KLUE Relation Extraction Project

8 조 MIML

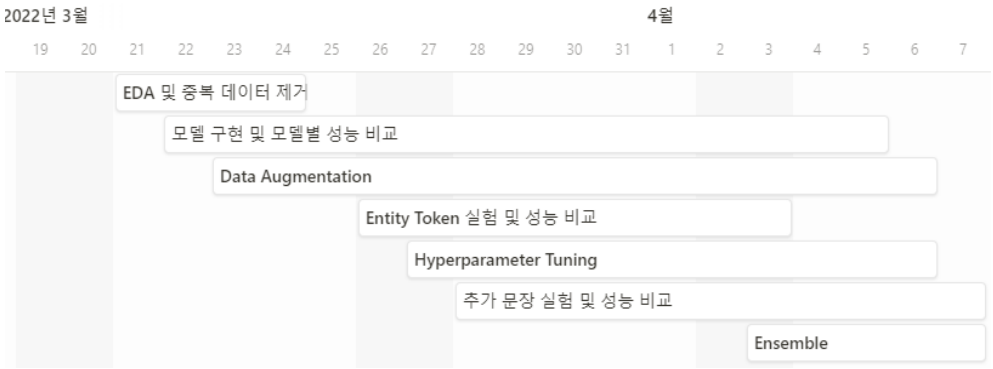
[1] 프로젝트 주제

문장의 단어(Entity)에 대한 속성과 관계를 예측하는 관계 추출(Relation Extraction) 문제
문장, 단어에 대한 정보를 통해 문장 속에서 단어 사이의 관계를 추론하는 모델 학습

[2] 프로젝트 팀 구성 및 역할

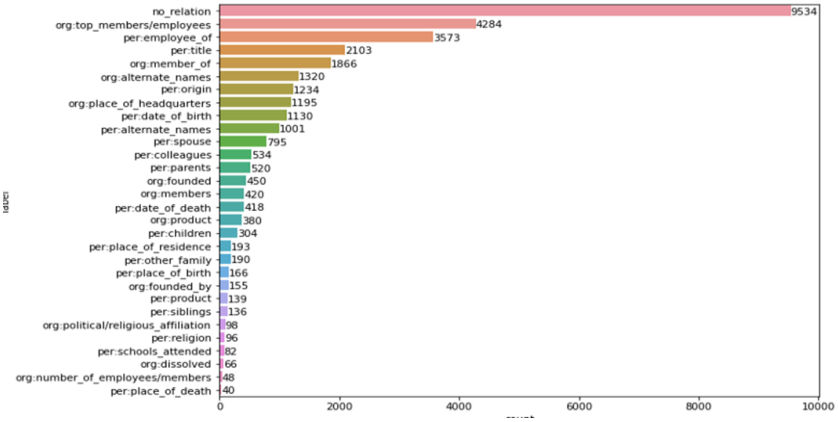
조원	역할
임동진	모델 구현, Github code refactoring
정재윤	Data 정제, Data augmentation, Hyperparameter tuning
조설아	Entity token 별 & 추가 문장 성능 실험, Voting
허치영	모델 구현, Loss function 성능 비교, Entity token 별 성능 실험
이보림	Entity token 별 &추가 문장 성능 실험, Ensemble, Voting, Loss

[3] 프로젝트 수행 절차 및 방법



1. EDA 및 중복 데이터 제거

전체 32470 개 데이터 중에 sentence, subject entity, object entity 가 중복되는 문장이 93 개 존재했습니다.
이외에도 mislabel data 가 존재를 확인한 후 label 을 고려하여 제거한 후 데이터셋을 재구성했습니다. 또한 data imbalance 가 극심해 이를 해결하는 것이 이번 대회의 핵심이라고 생각했습니다.
확인된 중복 데이터 중 sentence 와 entity 가 같고 label 이 no_relation 인 데이터를 제거했습니다. no_relation 이외의 중복 데이터는 남겨두었을 때 raw data 대비 micro_f1 score 2 점 가량 성능 향상이 있었습니다



2. Data Augmentation

1) EDA(Easy Data Augmentation)

SR(Synonym Replacement), RI(Random Insertion), RS(Random Swap), RD(Random Deletion) 중에서 SR 과 RI 는 유의어로 대체되거나 새로운 단어가 삽입되면서 문장의 문맥을 해치는 경우가 존재하여 성능을 오히려 하락시켰습니다. 이번 프로젝트에서는 약간의 성능 상승을 보인 **RS 와 RD 만을 사용**했습니다.

2) Back Translation

Papago 번역기를 crawling 을 활용해 데이터를 증강시켰습니다. 한국어->영어->한국어, 한국어->일본어->한국어, 한국어->중국어->영어->한국어 총 네 가지 경우에 대해 실험했으며, 모든 경우에서 약 1 점 가량 성능이 하락하여 최종 제출 모델에서는 사용하지 않았습니다.

Back translation 결과 문장에서 entity word 가 유의어로 바뀌는 경우가 잦아 좋은 성능을 얻어내지 못한 것으로 예측되며, crawling 을 사용했기에 한 번에 많은 시간이 걸려 충분한 실험을 진행하지 못했습니다.

3. 모델 구현 및 모델 별 성능 비교

1) Pre-trained Model

- KLUE/BERT-base (66.8111) : Baseline 성능 확인용
- KoBERT (67.3634) : 다른 한국어 pre-trained model 의 성능 확인 실험
- Multilingual BERT (53.2452) : 데이터셋에 다른 언어가 포함된 경우가 있어 여러 언어로 pre-train 된 모델의 성능 확인 실험
- KLUE/RoBERTa-large **(69.0621)** : 한국어 pre-trained model 의 크기에 따른 성능 향상 비교 실험
- XLM-RoBERTa-large (68.6212) : KLUE/RoBERTa-large 보다 더 큰 크기의 모델의 성능 실험

실험 결과 KLUE/RoBERTa-large 의 성능이 가장 좋게 나왔습니다. XLM-RoBERTa-large 를 제외한 나머지 모델의 경우 KLUE/RoBERTa-large 에 비해 작은 크기의 모델로 성능 향상폭이 크지 않았으며, XLM-RoBERTa-large 의 경우 다국어 pre-trained model 로 한국어 특화 모델인 KLUE/RoBERTa-large 에 비해 약간 낮은 성능을 보였습니다.

2) Classifier 변경

Method	실험 이유	결과 (micro_f1)	평가
Classifier 만 1 회 학습 후 모델 전체 train	Classifier 의 첫 학습으로 인한 큰 폭의 weight update 를 방지하고자 실험	65.6608	학습으로 인해 parameter update 방향이 이상해진 듯함
*Entity token 활용 classifier	Entity token 의 정보를 활용해서 분류하는 것이 성능에 영향을 줄 것이라 생각	**75.0477**	Entity token 에 담긴 entity 정보로 인해 대폭 성능향상됨

*손수현(2021),「엔터티 위치 정보를 활용한 한국어 관계추출 모델 비교 및 분석」, 한국정보과학회 언어공학연구회, 247-250p

4. Entity Token 실험 및 성능 비교

문장 내 entity 를 강조해 모델의 입력으로 사용해야 된다고 판단되어 entity 강조 방법을 고안했습니다. Special token 을 추가하는 방법을 포함해서 총 7 가지 entity token 삽입 방식을 고안해냈고, 7 가지 방법 모두에 대해 실험을 진행했습니다.

< Entity Token 추가 방식 목록 >

방식	원문 : 이순신 장군은 조선 출신이다.	Special Token 유무
Entity Marker	[subj]이순신[/subj] 장군은 [obj]조선[/obj] 출신이다.	O
Typed Entity Marker	[subj:PER]이순신[/subj] 장군은 [obj:ORG]조선[/obj] 출신이다.	O
Entity Mask	[subj:PER] 장군은 [obj:ORG] 출신이다.	O
Type Marker	[PER]이순신[/PER] 장군은 [ORG]조선[/ORG] 출신이다.	O
Typed Entity Marker (punct)	@*사람*이순신@ 장군은 #^단체^조선# 출신이다.	X

Typed Entity Marker_ordering (punct)	**@*사람*이순신@ 장군은 #^단체^조선# 출신이다.**	**X**
Typed Entity Suffix	*이순신[사람]* 장군은 *조선[단체]* 출신이다.	X

그 중 가장 높은 성능 개선을 이끌어낸 방식은 Typed Entity Marker_ordering (punct) 방식으로 micro_f1 score 7 점을 상승시켰습니다.

5. 추가 문장 실험 및 성능 비교

BERT 모델의 pre-train 방식과 유사한 입력을 만들고, 입력 문장에 더 많은 정보를 담아 학습을 용이하게 하고자 했습니다. KLUE/RoBERTa-large 모델에서도 KLUE/BERT-base 모델과 동일한 성능 향상을 가져왔습니다.

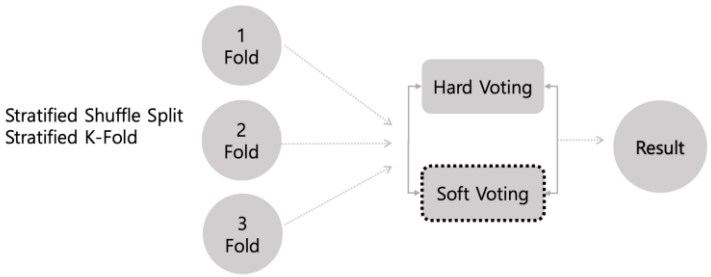
< 문장 추가 방식 목록 >

추가 문장 종류	Example
Baseline	Object Entity[SEP]Subject Entity
Without Token	이 문장에서 Subject Entity 와 Object Entity 의 관계를 고르시오.
Entity Marker_ver.1 (punct)	이 문장에서 *Subject Entity*와 *Object Entity*의 관계를 고르시오.
Typed Entity Marker_ver.1 (punct)	이 문장에서 *Subject Entity[Subject Type]*와 *Object Entity [Object Type]*의 관계를 고르시오.
Entity Marker_ver.2 (punct)	이 문장에서 @Subject Entity@와 #Object Entity#의 관계를 고르시오.
Typed Entity Marker_ver.2 (punct)	이 문장에서 @Subject Entity*Subject Type*@와 #Object Entity^Object Type^#의 관계를 고르시오.
Entity Marker with Type (only Object)	**이 문장에서 [Object Entity]는 [Subject Entity]의 [Object Type]이다.**
Entity Marker with Type	이 문장에서 [Object Entity]는 [Subject Type]인 [Subject Entity]의 [Object Type]이다.

문장 부호를 사용하여 entity 를 강조하고, 추가적인 정보를 제공할 수록 성능이 향상되었습니다.
각 entity 가 subject 인지 object 인지보다 entity 의 순서가 한국어 문장 구조에서 정보 추출에 유리함을 알 수 있었습니다.
Relation extraction 에서 object 의 의미가 관계의 분류에 있어 더 중요한 역할을 한다는 것을 알 수 있었습니다

6. Ensemble

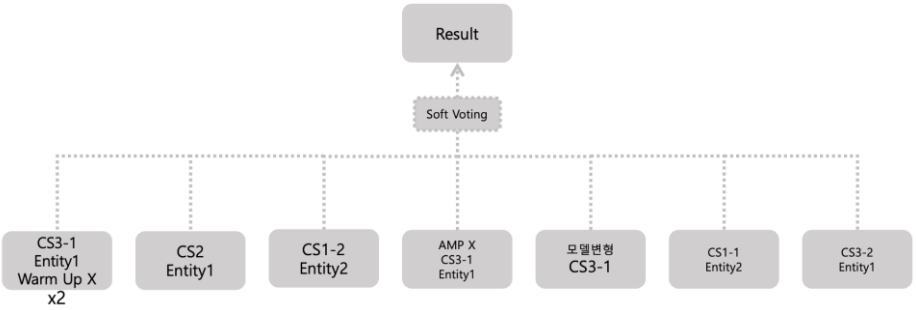
1) Fold 별 ensemble



주어진 train dataset 의 class 별 비율을 고려하여 9:1 비율로 각기 다른 3 개의 train, validation set 으로 나누고, 각각의 경우에 대해 학습한 모델들을 voting 을 통해 최종 예측 결과를 만들었습니다.
그 결과 micro_f1 score 3 점 가량 성능 향상을 보였으며 비슷한 방식의 K-Fold validation 기법보다 0.2 점 더 높은 향상을 보였습니다.
모든 데이터셋의 활용보다 다양한 데이터셋을 활용한 voting 방식이 더 효과적인 성능 향상을 불러옴을 알 수 있었습니다.

2) 성능 검증된 모델 내에서의 ensemble

각 모델들의 성능 차이가 크지 않아 상위 몇 개 모델들의 ensemble 은 성능의 향상을 크게 가져오지 못하리라 예상하여 리더 보드 기준 75 점 이상의 성능을 나타내는 모델 중 각기 다른 기법을 적용한 7 개의 모델에 대해서 soft voting 을 진행했고, 개별 모델보다 1 점 가량 성능 향상을 보였습니다.



3) Ensemble 사용 모델

base Hyperparameter (KLUE/RoBERTa-large model)

- learning rate : 2e-5
- epochs : 3
- warm up step : 400
- loss : CrossEntropy loss

< 사용 모델 >

모델	Entity Token	추가 문장	Details
KLUE/RoBERTa-large	Typed Entity Marker_ordering (punct)	Entity Marker with Type (only Object)	Warm up X x2 weighted ensemble
KLUE/RoBERTa-large	Typed Entity Marker_ordering (punct)	Entity Marker_ver.2 (punct) 변형	추가 문장을 질문 형태로 수정하여 사용
KLUE/RoBERTa-large	Typed Entity Suffix	Typed Entity Marker_ver.1 (punct)	
KLUE/RoBERTa-large	Typed Entity Marker_ordering (punct)	Entity Marker with Type (only Object)	3 Fold 로 나눈 데이터셋 각각을 학습한 모델 3 개 soft voting 한 모델
KLUE/RoBERTa-large 변형	Typed Entity Marker_ordering (punct)	Entity Marker with Type (only Object)	분류 시 [CLS] 토큰이 아닌 entity token 을 concatenate 하여 사용
KLUE/RoBERTa-large	Typed Entity Suffix	Entity Marker_ver.1 (punct)	amp 제거, 팀 내 private score 1 위
KLUE/RoBERTa-large	Typed Entity Marker_ordering (punct)	Entity Marker with Type	

[4] 자체 평가 의견

모든 팀원이 열정적으로 아이디어를 제안하며 많은 실험을 진행했습니다. 매일 피어세션에서 수시로 업무 진행 상황을 공유하며 각자의 업무에 충실하여 진행한 덕에 좋은 결과를 얻어냈습니다. 다만 대회 첫 주에 바로 협업을 진행하지 않고 각자 학습하다 보니 약간의 혼선이 생겼던 것이 아쉬웠습니다.

개인회고: 임동진

나의 목표

우리는 Huggingface 에서 Pretrain 된 Parameter 를 가져오기 때문에 모델 변경은 쉽게 할 수 없다고 생각하였다. 하지만, Model 구조를 뜯어보니 Classifier Layer 는 미리 학습된 Parameter 가 아닌 것 같아 이 Layer 를 변형시켜 성능 올리기를 시도하였다. 또한 Scheduler 및 Optimizer 도 변경시켜 성능 향상을 시도해보았다.

무엇을 어떻게 했는가?

1 주차 때 팀원끼리 토의하여 프로젝트 진행방식에 대해 얘기했다. 첫 주차 때는 각자의 프로젝트를 진행하고, 2 주차 때부터 모두의 코드를 합쳐 Github 를 활용하기로 하였다. 또한 멘토님의 조언대로 칸반 보드 및 Notion 을 적극적으로 활용해보기로 하였다.

1 주차 때 Model 의 Classifier 를 무작위로 변경시켜 보았다. 이 때 에러가 많이 떴고 가끔 제대로 수행되더라도 CSV 파일을 만들어 제출하면 0 ~ 10 점을 벗어나지 못했었다. 또한 Focal Loss, F1 Loss 도 활용해보고 Scheduler 도 Cosineannealing 을 활용해보는 등 많은 시도를 해보았지만 계속해서 에러가 발생하거나 내가 원하는 방향으로 변경이 일어나지 않았다.

2 주차 때 팀원의 조언을 받고 내가 활용하는 클래스 및 메서드에 대한 코드를 읽기 시작했다. 그러자 내가 어떤 부분을 잘못하여 에러가 발생하였고, 가끔씩 에러가 발생하지 않았더라도 어떤 부분 때문에 0 ~ 10 점을 벗어나지 못했다는 것이 서서히 보이기 시작하였다. 코드를 모두 이해하고 나서, Classifier 대신 LSTM 을 활용해보는 방식을 활용해보았고 Classifier 이외의 BERT 모델을 Freezing 시킨 이후 Classifier 만 학습을 시키고, 1 Epoch 이 지나면 Freezing 을 풀어주어 모델 전체에 대해 학습을 진행하는 방식에 대해서도 적용해보았다.

3 주차 때는 논문 중 [CLS] 토큰 대신 [SUB], [OBJ] 토큰을 Object 및 Subject Entity 앞 뒤에 넣어주어, 해당 토큰에서 나온 Output 2 개를 Concat 시켜 Classifier 에 학습시키는 방식이 더 좋은 성적을 냈다는 것을 알았다. 따라서 이를 시도해보았고, Micro f1 이 75 정도로 단일 모델에서는 가장 좋은 성능을 냈었다.

아쉬운 점 및 다음 대회 때 시도할 점

코드 읽는 것에 대한 필요성을 너무 늦게 알았다. 코드를 처음부터 읽었다면 생기지 않았을 시간 낭비와 스트레스 때문에 자괴감도 많이 생기고 코딩도 손에 안 잡혔다. 처음부터 코드를 읽었다면 더욱 좋은 결과 및 많은 실험을 할 수 있었을 것 같아 아쉽다. 다음부터는 활용하는 메서드에 대한 코드를 숙지하고 활용해야 할 것 같다.

쉬는 것에 대한 중요성도 깨달은 것 같다. 계속해서 결과가 안 좋으니 선잠만 자면서 코드를 붙잡고 있었는데, 괜찮은 성능은 나오지 않았다. 계속해서 결과가 안 좋아 동기도 떨어져 한눈도 많이 판 것 같다. 그 때 주말에 한 번 푹 쉬고 나니 다시 아이디어가 하나 둘 씩 나오기 시작했고 한 눈도 덜 팔게 되어 코드에 집중할 수 있었던 것 같다. 다음부터는 만약 아이디어가 안 나오거나 코딩이 막히면 쉬면서 머리를 비워봐야 할 것 같다.

세 번째로 아쉬운 점은 협업 실패이다. 먼저 Github 의 활용이다. 각자의 방법으로 성능 향상에 힘쓴 것은 실력 향상에 매우 큰 도움을 주었다고 생각한다. 하지만 2 주차에 코드를 합칠 때 아이디어도 잘 공유되지 않았고 코드적으로도 모호해져 GitHub 가 오히려 코딩할 시간을 뺏게 되었다. 협업 툴을 잘못 활용하면 오히려 코딩에 방해가 된다는 것을 느꼈다. 두 번째는 칸반 보드 활용을 제대로 하지 못한 것이다. 첫째 주에 점수가 계속해서 나오지 않자 칸반 보드에 올리기 부끄럽다는 생각으로 칸반 보드 Update 에 미진했고, 이후 점수가 좋아져도 습관이 안 들어 있어 칸반 보드 활용을 깜빡해 결국 활용하지 못했었다. 다음부터는 점수 여부와 관계없이 칸반 보드를 활용하여 내 실험 결과를 공유해야겠다. 또한 1~2 일 째는 아무것도 하지 않고 코드를 모듈화시켜 Github 를 활용한 협업에 조금 더 집중해야겠다.

개인회고: 정재윤

- **이번 프로젝트에서 나의 목표는 무엇이었는가?**
  1. NLP 도메인에서 처음 진행하는 프로젝트인만큼 프로젝트의 전체흐름 익히기
  2. Hugging Face 에 익숙해지기
  3. 팀에서 맡은 일을 끝까지 책임지고 완수하기

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**
  1. 팀원들과의 적극적인 소통 및 활발한 정보 공유
  2. 협업을 위한 Github 익히기

- **나는 어떤 방식으로 팀에 기여했는가?**
  1. 대회에 적합한 pretrained model 탐색 및 다양한 hyperparameter 변형을 통한 실험결과를 토대로 최적의 hyperparamter 를 탐색
  2. 공부한 여러 augmentaion 기법을 활용하여 모델을 학습시킬 데이터셋 제작
  dataset 관련 역할을 맡아 진행하면서 eda 결과 공유

- **내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?**
  1. 다양한 augmentation 기법, sampling 을 적용한 데이터셋으로 실험을 진행하였지만 모델개선에는 큰 영향을 주지 못했다. Imbalance data 를 잡으려 노력했지만 sampling 자체가 이 문제의 정답은 아니라는 것을 느꼈다.
  2. 많은 양의 dataset 을 다루다보니 일을 처리하는데 생각보다 많은 비용이 요구됨을 알게 되었고, 처음부터 계획을 세우고 제작에 진입해야 일을 더 효율적으로 처리할 수 있음을 배웠다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**
  1. 팀원들간 협업툴인 wandb 를 활용하여 결과를 토대로 다같이 분석하여 이를 다음 실험에 적용. 불필요한 실험을 줄여 효율적으로 진행할 수 있었다.
  2. eda 를 지난 대회보다 좀더 심도있게 진행하여 이를 바탕으로 raw data 를 가공. Mislabel, 중복된 데이터를 제거할 수 있었고, augmentation 도 수월하게 적용할 수 있었다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**
  1. dataset 가공이 비용이 많이 들어 개인 local 에서 진행되었고, 팀 코드에서 이뤄지지 않아 github 에 업로드하지 못했고, 팀원들간의 코드리뷰 상황에서 떨어지는 코드이해도 때문에 협업에 온전히 참여하지 못했다.
  2. 여건상 좀 더 다양한 augmentation 을 적용해보지 못했고, NLP 도메인에서 augmentation 기법으로 성능향상을 맛보지 못해 아쉬웠다.

대회 후기

Hugging Face 에 어느정도 친해진 계기였습니다. 또한 NLP 에서 주로 쓰이는 augmentation 기법에 대해 깊게 배울 수 있어 뜻깊은 시간이었습니다. 하지만 더 많은 augmentation 기법을 적용해보지 못한점과 다른 캠퍼분들에 비해 EDA 분석을 충분히 하지 못하고 augmentation 을 적용했던 점이 아쉬웠습니다. 데이터를 다루는데 있어 EDA 는 매우 중요한 절차이자 초석임을 다시한번 알게 된 계기였습니다.

개인회고: 조설아

Input 문장 변형

- 원본 문장에 special token 추가
  - 효과가 없었다. 토큰을 추가하고서 토큰 등록을 하지 않았을 때 오히려 미미한 성능 향상이 있었기 때문에 entity marker 를 시도하는 것으로 실험 방향을 변경하였다.
- 원본 문장에 entity marker 추가
  - 정보를 추가하면 추가할수록 성능이 늘었고 typed entity marker punctuation 이 가장 효과적이었다. 그러나 특정 점수 이상을 넘기는 무리였다. 우연히 typed entity marker punctuation 을 subject 와 object 가 아닌 앞뒤 순서에 따라 다르게 적용하였을 때 꽤 큰 성능 향상이 있었음을 발견하고 한국어 문장 구조와 국문법 특성 상 subject 와 object 식별보다는 문맥의 흐름 파악이 더 중요함을 깨달았다.

Multi Sentence

- 위의 Input 문장 변형에 따라 concat entity 문장 또한 베이스라인의 "Subject[SEP]Object"에서 비슷하게 변주를 주면서 성능을 올렸다.
- Subject 와 Object 의 식별보다 문맥의 순서 파악이 더 중요함을 깨닫고 나서는 순서에 따라 서로 다른 기호로 entity 를 감싸주었기 때문에 concat entity 부분에서 subject 와 object 에 다른 기호로 감싸주는 것이 무의미해졌다. 따라서 단어들을 모두 동일하게 대괄호로 감싸주었다.
- 대회의 label 을 생각했을 때 label 은 obejct 를 중심으로 결정된다. 중요한 점은 subject 와의 관계 속에서 형성되는 성질이 label 이 된다는 점이였다.
- 따라서 object 를 주어로 삼고 subject entity, 그리고 object entity type 이 포함된 문장을 생성했다. 결론적으로 object 에 대한 정보는 최대한으로 subject 에 대한 정보는 최소한으로 주었다.

시도하여 실패한 것과 아쉬운 것

- Entity Embedding
  - 사용하고 있는 모델(RoBERTa-large)에 맞게 entity embedding layer 를 추가하려 했지만 실패했다.
  - 모델 구조에 대한 이해가 부족했다.
  - pytorch 와 huggingfaec 에 대한 공부가 부족했다.
  - 자주 사용하며 친숙해지고 싶다.

개인적인 대회의 목표와 결과

NLP 인 만큼 인문계열 학생인 나의 지식 혹은 논리가 효과가 있을지 확인해보고 싶었다. 다른 분야에 비하여 문이과 융합적 성격이 강할 것이라 예상했고 내가 배운 것이 쓸모가 있었으면 했다.

결과적으로 국문법적 접근이 효과가 있었다. 인공지능 기술과 인문학적 지식이 시너지 효과를 내는 성과를 달성했다는 데에 개인적 의의가 있는 대회였다.

개인회고: 허치영

이전 대회에서 제대로 된 실험 기록, 관리가 되지 않았던 만큼 이번 대회에서는 높은 성적보다 보다 많은 실험을 체계적으로 기록, 관리하는 것이 목표였습니다. Notion 칸반보드와 엑셀을 통해 진행한 실험을 최대한 기록하려고 노력했고, 실험 결과를 팀원들과 공유했습니다. 또한 huggingface 에 익숙해지고자 많은 시간을 huggingface 공식문서를 읽는 데에 사용했고 그 결과 기존 huggingface 모델을 customizing 하는 방법을 알 수 있었습니다.
Huggingface 및 모델링에 있어 지식이 부족한 탓에 실험해보려 했던 TAPT, RECENT 에 대해 제대로 진행해보지 못한 것이 아쉬웠습니다. 또한 언어학적, 국문학적 지식이 부족하여 언어적 측면에서의 접근이 많이 부족했습니다.
논문을 읽는 것에만 그치지 않고 꾸준히 모델을 직접 만들어봐야 된다는 다짐을 하게 되었습니다. 또한 언어에 관한 학문을 공부하는 만큼 언어학에 관한 지식도 틈틈이 쌓아가겠다는 생각을 하게 되었습니다.

협업의 면에서는 총 3 주간의 대회 중 첫 주부터 협업을 시작하지 않았던 것이 아쉬움으로 남았습니다. 개인별 실험과 학습으로 인해 파일간 충돌이 많이 발생했고, 그로 인해 불필요한 시간을 쓰게 되었습니다. 또한 notion 을 이용한 실험 공유에도 아쉬움이 남았습니다. 몇몇 실험을 제대로 기록하지 못하거나 깜빡하고 공유하지 못한 실험들이 있어서 같은 실험을 여러 명이 진행하는 경우가 발생했습니다.
아직 팀원들 모두 협업에 익숙하지 않은 만큼 더 많이 신경 쓰고, 서로에게 실험 기록에 대해서 상기시키는 방법밖에 없다고 생각합니다. 프로젝트를 계속해서 진행해 나가면 더 원활한 기록 및 관리가 될 수 있으리라 생각합니다.

개인회고: 이보림

시도했던 내용 *모든 성능 비교는 micro f1 score 기준

1. 데이터 중복 제거 (가정 : no relation으로 잘못 분류된 중복 데이터를 지우고 문장 내용은 같지만 label이 no relation이 아니면서 다른 데이터는 살리면 약간의 Imbalanced Data가 좋아질 것이다.)
   3 가지 경우 실험 -> 1) raw data 2) 모든 칼럼이 같은 데이터만 제거 3) 2)의 데이터에 문장내용, 엔티티 내용만 같은 데이터 중 no relation 만 제거 => 3) -> 1) -> 2) 순으로 성능이 좋았다.
2. 문장 내 엔티티 토큰 표시 (가정 : 문장 내에서도 엔티티에 관한 부분을 강조하면 모델도 해당 부분을 집중해서 이해할 것이다.)
   TYPE 은 모두 한글로 변환하여 실험 => ORG':'단체', 'PER':'사람', 'DAT':'날짜', 'LOC':'위치', 'POH':'기타', 'NOH':'수량'
   5 가지 경우 실험 -> 1) 표시 X 2) *entity* 3) *entity[TYPE]* 4) *[TYPE]* 5) *entity[special token] TYPE [/special token]* => 3) -> 5) -> 4) -> 2) -> 1) 순으로 성능이 좋았다.
3. 추가 문장 실험 (가정: 문장으로 해당 task에 대한 질문을 주면 pre-training된 모델이 문제를 더 이해할 수 있을 것이다.)
   4 가지 경우 실험 -> 1) SEP 토큰 이후 Object, Subject Entity 연결 2) *e01* 와 *e02 * 사이의 관계를 구하시오. 3) *e01 와 e02* 사이의 관계를 구하시오. 4) *엔티티[타입]* 과 *엔티티[타입]*의 관계를 구하시오. => 4) -> 2) -> 3) -> 1) 순으로 성능이 좋았다.
4. LDAM Loss (가정 : 적은 데이터의 label에 더 많은 regularization을 주는 Imbalanced Dataset을 위해 나온 LDAM Loss를 적용하면 성능이 향상할 것이다.)
   논문에서 Hyper parameter tunning 이 잘 되지 않으면 오히려 성능이 저하될 수 있다고 명시되어 있었다. 시간이 부족하여 제대로 된 튜닝을 못해서 기존의 best model 의 성능을 넘지 못하였어서 사용하지 못하였다.
5. Ensemble
   (가정 : 학습이 되지 않는 validation set 을 나누는 것에 부담이 있었다. label 을 고려하는 split 방법과 최대한 모든 데이터를 사용하고자 fold 별 모델을 저장하면 이러한 영향이 줄어들어 성능이 향상될 것이다.)
   Train, validation set 으로 데이터를 나누기에는 데이터가 불충분하며 데이터 불균형도 심했기 때문에 주어진 train dataset 의 class 별 비율을 고려하여 9:1 비율로 각기 다른 3 개의 train,validation set 으로 나누었다. 각각의 경우에 대해 모델을 저장한 후, voting 을 통해 최종 예측 결과를 만들었다.
   사용 후 많은 성능향상이 있었다. Hard voting 보다 Soft voting 사용 시 score 가 성능이 향상되었습니다.

후기

Data 를 직접 보는 것과 Task 에 대한 이해가 얼마나 중요한 것인지에 대해 깨닫게 되었다. Data 를 직접 읽어보고 제대로 이해하기 전과 후의 접근이 완전히 달라졌으며, 이해하고 나니까 더 다양한 아이디어가 떠오르고 그에 따른 의미있는 성능향상도 만들 수 있었다. 실험을 하면서 생각해본 아이디어가 성능 향상을 야기했을 때, 성능이 오른 이유와 모델은 아이디어를 어떻게 이해할까에 초점을 두고 생각해봤다. 이러한 생각을 하려고 하니 지금 사용하는 모델이 어떤 데이터를 어떻게 학습하였는 지로 다시 되돌아가면서 재미있게 공부할 수 있었다. 하지만 이번 대회를 통해, 직접 실험을 통한 성능확인과 실험기록 정리의 중요성을 느끼게 되었다. 마지막으로 모델 이해의 중요성을 느끼고 이러한 부분이 약했어서 다시 강의를 찾아보는 등 복습에 있어서 제대로 동기부여가 되었다.