Benchmarking AutoGen with different large language models

Rafael Barbarroxa
GECAD – Research Group on Intelligent Engineering and Computing for Advancement Innovation and Development, LASI – Intelligent Systems Associate Laboratory, Polytechnic of Porto
Porto, Portugal
0000-0002-7245-6395

Bruno Ribeiro
GECAD – Research Group on Intelligent Engineering and Computing for Advancement Innovation and Development, LASI – Intelligent Systems Associate Laboratory, Polytechnic of Porto
Porto, Portugal
0000-0001-5105-8905

Luis Gomes
GECAD – Research Group on Intelligent Engineering and Computing for Advancement Innovation and Development, LASI – Intelligent Systems Associate Laboratory, Polytechnic of Porto
Porto, Portugal
0000-0002-8597-3383

Zita Vale
GECAD – Research Group on Intelligent Engineering and Computing for Advancement Innovation and Development, LASI – Intelligent Systems Associate Laboratory, Polytechnic of Porto
Porto, Portugal
0000-0002-4560-9544

Abstract— **Although a controversial solution for daily use, the large language models (LLM) appeared near the users as an incredible solution that demonstrates the usefulness of artificial intelligence in recurring tasks. While current LLM solutions are based on direct queries from the users, the use of multi-agent systems supported by LLMs could be a major evaluation in distributed solutions. The few solutions that support multi-agent systems for LLMs, such as ChatDev and AutoGen, were developed based solely on ChatGPT LLM. This paper explores the feasibility of changing the LLM used in a configuration of two agents developed in AutoGen. The results show that the GPT models provided by OpenAI are still the best performers, however, other models can be chosen to reduce costs and still have good results.**

Keywords— AutoGen, large language models, multi-agents

I. Introduction

The idea of intelligent agents and multi-agent systems (MAS) has been researched for several years. Their unique ability to simulate human behaviors helped to solve many complex problems. For example, in the energy domain, MAS are used in smart grids [1], auction markets [2], energy communities [3], and intelligent energy management systems in real buildings [4]. However, the reasoning capacity of agents was always limited by the technology of the time, both in terms of hardware and algorithms. Only more recently has the reasoning of these agents greatly improved thanks to machine learning (ML), namely deep learning models.

At the end of 2022, OpenAI published an artificial intelligence (AI) bot called ChatGPT [5] based on the third version of Generative Pre-trained Transformer (GPT-3) [6]. As revolutionary as it was, ChatGPT rapidly impacted the world and how people viewed and used AI and the internet. After this breakthrough, other companies launched their large language models (LLM), such as Google with Bard [7] and Microsoft with Copilot [8].

Following these developments, researchers in the MAS field started to wonder what endeavors several agents equipped with LLMs could do. By using agents with natural language capacities, they could work together as a team to create more accurate and full-fledged solutions, not only for text-based problems but, eventually, for code problems as well [9]. An example of this is ChatDev [10] which is a MAS framework based on LLMs that replicate a real software development company, where the employees are agents. The agents are organized by several departments, as normal companies, such as the product management, coding, testing, and documentation departments. Another approach to this is AutoGen [11], by Microsoft. AutoGen's focus is not only to produce software but to solve several complex tasks like math problems. The user can configure how many and which type of agents she/he wants. Both ChatDev and AutoGen use ChatGPT to empower the agent’s communication and reasoning skills.

The main motivation of this paper is to compare AutoGen agents’ performance using different LLMs. The models being used are the GPT-3.5-turbo, GPT-4, Llama 2 [12], Mistral [13], and Zephyr [14]. The case study (CS) is composed of a unique, but complex, ML problem where different configurations of agents will try to solve it. Further, the speed and quality of each configuration are analyzed.

The paper is divided into four sections: the Introduction, where the motivation and related works are described, the Methodology, where the approach used to validate the agents is described, the Experimentation, where the results of the experimentation are discussed, and finally the Conclusions describe the main conclusions of this work.

II. Methodology

AutoGen was used to define and implement the agents of the CS and text-gen-webui as an application programming interface (API) that hosted the LLMs used by the agents. Two servers were used to host the LLMs in addition to a computer that hosted AutoGen agents. The MAS used in this paper was composed of two agents, the product manager (PM) and the coder. The former is responsible for creating a solution plan for the user prompt, the latter developed the code to solve it. The LLMs used were GPT-3.5-turbo and GPT-4, from OpenAI, Llama 2 13B from Meta AI, Mistral 7B from MistralAI, and Zephyr 7B from Hugging Face. For the experiment, different LLM variations of the coder and PM roles were tested, which resulted in 25 combinations.

To test the LLM combinations, the agents were prompted to create a Python script that uses an ML model to forecast the energy consumption of a building. The dataset's data was gathered from a building [15] in the form of a comma-separated values (CSV) file with three columns: the datetime, the total consumption for that period, and the mean temperature of each period. The CSV file is given to the model manually. The performance of each MAS was evaluated by comparing the results of the trained ML models. The prompt was the following: "*If I wanted to forecast the energy consumption of my home, what is the best machine learning algorithm to use and why? How could I implement it if I had a csv file named: energy_consumption.csv with three columns, the first is datetime, the second "total" is the total consumption and the third "temp_x10" is temperature multiplied by 10? What measure/metric/function should I choose and how should I use it to validate the model performance?*"

III. Experimentation

Of all the 25 possible configurations, only eight provided a good solution to the problem, as shown in Table 1. The script of each configuration was run at least two times to guarantee consistency. The first two columns correspond to the LLMs used for the coder and the PM. Despite not mentioning any specific metric to be used, every agent configuration that provided working solutions automatically used at least the root mean square error (RMSE) to evaluate the model trained. Because of this, a test dataset, unknown to the agents, representing a week of consumption at 15-minute intervals was used to generate test predictions which were then used to calculate RMSE manually for comparison. The runtime is the time from the request input until the replied answer was given, in some cases, such as the first line of the table, the agents executed the model that they created during running, increasing the runtime above normal.

TABLE I. RESULTS FROM THE DIFFERENT AUTOGEN CONFIGURATIONS.

Coder	Product Manager	RMSE (Watts)	Runtime	Model
GPT-4	GPT-3.5	64.864	~24 hours	ARIMA
GPT-3.5	GPT-3.5	256.264	205,8 s	LR
Mistral	GPT-4	256.240	286,7 s	LR
GPT-4	Zephyr	312.501	300 s	ARIMA
GPT-3.5	GPT-4	312.501	128,2 s	ARIMA
GPT-3.5	Zephyr	312.501	132,3 s	ARIMA
Llama 13B	GPT-3.5	501.498	633 s	LSTM
GPT-4	GPT-4	913.470	160,4 s	RFR

Overall, it is shown that the GPT models can achieve, most of the time, the best results considering their individual performance. However, the authors think that because AutoGen was specifically developed for the GPT models, the role configuration can be better tailored and adjusted depending on the model being used. Despite the GPT-4 (coder) and GPT-3.5 (PM) configuration achieving the best results, two comments must be made, (i) the time spent to achieve the result was almost one day, and (ii) after verifying the code outputs from their solution it is very possible that the model was overfitting. In this experiment, only configurations with at least one GPT-based LLM, in any role, produced working solutions. However, costs can be reduced by switching one of the agents to an open-source model.

Several of the remaining configurations could not even come up with a runnable solution, only GPT-4 (coder) + Mistral (PM) could, however, the agent chose to normalize the data and did not reverse it at the end, and for that reason is not considered as a feasible solution in Table 1.

IV. Conclusions

This paper tests the AutoGen framework against different large language models and compares their performance. The GPT 3.5, GPT 4. Llama 2, Mistral and Zephyr models were used. The configuration consisted of two agents, the product manager and the coder. The results show that GPT models are still the best fit for AutoGen, however, it is possible to substitute one agent model to reduce costs and still have good reasonable results. For future research, the role description of the agents could be explored to be better tailored to the specific model being used.

Acknowledgment

This article was developed under the "NGS - New Generation Storage" Innovation Pact (C644936001-00000045), by the "NGS" Consortium, co-funded by the NextGenerationEU, through the investment "Agendas para a Inovação Empresarial" of the Portuguese Recovery and Resilience Plan (PRR)." We also acknowledge the work facilities and equipment provided by the GECAD research center (UIDB/00760/2020 - 10.54499/UIDB/00760/2020 - https://doi.org/10.54499/UIDB/00760/2020).

References

[1] H. Pereira, B. Ribeiro, L. Gomes, and Z. Vale, "Smart Grid Ecosystem Modeling Using a Novel Framework for Heterogenous Agent Communities," *Sustainability*, vol. 14, no. 23, p. 15983, Nov. 2022, doi: 10.3390/su142315983.

[2] B. Veiga, G. Santos, T. Pinto, R. Faia, C. Ramos, and Z. Vale, "Simulation tools for electricity markets considering power flow analysis," *Energy*, vol. 275, p. 127494, Jul. 2023, doi: 10.1016/j.energy.2023.127494.

[3] T. Pinto, L. Gomes, P. Faria, Z. Vale, N. Teixeira, and D. Ramos, "Intelligent Simulation and Emulation Platform for Energy Management in Buildings and Microgrids," in *Machine Learning for Smart Environments/Cities*, 2022, pp. 167–181. doi: 10.1007/978-3-030-97516-6_9.

[4] B. Ribeiro, R. Faia, L. Gomes, and Z. Vale, "Energy Community Integration of a Smart Home Based on an Open Source Multiagent System," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*, 2023, pp. 415–421. doi: 10.1007/978-3-031-37616-0_35.

[5] OpenAI, "ChatGPT: Optimizing Language Models for Dialogue." Accessed: Jan. 22, 2023. [Online]. Available: https://openai.com/blog/chatgpt/

[6] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *ArXiv*, May 2020.

[7] Google, "Bard." Accessed: Dec. 04, 2023. [Online]. Available: https://bard.google.com/chat?hl=pt-PT

[8] Microsoft, "Copilot." Accessed: Dec. 04, 2023. [Online]. Available: https://copilot.microsoft.com/

[9] L. Wang *et al.*, "A Survey on Large Language Model based Autonomous Agents," *ArXiv*, Aug. 2023.

[10] C. Qian *et al.*, "Communicative Agents for Software Development," *ArXiv*, Jul. 2023.

[11] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation," *ArXiv*, Aug. 2023.

[12] H. Touvron *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," *ArXiv*, Jul. 2023.

[13] A. Q. Jiang *et al.*, "Mistral 7B," *ArXiv*, Oct. 2023.

[14] L. Tunstall et al., "Zephyr: Direct Distillation of LM Alignment," ArXiv, Oct. 2023.

[15] C. Goncalves, R. Barreto, P. Faria, L. Gomes, and Z. Vale, "Dataset of an energy community's generation and consumption with appliance allocation," Data Brief, vol. 45, p. 108590, Dec. 2022, doi: 10.1016/J.DIB.2022.108590.