RAG의 계층적인 문서구조를 활용한 검색 방법

서론

- ChatGPT가 발표된 후 LLM을 이용한 RAG(Retrival Augmented Generation)를 이용한 챗봇이 많은 주목을 받고 있다.
- LLM의 환각현상을 줄이고 LLM이 학습하지 않은 최신의 정보들을 답변에 반영할 수 있다는 장점이 있기 때문이다.
- RAG의 답변 품질을 높히기 위해서는 근거 지식들을 검색하는 방법이 중요하다.
- 본 프로젝트에서는 지식을 계층화하여 저장한 후 이를 반영한 검색을 적용하여 LLM의 답변 품질을 올릴 수 있음을 살펴보고자 한다.

본론

계층화한 지식의 저장 및 검색 방안

- 일반적인 RAG 서비스에서는 레거시 문서에서 텍스트를 추출한 후 이를 청킹, 임베딩 과정을 거쳐 벡터 스페이스에 지식으로 저장한다.
- 사용자의 질의가 들어오면 벡터 스페이스에서 유사도 비교등의 방법을 이용하여 사용자의 질의에 근거가 될 지식들을 검색한다.
- 이렇게 검색된 지식들을 LLM 컨텍스트에 포함하여 답변을 생성하도록 한다.
- 지식 구축시 청킹의 단위등의 따라 검색되는 지식에 충분한 정보가 포함되어 있지 않아 정확하지 않은 답변이 생성되는 경우가 발생한다.
- 검색의 성능을 높히기 위해서 Query rewriting, Multi-query generation등의 기법들을 적용하기도 한다.
- 본 프로젝트에서는 가장 간단한 RAG 파이프라인 예제에서 근거 지식을 계층화하여 구축하고 이를 반영한 검색 방법을 적용해 본다.

테스트 방법

- 전처리 및 지식 구축
    1. 오픈 마켓의 모니터에 관한 상품 정보를 텍스트 및 계층화한 Markdown 형식의 문서로 변환
    2. 테스트에 사용할 질문 세트 작성
    3. 변환한 텍스트 문서와 Markdown 형식의 문서를 각각 청킹, 임베딩 과정을 거쳐서 PGVector에 임베딩하여 저장
- 검색 및 답변생성(질문 세트내 개별 질문에 대해)
    1. 텍스트 컬렉션과 Markdown 컬렉션에 대하여 각각 Retriver를 이용하여 답변의 근거가 될 지식들을 검색
        - 텍스트 형식를 저장한 컬렉션은 LangChain의 Retriver를 그대로 사용
        - Markdown 형식을 저장한 컬렉션은 아래에서 설명하는 검색방법을 더하여 사용
    2. 각각 LLM을 통해 답변 생성
- 결과 비교
    - 개별 질문들에 대해 담당자의 답변과 LLM이 생성한 답변들을 함께 비교

지식 저장

1. 텍스트 형식의 데이터 저장
    - LangChain 라이브러리를 이용하여 OpenAI의 Length 기반 청킹과 Embedding 모델을 사용한다.

    - 관련코드: `load-text.py`
2. Markdown 형식의 지식 저장
    - 본 프로젝트에서는 계층화한 지식의 형태로 Markdown 형식을 사용한다.
    - Markdown 형식이 계층을 표현 형식이면서, 이미 어느 정도 청킹이 다루기 편한 장점이 있다.
    - 레거시 문서에서 Markdown 형식으로 변환하는 것은 본 프로젝트의 범위에서 제외하였다.
    - LangChain의 Markdown Loader를 이용하여 PGVector Vector space에 로딩한다.
        - PGVector 컬렉센에 parent등의 정보가 추가된다.
        - 검색이 이 metadata를 활용한다.
    - 관련코드: `load-markdown.py`

지식의 검색 및 답변 생성

1. 텍스트 형식을 저장한 지식 검색
    - LanghChain Retriver를 이용하여 코사인 유사도 비교를 수행하여 지식을 검색하여 얻는 청크들을 LLM에 전달할 컨텍스트로 사용한다.
    - 관련코드: `answer-text.py`
2. 계층화한 지식 검색
    - LanghChain Retriver를 이용하여 코사인 유사도 비교를 수행하여 근거 지식을 검색하는 과정까지는 위의 테스트 형식을 저장한 지식의 검색과 동일하다.
    - 검색에서 반환한 각각의 청크에 대해
        - Root노드가 나올 때까지 parent 노드를 찾아 추가하여 Partial Tree를 구성한다.
    - 모든 Partial Tree를 병합한 결과를 LLM에 전달할 컨텍스트로 사용한다.
3. 각 컨텍스트를 포함하는 질문을 LLM에 전달하여 답변을 생성한다.
    - 관련코드: `run-test.py`

테스트 케이스를 통한 비교

결론

- RAG이용하여 LLM의 환각현상을 줄이고, LLM이 학습하지 않은 최신의 지식들을 답변헤 활용할 수 있는 장점이 있다. LLM이 정확한 답변을 제공하기 위해서는 Retrival 단계에서 보다 정확하게 근거 지식들을 검색할 수 있어야 한다.
- 본 프로젝트에서는 계층화한 지식을 벡터 스페이스에 저장하고 검색시 근거 지식에 연관이 있는 노드들을 추가함으로써 나은 품질의 답변을 얻을 수 있다는 점을 살펴보았다.
- PDF나 오피스 문서등의 레거시 문서에서 Markdown 형식이나 이와 비슷한 수준으로 청킹 및 계층화하는 방법등에 대한 연구 및 개발이 필요하다.

참고문헌

별첨

테스트 환경

- OS: Ubuntu 22.04 LTS(WSL2)
- Anaconda 3
- Python 3.11
- Vector Space: Postgres + PGVector
- Embedding API: OpenAI Embedding

- LLM: ChatGPT-4

테스트 프로그램

- `setup/`
  - `create-conda-env`
  - `requirements.txt` : 필요한 패키지들
- `data/`
  - `lg-monitor.md` : 오픈 마켓의 상품 정보를 Markdown 형식으로 변환한것 (https://www.11st.co.kr/products/1985839344)
  - `lg-monitor.txt`
  - `questions.json` : 테스트에 사용할 질문들
- `load-text.py`:
- `load-markdown.py`:
- `run-test.py`

Python 환경 생성

```
# conda virtual environment 생성
conda create --name hiarchical-knowledge python=3.12

# hiarchical-knowledge 환경을 활성화
conda activate hiarchical-knowledge

# 필요한 패키기 설치
# pip install
# pip freeze -r setup/requirements.txt

# hiarchical-knowledge 환경을 활성화
pip install -r setup/requirements.txt
```