Session 8 — Chain Two Models

Space Builder: Image Story Pipeline

What You'll Build

A two-model pipeline: upload an image, Model 1 (BLIP) writes a caption for it, then Model 2 reads that caption and judges whether it's positive or negative. Two AI models, connected in a chain.

Step 1 — Create a New Space

1. Go to huggingface.co/new-space (log in if needed)
2. In the "Space name" field, type: image-story-pipeline
3. Under "Select the Space SDK", choose Gradio
4. Under "Select the Space hardware", choose Free — CPU basic
5. Leave everything else as default
6. Click Create Space

Step 2 — Create the requirements.txt File

1. On your Space page, click the Files tab (near the top)
2. Click Add file → Create a new file
3. In the filename field at the top, type: requirements.txt
4. In the big text area below, copy and paste this exactly:

```
This Space needs Pillow in requirements.txt (for image processing). The requirements are slight
  ly different from previous sessions — make sure you copy them exactly.
```

5. Click Commit new file to main (the blue button at the bottom)

Step 3 — Create the app.py File

1. Click Add file → Create a new file (again)
2. In the filename field, type: app.py
3. In the big text area, copy and paste ALL of the code below:

> *Your Space page will open. It's empty right now — we're about to add the code.*

4. Click Commit new file to main

Step 4 — Wait for It to Build

1. Click the App tab (at the top of your Space page)
2. You'll see "Building" — this takes 2–5 minutes
3. When it's done, your Space will appear!

Step 5 — Try It Out!

1. Upload a happy photo (pets, sunshine) — does the pipeline get it right?
2. Upload something abstract or confusing — does Model 1 write a bad caption?
3. If the caption is wrong, the sentiment is meaningless — that's an error cascade!

Troubleshooting

```
transformers
torch
gradio
Pillow
```

```
import gradio as gr
from transformers import pipeline, BlipProcessor, BlipForConditionalGeneration
from PIL import Image

# Load BLIP captioning model (~1GB)
print("Loading BLIP captioning model (this takes a moment)...")
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
caption_model = BlipForConditionalGeneration.from_pretrained(
    "Salesforce/blip-image-captioning-base"
)
print("BLIP loaded!")

# Load sentiment model (~250MB)
print("Loading sentiment model...")
sentiment = pipeline(
    "sentiment-analysis",
    model="distilbert-base-uncased-finetuned-sst-2-english",
)
print("All models loaded!")

def analyze_image(image):
    if image is None:
        return "Upload an image first!", "", ""

    # Step 1: Generate caption
    inputs = processor(image, return_tensors="pt")
    out = caption_model.generate(**inputs, max_length=50)
    caption = processor.decode(out[0], skip_special_tokens=True)

    # Step 2: Analyze caption sentiment
    result = sentiment(caption)[0]
    sentiment_text = f"{result['label']} ({result['score']:.1%} confidence)"

    # Step 3: Show the full pipeline
    pipeline_view = (
        f"IMAGE\n"
        f"  -> Model 1 (BLIP captioner): \"{caption}\"\n"
        f"  -> Model 2 (sentiment): {result['label']} ({result['score']:.1%})"
    )

    return caption, sentiment_text, pipeline_view

with gr.Blocks(title="Image Story Pipeline") as demo:
    gr.Markdown(
        "# Image Story Pipeline\n"
        "Upload an image. Model 1 describes it in words. Model 2 reads those "
        "words and judges the tone. Two models, connected — the output of the "
        "first becomes the input of the second."
    )
```

```
    with gr.Row():
        with gr.Column():
            image_input = gr.Image(type="pil", label="Upload an Image")
            btn = gr.Button("Analyze", variant="primary")
        with gr.Column():
            caption_output = gr.Textbox(
                label="Step 1: Caption (BLIP)", interactive=False
            )
            sentiment_output = gr.Textbox(
                label="Step 2: Sentiment of Caption", interactive=False
            )
            pipeline_output = gr.Textbox(
                label="Full Pipeline View", lines=4, interactive=False
            )

    btn.click(
        fn=analyze_image,
        inputs=image_input,
        outputs=[caption_output, sentiment_output, pipeline_output],
    )

    gr.Markdown(
        "### How to break it\n"
        "Try uploading images that are hard to describe — abstract art, memes, "
        "dark photos. If Model 1 gets the caption wrong, Model 2's analysis is "
        "meaningless. That's an **error cascade**."
    )

demo.launch()
```

```
Make sure you copy the ENTIRE code block — from the very first line to the very last. Missing e
  ven one line can cause errors.
```

```
If you see a red error: Click the Logs tab to read the error message. The most common fix is to
   double-check that requirements.txt has the right contents and that app.py was copied complet
  ely.
```

Problem	Fix
"Runtime error"	Check the Logs tab. Usually means a typo in app.py. Re-copy the code carefully.
Space stuck on "Building"	Wait up to 5 minutes. Free CPU Spaces can be slow. If it’s been more than 10 minutes, try deleting the Space and starting over.
"ModuleNotFoundError"	Your requirements.txt is missing a library. Make sure it matches exactly what’s shown above.
Space loads but nothing happens	Make sure the last line of app.py is demo.launch() with no extra spaces before it.