<정보보호와시스템보안 프로젝트1 보고서>

팀원 : 장진우/20171694/소프트웨어학부
박영신/20202082/정보보안암호수학과
정혜승/20202105/정보보안암호수학과
작성자 : 장진우

<파일 주의사항>

- 아래의 내용은 README.md에도 나와 있는 내용입니다.
  - 실행하실 때는 testCode 폴더 안에 있는 Base_code를 실행시키면 됩니다.
  - personalSubmissions 폴더 안에는 각자가 만들어 본 코드가 있습니다.
    - ▶ 이는 각자 제대로 이 과제를 해봤다는 의미로 모아두었습니다.
    - ▶ 각 조원들이 자신들이 이해한 내용을 바탕으로 만든 코드라서 내용이 다릅니다.
    - ▶ 제대로 된 제출용 코드는 testCode 안에 있는 Base_code입니다. 이 코드는 속도 테스트 결과 가장 빨리 작동한 장진우 학생의 코드가 들어있습니다.

<작업 내용>

- 불필요한 웹 로그 부분을 읽지 않게 parsing 함수를 변경했습니다.
- 랜덤포레스트가 학습을 Trigram 단위로 할 수 있도록 TfidfVectorizer 함수에 추가적인 파라미터를 제공했습니다.

<파일 설명>

- 기본적으로 파일 자체는 ecampus에 올라와있는 베이스코드를 그대로 가져다 썼습니다.
- 파일에서 다른 부분은 두 군데입니다.

```python
def parsing(path):  # 파싱을 진행하는 함수
    with open(path, 'r', encoding='utf-8') as f:
        # 파일을 읽어드리고 ['로그','로그',...] 이런식으로 로그를 구조화
        # 아래 내용이 해석하기 좀 복잡하긴 한데,
        # 메서드 이름과 URL, POST나 PUT의 경우 body를 긁어오는 코드라고 보면 됩니다.
        train = []
        para = ""
        while True:
            l = f.readline()
            if not l:
                break  # 파일을 전부 읽으면 읽기를 중단합니다.
            if l != "\n":
                para += l
                while l != "\n":
                    l = f.readline()
                if para[:4] == 'POST' or para[:3] == 'PUT':
                    para += f.readline()
                train.append(para)
                para = ""
    return train
```

- 첫 번째 부분은 원본 Base_code에 있는 parsing 함수입니다.
- 기존의 코드는 이 함수로 웹로그의 모든 글자를 전부 저장해서 학습을 시켰는데, 로그를 살펴본 결과 이 로그는 HTTP 메소드들 옆에 붙어있는 URL, 그리고 POST나 PUT의 경우 아래쪽에 추가로 붙어 있는 BODY 부분을 제외하고는 다 같은 내용이거나 별로 쓸모없다는 사실을 알아냈습니다. 이 parsing 함수는 정확히 그 부분만 읽어내기 위한 함수입니다.
- 코드의 작동 방식
  - 기본적으로 한 번의 반복에 하나의 로그를 처리하도록 만들었습니다.
  - 먼저 l = f.readline()으로 로그를 한 번 읽습니다
    - ▶ 여기서 if not l에 걸린다면, 파일이 끝난 것이므로 함수를 종료합니다.
  - 이후 처음 읽은 줄은 l에 저장하고, 이번에 읽은 로그의 HTTP 메소드가 어떤 것이냐에 따라서 행동이 갈립니다.
    - ▶ GET인 경우

```
GET http://localhost:8080/tienda1/index.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like
Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml
+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=19581AE7FCCA7BAFDF4E18EF38B4C20D
Connection: close
```

- GET의 경우 맨 처음의 줄 외에는 전부 중복되거나 필요 없는 내용입니다. 따라서 첫 줄을 제외한 모든 줄을 버립니다.

- 따라서 Wn이 나올 때까지 readline으로 내려가고, 맨 첫 줄만 저장된 para를 리턴할 리스트에 추가합니다.

▶ 그 외의 경우 (POST, PUT)

```
POST http://localhost:8080/tienda1/publico/registro.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.8 (like
Gecko)
Pragma: no-cache
Cache-control: no-cache
Accept: text/xml,application/xml,application/xhtml
+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Encoding: x-gzip, x-deflate, gzip, deflate
Accept-Charset: utf-8, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: localhost:8080
Cookie: JSESSIONID=367891214D3D939AB37BC025F7D6A205
Content-Type: application/x-www-form-urlencoded
Connection: close
Content-Length: 248

modo=registro&login=gdowik2&password=29%3Fem2ti3a&nombre=Grecia&apellidos=Ar%E1n
+Pola&email=fravel%40filtraciondehumos.sz&dni=48171319G&direccion=El+Vito%2C
+90+&ciudad=Ampolla%2C+L%27&cp=50100&provincia=Castell
%F3n&ntc=1798713909636768&B1=Registrar
```

- POST와 PUT은 전부 추가적인 BODY가 존재합니다. 따라서 맨 첫 줄과 맨 마지막 줄을 읽어야 합니다.
- 따라서 Wn이 나올 때까지 readline으로 내려간 다음, 한 줄을 더 읽어서 para에 저장한 다음 리스트에 추가합니다.

■ 마지막으로, 저장된 리스트를 반환합니다. 이렇게 하면 필요한 만큼의 글자들만 읽어들일 수 있게 됩니다.

```
def vectorize(train_x, test_x):  # 문장을 벡터로 만듭니다 해당 코드에서는
    tf = TfidfVectorizer(analyzer="char", ngram_range=(3, 3))
```

● 두 번째 부분은 vectorize 함수에 있던 TfidfVectorizer 함수입니다.

● 이 부분은 대학원 영상을 보고 아이디어를 얻었는데, Unigram이나 Bigram보다는 Trigram이 더 효율적으로 학습을 시킬 수 있다는 부분을 참고해서 수정했습니다.

● TfidfVectorizer 함수에 analyzer = 'char'을 파라미터로 넣어서 학습을 글자 단위로 설정하였으며, ngram_range=(3,3)도 파라미터로 설정하여 글자를 3개씩 묶은 trigram을 학습 단위로 설정하였습니다.

■ ngram_range=(2,3)이나 ngram_range=(3,5), ngram_range=(4,4)같은 것으로도 테스트를 해보았는데, 결과적으로는 ngram_range=(3,3)이 가장 높은 정확도를 보여줬습니다.

<결과>

```
E:\Python\python.exe E:/Github/webLogSeparater/testCode/Base_code.py
No. 1
0.9910750839269631
0.989111976825492
No. 2
0.9905019241791534
0.9883883883883883
No. 3
0.9902562842872349
0.9881035689293213
No. 4
0.9909932039629903
0.989010989010989
No. 5
0.9913207238188815
0.9894127047542949
```

- Accuracy score(위쪽)는 평균 0.99082944403504464, 약 99% 정도로 나왔고,
- f1 score(아래쪽)는 평균 0.9888055255816971, 약 98.8% 정도로 나왔습니다.