LLM-Assisted Data Extraction (Human-in-the-Loop)

Note on data. This problem set uses **synthetic** (simulated) text snippets provided in the code tutorial. The goal is to practice structured extraction with LLMs, auditing, and evaluation—not to make substantive claims about real events.

LLM requirement. Use an LLM that is **free to use**. Recommended: run a local open-source model with `Ollama` (e.g., `llama3.1:8b` or `mistral:7b-instruct`). Do *not* use a paid API.

Conceptual Questions

Please write three to ten sentence explanations for each of the following questions. **You are only required to answer ONE of the two questions below.**

1. Explain why **schema-constrained** extraction (structured JSON fields with explicit missingness rules) can reduce hallucination relative to free-form summarization. Then explain one limitation: a way the model can still produce systematically wrong extractions even when it outputs "valid" JSON.
2. Human-in-the-loop extraction requires evaluation. Explain why **spot-audits** and **precision/recall**-style evaluation are both needed. In your answer, define one failure mode that would be missed by evaluating only on a small gold set, and one failure mode that would be missed by auditing only a small random sample.

Applied Exercises

Use the code in the week's code tutorial and the lecture slides to answer the following questions.

3. **Replace the API call with a free LLM and generate structured extractions.** Start from the provided script (`llm_human.py`) and modify it so it uses a **free** LLM.
   - Keep the `EventExtraction` schema and require the model to output a single JSON object that matches the schema.
   - Run extraction for all documents and save the output table to `outputs/extractions_raw.csv`.
   - Report (briefly) which model you used (e.g., `llama3.1:8b`) and the exact prompt you used.

   Implementation hint (Ollama). If using Ollama, you may call the local server from Python via HTTP. For example:

```
# pip install requests
import requests, json

payload = {
```

```
  "model": "llama3.1:8b",
  "prompt": "Return ONLY valid JSON matching this schema: ...",
  "stream": False
}
resp = requests.post("http://localhost:11434/api/generate", json=payload).json()
text = resp["response"]
record = json.loads(text)  # then validate with Pydantic
```

4. **Uncertainty flags + audit sheet (human-in-the-loop).** Using your extracted dataset:
   - Create at least **four** mechanical review flags (examples: low confidence, missing date, missing country, `geo_precision` is `unknown`, or empty actors list).
   - Create a **single** audit sheet CSV (similar to the tutorial) that includes:
     (a) the raw text,
     (b) the extracted fields,
     (c) the evidence quotes, and
     (d) blank columns for human corrections and failure-mode tags.
   - Fill out the audit sheet for at least **five** documents and tag a failure mode for any incorrect extraction (e.g., `date_missing`, `location_vague`, `event_type_wrong`, `actor_hallucination`).
   - Report two audit statistics:
     (a) the share of audited rows marked correct, and
     (b) the most common failure mode you observed (a small frequency table is fine).
5. **Evaluation + prompt iteration.** Use the small gold set in the tutorial to evaluate event-type classification:
   - Compute and report a classification report (precision/recall/F1) for `event_type`.
   - Create **two** prompt variants (e.g., different missingness instructions; stronger evidence requirements; a shorter event-type taxonomy explanation).
   - Re-run extraction and evaluation for both prompts, and present a **small table** comparing at least:
     (a) macro-F1,
     (b) accuracy, and
     (c) number of items flagged for human review.
   - In 6–10 sentences, defend which prompt you would use in a larger project and why. Your answer must reference both (i) quantitative evaluation and (ii) auditing considerations.
6. **Challenge Question (Optional — if you finish early):** Make the pipeline more robust to long documents or ambiguous text. Choose **ONE** option:
   (a) **Chunking.** Split each document into 2–3 chunks, run extraction per chunk, and then write a short rule-based aggregation step that outputs one final record (e.g., choose the highest-confidence chunk; union actors; keep the most specific location). Discuss 2–4 sentences on why chunking changes failure modes.
   (b) **Abstention policy.** Add an explicit "abstain" rule: if the model is unsure, it must set `event_type = other` and add an uncertainty flag. Compare (i) event-type macro-F1 and (ii) the review queue size before vs after abstention. Interpret the trade-off (5–8 sentences).