Week3

Created By	DongGu Kim
Last Edited	@Apr 10, 2020 11:04 PM
Property	연구방법론 추가, 기초개발환경 설치
Tags	Reasearch

Week03 - 개발환경 설정 및 EDA

1. 연구방법 참고자료 보강

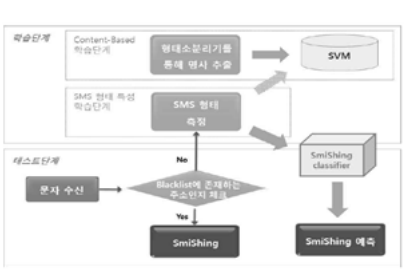
2. 주피터노트북 → 구글 코랩 전환

3. 탐색적 데이터 분석 기초

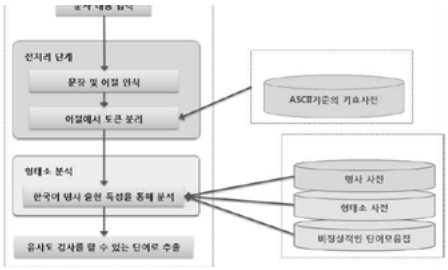
1. 연구방법 참고자료 보강

1. SVM을 이용한 스미싱 탐지기법 (이지원 외 2인. 2013)
    - 형태소 분리를 이용한 명사 추출
        - 스미싱 문자는 “인출”, “대출”, “공짜”와 같은 단어가 content based filtering에서 검출되는 것을 막기 위해 단어 사이에 아스키 코드를 삽입하여 검출되기 어렵게 만든다(ex. 공ㅉr). 일반 문자와 스미싱 문자간의 형태 차이가 분명히 존재하고, 문자 내용을 분석하여 이에 따라 SVM(support vector machine)을 분류했다. 형태소 분석, 워드임베딩을 진행하여 문자 메시지별 종류를 학습해야 한다. 연구 결과도 TP Rate가 90%를 상회하는 것을 참고하면, 유의미한 과정이라고 할 수 있다.

            참고자료



[그림 5] 스미싱 분류를 위한 구조
[Fig. 5] Structure of Smishing Classifier



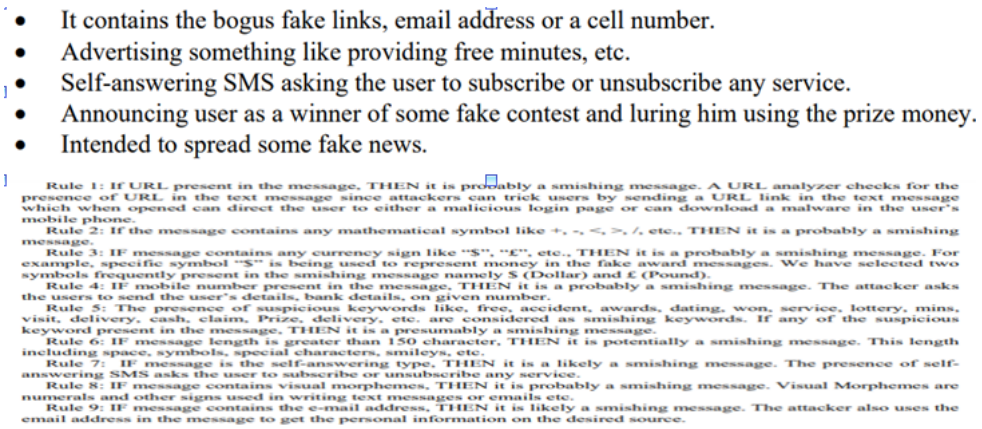
[그림 6] 형태소 분리기 구성
[Fig. 6] Structure of Morpheme Analyzer

```
case 1 "Mssage Type"
        SMS : 1, MMS : 2
case 2 "Message Attribute"
        TRUE, FALSE  /* 문자 메시지 내용중에 URL이나 번호가 있을 경우 TRUE */
case 3 "Number of Emoticons"
        INT  /* 0x80 이상값을 가지고 있는 확장 ASCII에 있는 이모티콘 */
case 4 "Phone Number Structure"
        TRUE, FALSE /* 대한민국 전화번호 체계에 맞는 수신번호 구조 여부 */
case 5 "Existence of apk"
        TRUE, FALSE /* HTML내의 Header tag 분석을 통해 apk 존재 여부 체크 */
case 6 "Number of word at Smishing message"
        INT /* 스미싱에서 많이 출현하는 단어 수 */
case 7 "Similarity between url domain and related company"
        TRUE, FALSE /* URL, 전화번호와 등록된 회사번호와 비교 */
case 8 "Received Time and Text"
        TRUE, FALSE /* 문자 메시지 수신시간과 내용 비교 */
```

[그림 8] 스미싱 탐지기준 값 세팅
[Fig. 8] The value setting of detection criteria for Smishing

2. Rule-Based Framework for Detection of Smishing Messages in Mobile Environment (Ankit Kumar Jain, B. B. Gupta, 2018)
    - 스미싱 문자 특징 분석
        - 스미싱 문자는 일반 문자와 대조되는 내용적 특징이 있다.

- 참고자료



- It contains the bogus fake links, email address or a cell number.
- Advertising something like providing free minutes, etc.
- Self-answering SMS asking the user to subscribe or unsubscribe any service.
- Announcing user as a winner of some fake contest and luring him using the prize money.
- Intended to spread some fake news.

Rule 1: If URL present in the message, THEN it is probably a smishing message. A URL analyzer checks for the presence of URL in the text message since attackers can trick users by sending a URL link in the text message which when opened can direct the user to either a malicious login page or can download a malware in the user's mobile phone.
Rule 2: If the message contains any mathematical symbol like +, -, <, >, /, etc., THEN it is a probably a smishing message.
Rule 3: IF message contains any currency sign like "$", "£", etc., THEN it is a probably a smishing message. For example, specific symbol "$" is being used to represent money in the fake award messages. We have selected two symbols frequently present in the smishing message namely $ (Dollar) and £ (Pound).
Rule 4: IF mobile number present in the message, THEN it is a probably a smishing message. The attacker asks the users to send the user's details, bank details, on given number.
Rule 5: The presence of suspicious keywords like, free, accident, awards, dating, won, service, lottery, mins, visit, delivery, cash, claim, Prize, delivery, etc. are considered as smishing keywords. If any of the suspicious keyword present in the message, THEN it is a presumably a smishing message.
Rule 6: IF message length is greater than 150 character, THEN it is potentially a smishing message. This length including space, symbols, special characters, smileys, etc.
Rule 7: IF message is the self-answering type, THEN it is a likely a smishing message. The presence of self-answering SMS asks the user to subscribe or unsubscribe any service.
Rule 8: IF message contains visual morphemes, THEN it is probably a smishing message. Visual Morphemes are numerals and other signs used in writing text messages or emails etc.
Rule 9: IF message contains the e-mail address, THEN it is likely a smishing message. The attacker also uses the email address in the message to get the personal information on the desired source.

3. 도출점
   - 스미싱 문자와 일반 문자는 '내용, 문법, 문자 길이' 등의 텍스트 자체의 차이가 분명히 존재한다. 이를 통해 스미싱, 일반 문자를 분류하는 모델링을 만들고자 한다.

---

2. 주피터노트북 → 구글 코랩 전환

1. tensorflow 설치 및 pip 자체 오류
   - 텍스트를 벡터화하는 토크나이저 과정에서 tensorflow 오류 발생
   - tensorflow 오류를 해결하기 위해 pip package업데이트 과정에서 오류 발생 → 복구하기 위해 업데이트 파일 제거하고 다시 실행하는 과정에서 무한 오류
   - pip 오류 발생 → 아나콘다 자체를 삭제하고 다시 설치하는 것에 부담감을 느낌
   - 이번 기회에 구글 코랩을 사용하는 것이 좋지 않을까 싶어서 코랩으로 전환함.

2. Mecab 라이브러리 이용
   - Konlpy에서 정확도, 속도면에서 가장 우수한 class는 Mecab이다.
   - Mecab은 윈도우 환경에서 설치가 안되는 단점이 있다.
   - 구글 코랩을 이용하면 윈도우 환경에서도 Mecab이 이용 가능하다.

```
!pip install git
! git clone [https://github.com/SOMJANG/Mecab-ko-for-Google-Colab.git](https://github.com/SOMJANG/Mecab-ko-for-Google-Colab
cd Mecab-ko-for-Google-Colab/
! bash install_mecab-ko_on_colab190912.sh
```

3. 코랩의 GPU 가속도 환경설정 이용가능
   - GPU를 통해 모델링을 하면 더 빠르게 무료로 진행할 수 있는데, 구글 코랩에서 GPU 환경에서 학습시킬 수 있다.

---

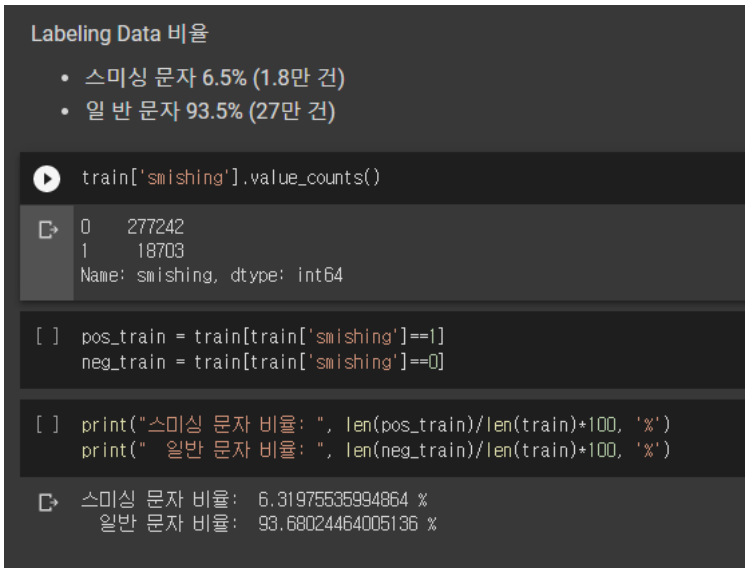
3. 탐색적 데이터 분석 기초

1. EDA 과정에 앞서서 문자메시지 내용을 형태소 분류를 진행한 칼럼을 추가했다.

```
from konlpy.tag import Mecab
mecab = Mecab()
from tqdm import tqdm_notebook
train['morph'] = 0

%%time
for idx in tqdm_notebook(range(len(train))):
  train['morph'][idx] = mecab.morphs(train['text'][idx])
```

2. Labeling Data 비율



3. 문자 메시지 길이 평균