HIGH prep

DevRev

AI AGENT 007: TOOLING UP FOR SUCCESS

INTER IIT TECH MEET MADRAS

AI Agent 007: Tooling up for Success

In the specialized domain of domain-specific question answering, AI Agent 007 - armed with a Language Model and a toolbox of specific tools, strives to adeptly address user queries. The plan hinges on a meticulous analysis of the domain-specific query, selecting tools that best match the unique context, specifying precise tool arguments, and sequencing tool execution intelligently. This context-aware approach is vital for providing accurate and relevant answers within specialized domains, enhancing user satisfaction, and potentially yielding substantial financial returns. Just as the automated bot message assists in general customer support, AI Agent 007's strategy is tailored to excel in the intricacies of domain-specific question answering.

Introduction

DevRev is dedicated to bridging the gap between developers and their customers (or "Revs") by developing intelligent systems that can handle the routine tasks associated with a product's lifecycle and reduce the need for manual intervention.

One of DevRev's ongoing projects is to create a user-friendly interface for an AI system capable of answering queries and performing operations on customer data. The goal is to enable customers to interact with DevRev using natural language. This involves understanding user queries, mapping them to a combination of existing capabilities within our AI agent, determining the necessary arguments, and executing the tasks to provide a final output.

Problem Statement

Task(s)

A Language model L has a set of tools T, and a user query Q is given. To answer query Q, we need to use existing tools. You need to output the subset of tools to be used to answer the query, the arguments that these tools should be called with, and how to compose the tools to answer the query. The query is conversational, like so

```
Unset
  user_message: "Hello!"
  agent_message: "Hello, how can I help you today?"
  user_message: "Can you tell me my P0 issues?",
  agent_message: "Sure, here is the list... ",
  user_message: "Okay, can you change this list to show only those that are
in triage stage?",
  agent_message: "Sure, this is the updated list ... ",
```

Note:
The set of tools T is dynamic, and new tools can be added, and existing ones could be modified or removed and the agent needs to be able to handle it gracefully.

To begin with, you have following set of tools available to answer the query:

Tool	Description	ArgumentName	Argument Description	ArgumentType	ArgumentValue Example
works_list	Returns a list of work items matching the request	applies_to_part	Filters for work belonging to any of the provided parts	array of strings	Example: ["FEAT-123", "ENH-123", "PROD-123", "CAPL-123"]
		created_by	Filters for work created by any of these users	array of strings	Example: ["DEVU-123"]
		issue.priority	Filters for issues with any of the provided priorities. Allowed values: p0, p1, p2, p3	array of strings	
		issue.rev_orgs	Filters for issues with any of the provided Rev organizations	array of strings	Example: ["REV-123"]
		limit	The maximum number of works to return. The default is '50'	integer (int32)	
		owned_by	Filters for work owned by any of these users	array of strings	Example: ["DEVU-123"]
		stage.name	Filters for records in the provided stage(s) by name	array of strings	
		ticket.needs_response	Filters for tickets that need a response	boolean	
		ticket.rev_org	Filters for tickets associated with any of the provided Rev	array of strings	Example: ["REV-123"]

			organizations		
		ticket.severity	Filters for tickets with any of the provided severities. Allowed values: blocker, high, low, medium	array of strings	
		ticket.source_channel	Filters for tickets with any of the provided source channels	array of strings	
		type	Filters for work of the provided types. Allowed values: issue, ticket, task	array of strings	
summarize_objects	Summarizes a list of objects. The logic of how to summarize a particular object type is an internal implementation detail.	objects	List of objects to summarize	array of objects	
prioritize_objects	Returns a list of objects sorted by priority. The logic of what constitutes priority for a given object is an internal implementation detail.	objects	A list of objects to be prioritized	array of objects	

add_work_items_to_sprint	Adds the given work items to the sprint	work_ids	A list of work item IDs to be added to the sprint.	array of strings	
		sprint_id	The ID of the sprint to which the work items should be added	str	
get_sprint_id	Returns the ID of the current sprint				
get_similar_work_items	Returns a list of work items that are similar to the given work item	work_id	The ID of the work item for which you want to find similar items	string	
search_object_by_name	Given a search string, returns the id of a matching object in the system of record. If multiple matches are found, it returns the one where the confidence is highest.	query	The search string, could be for example customer's name, part name, user name.	string	
create_actionable_tasks_from_text	Given a text, extracts actionable	text	The text from which the actionable	string	

	insights, and creates tasks for them, which are kind of a work item.		insights need to be created.		
who_am_i	Returns the string ID of the current user				

The table below is a sample from the dataset containing queries and the tools to use in order to answer the query. The **output** should be a list of **JSONs** conforming following jsonschema:

```
Unset
{
 "type": "array",
 "items": {
  "type": "object",
  "properties": {
   "tool_name": { "type": "string" },
   "arguments": {
    "type": "array",
    "items": {
     "type": "object",
     "properties": {
      "argument_name": { "type": "string" },
      "argument_value": { "type": "string" }
     },
     "required": ["argument_name", "argument_value"]
    }
   }
  },
  "required": ["tool_name", "arguments"]
 }
}
```

To reference the value of the `ith` tool in the chain, use `$$PREV[i]` as argument value. `i = 0, 1, .. j-1; j = current tool's index in the array`
If the query could not be answered with the given set of tools, output an empty list instead.

Query	Output

Summarize work items similar to don:core:dvrv-us-1:devo/0:issue/1	Unset
	[
	{
	"tool_name": "get_similar_work_items",
	"arguments": [
	{
	"argument_name": "work_id",
	"argument_value": "don:core:dvrv-us-1:devo/0:issue/1"
	}
	]
	},
	{
	"tool_name": "summarize_objects",
	"arguments": [
	{
	"argument_name": "objects",
	"argument_value": "$$PREV[0]"
	}
	]
	}
	]

What is the meaning of life?	Unset []
Prioritize my P0 issues and add them to the current sprint	Unset [{ "tool_name": "whoami", "arguments": [] }, { "tool_name": "works_list", "arguments": [{ "argument_name": "issue.priority",

	"argument_value": ["p0"]
	},
	{
	"argument_name": "owned_by",
	"argument_value": ["$$PREV[0]"]
	},
	{
	"argument_name": "type",
	"argument_value": ["issue"]
	}
	]
	},
	{
	"tool_name": "prioritize_objects",
	"arguments": [
	{
	"argument_name": "objects",
	"argument_value": "$$PREV[1]"
	}
	]
	},
	{
	"tool_name": "get_sprint_id",
	"arguments": []
	},
	{
	"tool_name": "add_work_items_to_sprint",
	"arguments": [
	{
	"argument_name": "work_ids",
	"argument_value": "$$PREV[2]"
	},
	{
	"argument_name": "sprint_id",
	"argument_value": "$$PREV[3]"
	}
	]
	}
	]

Summarize high severity tickets from the customer `UltimateCustomer`	Unset [{ "tool_name": "search_object_by_name", "arguments": [

<table>
<tr>
<td></td>
<td>

```
      {
        "argument_name": "query",
        "argument_value": "UltimateCustomer"
      }
    ]
  },
  {
    "tool_name": "works_list",
    "arguments": [
      {
        "argument_name": "ticket.rev_org",
        "argument_value": ["$$PREV[0]"]
      },
      {
        "argument_name": "ticket.severity",
        "argument_value": ["high"]
      },
      {
        "argument_name": "type",
        "argument_value": ["ticket"]
      }
    ]
  },
  {
    "tool_name": "summarize_objects",
    "arguments": [
      {
        "argument_name": "objects",
        "argument_value": "$$PREV[1]"
      }
    ]
  }
]
```

</td>
</tr>
<tr>
<td>What are my all issues in the triage stage under part FEAT-123? Summarize them.</td>
<td>

Unset

```
[
  {
    "tool_name": "whoami",
    "arguments": []
  },
  {
    "tool_name": "works_list",
    "arguments": [
      {
        "argument_name": "stage.name",
```

</td>
</tr>
</table>

	`"argument_value": ["triage"] }, { "argument_name": "applies_to_part", "argument_value": ["FEAT-123"] }, { "argument_name": "owned_by", "argument_value": ["$$PREV[0]"] }, { "argument_name": "type", "argument_value": ["issue"] }] }, { "tool_name": "summarize_objects", "arguments": [{ "argument_name": "objects", "argument_value": "$$PREV[1]" }] }]`
List all high severity tickets coming in from slack from customer `Cust123` and generate a summary of them.	Unset `[ { "tool_name": "search_object_by_name", "arguments": [ { "argument_name": "query", "argument_value": "Cust123" } ] }, { "tool_name": "works_list", "arguments": [ { "argument_name": "ticket.rev_org", "argument_value": ["$$PREV[0]"] },`

<table>
<tr>
<td></td>
<td>

```
      {
        "argument_name": "ticket.severity",
        "argument_value": ["high"]
      },
      {
        "argument_name": "ticket.source_channel",
        "argument_value": ["slack"]
      },
      {
        "argument_name": "type",
        "argument_value": ["ticket"]
      }
    ]
  },
  {
    "tool_name": "summarize_objects",
    "arguments": [
      {
        "argument_name": "objects",
        "argument_value": "$$PREV[1]"
      }
    ]
  }
]
```

</td>
</tr>
<tr>
<td>Given a customer meeting transcript T, create action items and add them to my current sprint</td>
<td>

Unset

```
[
  {
    "tool_name": "create_actionable_tasks_from_text",
    "arguments": [
      {
        "argument_name": "text",
        "argument_value": "T"
      }
    ]
  },
  {
    "tool_name": "get_sprint_id",
    "arguments": []
  },
  {
    "tool_name": "add_work_items_to_sprint",
    "arguments": [
      {
        "argument_name": "work_ids",
```

</td>
</tr>
</table>

	`"argument_value": "$$PREV[0]" }, { "argument_name": "sprint_id", "argument_value": "$$PREV[1]" } ] } ]`
Get all work items similar to TKT-123, summarize them, create issues from that summary, and prioritize them	Unset `[{ "tool_name": "get_similar_work_items", "arguments": [{ "argument_name": "work_id", "argument_value": "TKT-123" }] }, { "tool_name": "summarize_objects", "arguments": [{ "argument_name": "objects", "argument_value": "$$PREV[0]" }] }, { "tool_name": "create_actionable_tasks_from_text", "arguments": [{ "argument_name": "text", "argument_value": "$$PREV[1]" }] }, { "tool_name": "prioritize_objects", "arguments": [{ "argument_name": "objects", "argument_value": "$$PREV[2]"`

	<pre>}
	]
	}
	]</pre>

Extra/Bonus

- Not every user query could be potentially solved by taking composition of available functions, and might need some additional logic around combining the outputs of those functions, like mathematical operations, iterations, conditional logic etc.
- You would get bonus points if your solution can handle those cases/scenarios, rather than just being able to output the asked list of JSONs.

Testing

The provided code **should compile**, and you need to provide clear instructions on how to execute it. The **output** of the code given the query should be the **JSON with schema** defined above under Problem Statement section.

NOTE: It should also be easy to add a new tool, and you need to provide clear instructions for that, and your code should be able to handle the addition of new tools.

Deliverables

- Code
- Report on the experiments and research done to implement the solution.

Report

Apart from the above mentioned deliverables, you would also be required to submit the mid-term and end-term report which should **necessarily** include the following:

- Literature review
- Different techniques evaluated
- Final technique being used, latency metric corresponding to it
- Future work
- References

The tentative date for mid-term evaluation is **25th November 2023**. The final submissions should come in before 12th December 2023 11:59PM.

Scoring

Final score would comprise of following components:

- Midterm report (research + experiments) - 20%
- Code 40%
    - Extensibility
    - Correctness
    - Performant = cost (tokens/$) + time
- End-term report (research + experiments) (40%)

References

- On the Tool Manipulation Capability of Open-source Large Language Models
- ToolLLM (Facilitating large language models to master 16000+ real-world APIs)
- ToolQA (A Dataset for LLM Question Answering with External Tools)
- API-Bank: A Benchmark for Tool-Augmented LLMs
- Gorilla: Large Language Model Connected with Massive APIs
- ToolAlpaca: Generalized Tool Learning for Language Models with 3000 Simulated Cases
- https://github.com/Significant-Gravitas/AutoGPT/
- https://github.com/e2b-dev/awesome-ai-agents