UNIVERSITÄT ZU LÜBECK

Using AutoGPT for Information Retrieval Agents

Nutzung von AutoGPT für Informations-Recherche Agenten

Masterarbeit

verfasst am
Institut für Informationssysteme

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
Jakob Horbank

ausgegeben und betreut von
Prof. Dr. Ralf Möller

Lübeck, den 15. April 2024

IM FOCUS DAS LEBEN

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Jakob Horbank

Zusammenfassung

Es geht darum, der Welt »Hallo« zu sagen.

Abstract

It is about saying “hello” to the world.

Contents

1

Introduction

This is an introduction like not other. Information retrieval hard and stuff. Would be nice to have a chatbot that answer natural language questions and gives sources from research database and even web.

1.1 Contributions of this Thesis

Hopefully the above.

1.2 Related Work

There are different attempts at creating LLM outputs with sources. Perplexity AI hosts a question answering service, that gives source from websites.

Open Source LLM Agents

- AutoGPT
- babyagi

Information Retrieval Applications

In a variety of application contexts, the answers of an assistent have to be correct. This is espciacally true in reasearch contexts. A model that hallucinates isn't feasable in this case. But while hallucination can be reducing with fine-tuning, it can not be competely eliminated. Perplexity AI is an online service that leverages a language model to provide a search that generates an answer from different sources in the internet. The content of the answer is then linked to the found sources, so the user can see and verify the result.

As this is a propierty closed source product accessible through a web interface, this is not a useful to create our own research assistant. The provided API simply hosts different LLMs and allows for prompting them. There is no possibilty of of fine-tuning.

1.3 Structure of this Thesis

I do this then that and then that.

2

Analysis of AutoGPT for Information Retrieval Tasks

AutoGPT is an open source project that tries to 'make GPT fully autonomous'. It started out as collection of loose scripts but quickly gained alot of interest in the open source community and grew into a much bigger and now funded project. Initally it only contained the AutoGPT agent that I will anaylse in this chapter but over time has seen additions such as a agent framework called *forge*, a benchmarking system that was put in place to host an agent hackathon as well as branching out of the project to create the *agent protocol*. The agent protocol is an attempt of the community to standardize agent applications.

The AutoGPT was not built with a focus on information retrieval, which is why I will analyse the information retrieval capabilites in this chapter.

GPT langauge models are used to control an agent that works towards reaching a stated goal. The project contains a core general purpose agent with a predefined set of abilities. Addidtionaly a baseline sdk is beeing developed to build custom agents.

2.1 Agent Backgrounds

The concept of agents in computer science is not new. An agent is a system that acts towards reaching a goal in an environment. Agents can be implemented as software or as physical robots or even humans. In the same way different envirmonments are possible such as the real physical world, a web browser or a simulation. The agent needs ways to sense its environment, which can be done by sensors in a physical environment or programatically in a software enivrnment. A task has to be specified for to agent so it knows what his goal is. It can then employ different strategies to reach that goal. These strategies consist of planning steps to execute. While acting out these steps, the environment will probably change as time passes, so an agent needs to reevaluate its plan and the contained steps.

2.2 Architecture Overview

The AutoGPT agent is modeled after the classic agent architecture. After the start the user is asked to enter a task the AutoGPT agent should perform. Then the agent enters a loop of prompting the LLM, executing the proposed action, handling the result of the action and updating the agent state.

The LLM is prompted in a structured way. A base prompt template is defined and populated with current information before each prompting step. The information includes the task at hand, a list of possible actions, a history of previous actions and their results and some extra statements that are there to guide the language model. As the answer needs to be parsed, the system prompt defines a fixed format the LLM should answer in. The answer consists of the thoughts and the proposed next action.

AutoGPT is divided into four modules. The *brain* is the main module that controls the agent. In AutoGPT this is realized by prompting the language model in a structured way. Using the chat system prompt, the language model is prompted to answer a structred format. Different techniques are implemented in this structure. The language model is forced not only to plan the next step, but also to explain the choice for the chosen step and to add self-criticism. An extra output for the human user is also returned. The second part of the answer is the actual next action with the needed arguments. The action make up the second module of the agent. In this module the abilities of the agent are defined. These can be file operations, database queries or web search functionalities. The third module is the *memory.* Memories are modeled after humans which have short and long-term memory. Short term memory can be implemented as an in-memory list messages to the language model. Long-term memory needs a persistent store such as a database. A popular option for language models are vector databases that work with embeddings.

Currently, AutoGPT is only compatible with OpenAI models. Switching to a different model is not that hard, because only the API format would have to be changed. The challenge is to switch between different prompting styles, as every model needs to be prompted differently. For example OpenAI models benefit from profile sentences like "You are an expert in computer science", while Anthropics Claude does not...

2.3 Information Retrieval Capabilities

The AutoGPT Agent has different abilities that can be utilized for information retrieval. Notably, it is able to search the web and operate on the file system.

The web search is implemented by a two-step process. First a search API like DuckDuckGo is called to get a list of relevant pages. Then the page contents are scraped with a headless browser. It is possible to read and write to text files. Other document types are processed by basic text extraction tools to get the plain text.

For longer files such as scientific journals the extracted text is too long for the language model. The AutoGPT agent has no ability to chunk the text into smaller chunk or store it in a database. This is a limitation that needs to be addressed for information retrieval tasks over a research database repository.

Having a vector database would enable techniques such as retrieval augmented generation. The agent would get a prompt which a question over the RDR and choose an action to start a semantic search over the vector database. The result of the search are the chunks that are semantically closest to the question. These chunks can then be included as context for the LLM prompt to generate an answer.

The default agent has a tendency towards searching the web for information. We want an agent that prioritizes information that is present in the research repository. This needs to be addressed in the prompting techniques of the agent.

3

Retrieval Augmented Generation Agent

Large language models can embed a large amount of knowledge into the weights during the pre-training phase. It is possible to generate good answer for different kinds of prompts. ”write about what are good prompts for llms”. But when it comes to specific domain knowledge, the exactness of LLMs start to degrade.

When trying to prompt models about very specific topics common problems such as hallucationations start to show. Not only does the model generate wrong information, it does so with strong confidence. A promising method to make the model answer based on specific sources is called retrieval augmented generaton(RAG).

The RAG method combines an information store with in-context learning. A corpus of documents is stored in a database. When the user prompts the system the database is queried with that prompt matching documents are returned. These documents are then included in the prompt to the LLM as context.

To store the information a vector database is used. A vector database uses embedding models to embed each document into a high dimensional vectorspace. After a user query, the query string gets embedded by the same model, then a metric such as the cosine similarty is used to find semantically close documents. The top-k documents are then returned.

– what is a vectorestore compared to other databases
– how are documents stored and queried
– how is the information included in the prompt

3.1 Methods To Improve LLM Generation

Different approaches try to embed information sources into the generated text. One approach is to fine-tune the language model on a dataset that contains the information. The creation of fine-tuning data is an expensive task, as data needs to be gathered, checked an clean up to produce good results in training. There is some emerging work on generating synthetic datasets by using large language models as data augmentation tools. But even if the the fine-tuning data quality is sufficient, it is still a challenge to make a LLM an expert for a specific domain. Rather than learning new knowledge, fine-tuning is best suited to guide the model towards a certain answering style. Furthermore, fine-tuning billion parameter models is only possible on expensive hardware and therefore not feasable for on-demand tasks.

Following up on the callenges of fine-tuning researches have searched for ways to optimize prompts to get the best model performance. These methods make use of a phenomon called in-context learning. In-context learning describes the observation that giving a large language model more context surrounding the actual prompt can drastically improve the response quality. [1] found that adding a sentence that suggests step-by-step thinking to solve a problem makes the model better at solving logical questions. For GPT models, [xzy] showed an improvement after giving the language model a hint that it is an expert in a certain topic. The findings in the area of prompting techniques and in-context learning suggest that prompt optimizatons have a high potential of getting the most out of LLMs.

A more promising approach is retrieval augmented generation. Before generating an answer to a prompt, a vector database is searched for relevant information. The prompt then includes the information of the returned documents as context.

- Fine-tuning
- Prompting techniques (Chain-of-thought, in-context learning)
- Retrieval augmented generation

3.2 Retrieval Augmented Generation

Large language models can embed a large amount of knowledge into the weights during the pre-training phase. It is possible to generate good answer for different kinds of prompts. ”write about what are good prompts for llms”. But when it comes to specific domain knowledge, the exactness of LLMs start to degrade.

When trying to prompt models about very specific topics common problems such as hallucationations start to show. Not only does the model generate wrong information, it does so with strong confidence. A promising method to make the model answer based on specific sources is called retrieval augmented generaton(RAG).

The RAG method combines an information store with in-context learning. A corpus of documents is stored in a database. When the user prompts the system the database is queried with that prompt matching documents are returned. These documents are then included in the prompt to the LLM as context.

To store the information a vector database is used. A vector database uses embedding models to embed each document into a high dimensional vectorspace. After a user query, the query string gets embedded by the same model, then a metric such as the cosine similarty is used to find semantically close documents. The top-k documents are then returned.

- what is a vectorestore compared to other databases
- how are documents stored and queried
- how is the information included in the prompt

3.3 Extending AutoGPT with retrieval capailities

- Custom agent Loop
- Custom abilities
- Further guidances for the agent

4

Creating IR Agent Benchmarking Challenges

The evaluation of large language model agents is a difficult task, as evaluating LLMs themselves presents a challenge. There are different approaches to evaluate systems built around language models. Subjective evaluation is based on human feedback. As LLM systems are generally made to serve humans this is an important part of evaluation. On the other hand, quantiative metrics that can be computed are used for objective evaluation. Different metrics are used for different tasks. Another importat method are benchmarks. Benchmarks are a set of tasks or an environment that the agent is to move in.

4.1 Existing IR Agent Benchmarks

As the space of possible agent domains is large, lots of different benchmarks were proposed. Simulation enviroments like

Although lots of benchmarks for LLM applications have been proposed, there are few benchmarks that are designed to test the information retrieval capabilities of an agent. There are some benchmarks that are used to evaluate information retrieval in general and in a diverse domains like ”cite”.

4.2 The AutoGPT Benchmarking System

To evaluate AutoGPT and other agent systems that implment the agent protocol, the AutoGPT project has implemented its own benchmarking system. The system constists of a set of tasks that the agent has to complete. The tasks are designed to test different aspects of the agent, and are divided into different topics. Some tasks depend on the previous succesful completion of other tasks. A task consists of an input prompt and a expected output. The ouput is defined by certain words that should be contained.

– Level based system
– Dependencies

4.3 Custom Benchmarks for local IR over journals

- Retrieval benchmarks

5

Results

5.1 Points of Failiure

5.2 Benchmarking Results

5.3 Subjective Evalution

– Why is subjective evaluation needed
– What aspects of the agent can be evaluated subjectively

6

Conclusion

– What was the premise
– What was tried out
– What worked what did not and why

6.1 Future Work

Bibliography

[1] Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter, b., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 24824–24837. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.