Assignment 14 — Solutions

Evaluation of LLMs & RAG

Part C — Building GenAI Applications · 10 questions with answers

Q1. How do you evaluate an LLM application? (Offline vs online, reference-based vs reference-free.) ★ · Asked: Infosys, Google

Answer

Offline evaluation (before release):

- Build a **test set** of realistic inputs with reference answers or grading criteria.
- **Reference-based** metrics compare output with a known correct answer: exact match, F1, accuracy, BLEU/ROUGE, semantic similarity, LLM-judge "is it equivalent to the reference?".
- **Reference-free** metrics judge the output on its own: faithfulness to retrieved context, relevance, toxicity, format validity, LLM-as-judge rubric scores (helpfulness, correctness, tone).
- Component-level evaluation: retrieval (recall@k, MRR, context precision/recall) separately from generation.
- Non-functional: latency, cost per request, robustness, safety/red-team tests.

Online evaluation (in production):

- User feedback (thumbs up/down, ratings), task completion, escalation rate to humans, conversation length.
- A/B tests between prompts/models.
- Sampled LLM-judge scoring of live traffic, monitoring of drift and failures.

Tools: RAGAS, DeepEval, LangSmith, Langfuse, TruLens, promptfoo, OpenAI Evals.

Q2. What is RAGAS? What metrics evaluate the retriever and the generator? ★ · Asked: Google

Answer

RAGAS (Retrieval-Augmented Generation Assessment) is an open-source framework for evaluating RAG pipelines, largely **reference-free**, using LLMs (and embeddings) as judges.

Retriever metrics:

- **Context precision** – are the relevant chunks ranked at the top?
- **Context recall** – did retrieval find all the information needed for the ground-truth answer?
- (Context entities recall, noise sensitivity.)

Generator metrics:

- **Faithfulness** – is the answer supported by the retrieved context (no hallucination)?
- **Answer relevancy** – does the answer address the question?
- (Answer correctness / factual correctness and answer similarity vs a reference answer.)

```
from ragas import evaluate
from ragas.metrics import faithfulness, answer_relevancy, context_precision, context_recall
from datasets import Dataset
data = Dataset.from_dict({"question": qs, "answer": answers,
                          "contexts": contexts, "ground_truth": refs})
print(evaluate(data, metrics=[faithfulness, answer_relevancy, context_precision, context_recall]))
```

(Newer RAGAS versions rename fields to user_input / response / retrieved_contexts / reference.)

Q3. Explain faithfulness, answer relevancy, context precision and context recall. ★

Answer

- **Faithfulness** – measures whether every claim in the answer can be **inferred from the retrieved context**. An LLM splits the answer into individual statements and checks each against the context. Score = supported claims / total claims (0–1). Low faithfulness = hallucination. Example: context says "refund within 7 days", answer says "refund within 30 days" → unfaithful.
- **Answer relevancy** – measures how well the answer **addresses the question** (not whether it is true). RAGAS asks an LLM to generate possible questions from the answer and computes the average cosine similarity between them and the original question. Incomplete or off-topic/padded answers score low.
- **Context precision** – measures whether the **relevant retrieved chunks are ranked higher** than irrelevant ones (a precision@k averaged over ranks). High = little noise at the top of the results.
- **Context recall** – measures whether the retrieved context contains **all the information needed** to produce the ground-truth answer: each sentence of the reference answer is checked for whether it can be attributed to the context. Needs a reference answer. Low recall = retriever missed information.

Q4. What is LLM-as-a-judge? What are its biases and how do you reduce them? ·

Asked: Google

Answer

LLM-as-a-judge – using a strong LLM to **grade outputs** of another model/system, with a rubric: score 1–5 for correctness/helpfulness, pass/fail against criteria, or **pairwise comparison** ("Which answer is better, A or B?"). Scales human-like evaluation cheaply; agrees reasonably well with human raters when well designed.

Biases:

- **Position bias** – prefers the first (or second) answer.
- **Verbosity bias** – prefers longer answers.
- **Self-preference bias** – prefers outputs from its own model family.
- Inconsistency between runs; weak at checking facts/maths it doesn't know.

Mitigations:

- **Swap A/B order** and average (or count only consistent wins).
- Clear, specific **rubrics** with examples; ask for reasoning before the score; use binary/low-granularity scales.
- Use a judge from a **different model family**, temperature 0.
- Provide reference answers/context when available.
- **Calibrate against human labels** on a sample; use multiple judges.

Q5. Explain BLEU, ROUGE, METEOR and perplexity. Where is each used?

Answer

- **BLEU** – measures **n-gram precision** (1–4 grams) of the generated text against reference(s), with a brevity penalty for too-short outputs. 0–1 (or 0–100). Mainly for **machine translation**.
- **ROUGE** – measures **n-gram recall/overlap** against references: ROUGE-1 (unigrams), ROUGE-2 (bigrams), ROUGE-L (longest common subsequence). Mainly for **summarization**.

- **METEOR** – combines precision and recall (recall weighted higher) and matches **stems and synonyms** (WordNet), with a penalty for word-order fragmentation. Correlates better with human judgement than BLEU; used for **translation/captioning**.
- **Perplexity** – exp(average negative log-likelihood) of a text under a language model; how "surprised" the model is. **Lower = better** language modelling/fluency. Used to evaluate **LM pre-training** and compare models on the same tokenizer/dataset.

Limitation of n-gram metrics: they reward word overlap, not meaning – a correct paraphrase can score low; so for LLM outputs, semantic similarity (BERTScore) and LLM-as-judge are often preferred.

Q6. What metric is used to evaluate speech-to-text (WER, CER)?

Answer

- **WER (Word Error Rate)** – the standard metric: WER = (S + D + I) / N, where S = substitutions, D = deletions, I = insertions, N = number of words in the reference transcript. Lower is better (0 = perfect; can exceed 1).
- **CER (Character Error Rate)** – same formula at character level; better for languages without clear word boundaries (Chinese, Japanese) or for spelling-level accuracy.

```
import evaluate
wer = evaluate.load("wer")
print(wer.compute(predictions=["i love deep learning"], references=["i love deep learning
    models"]))  # 0.2
```

Normalize text (case, punctuation) before scoring. Also track latency (real-time factor).

Q7. How do you build a golden evaluation dataset? · Asked: Google

Answer

1. **Collect real inputs:** sample actual user queries/logs (anonymized), plus questions from domain experts covering all important topics.
2. **Cover the distribution:** easy, hard, multi-hop, ambiguous, out-of-scope ("should say I don't know"), adversarial/prompt-injection, different languages/phrasings.
3. **Create reference answers:** have domain experts write or verify the ideal answers and the source documents/chunks that contain them. LLMs can **draft synthetic Q&A pairs** from documents (e.g. RAGAS test-set generator), but humans must review them.
4. **Add metadata:** category, difficulty, expected source.
5. **Size:** start with ~50–200 high-quality examples; grow over time.
6. **Version it** and keep it separate from training data; **add every production failure** as a new test case.

Q8. How do you measure hallucination rate in production? · Asked: TCS, Google

Answer

- **Automated faithfulness checks on sampled traffic:** run an LLM-judge / RAGAS faithfulness or an NLI model to check whether each answer's claims are supported by the retrieved context; hallucination rate = % of answers with unsupported claims.
- **Citation verification:** check that quoted/cited sources exist and actually contain the cited text.
- **Detectors:** e.g. self-consistency checks (sample several answers, flag disagreement), dedicated hallucination-detection models (e.g. Vectara HHEM, Lynx).
- **User signals:** thumbs down, "that's wrong" corrections, escalations to human agents.
- **Human audit:** experts review a random sample weekly to calibrate the automated metrics.

- Track the rate on a **dashboard** over time, by topic and by model/prompt version, with alerts when it rises.

Q9. How do you evaluate retrieval quality? Explain Precision@K, Recall@K, MRR (and nDCG). ★ · Asked: TCS

Answer

Need a test set of queries with the known relevant chunks/documents.

- **Precision@K** = (relevant chunks in top K) / K – how much of what we retrieved is useful (noise).
- **Recall@K** = (relevant chunks in top K) / (all relevant chunks for that query) – did we find everything needed. Most important for RAG (missing context = wrong answer).
- **Hit rate@K** – % of queries where at least one relevant chunk is in the top K.
- **MRR (Mean Reciprocal Rank)** = average over queries of 1 / (rank of the first relevant result). Rank 1 → 1.0, rank 2 → 0.5, rank 4 → 0.25.
- **nDCG@K** – rewards relevant results at higher ranks, supports graded relevance (highly vs partly relevant).

Example: top-5 results, relevant at positions 2 and 4, total relevant = 3 → Precision@5 = 2/5 = 0.4, Recall@5 = 2/3 = 0.67, reciprocal rank = 1/2 = 0.5.

For LLM-based RAG evaluation without labels, use RAGAS context precision/recall.

Q10. How do you compare two prompt versions, and what are regression testing and shadow testing for LLM apps? ★ · Asked: Google

Answer

- **Comparing prompt versions:** run both on the same **golden dataset**; score with task metrics, format validity, LLM-as-judge (pairwise, order-swapped) and a human sample; compare latency and tokens/cost; inspect disagreements. Then A/B test in production.
- **Regression testing:** every change (prompt, model version, retriever, chunking) is automatically re-run against the golden set in **CI/CD**; the build fails if quality metrics drop below thresholds or previously fixed cases break.
- **Shadow testing:** the new version runs **in parallel on real production traffic without its output being shown to users**; responses are logged and compared with the current version (LLM-judge, metrics) – safe way to validate before rollout.
- **Canary release** follows: expose the new version to a small % of users, watch metrics, then expand or roll back.