

첫 번째 프로젝트. **Yelp 식당 리뷰 데이터**를 활용한

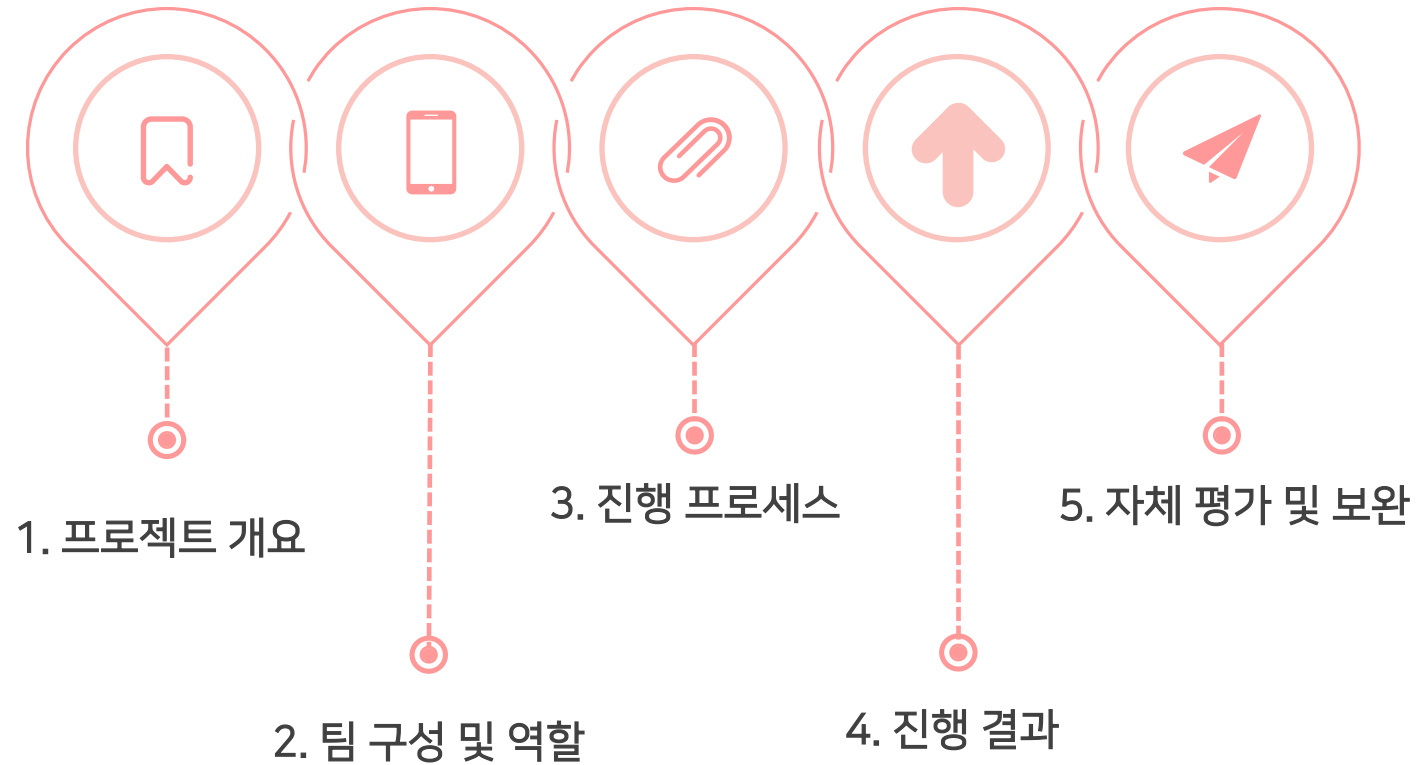
문장 분류기 성능 개선

Team 2. GNLP

(Good Night after Last Project)

김도연, 양은지, 오희주, 이다원, 이지수, 전효택, 정명찬

CONTENTS



1. 프로젝트 개요

프로젝트 주제

Yelp 식당 리뷰 데이터 를 활용한 문장 분류기 성능 개선

개발 환경

Python(Colab, VScode)

Torch, Transformers, Wandb, Matplotlib

진행 프로세스

데이터 전처리 > 모델 선정 > Hyperparameter 최적화 >
앙상블 적용 성능 개선 > 최종 평가, 모델 활용

목표

성능 개선 : 98.1% (baseline code) > 99%

2. 팀 구성 및 역할

※ 팀원 이름은 가나다순

★ 팀장



김도연

- 진행 사항 총 정리
- 데이터 정제 및 정규화
- 서비스 시스템 설계, 서버 구축



이지수

- 데이터 정제 및 정규화
- 데이터 증강 & 전처리 성과 검증

오희주

- 모델 선택 및 설계
- 모델 학습, 성능 평가



이다원

- 모델 선택 및 설계
- 모델 학습, 성능 평가

정명찬

- 모델 선택 및 설계
- 모델 학습, 성능 평가

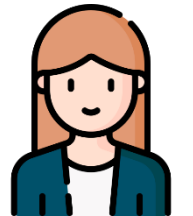


전효택

- 하이퍼파라미터 최적화
- 모델 종합, 비교

양은지

- 하이퍼파라미터 최적화
- 모델 종합, 비교



3. 진행 프로세스

프로젝트 총 개발기간 : 12/26(월) ~ 30(금) / 총 5일

The diagram illustrates a 6-step process for a machine learning project, with each step represented by a numbered hexagon and a dark blue rounded rectangle containing the step name and its tasks. The steps are connected by dashed lines. To the right of the steps is a 6x6 grid representing dependencies, where red squares indicate that a task in a given step depends on a task in a previous step.

Step	Task	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
1	사전 기획	Yes	No	No	No	No	No
2	데이터 전처리	No	Yes	No	No	No	No
3	모델 선정, 학습	No	No	Yes	No	No	No
4	최적화(Wandb)	No	No	Yes	Yes	No	No
5	모델 활용	No	No	No	Yes	Yes	No
6	프로젝트 발표	No	No	No	No	No	Yes

4. 진행 결과

데이터 전처리

1) Data Augmentation

neg	pos	total	neg_percent	pos_percent
177218	266041	443259	40.0	60.0

- 데이터 클래스 불균형
- Contextual Word Embeddings Augmenter 활용

예시 : Original: The quick brown fox jumps over the lazy dog .

Augmented Text:

- 1) the old quick brown fox instinctively jumps over for the lazy dog.
 - 2) the quick young brown fox dog jumps over the remaining lazy dog.
- Neg * 3 = 531654개
 - Pos * 2 = 532082개

2) TEXT 전처리 및 토큰화

- "_num_" 은 공백 대체
- 영어 외 특수문자는 공백 대체

4. 진행 결과

데이터 전처리

1) Data Augmentation

neg	pos	total	neg_percent	pos_percent
177218	266041	443259	40.0	60.0

- 데이터 클래스 불균형
- Contextual Word Embeddings Augmenter 활용

예시 : Original: The quick brown fox jumps over the lazy dog .

Augmented Text:

- 1) the old quick brown fox instinctively jumps over for the lazy dog.
 - 2) the quick young brown fox dog jumps over the remaining lazy dog.
- Neg * 3 = 531654개
 - Pos * 2 = 532082개

학습 시간이 너무 길어져
활용 어려움

부분적 적용(불균형 해소)

- Neg : 262476
- Pos : 266041

4. 진행 결과

데이터 전처리

1) Data Augmentation

- Neg : 262476
- Pos : 266041

2) TEXT 전처리 및 토큰화

- "_num_" 은 공백 대체
- 영어 외 특수문자는 공백 대체



Overfitting 발생

성능 개선 효과 없음



기존의 데이터셋 활용

4. 진행 결과

모델 선정, 학습

구조가 다른 다양한 모델을 통해 정확도를 확인

'bert-base-uncased'

'VictorSanh/Roberta-base-finetuned-yelp-polarity'

'google/electra-base-discriminator'

4. 진행 결과

모델 선정, 학습

구조가 다른 다양한 모델을 통해 정확도를 확인

'bert-base-uncased'

'VictorSanh/Roberta-base-finetuned-yelp-polarity'

'google/electra-base-discriminator'



각 모델 테스트 결과 98.2% ~ 98.9% 성능 확인

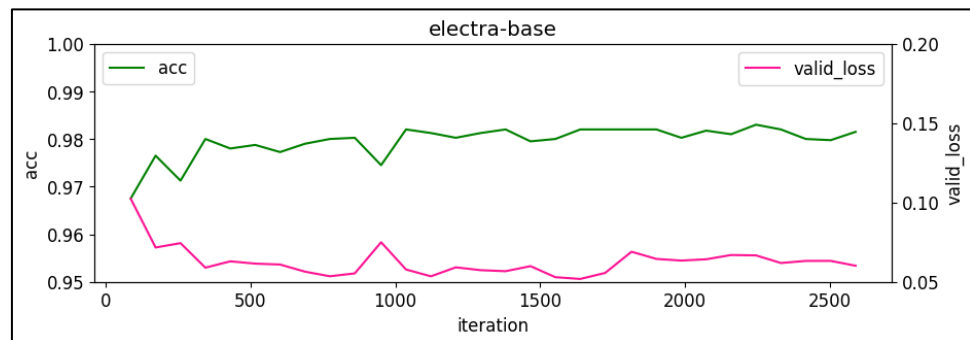
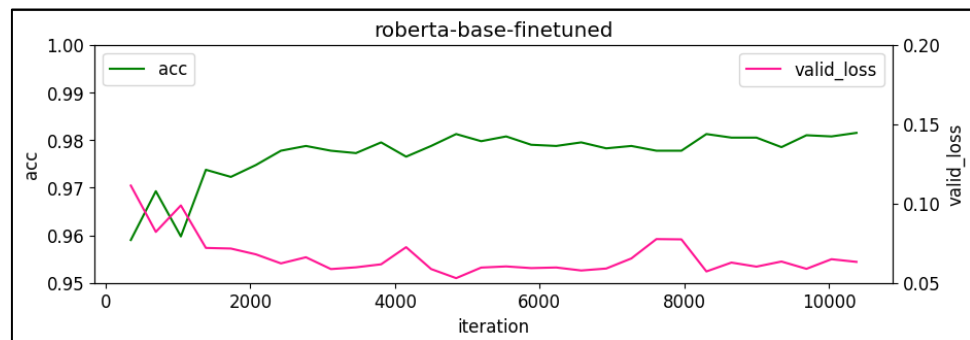
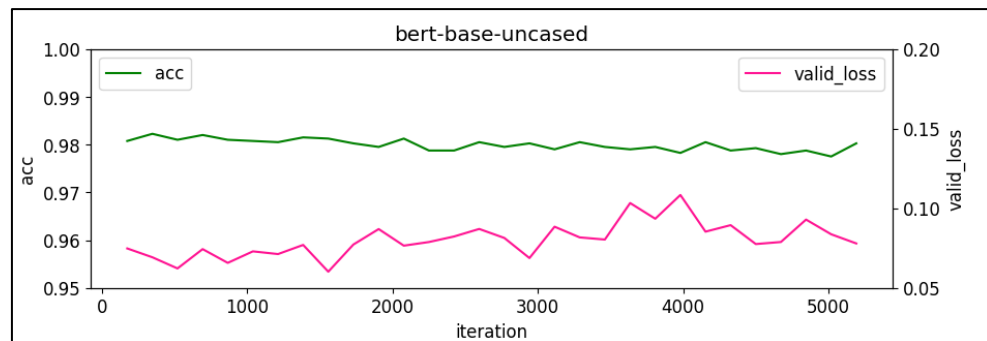
4. 진행 결과

모델 선정, 학습

Iteration에 따른 accuracy와 valid_loss 값 확인

```
#####  
# token_type_ids=token_type_ids,  
# position_ids=position_ids,  
# labels=labels)  
output = model(input_ids=input_ids,  
                attention_mask=attention_mask,  
                token_type_ids=token_type_ids,  
                position_ids=position_ids,  
                labels=labels)  
  
logits = output.logits  
loss = output.loss  
valid_losses.append(loss.item())  
  
batch_predictions = [0 if example[0] > example[1] else 1 for example in logits]  
batch_labels = [int(example) for example in labels]  
  
predictions += batch_predictions  
target_labels += batch_labels  
  
acc = compute_acc(predictions, target_labels)  
valid_loss = sum(valid_losses) / len(valid_losses)  
  
### matplotlib start ###  
  
iter_x.append(iteration)  
acc_y.append(acc)  
loss_y.append(valid_loss)  
  
### matplotlib end ###  
  
if lowest_valid_loss > valid_loss:  
    print('Acc for model which have lower valid loss: ', acc)  
    torch.save(model.state_dict(), "./pytorch_model.bin")  
    lowest_valid_loss = valid_loss
```

4. 진행 결과



```
cnt = 0
for i in range(len(iter_x)):
    if cnt >= iter_x[i]:
        iter_x[i] = max_iteration + iter_x[i]
    cnt = iter_x[i]

cnt = 0
for i in range(len(iter_x)):
    if cnt >= iter_x[i]:
        iter_x[i] = max_iteration + iter_x[i]
    cnt = iter_x[i]
print(iter_x)

plt.style.use('default')
plt.rcParams['figure.figsize'] = (10, 3)
plt.rcParams['font.size'] = 12

x = iter_x
y1 = acc_y
y2 = loss_y

fig, ax1 = plt.subplots()
ax1.set_xlabel('iteration')
ax1.set_ylabel('acc')
ax1.plot(x, y1, color='green', label='acc')
ax1.legend(loc='upper left')
ax1.set_ylim([0.950, 1])

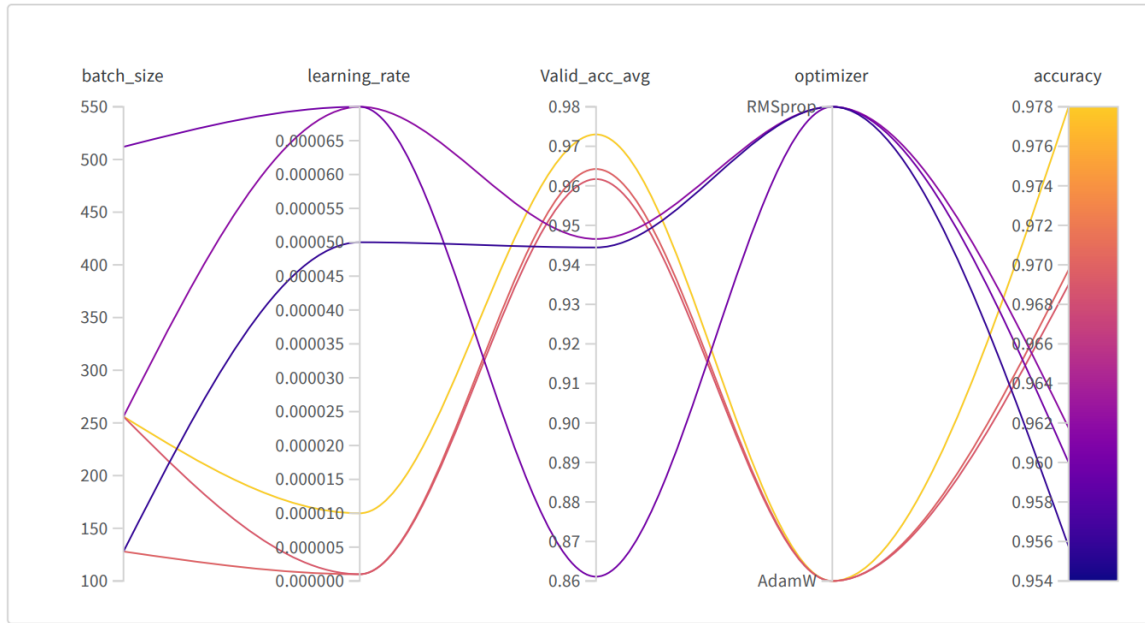
ax2 = ax1.twinx()
ax2.set_ylabel('valid_loss')
ax2.plot(x, y2, color='deeppink', label='valid_loss')
ax2.legend(loc='upper right')
ax2.set_ylim([0.05, 0.2])

plt.title("roberta-yelp-finetuned")

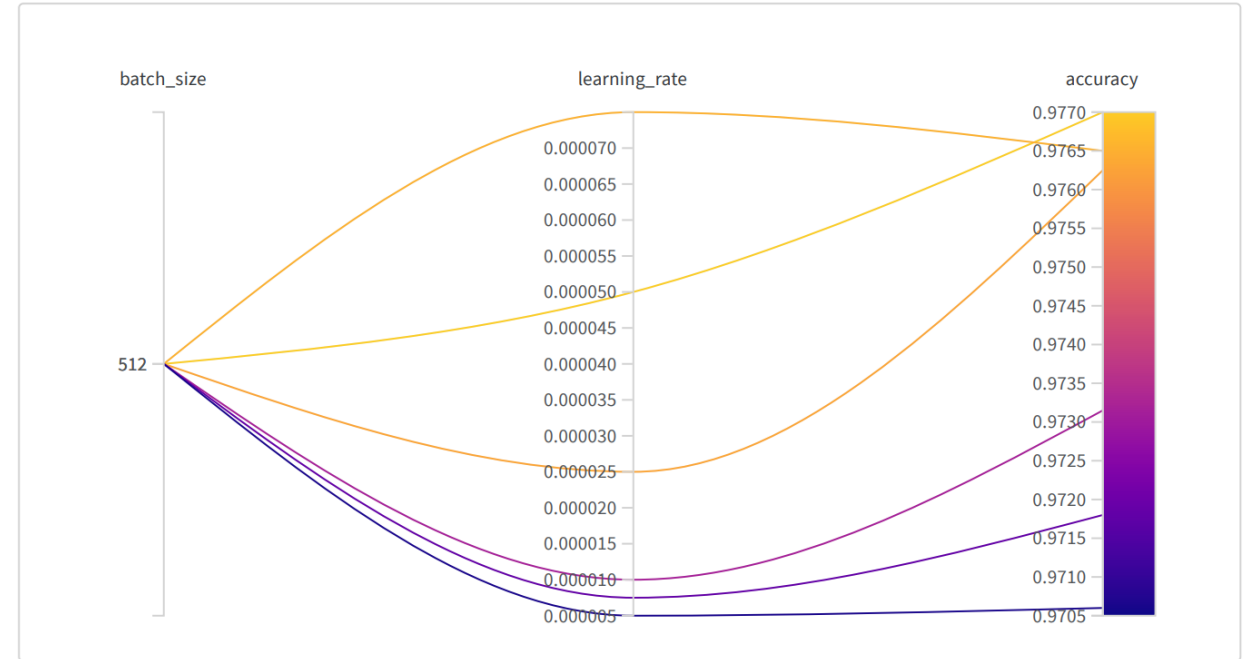
plt.show()
```

4. 진행 결과

하이퍼파라미터 최적화(Wandb)



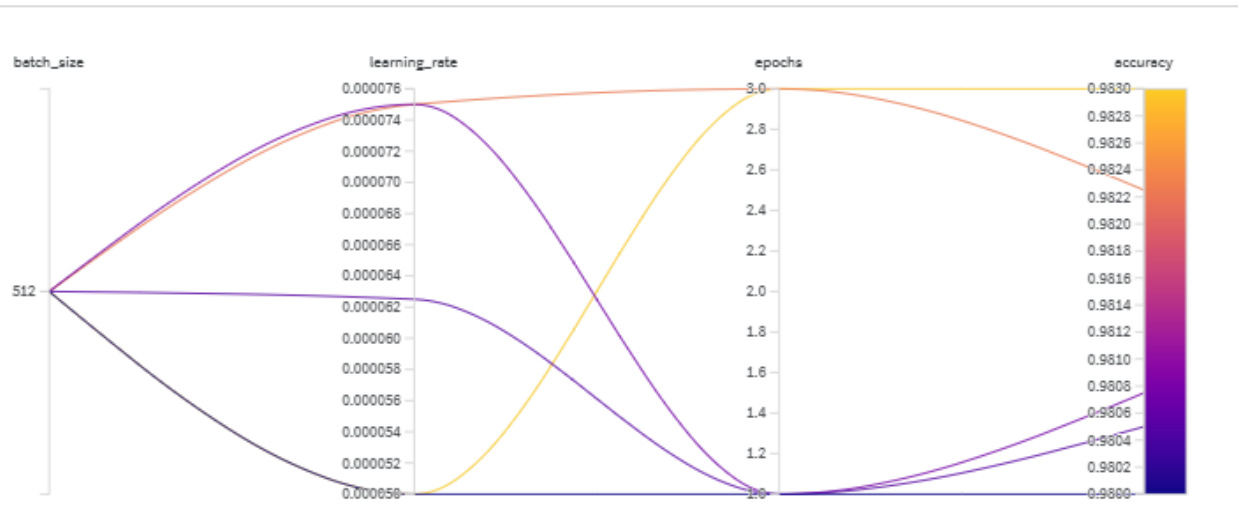
Optimizer: AdamW



LR : 5e-5 ~ 7e-5

4. 진행 결과

하이퍼파라미터 최적화(Wandb)



Optimizer: AdamW

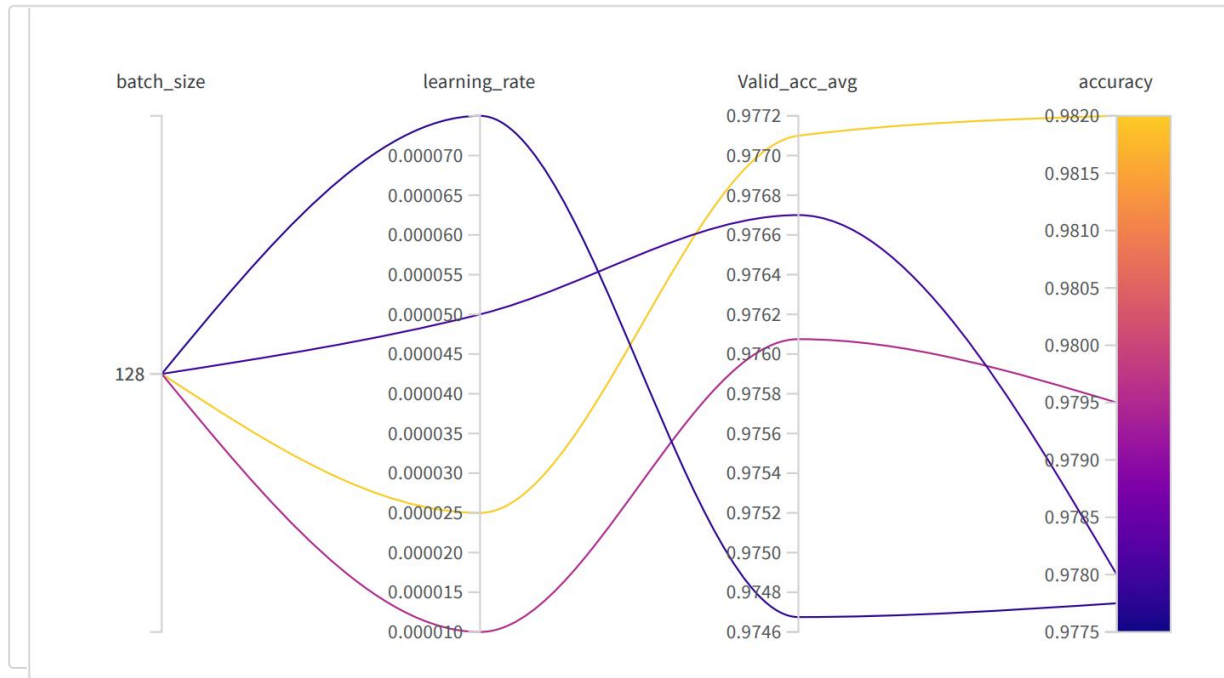
Batch-size: [32, 64, 128, 256, 512] 중 가장 큰 512

Epoch: 3 (epoch을 늘려도 training loss 크게 변동 없을 때 학습 종료)

LR: 5e-5 주변 값에서 scheduler 이용

4. 진행 결과

하이퍼파라미터 최적화(Wandb)



RoBERTa

Optimizer: AdamW

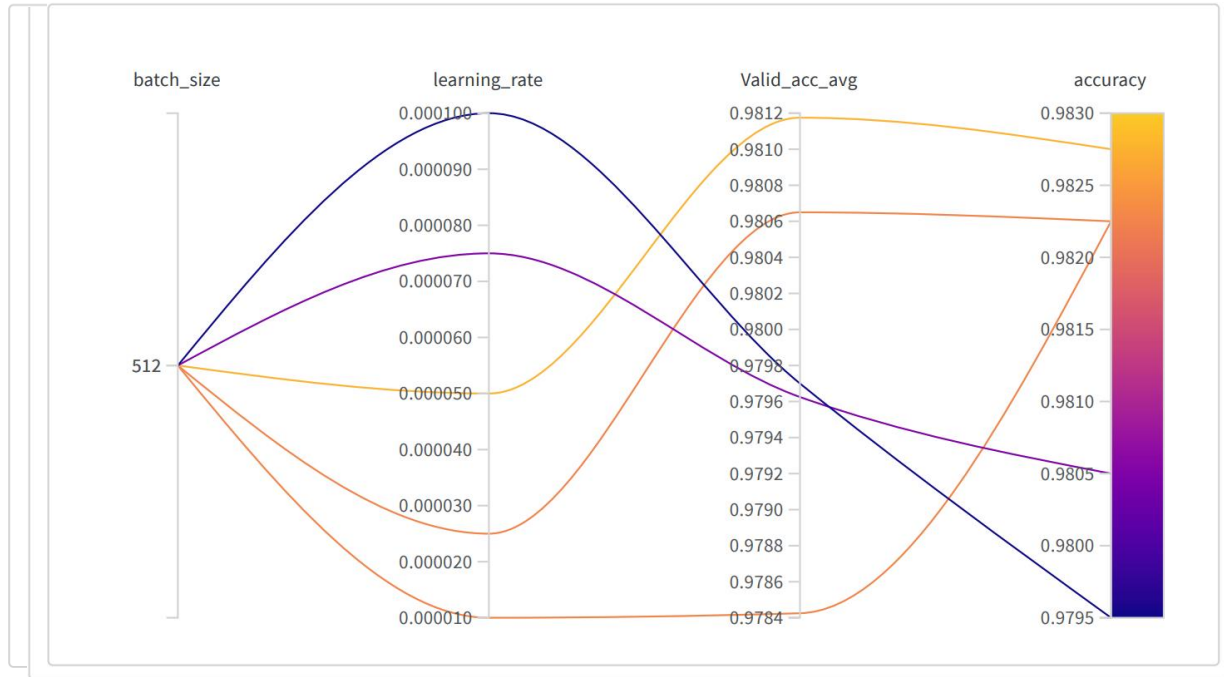
Batch-size: 128 (OOM 문제로 제한)

Epoch: 2

LR: 2.5e-5

4. 진행 결과

하이퍼파라미터 최적화(Wandb)



ELECTRA

Optimizer: AdamW

Batch-size: 512

Epoch: 2

LR: 5e-5

4. 진행 결과

앙상블 - Hard Voting

Hard Voting 기법

3개 모델의 예측값 0, 1 중 더 많이 나온
값을 최종 예측값으로 채택

테스트 결과 **99%** 성능 달성

```
[ ] pred_df = pd.DataFrame({'pred1':pred1, 'pred2':pred2, 'pred3':pred3})
```

```
def find_most_frequent(row):  
    counts = row.value_counts()  
    return counts.idxmax()  
  
# Apply the function to the DataFrame and store the result in a new column  
pred_df['most_frequent'] = pred_df.apply(find_most_frequent, axis=1)  
  
print(pred_df)
```

	pred1	pred2	pred3	most_frequent
0	1	1	1	1
1	0	0	0	0
2	1	1	1	1
3	0	0	0	0
4	1	1	1	1
...
995	1	1	1	1
996	1	1	1	1
997	0	0	0	0
998	0	0	0	0
999	0	0	0	0

[1000 rows x 4 columns]

ABOUT

Yelp 식당 리뷰 데이터를 활용한 문장 분류기 성능 개선 프로젝트



프로젝트 개요

dataset과 baseline code가 주어진 상태에서 모델의 성능을 개선 시킬 것

초기 정확도: 98.1%

train dataset: 443259

목표 정확도: 99.0%

valid dataset: 4000

주어진 기간: 5일

최종 정확도: 99.0%

식당 리뷰의 긍정/부정을 판단하는 이진분류모델의 성능 개선을 위한 여러가지 접근과 시도를 한다.

이를 통해 개발 전반적인 과정을 학습하고, 다양한 성능 향상 방법을 익히는 것을 목적으로 한다.

COLLABORATION TOOLS



5. 자체 평가 및 보완

- 평가 결과, 초기 목표였던 정확도 99%의 성능 달성
- 각 모델에 최적화된 전처리를 통해 전처리 효과 개선 필요
- 다양성을 부여한 점에서 효과가 있었으나, 모든 모델의 구조를 완벽히 이해하지는 못하고 이용한 점 보완 필요
- 하이퍼파라미터 최적화 도구(Wandb 등)에 관한 이해도를 높여 다음 프로젝트에 더욱 다양한 시도 기대
- 웹 서비스화 할 수 있도록 서버 작업 진행

첫 번째 프로젝트. Yelp 식당 리뷰 데이터 를 활용한 문장 분류기 성능 개선

2조. 김도연, 양은지, 오희주, 이다원, 이지수, 전효택, 정명찬

감사합니다