ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Final Project - Report

Group 1

Abdelrahman Saleh

Jacopo Maragna

01.08.2024

The Final Project implements a dutch auction using LLm-based agents as auctioneer and bedders.

How to run

Clone the repository and run the following Makefile targets:
- venv
- install
- run

Tech Stack

After careful consideration, our choice to employ AutoGen stemmed from its innate support for group chat communication—an ideal fit for our auction dynamics. Powered by OpenAI's GPT models, especially the formidable GPT-4, AutoGen promised the robustness and capabilities essential for our project's success.

Exploring open-source LLMs was part of our exploration, attempting to configure an inference server for local operations. Unfortunately, the computational demands surpassed our available resources by a significant margin, prompting us to pivot towards AutoGen for its efficiency and reliability.

Implementation

Libraries and Configuration

- The code imports necessary components from AutoGen, including agents, chat structures, and configuration utilities.
- It sets up LLM configuration parameters, initializing the AutoGen environment for the agents to interact.

Loading Prompts

- Three prompt files (`**auctioneer_prompt.txt**`, `**participant_prompt.txt**`, `**participant_prompt2.txt**`) are loaded to provide initial communication templates for the auctioneer and participants.

Agent Initialization

- `**UserProxyAgent**` is created to manage user interactions without human input, employing the LLM-based system for communication.
- `**AssistantAgent**` instances are initialized for the auctioneer and two participants (`**participant1**` and `**participant2**`), each configured with specific prompts and LLM settings.

Group Chat Setup

- A **`GroupChat`** instance is formed, incorporating the created agents (including the user proxy, auctioneer, and participants) and an empty message list.

Group Chat Manager

- The **`GroupChatManager`** is established, utilizing the configured group chat and LLM settings.

Chat Initiation

- The user proxy agent initiates the chat via the manager, sending an initial message "Run a dutch auction."

This code essentially sets up a simulated auction environment with LLM-powered agents, defining their roles (auctioneer, participants, and user proxy) and initializing communication structures within a group chat managed by AutoGen. It predefines initial messages and configurations for agents to simulate an auction scenario using AI-driven communication.

Prompts

Auctioneer

The prompt outlines the role and responsibilities of an auctioneer managing a Dutch auction, providing a structured guide to conduct the auction process effectively.

Role:

The participant assumes the role of an auctioneer in a Dutch auction, responsible for progressively decreasing the asking price until a participant accepts or until it reaches the predetermined reserve price.

Objective:

Auction Management: The auctioneer's primary objective is to manage the Dutch auction effectively by iteratively lowering the asking price based on participant responses.

Steps to Follow:

1. **Initialization:**
    - Start the auction by announcing the initial asking price.

2. **Participant Inputs:**
    - Participants respond with either 'accept' or 'refuse' to the proposed price.

3. **Evaluate Participant Responses:**

    - **All 'Refuse' Responses:**
        - Lower the asking price by a predetermined percentage.
        - Announce the updated price to encourage continued bidding.
    - **Even One 'Accept' Response:**
        - End the auction immediately.
        - Declare it 'closed' and announce the winning participant.

4. **Finalization:**
    - If the lowered price reaches the reserve price without any 'accept' responses, conclude the auction without a sale.

Provided Information:

- **Initial Parameters:** Initial asking price, lowering percentage, and reserve price are provided as placeholders within the prompt for the auctioneer to fill in with specific values.

Constraints:

- **Adherence to Instructions:** The auctioneer must strictly follow the outlined steps and procedures throughout the auction process.

Participants

The prompt outlines the role and responsibilities of a participant engaging in a Dutch Auction, providing guidelines on how to respond to the auctioneer's proposed prices.

Role:

The participant's role is to engage in the Dutch auction by evaluating the auctioneer's proposed prices and responding based on their predetermined maximum price they are willing to pay.

Objective:

- **Evaluation of Proposed Price:** The participant's goal is to assess each proposed price against their maximum price threshold ({price}) and respond accordingly.

Steps to Follow:

1. **Upon Price Proposal:**
    - The auctioneer presents a price for evaluation.

2. **Evaluate the Proposal:**
    - **If Below Maximum Price:**
        - Respond with 'accept' to signal willingness to purchase at that price.
    - **If Equal or Higher Than Maximum Price:**
        - Respond with 'refuse' to indicate the price exceeds the limit.

3. **Consistent Responses:**
    - Continuously assess each subsequent price proposal, ensuring responses align with the predetermined maximum price.

Notes:

- **Prompt and Clear Responses:** Participants must promptly and clearly respond to each proposed price with either 'accept' or 'refuse'.
- **Alignment with Maximum Price:** Decisions to accept or refuse should align with the participant's predefined maximum price ({price}).
- **Restrictions on Role Play:** Participants are prohibited from assuming the auctioneer's role or proposing new prices; their responses are limited to 'refuse' or 'accept'.

Results

```
user_proxy (to chat_manager):

Run a dutch auction.

------------------------------------------------------------------------
--------
auctioneer (to chat_manager):

Welcome everyone, we're starting a thrilling Dutch auction today! The
initial asking price for the item is 1000. Who'd like to start the
bidding at this price?

------------------------------------------------------------------------
--------
participant1 (to chat_manager):

refuse

------------------------------------------------------------------------
--------
participant2 (to chat_manager):

refuse

------------------------------------------------------------------------
--------
auctioneer (to chat_manager):

Alright! I understand. Let's lower the asking price by 5%. The new
asking price now stands at 950. Any takers at 950?

------------------------------------------------------------------------
--------
```

```
participant1 (to chat_manager):

refuse

------------------------------------------------------------------------
--------
participant2 (to chat_manager):

refuse

------------------------------------------------------------------------
--------
auctioneer (to chat_manager):

Got it, no takers at 950. Let's lower the price again by another 5%. The
new price now is 902.50. Does anyone accept the price of 902.50?

------------------------------------------------------------------------
--------
participant1 (to chat_manager):

accept

------------------------------------------------------------------------
--------
auctioneer (to chat_manager):

Congratulations, Participant1! You've won the auction. We close the
bidding at the price of 902.50. Everyone else, thank you for
participating. We hope to see you again at our next auction!
```

Discussion / Conclusion

The culmination of our Final Project marks a successful implementation of a Dutch auction orchestrated through LLM-based agents, functioning both as auctioneers and participants.

Execution Steps

To run the project, one can easily clone the repository and execute the Makefile targets: **`venv`**, **`install`**, and **`run`**.

Technology Stack

Our decision to utilize AutoGen was deliberate, driven by its proficiency in facilitating group chat communications, an ideal fit for the intricacies of our auction dynamics. Leveraging the robustness of OpenAI's GPT-4 models within AutoGen promised the essential capabilities for our project's triumph.

Implementation Overview

The code architecture involves the importation of necessary components from AutoGen, configuring LLM parameters to enable interaction among agents within the AutoGen environment. Initial communication templates were loaded to set the tone for interaction between the auctioneer and participants. Agent initialization and group chat setup ensued, with the GroupChatManager established to oversee interactions, facilitating the commencement of the auction through the user proxy agent's initiation.

Prompts and Role Definitions

The meticulously crafted prompts defined distinct roles and responsibilities for both the auctioneer and participants, outlining precise steps to follow. For the auctioneer, the objective centered on efficiently managing the Dutch auction by progressively adjusting the asking price based on participant responses until a bid was accepted or the reserve price was reached. Participants, on the other hand, were tasked with evaluating proposed prices against their predetermined maximum price, responding with 'accept' or 'refuse' accordingly.

Results and Conclusion of Auction

The conducted simulation showcased the robustness of our implemented system. Despite multiple price adjustments, the auction concluded successfully when a participant accepted the price of 902.50, thereby winning the auction. This highlights the efficacy of our design in orchestrating a dynamic and engaging auction environment.

In essence, our project demonstrates the potential of AI-powered agents in conducting auctions, showcasing their adaptability in managing complex bidding processes while maintaining the integrity of auction dynamics. This endeavor not only exemplifies the capabilities of AI-driven auctioneers but also underscores the participatory role of AI-guided participants in such settings.