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. You must use a **free local LLM**. You may use either:
- **Ollama** (e.g., `llama3.1:8b`, `mistral:7b-instruct`), or
- A **Hugging Face Transformers** model run locally (e.g., `google/flan-t5-small`).

Do *not* use a paid API.

Start Off: Verifying Your Environment

1. **Environment check (required).**
   Submit proof that you successfully ran the full tutorial code on your machine. You may submit *one* of the following:
   - A screenshot or text file showing console output that includes:
     - the first few rows of the extracted dataset,
     - review flag counts, and
     - the classification report.
   - A screenshot of your `outputs/` directory showing generated CSV files (e.g., `extractions_raw.csv`, `extractions_with_flags.csv`, `human_audit_sheet.csv`).
   - A Git commit (hash or screenshot) that includes at least one generated output file.
   - A short log file (e.g., `run_log.txt`) containing printed diagnostics and evaluation metrics.

   Your submission must clearly demonstrate that the full pipeline ran successfully.

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.**

2. 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.
3. 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 tutorial and lecture slides to answer the following questions. **You are only required to answer TWO of the three questions below.**

4. **Run structured extraction using a free local LLM.**
   Starting from the provided script (`llm_human.py`):
   - Use a **free local LLM** (Ollama or Hugging Face).
   - Keep the `EventExtraction` schema.
   - Require the model to output a single JSON object that matches the schema.
   - Run extraction for all documents.
   - Save the output table to `outputs/extractions_raw.csv`.

   Because local models are not perfectly reliable at producing valid JSON, you must:
   - Log the raw model output,
   - Report the number (or share) of parse failures (if any), and
   - Explain briefly how you handled invalid JSON outputs.

   In your submission, report:
   - Which model you used (e.g., `llama3.1:8b`, `flan-t5-small`),
   - The exact prompt you used.

5. **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 = unknown`, empty actors list, parse failure).
   - Create a **single audit sheet CSV** 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.
   - For any incorrect extraction, tag a failure mode (e.g., `date_missing`, `location_vague`, `event_type_wrong`, `actor_hallucination`, `parse_failure`).

   Report two audit statistics:
   (a) the share of audited rows marked correct, and
   (b) the most common failure mode (a small frequency table is sufficient).

6. **Evaluation + prompt iteration.**
   Using the small gold set in the tutorial:
   - Compute and report a classification report (precision/recall/F1) for `event_type`.
   - Create **two** prompt variants (e.g., different missingness instructions, stronger evidence requirements, shorter taxonomy explanation).
   - Re-run extraction and evaluation for both prompts.
   - 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. Your answer must reference both:
   - quantitative evaluation results, and

- auditing considerations.

Challenge Question (Optional — if you finish early)

7. 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 write a rule-based aggregation step that outputs one final record (e.g., choose the highest-confidence chunk, union actors, keep the most specific location). In 2–4 sentences, explain how chunking changes failure modes.
   (b) **Abstention policy.** Add an explicit abstention rule: if the model is unsure, it must set `event_type = other` and add an uncertainty flag. Compare:
      - event-type macro-F1, and
      - the review queue size,

      before vs. after abstention. Interpret the trade-off in 5–8 sentences.