**1. 프로세스**

1) Dataset

-Hugging Face의 트렌스포머 모델을 설치.

-네이버 영화리뷰 감정분석 데이터를 다운로드.

-pandas traing set과 testset data를 로드.

-data의 label 추출

-bert tokenizer로 문장을 분리하고 토큰을 숫자 인덱스로 변환

-문장을 LEN 길이에 맞게 자르고 모자란 부분 padding 처리.

-attention 마스크를 초기화 하고 훈련셋과 검증셋으로 분리

2) Model

- 초기화(init) 에서 모델의 환경 및 옵션을 세팅한다.

-포워드(forward) 에서 utterance 에 대하여 tokenize 작업과 [CLS], [SEP] 등 외부 정보를 앞뒤로 붙여준다.

포워드에서 조정된 utterance tokenize 정보를 id(숫자) 로 전환 하는 작업을 수행한다.

즉, text => id convert 작업 으로서 모델에 들어갈 input 데이터 생성 목적이다.

input 데이터를 bert model (neural network) 에 적용한다.

단어의 차원을 768(base 기준)으로 확대하고 해당 차원이 hidden layer 수가 된다.

즉, 5개의 단어가 있다면 (5,768) matrix 구조가 되고 768은 hidden layer 수이다.

단, 옵션에 따라 base => large 변경 할 경우 1024 hidden layer가 될 수 있다.

추가적으로 Hidden layer 정보를 feed forward 선형 결합(fully connected)을 수행 하여 8개의 분류 스코어를 낸다.

선형 결합의 matrix 구조는 (768,8) 가 된다.

1.3. Evaluation Metrics

싸이킷런 측정 도구를 불러온다. 그 중에서 precision, recall 그리고 f1 score 을 평가 기준으로 한다.

1.4. Hyper-parameters

- bert hidden layer 옵션 : base

=> 추가적으로 large 옵션 변경 가능. 단, Model 에서 hidden layer 갯수 변경 해야 함 (768 =>1024)

- 학습률 : 2e-5

- epoch : 5

학습률에 따라 진행 속도가 결정되며, 너무 커도 너무 작아도 모델 생성에 문제가 발생할 수 있다.

epoch 는 모델을 만든 횟수를 의미하며, 여러번 작업을 수행하면서 편향(bias) 된 결과를 지양 하고자 한다.

1.5. Training & evaluation

앞서 설정한 Model 의 정보를 불러온다.

Training을 위한 Cost 함수 선택과 Optimizing 수행하는데 있어서 어떤 옵션(학습률 포함)을 이용할지 선택을 한다.

주어진 내용에서는 Cross-entropy 를 Cost 함수로 적용 하되, weight 등의 추가옵션은 없다.

Optimizing은 Adam(Adagrad + RMSProp) method 이다.

모델을 만들기 위한 기본 옵션이 설정되면, 그 다음부터는 train 데이터를 기반으로 모델을 만들기 위한 훈련(training) 작업을 수행한다.

수행 내용은 train 데이터 중 utterance input 에 대하여 8개의 emotion 정보를 target (Ground Truth)으로 세팅해 주고, Cost 최소화를 목적으로 forward, backward 작업을 하면서 모델을 만들어 나간다.

모델의 검증은 dev 데이터를 가지고 한다. dev 파일의 utterance 정보를 입력해서 모델링 적용 후 나온 결과값과 dev 파일의 emotion 값(Ground Truth)을 비교하여 recall, precision을 생성한다. 이를 바탕으로 epoch 마다 f1 score 를 계산하고 모델에 대한 기본적인 검증 수치를 확인한다.

해당 작업은 하이퍼 파라미터에서 제공한 epoch 수에 따라 반복작업을 수행하게 된다.

1.6. Labeling

test 데이터를 앞서 나온 모델에 적용한다.

각 utterance 정보에 대하여 8개 감성 정보 중 하나의 결과 값이 나올 것이고, 해당 데이터는 파일에 남기게 된다.

추후 해당 Output 데이터는 실질 결과와 비교하여 모델의 적합성을 최종 판단하게 된다.

1.7. Proposal

- There is a class imbalance problem. (Use weighted cross-entropy etc.)

- Our model takes a single sentence. (Make it grasp its context as well.)

- Our model does not consider speaker information. (Make it consider the info.)

- Batch size is set as 1. (Increase the batch size.)

**2. 속도 개선을 목표로 한 프로그램 수정**

Proposal 을 포함하여, 옵션의 변경 및 추가, 데이터 확인 등의 작업을 수행하면서 개선점을 파악하고자 한다.

작업 수행 시, epoch 를 1로 두어서 f1 score 를 검증하면서 코드 조정을 하였다.

2.1. 데이터 분석 (W/ pandas )

데이터 전처리 수행 전, 별도의 데이터 Data set 를 생성하여 Download 받은 데이터를 분석해 본다.

- list data naming : data\_anal (Compare w/ “data”)

=> 별도의 파일명 생성

- data\_anal 의 pandas load(3 Cases) : data\_anal\_pd\_train / data\_anal\_pd\_dev / data\_anal\_pd\_test

=> 편의성을 부여하고자 pandas app. 활용 함.

각각의 데이터 건수는 아래와 같다.

- data\_anal\_pd\_train : 총 10561 건

=> train 데이터 : 모델을 만들기 위한 데이터

- data\_anal\_pd\_dev : 총 1178 건

=> dev 데이터 : 모델을 검증하기 위한 데이터

- data\_anal\_pd\_test : 총 2764 건

=> test 데이터 : 모델을 적합성을 확인하기 위한 데이터

1) data\_anal\_pd\_train

총 10561 건의 데이터가 존재하며, 264명의 speaker 가 존재한다.

264명의 speaker 중에서 주인공(hero) 6명 대화 개수가 80% 이상 이다.

emotion 건수에서는 neutral이 절반에 육박하며 앞도적인 1등 이다.

non-neutral / joy / surprise / anger... 순으로 구성되어 있으며, disgust 및 fear 등 부정적인 감정이 하위권을 형성한다.

2) data\_anal\_pd\_dev

speaker 발생 빈도 순위가 train 데이터와 일부 차이가 있다.

예를 들어, Joey(speaker 중 한명)가 다른 데이터 셋(train,test)에서는 1순위에올랐지만, 해당 데이터셋(dev)에서는 4위에 위치하고 있다.

emotion 에 대한 분포도는 최상위권인 neutral, non-neutral, joy 순서는 동일하나, 부정적인 의미의 하위권에 대해서는 일부 순위 차이가 있다.

3) data\_anal\_pd\_test

data\_anal\_pd\_train 데이터와 speaker 와 emotion 분포도와 유사하다.

2.2. proposal 3th line - speaker info.

대화 총 건수 중에서 주요 인물이 차지하는 비중은 80~90% 이다.

주요 인물은 Rachel, Monica, Phoebe, Chandler, Joey, Loss 이고, 그 외 나머지 약 250명의 인물들이 10~20% 를 구성한다.

주요 인물 캐릭터에 대해 인터넷 조사를 하였다. (도메인 분석)

※ 참고 : <https://namu.wiki/w/%ED%94%84%EB%A0%8C%EC%A6%88>

- Rachel : 카페 웨이트리스로 일하다가 블루밍데일 백화점으로 옮긴 후 지금은 의류회사인 랄프로렌에서 일하는 직장인

- Monica : 모니카는 깔끔과 질서 정연에 강박 관념을 가지고 있는 그 누구에게도 지기 싫어하는 요리사

- Phoebe : 피비는 낙천적이며 엉뚱한 면이 많은 포크 싱어 겸 안마 치료사

- Chandler : 챈들러는 조금은 잘난 체하지만 말을 잘하여 마음이 여린 정보처리사(데이터 프로세서)

- Joey : 여자와 스포츠, 뉴욕, 그리고 그 무엇보다 자기 자신을 너무나 사랑하는 약간은 지적인 면이 부족한 배우 겸 모델

- Loss : 로스는 꼼꼼한 성격에 수줍음이 많고 마음이 약한 고생물학자. 여동생 모니카와 정반대 성격을 지니고 있음

위 주요 인물에 대해서 utterance 정보를 수정 또는 조정하였다.

주요 인물에 대해서는 주인공(hero)을 표기 하였고, 성별 정보 (male/female) 를 포함한다. 위 정보 이외에는 etc 정보를 넣어서 구분한다.

해당 정보를 포함하기 전과 후의 f1 score 차이가 일부 있었다.

- 포함 전 : 0.579796

- 포함 후 : 0.580645

일단 해당 정보를 추가할 경우, score 변동이 크게 떨어지지 않은 상황이므로, 추가 정보를 보완해서 데이터 셋 구성을 한다.

2.3. 코드 수정 보완 - 코딩 미스 보완

- 변경 전 : ids = [tokenizer.convert\_tokens\_to\_ids(tokens)] # (bat=1, len)

- 변경 후 : ids = [self.bert\_tokenizer.convert\_tokens\_to\_ids(tokens)] # (bat=1, len)

변경 전 tokenizer 선언이 tutorial 에서 나온 것을 참조해서 사용 (잘못된 부분 캐치) 되었다.

tutorial 내용을 제거하면서 모델 내 초기화(init)선언한 것으로 읽을 수 있게 프로그램을 변경 하였다.

2.4. Cost Function 변경

The negative log likelihood loss 함수를 적용해 보았다.

분류학습기에 효율적인지 판단하기 위함이다.

※ 참고 : https://pytorch.org/docs/master/nn.html?highlight=nll%20loss#torch.nn.NLLLoss

1) CrossEntropyLoss => micro\_f1: 0.580645

2) NLLLoss => micro\_f1: 0.416808

Cost function 을 변경했지만, score 가 떨어졌기에, 기존 Cross Entropy 함수를 계속 사용한다.

단, CrossEngropyLoss 를 사용하되 Labeling 정보에 가중치(weight)를 부여해 보고자 한다.

2.5. Cost Function Labeling Weight 부여

※ 참조 : https://discuss.pytorch.org/t/passing-the-weights-to-crossentropyloss-correctly/14731

# emotion 코드 : 코드 의미 => 테스트 데이터 건수

# 0 : 'disgust’ => 240

# 1 : 'surprise' => 1220

# 2 : 'fear' => 185

# 3 : 'non-neutral' => 2017

# 4 : 'joy' => 1283

# 5 : 'sadness' => 351

# 6 : 'anger' => 513

# 7 : 'neutral' => 4752

- 8개의 label 에 대하여 weight 를 적용해 본다. (=> micro\_f1)

0) [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] => 0.128183

1) [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0] => 0.181664

2) [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0] => 0.072156

3) [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 0.024618

4) [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0] => 0.128183

5) [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0] => 0.104414

6) [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] => 0.072156

7) [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0] => 0.416808

a) [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0] => 0.580645

b) [2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0] => 0.584041

c) [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5] => 0.571307

d) [1.0, 1.0, 0.5, 0.5, 1.0, 1.0, 0.5, 1.0] => 0.566214

e) [0.5, 0.5, 1.0, 1.0, 0.5, 0.5, 1.0, 0.5] => 0.572156

0)~7) 정보는 labeling 각각의 f1 score 이다. (Idea : one-hot encoding)

a)~c) 정보는 건수에 따라 파라미터 일부를 조정한 후 f1 score 이다.

가중치(weight) 적용에 따른 여러 작업을 수행해 보았지만, f1 score 에 큰 영향이 없는 것으로 판단 된다

각각의 분류에 대한 가중치 부여(나머지는 미부여) 작업을 수행하였고, 가중치를 2개, 1/2배 해 보았지만, 큰 차이가 없음을 알 수 있었다.

일단 weight 옵션을 주었을때 여부에 대해서 큰 차이가 없기에, 옵션을 제거하고 작업을 수행하도록 한다.

2.6. hidden layer 옵션 변경

base 로 설정된 옵션에 대하여 large 로 변경하고자 한다. hidden layer 개수 증가에 따라 score 부분을 확인을 해 보고자 한다.

수정 부분은 model 내 768 값을 1024로 수정하였다. 또한 pretrained\_weights 옵션에 대해서 bert-large-uncased 로 수정 하였다.

※ 참고 : <https://github.com/google-research/bert>

Hidden layer 증가에 따라 작업 속도가 급격히 증가했다.

기존 base 옵션에서 1 epoch 당 10분 정도 걸렸다면, large 옵션에서는 20분 정도로 약 2배만큼의 시간이 소요되었다.

f1 score 는 0.580645 로 나오며, bert-base-uncased와 비교하여 큰 차이는 없었다.

2.7. epoch 수치 조정 ( 1 => 3) 최종 기동

bert-base-uncased 옵션으로 하여epoch 수치를 3으로 조정하였다.

test 데이터의 utterance 정보에 대한 emotion labeling 생성 목적이다.

해당 작업을 수행 하여, labeled.json 생성하였고, 과제로 제출한다.