**<프로젝트의 방향성>**

이번 stage2 이후의 제가 선택한 주제들이 Object Detection, 수식인식기로 CV에 관한 주제들이 남았습니다.

따라서 이번 stage2가 저에게는 NLP에만 초점을 맞추어 진행할 수 있는 유일한 시간이라는 생각으로 깊고 섬세하게 보자는 목표를 잡고 진행하였습니다.

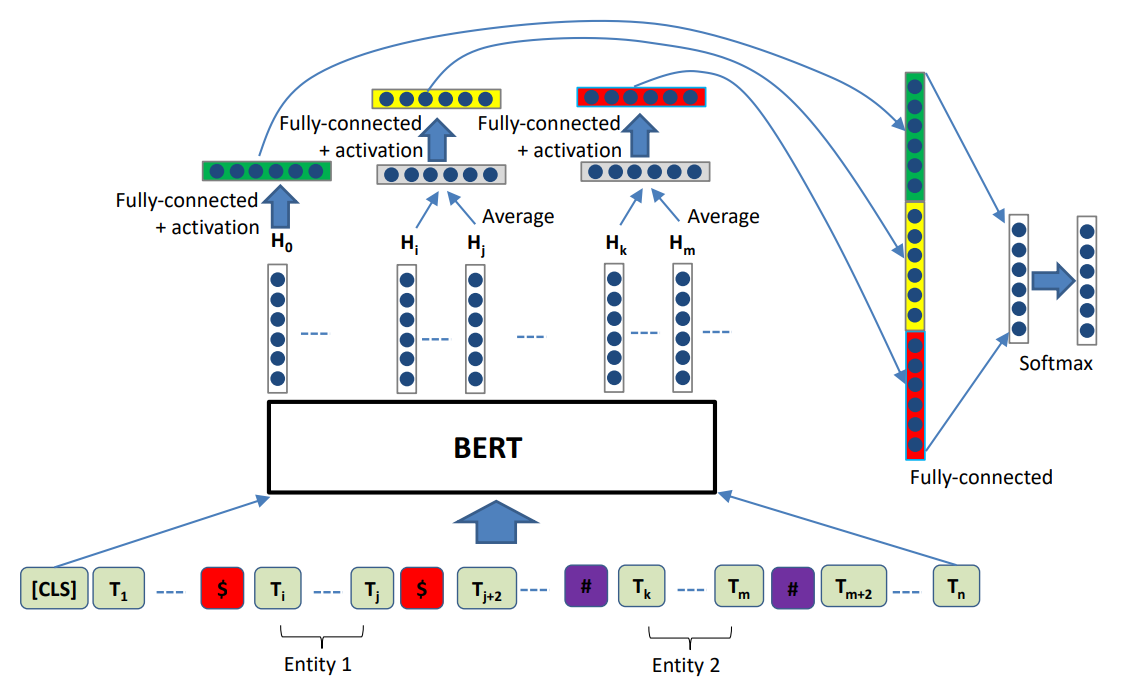
**<기술적인 도전>**

### 본인의 점수 및 순위

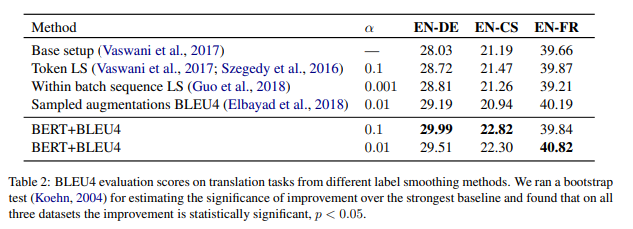
LB 점수 80.4, 26등

### 진행 과정

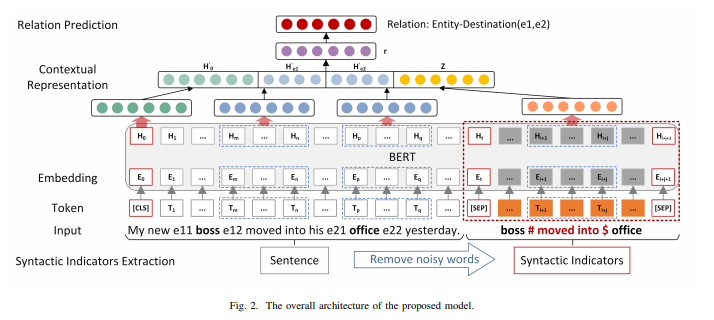
* Baseline을 통해 여러 모델을 바꾸어가며 진행 : kobert (acc. 58.8%) etc..
  + Pretrain을 받아오지 못하는 것을 알게되고 pretrain으로 진행시 acc. 71.7%
* Train Data를 들여다보니 영어와 한자 등이 다수 포함 되어있었고, multi-lingual model을 사용해야 겠다는 생각이 들었다.
* Baseline이 하이레벨 api로 구성이 되어있어 커스터마이징하기에 많은 불편함이 있어 relation extraction에 대해서 찾아보았고, R-BERT(https://github.com/monologg/R-BERT, Enriching Pre-trained Language Model with Entity Information for Relation Classification : https://arxiv.org/abs/1905.08284)를 참고하여 모델을 새로 구성하였다.



* 모델을 구성하여 학습을 진행하려 했지만 pretrained weight,bias를 상속받아오지 못하는 warning log가 엄청많이 발생하였다.
  + 해결법을 찾으며 코드를 자세히 들여다보다가 BertPretrainedModel class를 상속받아 구현되어진 것을 보고 BertModel class를 상속받아오도록 수정하고 내부적으로 pretrained를 호출하게 구성하여 해결했다.
  + Kobert acc. 73%, XLM-Roberta-large acc. 77.8%
    - Bert계열이 아닌 model적용시 에러가 발생하는 이유가 뭘까?
      * Transformers에서 bertforclassification, electraforclassification 함수의 내부 code를 확인했다.
      * Bert계열은 pooler\_output(CLS를 의미), last\_hidden\_state를 return.
      * Bert가 아닌 계열은 last\_hidden\_state만 return.
      * 따라서 last\_hidden\_state[:,0,:]를 통해 CLS임베딩을 뽑아내야 한다.
        + Koelectra acc. 68.6%
        + 이 방식으로 CLS임베딩을 뽑아내면 acc기준 60%후반대의 성능을 보이며 bert계열의 모델보다 크게 성능을 내지 못했다.
  + Maxlen이 어떤 영향을 끼칠까?
    - Train set max len : 345
    - Test set max len : 226
    - Test set max len에 비슷하도록 학습을 진행하면 성능이 좋아지지 않을까?
      * Treain set max len을 250으로 조절
      * R-BERT모델은 Entity가 빠지면 안되므로 250보다 긴 문장은 학습에서 제외 (9000 set -> 8990 set)
      * 결과는 acc기준 77.8% -> 74.2%
  + https://github.com/monologg/R-BERT 에서는 entity layer를 하나만 가지고 두개의 entity가 공유하도록 구성되어진 것을 확인했다.
    - 이 Layer를 나누어 주면 성능이 오르지 않을까?
      * Share acc. 77.8% -> Split acc. 80.4%
      * 이후로는 split layer를 통해 실험이 진행되었다.
      * 하나의 layer로 share하도록 구성되어진 이유가 뭘까?
        + Layer가 늘어나면 GPU메모리가 더 필요하게 된다.
        + 이를 줄여주기위해 이렇게 사용한 것이 아닐까? 하는 예상.
  + BERT모델은 GELU activation을 활용하여 학습이 진행되어진다.
    - 그럼 R-BERT에서도 Tanh대신 GELU를 사용하여 activation을 진행하면 어떨까?
      * Tanh Acc. 80.4% -> GELU Acc. 77.0%
  + 일반화를 높이기 위해 Cross Entropy loss를 Smooth해서 사용하면 어떨까?
    - <https://www.aclweb.org/anthology/2020.emnlp-main.405.pdf>



* + - 해당 논문을 참고하여 0.05 smoothing을 주어 실험해보았다.
    - Cross Entropy Acc. 80.4% -> Smoothing Acc. 78.5%
  + Papers with code(https://paperswithcode.com/sota/relation-extraction-on-semeval-2010-task-8)의 RANK를 보면 R-BERT는 RANK 7을 기록
    - RANK가 높은 모델을 사용하면 더 좋지않을까?
      * RANK4 Skeleton-Aware BERT(Enhancing Relation Extraction Using Syntactic Indicators and Sentential Contexts)가 R-BERT에서 Syntatic Indicators만 적용하면 되는 기술이라 적용하는데에 오래 걸리지 않을 것같다.



* + - * Modeling을 끝내고 batch 16으로 학습진행을 하려했지만 layer가 하나 늘어나게 되어 OOM이 발생하였다.
        + 어떻게 해결할까?

Batch 16 -> 8로 낮추어 학습 : OOM발생

FP32를 FP16으로 진행하면 어떨까?

FP16으로 적용하게 되면 성능적인 손해가 심할 것 같다.

U stage에서 배운 최적화 기술 중 쉽게 적용할 수 있는 방법이 있을까?

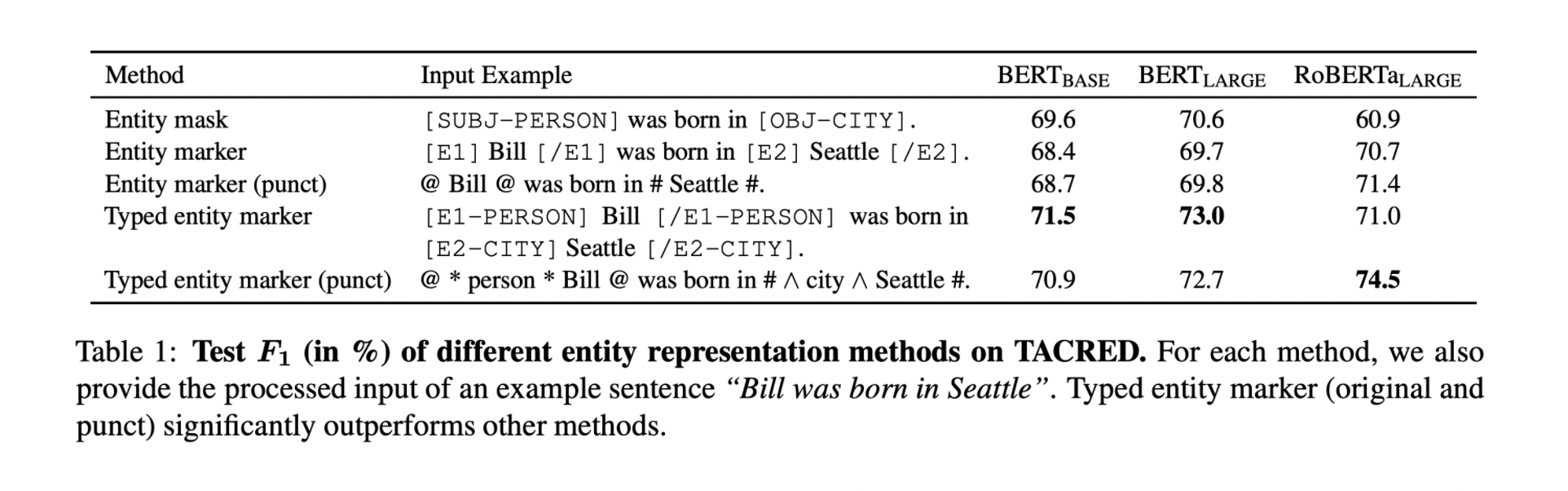
Pytorch 1.6버전에서 내장되어진 FP16과 FP32를 적절히 섞어주는 AMP(Automatic Mixed Precision)을 발견했고, 적용하는데에도 쉬웠다.(https://pytorch.org/docs/stable/notes/amp\_examples.html)

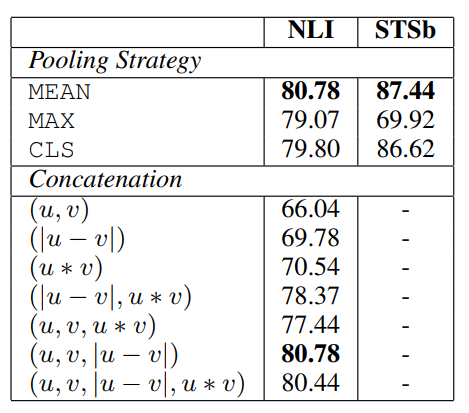
AMP를 적용하여도 batch 16으로는 진행이 불가하여 batch 8로 진행하였다.

Acc. 74.8%

모델을 large에서 base로 pretrain모델 성능을 조금 포기하고, model구조의 성능을 통해 올려볼수 있지 않을까?

XLM-Roberta-base, batch 8, no AMP : Acc. 75.6%

* + R-BERT structure with XLM-Roberta-large Pretrained로 돌아오게 되었다.
  + Roberta에 대해 자세히 설명된 글(<https://brunch.co.kr/@choseunghyek/7>)을 찾았고, batch size가 성능과 완전한 비례관계는 아니지만, 크면 클수록 성능이 좋아지는 경향이 있다고 한다.
    - 토론 게시판에 비슷하게 Batch size에 관한 글이 올라왔고, 작을수록 성능이 좋다는 의견이 있었다.
    - 뭐가 맞을까?
      * Batch 16 Acc. 80.4% -> Batch 8 Acc. 77.7%
  + 토론 게시판에 NER tag를 포함한 relation extraction에 대한 글이 올라왔다.
    - 오수지님께서 친절하게 special token과 NER tag를 추가한 dataset을 올려주셨다.
      * No NER Acc. 80.4% -> NER tag + NER special token Acc. 78.0%
    - NER special token을 빼고 NER tag만 적용하면 어떨까?
      * No NER Acc. 80.4% -> NER tag Acc. 77.9%
  + 기존 Train Data에서 잘못 labeling되어진 부분들을 master님께서 수정해서 올려주셨다.
    - No Modify data Acc. 80.4% -> Modify data Acc. 77.5%
    - 10개도안되는 dataset을 바꾸어 적용했는데에 비해 Acc가 너무 크게 감소했다.
    - 이전에 인턴 때 “현대 차 리뷰 긍정/부정/중립 감성분석”을 진행시에도 이와 비슷한 현상이 있었다.
    - Train data에 “역시 현대차는 비싸”와 비슷한 늬앙스의 문장에 대해 담당자님께서 “긍정”이라고 라벨링을 하셨고, 이에 대해 여쭈어 보았다.
    - 당시 현대 차 담당자분께서 브랜드적인 가치가 높게 평가되어지는 것이 현대차 내부에서는 긍정적으로 판단한다는 말씀을 들었고, 크게 생각을 바꿀수 있던 적이 있었다.
    - 지금 Train Data에서도 처음 laber분들께서 labeling할때의 기준에 맞추어 Train, Test labeling이 진행되어졌을 것이고, 시간이 지난 현재 이를 수정한다면 라벨링 당시의 편향에 맞추어지지 않을 것 같아 수정전의 Data로 나머지 실험들은 진행하였다.
  + 모델을 개선할 방법을 찾던 중, Sentence-BERT에 관한 글을 발견했다. (<https://velog.io/@ysn003/%EB%85%BC%EB%AC%B8-Sentence-BERT-Sentence-Embeddings-using-Siamese-BERT-Networks>)
    - 내용이 흥미로워 짧게 읽던 중, 지금 나의 모델에 적용할수 있는 부분을 발견하게 되었다.
    - R-BERT구조에서 CLS, Entity1, Entity2임베딩을 concat하여 Fully Connect Layer를 통과하게 되는데 이 Concat방법을 여러가지로 적용할수 있었다.

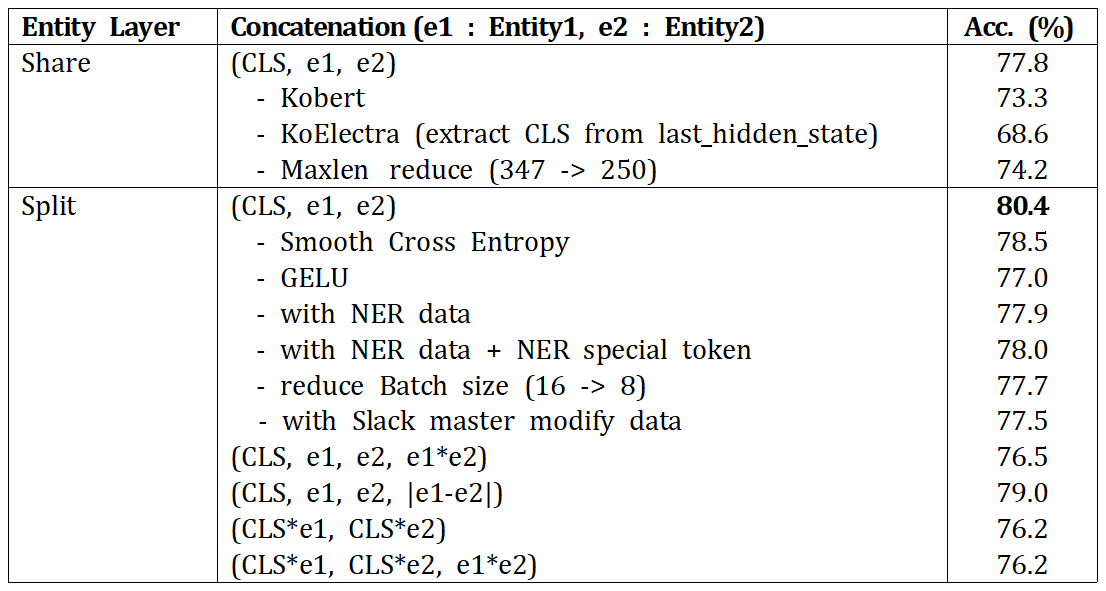


* + - 이에 대해 조원분들과 같이 분석을 진행하였고, 각자 적용해 볼 수있었다.
    - 대회 마지막쯔음 발견하게 되어 전부 실험해보기에는 시간이 많이 부족했지만 몇몇개는 실험해 볼 수 있었으며, 결과는 아래의 표에 기록해 두었다.
    - Train data에서 라벨 0이 ‘관계없음’으로 가장 많았고, 내적을 통해 유사도가 0에 가까울수록 관계없음으로 예측을 잘할 것이라는 가설을 바탕으로 내적이 가장 높은 정확도를 보일 것 같았지만 예상과는 다르게 두 임베딩간의 거리가 가장 높은 결과를 보였다.

### 사용한 모델 아키텍처 및 하이퍼 파라미터 (상위 2개)

1. 아키텍처: R-XLMRoberta
   1. LB 점수 : 80.4% (Accuracy)
   2. Base Pretrained Model : XLM-Roberta-large
   3. Hyperparameters
      1. Batch size : 16
      2. Num labels : 42
      3. Weight decay : 0.0
      4. Learning rate : 2e-5
      5. Warmup step : 0
      6. Max grad norm : 1.0
      7. Gradient accumulation step : 1
      8. Dropout rate : 0.1
      9. Max len : 347
   4. Activation : Tanh
   5. Optimizer : AdamW
   6. Concatenation : (CLS, Entity1, Entity2)
2. 아키텍처: R-XLMRoberta
   1. LB 점수 : 79.0% (Accuracy)
   2. Base Pretrained Model : XLM-Roberta-large
   3. Hyperparameters
      1. Batch size : 16
      2. Num labels : 42
      3. Weight decay : 0.0
      4. Learning rate : 2e-5
      5. Warmup step : 0
      6. Max grad norm : 1.0
      7. Gradient accumulation step : 1
      8. Dropout rate : 0.1
      9. Max len : 347
   4. Activation : Tanh
   5. Optimizer : Adamw
   6. Concatenation : (CLS, Entity1, Entity2, | Entity1 – Entity2 |)

### 결과 정리



### 앙상블 방법

* 점수적인 면보다는 다양한 실험 결과가 궁금하여 앙상블을 진행하지 못했습니다.

### 

**<학습과정에서의 교훈>**

### 학습과 관련하여 개인이 얻은 교훈

* 많은 실험과 자료들을 통해 시야를 크게 넓힐 수 있는 좋은 기회였습니다.

### 피어세션을 진행하며 좋았던 부분과 동료로부터 배운 부분

* 피어세션을 통해 다양한 시각과 실험결과들을 얻을 수 있었습니다.
* 위에서 얻은 정보들을 토대로 다양한 가설을 세워 볼 수 있었습니다.
* 네트워킹이 중요하다는 생각을 다시한번 느꼈습니다.

**<마주한 한계와 도전숙제>**

### 아쉬웠던 점들

* 개인적으로 구성한 Validation을 통해 정확도를 체크하는 것 보다 조금더 정확한 결과를 원해 제출을 통해 결과를 확인했지만, 제출기회가 너무 적었습니다.
* 제출기회도 적었지만, 기간이 짧아 아직 시도해보지 못한 가설들이 많아 아쉽습니다. (ex. R-BERT에서 Entity 임베딩을 위해 구성된 average에 last\_hidden\_state를 넣어주어 CLS를 뽑아내어 진행하는 실험, 다양한 concatenation 등)
* Syntatic Indicator를 추가하여 진행하는 실험에서 OOM이 발생하여 AMP를 사용하여 진행할 수 밖에 없던 점이 아쉽습니다.
* GPU가 조금 더 가능하거나, 좋은 최적화 기술을 구현 할 수 있었다면 NER with NER special token방법 진행시에 NER layer만 따로 추가하여 실험해 볼수도 있었지만, 해보지 못한점이 아쉽습니다.
* 리더보드 마감 2일전에 2등을 기록하고 있었지만, 앙상블이 시작되면서 최상위권에서 밀려내려온 점도 아쉽습니다.
* 위와 비슷한 내용이지만 앙상블을 시도해보지 못한점 또한 아쉽습니다.

### 한계/교훈을 바탕으로 다음 스테이지에서 새롭게 시도해볼 것

* 다음 스테이지 부터는 팀 프로젝트로 진행이 되므로 하나의 가설을 세울 때에도 팀원들과 의견을 주고받아 섬세한 가설을 세우도록 시도해볼 것입니다.
* 처음 모델을 구성할때에도 GPU memory적으로도 조금 생각을 해나아가며 줄여볼 수 있을 것 같은 부분들에 대해서는 줄여가며 모델을 구성해 볼 것입니다.