

# DLTHON DKTC 데이터 프로젝트

## 아이펠

MEMBER

김서영

성연우

윤수영

이창민



# INDEX

01

[프로젝트 목표](#)

02

[데이터 EDA](#)

03

[데이터 전처리](#)

04

[모델 선택 및 훈련](#)

05

[결과](#)

06

[새로운 시도](#)

07

[프로젝트 결과 회고](#)

08

[Q&A](#)

01

# 프로젝트 목표

| 1) 총괄 목표

## 총괄 목표

- 프로젝트 제출
- f-1 score 0.8 이상 달성

01

# 프로젝트 목표

| 2) 세부 목표

01. '일반 대화' 데이터

다양한 방법으로 "일반 대화" 클래스 데이터 수집

02. 데이터 EDA

데이터 EDA를 적절히 사용하여 모델 성능 개선

03. 모델 비교

다양한 모델 구조를 구축하여 모델 성능 모니터링

04. 성능

목표 성능 도달

05. 최적화

모델 튜닝을 통해 최적의 모델 도출

## 02

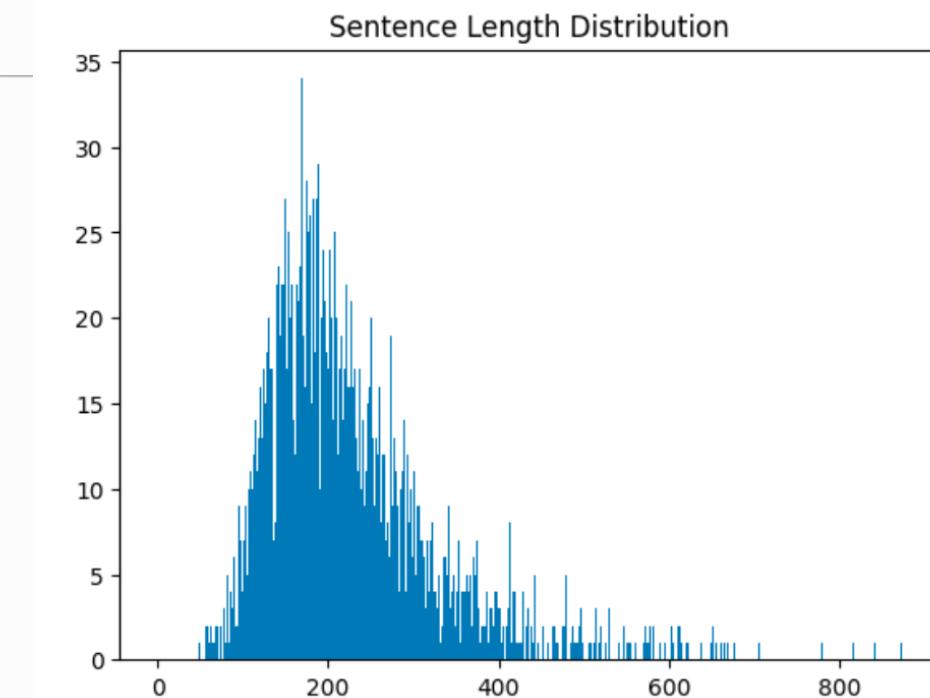
## 데이터 EDA

## | 1) 데이터 수집 : 기본 제공 데이터

DKTC\_train data

class	Class	total
협박	0	896
M갈취	1	981
직장 내 괴롭힘	2	979
기타 괴롭힘	3	1094

idx	class	conversation
0	0	협박 대화 지금 너 스스로를 죽여달라고 애원하는 것인가?\n 아닙니다. 죄송합니다.\n 죽을 ...
1	1	협박 대화 길동경찰서입니다.\n9시 40분 마트에 폭발물을 설치할거다.\n네?\n똑바로 들어 ...
2	2	기타 괴롭힘 대화 너 되게 귀여운거 알지? 나보다 작은 남자는 첨봤어.\n그만해. 니들 놀리는거 재미...
3	3	갈취 대화 어이 거기!\n예??\n너 말이야 너. 이리 오라고\n무슨 일.\n너 웃 좋아보인다?...
4	4	갈취 대화 저기요 혹시 날이 너무 뜨겁잖아요? 저희 회사에서 이 선크림 파는데 한 번 손등에 ...

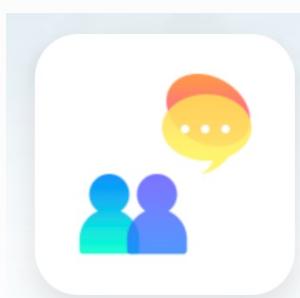


02

# 데이터 EDA

| 1) 데이터 수집 : 수집한 데이터

AIHUB

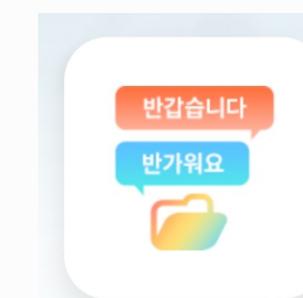


#코퍼스 #감성대화 #감성챗봇 #우울증 예방

## 감성 대화 말뭉치

분야 한국어 유형 오디오, 텍스트

구축년도: 2020 | 갱신년월: 2022-12 | 조회수: 103,178 | 다운로드: 11,463 | 용량: 20.35 MB



#대화 #주제별 대화 #일상대화 #대화주제 #화행

## 주제별 텍스트 일상 대화 데이터

분야 한국어 유형 텍스트

구축년도: 2021 | 갱신년월: 2022-07 | 조회수: 31,685 | 다운로드: 3,127 | 용량: 256.73 MB



#도메인 및 카테고리 정보 #의도 #엔티티 #시소스 #대화 데이터 #인공지능 챗봇 #민원 상담 데이터

## 한국어 대화

분야 한국어 유형 텍스트

구축년도: 2018 | 갱신년월: 2021-10 | 조회수: 19,682 | 다운로드: 8,003 | 용량: 7.76 MB

02

# 데이터 EDA

## | 2) 데이터 확인 및 분석



### • 대화가 아닌 행은 잘라낸 후 사용

14803	고객	요새 세일:A	음식점	셀프서비스	1	32	할인메뉴문의	Q	요새 세일하는 거 있어요?	세일
14804	점원	요새는 세:A	음식점	셀프서비스	0	33	할인메뉴문의	A	요새는 세일	
14805	고객	지금 기다:A	음식점	홀서빙음식	1	1	웨이팅여부문의	Q	지금 기다리면 자리 나올까요?	
14806	고객	6명 식사하:A	음식점	홀서빙음식	1	2	웨이팅여부문의	Q	6명 식사하려면 얼마나 기다려야 하나요?	
14807	고객	할인쿠폰:A	음식점	홀서빙음식	1	3	할인쿠폰사용	Q	할인쿠폰 주말에도 사용 가능한가요?	
14808	고객	4명 김치:A	음식점	홀서빙음식	1	4	할인쿠폰사용	Q	4명 김치 전골 시킬건데 할인 쿠폰 사용 가능하죠?	
14809	고객	여긴 2시간:A	음식점	홀서빙음식	1	5	주차비문의	Q	여긴 2시간 초과되면 주차비가 얼마죠?	
14810	고객	할인 메뉴:A	음식점	홀서빙음식	1	6	할인메뉴문의	Q	할인 메뉴는 매일 달라지나요?	
14811	고객	오늘의 할:A	음식점	홀서빙음식	1	7	할인메뉴문의	Q	오늘의 할인 메뉴는 뭐예요?	
14812	고객	이것도 할:A	음식점	홀서빙음식	1	8	할인메뉴문의	Q	이것도 할인 메뉴인가요?	
14813	고객	제 앞에 몇:A	음식점	홀서빙음식	1	9	대기자수문의	Q	제 앞에 몇 명 있어요?	
14814	고객	몇 명 남았:A	음식점	홀서빙음식	1	10	대기자수문의	Q	몇 명 남았어요?	
14815	고객	앞으로 두:A	음식점	홀서빙음식	1	11	대기자수문의	Q	앞으로 두 명 남은 거 맞나요?	
14816	고객	제가 몇 번:A	음식점	홀서빙음식	1	12	대기자수문의	Q	제가 몇 번째 대기순서죠?	
14817	고객	지금 런치:A	음식점	홀서빙음식	1	13	런치메뉴주문문의	Q	지금 런치메뉴 시킬 수 있죠?	
14818	고객	이 짐에 콩:A	음식점	홀서빙음식	1	14	양에대한요청	Q	이 짐에 콩나물 양이 너무 적어요 좀 더 넣어주세요	
14819	고객	한참 기다:A	음식점	홀서빙음식	1	15	대기시간문의	Q	한참 기다려야되죠?	
14820	고객	오래 기다:A	음식점	홀서빙음식	1	16	대기시간문의	Q	오래 기다려야 해요?	
14821	고객	많이 기다:A	음식점	홀서빙음식	1	17	대기시간문의	Q	많이 기다려야 되죠?	
14822	고객	짜장면이니:A	음식점	홀서빙음식	1	18	미리주문메뉴문의	Q	짜장면이나 짬뽕도 미리 주문 해놓을 수 있나요?	
14823	고객	요리 메뉴:A	음식점	홀서빙음식	1	19	미리주문메뉴문의	Q	요리 메뉴만 먼저 주문하면 되나요?	
14824	고객	미리 주문:A	음식점	홀서빙음식	1	20	미리주문메뉴문의	Q	미리 주문해야 좋은 메뉴는 뭐가 있을까요?	
14825	고객	삼계탕은:A	음식점	홀서빙음식	1	21	미리주문메뉴문의	Q	삼계탕은 미리 주문 해야하죠?	
14826	고객	똥암꿍은:A	음식점	홀서빙음식	1	22	미리주문메뉴문의	Q	똥암꿍은 미리 주문해야 되나요? 와서 주문하면 되나요?	
14827	고객	1시간 무료:A	음식점	홀서빙음식	1	23	무료주차시간문의	Q	1시간 무료 주차 인가요?	
14828	고객	30분 무료:A	음식점	홀서빙음식	1	24	무료주차시간문의	Q	30분 무료 주차 인가요?	
14829	고객	토요일은:A	음식점	홀서빙음식	1	25	무료주차시간문의	Q	토요일은 무료 주차 몇 시간해요?	

02

# 데이터 EDA

| 2) 데이터 확인 및 분석



- 분석에 방해될 수 있는 토큰 확인

웅 화해는 잘했지 근데 내일 곱창으로 화해의 마무리를 지으려고 했는데 시간이 안돼...  
그냥 곱창이 먹고 싶은 거지? 아쉽겠네  
화해 안 한 걸로 하자 그럼  
그래서 목요일은 제발 일이 빨리 끝나길 바라고 있어 그날 남자친구한테 곱창 얻어먹으려고  
쿠쿄쿄쿄쿄쿄 다음 주에 먹어 그냥 ,  
아마 안될 거 같아 내 생각엔  
안돼 나 남자친구랑 화해하게 도와주라 좀  
그래 그냥 다음 주에 먹어 포기하고  
싫어 화해하지 마 싸워 히히히  
아 왜 남자친구랑 화해하고 싶다고  
그냥 집에서 먹엉... 그게 답

## 03

## 데이터 전처리

| 1) 기본 전처리

- `remove_nan()` : 결측치 제거 함수
- `remove_punctuation()` : 특수문자, 공백 제거 함수. 대화자 분리 필요시 skip

```
# 결측치 제거
def remove_nan(x):
    # 결측치나 빈 문자열이 있으면 None을 반환 (삭제 효과)
    if pd.isna(x) or x == '':
        return None # None을 반환하면 해당 값이 삭제됨
    return x
```

```
# 특수문자, 공백 제거
def remove_punctuation(x):
    x = re.sub("[^ㄱ-ㅎ가-힣]+", " ", x)
    x = re.sub("[ ]+", " ", x)
    x = x.strip()
    return x
```

## 03

## 데이터 전처리

| 1) 기본 전처리

- 각 클래스에 정수를 매핑해서 라벨 데이터를 정수 인코딩하는 함수

```
# 라벨 정수 인코딩
def label_encoder(x):
    label_map = {'협박 대화': 0, '갈취 대화': 1, '직장 내 괴롭힘 대화': 2, '기타 괴롭힘 대화': 3, '일반 대화': 4}
    x['encoded_label'] = x['class'].map(label_map)
    print(x['encoded_label'].unique())

    return x
```

## 03

## 데이터 전처리

| 1) 기본 전처리

- “일반 대화” 파일을 로드해서 과도하게 짧거나 긴 대화 삭제 후 return

```
# AIHUB에서 수집한 데이터 로드
def load_collected_data(file_path):
    df = pd.read_csv(file_path)

    df['class'] = "일반 대화"

    # 대화를 몇 번 주고 받았는지 확인
    df["turn_num"] = df["conversation"].apply(lambda x : len(x.split("\n")))

    # 너무 긴 대화는 삭제
    df = df[(df["turn_num"] >= 4) & (df["turn_num"] <= 15)][["class", "conversation"]]
    print(len(df))

return df
```

## 03

## 데이터 전처리

| 1) 기본 전처리

- 분석에 방해되는 단어를 판단하여 삭제

```
# 숫자, 콜론, 자음, 특수문자 제거하는 함수
def remove_unwanted_characters(text):

    text = re.sub(r'₩b키키₩b', '', text) # 정확히 "키키"만 제거
    text = re.sub(r'₩b하핳₩b', '', text) # 정확히 "핳핳"만 제거
    text = re.sub(r'₩b呵呵₩b', '', text) # 정확히 "呵呵"만 제거
    text = re.sub(r'₩bଓଓ₩b', '', text) # 정확히 "ଓଓ"만 제거
    text = re.sub(r'₩bଓই₩b', '', text)

# 숫자, 콜론, 한글 자음 (ㄱ-ㅎ), 특수문자 제거하는 정규 표현식
cleaned_text = re.sub(r'[0-9:₩₩@#$%^&*()":{}|<>ㄱ-ㅎㅏ-ㅣ]+', '', text)
return cleaned_text
```

## 03

## 데이터 전처리

| 2) 데이터 증강

- 수집한 '일반 대화' 클래스의 데이터에 비해 사전에 주어진 train 데이터가 부족하기에 데이터 증강 사용
- 사용한 데이터 증강 기법
  - RD : 임의로 일부 데이터를 삭제하여 모델의 일반화 능력을 높이는 기법
    - 원본 : "지갑어디갔지 에이 버스에서 잃어버렸나보네 그럼 취소할까요"
    - RD 적용 : "지갑어디갔지 에이 버스에서 잃어버렸나보네 취소할까요" (그럼 삭제)
  - RDS : 삭제된 데이터를 다른 값으로 대체해 데이터의 다양성을 증가시키는 기법
    - 원본 : "아가씨 담배피는교"
    - RDS 적용 : "아가씨 담배피느냐"

03

# 데이터 전처리

| +) 대화자 구분 모델

- test 데이터셋은 train 데이터셋과 달리 '\n' 구분자가 없음
  - = 대화자를 구분할 수 없음
- test 데이터에서 대화자를 구분해서 의미있는 패턴을 도출해낼 수 있을 것이라 판단
- BERT 기반 모델을 사용해 대화자 구분 모델 구현

## 04

## 모델 선택 및 훈련

| 사용한 모델

## • BERT 기반

양방향 문맥이해를 통해 각 단어의 의미를 정확히 파악  
대규모 Corpus로 사전 학습된 모델을 미세 조정해 높은 성능 발휘

## • GPT 기반

자연스러운 텍스트 생성 기반으로 문맥 이해에 뛰어남  
긴 텍스트도 잘 처리하기에 복잡한 분류 작업에 높은 정확도를 보임

# 04 모델 선택 및 훈련

## multilingual-cased

- BERT, ALBERT 사용
- 다국어 전문 모델
- 학습 문제 (acc 20%)
- 한국어 전문 사전 훈련 모델 필요성

BERT 기반 모델

## KLUE

- bert-base 한국어 모델
- 학습은 제대로 진행
- test셋 예측 성능 문제
- 양질의 데이터 필요성

## GPT

- koGPT-2 사용
- 한국어 전문 모델
- test셋 예측 성능 문제
- 평가지표 교체 필요성
- 데이터 증강 필요성
- 대화 구분자 추가 필요성

## 04

## 모델 선택 및 훈련

| 1) BERT 기반 모델

multilingual-cased

- 기반 모델 : <https://huggingface.co/google-bert/bert-base-multilingual-cased>
- 모델 Summary :

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 200)]	0	
attention_mask (InputLayer)	[(None, 200)]	0	
tf_bert_model (TFBertModel)	TFBaseModelOutputWithPooling	177853440	input_ids[0] [0] attention_mask[0] [0]
dense (Dense)	(None, 32)	24608	tf_bert_model [0] [1]
dropout_37 (Dropout)	(None, 32)	0	dense[0] [0]
dense_1 (Dense)	(None, 5)	165	dropout_37[0] [0]
<hr/>			
Total params: 177,878,213			
Trainable params: 177,878,213			
Non-trainable params: 0			
<hr/>			

## 04

## 모델 선택 및 훈련

## | 1) BERT 기반 모델

multilingual-cased

## ● 모델 주요 함수 :

- BERT tokenizer 기반으로 토큰화 및 패딩 마스크 제작 함수
- BERT pre\_trained 모델을 기반으로 fine\_tuning 하는 모델을 구축하는 함수
- 위 함수들 기반으로 구축한 모델과, 토큰 인코딩 데이터 및 패딩 마스크를 입력
- 대화 유형 class를 label 데이터로 입력 받아 모델 학습



```
def bert_encode(datas, sent_max_length):
    input_ids = []
    attention_masks = []

    for sent in datas:
        encoded = tokenizer.encode_plus(sent,
                                        add_special_tokens = True,
                                        max_length = sent_max_length,
                                        padding='max_length',
                                        truncation = True,
                                        return_attention_mask=True)

        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])

    return np.array(input_ids), np.array(attention_masks)
```

```
def create_model(bert_model):
    input_ids = tf.keras.Input(shape=(200,), dtype=tf.int32)
    attention_mask = tf.keras.Input(shape=(200,), dtype=tf.int32)

    output = bert_model([input_ids, attention_mask])
    output = output.last_hidden_state[:,0,:]
    output = tf.keras.layers.Dense(32, activation='relu')(output)
    output = tf.keras.layers.Dropout(0.2)(output)
    output = tf.keras.layers.Dense(5, activation='softmax')(output)

    model = tf.keras.Model(inputs = [input_ids, attention_mask], outputs = output)
    model.compile(Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    return model
```

```
history = use_bert_model.fit([train_input_ids, train_attention_mask], train_label, validation_split=0.2, epochs = 10, batch_size=16)
```

## 04

## 모델 선택 및 훈련

| 1) BERT 기반 모델

KLUE

● **KLUE (Korean Language Understanding Evaluation) BERT**

- 한국어 자연어 처리(NLP) 작업을 위한 BERT 모델
- 한국어의 어순이나 형태소 분석에서 나타나는 특성을 반영하기 위해 추가적인 데이터 전처리 및 학습이 이루어짐

## ● 모델 아키텍처

- Layer: 12개의 Transformer 인코더 층
- Hidden size: 768
- Attention heads: 12개
- Parameter 수: 약 1억 1천 1백만 개 (110M)

## 04

## 모델 선택 및 훈련

| 1) BERT 기반 모델

## KLUE

- 기반 모델: <https://huggingface.co/klue/bert-base>
- 모델 Summary : Model: "tf\_bert\_for\_sequence\_classification"

Layer (type)	Output Shape	Param #
<hr/>		
bert (TFBertMainLayer)	multiple	110617344
dropout_37 (Dropout)	multiple	0
classifier (Dense)	multiple	3845
<hr/>		
Total params:	110,621,189	
Trainable params:	110,621,189	
Non-trainable params:	0	

## 04

## 모델 선택 및 훈련

| 1) BERT 기반 모델

## GPT

- 기반 모델 : <https://huggingface.co/skt/kogpt2-base-v2>
- 모델 Summary :

Model : "model_2"	Layer (type)	Output Shape	Param #
<hr/>			
input_6 (InputLayer)		[ (None, 300) ]	0
tfgpt2model (TFGPT2Model)		TFBaseModelOutputWithPast	125164032
tf.__operators__.getitem_4	( (None, 768) )		0
dense_2 (Dense)		(None, 5)	3845
<hr/>			
Total params: 125,167,877			
Trainable params: 125,167,877			
Non-trainable params: 0			

# 05 | 결과

## multilingual-cased

- BERT, ALBERT 사용
- 다국어 전문 모델
- 학습 문제 (acc 20%)
- 한국어 전문 사전 훈련 모델 필요성

BERT 기반 모델

## KLUE

- bert-base 한국어 모델
- 학습은 제대로 진행
- test셋 예측 성능 문제
- 양질의 데이터 필요성

## GPT

- koGPT-2 사용
- 한국어 전문 모델
- test셋 예측 성능 문제
- 평가지표 교체 필요성
- 데이터 증강 필요성
- 대화 구분자 추가 필요성

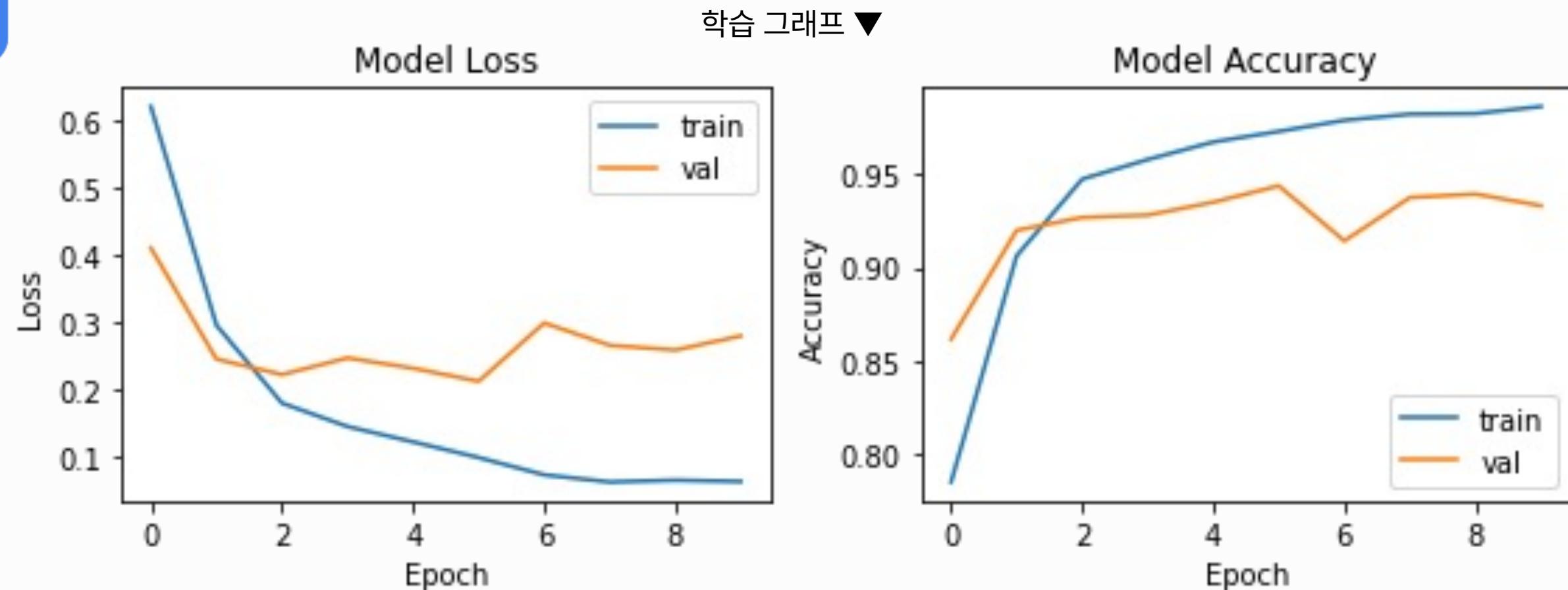
## 05

## 결과

## | 1) BERT 기반 모델

multilingual-cased

- 모델 결과:



미리 준비한 검정 세트로 성능 평가 ▼

```
use_bert_model.evaluate([test_input_ids, test_attention_mask], test_label)  
63/63 [=====] - 11s 174ms/step - loss: 0.2998 - accuracy: 0.9370  
[0.2998300790786743, 0.9369999766349792]
```

05

결과

| 1) BERT 기반 모델

multilingual-cased

## ● 모델 Submission :

**submission.csv**  
Complete · yengerche · 3d ago

Score:  
Public score: 0.61031

UPLOADED FILES  
 submission.csv (3 KiB)

DESCRIPTION  
Team2 Model by BERT result  
26 / 500

SELECT FOR FINAL SCORE  
 Select this submission to be scored for your final leaderboard score

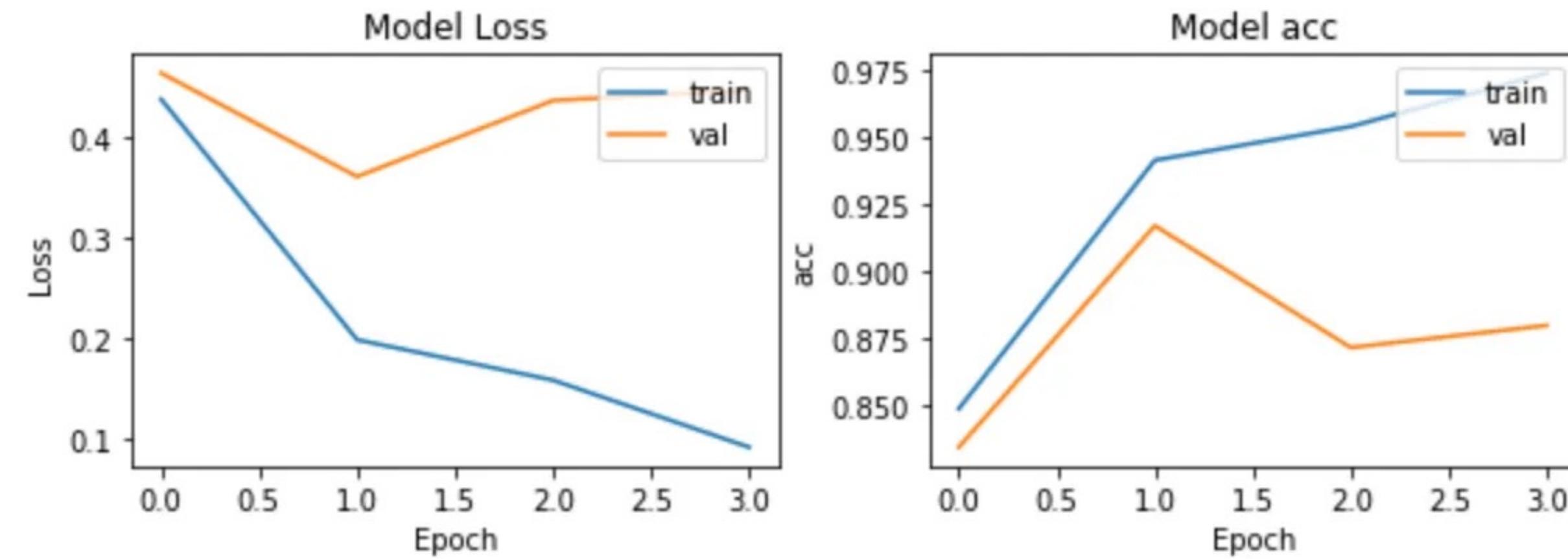
## 05

## 결과

| 1) BERT 기반 모델

KLUE

- 모델 결과:



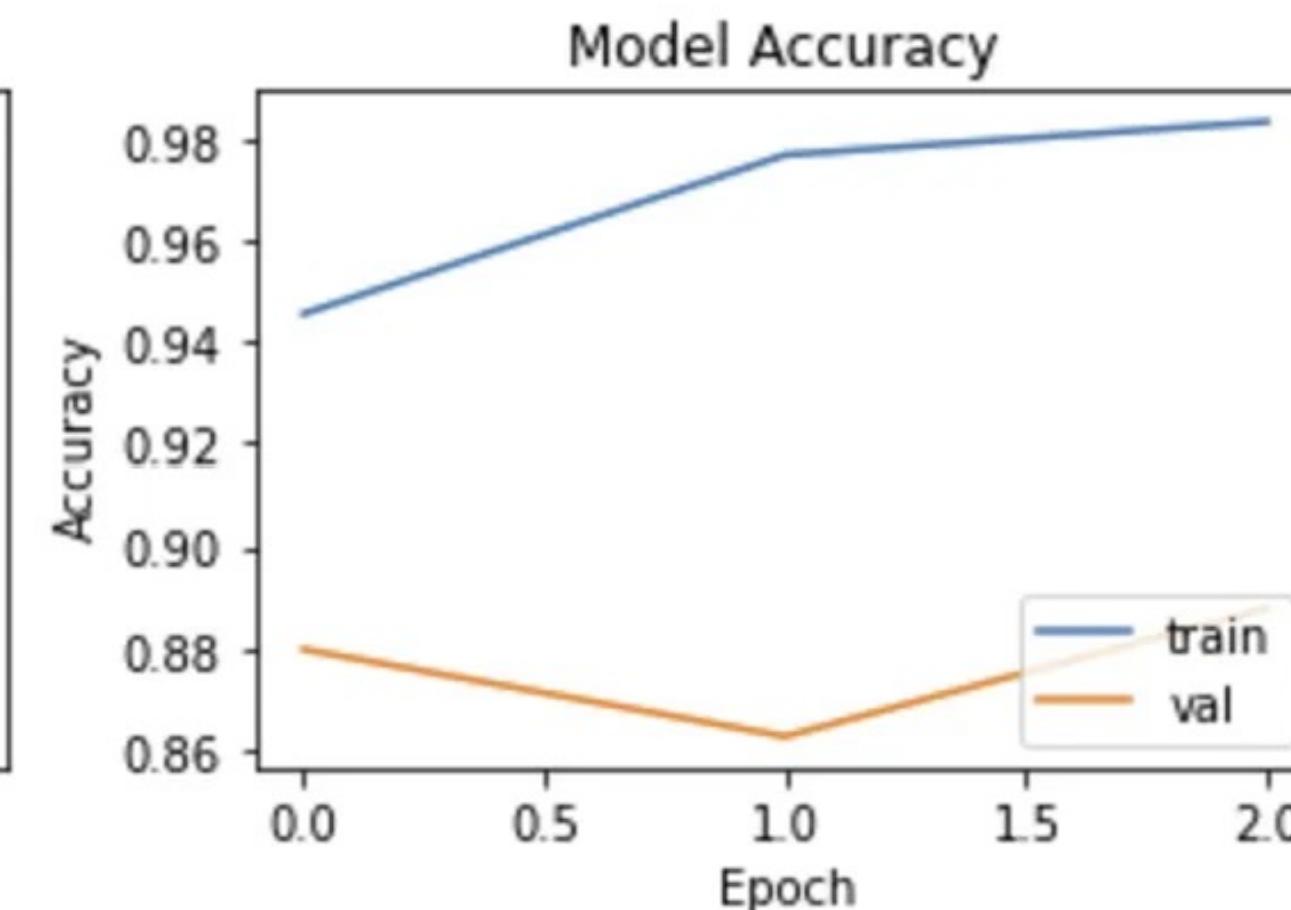
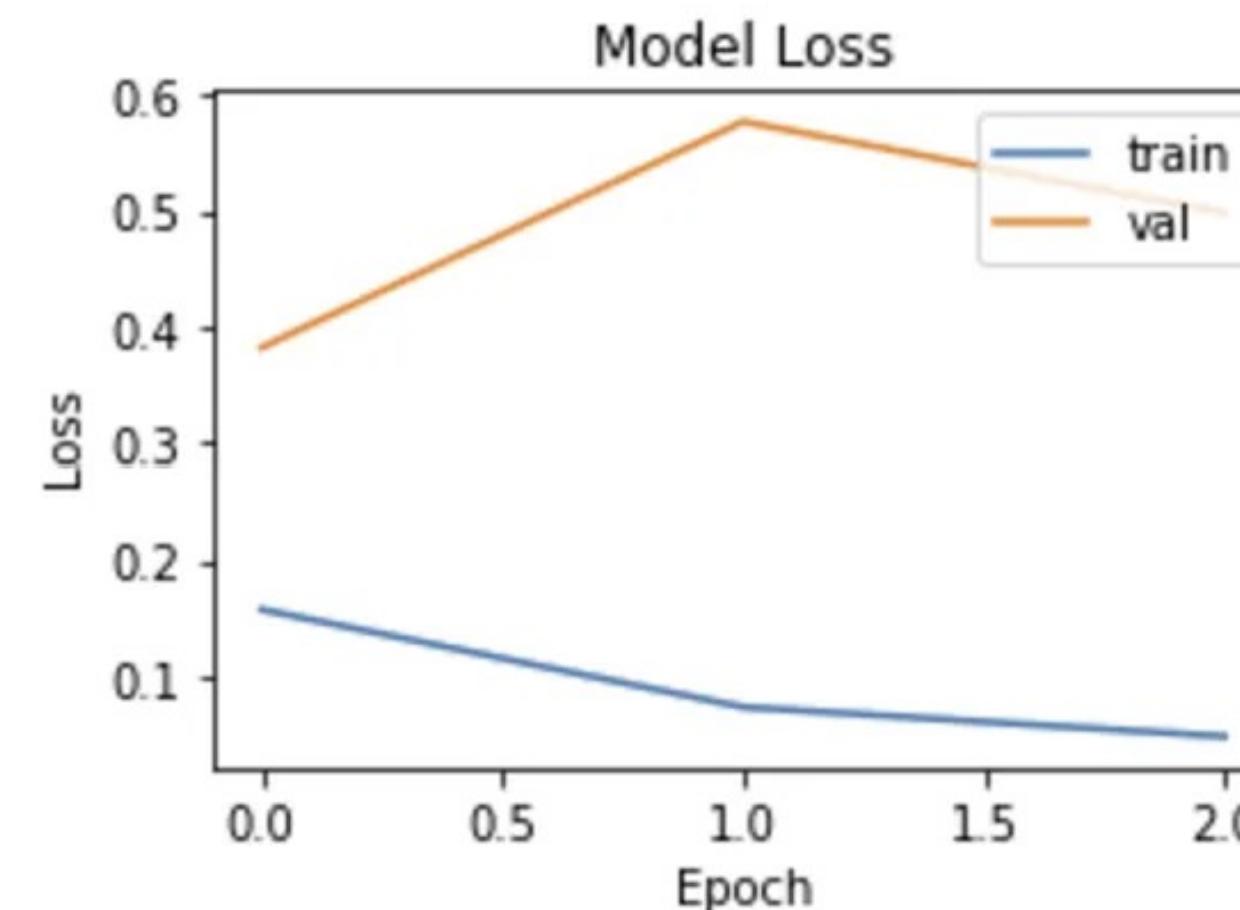
## 05

## 결과

| 1) BERT 기반 모델

## GPT

- 모델 결과:



05

결과

| 1) BERT 기반 모델

GPT

## ● 모델 Submission :



submission.csv

Complete · seoyeo · 4d ago

Score:

Public score: 0.56640

## UPLOADED FILES

submission.csv (3 KiB)



## DESCRIPTION

base\_gpt

8 / 500

## SELECT FOR FINAL SCORE

 Select this submission to be scored for your final leaderboard score

# 06 | 새로운 시도

## 대화자 구분 모델

- Test 데이터 대화자 구분
- SBERT 기반 모델 사용
- 다국어 전문 모델
- 증강된 데이터를 이용하여, 정확도 상승 (80% -> 90%)

## 데이터 증강

- RD, RSW 증강
- Back Translation
- MBART 기반 모델 사용
- 속도 증가의 필요성
- 다양성 증가의 필요성

## 최종 예측 모델

- koGPT2 모델 사용
- 구분자 test 데이터 사용
- RD, RSW 증강 데이터 사용
- 일반 대화 데이터 2만건
- SC-accuracy 사용

# 06 | 새로운 시도

## 대화자 구분 모델

- 기반 모델 : <https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2>
- 모델 Summary :

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 240)]	0	
input_2 (InputLayer)	[(None, 240)]	0	
tf_bert_model (TFBertModel)	TFBaseModelOutputWithLogits	117653760	input_1[0] [0] input_2[0] [0]
dense (Dense)	(None, 240, 64)	24640	tf_bert_model[0] [13]
dropout_37 (Dropout)	(None, 240, 64)	0	dense[0] [0]
dense_1 (Dense)	(None, 240, 3)	195	dropout_37[0] [0]
<hr/>			
Total params: 117,678,595			
Trainable params: 117,678,595			
Non-trainable params: 0			

# 06 | 새로운 시도

## 대화자 구분 모델

- Bert 모델의 last\_hidden\_state 출력 전체를 받아옴

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 240)]	0	
input_2 (InputLayer)	[(None, 240)]	0	
tf_bert_model (TFBertModel)	TFBaseModelOutputWith 117653760	input_1[0] [0] input_2[0] [0]	
dense (Dense)	(None, 240, 64)	24640	tf_bert_model[0] [13]
dropout_37 (Dropout)	(None, 240, 64)	0	dense[0] [0]
dense_1 (Dense)	(None, 240, 3)	195	dropout_37[0] [0]
<hr/>			
Total params:	117,678,595		
Trainable params:	117,678,595		
Non-trainable params:	0		

## 모델 구조상 차이점

## multilingual-cased

- last\_hidden\_state의  $[:, 0, :]$  슬라이싱 단일값을 받아옴

Layer (type)	Output Shape	Param #	Connected to
input_ids (InputLayer)	[(None, 200)]	0	
attention_mask (InputLayer)	[(None, 200)]	0	
tf_bert_model (TFBertModel)	TFBaseModelOutputWith 177853440	input_ids[0] [0] attention_mask[0] [0]	
dense (Dense)	(None, 32)	24608	tf_bert_model[0] [1]
dropout_37 (Dropout)	(None, 32)	0	dense[0] [0]
dense_1 (Dense)	(None, 5)	165	dropout_37[0] [0]
<hr/>			
Total params:	177,878,213		
Trainable params:	177,878,213		
Non-trainable params:	0		

# 06 | 새로운 시도

## 대화자 구분 모델

- 모델 중요 함수:
- BERT tokenizer 기반으로 토큰화 및 패딩 마스크 제작 함수
- BERT pre\_trained 모델을 기반으로 fine\_tuning 하는 모델을 구축하는 함수

```
def bert_encode(datas, sent_max_length):
```

```
    input_ids = []
    attention_masks = []
```

```
    for sent in datas:
```

```
        encoded = tokenizer.encode_plus(sent,
                                         add_special_tokens = True,
                                         max_length = sent_max_length,
                                         padding='max_length',
                                         truncation = True,
                                         return_attention_mask=True)
```

```
        input_ids.append(encoded['input_ids'])
        attention_masks.append(encoded['attention_mask'])
```

```
    return np.array(input_ids), np.array(attention_masks)
```

• 토큰 인코딩 데이터

• 문장 패딩 마스크

```
def create_model(bert_model, max_len):
```

```
    input_ids = tf.keras.Input(shape=(max_len,), dtype=tf.int32)
    attention_mask = tf.keras.Input(shape=(max_len,), dtype=tf.int32)
```

```
    output = bert_model([input_ids, attention_mask])
```

```
    output = output.last_hidden_state
```

```
    output = tf.keras.layers.Dense(64, activation='relu')(output)
```

```
    output = tf.keras.layers.Dropout(0.2)(output)
```

```
    output = tf.keras.layers.Dense(3, activation='softmax')(output)
```

• last\_hidden\_state의 모든 출력을 받기 때문에, 최종 출력이 문장의 토큰 길이와 같게됨

```
model = tf.keras.Model(inputs = [input_ids, attention_mask], outputs = output)
```

```
model.compile(Adam(learning_rate=0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
return model
```

# 06 새로운 시도

## 대화자 구분 모델

- ### ● 모델 중요 함수 :

```
changed_data = []

for sentence in proto_data:
    sentence = re.sub(r'\n', ' 엔 ', sentence)
    changed_data.append(sentence)

prot_data = np.array(changed_data)
```

- 토큰 인코딩이 되어진 input\_ids 데이터를 기반으로 label 데이터를 만드는 함수

```

def make_label(data, index_num, max_len):
    return_data = []
    for conv in data:
        config_num = 0
        return_sige_sent = []
        for sen in conv:
            if sen == index_num:
                if config_num == 0:
                    config_num = 1
                else:
                    config_num = 0
            elif sen == 1:
                return_sige_sent.append(2)
            else:
                return_sige_sent.append(config_num)
        return_sige_sent = np.array(return_sige_sent)
        if return_sige_sent.shape[0] != max_len:
            add_len = max_len - return_sige_sent.shape[0]
            num_in = return_sige_sent[-1]
            for i in range(add_len):
                return_sige_sent = np.append(return_sige_sent, int(num_in))
            return_sige_sent = np.reshape(return_sige_sent, (1, max_len))
        return_data.append(return_sige_sent)
    return return_data

```

- `label_set`를 만들기 위한 전처리,  
걍 '\n'을 넣게되면 토큰화에서  
러지기 때문에 다른 문자로 변환

- 0과 1로 화자가 바뀔 때마다  
마스크가 바뀌도록 하였으며,  
문장이 끝나고 패딩이 시작되면 2로  
마스크가 채워짐

# 06 | 새로운 시도

## 대화자 구분 모델

- 모델 중요 함수 :

- 앞의 함수들을 이용하여 input 데이터와 label 데이터를 제작하고, 모델을 구축하여, 제작된 데이터를 넣고 모델을 fine\_tuning

- \*이후 fine\_tuning된 모델(가중치)를 이용하여 test\_data의 화자를 구분함

```

use_bert_model = create_model(bert_model, 240)
use_bert_model.summary()

Model: "model"
_________________________________________________________________
Layer (type)        Output Shape         Param #     Connected to
_________________________________________________________________
input_1 (InputLayer) [(None, 240)]      0           input_1[0][0]
input_2 (InputLayer) [(None, 240)]      0           input_2[0][0]
tf_bert_model (TFBertModel)   TFBASEMODELOUTPUT 117653760  input_1[0][0]
                                         input_2[0][0]
dense (Dense)       (None, 240, 64)    24640       tf_bert_model[0][13]
dropout_37 (Dropout) (None, 240, 64)    0           dense[0][0]
dense_1 (Dense)     (None, 240, 3)     195         dropout_37[0][0]
_________________________________________________________________
Total params: 117,678,595
Trainable params: 117,678,595
Non-trainable params: 0
_________________________________________________________________

checkpoint = tf.keras.callbacks.ModelCheckpoint("dlthon2.keras", monitor='val_accuracy', verbose=1, save_best_only=True, mode='max', save_weights_only=True)

history = use_bert_model.fit([train_a_input_ids_b, train_a_attention_mask_b], train_label_ids_np, validation_split=0.2, epochs = 10, callbacks=[checkpoint], batch_size=8)

```

\* [https://github.com/seoyeokim/Aiffel\\_DLThon/blob/main/code/Suyoung/dlthon\\_4.ipynb](https://github.com/seoyeokim/Aiffel_DLThon/blob/main/code/Suyoung/dlthon_4.ipynb)

# 06 | 새로운 시도

## 데이터 증강

### | 1) RD, RSW 증강

- RD : 임의의 단어를 삭제

각 단어를 p의 확률로 임의로 삭제

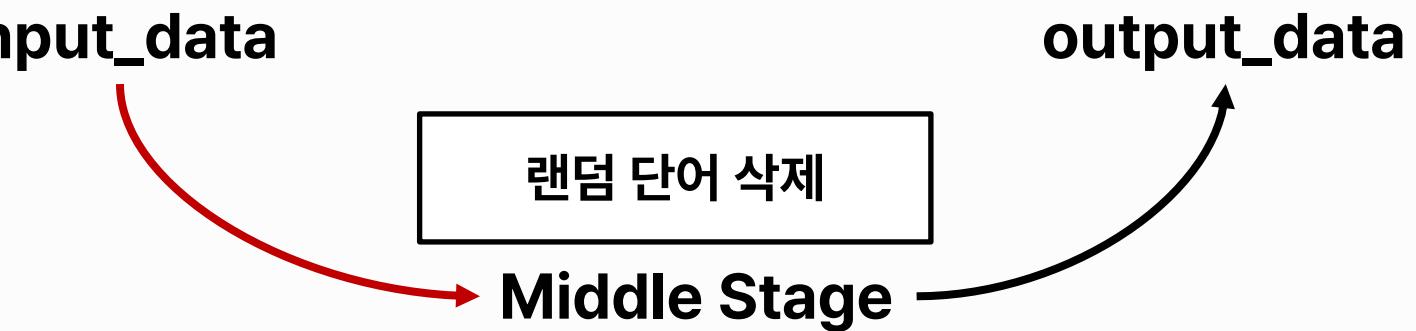
- 원본

정말 잘못했습니다.

너가 선택해. 너가 **죽을래** 네 가족을 **죽여줄까**.

죄송합니다. 정말 잘못했습니다.

**선택** 못하겠습니다. 한번만 도와주세요.



- RD 적용

정말 잘못했습니다.

너가 선택해. 너가 **네 가족을** (**죽을래**, **죽여줄까**)

죄송합니다. 잘못했습니다.

**못하겠습니다.** 한번만 도와주세요. (**선택**)

# 06 | 새로운 시도

데이터 증강

| 1) RD, RSW 증강

- RD : 임의의 단어를 삭제

30% 확률로 단어 삭제

```
# RD(Random Deletion) 함수
def random_deletion_with_newlines(text, p=0.3):
    lines = text.split("\n") # 텍스트를 줄 단위로 분리
    processed_lines = [] # 처리된 줄을 저장할 리스트

    # 단어를 RD 기법으로 처리
    retained_words = [word for word in words if random.random() > p]
    if len(retained_words) == 0: # 모든 단어가 삭제된 경우 최소 하나의 단어 유지
        retained_words = [random.choice(words)]

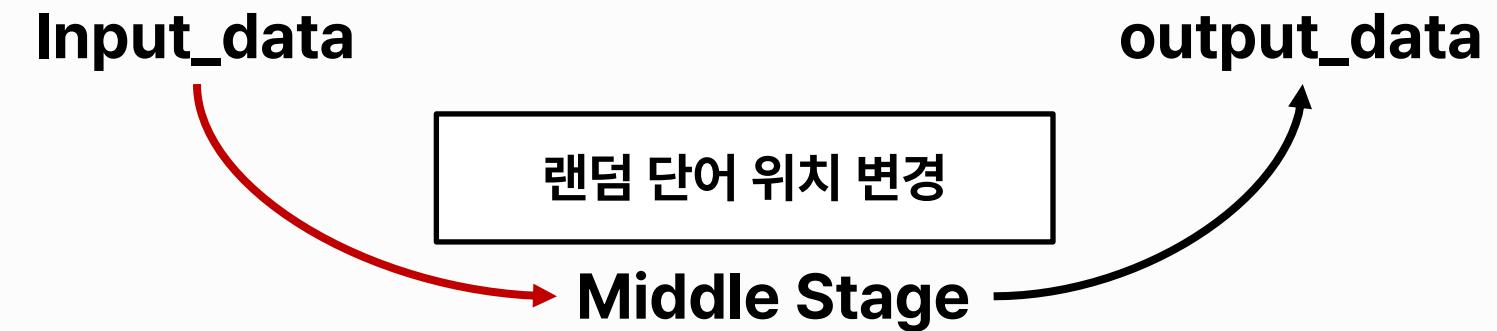
    # 줄바꿈 기호를 다시 연결하여 반환
    return "\n".join(processed_lines)
```

# 06 | 새로운 시도

## 데이터 증강

### | 1) RD, RSW 증강

- **RSW** : 문장 내 임의의 단어의 위치를 바꾸는 방식  
랜덤하게 2개의 단어를 선택하고 둘의 위치를 바꿈
- 원본  
정말 잘못했습니다.  
너가 선택해. 너가 **죽을래** 네 가족을 죽여줄까.  
죄송합니다. **정말** 잘못했습니다.  
선택 못하겠습니다. 한번만 도와주세요.



- RSW 적용  
정말 잘못했습니다.  
선택해. **죽을래** 너가 너가 네 가족을 죽여줄까. (**너가**, **죽을래**)  
잘못했습니다. 죄송합니다. **정말**  
도와주세요. 선택 한번만 못하겠습니다. (**정말**, **선택**)

# 06 | 새로운 시도

데이터 증강

| 1) RD, RSW 증강

- **RSW : 문장 내 임의의 단어의 위치를 바꾸는 방식**

단어들의 위치 변경 횟수

```
# RSW(Random Swap) 함수
def random_swap_with_newlines(text, num_swaps=1):
    lines = text.split("\n") # 텍스트를 줄 단위로 분리
    processed_lines = [] # 처리된 줄을 저장할 리스트

    # 단어를 RSW 기법으로 처리
    for _ in range(num_swaps):
        idx1, idx2 = random.sample(range(len(words)), 2) # 무작위로 두 인덱스 선택
        words[idx1], words[idx2] = words[idx2], words[idx1] # 단어 위치 교환

    # 줄바꿈 기호를 다시 연결하여 반환
    return "\n".join(processed_lines)
```

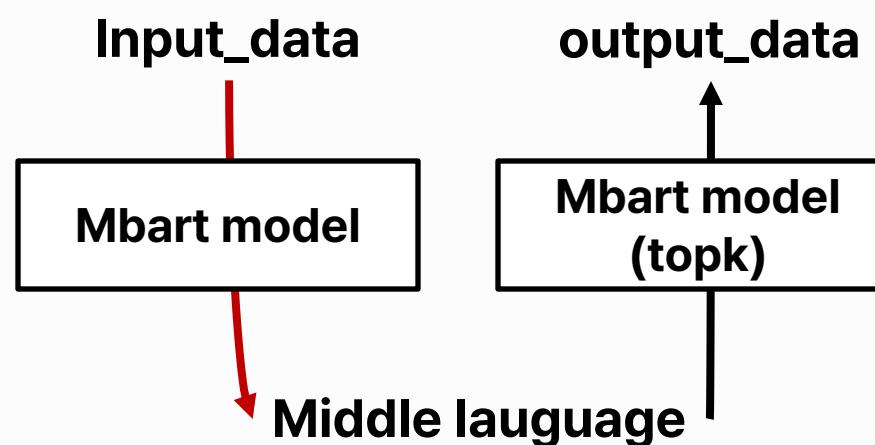
# 06 | 새로운 시도

데이터 증강

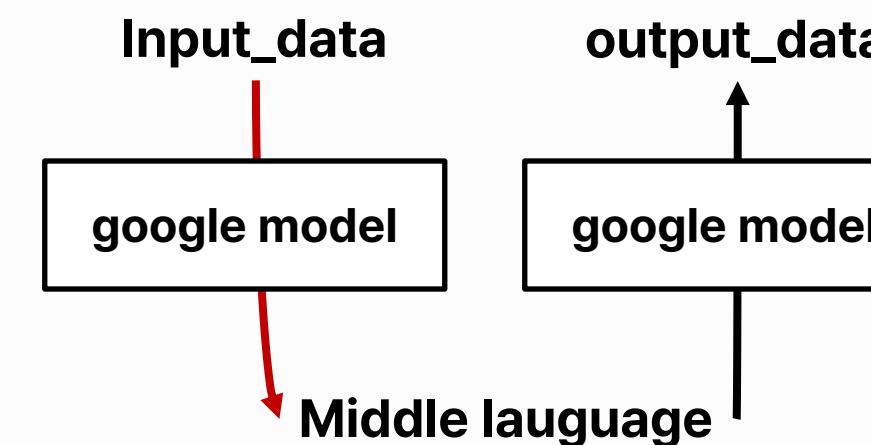
| 2) Back Translation 증강

- 기반 모델 1: <https://huggingface.co/facebook/mbart-large-50-many-to-many-mmt>
- 기반 모델 2: <https://pypi.org/project/googletrans/>
- Back Translation work flow(evaluation : sacrebleu):

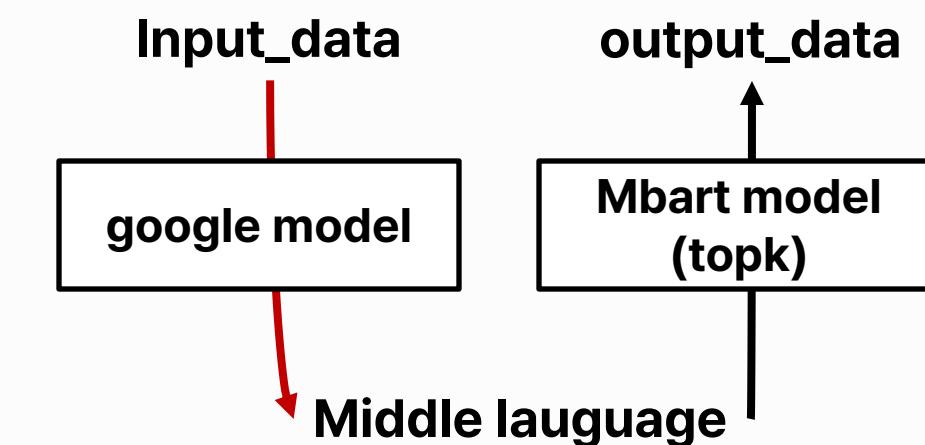
1) Mbart\_Back\_translation



2) google\_Back\_translation



3) collaboration\_Back\_translation



아이펠

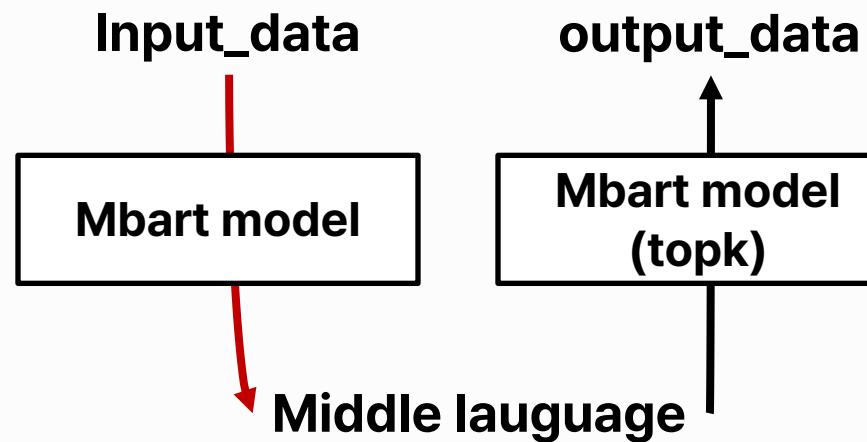
# 06 | 새로운 시도

## 데이터 증강

### | 2) Back Translation 증강

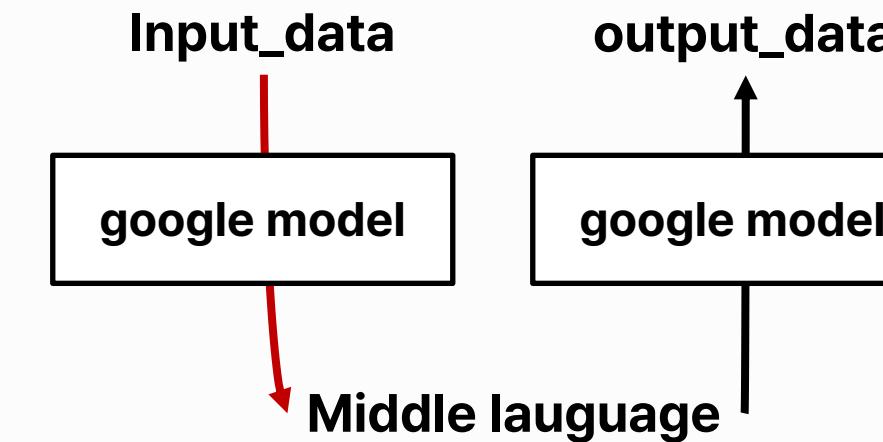
- Back Translation evaluation result

1) Mbart\_Back\_translation



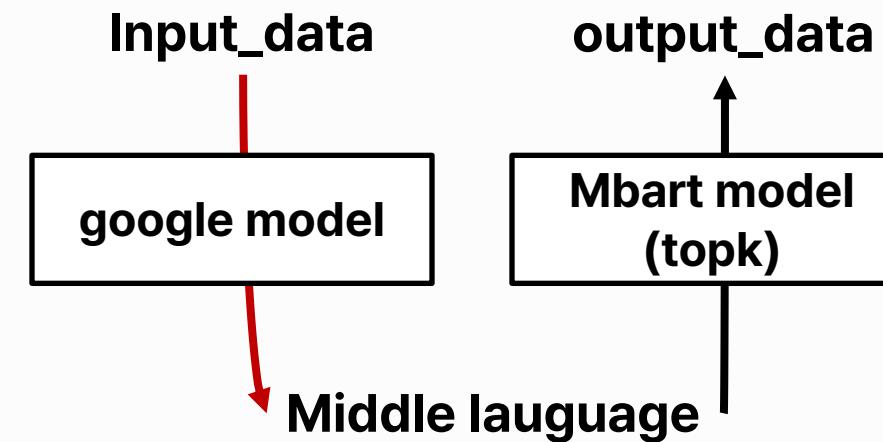
BLEU  
2.2771710719511318

2) google\_Back\_translation



BLEU  
3.8329663759924886

3) collaboration\_Back\_translation



BLEU  
2.9631089451969896

## 06

## 새로운 시도

## 데이터 증강

## ● 모델 주요 함수:

- Mbart 모델을 기반으로 정번역 수행하는 부분



- Mbart 모델을 기반으로 역번역 수행하는 부분  
(top\_k로 샘플링)



- Mbart 모델을 기반으로 정번역 및 역번역을 수행하는 함수

```
def single_translate(input_data, s_lang, t_lang, top_k_val, type_trans = 'forward'):
    original_list = []
    return_list = []
    class_list = []
    MBart_tokenizer.src_lang = s_lang
    if type_trans == 'forward':
        single_data = input_data['conversation']
        for i, row_input_data in enumerate(single_data):
            retrun_text_list = []
            text_sen = row_input_data
            text_list = text_sen.split('\n')
            for text in text_list:
                encoded_text = MBart_tokenizer(text, return_tensors="pt")
                generated_tokens = MBart_model.generate(**encoded_text, forced_bos_token_id = MBart_tokenizer.lang_code_to_id [t_lang])
                translation = MBart_tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)
                retrun_text_list.extend(translation)
            retrun_text = '\n'.join(retrun_text_list)
            class_list.append(input_data['class'][0])
            original_list.append(text_sen)
            return_list.append(retrun_text)
            print(i)
    else:
        for i, text_sen in enumerate(input_data['conversation']):
            retrun_text_list = []
            text_list = text_sen.split('\n')
            for text in text_list:
                encoded_text = MBart_tokenizer(text, return_tensors="pt")
                generated_tokens = MBart_model.generate(**encoded_text, forced_bos_token_id = MBart_tokenizer.lang_code_to_id [t_lang], do_sample=True, top_k=top_k_val)
                translation = MBart_tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)
                retrun_text_list.extend(translation)
            retrun_text = '\n'.join(retrun_text_list)
            class_list.append(input_data['class'][0])
            original_list.append(text_sen)
            return_list.append(retrun_text)
            print(i)

    original_data = pd.DataFrame({'class' : class_list, 'conversation' : original_list})
    translated_data = pd.DataFrame({'class' : class_list, 'conversation' : return_list})

    return original_data, translated_data
```

## 06

## 새로운 시도

데이터 증강

## ● 모델 주요 함수:

- googletrans 모델을 기반으로 정번역 수행



- googletrans 모델을 기반으로 역번역 수행



- googletrans 모델을 기반으로 정번역 및 역번역을 수행하는 함수

```
def single_translate_google(input_data, s_lang, t_lang, type_trans = 'forward'):
    original_list = []
    return_list = []
    class_list = []
    if type_trans == 'forward':
        single_data = input_data['conversation']
        for i, row_input_data in enumerate(single_data):
            retrun_text_list = []
            text_sen = row_input_data
            text_list = text_sen.split('\n')
            for text in text_list:
                translation = translator.translate(text, src=s_lang, dest=t_lang).text
                retrun_text_list.append(translation)
            retrun_text = '\n'.join(retrun_text_list)
            class_list.append(input_data['class'][0])
            original_list.append(text_sen)
            return_list.append(retrun_text)
            print(i)
    else:
        for i, text_sen in enumerate(input_data['conversation']):
            retrun_text_list = []
            text_list = text_sen.split('\n')
            for text in text_list:
                translation = translator.translate(text, src=s_lang, dest=t_lang).text
                retrun_text_list.append(translation)
            retrun_text = '\n'.join(retrun_text_list)
            class_list.append(input_data['class'][0])
            original_list.append(text_sen)
            return_list.append(retrun_text)
            print(i)

    original_data = pd.DataFrame({'class' : class_list, 'conversation' : original_list})
    translated_data = pd.DataFrame({'class' : class_list, 'conversation' : return_list})

    return original_data, translated_data
```

# 06 | 새로운 시도

## 데이터 증강

- 모델 주요 함수:

- Mbart\_Back\_translation
- Mbart 모델로 정번역 및 역번역 모두 수행



```
def back_translate(input_data, s_lang, t_lang, top_k_val):  
    original_data1, translated_data = sigle_translate(input_data, s_lang, t_lang, 'forward')  
    original_data2, return_data = sigle_translate(translated_data, t_lang, s_lang, top_k_val, 'backward')  
    return [original_data1, return_data]
```

- google\_Back\_translation
- googletrans 모델로 정번역 및 역번역 모두 수행



```
def back_translate_google(input_data, s_lang, t_lang):  
    original_data1, translated_data = sigle_translate_google(input_data, s_lang, t_lang, 'forward')  
    original_data2, return_data = sigle_translate_google(translated_data, t_lang, s_lang, 'backward')  
    return [original_data1, return_data]
```

- collaboration\_Back\_translation
- googletrans 모델로 정번역, Mbart 모델로 역번역을 수행



```
def back_translate_colabo(input_data, s_lang_g, t_lang_g, s_lang, t_lang, top_k_val):  
    original_data1, translated_data = sigle_translate_google(input_data, s_lang_g, t_lang_g, 'forward')  
    original_data2, return_data = sigle_translate(translated_data, t_lang, s_lang, top_k_val, 'backward')  
    return [original_data1, return_data]
```

## 06

## 새로운 시도

## 최종 예측 모델

- 기반 모델 : <https://huggingface.co/skt/kogpt2-base-v2>
- 모델 Summary :

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[None, 300]	0
tfgpt2model (TFGPT2Model)	TFBaseModelOutputWithPast	125164032
tf.__operators__.getitem_3	(None, 768)	0
dropout (Dropout)	(None, 768)	0
dense_1 (Dense)	(None, 5)	3845
<hr/>		
Total params: 125,167,877		
Trainable params: 125,167,877		
Non-trainable params: 0		

# 06 | 새로운 시도

## 최종 예측 모델

- Dense 레이어에 입력되기 전 0.2%  
Drop out 레이어 추가

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[None, 300]	0
tfgpt_t2model (TFGPT2Model)	TFBaseModelOutputWithPast	125164032
tf.__operators__.getitem_3 (	(None, 768)	0
dropout (Dropout)	(None, 768)	0
dense_1 (Dense)	(None, 5)	3845
Total params:	125,167,877	
Trainable params:	125,167,877	
Non-trainable params:	0	

## 모델 구조상 차이점

## Base GPT

- GPT 모델 출력 이후 바로 출력  
레이어를 통해 결과 도출

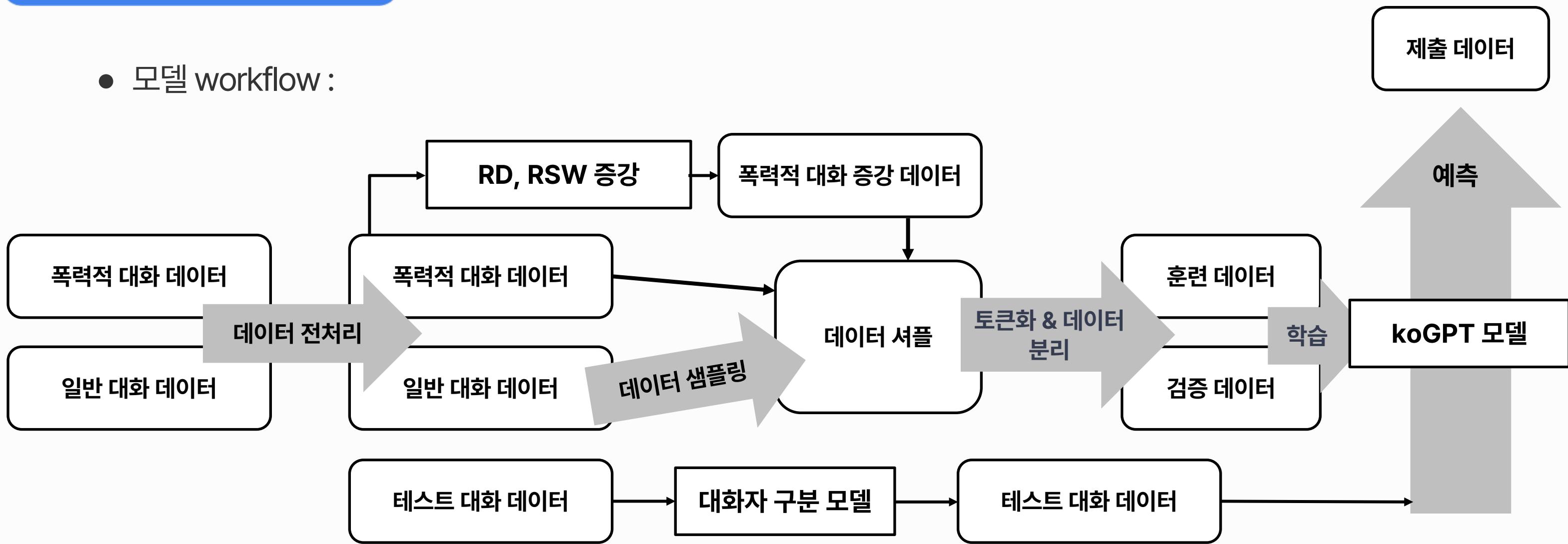
Model: "model\_2"

Layer (type)	Output Shape	Param #
input_6 (InputLayer)	[None, 300]	0
tfgpt_t2model (TFGPT2Model)	TFBaseModelOutputWithPast	125164032
tf.__operators__.getitem_4 (	(None, 768)	0
dense_2 (Dense)	(None, 5)	3845
Total params:	125,167,877	
Trainable params:	125,167,877	
Non-trainable params:	0	

# 06 | 새로운 시도

## 최종 예측 모델

- 모델 workflow :



## 06

## 새로운 시도

## 최종 예측 모델

- 토큰화 & 데이터 분리:

줄바꿈 문자로 사용하여  
화자와 청자의 문장을 분리하여 학습

```
def convert_examples_to_features2(sent_list, max_seq_len, tokenizer):
    input_ids = []

    for conversation in tqdm(sent_list, total=len(sent_list)):
        bos_token = [tokenizer.bos_token]
        eos_token = [tokenizer.eos_token]

        sentences = conversation.split("\n")

        tokens = []

        for i, sentence in enumerate(sentences):
            sentence_tokens = bos_token + tokenizer.tokenize(sentence) + eos_token
            tokens += sentence_tokens

        # 정수 인코딩
        input_id = tokenizer.convert_tokens_to_ids(tokens)
        # 패딩 추가
        input_id = pad_sequences([input_id], maxlen=max_seq_len, value=tokenizer.pad_token_id, padding='post')[0]

        # 길이 확인
        assert len(input_id) == max_seq_len, f"Error with input length {len(input_id)} vs {max_seq_len}"

        # 결과 저장
        input_ids.append(input_id)

    input_ids = np.array(input_ids, dtype=int)

return input_ids
```

# 06 | 새로운 시도

## 최종 예측 모델

- 모델 생성:

생성 모델을 다중 분류 모델로  
바꾸기 위한 클래스 활용

```
# GPT로 다중 분류 모델 만들기

class TFGPT2ForSequenceClassification(tf.keras.Model):
    def __init__(self, model_name, num_labels, dropout_rate=0.2):
        super(TFGPT2ForSequenceClassification, self).__init__()
        self.gpt = TFGPT2Model.from_pretrained(model_name, from_pt=True)
        self.dropout = tf.keras.layers.Dropout(dropout_rate, name='dropout')
        self.classifier = tf.keras.layers.Dense(num_labels,
                                                kernel_initializer=tf.keras.initializers.TruncatedNormal(0.02),
                                                activation='softmax',
                                                name='classifier')

    def call(self, inputs, training=False):
        outputs = self.gpt(input_ids=inputs)
        cls_token = outputs[0][:, -1]
        cls_token = self.dropout(cls_token, training=training)
        prediction = self.classifier(cls_token)

    return prediction
```

# 06 | 새로운 시도

## 최종 예측 모델

### ● 모델 결과 :

Checkpoint accuracy가 잘못  
입력되어 최적 가중치 저장이 안되었음

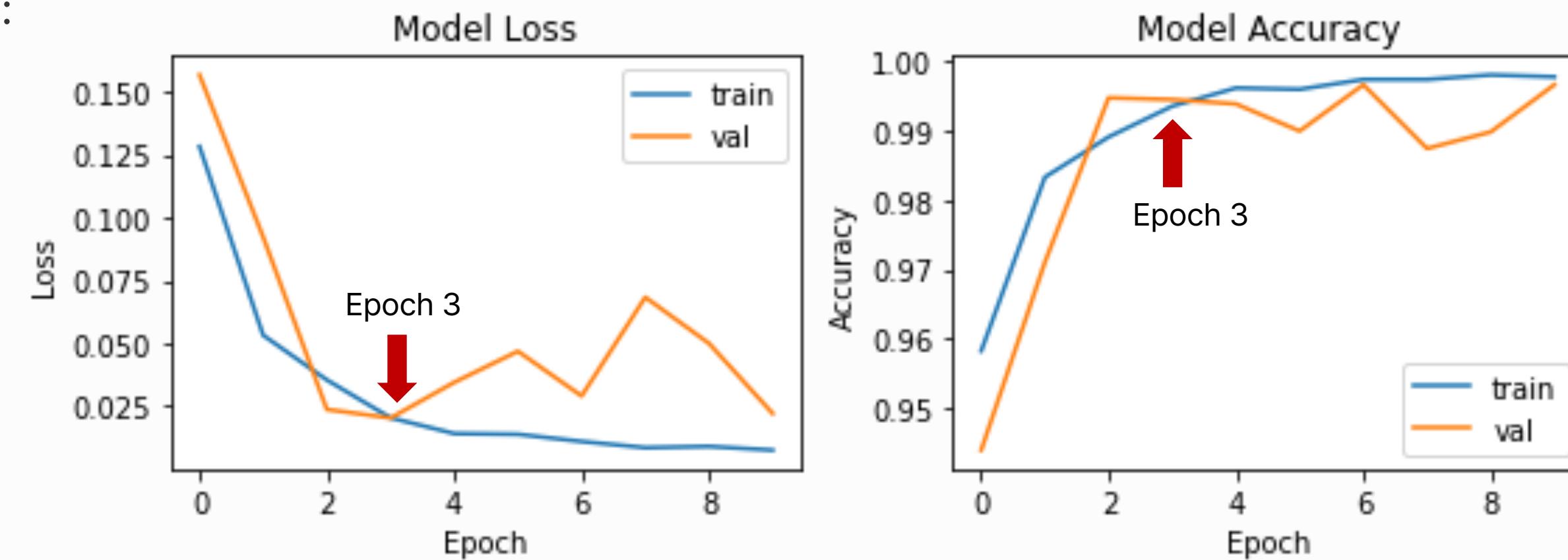
```
Epoch 1/10
1986/1986 [=====] - 2401s 1s/step - loss: 0.1284 - sparse_categorical_accuracy: 0.9583 - val_loss: 0.1569 - val_s
parse_categorical_accuracy: 0.9440
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 2/10
1986/1986 [=====] - 2397s 1s/step - loss: 0.0533 - sparse_categorical_accuracy: 0.9833 - val_loss: 0.0923 - val_s
parse_categorical_accuracy: 0.9710
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 3/10
1986/1986 [=====] - 2398s 1s/step - loss: 0.0355 - sparse_categorical_accuracy: 0.9891 - val_loss: 0.0238 - val_s
parse_categorical_accuracy: 0.9947
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 4/10
1986/1986 [=====] - 2397s 1s/step - loss: 0.0204 - sparse_categorical_accuracy: 0.9935 - val_loss: 0.0204 - val_s
parse_categorical_accuracy: 0.9945
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 5/10
1986/1986 [=====] - 2396s 1s/step - loss: 0.0142 - sparse_categorical_accuracy: 0.9961 - val_loss: 0.0345 - val_s
parse_categorical_accuracy: 0.9938
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 6/10
1986/1986 [=====] - 2396s 1s/step - loss: 0.0139 - sparse_categorical_accuracy: 0.9959 - val_loss: 0.0469 - val_s
parse_categorical_accuracy: 0.9899
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 7/10
1986/1986 [=====] - 2397s 1s/step - loss: 0.0110 - sparse_categorical_accuracy: 0.9974 - val_loss: 0.0291 - val_s
parse_categorical_accuracy: 0.9966
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 8/10
1986/1986 [=====] - 2398s 1s/step - loss: 0.0086 - sparse_categorical_accuracy: 0.9974 - val_loss: 0.0684 - val_s
parse_categorical_accuracy: 0.9874
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 9/10
1986/1986 [=====] - 2399s 1s/step - loss: 0.0090 - sparse_categorical_accuracy: 0.9980 - val_loss: 0.0501 - val_s
parse_categorical_accuracy: 0.9898
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
Epoch 10/10
1986/1986 [=====] - 2397s 1s/step - loss: 0.0076 - sparse_categorical_accuracy: 0.9977 - val_loss: 0.0221 - val_s
parse_categorical_accuracy: 0.9966
WARNING:tensorflow:Can save best model only with val_accuracy available, skipping.
```

최적 적합

# 06 | 새로운 시도

## 최종 예측 모델

- 모델 결과:



# 06 | 새로운 시도

최종 예측 모델

- 모델 Submission :

**submission.csv**  
Complete · yengerche · 40m ago

**Score:**  
Public score: 0.68234

UPLOADED FILES

submission.csv (3 KiB)

DESCRIPTION

final\_model

11 / 500

SELECT FOR FINAL SCORE

Select this submission to be scored for your final leaderboard score

## 07

# 프로젝트 회고

## 인터넷에서 수집한 대화와 실생활에서 수집한 대화의 차이 보간

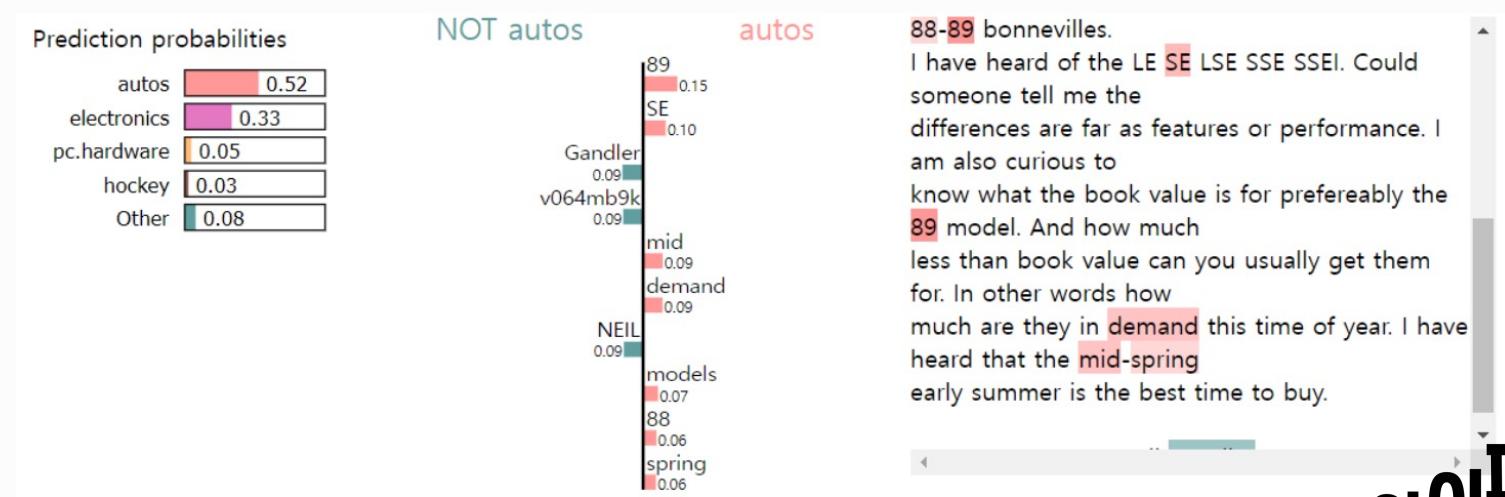
- 말투나 사용하는 단어 등을 통일하거나 처리해 학습을 용이하게 만드는 텍스트 전처리 기법 찾기

## Fine Tuning

- 대화자 구분이 잘 적용될 수 있도록 더 많은 데이터를 이용하여 Fine Tuning
- W&B를 활용해서 모델 Fine Tuning

## 설명가능한 인공지능(XAI) 활용

- 분류 문제의 근거가 되는 텍스트 토큰 확인 가능 (LIME)



07

## 프로젝트 회고

### 다양한 문제 정의 방식 시도

- 폭력적 대화와 일반 대화를 우선 분류하고 이후 폭력적 대화를 유형별로 분류
- 폭력 내용은 가해자의 발화에서 뚜렷하게 드러나므로 가해자 발언을 통해 패턴 학습
  - 폭력 대화를 위주로 대화자 구분 모델 Fine Tuning
  - 일반 대화 분류 시 대화자 구분하지 않고, 폭력 class 분류 시 대화자 구분 사용

### 더 다양한 모델 시도

- 충분한 메모리 확보 후 LLM 기반 고용량 모델 기용 (e.g. Llama, ...)
- Stacking, Voting 등의 앙상블 기법 기용

# Q & A