



gpt-researcher 설치 및 운영

Status Draft

GPT-Researcher 상세 매뉴얼

본 매뉴얼은 GPT-Researcher의 설치, 사용 및 다양한 활용 예시를 상세히 다루어, 사용자가 이 강력한 AI 연구 에이전트를 효과적으로 활용할 수 있도록 돕는 것을 목표로 합니다.

1. GPT-Researcher 개요

GPT-Researcher는 웹 및 로컬 환경에서 심층적인 연구를 수행하도록 설계된 오픈 소스 AI 에이전트입니다. 그 주된 목적은 상세하고 사실적이며 편향되지 않은 연구 보고서를 인용과 함께 생성하는 것입니다. 이 에이전트는 광범위한 사용자 정의 옵션을 제공하여, 사용자가 특정 도메인에 맞춤화된 연구 에이전트를 생성할 수 있도록 지원합니다.

1.1. GPT-Researcher란 무엇인가?

GPT-Researcher는 웹 및 로컬 환경에서 주어진 모든 작업에 대해 심층 연구를 수행하는 개방형 에이전트입니다. 그 핵심 목적은 상세하고, 사실적이며, 편향되지 않은 연구 보고서를 인용과 함께 생성하는 것입니다. 이 에이전트는 광범위한 사용자 정의 옵션을 제공하며, 사용자가 특정 도메인에 맞춤화된 연구 에이전트를 생성할 수 있도록 지원합니다. 이 시스템은 Plan-and-Solve 및 RAG(Retrieval-Augmented Generation) 논문에서 영감을 받아 개발되었으며, 병렬화된 에이전트 작업을 통해 안정적인 성능과 향상된 속도를 제공함으로써 정보의 오용, 처리 속도, 결정론, 신뢰성 등의 문제를 해결하고자 합니다. 궁극적인 임무는 AI를 통해 개인과 조직에 정확하고 편향되지 않으며 사실적인 정보를 제공하는 것입니다.

GPT-Researcher의 개발은 현재 정보 환경에서 나타나는 여러 문제점을 해결하는 데 중점을 둡니다. 첫째, 수동 연구를 통해 객관적인 결론을 도출하는 과정은 종종 수 주가 소요될 수 있으며 상당한 자원과 시간을 요구합니다. 둘째, 오래된 정보로 학습된 대규모 언어 모델(LLMs)은 때때로 "환각" 현상을 일으켜 부정확하거나 관련 없는 정보를 생성할 수 있으며, 이는 현재 연구 작업에 부적합할 수 있습니다. 셋째, 현재 LLMs는 긴 연구 보고서를 생성하기에 불충분한 토큰 제한을 가지고 있어, 심층적인 분석을 어렵게 만듭니다. 넷째, 기존 서비스의 제한적인 웹 소스 사용은 오정보와 피상적인 결과로 이어질 수 있으며, 선택적인 웹 소스는 연구 작업에 편향을 도입할 수 있습니다. GPT-Researcher는 이러한 문제들을 해결함으로써, 정보의 신뢰성과 효율성을 높이는 것을 목표로 합니다.

이 도구는 연구 보고서 생성, 개요 작성, 자료 보고서 취합, 강의 보고서 개발 등 다양한 일반적인 활용 사례를 지원합니다. 특히 2,000단어 이상의 장문 보고서 생성 능력을 갖추고 있으며, 연구 작업당 20개 이상의 웹 소스를 취합하여 객관적이고 사실적인 결론을 도출합니다. 또한, 사용자 친화적인 웹 인터페이스를 제공하고, JavaScript를 지원하는 웹 소스를 스크래핑할 수 있습니다. 방문 및 사용된 웹 소스를 추적하여 투명성을 보장하며, PDF, Word 등 다양한 형식으로 보고서를 내보낼 수 있습니다.

GPT-Researcher가 "사실적이고 편향되지 않은 연구 보고서와 인용"을 반복적으로 강조하고, LLM의 고질적인 문제인 "오정보"와 "환각"을 해결하겠다는 점은 이 도구가 명확한 전략적 포지셔닝을 가지고 있음을 나타냅니다. 이는 단순히 LLM을 활용한 도구가 아니라, 정확성과 검증 가능성이 필수적인 중요한 애플리케이션에서 순수 LLM의 알려진 약점을 보완하기 위해 설계되었습니다. Plan-and-Solve 및 RAG 논문에서 영감을 받았다는 언급은 이 도구가 정보 검색 및 합성에 있어 견고하고 원칙적인 접근 방식을 기반으로 함을 보여주며, 이는 경험적 접근 방식보다 더 신뢰할 수 있는 결과를 제공할 수 있습니다. 이러한 설계는 GPT-Researcher가 신뢰할 수 있는 AI 생성 콘텐츠에 대한 중요한 시장 요구를 충족시키는 도구로 자리매김하게 합니다.

1.2. 주요 기능 및 아키텍처

GPT-Researcher는 자율 에이전트들이 복잡한 작업을 수행하기 위해 소통하고 협력하는 다중 에이전트 프레임워크 내에서 작동하도록 설계되었습니다. 이 시스템은 Langchain에서 구축된 Python 라이브러리인 LangGraph와 통합되어 있으며, 이는 상태 저장형, 다중 액터 LLM 애플리케이션을 구축하는 데 사용됩니다. LangGraph의 DAG(Directed Acyclic Graph) 아키텍처는 전문화된 에이전트들이 통신하고 작업을 트리거할 수 있도록 함으로써 복잡한 연구 프로세스를 효율적으로 관리합니다.

GPT-Researcher의 연구 팀은 총 7개의 AI 에이전트로 구성되어 있으며, 각 에이전트는 특정 역할을 수행합니다:

- **Chief Editor (총괄 편집자):** 연구 프로세스를 감독하고 팀을 관리하는 "마스터" 에이전트로서, LangGraph를 사용하여 다른 에이전트들을 조정합니다.
- **Researcher (연구원 - gpt-researcher):** 주어진 주제에 대해 심층 연구를 수행하는 전문 자율 에이전트입니다.
- **Editor (편집자):** 연구 개요 및 구조를 계획하는 역할을 담당합니다.
- **Reviewer (검토자):** 주어진 기준에 따라 연구 결과의 정확성을 검증하고 피드백을 제공합니다.
- **Revisor (수정자):** 검토자의 피드백을 기반으로 연구 결과를 수정합니다.
- **Writer (작성자):** 서론, 결론, 참고 문헌 섹션을 포함하여 최종 보고서를 작성하고 편집하는 역할을 담당합니다.
- **Publisher (출판자):** 최종 보고서를 PDF, Docx, Markdown 등 다양한 형식으로 출판하는 역할을 담당합니다.

이러한 다중 에이전트 팀은 다음 5단계의 연구 프로세스를 통해 작업을 수행합니다:

1. **Planning stage (계획 단계):** Chief Editor가 전체 프로세스를 감독하며 연구의 방향을 설정합니다.
2. **Data collection and analysis (데이터 수집 및 분석):** Researcher가 심층 연구를 수행하고 초안을 작성합니다.
3. **Writing and submission (작성 및 제출):** Writer가 서론, 결론, 참고 문헌을 포함한 최종 보고서를 작성합니다.
4. **Review and revision (검토 및 수정):** Reviewer가 초안의 정확성을 검증하고 피드백을 제공하며, Revisor가 이 피드백을 기반으로 수정합니다.

5. **Publication (출판):** Publisher가 최종 보고서를 다양한 형식으로 출판합니다.

7개의 에이전트로 구성된 상세한 프레임워크와 그들의 순차적인 프로세스는 GPT-Researcher의 중요한 설계 선택입니다. 이는 여러 역할이 정보에 기여하고 이를 검증하는 인간의 편집 프로세스를 모방합니다. LangGraph를 활용한 이 다중 에이전트 아키텍처는 단순히 작업 분배를 넘어, 내부적인 점검과 균형을 도입하여 결과물의 신뢰성과 품질을 향상시키기 위한 의도적인 전략입니다. 특히 "Reviewer"와 "Revisor" 에이전트는 비편향적이고 사실적이라는 주장을 뒷받침하기 위해 비판적 검토 주기를 시뮬레이션합니다. 이는 단일 에이전트 LLM 시스템에 비해 더 높은 수준의 결정론과 신뢰성을 의미하며, 이는 개요에서 강조된 정보의 오용 및 편향 문제를 직접적으로 해결합니다. 이러한 설계는 GPT-Researcher의 강력한 차별화 요소로 작용합니다.

2. 설치 방법

GPT-Researcher는 다양한 방법으로 설치 및 실행할 수 있으며, 사용자의 환경과 선호도에 따라 적절한 방법을 선택할 수 있습니다.

2.1. 사전 준비 사항 (Prerequisites)

GPT-Researcher를 설치하기 위해서는 Python 3.11 이상 버전이 필요합니다. Python 공식 웹사이트에서 최신 버전을 다운로드하여 설치할 수 있습니다. Python 3.11+의 엄격한 요구 사항은 GPT-Researcher가 이 버전에서 도입된 특정 기능이나 최적화에 의존하고 있음을 의미합니다. 이는 사용자에게 매우 중요하며, 이전 Python 버전을 사용하면 호환성 문제와 오류가 발생할 가능성이 높습니다.

또한, OpenAI 및 Tavily API 키를 준비해야 합니다. GPT-Researcher는 기본적으로 OpenAI를 LLM 호출에 사용하고 Tavily API를 실시간 온라인 정보 검색에 사용합니다. Tavily는 연구 작업에 있어 사실적이고 편향되지 않은 정보에 중점을 두기 때문에 추천됩니다.

2.2. 기본 설치 및 실행 (Standard Installation and Execution)

이 방법은 가장 일반적인 설치 방식으로, GitHub 저장소를 클론하여 직접 실행합니다.

프로젝트 클론: 먼저 GPT-Researcher 프로젝트를 GitHub 저장소에서 클론하고 해당 디렉토리로 이동합니다.

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

1. **API 키 설정:** OpenAI 및 Tavily API 키를 환경 변수로 설정하거나 .env 파일에 저장해야 합니다.

환경 변수로 내보내기 (임시, Linux/Windows): 현재 세션에서만 유효한 임시 설정을 위해 다음 명령어를 사용합니다.

```
export OPENAI_API_KEY={여기에 OpenAI API 키 입력}export TAVILY_API_KEY={여기에 Tavily API 키 입력}
```

- **.env 파일 사용 (영구 설정):** gpt-researcher 폴더 내에 .env 파일을 생성하고 다음 형식으로 키를 입력합니다. 이 방법은 민감한 자격 증명이 코드 저장소나 명령 기록에 우발적으로 노출되는 것을 방지하는 중요한 보안 모범 사례입니다. *참고: 다른 LLM 또는 검색 엔진 공급자를 사용하려면 해당 API 키를 환경 변수로 설정해야 할 수 있습니다.*

```
OPENAI_API_KEY={여기에 OpenAI API 키 입력}
TAVILY_API_KEY={여기에 Tavily API 키 입력}
```

의존성 설치 및 서버 시작: API 키 설정을 완료한 후, 필요한 의존성을 pip를 사용하여 설치하고 서버를 시작합니다.

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

- **웹 인터페이스 접속:** 서버가 성공적으로 실행되면 웹 브라우저에서 `http://localhost:8000`으로 접속하여 GPT-Researcher의 웹 인터페이스를 사용할 수 있습니다.

2.3. PIP 패키지로 설치 및 사용 (Installation and Usage as a PIP Package)

GPT-Researcher는 PIP 패키지로도 설치하여 Python 프로젝트에 통합할 수 있습니다. 이는 재사용성과 통합에 중점을 둔 설계 철학을 나타냅니다. 이를 통해 개발자는 GPT-Researcher의 핵심 기능을 기존 Python 애플리케이션이나 워크플로에 쉽게 포함할 수 있으며, 독립형 웹 인터페이스에 국한되지 않습니다. 이는 데이터 파이프라인, 콘텐츠 생성 플랫폼 또는 분석 도구와 같은 더 큰 시스템 내에서 연구 작업을 자동화할 수 있도록 하여 유용성을 크게 확장합니다.

PIP 패키지 설치:

```
pip install gpt-researcher
```

예시 사용 (Python 스크립트):

GPTResearcher 클래스를 임포트하여 연구를 수행할 수 있습니다.

```
from gpt_researcher import GPTResearcher
import asyncio

async def run_research():
    query = "왜 엔비디아 주식이 오르는가?"
    researcher = GPTResearcher(query=query, report_type="research_report")
    research_result = await researcher.conduct_research()
    report = await researcher.write_report()
    print(report)

if __name__ == "__main__":
    asyncio.run(run_research())
```

2.4. Docker를 이용한 설치 및 실행 (Installation and Execution with Docker)

Docker를 사용하여 GPT-Researcher를 컨테이너화된 환경에서 실행할 수 있습니다. Docker 지원은 배포 및 환경 관리에 있어 상당한 이점을 제공합니다. 모든 의존성과 구성을 캡슐화하여 "내 컴퓨터에서는 잘 작동하는데..."와 같은 문제를 제거하고, 다양한 운영 체제나 프로덕션 환경에서 설정을 간소화합니다. 이는 특히 시스템 수준 라이브러리 의존성으로 인해 어려움을 겪을 수 있는 사용자에게 설치 마찰을 줄여줍니다.

1. **Docker 설치:** 시스템에 Docker가 설치되어 있는지 확인합니다.
2. **.env 파일 구성:** .env.example 파일을 .env로 복사하고, 필요한 API 키를 추가합니다.
3. **원치 않는 서비스 주석 처리:** docker-compose 파일 내에서 Docker로 실행하고 싶지 않은 서비스를 주석 처리합니다. 이 기능은 모듈식 설계를 의미하며, 사용자가 필요한 구성 요소만 실행하여 리소스 사용을 최적화할 수 있도록 합니다.

Docker Compose 빌드 및 실행: 다음 명령어 중 하나를 사용하여 애플리케이션을 빌드하고 실행합니다.
`docker-compose up --build#` 또는 `docker compose up --build`

1. **접속:** 기본적으로 이 Docker 흐름은 두 가지 프로세스를 시작합니다: localhost:8000의 Python 서버와 localhost:3000의 React 앱. localhost:3000에 접속하여 연구를 시작할 수 있습니다.

2.5. 가상 환경 또는 Poetry 사용 (Using Virtual Environment or Poetry)

프로젝트별 의존성의 깔끔한 분리를 위해 가상 환경이나 Poetry 사용이 권장됩니다. 이는 시스템 전체 패키지와의 충돌을 방지하고 프로젝트 수명 주기 전반에 걸쳐 더 간단한 의존성 관리를 가능하게 합니다. Poetry 셸을 활성화하는 것은 올바른 버전의 의존성이 사용되고 효율적인 개발 및 테스트에 도움이 되는 제어된 환경을 제공하므로 프로젝트 작업에 필수적입니다. 가상 환경 또는 Poetry 사용에 대한 명시적인 권장 사항은 단순한 설치 단계를 넘어 장기적인 프로젝트 건전성과 유지보수성을 위한 능동적인 조치입니다. 의존성을 격리함으로써 사용자는 동일한 컴퓨터에서 다른 프로젝트가 충돌하는 라이브러리 버전을 요구하는 "의존성 지옥"을 피할 수 있습니다. 이러한 선견지명은 강력한 개발 관행을 나타내며, 통합 및 지속적인 사용을 위한 도구에 있어 안정적이고 예측 가능한 개발 경험을 제공하는 것을 목표로 합니다.

3. 사용 방법

GPT-Researcher는 웹 인터페이스를 통해 직관적으로 사용하거나, Python 스크립트를 통해 프로그래밍 방식으로 활용할 수 있습니다. 또한, 특정 소스나 로컬 문서를 기반으로 연구를 수행하는 등 다양한 고급 사용 시나리오를 지원합니다.

3.1. 웹 인터페이스를 통한 사용 (Using the Web Interface)

서버를 실행한 후(<http://localhost:8000> 또는 Docker의 경우 <http://localhost:3000> React 앱), 웹 브라우저를 통해 GPT-Researcher의 사용자 친화적인 웹 인터페이스에 접속할 수 있습니다.

localhost:8000을 통해 접근 가능한 "사용자 친화적인 웹 인터페이스"의 제공은 순수 개발자 외의 사용자층으로 확장하기 위한 전략적 움직임입니다. PIP 패키지가 프로그래밍 방식의 통합을 지원하는 반면, 웹 인터페이스는 명령줄 작업이나 Python 스크립팅에 익숙하지 않은 연구원, 콘텐츠 제작자 또는 비즈니스 분석가도 GPT-Researcher에 접근할 수 있도록 하여 채택 잠재력을 극대화합니다.

3.2. Python 스크립트를 통한 기본 사용 예시 (Basic Usage Examples via Python Script)

PIP 패키지로 설치한 경우, Python 스크립트 내에서 GPTResearcher 클래스를 임포트하여 연구를 수행할 수 있습니다. GPTResearcher 인스턴스 생성, `conduct_research()`, `write_report()` 호출로 구성된 핵심 Python API는 고급 AI 기반 워크플로우를 구축하기 위한 기반입니다. 이는 단순히 보고서를 얻는 것을 넘어, "보고서를 어떻게 활용하는가"에 대한 문제입니다.

```
from gpt_researcher import GPTResearcher
```

```
import asyncio
```

```
async def get_report(query: str, report_type: str)
-> str:
```

```
# GPTResearcher 인스턴스 생성
```

```
# query: 연구할 주제
```

```
# report_type: 생성할 보고서의 유형 (예:
"research_report", "resource_report",
"outline_report", "lesson_report",
"custom_report")
```

```

researcher = GPTResearcher(query=query,
report_type=report_type)

# 연구 수행

research_result = await
researcher.conduct_research()

# 보고서 작성

report = await researcher.write_report()

return report

if __name__ == "__main__":

# 예시 1: 일반 연구 보고서 생성

query_general = "엔비디아 주식이 왜 오르는가?"

report_type_general = "research_report"

print(f"--- 일반 연구 보고서: {query_general} ---")

report_general =
asyncio.run(get_report(query_general,
report_type_general))

print(report_general)

# 예시 2: 특정 주제에 대한 개요 보고서 생성

```

```

query_outline = "인공지능의 최신 발전 동향"

report_type_outline = "outline_report"

print(f"\n--- 개요 보고서: {query_outline} ---")

report_outline =
asyncio.run(get_report(query_outline,
report_type_outline))

print(report_outline)

```

생성된 보고서는 뉴스 기사, 마케팅 콘텐츠, 이메일 템플릿, 뉴스레터 등 가치 있는 콘텐츠를 생성하기 위한 맥락으로 활용될 수 있습니다. 또한 코드 문서, 비즈니스 분석, 금융 정보 등에 대한 정보를 수집하는 데에도 사용될 수 있습니다. GPT-Researcher의 결과물이 이러한 다양한 콘텐츠를 생성하기 위한 맥락으로 활용될 수 있다는 명시적인 언급은 GPT-Researcher가 더 큰 AI 시스템 내에서 활성화 구성 요소로서의 역할을 수행함을 보여줍니다. 이는 독립형 연구를 넘어, 후속 생성형 AI 작업을 위한 중요한 데이터 보강 또는 지식 검색 계층 역할을 수행하여 그 가치를 확장합니다.

3.3. 로컬 문서 기반 연구 (Research on Local Documents)

GPT-Researcher는 PDF, 일반 텍스트, CSV, Excel, Markdown, PowerPoint, Word 문서를 포함한 로컬 문서를 기반으로 연구 작업을 지원합니다. 이 기능은 GPT-Researcher의 유용성을 순수 웹 기반 에이전트에서 조직의 내부 지식 기반을 활용할 수 있는 도구로 크게 확장합니다. 이는 민감하거나 독점적인 정보가 공개 인터넷에 노출될 수 없는 기업 애플리케이션에 매우 중요합니다. 이는 GPT-Researcher를 강력한 내부 RAG(Retrieval-Augmented Generation) 시스템으로 변모시켜, 기업이 자체 비공개 데이터를 기반으로 보고서, 분석 또는 콘텐츠를 생성할 수 있도록 함으로써 내부 효율성과 지식 관리를 향상시킵니다.

DOC_PATH 환경 변수 설정: 문서가 있는 폴더를 가리키도록 DOC_PATH 환경 변수를 추가합니다.
export DOC_PATH="./my-docs" # 예시: 내 문서 폴더 경로

1. 소스 선택:

- localhost:8000에서 프론트엔드 앱을 사용하는 경우, "Report Source" 드롭다운 옵션에서 "My Documents"를 선택합니다.

- PIP 패키지로 GPT-Researcher를 실행하는 경우, GPTResearcher 클래스를 인스턴스화할 때 report_source 인수를 "local"로 전달합니다.

3.4. 특정 소스 기반 연구 (Research on Specific Sources)

사용자는 URL 목록을 제공하여 GPT-Researcher가 지정된 소스에서만 연구를 수행하도록 지시할 수 있습니다. source_urls를 지정할 수 있는 기능은 연구 프로세스에 대한 중요한 수준의 제어를 제공합니다. 이는 GPT-Researcher가 해결하고자 하는 핵심 문제 중 하나인 "선택적인 웹 소스는 연구 작업에 편향을 도입할 수 있습니다"를 해결합니다. 사용자가 소스를 큐레이션할 수 있도록 함으로써, 편향을 적극적으로 완화하고, 권위 있는 도메인에 집중하거나, 특정 관점에 대한 비교 분석을 수행할 수 있습니다. 이 기능은 대상 연구, 학술적 무결성 또는 알려진 소스에서 정보를 검증할 때 매우 유용하며, 생성된 보고서의 신뢰성과 관련성을 향상시킵니다.

```
from gpt_researcher import GPTResearcher

import asyncio

async def get_report_from_sources(query: str,
report_type: str, sources: list) -> str:

researcher = GPTResearcher(query=query,
report_type=report_type, source_urls=sources)

await researcher.conduct_research()

report = await researcher.write_report()

return report

if __name__ == "__main__":

query = "인공지능의 최신 발전 동향은 무엇인가?"

report_type = "research_report"
```

```
# 위키피디아와 IBM Watson AI 페이지에서만 연구 수행
```

```
sources = ["https://ko.wikipedia.org/wiki/인공지능", "https://www.ibm.com/kr-ko/watson/ai"]
```

```
print(f"--- 특정 소스 기반 연구 보고서: {query} (소스: {sources}) ---")
```

```
report_specific_sources =  
asyncio.run(get_report_from_sources(query,  
report_type, sources))
```

```
print(report_specific_sources)
```

3.5. 사용자 정의 프롬프트 및 보고서 유형 (Custom Prompts and Report Types)

사용자는 query 인수로 사용자 정의 프롬프트를 전달하고 report_type을 "custom_report"로 설정하여 연구 방향을 안내하고 보고서 레이아웃을 맞춤화할 수 있습니다. 사용자 정의 query와 결합된 custom_report 유형은 보고서의 형태를 구성하는 데 있어 탁월한 유연성을 제공합니다. 이를 통해 사용자는 미리 정의된 보고서 형식을 넘어 연구의 정확한 범위, 어조 및 구조를 지시할 수 있습니다. 이는 틈새 애플리케이션, 특정 학술적 요구 사항(예시에서 언급된 APA 형식), 또는 매우 구체적인 출력 특성을 요구하는 콘텐츠 생성 파이프라인에 GPT-Researcher를 통합할 때 특히 유용합니다. 이 기능은 사용자가 연구 에이전트를 진정으로 "맞춤 제작"할 수 있도록 하여, 개요에서 약속된 사용자 정의 가능성을 충족시킵니다.

```
from gpt_researcher import GPTResearcher
```

```
import asyncio
```

```
async def get_custom_report(prompt: str,  
report_type: str) -> str:
```

```

researcher = GPTResearcher(query=prompt,
report_type=report_type)

await researcher.conduct_research()

report = await researcher.write_report()

return report

if __name__ == "__main__":

report_type = "custom_report"

prompt = "인공지능의 최신 발전 동향을 연구하고,
APA 형식으로 소스를 포함한 상세 보고서를 제공하시
오."

print(f"--- 사용자 정의 프롬프트 보고서: {prompt} --
-")

report_custom_prompt =
asyncio.run(get_custom_report(prompt=prompt,
report_type=report_type))

print(report_custom_prompt)

```

4. 고급 설정 및 사용자 정의

GPT-Researcher는 config.py 파일을 통해 광범위한 사용자 정의 옵션을 제공하며, 이를 통해 LLM, 검색 엔진, 보고서 형식, 연구 깊이 등을 세밀하게 조정할 수 있습니다. 외부 JSON 설정 파일을 사용하여 이러한 설정을 관리하는 것도 가능합니다.

4.1. config.py 파일 이해 (Understanding config.py)

config.py 파일은 GPT-Researcher의 동작을 제어하는 다양한 매개변수를 포함하고 있으며, 이를 수정하여 에이전트의 성능, 품질 및 출력을 최적화할 수 있습니다. 광범위한 사용자 정의를 위한 중앙 집중식 config.py 파일의 존재는 유지보수 가능하고 확장 가능한 소프트웨어의 핵심 설계 패턴입니다. 이는 운영 매개변수에 대한 단일 진실 소스를 제공하여, 다양한 실행이나 환경에서 연구 설정을 관리하고 재현하기 쉽게 만듭니다. 이 접근 방식은 복잡한 다중 에이전트 시스템을 다룰 때 분산된 구성보다 우수하며, 일관성을 보장하고 디버깅을 간소화합니다.

4.2. LLM 및 검색 엔진 설정 (LLM and Retriever Configuration)

GPT-Researcher는 다양한 LLM 공급자와 검색 엔진을 지원하며, 이를 config.py 파일에서 설정할 수 있습니다.

- **LLM 공급자 및 모델:**

- llm_provider: 기본값은 "openai"입니다. 다른 Langchain 지원 LLM으로 변경할 수 있습니다 (예: "huggingface", "cohere").
- fast_llm_model: 기본값은 "gpt-3.5-turbo"입니다. 빠르고 덜 복잡한 작업에 사용됩니다.
- smart_llm_model: 기본값은 "gpt-4o"입니다. 더 복잡하고 심층적인 작업에 사용됩니다.
- 시스템은 gpt-3.5-turbo 및 gpt-4-turbo (128K 컨텍스트)를 필요할 때만 사용하여 비용을 최적화하도록 설계되었습니다. 평균 연구 작업은 약 0.1달러의 비용이 들고 약 3분 정도 소요됩니다.

- **검색 엔진 (Retriever):**

- retriever: 기본값은 "tavily"입니다. duckduckgo, googleAPI, googleSerp, searx, bing 등으로 변경할 수 있습니다. 다른 공급자는 API 키를 요구할 수 있습니다.

llm_provider, fast_llm_model, smart_llm_model, retriever를 구성할 수 있는 기능은 단순히 선택의 폭을 넓히는 것을 넘어 전략적 유연성과 비용-성능 최적화를 의미합니다. "빠른" 작업에 gpt-3.5-turbo를, "스마트한" 작업에 gpt-4o를 기본으로 사용하는 것은 명시적인 비용 최적화 전략을 보여줍니다. 이 계층적 모델 사용은 사용자가 비용과 품질의 균형을 맞출 수 있도록 하여, 덜 중요한 작업에는 저렴한 모델을, 복잡하고 심층적인 연구에는 더 강력하고 비용이 많이 드는 모델을 선택할 수 있게 합니다. 마찬가지로, 검색 엔진을 전환하는 것은 특정 검색 요구 사항, 지역 가용성 또는 비용 고려 사항에 대한 적응을 가능하게 합니다. 이러한 설계 선택은 AI 리소스 관리에 대한 실용적인 접근 방식을 강조합니다.

4.3. 보고서 형식, 단어 수, 연구 깊이 조정 (Adjusting Report Format, Word Count, Research Depth)

사용자는 config.py 파일을 통해 보고서의 출력 특성을 세밀하게 제어할 수 있습니다.

- report_format: 기본값은 "apa"입니다. 다양한 보고서 형식에 맞게 조정할 수 있습니다.
- total_words: 생성될 보고서의 총 단어 수를 제어합니다. 기본값은 1000입니다.

- `max_iterations`: 연구 프로세스의 깊이를 제어합니다. 기본값은 1입니다.

이러한 매개변수들은 보고서의 출력 특성에 대한 세분화된 제어를 제공합니다. APA 형식을 지정할 수 있다는 것은 특정 형식 요구 사항이 있는 학술 또는 전문 사용자를 충족시킵니다. `total_words`를 조정하면 간결한 요약 또는 광범위한 심층 분석이 가능합니다. `max_iterations`를 수정하면 연구의 깊이와 철저함에 직접적인 영향을 미쳐, 사용자가 속도와 포괄성 사이에서 균형을 맞출 수 있습니다. 이러한 수준의 제어는 생성된 보고서가 일반적이지 않고 다양한 사용자 요구 사항 및 사용 사례(빠른 개요부터 포괄적인 학술 논문까지)에 정확하게 부합하도록 보장합니다.

4.4. 외부 JSON 설정 파일 사용 (Using External JSON Configuration Files)

`config.py` 파일을 직접 편집하는 대신, 외부 JSON 설정 파일을 사용하여 구성을 제공할 수 있습니다. `GPTResearcher` 클래스를 초기화할 때 `config_file` 매개변수에 JSON 파일의 경로를 전달합니다.

```
researcher = GPTResearcher(query=query,  
report_type=report_type,  
config_path="path/to/your/config.json")
```

외부 JSON 구성 파일을 사용할 수 있는 옵션은 `config.py`를 직접 편집하는 것보다 훨씬 더 큰 유연성을 제공합니다. 이를 통해 구성의 동적 로딩, 다양한 설정의 쉬운 A/B 테스트, 그리고 핵심 프로젝트 파일을 수정하지 않고도 버전 제어 시스템(예: Git)과의 원활한 통합이 가능합니다. 복잡한 배포나 협업 환경에서 이러한 구성의 외부화는 더 나은 관리, 재현성 및 유연성을 촉진하여 사용자가 코드 변경 없이 다른 운영 프로필 간에 전환할 수 있도록 합니다.

4.5. 표: 주요 설정 매개변수

다음 표는 GPT-Researcher의 주요 설정 매개변수와 그 목적, 기본값, 그리고 가능한 옵션을 요약하여 보여줍니다. 이 표는 `config.py` 파일에 포함된 수많은 매개변수들을 사용자에게 명확하고 간결하게 제시합니다. 이 매개변수들은 GPT-Researcher의 동작, 품질 및 비용을 직접적으로 제어합니다. 표는 각 매개변수의 목적, 기본값, 그리고 가능한 옵션/설명을 한눈에 볼 수 있는 빠른 참조를 제공합니다. 이러한 구조화된 프레젠테이션은 텍스트 설명보다 구성 세부 사항을 전달하는 데 훨씬 더 효과적이며, 사용자가 코드나 긴 설명을 일일이 분석할 필요 없이 자신의 특정 요구 사항과 관련된 설정을 신속하게 식별하고 조정할 수 있도록 합니다. 이는 사용성을 향상시키고 사용자 정의 프로세스를 가속화합니다.

매개변수 이름	목적	기본값	가능한 옵션/설명
<code>OPENAI_API_KEY</code> , <code>TAUVILY_API_KEY</code>	LLM 및 검색 엔진 인증	{Your API Key}	환경 변수 또는 .env 파일에 설정
<code>llm_provider</code>	LLM 공급자 지정	"openai"	Langchain 지원 LLM (예: "huggingface", "cohere")
<code>fast_llm_model</code>	빠르고 덜 복잡한 작업용 LLM	"gpt-3.5-turbo"	다른 빠른 모델

smart_llm_model	복잡하고 심층적인 작업용 LLM	"gpt-4o"	다른 스마트 모델
retriever	실시간 온라인 정보 검색 엔진	"tavily"	"duckduckgo", "googleAPI", "googleSerp", "searx", "bing" 등 (일부 API 키 필요)
report_format	생성될 보고서의 형식	"apa"	사용자 정의 가능한 보고서 형식
total_words	보고서의 총 단어 수	1000	원하는 단어 수로 조정
max_iterations	연구 프로세스의 깊이	1	더 깊은 연구를 위해 조정
fast_token_limit	빠른 LLM의 토큰 제한	2000	토큰 사용량 관리
smart_token_limit	스마트 LLM의 토큰 제한	4000	토큰 사용량 관리
summary_token_limit	요약 토큰 제한	700	요약 토큰 사용량 관리
temperature	LLM 출력의 무작위성 제어	0.6	창의성 조절 (높을수록 무작위)
user_agent	웹 브라우징 요청을 위한 사용자 에이전트 문자열	"Mozilla/5.0 (...)"	필요에 따라 변경
memory_backend	에이전트 메모리 저장 방식	"local"	(현재 "local"만 지원)
browse_chunk_max_length	웹 콘텐츠 브라우징 시 청크의 최대 길이	8192	필요에 따라 조정
config_file	외부 JSON 설정 파일 경로	None	사용자 정의 JSON 파일 경로
source_urls	연구할 특정 URL 목록	None	URL 리스트 (예: ["url1", "url2"])
report_type (custom)	사용자 정의 보고서 유형	"research_report"	"custom_report"로 설정 시 query가 프롬프트로 활용

5. 활용 예시 및 시나리오

GPT-Researcher는 다양한 실용적인 시나리오에서 사용될 수 있으며, 여러 애플리케이션에 통합될 수 있는 자율 에이전트입니다.

5.1. 독립형 에이전트로서의 활용 (Utilizing as a Standalone Agent)

GPT-Researcher는 기존의 어떤 Python 프로젝트에도 독립형 에이전트로 임포트되어 연구 보고서를 생성할 수 있습니다. 예를 들어, "최근 버닝맨 홍수에서 무슨 일이 일어났는가?"와 같은 쿼리에 대한 연구 보고서를 가져올 수 있습니다. 생성된 보고서는 뉴스 기사, 마케팅 콘텐츠, 이메일 템플릿, 뉴스레터 등 가치 있는 콘텐츠를 생성하기 위한 맥락으로 활용될 수 있습니다. 또한 코드 문서, 비즈니스 분석, 금융 정보 등 사실적이고 고품질의 실시간 정보가 필요한 복잡한 작업을 완료하기 위한 정보를 수집하는 데에도 사용될 수 있습니다. GPT-Researcher의 출력을 다른 생성 작업의 입력으로 사용하는 것에 대한 강조는

GPT-Researcher가 더 큰 콘텐츠 생성 및 지식 관리 시스템에서 핵심 구성 요소로서의 역할을 수행함을 강조합니다. 이는 단순한 보고서 생성기가 아니라 "지식 기반 구축자" 또는 "콘텐츠 공급자"로서, 기업이 콘텐츠 생성 워크플로우를 자동화하거나 증강하려는 경우 중요한 미들웨어로 자리매김합니다.

5.2. 다중 에이전트 프레임워크 통합 (Integration with Multi-Agent Frameworks)

GPT-Researcher는 복잡한 작업을 처리하기 위해 자율 에이전트들이 협력하는 다중 에이전트 프레임워크에 통합될 수 있습니다. 특히 심층 연구가 필요한 작업에 유용합니다. LangGraph와 통합된 편집 에이전트 팀은 주어진 작업에 대한 연구 보고서를 완성하기 위해 협력하는 대표적인 예시입니다. 이 팀은 총괄 편집자가 연구 프로세스를 감독하고, 연구원이 심층 연구를 수행하고 초안을 작성하며, 작성자가 최종 보고서를 편집하고 작성합니다. 검토자가 초안의 정확성을 검증하고 피드백을 제공하며, 수정자가 이 피드백을 기반으로 초안을 수정합니다. 마지막으로 출판자가 최종 보고서를 다양한 형식으로 출판합니다.

이러한 접근 방식은 금융, 비즈니스 분석, 의료, 마케팅, 법률 등 다양한 분야에 적용될 수 있습니다. 예를 들어, 코딩 작업을 담당하는 코딩 에이전트는 기술 설계를 작성하기 전에 최신 및 관련 API 문서와 모범 사례를 큐레이션하기 위해 연구 에이전트(GPT-Researcher)를 통합할 수 있습니다. 다중 에이전트 프레임워크, 특히 "편집 에이전트 팀"은 AI 시스템이 인간 조직 구조와 워크플로우를 모방하도록 어떻게 진화하고 있는지를 보여주는 심오한 예시입니다. 이는 단순히 병렬 처리가 아니라, 전문화된 역할과 순차적이고 반복적인 프로세스를 통한 노동 분업입니다. 이 설계 선택은 편향되지 않은 연구 보고서와 같은 복잡하고 고품질의 결과물이 인간 팀처럼 내부 검증을 포함하는 협력적이고 다단계적인 프로세스를 통해 가장 잘 달성될 수 있다는 믿음을 내포합니다. 이는 AI 에이전트가 단순히 작업을 수행하는 것을 넘어, 단일 AI의 범위를 넘어서는 문제를 해결하기 위해 "디지털 조직"을 형성하는 미래를 시사합니다. 다양한 분야에 걸친 적용은 복잡한 도메인별 문제 해결을 위한 광범위한 유용성을 강조합니다.

5.3. 웹 프레임워크 통합 (Integration with Web Frameworks: FastAPI, Flask)

GPT-Researcher는 FastAPI 및 Flask와 같은 인기 있는 웹 프레임워크와 통합될 수 있습니다. FastAPI 예시는 query와 report_type을 받아 생성된 보고서를 반환하는 엔드포인트(/report/{report_type})를 생성하는 방법을 보여줍니다. Flask와의 유사한 통합도 가능하며, 이를 위해서는 flask[async] 설치가 필요합니다. FastAPI 및 Flask와 같은 인기 있는 웹 프레임워크와의 통합은 GPT-Researcher를 로컬 도구에서 배포 가능한 서비스로 전환하는 데 중요합니다. 이를 통해 조직은 API를 통해 연구 기능을 노출하여 다른 애플리케이션이나 사용자가 프로그래밍 방식으로 연구 보고서를 요청할 수 있도록 합니다. 이는 "서비스형 연구(Research-as-a-Service)"를 향한 단계로, 기업 전체 또는 상업적 제공을 위해 그 기능에 대한 확장 가능한 접근을 용이하게 합니다. 이는 GPT-Researcher를 데스크톱 유틸리티에서 더 크고 분산된 시스템의 백엔드 구성 요소로 전환시킵니다.

6. 문제 해결

GPT-Researcher를 사용하는 동안 발생할 수 있는 일반적인 문제와 그 해결책을 이해하는 것은 중요합니다.

6.1. 일반적인 문제 및 해결책 (Common Issues and Solutions)

문제가 발생하면 먼저 Discord 커뮤니티 에서 해결된 문제를 확인하거나 도움을 요청하는 것이 좋습니다. 이는 오픈 소스 프로젝트의 특성과 커뮤니티 기반 지원 모델을 강조합니다.

- **"Model: gpt-4 does not exist" 오류:** 이 오류는 gpt-4를 사용할 권한이 아직 없을 수 있음을 나타냅니다. OpenAI는 7월 말까지 광범위하게 사용 가능할 것으로 예상합니다.
- **"Error processing the url" (Selenium 문제):** GPT-Researcher는 웹 스크래핑에 Selenium을 사용하며, 일부 사이트는 스크래핑에 실패할 수 있습니다. 애플리케이션을 다시 시작하고 다시 시도하십시오. 이 문제는 웹 스크래핑의 본질적인 어려움과 외부 브라우저 구성 요소에 대한 의존성을 지적하며, 사용자가 이러한 외부 요인을 관리해야 함을 의미합니다.
- **Chrome 버전 문제:** 호환되지 않는 Chrome 브라우저 및 Chromedriver 버전은 문제를 일으킬 수 있습니다.
- Chrome 웹 브라우저를 다운그레이드하려면 [slimjet](#)을 방문하여 운영 체제와 호환되는 이전 Chrome 버전을 찾으십시오.
- 충돌을 피하기 위해 이전 버전을 설치하기 전에 현재 Chrome 버전을 제거하십시오.
- 가장 중요하게, 다운그레이드하는 Chrome 버전이 공식 [chromedriver 웹사이트](#)에서 호환되는 Chromedriver를 사용할 수 있는지 확인하십시오.

6.2. 특정 오류 메시지 해결 (Resolving Specific Error Messages)

- **"Cannot load library 'gobject-2.0-0'" 오류:** 이 문제는 연구 보고서에서 PDF를 생성하는 데 사용되는 WeasyPrint 라이브러리와 관련이 있습니다.
- https://doc.courtbouillon.org/weasyprint/stable/first_steps.html의 가이드를 따르거나 패키지를 수동으로 설치하여 해결하십시오.
- **MacOS의 경우:** brew install glib gobject-introspection.
- **Linux의 경우:** sudo apt install libglib2.0-dev.
- **"Cannot load library 'pango'" 오류:**
- **MacOS의 경우:** brew install pango.
- **Linux의 경우:** sudo apt install libpango-1.0-0.

이러한 특정 문제 해결 단계는 시스템 수준 라이브러리, 특히 PDF 생성을 위해 사용되는 WeasyPrint에 대한 의존성을 보여줍니다. 이는 GPT-Researcher가 Python 기반이지만, 풍부한 출력 기능(예: PDF 내 보내기)이 기본 OS 구성 요소에 의존한다는 것을 나타냅니다. 이는 크로스 플랫폼 애플리케이션에서 흔히 발생하는 문제이며, 이러한 외부 라이브러리에 대한 상세한 OS별 설치 지침이 필요합니다.

6.3. Mac M 칩 사용자 워크어라운드 (Workaround for Mac M Chip Users)

위 해결책이 작동하지 않는 경우, 다음 단계를 따르십시오:

1. brew를 가리키는 Python 3.11의 새 버전을 설치합니다: `brew install python@3.11`.
2. 필요한 라이브러리를 설치합니다: `brew install pango glib gobject-introspection`.
3. 필요한 GPT Researcher Python 패키지를 설치합니다: `pip3.11 install -r requirements.txt`.
4. Python 3.11(brew 사용)으로 앱을 실행합니다: `python3.11 -m uvicorn main:app --reload`.

Mac M 칩 사용자를 위한 전용 해결책은 새로운 하드웨어 아키텍처와 기존 소프트웨어 생태계와의 호환성으로 인해 발생하는 문제에 대한 직접적인 대응입니다. 이는 광범위한 플랫폼 지원과 일반적인 사용자 문제점에 대한 개발자의 대응성을 보여줍니다. 이는 Python의 크로스 플랫폼 특성에도 불구하고, 기본 네이티브 의존성(예: WeasyPrint 또는 특정 Python 빌드)이 여전히 새로운 칩셋에서 문제를 일으킬 수 있으며, 최적의 성능을 위해 특정 지침이 필요함을 강조합니다.

6.4. 표: 일반적인 문제 해결 가이드

문제 해결은 모든 기술 매뉴얼의 중요한 부분입니다. 다음 표는 일반적인 문제와 그 해결책을 요약하여 사용자에게 문제를 진단하고 해결하는 데 효율적이고 사용자 친화적인 방법을 제공합니다. 사용자는 긴 문단을 검색하는 대신, 특정 오류나 증상을 빠르게 찾아 해당 해결책을 찾을 수 있습니다. 이러한 구조화된 접근 방식은 설정 및 작동 중 사용자 경험을 크게 향상시키고, 좌절감을 줄이며, 문제 해결 시간을 단축합니다. 이는 여러 외부 의존성에 의존하는 도구에 있어 가장 중요합니다.

문제/오류 메시지	증상	해결책/워크어라운드	관련 스니펫 ID
일반적인 문제	특정 오류 메시지 없이 작동하지 않음	Discord 커뮤니티 확인	
"Model: gpt-4 does not exist"	gpt-4 모델 사용 불가	gpt-4 사용 권한 확인 (OpenAI 정책)	
"Cannot load library 'gobject-2.0-0'"	PDF 생성 실패	WeasyPrint 가이드 따르기; MacOS: <code>brew install glib gobject-introspection</code> ; Linux: <code>sudo apt install libglib2.0-dev</code>	
"Cannot load library 'pango'"	PDF 생성 실패	MacOS: <code>brew install pango</code> ; Linux: <code>sudo apt install libpango-1.0-0</code>	
"Error processing the url"	웹 스크래핑 실패	애플리케이션 재시작 후 재시도	
Chrome/Chromedriver 버전 불일치	웹 스크래핑 실패, 브라우저 관련 오류	Chrome 다운그레이드 및 호환되는 Chromedriver 설치	
Mac M 칩 성능/설치 문제	특정 라이브러리 로드 오류 또는 비정상 작동	brew로 Python 3.11 재설치, 필요한 라이브러리 설치, pip3.11 및 python3.11로 실행	

7. 결론

GPT-Researcher는 상세하고, 사실적이며, 편향되지 않은 연구 보고서를 인용과 함께 제공하도록 설계된 혁신적인 오픈 소스 심층 연구 에이전트입니다. 이는 정보의 오용 및 LLM의 한계와 같은 현대 정보 환경의 주요 과제를 해결합니다.

본 매뉴얼에서 살펴보았듯이, GPT-Researcher는 표준 설치, PIP 패키지, Docker를 포함한 유연한 설치 옵션을 제공하며, LLM, 검색 엔진, 보고서 형식, 연구 깊이 등 광범위한 사용자 정의가 가능합니다. 특히, Chief Editor, Researcher, Editor, Reviewer, Revisor, Writer, Publisher로 구성된 다중 에이전트 아키텍처는 인간의 편집 프로세스를 모방하여 보고서의 신뢰성과 품질을 크게 향상시킵니다.

로컬 문서나 특정 웹 소스를 기반으로 연구를 수행하는 기능은 GPT-Researcher의 적용 범위를 확장하며, 독립형 에이전트로서의 활용부터 복잡한 AI 프레임워크와의 통합, 웹 프레임워크를 통한 서비스 제공까지 다양한 시나리오에서 그 가치를 발휘합니다. 또한, 발생할 수 있는 일반적인 문제에 대한 포괄적인 문제 해결 가이드는 사용자가 원활하게 도구를 사용할 수 있도록 지원합니다.

GPT-Researcher는 더 신뢰할 수 있고 정교한 AI 기반 정보 합성을 향한 중요한 발걸음을 나타냅니다. 고급 LLM 기능과 구조화된 연구 방법론(다중 에이전트 프레임워크, RAG 원칙) 및 강력한 웹 스크래핑을 결합함으로써, 단순한 생성형 AI를 넘어 검증 가능한 고품질 연구를 제공합니다. 배포 및 사용자 정의의 유연성은 개인과 조직이 중요한 정보 수집, 콘텐츠 생성 및 지식 관리를 위해 AI를 활용하려는 경우 다재다능한 도구로 자리매김하며, 궁극적으로 점점 더 복잡해지는 정보 환경에서 사용자에게 정확하고 편향되지 않은 통찰력을 제공합니다. 이 도구는 단순히 무엇을 연구하는지가 아니라, *어떻게* 연구하는지에 중점을 두어 정확성과 검증 가능성을 강조함으로써 AI 기반 정보 합성의 패러다임 전환을 이끌어낼 것입니다.

참고 자료

1. assafelovic/gpt-researcher: LLM based autonomous agent ...

- GitHub, <https://github.com/assafelovic/gpt-researcher>

2. GPT Researcher

- Tavily Docs, <https://docs.tavily.com/examples/open-sources/gpt-researcher>