Preprocessing_news

August 31, 2021

1 Pre-processing News Data

1.1 Bài toán

Dữ liệu gồm n văn bản phân vào 10 chủ đề khác nhau. Cần biểu diễn mỗi văn bản dưới dạng một vector số thể hiện cho nội dụng của văn bản đó.

1.2 Mục lục

- Load dữ liệu từ thư mục
- Loại bỏ các stop words
- Sử dụng thư viện để mã hóa TF-IDF cho mỗi văn bản

1.3 Phương pháp mã hóa: TF-IDF

Cho tập gồm n văn bản: $D = \{d_1, d_2, ...d_n\}$. Tập từ điển tương ứng được xây dựng từ n văn bản này có độ dài là m - Xét văn bản d có $|d|$ từ và t là một từ trong d. Mã hóa tf-idf của t trong văn bản d được biểu diễn:

$$\begin{aligned} \text{tf}_{t,d} &= \frac{f_t}{|d|} \\ \text{idf}_{t,d} &= \log \frac{n}{n_t}, \qquad n_t = |\{d \in D : t \in d\}| \\ \text{tf-idf}_{td} &= \text{tf}_{t,d} \times \text{idf}_{t,d} \end{aligned} \tag{1}$$

- Khi đó văn bản d được mã hóa là một vector m chiều. Các từ xuất hiện trong d sẽ được thay bằng giá trị tf-idf tương ứng. Các từ không xuất hiện trong d thì thay là 0

[ ]:
```
import os
import matplotlib.pyplot as plt
import numpy as np

from sklearn.datasets import load_files
from pyvi import ViTokenizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline

%matplotlib inline
```

1.4 Load dữ liệu từ thư mục

Cấu trúc thư mục như sau

- data/news_vnexpress/
  - Kinh tế:
    - bài báo 1.txt
    - bài báo 2.txt
  - Pháp luật
    - bài báo 3.txt
    - bài báo 4.txt

[ ]:
```
INPUT = 'data/news_vnexpress'
os.makedirs("images",exist_ok=True)  # thư mục lưu các các hình ảnh trong quá
↪trình huấn luyện và đánh gía
```

[ ]:
```
# statistics
print('Các nhãn và số văn bản tương ứng trong dữ liệu')
print('-------------------------------------------')
n = 0
for label in os.listdir(INPUT):
    print(f'{label}: {len(os.listdir(os.path.join(INPUT, label)))}')
    n += len(os.listdir(os.path.join(INPUT, label)))

print('-------------------------')
print(f"Tổng số văn bản: {n}")
```

```
Các nhãn và số văn bản tương ứng trong dữ liệu
-------------------------------------------
doi-song: 120
du-lich: 54
giai-tri: 201
giao-duc: 105
khoa-hoc: 144
kinh-doanh: 262
phap-luat: 59
suc-khoe: 162
the-thao: 173
thoi-su: 59
-------------------------
Tổng số văn bản: 1339
```

[ ]:
```
# load data
data_train = load_files(container_path=INPUT, encoding="utf-8")
print('mapping:')
for i in range(len(data_train.target_names)):
    print(f'{data_train.target_names[i]} - {i}')
```

```
print('--------------------------')
print(data_train.filenames[0:1])
# print(data_train.data[0:1])
print(data_train.target[0:1])
print(data_train.data[0:1])

print("\nTổng số  văn bản: {}" .format( len(data_train.filenames)))
```

```
mapping:
doi-song - 0
du-lich - 1
giai-tri - 2
giao-duc - 3
khoa-hoc - 4
kinh-doanh - 5
phap-luat - 6
suc-khoe - 7
the-thao - 8
thoi-su - 9
--------------------------
['data/news_vnexpress\\khoa-hoc\\00133.txt']
[4]
['Mời độc giả đặt câu hỏi tại đây\n']

Tổng số  văn bản: 1339
```

1.5 Chuyển dữ liệu dạng text về ma trận (n x m) bằng TF-IDF

- Bạn cần viết đoạn mã tương ứng trong cell bên dưới. Theo các bước được gợi ý

[ ]:
```
# load dữ liệu các stopwords

#---> Code ở đây

# Chuyển hoá dữ liệu text về dạng vector TF
#     - loại bỏ từ dừng
#     - sinh từ điển

#---> Code ở đây

# Hàm thực hiện chuyển đổi dữ liệu text thành dữ liệu số dạng ma trận
# Input: Dữ liệu 2 chiều dạng numpy.array, mảng nhãn id dạng numpy.array

data_preprocessed = #Code ở đây

X = data_preprocessed # thuoc tinh
Y = data_train.target #nhan
```

```
print(f"\nSố lượng từ trong từ điển: {len(module_count_vector.vocabulary_)}")
print(f"Kích thước dữ liệu sau khi xử lý: {X.shape}")
print(f"Kích thước nhãn tương ứng: {Y.shape}")
```

```
Số lượng stopwords: 2063
['a_lô', 'a_ha', 'ai', 'ai_ai', 'ai_nấy', 'ai_đó', 'alô', 'amen', 'anh',
'anh_ấy']

Số lượng từ trong từ điển: 12796
Kích thước dữ liệu sau khi xử lý: (1339, 12796)
Kích thước nhãn tương ứng: (1339,)
```

[ ]:
```
print(X[100].toarray())
print(Y[100])
```

```
[[0.         0.         0.         … 0.         0.14048828 0.        ]]
5
```

[ ]:
```
sum(sum(X[100].toarray() != 0))
```

[ ]:
```
289
```

[ ]:
```
print(X[100])
```

```
  (0, 12794)    0.14048828324700804
  (0, 12724)    0.051226678060487627
  (0, 12714)    0.034379239518190156
  (0, 12705)    0.024927343279465615
  (0, 12697)    0.03935911209707954
  (0, 12692)    0.013885134230282647
  (0, 12691)    0.02076954755505395
  (0, 12672)    0.03173992101554847
  (0, 12646)    0.04268947993761032
  (0, 12643)    0.030193779677554416
  (0, 12629)    0.024173036345759045
  (0, 12626)    0.01928809379275951
  (0, 12624)    0.3318224864003995
  (0, 12617)    0.08000423234784886
  (0, 12591)    0.07519534686809994
  (0, 12584)    0.03876774373554222
  (0, 12566)    0.033240367004725005
  (0, 12558)    0.03206234356763185
  (0, 12547)    0.04575286598942787
  (0, 12535)    0.05488370325838488
  (0, 12521)    0.09355442947181113
  (0, 12517)    0.03883219864696093
  (0, 12509)    0.017786174579851665
```

```
(0, 12454)    0.07589970050190288
(0, 12272)    0.02125953768208212
:     :
(0, 2170)     0.029508397725910254
(0, 2159)     0.016084504788746505
(0, 2140)     0.015661963686282587
(0, 2135)     0.04322051581452054
(0, 2111)     0.025775634654335113
(0, 2101)     0.026936724711167648
(0, 2076)     0.014547704347804934
(0, 1866)     0.0511703119570894 7
(0, 1783)     0.0473450956087146
(0, 1631)     0.023493949966261328
(0, 1590)     0.03492662390306206
(0, 1271)     0.01600570513441007
(0, 1219)     0.0511703119570894 7
(0, 1209)     0.10234062391417895
(0, 1194)     0.0511703119570894 7
(0, 909)      0.03318224864003995
(0, 662)      0.022769929356223212
(0, 418)      0.04891806652384772
(0, 397)      0.03423295419235291
(0, 392)      0.024783841259467598
(0, 269)      0.033240367004725005
(0, 188)      0.04722498744523732
(0, 156)      0.023262873797037946
(0, 100)      0.02076954755505395
(0, 81)       0.015211595534715629
```