DKTC (Dataset of Korean Threatening Conversations)

DLthon - MINIONS

김나경, 고대현, 최호재

DLthon-MINIONS 목차

- **팀소개**
- **데이터**
- **EDA**
- **전처리**
- **BaseLine Model**
  - 학습함수 구현하기
  - 하이퍼파라미터 구현하기
- **Custom Model**
  - 학습 및 결과
- **Pre-trained Model**
  - 학습 및 결과
- **추가 실험**
  - 실험 설계
  - 데이터
    - 다른 프롬프트
    - Augmentation

미니언즈를 소개합니다! 팀원 소개

고대현 (BOB)

김나경 (KEVIN)

최호재 (STUART)

이번 해커톤의 주제는?

DKTC 데이터셋 데이터

대화의 성격을 **위협 세부 클래스 4개** 또는 **일반 대화** 중 하나로 예측하는 과제

- **평가 지표 :** 모델이 분류한 결과와 정답 간의 **"F1-Score"**로 측정

불균형한 데이터셋에서 정밀도와 재현율을 균형 있게 평가하기 위해서

- **데이터**
  - **학습 데이터**
    '협박', '갈취', '직장 내 괴롭힘', '기타 괴롭힘' 등
    4개 클래스 각 약 1천 개로 구성
  - **테스트 데이터**
    '협박', '갈취', '직장 내 괴롭힘', '기타 괴롭힘', '일반 대화' 등
    5개 클래스 각 1백여 개로 구성

일반대화 수집 프롬프트 만들기 데이터

무엇이든 만들어주는 착한 비서

ChatGPT (4o, OpenAI)

“ "한국어 위협 대화 분류"를 위한
데이터셋을 구축하려고 해. ”

Needs

- 현재 학습 데이터에는 ‘협박’, ‘갈취’, ‘직장 내 괴롭힘’, ‘기타 괴롭힘’ 등 4개 클래스 각 약 1천 개로 구성되어 있고 첨부된 ‘train.csv’ 파일에서 확인할 수 있어.
- 동일한 csv 형식으로 ‘일반 대화’ 클래스의 대화 데이터셋을 구성해 줄 수 있어? 최대 천 개의 대화를 생성해줘.
- ‘일반 대화’ 클래스에는 협박, 갈취, 괴롭힘 등의 위협적이거나 공격적인 언어가 없어야 해.
- ‘train.csv’에서는 두 화자가 10번에 걸쳐 주고 받는 대화가 이어지는 데, 이러한 형식으로 데이터셋을 만들어줘.
- 두 화자 간 대화에는 ‘₩n’을 넣어줘.
- 기존 ‘train.csv’와 새로운 데이터셋(일반 대화)을 합친 후, 이를 셔플링해줘.

‘conversation.csv’ 파일 생성

데이터 살펴보기 데이터

idx값, 꼭 필요할까? → 삭제해야 한다.

idx값은 고유값 이므로, 향후 데이터 분석이나 모델 구현에 도움이 되지 않는다.

A, B 간의 대화, A가 대부분 가해자인듯

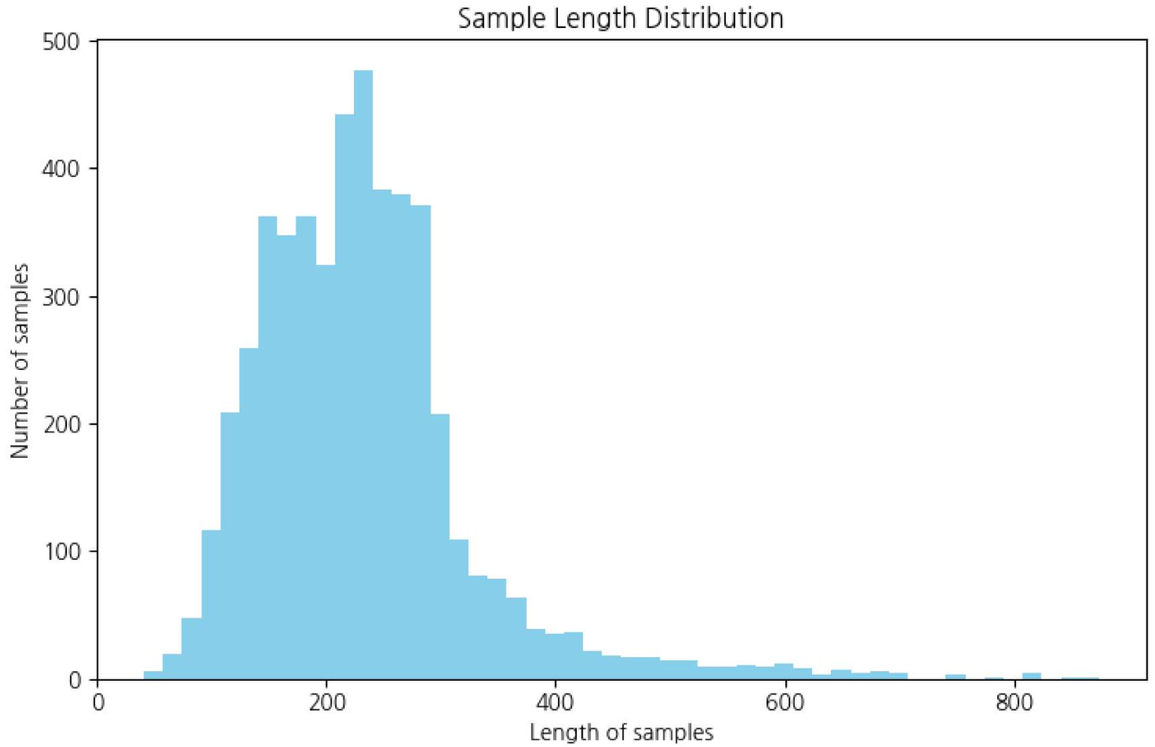
\n을 기준으로 화자가 전환되며, 보통 가해자의 대화로부터 출발한다.

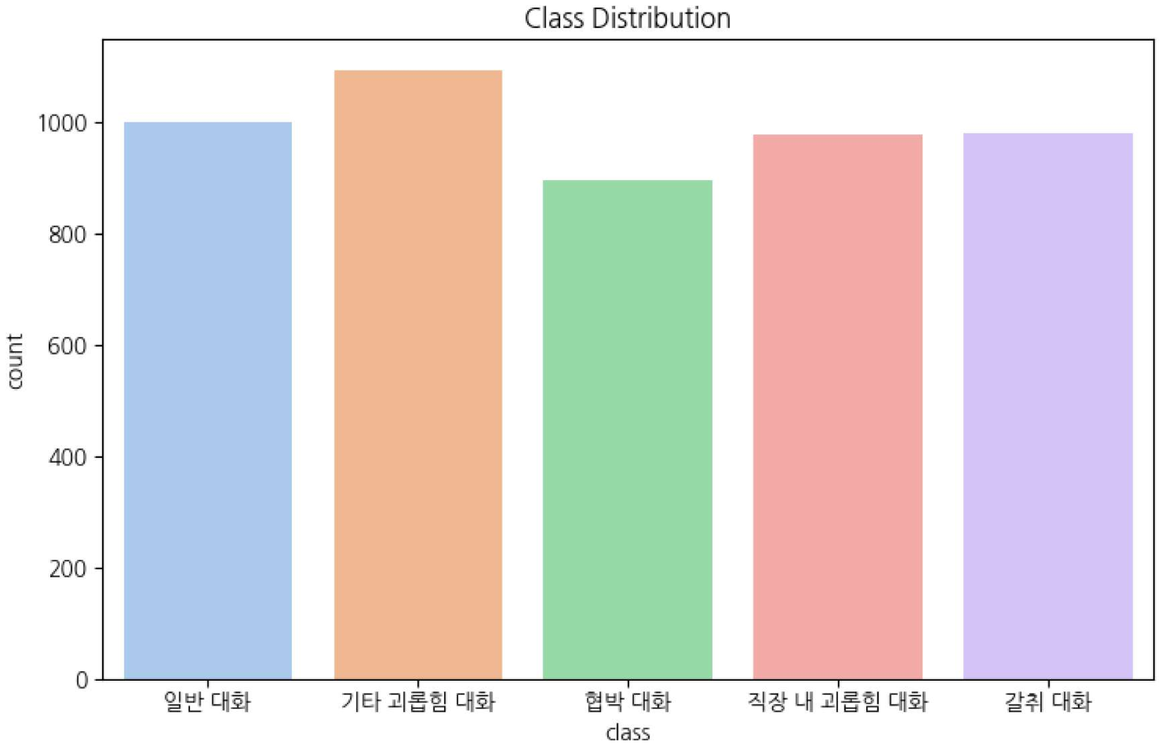
class는 총 5개, 위협적인 대화가 대부분

'협박', '갈취', '직장 내 괴롭힘', '기타 괴롭힘', '일반 대화' 등 5개 클래스로 구성된다.

	idx	class	conversation
4886	4886	일반 대화	여름에 어디 갈 거야?₩n제주도 가려고 해.₩n어디로 여행 가고 싶어?₩n유럽 여행...
2245	2245	기타 괴롭힘 대화	야돼지₩n나.?₩n그래 여기 급식소에서 돼지가 너 빼고 더 있냐?₩n야 그 피자빵은...
4173	4173	협박 대화	어머나 변태인가봐₩n이익₩n진짜 이상한 학생을 만났네 기분이 이상해서 버스에서 빨리...
340	340	기타 괴롭힘 대화	야! 너희 아빠 외국인 노동자라면서? 어느 나라 사람이야?₩n우리아빠 그냥 일하는 ...
3474	3474	협박 대화	음식 리뷰를 그렇게 쓰면 어떻합니까? 당장 글 삭제해!₩n 솔직하게 음식먹은 내용을...
1571	1571	갈취 대화	아 네 안녕하세요! 혹시 가지고 계신 폰이 제 건데 어디서 주웠는지 알 수 있을까요...
912	912	직장 내 괴롭힘 대화	지은씨 커피 좀 사와요.₩n그건 싫어요.₩n사오라면 사와야지 인턴주제.₩n그러니까요...
130	130	기타 괴롭힘 대화	야 대머리 독수리 니 머리카락 어디갔냐₩n그만 놀려라₩n사실을 말하는데 왜 그러냐₩...
4500	4500	협박 대화	어이 세치기하지 말고 뒤로 가.₩n어따대고 반말이야?₩n세치기한 주제에 말이 많네....
1909	1909	협박 대화	야이새끼야 죽고싶어 환장했어?? 내가 잘 따돌리랬지₩n제딴에는 잘했다고 생각했는데....

문장 기준 분석 EDA

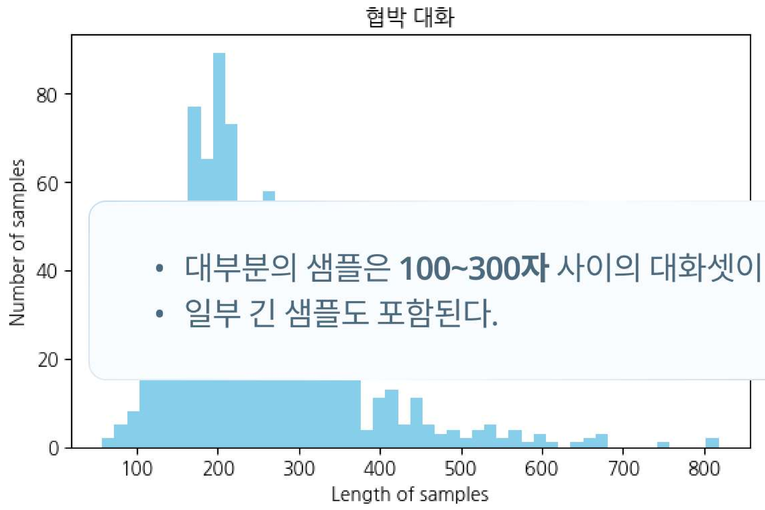




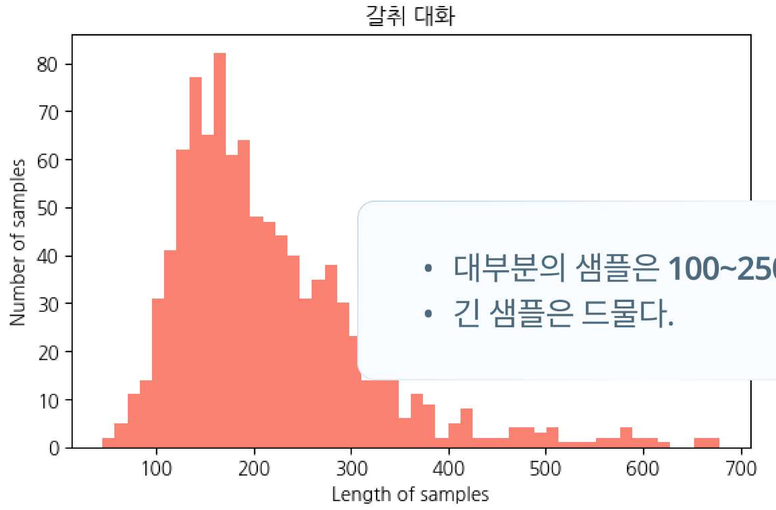
- 대부분의 샘플은 **100~300자** 사이의 대화셋이다.
- 대부분 비슷한 길이의 대화셋이나,
  500자를 넘어서는 일부 데이터셋도 있다.

- 각 클래스의 **샘플 수가 고르게 분포**된다.
- **'기타 괴롭힘 대화'**가 가장 많고, **'협박 대화'**가 가장 적다.

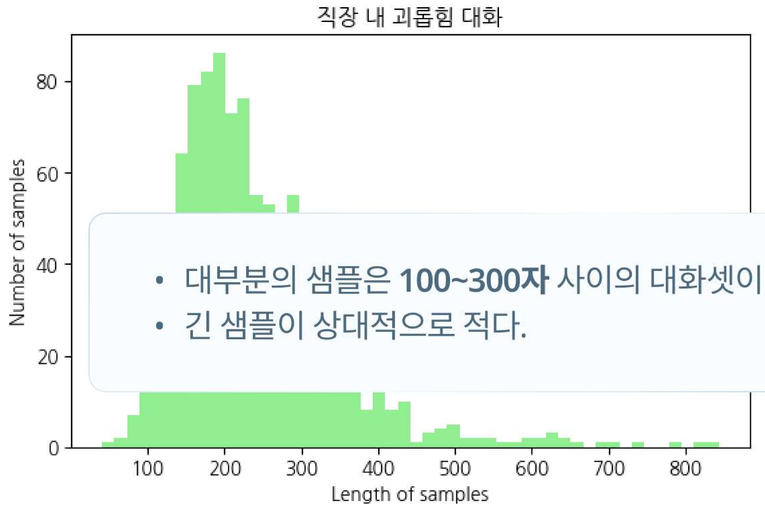
문장 기준 분석 EDA



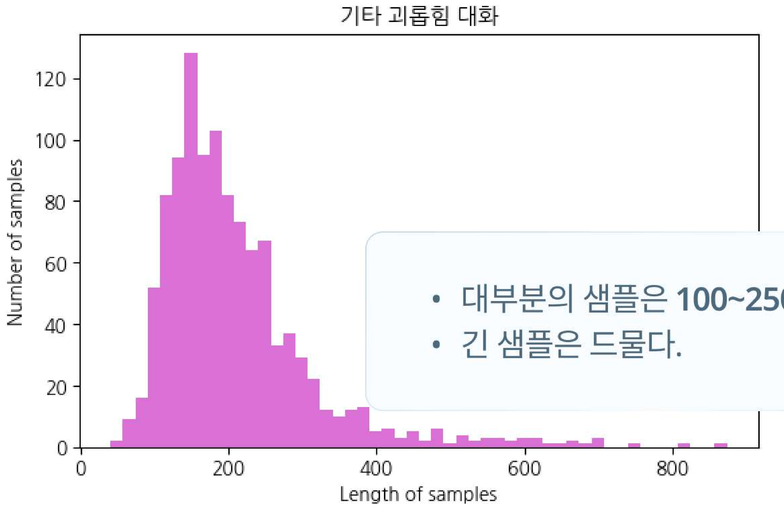
- 대부분의 샘플은 **100~300자** 사이의 대화셋이다.
- 일부 긴 샘플도 포함된다.



- 대부분의 샘플은 **100~250자** 사이의 대화셋이다.
- 긴 샘플은 드물다.

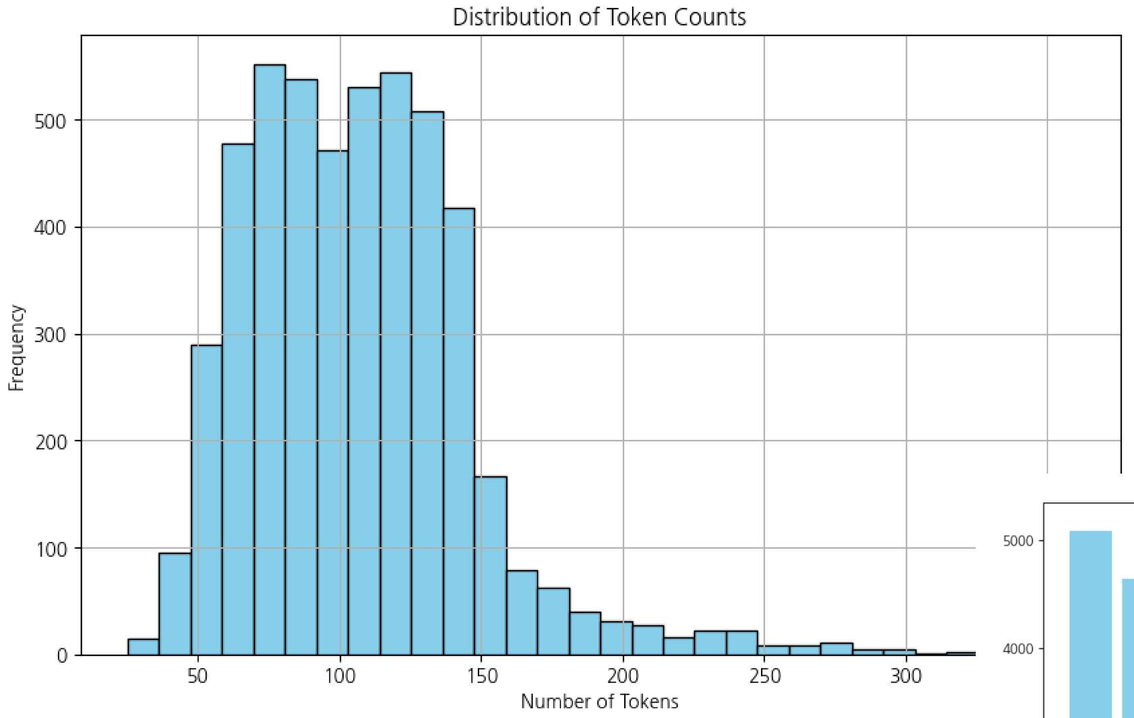


- 대부분의 샘플은 **100~300자** 사이의 대화셋이다.
- 긴 샘플이 상대적으로 적다.



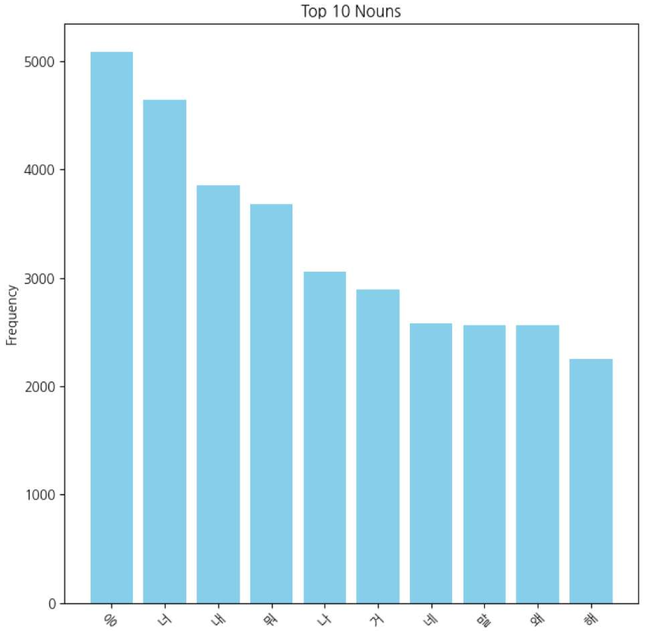
- 대부분의 샘플은 **100~250자** 사이의 대화셋이다.
- 긴 샘플은 드물다.

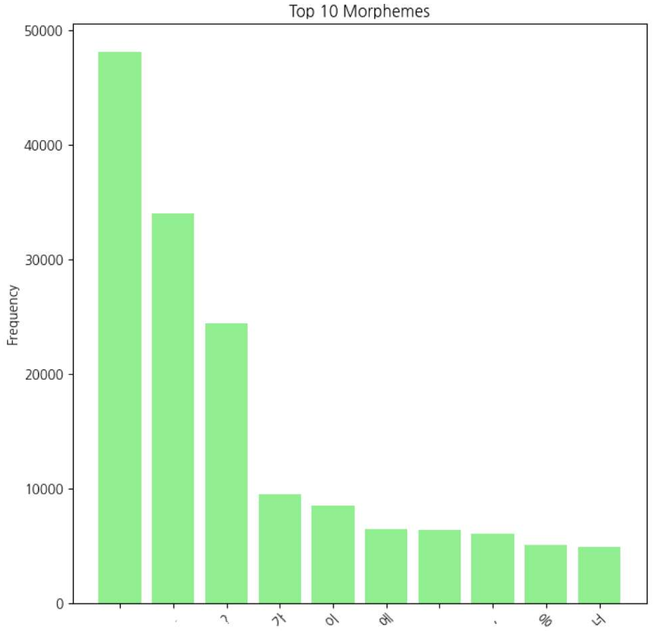
단어 기준 분석 EDA



데이터셋 특성 상, 대화체가 많기 때문에 **토크나이저로 'Okt'를 사용함.**

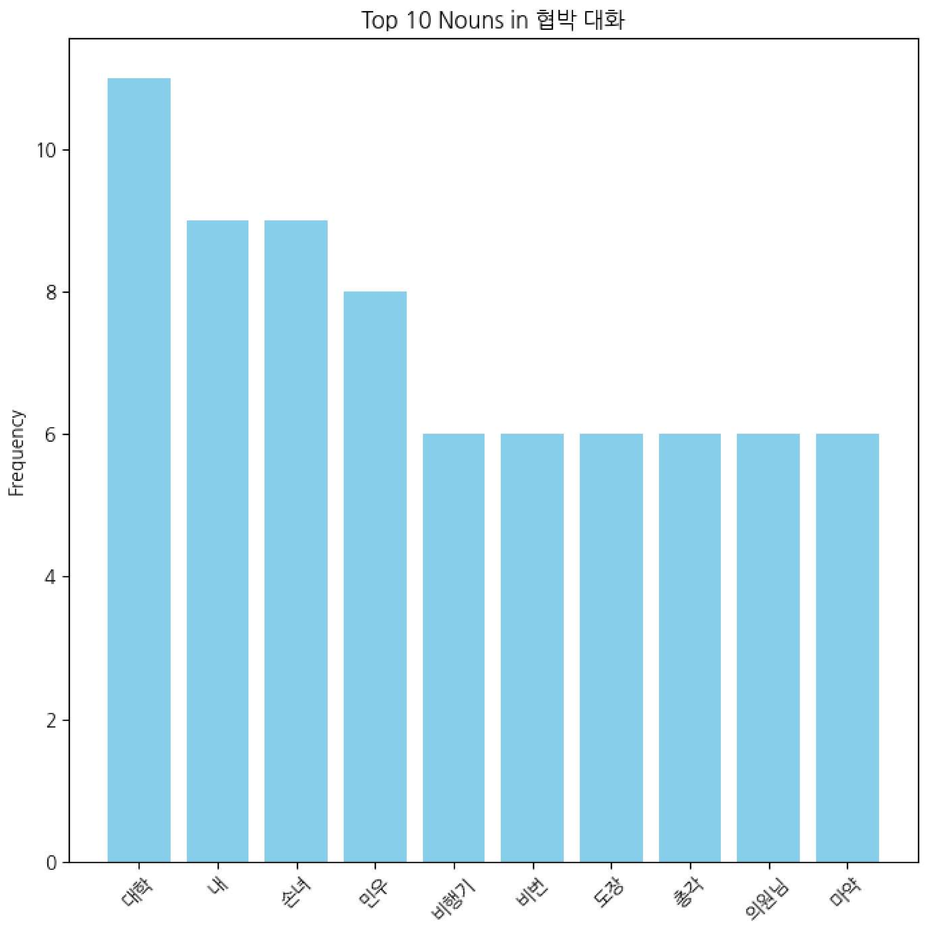
- 대부분의 샘플은 **50~150** 토큰 사이에 분포된다.
- 약 **90~120** 토큰의 빈도수가 가장 많다.
- **200** 토큰을 넘는 샘플은 드물다.

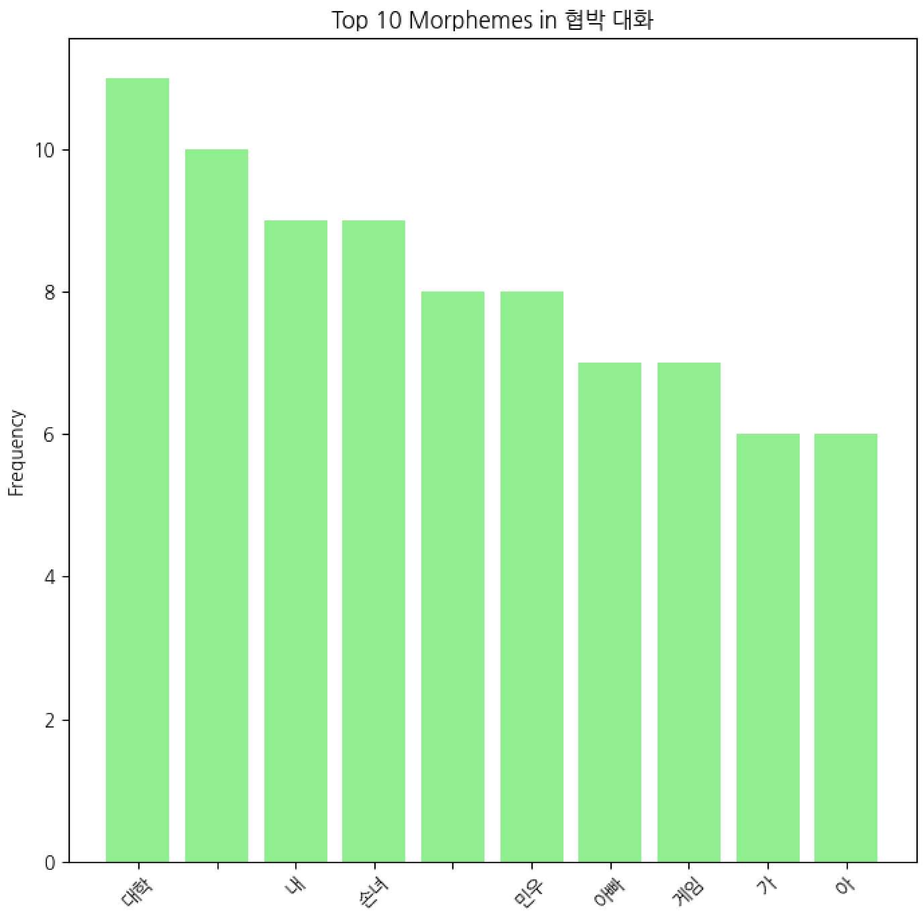




- **쓸모 없는 명사가 많아** 의미 있는 분석을 위해 전처리가 필요하다.
- **공백(" ")이나 줄바꿈(\N)이 상위권**에 포함되어, 전처리가 필요하다.

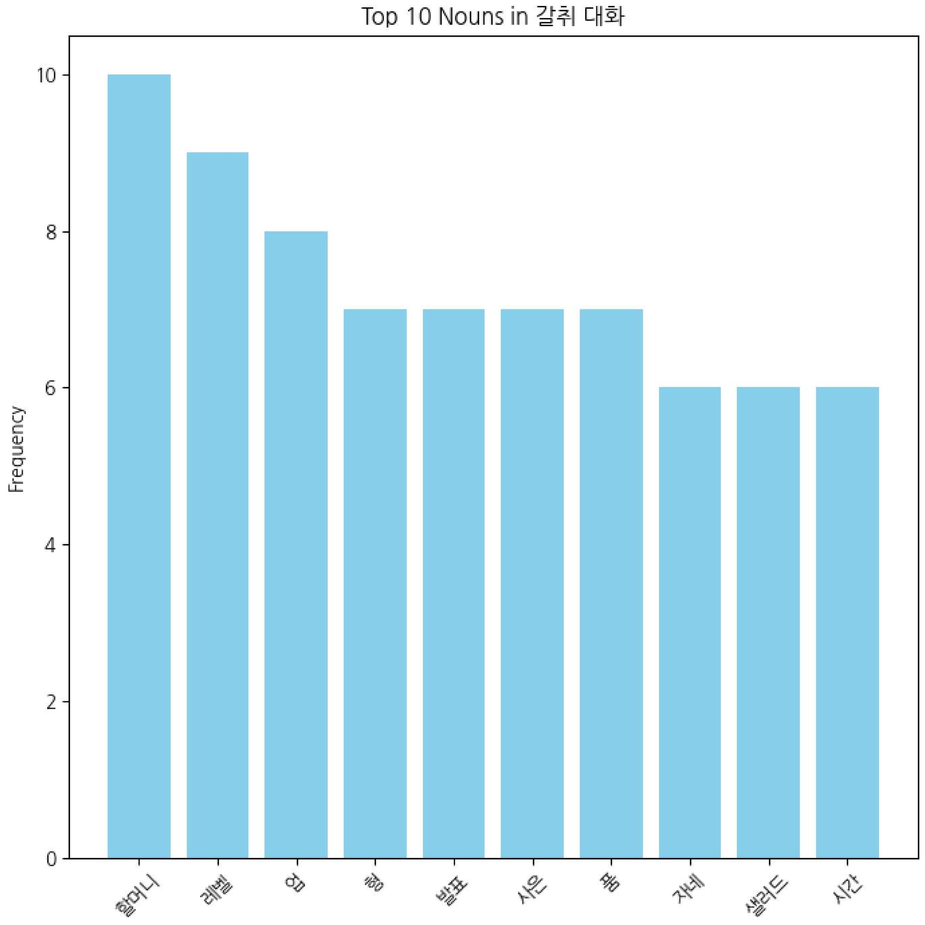
단어 기준 분석 EDA

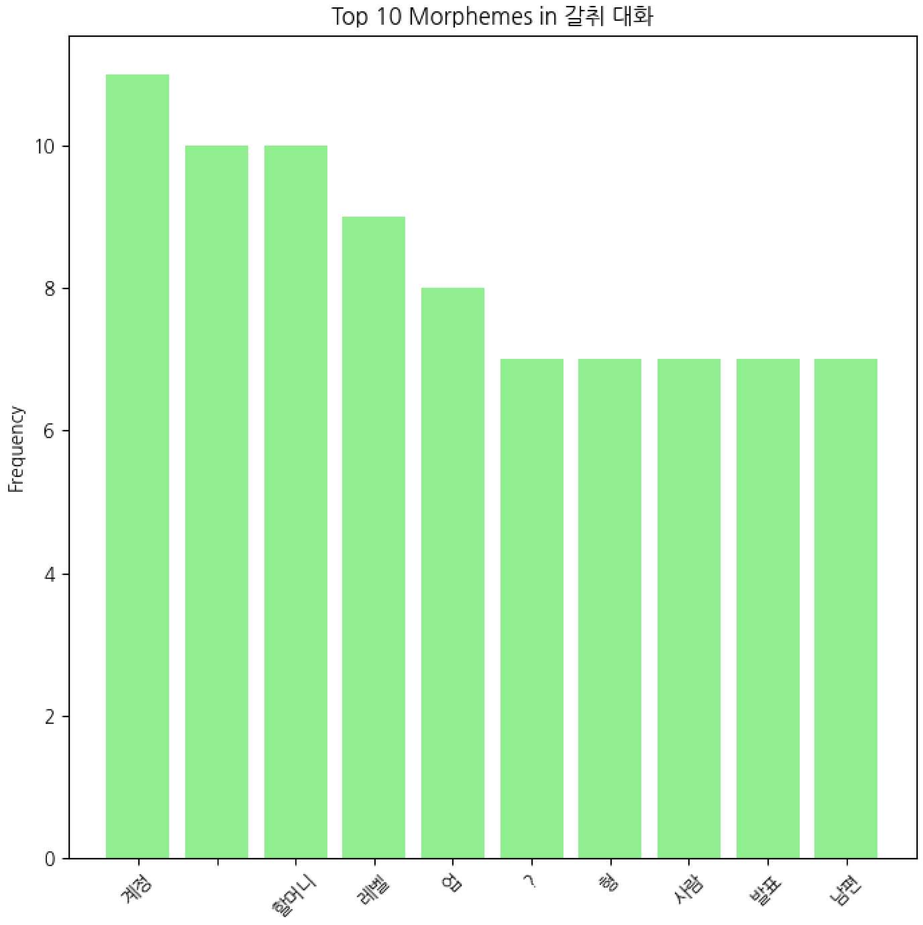




- 협박에서는 **사람을 지칭**하거나, 혹은 **대학에 관련된 내용**이 많이 등장한다.
- 데이터에서 공백 및 줄 바꿈이 상위 형태소로 나타나 **추가적인 전처리가 필요**하다.

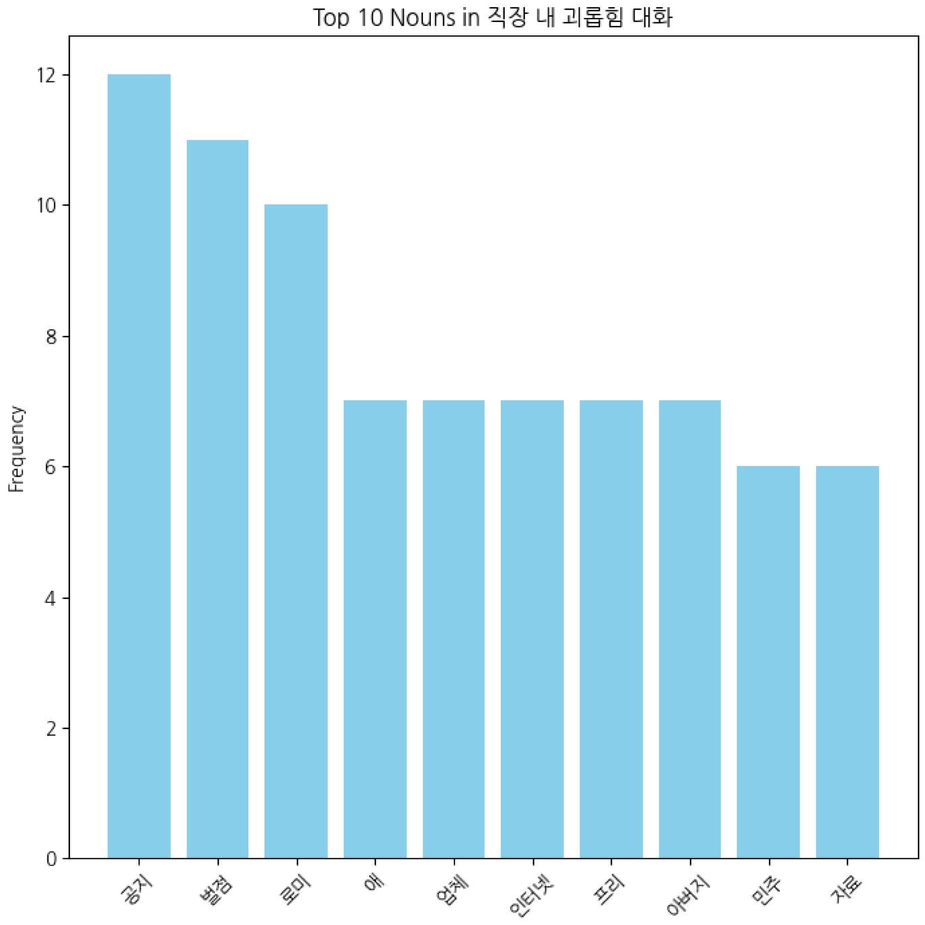
단어 기준 분석 EDA

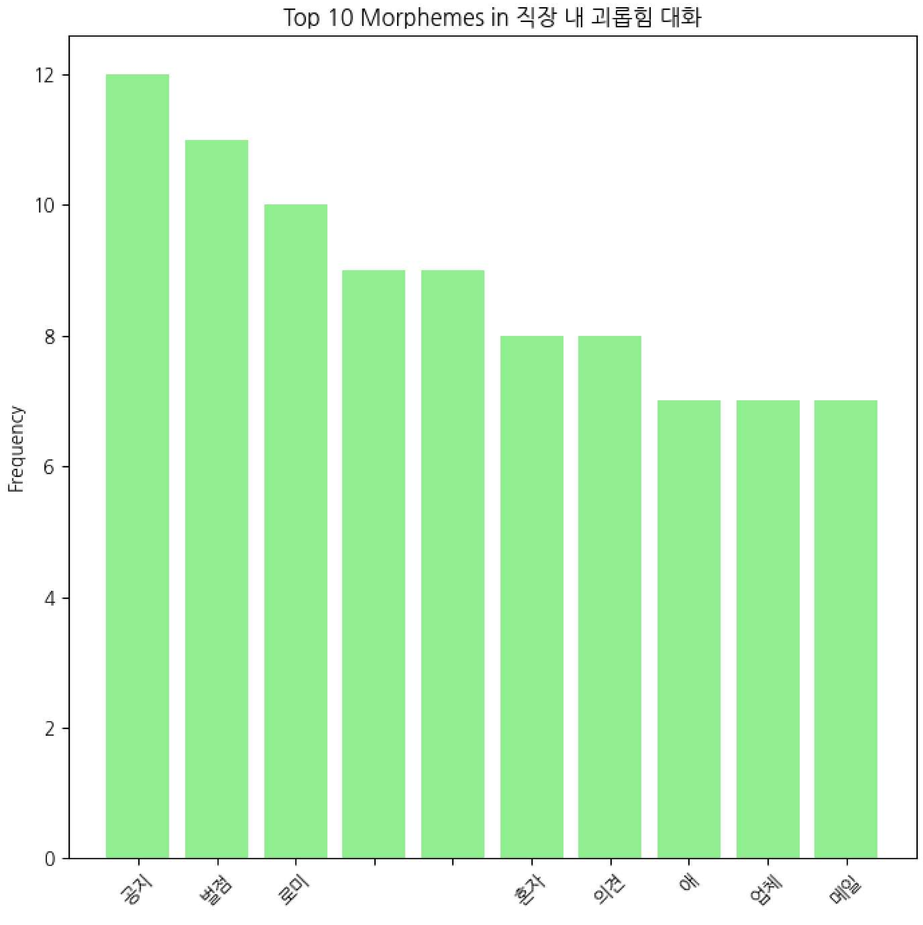




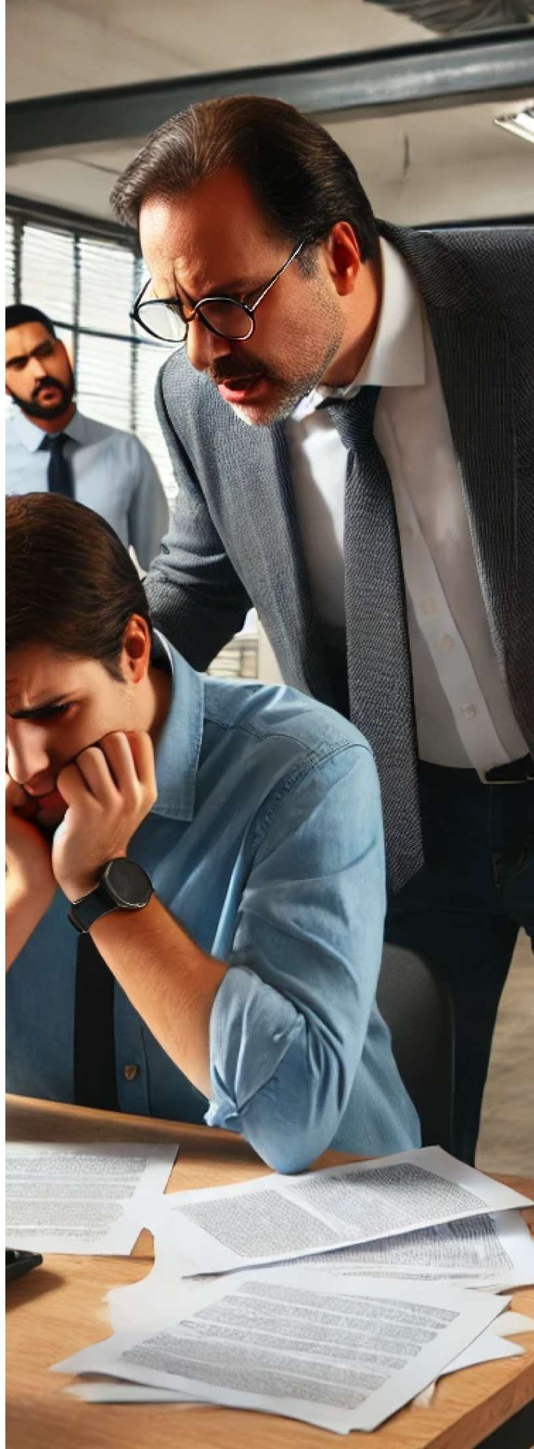
- 갈취에서는 **할머니를 대상/사칭**하거나, 혹은 **사람과 관련된 내용**이 많이 등장한다.
- 데이터에서 공백 및 줄 바꿈이 상위 형태소로 나타나 **추가적인 전처리가 필요**하다.

단어 기준 분석 EDA

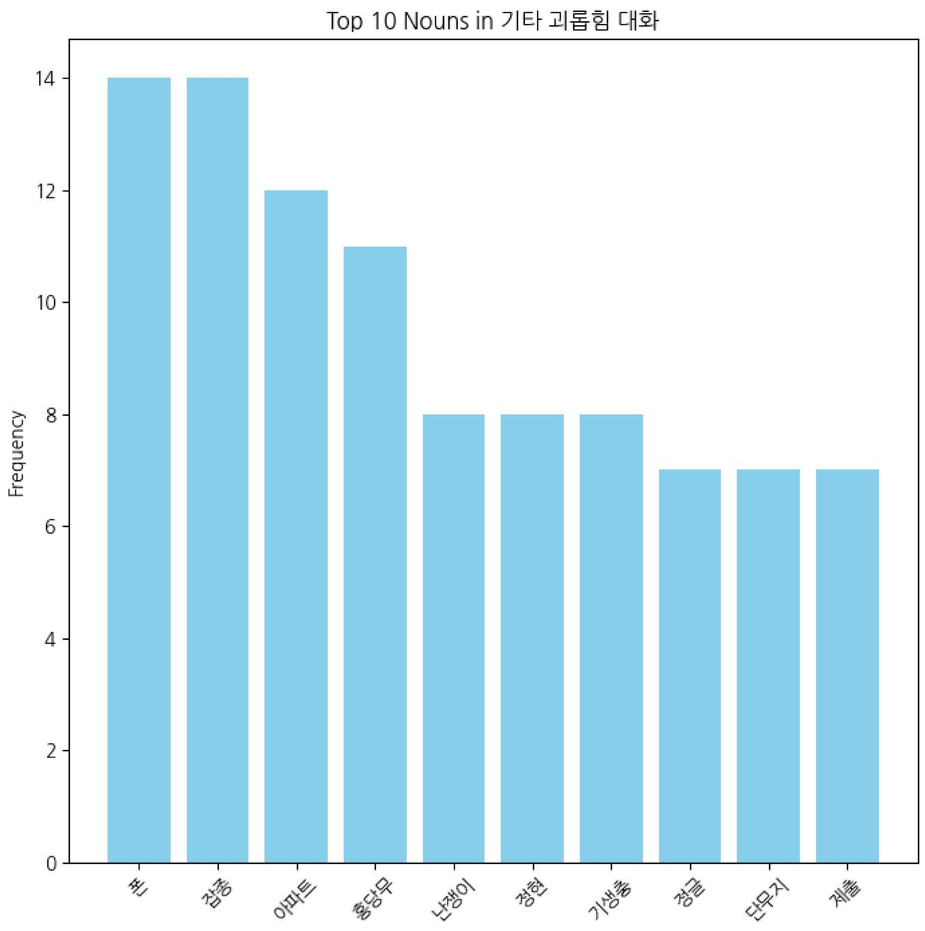


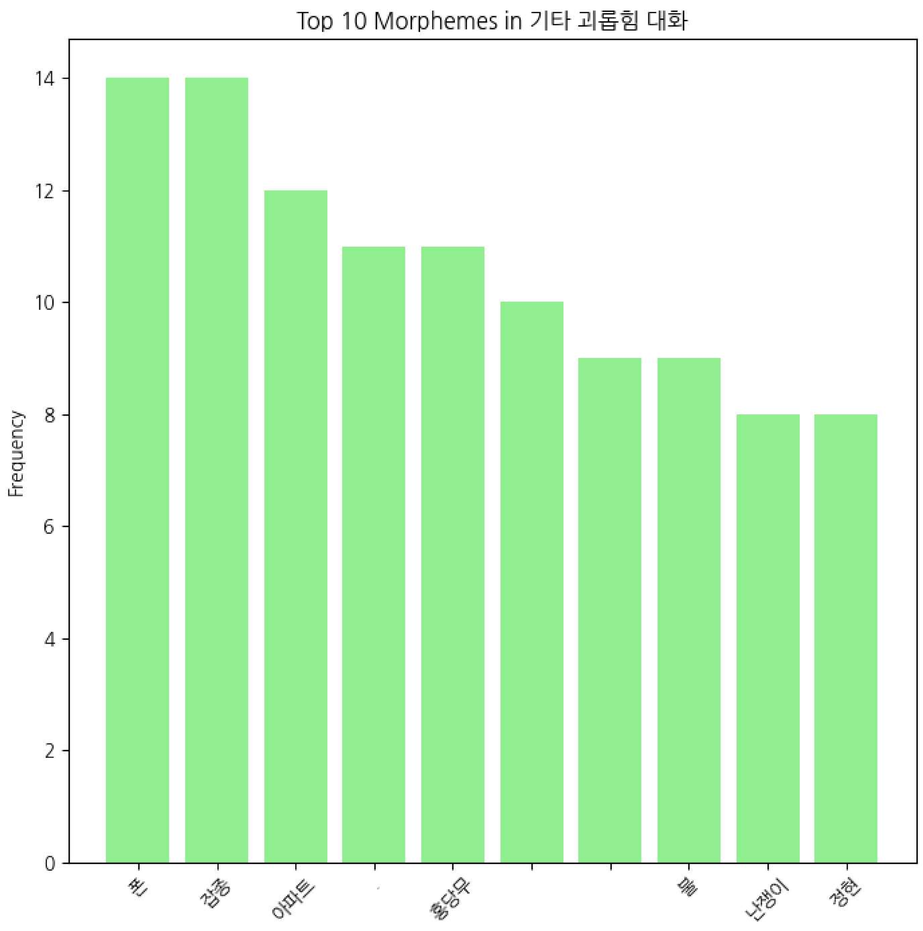


- 직장에서는 **벌점으로 위협**하거나, 혹은 **업체 등 직장 관련 용어**이 많이 등장한다.
- 데이터에서 공백 및 줄 바꿈이 상위 형태소로 나타나 **추가적인 전처리가 필요**하다.



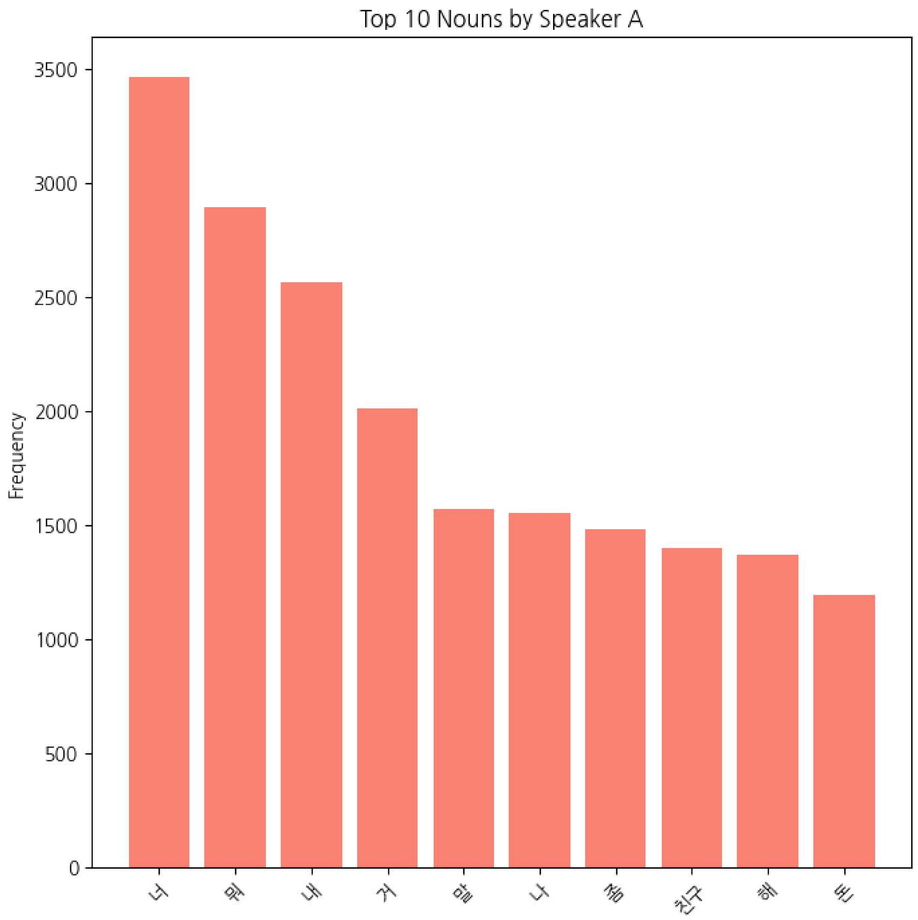
단어 기준 분석 EDA

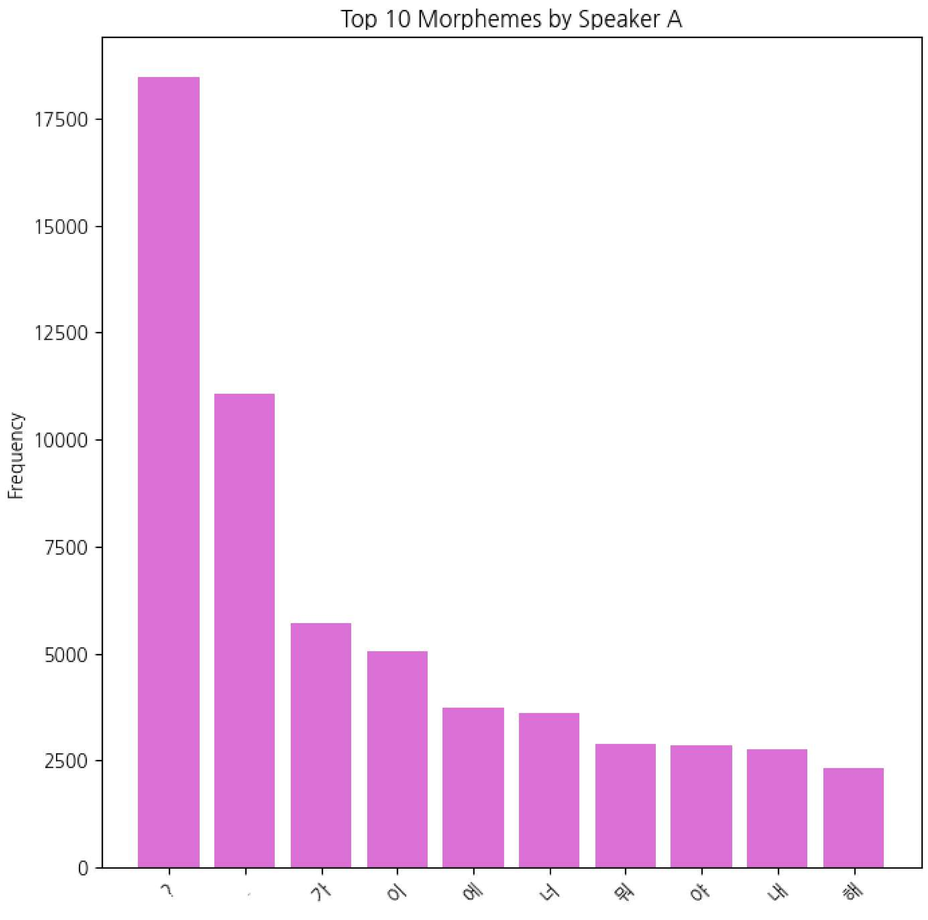




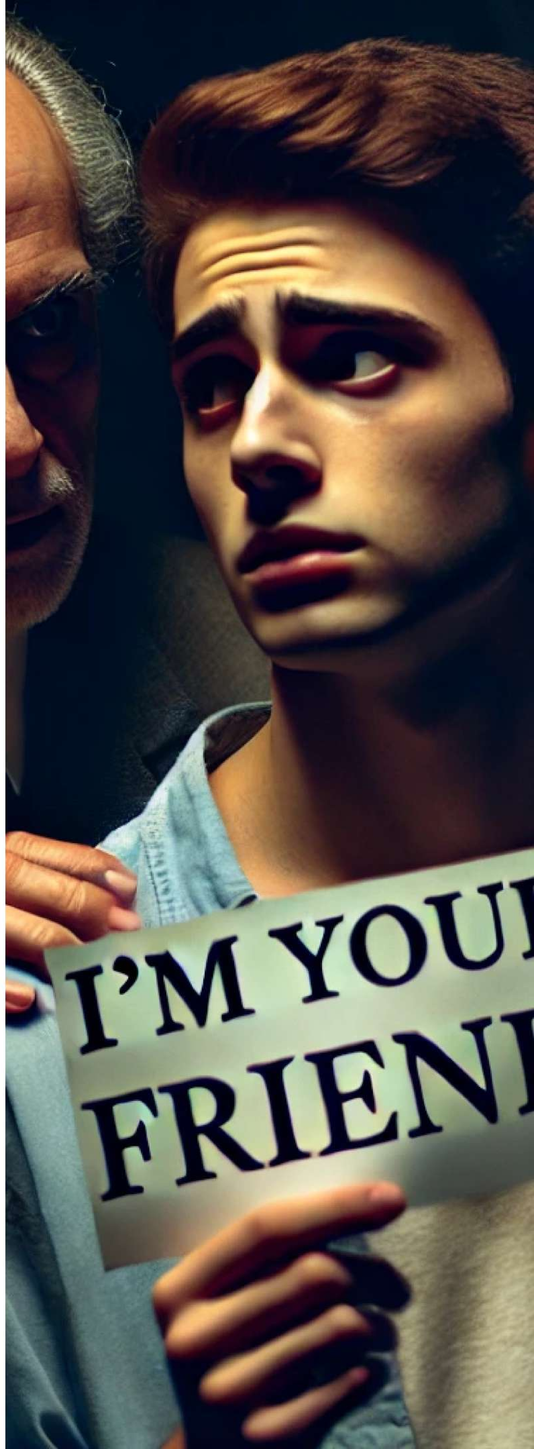
- 이외에도 **폰, 아파트 등** 생활 속 용어 혹은 **기생충 등 비하 용어**가 많이 등장한다.
- 데이터에서 공백 및 줄 바꿈이 상위 형태소로 나타나 **추가적인 전처리가 필요**하다.

단어 기준 분석 EDA

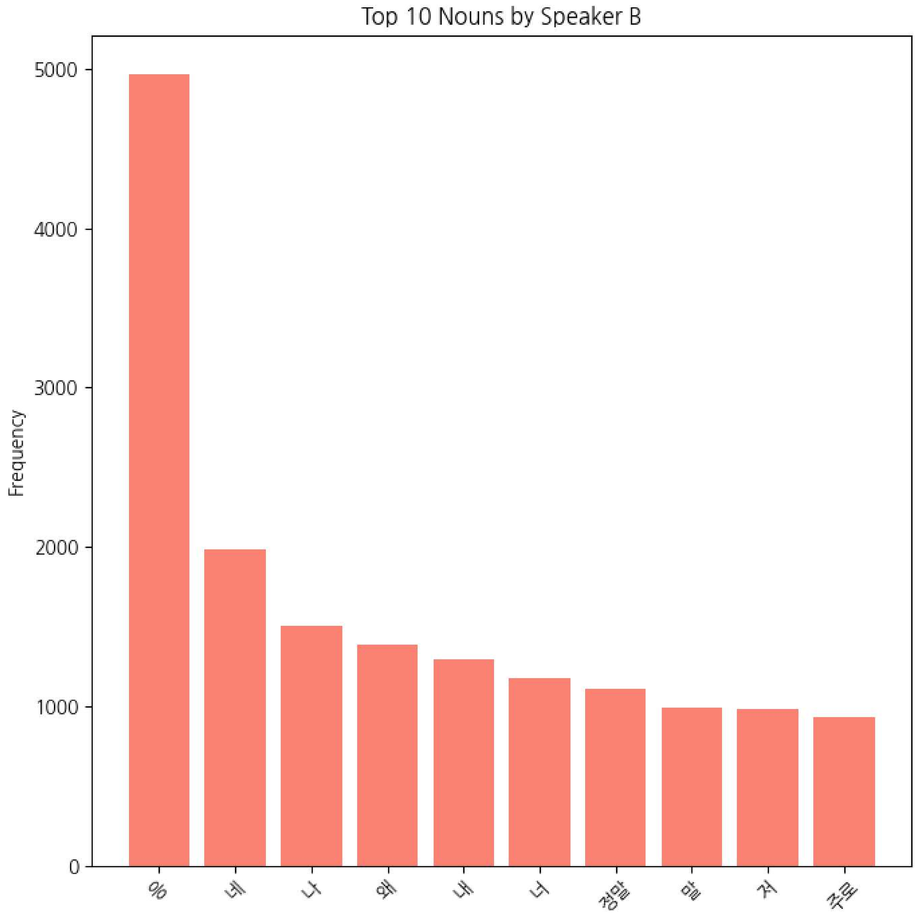


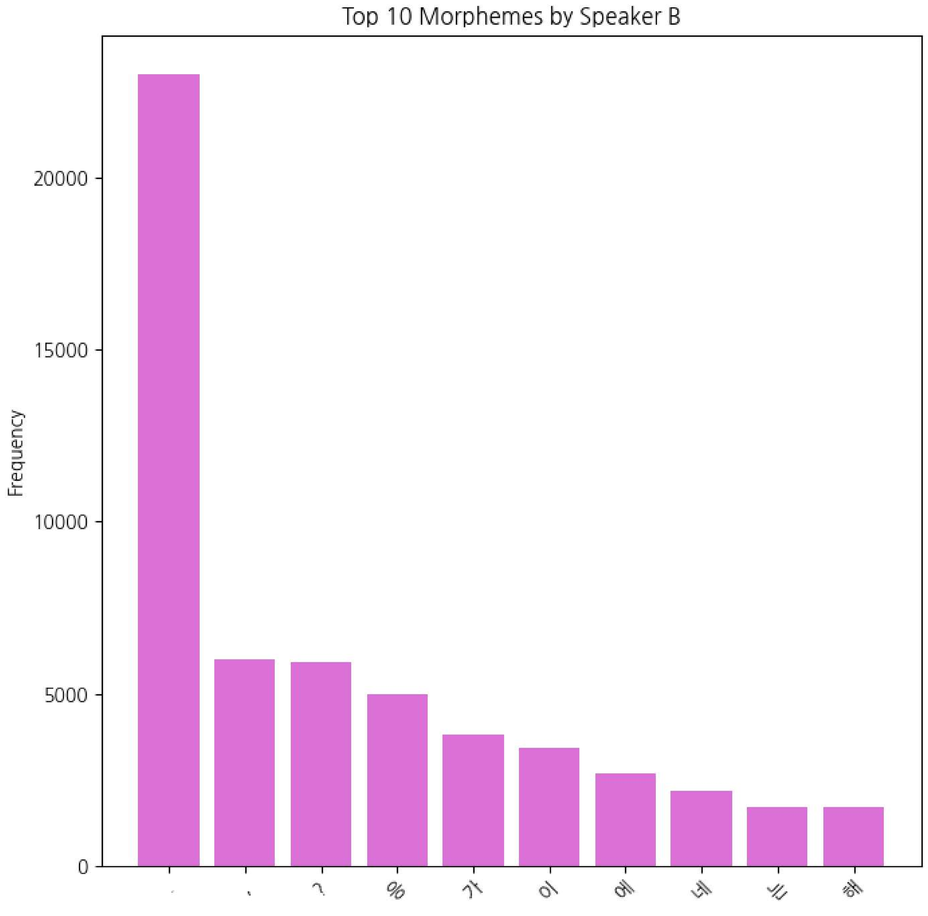


- 주로 **'친구'**, **'돈'** 등 협박이나 갈취에서 들을 수 있는 말이 주로 나왔었다.
- 하지만 분석에 필요 없는 명사와 공백 등의 문제로 **전처리는 반드시 필요**하다.

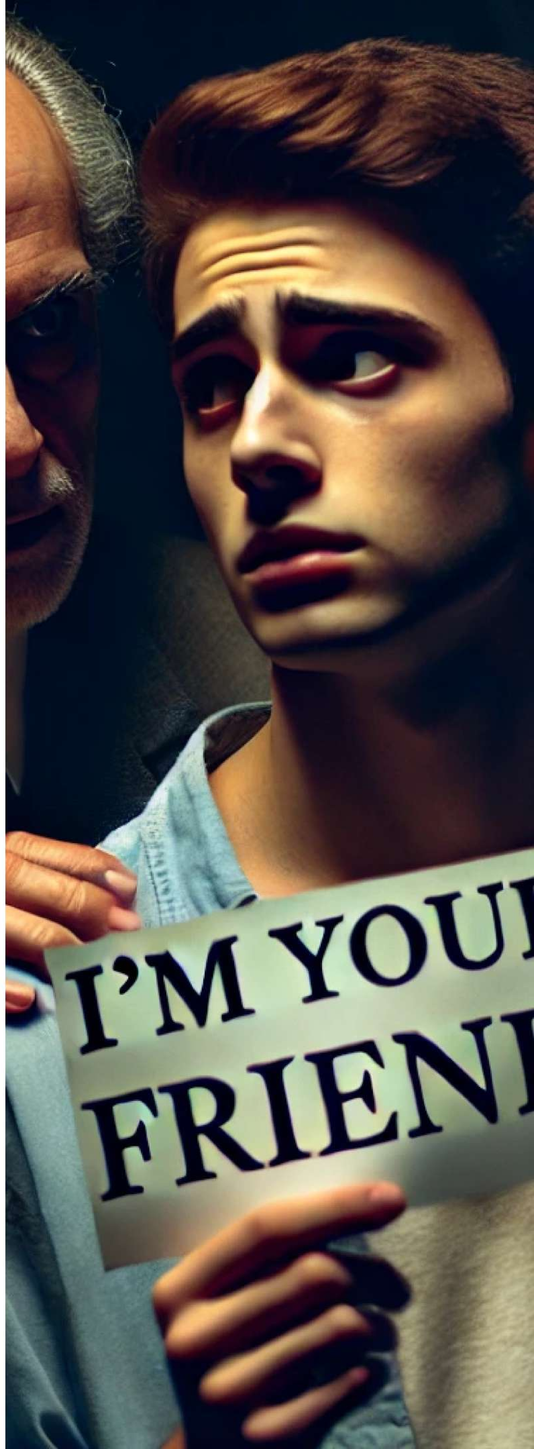


단어 기준 분석 EDA





- 주로 A 화자에 의해서 대화가 진행되기에 **'왜'** 그리고 자신의 진정성을 증명하기 위한 **'정말'** 등이 포함되어 있었다.
- 마찬가지로 공백, 특수 문자, 쓸모 없는 명사 등 **전처리가 필요할 것**으로 보인다.



데이터 전처리 데이터 전처리

기본 전처리	**중복치, 결측치 없음**	**class column 라벨 인코딩**
텍스트 전처리	한글, '?', '!', '.', ',', **공백 유지**	**불용어 삭제** 기존 불용어 리스트 +추가 불용어 리스트

```python
# 전처리 함수
def preprocess_sentence(sentence):
    # \n을 공백으로 바꾸기
    sentence = re.sub("\n", " ", sentence)

    # (ㄱ-ㅎ, ㅏ-ㅣ, ".", "?", "!", ",", ' ')를 제외한 모든 문자를 없애기
    sentence = re.sub("[^ㄱ-ㅣ가-힣.?!, ]", "", sentence)

    # 단어와 구두점(punctuation) 사이에 공백 추가
    sentence = re.sub(r"([?.!,])", r" \1 ", sentence)

    return sentence
```

위의 사항을 구현한 전처리 함수

토큰화하기 데이터 전처리

- **sentence**
  지금 뭐하는 건가 ? 꼼짝마 손들어 내가 그럴 것 같니 ? 딸에게 독약이라도 줄 거야 헉 그건 안 되 딸 살리고 싶으면 내말 들어
  알았어 지금 당장 듣지 않으면 여기 있는 물건 던질거야 알았어 두고 보지
- **SentencePiece Tokenizer**
  ['지금', '뭐하는', '건가', ' ?', ' 꼼짝', '마', ' 손들어', ' 내가', ' 그럴', ' 것', ' 같니', ' ?', ' 딸에게', ' 독약', '이라도', ' 줄', ' 거야', ' 헉', ' 그건', ' 안', ' 되', ' 딸', ' 살리고', ' 싶으면', ' 내말', ' 들어', ' 알았어', ' 지금', ' 당장', ' 듣지', ' 않으면', ' 여기', ' 있는', ' 물건', ' 던', '질거야', ' 알았어', ' 두고', ' 보지']
- **Okt Tokenizer**
  ['지금', '뭐', '하는', '건가', '?', '꼼짝', '마', '손', '들어', '내', '가', '그럴', '것', '같니', '?', '딸', '에게', '독약', '이라도', '줄', '거야', '헉', '그건', '안', '되', '딸', '살리고', '싶으면', '내', '말', '들어', '알았어', '지금', '당장', '듣지', '않으면', '여기', '있는', '물건', '던질거야', '알았어', '두고', '보지']
- **Mecab Tokenizer**
  ['지금', '뭐', '하', '는', '건가', '?', '꼼짝', '마', '손들', '어', '내', '가', '그럴', '것', '같', '니', '?', '딸', '에게', '독약', '이', '라도', '줄', '거', '야', '헉', '그건', '안', '되', '딸', '살리', '고', '싶', '으면', '내', '말', '들', '어', '알', '았', '어', '지금', '당장', '듣', '지', '않', '으면', '여기', '있', '는', '물건', '던질', '거', '야', '알', '았', '어', '두', '고', '보', '지']

예시 문장과 토큰화된 문장들

custom model에 사용하기 위해 직접 토큰화해보자.

- **빨간색**: 원본 문장이 띄어쓰기가 잘 못 되어 있을 때 Okt 토크나이저가 가장 알맞은 띄어쓰기로 토큰을 나누었다
- **파란색**: Okt에 비해 Mecab 토크나이저는 모든 문장성분을 다 나누는 경향이 있다

토크나이저로 'Okt'를 사용함.

토큰화하기 데이터 전처리

	conversation	type	preprocessed	tokenized	encoded
2029	성오야 미안한데 나 만원만 빌려주면 안 돼 ? 미안해 . 나도 이번달에 지출이 많...	1	성오야 미안한데 나 만원만 빌려주면 안 돼 ? 미안해 . 나도 이번달에 지출이 많...	[성오, 미안한데, 만원, 만, 빌려주면, 돼, ?, 미안해, ., 나도, 달, 지...	[18148, 1090, 90, 24, 1287, 42, 3, 132, 2, 84,...
1958	야 너네 집 돈 많다며 아닌데 누가 그래 너 집 대기업 한다고 소문이 자자하던데 맞...	1	야 너네 집 돈 많다며 아닌데 누가 그래 너 집 대기업 한다고 소문이 자자하던데 맞...	[집, 돈, 많다며, 아닌데, 누가, 집, 대기업, 한다고, 소문, 자자하던데, 맞...	[43, 16, 10269, 414, 175, 43, 3739, 394, 487, ...
1705	친구랑 잘 지내 ? 응 , 다들 잘 지내고 있어 . 친구랑 같이 무슨 활동 해...	4	친구랑 잘 지내 ? 응 , 다들 잘 지내고 있어 . 친구랑 같이 무슨 활동 해...	[친구, 랑, 잘, 지내, ?, ,, 다, 잘, 지내고, 있어, ., 친구, 랑, ...	[17, 21, 18, 136, 3, 4, 14, 18, 372, 20, 2, 17...

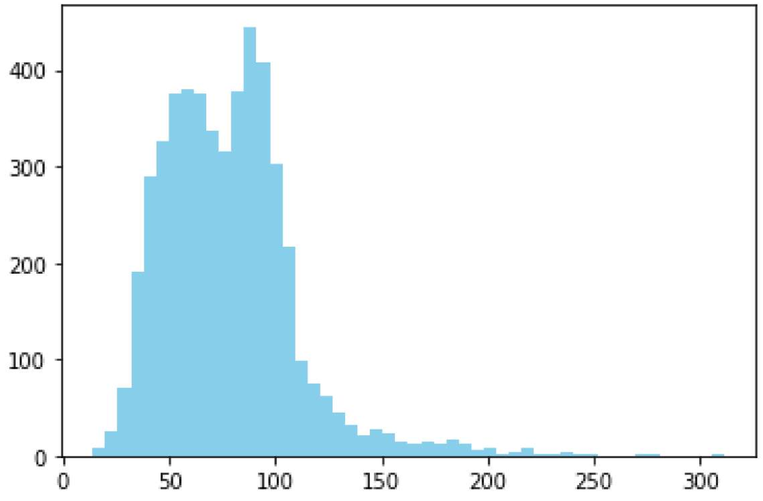
- Vocab.Txt 파일로 단어 사전 저장한다.
- 이를 이용하여 정수 인코딩을 한다.

데이터 분할하기 데이터 전처리

문장의 최대 길이를 정하기 위해 토큰화된 데이터 길이 분포를 확인해본다.

최대 길이 150으로 설정하고 패딩을 시도한다.

- **Max Length** : 311
- **Min Length** : 14
- **Average Length** : 77.63575757575758
- **Standard Deviation** : 31.53093869760893



훈련, 검증, 테스트셋으로 데이터를 분할하여 실험을 진행한다.

훈련, 검증, 테스트 데이터셋 0.8/0.1/0.1로 분할한다.

훈련 데이터의 개수 : 3960

검증 데이터의 개수 : 495

테스트 데이터의 개수 : 495

훈련 레이블의 개수 : 3960

검증 데이터의 개수 : 495

테스트 레이블의 개수 : 495

BaseLine Model 모델

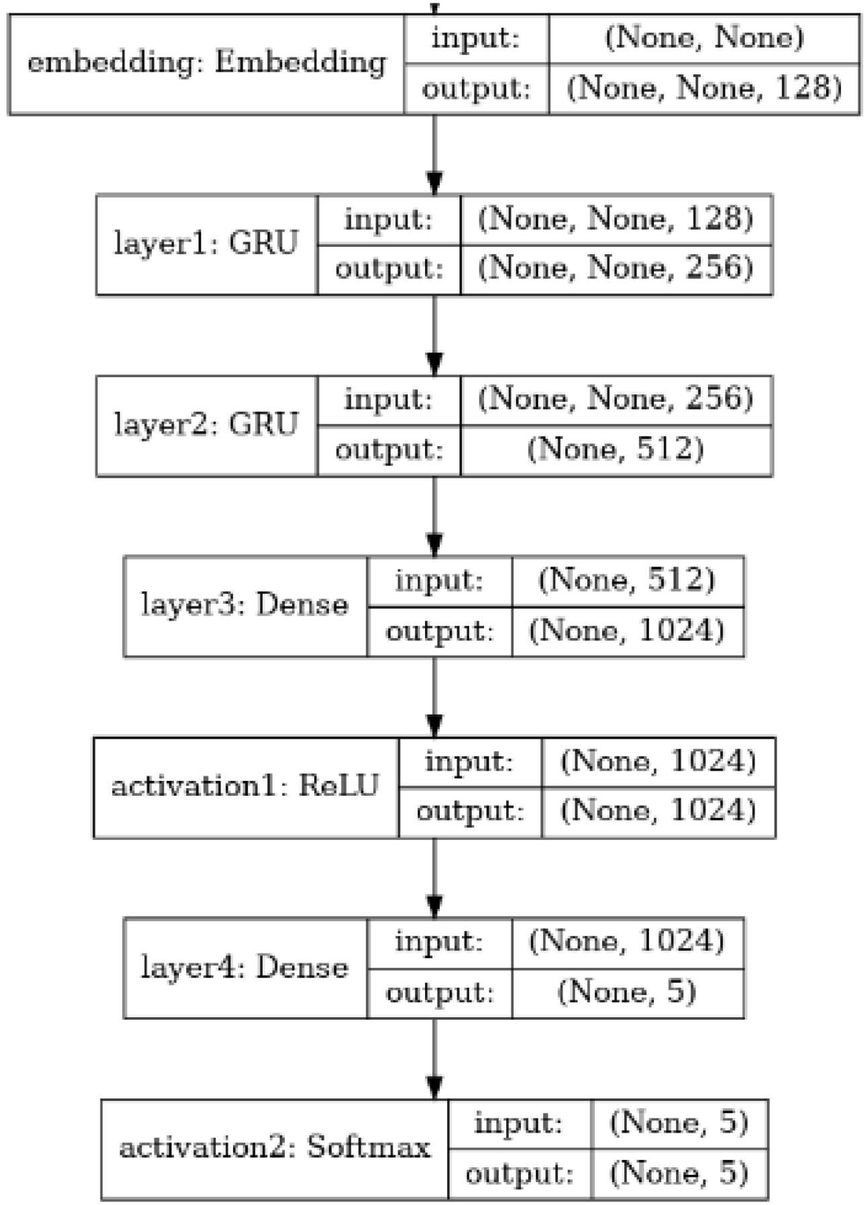
BaseLine Model

Model Structure

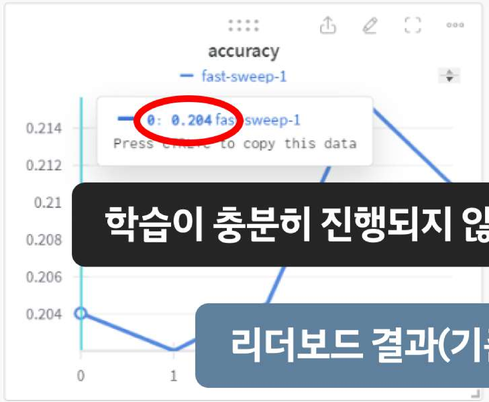
- 2개의 GRU 레이어 및 2개의 dense 레이어
- dropout 없음

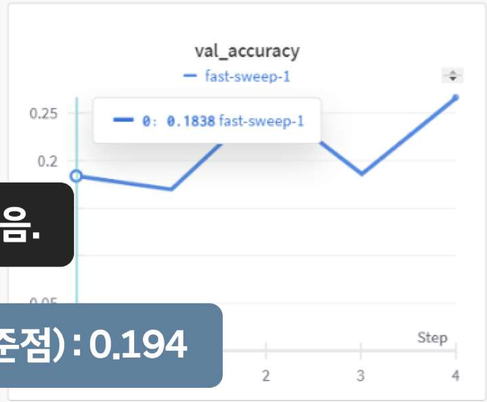
Training Spec

- batch_size: 32
- dropout_rate: 0.1
- epoch: 4
- learning_rate: 0.05
- optimizer: rmsprop
- loss: sparse categorical_crossentropy
  - 다중 클래스 분류 문제
- Evaluation Matrix : Accuracy(정확도), F1-Score
  - Accuracy(정확도) : 분류 모델 성능을 직관적이고 전반적으로 평가 가능
  - F1-Score : 데이터의 불균형을 반영, 리더보드 평가 지표



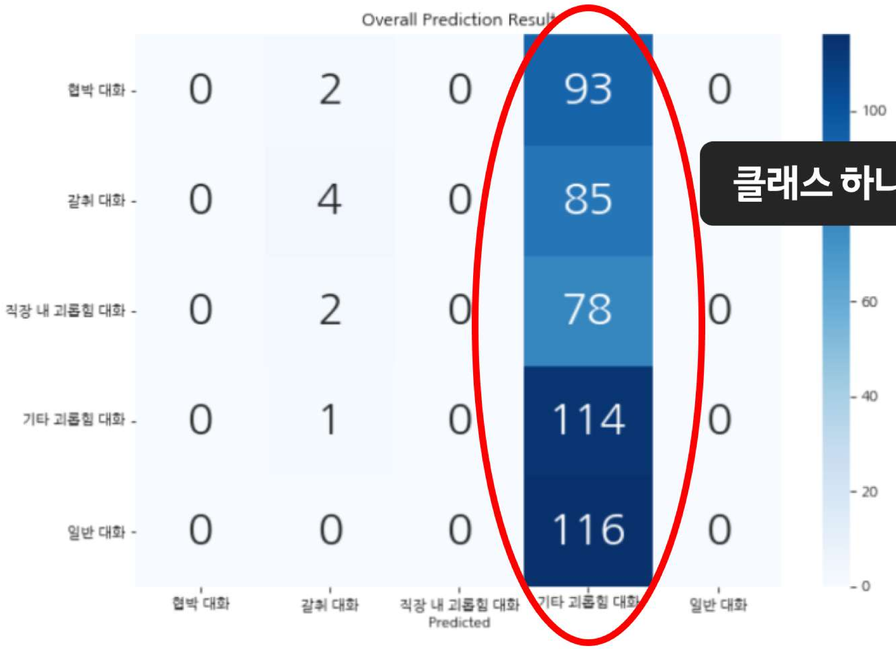
BaseLine Model 모델



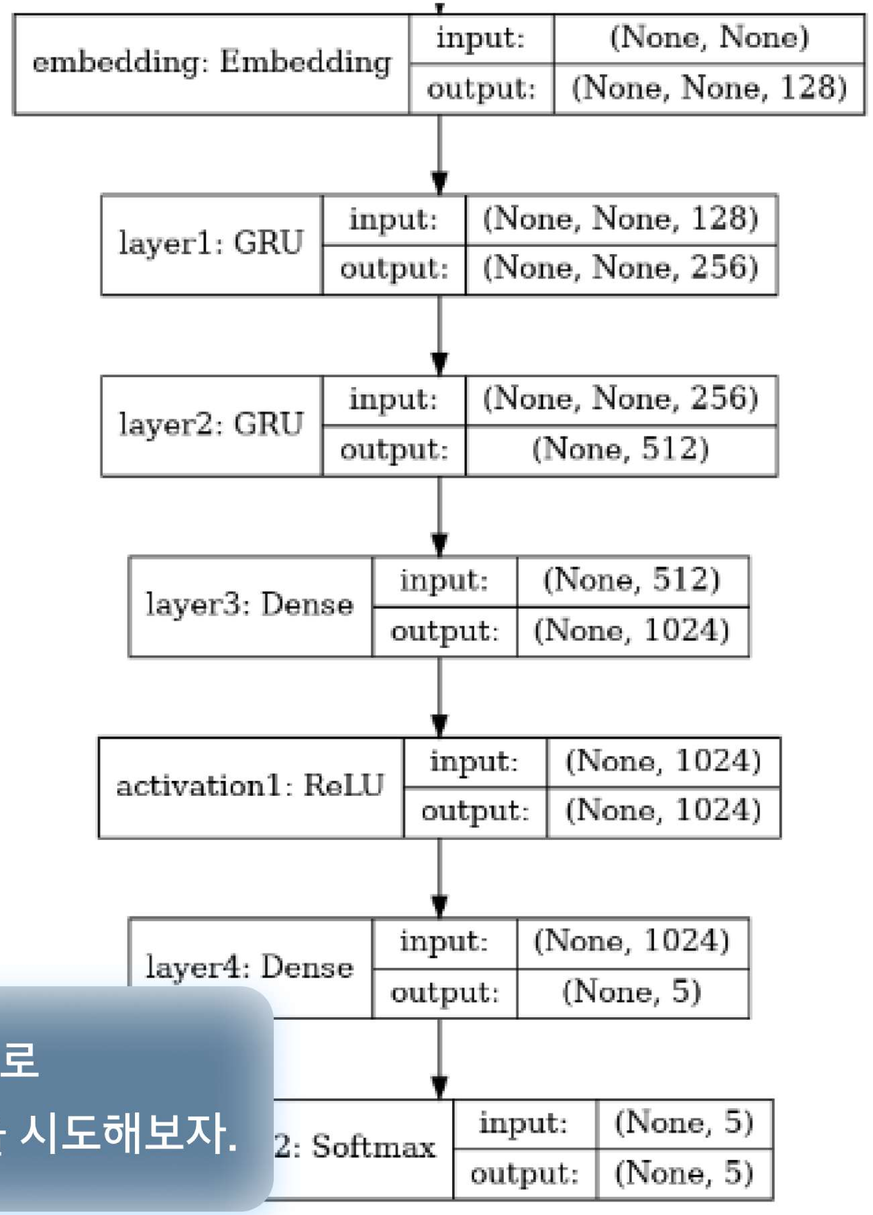


학습이 충분히 진행되지 않음.

리더보드 결과(기준점) : 0.194



클래스 하나로 예측하는 경향이 있음.



동일한 모델로
하이퍼파라미터 튜닝을 시도해보자.

BaseLine Model_Hyperparameter Tuning 모델

BaseLine Model

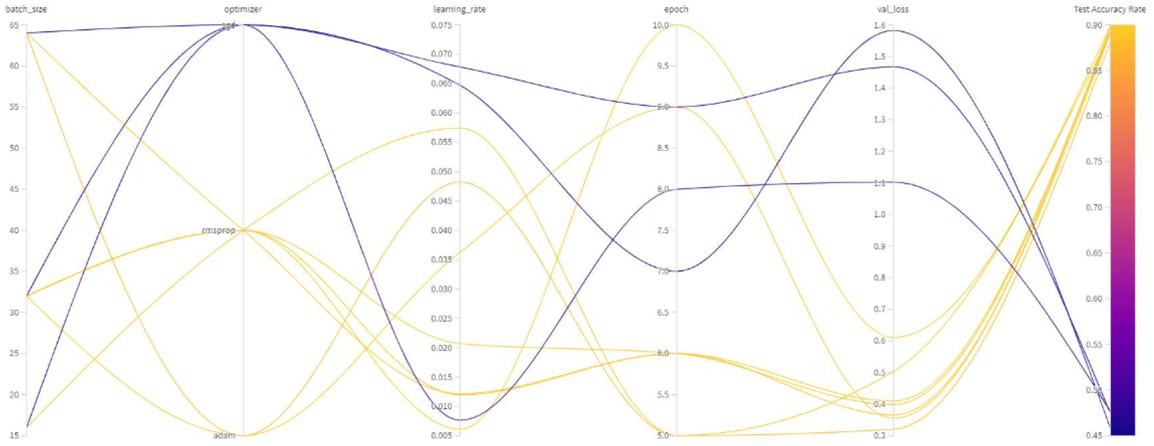
Hyperparameter Tuning

- How To : RandomSearch
- Parameters
  - Optimizer
    - 처음에는 sgd, adam, rmsprop 세 개의 옵티마이저 활용
      - sgd가 다른 두 옵티마이저에 비해 성능 확연히 안 좋음
    - adam, rmsprop만 활용
  - Epoch
    - baseline 모델에서 epoch 5로 했을 때 학습이 충분히 진행되지 않았으므로 최대 10으로 설정
  - Others
    - 개별적으로는 성능에 영향을 크게 미치지 않음.
    - 다른 파라미터와의 조합이 영향을 미침.
    - ex. learning rate가 작을 때에는 epoch이 커야 성능이 좋음

```
"paameters": {
    "learning_rate" : {
        "min" : 0.01,
        "max" : 0.1
        },
    "epoch" : {
        "distribution" : "int_uniform",
        "min" : 5,
        "max" : 10
        },
    "batch_size": {
        "values": [16, 32, 64]
        },
    "optimizer": {
        "values": ["adam", "rmsprop"]
        },
    "dropout_rate":{
        "values": [0.1, 0.2, 0.3]
        }
    }
```

사용한 하이퍼파라미터

하이퍼파라미터 튜닝 시각화



BaseLine Model_학습 함수 구현하기 모델

config 설정하여 하이퍼파라미터 튜닝 준비하기

W&B 활용하여 모델 학습 관리하기

```python
# 학습
# CompareResultsCallback 테스트 데이터셋을 활용하므로 인자로 넣어줌
def train(default_config, X_test, y_test):

    wandb.init(config = default_config)
    config = wandb.config

    keras.backend.clear_session()

    # Model
    model = build_model_1DCNN(config)

    # Compile
    model.compile(optimizer = config.optimizer,
                  loss = config.loss,
                  metrics = config.metrics)

    # 비교 결과 그리는 콜백
    cr_callback = CompareResultsCallback(X_test, y_test, config.class_num)
    # early stopping 콜백
    es_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=2)

    # 학습
    history = model.fit(X_train, y_train,
              epochs = config.epoch,
              batch_size = config.batch_size,
              validation_data = (X_val, y_val),
              callbacks=[wandb.keras.WandbCallback(), cr_callback, es_callback])

    # test dataset으로 accuracy 계산
    test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
    pred_test = model.predict(X_test).argmax(axis=1)
```

분류 결과 확인하기

early stopping을 위한 콜백 추가하기

```python
    model.compile(optimizer = config.optimizer,
                  loss = config.loss,
                  metrics = config.metrics)

    # 비교 결과 그리는 콜백
    cr_callback = CompareResultsCallback(X_test, y_test, config.class_num)
    # early stopping 콜백
    es_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=2)

    # 학습
    history = model.fit(X_train, y_train,
              epochs = config.epoch,
              batch_size = config.batch_size,
              validation_data = (X_val, y_val),
              callbacks=[wandb.keras.WandbCallback(), cr_callback, es_callback])

    # test dataset으로 accuracy 계산
    test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
    pred_test = model.predict(X_test).argmax(axis=1)
    # test dataset으로 f1 score 계산
    f1_score_res = f1_score(y_test, pred_test, average='micro')

    # wandb에 log 추가
    wandb.log({
        "Test Accuracy Rate": test_accuracy,
        "Test F1 Score": f1_score_res,
        "Test Error Rate": 1 - test_accuracy
    })

    return history
```

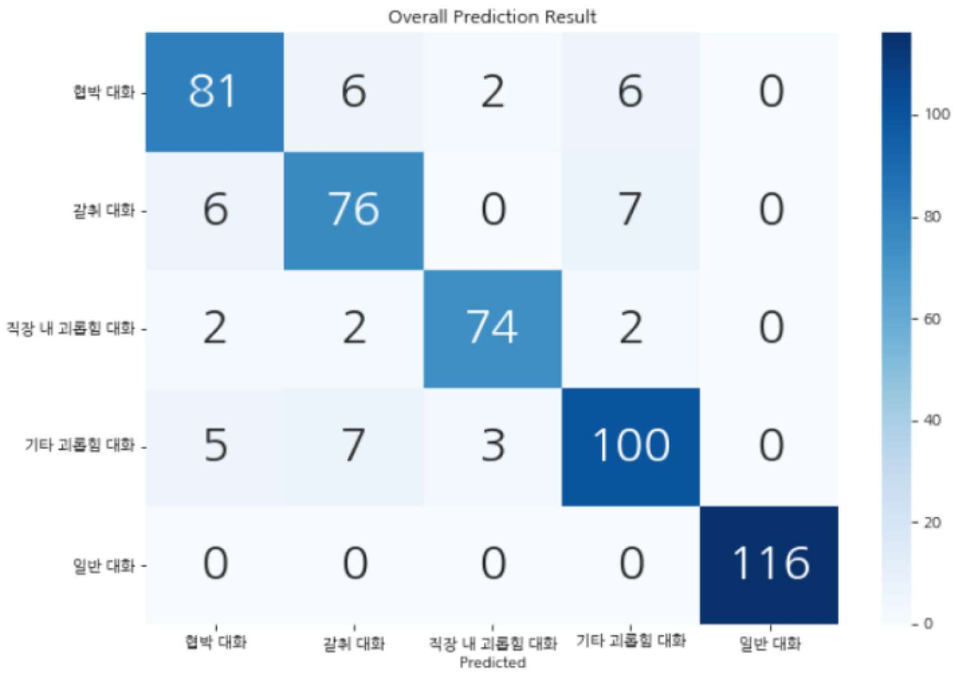
F1-Score 계산하기

wandb에 log으로 추가하기

train()함수 - CompareResult Callback 모델

```python
# epoch이 모두 끝나면 각 클래스별 GT와 예측값을 비교한 표 시각화
# 5x5의 표의 [i,j]의 위치는 실제값은 i번째 클래스이고, 예측값은 j번째 클래스임을 나타냄
class CompareResultsCallback(keras.callbacks.Callback):
    def __init__(self, X_test, y_test, class_num):
        super().__init__()
        self.X_test = X_test
        self.y_test = y_test
        self.class_num = class_num
        # 전체 표 초기화
        self.table = np.zeros((self.class_num, self.class_num), dtype=np.int32)

    def on_train_end(self, epoch, logs=None):
        # 학습 끝났을 때 실행됨
        pred_test = self.model.predict(self.X_test).argmax(axis=1)
        self.y_test = np.array(self.y_test)
        pred_test = np.array(pred_test)

        # 5개의 클래스에서 두 개씩 뽑아내어 비교
        # 실제값이 class_a일 때, 예측값을 claas_b로 예측한 횟수
        for class_a in range(self.class_num):
            for class_b in range(self.class_num):
                num = len(np.where((self.y_test == class_a) & (pred_test == class_b))[0])
                self.table[class_a, class_b] += num

        # 표 그리기
        cr_plot = plot_table(self.table)

        # wandb에 로그로 저장
        cr_image = wandb.Image(cr_plot)
        wandb.log({"Overall Prediction Result": cr_image})
```

실제 클래스

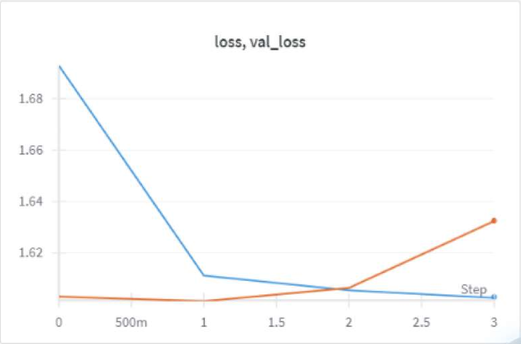


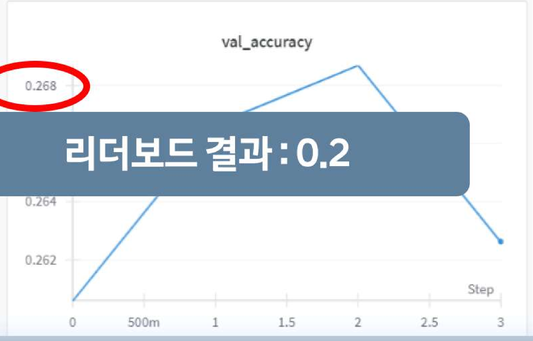
예측된 클래스

BaseLine Model_Hyperparameter Result 모델

best model 파라미터

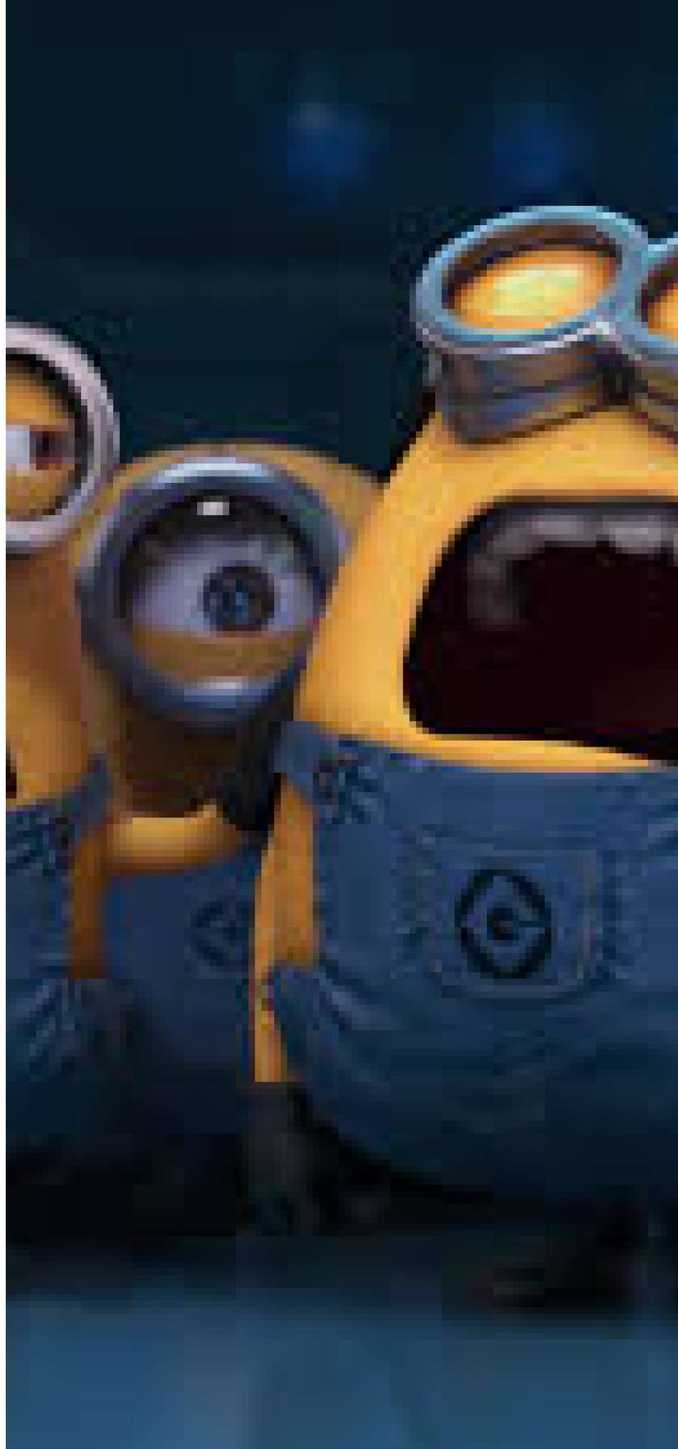
튜닝을 거쳐도 유의미한 성능이 나오지 않음





리더보드 결과 : 0.2

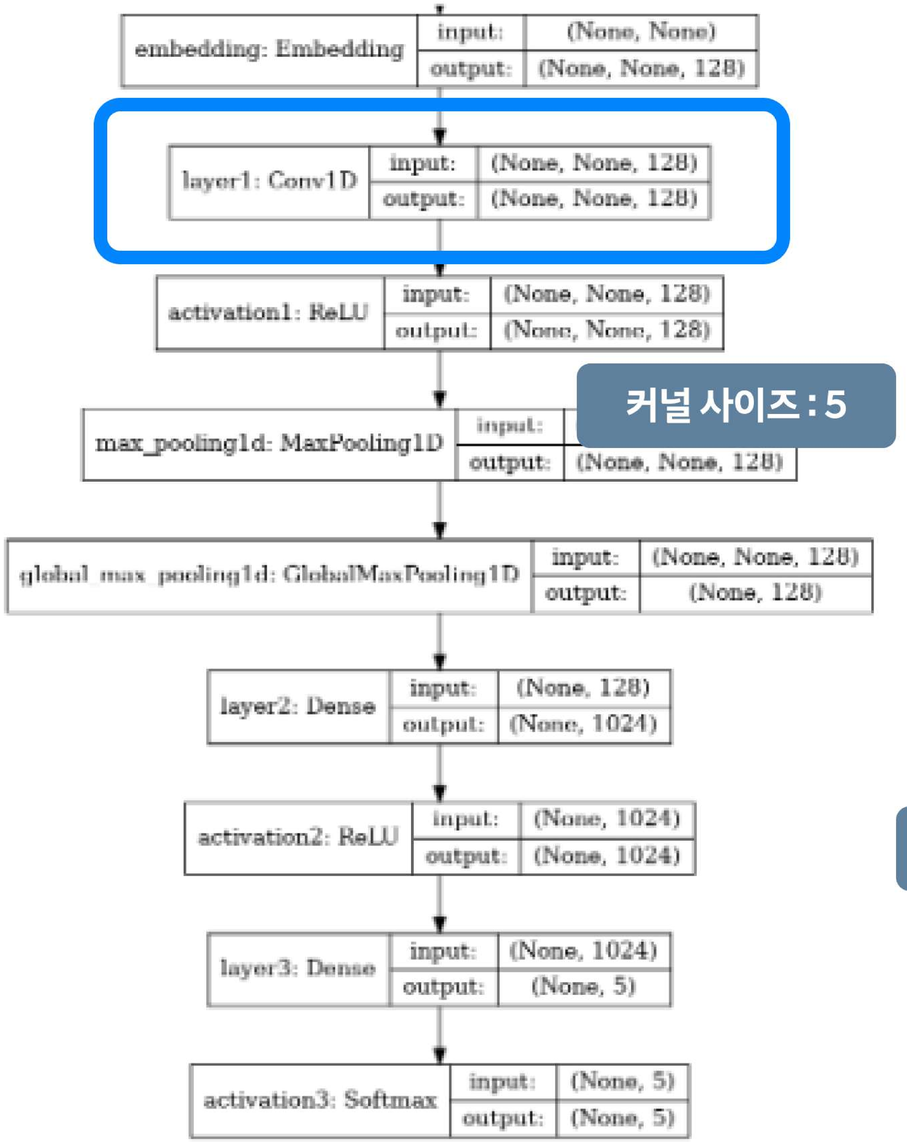
모델의 크기를 키우거나 dropout 등을 추가할 필요 있음



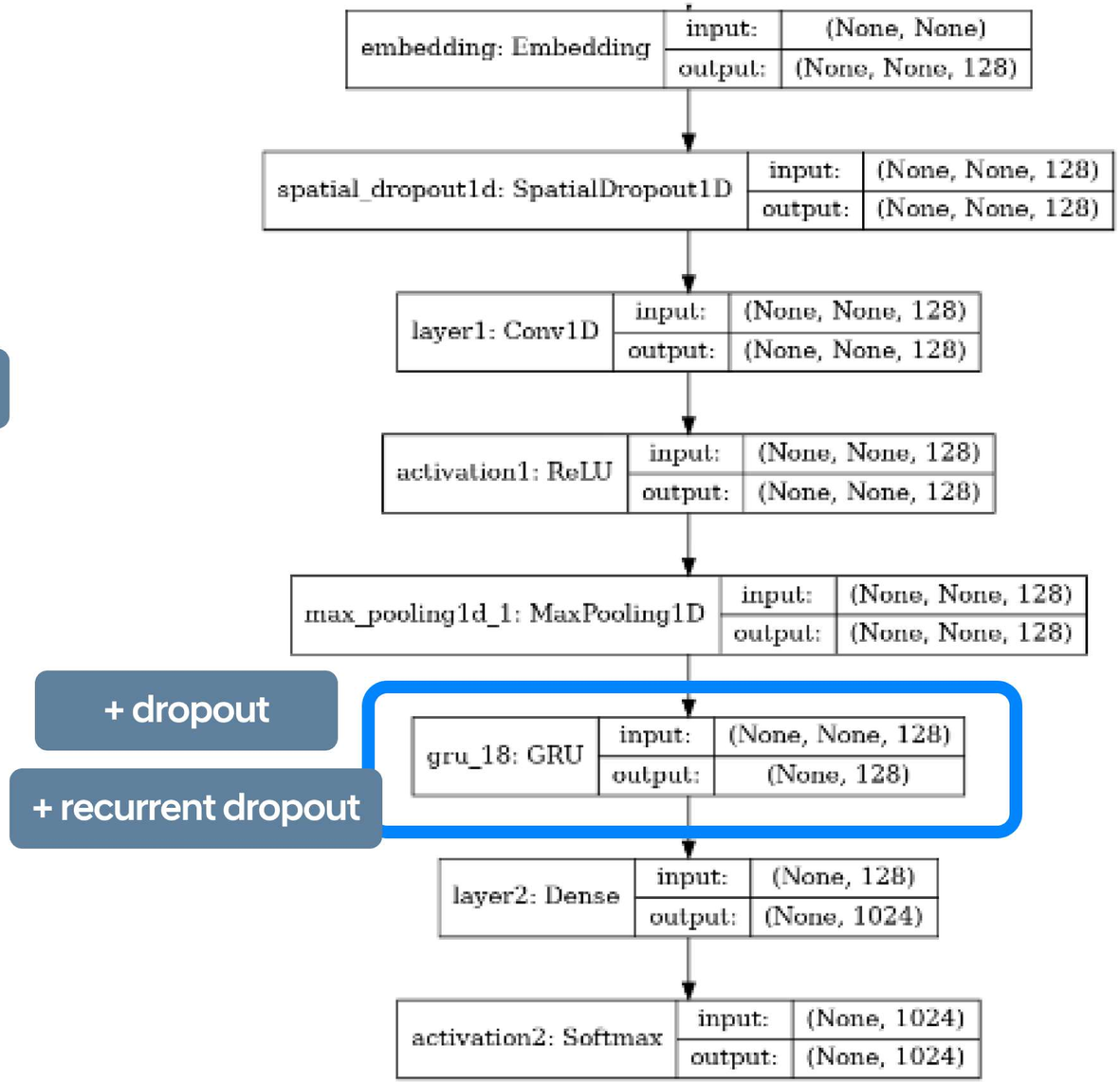
다른 모듈을 추가하거나,
사전학습된 모델을 사용해보자!

Custom Models 모델

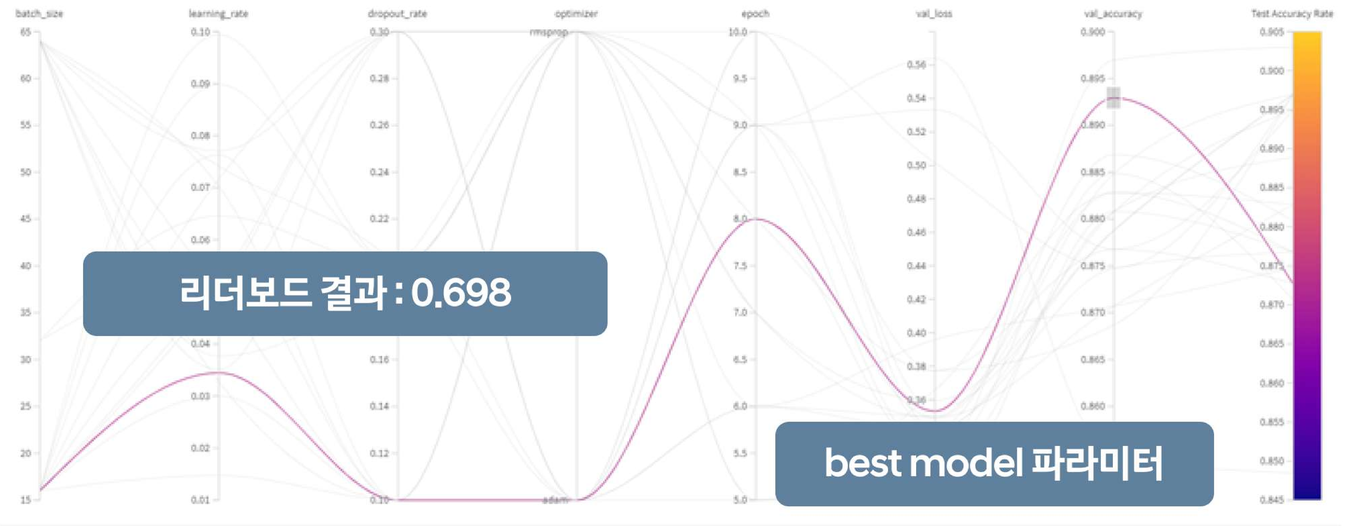
1. 1D CNN

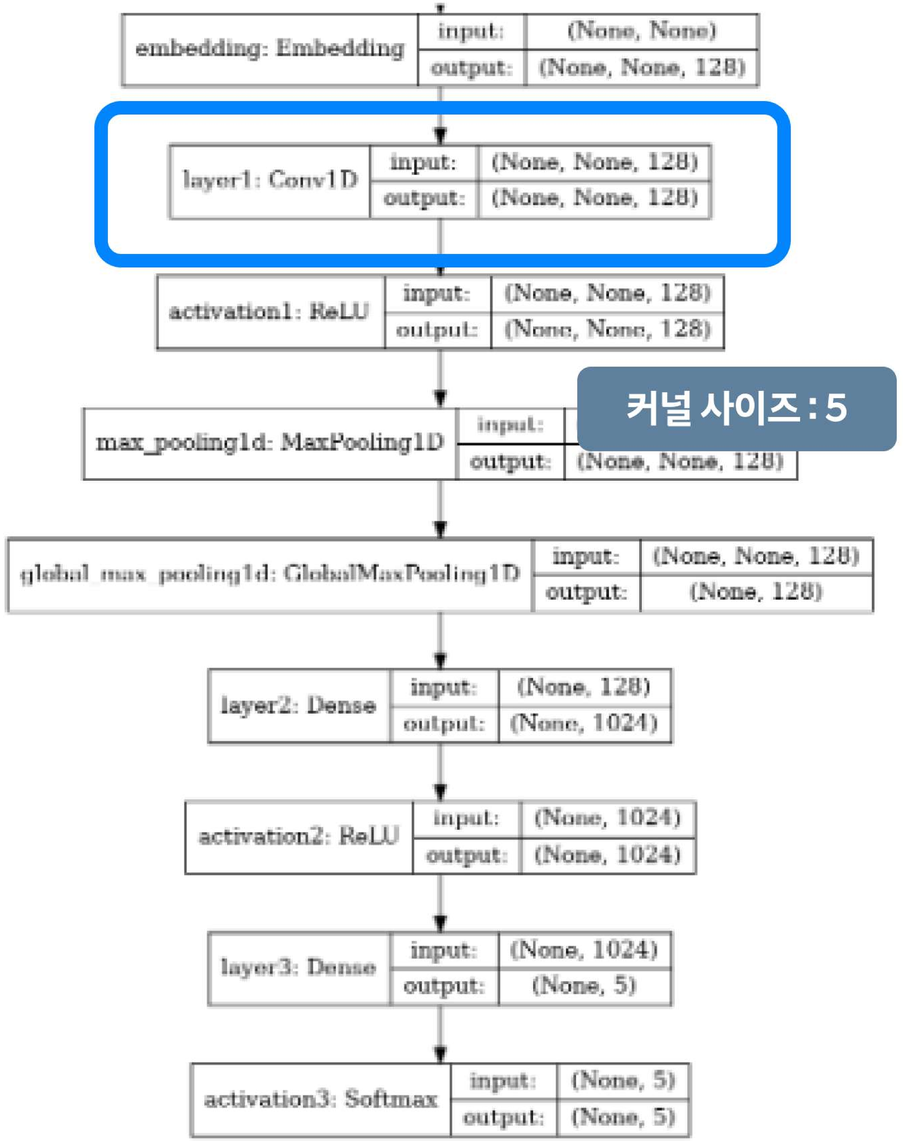


2. 1D CNN+GRU



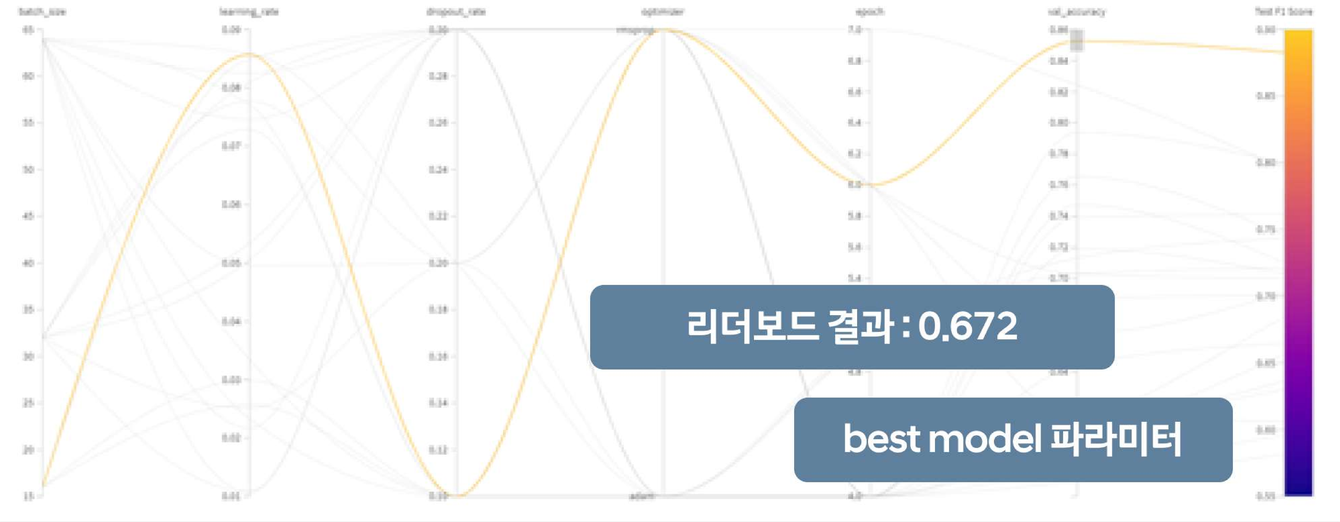
1D CNN Model_Train Result 모델



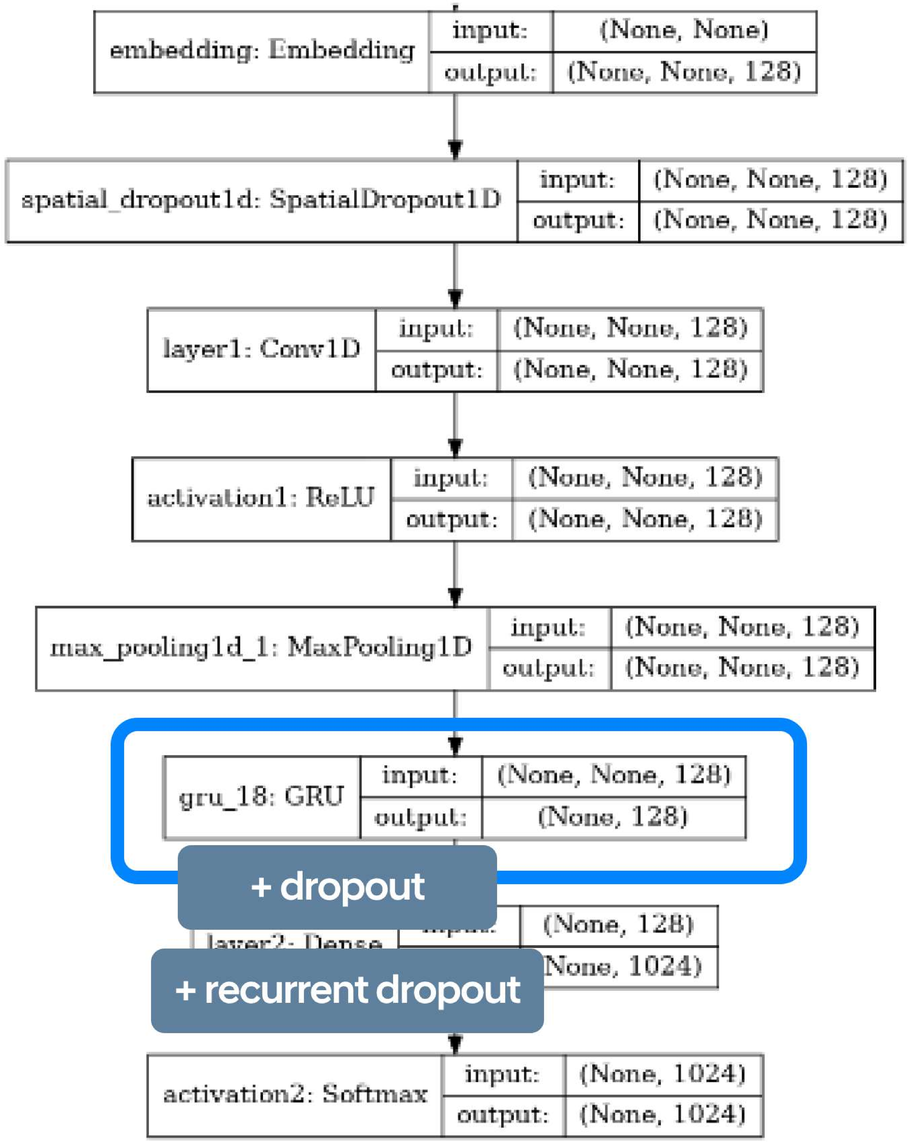


- BaseLine 모델보다 큰 성능 향상이 있었다.
- 의문점 : Test Accuracy나 Test F1 Score가 높은 다른 CNN 모델들보다 현재 모델이 리더보드 상에서 성능이 더 좋게 나왔다.

1D CNN+GRU_Train Result 모델



- BaseLine 모델보다는 나아졌지만 1D CNN 모델보다 성능이 떨어진다.
- 의문점 : 직접 계산한 F1-Score와 리더 보스 상에서의 F1-Score가 많이 차이난다.



Pre-trained Model Result_KoGPT2 모델

KoGPT2

한국어 GPT-2 (SKT-AI)

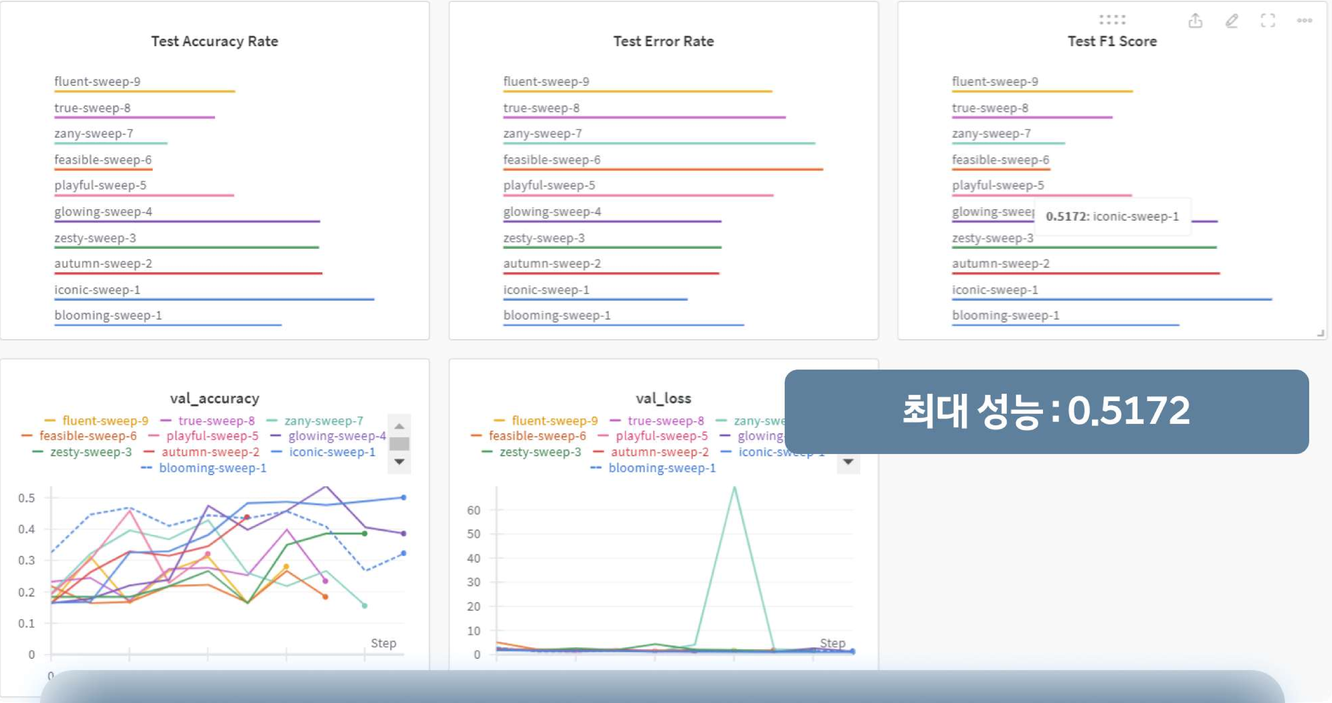
"기존 GPT-2의 특성을 살리고 한국어 성능을 높이기 위한 모델"

Explaination

- 부족한 한국어 성능을 극복하기 위해 40GB 이상의 텍스트로 학습된 한국어 디코더 언어 모델
- 기존 GPT-2의 특성처럼 텍스트의 다음 단어를 잘 예측할 수 있도록 학습된 언어 모델
- 네이버 영화 감성 분석에서 89.1점을 받음.

Model

Model	# of params	Type	# of layers	# of heads	ffn_dim	hidden_dims
kogpt2-base-v2	125M	Decoder	12	12	3072	768



최대 성능 : 0.5172

문장 생성에 특화된 모델이어서 그런지 최대 성능이 낮았다.
결론적으로 이 모델로는 **학습이 불가능하다고 판단**했다.

Pre-trained Model KoBART 모델

Model

Model	# of params	Type	# of layers	# of heads	ffn_dim	hidden_dims
KoBART-base	124M	Encoder	6	16	3072	768
		Decoder	6	16	3072	768

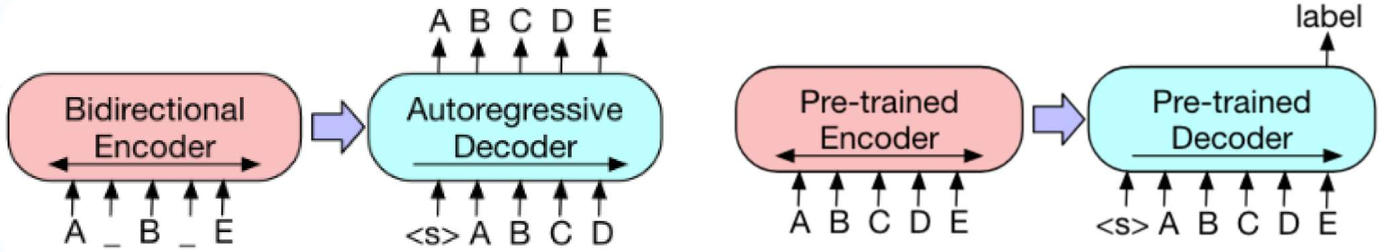
KoBART

KoBART (SKT-AI)

"기존 BART의 특성을 살리고
한국어 성능을 높이기 위한 모델"

Explaination

- BART는 입력 텍스트 일부에 노이즈를 추가하여 이를 다시 원문으로 복구하는 autoencoder의 형태로 학습이 되는 모델
- 기존 BART의 특성을 살려 Text Infilling 노이즈 함수를 사용하여 40GB 이상의 한국어 텍스트에 대해서 학습한 한국어 encoder-decoder 언어 모델
- 네이버 영화 감성 분석에서 90.24점을 받음.




```
tokenizer from pretrained (max_length = 128,
                           vocab_size = 30,000)
Text Classification model =  classifier head (bart_model)
bart_model = from pretrained -> freezed
classifier head = drop_out (linear(num_class)) -> trainable
```

Pre-trained Model KoBART 모델

KoBART

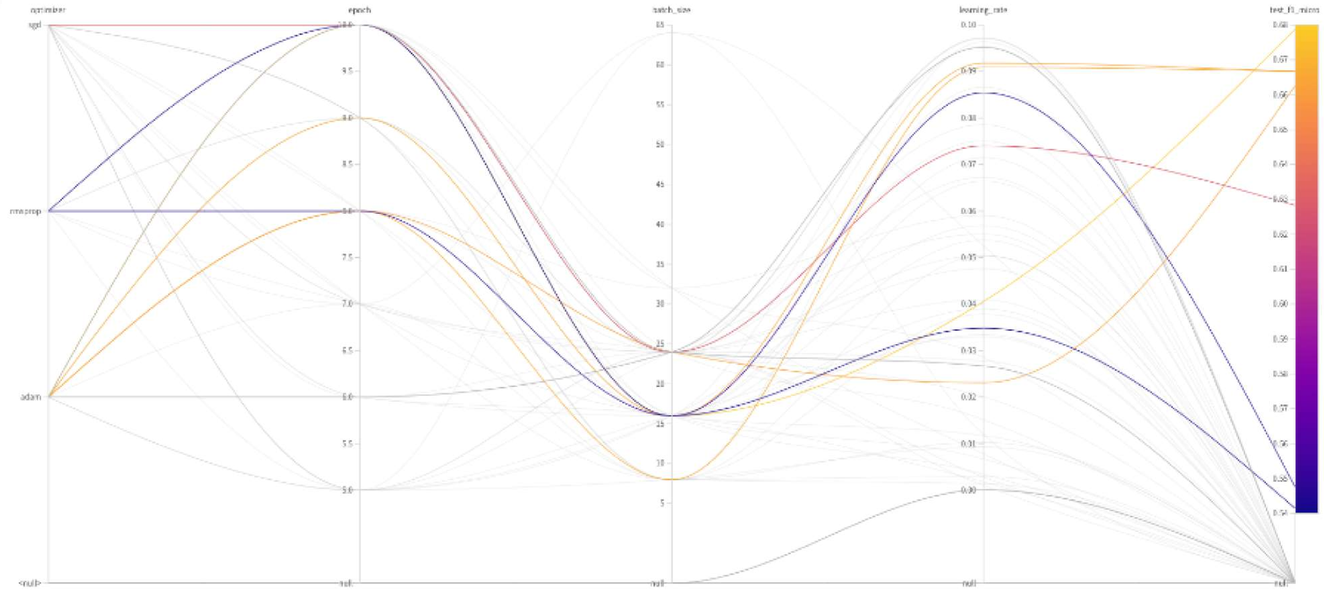
KoBART (SKT-AI)

**"기존 BART의 특성을 살리고
한국어 성능을 높이기 위한 모델"**

max_length = 128
OOM 을 피하기 위해서 max_length 를 줄여서 실험을 진행했다.

Explaination

- BART는 입력 텍스트 일부에 노이즈를 추가하여 이를 다시 원문으로 복구하는 autoencoder의 형태로 학습이 되는 모델
- 기존 BART의 특성을 살려 Text Infilling 노이즈 함수를 사용하여 40GB 이상의 한국어 텍스트에 대해서 학습한 한국어 encoder-decoder 언어 모델
- 네이버 영화 감성 분석에서 90.24점을 받음.



앞서 본 실험에서의 문제점 실험 설계

문제

학습 할 때 확인한 평가지표와 submission에서의 평가지표 결과가 달라 모델 선정에 장애물이 된다.

가설

`test.json`의 데이터셋이 학습한 데이터셋과 많이 다를 수 있다.

예상 원인 실험 설계

- **"t_011":**
  {"text": "아 취업하고 싶은데 누가 나 좀 안 데려가냐 아 그니까 나도 취업하고싶다 코로나때문에 미친거같아 진짜 너무 안 뽑아 그니까 동기들중에 취업한애들 아는애들 있어 뭐 지한이 삼성갔다 그러고 그 여자애 이름 기억 안나는데 카카오 들어갔다던데 씨발 다들 잘갔네 부럽다 그니까 그냥 그럭저럭 학교생활 했는데 쫌 더 열씸히 할껄 아 난 나름 열심히해서 학점도 좋고한데 왜 안되냐 맨날 면접가서 떨어지니까 더 힘드네 너는 면접연습만 하면 될거같은데 나는 서류에서 광탈이야 아 그러게 면접연습 해야되는데 타고난게 말을 잘 못하는데 어쩌라는거야 그니까 공부도 잘해야되고 말도잘해야 취업되는 더러운세상 이휴 부모님한테도 죄송하고 얼른 취업해야되는데"}

예상 원인 1

"생성한 '일반 대화'가 test.json의 '일반 대화'와 다르다"

Explaination

- 일반 대화라고 하더라도 공격적인 언어나 욕설이 포함될 수 있다. (대상이 사람이 아닌 경우).
- 기존 데이터셋은 '일반 대화'가 다른 클래스의 대화와 쉽게 구분되므로 모델이 '일반 대화'에 대해 깊게 학습하지 않았을 수 있다.

구분하기 어려운 '일반 대화'를 생성하도록 하는 프롬프트로 **새로운 '일반 대화' 합성 데이터를 생성**하고, 이를 활용한다.

새로운 장소에서 '일반 대화' 데이터셋을 수집한다.
(기제공된 데이터셋 이용)

예상 원인 실험 설계

예상 원인 2

"test.json의 클래스 간의 불균형이 심하다"

Explaination

- 현재 학습한 데이터는 각 클래스 별 1000개의 데이터를 사용했으므로 균형잡힌 클래스에 대한 예측을 학습한다.
- 하지만 예측한 결과를 봤을 때 '일반 대화' 클래스로 예측한 수가 다른 클래스로 예측한 수보다 현저히 적었다.

1. 1D CNN 모델 예측 결과

[model-solar-sweep-7:v1] predictions:
Counter({1: 120, 2: 121, 3: 140, 0: 116, 4: 3})

2. 1D CNN+GRU 모델 예측 결과

[model-swept-sweep-10:v3] predictions:
Counter({1: 115, 2: 117, 3: 145, 0: 123})

생길 수 있는 의문점 실험 설계

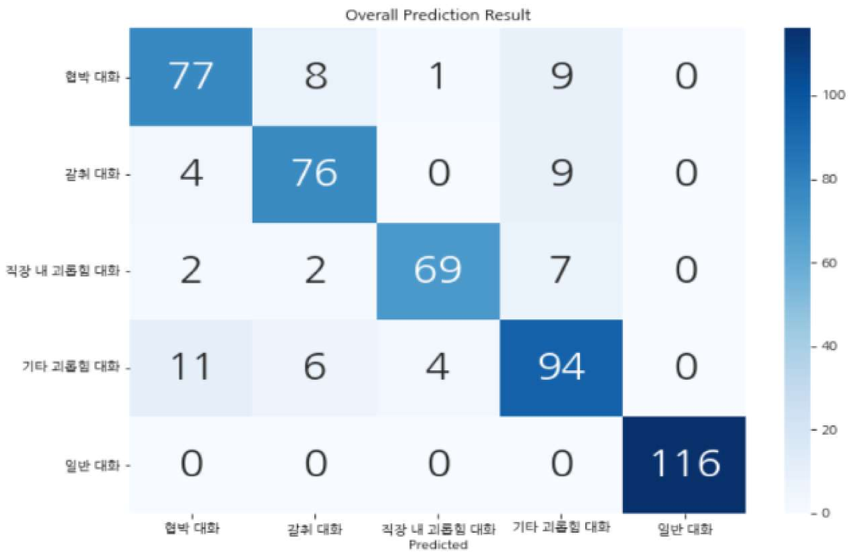
그렇다면 분류기 자체가 클래스 4('일반 대화')를 예측하지 않도록 설계된 것 아닌가요?

모델은 오히려 '일반 대화'를 더 잘 잡을 수 있도록 학습되었기 때문에 분류기 자체가 '일반 대화' 클래스를 예측하지 않는 것이라고는 생각할 수 없다.

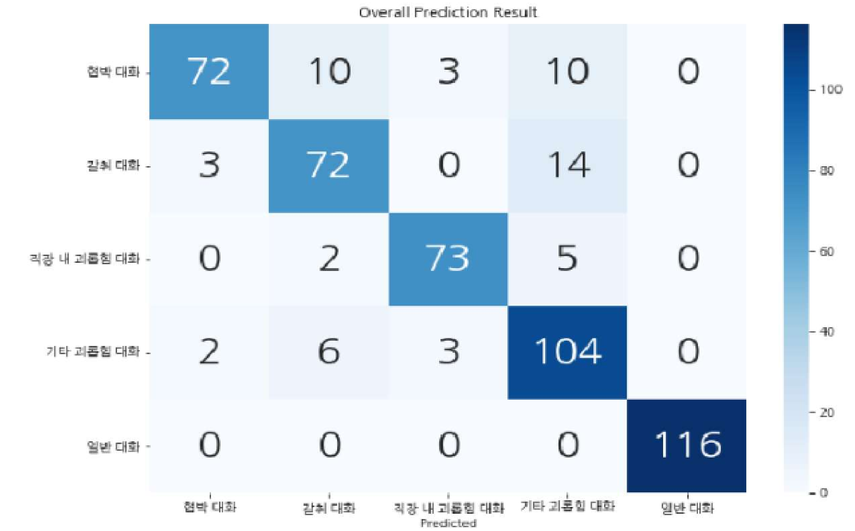
학습 데이터도 '일반 대화' 클래스가 다른 클래스보다 적도록 만들어서 학습 데이터와 추론 데이터의 분포를 비슷하게 만들어보자.

'일반 대화' 클래스의 데이터셋을 줄이는 것보다는 Augmentation 활용하여 다른 클래스의 데이터셋을 늘리는 게 좋다.

1D CNN



1D CNN+GRU



실험 1_New Prompt 실험

idx	class	conversation
3950	일반 대화	이번 주말에 영화 볼래? 좋아, 무슨 영화 볼까? 시간은 언제가 좋아? 저녁 7시 어때? 팝콘은 어떤 맛으로 할까? 나는 버터 맛이 좋아. 예매는 내가 할게. 고마워, 기대된다! 새로 나온 액션 영화 어때? 좋지! 나도 그거 보고 싶었어.
3951	일반 대화	좋아, 저번에 갔던 파스타 집 어때? 거기 정말 맛있었지. 디저트도 먹을까? 응, 티라미수 먹자. 오늘 점심 뭐 먹을래? 나는 파스타 먹고 싶은데. 메뉴는 뭐로 할까? 나는 카르보나라! 나는 봉골레로 할게. 그럼 오늘 점심은 파스타로 결정!
3952	일반 대화	오늘 날씨 정말 좋다! 맞아, 햇볕이 따뜻해서 기분이 좋아. 주말에 산책 갈까? 좋아! 어디로 갈까? 한강 공원 어때? 좋아, 거기서 피크닉도 할 수 있잖아. 내일도 날씨가 좋았으면 좋겠다. 일기예보 보니까 내일도 맑대. 저녁에 비 온대. 아, 우산 챙겨야겠다.
		오늘 날씨 정말 좋다! 맞아, 햇볕이 따뜻해서 기분이 좋아. 저녁에 비 온대.

데이터셋을 잘 생성하지만,

이후 비슷한 문장, 혹은 대화 만을 생성한다

주제를 미리 정해서 데이터셋을 다양화해보면 어떨까?

쇼핑, 공부, 지하철, 날씨 등 약 50개의 주제를 미리 정해주고 데이터셋을 형성해보자.

```
A: 최근에 본 애니메이션 있어?
B: 응, '너의 이름은' 정말 감동적이었어.
A: 맞아, 나도 그거 봤어. 작화가 정말 예술이더라.
B: 스토리도 흥미롭고 눈물 나더라.
A: 다른 추천할 만한 애니 있어?
B: '센과 치히로의 행방불명'도 명작이야.
A: 그거 꼭 봐야겠네. 고마워!
B: 천만에. 같이 애니메이션 얘기하니 좋다.
A: 그러게. 나중에 또 얘기하자
B: 그래, 다음에 또 보자!
```

처음에는 데이터셋을 잘 작성해주지만...

```
A: 다음 여행 계획 있어?
B: 응, 호주 여행 가려고 해.
```

```
A: 어떤 예술 좋아해?
B: 나는 조각을 좋아해.
```

점점 성의 없어지거나..

같은 포맷의 데이터셋만 생성한다..

```
A: 쇼핑에 대해 어떻게 생각해?
B: 쇼핑은(는) 정말 흥미로운 주제야.
A: 맞아, 나도 그렇게 생각해.
B: 쇼핑에 대해 더 이야기해볼까?
A: 좋아, 어떤 점이 궁금해?
B: 모든 것이 궁금해. 더 말해줘.
```

실험 2_기제공된 데이터셋 실험

AI HUB에 있는 `주제 별 텍스트 일상 대화 데이터`

정보 추출 방법

- speaker_type == '1:1'의 경우만 추출했다.
- 화자의 id 정보를 제거하고 두 화자의 발화를 순서대로 '\n' 을 구분자로 연결했다. (`train.csv`와 같은 형태로 만들어주었다.)
- 총 1,005개의 sample을 추출하였다.

리더보드 결과 : 0.232

- 기존에 제공된 데이터셋을 활용해보았다.
- 하지만 BaseLine 모델과 크게 차이 없었고, 성능도 개선되지 않았다.

실험 3_Augmentation 실험

Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks

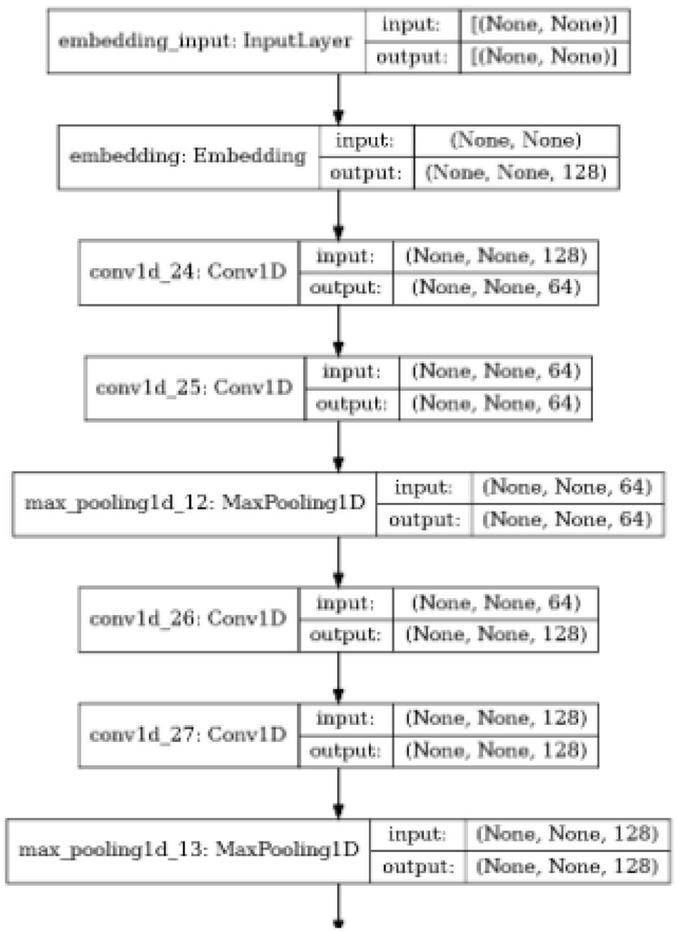
RS, RD 방법을 사용해보자.

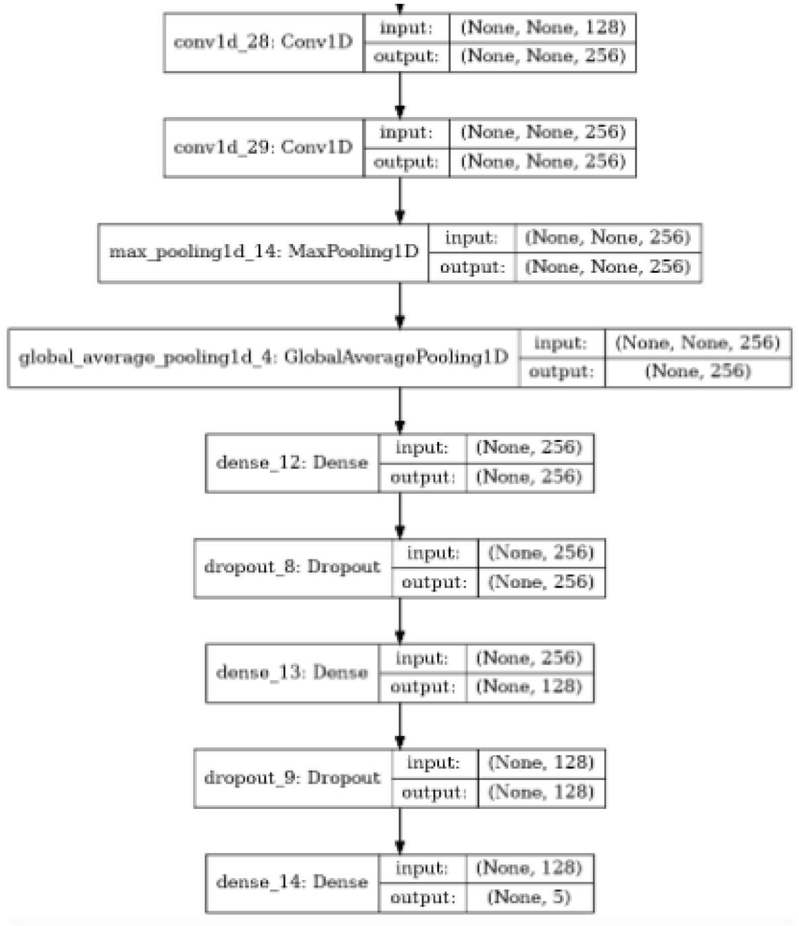
- RS(Random Swap) : 문장 내 임의의 두 단어의 위치를 바꿈
- RD(Random Deletion) : 임의의 단어를 삭제

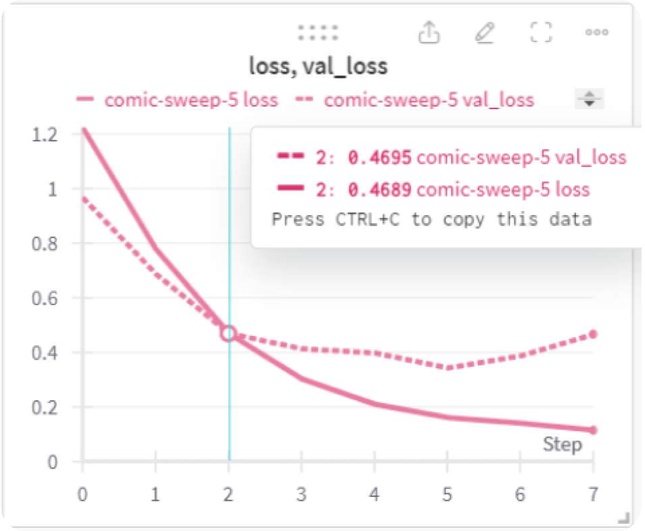
약간의 성능 향상이 있었음.

다만, 오버피팅이 너무 빨리 일어났다.
→ 모델에 드롭아웃을 추가하거나
큰 모델을 쓰면 성능이 더 좋아질 것이다.

기존 1D CNN을 키우고 Dropout Rate를 키우기 실험







리더보드 결과 : 0.646

- 오버피팅은 완화되었다.
- 하지만 기존 모델보다 점수가 낮아졌다.

DKTC (Dataset of Korean Threatening Conversations)

DLthon - MINIONS

감사합니다