GENAI HANDS-ON -1

Name: SPOORTHI V

SRN: PES2UG23CS594

Section: J

Email ID: spurthiv33@gmail.com

Mobile Number: 7483394221

Github Link: https://github.com/s-p-o-o-r-t-h-i-v/PES2UG23CS594_GENAI_Hands-on-1

Assignment 1: screenshots

Text Generation

Task	Model	Classification (Success/Failure)	Observation (What actually happened?)	Why did this happen? (Architectural Reason)
Generation	**BERT**	**Failure**	Generated only a long sequence of dots (.....) with no meaningful continuation.	BERT is an **encoder-only model** trained for understanding tasks (MLM), not for autoregressive next-token generation.
Generation	**RoBERTa**	**Failure**	Returned only the original prompt without generating any new text.	RoBERTa is also **encoder-only** and lacks a decoder to generate tokens sequentially.
Generation	**BART**	**Failure (Unstable Output)**	Generated long, incoherent, and repetitive text with random words and symbols.	Although BART has a decoder, bart-base is **not trained as a causal language model**; its LM head is partially uninitialized, leading to unstable generation.

Masked Language Modeling (Missing Word):

Task	Model	Classification (Success/Failure)	Observation (What actually happened?)	Why did this happen? (Architectural Reason)
Fill-Mask	**BERT**	**Success**	Predicted high-confidence words such as **"create" (0.54)**, **"generate"**, and **"produce"**, all contextually correct.	BERT is an **encoder-only model trained with Masked Language Modeling (MLM)**, making it highly effective at predicting missing tokens.
Fill-Mask	**RoBERTa**	**Success**	Accurately predicted **"generate" (0.37)** and **"create" (0.36)** with strong confidence and correct context.	RoBERTa is an optimized **MLM-based encoder**, trained on more data without NSP, improving masked token prediction.
Fill-Mask	**BART**	**Partial Success**	Predicted words like **"create"**, **"help"**, and **"provide"**, but with **lower confidence scores** and weaker semantic fit.	BART is an **encoder-decoder model trained via denoising autoencoding**, not pure MLM, so fill-mask is not its primary strength.

Question Answering:

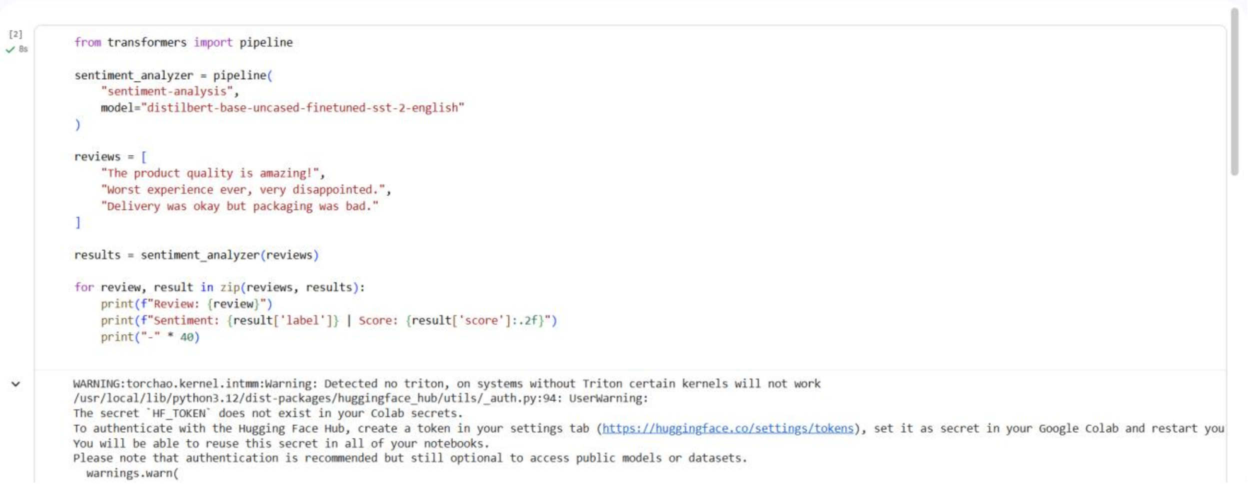
Task	Model	Classification (Success/Failure)	Observation (What actually happened?)	Why did this happen? (Architectural Reason)
QA	**BERT**	**Partial Success**	Correctly extracted **"hallucinations, bias, and deepfakes"**, but with a **very low confidence score (0.01)**.	The QA head (qa_outputs) was **randomly initialized** because the base model is **not fine-tuned for QA**, limiting confidence.
QA	**RoBERTa**	**Partial Success**	Returned a **partial span**: ", bias, and deepfakes." with **very low confidence (0.0039)**.	RoBERTa base lacks QA fine-tuning, so span selection is unstable despite strong encoding.

Task	Model	Classification (Success/Failure)	Observation (What actually happened?)	Why did this happen? (Architectural Reason)
QA	**BART**	**Failure**	Returned an **incorrect answer** (“AI poses”) unrelated to the risks.	Although BART has a decoder, it is **not trained for extractive QA**, and the QA head is randomly initialized.

Assignment 2: Screenshots

Customer Feedback Analyzer

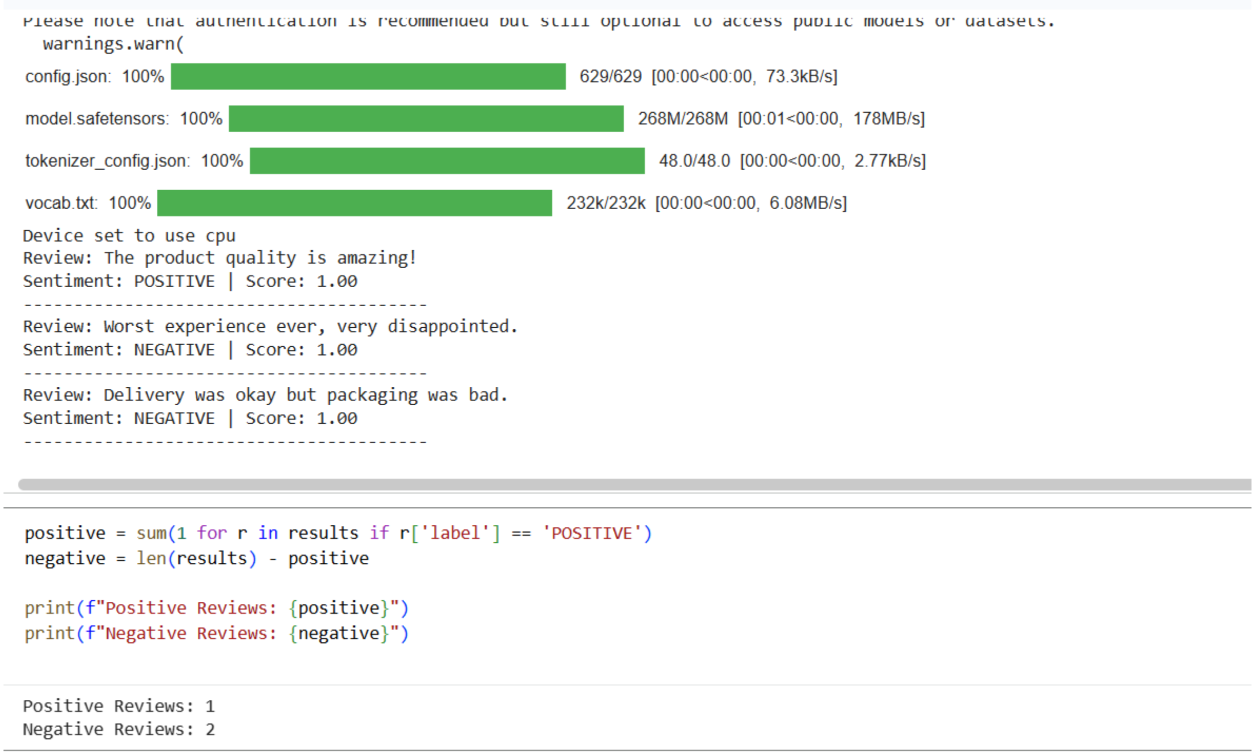
- **Goal**: Analyze 100s of product reviews to see if people are happy or angry.
- **Tech**: `pipeline('sentiment-analysis')` (Positive/Negative classification).



[2]
✓ 8s

```
from transformers import pipeline

sentiment_analyzer = pipeline(
    "sentiment-analysis",
    model="distilbert-base-uncased-finetuned-sst-2-english"
)

reviews = [
    "The product quality is amazing!",
    "Worst experience ever, very disappointed.",
    "Delivery was okay but packaging was bad."
]

results = sentiment_analyzer(reviews)

for review, result in zip(reviews, results):
    print(f"Review: {review}")
    print(f"Sentiment: {result['label']} | Score: {result['score']:.2f}")
    print("-" * 40)
```

```
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernels will not work
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning: 
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart you
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```



```
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%  629/629 [00:00<00:00, 73.3kB/s]
model.safetensors: 100%  268M/268M [00:01<00:00, 178MB/s]
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 2.77kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 6.08MB/s]
Device set to use cpu
Review: The product quality is amazing!
Sentiment: POSITIVE | Score: 1.00
----------------------------------------
Review: Worst experience ever, very disappointed.
Sentiment: NEGATIVE | Score: 1.00
----------------------------------------
Review: Delivery was okay but packaging was bad.
Sentiment: NEGATIVE | Score: 1.00
----------------------------------------
```

```python
positive = sum(1 for r in results if r['label'] == 'POSITIVE')
negative = len(results) - positive

print(f"Positive Reviews: {positive}")
print(f"Negative Reviews: {negative}")
```

```
Positive Reviews: 1
Negative Reviews: 2
```