Conference 자료

BERT는 NLP 분야에서 현재 매우 보편적이고 강력한 성능을 내는 딥러닝 모델이다. 이 딥러닝 모델은 transformer라는 기존 seq2seq와 다른 형태를 지니며 분류에서부터 번역까지 아우르는 분야에서 활용이 가능하다.

이 모델은 구글에 의해 공개되었고 huggingface라는 다른 팀에서 해당 모델을 친히 구현해 주었다. 이 모델에 들어가는 Input data를 위한 인코딩 작업 역시도 해당 팀에서 배포하였다. 이번 인공지능 대회에서 학습할 수 있는 단어 개수가 매우 적어 이를 어떻게 단어 인코딩을 해야 할지 고민이었고 많은 자료를 찾아본 결과 Sentence Piece 방식으로 인코딩 하는 것이 현재 상황에 적합하다 판단하였다. 이 방식은 어떤 학습할 단어 셋을 주면 해당 단어 세트에서 가장 작은 단위로 문자를 쪼갠 뒤 단위로 조립하여 나오는 단어들을 vocab 파일로 변환해주는 방식이다. ->나중에 더 자세한 설명 추가할 예정.

대회에서 제공한 학습 문장 세트를 넣어주어서 해당 문장들을 모두 스캔하면서 최대한 많은 조합의 단어를 학습하고 그 결과를 vocab 파일로 저장해 두었다. 이는 추후에 Bert 모델에 넣을 때 사용하기 위한 단어로서 BERT 모델은 결국 단어 단위로 임베딩을 실시하기 때문에 최대한 좋은 형태의 vocab 파일을 얻는 것이 곧 좋은 성능을 내는 지름길이었다.

그러나 BERT Tokenizer는 어디까지나 영어에 맞추어서 나온 알고리즘이다. 한국어의 경우 NLP는 매우 힘든데 영어처럼 단순히 띄어쓰기로 의미가 분절되는 것이 아니라 다양한 형태소의 조합으로 의미가 결정되고 분절되기 때문이다. 따라서 좀 더 좋은 성능을 내기 위해서 이 버트 토크나이저와 기존 한국어 NLP와 결합하여 쓸 수 있는 방법을 찾았다.

<https://github.com/lovit/huggingface_konlpy> 깃허브에서 좋은 자료가 있어 참고하였다. 이 분은 한국어 형태소 분석기 중 하나인 Konlpy와 버트 토크나이저를 융합하여 좀 더 좋은 워드 분절을 위한 코드를 짜셨다. 한국어 형태소 분석기 Tokenizer를 사용해 토크나이징을 결정하는 내부를 한국어에 최적화 된 형태로 인식하게 하여 Bert Tokenizer의 바탕이 되는 Word Piece Tokenizer와 결합해 좋은 성능을 보여주었다고 한다. 실제로 Konlpy Word Piece Tokenizer를 사용하니 [Unk] 마킹이 된 단어가 약간 줄었고 조사와 용언 어간 및 어미의 구분이 기존 Word Piece Tokenizer보다 좋아졌다. -> 추후에 사진 자료 첨부할 예정.

이를 이용하여 버트 모델에 단어를 임베딩하여 넣었고 파인 튜닝을 진행하였으나 NLP 딥러닝이 처음이라 오버피팅을 해결하지 못해 결국 매우 쓰레기 같은 결과를 냈다. 결과적으로 다른 두 분이서 진행한 scikit learn 기반의 텍스트 분류가 훨씬 더 좋은 결과를 내었고 그것을 최종 결과로 제출하였다.

이번 컨퍼런스 주제인 영어 소설작가 분류는 다행이도 위와 같은 더 복잡한 과정이 필요가 없었다. 영어 기반으로 만들어진 BERT 모델과 토크나이저를 그대로 쓰면 됐기 때문이다. 대회에서 pre-trained 모델을 사용하지 말라고 하였긴 하지만 어디까지나 지금은 컨퍼런스를 위한 작업이기 때문에 그냥 했다.

데이터에는 Null 데이터는 없는 것으로 파악됐다. 또한 단어의 길이를 봤을 때 대부분 500 자 이하에 분포하였고 대략적으로 10% 정도 그보다 더 길었다. 마찬가지로 학습 속도와 메모리 활용 개선을 위해서 해당 긴 텍스트를 어떻게 처리할지 좀 더 생각해 봐야했다. 두 가지 안이 있는데 해당 텍스트를 임의로 잘라서 작가 레이블과 매칭을 시키는 방법이다. 이는 매우 효과적일 수 있으나 테스트 셋에서 500자보다 긴 문장이 존재한다면 이에 대한 예측에 변수가 생길 수 있다는 점을 고려해야 했다. -> 어떻게 할지 결정하고 뒤에 덧붙이기

마찬가지로 BERT 모델이 과연 영어에서 Fine-tuning 만으로 얼만큼의 효과를 거두는지 확인하고 싶었다. 따라서 나는 또 BERT를 고집했다. 우선 앞서 얘기한 대로 Word Piece Tokenizing인 BERT Tokenizer를 활용했다. 따로 이상한 전처리나 그런걸 해줄 필요 없이 그냥 토크나이징을 하기만 하면 모델에 넣을 준비가 되기 때문에 매우 효율적인 방법이다.

files=training\_file,

vocab\_size=vocab\_size, # 학습할 단어 크기

min\_frequency=min\_frequency, # 단위어 최소 출현 빈도

limit\_alphabet=limit\_alphabet, # 알파벳 학습의 최대 개수

special\_tokens=["[PAD]", "[UNK]", "[CLS]", "[SEP]", "[MASK]"],

show\_progress=True,

wordpieces\_prefix="##"

이는 토크나이저의 input parameter이다. 단어를 학습시킬 파일을 제공하고 단어 사전의 크기를 정하고 최소 몇 번 출현하는 단어에 대해서만 학습시킬 것인지 정하고 다양한 버트를 위한 토큰을 집어 넣었다. 이렇게 학습한 모델은 그 결과를 vocab 파일로 저장하고 후에 load를 한 뒤 인코딩을 해보았다. 우리는 self train을 위해 위키피디아의 다양한 코퍼스와 문장을 가져온 뒤 학습을 시킬 예정이다.

BERT Tokenizer는 또한 이렇게 직접 학습하는 방법 말고도 미리 학습된 단어를 가져올 수 있다. 한국어 NLP BERT Tokenizer에서는 사용할 수 없는 방법이었지만 영어기 때문에 그냥 가져오면 어느 정도 구글에서 잘 만들어둔 vocab을 사용할 수 있었다. 이 둘의 결과를 한 번 비교해 봐서 어떤 방법이 좀 더 효과적인지 판단하고서 해당 방법으로 토크나이징을 진행하기로 하였다.