Intelligent Human Systems Integration (IHSI 2024), Vol. 119, 2024, 144–153
https://doi.org/10.54941/ahfe1004478

Magenta: Metrics and Evaluation Framework for Generative Agents Based on LLMs

Sudarshan Kamath Barkur, Pratik Sitapara, Sven Leuschner, and Sigurd Schacht

University of Applied Sciences, 91522 Ansbach, Germany

ABSTRACT

Large Language Models (LLMs) have gained growing significance within the realm of Natural Language Processing (NLP) and have showcased their immense potential across a wide spectrum of applications, including the development of Autonomous Agents. Through the meticulous assessment of their performance and subsequent enhancements, researchers and engineers can continue to develop advanced AI systems that can address user needs and perform tasks with high accuracy and efficiency. In this paper, we discuss the current state of evaluation of language models and autonomous agents and propose a new evaluation benchmark for autonomous agents, introducing two metrics - effort and success rate. Effort measures the number of steps or actions the agent takes to complete a given task, whereas the success rate measures the correctness or quality of the agent's output. We argue that these two metrics capture important aspects of the agent's performance and trade-offs. We show that our benchmark can provide useful insights into the strengths and weaknesses of different agents. As a work in progress, we suggest standardizing the evaluation process of LLMs by leveraging existing frameworks and transforming them into an automatic evaluation framework that evaluates all kinds of language models, agents, and LLM-based applications. This approach aims to provide a comprehensive and unified evaluation method, enabling the users to make informed decisions when selecting and fine-tuning LLMs for their specific needs and calculating the accuracy of language-based models and applications.

Keywords: Large language models, Autonomous agents, Evaluation, Llama, Generative agents, LLMs, GPT, Framework

INTRODUCTION

Natural Language Processing (NLP) is a field of research that aims to create systems and applications that can understand and generate natural language. It forms a basis for the Large Language Models (LLMs) which are probabilistic models that can produce fluent and coherent text and perform various tasks with high accuracy. LLMs are trained on huge amounts of text data, which gives them the ability to generate realistic responses and handle difficult and complex tasks. LLMs can be applied to many NLP tasks, such as text summarization, machine translation, natural language generation, question answering, and more.

A prompt serves as a textual input to the model, functioning as the initial cue or framework for its response. Furthermore, the prompt can be used to establish the context before presenting the task. This enables LLMs to generate more coherent and contextually relevant responses, making them suitable for interactive and conversational applications (Kaddour et al., 2023) without any fine-tuning (re-training the model for a particular task/dataset). This leads us to the Retriever Augmented Generation.

Retrieval Augmented Generation (RAG)

RAG can be used to augment the prompt with real-time information, such as information from the databases, leading to accurate and relevant content. In addition, the retrieval indices can be kept up-to-date rather than continuously fine-tuning an LLM. The context length of an LLM determines the extent of text that the model can process at once. Typically, this limit is relatively short ranging from 2048 - 4096 tokens. Therefore, the documents are chunked (split into shorter paragraphs) and given as the context in the prompt to the model.

Autonomous Agents

In addition to retrieval capabilities, LLMs can be integrated with tools such as calculators and internet search engines, giving rise to Autonomous Agents. These agents, capable of following natural language instructions, complete tasks within visually perceived environments. LLMs play a key role in the development of autonomous agents that can effectively and efficiently meet user needs. Some examples of such agents include HuggingGPT (Shen et al., 2023), Auto-GPT (Yang, Yue, and He, 2023), Baby-AGI (Nakajima, 2023), and GPT-Engineer (Osika, 2023).

These agents are smarter and capable of solving complicated problems that require multiple steps and sources of information. These agents are adept at tasks such as code generation, data retrieval, text summarization, question answering, crafting fluent responses, managing emails, scheduling appointments, making online purchases, and adapting based on user feedback.

Problem-Solving Paradigms for Autonomous Agents

Here, we discuss a few of the paradigms for constructing an autonomous agent. These are:

- **Chain of thought**: It involves instructing the model to break down challenging tasks into smaller and more manageable steps, providing insights into the model's thought process (Zhang et al., 2023).
- **Tree of Thoughts**: An extension of CoT, Tree of Thoughts takes the idea further by exploring multiple possible reasoning paths at each step. It decomposes problems into multiple thought steps, generating multiple thoughts at each step, and creating a tree-like structure (Yao et al., 2023).
- **Self-Refine:** An approach that improves LLM outputs through iterative feedback and refinement by generating an initial output and refining itself based on feedback (Madaan et al., 2023).

- **ReAct**: Reasoning and Acting (ReAct) integrates reasoning and action within LLM by expanding the action space to include a combination of task-specific discrete actions and language-based actions. The former allows LLM to interact with the environment, such as using the Wikipedia search API, use tools like calculator and Python, whereas the latter prompts LLM to generate reasoning traces in natural language (Yao et al., 2022).

Use-Case: Powering an Autonomous Agent Based Chatbot Using the ReAct Paradigm

The University of Applied Sciences at Ansbach used a Chatbot called DIAS built with the Rasa, which uses a DIET classifier (Dual Intent and Entity Transformer), which is trained to recognize the entities and the intents in a sentence. A fixed answer is denoted for each intent by using rules. Additional training is done also for the flow of the conversation. For the DIAS bot, automation was done to create the required files and rules for training by programmatically extracting the data from a CSV file, which contained the information required. Using Rasa actions, a knowledge graph was also connected to the bot, which contained the information about the persons at the university, and the study programs, which was obtained by scraping the website of the university.

As LLMs like LLaMA became popular, along with frameworks like LangChain (LangChain, 2023), it was then decided to migrate the bot to a generative language model, with the ability to retrieve the information directly from various files. This would also help reduce the perception that the chatbots are "robotic" and reduce the efforts to maintain the bot. For the autonomous agent based bot (referred to as DIASv2), the existing data sources of DIASv1 were used, by referring to each data source as a tool in the ReAct prompt.

The text information made available to the bot consisted of the FAQs, the Study Programs, information about the persons working at the university, Mensa Menu (Canteen Menu), and the opening times of different organizations within the university.

For the retrieval, the embeddings were created by the text-ada-embedding-002 model from OpenAI. Initial data creation was done by constructing an agent using GPT-4 in the background. Discord was used as a front-end, for testing and data collection. Later, the tools were themselves bundled with LLMs, therefore creating an agents-calling-agents network, which helped in shortening the overall context length, before passing the information to the main agent. The context to train the main agent needs to be less than 2048 tokens, as the target LLM was a LLaMA-13B based model, which had shorter context as opposed to GPT-4. The LangChain framework was used to create the agents with ReAct paradigm, as it offers customization and easy integration with different vector databases.

Once the data collection was finished, it was then augmented by changing the names of the tools, leading to 900 examples to train the model. Guanaco-13B was the target model, trained using QLoRa with 4-bit precision

(Dettmers et al., 2023). A single H100 GPU was used and training was done for 4 epochs. The adapter generated was then merged with the base model to create the DIAS-13B model.

As the agent has access to the tools, it needs to select the correct tool and then arrive at the correct answer. For some questions, the answer is arrived at by taking the output from multiple tools and reaching a conclusion. Therefore, the decision-making process of the agent also decides the overall execution time.

Two factors are important in deciding whether a given LLM can be used as a basis for the agent:

- Ability to follow the prompts correctly
- Target language and target domain

For example, some language models are not trained equally in other languages compared to English and require additional fine-tuning for the target language or a specialized domain such as law or medicine. Therefore, evaluations are necessary to identify and address these weaknesses.

EVALUATION

As discussed in the previous section, evaluations help in selecting the base LLM that powers an agent. The evaluations allow us to compare the different models for a given domain and tasks, detect biases, and also look at how models follow instructions. In the case of agents, evaluations make sure that the agents are explainable and check for safety and reliability before deployment.

Current Frameworks for Evaluating LLMs

- **MMLU**: A set of 57 tasks that span elementary math, US history, computer science, law, and more. To perform well, models must possess extensive world knowledge and problem-solving ability (Hendrycks et al., 2020).
- **LM-evaluation-harness**: A unified framework to test models via zero/few-shot settings on 200 tasks. Incorporates a large number of evaluations including BigBench, MMLU, etc. (EleutherAI, 2023).
- **HELM**: Instead of specific tasks and metrics, HELM offers a comprehensive assessment of LLMs by evaluating them across domains. Metrics include accuracy, calibration, robustness, fairness, bias, toxicity, etc. Tasks include Q&A, information retrieval, summarization, text classification, etc. (Liang et al., 2022).
- **AlpacaEval**: An automated evaluation framework that measures how often a strong LLM (e.g., GPT-4) prefers the output of one model over a reference model. Metrics include win rate, bias, latency, price, variance, etc. Validated to have a high agreement with 20k human annotations (Tatsu-Lab, 2023).

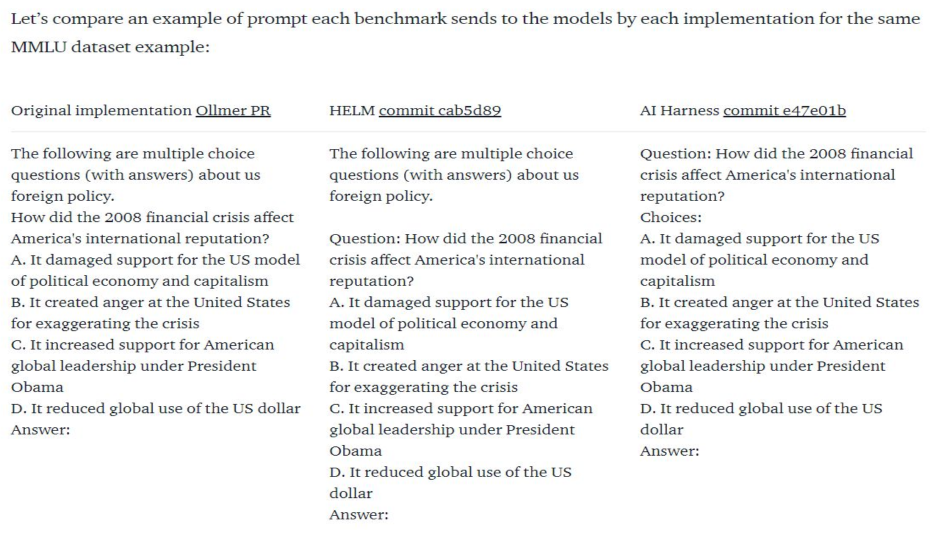
Current Frameworks for Evaluating Agents

- **ToolLLM:** Enabling Large Language Models with 16,000+ Real-world APIs: ToolBench, a dataset of 16,464 real-world RESTful APIs with human instructions, DFSDT, a novel depth-first search-based decision tree algorithm for efficient problem-solving, and ToolEval, an automatic evaluator for assessing LLMs' tool usage. A LLaMA model fine-tuned using ToolBench, outperforms other benchmarks, showcasing performance akin to ChatGPT and surpassing them in tool usage (Qin et al., 2023).

- **AgentBench**: AgentBench addresses the lack of systematic LLM evaluations in interactive settings with eight evolving environments. These environments test LLM-as-Agent's reasoning and decision-making across text adventure games, web navigation, and lateral thinking puzzles. AgentBench assesses understanding, planning, execution, and evaluation. The study evaluates 25 LLMs, including GPT-3.5, GPT-4, and open-source models like LLaMA. Results show that top commercial LLMs excel in complex environments, outperforming open-source competitors. The authors also analyze strengths, weaknesses, and common challenges across LLMs (Liu et al., 2023).

Challenges in Evaluating LLMs

According to a recent blog by HuggingFace, the comparison of the implementation of MMLU with HELM and LM-Evaluation-Harness led to the conclusion that the same examples have different prompts across different frameworks and minor differences in the tokenization and punctuation affect a model's performance (Fourrier et al., 2023).



Let's compare an example of prompt each benchmark sends to the models by each implementation for the same MMLU dataset example:

Original implementation Ollmer PR	HELM commit cab5d89	AI Harness commit e47e01b
The following are multiple choice questions (with answers) about us foreign policy. How did the 2008 financial crisis affect America's international reputation? A. It damaged support for the US model of political economy and capitalism B. It created anger at the United States for exaggerating the crisis C. It increased support for American global leadership under President Obama D. It reduced global use of the US dollar Answer:	The following are multiple choice questions (with answers) about us foreign policy. Question: How did the 2008 financial crisis affect America's international reputation? A. It damaged support for the US model of political economy and capitalism B. It created anger at the United States for exaggerating the crisis C. It increased support for American global leadership under President Obama D. It reduced global use of the US dollar Answer:	Question: How did the 2008 financial crisis affect America's international reputation? Choices: A. It damaged support for the US model of political economy and capitalism B. It created anger at the United States for exaggerating the crisis C. It increased support for American global leadership under President Obama D. It reduced global use of the US dollar Answer:

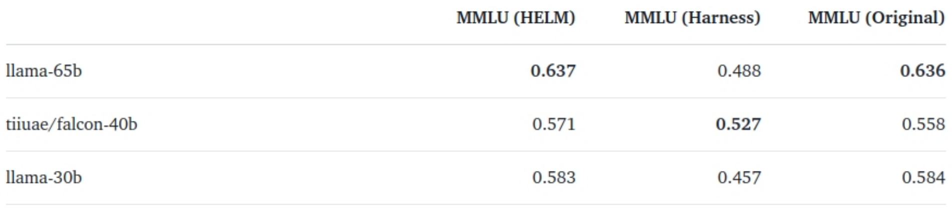
Figure 1: Comparing the prompts from different frameworks (Fourrier et al., 2023).

The prompt used in different benchmarks may have a different structure for the same dataset. As we can see in Figure 1, the prompt is the same in all three benchmarks but has minor differences. LM-evaluation-harness does not include any topic line or instruction. Also, the same benchmark adds a small keyword 'Choices' compared to the other two benchmarks, in HELM and

LM-evaluation-harness add a 'Question:' prefix. In the case of the original Implementation by MMLU, there is no newline separating the instruction and question compared to HELM.

Method of Evaluation in Each Benchmark for the Same Prompt:

MMLU: Compares predicted probabilities on the answers only (A, B, C, D)
HELM: Uses the next token probabilities from the model and picks the token with the highest probability, even if it's not one of the options.
LM-evaluation-harness: Computes the probability of the full answer sequence (i.e., a letter followed by the answer text) for each answer, picking the answer with the highest probability.



	MMLU (HELM)	MMLU (Harness)	MMLU (Original)
llama-65b	**0.637**	0.488	**0.636**
tiiuae/falcon-40b	0.571	**0.527**	0.558
llama-30b	0.583	0.457	0.584

Figure 2: Shows the result of 3 different models which are evaluated with the same dataset using different benchmarks that we discussed. Also, the evaluation methods for each benchmark are different as discussed. Reason for different numbers as results which are not comparable with the same prompt.

Therefore, the same dataset could have different scores based on the evaluation framework used, leading to different absolute scores and different model rankings on different evaluation frameworks (Fourrier et al., 2023). This poses a challenge for the Open LLM Leaderboard and calls for standardization and transparency in the evaluation process.

METHODOLOGY AND RESULTS

Methodology

To evaluate the bot (agent), we follow the following steps:

- Data preparation: Prepare the fictional data for evaluating the bot. The prepared data should be embedded and made available to the retriever used by the bot.
- Questions and Gold Answers preparation.
- Definition of the metrics.

For the test data preparation, we created 100 fictional persons and 20 fictional study programs, such that they mimic the real data. This was done to ensure the train data was not present in the test data. We created 10 single-step questions and 10 multi-step questions to test the agent on these questions. By single-step, it means that only one step is taken to answer the question (usually by the data from the retriever). Likewise, for the multi-step, it means that the retriever model has to be accessed twice, combine the data, and then conclude. For each question, we define the gold answer i.e. the model answer.

We then defined the metrics for the evaluation: Effort and Success rate.

- **Effort**: We count the number of steps the model takes to complete the task and then divide it by the steps taken by the gold answer. 1 is the ideal score obtained here. Lower effort indicates skipping of steps, which can be due to hallucination.
- **Success rate**: We check whether the end answer of the model matches the gold answer. If only half of the answer is present (if two answers are required, then the success rate is 0.5).

For testing, we based different models as the backbone of the agent. Most models are hosted by TogetherAI (Together AI, 2023), whereas the DIASv2 model is hosted on our servers.

In addition, we test the performance of such an agent in the English language, by giving it twenty such questions with access to the following tools: Calculator, Wikipedia, Arxiv, PythonRepl, Random Number Generator, and DuckDuckSearch. The use of these tools is required to obtain the answers to the questions.

We build this approach programmatically by defining a task file in JSON. The file consists of the following fields: Models to be evaluated, questions, gold steps, and gold answers. The generation of gold steps and gold answers is done with the help of GPT-4. Once this file is defined, it is then used by a Python script to evaluate these models and come up with the results. The result is again stored in a JSON file.

RESULTS

This section discusses the comparison of the various models of the LLaMA family, ranging from 7 Billion parameters to 70 Billion parameters. Initially, the evaluation is done for the DIASv2 agent, which is fine-tuned, and compared with the other models from the LLaMA family. Later, the evaluation is done keeping generic English language questions in mind.

As discussed earlier, we define two metrics, the success rate and effort. The Success rate metric can have values (0, 0.5, and 1). 1 indicates a correct answer and 0 indicates an incorrect response, 0.5 indicates that one part of the answer is correct. In parallel, the Effort metric quantifies the steps taken by the model divided by the gold-standard steps, with 0 signifying either zero steps taken or parsing errors. A lower effort score often suggests hallucination, indicating that the model skipped essential intermediate steps to arrive at an answer. Combining these metrics provides a comprehensive evaluation of the model's correctness and the extent to which it accurately follows the desired prompt.

When it became clear that the agents could be evaluated in this manner, we also tested the agents with tools for questions in English. The results are mentioned for the Single-step problems in Table 3 and the multi-step problems in Table 4.

Table 1. Overall performance for single-step problems for the university agent in the German language. Values indicated are in the form of success rate and effort (success rate, effort). The orange highlight indicates successful completion.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
LLaMA-2-7b-chat	0, 0	0, 1	0, 0	0, 1	0, 0	0, 0	0, 0	0, 1	0, 0	0, 0
LLaMA-2-13b-chat	0, 0	0, 0	0, 0	0, 0	0, 0	0, 2	0, 5	0, 0	0, 1	0, 1
LLaMA-2-70b-chat	**1, 1**	0, 4	0, 1	**1, 1**	**1, 2**	0, 1	0, 1	0, 0	0, 0	0, 0
WizardLM-70B-V1.0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0
Vicuna-13b-V1.5	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0
DIAS-13B	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 2**	**1, 1**	**1, 1**

Table 2. Overall performance for multi-step problems for the university agent in the German language. Values indicated are in the form of success rate and effort (success rate, effort).

	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
LLaMA-2 7b-chat	0, 0.5	0, 0.5	0, 0	0, 0.5	0, 0.5	0, 0	0, 0	0, 0.5	0, 1	0, 0
LLaMA-2 13b-chat	0, 0	0, 0.5	0, 0.5	0, 0	0, 0.5	0, 0.5	0, 0	0, 0	0, 0	0, 0
LLaMA-2 70b-chat	0, 0	0, 0	0, 0	0, 0	0, 0.5	**0.5, 1**	0, 1	0, 1	0, 0	0, 0
WizardLM- 70B-V1.0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0
Vicuna-13b -V1.5	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0	0, 0
DIAS-13B	**0.5, 0.5**	**1, 1**	**1, 2**	**0.5, 0.5**	**1, 1**	**1, 1**	0, 0.5	0, 1	**0.5, 0.5**	**0.5, 0.5**

Table 3. Overall performance for single-step problems for generic questions in the English language. Values indicated are in the form of success rate and effort (success rate, effort).

	GQ11	GQ12	GQ13	GQ14	GQ15	GQ16	GQ17	GQ18	GQ19	GQ20
LLaMA-2 7b-chat	0, 3	0, 2	0, 5	**1,1**	**1, 1**	0, 5	0, 0	**1, 1**	**1, 1**	**1, 2**
LLaMA-2 13b-chat	0, 1	0, 1	0, 0	**1, 1**	**1,1**	0, 2	0, 0	0, 0	**1, 0**	**1, 2**
LLaMA-2 70b-chat	0, 0	0, 1	**1, 1**	**1, 1**	0, 5	**1, 1**	**1, 1**	0, 0	**1, 0**	**1, 1**
Wizard LM-70B-V1.0	0, 3	0, 2	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 1**	**1, 2**	**1, 1**	**1, 1**

Table 4. Overall performance for multi-steps problems for generic questions in the English language. Values indicated are in the format of success rate and effort (success rate, effort).

	GQ1	GQ2	GQ3	GQ4	GQ5	GQ6	GQ7	GQ8	GQ9	GQ10
LLaMA-2 7b-chat	0, 0.5	0, 1	**1, 0.5**	0, 1	0, 2	0, 0.5	**1, 0.05**	**1, 1**	0, 1.5	0, 1
LLaMA-2 13b-chat	0, 0.5	0, 0.5	0, 0	0, 0	0, 0	**1, 1.5**	**1, 1**	**1, 1.5**	0, 0	**1, 1**
LLaMA-2 70b-chat	0, 0.5	0, 0	0, 0	0, 0	0, 0	**1, 1.5**	**1, 1**	0, 0	0, 2	**1, 1**
Wizard LM-70B-V1.0	**1, 2**	0, 1	0, 1	0, 2.5	0, 2	0, 1	**1, 1**	**1, 0.5**	0, 1.5	**1, 1**

From Tables 1 and 2, fine-tuning a smaller model makes it possible to use it for the ReAct paradigm and deploy it on-site locally. Fine-tuning also helps it strengthen its abilities for the languages it was trained unequally on, indicated by the differences in the success rates of generic questions in English, compared to the prompts in German. These metrics also highlight the issue of the language model being undertrained for certain languages. If the datasets could be translated and the custom datasets added, it could lead to better training of the language models.

Therefore, we suggest our framework called Magenta with these goals in mind, by extending the program created for automatic evaluation of the language models.

CONCLUSION AND FUTURE WORK

Evaluation of agents and LLMs is an important step to bring these agents into the production world. This helps to make sure, that the results are reproducible and scientific. LLMs form the backbone of the agents, therefore making it necessary to choose those models that can follow the prompts and have the target domain knowledge where the agent is being deployed. With a use-case of an agent for the university, we show that the fine-tuning allows the model to follow the prompts and deploy a smaller fine-tuned model locally. We also highlight that the language models are under-trained for other languages. Continuing with evaluation, we built a tool from scratch, to evaluate the agents and then proposed our framework Magenta to integrate all other datasets for evaluation, add custom datasets, and build a unified framework for evaluating LLMs and agents in a single framework. As a work in progress, we are integrating other publicly available datasets and introducing automatic translation for the datasets, by defining them in the task file. As a future work, we propose Magenta++, a framework that defines these tasks programmatically and then constrains the generated output, to check the logical deductions made by the LLM.

REFERENCES

Dettmers, T. et al. (2023) 'QLoRA: Efficient Finetuning of Quantized LLMs'. Available at: https://arxiv.org/abs/2305.14314

EleutherAI (2023) 'lm-evaluation-harness'. Available at: https://github.com/EleutherAI/lm-evaluation-harness

Fourrier, C., Habib, N., Launay, J., & Wolf, T. (2023) 'What's going on with the Open LLM Leaderboard?'. Available at: https://huggingface.co/blog/evaluating-mmlu-leaderboard

Hendrycks, D. et al. (2020) 'Measuring Massive Multitask Language Understanding'. Available at: http://arxiv.org/abs/2009.03300

Kaddour, J. et al. (2023) Challenges and Applications of Large Language Models.

LangChain, (2023). Available: https://www.langchain.com/

Liang, P. et al. (2022) 'Holistic Evaluation of Language Models'. Available at: http://arxiv.org/abs/2211.09110.

Liu, X. et al. (2023) 'AgentBench: Evaluating LLMs as Agents'. Available at: https://arxiv.org/abs/2308.03688

Osika, A. (2023) 'gpt-engineer'. Available at: https://github.com/AntonOsika/gpt-engineer

Nakajima, Y. (2023) 'babyagi'. Available at: https://github.com/yoheinakajima/babyagi.

Qin, Y. et al (2023) 'ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. Available at: https://arxiv.org/abs/2307.16789

Shen, Y. et al. (2023) 'HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in Hugging Face'. Available at: http://arxiv.org/abs/2303.17580

Tatsu-Lab (2023) 'alpaca_eval'. Available at: https://github.com/tatsu-lab/alpaca_eval

Together AI (2023) 'Together AI'. Available at: https://together.ai/

Yang, H., Yue, S. and He, Y. (2023) 'Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions'. Available at: http://arxiv.org/abs/2306.02224

Yao, S. et al. (2022) 'ReAct: Synergizing Reasoning and Acting in Language Models'. Available at: http://arxiv.org/abs/2210.03629

Yao, S. et al. (2023) 'Tree of Thoughts: Deliberate Problem Solving with Large Language Models'. Available at: http://arxiv.org/abs/2305.10601

Zhang, Z. et al. (2023) 'Multimodal Chain-of-Thought Reasoning in Language Models'. Available at: http://arxiv.org/abs/2302.00923

Madaan, A., et al. (2023) 'Self-refine: Iterative refinement with self-feedback'. Available at: https://arxiv.org/abs/2303.17651