Unit 1 Hands-on: Generative AI & NLP Fundamentals

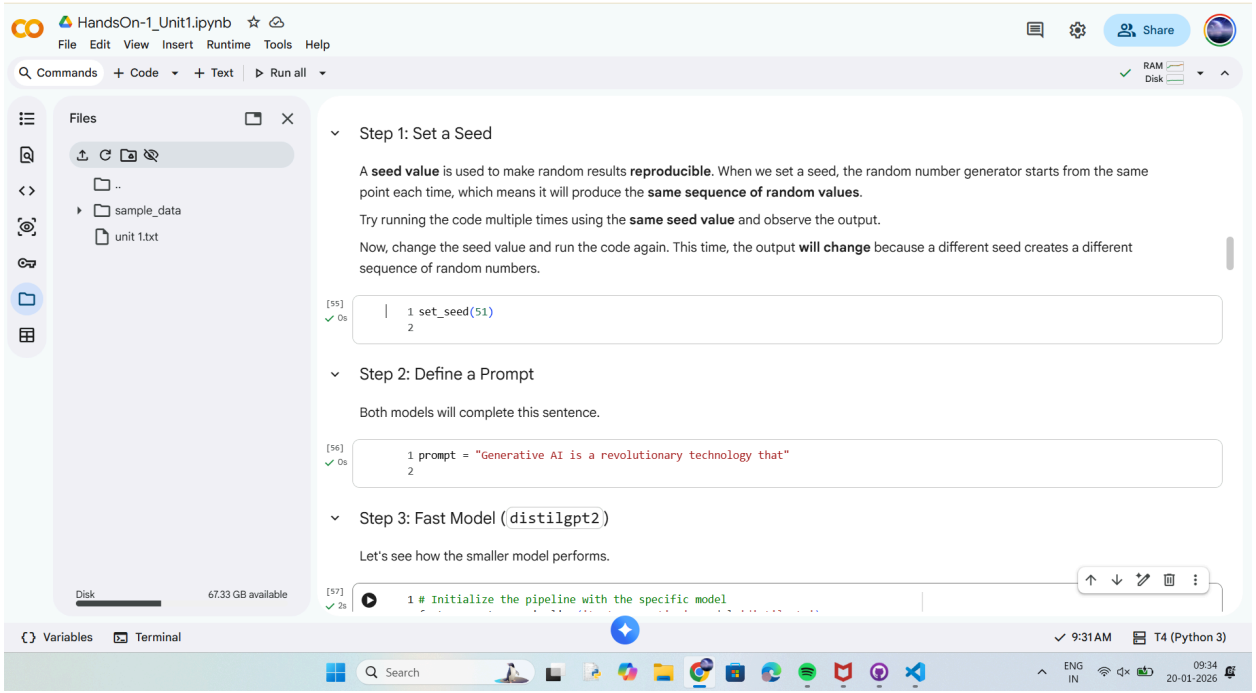
A R KEERTHANA
PES2UG23CS001

Observations from the HandsOn-1_Unit1.ipynb:

Section 1: Introduction & Setup

- Hugging Face acts as a centralized platform for accessing pretrained AI models, datasets, and demos, reducing the need to train models from scratch.
- The transformers library provides a simple interface to load and use state-of-the-art NLP models across different frameworks.
- The pipeline() function abstracts preprocessing, model inference, and post-processing into a single high-level call.
- External libraries like nltk and os support traditional NLP tasks and file handling.
- Loading external text files allows the notebook to treat them as a knowledge base for downstream NLP tasks.

Section 2: Generative AI – Dumb vs Smart Models



- Setting a random seed ensures reproducible text generation outputs.
- Changing the seed value results in different generated text, highlighting the stochastic nature of language models.
- Smaller distilled models like distilgpt2 generate text faster but may lose coherence.
- Larger models like gpt2 generally produce more context-aware and meaningful text.
- Model size and training depth directly affect output quality and relevance.

Section 3: NLP Fundamentals – Under the Hood

Tokenization

- Text must be converted into tokens before being processed by a model.
- Each token is mapped to a unique numerical ID understood by the model.
- Tokenization can split words into subword units rather than whole words.

Part-of-Speech (POS) Tagging

- POS tagging assigns grammatical labels such as noun, verb, or adjective to words.
- It helps models understand sentence structure and grammatical roles.
- The same word can receive different tags depending on context.
- Example of POS Tagging

  Consider the sentence: "The quick brown fox jumps over the lazy dog."

  After performing POS Tagging, we get:

  "The" is tagged as determiner (DT)

  "quick" is tagged as adjective (JJ)

  "brown" is tagged as adjective (JJ)

  "fox" is tagged as noun (NN)

  "jumps" is tagged as verb (VBZ)

  "over" is tagged as preposition (IN)

  "the" is tagged as determiner (DT)

  "lazy" is tagged as adjective (JJ)

  "dog" is tagged as noun (NN)

Named Entity Recognition (NER)

- NER identifies and classifies entities like names, organizations, and locations.
- Pretrained BERT-based models can accurately extract structured information from raw text.
- Filtering by confidence score improves the reliability of extracted entities.

Section 4: Advanced Applications

Summarization

- Different summarization models balance speed and output quality.
- Distilled models generate faster summaries with less detail.
- Larger models produce more coherent and informative summaries.
- Model choice depends on performance requirements and resource constraints.

Question Answering

- Question answering models extract answers directly from a given context.
- The quality of answers depends on how clearly the information appears in the context.
- This task demonstrates practical information retrieval from unstructured text.

Masked Language Modeling

- Masked language modeling predicts missing words based on surrounding context.
- It is the core training objective behind BERT-style models.
- The model assigns probabilities to multiple possible token predictions, showing contextual understanding.

Observations from the PES2UG23CS001_model_comparison.ipynb

This experiment demonstrates that model architecture strongly influences task performance, and models fail when forced outside their training objectives.

Task	Model	Classification	Observation	Architectural Reason
Generation	BERT	Failure	Generated only repeated punctuation (.....) instead of meaningful text.	BERT is an encoder-only model trained for understanding, not autoregressive text generation.
Generation	RoBERTa	Failure	Returned only the input prompt without generating new tokens.	RoBERTa is an optimized encoder-only model and lacks a decoder for text generation.
Generation	BART	Success	Generated new tokens beyond the prompt, but the text was partially incoherent and repetitive.	BART has an encoder–decoder architecture, allowing text generation, but the base model is not fine-tuned for fluent open-ended generation.
Fill-Mask	BERT	Success	Predicted highly relevant masked words like create and generate with high confidence.	BERT is trained using Masked Language Modeling (MLM), making it well-suited for fill-mask tasks.
Fill-Mask	RoBERTa	Success	Accurately predicted appropriate masked words with strong confidence scores.	RoBERTa is an optimized encoder-only model with improved MLM training on larger datasets.
Fill-Mask	BART	Partial Success	Generated reasonable masked word predictions but with lower confidence than BERT and RoBERTa.	BART is primarily designed for sequence-to-sequence tasks, not masked language modeling.
QA	BERT	Partial Success	Returned only a partial answer ("and deepfakes") instead of the full list of risks.	Base BERT is not fine-tuned for question answering and struggles to extract complete spans from context.
QA	RoBERTa	Partial Success	Produced an incomplete and vague answer fragment ("and").	Although a strong encoder, RoBERTa requires QA-specific fine-tuning to accurately locate answer spans.
QA	BART	Partial Success	Returned a longer and more fluent answer but included extra context beyond the exact answer.	As an encoder–decoder model, BART tends to generate or paraphrase text rather than precisely extract answer spans when not fine-tuned for QA.

(BERT vs RoBERTa vs BART Benchmarking)

Objective

To understand how model architecture (encoder-only vs encoder–decoder) affects performance across different NLP tasks: **text generation, masked language modeling, and question answering**.

Experiment 1: Text Generation

Observation

- **BERT** generated only repetitive punctuation and failed to produce meaningful text.
- **RoBERTa** returned only the input prompt without generating new tokens.
- **BART** generated new tokens beyond the prompt, but the output was partially incoherent and repetitive.

Insight

- Encoder-only models (BERT, RoBERTa) are not designed for autoregressive text generation.
- BART's encoder–decoder architecture enables generation, but the base model lacks fine-tuning for fluent open-ended text generation.

Experiment 2: Masked Language Modeling (Fill-Mask)

Observation

- **BERT** predicted highly relevant words such as *create* and *generate* with high confidence.
- **RoBERTa** also produced accurate and confident masked word predictions.
- **BART** generated reasonable predictions but with noticeably lower confidence.

Insight

- BERT and RoBERTa perform best on fill-mask tasks because they are explicitly trained using Masked Language Modeling (MLM).
- BART is not primarily trained for MLM, as its architecture focuses on sequence-to-sequence learning.

Experiment 3: Question Answering

Observation

- **BERT** returned only a partial answer span from the context.
- **RoBERTa** produced an incomplete and vague answer fragment.
- **BART** returned a longer, more fluent answer but included extra context beyond the exact answer.

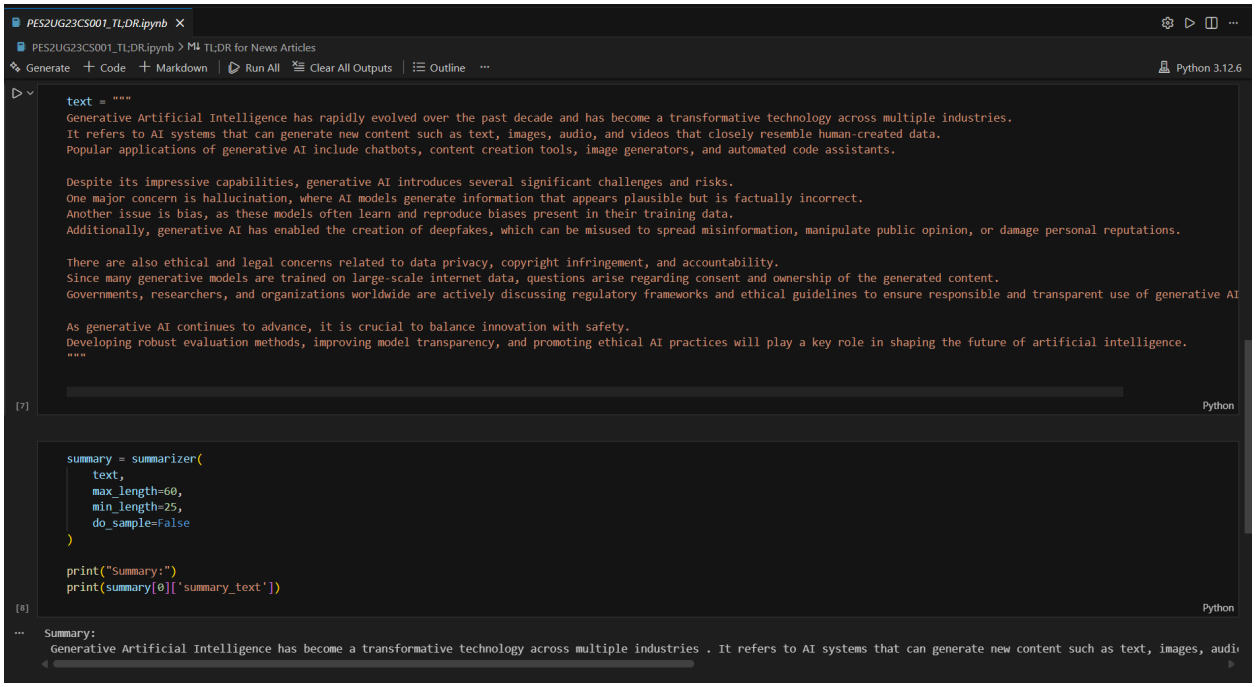
Insight

- All three base models performed poorly because they were **not fine-tuned for question answering tasks** such as SQuAD.
- Encoder–decoder models like BART tend to paraphrase or generate responses rather than precisely extract answer spans when not QA-trained.

Overall Conclusion (Model Comparison)

- Model architecture strongly influences task performance.
- Models perform well only on tasks aligned with their training objectives.
- Forcing models outside their intended design leads to degraded or misleading outputs.
- Encoder-only models excel at understanding tasks, while encoder–decoder models are better suited for generation and transformation tasks.

Observations from the PES2UG23CS001_TL;DR.ipynb



(TL;DR for News Articles using DistilBART)

Objective

To build a simple abstractive summarization system that converts long articles into concise summaries using a pre-trained transformer model.

Observation

- The summarization pipeline successfully reduced a long, multi-paragraph article into a short, coherent summary.
- The generated summary preserved the main ideas while removing redundant details.
- The output was grammatically correct and readable without manual post-processing.

Insight

- **DistilBART** is a lightweight encoder–decoder model optimized for summarization tasks.
- Encoder–decoder architectures are well-suited for abstractive summarization because they learn to rewrite and compress input sequences.

- Hugging Face pipelines allow rapid development of practical NLP applications without model training.

Key Takeaway (TL;DR Project)

- Pre-trained transformer models can be directly applied to real-world productivity tasks.
- Summarization is a natural fit for encoder–decoder architectures.
- Hugging Face pipelines significantly reduce development complexity while maintaining strong performance.

Final Learning Summary (Across Both Files)

- Understanding **model architecture** is critical when selecting a model for a given task.
- Encoder-only models specialize in comprehension-based tasks.
- Encoder–decoder models enable generation and transformation but require appropriate fine-tuning for high-quality results.
- Hands-on experimentation reveals limitations that are not obvious from theory alone.