Agent-Based AI Ticket Documentation

Jhonata Miranda da Costa
Universidade Federal de Viçosa - Campus Florestal
jhonata.miranda@ufv.br

1 Findings in project

1.1 The Crew

This project implements three agents—a researcher, an analyst, and a summarizer—as defined in the ticket. Each agent requires a role description, goal, and backstory aligned with the framework’s specifications. Each agent will automatically be assigned a specific and distinct assignment for this project. The role, aim, and backstory must be set for the agent setup. The description, the desired result, and the agent responsible for completing the task must all be set for the task configuration.

The researcher agent is configured with the following prompts:

- Role: <topic> Senior Data Researcher.
- Goal: Search for information on <topic>.
- Backstory: You’re a seasoned researcher with a knack for uncovering the information about <topic>. Known for your ability to find the most relevant information and present it in a clear and concise manner. You will search for information on the internet, collect it and group it together.

where, <topic> is the subject of the report. And for the task:

- Description: Conduct a thorough research about <topic> Make sure you find any relevant information of renowned sources.
- Expected output: A set of relevant information about <topic>.
- Agent: researcher.

The analyst agent is configured with the following prompts:

- Role: <topic> Senior Analyst.
- Goal:Extract the most important information about <topic>.
- Backstory: You’re a meticulous analyst with a keen eye for detail. You’re known for your ability to extract the key information. You will receive information and extract the most relevant topics.

where, <topic> is the subject of the report. And for the task:

- Description: Review the context you got and expand each topic into a full section for a report. Make sure the report is detailed and contains any and all relevant information.
- Expected output: A fully fledged report with the main topics, each with a full section of information. Formatted as markdown without ’```’.
- Agent: analyst.

The summarizer agent is configured with the following prompts:

- Role: <topic> Senior Summarizer.
- Goal: Generate a summary with the most important information about <topic>.
- Backstory: You’re a report writer who organizes incoming information well, creates concise texts and translates from technical terms to more common ones. You’ll write reports in easy-to-understand language.

where, <topic> is the subject of the report. And for the task:

- Description: Using the information received, write a report, translating very technical terms into more common language. Make sure the report is detailed and contains any and all relevant information.
- Expected output: A report separated into the most important topics written in web page format, with text and bullet points.
- Agent: summarizer.

The researcher and analyst agents are configured with the Serper Dev API (for search) and a web scraping tool, both integrated into crew.py for data retrieval.

1.2 How to run the project

Add your ‘OPENAI_API_KEY‘, ‘SERPER_API_KEY‘ and ‘MODEL‘ settings to the ‘.env‘ file (create this file in the Agents-Test directory after cloning this repository). To execute this project, first install the AI Crew library if not already available. Anaconda may used to create a conda environment to run this project. Install dependencies via:

```
pip install crewai crewai-tools
```

to install the library, and:

```
pip install mlflow
```

to install mlflow and monitor execution. Make sure you run mlflow in the root directory of the repository to access the artifacts already generated in this experiment. Use one terminal to run the mlflow in localhost with:

```
mlflow server
```

command and other terminal to run the project with:

```
crewai run
```

When the program is running, you will be asked to choose the type of execution you want, just type:

```
sync
```

or

```
async
```

for synchronous or asynchronous execution, respectively.

1.3 Metrics and evaluation

To evaluate the project, the following metrics are used: Response time, Accuracy(divided in Completeness, Correctness and Conciseness), Relevance, Efficiency(divided in number of API Calls, Money Spent and Hardware used) and Scalability. To monitor the scalability, the number of reports generated in each run is: [5,3,1] for each execution strategy.

The main goal is compare the crew generating each quantity of reports synchronously and asynchronously. In the table 1, we have the results of this evaluation.

Accuracy is evaluated using three criteria (Completeness, Correctness, and Conciseness), each scored on a 0–10 scale:

- **Completeness:** Coverage of 11 key topics (e.g., Symptoms, Treatment). Reports missing sections or do not mention the topics are penalized proportionally.
- **Correctness:** Penalizes factual errors (>1% word errors = score 1).
- **Conciseness:** Compares word count to human-written baselines (±10% deviation = score 10).

The criteria of Completeness is:

- Overview
- Symptoms
- Treatment
- Diagnosis
- Sick time
- When to see a doctor
- Causes
- Risks Factors
- Complications
- Prevention
- Transmission

being the last item only applicable to Common Cold and Influenza reports. With 11 metrics for these two type of reports, we apply rule of three to scale the evaluate to the same range.

The Correctness score is generate by:

- >1% of number of words wrong → evaluate = 1
- $\leq$1% of number of words wrong → evaluate = 5
- 0 → evaluate = 10

and, to know the number of words of the AI-generated or the humman-writted reports, I use the word counter of LibreOffice. Human-written reference reports are saved as .odt files in the repository.

Finally, to generate the Conciseness score:

- $\pm > 30\%$ of number of words compared to human-written text → evaluate = 1
- ± between 10% and 30% of number of words compared to human-written text → evaluate = 5
- $\pm < 10\%$ of number of words compared to human-written text → evaluate = 10

With this, we want to know if the number of words in the ai-generated reports is similar to the humman-writted reports.

2 Challenges and Potential Improvements

Using the framework is very simple. By creating a base project, I was able to change the number of agents and the number of reports easily. I just had to make a code to save each of the reports in an organized way. A runtime error occurred during web content extraction, triggered by excessively large or malformed input data that exceeded the model's context window.

I believe we can explore the framework a little more for other tasks, given the large number of tools available. For this specific experiment, improving the prompts used could greatly increase the ratings of the AI-generated reports. Another solution would be to use the option to train the crew, so that better reports are probably generated.

Run	# of reports	Report Folder Name	Total Time(seconds) + ID in mflow(the first four characters)	# of API Calls	Cost($) + Serper credits	# of Tokens	Type
1	5	reports_20250317_104005	306.82+73af	40	$0.04+5	total_tokens=213711, prompt_tokens=198562, cached_prompt_tokens=49152, completion_tokens=15149	Sync
2	5	reports_20250317_105326	327.46+e7fa	41	$0.03+5	total_tokens=173349, prompt_tokens=156373, cached_prompt_tokens=39680, completion_tokens=16976	Sync
3	3	reports_20250322_093232	258.91+896a	30	$0.04+3	total_tokens=336932, prompt_tokens=326738, cached_prompt_tokens=183168, completion_tokens=10194	Sync
4	1	reports_20250322_093754	55.43+fe44	6	$0.01+1	total_tokens=17120, prompt_tokens=13841, cached_prompt_tokens=2432, completion_tokens=3279	Sync
5	5	reports_20250322_094454	84+4772	47	$0.04+5	total_tokens=284820, prompt_tokens=267762, cached_prompt_tokens=137856, completion_tokens=17058	Async
6	3	reports_20250322_094906	66+e3bd	24	$0.01+3	total_tokens=93348, prompt_tokens=83915, cached_prompt_tokens=30720, completion_tokens=9433	Async
7	1	reports_20250322_095318	66+e969	9	$0.01+1	total_tokens=40820, prompt_tokens=36729, cached_prompt_tokens=19072, completion_tokens=4091	Async

Table 1: Experimental Reports Table