Multi Layer Perceptron

Ko, Youngjoong

Sungkyunkwan University

nlp.skku.edu, nlplab.skku.edu

Contents

Introduction

❖ Introduction

- The assignment is to implement the ‘Multi Layer Perceptron’ model for text classification.

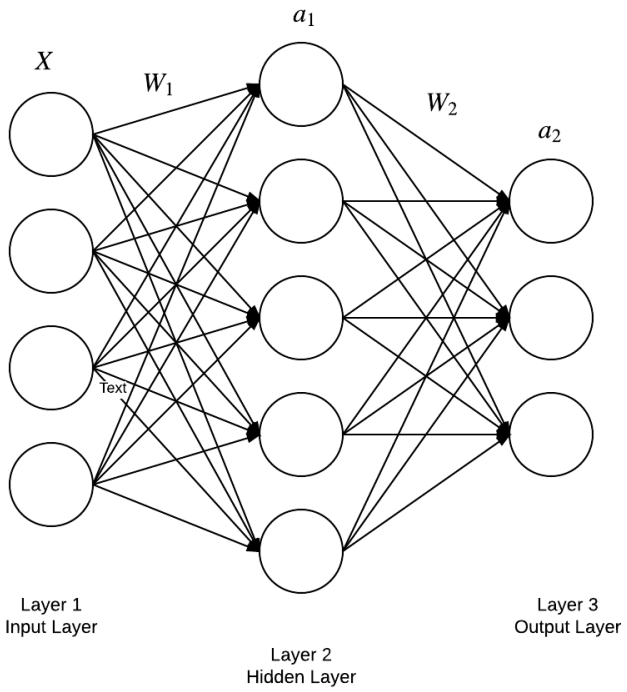
  ※ See page 10 for specifics

- In this PDF, we will show what the ‘Multi Layer Perceptron’ is, and how to implement the model with Scikit-learn.

Multi Layer Perceptron

❖ Multi Layer Perceptron

- Multi Layer Perceptron (MLP) is a neural network consisting of more than two layers
- Each layer has several perceptrons.



Multi Layer Perceptron with Scikit-Learn

❖ Implementation of the model

- Implementation of the Multi Layer Perceptron model consists of 3 steps

1. Preprocessing Train/Test data and calculating TF-IDF
2. Learning the model with Train data
3. Evaluating the model with Test data

Multi Layer Perceptron with Scikit-Learn

❖ Python Classes of ‘sklearn.neural_network’

- neural_network.MLPClassifier (Multi Layer Perceptron classifier)
  - For implementing the Multi Layer Perceptron, there are some hyperparameters to be tuned such as the number of hidden neurons, layers, and iterations.
  - You have to use the above library to implement the MLP.

Multi Layer Perceptron with Scikit-Learn

❖ Parameters of neural_network.MLPClassifier module

- hidden_layer_sizes : tuple, (default =100,)

  -The i-th element represents the number of neurons in the i-th hidden layer.

  ex. hidden_layer_sizes=$(i_1, i_2, \dots, i_n)$

- activation : {'identity,' 'logistic,' 'tanh,' 'relu'}, (default='relu')

  - An activation function for the hidden layer.

- solver : {'lbfbs,' 'sgd,' 'adam,'}, (default='adam')

  - A method for weight optimization.

- batch_size : int, (default='auto')

  - The size of minibatches for stochastic optimizers.

  - 'auto' : batch_size=min(200,n_samples)

Multi Layer Perceptron with Scikit-Learn

❖ neural_network.MLPClassifier (Parameters)

- learning_rate_init :double, (default =0.001)
  - The initial learning rate used. It controls the step-size in updating the weights.

- max_iter : int, (default=200)
  - Maximum number of iterations.

Multi Layer Perceptron with Scikit-Learn

❖ Fixing random numbers

- random.seed(), np.random.seed()
  - Above functions generate random numbers. You can fix those random numbers by aligning same seed value.
  - You have to align a same seed value to every random number generator for this assignment. If you don't, you may get significantly different results whenever you run the model.

```
import random
import numpy as np

random.seed(777)
np.random.seed(777)
```

Multi Layer Perceptron with Scikit-Learn

❖ Learning the Multi Layer Perceptron model with Train data

- Define the MLP model

```
from sklearn.neural_network import MLPClassifier
classfier = MLPClassifier(hidden_layer_sizes=(100, ),
                          activation='tanh',solver='adam',
                          batch_size='auto',learning_rate_init=0.001,
                          max_iter=200)
classfier.fit(train_tf_idf, train_labels)
```


The above code is a just example. so parameter and function for this assignment can be differ.

Multi Layer Perceptron with Scikit-Learn

❖ Learning Multi Layer Perceptron model with Train data

- Input ”Train data input list” and ”Train labels list” to Multi Layer Perceptron model

```
from sklearn.neural_network import MLPClassifier
classfier = MLPClassifier(hidden_layer_sizes=(100, ),
                          activation='tanh',solver='adam',
                          batch_size='auto',learning_rate_init=0.001,
                          max_iter=200)
classfier.fit(train_tf_idf, train_labels)
```

“Train data input list” “Train labels list”

The above code is a just example. so parameter and function for this assignment can be differ.

Multi Layer Perceptron with Scikit-Learn

❖ Evaluating the model with Test data

```
1 prediction = classifier.predict(test_tf_idf)
2 print(prediction)
```

```
['business' 'business' 'business' 'tech' 'business' 'business' 'politics'
 'politics' 'politics']
```

"Test data input list"

```
1 from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
2
3 print('Confusion metrics : ', confusion_matrix(test_labels,prediction))
4 print('Accuracy : ', accuracy_score(test_labels,prediction))
5 print('Macro averaging precision : ', precision_score(test_labels,prediction, average='macro'))
6 print('Micro averaging precision : ', precision_score(test_labels,prediction, average='micro'))
7 print('Macro averaging recall : ', recall_score(test_labels,prediction, average='macro'))
8 print('Micro averaging recall : ', recall_score(test_labels,prediction, average='micro'))
9 print('Macro averaging f1-score : ', f1_score(test_labels,prediction, average='macro'))
10 print('Micro averaging f1-score : ', f1_score(test_labels,prediction, average='micro'))
```

```
Confusion metrics :  [[20  0  0]
 [ 7 13  0]
 [ 5  0 15]]
Accuracy :  0.8
Macro averaging precision :  0.875
Micro averaging precision :  0.8
Macro averaging recall :  0.7999999999999999
Micro averaging recall :  0.8
Macro averaging f1-score :  0.8047508047508048
Micro averaging f1-score :  0.8000000000000002
```

Assignment

❖ Assignment

1) The attached JSON files consist of 240 and 60 articles respectively.
   - See page 17-18 for input file format.

2) This assignment is based on Assignment 4 (TF-IDF) and Assignment 6 (Evaluation).

3) Implement Multi Layer Perceptron and evaluate the result of the model, your model **must result in at least 90% of Macro averaging f1-score**

4) You have to use **'MLPClassifier' function** of scikit-learn with appropriate parameters.

5) You can import below functions from Scikit-Learn library for evaluation
   - confusion matrix, accuracy, precision, recall, F1-score

Assignment

❖ Assignment

6) You have to create .txt file containing results of above metrics in form of percentage (file I/O) by using your code.

   - See page 19 for output file format.

7) You are **allowed to use only 'scikit-Learn'** library when calculating performances, calculating TF-IDF and implementing the Multi Layer Perceptron model.

8) You **are not allowed** to use any library except scikit-learn, json and nltk

9) Teaching assistants have been checking out "Copy" to protect plagiarism by a copy detecting system. 0 scores will be assigned to all "Copy" results.

10) Round down to the fourth digit after the decimal point.

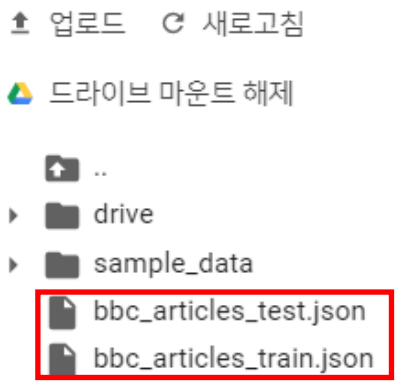
   Ex) 78.42159 → 78.4215 / 51.172426 → 51.1724

Assignment

❖ Assignment

11) For this assignment, If there is any evidence that you upload your source code or download it from any public internet site (Stack Overflow, Github, etc.) we will regard it as copy and you will get 0 score.

12) Upload the input file to the same folder where the source code exists as shown on the right.



13) All paths in the code use a relative path as below.

Ex)
```
with open('./bbc_articles_test.json', 'r') as f:
```

12) From this assignment, we need what we learned from the previous assignments. so if your code of previous assignments was deducted, you should analyze and modify your code

Assignment

❖ Submission File (submit following files as a compressed .zip file)

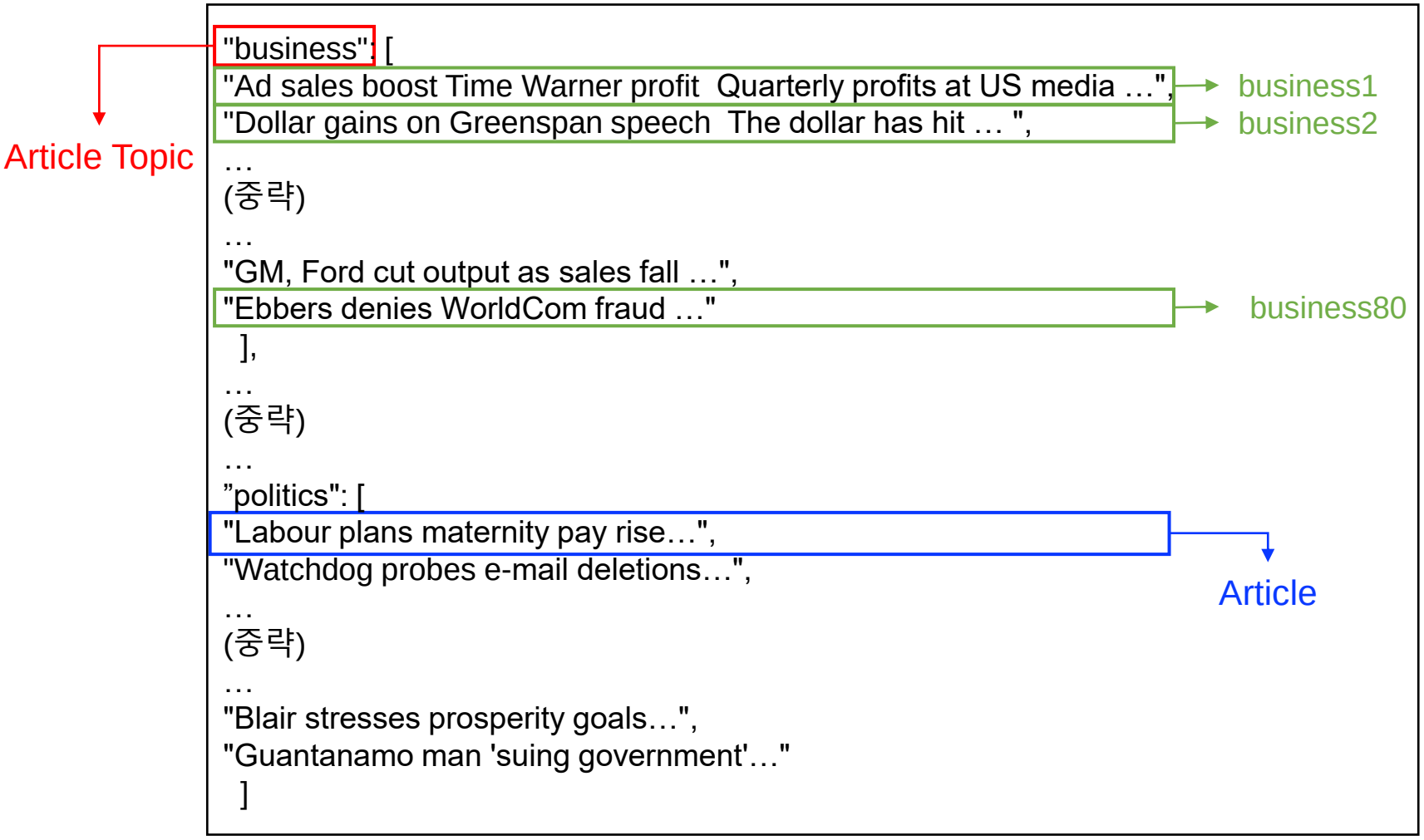
1) Python code file (.py) (python version 3.x , not .ipynb file)
   - Format: “Student Number_Name_MLP.py”.
     - Ex) “2020000000_홍길동_MLP.py”
   - **You will get deduction score if you submit the code with any other format such as .ipynb.**
   - Should develop your code **on Colab**
   - You will get **deduction score if** you submit different outputs from different environments (not Colab).

2) TEXT file (.txt)
   - Format: “Student Number_Name_MLP.txt”.
   - You will get **deduction score if you copy and paste the results from print() function.** This means that your code has to create .txt file by file I/O mechanism as below.

Ex)
```
fw = open('./Student-Number_NAME_MLP.txt', 'w', encoding="UTF-8")
fw.write(result)
fw.close()
```

JSON Input File

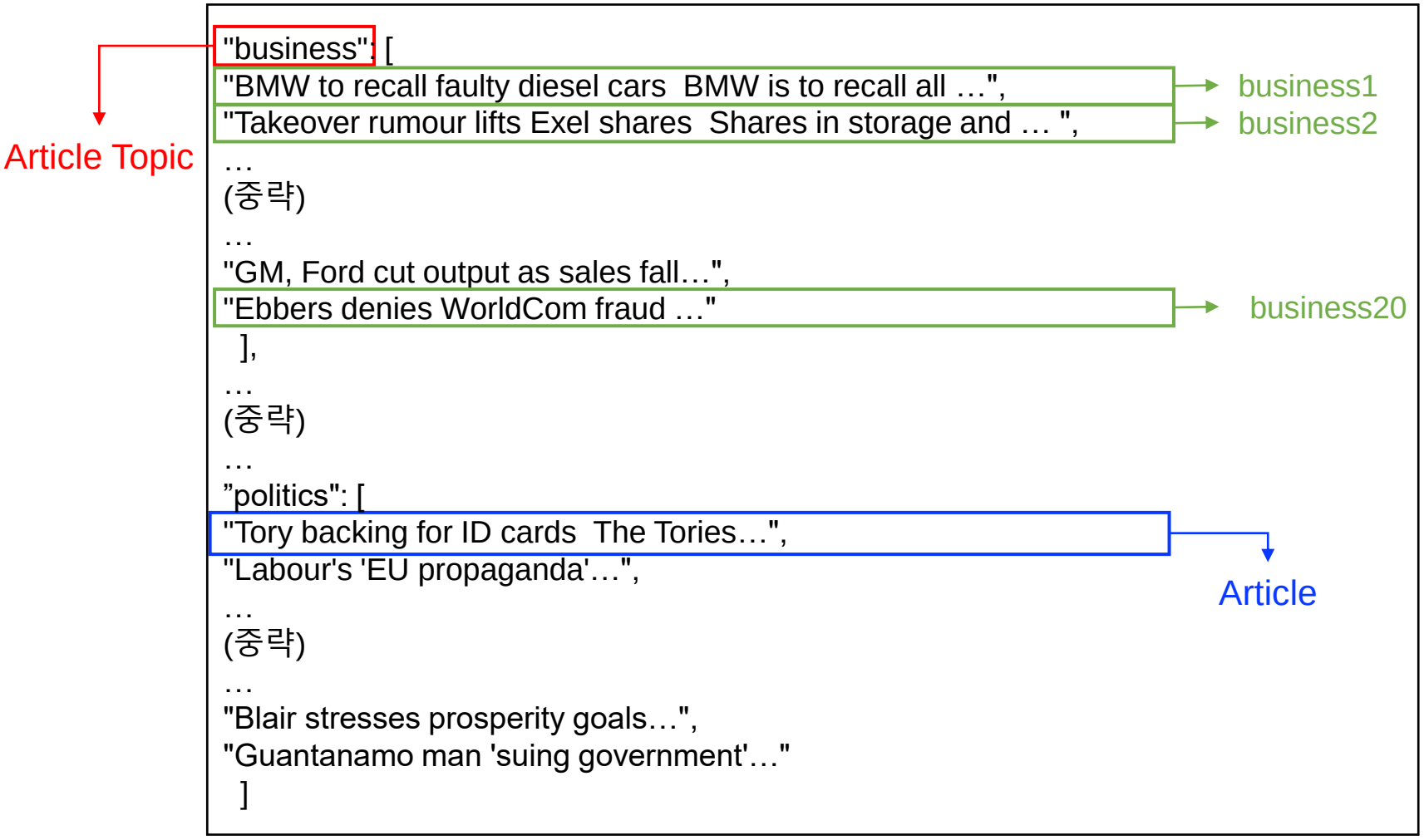
❖ bbc_articles_train.json



- The file for training includes 240 articles of business, politics, and tech topics.

JSON Input File

❖ bbc_articles_test.json



- The file for test includes 60 articles of business, politics, and tech topics.

An Example of Output File

Student Number_Name_MLP.txt

Confusion matrix :
25 12 13
16 21 13
14 16 20

Accuracy : 44.0000%

Macro averaging precision : 43.9298%
Micro averaging precision : 42.8571%

Macro averaging recall : 44.0000%
Micro averaging recall : 45.2054%

Macro averaging f1-score : 43.9649%
Micro averaging f1-score : 44.0000%

Delimiter: Newline

Delimiter : tab

Delimiter : space

You must achieve at least 90%

- Above examples can be different with actual answers.
- Round down to the fourth digit after the decimal point.