

```
!pip install transformers torch streamlit matplotlib
```



```
Attempting uninstall: nvidia-cusolver-cu12
Found existing installation: nvidia-cusolver-cu12 11.6.3.83
Uninstalling nvidia-cusolver-cu12-11.6.3.83:
Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-12.4.127 nvidi
```

Part 1: Simple Pipeline Approach

✓ Create a summarization pipeline using DistilBART model

```
from transformers import pipeline

summarizer = pipeline("summarization", model="sshleifer/distilbart-cnn-12-6")
```

```
🔄 /usr/local/lib/python3.11/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.
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(
config.json: 100% 1.80k/1.80k [00:00<00:00, 151kB/s]
pytorch_model.bin: 100% 1.22G/1.22G [00:07<00:00, 199MB/s]
model.safetensors: 100% 1.22G/1.22G [00:14<00:00, 138MB/s]
tokenizer_config.json: 100% 26.0/26.0 [00:00<00:00, 1.72kB/s]
vocab.json: 100% 899k/899k [00:00<00:00, 5.60MB/s]
merges.txt: 100% 456k/456k [00:00<00:00, 7.02MB/s]
Device set to use cpu
```

✓ Sample incident report text

```
incident_text = """
On June 5, 2024, Unit 3 experienced a temporary loss of power due to a failed generator control module.
Operators responded immediately and shifted cooling operations to backup systems.
No radiation was released. The NRC was notified, and full power was restored within 3 hours.
Investigation revealed that the failure was due to aged circuit components.
"""
```

✓ Generate summary with constraints

```
summary = summarizer(incident_text, max_length=100, min_length=30, do_sample=False)
print(summary[0]['summary_text'])
```

⚠ Your max_length is set to 100, but your input_length is only 78. Since this is a summarization task, where outputs shorter than the input are typically want
Unit 3 experienced a temporary loss of power due to a failed generator control module . Operators responded immediately and shifted cooling operations to b

Part 2: Manual Model Loading with Attention Analysis

✓ Load tokenizer and model separately for more control

```
from transformers import BartTokenizer, BartForConditionalGeneration

tokenizer = BartTokenizer.from_pretrained("facebook/bart-large-cnn")
model = BartForConditionalGeneration.from_pretrained(
    "facebook/bart-large-cnn",
    attn_implementation="eager"
)
```

✓ Same incident text

```
incident_text = """
On June 5, 2024, Unit 3 experienced a temporary loss of power due to a failed generator control module.
Operators responded immediately and shifted cooling operations to backup systems.
No radiation was released. The NRC was notified, and full power was restored within 3 hours.
Investigation revealed that the failure was due to aged circuit components.
"""

# Tokenize input text
```


```
inputs = tokenizer(incident_text, return_tensors="pt", truncation=True, max_length=1024)

# Generate summary with attention tracking
output = model.generate(
    **inputs,
    max_length=50,
    min_length=20,
    num_beams=4,
    early_stopping=True,
    output_attentions=True,
    return_dict_in_generate=True
)
```

✓ Decode the generated summary

```
summary_text = tokenizer.decode(output['sequences'][0], skip_special_tokens=True)
print("Summary:", summary_text)
```

🔗 Summary: On June 5, 2024, Unit 3 experienced a temporary loss of power due to a failed generator control module. No radiation was released. Investigation re



✓ Part 3: **Attention Visualization**

Extract cross-attention from last decoder layer

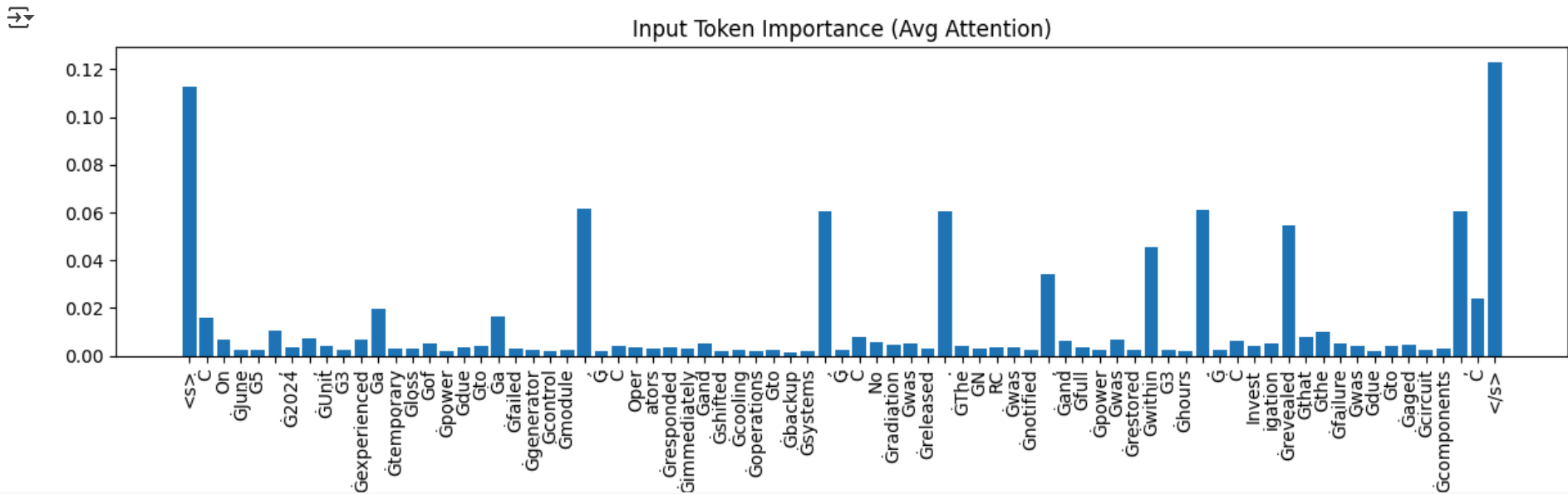
```
# Cross-attention from last decoder layer (list: layers → batch → heads → target → source)
cross_attn = output.cross_attentions[-1][0] # shape: [num_heads, target_len, source_len]
```

```
import numpy as np
import matplotlib.pyplot as plt

# Average across heads and target tokens
avg_attn = cross_attn.mean(dim=0).mean(dim=0).detach().numpy().flatten() # shape: [source_len]
# Get the original tokens for labeling
tokens = tokenizer.convert_ids_to_tokens(inputs['input_ids'][0])
```

✓ Create visualization

```
plt.figure(figsize=(12, 4))
plt.bar(range(len(tokens)), avg_attn)
plt.xticks(range(len(tokens)), tokens, rotation=90)
plt.title("Input Token Importance (Avg Attention)")
plt.tight_layout()
plt.show()
```



Start coding or [generate](#) with AI.