AI Research Engineer Take-Home Assessment

Instructions:

- This assignment consists of two tracks with a total of 2 problems. Please choose **any one.**
- Time Limit: 1 week from date of receipt

Submission Format

- Zipped file with:
  - `track_{tracknum}/` directory for your chosen problem (for example `track_1` if you choose to solve)
  - README with setup instructions
  - Requirements file (can be .txt or .toml) for dependencies. Kindly make sure your results are reproducible and that it's easy for us to run your code.
  - Jupyter notebooks or scripts with inline documentation
  - Analysis report (PDF or markdown)
  - Please be careful to hide/remove any sensitive information like API keys before submitting

Track 1: Tree-Based Planning with LLMs

Objective

Develop a tree-based planning system that uses an LLM as a one-step action predictor to solve Sokoban puzzles of increasing complexity.

Task Description

Sokoban is a classic puzzle game where the player pushes boxes onto target locations. Your system should:

1. Parse Sokoban puzzle states from text format
2. Use an LLM to predict single actions given the current state
3. Implement a tree search algorithm that leverages these predictions

4. Solve puzzles with increasing difficulty

Constraints and Requirements

- **LLM Usage**: The LLM can only be used as a **one-step** predictor. Given a board state, it should output a single action (up, down, left, right) with optional confidence scores or reasoning
- **Model Selection**: Smaller open-source models preferred (e.g., Llama-7B, Mistral-7B, or smaller)
- **Data Source**: Use puzzles from David Skinner's Microban collection: http://www.abelmartin.com/rj/sokobanJS/Skinner/David%20W.%20Skinner%20-%20Sokoban_files/Microban.txt

Technical Specifications

- **State Representation**: Design how to represent the board state for the LLM (text, ASCII art, structured format, etc.)
- **Action Space**: Standard Sokoban actions (up, down, left, right)
- **Search Algorithm**: Implement a tree-based search (e.g., MCTS, A*, beam search) that uses LLM predictions to guide exploration
- **[Optional] Training with RL:** Train the model with RL and evaluate your new model against the search powered pipeline.
- **Evaluation**: Test on at least 10 puzzles of varying difficulty from Microban

Deliverables

1. **Code Implementation**:
    - Sokoban environment parser and simulator
    - LLM integration for action prediction
    - Tree search algorithm implementation
    - Evaluation framework

2. **Experimental Analysis**:
    - Compare different state representations for the LLM
    - Analyze the effect of different search strategies
    - Study how LLM prediction quality affects solving performance
    - Success rate vs. puzzle complexity analysis

3. **Discussion**:

    - How does your approach handle the constraint of single-step LLM usage?
    - What are the computational trade-offs of your search strategy?
    - How might you improve the system with more LLM calls or different architectures?
    - [Optional] RL-trained model vs Tree-Search

Track 2: Uncertainty Quantification in Language Models

Objective

Develop methods to estimate and evaluate uncertainty at both sentence and token levels in language model outputs, focusing on practical applications where knowing confidence is crucial.

Task Description

Language models often generate confident-sounding text even when uncertain. Your system should:

1. Implement uncertainty estimation techniques for an open-source LLM
2. Evaluate uncertainty quality across different types of tasks
3. Demonstrate practical applications of uncertainty-aware generation

Requirements

- **Model Selection**: Use smaller open-source models (e.g., Llama-7B, Mistral-7B, or smaller)
- **Granularity**: Implement both token-level and sentence-level uncertainty estimation
- **Evaluation**: Test on at least two different task types (e.g., factual QA, creative writing, summarization, math problems). It is up to you how you evaluate this system.

Deliverables

1. **Code Implementation**:

    - Uncertainty estimation methods
    - Evaluation framework with multiple metrics
    - Visualization tools for uncertainty analysis

2. **Experimental Analysis**:
    - Compare different uncertainty estimation methods
    - Analyze uncertainty patterns across different task types
    - Study the relationship between uncertainty and model performance
    - Calibration analysis with reliability diagrams

3. **Discussion**:
    - Which uncertainty methods work best for different scenarios?
    - How does computational cost scale with uncertainty quality?
    - What are the limitations of current approaches?

Dataset Suggestions

- **Factual QA**: Natural Questions, TriviaQA (for factual accuracy)
- **Math**: GSM8K, MATH (for reasoning tasks)
- **Summarization**: CNN/DailyMail, XSum (for abstractive tasks)
- **Open-ended**: Writing prompts, creative tasks (for subjective evaluation)

Evaluation Criteria:

- Provide working code implementation
- Include detailed analysis of your approach including motivation of the choice, design decisions, and any potential trade-offs
- Document experimental results with metrics and visualizations
- Discuss limitations and potential improvements
- We value creativity, rigorous analysis, and novel approaches over perfect solutions

---