2025.05.14.(수)

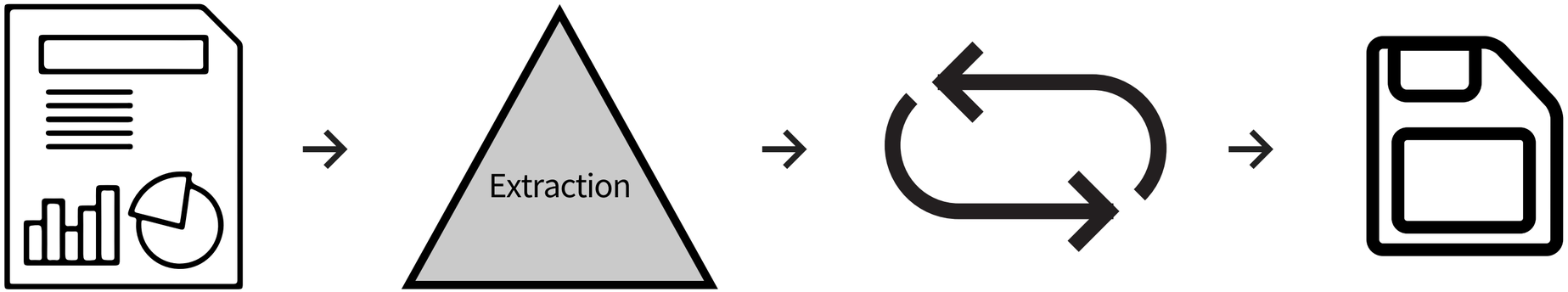
NS3 Code Generator

: LLM 기반의 네트워크 시뮬레이션 코드 자동 생성

202101109 박수화
202102722 최희원

목차

RAG DB 구축 흐름



수집 → 추출 → 변환 → 저장

접근 방식

Data 추출 기준

① 크롤링은 하되 구조화가 가능만 문서만 골라서 처리

- 공식 문서나 블로그 중에서 헤더나 소제목이 잘 되어있는 글만 처리
- 마크다운 언어 활용해서 분리

② 논문 파일(pdf 등) 기반 문서 처리

- 논문, 공식 문서에서 파일 직접 수집
- LLM(KeyBERT 등) 활용해 설정 관련 키워드 추출 → 자동 태깅 또는 웹 검색 쿼리 생성에 활용

접근 방식

RAG DB 구축

1. NS-3 에서 자주 등장하는 입력 설정 커버용 키워드 정리
    - 논문 5편 다운로드 및 텍스트 추출
    - pdf_keyword.py 를 통해 키워드 추출 + 저장 (json 형식)
    - 문제점: 추출된 키워드가 NS-3 코드와 직접 연관이 낮음

2. 수동 정의 키워드 기반 자동 쿼리 생성
    - 예시 키워드: 802.11ac, UdpEchoClient, PropagationDelayModel 등
    - query.py site:medium.com 등과 조합하여 쿼리 텍스트 자동 생성

```
802.11ac site:medium.com
UdpEchoClient site:velog.io
NetDeviceHelper site:tistory.com
```

3. 구조화된 문서만 크롤링

    → 예를 들어 <h1> 이런 마크다운 언어 단위로 청킹

4. 소제목 같은 것은 자동으로 태그를 통해 매핑

5. 임베딩 후 DB에 저장

접근 방식

1. NS-3 에서 자주 등장하는 입력 설정 커버용 키워드 정리

pdf_keyword.py

```python
import os
import fitz  # PyMuPDF
from keybert import KeyBERT

# PDF 파일들이 있는 폴더 경로
pdf_dir = "../rag_db_1/papers/"

# 모델 초기화
kw_model = KeyBERT()

# PDF 폴더 내 모든 파일 반복
def extract_section(text, keywords):
    lines = text.split("\n")
    for i, line in enumerate(lines):
        if any(k.lower() in line.lower() for k in keywords):
            return "\n".join(lines[i:i+15])
    return ""

for filename in os.listdir(pdf_dir):
    if filename.endswith(".pdf"):
        pdf_path = os.path.join(pdf_dir, filename)
        print(f"\n처리 중: {filename}")

        # 1. 텍스트 추출
        doc = fitz.open(pdf_path)
        full_text = "\n".join([page.get_text() for page in doc])

        # 2. Simulation 관련 섹션 추출
        setup_text = extract_section(full_text, ["Simulation", "Experiment", "Parameter", "Setup"])  # 또는 full_text(전체 내용) 사용

        if not setup_text.strip():
            print("관련 섹션이 없음")
            continue

        # 3. 키워드 추출
        keywords = kw_model.extract_keywords(setup_text, top_n=15, stop_words='english')
        print("추출된 키워드:")
        for kw, score in keywords:
            print(f"  - {kw} ({score:.2f})")

        # 4. 추출된 키워드 저장
        import json
        output_dir = "../rag_db_1/keywords/"
        os.makedirs(output_dir, exist_ok=True)

        output_path = os.path.join(output_dir, filename.replace(".pdf", ".json"))
        keyword_list = [kw for kw, _ in keywords]

        with open(output_path, "w") as f:
            json.dump(keyword_list, f, indent=2)

        print(f"저장 완료 → {output_path}")

# 추출된 키워드 저장
import json
output_dir = "../rag_db_1/keywords/"
os.makedirs(output_dir, exist_ok=True)

# 저장할 경로 지정
output_path = os.path.join(output_dir, filename.replace(".pdf", ".json"))
keyword_list = [kw for kw, _ in keywords]

# JSON으로 저장
with open(output_path, "w") as f:
    json.dump(keyword_list, f, indent=2)

print(f"저장 완료 → {output_path}")
```

접근 방식

2. 수동 정의 키워드 기반 자동 쿼리 생성

query.py

```python
import os
import json

# 1. 경로 설정
keyword_dir = "../rag_db_1/keywords/"
output_path = "../rag_db_1/keyword_queries.txt"

# 2. 사용할 site 목록
sites = ["medium.com", "velog.io", "tistory.com", "site:blog.naver.com", "site:github.io"]

# 3. 쿼리 생성
queries = []

for filename in os.listdir(keyword_dir):
    if filename.endswith(".json"):
        with open(os.path.join(keyword_dir, filename), "r") as f:
            keywords = json.load(f)

        for keyword in keywords:
            for site in sites:
                queries.append(f"{keyword} site:{site}")

# 4. 결과 저장
with open(output_path, "w") as f:
    f.write("\n".join(queries))

print(f"쿼리 {len(queries)}개 생성 완료 → {output_path}")
```

접근 방식

3. 구조화된 문서만 크롤링

```python
import os
import requests
from bs4 import BeautifulSoup
from langchain.docstore.document import Document
from langchain.vectorstores import Chroma
from langchain.embeddings import HuggingFaceEmbeddings

# URL 목록 불러오기
with open("../rag_db_1/url_list.txt", "r") as f:
    urls = [line.strip() for line in f if line.strip()]

# 크롤링 + 청크 추출 함수
def extract_chunks_from_url(url):
    try:
        response = requests.get(url, timeout=10)
        soup = BeautifulSoup(response.text, "html.parser")

        # 구조화된 콘텐츠만 추출
        content = []
        for tag in soup.find_all(["h1", "h2", "p"]):
            text = tag.get_text(strip=True)
            if len(text) > 50:  # 너무 짧은 건 제외
                content.append(text)

        # 간단한 청크 나누기
        chunks = []
        buffer = ""
        for paragraph in content:
            if len(buffer + paragraph) > 500:
                chunks.append(buffer.strip())
                buffer = paragraph
            else:
                buffer += " " + paragraph
        if buffer:
            chunks.append(buffer.strip())

        return chunks
    except Exception as e:
        print(f"{url} 에서 오류 발생: {e}")
        return []
```

builder.py

```python
# 문서 생성 + 태그 추가
documents = []
for url in urls:
    chunks = extract_chunks_from_url(url)
    for chunk in chunks:
        doc = Document(
            page_content=chunk,
            metadata={"source": url}  # 이후 자동 태깅 시 참고
        )
        documents.append(doc)

print(f"총 {len(documents)}개 청크 생성 완료")

# Chroma DB 저장
persist_dir = "../rag_db_1/chroma_db"
os.makedirs(persist_dir, exist_ok=True)

embedding_model = HuggingFaceEmbeddings(model_name="meta-llama/Llama-3.3-70B-Instruct")
db = Chroma.from_documents(documents, embedding=embedding_model, persist_directory=persist_dir)
db.persist()

print(f"Chroma DB 저장 완료 → {persist_dir}")
```

접근 방식

결과

```
처리 중: Deep_Reinforcement_Learning-Based_Mobility-Aware_UAV_Content_Caching_and_Placement_in_Mobile_Edge_Networks.pdf
추출된 키워드:
  - mdp (0.46)
  - reinforcement (0.40)
  - mobile (0.36)
  - edge (0.32)
  - bandwidth (0.32)
  - caching (0.32)
  - uav (0.30)
  - qoe (0.29)
  - algorithms (0.28)
  - cache (0.27)
  - optimal (0.27)
  - learning (0.26)
  - unmanned (0.26)
  - aerial (0.24)
  - network (0.23)
저장 완료 → ../rag_db/keywords/
```

```
처리 중: 3592149.3592161.pdf
추출된 키워드:
  - simulated (0.43)
  - openran (0.40)
  - simulation (0.38)
  - xapps (0.35)
  - 5g (0.34)
  - simu (0.33)
  - availability (0.31)
  - infrastructure (0.31)
  - software (0.30)
  - platform (0.30)
  - distributed (0.28)
  - stations (0.28)
  - telco (0.25)
  - datasets (0.25)
  - interface (0.24)
저장 완료 → ../rag_db/keywor
```

```
처리 중: electronics-09-00272-v2.pdf
추출된 키워드:
  - simulation (0.38)
  - network (0.34)
  - fisica (0.20)
  - università (0.17)
  - ns (0.17)
  - bioingegneria (0.16)
  - mauro (0.15)
  - studi (0.14)
  - computer (0.14)
  - matematica (0.13)
  - systematic (0.13)
  - milano (0.13)
  - campania (0.12)
  - informatica (0.11)
  - italy (0.11)
저장 완료 → ../rag_db/keywor
```

```
처리 중: 978-3-642-12331-3_2.pdf
추출된 키워드:
  - networking (0.43)
  - protocols (0.43)
  - network (0.41)
  - simulators (0.40)
  - simulator (0.40)
  - simulation (0.38)
  - ns (0.27)
  - testbeds (0.23)
  - prototyping (0.23)
  - scalability (0.22)
  - reproducibility (0.22)
  - scalable (0.21)
  - reproducible (0.18)
  - technologies (0.17)
  - research (0.17)
저장 완료 → ../rag_db/keywords/978-3-642-12331-3_2.json
```

⇒ 코드 생성에는 도움이 되지 않을 듯한 영양가 없는 키워드가 추출된 느낌
→ 논문이 직접적으로 ns3 코드에 사용되는 내용 중심이 아니기 때문 ?

접근 방식

피드백 및 진행 상황

- 커뮤니티 중심의 코드 수집 (StackOverflow, GitHub, NS-3 커뮤니티 등)
- parameter 설정은 대부분 experiment 섹션 이후에 존재 → 해당 부분 중심으로 수집
- 현재 해당 방향으로 코드 개선 중

감사합니다 :)