# 2022 년 k-ium 의료인공지능경진대회 개발 문서

팀명 : 문학산
소속 : 가천대학교 의용생체공학과
팀원 : 박창선, 신가영, 이성재, 이유빈, 정민규

## 1. 전처리

### 가. 중복 데이터 제거

1) 1 차 train dataset 에서 중복되는 데이터를 삭제함. 해당 파일은 압축 파일의 data 폴더에 있는 train_set_no_duplicates.csv 임.

2) *팀 문학산*은 'conclusion'이 가장 주요한 정보를 담고 있다고 판단하여 'Findings'는 전부 삭제하고 'conclusion'의 데이터만 사용함.

3) 중복 제거는 학습 시에만 사용하고, inference 시에는 사용하지 않음.

### 나. 문장 전처리

1) 문장 전체를 소문자화하고 한글을 제거함.

2) 동일한 단어가 다른 벡터로 임베딩되는 것을 방지하기 위해 빈도 수가 높은 단어 중 뜻은 동일하지만 '.'이 포함되거나 줄임말이 사용된 등의 경우에는 동일한 단어로 처리함.

### 다. 단어 전처리

1) 불용어 처리

NLTK 모듈을 이용하여 불용어 처리를 진행함. 큰 의미가 없는 단어를 제거해주는 작업임.

- from nltk.corpus import stopwords

2) 토크나이저

토크나이저는 nltk 모듈의 TreebankWordTokenizer 를 사용함. TreebankWordTokenizer 는 hyphen(-)으로 구성된 단어는 하나로 유지하고 apostrophe(')로 접어가 함께하는 단어는 서로 분리하는 규칙을 가짐. 이런 규칙이 의료데이터에 적합할 것으로 판단하였음.

3) 문자 벡터화

문자를 벡터화 할 때는 keras 에서 제공하는 TextVectorization layer 를 사용함. 이를 이용하여 데이터의 문자를 벡터화 함.

**라. 단어 임베딩**

1) Pre-trained word embedding 사용

가) 사용 목적

케라스(Keras)에서 제공하는 Embedding 레이어를 사용하여 처음부터 임베딩 벡터를 학습할 수도 있음. 그러나 훈련 데이터의 양이 많지 않은 경우 처음부터 학습하여 최적화된 임베딩 벡터를 얻는 것은 쉽지 않음. 대신 규모가 큰 훈련 데이터로 사전 훈련된 임베딩 벡터(pre-trained word embedding)를 사용하면 적은 양의 데이터를 가지고 있는 상황에서도 높은 성능을 기대할 수 있음.

나) Embedding model 선정

경진 대회에서 제공한 허혈성 뇌졸중 판독문은 의료 데이터셋에 해당함. 따라서 의료 데이터셋으로 사전 훈련된 임베딩 모델을 선택하는 것이 합리적임. 이를 위해 *팀 문학산*은 의학, 간호학, 치의학 등 생의학 관련 의료서지정보 데이터베이스 서비스를 제공하고 있는 *'PubMed'*와 4,000 명 이상의 환자 데이터로 구성된 오픈 데이터셋인 *'MIMIC III Clinical notes'*로 학습된 단어 임베딩 모델 *BioWordVec*[1]을 선택함.

다) 사용 방법

[1] https://github.com/ncbi-nlp/BioSentVec

*BioWordVec* 에 일치하는 단어가 있는 경우 사전 훈련된 단어 벡터를 사용함. 일치하는 단어가 없는 경우 Facebook 에서 개발한 텍스트 임베딩 기법인 내장된 fastText 모델을 활용함. fastText 는 훈련이 완료된 후 데이터셋의 모든 단어의 각 n-gram 에 대해서도 워드 임베딩이 가능함. 따라서 일치하는 단어가 없는 경우(Out of Vocabulary, OOV)에도 대처가 가능하다는 장점이 있음.

## 2. 모델

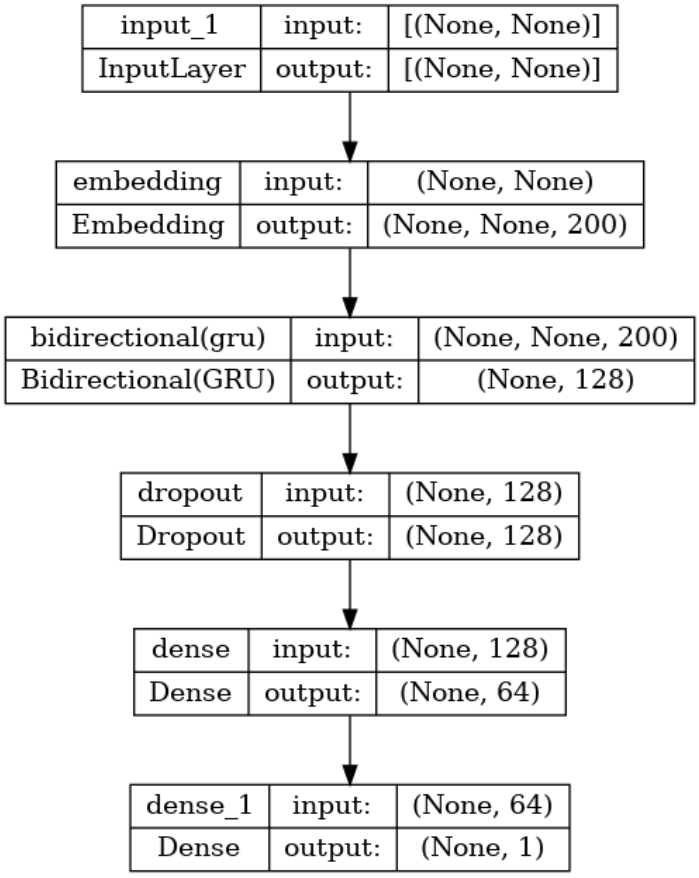


**그림 1 모델의 구조**

### 가. Layer

1) input_1 : 입력 층

2) embedding

임베딩 층. 사전 훈련된 단어 임베딩 모델인 BioWordVec 을 활용함. 과적합을 막기 위해 문서에서 많이 등장한 상위 25000 개의 단어만 사용함. 각 단어는 200 차원으로 표현됨.

3) bidirectional

unit 수가 128 개인 GRU layer 를 사용함. LSTM, GRU 등은 입력 순서를 시간 순대로 입력하기 때문에 결과물이 직전 패턴을 기반으로 수렴하는 경향을 보인다는 한계가 있음. 기존 GRU 계층에 역방향으로 처리하는 GRU 층을 추가함으로써 보다 다양한 학습을 기대할 수 있음.

4) Dropout

모델 학습 중 과적합(overfitting)을 막기 위한 layer. rate=0.3 으로 설정됨. 모델 학습시에만 활성화되며, inference 시에는 동작하지 않음.

5) dense, dense_1

bidirectional GRU 를 통해 나온 128 차원의 벡터를 최종 목표인 뇌졸중 유무를 판단하기 위한 완전연결층. 첫번째 dense 층은 64 개의 노드로 이뤄져 있고, 활성화 함수로 ReLU(Rectified Linear Unit) 사용. 마지막 dense 층은 뇌졸중 유무를 판단하기 위한 층으로, 활성화 함수로 sigmoid 사용.

**나. 학습 관련 세부 사항**

1) 전처리된 train_set_no_duplicates.csv 전체로 학습 진행.

2) 12 번째 epoch 의 weight 을 사용.

3) Adam optimizer 사용.

4) learning rate=0.001 사용.

5) loss function 으로 binary cross entropy 사용.

## 3. 결과

| | *Precision* | *Recall* | *F1-score* | *support* |
|---|---|---|---|---|
| *0* | 0.993013 | 0.996289 | 0.994648 | 2425 |
| *1* | 0.959091 | 0.925439 | 0.941964 | 228 |
| *Accuracy* | | | 0.990200 | 2653 |
| *Macro avg* | 0.976052 | 0.960084 | 0.968306 | 2653 |
| *Weighted avg* | 0.990097 | 0.990200 | 0.990120 | 2653 |

**표 1 classification report**

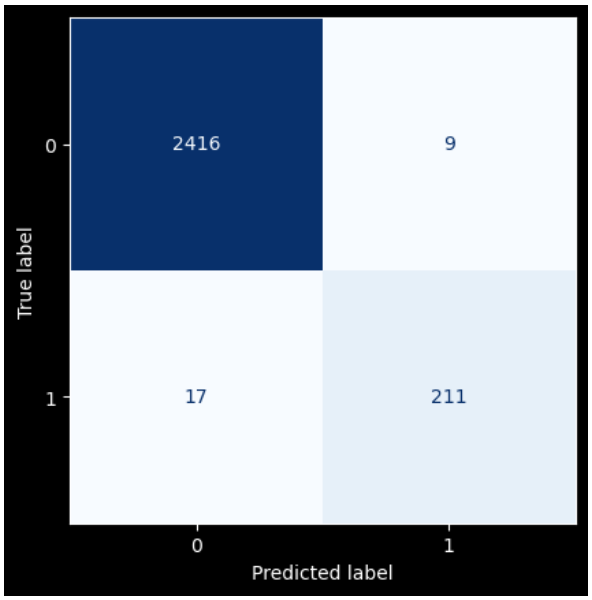


**그림 2 confusion matrix**

º 표 1(classification report)과 그림 2(confusion matrix)는 scikit-learn 라이브러리에서 제공하는 metrics.classification_report, confusion_matrix 를 이용하여 계산함.

### 가. C-statistics: 0.960864

1) AUROC 결과는 scikit-learn 라이브러리에서 제공하는 metrics.roc_auc_score 함수를 이용하여 계산함.

2) 2 차 validation dataset 에 대한 모델 auroc 결과는 마찬가지로 제출한 압축 파일에 'c-statistics.txt'로 저장되어 있음. 1 차 제출 때와는 다르게 python bi-gru.py 명령어를 입력했을 때 terminal 에 c-statistics 가 출력되지 않으므로 cat c-statistics.txt 명령어를 입력하여 auroc 결과를 확인해야 함.

### 나. 출력 txt file

제출한 압축 파일의 경로에서 python bi-gru.py > output.txt 명령어를 terminal 에 입력하면 해당 폴더로 output.txt 파일이 만들어지고, predict 결과가 저장됨. 현재는 폴더에 output.txt 파일이 저장되어 있는 상태임. bracket 이 없는, 조금 더 보기 편한 결과는 폴더에 result.txt 로 저장돼 있음.

## 4. 분석 및 소감

### 가. 1,2 차 결과 분석

1) 1 차 train 에서 0.996 의 성능을 보였으나 2 차 test 에서는 이에 살짝 못미치는 0.96 의 성능을 보임.

2) 일단, 1 차 C-statistics 가 0.996 을 보인 것은 1 차 제공 데이터 전체로 train 과 prediction 을 모두 시행했기 때문으로 보임.

3) 1 차 모델 평가 과정에서 오답 항목 분석 시 오답 간에 서로 연관이 없었으며, 뇌졸중 판독문을 보고 왜 뇌졸중으로 판독이 되었는지 팀 내에서 판단하기에는 의학 지식적 한계가 있었음.

4) 위 과정에서 해결하지 못한 confusion matrix 의 False 항목들이 2 차 데이터에서 추가된 것이 2 차 test 에서의 성능 저하 원인으로 판단됨.

5) 잘못된 임베딩의 가능성

   대부분의 의학용어들은 줄임말이 많음.(ex: DAI: Diffusion Axonal Injury) 전처리 과정에서 대문자를 소문자로 바꾸면서 *BioWordVec* 의 임베딩을 적용하지 못한 사례가 있을 가능성이 있음.

   전처리 과정의 오류로 2 차 data 의 오답을 확인했을 때 전반적으로 left 뒤에 오는 단어들이 띄어쓰기 없이 합쳐짐.(ex: left temporal -> lefttemporal) 같은 단어임에도 불구하고 다른 단어로 임베딩 됐을 가능성이 있음.

### 나. 소감

- 2 차 오답 분석을 진행하면서 전처리 과정의 미흡함을 발견해서 아쉬움. 전처리 과정을 꼼꼼히 확인해서 정교한 임베딩을 했으면 성능이 올라갔을 것이라고 예상함.

- 모델의 성능을 최대치로 끌어올리기 위해 다양한 모델과 전처리 방법을 시도하고, fine-tuning 도 끊임없이 진행하며 책으로는 배울 수 없는 많은 것을 배웠음. 비록 1 차 결과에 비해 아쉬운 2 차 결과이지만 이 대회를 진행하는 과정에서 배운 것들은 잊지 않고 발전의 발판으로 삼겠음.

- 1 차에 비해 아쉬운 2 차 결과를 통해 epoch 선정에서 조금 더 좋은 판단을 할 수 있지 않았을까 하는 아쉬움이 남음.

- 1 차 훈련을 진행할 당시 'Findings'는 환자의 초기 증상 및 촬영한 영상 종류에 대한 정보를 담고 있고 'conclusion'은 그에 대한 의사의 종합 소견이라고 생각함. 그리하여 'conclusion' 데이터만으로도 충분히 높은 성능을 낼 수 있을 거라 판단, 해당 데이터셋만으로 학습을 진행함. 'Findings'와 'conclusion'을 모두 이용하여 모델을 훈련하게 되면 오히려 방해가 될 거라고 판단함. 2 차 검증 데이터셋으로 모델을 테스트하면서, 모델이 중요한 feature 를 직접 선택할 수 있도록 전체 데이터셋을 사용했다면 보다 좋은 성능을 낼 수 있지 않았을까 생각함.

- 최근 자연어 처리 분야에서 높은 활약을 하고 있는 트랜스포머 계열의 모델에 대해 공부하고, 추후에 열릴 2 회 경진대회에서 적용해볼 예정임.