MRC Wrap-Up Report

Team : 자만추 (NLP-01)

1. 프로젝트 개요

A. 프로젝트 주제

> 💡 **Open-Domain Question Answering (ODQA)**

- **Question Answering (QA)**은 다양한 종류의 질문에 대해 대답하는 인공지능을 만드는 연구 분야입니다.
- 다양한 QA 시스템 중 ODQA는 주어지는 지문이 따로 존재하지 않고 사전에 구축되어있는 Knowledge resource 에서 질문에 대답할 수 있는 문서를 찾는 과정이 추가되기 때문에 더 어려운 문제입니다.
- 질문에 관련된 문서를 찾아주는 "retriever" 와 관련된 문서를 읽고 적절한 답변을 찾거나 만들어주는 "reader"를 구현하여 질문에 대한 원하는 답변을 얻을 수 있습니다.

B. 데이터셋

- **데이터 유형 (Source)** : `wikipedia_documents` (약 57,000개)
- **Train_dataset 데이터 개수 : train (**3,952), **validation** (240)
- **Test_dataset 데이터 개수** : **validation** (600)

분류(디렉토리 명)	세부 분류	샘플 수	용도	공개여부
train_dataset	train	3952	학습용	모든 정보 공개 (id, question, context, answers, document_id, title)
	validation	240		
test_dataset	validation	240 (Public) 360 (Private)	제출용	id, question 만 공개

제공된 학습 데이터셋 구성

C. 평가지표

- **Exact Match (EM)** : 모델의 예측과, 실제 답이 정확하게 일치할 때만 점수가 주어지는 평가방식
- **F1 Score** : 일치하지 않더라도 겹치는 단어도 있는 것을 고려해 부분 점수가 주어지는 평가방식

2. 프로젝트 팀 구성 및 역할

- 김효연 : Haystack, Elasticsearch(Retrieval), DPR(Retrieval)
- 서유현 : 데이터 전처리/후처리, Question Generation, PEFT
- 손무현 : 데이터 증강 (BackTranslation), Retrieval 실험 및 앙상블, PEFT
- 이승진 : 데이터 전처리, Haystack, BM25(Retrieval), Reranker, DAPT-TAPT-Fine tuning(twice)
- 최규빈 : 프로젝트 서칭, Base-code 작성, Curriculum learning
- 황지원 : PM, 데이터 증강 (Hanspell), 데이터 샘플링 및 전처리, Bigbird(Reader), DPR(Retrieval)

3. 프로젝트 수행 절차 및 방법

팀 목표 설정

(1주차) 대회 간 팀 규칙을 정하였고 협업을 위한 버전관리 방법론으로 Github Flow를 선정하였습니다. 학습 데이터 증강 및 분석을 진행하였고, 추가 학습을 위해 외부 데이터 조사를 진행하였습니다. 또한 원활한 학습을 진행하기 위해 Retrieval 및 Reader에 대한 베이스 코드 작업을 진행하였습니다.

(2주차) 이전에 조사한 외부 데이터와 증강 데이터를 활용한 모델 학습을 진행하였습니다. 또한 베이스라인 코드를 재작성하여 TF-IDF, BM-25 등 다양한 Retrieval에 대한 연구와 함께 Roberta-large, KoBigbird, Fine-tuned-Roberta-large 등 다양한 모델을 실험하고 튜닝을 진행하였습니다. 또한 다양한 시각에서 QA Task를 바라보기 위해 curriculum learning, TAPT, DAPT, ReRanking에 대한 연구를 진행하였습니다.

(3주차) 이전에 진행하였던 실험들을 마무리하고 성능 향상에 집중하였고, 마무리 작업을 위해 voting 방식을 통한 앙상블 작업을 진행하였습니다.

4. 프로젝트 수행 결과

1. 데이터 분석

- **데이터 source와 특징**
  - 이번 대회에 사용한 데이터는 Huggingface에서 제공하는 `wikipedia_documents` 데이터를 사용하였습니다.
  - 각 데이터에 대해 title과 그와 관련된 context를 주제로 question과 그에 대한 answers 데이터를 가지고 있습니다.
- **Raw Data 확인**
  - 중복된 context를 사용하여 다른 question을 묻는 데이터가 다수 있었습니다. (`579` 개)
  - 또한 한글/영어 외에 다른 문자(특수문자, 일본어, 한자 등)가 학습에 영향을 줄 수 있다고 판단해 분석을 진행하였고, 다음과 같이 분포한 것을 확인하였습니다.
    - `특수문자`가 있는 데이터 : `3952`개
    - `한자`가 있는 데이터 : `924`개
    - `일본어`가 있는 데이터 : `58`개
- **텍스트 길이**
  - 학습 데이터를 분석하면서 전체적으로 텍스트들의 길이 분포를 확인해본 결과 **가장 긴 텍스트의 길이**가 `2,059 자`, **짧은 텍스트의 길이**는 `512 자` **(평균 :** `920 자`**)**임을 확인했습니다.
  - 이는 긴 context를 한번에 찾는 과정에서 tokenizer의 seq_length와 연관이 있다고 생각했으며, length를 고려하여 tokenieze를 진행한 데이터를 조율하고자 하였습니다.

```
20]: train['train'][0]

20]: {'title': '미국 상원',
 'context': '미국 상의원 또는 미국 상원(United States Senate)은 양원제인 미국 의회의 상원이다.\\n\\n미국 부통령이 상원의장이 된다. 각 주당 2명의 상원의원이 선출되어 100명의 상원의원으로 구성되어 있다. 임기는 6년이며, 2년마다 50개주 중 1/3씩 상원의원을 새로 선출하여 연방에 보낸다.\\n\\n미국 상원은 미국 하원과는 다르게 미국 대통령을 수반으로 하는 미국 연방 행정부에 각종 동의를 하는 기관이다. 하원이 세금과 경제에 대한 권한, 대통령을 포함한 대다수의 공무원을 파면할 권한을 갖고 있는 국민을 대표하는 기관인 반면 상원은 미국의 주를 대표한다. 즉 캘리포니아주, 일리노이주 같이 주 정부와 주 의회를 대표하는 기관이다. 그로 인하여 군대의 파병, 관료의 임명에 대한 동의, 외국 조약에 대한 승인 등 신속을 요하는 권한은 모두 상원에게만 있다. 그리고 하원에 대한 견제 역할(하원의 법안을 거부할 권한 등)을 담당한다. 2년의 임기로 인하여 급진적일 수밖에 없는 하원은 지나치게 급진적인 법안을 만들기 쉽다. 대표적인 예로 건강보험 개혁 당시 하원이 미국 연방 행정부에게 퍼블릭 옵션(공공건강보험기관)의 조항이 있는 반면 상원의 경우 하원안이 지나치게 세금이 많이 든다는 이유로 퍼블릭 옵션 조항을 제외하고 비영리건강보험기관이나 보험회사가 담당하도록 한 것이다. 이 경우처럼 상원은 하원이나 내각책임제가 빠지기 쉬운 국가들의 국회처럼 걸핏하면 발생하는 의회의 비정상적인 사태를 방지하는 기관이다. 상원은 급박한 처리사항의 경우가 아니면 법안을 먼저 내는 경우가 드물고 하원이 만든 법안을 수정하여 다시 하원에 되돌려보낸다. 이러한 방식으로 단원제가 빠지기 쉬운 함정을 미리 방지하는 것이다.날짜=2017-02-05',
 'question': '대통령을 포함한 미국의 행정부 견제권을 갖는 국가 기관은?',
 'id': 'mrc-1-000067',
 'answers': {'answer_start': [235], 'text': ['하원']},
 'document_id': 18293}
```

2. 데이터 전처리(Preprocessing) / 후처리(postprocessing)

2.1) Tokenizer vocab 교체

- Train 데이터셋에 있는 특수문자 중 Roberta-large 모델 기준 vocab에 없는 특수문자가 `210`개 있었습니다.
- 이중 유의미한 특수문자가 있을지도 모른다고 판단해, Unknown 토큰 (`[UNK]`)이 되지 않게 해당하는 특수문자를 모두 모델의 vocab에 추가하였습니다.
- Vocabulary를 수정하고 동일한 Train 데이터셋으로 훈련한 결과 **EM**은 더 좋은 결과가 나왔고, **F1 Score**는 하락하였습니다.

No.	Data	EM	F1 Score
1	base-data (roberta-large)	35.42	48.41
2	vocab 교체 (roberta-large)	37.08 **(+1.66)**	47.42 **(+0.99)**

2.2) 개행 문자 제거 / 전각 문자 제거

- 일부 Context를 확인했을때, **개행문자** (`\n`)가 다수 있는 것을 확인하였고, 일부에 대해서는 **전각문자**와 문맥에 상관없는 일부 **특수문자**(`\”`, `\’`)가 있는 것을 확인하였습니다.
- 개행 문자의 경우는 context의 불필요한 요소라 생각을 하였고, 전각문자와 같은 특수문자 들은 tokenizing을 진행할 때에 잘못 분류할 수 있을것이라 판단하였습니다.
- 이런 가설하에 학습 데이터에 대해 전처리를 진행하면 좀더 **양질의 데이터**를 확보해서 학습 결과가 오르지 않을까하고 실험을 진행하였습니다.
- 예상과 다르게 전처리를 진행한 데이터에 대해 안좋은 결과가 나왔었습니다. 이는 여러 이유가 있겠으나 원본 데이터을 해치지 않은 범위에서 전처리를 하는 것이 Wiki 데이터로부터의 QA 작업에서 더욱 좋은 효율을 보인다는 것을 확인하였습니다.

No.	Data	EM	F1 Score
1	base-data (roberta-large)	60.42	70.32
2	data-preprocessing (roberta-large)	57.50 **(-2.72)**	69.42 **(-0.90)**

2.3) Output 데이터 조사 후처리

- 예측한 결과에서 일부 정답 끝에 조사가 붙어있는 것을 확인하였고, 실제 정답에는 필요하지 않다고 생각했습니다.
- **mecab**등 한국어 형태소 분석기를 통해 몇몇 정답의 조사를 찾아서 직접 제거하였습니다.
- 직접 일부 데이터에 대해 수정 작업을 진행하여 제출한 결과 약간의 점수 상승을 확인할 수 있었습니다.
- `‘바이샬리에’`, `‘아루에’`에서 ‘에’같이 조사인지 고유명사의 일부인지 헷갈리는 경우가 많아서 후처리 작업 자동화를 처리하지 못했습니다.

No.	Data	EM	F1 Score
1	base-data (roberta-large)	35.42	48.41
2	data-postprocessing (roberta-large)	35.83 **(+0.41)**	48.50 **(+0.09)**

3. 데이터 증강

3.1) 외부 데이터 활용

- 주어진 학습데이터 외에도 다양한 데이터를 통해 학습을 진행하여 `MRC Task`에 맞는 적합한 데이터셋을 찾고자 하였습니다.
- ODQA Task를 고려하여 관련된 다른 외부데이터를 활용해 모델 학습을 진행해보았습니다.
  1. `KorQuAD v1` : 1,560 개의 **Wikipedia article**에 대해 10,645 건의 문단과 66,181 개의 질의응답 쌍으로, **Training set 60,407 개, Dev set 5,774 개**의 질의응답쌍으로 구분한 데이터 *(CC BY-ND 2.0 KR)*
  2. `AI HUB [기계독해]` : 뉴스 본문 기반 학습 데이터셋 45만 건을 구축한 데이터
- 모델을 동일하게 설정하여 외부 데이터에 대하여 학습을 진행하였습니다.

No.	Data	EM	F1 Score	**데이터량**
1	**Base**	35.42	48.41	`3952`
2	**KorQuAD v1**	32.08 **(-3.34)**	45.84 **(-2.57)**	`60407`
3	**AIHUB MT**	26.67 **(-8.75)**	40.31 **(-8.10)**	`243425`

- 다양한 양질의 데이터를 찾기 위해 외부데이터를 탐색하였지만, MRC Task에 맞지 않았는지 예상과 다르게 데이터셋의 결과가 좋지 않았습니다.

3.2) Back-Translation 데이터 증강

- QA task에서는 모델을 다양한 예제와 문장 구조에 노출시켜 일반화 능력을 향상시키는 것이 중요하기 때문에, 다양한 언어로 번역하고 역번역하여 모델이 다양한 문장 구조와 어휘를 이해하고 처리할 수 있도록 해주는 **Back-Translations 기법**을 시도하고자 했습니다.
- **Back-Translations**을 진행하면서 번역된 문장에서 answer의 위치 인덱스 정보가 기존 answer_start 위치와 다르게 바뀌어 올바르게 존재하지 않은 경우가 있다는 것을 알 수 있었습니다.
  - 이를 해결하기 위해 answer를 문장 내에서 찾아서 그것의 인덱스를 찾고자 answer 부분을 마스킹하고 번역한 뒤, 번역이 완료된 문장에 마스킹 부분을 원래 answer text로 다시 채워 넣는 방법을 채택하여 작업을 진행하였습니다.
  - 작업을 진행하면서 기존의 문제점이었던 번역 후 answer_start 정보가 변경되는 문제를 해결할 수 있었습니다.
- 또한 영어와 일본어 두 언어 중 일본어로 우선 실험을 진행했는데, 번역 품질이 좋지 않은 것을 확인하였습니다. 특히 좋지 않은 번역 품질로 인해 Answer Masking이 되지 않는 경우가 생기는 것을 확인하였습니다.
- 해당 문제를 해결하기 위해 여러 번역 방법들을 시도해보았고, 웹(Web) 상에서 수행한 구글 번역이 더 나은 번역 결과를 보여준다고 판단하여 웹 크롤링을 통한 **역번역(Back-Translations)**을 다시 시도하였습니다.
- 라이브러리와 웹 크롤링 두 경우의 번역 결과를 비교해보았는데 두 경우 중 라이브러리를 사용한 방법이 좀더 효율적인 방식이라는 것을 확인할 수 있었습니다.
  - 번역 품질 및 효율성 등을 종합적으로 고려하여 최종적으로 **googletrans 라이브러리**를 사용하여 최종적인 **Back-Translation** 데이터 증강을 수행했습니다.
  - 또한 일본어에 비해 Answer Masking 이슈가 덜 발생했던 영어로 역번역을 진행하였습니다.
- 결과
  - 역번역의 결과로 Evaluation에서는 EM 수치가 **+0.42** 만큼 상승한 것을 확인하여 긍정적으로 보았습니다.

- 하지만 아래의 리더보드 제출 결과에서는 EM 수치가 **-3.34** 하락하였습니다.

No.	Data	EM	F1 Score	데이터량
1	Base Data	35.42	48.41	`3952`
2	BTS_증강_English	32.08 **(-3.34)**	45.84 **(-2.57)**	`7480`

- 리더보드 제출 결과가 좋지 않았던 이유로 불필요한 특수문자 등을 제거하지 않은 상태에서 역번역을 수행했기 때문에, 번역품질이 좋지 않았음을 확인하였습니다.
- 또한 역번역 과정에서 Reader 모델이 Answer를 찾기 위해 필요한 문맥 정보들이 많이 훼손되어 원하는 데이터를 가져올 수 없어 나온 결과라고 판단했습니다.

3.3) Hanspell 데이터 증강

- 일부 데이터를 확인해봤을 때, 문법 또는 띄어쓰기 등 오류가 있는 데이터가 있는 것을 확인하였습니다.
- 이런 일부 에러 데이터가 학습을 진행하는데 **노이즈(noise)**로서 학습 결과에 영향을 줄 수 있다는 생각 데이터를 정제하여 양질의 데이터를 확보하고자 하였습니다.
- py-hanspell 라이브러리
  - `네이버 맞춤법 검사기`를 이용한 한글 맞춤법 검사 라이브러리 입니다.
  - 좀더 정교한 데이터로 문장을 수정하여 학습에 활용할 수 있습니다.
- 학습 데이터에 대해 hanspell 라이브러리를 통한 맞춤법 검사 전처리를 진행하였습니다.

No.	Data	EM	F1 Score
1	Base	35.42	48.41
2	Hanspell 전처리	21.25 **(-14.17)**	32.11 **(-16.30)**
3	Hanspell 증강	31.67 **(-3.75)**	45.20 **(-3.21)**

- Hanspell 라이브러리를 통해 정제된 데이터를 만들면 더 좋은 학습결과를 만들 수 있을 것이라 생각을 했는데, 예상과 다르게 결과가 좋지 않았습니다.
- 이에 일부 노이즈가 포함된 데이터가 더 좋은 학습결과를 만들수 있다라는 결론을 내렸습니다.

3.4) Question Generation 데이터 증강

- 동일한 context를 이용해 question를 새로 만들면 데이터 증강이 되지 않을까 생각하여 적용한 기법입니다.
- 기존의 Train 데이터셋에서 **context**와 **answer**를 input 데이터로 사용하였고, `SKT-AI/KoGPT2`을 기반으로 Fine-tuning된 Question Generation 모델을 이용해 데이터 증강을 시도하였습니다.
- 증강한 데이터를 활용해 동일한 Roberta 모델을 사용해 실험을 진행하였습니다.

No.	Data	EM	F1 Score
1	base-data (roberta-large)	35.42	48.41
2	question-generated data (roberta-large)	26.25 **(-9.17)**	26.25 **(-22.16)**

- 증강한 데이터로 제출해보았으나 점수가 크게 하락하였고 분석 결과 다음과 같은 문제를 발견하였습니다.
  1. question에 정답이 들어가는 경우가 있었습니다. (위 부분은 ODQA 데이터 룰에 어긋나는 부분임을 확인하였습니다.)
  2. Context와 Answer 데이터를 사용하였으나 이와 관계없는 질문을 생성하였습니다.
  3. Context에 Answer가 여러번 나오는 경우, 다른 위치의 문구를 Answer로 하는 경우가 있었습니다.
- 이를 통해 질문 생성으로 증강된 데이터에 정제 작업을 거쳐야 함을 확인하였습니다.

3.5) 데이터 증강 결론

- 생각보다 정제된 데이터를 사용하면 양질의 데이터를 확보하여 학습 결과가 좋게 나올것이라 생각하였습니다.
- 하지만 결과적으로 데이터 증강을 한 결과가 더 떨어진 것을 확인하였습니다.
- 데이터의 전체적인 에러를 제거한 것보다 일부 노이즈 데이터가 있는 것이 더 좋을 결과가 나올 수 있다는 것을 확인할 수 있었습니다.

---

4. retrieval

4.1) Haystack

- **배경**
  - Baseline code가 보기 어렵고 확장이 힘들다고 생각해서 확장 가능성이 넓은 Haystack이라는 라이브러리를 차용하여 팀 만의 새로운 QA 시스템 Baseline code 설계를 위한 시도를 하였습니다.
  - Haystack은 NLP Framework 중 하나로, QA Pipeline을 쉽게 만들 수 있다는 점과 다양한 **Retrieval-Reader** 전략과 모델을 간편하게 다룰 수 있어 실험하기에 용이하다는 장점이 있습니다.
- **Haystack Workflow**
  1. 검색의 대상이 되는 document들을 **documentStore**에 넣습니다. 이때, **preprocessor**를 추가하여 전처리를 진행할 수 있습니다.
  2. Haystack은 `Query - Retriever - Reader`로 이루어진 파이프라인을 제공하며, 그 외 `Reranking`과 같은 파이프라인을 손쉽게 추가할 수 있어 넓은 확장성을 제공합니다.
  3. Retriever와 Reader를 한눈에 평가할 수 있는 보고서를 지원합니다.
- **결과**
  - Haystack으로 훈련된 Reader가 Baseline code로 훈련한 Reader보다 성능이 좋지 못했습니다.
  - 그 이유로 Baseline code는 Huggingface의 **Transformers**로 훈련을 진행했고, Haystack은 **FARM**이라는 자체 API를 사용했기 때문에 내부적인 디테일이 달랐다고 생각했습니다.
  - Haystack으로 **TF-IDF, BM25, Embedding, DPR** 등 다양한 Retrever를 시도해보았으나 모두 Baseline code보다 성능이 안좋거나 답변이 이상했습니다.
  - 팀원 중 3인이 참여하였고 대략 10일 정도가 개발기간으로 소요되었습니다. 하지만 기존 Baseline code의 추론 결과와 비교하면 같은 모델과 같은 train 데이터를 사용하였음에도 성능에 다소 차이가 있었습니다.
  - 실패 원인을 해결하고자 나름의 가설도 세워가며 low level 파일까지 Custom해보는 등 여러 시도를 해보았지만 끝내 밝히지 못해 결론적으로 실패로 남은 시도였습니다.

No.	Retriever	EM
1	**BM25+Base Reader**	36.25
2	**TF-IDF**(custom) + B**ase Reader**	30.42
3	**TF-IDF**(no_custom) + B**ase Reader**	26.67
4	B**aseline pipeline**	35.42
5	H**aystack pipeline**	29.58

4.2) BM25 / TF-IDF

- 배경
  - TF-IDF
    - 단어의 빈도와 역 문서 빈도를 사용하여 DTM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법으로 특징 추출(Feature extraction) 기법으로 raw data에서 특징을 뽑아내는 것을 목적으로 하고 있습니다.
  - BM25
    - TF-IDF의 문서의 길이를 고려하지 못한다는 점과 TF값이 너무 커져서 관련성이 떨어지는 문서들이 점수가 높아지는 문제를 개선하기 위해 도입하였습니다.
  - 다음 두 Retriever는 MRC Task에서 일반적으로 많이 사용하는 Retriever 입니다.
  - Retriever 테스트를 진행하기전 우선적으로 두 Retriever에 대한 실험을 진행하였습니다.
- 실험

No.	Retriever	EM	F1 Score
1	TF-IDF (top10)	35.4200	48.4100
2	BM25 (top100)	40.4200	52.3500

  - 두 Retriever로 실험을 진행한 결과 `BM25`가 `TF-IDF`보다 더 좋은 결과를 보여주는 것을 확인하였습니다.
  - 이후 Task 실험에 대해 BM25을 우선적으로 적용하여 진행하는 것이 SOTA 모델을 찾을 수 있는 기댓값이 높을 것이라 생각하였습니다.

4.3) Elasticsearch

- **배경**
  - Haystack의 documentStore로 Elasticsearch가 지원되지만, Haystack 도입을 취소하면서 Elasticsearch 사용이 어려워졌습니다.
  - 직접 Elasticsearch를 구현하여 다양한 Retriever에 대해 비교하며 실험을 진행하였습니다.
- **결과**
  - Baseline code의 BM25와 비교했을때, 성능이 떨어지는 것을 확인했습니다.
  - 그 이유는 Elasticsearch는 노리 토크나이저(Nori Tokenizer)를 사용하지만, Baseline code의 BM25는 PTM의 Tokenizer를 사용하기 때문이라고 생각했습니다.
  - Elasticsearch는 다양한 커스터마이징이 가능하지만, 스코어링에 BM25를 사용하기 때문에 결국, Baseline code에서 BM25를 사용하는 것과 마찬가지라고 생각하고 BM25 방향으로 연구를 진행하였습니다.

No.	Retriever	EM	F1 Score
1	BM25	39.17	53.68
2	elasticsearch	26.67 **(-12.50)**	40.40 **(-13.28)**

4.4) Extra Retriever Test (BM25L / BM25+)

- 배경
  - BM25의 긴 문서에 과도한 불이익을 주는, 즉 문서 길이 정규화가 긴 문서보다 짧은 문서를 unfairly하게 선호하는 것을 방지하기 위한 기법으로 `BM25L`을 도입하였습니다.
  - 긴 문서의 처벌(the penalization of long documents)이 `BM25`뿐만 아니라 다른 ranking functions들에서도 발생한다는 것을 관찰 되는 문제를 해결하기 위해 나온 `BM25+`를 추가로 도입하였습니다.
  - 동일한 Reader모델을 사용하여 위 3가지 retriever와 TF-IDF를 비교하는 실험을 진행했습니다.
- 결과
  - 리더보드에서 EM을 기준으로 가장 높은 결과를 보여준 것은 `BM25`로 **58.75**의 점수를 기록하였으며, `BM25+`는 `BM25` 보다 적었지만 **57.50**를 기록하여 두 기법 모두 `TF-IDF`와 비교하였을때 보다 지표상 **17.50** 이상 높은 것을 확인할 수 있었습니다. 그러나 `BM25L`의 경우, `TF-IDF`보다 약 **22.50** 만큼 낮은 점수를 확인할 수 있었습니다.

No.	Retriever	EM	F1 Score
1	TF-IDF	40.00	53.14
2	BM25	**58.75**	**70.39**
3	BM25+	57.50	68.75
4	BM25L	17.50	27.61

4.5) Retriever Ensemble (BM25, BM25+, BM25L)

- 배경
  - 동일한 Reader 모델을 사용하여 Retriever들을 비교하였을 때, 각각 비슷한 Passage를 가져올 수도 있지만 이와 반대로 Retreiver마다 retrieve하는 passage들이 다를 수 있음을 고려하여, **Retrieval Ensemble**을 통해 각각의 Retriever들의 결과를 고려하여 Task를 진행할 수 있도록 실험을 하였습니다.
  - 이번 프로젝트의 평가지표인 EM / F1 Score와 다른 평가 방식으로 Ensemble의 성능을 비교해보고자 새로운 Score 방식을 도입하였습니다.
    - 각 Retriever가 top-k의 Passage를 가져온 결과에 대해서 정답이 포함되어 있는 경우를 전체 Query를 고려해 계산한 Accuracy Score 점수로 각각의 Retriever를 비교하였습니다.
  - query로 사용한 데이터는 Base Dateset으로 주어진 데이터들 중 Answer 정보를 포함하고 있는 Train_dataset을 사용하여 실험을 진행하였습니다.
- 결과
  - 리더보드 결과와 다르게 위 Accuracy Score로 테스트를 진행한 결과 `BM25+`가 약 **0.933**로 가장 높은 Score를 기록한 것을 확인할 수 있었습니다.
  - 위 결과와 `BM25`가 리더보드 제출 결과에서 가장 높은 EM을 보였던 점을 고려했을 때, 정답이 포함된 Passage를 더 많이 포함한다 하더라도 항상 **EM**이 더 높게 나온다는 것을 보장할 수 없다는 것을 알게 되었습니다. 마찬가지로 Retriever Ensemble 결과가 **EM**에서도 `BM25` , `BM25+`보다 낮게 나올지는 직접 확인해보지 않고는 결과를 단정지을 수 없다는 것을 알 수 있었습니다.

No.	Retriever	**Score (accuracy)**
1	TF-IDF	0.796
2	BM25	0.932
3	BM25+	**0.933**
4	BM25L	0.464
5	Ensemble (BM25, BM25+, BM25L)	0.925

---

5. reader

5.1) Bigbird

- 개요
  - 기존 Bert 모델들은 Token 간의 Attention을 계산하여 연산 비용이 크고 메모리 사용량이 많아 긴 Sequence 처리에 어려움이 있었습니다. 그래서 Tokenizing을 진행할 때에도 512자에 맞춰 진행하였지만, BigBird의 경우 Sequence를 작은 블록으로 분할하고, 블록 간의 Attention을 집중적으로 계산함으로써 효율성을 높일 수 있습니다.

    - 이러한 Bigbird 모델의 효율적인 Attention Mechanism을 활용하여 긴 Sequence 처리와 대규모 모델을 가능하게 합니다. 이를 통해 다양한 자연어 처리 작업에 활용된다는 점을 고려하여 이번 MRC Task에도 도입해보고자 하였습니다.
- 실험 내용
    - BigBird의 핵심 아이디어인 Block-Sparse Attention을 이용하여 특정 블록 간의 Attention만 계산하고, 다른 블록들은 건너뛰어 Attention 연산 비용을 줄입니다. 이를 통해 긴 Sequence 처리에 있어서 메모리 사용량과 연산 비용을 크게 절감할 수 있을것으로 기대하였습니다.
    - 또한 BigBird는 Global-Local Attention 패턴을 도입하여 긴 Sequence 처리를 개선합니다. 이 패턴은 각 블록에서 Global Attention과 Local Attention을 혼합하여 사용하는 방식으로, 전체적인 문맥 파악과 지역적인 상세 정보를 모두 고려하였습니다.
    - 모델 탐색 중 한국어 데이터로 Fine-tuning된 모델인 `monologg/kobigbird-bert-base` 을 확인하였고, 이를 활용해 실험을 진행하였습니다.

No.	Model	EM	F1 Score
1	klue/bert-base	35.42	48.41
2	klue/roberta-large	**66.25**	**75.96**
3	monologg/kobigbird-bert-base	45.42 (-20.83)	60.87 (-15.09)

- 결론
    - 기존 bert-base 베이스 모델보다는 좋은 성능을 보였지만 SOTA 모델 중 하나였던 Roberta-large 모델에는 못미치는 결과를 보여주었습니다.
    - 또한 텍스트의 길이가 `4,096자` 까지 수용가능한 모델이지만 **GPU 메모리 한계**로 그에 못미치는 `512자`로 tokenize를 진행한 것도 하나의 이유가 되지 않을까 싶습니다.

5.2) DAPT / TAPT / Finetuning twice

- 배경
    - 도메인 특화 영역에서 Domain knowledge는 **OOV**(out of vocabulary)등의 문제와 함께 일반적인 데이터셋으로만 학습된 사전모델은 이를 커버하지 못한다는 문제점이 있습니다.
    - 이를 해결하기 위해 도메인이 특정된 데이터셋으로 추가적인 further pre-training하여 도메인 영역을 부가하려는 시도들이 있었고 효과를 보았다는 논문 내용을 참고하여 실험을 진행하였습니다.
    - 또한 국내외 기계독해 경진대회에서 우수한 성적을 거둔 **삼성 SDS**의 노하우를 참고하여 두번에 거친 fine-tuning 방법을 채택하였습니다.
- 진행 과정
    - **AI hub의 뉴스 기사 기계독해 데이터**와 **Korpora의 나무 위키 텍스트 데이터**를 합하여 총 `42만개` 데이터를 별도의 전처리 없이 MLM방식으로 `DAPT(domain-adaptive pretraining)` 훈련을 12 epoch 진행
    - **Korquad_v1 데이터**로 정답 부분에 해당하는 영역에만 `[MASK]` 처리를 시킨 후 `TAPT(task-adaptive pretraining)` 훈련을 12 epoch 진행
    - 위키피디아 데이터와 매우 흡사한 카테고리를 갖는 AI hub의 기계독해 데이터를 선별하여 1차 fine-tuning을 진행하였으나 시간 관계상 0.5 epoch를 진행
    - 마지막으로 주어진 대회 데이터셋으로 2차 fine-tuning을 진행
- 결과
    - 2차 fine-tuning까지 진행된 모델은 시간관계상 제출하지 못하였습니다.
    - 따라서, dapt-tapt 까지 진행된 모델로 비교해 보았을땐 베이스가 되는 모델인 klue/roberta-large보다 성능이 개선되었음을 확인 하였습니다 (평가 데이터가 다른점 참고).

No.	Model	EM	F1 Score	평가 데이터
1	klue/roberta-large	56.67	68.49	**test dataset**
2	knlpscience/klue-roberta-large-dapt-tapt	64.58	73.95	**validation dataset**

5.3) Curriculum-learning

- 배경
    - Curriculum Learning은 쉬운 내용을 먼저 학습한 후 점진적으로 어려운 내용을 학습하는 인간의 방식을 모방해 기계학습 과정에서 상대적으로 쉬운 데이터를 먼저 학습한 후 어려운 데이터의 양을 늘려가는 학습 기법입니다.
    - 이러한 방식을 이번 프로젝트에 적용하면 좀더 학습을 잘하지 않을까라는 생각을 하였습니다.
- 방식
    - 본 연구에서는 `klue/roberta-large` 데이터를 1 epoch 학습한 후 동일한 데이터로 evaluation을 거쳐 한번에 학습하지 못한 데이터를 easy data와 hard data로 구분하였습니다.
        - 학습에 사용한 easy data와 hard data는 6 : 4 비율로 선별되었습니다.
- 결과

No.	Model	Data	EM	F1 Score
1	*Nonegom/roberta_finetune_twice*	total	66.67	76.71
2	*Nonegom/roberta_finetune_twice*	easy → total	66.25 **(-0.42)**	75.46 **(-1.25)**
3	klue/roberta-large	easy only	66.25 **(-0.42)**	75.46 **(-1.25)**
4	klue/roberta-large	easy → total	65.42 **(-0.25)**	75.80 **(-0.91)**

    - 분석 결과 쉬운 데이터의 경우 괄호가 없고 어려운 데이터의 경우 괄호가 있는, 즉 annotation 과정에서 생겨난 오류가 존재하여 curriculum learning을 올바르게 수행하는 데에 어려움을 겪었습니다.
    - 최종적으로 단일 모델의 성능 향상은 아니더라도 다양성의 측면에서 다른 모델을 얻었습니다.

---

6. Model Ensemble

- 여러개의 모델의 결과를 하나의 결과로 만들어 각각의 모델에 대한 예측값을 집계하여 결과물을 산출하는 작업을 진행하고자 Ensemble 기법을 활용하였습니다.
- Ensemble 기법을 통해 오버피팅의 경우를 완화하고 좀 더 Task에 맞추어 좋은 결과를 얻을 수 있기에 다양한 voting 방식을 사용한 Ensemble을 진행하였습니다.
    - 모델의 output의 probablity를 활용하였으며 Ensemble의 방법으로 `soft/hard/weighted-soft voting`을 도입하였습니다.
- 각 개별 모델이 전체 데이터에 대한 성능은 조금 떨어지더라도 특정 분야에 대해 뛰어나다면, 이들을 결합함으로써 에러는 줄이고 전체 성능을 향상시킬 수 있다는 가정하에 진행하게 되었습니다.
- 결과

No.	Ensemble	EM	**F1 Score**
1	TF-IDF(40/10) + BM25(40/10) (soft-voting)	70.00	78.56
2	TF-IDF(40/10) + BM25(40/10) (hard-voting)	70.83	81.12
3	TF-IDF(40/10) + BM25(40/10) + Fine-tuned Model (hard-voting)	70.00	80.02

No.	Ensemble	EM	**F1 Score**
4	TF-IDF(40/10) + BM25(40/10) + Fine-tuned Model (soft-voting)	72.08	80.98
5	TF-IDF(40/10) + BM25(40/10) + Curriculum Model (soft-voting)	**72.50**	**81.15**

- 기존에도 Ensemble을 진행하면 모델의 결과가 더 향상된 것을 확인했었는데, 이번 시도에서도 SOTA의 결과를 얻을 수 있었습니다.

---

5. 최종 결과

최종 제출 모델

1. **Ensemble Model**
   - 다음 6개의 모델을 활용하여 다양한 방법으로 hard voting, soft voting을 시도했습니다.

No.	Retrieval	Reader	EM	**F1 Score**
1	BM25 (top10)	*Nonegom/roberta_finetune_twice*	53.75	66.71
2	BM25 (top40)	*Nonegom/roberta_finetune_twice*	66.25	75.96
3	BM25 (top100)	*Nonegom/roberta_finetune_twice*	58.33	68.32
4	TF-IDF (top10)	*Nonegom/roberta_finetune_twice*	-	-
5	TF-IDF (top40)	*Nonegom/roberta_finetune_twice*	-	-
6	BM25 (top40)	*Nonegom/roberta_curriculum_learn*	70.42	78.28
⇒		Ensemble Model	**72.50**	**81.15**

2. **Curriculum Model**

Retrieval	Reader	EM	**F1 Score**
BM25 (top40)	*Nonegom/roberta_curriculum_learn*	70.42	78.28

- **평가 지표**는 **EM** (Sub : F1 Score) 로 진행되었습니다.
  - **public** : **72.5000** ⇒ **private** : **65.2800**

Public Data에 대해 가장 점수가 높았던 결과

🖐 1~6 번째 모델들을 soft voting한 결과가 SOTA

Private Data에 대해 가장 점수가 높았던 결과

🖐 6번째 단일 모델(*Nonegom/roberta_curriculum_learn*)이 SOTA

---

6. 자체 평가 의견

A. 잘한 점들

- 칸반보드, 노션과 같은 다양한 협업 툴을 활용한 것이 좋았습니다.
- 지난 대회보다 팀원 간의 소통이 잘 이루어졌다고 느꼈습니다.
- 모델과 하이퍼 파라미터, 리트리버 등을 '모델 제출 결과'로 잘 공유하여 프로젝트를 원활히 진행할 수 있었습니다.
- 여러 인사이트를 참고하여 프로젝트에 흡수시켜 좋은 결과를 얻을 수 있었다.
- 프로젝트 중 어려운 부분에 대해서 페어코딩과 피드백을 통해 원활히 해결해나갈수 있었습니다.

B. 아쉬웠던 점들

- Generation 모델을 사용해보지 못한 점이 아쉬웠습니다.
- 다양한 Retrieval Task를 도입하지 못한점이 아쉬웠다.
- 여러 이슈들로 인해 시간이 부족했습니다.
- 시간이 어느정도 걸릴지 예상하지 못해서 프로젝트 진행에 지연이 있었던 점이 아쉬웠다.
- 전반적인 프로젝트 계획 설계를 효율적으로 설정하지 못했다.

C. 시도 했으나 잘 되지 않았던 것들

- DPR, Elasticsearch
- haystack
- PEFT (LoRa)
- Retrieval Ensemble

D. 프로젝트를 통해 배운 점 또는 시사점

- 다양한 툴을 사용해 협업하는 경험을 할 수 있었습니다.
- Dataset에 대한 완벽한 이해가 모델 설계 및 학습 결과에 영향을 줄 수 있다.
- 페어 코딩과 같은 협업의 중요성을 다시 한번 느낄 수 있었습니다.
- Task 진행 테스트 또는 경험을 통해 시간이 얼마나 걸릴지 예측하는 것이 프로젝트 전반적인 계획에 도움을 줄 수 있다고 생각합니다.

MRC Wrap-up Report (T5231_황지원)

이번 프로젝트에서 나의 목표

- PM으로서 프로젝트 목표와 계획을 수립하고 진행상황을 모니터링하며 소통하고 조정하는 경험을 해보고 싶습니다.
- Git 방법론에 대해 이해하고 프로젝트마다 적합한 방법론을 적용하기 위한 경험을 하고싶습니다.
- 다양한 데이터를 활용하여 양질의 데이터를 확보하고 이를 바탕으로 모델 개선을 진행하고 싶습니다.
- 데이터 전처리의 중요성을 알고 전처리 유무에 대한 결과 차이를 확인하고 싶습니다.
- 다양한 Reader 모델을 탐색하여 Task에 적합한 SOTA를 얻고자 실험을 진행하고 싶습니다.
- 추가적으로 모델 성능 개선을 위해 다양한 인사이트를 탐색하고 싶습니다.

내 학습목표를 달성하기 위해 진행한 내용

- Git 브랜치 전략에 대해 조사를 진행하였고, 이번 프로젝트에 적합한 Flow를 선정하였습니다.
- 대회 초반에는 강의와 관련 논문을 확인하며 최대한 MRC Task를 이해하려고 노력하였습니다.
- 외부데이터를 활용하고 Hanspell 데이터 증강을 시도하는 등 데이터를 활용한 성능 개선을 하고자 하였습니다.
- 모델을 탐색하여 적합한 tokenize 방식을 선정하고 Task에 적합한 SOTA를 조사하였습니다.
- huggingface에 익숙해지는 것을 목표로 하였습니다.

연구 및 실험 내용

1. **데이터 분석 및 전처리, 증강 작업**
   - Raw Data를 분석하여 문서 길이에 대한 tokenize 전략을 수립하였고, 다른 모델을 실험하는데 참고하도록 하였습니다.
   - `KorQuAD`, `AI Hub` 등 외부데이터를 사용하여 양질의 데이터를 수집하였습니다.
   - Hanspell 데이터 증강 방식을 활용해 노이즈를 제거하는 작업을 진행하였습니다.
   - 특수문자, 전각문자 등 Tokenize 과정에서 학습에 영향을 줄 수 있는 부분들은 전처리를 context의 노이즈를 제거하는 작업을 진행하였습니다.
   - **결과**
     - 노이즈를 제거한 정제된 데이터로 학습시킨 결과가 생각보다 좋지 않았습니다. 이를 바탕으로 언어모델을 학습시키는데 일부 노이즈가 포함된 데이터가 더 결과가 좋다는 것을 확인할 수 있었습니다.
     - 외부데이터를 활용한 데이터 증강 작업과 Task에 적합한 데이터를 탐색하였는데 결과가 생각보다 좋지는 않았습니다. 이를 바탕으로 각 Task마다 적합한 데이터가 따로있고, 그에 맞춰 기준을 선정해야 한다는 점을 다시 확인하였습니다.
2. **Reader 모델 연구 진행**
   - Reader 모델 연구를 위해 Roberta-large, Bigbird 모델에 대해 조사를 진행하였고, 각 모델별로 테스트를 진행하였습니다.
     - Roberta-large : `klue/roberta-large`, Bigbird : `monologg/kobigbird-bert-base`
   - 모델의 SOTA를 확인하기 위해 파라미터 튜닝을 진행하였습니다.
   - 결과
     - 효율적인 Attention Mechanism을 활용하여 긴 Sequence 처리와 대규모 모델을 가능하다는 장점을 보여 bigbird 모델에 기대를 가지고 테스트를 진행하였는데, 베이스 모델보다는 좋은 성능을 보였지만 Roberta-large 모델이 역시 더 좋은 결과를 보여주었습니다.
     - 서버 Infra의 한계로 Tokenizer에 큰 Sequence로 테스트를 진행하지 못한점이 아쉽게 느껴졌습니다.
3. **DPR (Dense Passage Retrieval)**
   - ODQA Task에서 기존의 Sparse Retrieval을 능가하는 성능을 보이는 DPR 모델을 확인하였고, 이를 저희 프로젝트에 접목하기 위해 논문을 확인하고 서칭을 진행하였습니다.
   - DPR 모델 테스트를 진행하기 위해 `KoDPR` 오픈소스를 이용하여 테스트를 진행하였고, `ko-wiki` 데이터 전체를 사용해 입력한 문장에 대해 원하는 결과가 나오는 것을 볼 수 있었습니다.
   - 결과
     - 테스트를 진행할 때에는 1차적으로 적절한 Passage를 가져오는 것을 확인하였지만, 저희 프로젝트 환경에 접목하는 과정에서 역량부족으로 에러를 해결하지 못해 결국 적용하지 못했습니다.
     - 테스트로 마쳤지만 완성하지 못한 부분에서는 아쉽게 느껴졌습니다.
4. **Model Ensemble**
   - 프로젝트 과정에서 팀 전체가 다양한 테스트를 통해 얻은 모델들의 결과를 확인해보았습니다.
   - 기존 Roberta 모델보다 향상된 결과도 있었고, 부족한 부분도 있었지만 다양한 인사이트를 가져올 수 있었고 오버피팅 문제를 완화하고 더 좋은 결과를 얻기위해 voting 방식을 사용한 Ensemble을 진행하였습니다.
     - Ensemble의 방법으로 `soft/hard/weighted-soft voting`을 도입하였습니다.
   - 모델의 output의 probablity를 활용하여 다양한 `앙상블 기법`으로 마무리 작업을 하였습니다.
   - 결과
     - 기존에도 Ensemble을 진행하면 모델의 결과가 더 향상된 것을 확인했었는데, 이번 시도에서도 SOTA의 결과를 얻을 수 있었습니다.
     - 다만 Public이 아닌 Private 데이터로 다시 확인할 때에는 결과가 많이 감소한 것을 보았고, Ensemble을 진행해도 일반화의 문제는 해결할 수 없다는 생각이 들었습니다.

이번 프로젝트의 새로운 시도

- 브랜치 운영 및 버전 관리에 대해 이전부터 중요하게 생각하였는데 신경써서 관리했던 것이 팀원 전체가 안정적으로 프로젝트 진행하는데 많은 부분 도움이 되었다고 생각합니다.
- 이번에는 이전 프로젝트와 다르게 데이터를 우선적으로 접근하고 탐색하는 과정을 진행하였습니다. 그 과정에서 데이터 전처리, 샘플링 작업과 외부데이터 활용에 대해 다양한 실험을 진행할 수 있었습니다.
- 또한 기존에 잘 사용된 모델이 아닌 `Bigbird` , `DPR` 모델과 같이 Task에 특화된 모델을 서칭하고 실험하는 과정을 진행하였고 역량을 향상시킬 수 있던 계기가 있었다고 생각합니다.

이번 프로젝트의 한계점

- PM의 역할이라고 생각하고 진행했던 작업들이 되돌아 생각해보니 PL의 업무와 다를 것이 없었다고 생각하였습니다. 둘다 프로젝트에서 중요한 역할이지만 좀더 위 직무에 대해 조사를 해볼 필요가 있다고 생각하였습니다.
- 데이터에 대해 집중해서 다양한 시도를 하다보니 모델 부분에 대해 테스트를 진행을 많이 하지 못하였고, 또한 그렇게 테스트를 진행한 데이터 부분에서 좋은 결과를 얻지 못하였음에도 놓지 못하고 계속 잡고 있던 것이 되돌아 생각하면 아쉽게 느껴졌습니다.
- Retrieval 또는 Reader 부분에 대해 하나만 깊게 연구하기 보다는 각각 서칭하고 싶은 부분들에 대해 실험을 진행한 것에 원하는 연구 결과를 얻을 수 없었던 이유이지 않을까 라는 생각을 하였습니다.
- huggingface를 적극적으로 활용해보고 싶었는데 모델 및 데이터를 서칭하는데는 사용했지만 실험한 모델이나 데이터를 로드하는 부분의 Task는 진행하지 못한 점에서는 아쉽게 느껴졌습니다.

느낀점과 다음 프로젝트에 시도해볼 내용

🖐 모든 프로젝트를 마치고 되돌아보면 강의를 통해 배운 도메인 지식에 대해 직접 테스트를 하고 적용하며 온전히 나만의 실험을 진행했다는 점에서 많은 점을 배울 수 있었습니다. 또한 멘토님과 동료들의 피드백을 통해 부족한점을 채워나갈 수 있었고, 더 성장할 수 있는 계기가 되었다고 생각합니다.

아쉬운 점으론 이번 프로젝트에서는 다양한 시도를 한 것은 맞지만 MRC Task에 대한 난이도 때문인지 많은 실험을 성공적으로 마무리 하지 못했습니다. 하지만 팀원들의 인사이트를 바탕으로 배워가는 것도 있었고 배운 것도 많았다고 생각합니다.

마지막 최종 프로젝트를 앞두고 모델 테스트를 경험한 것도 있지만 브랜치 규칙 및 코드리뷰와 관련해서도 협업적인 부분에서 step by step으로 진행하였고 이를 바탕으로 프로젝트를 진행하는데 무리가 없을 것이라 생각합니다.

또한 최종프로젝트가 마지막 프로젝트인 만큼 후회없이 제가 하고 싶은 공부 및 실험을 하고 싶다는 생각을 하였습니다.