

```
"""
```

```
$ pip install swarms
```

- Add docs into the database
- Use better llm
- use better prompts [System and SOPs]
- Use a open source model like Command R
- Better SOPS ++ System Prompts
- 

```
"""
```

```
from swarms import Agent
```

```
from swarm_models import OpenAIChat
```

```
from swarms_memory import ChromaDB
```

```
from swarms.tools.prebuilt.bing_api import fetch_web_articles_bing_api
```

```
import os
```

```
from dotenv import load_dotenv
```

```
load_dotenv()
```

```
# Let's create a text file with the provided prompt.
```

```
research_system_prompt = """
```

```
Research Agent LLM Prompt: Summarizing Sources and Content
```

```
Objective:
```

Your task is to summarize the provided sources and the content within those sources. The goal is to create concise, accurate, and informative summaries that capture the key points of the original content.

Instructions:

1. Identify Key Information:

- Extract the most important information from each source. Focus on key facts, main ideas, significant arguments, and critical data.

2. Summarize Clearly and Concisely:

- Use clear and straightforward language. Avoid unnecessary details and keep the summary concise.
- Ensure that the summary is coherent and easy to understand.

3. Preserve Original Meaning:

- While summarizing, maintain the original meaning and intent of the content. Do not omit essential information that changes the context or understanding.

4. Include Relevant Details:

- Mention the source title, author, publication date, and any other relevant details that provide context.

5. Structure:

- Begin with a brief introduction to the source.
- Follow with a summary of the main content.

- Conclude with any significant conclusions or implications presented in the source.

```
"""
```

```
# Initialize
```

```
memory = ChromaDB(  
    output_dir="research_base",  
    n_results=2,  
)
```

```
llm = OpenAIChat(  
    temperature=0.2,  
    max_tokens=3500,  
    openai_api_key=os.getenv("OPENAI_API_KEY"),  
)
```

```
# Initialize the agent
```

```
agent = Agent(  
    agent_name="Research Agent",  
    system_prompt=research_system_prompt,  
    llm=llm,  
    max_loops="auto",  
    autosave=True,
```

```
dashboard=False,  
  
interactive=True,  
  
long_term_memory=memory,  
  
# tools=[fetch_web_articles_bing_api],  
  
)
```

```
def perplexity_agent(task: str = None, *args, **kwargs):
```

```
    """
```

This function takes a task as input and uses the Bing API to fetch web articles related to the task. It then combines the task and the fetched articles as prompts and runs them through an agent. The agent generates a response based on the prompts and returns it.

Args:

task (str): The task for which web articles need to be fetched.

Returns:

str: The response generated by the agent.

```
    """
```

```
    out = fetch_web_articles_bing_api(  
  
        task,  
  
        subscription_key=os.getenv("BING_API_KEY"),  
  
    )
```

```
# Sources
```

```
sources = [task, out]
```

```
sources_prompts = "".join(sources)
```

```
# Run a question
```

```
agent_response = agent.run(sources_prompts)
```

```
return agent_response
```

```
out = perplexity_agent(
```

```
    "What are the indian food restaurant names in standford university avenue? What are their cost  
ratios"
```

```
)
```

```
print(out)
```