Week5-2: RAG — Multimodal (Text + Images/Charts)

0) Setup & Demo Images

```
#@title 0) Setup & Demo Images
import os, json, numpy as np, pandas as pd
from PIL import Image

# Point to your folder that already has images
IMG_DIR = '/content/project_files'   # <-- change to your actual folder with images
os.makedirs(IMG_DIR, exist_ok=True)

meta = []
# Loop through all PNG/JPG images in the folder
for i, fname in enumerate(sorted(os.listdir(IMG_DIR)), start=1):
    if fname.lower().endswith((".png", ".jpg", ".jpeg")):
        path = os.path.join(IMG_DIR, fname)
        meta.append({
            "image_id": f"img{i}",
            "path": path,
            "caption": f"Project image {i}: {fname}"
        })

df_imgs = pd.DataFrame(meta)
print('✅ Images:', len(df_imgs))
print(df_imgs.head(2))
```

```
✅ Images: 3
  image_id                                              path  \
0     img1       /content/project_files/context_length[1].png
1     img2  /content/project_files/latency_vs_recall_trade...

                                          caption
0             Project image 1: context_length[1].png
1  Project image 2: latency_vs_recall_tradeoff[1]...
```

```
# 1) Embeddings (placeholders) + joint index
import numpy as np, pandas as pd
np.random.seed(0); emb_dim = 128

# Build text corpus aligned with your images
text_corpus = pd.DataFrame({
    'doc_id': [f'doc{i+1}' for i in range(len(df_imgs))],
    'text': [f'This document references {row.caption}.' for _, row in df_imgs.iterrows()]
})

# Image embeddings (normalized random vectors as placeholders)
img_emb = {row.image_id: np.random.randn(emb_dim).astype('float32') for _, row in df_imgs.iterrows()}
for k in img_emb:
    img_emb[k] /= (np.linalg.norm(img_emb[k])+1e-9)

# Text embeddings (normalized random vectors as placeholders)
text_emb = {row.doc_id: np.random.randn(emb_dim).astype('float32') for _, row in text_corpus.iterrows()}
for k in text_emb:
    text_emb[k] /= (np.linalg.norm(text_emb[k])+1e-9)

# Cosine similarity
def cosine(a,b):
    return float(a @ b / (np.linalg.norm(a)+1e-9) / (np.linalg.norm(b)+1e-9))

# Encode query text (placeholder)
def encode_text(q):
    v = np.random.randn(emb_dim).astype('float32')
    return v/(np.linalg.norm(v)+1e-9)

print('✅ Text docs:', len(text_corpus))
print(text_corpus.head(2))
```

```
✅ Text docs: 3
  doc_id                                               text
0   doc1  This document references Project image 1: con...
1   doc2  This document references Project image 2: late...
```

```
# 2) Retrieval modes (unchanged)
def retrieve_text(query, k=3):
    q = encode_text(query); scores = [(doc, cosine(q, text_emb[doc])) for doc in text_emb]
    return sorted(scores, key=lambda x: -x[1])[:k]

def retrieve_image_by_text(query, k=3):
    q = encode_text(query); scores = [(img, cosine(q, img_emb[img])) for img in img_emb]
    return sorted(scores, key=lambda x: -x[1])[:k]

def retrieve_by_image(image_id, k=3):
    q = img_emb[image_id]
    t = sorted([(doc, cosine(q, text_emb[doc])) for doc in text_emb], key=lambda x: -x[1])[:k]
    i = sorted([(img, cosine(q, img_emb[img])) for img in img_emb if img != image_id], key=lambda x: -x[1])[:k]
    return t, i

# --------------------------
# Project-related display
# --------------------------
print("\n🔎 Text Query → Retrieved Docs")
for doc, score in retrieve_text("trend in recall vs latency", 3):
    print(f" Doc {doc}: {text_corpus.loc[text_corpus.doc_id==doc,'text'].values[0]} | score={score:.3f}")

print("\n🖼️ Text Query → Retrieved Images")
for img, score in retrieve_image_by_text("recall performance chart", 3):
    cap = df_imgs.loc[df_imgs.image_id==img,'caption'].values[0]
    print(f" Image {img}: {cap} | score={score:.3f}")

print("\n📊 Image Query (img2) → Retrieved Docs + Images")
docs, imgs = retrieve_by_image("img2", 3)

print(" Related Docs:")
for doc, score in docs:
    print(f"  Doc {doc}: {text_corpus.loc[text_corpus.doc_id==doc,'text'].values[0]} | score={score:.3f}")

print(" Related Images:")
for img, score in imgs:
    cap = df_imgs.loc[df_imgs.image_id==img,'caption'].values[0]
    print(f"  Image {img}: {cap} | score={score:.3f}")
```

```
🔎 Text Query → Retrieved Docs
 Doc doc1: This document references Project image 1: context_length[1].png. | score=0.117
 Doc doc2: This document references Project image 2: latency_vs_recall_tradeoff[1].png. | score=-0.089
 Doc doc3: This document references Project image 3: recall_scores[1].png. | score=-0.109

🖼️ Text Query → Retrieved Images
 Image img1: Project image 1: context_length[1].png | score=0.257
 Image img2: Project image 2: latency_vs_recall_tradeoff[1].png | score=0.061
 Image img3: Project image 3: recall_scores[1].png | score=-0.098

📊 Image Query (img2) → Retrieved Docs + Images
 Related Docs:
  Doc doc1: This document references Project image 1: context_length[1].png. | score=0.034
  Doc doc3: This document references Project image 3: recall_scores[1].png. | score=-0.081
  Doc doc2: This document references Project image 2: latency_vs_recall_tradeoff[1].png. | score=-0.085
 Related Images:
  Image img1: Project image 1: context_length[1].png | score=0.038
  Image img3: Project image 3: recall_scores[1].png | score=0.005
```

```
# 3) Prompt assembly
def assemble_prompt(query, text_hits, image_hits):
    tbits = [f'[{doc}] '+text_corpus[text_corpus.doc_id==doc].iloc[0].text for doc,_ in text_hits]
    ibits = [f'[{img}] '+df_imgs[df_imgs.image_id==img].iloc[0].caption for img,_ in image_hits]
    return f"""System: Answer using ONLY the evidence below. Cite [doc_id] or [image_id].
Query: {query}
Evidence:
- Text: {' | '.join(tbits)}
- Images: {' | '.join(ibits)}
Answer:
"""
q='What recall_scores[1]  indicate?'; print(assemble_prompt(q, retrieve_text(q,2), retrieve_image_by_text(q,2)))
```

```
dence below. Cite [doc_id] or [image_id].
icate?

ences Project image 3: recall_scores[1].png. | [doc1] This document references Project image 1: context_length[1].png.
```

```
latency_vs_recall_tradeoff[1].png | [img1] Project image 1: context_length[1].png
```

```
print('A) Text-only -> retrieve text + images-by-text')
q1 = 'Summarize trend in average context length for pipeline variants'
print('Text:', retrieve_text(q1,3))
print('Images-from-text:', retrieve_image_by_text(q1,3))

print('\nB) Image-only -> retrieve related docs & similar images')
t_hits, i_hits = retrieve_by_image('img3', 3)  # img3 = recall_scores.png in your charts
print('Docs:', t_hits)
print('Images:', i_hits)
```

```
A) Text-only -> retrieve text + images-by-text
Text: [('doc2', 0.04642634838819504), ('doc3', -0.04017006978392601), ('doc1', -0.12806814908981323)]
Images-from-text: [('img2', 0.028183303773403168), ('img3', 0.024671155959367752), ('img1', 0.014205452986061573)]

B) Image-only -> retrieve related docs & similar images
Docs: [('doc2', 0.1648528128862381), ('doc1', 0.009096058085560799), ('doc3', -0.1570434719324112)]
Images: [('img2', 0.004841104615479708), ('img1', -0.08821874856948853)]
```

Double-click (or enter) to edit