MetaMind: A Cognitively Motivated Framework for Human-Like Social Reasoning

Overview

MetaMind is a novel multi-agent framework designed to endow Large Language Models (LLMs) with more human-like social reasoning capabilities. Traditional LLMs often struggle with the ambiguity and indirectness inherent in real-world human communication, failing to grasp unspoken intents, implied emotions, or culturally sensitive cues. MetaMind addresses this gap by explicitly modeling the multi-stage cognitive processes humans use to reason about unobservable mental states and navigate socially complex environments.

Inspired by psychological theories of metacognition and Theory of Mind (ToM), MetaMind employs a staged and collaborative system of three specialized agents to interpret, refine, and respond to social interactions with greater nuance, empathy, and cultural sensitivity.

> *"What is meant often goes far beyond what is said, and that is what makes conversation possible."* - H. P. Grice

Key Features & Core Concepts

MetaMind's architecture is built upon a layered reasoning process involving three distinct agents:

1. **Theory-of-Mind (ToM) Agent**: This agent initiates the reasoning process by generating multiple hypotheses about the user's latent mental state (beliefs, desires, intentions, emotions, thoughts) based on contextual and social cues. It aims to move beyond literal interpretations to infer what the speaker might be truly trying to convey.
    - **Hypothesis Generation**: Produces a set of candidate mental state interpretations with explanations and type labels.
    - **Mental-State Reasoning**: Involves commonsense-based hypothesis generation, cross-referencing with social memory, identifying ToM markers, and diversifying hypotheses.
2. **Domain Agent**: This agent refines the hypotheses generated by the ToM Agent by incorporating socially grounded constraints. It assesses the appropriateness of interpretations against broader norms like cultural expectations, ethical guidelines, and situational context, ensuring socially responsible and domain-aware reasoning.
    - **Hypothesis Refinement and Selection**: Revises hypotheses based on domain rules (cultural, ethical, role-based) and selects the most appropriate one using a scoring mechanism that balances contextual plausibility and information gain.
3. **Response Agent**: This agent generates a contextually appropriate output based on the refined optimal hypothesis and the user's social memory (e.g., emotional patterns, prior preferences). It also includes a self-validation mechanism to ensure the response aligns with the inferred intent and maintains social and semantic quality.

- **Generation and Validation**: Produces a natural language response conditioned on the selected hypothesis and social memory, then validates it for empathy and coherence.

Methodology

The MetaMind framework operates in a staged metacognitive loop:

1. **Stage 1: Generating Mental State Hypotheses (ToM Agent)**
    - Analyzes user input and conversational context.
    - Integrates information from a social memory module (storing user preferences and emotional markers).
    - Classifies potential mental states (Belief, Desire, Intention, Emotion, Thought).
    - Generates a diverse set of k candidate hypotheses.
2. **Stage 2: Refining Hypotheses (Domain Agent)**
    - Applies domain-specific rules (cultural, ethical, role-based) to revise each hypothesis.
    - Selects the optimal revised hypothesis based on contextual plausibility and information gain.
3. **Stage 3: Generating and Validating Output (Response Agent)**
    - Generates a response conditioned on the selected hypothesis and social memory.
    - Self-validates the response for emotional alignment (empathy) and contextual coherence.

Theoretical Grounding

MetaMind is grounded in established psychological theories:

- **Theory of Mind (ToM)**: The human capacity to attribute mental states—beliefs, desires, intentions, emotions—to oneself and to others, and to understand that others have beliefs, desires, intentions, and perspectives that are different from one's own.
- **Metacognition**: Awareness and understanding of one's own thought processes. MetaMind mirrors this by reflecting on, revising, and adapting its understanding in light of social norms and constraints.

Contributions

- Proposes a cognitively grounded, multi-agent framework that models human-like social reasoning by inferring mental states, incorporating social/ethical constraints, and adapting to user-specific patterns.
- Demonstrates through comprehensive evaluations significant improvements in contextual accuracy and social appropriateness in real-world scenarios, achieving state-of-the-art results on challenging benchmarks and matching human performance on key tasks.
- Provides in-depth ablation studies justifying the necessity of each agent and design choice for the framework's performance and generalization.

Running the Project

Follow these steps to get MetaMind up and running on your local machine.

1. Clone the Repository

First, clone the MetaMind repository to your local system using Git:

```
git clone <repository-url> # Replace <repository-url> with the actual URL of
your Git repository
cd MetaMind
```

2. Setup and Configuration

It's recommended to set up a virtual environment for Python projects.

```
python -m venv venv
# On Windows
.\venv\Scripts\activate
# On macOS/Linux
source venv/bin/activate
```

Install the required dependencies:

```
pip install -r requirements.txt
```

Configure the project by editing the `config.py`file. This file likely contains settings for API keys, model parameters, or other essential configurations. Please refer to the comments within `config.py` for specific instructions on what to set.

3. Running the Terminal Version

To run the terminal-based version of MetaMind, execute the `main.py` script:

```
python main.py
```

This will typically start an interactive command-line interface where you can interact with the MetaMind agents.

4. Running the Web Version

To run the web-based interface for MetaMind (if available), execute the `app.py` script:

```
python app.py
```

This will usually start a local web server. Open your web browser and navigate to the URL provided in the terminal output (commonly `http://127.0.0.1:5000` or similar).