Model_builder

January 9, 2022

1 Đồ án 3: Fake news detection

1.1 Link trang web đã deploy

Tại đây

Lưu ý: Trang web có yêu cầu 2 input: văn bản và domain nguồn tin. Đối với domain, ta cần nhập vào domain_raw theo dạng `domain.tên_miền` mà không có https:// hay chứa các dấu '/' (VD: thanhnien.vn thay vì https://thanhnien.vn hoặc thanhnien.vn/nguon_tin.html)

1.2 Danh sách thành viên

MSSV	Họ Tên	Công việc
1712718	Huỳnh Thanh Sang	Lựa chọn và xây dựng model phù hợp
19120068	Dương Nam Hải	Lựa chọn và xây dựng model phù hợp
19120096	Lưu Gia Minh	Hợ trợ tiền xử lý dữ liệu, xây dựng streamlit
19120202	Võ Tiến Dũng	Tiền xử lý dữ liệu
19120267	Hoàng Dược Lam	Xây dựng pipeline cho streamlit, viết báo cáo

1.3 Import thư viện

[1]:
```
import os
import sys
import preprocessing
from preprocessing import *
import json
sys.executable
```

[1]:
```
'/home/hailinux/.conda/envs/test/bin/python'
```

1.4 Chiến lược thực hiện

Sau khi quan sát tập dữ liệu, nhóm nhận thấy domain của nguồn tin đóng vai trò rất quan trọng trong việc quyết định tin là thật hay giả. Vì thế, chúng em quyết định tạo vector đặc trưng page_rank để đánh giá độ tin cậy của domain. Vector này sẽ có vai trò rất lớn trong việc predict văn bản truyền vào.

1.5 Tiền xử lý dữ liệu

- Đọc dữ liệu từ file csv và lấy ra 3 thông tin **raw_document**, **raw_domain**, **raw_label**
- Trước đó, chúng em đã tiến hành xử lý và chấm điểm độ tin cậy của các domain, lưu trong file `page_rank.json`
- Quy đổi điểm của raw_domain dựa vào page_rank.json
- Tiền xử lý dữ liệu với các bước sau (thực hiện trong file 'preprocessing.py')
  - Chuẩn hóa Unicode tiếng việt - đưa văn bản về chuẩn Unicode dựng sẵn
  - Chuyển câu tiếng việt về chuẩn gõ dấu kiểu cũ (dùng `òa úy` thay `oà uý`)
  - Tách từ tiếng Việt thành các từ đơn và từ ghép (với sự hỗ trợ của underthesea)
  - Đưa toàn bộ văn bản về chữ thường
  - Xóa các ký tự không cần thiết (các dấu ngắt câu, số đếm,...)
  - Xóa các khoảng trắng thừa
  - Loại bỏ các stopword (những từ xuất hiện nhiều trong các văn bản - vì nó không ảnh hưởng đếu hiệu suất của việc phân loại, nên ta có thể bỏ đi để giảm thời gian training model)

[2]:
```
# Lấy dữ liệu từ file csv
raw_document, raw_domain , raw_label = read_data("data/data.csv")

# Lấy phần page_rank đã chấm điểm trước đó
f = open("page_rank.json", "r")
pr = json.loads(''.join(f.readlines()))
page_rank = []

# Thực hiện đổi phần raw_domain thành điểm dựa vào page_rank.json
for dm in raw_domain:
    try:
        page_rank.append(pr[dm])
    # Nếu domain không tồn tại trong page_rank.json, điểm của domain sẽ được␣
↪gán là 0
    except KeyError as e:
        page_rank.append(0.0)
print(page_rank)

# Tiền xử lý dữ liệu
preprocess_document = list(map(text_preprocess, raw_document))
```

```
[2.6, 0.0, 0.0, 0.0, 0.0, 2.53, 2.53, 0.0, 0.0, 3.71, 2.31, 2.31, 2.34, 2.31,
0.0, 4.55, 4.55, 0.0, 4.21, 4.53, 4.53, 2.99, 4.6, 0.62, 4.24, 4.24, 0, 4.53,
4.53, 4.53, 4.53, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79,
2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79,
2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79,
2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79,
2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 2.79,
2.79, 2.79, 2.79, 2.79, 2.79, 2.79, 0.0, 0.0, 4.76, 4.76, 4.76, 4.76, 4.76,
4.78, 5.24, 4.59, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 4.24,
5.15, 5.15, 5.15, 5.15, 4.78, 4.78, 5.24, 5.24, 5.24, 5.24, 4.76, 4.78, 5.24,
```

```
5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 4.78,
5.24, 4.57, 5.24, 5.24, 5.24, 4.54, 5.24, 5.15, 5.24, 4.57, 5.24, 4.06, 5.24,
5.24, 5.24, 3.65, 3.65, 4.59, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24,
5.24, 5.24, 5.24, 5.24, 5.24, 4.57, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24,
5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24, 5.24,
5.24, 5.24, 5.24, 4.24, 4.24, 4.24, 4.24, 4.24, 4.24, 4.24, 4.24, 4.24, 5.23,
5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23, 5.23]
```

[5]:
```python
# Viết các phần văn bản đã tiền xử lý xuống file txt
# Hỗ trợ cho streamlit sau này
with open('preprocess_document.txt', 'w') as f:
    for item in preprocess_document:
        f.write("%s\n" % item)
```

Tiến hành Tokenizer văn bản, bằng cách lưu mỗi từ trong toàn bộ văn bản thành 1 ID, sau đó quy đổi các từ trong văn bản thành các ID đã có ở trên

[7]:
```python
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

tokenizer = Tokenizer(num_words=15000,
                      oov_token='<OOV>',
                      filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')

#tokenize toàn bộ văn bản thành
tokenihzer.fit_on_texts(preprocess_document)

print(tokenizer)

#lấy ra word index, là 1 dict chứa ID và các từ tương ứng
word_index = tokenizer.word_index

#print(word_index)
```

```
2022-01-09 04:43:03.495777: W
tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load
dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open
shared object file: No such file or directory
2022-01-09 04:43:03.495794: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
Ignore above cudart dlerror if you do not have a GPU set up on your machine.
```

```
<keras_preprocessing.text.Tokenizer object at 0x7fb9fddf0b50>
```

[8]:
```python
#Chuyển các từ trong preprocess_document thành các ID
sequences_data = tokenizer.texts_to_sequences(preprocess_document)

#Thêm các biến đệm, hoặc loại bỏ dữ liệu
#Mục đích: khiến toàn bộ các văn bản có cùng một độ dài chuẩn
```

```
sequences_data = pad_sequences(sequences_data, padding="post")
```

[9]:
```
import numpy as np
page_rank = np.array(page_rank).reshape(len(page_rank),1)
```

Ta chuẩn hóa dữ liệu theo kiểu chuẩn hóa min-max. final_data sẽ có dạng là một ma trận gồm 223 dòng văn bản (đã cung cấp trong data), với mỗi dòng chứa các giá trị sau:

- Từ giá trị 0 đến 1849 là thông tin đã được chuẩn hóa
- Giá trị 1850 là điểm của domain
- Giá trị 1851 là kết quả tin thật hay giả

[10]:
```
#Chuẩn hóa dữ liệu theo chuẩn hóa min-max
def normalization(array):
    return ( array - np.min(array) ) / (np.max(array) - np.min(array))
```

[11]:
```
# Chuẩn hóa dữ liệu
normalized_data = normalization(sequences_data)

final_data = np.hstack((normalized_data,
                        np.array(page_rank),
                        np.array(raw_label).reshape(-1,1).astype(int)))

print(final_data.shape)
```

```
(223, 1852)
```

[12]:
```
np.set_printoptions(suppress=True)
final_data
```

[12]:
```
array([[0.09985576, 0.23321868, 0.11683124, …, 0.        , 2.6       ,
        1.        ],
       [0.09985576, 0.02474204, 0.11683124, …, 0.        , 0.        ,
        1.        ],
       [0.26173305, 0.21169422, 0.05702874, …, 0.        , 0.        ,
        1.        ],
       …,
       [0.1765228 , 0.11616554, 0.34916232, …, 0.        , 5.23      ,
        0.        ],
       [0.02252302, 0.00288472, 0.07733274, …, 0.        , 5.23      ,
        0.        ],
       [0.58959281, 0.06790192, 0.00321757, …, 0.        , 5.23      ,
        0.        ]])
```

[13]:
```
#save dữ liệu về file
np.savetxt('data/final_data.csv', final_data, delimiter=',')
```

1.5.1 Tách thành tập train và test

Tách dữ liệu thành tập train và tập test theo tỷ lệ 80:20, random_state=50

[14]:
```
import numpy as np
mydata = np.genfromtxt('data/final_data.csv', delimiter=',')
X = mydata[:,:-1]
y = mydata[:, -1]
```

[15]:
```
from sklearn.model_selection import train_test_split
import pickle
test_percent = 1/5
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=test_percent,
                                                    random_state=50)
```

[16]:
```
X_train.shape, X_test.shape, len(y_train), len(y_test)
```

[16]:
```
((178, 1851), (45, 1851), 178, 45)
```

1.6 Train model và so sánh kết quả

Thực hiện training theo 3 model: - Logistic Regression - MLP Classifier - SVM với kernel RBF

1.6.1 Logistic Regression

[17]:
```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import LogisticRegression
import time
start_time = time.time()
clf = LogisticRegression(solver='lbfgs',
                         multi_class='auto',
                         max_iter=10000).fit(X_train, y_train)

train_time = time.time() - start_time
print('Done training Linear Classifier in', train_time, 'seconds.')

# Save model
pickle.dump(clf, open("Model/linear_classifier.pkl", 'wb'))
```

```
Done training Linear Classifier in 0.09663009643554688 seconds.
```

[18]:
```
clf.score(X_train,y_train), clf.score(X_test,y_test)
```

[18]:
```
(0.9887640449438202, 0.9111111111111111)
```

1.6.2 MLP Classifier

[19]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification

start_time = time.time()

clf = MLPClassifier(hidden_layer_sizes=(2**11, 2**8, 2**4, 2**3),
                    learning_rate='adaptive',
                    batch_size=64,
                    verbose=True,
                    max_iter=100).fit(X_train, y_train)
train_time = time.time() - start_time
print('Done training MLP Classifier in', train_time, 'seconds.')

pickle.dump(clf, open("Model/MLP.pkl", 'wb'))
```

```
Iteration 1, loss = 0.70128353
Iteration 2, loss = 0.59516897
Iteration 3, loss = 0.42799757
Iteration 4, loss = 0.25890119
Iteration 5, loss = 0.13928159
Iteration 6, loss = 0.07312959
Iteration 7, loss = 0.03995604
Iteration 8, loss = 0.01923226
Iteration 9, loss = 0.00557211
Iteration 10, loss = 0.00389161
Iteration 11, loss = 0.00305137
Iteration 12, loss = 0.00263197
Iteration 13, loss = 0.00238609
Iteration 14, loss = 0.00226865
Iteration 15, loss = 0.00218895
Iteration 16, loss = 0.00213142
Iteration 17, loss = 0.00207965
Iteration 18, loss = 0.00203962
Iteration 19, loss = 0.00200683
Iteration 20, loss = 0.00197582
Iteration 21, loss = 0.00195473
Iteration 22, loss = 0.00193588
Iteration 23, loss = 0.00191962
Iteration 24, loss = 0.00190475
Iteration 25, loss = 0.00189209
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs.
Stopping.
Done training MLP Classifier in 37.891682624816895 seconds.
```

[20]:
```
clf.score(X_train,y_train), clf.score(X_test,y_test)
```

[20]: (1.0, 0.8222222222222222)

1.6.3 SVM với kernel RBF

[21]:
```
from sklearn.svm import SVC
start_time = time.time()
clf = SVC(kernel='rbf', gamma=0.001, C=1000).fit(X_train, y_train)
train_time = time.time() - start_time
print('Done training SVC Classifier in', train_time, 'seconds.')

pickle.dump(clf, open('Model/svm_rbf_kernel.pkl', 'wb'))
```

```
Done training SVC Classifier in 0.07808637619018555 seconds.
```

[22]:
```
clf.score(X_train,y_train), clf.score(X_test,y_test)
```

[22]: (1.0, 0.9111111111111111)

1.6.4 Đánh giá

- Cả 3 model với test_size = 20%, đều cho accuracy khá tốt (trên 80%).
- Với Logistic Regression và SVM với kernel RBF, 2 model cho accuracy cao hơn so với MLP Classifier. Tuy nhiên, ta chưa đủ căn cứ để đánh giá trong 3 mô hình, mô hình nào thật sự tốt hơn.

1.7 Xây dựng pipeline cho streamlit

Phần code được thực hiện trong file 'pipeline.py' để hỗ trợ cho việc xây dựng streamlit, gồm các bước như sau: - Đọc dữ liệu đã xử lý từ 'preprocess_document.txt' và tiến hành tokenize. - Từ phần văn bản đã được tokenize, biến đổi phần văn bản nhập vào thành list các ID, thêm các khoảng ID 0 hoặc cắt bớt văn bản sao cho văn bản sau khi xử lý sẽ có độ dài là 1850. - Chuẩn hóa min-max đoạn văn bản đã xử lý . - Tìm điểm của domain và thêm vào cuối array văn bản đã xử lý. - Dự đoán tin thật hay giả và thêm vào cuối array của văn bản.

1.8 Tài liệu tham khảo

- tensorflow, underthesea, scikitlearn, Protonx document
- nguyenvanhieu.vn