ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN 1

Phát hiện mail rác

Môn: Trí tuệ nhân tạo cho an ninh thông tin

Người thực hiện:
21120205 - Nguyễn Tạ Bảo
21120511 - Lê Nguyễn

Thành phố Hồ Chí Minh - 2024

Mục lục

Danh sách hình vẽ

Chương 1

Phân tích dữ liệu

Nhóm thực hiện load dữ liệu bằng thư viện `pandas` của Python:

```
train_data = pd.read_csv ('./train.csv')
test_data = pd.read_csv ('./val.csv')
```

Theo quan sát, dữ liệu sẽ gồm một cột trống để thể hiện index của từng dòng dữ liệu, các cột còn lại lần lượt sẽ là `Message ID, Subject, Message, Spam/Ham, Split`. Trong đó `Spam/Ham` là label để chỉ ra một email là rác hoặc không, còn cột `Subject` và `Message` dùng để chỉ nội dung và tiêu đề của email. Phía dưới là 5 dòng đầu của tập dữ liệu trong file `train.csv`.

Message ID	Subject	Message	Spam/Ham
0	christmas tree farm pictures	NaN	ham
1	vastar resources , inc .	gary , production fr ...	ham
2	calpine daily gas nomination	- calpine daily gas ...	ham
3	re : issue	fyi - see note below ...	ham
4	mcmullen gas for 11 / 99	jackie , since the i ...	ham

Bảng 1.1: 5 dòng đầu của tập dữ liệu trong file `train.csv`, phần `Message` đã được bỏ bớt ký tự (ngoài ra cũng đã loại bỏ một vài cột của dữ liệu).

Để xử lý dữ liệu và chuẩn bị dữ liệu cho phần mô hình, nhóm sẽ làm thông qua 3 bước:

1. Loại bỏ các phần có dữ liệu bị hư (`NaN`) và thay thế bằng chuỗi rỗng.

   ```
   train_data['Subject'] = train_data['Subject'].fillna ('')
   test_data['Subject'] = test_data['Subject'].fillna ('')

   train_data['Message'] = train_data['Message'].fillna ('')
   test_data['Message'] = test_data['Message'].fillna ('')
   ```

2. Nhóm tiến hành gộp cột `Subject` và `Message` để tạo thành một cột duy nhất `Text`, điều này sẽ giúp mô hình dễ xử lý hơn, khi ta có một văn bản duy nhất và đưa thành một vector đặc trưng, nếu 2 văn bản có thể đưa ra 2 vector đặc trưng và gây khó cho mô hình:

   ```
   train_data['Text'] = train_data['Subject'] + ' ' + train_data['Message']
   test_data['Text'] = test_data['Subject'] + ' ' + test_data['Message']
   ```

3. Chia ra tập train và tập test:

```
X_train = train_data['Text']
Y_train = train_data['Spam/Ham']

X_test = test_data['Text']
Y_test = test_data['Spam/Ham']
```

Chương 2

Mô hình

2.1 Vector đặc trưng

Ta đã biết các mô hình học máy chỉ có thể nhận đầu vào là số, do đó trước khi đưa vào mô hình, các đoạn email, lúc này là một cột `Text` sẽ được chuyển thành một vector số, gọi là vector đặc trưng. Để tìm vector đặc trưng này, nhóm sử dụng kỹ thuật TF-IDF bằng thư viện `scikit-learn` như sau:

```
# Trích xuất đặc trưng bằng TF-IDF
# Giới hạn số lượng từ vựng tối đa là 5000
tfidf = TfidfVectorizer(max_features=5000)

X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

Ta thấy `max_features` chính là số chiều của vector đặc trưng được tạo bằng kỹ thuật TF-IDF, việc gán giá trị này càng cao thì chiều của vector càng lớn, ngoài việc làm nặng phần tính toán, thì vector càng lớn dẫn đến vector càng thưa, làm giảm hiệu suất của mô hình, sau nhiều lần thử nghiệm thì giá trị 5000 được chọn.

Hiểu kĩ hơn, $\text{TF}(t, d)$ hay *term-frequency* của từ t trong văn bản d là tần số mà từ t xuất hiện trong văn bản d, khi ta chọn `max_features = n` thì `scikit-learn` sẽ chọn `n` từ có TF cao nhất. Tiếp theo IDF được thư viện `scikit-learn` tính bằng công thức:

$$\text{IDF}(t) = \log\left(\frac{N}{\text{DF}(t)+1}\right)$$

Trong đó N là tổng số email (hay còn gọi là corpus) mà ta đưa vào `TfidfVectorizer` và $\text{DF}(t)$ là document-frequency của từ t, tức là số văn bản (hay email) mà t xuất hiện trong toàn bộ corpus. Cuối cùng, giá trị TF-IDF của một từ t của một email d sẽ được tính bằng:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t)$$

Vậy với mỗi email d được đưa vào, ta tạo một vector gồm `n` chiều (với `n` được chọn ở `max_features`), mỗi chiều tương ứng một từ mà `scikit-learn` chọn cho ta. Sau đó tại mỗi từ t_i, ta tính $\text{TF-IDF}(t_i, d)$ và gán vào vị trí thứ i trong vector, cuối cùng ta được một vector đặc trưng tương ứng với email d.

2.2 Huấn luyện mô hình

Nhóm chọn mô hình *Multinomial Naive Bayes* để phân loại email. Trước đó, ta phải tìm hiểu qua Naive Bayes, giả sử ta có M lớp, ta hỏi rằng xác suất một điểm dữ liệu $\mathbf{x} = (x_1, x_2, \ldots, x_N)$

gồm N chiều có lớp i là bao nhiêu, tức là tìm xác suất $p(i \mid \mathbf{x}) = p(i \mid x_1, x_2, \ldots, x_N)$. Vậy để tìm xem $\mathbf{x}$ thuộc lớp nào, ta chỉ cần tìm $p(i \mid \mathbf{x})$ lớn nhất.

Áp dụng công thức Bayes, ta có:

$$p(i \mid x_1, x_2, \ldots, x_N) = \frac{p(x_1, x_2, \ldots, x_N \mid i)p(i)}{p(x_1, x_2, \ldots, x_N)}$$

Do mẫu số là một hằng số, thế nên nếu $p(i \mid x_1, x_2, \ldots, x_N)p(i)$ lớn nhất thì $p(x_1, x_2, \ldots, x_N \mid i)$ phải lớn nhất. Ở kỹ thuật Naive Bayes, bất kể $p(x_n \mid i)$ với $n \in [1, N]$ nào thì ta đều giả sử các xác suất ấy độc lập với nhau, tức là:

$$p(x_1, \ldots, x_N \mid i) = \prod_{n=1}^{N} p(x_n \mid i)$$

Do đó, ta cần tối đa xác suất dưới đây để tìm ra được lớp mà $\mathbf{x}$ thuộc về:

$$\prod_{n=1}^{N} p(x_n \mid i)p(i)$$

Do đó, ta chỉ cần tính $p(x_n \mid i)$ là xong. Ở mô hình mà nhóm chọn, Multinomial Bayes, $p(x_n \mid i)$ sẽ được tính bằng cách sau:

$$p(x_n \mid i) = \frac{T_{in}}{T_i}$$

Ta đặt x_n là từ thứ n trong vector đặc trưng của ta và i là nhãn (rác hay không rác). Khi đó T_{in} là tổng số lần từ x_n xuất hiện trong các email mà có nhãn là i. Còn T_i là tổng độ dài của các email mà có nhãn là i. Thế nhưng đôi khi từ n là từ chưa bao giờ xuất hiện trong corpus, thì khi đó $p(x_n \mid i) = 0$, để tránh điều này, ta dùng thêm siêu tham số α, khi đó:

$$p(x_n \mid i) = \frac{T_{in} + \alpha}{T_i + N\alpha}$$

Theo `scikit-learn`, α sẽ có giá trị là 1 và chúng em chọn các siêu tham số mặc định, mô hình được cài đặt như dưới:

```
model = MultinomialNB()
model.fit(X_train_tfidf, Y_train)
```

Sau khi đã huấn luyện mô hình, nhóm lưu lại tham số của mô hình và cả vector đặc trưng thông qua thư viện `joblib` để có thể dùng ở phần thử nghiệm thực tế.

```
joblib.dump(model, 'spam_classifier_model.pkl')
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
```

Tiếp theo nhóm đánh giá mô hình bằng các độ đo là Accuracy, Precision, Recall và F1. Để hiểu rõ, đầu tiên ta nhìn qua Confusion Matrix như dưới:

	Lớp dự đoán	Class = 1	Class = 0
Lớp thật	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

Trong đó:

- f_{11} dự đoán đúng lớp 1, còn gọi là TP (True positive).
- f_{10} dự đoán lớp 1 thành 0, còn gọi là FN (False negative).
- f_{01} dự đoán lớp 0 thành 1, còn gọi là FP (False positive).
- f_{00} dự đoán đúng lớp 0, còn gọi là TN (True negative).

Khi đó, các độ đo được tính như sau:

$$\text{Accuracy} = \frac{\text{số mẫu dự doán đúng}}{\text{tổng mẫu}} = \frac{TN + TP}{TN + TP + FP + FN}$$

Thế nhưng accuracy sẽ không còn đúng nếu tỉ lệ dữ liệu của mỗi nhãn không đều nhau, do đó nhóm dùng thêm độ đo:

$$\text{Precision} = p = \frac{TP}{TP + FP}$$
$$\text{Recall} = r = \frac{TP}{TP + FN}$$
$$\text{F1} = \frac{2pr}{p + r}$$

Ở phía dưới là phần cài đặt:

```
# Dự đoán nhãn của tập kiểm thử
Y_pred = model.predict(X_test_tfidf)

accuracy = accuracy_score(Y_test, Y_pred)
precision = precision_score(Y_test, Y_pred , pos_label='spam')
recall = recall_score(Y_test, Y_pred, pos_label='spam')
f1 = f1_score(Y_test, Y_pred , pos_label='spam')

print(f"Accuracy: {accuracy}")
# Accuracy: 0.9818417639429312
print(f"Precision: {precision}")
# Precision: 0.9753943217665615
print(f"Recall: {recall}")
# Recall: 0.9891234804862444
print(f"F1 Score: {f1}")
# F1 Score: 0.9822109275730623
```

Nhóm đạt được Accuracy là 0.98, Precision là 0.975, Recall là 0.989 và F1 là 0.98 ở trên tập test. Làm tương tự với tập val, nhóm chung em được Accuracy là 0.98, Precision là 0.975, Recall là 0.989 và F1 Score 0.982.