assafelovic / **gpt-researcher** Public

Code | Issues 164 | Pull requests 43 | Discussions | Actions | Projects

57 Branches **68** Tags | Go to file | Go to file | Code

assafelovic Modify contributor image link in README

7c32174 · 3 weeks ago

.claude	refactored skills diretory strcu…	2 months ago
.github	removed docker push entirely	3 months ago
backend	Merge pull request #1665 fro…	3 weeks ago
docs	fix docs links and add ag2 pi…	last month
evals	Cleaned up imports.	10 months ago
frontend	Merge pull request #1657 fro…	3 weeks ago
gpt_researcher	Merge pull request #1665 fro…	3 weeks ago
mcp-server	moved mcp to dedicated repo	11 months ago
multi_agents	#1673: Fixed the reference e…	3 weeks ago
multi_agents_ag2	version	last month
terraform	refactor: Implement ECR and…	3 months ago
tests	Add aggregated summary fla…	3 months ago
.cursorignore	feat: Add support for custom …	7 months ago
.cursorrules	Update to Cursor Rules - giti…	last year
.dockerignore	add support markdown downl…	2 years ago
.env.example	perf: optimize context compr…	2 months ago
.gitignore	Update .gitignore	11 months ago
.python-version	Add aggregated summary fla…	3 months ago

CODE_OF_CONDUC…	Updates Code of Conduct an…	2 years ago
CONTRIBUTING.md	Updated Contributing.md	2 years ago
CURSOR_RULES.md	Update to Cursor Rules - giti…	last year
Dockerfile	added image generation with…	3 months ago
Dockerfile.fullstack	Add `nginx` routing	10 months ago
LICENSE	Update LICENSE	2 years ago
Procfile	procfile for heroku	last year
README-ja_JP.md	feat: enable LangSmith tracin…	3 months ago
README-ko_KR.md	feat: enable LangSmith tracin…	3 months ago
README-zh_CN.md	feat: enable LangSmith tracin…	3 months ago
README.md	Modify contributor image link …	3 weeks ago
citation.cff	added citation	2 years ago
cli.py	Add output format control fla…	5 months ago
docker-compose.yml	added image generation with…	3 months ago
json_schema_generat…	fix: Changed agent selection …	10 months ago
langgraph.json	styling and edited langgraph …	2 years ago
main.py	fixed run issue	7 months ago
poetry.toml	Virtual Environ and Poetry S…	3 years ago
pyproject.toml	fix docs links and add ag2 pi…	last month
requirements.txt	added image generation with…	3 months ago
setup.py	updated project version	2 months ago

README Code of conduct Contributing Apache-2.0 license

🔎 GPT Researcher

GPT Researcher is an open deep research agent designed for both web and local research on any given task.

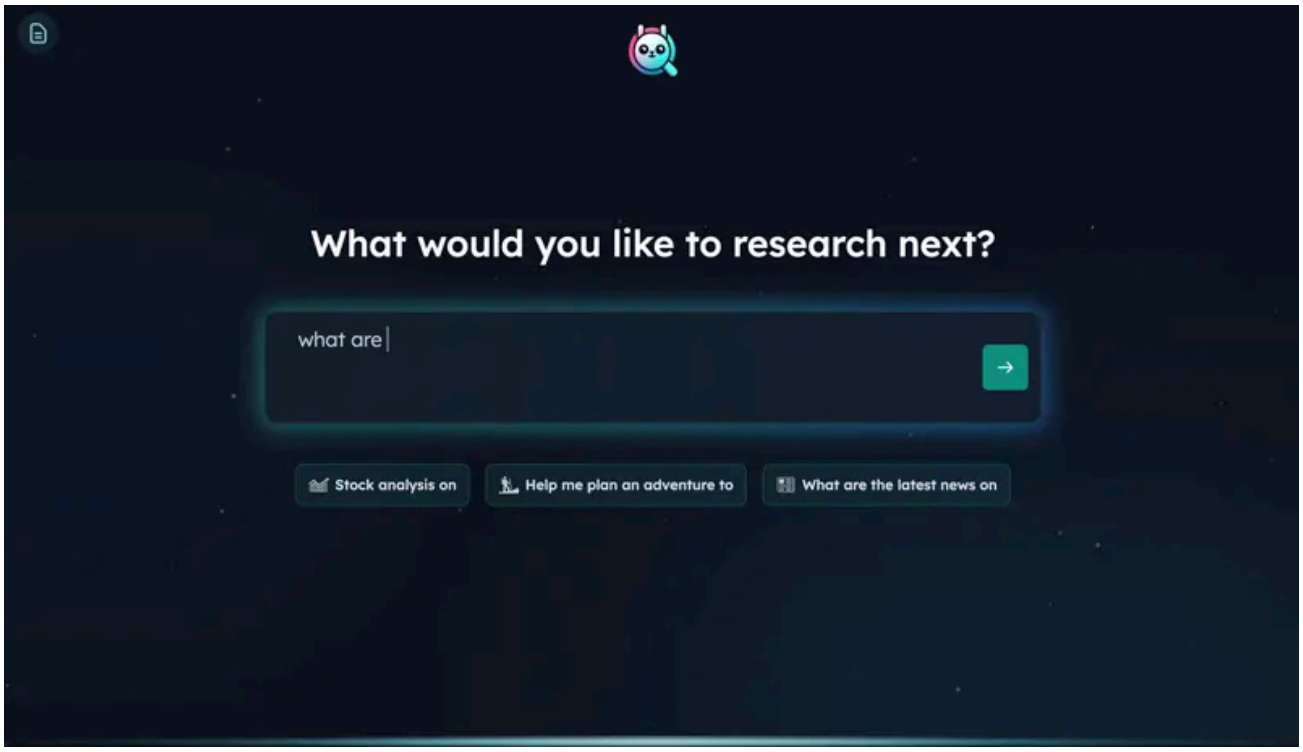
The agent produces detailed, factual, and unbiased research reports with citations. GPT Researcher provides a full suite of customization options to create tailor made and domain specific research agents. Inspired by the recent Plan-and-Solve and RAG papers, GPT Researcher addresses misinformation, speed, determinism, and reliability by offering stable performance and increased speed through parallelized agent work.

Our mission is to empower individuals and organizations with accurate, unbiased, and factual information through AI.

Why GPT Researcher?

- Objective conclusions for manual research can take weeks, requiring vast resources and time.
- LLMs trained on outdated information can hallucinate, becoming irrelevant for current research tasks.
- Current LLMs have token limitations, insufficient for generating long research reports.
- Limited web sources in existing services lead to misinformation and shallow results.
- Selective web sources can introduce bias into research tasks.

Demo



Install as Claude Skill

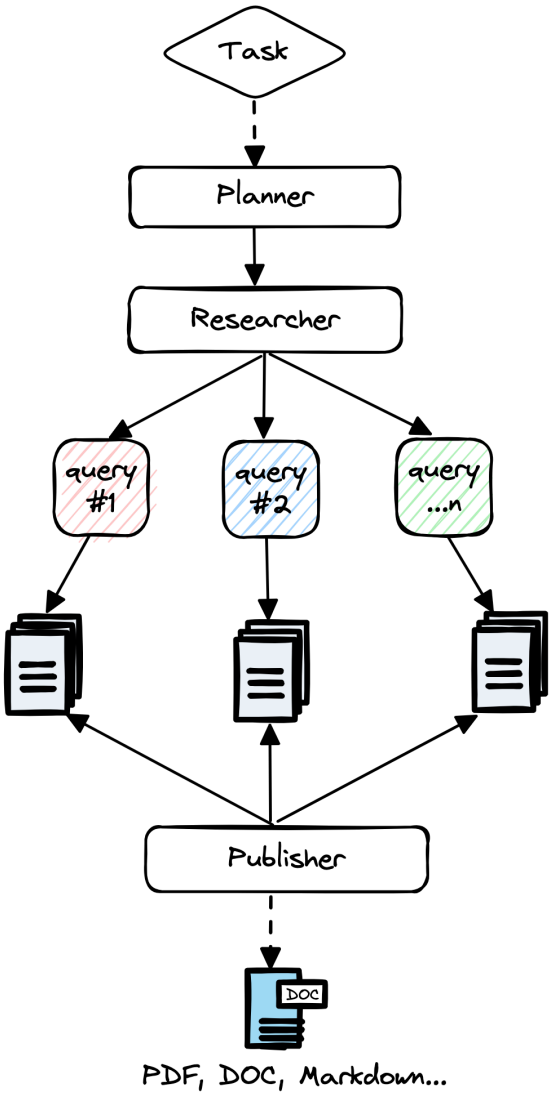
Extend Claude's deep research capabilities by installing GPT Researcher as a Claude Skill:

```
npx skills add assafelovic/gpt-researcher
```

Once installed, Claude can leverage GPT Researcher's deep research capabilities directly within your conversations.

Architecture

The core idea is to utilize 'planner' and 'execution' agents. The planner generates research questions, while the execution agents gather relevant information. The publisher then aggregates all findings into a comprehensive report.



Steps:

- Create a task-specific agent based on a research query.
- Generate questions that collectively form an objective opinion on the task.
- Use a crawler agent for gathering information for each question.
- Summarize and source-track each resource.
- Filter and aggregate summaries into a final research report.

Tutorials

- How it Works
- How to Install
- Live Demo

Features

- 📝 Generate detailed research reports using web and local documents.
- 🖼️ Smart image scraping and filtering for reports.
- 🍌 **AI-generated inline images** using Google Gemini (Nano Banana) for visual illustrations.
- 📜 Generate detailed reports exceeding 2,000 words.
- 🌐 Aggregate over 20 sources for objective conclusions.
- 🖥️ Frontend available in lightweight (HTML/CSS/JS) and production-ready (NextJS + Tailwind) versions.
- 🔍 JavaScript-enabled web scraping.
- 📂 Maintains memory and context throughout research.
- 📄 Export reports to PDF, Word, and other formats.

📖 Documentation

See the Documentation for:

- Installation and setup guides
- Configuration and customization options
- How-To examples
- Full API references

⚙️ Getting Started

Installation

1. Install Python 3.11 or later. Guide.
2. Clone the project and navigate to the directory:

```
git clone https://github.com/assafelovic/gpt-researcher.git
cd gpt-researcher
```

3. Set up API keys by exporting them or storing them in a `.env` file.

```
export OPENAI_API_KEY={Your OpenAI API Key here}
export TAVILY_API_KEY={Your Tavily API Key here}
```

(Optional) For enhanced tracing and observability, you can also set:

```
# export LANGCHAIN_TRACING_V2=true
# export LANGCHAIN_API_KEY={Your LangChain API Key here}
```

For custom OpenAI-compatible APIs (e.g., local models, other providers), you can also set:

```
export OPENAI_BASE_URL={Your custom API base URL here}
```

4. Install dependencies and start the server:

```
pip install -r requirements.txt
python -m uvicorn main:app --reload
```

Visit http://localhost:8000 to start.

For other setups (e.g., Poetry or virtual environments), check the Getting Started page.

Run as PIP package

```
pip install gpt-researcher
```

Example Usage:

```
...
from gpt_researcher import GPTResearcher

query = "why is Nvidia stock going up?"
researcher = GPTResearcher(query=query)
# Conduct research on the given query
research_result = await researcher.conduct_research()
# Write the report
report = await researcher.write_report()
...
```

For more examples and configurations, please refer to the PIP documentation page.

🔧 MCP Client

GPT Researcher supports MCP integration to connect with specialized data sources like GitHub repositories, databases, and custom APIs. This enables research from data sources alongside web search.

```
export RETRIEVER=tavily,mcp  # Enable hybrid web + MCP research
```

```
from gpt_researcher import GPTResearcher
import asyncio
import os

async def mcp_research_example():
    # Enable MCP with web search
    os.environ["RETRIEVER"] = "tavily,mcp"

    researcher = GPTResearcher(
        query="What are the top open source web research agents?",
        mcp_configs=[
            {
                "name": "github",
                "command": "npx",
                "args": ["-y", "@modelcontextprotocol/server-github"],
                "env": {"GITHUB_TOKEN": os.getenv("GITHUB_TOKEN")}
            }
        ]
    )

    research_result = await researcher.conduct_research()
    report = await researcher.write_report()
    return report
```

> For comprehensive MCP documentation and advanced examples, visit the MCP Integration Guide.

🍌 Inline Image Generation

GPT Researcher can automatically generate and embed AI-created illustrations in your research reports using Google's Gemini models (Nano Banana).

```
# Enable in your .env file
IMAGE_GENERATION_ENABLED=true
GOOGLE_API_KEY=your_google_api_key
IMAGE_GENERATION_MODEL=models/gemini-2.5-flash-image
```

When enabled, the system will:

1. Analyze your research context to identify visualization opportunities
2. Pre-generate 2-3 relevant images during the research phase
3. Embed them inline as the report is written

Images are generated with dark-mode styling that matches the GPT Researcher UI, featuring professional infographic aesthetics with teal accents.

Learn more about Image Generation in our documentation.

✨ Deep Research

GPT Researcher now includes Deep Research - an advanced recursive research workflow that explores topics with agentic depth and breadth. This feature employs a tree-like exploration pattern, diving deeper into subtopics while maintaining a comprehensive view of the research subject.

- 🌳 Tree-like exploration with configurable depth and breadth
- ⚡ Concurrent processing for faster results
- 🤝 Smart context management across research branches
- ⏱️ Takes ~5 minutes per deep research
- 💰 Costs ~$0.4 per research (using `o3-mini` on "high" reasoning effort)

Learn more about Deep Research in our documentation.

Run with Docker

> **Step 1** - Install Docker

> **Step 2** - Clone the '.env.example' file, add your API Keys to the cloned file and save the file as '.env'

> **Step 3** - Within the docker-compose file comment out services that you don't want to run with Docker.

```
docker-compose up --build
```

If that doesn't work, try running it without the dash:

```
docker compose up --build
```

> **Step 4** - By default, if you haven't uncommented anything in your docker-compose file, this flow will start 2 processes:

- the Python server running on localhost:8000
- the React app running on localhost:3000

Visit localhost:3000 on any browser and enjoy researching!

📄 Research on Local Documents

You can instruct the GPT Researcher to run research tasks based on your local documents. Currently supported file formats are: PDF, plain text, CSV, Excel, Markdown, PowerPoint, and Word documents.

Step 1: Add the env variable `DOC_PATH` pointing to the folder where your documents are located.

```
export DOC_PATH="./my-docs"
```

Step 2:

- If you're running the frontend app on localhost:8000, simply select "My Documents" from the "Report Source" Dropdown Options.
- If you're running GPT Researcher with the PIP package, pass the `report_source` argument as "local" when you instantiate the `GPTResearcher` class code sample here.

🤖 MCP Server

We've moved our MCP server to a dedicated repository: gptr-mcp.

The GPT Researcher MCP Server enables AI applications like Claude to conduct deep research. While LLM apps can access web search tools with MCP, GPT Researcher MCP delivers deeper, more reliable research results.

Features:

- Deep research capabilities for AI assistants
- Higher quality information with optimized context usage
- Comprehensive results with better reasoning for LLMs
- Claude Desktop integration

For detailed installation and usage instructions, please visit the official repository.

👨‍👩‍👦 Multi-Agent Assistant

As AI evolves from prompt engineering and RAG to multi-agent systems, we're excited to introduce multi-agent assistants built with LangGraph and AG2.

By using multi-agent frameworks, the research process can be significantly improved in depth and quality by leveraging multiple agents with specialized skills. Inspired by the recent STORM paper, this project showcases how a team of AI agents can work together to conduct research on a given topic, from planning to publication.

An average run generates a 5-6 page research report in multiple formats such as PDF, Docx and Markdown.

Check it out here or head over to our documentation for LangGraph and AG2 for more information.

🔍 Observability

GPT Researcher supports **LangSmith** for enhanced tracing and observability, making it easier to debug and optimize complex multi-agent workflows.

To enable tracing:

1. Set the following environment variables:

```
export LANGCHAIN_TRACING_V2=true
export LANGCHAIN_API_KEY=your_api_key
export LANGCHAIN_PROJECT="gpt-researcher"
```

2. Run your research tasks as usual. All LangGraph-based agent interactions will be automatically traced and visualized in your LangSmith dashboard.

🖥️ Frontend Applications

GPT-Researcher now features an enhanced frontend to improve the user experience and streamline the research process. The frontend offers:

- An intuitive interface for inputting research queries
- Real-time progress tracking of research tasks
- Interactive display of research findings
- Customizable settings for tailored research experiences

Two deployment options are available:

1. A lightweight static frontend served by FastAPI
2. A feature-rich NextJS application for advanced functionality

For detailed setup instructions and more information about the frontend features, please visit our documentation page.

🚀 Contributing

We highly welcome contributions! Please check out contributing if you're interested.

Please check out our roadmap page and reach out to us via our Discord community if you're interested in joining our mission.

Releases 68

v3.4.3 Latest

3 weeks ago

+ 67 releases

Packages 3

- gpt-researcher
- gpt-researcher-frontend-nextjs
- gpt-researcher-docs-discord-bot

Used by 254

+ 246

Contributors 205

+ 191 contributors

Languages

Python 54.8% **TypeScript** 27.8% **JavaScript** 7.1% **CSS** 6.3% **HCL** 1.9% **HTML** 1.8%
Other 0.3%