perplexity

Make a detailed analysis of the recently provided Jupyter Notebook "L7_job_application_crew.ipynb", broken down into the sections I requested?

Here is a detailed analysis of the provided Jupyter Notebook, broken down into the sections you requested:

Cell-by-Cell Analysis

The Jupyter Notebook "L7_job_application_crew.ipynb" demonstrates how to build a multi-agent system using the `crewAI` framework to tailor job applications.

1. **Markdown Cells (Introduction and Setup)**:
   - The initial markdown cells introduce the lesson's objective: to build a multi-agent system.
   - It specifies the installation command for the required libraries if running the notebook locally: `!pip install crewai==0.28.8 crewai_tools==0.1.6 langchain_community==0.0.29`[1] [2].
2. **Code Cell 1: Warning Control**:
   - This cell imports the `warnings` module and uses `warnings.filterwarnings('ignore')` to suppress warning messages during notebook execution, ensuring a clean output[1] [2].

```
import warnings
warnings.filterwarnings('ignore')
```

3. **Markdown Cell: Import Libraries, APIs, and LLM**:
   - This cell indicates that the next step is to import libraries, APIs, and the LLM[1] [2].
4. **Code Cell 2: Importing Libraries**:
   - This cell imports the necessary classes (`Agent`, `Task`, `Crew`) from the `crewai` library[1] [2].

```
from crewai import Agent, Task, Crew
```

5. **Markdown Cell: Note about LLM**:
   - This cell specifies that while the video uses `gpt-4-turbo`, the code will run on `gpt-3.5-turbo` due to certain constraints[1] [2]. It notes that `gpt-4-turbo` can be used when running the notebook locally but will not work on the platform[1] [2].

6. **Code Cell 3: OpenAI API Key and Model Setup**:

```
import os
from utils import get_openai_api_key, get_serper_api_key

openai_api_key = get_openai_api_key()
os.environ["OPENAI_MODEL_NAME"] = 'gpt-3.5-turbo'
os.environ["SERPER_API_KEY"] = get_serper_api_key()
```

   - This cell imports the `os` module and functions (`get_openai_api_key`, `get_serper_api_key`) from a local `utils.py` file (not included in the provided content)[1] [2]. It retrieves the OpenAI and Serper API keys and sets the `OPENAI_MODEL_NAME` environment variable to `gpt-3.5-turbo` and the `SERPER_API_KEY`[1] [2].
   - **Note**: The notebook mentions that `gpt-4-turbo` will not work on the platform[1] [2].
7. **Markdown Cell: crewAI Tools**:
   - This cell introduces `crewAI` Tools[1] [2].
8. **Code Cell 4: Importing Tools**:

```
from crewai_tools import (
 FileReadTool,
 ScrapeWebsiteTool,
 MDXSearchTool,
 SerperDevTool
)

search_tool = SerperDevTool()
scrape_tool = ScrapeWebsiteTool()
read_resume = FileReadTool(file_path='./fake_resume.md')
semantic_search_resume = MDXSearchTool(mdx='./fake_resume.md')
```

   - This cell imports `FileReadTool`, `ScrapeWebsiteTool`, `MDXSearchTool`, and `SerperDevTool` from the `crewai_tools` library[1] [2]. It also initializes the tools[1] [2].
9. **Markdown Cell: Uncomment and Run the Cell Below...**:
   - This cell includes a note stating, "Uncomment and run the cell below if you wish to view `fake_resume.md` in the notebook"[1] [2].
10. **Code Cell 5: Display fake_resume.md**:

```
# from IPython.display import Markdown, display
# display(Markdown("./fake_resume.md"))
```

    - This cell contains commented-out code to display `fake_resume.md` using `IPython.display.Markdown`[1] [2].
11. **Markdown Cell: Creating Agents**:
    - This cell introduces the next step: creating agents[1] [2].
12. **Code Cell 6: Agent Creation (Researcher)**:

```
# Agent 1: Researcher
researcher = Agent(
 role="Tech Job Researcher",
 goal="Make sure to do amazing analysis on "
  "job posting to help job applicants",
 tools = [scrape_tool, search_tool],
 verbose=True,
 backstory=(
  "As a Job Researcher, your prowess in "
  "navigating and extracting critical "
  "information from job postings is unmatched."
  "Your skills help pinpoint the necessary "
  "qualifications and skills sought "
  "by employers, forming the foundation for "
  "effective application tailoring."
 )
)
```

- This cell instantiates an `Agent` named `researcher`[1] [2]. The agent is assigned a `role` ("Tech Job Researcher"), a `goal` (analyzing job postings), and a `backstory`[1] [2]. It is set to `verbose=True` and is assigned `scrape_tool` and `search_tool`[1] [2].

13. **Code Cell 7: Agent Creation (Profiler)**:

```
# Agent 2: Profiler
profiler = Agent(
 role="Personal Profiler for Engineers",
 goal="Do incredible research on job applicants "
  "to help them stand out in the job market",
 tools = [scrape_tool, search_tool,
  read_resume, semantic_search_resume],
 verbose=True,
 backstory=(
  "Equipped with analytical prowess, you dissect "
  "and synthesize information "
  "from diverse sources to craft comprehensive "
  "personal and professional profiles, laying the "
  "groundwork for personalized resume enhancements."
 )
)
```

- This cell creates an `Agent` named `profiler`[1] [2]. The agent's `role` is "Personal Profiler for Engineers"[1] [2]. The assigned tools are `scrape_tool`, `search_tool`, `read_resume`, and `semantic_search_resume`[1] [2].

14. **Code Cell 8: Agent Creation (Resume Strategist)**:

```
# Agent 3: Resume Strategist
resume_strategist = Agent(
 role="Resume Strategist for Engineers",
 goal="Find all the best ways to make a "
  "resume stand out in the job market.",
 tools = [scrape_tool, search_tool,
  read_resume, semantic_search_resume],
```

```
 verbose=True,
 backstory=(
  "With a strategic mind and an eye for detail, you "
  "excel at refining resumes to highlight the most "
  "relevant skills and experiences, ensuring they "
  "resonate perfectly with the job's requirements."
 )
)
```

- This cell creates an `Agent` named `resume_strategist`[1] [2]. The agent's `role` is "Resume Strategist for Engineers"[1] [2]. The assigned tools are `scrape_tool`, `search_tool`, `read_resume`, and `semantic_search_resume`[1] [2].

15. **Code Cell 9: Agent Creation (Interview Preparer)**:

```
# Agent 4: Interview Preparer
interview_preparer = Agent(
 role="Engineering Interview Preparer",
 goal="Create interview questions and talking points "
  "based on the resume and job requirements",
 tools = [scrape_tool, search_tool,
  read_resume, semantic_search_resume],
 verbose=True,
 backstory=(
  "Your role is crucial in anticipating the dynamics of "
  "interviews. With your ability to formulate key questions "
  "and talking points, you prepare candidates for success, "
  "ensuring they can confidently address all aspects of the "
  "job they are applying for."
 )
)
```

- This cell creates an `Agent` named `interview_preparer`[1] [2]. The agent's `role` is "Engineering Interview Preparer"[1] [2]. The assigned tools are `scrape_tool`, `search_tool`, `read_resume`, and `semantic_search_resume`[1] [2].

16. **Markdown Cell: Creating Tasks**:

- This cell introduces the next step: creating tasks[1] [2].

17. **Code Cell 10: Task Creation (Research Task)**:

```
# Task for Researcher Agent: Extract Job Requirements
research_task = Task(
 description=(
  "Analyze the job posting URL provided ({job_posting_url}) "
  "to extract key skills, experiences, and qualifications "
  "required. Use the tools to gather content and identify "
  "and categorize the requirements."
 ),
 expected_output=(
  "A structured list of job requirements, including necessary "
  "skills, qualifications, and experiences."
 ),
 agent=researcher,
```

```
 async_execution=True
)
```

- This cell defines the `research_task` for the `researcher`[1] [2]. It includes a `description`, `expected_output`, sets the `agent` to `researcher`, and sets `async_execution=True`[1] [2].

18. **Code Cell 11: Task Creation (Profile Task)**:

```
# Task for Profiler Agent: Compile Comprehensive Profile
profile_task = Task(
 description=(
  "Compile a detailed personal and professional profile "
  "using the GitHub ({github_url}) URLs, and personal write-up "
  "({personal_writeup}). Utilize tools to extract and "
  "synthesize information from these sources."
 ),
 expected_output=(
  "A comprehensive profile document that includes skills, "
  "project experiences, contributions, interests, and "
  "communication style."
 ),
 agent=profiler,
 async_execution=True
)
```

- This cell defines the `profile_task` for the `profiler`[1] [2]. It includes a `description`, `expected_output`, sets the `agent` to `profiler`, and sets `async_execution=True`[1] [2].

19. **Markdown Cell: Passing List of Tasks as Context**:
    * This cell explains how to pass a list of tasks as `context` to a task, noting that the task will not run until it has the output from those tasks[1] [2].

20. **Code Cell 12: Task Creation (Resume Strategy Task)**:

```
# Task for Resume Strategist Agent: Align Resume with Job Requirements
resume_strategy_task = Task(
 description=(
  "Using the profile and job requirements obtained from "
  "previous tasks, tailor the resume to highlight the most "
  "relevant areas. Employ tools to adjust and enhance the "
  "resume content. Make sure this is the best resume even but "
  "don't make up any information. Update every section, "
  "inlcuding the initial summary, work experience, skills, "
  "and education. All to better reflrect the candidates "
  "abilities and how it matches the job posting."
 ),
 expected_output=(
  "An updated resume that effectively highlights the candidate's "
  "qualifications and experiences relevant to the job."
 ),
 output_file="tailored_resume.md",
 context=[research_task, profile_task],
 agent=resume_strategist
)
```

- This cell defines the `resume_strategy_task` for the `resume_strategist`[1] [2]. It includes a `description`, `expected_output`, sets `output_file="tailored_resume.md"`, sets `context` to `[research_task, profile_task]`, and sets the `agent` to `resume_strategist`[1] [2].

21. **Code Cell 13: Task Creation (Interview Preparation Task)**:

```
# Task for Interview Preparer Agent: Develop Interview Materials
interview_preparation_task = Task(
 description=(
  "Create a set of potential interview questions and talking "
  "points based on the tailored resume and job requirements. "
  "Utilize tools to generate relevant questions and discussion "
  "points. Make sure to use these question and talking points to "
  "help the candiadte highlight the main points of the resume "
  "and how it matches the job posting."
 ),
 expected_output=(
  "A document containing key questions and talking points "
  "that the candidate should prepare for the initial interview."
 ),
 output_file="interview_materials.md",
 context=[research_task, profile_task, resume_strategy_task],
 agent=interview_preparer
)
```

- This cell defines the `interview_preparation_task` for the `interview_preparer`[1] [2]. It includes a `description`, `expected_output`, sets `output_file="interview_materials.md"`, sets `context` to `[research_task, profile_task, resume_strategy_task]`, and sets the `agent` to `interview_preparer`[1] [2].

22. **Markdown Cell: Creating the Crew**:

- This cell introduces the creation of the crew[1] [2].

23. **Code Cell 14: Crew Creation**:

```
job_application_crew = Crew(
 agents=[researcher,
  profiler,
  resume_strategist,
  interview_preparer],

 tasks=[research_task,
  profile_task,
  resume_strategy_task,
  interview_preparation_task],

 verbose=True
)
```

- This cell instantiates the `Crew` object, linking the agents and their tasks[1] [2]. `verbose=True` is set[1] [2].

24. **Markdown Cell: Running the Crew**:

- This cell prepares for running the Crew and explains how to set the inputs for execution[1] [2].

25. **Code Cell 15: Job Application Inputs**:

```
job_application_inputs = {
 'job_posting_url': 'https://jobs.lever.co/AIFund/6c82e23e-d954-4dd8-a734-c0c2c5ee00
 'github_url': 'https://github.com/joaomdmoura',
 'personal_writeup': """Noah is an accomplished Software
 Engineering Leader with 18 years of experience, specializing in
 managing remote and in-office teams, and expert in multiple
 programming languages and frameworks. He holds an MBA and a strong
 background in AI and data science. Noah has successfully led
 major tech initiatives and startups, proving his ability to drive
 innovation and growth in the tech industry. Ideal for leadership
 roles that require a strategic and innovative approach."""
}
```

- This cell defines a dictionary of job application inputs to be used as input for the crew[1] [2].

26. **Markdown Cells: Note**:
    - This cell notes that LLMs can provide different outputs for the same input[1] [2].
27. **Code Cell 16: Running the Crew**:

```
### this execution will take a few minutes to run
result = job_application_crew.kickoff(inputs=job_application_inputs)
```

- This cell executes the crew's workflow using the `crew.kickoff()` method[1] [2]. It passes a dictionary with job application inputs as input[1] [2].

28. **Markdown Cell: Display tailored_resume.md**:
    - This cell prepares to display the generated `tailored_resume.md` file[1] [2].
29. **Code Cell 17: Displaying Tailored Resume**:

```
from IPython.display import Markdown, display
display(Markdown("./tailored_resume.md"))
```

- This cell imports the `Markdown` class from `IPython.display` and displays the `tailored_resume.md` file as markdown output within the notebook[1] [2].

30. **Markdown Cell: Display interview_materials.md**:
    - This cell prepares to display the generated `interview_materials.md` file[1] [2].
31. **Code Cell 18: Displaying Interview Materials**:

```
display(Markdown("./interview_materials.md"))
```

- This cell displays the `interview_materials.md` file as markdown output within the notebook[1] [2].

Technical Explanations

- **Multi-Agent System**: The notebook implements a multi-agent system where a Researcher, Profiler, Resume Strategist, and Interview Preparer collaborate to tailor a job application[1] [2].
- **CrewAI Framework**: The `crewAI` framework is used to create and manage the multi-agent system[1] [2].
- **LLM (Language Model)**: The agents use OpenAI's `gpt-3.5-turbo` as their LLM[1] [2].
- **Tools**: The notebook demonstrates the use of `SerperDevTool` and `ScrapeWebsiteTool` for web searches and scraping, `FileReadTool` for reading local files, and `MDXSearchTool` for semantic search within a document[1] [2].
- **Task Context**: The notebook showcases how to pass the output of previous tasks as context to subsequent tasks, enabling the agents to build upon each other's work[1] [2].
- **Asynchronous Execution**: The `async_execution=True` setting on the initial tasks allows them to run in parallel, improving overall efficiency[1] [2].
- **Output Files**: The `output_file` parameter is used to save the results of the resume strategist and interview preparer tasks to markdown files[1] [2].
- **Computational Complexity**: The computational complexity depends on the LLM used and the complexity of the tasks. LLM inference can be computationally expensive. The use of tools, especially web searches, can also add to the computational cost.

Environment and Dependencies

- **External Libraries**:
  - `crewai==0.28.8`: The core framework for creating multi-agent systems[1] [2].
  - `crewai_tools==0.1.6`: Tools and utilities for `crewAI`, such as `SerperDevTool`, `ScrapeWebsiteTool`, `FileReadTool`, and `MDXSearchTool`[1] [2].
  - `langchain_community==0.0.29`: A library providing components for working with language models[1] [2].
  - `IPython`: Used for displaying Markdown output in Jupyter Notebooks.
- **Environment Variables**:
  - `OPENAI_API_KEY`: The OpenAI API key is required to access the `gpt-3.5-turbo` model[1] [2].
  - `SERPER_API_KEY`: The Serper API key is required to use the `SerperDevTool`[1] [2].
  - `OPENAI_MODEL_NAME`: Specifies the OpenAI model to be used (`gpt-3.5-turbo`)[1] [2].
- **Version Considerations**: The notebook specifies exact versions for the `crewai`, `crewai_tools`, and `langchain_community` libraries[1] [2].

Error Handling and Optimization

- **Error Handling**: The output shows that the `Tech Job Researcher` agent encountered a "404 error" while trying to access the job posting URL[1] [2]. The agent was able to identify that the job posting had been removed or closed[1] [2].
- **Optimization**:
  - **Efficient LLM Usage**: Optimize the prompts and instructions given to the agents to reduce the number of tokens used and the overall cost of LLM inference.
  - **Caching**: Implement caching mechanisms for tool results to avoid redundant computations.

Documentation and Comments

- **Markdown Cells**: The markdown cells provide a high-level overview of the notebook's purpose, explain the steps involved in creating a multi-agent system, and provide context for the code[1] [2].
- **Inline Comments**: The code includes comments to explain specific lines or sections of code.

Overall Summary

The Jupyter Notebook "L7_job_application_crew.ipynb" demonstrates how to use the `crewAI` framework to create a multi-agent system for tailoring job applications[1] [2].

- **Input**: The notebook takes a job posting URL, GitHub URL, and a personal write-up as input[1] [2].
- **Process**: It defines four agents (researcher, profiler, resume strategist, and interview preparer) and assigns them tasks to analyze the job posting, compile a profile, tailor the resume, and prepare interview materials[1] [2]. The agents use OpenAI's `gpt-3.5-turbo` to generate text and various tools to gather information[1] [2]. The tasks are executed by the `crew.kickoff()` method[1] [2].
- **Output**: The notebook outputs a tailored resume and interview materials[1] [2].
- **Edge Cases**: The notebook assumes the OpenAI and Serper API keys are properly configured and that the required libraries are installed. The quality of the generated outputs depends on the capabilities of the LLM and the clarity of the task descriptions.
- **Future Extensions**:
  - **More Robust Error Handling**: Implement more robust error handling to gracefully handle exceptions and provide informative error messages.
  - **Implement Caching**: Add caching to avoid redundant API calls and improve performance.
  - **Allow for User Feedback**: Add a mechanism for users to provide feedback on the generated resume and interview materials, and use this feedback to improve the system.

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/10259221/c39d44f4-9ea7-4dcc-beae-be38042bf37f/L7_job_application_crew.ipynb
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/10259221/c39d44f4-9ea7-4dcc-beae-be38042bf37f/L7_job_application_crew.ipynb