
- 출처: LangChain 공식 문서 또는 해당 교재명
- 원본 URL: https://smith.langchain.com/hub/teddynote/summary-stuff-documents

---

`Arxiv`

- **`arXiv`**
  - 물리학, 수학, 컴퓨터 과학, 정량 생물학, 정량 금융, 통계, 전기공학 및 시스템 과학, 경제학 분야의 200만 편의 학술 논문을 위한 오픈 액세스 아카이브
  - `Arxiv` 문서 로더에 접근하려면 `arxiv`, `PyMuPDF` 및 `langchain-community` 통합 패키지 설치 필요
- **`PyMuPDF`**: `arxiv.org` 사이트에서 다운로드한 `PDF 파일`을 **`텍스트 형식`**으로 `변환`
- 공식 도큐먼트

```
# API KEY를 환경변수로 관리하기 위한 설정 파일
import os
from dotenv import load_dotenv

# API KEY 정보로드
load_dotenv()               # true
```

- 사전에 `VS Code` 터미널에 설치할 것

```
pip install -qU langchain-community arxiv pymupdf
```

`객체 생성`

- 이제 `model` 객체를 인스턴스화 및 문서 로드 가능

```
from langchain_community.document_loaders import ArxivLoader     # ArxivLoader 임포트
from langchain_core.documents import Document
import fitz                                                      # fitz 모듈 임포트

# try-except 블록을 사용하여 오류 발생 시 처리
try:
    # ArxivLoader 인스턴스 생성
    # Query에 검색하고자 하는 논문의 주제 입력
    loader = ArxivLoader(
        query="Chain of thought",           # 검색할 논문의 주제
        load_max_docs=2,                    # 최대 문서 수
        load_all_available_meta=True,       # 메타데이터 전체 로드 여부
    )

    # 문서 로드
    docs = loader.load()

    # 로드된 문서 목록 출력
    if docs:
        print(f"총 {len(docs)}개의 논문을 로드했습니다.\n")

        for i, doc in enumerate(docs):
            if isinstance(doc, Document):
                print(f"--- 논문 {i+1} ---")
                print(f"제목: {doc.metadata.get('Title', '제목 없음')}")
                print(f"저자: {doc.metadata.get('Authors', '저자 정보 없음')}")
                print(f"요약 (일부):\n{doc.page_content[:300]}...\n")
            else:
```

```
                print(f"--- 논문 {i+1} (로드 실패) ---")
                print(doc)
    else:
        print("조건에 맞는 논문을 찾지 못했습니다.")

except Exception as e:
    print(f"논문 로드 중 오류가 발생했습니다: {e}")
    print("\n[해결 제안]")
    print("1. 검색어가 정확한지 확인해 주세요.")
    print("2. 네트워크 상태를 확인하고, 방화벽이나 프록시 설정이 논문 다운로드를 방해하지 않는지 확인해 주세요.")
    print("3. 'pip install -U PyMuPDF' 명령어를 실행하여 PyMuPDF를 최신 버전으로 업데이트해 보세요.")
    print("4. 그래도 문제가 해결되지 않으면, 'ArxivLoader' 대신 'ArxivAPIWrapper'를 사용하여 논문 정보를 먼저 확인해 볼 수 있습니다.")
```

- 셀 출력 (1.2s)

```
논문 로드 중 오류가 발생했습니다: module 'fitz' has no attribute 'fitz'

[해결 제안]
1. 검색어가 정확한지 확인해 주세요.
2. 네트워크 상태를 확인하고, 방화벽이나 프록시 설정이 논문 다운로드를 방해하지 않는지 확인해 주세요.
3. 'pip install -U PyMuPDF' 명령어를 실행하여 PyMuPDF를 최신 버전으로 업데이트해 보세요.
4. 그래도 문제가 해결되지 않으면, 'ArxivLoader' 대신 'ArxivAPIWrapper'를 사용하여 논문 정보를 먼저 확인해 볼 수 있습니다.
```

- 오류 원인과 해결 방법

```
- 오류 원인: **`langchain_community.document_loader.arxiv`모듈과 `PyMuPDF`라이브러리 버전의 호환성 충돌 문제일 가능성 높음**
- `langchain_community`의 `ArxivLoader` 코드가 오래된 버전의 `PyDF`(모듈 이름:`fitz`)를 기반으로 작성되었기 때문
- `ArxivLoader`
- 내부적으로 `try-except fitz.fitz.FileDataError`와 같은 코드를 사용
- `fitz`모듈 아래에 `fitz`라는 속성(`attribute`)이 더 이상 존재하지 않기 때문

- 해결 방법: **`ArxivLoader`** 의 코드를 수정해야 함
- `ArxivLoader`의 소스 코드를 직접 수정하는 것은 어려움
- `langchain-community` 라이브러리가 이 문제를 해결하기 전까지는 `ArxivLoader`를 직접 사용하지 않고 `ArxivAPIWrapper`를 사용하여 논
- `ArxivAPIWrapper`
    - **논문 검색 결과 자체를 반환**
    - **`PDF` 파일을 `직접 다운로드`하고 `파싱`하는 로직을 직접 구현해야 함**
```

`ArxivAPIWrapper`로 시도

```
from langchain_community.utilities import ArxivAPIWrapper
import requests
import fitz

# 검색할 논문의 주제
query_topic = "Chain of thought"

# 최대 문서 수
max_docs_to_load = 2

# ArxivAPIWrapper 인스턴스 생성
arxiv_wrapper = ArxivAPIWrapper()

try:
    # 1. Arxiv에서 논문 검색
    print(f"'{query_topic}' 주제로 논문을 검색 중입니다...")
    search_results = arxiv_wrapper.run(query=query_topic)

    # ArxivAPIWrapper의 run 메서드는 문자열을 반환하므로, 원하는 메타데이터 추출이 어려움
    # 따라서 arxiv 패키지를 직접 사용하기
    import arxiv
    search = arxiv.Search(
        query=query_topic,
        max_results=max_docs_to_load,
        sort_by=arxiv.SortCriterion.Relevance
    )

    docs = []

    # 2. 검색된 논문 목록을 순회하며 PDF 다운로드 및 파싱
    for i, result in enumerate(arxiv.Client().results(search)):
```

```
            try:
                # PDF 다운로드 URL
                pdf_url = result.pdf_url
                print(f"\n{i+1}/{max_docs_to_load} 논문 다운로드 중: {pdf_url}")

                # 논문 PDF 다운로드
                response = requests.get(pdf_url)
                response.raise_for_status()                 # HTTP 오류가 발생하면 예외 발생

                # 다운로드된 PDF 데이터를 메모리에서 파싱
                with fitz.open(stream=response.content, filetype="pdf") as doc_file:
                    # PDF 페이지의 텍스트를 모두 추출
                    text_content = "".join(page.get_text() for page in doc_file)

                    # 추출된 정보를 Document 객체로 저장
                    from langchain.docstore.document import Document
                    doc = Document(
                        page_content=text_content,
                        metadata={
                            "Title": result.title,
                            "Authors": ", ".join(author.name for author in result.authors),
                            "Published": result.published,
                            "pdf_url": pdf_url,
                        }
                    )
                    docs.append(doc)

                print(f"성공적으로 로드된 논문: '{result.title}'")

            except requests.exceptions.HTTPError as errh:
                print(f"HTTP 오류: {errh}")
                print(f"'{result.title}' 논문 다운로드 실패. 다른 논문을 시도합니다.")
                continue
            except Exception as e:
                print(f"논문 처리 중 오류 발생: {e}")
                print(f"'{result.title}' 논문 처리 실패. 다른 논문을 시도합니다.")
                continue

        # 3. 로드된 문서 목록 출력
        if docs:
            print("\n=== 최종 로드된 문서 목록 ===")
            for i, doc in enumerate(docs):
                print(f"--- 논문 {i+1} ---")
                print(f"제목: {doc.metadata.get('Title', '제목 없음')}")
                print(f"저자: {doc.metadata.get('Authors', '저자 정보 없음')}")
                print(f"요약 (일부):\n{doc.page_content[:300]}...\n")
        else:
            print("\n조건에 맞는 논문을 찾지 못했습니다.")

    except Exception as e:
        print(f"전체 프로세스 중 오류가 발생했습니다: {e}")
        print("\n[해결 제안]")
        print("1. 검색어가 정확한지 확인해 주세요.")
        print("2. 네트워크 상태를 확인하고, 방화벽이나 프록시 설정이 논문 다운로드를 방해하지 않는지 확인해 주세요.")
        print("3. 필요한 라이브러리(requests, PyMuPDF, arxiv)가 모두 설치되었는지 확인해 주세요.")
        print("   'pip install requests PyMuPDF arxiv'")
```

- 셀 출력 (2.3s)

```
'Chain of thought' 주제로 논문을 검색 중입니다...

1/2 논문 다운로드 중: http://arxiv.org/pdf/2311.09277v1
성공적으로 로드된 논문: 'Contrastive Chain-of-Thought Prompting'

2/2 논문 다운로드 중: http://arxiv.org/pdf/2305.16582v2
성공적으로 로드된 논문: 'Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in Language Models'

=== 최종 로드된 문서 목록 ===
--- 논문 1 ---
제목: Contrastive Chain-of-Thought Prompting
저자: Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, Lidong Bing
요약 (일부):
Contrastive Chain-of-Thought Prompting
Yew Ken Chia*1,
Guizhen Chen*1, 2
Luu Anh Tuan2
```

```
Soujanya Poria
Lidong Bing† 1
1DAMO Academy, Alibaba Group, Singapore
Singapore University of Technology and Design
2Nanyang Technological University, Singapore
{yewken_chia, sporia}@sutd.edu.sg
{guizhen001, anhtu...

--- 논문 2 ---
제목: Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in Language Models
저자: Yao Yao, Zuchao Li, Hai Zhao
요약 (일부):
Beyond Chain-of-Thought, Effective Graph-of-Thought Reasoning in
Language Models
Yao Yao1,2, Zuchao Li3,∗and Hai Zhao1,2,∗
1Department of Computer Science and Engineering, Shanghai Jiao Tong University
2MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
3National Eng...
```

메타데이터: 전체 출력, 부분 출력

- 전체 메타데이터 출력하기

```
# 관련 코드
# ...중략...
        # 모든 메타데이터를 저장
        full_metadata = {
            "Title": result.title,
            "Authors": ", ".join(author.name for author in result.authors),
            "Published": result.published,
            "Updated": result.updated,
            "Summary": result.summary,
            "Categories": ", ".join(result.categories),
            "Entry ID": result.entry_id,
            "DOI": result.doi,
            "pdf_url": pdf_url,
        }

        doc = Document(
            page_content=text_content,
            metadata=full_metadata
        )
        docs.append(doc)
    print(f"성공적으로 로드된 논문: '{result.title}'")
```

- `PyMuPDF` 에서의 방법

```
loader = ArxivLoader(
    query="ChatGPT",
    load_max_docs=2,                    # 최대 문서 수
    load_all_available_meta=True,       # 메타데이터 전체 로드
)
```

```
# 메타데이터 전체 출력하기_1
# 다른 주제: ChatGPT

from langchain_community.utilities import ArxivAPIWrapper
import requests
import fitz
from langchain.docstore.document import Document
import arxiv
```

```
# 검색할 논문의 주제
query_topic = "ChatGPT"

# 최대 문서 수
max_docs_to_load = 2

# ArxivAPIWrapper 인스턴스 생성
arxiv_wrapper = ArxivAPIWrapper()

try:
    print(f"'{query_topic}' 주제로 논문을 검색 중입니다...")

    search = arxiv.Search(
        query=query_topic,
        max_results=max_docs_to_load,
        sort_by=arxiv.SortCriterion.Relevance
    )

    docs = []

    for i, result in enumerate(arxiv.Client().results(search)):
        try:
            pdf_url = result.pdf_url
            response = requests.get(pdf_url)
            response.raise_for_status()

            with fitz.open(stream=response.content, filetype="pdf") as doc_file:
                text_content = "".join(page.get_text() for page in doc_file)

                # 모든 메타데이터를 저장
                full_metadata = {
                    "Title": result.title,
                    "Authors": ", ".join(author.name for author in result.authors),
                    "Published": result.published,
                    "Updated": result.updated,
                    "Summary": result.summary,
                    "Categories": ", ".join(result.categories),
                    "Entry ID": result.entry_id,
                    "DOI": result.doi,
                    "pdf_url": pdf_url,
                }

                doc = Document(
                    page_content=text_content,
                    metadata=full_metadata
                )
                docs.append(doc)
            print(f"성공적으로 로드된 논문: '{result.title}'")

        except requests.exceptions.HTTPError as errh:
            print(f"HTTP 오류: {errh}")
            continue
        except Exception as e:
            print(f"논문 처리 중 오류 발생: {e}")
            continue

    if docs:
        print("\n=== 로드된 문서 목록 (전체 메타데이터) ===")
        for i, doc in enumerate(docs):
            print(f"--- 논문 {i+1} ---")
            for key, value in doc.metadata.items():
                print(f"{key}: {value}")
            print("-" * 20)
    else:
        print("\n조건에 맞는 논문을 찾지 못했습니다.")

except Exception as e:
    print(f"전체 프로세스 중 오류가 발생했습니다: {e}")
```

- 셀 출력 (5.4s)

```
'ChatGPT' 주제로 논문을 검색 중입니다...
성공적으로 로드된 논문: 'In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT'
성공적으로 로드된 논문: 'Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text'

=== 로드된 문서 목록 (전체 메타데이터) ===
```

```
--- 논문 1 ---
Title: In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT
Authors: Xinyue Shen, Zeyuan Chen, Michael Backes, Yang Zhang
Published: 2023-04-18 13:20:45+00:00
Updated: 2023-10-05 13:27:12+00:00
Summary: The way users acquire information is undergoing a paradigm shift with the
advent of ChatGPT. Unlike conventional search engines, ChatGPT retrieves
knowledge from the model itself and generates answers for users. ChatGPT's
impressive question-answering (QA) capability has attracted more than 100
million users within a short period of time but has also raised concerns
regarding its reliability. In this paper, we perform the first large-scale
measurement of ChatGPT's reliability in the generic QA scenario with a
carefully curated set of 5,695 questions across ten datasets and eight domains.
We find that ChatGPT's reliability varies across different domains, especially
underperforming in law and science questions. We also demonstrate that system
roles, originally designed by OpenAI to allow users to steer ChatGPT's
behavior, can impact ChatGPT's reliability in an imperceptible way. We further
show that ChatGPT is vulnerable to adversarial examples, and even a single
character change can negatively affect its reliability in certain cases. We
believe that our study provides valuable insights into ChatGPT's reliability
and underscores the need for strengthening the reliability and security of
large language models (LLMs).
Categories: cs.CR, cs.LG
Entry ID: http://arxiv.org/abs/2304.08979v2
DOI: None
pdf_url: http://arxiv.org/pdf/2304.08979v2
--------------------
--- 논문 2 ---
Title: Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text
Authors: Lingyi Yang, Feng Jiang, Haizhou Li
Published: 2023-07-21 06:38:37+00:00
Updated: 2023-12-30 13:17:52+00:00
Summary: The remarkable capabilities of large-scale language models, such as ChatGPT,
in text generation have impressed readers and spurred researchers to devise
detectors to mitigate potential risks, including misinformation, phishing, and
academic dishonesty. Despite this, most previous studies have been
predominantly geared towards creating detectors that differentiate between
purely ChatGPT-generated texts and human-authored texts. This approach,
however, fails to work on discerning texts generated through human-machine
collaboration, such as ChatGPT-polished texts. Addressing this gap, we
introduce a novel dataset termed HPPT (ChatGPT-polished academic abstracts),
facilitating the construction of more robust detectors. It diverges from extant
corpora by comprising pairs of human-written and ChatGPT-polished abstracts
instead of purely ChatGPT-generated texts. Additionally, we propose the "Polish
Ratio" method, an innovative measure of the degree of modification made by
ChatGPT compared to the original human-written text. It provides a mechanism to
measure the degree of ChatGPT influence in the resulting text. Our experimental
results show our proposed model has better robustness on the HPPT dataset and
two existing datasets (HC3 and CDB). Furthermore, the "Polish Ratio" we
proposed offers a more comprehensive explanation by quantifying the degree of
ChatGPT involvement.
Categories: cs.CL
Entry ID: http://arxiv.org/abs/2307.11380v2
DOI: None
pdf_url: http://arxiv.org/pdf/2307.11380v2
--------------------
```

- 전체 메타데이터 출력하기

```
    # 관련 코드
    # ...중략...
            # 메타데이터 일부를 저장
            doc = Document(
                page_content=text_content,
```

```
                metadata={
                    "Title": result.title,
                    "Authors": ", ".join(author.name for author in result.authors),
                    "Published": result.published,
                    "pdf_url": pdf_url,
                }
            )
            docs.append(doc)
```

- `PyMuPDF`에서의 방법

```
    loader = ArxivLoader(
        query="ChatGPT",
        load_max_docs=2,                    # 최대 문서 수
        load_all_available_meta=False,      # 메타데이터 전체 로드
    )
```

```
# 메타데이터 전체 출력하기_2
# 다른 주제: ChatGPT

from langchain_community.utilities import ArxivAPIWrapper
import requests
import fitz
from langchain.docstore.document import Document
import arxiv

# 검색할 논문의 주제
query_topic = "ChatGPT"

# 최대 문서 수
max_docs_to_load = 2

# ArxivAPIWrapper 인스턴스 생성
arxiv_wrapper = ArxivAPIWrapper()

try:
    print(f"'{query_topic}' 주제로 논문을 검색 중입니다...")

    search = arxiv.Search(
        query=query_topic,
        max_results=max_docs_to_load,
        sort_by=arxiv.SortCriterion.Relevance
    )

    docs = []

    for i, result in enumerate(arxiv.Client().results(search)):
        try:
            pdf_url = result.pdf_url
            response = requests.get(pdf_url)
            response.raise_for_status()

            with fitz.open(stream=response.content, filetype="pdf") as doc_file:
                text_content = "".join(page.get_text() for page in doc_file)

                doc = Document(
                    page_content=text_content,
                    metadata={
                        "Title": result.title,
                        "Authors": ", ".join(author.name for author in result.authors),
                        "Published": result.published,
                        "pdf_url": pdf_url,
                    }
                )
                docs.append(doc)
            print(f"성공적으로 로드된 논문: '{result.title}'")

        except requests.exceptions.HTTPError as errh:
            print(f"HTTP 오류: {errh}")
            continue
        except Exception as e:
            print(f"논문 처리 중 오류 발생: {e}")
            continue
```

```
    if docs:
        print("\n=== 로드된 문서 목록 (간결한 메타데이터) ===")
        for i, doc in enumerate(docs):
            print(f"--- 논문 {i+1} ---")
            print(f"제목: {doc.metadata.get('Title', '제목 없음')}")
            print(f"저자: {doc.metadata.get('Authors', '저자 정보 없음')}")
            print(f"URL: {doc.metadata.get('pdf_url', 'URL 없음')}")
            print("-" * 20)
    else:
        print("\n조건에 맞는 논문을 찾지 못했습니다.")

except Exception as e:
    print(f"전체 프로세스 중 오류가 발생했습니다: {e}")
```

- 셀 출력 (1.1s)

```
'ChatGPT' 주제로 논문을 검색 중입니다...
성공적으로 로드된 논문: 'In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT'
성공적으로 로드된 논문: 'Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text'

=== 로드된 문서 목록 (간결한 메타데이터) ===
--- 논문 1 ---
제목: In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT
저자: Xinyue Shen, Zeyuan Chen, Michael Backes, Yang Zhang
URL: http://arxiv.org/pdf/2304.08979v2
--------------------
--- 논문 2 ---
제목: Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text
저자: Lingyi Yang, Feng Jiang, Haizhou Li
URL: http://arxiv.org/pdf/2307.11380v2
--------------------
```

요약(Summaray)

- 논문의 전체 내용이 아닌 요약본을 출력시 → `get_summaries_as_docs()` 함수 호출

```
from langchain_community.document_loaders import ArxivLoader
from langchain_community.utilities import ArxivAPIWrapper
from langchain_core.documents import Document

# 검색할 논문의 주제
query_topic = "ChatGPT"

# 최대 문서 수
max_docs_to_load = 2

# ArxivAPIWrapper 인스턴스 생성
arxiv_wrapper = ArxivAPIWrapper()

try:
    print(f"'{query_topic}' 주제로 논문을 검색 중입니다...")

    # ArxivLoader를 사용해 요약본을 문서로 로드
    # get_summaries_as_docs() 함수는 논문 초록을 가져와 Document 객체로 반환
    docs = arxiv_wrapper.get_summaries_as_docs(query=query_topic)

    if docs:
        print("\n=== 요약된 문서 목록 ===")
        for i, doc in enumerate(docs):
            if isinstance(doc, Document):
                print(f"--- 문서 {i+1} ---")
                print(f"제목: {doc.metadata.get('Title', '제목 없음')}")
                print(f"저자: {doc.metadata.get('Authors', '저자 정보 없음')}")
                print(f"요약:\n{doc.page_content}\n")
                print("-" * 20)
            else:
                print(f"--- 문서 {i+1} (로드 실패) ---")
                print(doc)
    else:
        print("\n조건에 맞는 논문을 찾지 못했습니다.")
```

```
except Exception as e:
    print(f"논문 요약 중 오류가 발생했습니다: {e}")
    print("\n[해결 제안]")
    print("1. 검색어가 정확한지 확인해 주세요.")
    print("2. 네트워크 상태를 확인하고, 방화벽이나 프록시 설정이 논문 다운로드를 방해하지 않는지 확인해 주세요.")
```

- 셀 출력 (0.1s)

```
'ChatGPT' 주제로 논문을 검색 중입니다...

=== 요약된 문서 목록 ===
--- 문서 1 ---
제목: In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT
저자: Xinyue Shen, Zeyuan Chen, Michael Backes, Yang Zhang
요약:
The way users acquire information is undergoing a paradigm shift with the
advent of ChatGPT. Unlike conventional search engines, ChatGPT retrieves
knowledge from the model itself and generates answers for users. ChatGPT's
impressive question-answering (QA) capability has attracted more than 100
million users within a short period of time but has also raised concerns
regarding its reliability. In this paper, we perform the first large-scale
measurement of ChatGPT's reliability in the generic QA scenario with a
carefully curated set of 5,695 questions across ten datasets and eight domains.
We find that ChatGPT's reliability varies across different domains, especially
underperforming in law and science questions. We also demonstrate that system
roles, originally designed by OpenAI to allow users to steer ChatGPT's
behavior, can impact ChatGPT's reliability in an imperceptible way. We further
show that ChatGPT is vulnerable to adversarial examples, and even a single
character change can negatively affect its reliability in certain cases. We
believe that our study provides valuable insights into ChatGPT's reliability
and underscores the need for strengthening the reliability and security of
large language models (LLMs).

--------------------
--- 문서 2 ---
제목: Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text
저자: Lingyi Yang, Feng Jiang, Haizhou Li
요약:
The remarkable capabilities of large-scale language models, such as ChatGPT,
in text generation have impressed readers and spurred researchers to devise
detectors to mitigate potential risks, including misinformation, phishing, and
academic dishonesty. Despite this, most previous studies have been
predominantly geared towards creating detectors that differentiate between
purely ChatGPT-generated texts and human-authored texts. This approach,
however, fails to work on discerning texts generated through human-machine
collaboration, such as ChatGPT-polished texts. Addressing this gap, we
introduce a novel dataset termed HPPT (ChatGPT-polished academic abstracts),
facilitating the construction of more robust detectors. It diverges from extant
corpora by comprising pairs of human-written and ChatGPT-polished abstracts
instead of purely ChatGPT-generated texts. Additionally, we propose the "Polish
Ratio" method, an innovative measure of the degree of modification made by
ChatGPT compared to the original human-written text. It provides a mechanism to
measure the degree of ChatGPT influence in the resulting text. Our experimental
results show our proposed model has better robustness on the HPPT dataset and
two existing datasets (HC3 and CDB). Furthermore, the "Polish Ratio" we
proposed offers a more comprehensive explanation by quantifying the degree of
ChatGPT involvement.

--------------------
--- 문서 3 ---
제목: When ChatGPT is gone: Creativity reverts and homogeneity persists
저자: Qinghan Liu, Yiyong Zhou, Jihao Huang, Guiquan Li
요약:
ChatGPT has been evidenced to enhance human performance in creative tasks.
Yet, it is still unclear if this boosting effect sustains with and without
ChatGPT. In a pre-registered seven-day lab experiment and a follow-up survey
after 30 days of experiment completion, we examined the impacts of ChatGPT
```

```
presence and absence on sustained creativity using a text dataset of 3302
creative ideas and 427 creative solutions from 61 college students.
Participants in the treatment group used ChatGPT in creative tasks, while those
in the control group completed the tasks by themselves. The findings show that
although the boosting effect of ChatGPT was consistently observed over a
five-day creative journey, human creative performance reverted to baseline when
ChatGPT was down on the 7th and the 30th day. More critically, the use of
ChatGPT in creative tasks resulted in increasingly homogenized contents, and
this homogenization effect persisted even when ChatGPT was absence. These
findings pose a challenge to the prevailing argument that ChatGPT can enhance
human creativity. In fact, generative AI like ChatGPT lends to human with a
temporary rise in creative performance but boxes human creative capability in
the long run, highlighting the imperative for cautious generative AI
integration in creative endeavors.

--------------------
```

`lazy_load()`

- 문서를 대량으로 로드 시
  - 모든 로드된 문서의 부분 집합에 대해 하류 작업을 수행할 수 있다면 **`메모리 사용량을 최소화`하기 위해 `문서를 한 번에 하나씩 지연 로드`할 수 있음**

```
import fitz
print(dir(fitz))
print(type(dir(fitz)))                          # <class 'list'>
```

- 셀 출력

```
['ASSERT_PDF', 'Annot', 'AnyType', 'Archive', 'Base14_fontdict', 'Base14_fontnames', 'ByteString', 'CS_CMYK', 'CS_
```

```
from langchain_community.utilities import ArxivAPIWrapper
import requests
import fitz
from langchain.docstore.document import Document
import arxiv

# 검색할 논문의 주제
query_topic = "ChatGPT"

# 최대 문서 수
max_docs_to_load = 5                                            # 한 번에 5개까지 논문을 가져오기

# arxiv 검색 설정
search = arxiv.Search(
    query=query_topic,
    max_results=max_docs_to_load,
    sort_by=arxiv.SortCriterion.Relevance
)

def lazy_load_arxiv_docs():
    """
    arxiv.Client().results()를 사용하여 논문을 하나씩 로드하고 반환합니다.
    """
    print(f"'{query_topic}' 주제로 논문을 검색 중입니다...")

    for i, result in enumerate(arxiv.Client().results(search)):
        try:
            pdf_url = result.pdf_url
            print(f"\n{i+1}/{max_docs_to_load} 논문 다운로드 및 파싱 중: {result.title}")

            # 논문 PDF 다운로드
            response = requests.get(pdf_url)
            response.raise_for_status()

            # 다운로드된 PDF 데이터를 메모리에서 파싱
            with fitz.open(stream=response.content, filetype="pdf") as doc_file:
```

```
                text_content = "".join(page.get_text() for page in doc_file)

                # 추출된 정보를 Document 객체로 저장
                doc = Document(
                    page_content=text_content,
                    metadata={
                        "Title": result.title,
                        "Authors": ", ".join(author.name for author in result.authors),
                        "Published": result.published,
                        "pdf_url": pdf_url,
                    }
                )

                # 제너레이터(generator)를 사용하여 하나씩 반환
                yield doc

        except requests.exceptions.HTTPError as errh:
            print(f"HTTP 오류: {errh}")
            continue
        except Exception as e:
            print(f"논문 처리 중 오류 발생: {e}")
            continue

# 'docs' 리스트에 문서를 하나씩 추가 (lazy_load()와 유사한 방식)
docs = []
for doc in lazy_load_arxiv_docs():
    docs.append(doc)

# 최종 로드된 문서 목록 출력
if docs:
    print("\n=== 최종 로드된 문서 목록 ===")
    for i, doc in enumerate(docs):
        print(f"--- 논문 {i+1} ---")
        print(f"제목: {doc.metadata.get('Title', '제목 없음')}")
        print(f"요약 (일부):\n{doc.page_content[:300]}...\n")
        print("-" * 20)

else:
    print("\n조건에 맞는 논문을 찾지 못했습니다.")
```

- 셀 출력 (7.7s)
- 논문 5개는 순차적으로, 1개씩 검색됨

```
'ChatGPT' 주제로 논문을 검색 중입니다...

1/5 논문 다운로드 및 파싱 중: In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT

2/5 논문 다운로드 및 파싱 중: Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text

3/5 논문 다운로드 및 파싱 중: When ChatGPT is gone: Creativity reverts and homogeneity persists

4/5 논문 다운로드 및 파싱 중: Pros and Cons! Evaluating ChatGPT on Software Vulnerability

5/5 논문 다운로드 및 파싱 중: Unveiling the Role of ChatGPT in Software Development: Insights from Developer-ChatGPT Ir

=== 최종 로드된 문서 목록 ===
--- 논문 1 ---
제목: In ChatGPT We Trust? Measuring and Characterizing the Reliability of ChatGPT
요약 (일부):
In ChatGPT We Trust? Measuring and Characterizing
the Reliability of ChatGPT
Xinyue Shen1 Zeyuan Chen2 Michael Backes1 Yang Zhang1
1CISPA Helmholtz Center for Information Security
2Individual Researcher
Abstract
The way users acquire information is undergoing a paradigm
shift with the advent of Chat...

--------------------
--- 논문 2 ---
제목: Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text
요약 (일부):
IS CHATGPT INVOLVED IN TEXTS? MEASURE THE POLISH
```

```
RATIO TO DETECT CHATGPT-GENERATED TEXT
Lingyi Yang1, Feng Jiang 1 2 3*, Haizhou Li 1 2
1 School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
2 Shenzhen Research Institute of Big Data, China
3 School of Information Science and ...

--------------------
--- 논문 3 ---
제목: When ChatGPT is gone: Creativity reverts and homogeneity persists
요약 (일부):




1
When ChatGPT is gone: Creativity reverts and homogeneity persists
Qinghan Liu
Yiyong Zhou
Jihao Huang
Guiquan Li*
Peking U., School of
Psychological and
Cognitive Sciences
Peking U., School of
Psychological and
Cognitive Sciences
Beijing Yuxin
Technology Company
Peking U., ...

--------------------
--- 논문 4 ---
제목: Pros and Cons! Evaluating ChatGPT on Software Vulnerability
요약 (일부):
Pros and Cons! Evaluating ChatGPT on Software
Vulnerability
XIN YIN, Zhejiang University, China
This paper proposes a pipeline for quantitatively evaluating interactive LLMs such as ChatGPT using publicly
available dataset. We carry out an extensive technical evaluation of ChatGPT using Big-Vul cove...

--------------------
--- 논문 5 ---
제목: Unveiling the Role of ChatGPT in Software Development: Insights from Developer-ChatGPT Interactions on GitHu
요약 (일부):
Unveiling the Role of ChatGPT in Software Development:
Insights from Developer-ChatGPT Interactions on GitHub
RUIYIN LI, School of Computer Science, Wuhan University, China
PENG LIANG, School of Computer Science, Wuhan University, China
YIFEI WANG, School of Computer Science, Wuhan University, China...

--------------------
```

- *next:* `DirectoryLoader`
