s]
vocab.json: 100% 1.04M/1.04M [00:00<00:00, 8.38MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 3.32MB/s]
tokenizer.json: 100% 1.36M/1.36M [00:00<00:00, 10.3MB/s]
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `tr
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take
Generative AI is a revolutionary technology that enables a wide range of intelligent systems to work

In this article, we will discuss the main features of the new AI platform, and how it can be used to

1. How Can I Use It?

The concept of AI is not new. It has been used by many people to measure their mental health and heal

It is based on the premise that AI is a way for humans to move towards a more efficient way of thinki

In this article, we will explain what AI can do.

What does it do

In this article, we will explain how all of our cognitive and emotional systems interact with the AI

A new way of thinking about AI

A new paradigm for the development of intelligent AI

A new way of thinking about mental health and health-related behaviors
```

What is POS Tagging?

Part-of-Speech (POS) Tagging is a Natural Language Processing (NLP) technique that assigns grammatical labels—such as noun, verb, adjective, adverb, or preposition—to each word in a sentence. It helps machines understand the role and meaning of words based on their context. POS tagging is a fundamental step in many NLP tasks like parsing, information extraction, and text analysis. It improves language understanding by identifying how words function within a sentence.

Word	POS Tag	Full Form	Explanation
Modern	JJ	Adjective	Describes the noun
AI	NNP	Proper Noun	Name of a specific field
systems	NNS	Noun, Plural	Indicates more than one system
analyze	VB	Verb, Base Form	Shows an action being performed
language	NN	Noun, Singular	Refers to a single concept

• Intelligent (JJ – Adjective)
• systems (NNS – Noun, Plural)
• are (VBP – Verb, present tense)
• transforming (VBG – Verb, -ing form)
• modern (JJ – Adjective)
• computing (NN – Noun, Singular)
• by (IN – Preposition)
• enabling (VBG – Verb, -ing form)
• automation (NN – Noun, Singular)

Documenting the Gen-AI Handson

Hugging Face can be viewed as the GitHub of Artificial Intelligence, hosting a vast collection of open-source models, datasets, and AI projects. These resources showcase how modern AI systems function and enable developers to leverage pre-trained models rather than building solutions entirely from the ground up.

The Transformers library serves as the connection between Hugging Face and our code. It offers easy-to-use APIs for downloading and applying pre-trained models in custom projects, while supporting multiple deep-learning frameworks.

One of the most useful features of this library is the pipeline() function. It abstracts the entire workflow—preprocessing input data, running model inference, and postprocessing outputs—into a single callable interface, allowing complex NLP tasks to be performed with minimal code.

During the hands-on session, we compared two generative language models: distilgpt2 and gpt2. The distilgpt2 model is a compressed and faster version of gpt2, designed to use less memory and deliver quicker responses. In contrast, the full gpt2 model is larger and generally produces higher-quality and more fluent text.

To ensure reproducibility, a random seed was set at the beginning of the experiment. This guarantees that identical inputs generate the same outputs across multiple runs, which is essential for consistent evaluation.

When both models were given the same prompt, the gpt2 model generated text that was more coherent and grammatically accurate. Although distilgpt2 performed reasonably well, it tended to repeat itself and lose contextual understanding as the generated text became longer.

Tokenization plays a crucial role in preprocessing. Since models cannot directly interpret raw text, sentences are divided into smaller units called tokens, which are then mapped to numerical IDs that the model can process efficiently.

The temperature parameter influences the randomness of text generation. Higher temperature values encourage more diverse and creative outputs by allowing less likely word choices, while lower values result in more predictable and focused text.

Part-of-Speech (POS) tagging is another important preprocessing technique, where grammatical categories are assigned to words. For example, distinguishing whether the word “will” functions as a noun or a modal verb helps the model better understand sentence structure.

Named Entity Recognition (NER) is used to detect and classify entities such as people, organizations, and locations. For instance, NER helps determine whether the word “Apple” refers to a fruit or a technology company by analyzing the surrounding context.

Documenting the Benchmark Assignment

The goal of the benchmark assignment was to analyze how different transformer architectures behave when they are applied to tasks they are not equally optimized for. This exercise helps illustrate why model architecture plays a critical role in Generative AI.

Three transformer models were selected for the benchmark:

- **BERT (bert-base-uncased)** – an encoder-only architecture
- **RoBERTa (roberta-base)** – an improved and optimized encoder-only architecture
- **BART (facebook/bart-base)** – an encoder–decoder architecture

All three models were evaluated on the same set of tasks using Hugging Face pipeline interfaces to ensure a fair comparison.

Tasks Conducted

Text Generation

The models were prompted to generate text. BERT and RoBERTa were unable to produce meaningful output because encoder-only models are not designed for next-token prediction. BART, which includes a decoder, was capable of generating text, although the quality was relatively limited.

Fill-Mask (Masked Language Modeling)

In this task, sentences with masked tokens were provided. BERT and RoBERTa

performed effectively, accurately predicting appropriate words due to their training objective centered on masked language modeling. BART, however, showed weaker performance since this task is not its primary focus.

Question Answering
The models were asked to answer questions based on a given context. The results were inconsistent, as the base versions of these models were not fine-tuned specifically for question answering. Nevertheless, they occasionally produced partial or approximate answers.

Task	Model	Result (Success / Failure)	Observation (What actually happened?)	Reason (Architectural Explanation)
Generation	**BERT**	**Failure**	**Generated nonsensical or random output**	**BERT is an encoder-only model and is not trained for next-word prediction**
Generation	**RoBERTa**	**Failure**	**Did not generate new text; returned the input prompt**	**RoBERTa is also an encoder-only architecture, unsuitable for text generation**
Generation	**BART**	**Failure**	**Generated text, but it was meaningless**	**Although BART is encoder–decoder, the base model was not fine-tuned for coherent generation**

Fill-Mask	**BERT**	**Success**	**Predicted words like “create”, “generate” with high confidence (0.5397)**	**Trained using Masked Language Modeling (MLM)**
Fill-Mask	**RoBERTa**	**Success**	**Produced consistent synonym predictions**	**Trained on more robust and larger datasets than BERT**
Fill-Mask	**BART**	**Success**	**Predicted relevant words but with lower confidence**	**Flexible architecture but primarily optimized for sequence-to-sequence tasks**
QA	**BERT**	**Partial Success**	**Extracted biased or hallucinated answers (Score: 0.017)**	**NSP training helped, but randomly initialized heads caused unstable outputs**
QA	**RoBERTa**	**Failure**	**Extracted only a single keyword (“deepfakes”) (Score: 0.012)**	**Removal of NSP reduced its ability to link questions and answers**
QA	**BART**	**Partial Failure**	**Extracted incomplete phrase (Score: 0.064)**	**Decoder handled longer inputs, but random head**

				initialization led to weak answers

Key Observations

The benchmark clearly showed that:

- Encoder-only models like BERT and RoBERTa excel at understanding-oriented tasks such as fill-mask.
- Encoder–decoder models like BART are more suitable for generation-based tasks.
- Applying a model to tasks outside its intended design leads to suboptimal or failed outputs, which is an expected and meaningful outcome.

Conclusion

This benchmark assignment emphasizes the significance of transformer architecture in Generative AI. It demonstrates that a model's effectiveness depends on how closely the task aligns with its architectural design, reinforcing the foundational concepts introduced in Unit 1.

Project Overview: TL;DR News Article Summarizer

The **TL;DR News Article Summarizer** is a Productivity Agent designed to help users consume information more efficiently by condensing lengthy news articles into concise, three-sentence summaries. Built by Anushka Mandal, this Python-based tool utilizes state-of-the-art Natural Language Processing (NLP) to provide instant, high-quality insights.

The core functionality is powered by the Hugging Face transformers library, specifically leveraging the pipeline('summarization') task. The project offers a dual-model approach: sshleifer/distilbart-cnn-12-6 for rapid, resource-efficient processing and facebook/bart-large-cnn for more nuanced, high-fidelity results.

This flexibility allows the tool to cater to different hardware capabilities and user needs.

A central Python function, summarize_article, handles the logic for text processing and summary generation. It provides a structured output that includes the original article length, word count, and a formatted "TL;DR" section. Additionally, the tool calculates helpful productivity metrics, such as the compression ratio and the estimated reading time saved for the user.

To demonstrate its versatility, the project includes several examples across diverse domains:

- **Technology**: Summarizing the rapid evolution of AI and global regulatory frameworks.
- **Health & Science**: Distilling breakthroughs in Alzheimer's research and clinical trials.
- **Environment**: Condensing complex climate reports on rising sea levels and renewable energy.
- **Business**: Highlighting global economic recovery signs and central bank adjustments.
- **Education**: Outlining the shift toward online learning and student well-being programs.

The implementation is housed in a Jupyter Notebook, featuring a clean hierarchy of setup, model loading, and testing cells. By automating the extraction of key points, this project serves as a powerful utility for professionals and students overwhelmed by information density in the digital age.