Naive Bayes

Ko, Youngjoong

Sungkyunkwan University

nlp.skku.edu, nlplab.skku.edu

成均館大學校
1398
SUNG KYUN KWAN UNIVERSITY

Contents

Introduction

❖ Introduction

- The assignment is to implement the ‘Naive Bayes’ model for text classification.

  (See page 12 for specifics)

- In this PDF, we will explain what ‘Naive Bayes’ is, and how to implement the Naive Bayes model using Scikit-learn

Naive Bayes

❖ Naive Bayes

- The Naive Bayes algorithm is one of the simplest machine learning algorithms based on Bayes theorem for classification tasks.

Naive Bayes

❖ Naive Bayes of Text Classification

- Text Classification is a task that classifies the topic of text.

TF-values

	feature 1	feature 2	feature 3	Topic
atricle 1	1	5	0	business
atricle 2	5	2	3	tech
atricle 3	3	0	1	politics
atricle 4	2	6	2	business
atricle 5	5	2	1	tech
atricle 6	5	0	9	politics
new article	2	1	7	?

what is the Topic of new article?

- If feature1, feature2, feature3 are independent,

$$p(topic|feature1, feature2, feature3) = \frac{p(topic)\, p(feature1, feature2, feature3|topic)}{p(feature1, feature2, feature3)}$$

$$p(feature1, feature2, feature3|topic) = p(feature1|topic)p(feature2|topic)\; p(feature3|topic)$$

Naive Bayes with Scikit-Learn

❖ Model Implement

- Naive Bayes model implementation consist of 3 steps
  1. Preprocess Train/Test data and calculate TF
  2. Learn Naive Bayes model using Train data
  3. Evaluate model using Test data

- ‘sklearn.naive_bayes’ provides following python classes.
  1. GaussianNB
     - For Normal Distribution
  2. BernoulliNB
     - For Bernoulli Distribution
  3. MultinomialNB
     - For Multinomial Distribution

Naive Bayes with Scikit-Learn

❖ Python Classes of ‘sklearn.naive_bayes’

- GaussianNB
  - It is for special cases of Naive Bayes like **continuous data**.
- BernoulliNB
  - It used **for binary/boolean representation** data, such as whether or not certain words exist.
- MultinomialNB
  - It is used **for Multi Label Text Classification.**
  - In this assignment, you should use it

Naive Bayes with Scikit-Learn

❖ Learning Naive Bayes model

- sklearn.naive_bayes.MultinomialNB(*alpha=1.0*)
  - Alpha : float (default = 1.0)

    - Additive smoothing parameter (0 for no smoothing).
    - The smoothing priors $\alpha \geq 0$ accounts for features not present in the learning samples and prevents zero probabilities in further computations.
    - Setting $\alpha = 1$ is called Laplace smoothing, while $\alpha < 1$ is called Lidstone smoothing.

Naive Bayes with Scikit-Learn

❖ Learning Naive Bayes model using Train data

- Define model

```
from sklearn.naive_bayes import MultinomialNB,BernoulliNB,GaussianNB

classifier = BernoulliNB(alpha=0.001)

classifier.fit(train_tf, train_labels)
prediction = classifier.predict(test_tf)
```

- alpha (default = 1.0)

  - It is for smoothing parameter to prevent features would not become 0

  - Larger alpha reduces model complexity, while lower alpha increases model complexity.

  - The performance of the model may change depending on the alpha.

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

Naive Bayes with Scikit-Learn

❖ Learning Naive Bayes model using Train data

- Apply Train data and labels to Naive Bayes models

```
from sklearn.naive_bayes import MultinomialNB,BernoulliNB,GaussianNB

classifier = BernoulliNB(alpha=0.001)

classifier.fit(train_tf, train_labels)
prediction = classifier.predict(test_tf)
```

TF values of Test data

Labels of Train data

TF values of Train data

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

Naive Bayes with Scikit-Learn

❖ Evaluate model using Test data

```
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
```


The above result is a just example. so your result for this assignment can be differ.

Assignment

❖ Assignment

1) The two attached JSON files consists of 240 and 60 articles, respectively.
   - See page 16-17 for input file format.

2) This assignment is based on TF and Evaluation assignments using Scikit-Learn in Assignment 3(TF) and 6(Evaluation).

3) Implement Naive Bayes and evaluate the result of the model, your model **must achieve at least 85% (Macro averaging f1-score)** with **'MultinomialNB' function** of scikit-learn and appropriate parameter.

4) You can use the functions of the Scikit-Learn library for evaluation
   - confusion matrix, accuracy, precision, recall, F1-score

Assignment

❖ Assignment

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

   - See page 18 for output file format.

6) You are **allowed to use only 'Scikit-Learn'** for calculating metrics, TF and implementing the Naive Bayes model. So you cannot use any library except scikit-learn and json and nltk

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

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

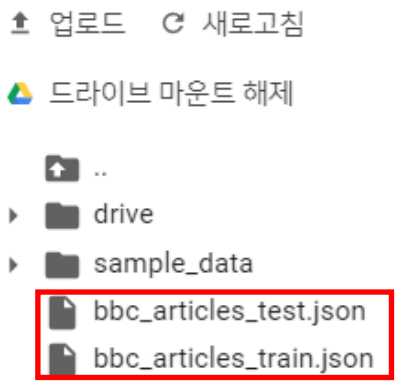
   Ex) 78.42159 → 78.4215 / 51.172426 → 51.1724

Assignment

❖ Assignment

9) 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.

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



11) 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)

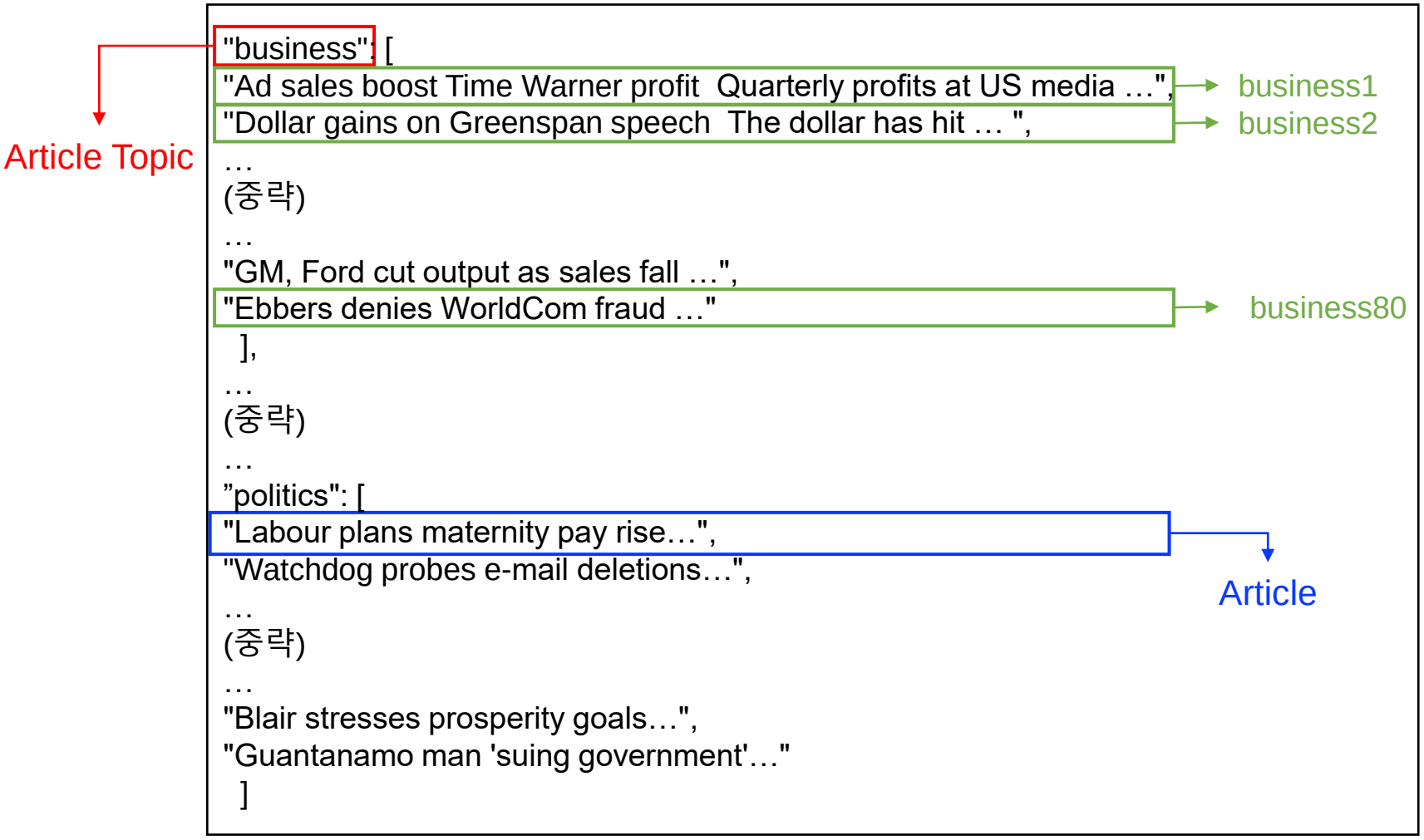
1) Python code file (.py) (python version 3.x , not .ipynb file)
   - Format: “Student Number_Name_NB.py”.
     - Ex) “2020000000_홍길동_ NB.py”
   - **You will get deduction score if you submit the code with any other format such as .ipynb.**
   - Should develop your code **on Colab**
   - You will get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

2) TEXT file (.txt)
   - Format: “Student Number_Name_NB.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_NB.txt', 'w', encoding="UTF-8")
fw.write(result)
fw.close()
```

JSON Input File

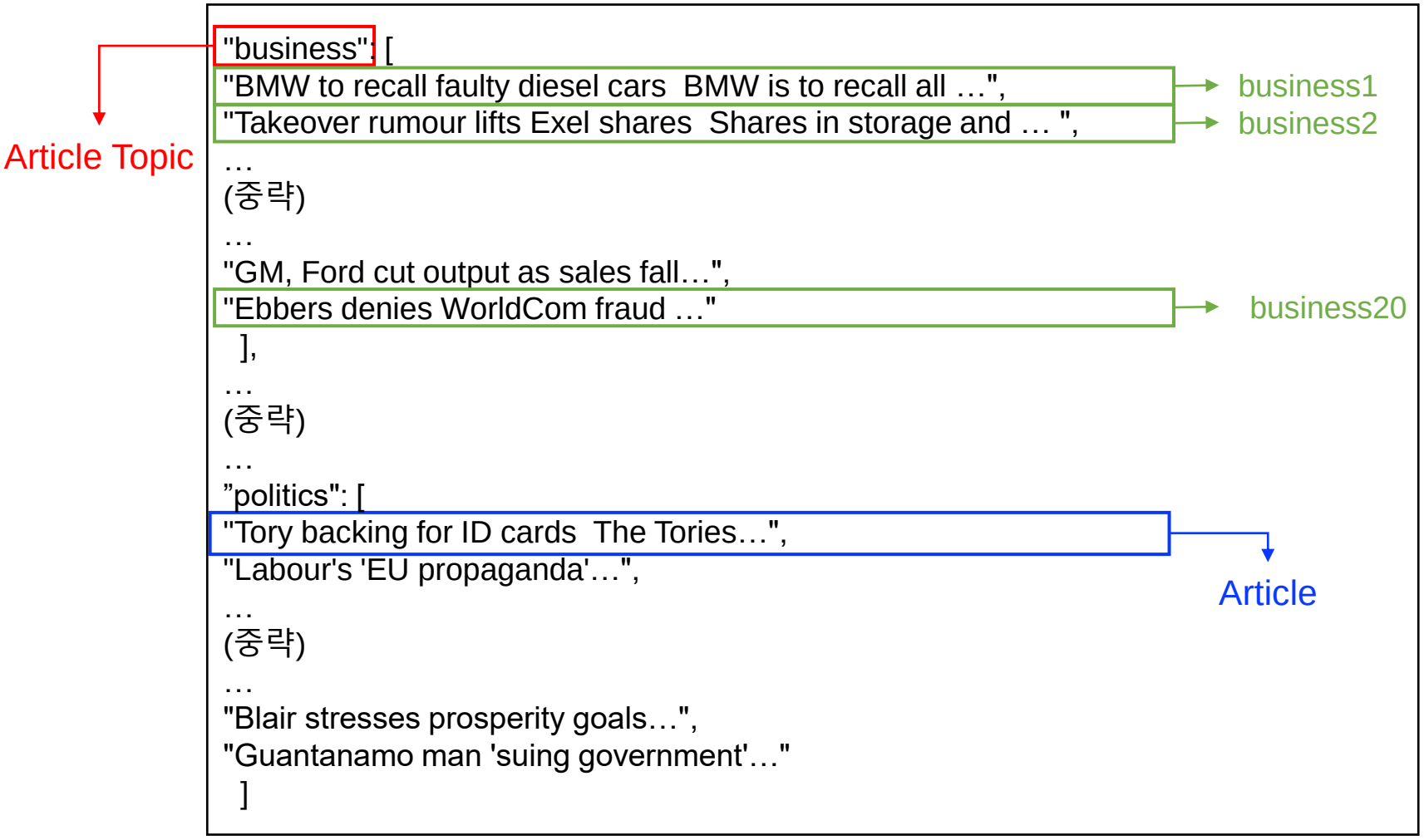
❖ bbc_articles_train.json



- JSON input file includes 240 articles on business, politics, and tech topics.

JSON Input File

❖ bbc_articles_test.json



- JSON input file includes 60 articles on business, politics, and tech topics.

An Example of Output File

Student Number_Name_NB.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 85%

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