🤔

MRC_*NLP*_팀 리포트(10조)

○ 프로젝트 개요

- **프로젝트 주제**

  Open-Domain Question Answering (ODQA) 주어지는 지문이 따로 존재하지 않고 사전에 구축되어있는 Knowledge resource 에서 질문에 대답할 수 있는 문서를 찾는 Task

- **프로젝트 개요**
  - 프로젝트 목표

    사용자가 원하는 질문에 답변을 해주는 ODQA 시스템을 구축하는 것

  - 구현 내용
    - EDA
      - Jupyter Notebook을 이용하여 데이터 특성 분석
    - Data Processing
      - 정규식 및 한글 관련 라이브러리 활용하여 데이터 전/후처리 구현
      - 위키피디아 데이터 문단 단위로 분리 (DPR, BERTSerini 논문 참고)
    - Reader Model
      - pretrained model 에 cnn layer를 추가 시도
    - Retrival Model
      - Sparse Retrieval (BM25)
      - Dense Retrieval (논문참조, Dense Passage Retrieval for Open-Domain Question Answering)
  - 교육 내용의 응용

    사전 구축된 대규모 데이터를 이용해 ODQA 시스템 구축이 가능

- **활용 장비 및 재료**(개발 환경, 협업 tool 등)
  - 서버환경 : Ubuntu 18.04.5 LTS , GPUv100
  - 개발툴 : vscode, jupyter notebook

- 협업툴 : Git, Github Project Slack, Zoom

- **프로젝트 File tree 및 Workflow**

○ 프로젝트 팀 구성 및 역할

- 김남현(T3021) : DPR+Reader Model 실험
- 민원식(T3079) : DPR 모델, BM25 구현 및 실험
- 전태양(T3194) : EDA, retriever 관련 실험 진행
- 정기원(T3195) : Data Processing, Tokenizer/MLM 관련 실험, 협업 환경 구축
- 주정호(T3211) : Reader Modeling & Fine-Tuning, Ensemble, Augmentation
- 최지민(T3223) : EDA, DPR(In-batch negative) 구현 및 Fine-Tuning

○ 프로젝트 수행 절차 및 방법

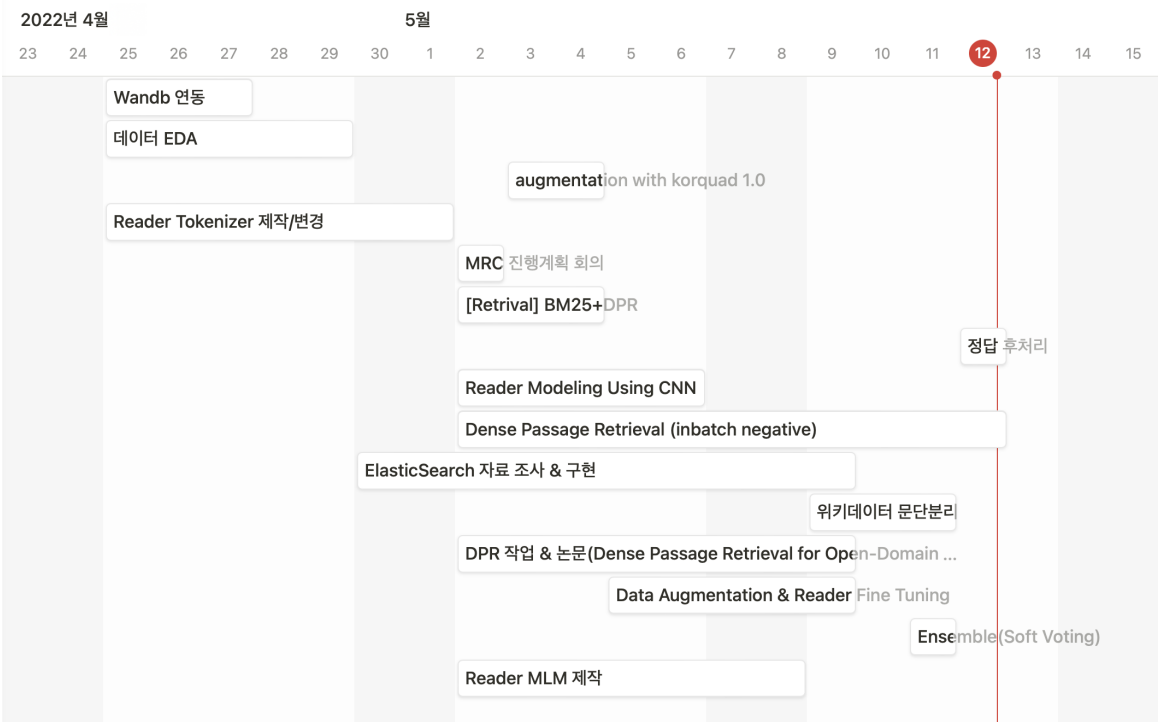
- **프로젝트 개발 Process**

  개발 과정을 아래와 같이 크게 5가지 파트로 분류함.
  - EDA : Jupyter Notebook을 이용하여 데이터 특성 및 이상치 분석
  - Data Processing : 모델 학습에 유용한 형태로 데이터를 처리
  - Modeling : 모델을 구현하고 성능 향상을 위해 Parameter Tunning 및 다양한 기능 추가
  - Model Test & Monitor : Monitoring Tool을 이용하여 모델을 다양한 환경에서 테스트
  - 협업 Tool 관리 및 기타(문서 정리) : Git Flow 적용

- **프로젝트 역할분담**

  모든 Process를 경험하고 싶다는 팀원들의 의견에 따라 팀원 별로 파트를 나누지 않고 모든 파트에 모든 팀원이 언제든 참여할 수 있도록 자유롭게 진행

- **프로젝트 수행 및 완료 과정**(Work Breakdown Structure)



○ 프로젝트 수행 결과

1. EDA

- 데이터셋 구성

분류	세부 분류	샘플 수	용도	공개여부
train_dataset	train	3952	학습용	모든 정보 공개 (id, question, context, answers, title)
	validation	240		
test_dataset	validation	600	제출용	id, question 만 공개

train

	con_len	qu_len	ans_len
count	3952.000000	3952.000000	3952.000000
mean	920.220648	29.322368	6.275051
std	356.500514	8.727421	5.346842
min	512.000000	8.000000	1.000000
25%	645.000000	23.000000	3.000000
50%	819.000000	29.000000	5.000000
75%	1099.250000	35.000000	8.000000
max	2059.000000	78.000000	83.000000

validation

	con_len	qu_len	ans_len
count	240.000000	240.000000	240.000000
mean	916.725000	29.195833	6.912500
std	360.032122	8.728301	6.858755
min	517.000000	9.000000	1.000000
25%	616.750000	23.000000	3.000000
50%	820.500000	29.000000	5.000000
75%	1107.250000	35.000000	8.000000
max	2064.000000	59.000000	64.000000

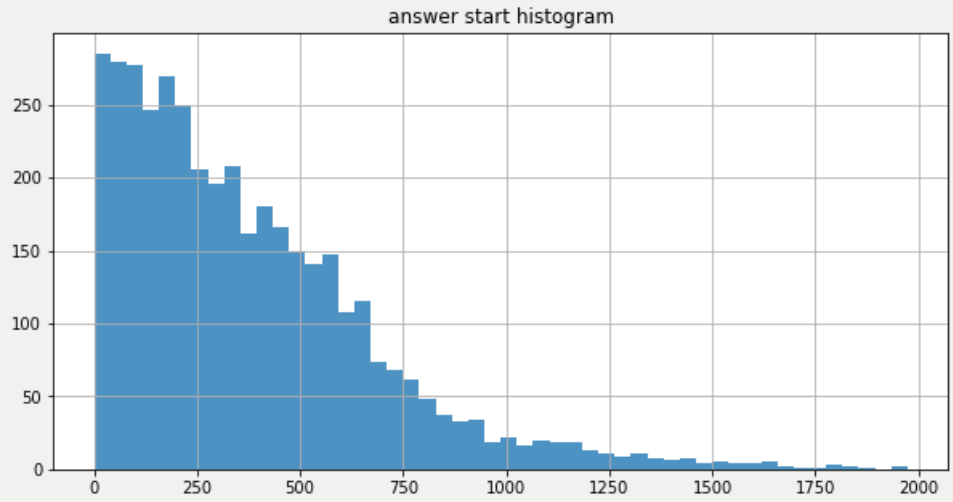
test

	qu_len
count	600.000000
mean	29.555000
std	8.962376
min	8.000000
25%	23.000000
50%	29.000000
75%	35.000000
max	62.000000

- context, question, answer 길이 모두 train, validation 데이터 유사
- test 데이터의 question 길이 또한 train, validation 데이터와 유사

- Context, Answer, Wiki 데이터셋 길이 파악
  - Context 내 Answer의 시작 위치 파악

💡 Answer start 위치는 0~1974이며 대부분 문장의 앞부분에 분포했다.



💡 **Context, Answer 내 존재하는 특수문자**

1. 한문, 일어, 러시아어
2. 개행문자 : \n, \\n, \n\n, \\n\\n
3. 특수 개행문자 : *, **
4. 따옴표 : “, ‘
5. 괄호 : 《》, 〈〉, ()
6. 기타 문자 : •, 『』, ⑥, ↑, ≪, °˙, ç

- Retrieval 과정에서 사용하는 wiki corpus

  개수 및 text 길이

	text_length
count	60613.000000
mean	755.565044
std	762.962671
min	184.000000
25%	414.000000
50%	577.000000
75%	857.000000
max	46099.000000

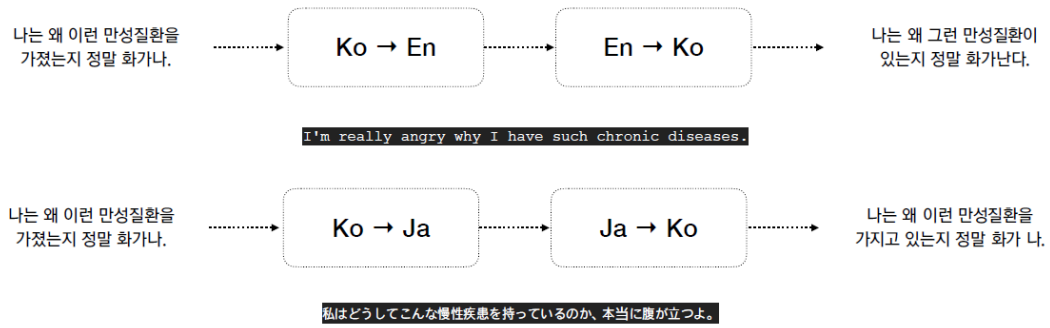
  예시

	text	corpus_source	url	domain	title	author	html	document_id
0	이 문서는 나라 목록이며, 전 세계 206개 나라의 각 현황과 주권 승인 정보를 개...	위키피디아	TODO	None	나라 목록	None	None	0
1	이 목록에 실린 국가 기준은 1933년 몬테비데오 협약 1장을 참고로 하였다. 협정...	위키피디아	TODO	None	나라 목록	None	None	1
2	현 서울특별시 종로구 서린동 (구 일제 강점기 경기도 경성부 서린정) 출신이다. 친...	위키피디아	TODO	None	백남준	None	None	2
3	아오조라 문고(青空文庫, あおぞらぶんこ|아오조라 문고)는 '일본어판 구텐베르크 프로...	위키피디아	TODO	None	아오조라 문고	None	None	3

- 중복을 제거하면 56737개의 unique한 문서로 이루어져 있음

2. **데이터 처리**

- 전처리 (KoBERT clean 함수 참고)
  - 정규식을 활용하여 문장기호, ASCII문자, 한글, 히라가나, 가타카나, 한자, 영어를 제외한 문자 및 URL 주소형태의 문장을 제거
  - Soynlp 기능을 통해 'ㅋㅋㅋㅋ'와 같이 동일한 문자가 중복해서 발생하는 데이터를 제거
  - 전각 문자를 반각 문자로 치환
- 후처리
  - utils_qa.py에서 정답을 기록할 때 Mecab을 활용하여 형태소 분리 후 한국어 불용어 사전을 통해 불용어를 제거한 값을 넣어줌
  - 한국어 불용어 사전은 https://bab2min.tistory.com/544 참고하였음
- Augmentation
  - KorQuAD 1.0
  - 학습 데이터의 Question을 영어와 일본어로 Backtranslation (Pororo의 Machine Translation 이용)

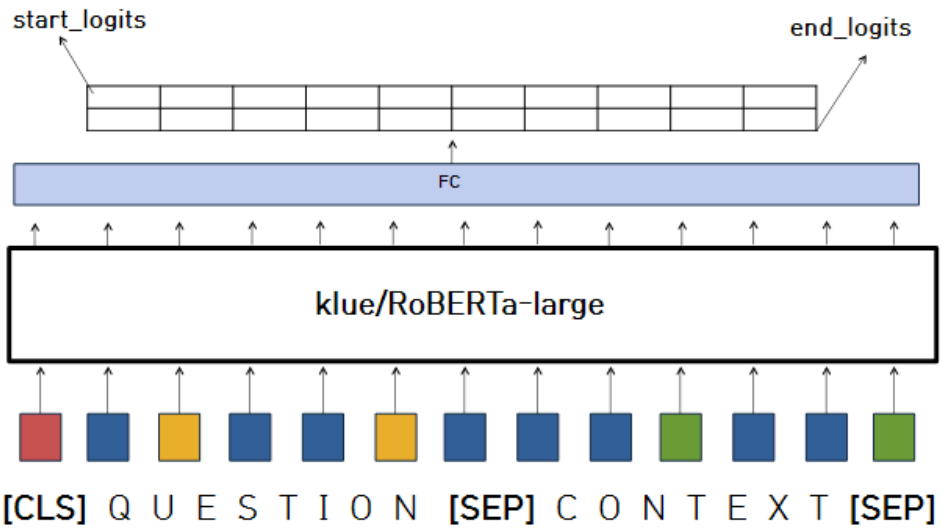


  - 성능 비교
    - 학습 데이터만을 학습한 것보다 성능이 하락하였다.
- 위키피디아 데이터 문단 단위 분리
  - DPR, BERTserini 논문에서 Retrieval 모델에 활용한 위키피디아 데이터를 문장 단위로 묶어 Passage에서 Paragraph로 데이터 단위를 축소할 때 성능이 향상된 것을 확인
  - 주어진 위키 데이터가 '\n' 개행문자를 통해 문장을 분리하고 있어, 하나의 문단에 포함할 문장의 개수 및 최소 문단길이를 조절하며 실험
  - 실험 결과 문장 개수 = 5, 최소 문단길이 = 15로 설정할 때 EM, F1 값이 비슷하면서 예측값은 다른 경우를 발견하였음. 이를 앙상블에 활용함.

3. **모델 개요**

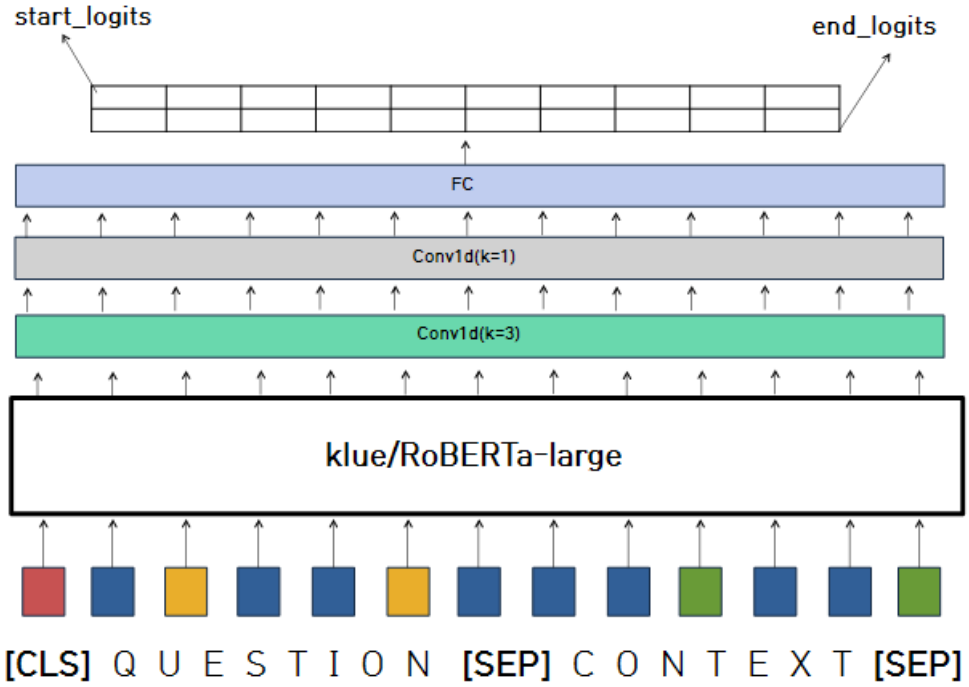
- Reader Model
  - AutoModelForQuestionAnswering

- Klue/bert-base
- Klue/roberta-small
- Klue/roberta-large를 이용해 학습 데이터를 임베딩



◦ CNN을 이용한 모델

- Klue/roberta-large를 이용해 학습 데이터를 임베딩



◦ 성능 비교(EM / micro f1)

- **AutoModelForQuestionAnswering - Klue/roberta-large : 70 / 78.64**

        - CNN을 이용한 모델 : 67.91 / 78.76

    - Reader Tokenizer 교체
        - 허깅페이스에 등록된 토크나이저 및 Konlpy.tag 라이브러리(Mecab, KKma, Komoran)를 사용한 Reader 성능을 실험해봄.
        - 실험 결과 Klue/roberta-large 토크나이저의 성능이 제일 좋아 배제하였음.

    - Vocab 교체
        - Klue/roberta-large 토크나이저의 vocab을 교체하는 작업을 진행함.
        - Mecab을 활용하여 형태소 단위로 단어를 자르고 허깅페이스 토크나이저 학습 함수를 활용하여 새로운 32000개의 vocab을 제작하였음.
        - 성능은 미흡하여 원인을 분석하다 새로운 vocab을 사용할 경우 MLM을 통해 pretrain을 진행해야 되는 것을 알게 되어 후속실험으로 MLM을 진행함.

    - MLM 학습
        - Mecab을 통해 새로 구축한 Vocab을 적용한 RoBERTa 모델을 제작하기로 함.
        - fairseq 깃헙 게시물을 참고하여 진행
        - train+valid data, KLUE-MRC data 두 종류로 실험하였으나, GPU 성능 한계상 fairseq에서 제시한 실험 설정을 충족하지 못하여 성능이 미흡하였음.

- Retrival Model
    - Spaerse Retrieval
        - TF-IDF
          단어의 등장빈도와 단어가 제공하는 정보의 양을 이용하여 벡터화된 passage embedding BoW를 구성
        - BM25
          TF-IDF의 개념을 바탕으로 문서의 길이까지 고려하여 점수를 산출
        - 토크나이저
            - sangrimlee/bert-base-multilingual-cased-korquad
            - monologg/koelectra-base-v3-finetuned-korquad
            - klue/roberta-large

    - Dense Retrieval
      사용자 Question과 찾고자 하는 Passage를 각각 Embedding하는 encoder 구현

      In-batch negative 방식 사용
        - Encoder Model
            - 실험 모델

        - bert-base-multilingual-cased
        - klue/roberta-base
        - klue/bert-base
    - 토크나이저
      각 실험 모델별 기본 토크나이저 사용
    - KorQuAD 1.0 데이터로 추가 학습

- Reader Model + Retrieval Mode 성능 비교(EM / micro f1)

  reader modeld를 동일한 환경으로 하였을때 아래와 같은 결과를 관찰함

retrieval	dense encoder model	epoch	val_EM	val_F1	토크나이저
DPR	klue/roberta-base	5	26.25	30.03	DPR 기본 토크나이저
DPR	bert-base-multilingual-cased	5	30	35.58	DPR 기본 토크나이저
DPR	bert-base-multilingual-cased	20	39.58	45.08	DPR 기본 토크나이저
DPR	klue/bert-base	10	45.83	53.76	DPR 기본 토크나이저
DPR+BM25(2-gram) (스코어비율 1:1)	klue/bert-base	30	54.58	62.43	DPR 기본 토크나이저
Tf-idf			51.66	60.34	klue/bert-base
Tf-idf			53.75	61.09	klue/roberta-large
Tf-idf			53.33	60.83	monologg/koelectra-base-v3-finetuned-korquad
bm25			52.91	61.57	klue/bert-base
bm25			50.00	56.51	klue/roberta-large
bm25			60.83	68.22	monologg/koelectra-base-v3-finetuned-korquad

  - Dense Retrieval 인코더 모델 간 비교
    - 총 3개의 인코더 모델('klue/bert-base' , 'klue/roberta-base', 'bert-base-multilingual-cased') 을 사용함.
    - epoch 10인 'klue/bert-base'가 epoch 20인 'bert-base-multilingual-cased' 보다 성능이 우수하여 인코더 모델로 선정함.
  - Sparse Retrieval 성능 비교
    - TF-IDF : 'klue/roberta-base'와 'monologg/koelectra-base-v3-finetuned-korquad'의 성능이 비슷하였음
    - BM25 : 'monologg/koelectra-base-v3-finetuned-korquad'의 성능이 가장 우수하였음
  - Dense Retrieval & Sparse Retrieval 비교
    참고 논문(Dense Passage Retrieval for Open-Domain Question Answering)의 내용을 구현해 보고자 하였지만 본 대회 실험 환경에서는 Sparse Retrieval가 더 나은 성능이 나왔음.

- SOTA 모델
  - Reader Model
    - Model
      - AutoModelForQuestionAnswering - Klue/roberta-large
    - Hyper Parameter
      - Learning Rate : 3.00e - 6
      - Batch Size : 8

- max_seq_len : 512
- Epoch : 5
- Loss Function : CrossEntropy
- Optimizer : AdamW
- fp16 : True

  - Retrieval
    - BM25
    - 토크나이저 : monologg/koelectra-base-v3-finetuned-korquad
- Ensemble
  - Reader Model은 AutoModelForQuestionAnswering - Klue/roberta-large로 고정
  - Retrival Model
    - Tokenizer, top_k, 데이터 처리 방식으로 경우를 나누어 앙상블을 진행하였음.
    - 시간 제한 상 네 개의 모델에 대해 앙상블을 진행하였음.

앙상블 실험 테이블

Aa 토크나이저	데이터 처리	# top_k	# val_em	# val_f1
koelectra	x	40	60.83	68.22
roberta-large	전처리	40	58.33	65.89
roberta-large	전처리 + 문단분리 (4문장 단위, 최소길이 10)	20	55	60.83
roberta-large	전처리 + 문단분리 (5문장 단위, 최소길이 15)	20	56.67	63.54

  - 성능 비교
    - 단일 SOTA 모델에 비해 EM이 Public 2.9%, Private 6.7% 상승하였음 (Public: 55.83→58.75%, Private: 53.89→60.56)

○ 자체 평가 의견

- **프로젝트의 의도 및 달성 정도**
  - 프로젝트의 의도 : 사전에 구축되어있는 Wikipedia 에서 질문에 대답할 수 있는 문서를 찾고, 해당 문서에서 질문에 대한 답을 하는 인공지능 구현
  - 달성 정도
    - Public (11팀 中 6등)

순위	팀 이름	팀 멤버	EM	F1
6 (-)	NLP_10조	원식 태양 희 지민	63.7500	73.5800

- Private (11팀 中 7등)

순위	팀 이름	팀 멤버	EM	F1
7 (1 ▼)	NLP_10조	원식 태양 희 지민	58.8900	70.9500

- **계획 대비 달성도, 완성도 등**(자체적인 평가 의견과 느낀 점)
  - 성능 향상으로 이루어지진 않았지만, 계획에 대해 대부분의 실험을 모두 수행하였다.

- **잘한 점과 아쉬운 점**(팀 별 공통 의견 중심으로 작성하며, 2~3장 분량을 고려하여 개인적인 의견은 개인 회고 부분에서 작성할 수 있도록 합니다.)
  - **잘한 점들**
    - 데이터 처리를 통해 앙상블 결과를 올렸다.
    - 비록 성능 향상으로 이루어지진 않았지만, 여러 외부 데이터를 이용한 Augmentation을 하여 실험해 보았다.
  - **시도 했으나 잘 되지 않았던 것들**
    - ElasticSearch 모듈화를 시도했으나 잘 되지 않았다.
    - “Dense Passage Retrieval for Open-Domain Question Answering” 논문 구현을 하였으나 GPU 메모리 한계로 인해 batch size를 늘리지 못해 논문 성능 재현이 되지 않았다.
  - **아쉬웠던 점들**
    - Reader의 성능을 많이 개선하지 못했다.
  - **프로젝트를 통해 배운 점 또는 시사점**
    - ODQA에서는 Retrieval의 성능이 매우 중요하다.

개인회고록(민원식)

개인회고록(김남현)

개인회고록(전태양)

개인회고록(정기원)

개인회고록(최지민)

개인회고록(주정호)

개인회고록(민원식)

○ 이번 프로젝트에서 나의 목표는 무엇이었는가?

- 베이스라인 코드를 완벽하게 이해하는 것
- Dense Passage Retrieval 를 구현하고 커스텀 해보는 것
- Sparse Retrival를 구현해 보는 것
- 외부 데이터셋을 이용해 학습을 진행해 보는 것

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 대회를 본격적으로 진행하기 전에 모든 베이스라인 코드를 읽어가며 주석을 다는 작업을 하였고, 특히 함수 호출과 반환 부분의 메개변수 타입을 파악함.
- 제공된 실습코드를 모듈 파일로 만들어 베이스라인 코드와 연동하도록 함.
- BM25를 완전히 새로 만들지는 못하였지만 기존에 구현된 모듈을 활용함.
- AIhub 기계독해 데이터셋을 이용하여 retrieval 모델 학습에 이용해봄.

○ 나는 어떤 방식으로 모델을 개선했는가?

- BM25를 이용하여 TF-IDF 보다 좋은 성능을 냄.
- Retrival Encoder Model을 모듈화 함.

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 이미 구현된 모듈을 가져다 사용하는데 두려움이 많이 사라졌음.
- 혼자만 사용하는 것이 아닌 모두가 같이 사용할 수 있는 코드 작성에 대해 많이 고민을 해봄

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 성능이 오르진 않았지만 주어진 데이터 이외에 다른 데이터 셋을 조사하고 학습에 이용해 봄.
- 논문을 보고 완벽하지는 않아도 비슷한 방식으로 구현해 보고 튜닝도 해보고자 시도함.
- 가독성이 높고 편리하게 사용하기 위한 코드를 만들기 여러 방법을 고민해봄.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 이번에 사용해 보지 못한 엘라스틱 서치와 같은 도구 사용의 필요성을 절감함.

- 새로운 기술과 논문에 대한 공부가 필요함을 느낌.
- 실험을 체계적으로 하지 못해 마지막에 자료를 정리하면서 부족한 부분을 발견함.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 효율적인 프로젝트 진행을 위해 중간 중간 팀원간 현재 진행 중인 사항을 점검하는 시간을 가져보려함.
- 자료조사를 효율적으로 하기 위해 도움이 될만한 사이트나 멘토님의 조언을 적극적으로 구하려함.
- 최종 프로젝트에는 새로운 도구를 선정해 하나라도 익혀보려 함.

개인회고록(김남현)

○ 이번 프로젝트에서 나의 목표는 무엇이었는가?

- MRC와 ODQA 흐름 이해하기
- MRC 베이스라인 코드 이해하기
- 논문 구현

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- MRC 공부
  - 베이스라인 코드에서 강의 자료 및 오피스아워에서 강조된 내용을 집중적으로 읽었다.
  - ODQA 관련한 자료를 찾아서 읽었다.
- 베이스라인 코드 뜯어보기
  - 이번 베이스라인 코드는 이전 대회에 비해 상당히 복잡했기에, 각각의 모듈이 어떻게 연관되어 있는지 파악하려고 노력했다.
- 관련 논문을 읽었다.
  - “Dense Passage Retrieval for Open-Domain Question Answering”
  - 논문 구현 시도.

○ 나는 어떤 방식으로 모델을 개선했는가?

- 논문에서 소개한 내용에 따라 베이스라인 코드를 바탕으로 DPR을 구현하려고 했다.
- 하이퍼파라미터 튜닝.

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 바라는대로 모델 성능 향상이 이루어지지 않아서 실망했다.
- 마지막에 팀원들과 앙상블 작업을 했는데 최종적으로 좋은 성적을 거두었다. EM이랑 f1 스코어 모두 60대를 달성했다. 지난 대회들 경험까지 고려해볼 때 다양한 모델로 앙상블을 풍부하게 하는 것이 성능 향상에 많이 도움이 된다는 걸 깨달았다.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전과 달리 이번 대회 때는 관련 논문을 자세하게 읽고, 모델 성능 향상을 위한 중간 목표를 설정했다.

- MRC 관련 깃허브 자료(e.g. monologg)를 탐색하고 대회 때 써보려고 시도했다. 다만 자료를 활용할 능력이 부족했고, 결과적으로 실패했다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 논문에 따라 DPR 구현을 하려 했으나 논문에서 나온 batch 사이즈를 서버가 감당할 수 없어서 그대로 구현하지 못했다.
- 베이스라인 코드가 상당히 복잡해서 각 모듈 간 흐름을 이해하는 데에 어려움을 많이 느꼈다.
- 베이스라인 코드에 내가 구현하고자 하는 기능을 추가하는 데에 어려움을 느꼈다.
- 무엇보다 MRC 대회 자체가 어려웠다고 생각한다. 강의와 오피스아워에서 자세하게 알려주지 않은 거 같아서 아쉽다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇일까?

- 더 적극적인 관련 자료 탐색
- 잘 안되면 팀원에게 빠르게 도움 요청하기
- 코드 모듈화 연습

개인회고록(전태양)

○ 이번 프로젝트에서 나의 목표는 무엇이었는가?

- 우선적으로 Baseline 코드를 이해하고자 했다.
- 성적을 올리기 위해 적극적으로 자료 조사하여 팀원들과 공유하고자 했다.
- Sparse Retrival, Dense Passage Retrieval, Elastic Search를 이해하고 구현해보려 했다.

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- huggingface Question Answering 공식 문서를 참고하여 baseline 코드의 흐름을 이해하려 노력했다.
- EDA를 활용하여 데이터를 살펴봤고, context와 answer에 들어가 있는 특수문자들, 빈번히 나타나는 단어들과 같은 정보를 팀원들에게 공유했다.
- 과제로 나온 실습코드를 모듈화하여 베이스라인 코드에 적용하려 했다.

○ 나는 어떤 방식으로 모델을 개선했는가?

- Dense Passage Retrieval 구현
- TF-IDF의 max_features, Retrieval topk와 같은 하이퍼파라미터 튜닝을 진행했다.

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 생각보다 DPR(Dense Passage Retrieval)의 성능이 좋지 않았고 오히려 BM25를 활용한 Sparse Retrieval의 성능이 더 좋게 나왔다.
- max_features 제한으로 인해서 embedding vector의 표현력이 떨어진다고 생각했고, 50,000 → 100,000으로 늘렸으나 효과가 미비했다.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전 대회와 달리 훈련 시 사용한 hyperparameter까지 실험 기록을 더 세밀하게 기록했고, 더 정갈하게 실험을 진행할 수 있었다.
- Github와 Kaggle을 활용하여 더 적극적으로 자료를 찾았고, reader model + CNN layer 구조와 BM25에 대한 정보를 얻을 수 있었다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 찾은 자료를 구현하고 디버깅하는 것에 대해 어려움을 느꼈다. 특히 ElasticSearch의 많은 기능들을 다루지 못한 것이 아쉬웠다.
- Retrieval에 비중을 많이 실었으나 큰 성능을 올리지 못했다.
- Retrieval에 Reader를 다뤄볼 여력이 없었던 것이 아쉬웠다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 이번에 ElasticSearch 구현하는 부분에서 홀로 긴 시간을 소모하였다. 다음 프로젝트에서는 막히는 부분이 있다면 이번 대회처럼 혼자 고민하는 것이 아닌 팀원들에게 공유하여 시간을 효율적으로 쓸 수 있도록 노력해야겠다.

개인회고록(정기원)

○ 이번 프로젝트에서 나의 목표는 무엇이었는가?

- 허깅페이스 인터페이스 익히기
- 수업에서 제시한 논문 구현
- 한국어 형태소 분석기(Konlpy, Soynlp) 사용해보기

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 허깅페이스 인터페이스 익히기
  - KoBigBird 인코더, RoBERTa MLM, 허깅페이스 토크나이저 학습 구현
- 논문 구현
  - DPR, BM25 선형결합 방식 구현
  - 문단 분리 구현
- 한국어 형태소 분석기 사용
  - Konlpy를 활용하여 Vocab 제작 및 텍스트 전/후 처리 구현
  - Soynlp를 활용하여 Vocab 제작 및 텍스트 처리 구현

○ 나는 어떤 방식으로 모델을 개선했는가?

- 논문에 근거하여 다양한 문단 길이를 실험하였다
  - 한 문단 당 최소 문장 개수 5개, 최소 길이 15으로 실험한 값으로 모델을 개선하였다
- Reader, Retriever 각각에 토크나이저를 교체하거나 Vocab을 개선거나, MLM을 통해 아예 모델까지 새로 학습하려고 시도하였다
  - 실험을 통해 최적의 토크나이저 활용 방법을 찾을 수 있었다. (Retriever 모델 토크나이저 교체)
- BM25, TFIDF, DPR 등 다양한 조건에 대해 실험을 진행하여 최적의 결과를 얻을 수 있었다

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- Reader에서 토크나이저를 활용하였지만, 성능 개선이 이뤄지지 않았다. 기한이 정해진 프로젝트의 경우 잘 안되는 방법에 매달리기 보다 다른 파트에 적용하거나 다른 실험을 진행하는 것이 좋다는 것을 깨닫게 되었다.
- 다양한 조건으로 실험을 진행한 결과 앙상블에서 좋은 결과를 얻을 수 있었다.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전 기수 글을 안보고 수업 관련 자료만 활용하여 프로젝트 진행
  - 주도적으로 진행할 수 있었지만, 프로젝트 진행 속도가 느린 단점이 있었다
- 실험 일지를 작성하며 모든 실험을 기록하였음
  - 프로젝트 말미에 앙상블을 진행할 때 도움이 되었음

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 토크나이저를 통해 Reader 모델 개선을 하지 못했음
- 베이스라인 코드가 복잡해서 BERTSerini 구조를 적용하지 못했음

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 성능향상이 이뤄지지 않으면 다음 실험으로 빠르게 넘어가기
- 간단한 베이스라인 코드를 바탕으로 제작하기

개인회고록(주정호)

○ 이번 프로젝트에서 나의 목표 & 달성하기 위해 무엇을 어떻게 했는가?

- Reader Modeling & Fine-Tuning
- Augmentation
- Ensemble

○ 나는 어떤 방식으로 모델을 개선했는가?

- KorQuAD 1.0의 1위 모델인 삼성 SDS의 SDS-XFormer+ (single model)을 차용하여 Reader Modeling
  - Klue/roberta-large를 이용해 학습 데이터 임베딩
  - CNN-relu-resnet-layernorm을 이용한 신경망 추가
- Augmentation
  - KorQuAD 1.0
  - 학습 데이터의 Question을 영어, 일본어으로 Backtranslation - Pororo의 Machine Translation 이용
- Ensemble
  - Reader은 AutoModelForQuestionAnswering - Klue/roberta-large 고정
  - Retrieval은 성능이 좋은 4개의 모델을 이용해 예측
  - 4개의 결과를 Soft Voting해 점수 향상

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 다양한 방법으로 Reader을 Modeling하고, 다양한 데이터를 이용해 추가로 학습시켰지만, 기존 학습 데이터 셋으로 학습 시킨 AutoModelForQuestionAnswering - Klue/roberta-large가 성능이 가장 좋았다. → 학습 데이터 셋인 Klue MRC와 성질이 다른 학습 데이터로 학습을 시키면 오히려 성능 하락으로 이어지는 것 같다.
- 성능이 좋은 다수의 결과를 Ensemble했을 때 SOTA를 달성하였다.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- github를 이용한 협업으로 팀원들끼리 다양한 시도의 코드를 쉽게 공유하고 접근할 수 있었다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- Reader을 끝내 기본 모델보다 좋은 성능으로 끌어내지 못했다.
- Reader에 몰두하느라, Retrieval을 공부하는 시간이 부족하였다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- Retrieval에 대한 다양한 논문을 공부하여, Retrieval을 다양한 방법론으로 시도해볼 것이다.

개인회고록(최지민)

○ 이번 프로젝트에서 나의 목표는 무엇이었는가?

- ODQA(Open-Domain Question Answering) task를 Retriever+Reader 접근 방식으로 해결하기
- Retriever
  - 어휘적 유사도가 아닌 의미적 유사도를 통해 적절한 passage를 찾아오고자 함

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- DPR (Dense Passage Retrieval) 구현 및 기학습된 encoder Fine-tuning

  Dense Passage Retrieval for Open-Domain Question Answering 논문 참조
  - In-batch negative 방식으로 연산량을 줄임

○ 나는 어떤 방식으로 모델을 개선했는가?

- 논문에서 batch size가 커질 수록 성능 향상이 된다고 보고되었으나, 주어진 GPU 메모리 한계로 인해 batch size 16이 최대였음

  ⇒ HuggingFace의 Accelerate를 활용하여 batch size 22까지 늘리는 동시에 학습 시간 단축
- Dense encoder model을 KorQuAD 1.0 데이터로 추가 학습
  - KorQuAD 데이터에 context 당 여러 question이 있어서, in-batch negative 방식 사용 시 어떤 context가 positive인 동시에 negative일 수 있음

    ⇒ context 당 하나의 question만 가져와서 학습하였음

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- DPR이 sparse retrieval 보다 성능이 낮게 나왔는데 두 가지 요인 때문이라고 분석함
  - Accelerate를 활용하여 batch size를 조금 늘리긴 하였지만(22), 논문에서 제시된 것(128)과 많이 차이가 나서 논문 성능을 재현하지 못함
  - question에 대한 answer를 context에서 추출하는 extraction-based MRC 대회여서 어휘적 유사도가 높은 sparse retrieval이 유리함

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 외부 데이터를 이용한 추가 학습을 통해 DPR 중 최고 성능 달성함

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 강의에서 배운 Bi-encoder model이 아닌 다른 retrieval 모델(Cross-encoder, ColBERT 등)을 찾아보지 못한 것이 아쉬움

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 강의에서 제공된 것 이외의 관련 자료 탐색