Multi-Turn Interaction for Gorilla OpenFunctions

The current model (OpenFunctions-v2) is single-shot: one user prompt, one LLM response.
Objective: Enable multi-turn interactions to allow continuous conversations for debugging.
Goal: Collect self-instruct training data where users interact over multiple prompts for complex debugging tasks.

Challenges

Single-Shot Limitation: Currently, the model processes one query and returns a response without the ability to carry context forward for multi-step debugging.
Error Handling: When changing context (e.g., modifying items or switching restaurants), the model struggles to adjust effectively, as seen in soda vs. pizza test cases.
Server Errors: Large context switches or adding unexpected items sometimes cause internal server errors.

Proposed Techniques for Data Collection

Simulated Debugging Conversations:

1. User asks for a task (e.g., ordering items), followed by feedback to correct mistakes.
2. Training the model to handle feedback and adjust API calls iteratively.

Error-Driven Feedback

1. Gather examples where users provide specific error messages (e.g., "quantity is wrong") to simulate correction in subsequent turns.
2. Focus on collecting scenarios where changes are small (e.g., quantities) vs. big (e.g., switching items).

Paraphrasing/Backtranslation:

1. Collect different ways users express errors to make the model adaptable to various inputs.
    a. Ex: "I want 3 pizzas and 1 garlic bread" becomes "Make it 3 pizzas and 1 bread."

Real-World Crowdsourcing:

1. Collect debugging conversations from users in real environments to capture diverse use cases.

Supporting Results (Based on Code and Testing)

1. Initial tests show the model successfully adjusts quantities but fails to handle larger context changes (e.g., switching between pizza and soda).
2. Test cases show internal server errors when adding unexpected inputs, demonstrating the need for robust error handling in multi-turn settings.
3. Paraphrasing is effective for minor input variations, but more complex changes require further adaptation.

Next Steps

1. Expand the training dataset with simulated multi-turn conversations, focusing on edge cases (e.g., switching restaurants, modifying items).
2. Refine the model's ability to adjust based on user feedback across multiple turns.
3. Implement mechanisms to handle abrupt context changes without causing server errors.

Conclusion

By addressing these challenges and collecting high-quality self-instructing data, we can transform OpenFunctions into a more dynamic, multi-turn interaction system.