ML Evaluation Engineer - Technical Assessment

Estimated Time: 10-12 hours

Format: Take-home pipeline & report

Overview

You are building an automated system to analyze historiographical divergence. Your goal is to ingest raw, messy text from diverse sources and build an "LLM Judge" that can quantify how different authors describe the same 10 historical events.

We are testing three specific competencies:

- **Engineering Grit:** Can you scrape and structure data from difficult, real-world sources?
- **Evaluation Design:** Can you design an LLM system that reliably extracts relevant information?
- **Statistical Rigor:** Can you prove your evaluation system is trustworthy using concrete metrics?

Part 1. Data Acquisition & Normalization

Goal: Acquire, reconcile, and normalize information about Abraham Lincoln. You will need first-person accounts (letters, speeches, writings by Abraham Lincoln himself) and accounts by other authors who wrote about him.

Required Data Sources

A. Project Gutenberg: Other Authors

Acquire the following books from this source. Do NOT download these by hand; write a script to download them programmatically.

1. https://www.gutenberg.org/ebooks/6812
2. https://www.gutenberg.org/ebooks/6811
3. https://www.gutenberg.org/ebooks/12801/
4. https://www.gutenberg.org/ebooks/14004/
5. https://www.gutenberg.org/ebooks/18379

B. Library of Congress (LoC): First Person Accounts

Get the following files from LoC. Do NOT download these by hand; write a script to download them programmatically.

1. **Letter about Election night 1860:** https://www.loc.gov/item/mal0440500/

2. **Fort Sumter Decision:** https://www.loc.gov/resource/mal.0882800
3. **Gettysburg Address:** https://www.loc.gov/exhibits/gettysburg-address/ext/trans-nicolay-copy.html
4. **Second Inaugural Address:** https://www.loc.gov/resource/mal.4361300
5. **Last Public Address:** https://www.loc.gov/resource/mal.4361800/

Deliverable: Two Normalized JSON Datasets

Create one dataset from each source. Each item (a letter, a speech, a piece of writing, etc.) must be formatted according to the schema below:

JSON

```
None


{
  "id": "unique_identifier",
  "title": "Human-readable title",
  "reference": "something that points to this item in the raw
data",
  "document_type": "Letter | Speech | Note | etc.",
  "date": "As in the source (do not add or remove precision)",
  "place": "If known",
  "from": "If applicable",
  "to": "If applicable",
  "content": "Full source text as formatted in this source"
}
```

Requirements

- Respect LoC API rate limits.
- Keep raw text as faithful to the source as possible (no aggressive cleaning).
- Normalize metadata consistently.
- Make your dataset as complete as possible. If you can only obtain either dataset partially, explain why.

Part 2: Event Extraction

Objective: Extract information you can find on the following events from the data sources you have built above.

5 Key Events:

1. Election Night 1860
2. Fort Sumter Decision
3. Gettysburg Address
4. Second Inaugural Address
5. Ford's Theatre Assassination

Task:

For each data source in your dataset (book/letter), make LLM calls to extract all the information you can find on the 5 incidents. Build an extraction pipeline (using an LLM of your choice) that processes the raw text for each (Book, Event) pair and outputs:

JSON

```
None


{
  "event": "fort_sumter",
  "author": "Goodwin",
  "claims": [
    "Lincoln waited for the South to fire first",
    "Seward opposed the resupply mission",
    "The cabinet vote was 5 to 2"
  ],
  "temporal_details": {
    "date": "April 12, 1861",
    "time": "4:30 AM"
  },
  "tone": "Sympathetic"
}
```

Engineering Requirements:

- **Handle context window limits:** Books are long; how do you find the needle in the haystack?
- **Prompt:** Design your prompt to extract high-quality information. In the video walkthrough, explain your prompt design and why you think it led to better quality event information extraction.

Part 3: The LLM Judge & Statistical Validation (The Core)

Objective: Build and validate a system that detects inconsistencies between information presented by Lincoln himself and by other authors.

If you are new to the concept of LLM-as-a-judge, you can read more here:

- Confident AI: Why LLM as a Judge is the best method
- Evidently AI: LLM Guide
- HuggingFace Cookbook: LLM Judge

3A. Design the Judge

Create an LLM-based evaluator that takes two claim sets (Lincoln vs. Others for the same event) and outputs:

1. **Consistency Score (0-100):** 0 = Total contradiction, 100 = Perfect alignment.
2. **Contradiction Classification:** Identify if differences are:
    - *Factual* (Dates, numbers, locations).
    - *Interpretive* (Motivations, feelings).
    - *Omission* (Author A included it, B ignored it).

3B. Statistical Validation (Crucial)

An LLM Judge is useless if it isn't reliable. You must provide statistical evidence of your system's performance.

Required Experiments:

1. **Prompt Robustness (Ablation Study):**
    - Compare 3 prompt strategies (e.g., Zero-Shot vs. Chain-of-Thought vs. Few-Shot).
    - Which yields more stable results?
2. **Self-Consistency (Reliability):**
    - Run the same comparison 5 times with temperature > 0.
    - Calculate the standard deviation of the scores. Is your judge deterministic or noisy?
3. **Inter-Rater Agreement (Cohen's Kappa):**
    - Manually label a small subset (e.g., 10 event pairs) yourself as "Consistent" or "Contradictory".
    - Calculate Cohen's Kappa between your LLM Judge and your manual labels. This is your "Human Alignment" score.

Deliverables

1. **The Report (Primary Deliverable):** Your analysis, charts, and error breakdown.

2. **Code:** Python scripts for scraping, extraction, and evaluation.
3. **Data:** The extracted JSONs (so we can see what you managed to scrape).

How We Will Evaluate You

- **Scraping Difficulty:** Did you actually solve the hard engineering problem of getting modern text, or did you take the easy path?
- **Statistical Literacy:** Did you apply the metrics (Kappa, Variance) correctly? Do you understand what they mean?
- **Prompt Engineering:** Is your "Judge" prompt sophisticated? Did you use techniques like CoT or DSPy?
- **Insight:** Can you distinguish between a "noise" error (LLM hallucination) and a real data insight?