Natural Language Processing with Disaster Tweets

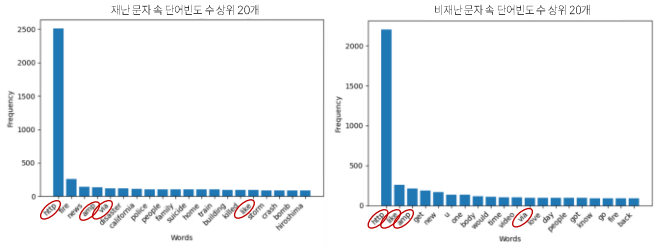
kaggle

부산대학교 통계학과 박소정 이호창 차영호

데이터 설명 : 트위터와 같은 SNS가 활발해지면서 긴급 재난 문자도 SNS를 통해서 전달 되는 경우가 늘고 있다. 해당 데이터는 국가에서 공식적으로 발송하는 재난 문자가 아닌 트위터를 통해 개인이 알리는 재난 문자로 국가에서 보내는 재난문자와 차이가 존재할 수 있다. 이러한 상황에서 개인이 알리는 재난 문자에는 어떠한 특징이 있는지 살펴보며 수많은 트윗 중 재난문자를 분류하고자 한다. (변수 : id, text, location, keyword,target) Natural Language Processing with Disaster Tweets | Kaggle

EDA

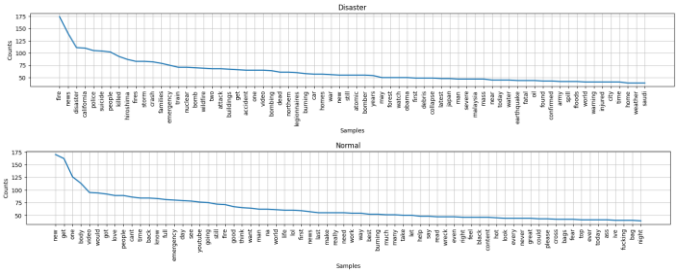
[전처리 전 train 데이터에서 text 변수의 재난문자와 비재난 문자 별 단어 빈도 수 확인]



- 재난 문자와 비재난 문자에서 모두 많이 나온 단어들 : http, amp, via, like

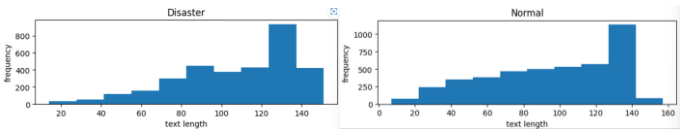
→ 학습 시 방해되는 요소라 판단하여 제거함

[위 단어들 제거 후 train 데이터에서 text 변수의 재난문자와 비재난 문자 별 단어 빈도 수 확인]



- 재난 문자가 비재난 문자보다 부정적인 단어들이 많이 보임 ex.killed, crash, wildfire 등

[text 변수 길이 비교]



- 재난 문자와 비재난 문자의 길이를 비교하기 위해 확인해 본 결과 차이가 없었음

→ 파생변수로 사용할 수 없음

Preprocessing

[전처리 과정1]

- 대문자를 소문자로 수정
- 중복처리 : text값은 같은데 target이 다른 경우
  1. text 값을 해석해서 target 값을 직접 수정
  2. 빈도수가 높은 target 값으로 처리 (ex. 5개의 똑같은 text가 있을 경우 1이 3개, 0이 2개가 나왔다면 1로 처리함)
- 아래의 문자들 공백으로 처리
  - (@\\[A-Za-z0-9]+): "@"로 시작하는 문자
  - ([^0-9A-Za-z \\t]): 알파벳, 숫자, 공백 및 탭 문자를 제외한 모든 문자
  - (\\w+:\\/\\/\\S+): URL
  - ^rt: 문자열의 시작이 "rt"인 문자
  - http.+?: " http " 로 시작하는 문자

[전처리 과정2]

- 숫자와 영어가 붙어있는 이상한 단어들을 분리하고, 숫자와 단어의 글자 수가 작은 것을 삭제
  - split_regex = r'([a-zA-Z]+|\\d+)' : 숫자와 영어를 구분
  - re.sub(r'\\d+', '', processed_sentence) : 숫자를 삭제
  - re.sub(r'\\b\\w{1,2}\\b', '', processed_sentence) : 단어의 글자 수가 2개 이하인 것 삭제

[사용하지 않는 변수]

- Keyword
  - disaster과 normal문자에서 나타나는 keyword의 차이가 확연하게 존재하였음
  - text에 keyword를 넣어서 사용해보기도하고, disaster/normal 각 각 에서의 keyword 빈도수를 이용해 후처리를 진행해보기도 하였음
  - 하지만 이 방법 모두 오히려 성능을 떨어트리는 결과를 가져오거나 변화가 없었음
- location
  - disaster와 normal 문자에서의 빈도수 그래프에서 알 수 있듯이, 눈에 보이는 차이가 없음
  - 두 문자 모두에서 가장 많이 나타나는 location은 nan, 즉 locatoin이 명시되어있지 않은 문자가 대다수였음.(나머지는 고만고만함)

[변수 토큰화 처리 방법]

- 토크나이저로 lemmatizer를 사용해 복수형을 단수형으로 과거형, 미래형을 현재형으로 비교급을 원형으로 변환된 token들을 얻음
  - 복수 → 단수
  - 과거, 미래 → 현재
  - 비교급 → 원형
- CountVectorizer를 통해 문자형을 수치형으로 변환하여 모델의 입력 데이터 셋을 만들었음
  - text내에서 특정 단어의 중요도를 구하는 TF-IDF도 사용해보았으나 CountVectorizer에 비해 좋지 않은 성능을 보였음

Modeling

- 사용 모델
  - RandomForestClassifier, MultinomialNB, LogisticRegression 모델 사용
  - 각각의 random_state는 모두 0으로 설정하고 iteration의 경우 1000으로 설정
- 앙상블
  - VotingClassifier를 사용했고 voting 방법은 가장 많이 나온 예측값으로 결정하는 hard 방법을 사용하였음

- 소정 : 자연어 처리를 접한 경험이 적어서 데이터를 다룰 때 어디서 부터 시작해야하는지 당황스러웠다. 하다보니 데이터 전처리가 매우 중요하다는 것을 깨달았다. 또한 모델링에서 앙상블을 사용하였더니 성능이 높아졌다. 그러나 예측하는 개수가 작아서 과적합이 덜 된 것일 수도 있다고 생각했다.
- 호창 : 자연어처리의 경우 모델의 사용보다 전처리가 더 중요하다고 느꼈고 다양한 모델을 사용해보고 싶었지만 전처리 과정에서 너무 많은 시간을 할애했다.
- 영호 : 처음하는거보다는 데이터분석에 익숙해진 것 같았다.. 물론 아직 많이 부족해서, 처음에 삽질을 많이 했지만 이번에는 유의미한 결과를 가져온 것 같아서 기분이 좋았다. 다음에는 챗 지피티 말고 내가 직접 문서같은 것을 보고 코드를 짜보고싶다!