BERT 기반 Text Classification

- 소설 작가 분류를 중심으로 -

2023학년도 1학기 인공지능 응용 (ICE4104)
팀 프로젝트 보고서

2023. 06. 10.

학번	12181879	**이름**	이동건
학번	12201921	**이름**	이가은
학번	12211722	**이름**	허지원

Introduction

본 보고서는 작문 스타일을 기반으로 소설 작가를 예측하고 분류하는 작업을 수행하기 위해 Bidirectional Encoder Representations from Transformers (BERT)[1]를 사용하여 텍스트 분류 모델을 개발하는 데 중점을 둔 프로젝트의 결과를 제시한다. 이번 프로젝트는 AI 경진대회 플랫폼인 데이콘(DACON)[a]이 주최한 경진대회[b]의 일환으로 진행하였다.

프로젝트의 주요 목표는 BERT의 기능을 활용하여 영문 소설 텍스트 데이터의 복잡한 뉘앙스를 캡처하고 각 소설 작가가 보여주는 특징적인 문체를 학습하는 것이다. 다양한 소설에서 추출한 개별 문장으로 구성된 데이터셋에서 모델을 훈련함으로써, 주어진 텍스트 데이터의 저자를 식별할 수 있는 정확하고 강력한 분류기를 구축하는 것을 목표로 한다. 이를 바탕으로 본 보고서는 데이터 전처리, 모델 구조, fine tuning 전략 및 평가 metrics를 포함하여 방법론에 대한 포괄적인 개요를 제공한다.

Method

1. Data Analysis

1.1 Dataset Information

index	text	Author
0	He was almost choking. There was so much, so much he wanted to say, but strange exclamations were all that came from his lips. The Pole gazed fixedly at him, at the bundle of notes in his hand; lo...	3
1	"Your sister asked for it, I suppose?"	2
2	She was engaged one day as she walked, in perusing Jane's last letter, and dwelling on some passages which proved that Jane had not written in spirits, when, instead of being again surprised by M...	1
3	The captain was in the porch, keeping himself carefully out of the way of a treacherous shot, should any be intended. He turned and spoke to us, "Doctor's watch on the lookout. Dr. odin take the n...	4
4	"Have mercy, gentlemen!" odin flung up his hands. "Don't write that, anyway; have some shame. Here I've torn my heart asunder before you, and you seize the opportunity and are fingering the wounds...	3

(그림 1) 학습용 데이터셋 예시

(그림 1)은 주최측에서 학습용 데이터셋으로 제공한 데이터 중 일부이다. 텍스트 형식의 영문 소설 데이터가 'text' 열에 주어지고, 'Author' 열에는 0부터 4까지의 5명의 서로 다른 작가에 대한 Author code가 주어진다. 각 작가별 데이터의 분포 수는 <표 1>에 나타내었다.

[a] https://dacon.io/

[b] 월간 데이콘 소설 작가 분류 AI 경진대회, https://dacon.io/competitions/open/235670/overview/.

<표 1> 학습용 데이터셋의 클래스 분포

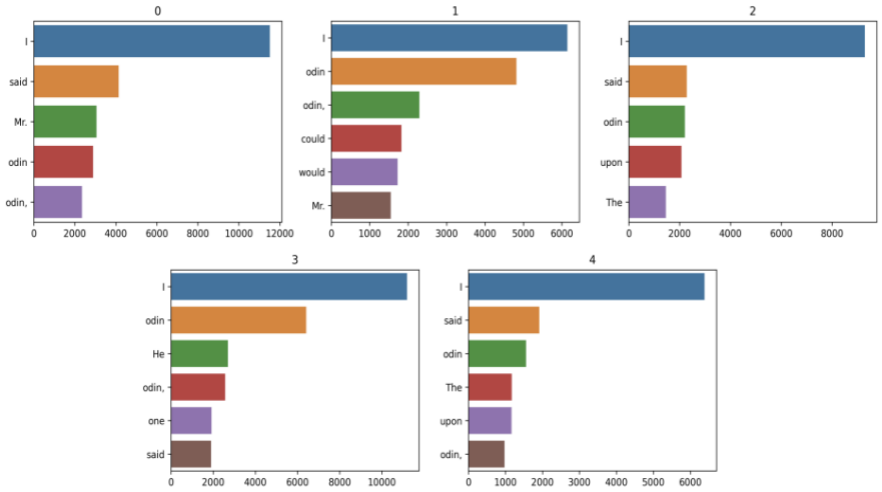
Label	Text data
0	13,235
1	7,222
2	11,554
3	15,063
4	7,805
Total	54,875

1.2 Data Feature 시각화

데이터 전처리 및 모델 학습에 앞서, 학습용 데이터셋의 특성을 탐색하고자 여러 방법론[2]을 적용하여 다음과 같이 데이터 분석을 진행하였다.

1. 작가 별 단어 빈도 수 분석

① 불용어(stopword)를 처리하지 않은 데이터의 작가 별 단어 빈도 수



(그림 2) 작가 별 단어 출현 빈도 수

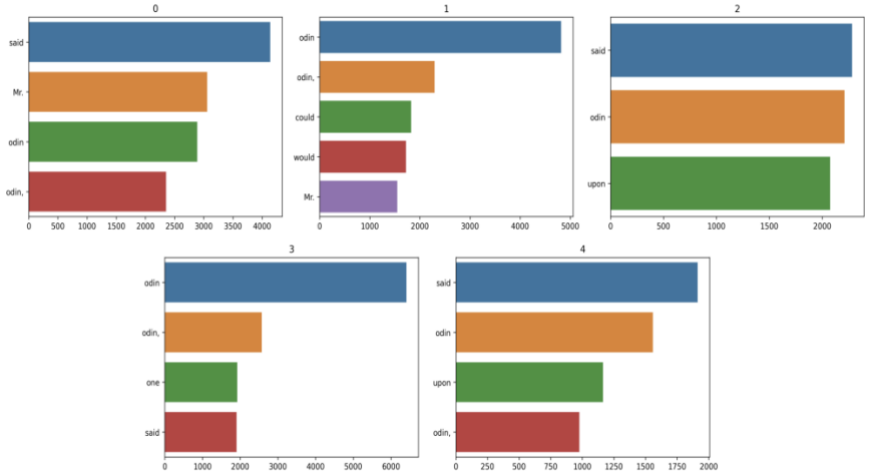
학습용 데이터셋의 소설 텍스트에서 등장 빈도수가 높은 상위 5개의 단어들을 작가 별로 시각화하여 (그림 2)에 나타내었다. “I” 라는 단어가 공통적으로 출현 빈도가 가장 높았으며, “Odin” 이라는 단어 역시 공통적으로 등장하였다. “Odin” 은 북유럽 신화에 나오는 신의 이름인데, 소설 속 등장인물의 이름을 익명화 한 것으로 보인다.[2] 이외에도 “the” 같은 핵심적인 의미를 가지지 않는 단어들도 상위 5개의 단어에 속하는 것을 확인할 수 있다.

② 불용어(stopword)를 삭제 처리 후 작가 별 단어 빈도 수

소설과 같은 글을 쓰는 상황에서 우리가 필수적으로 쓰는- 또는 쓸 수밖에 없는- 어휘들[c], 혹은 비교적 문맥상 중요하지 않은 단어들을 불용어라고 한다. 학습용 데이터셋에서 이러한 불용어를 제거한 후[2][3], 작가 별 단어 빈도 수를

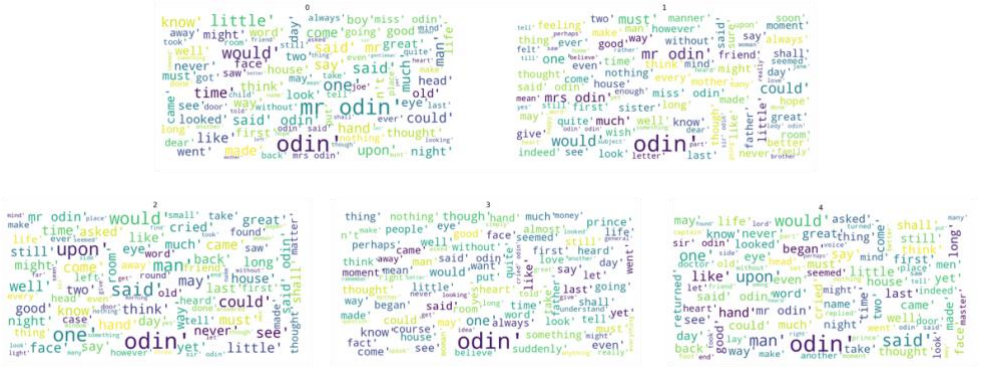
[c] I, you, also, the 등

확인하여 (그림 3)에 나타내었다. 전체적으로 “said”, “Mr.”, “Odin” 등의 빈도 수가 상대적으로 높은 빈도 수를 가지는 것을 볼 수 있다.



(그림 3) 불용어(stopword)를 삭제 처리 후 작가 별 단어 빈도 수

2. WordCloud[4] 분석



(그림 4) 소설 데이터의 작가 별 WordCloud (왼쪽 위부터 순서대로 0, 1, 2, 3, 4, 5)

WordCloud[4]는 텍스트 데이터에서 주요 단어를 시각적으로 나타내는 방법이다. 주어진 텍스트에서 가장 빈도가 높은 단어일수록 더 크게 표시되며, 텍스트 내에서 상대적인 중요도를 시각화 하는 데 사용된다. (그림 4)는 불용어를 삭제한 소설 데이터를 이용하여 WordCloud 분석한 결과이다.

2. Data Preprocessing

2.1 BERT 모델 사용을 위한 data 전처리

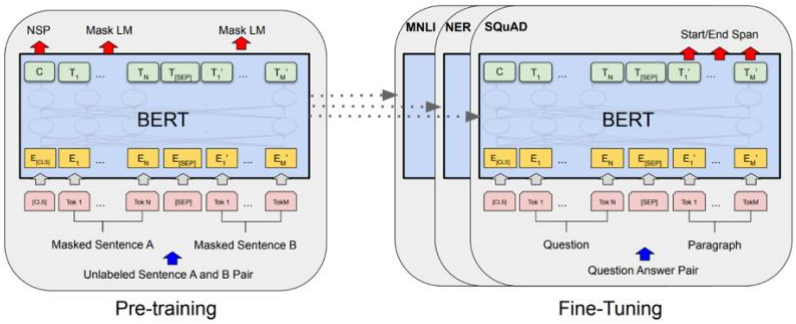
BERT를 사용하기 위해서는 텍스트 데이터를 BERT 모델에 맞게 전처리 해주는 과정이 필요하다. Transformers.BertTokenizer[5] 모듈을 사용하여, 다음과 같은 전처리 과정을 수행하였다.

- 인스턴스화한 tokenizer를 사용하여, 주어진 텍스트를 토큰화
- 입력 데이터의 최대 길이를 512로 제한, 입력 데이터가 최대 길이보다 작은 경우 padding을 수행
- 토큰화 된 입력 데이터를 정수 인코딩

2.2 Train / Validation data 선정

학습용 데이터셋 54,879건의 10%인 549건을 검증 데이터(Validation Data)로 설정하고, 나머지 54,330건은 훈련 데이터(Train Data)로 사용하였다.

3. Model



(그림 5) BERT의 전반적인 pre-training과 fine-tuning 과정[1]

BERT[1]는 2018년 구글에서 개발한 자연어 처리 모델이다. BERT는 transformer[6]의 인코더 구조를 활용하여 개발된 모델로 Pre-training 단계와 Fine-tuning 단계로 이루어진다.

Pre-training 단계에서는 Next Sentence Prediction (NSP)와 Masked Language Model (MLM)이라는 두 학습 기법이 동시에 이루어지며 학습을 수행한다. NSP 기법은 두 문장이 연결되어 있던 문장인지 아닌지를 예측하는 방식이다. 이때, 50%는 연결된 문장으로, 50%는 랜덤하게 뽑은 문장으로 학습된다. MLM 기법은 텍스트의 양방향 문맥을 학습하기 위해 input 토큰의 15%를 랜덤하게 masking 처리 후, masking 한 부분을 예측하는 방식이다. 이때, masking 비율의 80%는 [MASK] 토큰으로, 10%는 랜덤한 단어로, 나머지 10%는 일부러 변경하지 않고 예측한다.

Fine-tuning 단계에서는 학습된 모델을 세부적인 작업에 사용하기 위해 학습된 모델 뒤에 세부적인 작업을 위한 레이어를 하나 추가한 뒤, 원하는 작업에 맞는 데이터를 학습시키는 과정이다. 이때, 이전에 학습했던 데이터로 웨이트를 미세 조정하기 때문에, 더 좋은 성능을 낼 수 있다.

본 프로젝트에서는 다양한 BERT 계열의 모델들 중 BERT-Base-Cased 모델을 Fine-tuning 하여 적은 train data와 적은 epoch만으로 학습이 가능하도록 하였다. 하이퍼 파라미터 튜닝 시, 공통적으로 AdamW 옵티마이저를 사용하였으며, 학습 시, GPU는 NVIDIA RTX A6000을 사용했다.

4. Evaluation Metric

$$L(y,p) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log p_i + (1-y_i)\log(1-p_i)] \quad (1)$$

모델을 평가하기 위한 지표로, 모델이 예측한 확률 분포와 실제 레이블의 확률 분포 간의 차이를 측정하는 Log Loss를 이용하였다. 식 (1)에서 N은 샘플 수, y_i는 i번째 샘플의 실제 레이블, p_i는 i번째 샘플에 대한 모델의 예측 확률을 의미한다. Log Loss 값이 작을수록 모델의 성능이 좋다고 판단한다.

Result

1. Baseline Model

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 16)           320000
_________________________________________________________________
global_average_pooling1d     (None, 16)                0
_________________________________________________________________
ReLU (Dense)                 (None, 24)                408
_________________________________________________________________
Softmax (Dense)              (None, 5)                 125
=================================================================
Total params: 320,533
Trainable params: 320,533
Non-trainable params: 0
```

(그림 6) 주최 측에서 제공한 기본 베이스라인 모델의 구조[3]

대회를 주최한 DACON은 기본 베이스라인 모델을 제공하였다.[3] 해당 모델은 임베딩 레이어를 통과시킨 후, 데이터를 1차원 벡터로 만들어 주기 위해 GAP를 통과시킨 후, ReLU와 Softmax를 거쳐 작가를 예측하는 간단한 모델이다. 베이스라인 모델은 0.58012의 Log Loss를 달성하였으며, 해당 Log Loss보다 작은 값이 나오도록 하는 것을 목표로 실험을 진행하였다.

2. Model Performance (Hyper parameter tuning)

2.1. 고정된 학습률(Learning Rate) 사용

<표 2> 고정된 학습률(Learning Rate; LR) 사용에 따른 모델의 성능 비교

Learning Rate (LR)		Epoch	Batch size	Test Log Loss
Constant LR	5e-6	3	16	0.7087
	5e-6	7	16	0.6752
	5e-5	2	32	0.6815
	5e-5	3	16	0.7542
	5e-5	10	32	0.7088
	1e-5	3	16	0.5893

<표 2>에서 학습률을 고정된 값으로 설정하여 학습했을 때의 최고 성능은 0.5893임을 알 수 있다. 이것은 베이스라인 모델의 성능인 0.58012와 비교했을 때, 낮은 성능이다. 따라서, epoch마다 학습률을 변화시키는 학습률 스케쥴러(Learning Rate Scheduler)를 사용했다.

2.2. 학습률 스케쥴러(Learning Rate Scheduler) 적용

<표 3> 학습률 스케쥴러 적용에 따른 모델의 성능 비교

Learning Rate (LR)		Epoch	Batch size	Test Log Loss
Cosine annealing		3	16	0.8017
Exponential LR		3	16	7.5200
Lambda LR		3	16	0.7450
Step LR (Init LR / gamma)	5e-5 / 0.1	3	16	0.5092
	5e-5 / 0.1	6	16	**0.4509**
	5e-5 / 0.2	3	16	0.5262
	1e-5 / 0.1	3	16	0.6908
	1e-4 / 0.1	3	16	0.5040
	1e-4 / 0.5	3	16	0.5855

학습률 스케쥴러로 Cosine Annealing, Exponential LR, Lambda LR, Step LR을 사용하였다. 이 중, 성능이 가장 뛰어난 방법은 Step LR이었으며 epoch = 6, Batch size = 16에서 베이스라인 모델의 성능을 뛰어넘는 가장 좋은 결과를 얻을 수 있었다. 최고 성능을 보인 모델의 파라미터 수는 108,314,117이다.

2.3. 불용어(stopword) 제거 시 성능

<표 4> 불용어를 제외한 데이터를 사용했을 때의 Log Loss

Learning Rate (LR)		Epoch	Batch size	Test Log Loss
Constant LR	5e-5	3	16	1.03
	1e-5	3	16	0.88

핵심 의미를 가지지 않는 불용어들을 없애는 작업을 전처리 단계에서 수행하고, 결과를 비교해보았다. <표 4>에서 불용어를 삭제했을 때, Log Loss 값이 올라가면서 성능이 낮아지는 것을 확인하였다. 이를 통해, 소설 데이터에서 불용어를 삭제할 경우, 작가의 문체를 판단하기 어려워진다고 추측하였다.

Conclusion

1. 결론

본 보고서에서 진행한 실험에서는 사전 학습된 BERT-Base-Cased 모델을 데이콘에서 제공한 학습용 데이터셋 54,875건을 이용하여 fine-tuning하였고, 이를 통해 작가를 예측하는 모델을 개발했다. 성능을 높이기 위해, 하이퍼 파라미터 튜닝을 시도하였고, StepLR 스케쥴러를 사용했을 때 0.45089의 Test Log Loss를 달성함으로써 베이스라인 모델에 비해 성능이 향상된 모델을 얻을 수 있었다.

2. 성능 향상 이유

1) Step LR Scheduler의 사용으로 최적화된 weight를 얻을 수 있었다.
2) Foundation Model을 사용하여 최적화된 weight를 얻을 수 있었다.
3) BERT는 BookCorpus라는 도서 데이터로 학습되었기 때문에, 소설 데이터의 특성을 잘 학습할 수 있었다.
4) 불용어의 삭제가 성능 저하의 원인이 됨을 확인하고 불용어를 data에 포함시켰다.
5) 베이스라인 모델에 비해 약 337배 정도 차이나는 BERT의 파라미터 수가 성능 향상에 영향을 주었을 것이라 추측했다.

References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT* (pp. 4171-4186).

[2] 자연어 처리를 위한 탐색적 데이터 분석 - DACON. (2020). https://dacon.io/competitions/official/235670/codeshare/1822.

[3] [코드] 데이콘 기초 베이스라인 - DACON. (2020). https://dacon.io/competitions/official/235670/codeshare/1738.

[4] Andreas Mueller, Jean-Christophe Fillion-Robin, Raphael Boidol, et al. (2023). amueller/word_cloud: 1.9.1.1 (1.9.1.1). Zenodo. https://doi.org/10.5281/zenodo.7901644.

[5] BERT - Hugging Face. (2023). https://huggingface.co/docs/transformers/v4.26.0/model_doc/bert - transformers.BertTokenizer.

[6] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[7] 다양한 Learning Rate Scheduler(pytorch) - DACON. (2020). https://dacon.io/codeshare/2373.