**기초 자연어처리 Final Project Report**

텍스트, 도표, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명**Team 17**

그림 1. RAG Pipeline 구조

**1. Proposed RAG Architecture**

먼저 검색에 사용할 제공된 Wikipedia 데이터셋을 전처리후 noise가 있는 dataset과 나머지 clean dataset으로 분리했습니다. 이는 두 가지 데이터셋을 같이 embedding하게 되면 noise dataset이 clean dataset에 비해 **항상 query와의 similarity가 낮게** 나오므로, noise dataset에 query에 대한 관련 정보가 있음에도 불구하고 retrieve 하는 검색순위에서 밀리기 때문입니다. 따라서 noise 혹은 clean dataset 각각의 similarity는 noise dataset 안에서만 비교되도록 preprocessing을 거쳐 두 개의 dataset으로 구분했습니다(그림 1). 구분된 dataset에서 noise context와 clean context를 가져와서 이 context들과 user query의 similarity를 측정해, 이를 넘으면 prompt에 들어갈 context로 활용됩니다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명**2. Retrieval system**

그림 2. High Similarity Context Search

생성된 두 embedding vector에서 query와 가까운 vector를 용이하게 찾기 위해 FAISS library를 사용해 indexing을 진행했습니다. query가 입력되면, query를 embedding 후 FAISS로 가장 가까운 vector의 index를 출력하여 dataset에서 관련된 context들을 가져오는 retrieval를 구현했습니다..

**3. Query design and LLM prompt engineering**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**그림 3. Final LLM Prompt**

프로젝트를 진행하면서, retrieval를 통해 가져오는 context가 항상 답변의 quality를 올려주진 않는다는 점을 확인한 후, 일정 similarity 이상의 context만 가져오도록 설정했습니다. 또한 noise가 섞여 정보가 가려진 부분을 **(unknown)**으로 masking(그림 2)하여 **LLM이 유추**해볼 수 있도록 query를 설계했습니다. 또한 간결한 대답과 정확도 향상을 위해 prompt engineering을 시도했습니다(그림3).

**4. Post-processing**텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그림 4. Post-processing Rule

EM과 F1 score 향상을 위해 다양한 룰(그림 4)을 추가했습니다. 파일은 각각 post\_processing.py와 check\_submission.ipynb(.py)로 되어있으며, 두 파일을 순서대로 실행시켜야 정상적으로 submission이 됐습니다.

**5. Result**텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명

그림 5. 다양한 Prompt의 시도들

점수를 올리기 위해 다양한 prompt로 시도를 해보았습니다. Prompt의 길이가 모델의 성능을 좌우하는 것으로 알았으나, LLM의 모델의 크기에 비례해서 long-context를 잘 기억하는 것으로 보였습니다. 이번 Project에서 사용한 Gemma-2B의 경우에도 그리 큰 모델은 아니므로 많은 context를 집어넣을 경우 오히려 성능이 안 좋아지는 것을 볼 수 있었습니다. 결국, 모델 크기에 맞는 적절한 context의 길이도 찾는 것이 중요합니다..