LLM-as-a-Judge: Chatbot Evaluation Framework

Colin Gibbons-Fly

Table of contents

1 Objective

This framework serves as a prototype template to evaluate chatbot responses in a highly regulated insurance context. While CSAA currently does not have AI chatbots in production, this Judge system provides a forward-looking architecture for evaluating their performance safely and objectively.

- **Use Case**: Research-grade benchmarking for insurance chatbots
- **Data**: Dummy data representing realistic customer scenarios
- **Goal**: Provide structured, domain-specific evaluation to ensure future AI assistants are accurate, empathetic, and trustworthy

2 Judge Design

2.1 `ChatbotJudge`

This class uses a large language model (Claude 3.5 Sonnet via AWS Bedrock) to score chatbot responses across three dimensions:

- **Accuracy (40%)**
  - Is the information factually correct for insurance?
  - Is it compliant with industry regulations?
  - Is the terminology precise?
- **Sentiment (40%)**
  - Is the response emotionally appropriate?
  - Does it demonstrate empathy, tone, and professionalism?
- **Confidence (20%)**
  - How certain is the evaluator about the judgment?
  - Is the LLM reasoning clear and consistent?

2.2 Scoring System

```json
{
  "overall_score": 7.5,
  "confidence": 8.5,
  "justification": "Brief explanation focusing on key strengths and weaknesses",
  "accuracy": 8.0,
  "sentiment": 7.0,
  "strengths": ["specific strength 1", "specific strength 2"],
  "weaknesses": ["specific weakness 1", "specific weakness 2"],
  "recommendations": ["actionable improvement 1", "actionable improvement 2"]
}
```

2.3 Threshold Handling

Responses scoring below a confidence threshold (default 0.7) are flagged and included in performance summaries.

3 Prompt Structure

The LLM receives structured prompts in the following format:

1. **Role Definition**: Claims the role of an expert insurance chatbot evaluator.
2. **Scoring Criteria**: Lays out the rubric for evaluating chatbot responses.
3. **Chain-of-Thought Reasoning**: Forces structured breakdown of:
   - Customer intent
   - Regulatory needs
   - Emotional tone
   - Industry best practices
4. **Evaluation Template**: Requires structured JSON output for tracking

4 Dataset Preparation

```python
def load_data(filename='chatbot_responses.csv'):
    # Loads prompts and responses from a CSV file
    # Validates label integrity and format
```

- Columns expected: `prompt`, `response`, `accurate`, `sentiment`
- Ground-truth labels are used only for correlation metrics, not decision-making.

5 Evaluation Flow

```python
def evaluate_all(df, judge):
    # Evaluates each prompt/response row using the ChatbotJudge
    # Computes weighted composite score and logs results
```

Each evaluation prints: - Customer prompt and chatbot response - Chain-of-thought reasoning (if enabled) - Subscores and justification - Strengths, weaknesses, recommendations

6 Metrics & Reporting

6.1 `print_summary(results)` and `calculate_research_metrics(results)`

Reports: - **Composite score distribution** - **Per-dimension averages** (accuracy, sentiment, confidence) - **Correlations** with ground-truth labels - **Score variance and standard deviation** - **False positives / negatives** for system reliability - **Confusion matrix breakdowns** by response category

7 Output Format

All detailed results are saved as:

```
research_evaluation_results.json
```

This includes: - LLM judgments - Timing - Recommendations - Evaluation confidence - Ground truth for offline analysis

8 Production Considerations

- Current data is simulated, not live chat logs.
- This framework is designed to generalize well once real chatbot data becomes available.
- Could be extended into real-time QA or model comparison in CI/CD pipelines.

9 Future Work

- Enable API-based chatbot evaluation for deployed systems
- Visualize score drift over time
- Add annotation tool for human-in-the-loop overrides
- Expand rubric with fairness/bias dimensions