CMPE493 - Introduction to Information Retrieval
Term Project Presentation

Text Classification for Covid19 Scientific Literature

Emin Güre	2020700252
Ozan Kılıç	2016400144
Barış Mutlu	2016400192

The tools used

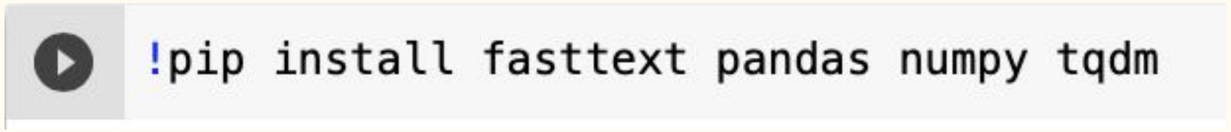
fastText

Library for efficient text classification and representation learning

Colaboratory

BioBert & SciBert

Libraries we used



```
!pip install fasttext pandas numpy tqdm
```

NLTK

Preprocessing Function

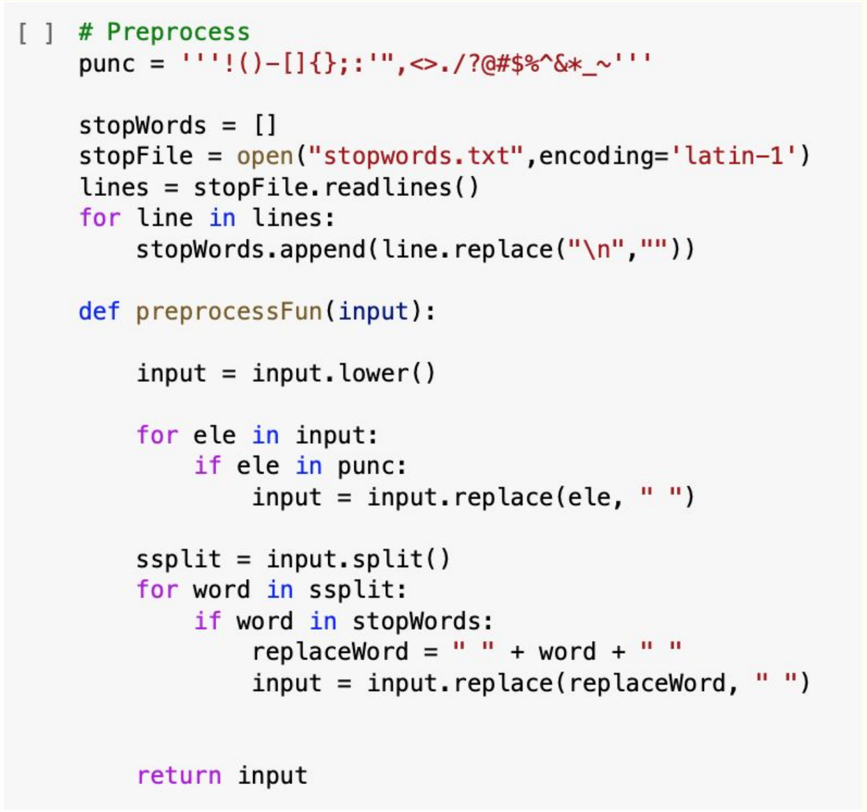
Takes raw string and returns processed string

-Taking columns

-Making words lower case

-Clearing punctuations

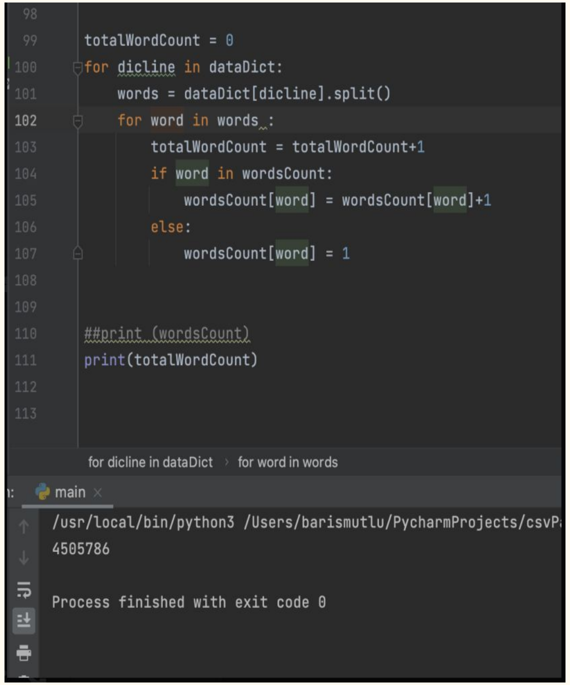
-Clearing stopwords



```
[ ] # Preprocess
    punc = '''!()-[]{};:'",<>./?@#$%^&*_~'''

    stopWords = []
    stopFile = open("stopwords.txt",encoding='latin-1')
    lines = stopFile.readlines()
    for line in lines:
        stopWords.append(line.replace("\n",""))

    def preprocessFun(input):

        input = input.lower()

        for ele in input:
            if ele in punc:
                input = input.replace(ele, " ")

        ssplit = input.split()
        for word in ssplit:
            if word in stopWords:
                replaceWord = " " + word + " "
                input = input.replace(replaceWord, " ")


        return input
```

Preprocessing Results

Total Word Count :



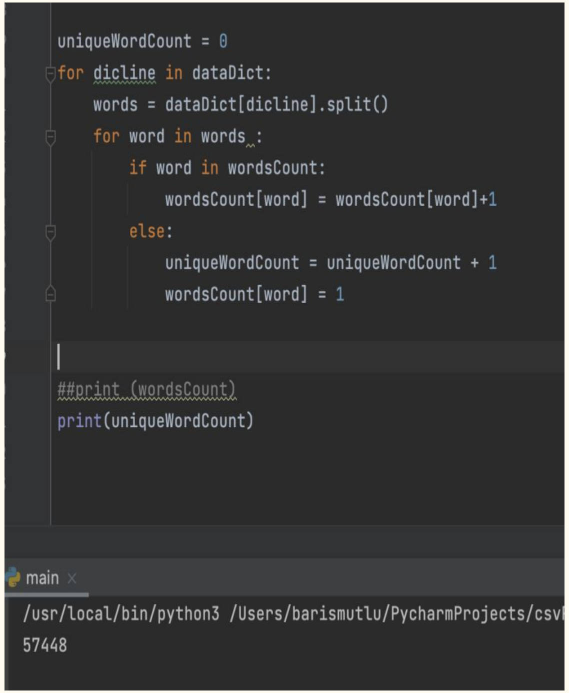
```
totalWordCount = 0
for dicline in dataDict:
    words = dataDict[dicline].split()
    for word in words :
        totalWordCount = totalWordCount+1
        if word in wordsCount:
            wordsCount[word] = wordsCount[word]+1
        else:
            wordsCount[word] = 1


##print (wordsCount)
print(totalWordCount)
```

for dicline in dataDict › for word in words

main

```
/usr/local/bin/python3 /Users/barismutlu/PycharmProjects/csvP
4505786

Process finished with exit code 0
```

Unique Word Count:



```
uniqueWordCount = 0
for dicline in dataDict:
    words = dataDict[dicline].split()
    for word in words :
        if word in wordsCount:
            wordsCount[word] = wordsCount[word]+1
        else:
            uniqueWordCount = uniqueWordCount + 1
            wordsCount[word] = 1


##print (wordsCount)
print(uniqueWordCount)
```

main

```
/usr/local/bin/python3 /Users/barismutlu/PycharmProjects/csv
57448
```

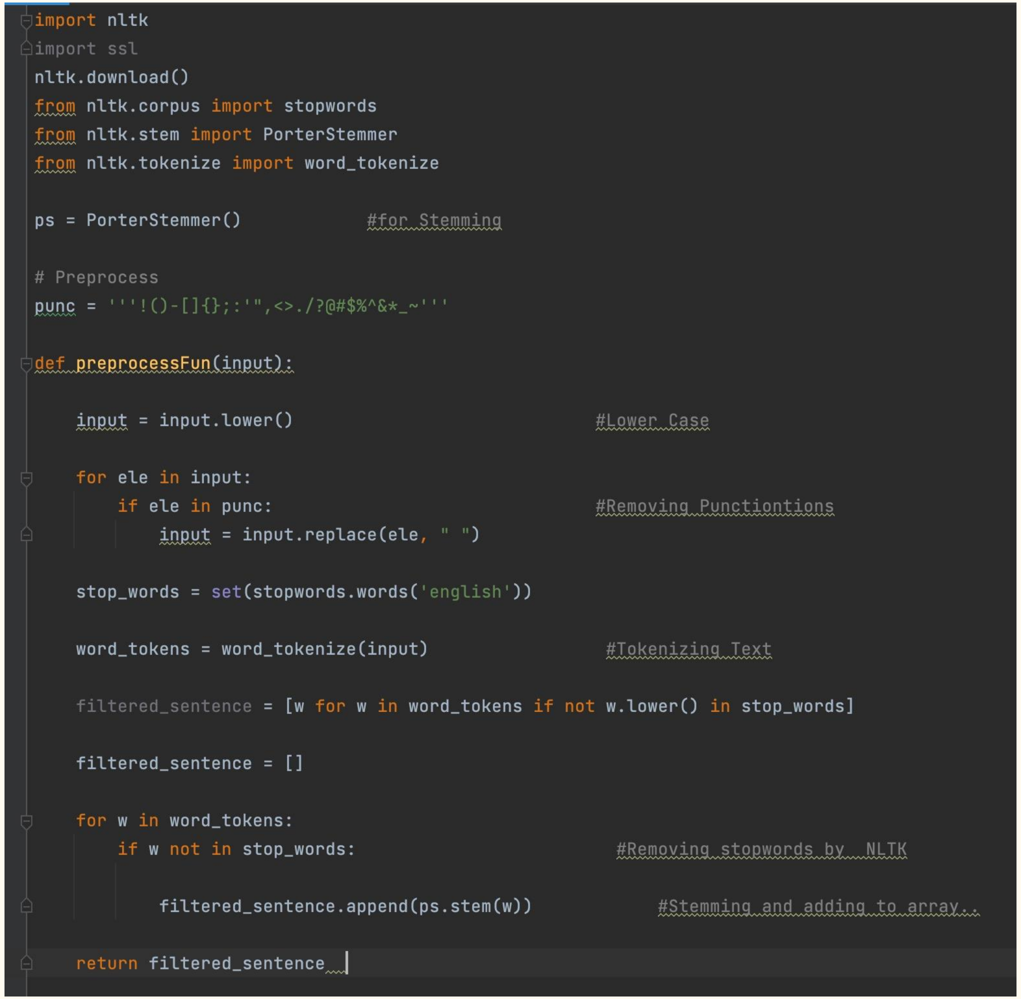
Most Frequent Words:

WORD	COUNT
19	106881
covid	106025
2	55925
patients	53211
sars	40387
cov	39217
coronavirus	31528
disease	30266
pandemic	25205
infection	18872
health	18213
2019	17707
clinical	17064
severe	16746
respiratory	16276
care	16142

After first presentation:

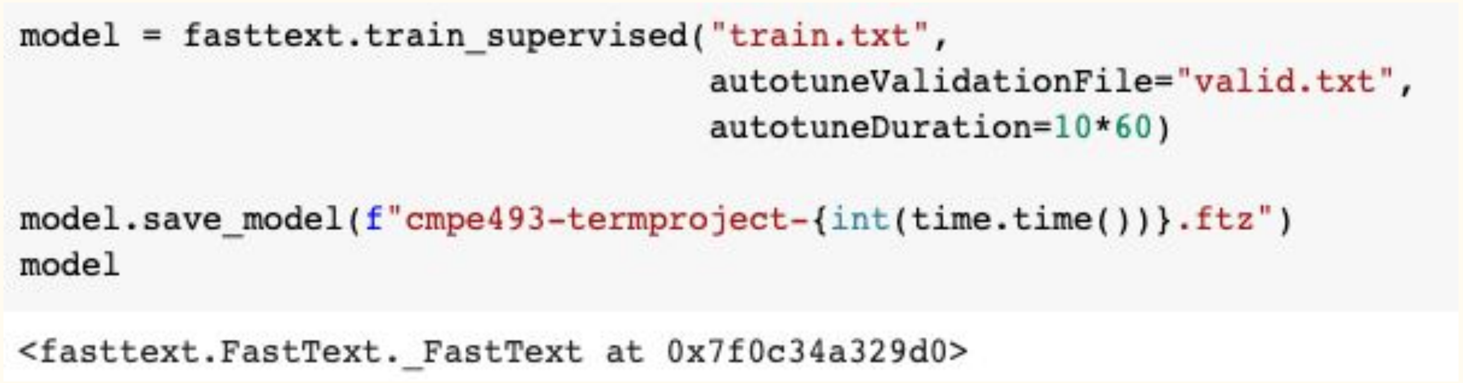
-Decided to improve by NLTK library.

-Library has more stopwords to remove them, and let us to stemming the raw data.



```python
import nltk
import ssl
nltk.download()
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()            #for Stemming

# Preprocess
punc = '''!()-[]{};:'",<>./?@#$%^&*_~'''

def preprocessFun(input):

    input = input.lower()                             #Lower Case

    for ele in input:
        if ele in punc:                               #Removing Punctiontions
            input = input.replace(ele, " ")

    stop_words = set(stopwords.words('english'))

    word_tokens = word_tokenize(input)                 #Tokenizing Text

    filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]

    filtered_sentence = []

    for w in word_tokens:
        if w not in stop_words:                         #Removing stopwords by  NLTK

            filtered_sentence.append(ps.stem(w))            #Stemming and adding to array..

    return filtered_sentence
```

Training



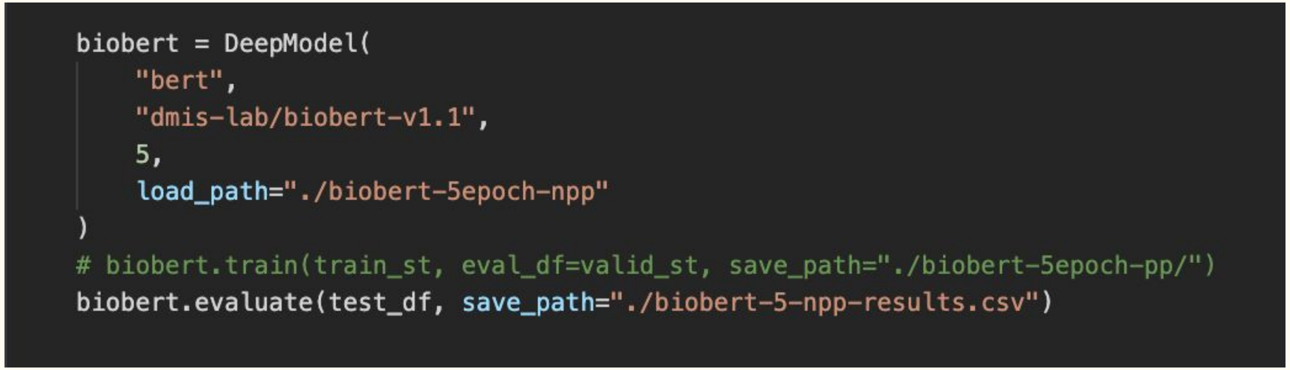
```
model = fasttext.train_supervised("train.txt",
                                  autotuneValidationFile="valid.txt",
                                  autotuneDuration=10*60)

model.save_model(f"cmpe493-termproject-{int(time.time())}.ftz")
model
```

```
<fasttext.FastText._FastText at 0x7f0c34a329d0>
```

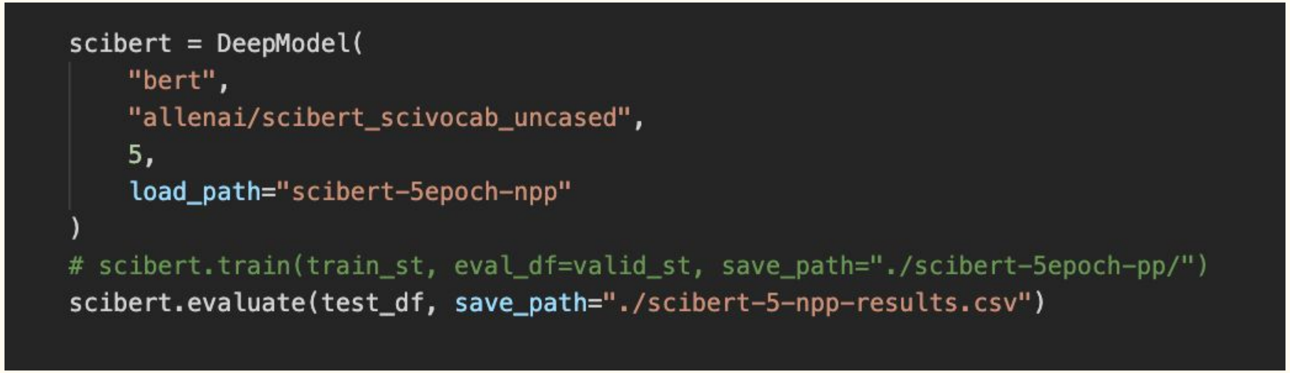
-Train data splitted into train(7x) and validation sets(2x)

-We used fastText with default parameters and autotuning as baseline approach

Training

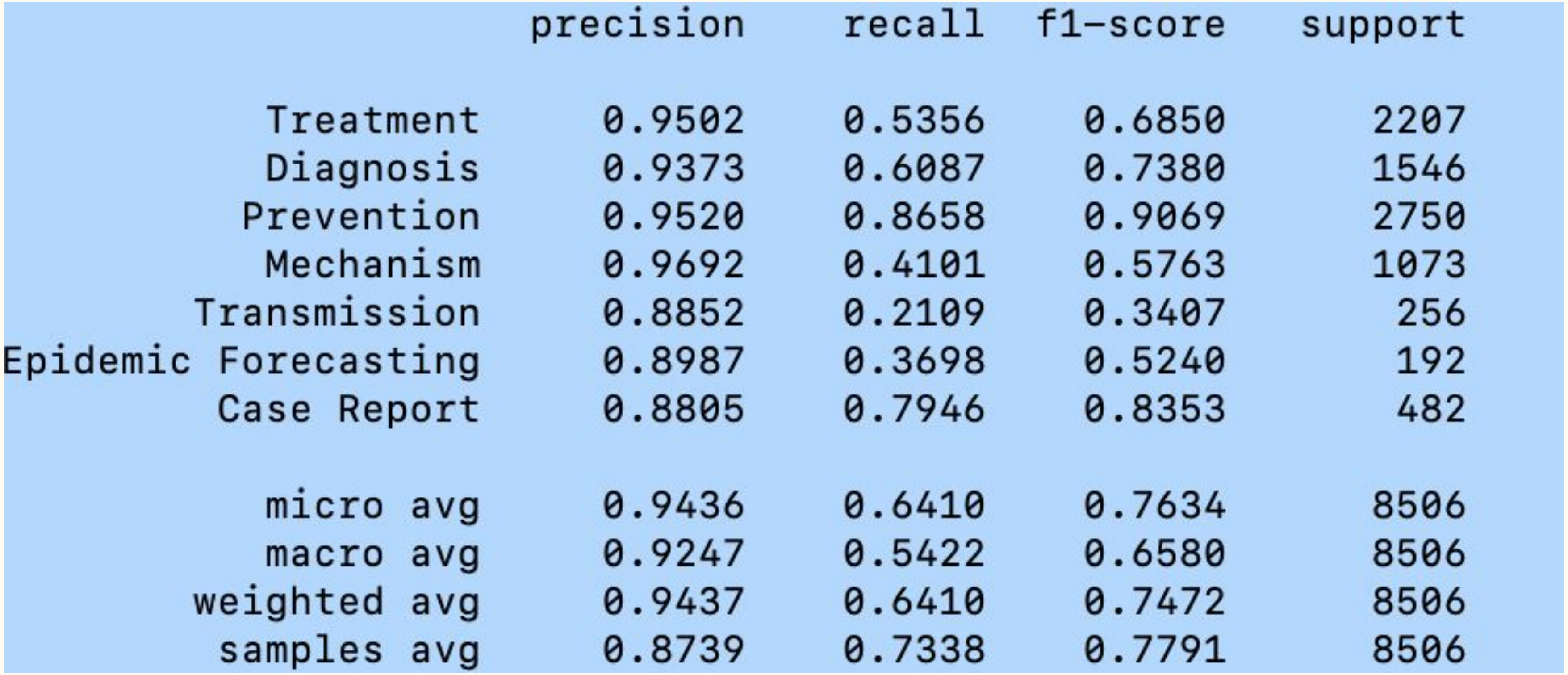


```
biobert = DeepModel(
    "bert",
    "dmis-lab/biobert-v1.1",
    5,
    load_path="./biobert-5epoch-npp"
)
# biobert.train(train_st, eval_df=valid_st, save_path="./biobert-5epoch-pp/")
biobert.evaluate(test_df, save_path="./biobert-5-npp-results.csv")
```



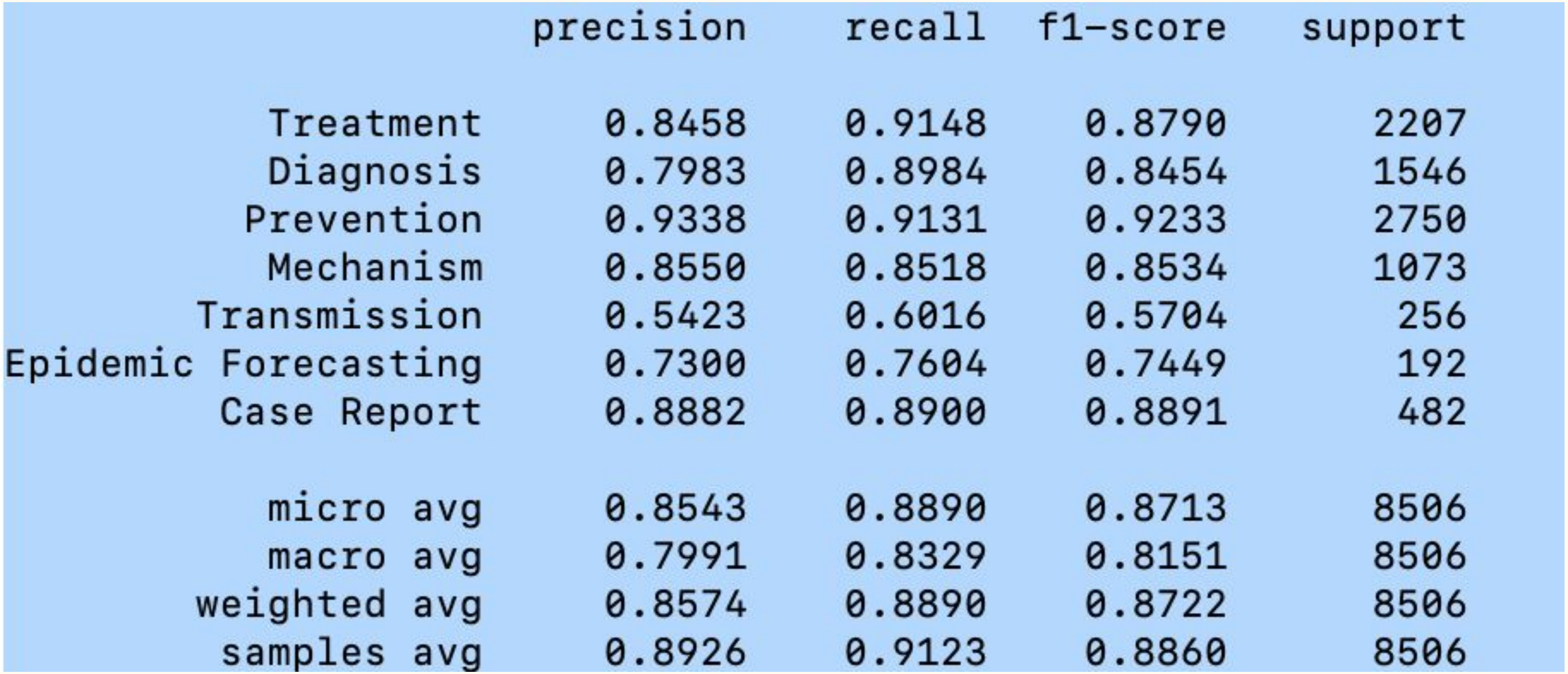
```
scibert = DeepModel(
    "bert",
    "allenai/scibert_scivocab_uncased",
    5,
    load_path="scibert-5epoch-npp"
)
# scibert.train(train_st, eval_df=valid_st, save_path="./scibert-5epoch-pp/")
scibert.evaluate(test_df, save_path="./scibert-5-npp-results.csv")
```

Test Results - FastText



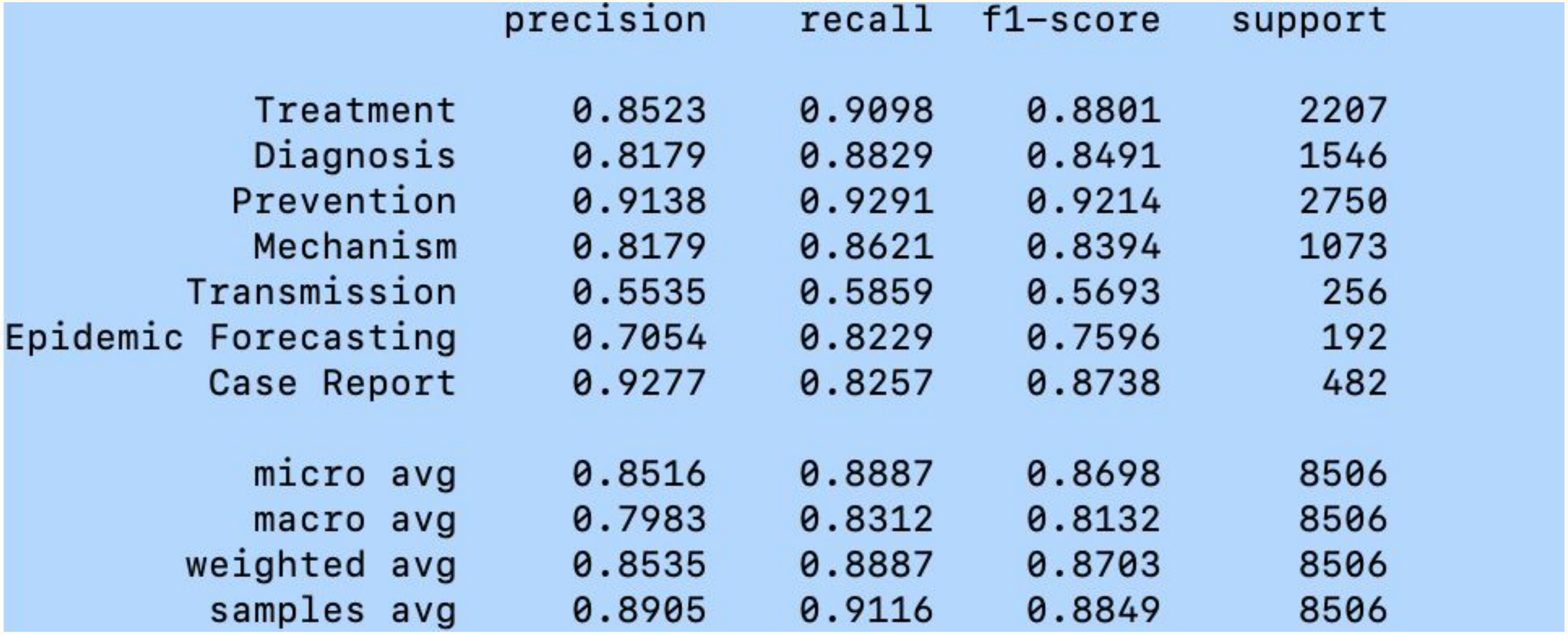
```
                    precision    recall  f1-score   support

          Treatment    0.9502    0.5356    0.6850      2207
          Diagnosis    0.9373    0.6087    0.7380      1546
         Prevention    0.9520    0.8658    0.9069      2750
          Mechanism    0.9692    0.4101    0.5763      1073
       Transmission    0.8852    0.2109    0.3407       256
Epidemic Forecasting    0.8987    0.3698    0.5240       192
        Case Report    0.8805    0.7946    0.8353       482

          micro avg    0.9436    0.6410    0.7634      8506
          macro avg    0.9247    0.5422    0.6580      8506
       weighted avg    0.9437    0.6410    0.7472      8506
        samples avg    0.8739    0.7338    0.7791      8506
```

Test Results - SciBert



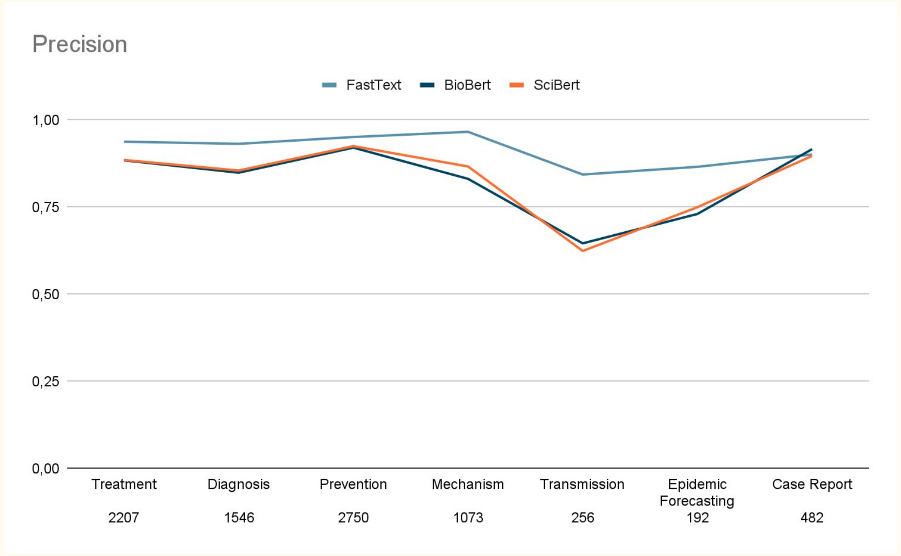
```
                     precision    recall  f1-score   support

          Treatment     0.8458    0.9148    0.8790      2207
          Diagnosis     0.7983    0.8984    0.8454      1546
         Prevention     0.9338    0.9131    0.9233      2750
          Mechanism     0.8550    0.8518    0.8534      1073
       Transmission     0.5423    0.6016    0.5704       256
Epidemic Forecasting    0.7300    0.7604    0.7449       192
        Case Report     0.8882    0.8900    0.8891       482

          micro avg     0.8543    0.8890    0.8713      8506
          macro avg     0.7991    0.8329    0.8151      8506
       weighted avg     0.8574    0.8890    0.8722      8506
        samples avg     0.8926    0.9123    0.8860      8506
```

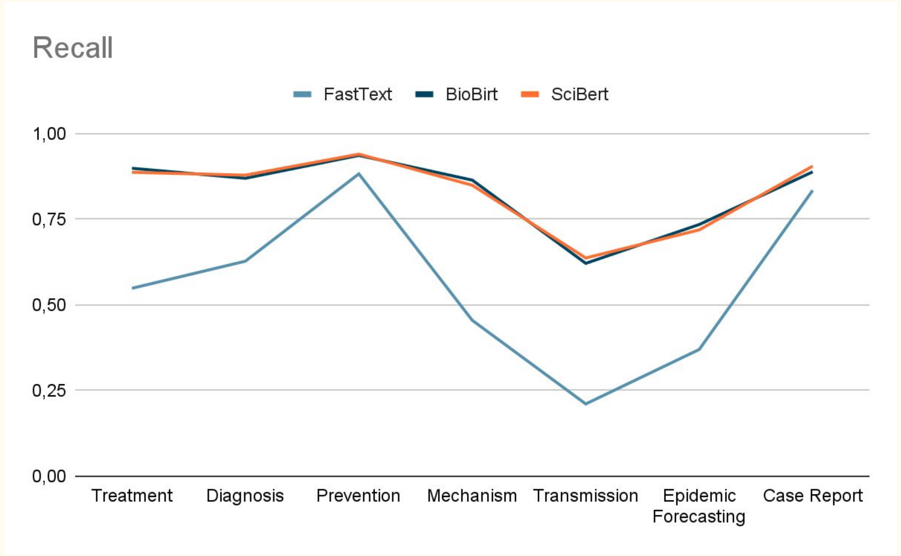
Test Results - BioBert



```
                    precision    recall  f1-score   support

          Treatment    0.8523    0.9098    0.8801      2207
          Diagnosis    0.8179    0.8829    0.8491      1546
         Prevention    0.9138    0.9291    0.9214      2750
          Mechanism    0.8179    0.8621    0.8394      1073
       Transmission    0.5535    0.5859    0.5693       256
Epidemic Forecasting    0.7054    0.8229    0.7596       192
        Case Report    0.9277    0.8257    0.8738       482

          micro avg    0.8516    0.8887    0.8698      8506
          macro avg    0.7983    0.8312    0.8132      8506
       weighted avg    0.8535    0.8887    0.8703      8506
        samples avg    0.8905    0.9116    0.8849      8506
```

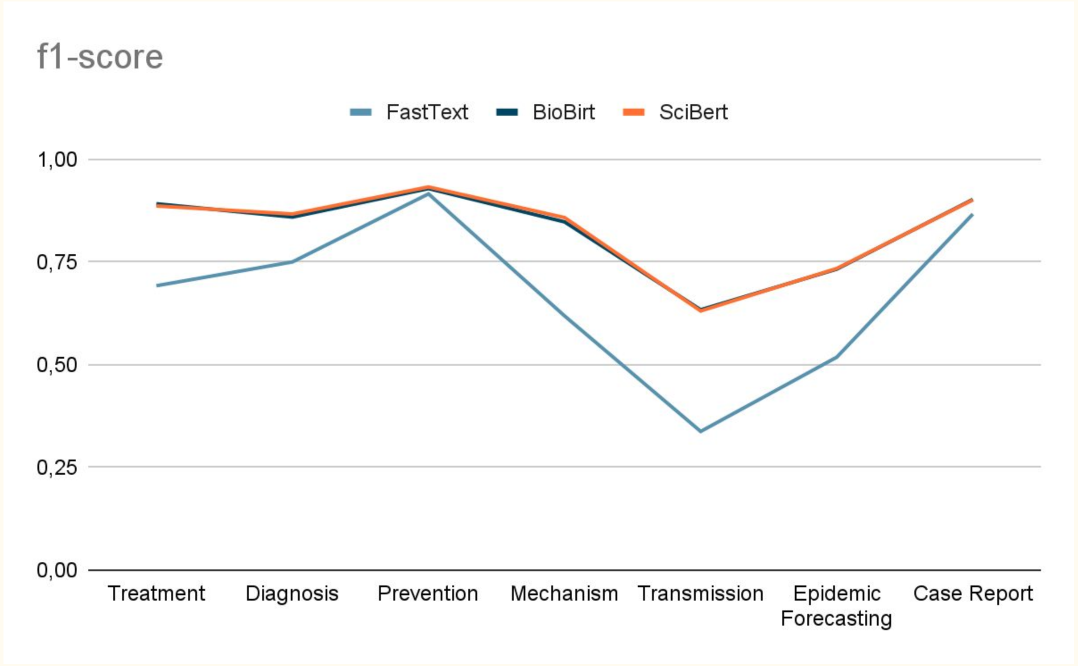
Model Comparison

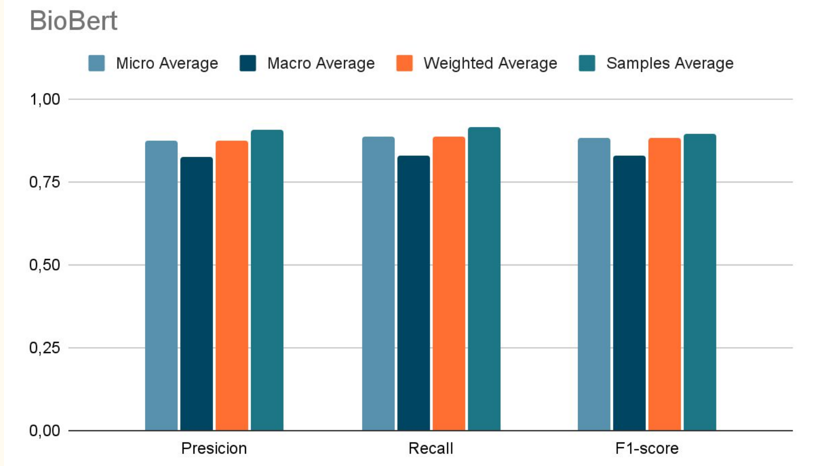


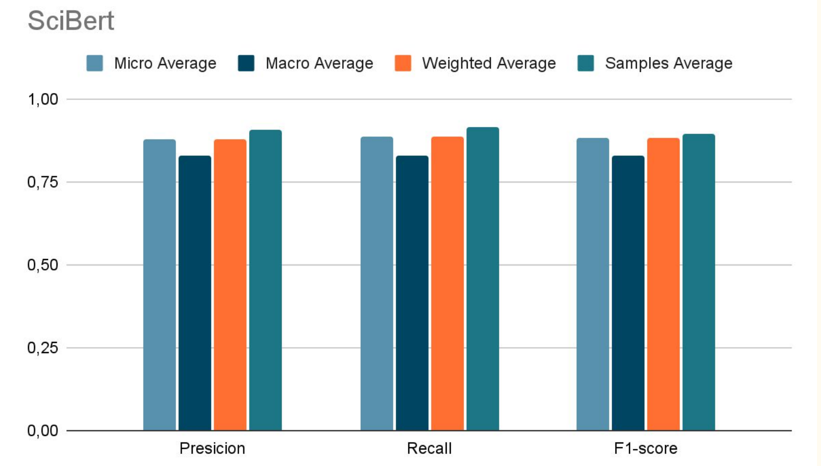


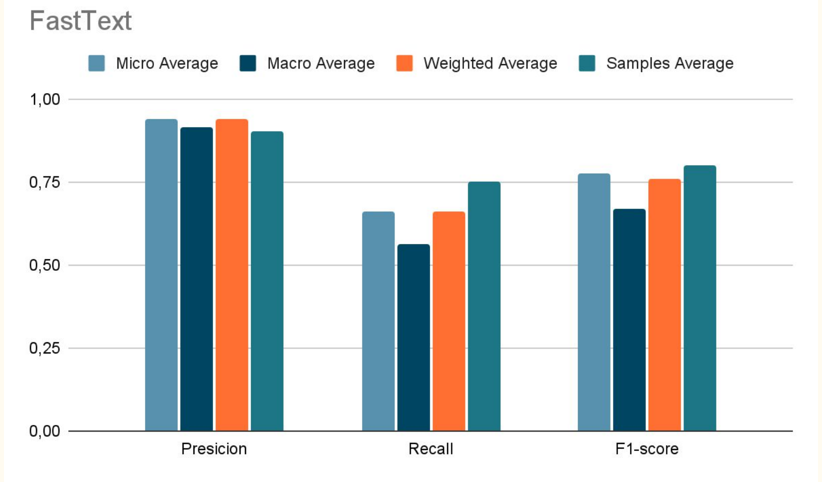
Gap between precision and recall values of the FastText is greater

Model Comparison









Macro and micro average changes of

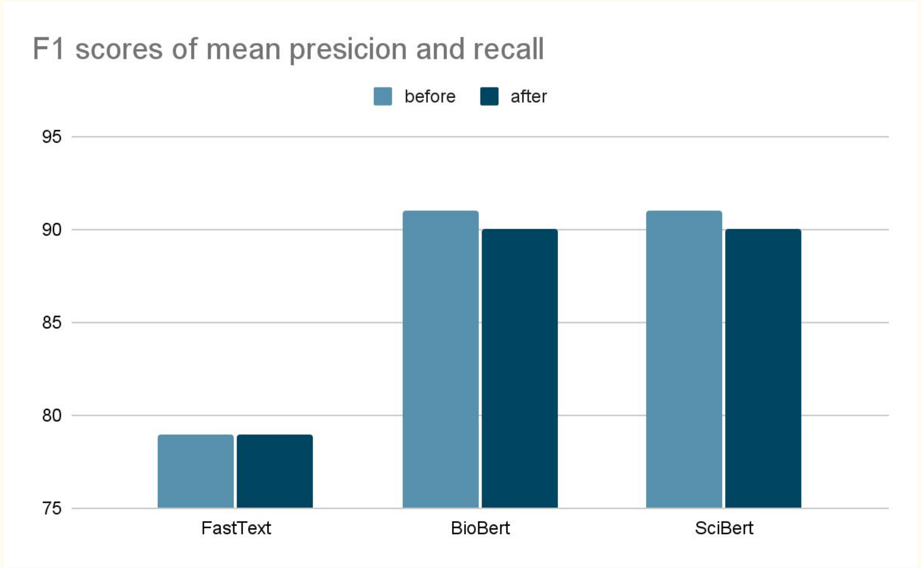
F1 - Scores:

%18 FastText Model

%6 Bert Models

-FastText is affected from minority classes more.

Effect of Stemming and NLTK library



Although the difference is not sharp, stemming decreases the f1 scores of Bert models while it does not affect the FastText model

How to improve the models?

In our data set, while some classes are estimated successfully, some are failed.

The difference between these classes are data sizes.

Ways to improve this unbalanced learning :

-Random over-sampling

Duplicate random data from minority classes

-Random under-sampling

Removing random data from majority classes

How to improve the models?

-Fast text has high precision scores while recall scores are low.

Threshold value can be decreased to make a tradeoff between precision and recall scores.