

Open-Domain Question Answering Competition

2021/10/12 ~ 2021/11/04

9조 Quarter100

프로젝트 개요

- 주어지는 지문이 따로 존재하지 않고 사전에 wiki data로 구축되어있는 knowledge resource에서 질문에 관련된 문서를 찾아주는 Retriever와 관련된 문서를 읽고 적절한 답변을 찾아주는 Reader를 만드는 프로젝트이다.

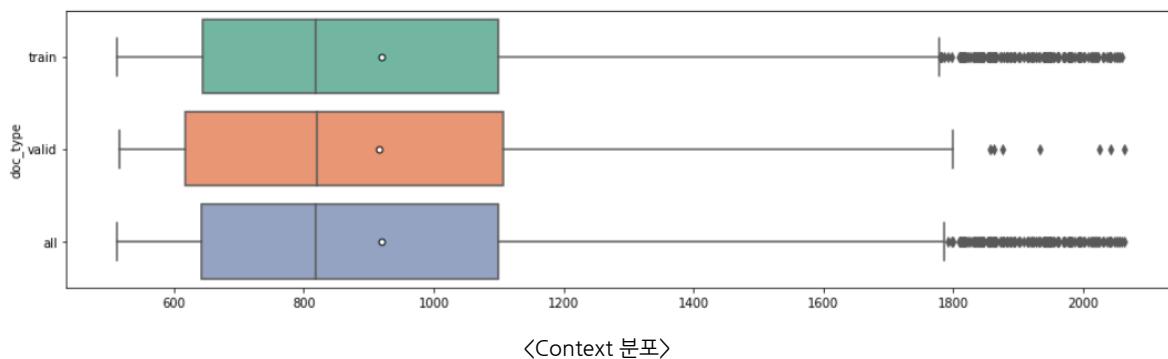
프로젝트 팀 구성 및 역할

| 조원 | Role, Work |
|-----|---|
| 김다영 | EDA, Data Pre-processing |
| 김다인 | ElasticSearch, K-fold validation set 구성 및 양상을 |
| 박성호 | Question Generation 구현 및 실험, Post-processing |
| 박재형 | ElasticSearch, Question Generation 구현 및 실험 |
| 서동건 | Reader, Post-processing |
| 정민지 | EDA, ElasticSearch, 양상을 |
| 최석민 | Reader, Post-processing |

프로젝트 수행

1. EDA

- wikipedia-documents에 대하여 총 60613개의 data중 중복된 문장 3876를 발견하였다. 이 문장 elastic setting 단계에서 제거해주었다.
- train, valid, train+valid(이하 all)에 대하여 Context 길이를 살펴보았다.

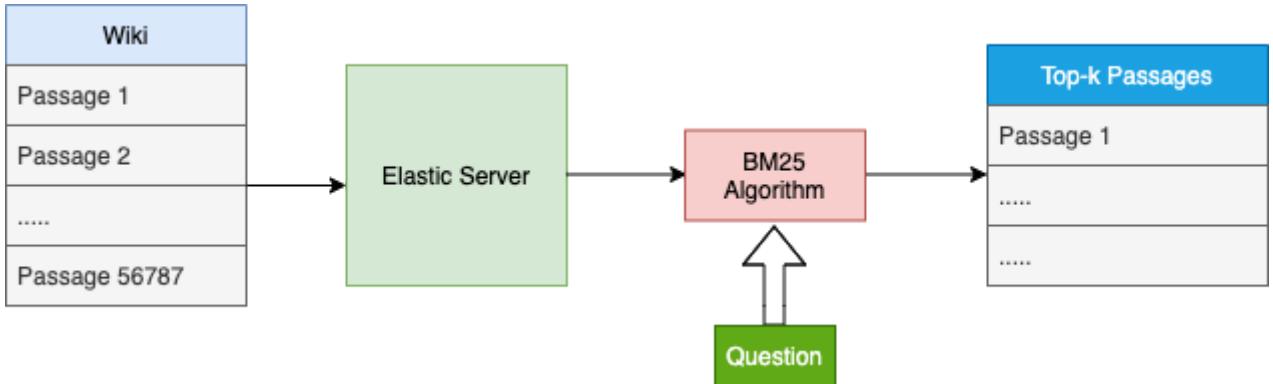


- Context의 경우 최소 길이 512, 최대 길이 2064로 사용한 모델의 max_length(512)보다 모두 큰 것을 알 수 있었다.

2. 전처리 :

- 다른 캠퍼님의 게시판 글을 통해 정답 텍스트에 들어가는 특수문자를 파악하였고, ‘₩n’ 등과 정답 산출에 필수적이지 않은 특수문자를 제외하였다. Retriever와 Reader 모두 전처리를 통해 예측 성능이 높아짐을 확인할 수 있었다.

3. Retriever

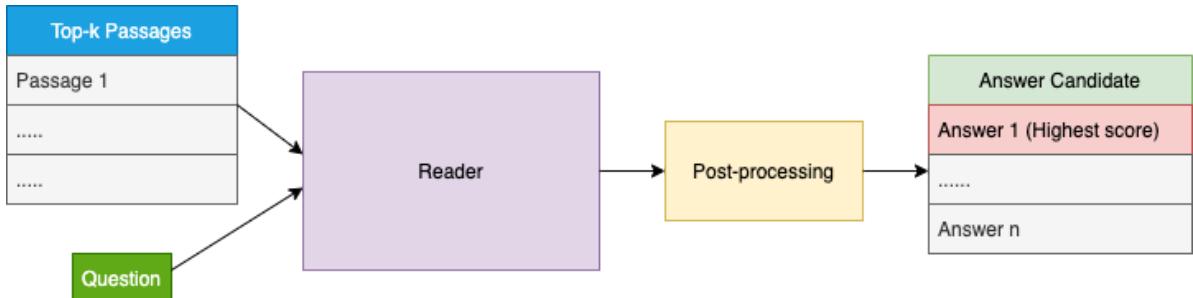


- **Elastic Search** : BM25를 default로 적용해 점수를 산출하는 검색엔진이다. 이를 활용해 직접 구현한 BM25보다 큰 점수 향상을 경험할 수 있었다. Analyzer로 nori tokenizer를 사용하고 Filter로 Stemmer와 Shingle을 사용했을 때 가장 좋은 성능을 보였다. 특히 pororo의 ner 태깅을 이용해 태깅이 가능한 단어가 들어있는 passage에는 부가 점수를 주도록 bool query의 should 구문을 구성하였다. 아래는 최종적으로 완성한 Elasticsearch Retriever의 topK 문서에 대한 성능이다.

| topK | 1 | 5 | 10 | 15 |
|----------|--------|--------|--------|--------|
| Accuracy | 0.7347 | 0.8839 | 0.9160 | 0.9325 |

- **Dense Retriever** : Passage Encoder와 Question Encoder를 통해 유사도가 높은 문서와 질문의 임베딩 값의 내적(Inner product)가 높게 나오도록 두 모델을 학습시킨다. 문서의 문맥을 임베딩화 할 수 있다는 장점을 가진다. 하지만 훈련 데이터 크기가 매우 작아 사용하지 못하였다.

4. Reader



- a. **Data augmentation** : 주어진 MRC dataset size가 작아 Reader 모델이 적은 양의 context에만 biased 되어 Inference 과정에서 새로운 context를 읽고 이해하는 능력이 떨어지는 문제가 있었다. 외부 데이터셋을 사용할 수 없는 본 대회 규정 상, 주어진 wiki documents 들을 이용해 Data를 늘리는 과정에 집중하였다.

- (1) **Negative Sampling** : Retriever를 이용해 Train dataset의 Question들과 유사도 점수가 높지만 정답을 갖고 있지 않은 context들(Negative passages)을 활용하였다. 이때 해당 question & context pair는 No Answer인 data이므로 Answer span position을 [CLS] 토큰으로 labeling 한다. 이 방법은 단순히 Reader가 여러 context들을 접할 수 있게 하는 것을 넘어 모델에게 더 어려운 Data를 제공함으로써 Robustness를 향상시킬 수 있다는 장점을 가진다.

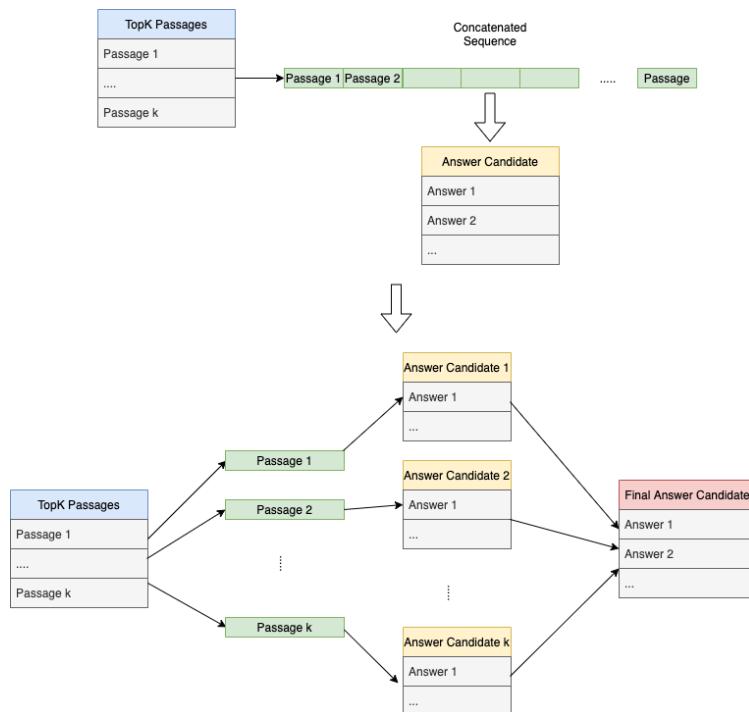
(2) Question Generation : 다른 캠퍼님의 QG관련 토론게시판 글에서 착안한 방법이다.

wiki_documents 데이터셋에서 title을 답으로 사용하고 question을 generation 한다. 또한 title이 같은 문서들은 하나로 합쳐 데이터로 활용했다. 이 때, extract base mrc에 맞게 문서 안에 title이 포함된 것만 사용했다.

KorQuAd v1.0을 학습한 KoGPT2 모델의 가중치를 사용하여 question을 generation 했다. 이 중 질문에 답이 있는 경우를 제외하고, 질문과 글의 bm25 점수를 계산하여 상위 1200개를 학습에 사용했다.

| Answer | Generated Question |
|---------|------------------------------|
| 영광 묘정영당 | 1616년 조선조 건국공신 이천우가 봉안한 영정은? |

b. Post-Processing



(1) Top-k Passages Separate

기존 Baseline code의 Inference과정은 Top-k개의 Passage들을 쭉 이어 concatenate하여 하나의 Passage로 취급한 뒤 Reader 모델을 통해 Answer candidates를 만드는 과정이었다. 하지만 이렇게 할 경우 concatenate한 긴 Passage가 tokenizer를 거치며 여러 feature로 쪼개는 과정에서 특정 feature에 여러 Passage가 겹친 채로 들어갈 수 있는 문제가 존재했다. Answer Span은 반드시 한 문서 안에서만 나온다는 점을 이용, Top-k passage들을 분리하여 Question과 함께 모델에 넣어주고 각 question & passage에서 나오는 answer candidate들을 모아 마지막으로 정렬하는 과정을 거쳐 최종 Answer candidates를 결정하는 방법을 택하였다.

(2) Answer Score

기존 Baseline code에서 각 answer 후보들의 score를 start_logit과 end_logit을 더하는 방식으로 연산하였다. 하지만 (1)의 방법을 사용할 경우, 각 Passage마다 출력되는 Logit 값들의 분포는 각각 다를 것이고 이를 단순히 더함으로써 대소관계를 비교해 score를 판단하는 것은 정확하지 않다고 판단하였다. 따라서 각 feature마다의 start_logit과 end_logit들을 softmax 연산을 통해 확률로 바꾸고 이 Start Probability와 End Probability를 곱하는 방식으로 바꾸었다. 이를 통해 다른 문서들의 정답 후보들을

합치고 비교하는 과정에서 확률이라는 동일한 기준을 가진 값으로 정렬하기 때문에 더욱 정확한 정렬이 가능하다는 장점이 있다.

(3) KSS (Korean Sentence Splitter)

retrieval 나온 top k 문서에 대해 KSS로 문장을 나누고, question과의 유사도를 계산하였다. 이후 SentenceTransformer를 이용하여 유사도가 0 이상인 문장만 문서로 간주하였다.

(4) Some minor Post-processing

mecab을 이용하여 모델이 예측한 단어 앞 뒤로 불필요한 특수 문자, 조사 등을 제거하였다.

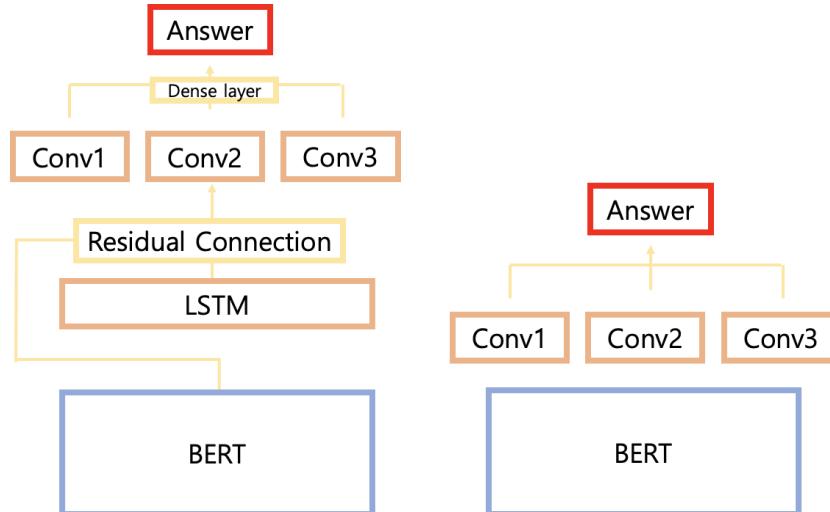
5. Ensemble :

퍼블릭 스코어가 높았던 20개 가량의 json파일을 hard voting했으며 후처리를 통해 동률인 경우에 대응했다. 이 방법을 통해 대회 마지막날 6점 가량의 점수 향상을 이루어낼 수 있었다.

6. 시도했지만 적용하지 못한 방법

Reader

모델 변경 실험을 하던 중 LSTM은 BERT에서 부족한 문장의 Sequential한 정보를 더 뽑아내기 위해 추가를 하였고 이 정보만으로는 부족하다고 생각해서 기존 BERT에서 나온 Output을 Residual Block과 같은 방식으로 더해준 다음 Conv Layer에 넣어주는 방식으로 아래와 같이 모델을 구성하였지만 성능이 좋지 않았습니다.



7. 자체 평가 의견(프로젝트 소감 및 느낀점 등)

- 서동건 : 4주동안의 긴 대회였는데도 끝까지 열심히 해 좋은 성과를 내준 팀원들에게 고맙다는 말을 하고싶다. NLP 분야의 Trendy한 Task인 Open-domain QA를 직접 진행해보는 좋은 경험을 얻었다.
- 최석민 : 다양한 시도를 했지만 모든 결과가 좋지 않아 많은 아쉬움이 남는 대회였다. 4주 동안 몰입한 대회가 처음이라 지치고 힘들었지만 우리 쿼터백 팀원들과 함께라서 잘 마무리 할 수 있었던 것 같다. 이번 대회에서 배우고 시도했던 것들을 다음 대회에 다시 꼭 도전해 볼 것이다.
- 정민지: 대회 기간이 길어서 그런가 저번 대회에 비해 지쳤던 것 같다. 기간이 길어서 다양한 시도를 해볼 수 있어서 좋았다. 하지만 실험 관리가 후반부에 약간 흐지부지 된 것 같아서 아쉽기도 했다.
- 박성호: Dense retriever 구현이 잘 되지 않았을 때 정말 힘들었다. 그래도 팀원들의 여러 시도들이 성공했고 이것들이 밑바탕이 되어 좋은 Ensemble 재료로 사용됐다는 점이 다행이다. 무사히 끝마쳐서 후련하다.

- 김다인 : 긴 기간동안 대회에 몰두하다보니 몸과 마음이 지치는 순간이 왔었다. 하지만 팀원들이 옆에서 큰 힘이 되어주어 대회를 성공적으로 마칠 수 있던 것 같다. 개인적으로 좋은 성능 평가와 밸리데이션 셋의 중요성, 앙상블의 위력 등을 다시 한 번 경험으로 확인할 수 있는 기회였으며 Elasticsearch를 비교적 깊게 이해하게 된 점이 뿌듯하다.
- 박재형: 이번 대회가 어렵고 기간이 길어서 지칠때가 많았는데 어려웠던 부분을 다같이 해결해서 대회를 좋은 성적으로 마무리할 수 있었다. 다양한 시도들을 할 수 있었고 공유도 잘되어서 좋았다. 또한 git을 지난번보다 더 잘 활용해서 뿌듯하다.
- 김다영: 따로 준비하는 일이 있어서 깊게 참여하지 못한 부분이 아쉽다.