Swarm Models

[![Join our Discord](https://img.shields.io/badge/Discord-Join%20our%20server-5865F2?style=for-the-badge&logo=discord&logoColor=white)](https://discord.gg/agora-999382051935506503) [![Subscribe on YouTube](https://img.shields.io/badge/YouTube-Subscribe-red?style=for-the-badge&logo=youtube&logoColor=white)](https://www.youtube.com/@kyegomez3242) [![Connect on LinkedIn](https://img.shields.io/badge/LinkedIn-Connect-blue?style=for-the-badge&logo=linkedin&logoColor=white)](https://www.linkedin.com/in/kye-g-38759a207/) [![Follow on X.com](https://img.shields.io/badge/X.com-Follow-1DA1F2?style=for-the-badge&logo=x&logoColor=white)](https://x.com/kyegomezb)

Swarm Models provides a unified, secure, and highly scalable interface for interacting with multiple LLM and multi-modal APIs across different providers. It is built to streamline your API integrations, ensuring production-grade reliability and robust performance.

Key Features:

- **Multi-Provider Support**: Integrate seamlessly with APIs from OpenAI, Anthropic, Azure, and more.

- **Enterprise-Grade Security**: Built-in security protocols to protect your API keys and sensitive data, ensuring compliance with industry standards.

- **Lightning-Fast Performance**: Optimized for low-latency and high-throughput, Swarm Models

delivers blazing-fast API responses, suitable for real-time applications.

- **Ease of Use**: Simplified API interaction with intuitive `.run(task)` and `__call__` methods, making integration effortless.

- **Scalability for All Use Cases**: Whether it's a small script or a massive enterprise-scale application, Swarm Models scales effortlessly.

- **Production-Grade Reliability**: Tested and proven in enterprise environments, ensuring consistent uptime and failover capabilities.

Onboarding

Swarm Models simplifies the way you interact with different APIs by providing a unified interface for all models.

1. Install Swarm Models

```bash
$ pip3 install -U swarm-models
```

2. Set Your Keys

```bash
OPENAI_API_KEY="your_openai_api_key"
GROQ_API_KEY="your_groq_api_key"
ANTHROPIC_API_KEY="your_anthropic_api_key"
AZURE_OPENAI_API_KEY="your_azure_openai_api_key"
```

3. Initialize a Model

Import the desired model from the package and initialize it with your API key or necessary configuration.

```python
from swarm_models import YourDesiredModel

model = YourDesiredModel(api_key='your_api_key', *args, **kwargs)
```

4. Run Your Task

Use the `.run(task)` method or simply call the model like `model(task)` with your task.

```python
task = "Define your task here"
result = model.run(task)
```

Or equivalently

#result = model(task)

```

### **5. Enjoy the Results**

```python

print(result)

```

Full Code Example

```python

from swarm_models import OpenAIChat

import os

# Get the OpenAI API key from the environment variable

api_key = os.getenv("OPENAI_API_KEY")

# Create an instance of the OpenAIChat class

model = OpenAIChat(openai_api_key=api_key, model_name="gpt-4o-mini")

# Query the model with a question

```
out = model(
    "What is the best state to register a business in the US for the least amount of taxes?"
)

# Print the model's response
print(out)
```

`TogetherLLM` Documentation

The `TogetherLLM` class is designed to simplify the interaction with Together's LLM models. It provides a straightforward way to run tasks on these models, including support for concurrent and batch processing.

Initialization

To use `TogetherLLM`, you need to initialize it with your API key, the name of the model you want to use, and optionally, a system prompt. The system prompt is used to provide context to the model for the tasks you will run.

Here's an example of how to initialize `TogetherLLM`:

```python
import os
from swarm_models import TogetherLLM
```

```
model_runner = TogetherLLM(
    api_key=os.environ.get("TOGETHER_API_KEY"),
    model_name="meta-llama/Meta-Llama-3.1-70B-Instruct-Turbo",
    system_prompt="You're Larry fink",
)
```

Running Tasks

Once initialized, you can run tasks on the model using the `run` method. This method takes a task string as an argument and returns the response from the model.

Here's an example of running a single task:

```python
task = "How do we allocate capital efficiently in your opinion Larry?"
response = model_runner.run(task)
print(response)
```

Running Multiple Tasks Concurrently

`TogetherLLM` also supports running multiple tasks concurrently using the `run_concurrently` method. This method takes a list of task strings and returns a list of responses from the model.

Here's an example of running multiple tasks concurrently:

```python
tasks = [
```

```
    "What are the top-performing mutual funds in the last quarter?",
    "How do I evaluate the risk of a mutual fund?",
    "What are the fees associated with investing in a mutual fund?",
    "Can you recommend a mutual fund for a beginner investor?",
    "How do I diversify my portfolio with mutual funds?",
]
responses = model_runner.run_concurrently(tasks)
for response in responses:
    print(response)
```

Enterprise-Grade Features

1. **Security**: API keys and user data are handled with utmost care, utilizing encryption and best security practices to protect your sensitive information.

2. **Production Reliability**: Swarm Models has undergone rigorous testing to ensure that it can handle high traffic and remains resilient in enterprise-grade environments.

3. **Fail-Safe Mechanisms**: Built-in failover handling to ensure uninterrupted service even under heavy load or network issues.

4. **Unified API**: No more dealing with multiple SDKs or libraries. Swarm Models standardizes your interactions across providers like OpenAI, Anthropic, Azure, and more, so you can focus on what matters.

Available Models

Model Name	Import Path
BaseLLM	`from swarm_models.base_llm import BaseLLM`
BaseMultiModalModel	`from swarm_models.base_multimodal_model import BaseMultiModalModel`
GPT4VisionAPI	`from swarm_models.gpt4_vision_api import GPT4VisionAPI`
HuggingfaceLLM	`from swarm_models.huggingface import HuggingfaceLLM`
LayoutLMDocumentQA	`from swarm_models.layoutlm_document_qa import LayoutLMDocumentQA`
llama3Hosted	`from swarm_models.llama3_hosted import llama3Hosted`
LavaMultiModal	`from swarm_models.llava import LavaMultiModal`
Nougat	`from swarm_models.nougat import Nougat`
OpenAIEmbeddings	`from swarm_models.openai_embeddings import OpenAIEmbeddings`
OpenAITTS	`from swarm_models.openai_tts import OpenAITTS`
GooglePalm	`from swarm_models.palm import GooglePalm as Palm`
Anthropic	`from swarm_models.popular_llms import Anthropic as Anthropic`
AzureOpenAI	`from swarm_models.popular_llms import AzureOpenAILLM as AzureOpenAI`
Cohere	`from swarm_models.popular_llms import CohereChat as Cohere`
OctoAIChat	`from swarm_models.popular_llms import OctoAIChat`

OpenAIChat	`from swarm_models.popular_llms import OpenAIChatLLM as OpenAIChat`
OpenAILLM	`from swarm_models.popular_llms import OpenAILLM as OpenAI`
Replicate	`from swarm_models.popular_llms import ReplicateChat as Replicate`
QwenVLMultiModal	`from swarm_models.qwen import QwenVLMultiModal`
FireWorksAI	`from swarm_models.popular_llms import FireWorksAI`
Vilt	`from swarm_models.vilt import Vilt`
TogetherLLM	`from swarm_models.together_llm import TogetherLLM`
LiteLLM	`from swarm_models.lite_llm_model import LiteLLM`
OpenAIFunctionCaller	`from swarm_models.openai_function_caller import OpenAIFunctionCaller`
OllamaModel	`from swarm_models.ollama_model import OllamaModel`
GroundedSAMTwo	`from swarm_models.sam_two import GroundedSAMTwo`

Support & Contributions

- **Documentation**: Comprehensive guides, API references, and best practices are available in our official [Documentation](https://docs.swarms.world).
- **GitHub**: Explore the code, report issues, and contribute to the project via our [GitHub repository](https://github.com/The-Swarm-Corporation/swarm-models).

License

Swarm Models is released under the [MIT License](https://github.com/The-Swarm-Corporation/swarm-models/LICENSE).

Todo

- [ ] Add cohere models command r
- [ ] Add gemini and google ai studio
- [ ] Integrate ollama extensively