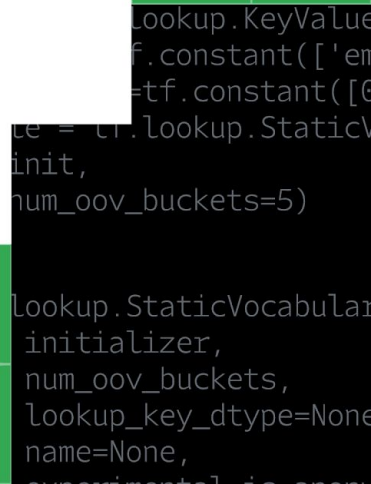
Google Developer Student Clubs

4주차 1조

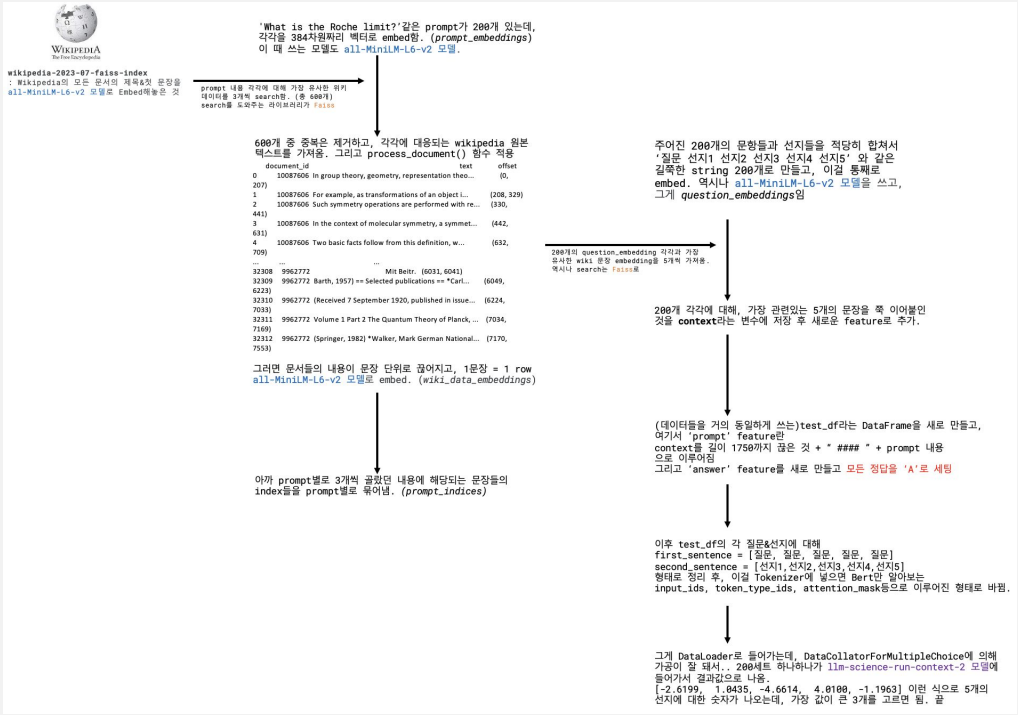
팀원: 강용진, 조현진, 조선빈



강용진

Code Review

- 리뷰한 코드 : https://www.kaggle.com/code/mgoksu/0-807-sharing-my-trained-with-context-model/notebook
- 작동 과정 구조도 :



Code Review

- 성능 개선 시도
  - 수정 없이 첫 시도: 정확도 0.807
  - 하이퍼파라미터 수정1: ‘NUM_SENTENCES_INCLUDE’를 5 -> 6, context의 길이를 1750 -> 2000
  - 정확도 0.812
  - 하이퍼파라미터 수정2: ‘NUM_SENTENCES_INCLUDE’를 6 -> 7, context의 길이를 2000 -> 2250
  - 정확도 0.815
- 향후 개선 가능한 포인트
  - fork한 코드에서는 test.csv만 사용하고, train.csv를 이용한 파인튜닝은 하지 않음.
    따라서 이 코드에서 사용된 AutoModelForMultipleChoice 모델에 파인튜닝을 시도해볼 수 있음
    Docs : https://huggingface.co/docs/transformers/tasks/multiple_choice

조현진

기본적인 전개

캐글에서 이미 주어진 코드를 프로토타입 삼아서 개선 시키려 한다.

1. 주어진 test file에서 prompt와 유사도가 큰 문장 상위 3개를 위키에서 가지고옴 그리고 그 문장이 포함된 parquet 파일을 가지고 옴
2. wikipedia_file_data에 위에서 뽑은 문장이 포함된 parquet파일과 넣어줌 파일에서의 문장 위치(id)를 넣어줌
3. wiki_text_data에 1번에서의 상위 문장 3개를 넣어줌
4. process_documents 함수를 통해서 wiki_text_data를 잘 다듬어 줌(processed_wiki_text_data)
5. processed_wiki_text_data를 임베딩한 것 -> wiki_data_embeddings
6. 기존 prompt와 보기 ABCDE를 합쳐주는 col 만듦 -> prompt_answer_stem
7. prompt_answer_stem을 임베딩함 -> question_embeddings
8. contexts에다가 5단어 겹치는 애들(위키에서 뽑은 문장들) 넣어줌(faiss로 효과적으로 찾아줌)
9. 이 중에서 1750 단어를 사용할 것임
10. 이제 trn('prompt', 'A', 'B', 'C', 'D', 'E', 'answer')의 value를 토큰화 해줌
11. 이 토큰화 된 것을 기반으로 predict를 함

하이퍼 파라미터 조정

1. 9번에서 1750단어만 사용했는데 2000단어 사용
   0.807->0.809
2. 유사도 상위 문장 3개 사용했는데 4개로 늘림
   0.809->0.813
3. ‘NUM_SENTENCES_INCLUDE’를 7개로 늘리기
   0.813->0.812
4. 9번에서 1750단어만 사용했는데 2250단어 사용
   성능 변화 없음
5. 유사도 상위 문장 3개 사용했는데 5개로 늘림
   0.812->0.818

조선빈

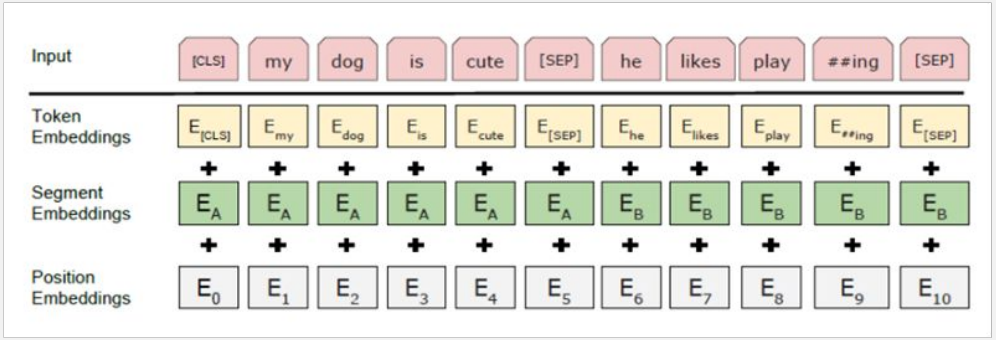
Transformer

Transformer Model

= Encoder + Decoder

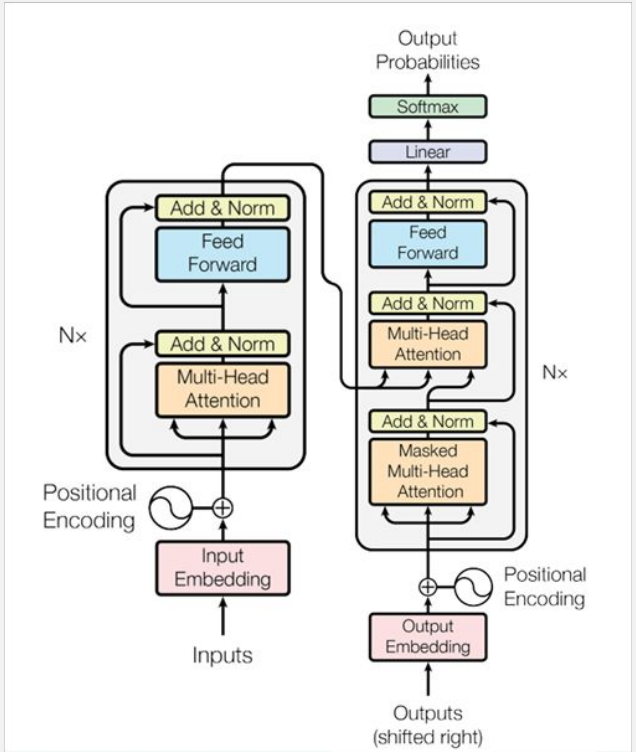
- Encoder : Input sequence > 1 vector

- Decoder : vector representation > Output sequence



Token Embeddings : vector representation for certain dimension

Input Sequence : Token + Segment + Position



Transformer Model 참고자료1
Transformer Model 참고자료2

FAISS

Sentence Transformers

compare the similarity between sentences

FAISS

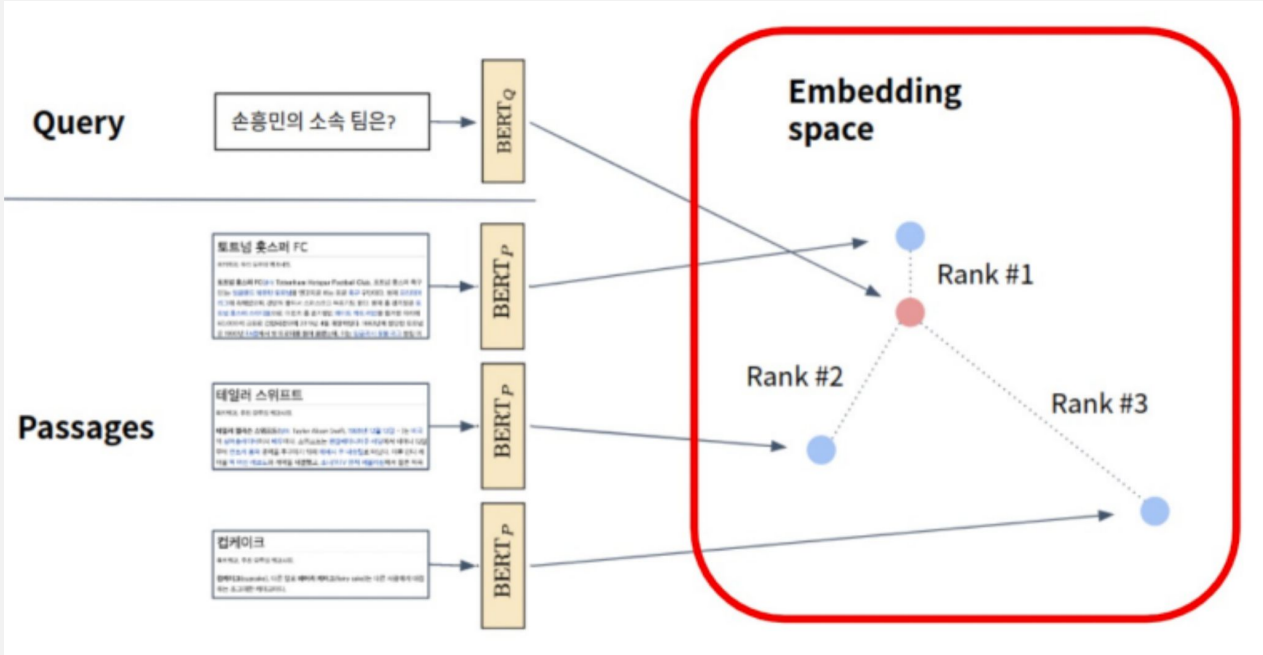
- similarity search library
- Facebook AI

1. Train index & Map vectors

- compression (SQ)
- clustering
- adding SQ8

2. Search based on FAISS index

- query > top-k cluster > search



FAISS 참고자료

FAISS

- 큰 데이터 처리 ~ 유사한 이웃을 일반적으로 이용되는 **cosine similarity**보다 빠르게 찾을 수 있음
- 고차원 벡터 공간에서 효율적으로 작동

  전체 벡터 공간을 **k**개로 나눈 후 가장 거리가 가까운 **N**개를 방문 및 유사성 확인

  Index.nprobe(N)

- 인덱싱 **& training** 과정 > 데이터셋의 특성에 맞게 인덱스 구축 가능