SC

Search for computer vision

V

DLThon 발표자료

구성원 : 이경규 권영찬 이종민 김상호

목차

- 01 분석배경
- 02 EDA
- 03 Base Model (Bert multilingual)
- 04 Base Model 기반 실험
- 05 Klue Bert Model 기반 실험
- 05 성능 결과표
- 06 성능 향상 시도

분석배경

- 문제정의
- 데이터 수집 방법 출처

문제 정의

DKTC (Dataset of Korean Threatening Conversations)

4개의 클래스로 구성된 한국어 대화 데이터를 활용하여

5개의 클래스로 다중 분류하는 모델 생성하여

한국어 협박, 갈취, 직장 내 괴롭힘, 기타 괴롭힘, 일반 대화의

5가지 대화 유형 Class를 분류

데이터 수집 방법, 출처

DKTC

제공 train data (DKTC)

3950개의 대화

Class : 4개 (협박, 갈취, 직장 내 괴롭힘, 기타 괴롭힘)

일반 대화

추가할 train data (AI Hub)

Kakao, Facebook, Nateon, …

약 1000개의 대화

Class : 1개 (일반 대화)

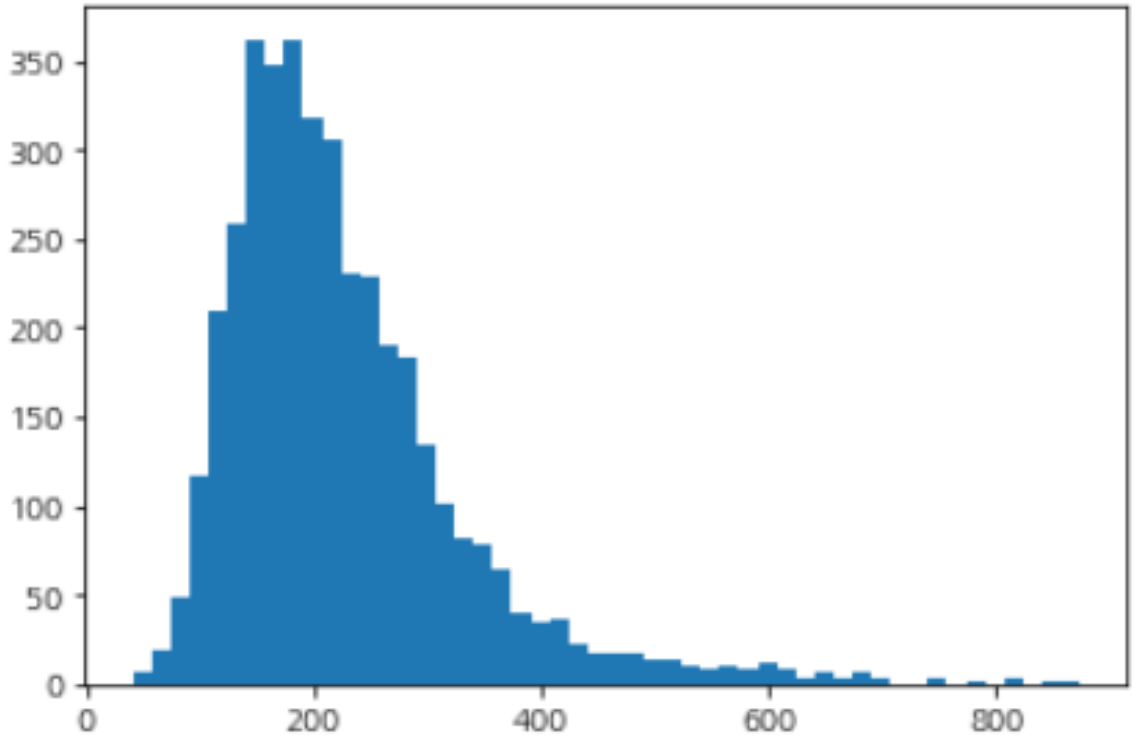
EDA

- 훈련 데이터
- 일반 대화 데이터
- 병합 데이터

EDA

훈련 데이터

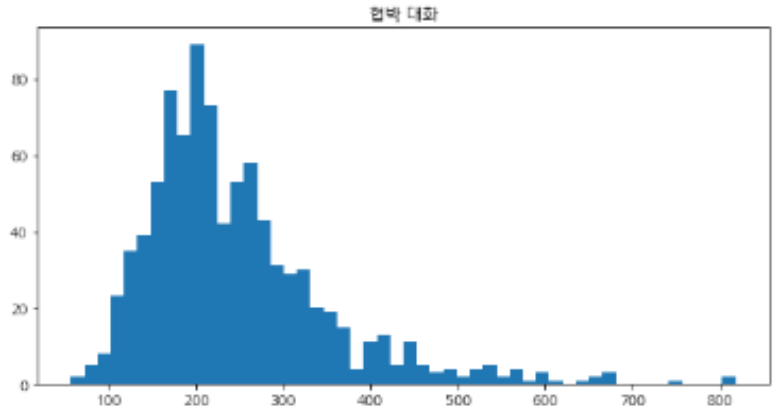
훈련 데이터

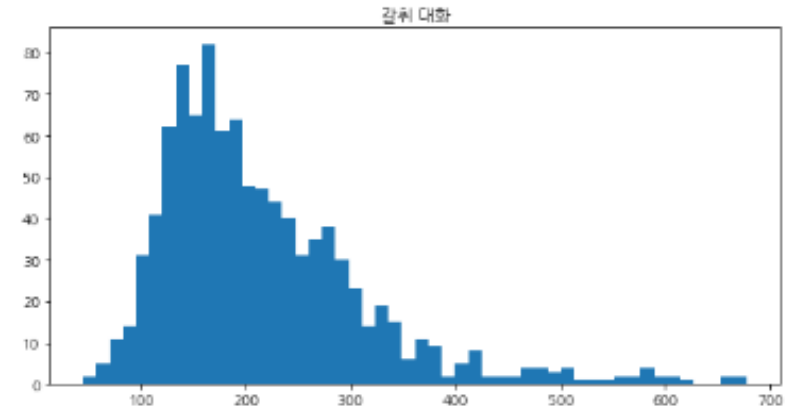


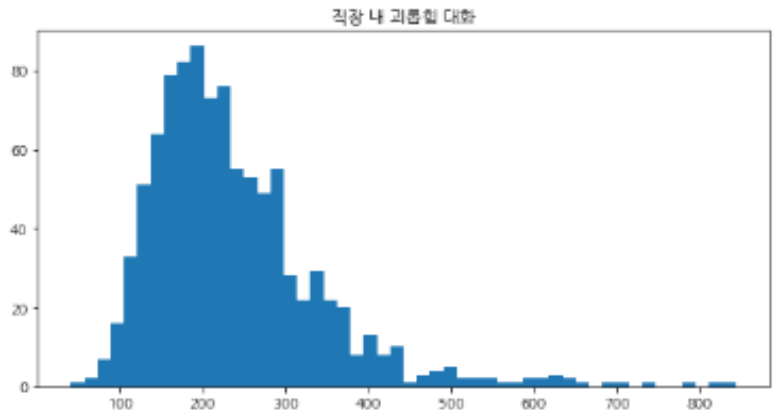

```
max len:  874
total mean len: 227
```

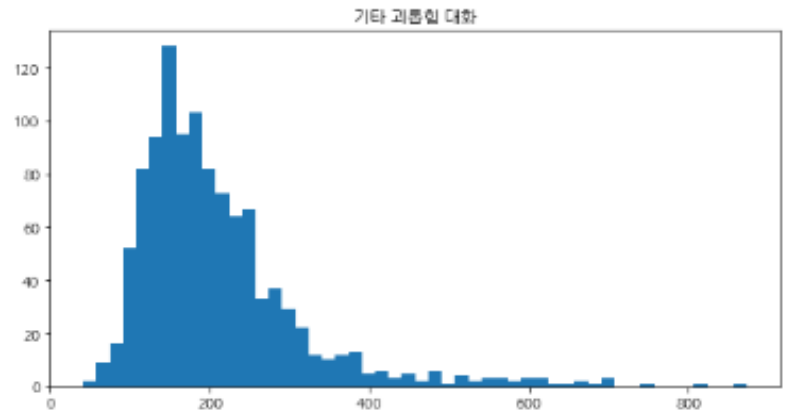
class	conversation
갈취 대화	981
기타 괴롭힘 대화	1094
직장 내 괴롭힘 대화	979
협박 대화	896

훈련 데이터









협박 대화 mean len: 246
기타 괴롭힘 대화 mean len: 210
갈취 대화 mean len: 216
직장 내 괴롭힘 대화 mean len: 238

훈련 데이터

'협박 대화'
: 지금 너 스스로를 죽여달라고 애원하는 것인가?
아닙니다. 죄송합니다.
죽을 거면 혼자 죽지 우리까지 사건에 휘말리게 해? 진짜 죽여버리고 싶게.
정말 잘못했습니다.
너가 선택해. 너가 죽을래 네 가족을 죽여줄까.
죄송합니다. 정말 잘못했습니다.
너에게는 선택권이 없어. 선택 못한다면 너와 네 가족까지 모조리 죽여버릴거야.
선택 못하겠습니다. 한번만 도와주세요.
그냥 다 죽여버려야겠군. 이의 없지?
제발 도와주세요.

'갈취 대화'
: 어이 거기
예??
너 말이야 너. 이리 오라고
무슨 일.
너 옷 좋아보인다?
얘 돈 좀 있나봐
아니에요.돈 없어요
뒤져서 나오면 넌 죽는다
오늘 피시방 콜?
콜. 마지막 기회다. 있는거 다 내놔
정말 없어요

훈련 데이터

'직장 내 괴롭힘 대화'
: 나 이틀뒤에 가나다 음식점 예약좀 해줘. 저녁7시로.
가나다 음식점이요.?
응. 남자친구 부모님한테 인사드리려는데 거기가 예약이 좀 힘들어? 그러니까 수진씨가 좀 해줘.
저.팀장님. 저도 웰 말 프로젝트로 정신없어서.죄송하지만.
사회생활 안 해본 티를 너무 내는거 아니야? 프로젝트만 백날 잘하면 뭐해? 윗 상사한테 잘 보이기도 해야지!
하지만 팀 프로젝트라서 이번엔.
말 참 이상하게 하네? 이번엔? 내가 뭐 매일같이 이런 심부름이나 시킨다는거야? 뭐야?!
아닙니다. 제가 말 실수 했습니다. 말씀하신 예약 꼭 해두겠습니다.
이러면 하고도 욕먹는거야! 한번에 네네 알겠습니다 하면 좀 좋아?!
죄송합니다. 알겠습니다.

'기타 괴롭힘 대화'
: 너 되게 귀여운거 알지? 나보다 작은 남자는 첨봤어.
그만해. 니들 놀리는거 재미없어.
지영아 너가 키 160이지? 그럼 재는 160도 안돼는거네?
너 군대도 안가고 좋겠다.
니들이 나 작은데 보태준거 있냐?
난쟁이들도 장가가고하던데. 너도 희망을 가져봐
더이상 하지마라.
그 키크는 수술도 있대잖아? 니네 엄마는 그거 안해주디?
나람 해줬어. 저 키로 어찌살아.
제발 그만 괴롭히라고!

EDA

일반 대화 데이터

- **카카오 대화 데이터**

일반 대화 데이터

GPT 생성 데이터

- 대화의 길이를 원하는 만큼 지정할 수 있어 좋지만, 중복 데이터가 너무 많다

- 사용 X

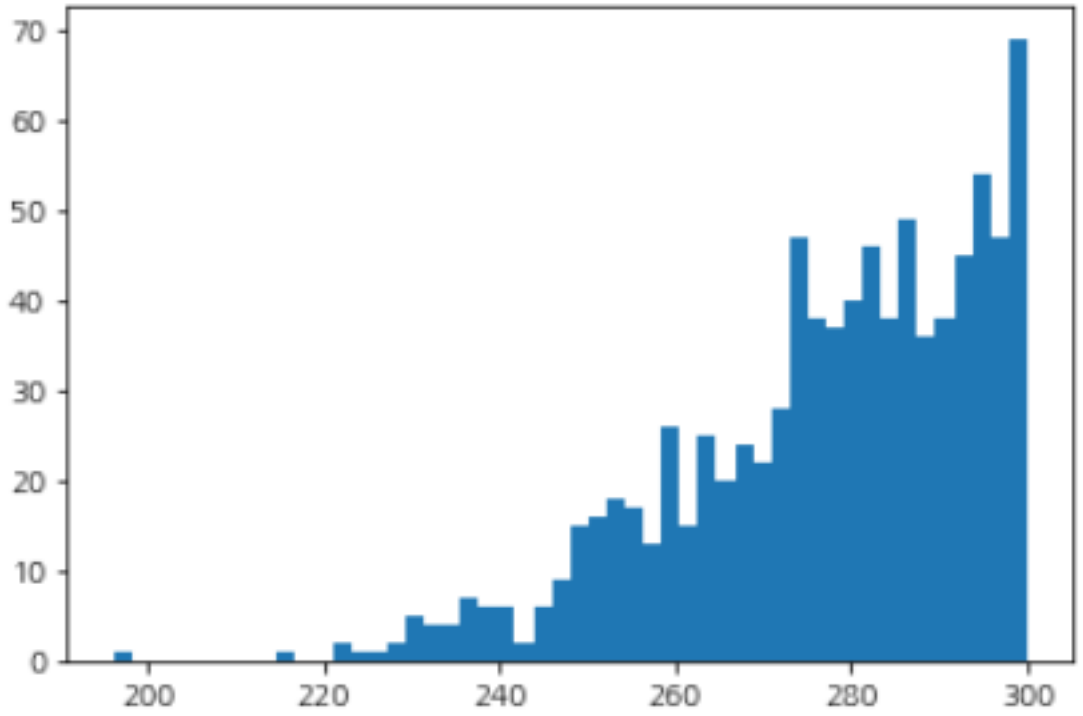
```
중복된 데이터가 conversation 열에 있습니다:
      idx     class                                              conversation
0       1      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
1       2      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
2       3  기타 괴롭힘 대화  너 피아노 좀 쳐봐.\n싫어.\n왜 손가락 없다고 유세 떠는 거야?\n이씨.\n비행...
8       9  기타 괴롭힘 대화  새파랗게 젊은게 어디 여길 앉아있어\n저 임산부에요\n사지 멀쩡한게! 임신이 벼슬이...
14     15      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
...   ...        ...                                                ...
4925  4926  기타 괴롭힘 대화  야 니 왤캐 못생겼냐?\n뭐라그랬냐?\n으 나 보고 말하지마 니 얼굴보면 토나올거 ...
4927  4928      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
4929  4930      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
4930  4931      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
4948  4949      일반 대화  오늘 날씨 어때?\n맑고 따뜻해.\n좋네! 주말에 계획 있어?\n등산 갈 생각이야....
```

일반 대화 데이터

kakao 대화 데이터 사용

```
df['conversation_length'] = df['conversation'].str.len()
filtered_df = df[(df['conversation_length'] >= 50) & (df['conversation_length'] <= 300)]
filtered_df = filtered_df[['idx', 'class', 'conversation']]
filtered_df.to_csv('~/aiffel/dktc/data2/normal_data300.csv', index=False)
```

```
shuffled_df.to_csv('~/aiffel/dktc/data2/train0.csv', index=False)
```



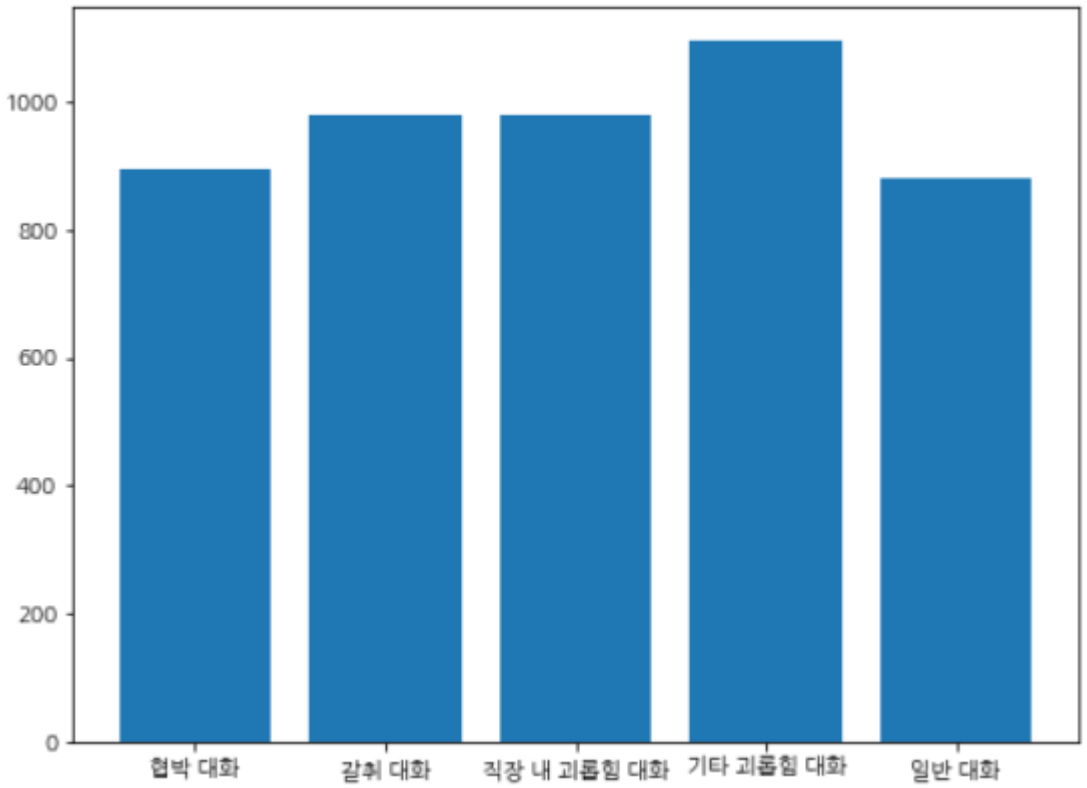
일반 대화 mean len: 277

EDA

병합 데이터

- **train0 : 훈련 데이터 + 카카오 데이터**
- **성능향상 시도 시 더 다양한 일반 대화 데이터를 추가한 데이터 사용 예정 (train4, ...)**

병합 데이터



class	conversation
갈취 대화	981
기타 괴롭힘 대화	1094
직장 내 괴롭힘 대화	979
협박 대화	896

Base Model

- 선정 이유
- 전처리, 모델 파라미터

Base Model

Bert mulitilingual

선정 이유

- 잘 알려진 대화 판별용 사전학습모델
- 다양한 언어
- tensorflow 환경 지원

Base Model

전처리

```
def preprocess_sentence(sentence):
    sentence = re.sub(r'([^a-zA-Zㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    sentence = sentence.strip()
    return sentence
```

하이퍼 파라미터

```
BATCH_SIZE = 16
lr = 5e-5
EPOCH = 10
```

콜백 설정

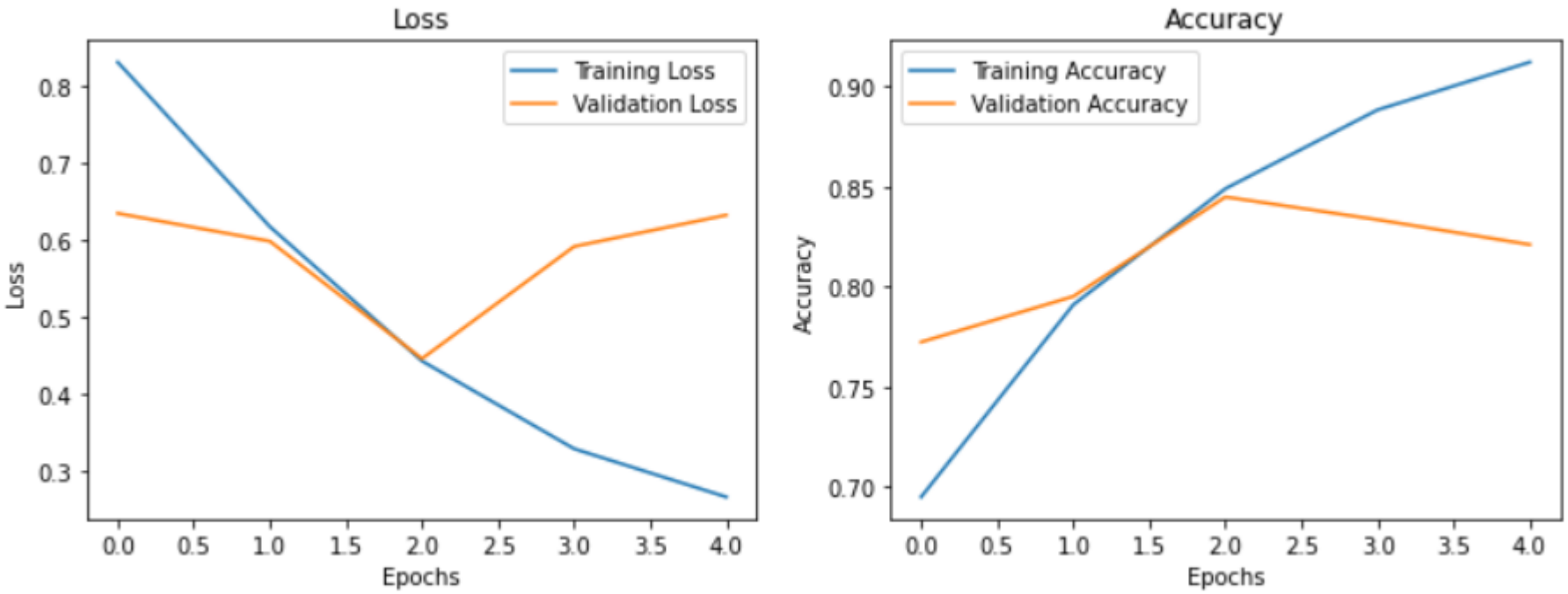
```
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=2,
    restore_best_weights=True
)

checkpoint = ModelCheckpoint(
    filepath='best_model_weights.h5',
    monitor='val_loss',
    save_best_only=True,
    save_weights_only=True,
    mode='min',
    verbose=1
)
```

Base Model

val loss , val acc

```
# 모델 평가
evaluation = model.evaluate(val_dataset)
print("평가 결과:", evaluation)
```

```
61/61 [==============================] - 21s 346ms/step - loss: 0.4453 - accuracy: 0.8447
평가 결과: [0.44534143805503845, 0.8447204828262329]
```



Base Model

F1-socre

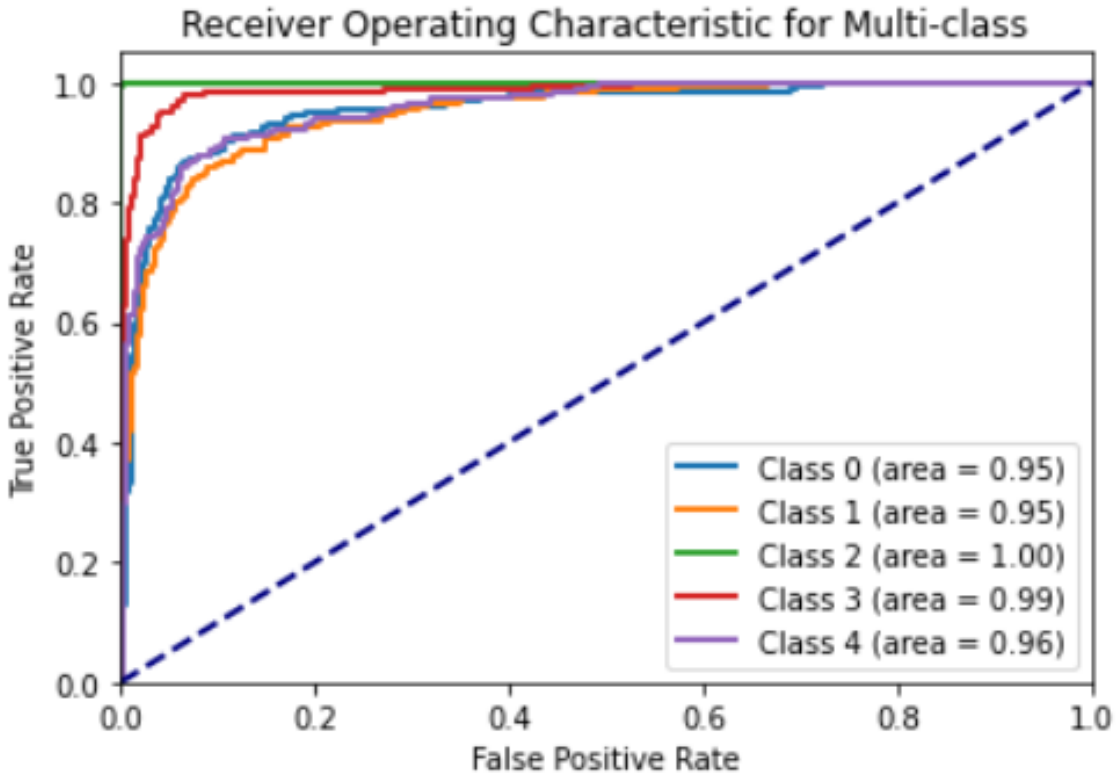
Real Accuracy: 0.8323

	precision	recall	f1-score	support
Class 0	0.75	0.83	0.79	211
Class 1	0.75	0.69	0.72	208
Class 2	0.99	0.99	0.99	169
Class 3	0.85	0.93	0.89	195
Class 4	0.85	0.75	0.80	183
accuracy			0.83	966
macro avg	0.84	0.84	0.84	966
weighted avg	0.83	0.83	0.83	966

Weighted F1 Score (based on real predictions): 0.8308

Base Model

AUC-ROC



Class2 = 일반 대화

100퍼센트 구분한다는게 이상하게 느껴지지만

다른 대화들과의 차이가

도드라지기 때문이라고 추측

Class3 = 직장 내 괴롭힘 대화

다름으로 구분을 잘하는데,

직장에서만 쓰는 용어들이

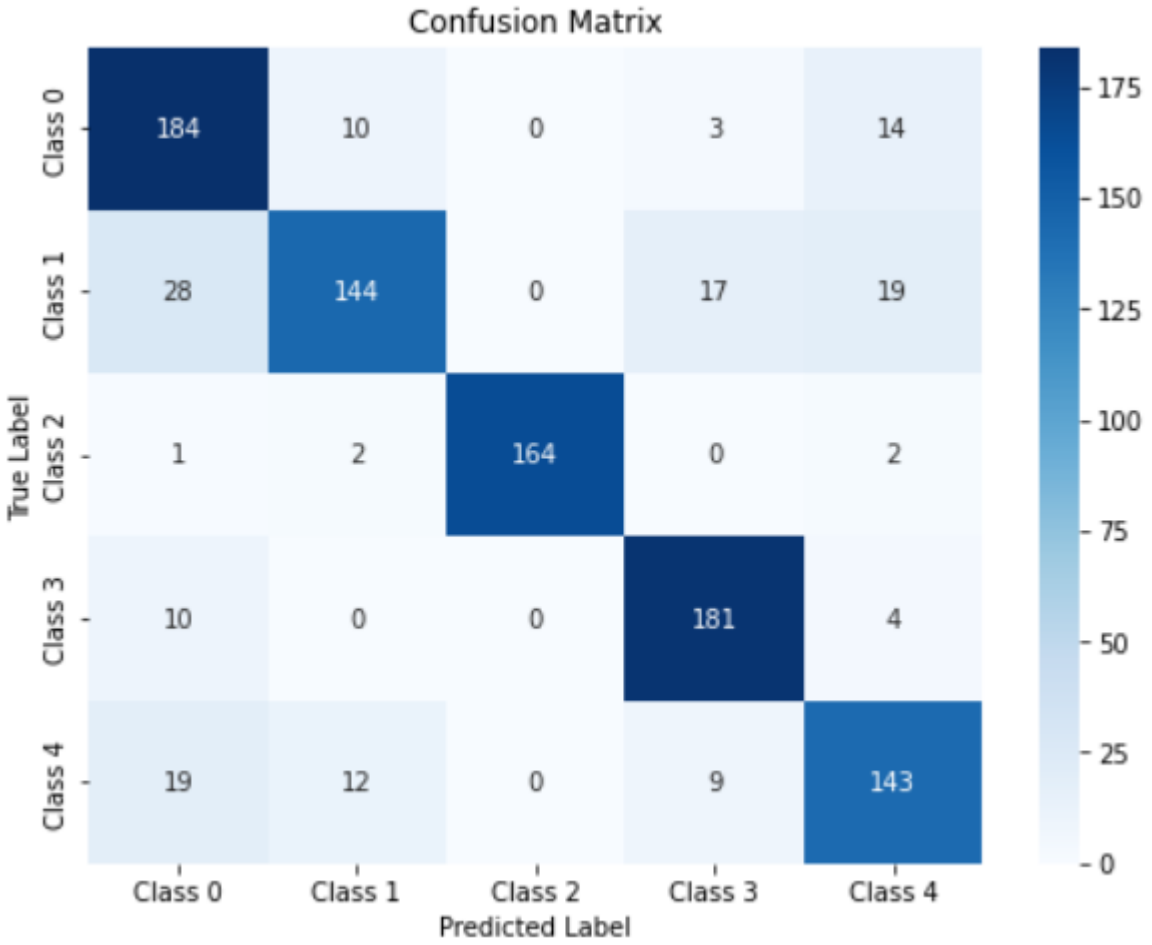
있기 때문이라고 추측

```
class_mapping = {class_name: encoder.transform([class_name])[0] for class_name in CLASS_NAMES}
print("Class mapping:", class_mapping)
```

```
Class mapping: {'협박 대화': 4, '갈취 대화': 0, '직장 내 괴롭힘 대화': 3, '기타 괴롭힘 대화': 1, '일반 대화': 2}
```

Base Model

Confusion Matrix



Base 모델 기반 다양한 실험

- **전처리 변경1**
- **전처리 변경2**
- **데이터 증강**

Base Model 기반 실험

Base Model

```
def preprocess_sentence(sentence):
    sentence = re.sub(r'([^a-zA-Z ㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    sentence = sentence.strip()
    return sentence
```

```
Real Accuracy: 0.8323
              precision    recall  f1-score   support

     Class 0       0.75      0.83      0.79       211
     Class 1       0.75      0.69      0.72       208
     Class 2       0.99      0.99      0.99       169
     Class 3       0.85      0.93      0.89       195
     Class 4       0.85      0.75      0.80       183

    accuracy                           0.83       966
   macro avg       0.84      0.84      0.84       966
weighted avg       0.83      0.83      0.83       966


Weighted F1 Score (based on real predictions): 0.8308
```

전처리 변경1

```
def preprocess_sentence(sentence):
    # synolp
    emoticon_normalize(sentence)
    repeat_normalize(sentence)
    sentence = re.sub(r'[^₩w₩s]', '', sentence)
    # base preprocess
    sentence = re.sub(r'([^a-zA-Z ㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    # 엔터 구분 (₩n)
    entence = sentence.replace("₩n", " ")
    sentence = sentence.strip()
    return sentence
```

```
Real Accuracy: 0.7174
              precision    recall  f1-score   support

     Class 0       0.70      0.83      0.76       211
     Class 1       0.55      0.59      0.57       208
     Class 2       0.91      0.98      0.95       169
     Class 3       0.84      0.86      0.85       195
     Class 4       0.54      0.33      0.41       183

    accuracy                           0.72       966
   macro avg       0.71      0.72      0.71       966
weighted avg       0.70      0.72      0.70       966


Weighted F1 Score (based on real predictions): 0.7047
```

Base Model 기반 실험

전처리 변경1

```python
def preprocess_sentence(sentence):
    # synolp
    emoticon_normalize(sentence)
    repeat_normalize(sentence)
    sentence = re.sub(r'[^₩w₩s]', '', sentence)
    # base preprocess
    sentence = re.sub(r'([^a-zA-Zㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    # 엔터 구분 (₩n)
    entence = sentence.replace("₩n", " ")
    sentence = sentence.strip()
    return sentence
```

Real Accuracy: 0.7174

	precision	recall	f1-score	support
Class 0	0.70	0.83	0.76	211
Class 1	0.55	0.59	0.57	208
Class 2	0.91	0.98	0.95	169
Class 3	0.84	0.86	0.85	195
Class 4	0.54	0.33	0.41	183
accuracy			0.72	966
macro avg	0.71	0.72	0.71	966
weighted avg	0.70	0.72	0.70	966

Weighted F1 Score (based on real predictions): 0.7047

전처리 변경2

```python
def preprocess_sentence(sentence):
    # synolp
    emoticon_normalize(sentence)
    repeat_normalize(sentence)
    # base preprocess
    sentence = re.sub(r'([^a-zA-Zㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    # 엔터 구분 (₩n)
    sentence = sentence.replace('₩n', ' <EOL> ')
    sentence = sentence.strip()
    return sentence
```

Real Accuracy: 0.8530

	precision	recall	f1-score	support
Class 0	0.80	0.85	0.82	211
Class 1	0.79	0.75	0.77	208
Class 2	1.00	0.96	0.98	169
Class 3	0.85	0.92	0.89	195
Class 4	0.86	0.80	0.83	183
accuracy			0.85	966
macro avg	0.86	0.86	0.86	966
weighted avg	0.85	0.85	0.85	966

Weighted F1 Score (based on real predictions): 0.8526

Base Model 기반 실험

데이터 증강

```python
def random_deletion(words, p=0.3):
    if len(words) == 1:
        return words

    new_words = []
    for word in words:
        r = random.uniform(0, 1)
        if r > p:
            new_words.append(word)

    if len(new_words) == 0:
        rand_int = random.randint(0, len(words) - 1)
        return [words[rand_int]]

    return "".join(new_words)

def swap_word(new_words):
    random_idx_1 = random.randint(0, len(new_words) - 1)
    random_idx_2 = random_idx_1
    counter = 0

    while random_idx_2 == random_idx_1:
        random_idx_2 = random.randint(0, len(new_words) - 1)
        counter += 1
        if counter > 3:
            return new_words

    new_words[random_idx_1], new_words[random_idx_2] = (
        new_words[random_idx_2],
        new_words[random_idx_1],
    )
    return new_words
```

```python
def random_swap(words, n=3):
    new_words = words.copy()
    for _ in range(n):
        new_words = swap_word(new_words)

    return new_words


print("before data augmentation: ", len(train_sentences))

train_splted = pd.DataFrame({ "sentence": train_sentences, "class": train_labels })

# random deletion
train_splted_rd = train_splted.copy()
train_splted_rd["sentence"] = train_splted_rd["sentence"].apply(random_deletion)

# random swap
train_splted_rs = train_splted.copy()

# with data augmentation
train_concated = pd.concat([train_splted , train_splted_rd , train_splted_rs])

print("after data augmentation: ", len(train_concated))

train_concated
```

```
before data augmentation:  3864
after data augmentation:  11592
```

Base Model 기반 실험

데이터 증강

```
# 모델 평가
evaluation = model.evaluate(val_dataset)
print("평가 결과:", evaluation)
```

```
61/61 [==============================] - 21s 344ms/step - loss: 0.3825 - accuracy: 0.8861
평가 결과: [0.38247063755989075, 0.8861283659934998]
```

```
Real Accuracy: 0.8861
              precision    recall  f1-score   support

     Class 0       0.85      0.84      0.85       196
     Class 1       0.79      0.87      0.83       219
     Class 2       1.00      0.99      0.99       176
     Class 3       0.96      0.92      0.94       196
     Class 4       0.86      0.82      0.84       179

    accuracy                           0.89       966
   macro avg       0.89      0.89      0.89       966
weighted avg       0.89      0.89      0.89       966


Weighted F1 Score (based on real predictions): 0.8869
```

Base Model 기반 실험

비교

Base model

Real Accuracy: 0.8323

	precision	recall	f1-score	support
Class 0	0.75	0.83	0.79	211
Class 1	0.75	0.69	0.72	208
Class 2	0.99	0.99	0.99	169
Class 3	0.85	0.93	0.89	195
Class 4	0.85	0.75	0.80	183
accuracy			0.83	966
macro avg	0.84	0.84	0.84	966
weighted avg	0.83	0.83	0.83	966

Weighted F1 Score (based on real predictions): 0.8308

전처리 변경1

Real Accuracy: 0.7174

	precision	recall	f1-score	support
Class 0	0.70	0.83	0.76	211
Class 1	0.55	0.59	0.57	208
Class 2	0.91	0.98	0.95	169
Class 3	0.84	0.86	0.85	195
Class 4	0.54	0.33	0.41	183
accuracy			0.72	966
macro avg	0.71	0.72	0.71	966
weighted avg	0.70	0.72	0.70	966

Weighted F1 Score (based on real predictions): 0.7047

전처리 변경2

Real Accuracy: 0.8530

	precision	recall	f1-score	support
Class 0	0.80	0.85	0.82	211
Class 1	0.79	0.75	0.77	208
Class 2	1.00	0.96	0.98	169
Class 3	0.85	0.92	0.89	195
Class 4	0.86	0.80	0.83	183
accuracy			0.85	966
macro avg	0.86	0.86	0.86	966
weighted avg	0.85	0.85	0.85	966

Weighted F1 Score (based on real predictions): 0.8526

데이터 증강

Real Accuracy: 0.8861

	precision	recall	f1-score	support
Class 0	0.85	0.84	0.85	196
Class 1	0.79	0.87	0.83	219
Class 2	1.00	0.99	0.99	176
Class 3	0.96	0.92	0.94	196
Class 4	0.86	0.82	0.84	179
accuracy			0.89	966
macro avg	0.89	0.89	0.89	966
weighted avg	0.89	0.89	0.89	966

Weighted F1 Score (based on real predictions): 0.8869

Klue Bert 모델 기반 실험

- **Klue bert 모델**
- **데이터 증강**
- **데이터 증강 + 전처리 변경**

Klue Bert Model 기반 실험

Klue Bert Model

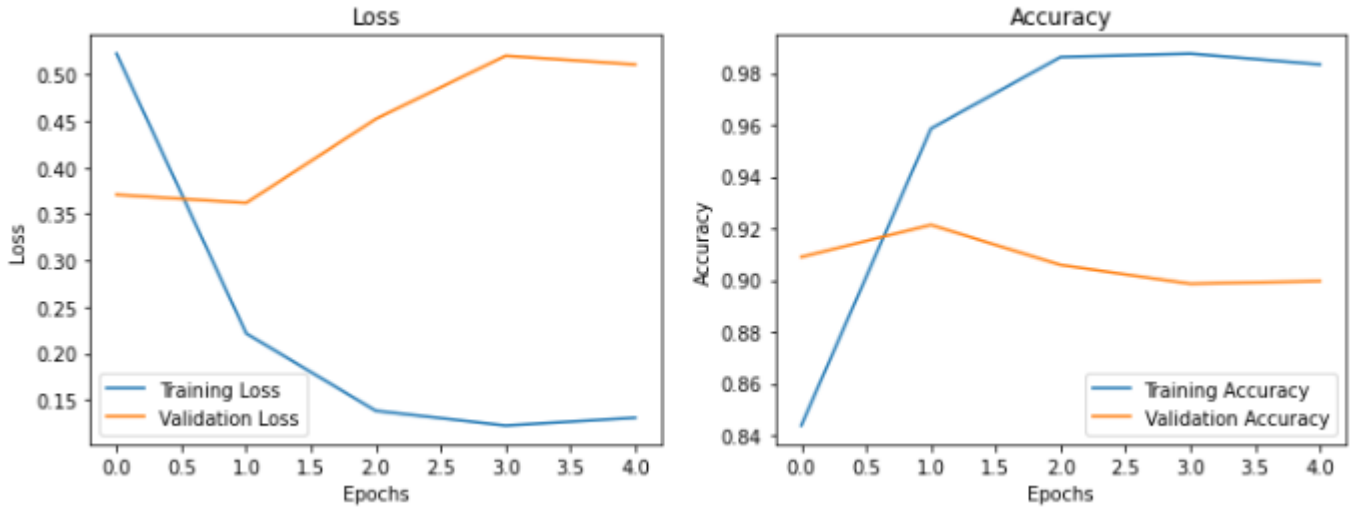
```
class TFBertForMultiClassClassification(tf.keras.Model):
    def __init__(self, model_name, num_classes, dropout_rate=0.1):
        super(TFBertForMultiClassClassification, self).__init__()
        self.bert = TFBertModel.from_pretrained(model_name, from_pt=True)
        self.dropout = Dropout(dropout_rate)
        self.classifier = tf.keras.layers.Dense(num_classes,
                                                kernel_initializer=tf.keras.initializers.TruncatedNormal(0.02),
                                                kernel_regularizer=l2(0.01),
                                                activation='softmax',
                                                name='classifier')

    def call(self, inputs):
        input_ids, attention_mask, token_type_ids = inputs
        outputs = self.bert(input_ids=input_ids,
                            attention_mask=attention_mask,
                            token_type_ids=token_type_ids)
        cls_token = outputs[1]
        dropped = self.dropout(cls_token)
        prediction = self.classifier(dropped)
        return prediction
```

- Klue (Korean Language Understanding Evaluation) 프로젝트에서 제공하는 bert 모델
- 한국어 NLP에서 좋은 성능을 보이는 모델
- Pytorch 기반 모델

Klue Bert Model 기반 실험

Klue Bert Model

```
def preprocess_sentence(sentence):
    emoticon_normalize(sentence)
    repeat_normalize(sentence)
    sentence = re.sub(r'([^a-zA-Z ㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    sentence = sentence.replace("₩n", " ")
    sentence = sentence.strip()
    return sentence
```



Klue Bert Model 기반 실험

F1-socre

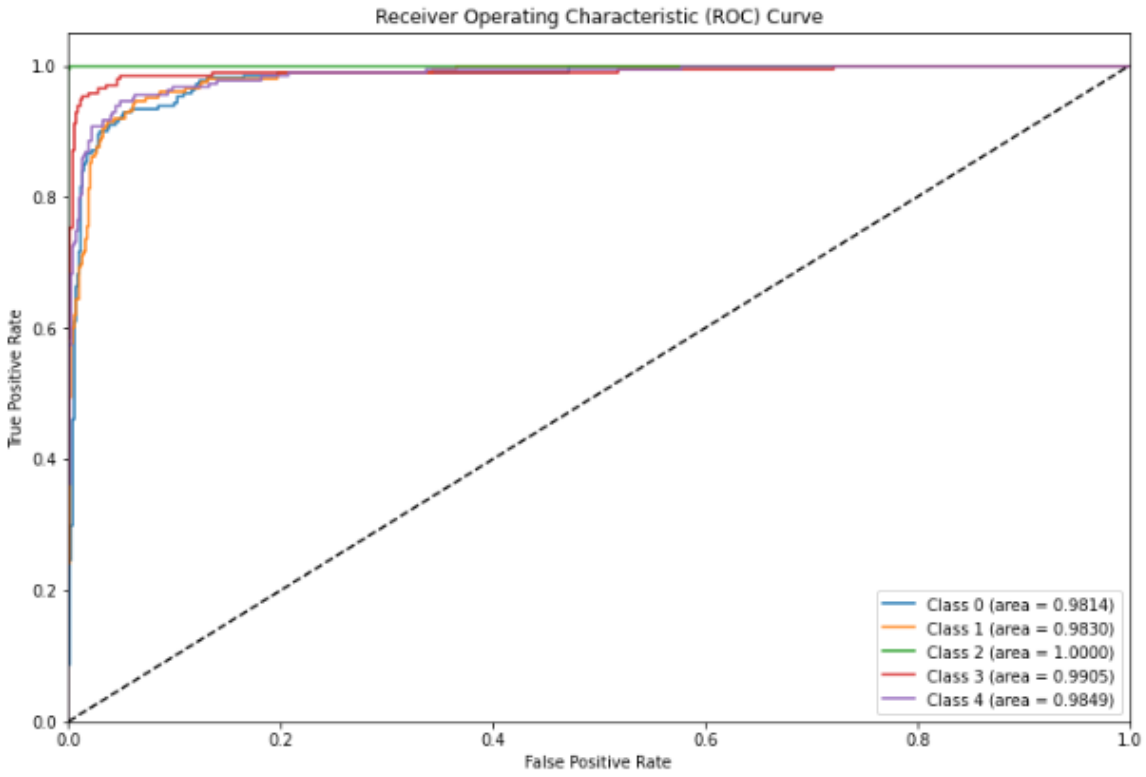
Real Accuracy: 0.9213

	precision	recall	f1-score	support
Class 0	0.89	0.90	0.90	211
Class 1	0.91	0.86	0.88	208
Class 2	0.99	1.00	1.00	169
Class 3	0.94	0.96	0.95	195
Class 4	0.88	0.91	0.89	183
accuracy			0.92	966
macro avg	0.92	0.92	0.92	966
weighted avg	0.92	0.92	0.92	966

Weighted F1 Score (based on real predictions): 0.9211

Klue Bert Model 기반 실험

AUC-ROC

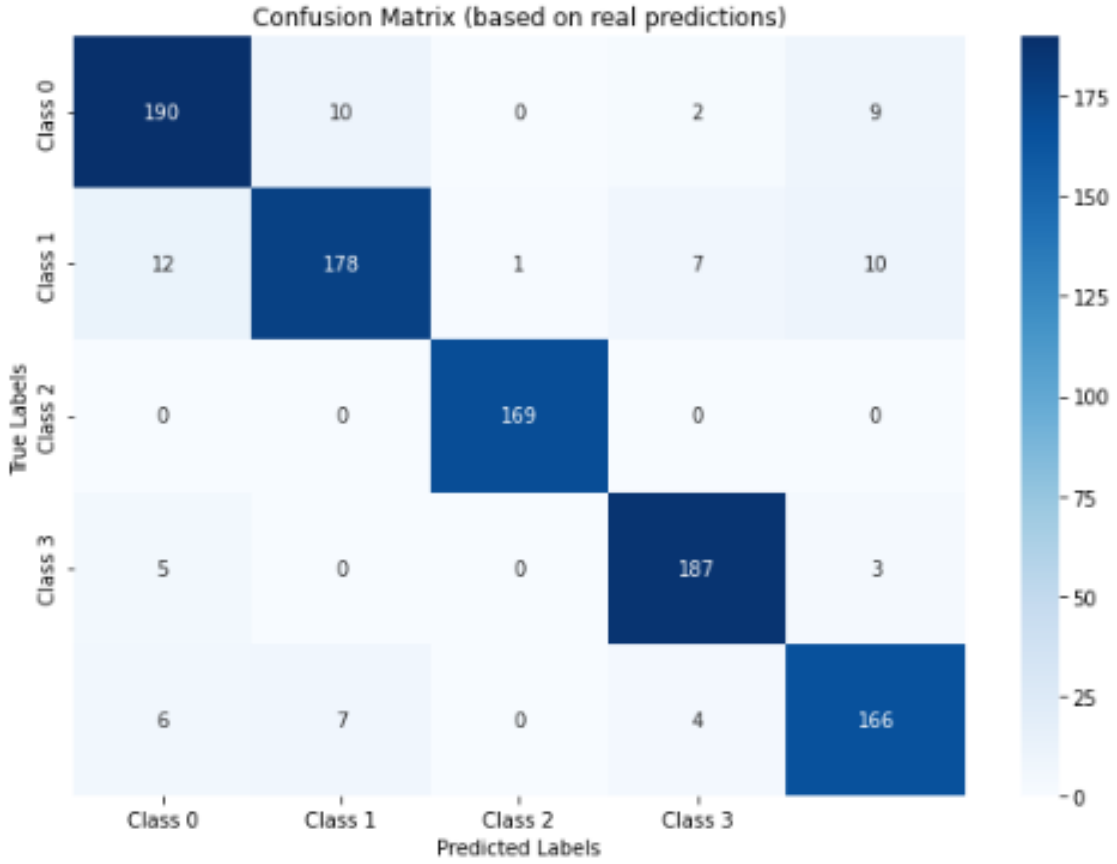



```
class_mapping = {class_name: encoder.transform([class_name])[0] for class_name in CLASS_NAMES}
print("Class mapping:", class_mapping)
```

```
Class mapping: {'협박 대화': 4, '갈취 대화': 0, '직장 내 괴롭힘 대화': 3, '기타 괴롭힘 대화': 1, '일반 대화': 2}
```

Klue Bert Model 기반 실험

Confusion Matrix



Klue Bert Model 기반 실험

데이터 증강+전처리

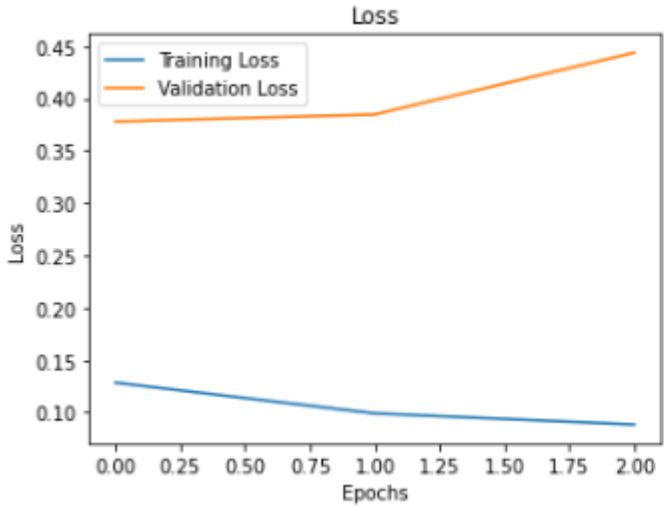
시행1 : train0
train0 : 훈련데이터 + kakao 대화
시행2 : train4
Train4 : 훈련데이터 + 다양한 플랫폼 대화

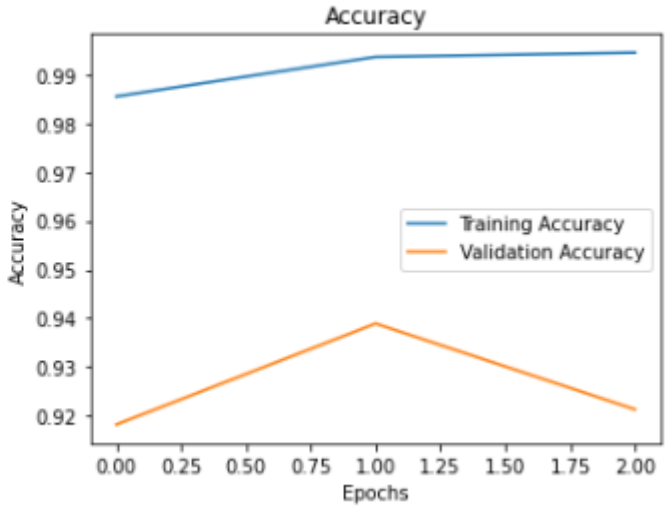
```
def preprocess_sentence(sentence):
    # synonym
    emoticon_normalize(sentence)
    repeat_normalize(sentence)
    #santance = re.sub(r'[^₩₩s]', '', santance)
    # base preprocess
    sentence = re.sub(r'([^a-zA-Zㄱ-ㅎ가-힣?.!,])', " ", sentence)
    sentence = re.sub(r'!+', '!', sentence)
    sentence = re.sub(r'₩?+', '?', sentence)
    sentence = re.sub(r"([?.!,])", r" ₩1 ", sentence)
    sentence = re.sub(r'[" "]+', " ", sentence)
    # 엔터 구분 (₩n)
    sentence = sentence.replace("₩n", "<EOL>")
    sentence = sentence.strip()
    return sentence
```

Klue Bert Model 기반 실험

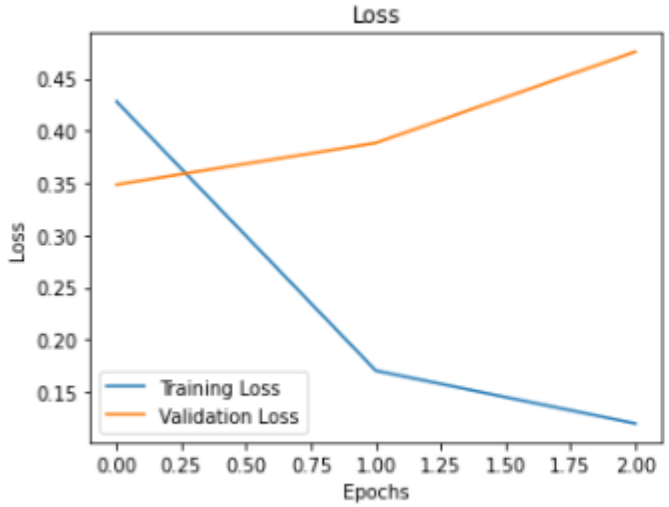
val_loss, val_acc

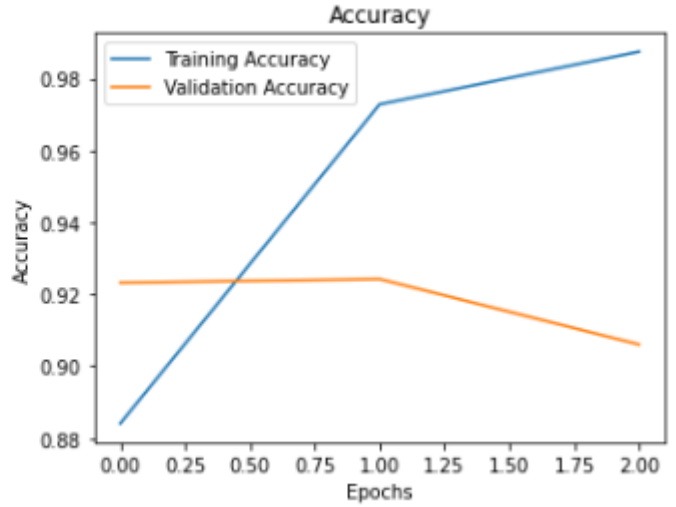
train0





train4





Klue Bert Model 기반 실험

F1-score

train0

Real Accuracy: 0.9182

	precision	recall	f1-score	support
Class 0	0.87	0.89	0.88	196
Class 1	0.85	0.93	0.89	219
Class 2	1.00	1.00	1.00	176
Class 3	0.97	0.92	0.94	196
Class 4	0.92	0.85	0.88	179
accuracy			0.92	966
macro avg	0.92	0.92	0.92	966
weighted avg	0.92	0.92	0.92	966

Weighted F1 Score (based on real predictions): 0.9185

train4

Real Accuracy: 0.9232

	precision	recall	f1-score	support
Class 0	0.87	0.91	0.89	196
Class 1	0.90	0.86	0.88	219
Class 2	1.00	0.99	0.99	200
Class 3	0.95	0.97	0.96	196
Class 4	0.90	0.89	0.89	179
accuracy			0.92	990
macro avg	0.92	0.92	0.92	990
weighted avg	0.92	0.92	0.92	990

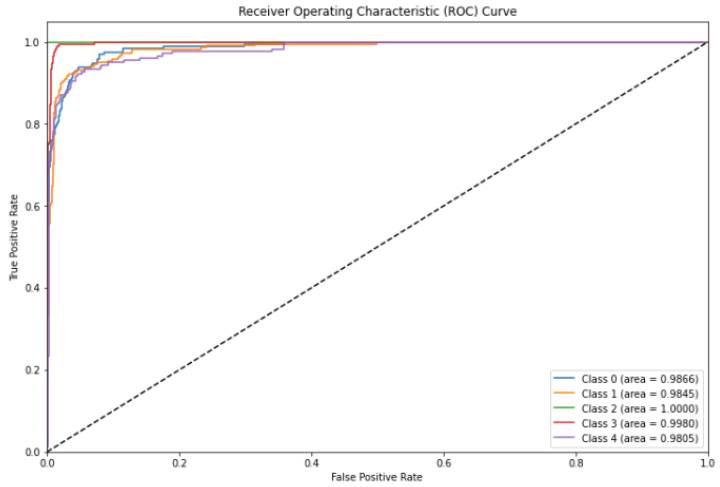
Weighted F1 Score (based on real predictions): 0.9232

Klue Bert Model 기반 실험

AUC-ROC

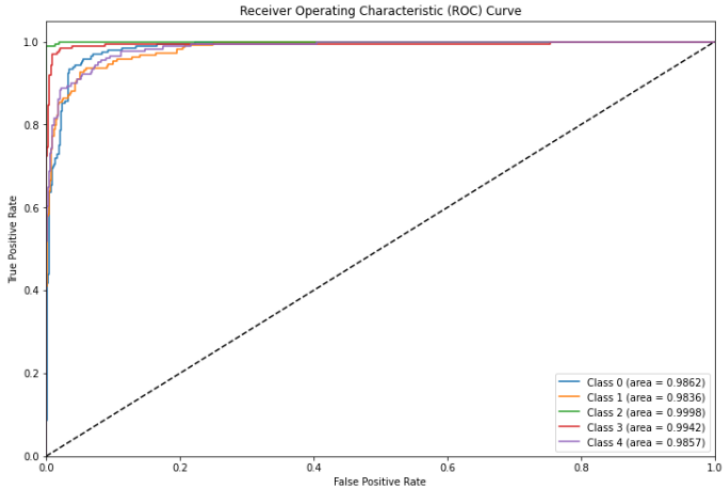
train0

AUC Score: 0.9899



train4

AUC Score: 0.9899



성능 결과 표

- **Base model**
- **Bert model**
- **Klue model**
- **Distil bert model**
- **TFXLMRobert model**

성능 결과 표

	data	구두점삭제유무 (X: 포함 O:삭제	백슬래시 변환유무	Soynlp 포함 여부	증강여부	max_len	**val_accuracy**	**f1 score**	**Submit acc**	
Base(BERT)	train0	X	X	X	X	300	0.8405	0.8401	0.666	
base_preprocess	train0	O	O	O	X	300	0.7173	0.7047	0.578	
base_preprocess_EOL	train0	X	O (EOL)	O	X	300	0.8530	0.8526	0.674	
base_preprocess_aug	train0	O	O	O	O	300	0.8892	0.8869	0.658	
klue_bert	train0	X	O	O	X	250	0.9213	0.9211	0.758	
klue_bert_aug	train0	O	O	O	O		0.9327	0.9349	0.764(250) 0.756(260)	
klue_bert_aug_EOL	train0	X	O (EOL)	O	O	250	0.9110	0.9285	0.778	
distil bert	train0	O	O	O	X		0.8602	0.8616	0.654	
TFXLMRobert	train0	O	O	O	X		0.8664	0.8667	0.668	

성능 향상 시도

- 데이터 변경
- Batch size
- Learning rate
- Max length

	data	batch size	learning rate	max_len	val_acc	f1score	submit acc	비고
✅	train0	8	1e-5	200	0.9265	0.9262	0.766	
✅	train0	8	1e-5	250	0.9265	0.9264	0.768	
✅	train0	8	5e-5	200	0.9326	0.9327	0.778	
✅	train0	8	5e-5	250	0.9161	0.9169	0.74	
✅	train0	16	1e-5	200	`0.9451345801353455`	`0.9452`	0.79	
✅	train0	16	1e-5	250	`0.9265010356903076`	`0.9267`	0.75	
	train0	16	5e-5	200				
▶️	train0	16	5e-5	250				
✅	train4	8	5e-5	250	0.9282	0.9283	0.804	
✅	train4	32	5e-5	250	0.9101	0.9097	0.794	증강 X
❌	train4	8	5e-5	250				
✅	train4	32	5e-5	200	0.9181	0.9176	0.83	
✅	train5	32	5e-5	200	0.9111	0.9101	0.834	
✅	train5	32	5e-5	250	0.9192	0.9185	0.846	
✅	train5	32	5e-5	300	0.9172	0.9167	0.844	
✅	train5	32	schedule	250	0.9253	0.9248	0.824	
✅	train5	8	5e-5	300	0.9263	0.9263	0.844	
✅	train5	16	5e-5	250	0.9152	0.9147	0.826	
▶️	train5	32	5e-5	300				

성능 향상 시도

- .
- .
- .
- .
- .

성능 향상 시도

- .
- .
- .
- .
- .

05 성능 결과표

	data	batch size	learning rate	max_len	val_acc	f1score	submit acc
✔	train0	8	1e-5	200	0.9265	0.9262	0.766
✔	train0	8	1e-5	250	0.9265	0.9264	0.768
✔	train0	8	5e-5	200	0.9326	0.9327	0.778
✔	train0	8	5e-5	250	0.9161	0.9169	0.74
	train0	16	1e-5	200	0.9451345801353455	0.9452	0.79
	train0	16	1e-5	250			
	train0	16	5e-5	200			
	train0	16	5e-5	250			
✔	train4	8	5e-5	250	0.9282	0.9283	0.804
✔	train4	32	5e-5	250	0.9101	0.9097	0.794