Intelligent agents module

Video presentation transcript (individual assignment)

Guilherme Amorim

Slide 0

Hello, my name is Guilherme, and welcome to my individual assignment for the intelligent agents module.

Slide 1

So just by way of introduction, intelligent agents are systems that are able to sense the environment that they are located in and then take action autonomously to achieve certain goals that they have been pre specified for them. Intelligent Agents are considered a component of artificial intelligence, and they're able to generate potential value in many different ways, including improving efficiency through automation and also fostering innovation by allowing for more complex tasks to be performed and to generate additional value across many use cases, some of those are included here, so in healthcare e commerce and finance, where agent based systems are being used to perform complex tasks to improve personalization in terms of care and eCommerce experience as well, and in managing or processing complex systems such as fraud detection and algorithmic training and finance.

For this assignment, we had a series of instructions or goals that our agents needed to accomplish. My team chose the domain of academic research, where we were tasked with building an agent that could find some results on a website, extract data, and then send that data to an offline location.

Slide 2

So for this system that we've called SAGE, so the Scalable Academic Goal-driven Explorer, we have to see layout where we start with a user that specifies to a graphical user interface, the username, a search prompt and an open AI API key, and then these are passed to a prompt preparation agent, an LLM based agent that simplifies the prompt, passes it to a search agent that does run a series of API calls to database engines, and then, using the information retrieved and a summary that is generated, it passes those to semantic parsing agent that will reason upon that information and shows the most relevant searches for that specific prompt the selected references. So in this case, I've instructed it to select five, but this can be changed in the future. Are passed to a referencing agent that formats those according to a pre-specified schema and then passes it to a validation agent that does a series of validations upon those references.

These are then passed to a, what we can call a writing agent that stores it store those in a database in both a structured and a vectorized form based on the reference summaries it created, and also in a CSV file for the user to access in a more ready format. And then all of those informations are also provided to the user via the graphic user interface.

Then an additional step here is that for after the first time that the agent is run, we can now actually do a bit of context searching upon the database to identify previous references that that might be relevant for this prompt, and then pass those as well to the semantic parsing agent.

Slide 3

So now I want to just do a brief run through of my code, just to show you how it looks like when you put everything together. I will be submitting as part of my assignment this Jupyter notebook that includes all the functions that I've specified, along with some simple unit testing that I've performed, you can see here the for each of those cells, the outputs as we go through just showing how everything was what it's outputting and how it's connecting to the next functions.

So this was my development tool. I then transpose all those functions into SAGE_functions.py and those are downloaded through SAGE_main. So if we then just run this, we can see how it all comes together, and I've included some log, console logging and some debugging as much as possible throughout my code, just to help with troubleshooting and to orientate ourselves in terms of what's doing it each time. So first, obviously checking dependencies and loading all necessary libraries. Langchain Community always takes some time to load. That's the code chunk that always takes the longest. But once that's done, it quickly loads. So you can see here the graphical user interface I've created. You can specify our username, let's say if we multiple users using this at some point, my OpenAI API key, and here we have a prompt I've created. And so I'm asking it to please tell me about five good academic references discussing strengths and weaknesses of using first order logic versus natural language processing approaches when designing intelligent agents, which I thought was very topical for this assignment, so we can ask it to run.

So it's telling us that it finished retrieving my GUI inputs and finished preparing the prompt. Here is the simplified prompt that it created. It then finished searching for all those across the databases that I specify the APIs here we can see, just going to minimize that for a moment, but you can see it's already finished running. Here are all the 15 references that it treaty retrieved, along with the with the abstracts, then in the when we load the semantic parsing agent, the first thing we do is to look for the for the previous context in our database. So we can see here it's found those five references that we wanted to consider as well. Then semantic parsing is performed, and it tells us that selected these five references. So reference number so from the database indexing reference number three, number four, number six, number eight and number seven, then reference agent completes. This is these are the formatted references validation completed as well. So this is the output from the validation. And then it finished shaping the results to the database and attempting and managing to write our CSV file with these individual results. Once this is done, it loads the graphical user interface to show the user what it's managed to find, so we can see all the structured information for all of those five references. I've also included here a disclaimer, just so people remember that this was all processed using AI and that they need to be careful. And then here you have, as we said, the five references, including the summary generated by the LM, just makes it easier to interpret what's been provided, rather than having to read through all the abstract but obviously you can do that afterwards, if you want, and it's stored in the CSV file. And then we can copy the URLs and just actually look for those. And since those were extracted through API calls to database themselves, we can be fairly confident that none of these are a result of hallucination. But it's always said here, it's always good to double check. So this is stage in action, and I hope you've enjoyed seeing my implementation.

Slide 4

Now, having seen my code implementation, I wanted to take you through some of the issues I found during development, and I've tried to troubleshoot those. First, is it related to using LLMs for reference search? So my team had originally specified that we would not be using API calls, and they start just using, well, API calls directly to databases, and just using or asking an LLM to do the reference search for us. I tried to implement this, but I found they would just keep coming up with fake references or making up information about true references. For example, the do wise or URLs, which was just really creating a lot of issues in terms of the accuracy and reliability of my system, as well as the tendency to look for or to provide outdated references that I was not happy with. And unfortunately, I could not really circumvent this, no matter how much prompt engineering I tried to do. And so I tried, I decided to do a massive revamp of the architecture and actually start with structured API calls to well established databases. And so in that way, I could ensure that I started with a good grounding in terms of knowing that the references and the information I found were true and accurate, and then I could provide those to the LLM agents to for reasoning and for prioritization, but knowing that the information that we were working with was true and correct.

Then in terms of referencing, we had originally wanted to do referencing according to a specific formatting, such as the cite them right harvest style, but I found that these are complicated formats to implement, because they have a number of different rules depending on the type of source they're working with. And implementing these would be rather challenging, also because we would ideally want users to be able to specify different formats. And so for this implementation, I've just decided to do a simple and versatile scheme that would be able to capture the information, the information the most interest for the most common reference types, and then keep that as an idea for subsequent improvement.

So in terms of validation, this was also challenging for me because I was trying to implement some rule based logic to validate the reference information. But besides some simple checks in terms of data structures and you and checking for valid URLs, it's difficult when to be sure that, for example, an authors or page list for reference is incorrect. You'd need to do some more sophisticated checking for that which I was not able to implement here, and ideally, if there were issues with references, we want the user to be notified of those and potentially ask to correct the references themselves. But this was very complex to implement, so I've just decided to go with some simple checks as specified, and then asking an LLM based agent to do some cross checking in terms of the formatting, just to add a second layer of verification in terms of working with the LLMs.

Another challenge was making sure that was able to format the outputs correctly, because obviously, they are always spit out as strings, and so there's a lot of formatting that needs to go into ensuring that those are formatted correctly as Python lists or dictionaries, then further or deeper down into my implementation.

One of the things I've struggled with was actually ensuring that I was not saving duplicated references into my database. So for this, I would need to have a way of identifying duplicates for performance in terms of efficiency and storage and computing. But I was not able to, although I could do this with the DOIs, for example, I was not entirely sure that I could completely trust the DOI that I was finding, especially, for some reason, some DOIs were not found. So I have not implemented this, and I've kept it as an idea for further improvement.

And similarly, for context search, I think that this was a possibly a helpful feature, but it might not be entirely helpful depending on the specific use case of a particular user. So I thought it would be helpful to, at some point, allow the user to turn this off if they only wanted to consider new references found by the API calls.

And so then just another mention, just to say a lot of issues in terms during development, in terms of setting up the API connections, correctly managing all the library dependencies and the constant updates, and navigating the complicated line chain documentation.

Slide 6

I wanted to also leave some critical reflections on my work. I think there are numbers of number of strengths in my implementation, so the fact that you can interact with the tool using natural language is a plus. And at the same time, I've used API calls to a number of databases that allows to get good quality and structured reference data that can overcome the issues with hallucination by the LLMs and then provide a solid foundation for parsing by the LLM. Those database calls are flexible and scalable, and we can add other databases as needed. The NLP based agents allow for very easy prioritization and flexible formatting of the references with simple natural language instructions. I've implemented a simple set of data validation rules, but which are flexible, and that allows to just reduce some of the most dramatic issues that one can have with the data. And I've implemented the reference summaries provided by the LLMs, which simplify inspection by the user once they have the references selected. So here I've retrieved public data only, so there are really no privacy or ethical concerns in terms of the data we're saving, so that's a plus. And in terms of the data storage, I've employed a local data storage facility, which obviously very low cost and allows for a high degree of control by the user. And I've implemented a modular strategy in my code that allows for, as we've seen, easy debugging and updating as needed.

However, some weaknesses, obviously, so my graphic interface is very archaic. Obviously, there are a number of dependencies that my code runs on which might create issues on the line. The database scope is flexible, but it's still limited currently, and really the in terms of the validation rules implemented, these are really sub optimal, and there is limited capacity within this current implementation to handle any issues with the references found and to resolve those issues. And so my implementation relies a lot on NLP based agents, and these may become problematic so that they have trouble with handling ambiguous or poorly defined prompt by the user. There are biases in the LLMs that might lead to spurious summaries or results presented. And as we discussed, the possibility of hallucination is always present. My code has very poor scalability and real time processing capacity, so there's here trade off in terms of simplicity and scalability and efficiency in my implementation. And similarly, the vector database indexing is performed in memory, and it's a very simple approach that I've used in my system. And also there is, in terms of the code itself, limited error handling and debugging features and unit testing performed.

Slide 6

So finally, this is a non-exhaustive list of possible improvements that I've identified for my implementation, some of these I've already mentioned previously, but I just wanted to add a few more. So I think it would be important to allow the LLMs to help the user disambiguate its prompts by providing additional prompt with queries to ask for additional information, and allowing the user to

specify which agent they want and to also customize the prompt instructions that have been provided by the developer, in this case, myself. Also, allowing the user to verify the references that have been retrieved before they are stored in the database, because some of those might not be relevant or not might be correctly formatted and not being not have been identified by the agents. And then finally, there's a few things that could and should be done in terms of improving API handling efficiency and error management in terms of my implementation.

Slide 7

I just wanted to end with some conclusions and reflections upon my learning. So I think I can say that it developed a successfully a multi agent system that can successfully achieve all the aims that have been specified for this assignment. So it's able to find results on a few websites based on search terms that extracts and processes that data and stores it in a few offline locations. And so in doing so, I was able to really explore a lot of crucial, very important computer science concepts, which were all new to me, and so there was a lot of learning to be taken from that. And I really enjoyed it. I've experienced really firsthand the advantages and pitfalls of different ways of building agent-based architectures, and I've was able to merge those and implement a bit of the best of both worlds in some aspects into my code. And also, I was able to critically identify a number of both the strengths but also the weaknesses and areas for potential improvement in my code, and I hope that I specified those clearly in my presentation. And then finally, just to say, I was able to write my first computer program, which I'm very proud of, even though it has a lot of limitations, but I have to say I have a little help from a number of LLMs in doing this work, so thanks for listening. You.