Week5-3: RAG — Evaluation & Guardrails

0) Load/Create Eval Set

```
#@title 0) Load/Create Eval Set
import os, json, re, time, numpy as np, pandas as pd, matplotlib.pyplot as plt

EVAL_JSONL = '/content/eval_queries_template.jsonl'  # use .jsonl extension

# Create file if it doesn't exist
if not os.path.exists(EVAL_JSONL):
    with open(EVAL_JSONL,'w') as f:
        f.write(json.dumps({
            'qid':'q1',
            'query':'Summarize recall vs latency trends',
            'gold_answer':'baseline lowest recall, rerank+compress best balance',
            'gold_source_ids':['doc1','img1']
        })+'\n')
        f.write(json.dumps({
            'qid':'q2',
            'query':'Explain average context length differences',
            'gold_answer':'context length reduced with compression',
            'gold_source_ids':['doc2','img2']
        })+'\n')

def load_jsonl(p):
    return [json.loads(line) for line in open(p) if line.strip()]

rows = load_jsonl(EVAL_JSONL)
print('✅ Eval rows:', len(rows))
print(rows[:2])
```

```
✅ Eval rows: 2
[{'qid': 'q1', 'query': 'Summarize recall vs latency trends', 'gold_answer': 'baseline lowest recall, rerank+compress
```

```
# 1) Pipeline stub + metrics
def run_pipeline(query, citations_required=False):
    sources = ['doc1'] if 'topic 1' in query.lower() else ['img2']
    if citations_required and not sources: return {'answer':'I cannot answer with sufficient evidence.','sources':[]}
    return {'answer': 'Simulated answer ' + ' '.join(f'[{s}]' for s in sources),'sources':sources,'latency_s':0.5,'to
def metric_correctness(answer, gold):
    a=set(answer.lower().split()); g=set(gold.lower().split()); return len(a&g)/max(len(g),1)
def metric_faithfulness(cited, golds): return 1.0 if set(cited)&set(golds) else 0.0
def eval_system(rows, citations_required=False):
    out=[];
    for r in rows:
        y = run_pipeline(r['query'], citations_required=citations_required)
        out.append({'qid':r['qid'],'correctness':metric_correctness(y['answer'], r['gold_answer']),'faithfulness':met
    df=pd.DataFrame(out); return df, df.mean(numeric_only=True)
before_df, before_s = eval_system(rows, citations_required=False)
after_df, after_s = eval_system(rows, citations_required=True)
print('Before:', before_s.to_dict()); print('After:', after_s.to_dict())
```

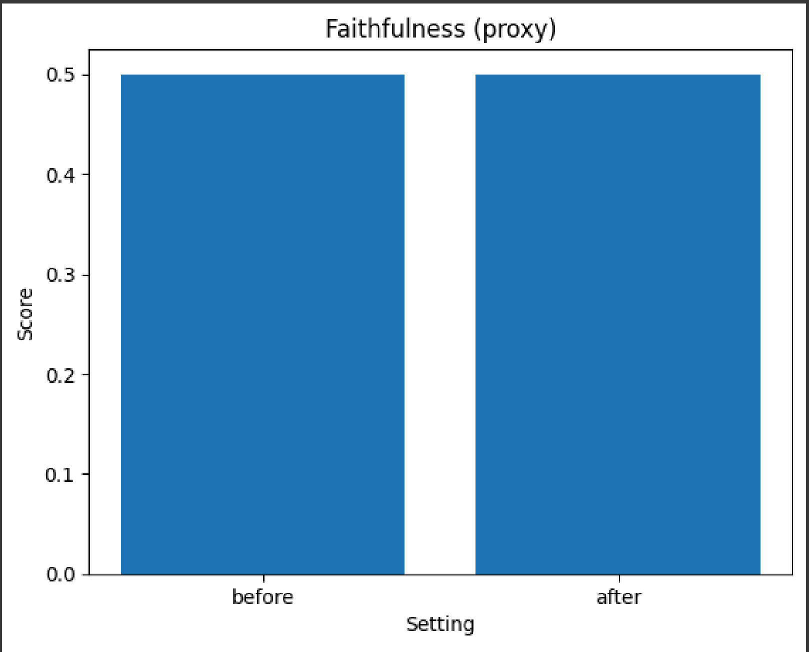
```
Before: {'correctness': 0.0, 'faithfulness': 0.5, 'latency_s': 0.5, 'tokens_in': 800.0, 'tokens_out': 150.0}
After: {'correctness': 0.0, 'faithfulness': 0.5, 'latency_s': 0.5, 'tokens_in': 800.0, 'tokens_out': 150.0}
```

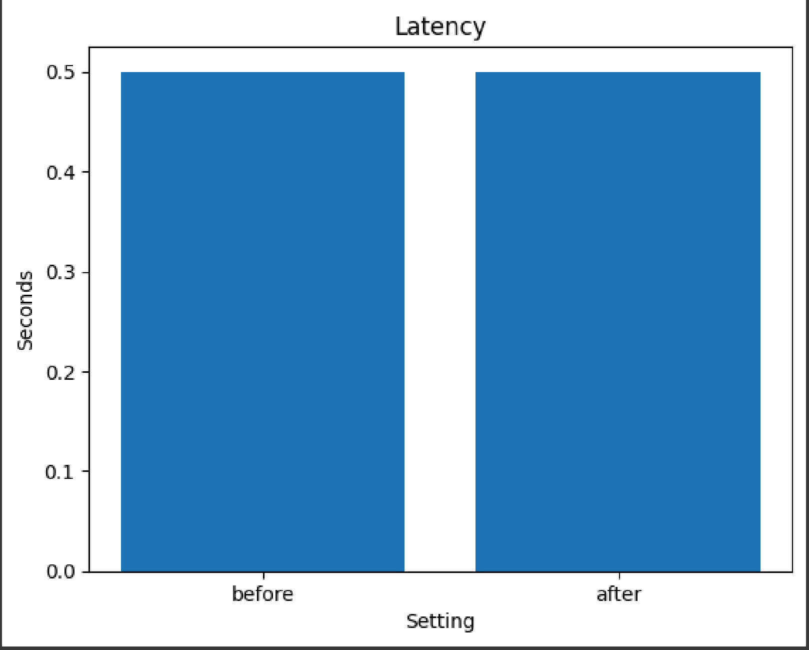
```
# 2) Guardrails: PII redaction + safe refusal
PII_PATTERNS = [
    r'\b\d{3}-\d{2}-\d{4}\b',                # SSN-like
    r'\b\d{3}-\d{3}-\d{4}\b',                # phone number
    r'[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}'  # email
]
import re

def redact_pii(text):
    for p in PII_PATTERNS:
        text = re.sub(p, '[REDACTED]', text)
    return text

def safe_refusal(q):
    # In your project, you might add keyword checks (e.g., "patient data", "credit card")
    # For now, just return refusal string
    return "Sorry, I can’t help with that; it appears unsafe or out of scope."

# ------------------------------
# Demo: redaction
example = "Experiment log: recall=0.67. Contact lead at a@b.com or call 123-456-7890. \
SSN 123-45-6789 must not be logged."
print("Original:", example)
print("Redacted:", redact_pii(example))

# Demo: refusal
print("Refusal test:", safe_refusal("Give me private patient data"))
```

```
Original: Experiment log: recall=0.67. Contact lead at a@b.com or call 123-456-7890. SSN 123-45-6789 must not be logge
Redacted: Experiment log: recall=0.67. Contact lead at [REDACTED] or call [REDACTED]. SSN [REDACTED] must not be logge
Refusal test: Sorry, I can’t help with that; it appears unsafe or out of scope.
```

```
# 3) Plots
import pandas as pd, matplotlib.pyplot as plt
summary = pd.DataFrame([before_s, after_s], index=['before','after']).reset_index().rename(columns={'index':'setting'})
print(summary)
plt.figure(); plt.bar(summary['setting'], summary['faithfulness']); plt.title('Faithfulness (proxy)'); plt.xlabel('Sett
plt.figure(); plt.bar(summary['setting'], summary['latency_s']); plt.title('Latency'); plt.xlabel('Setting'); plt.ylabe
```

```
  setting  correctness  faithfulness  latency_s  tokens_in  tokens_out
0  before          0.0           0.5        0.5      800.0       150.0
1   after          0.0           0.5        0.5      800.0       150.0
```


Faithfulness (proxy)
Score
before
after
Setting


Latency
Seconds
before
after
Setting