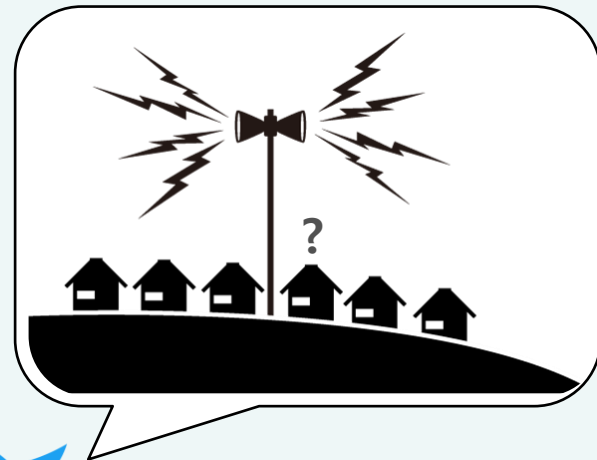


# REAL or NOT?

NLP 를 통한 자연재해 트윗 분류

이지은, 김00, 나00, 백00



## 목 차

01

프로젝트 선정 배경

업데이트된 BERT  
기반 NLP 를  
실험하기 위함

02

프로젝트 개요

프로젝트 timeline  
및  
사용기술 소개

03

데이터 탐색/정제

시각화 기반  
주어진 자료 탐색  
및 학습에 필요한  
문자열 변환

04

데이터 분석 /  
딥러닝

기초 모델과 BERT  
모델을 통한  
기계학습 수행

05

학습결과 /  
개선사항

Confusion Matrix  
를 통한 결과값  
비교

## 프로젝트 선정 배경

### BERT 개요

#### BERT는 구글의 검색어 처리 알고리즘임

- BERT 는 Bidirectional Encoder Representations from Transformers 의 약자
- 구글이 개발한 신경망 기반 자연어 학습 알고리즘
- 2018.11 오픈소스화 됨
- 기존 모델과 달리 단어 전후의 맥락을 고려한 유추가 가능

#### [기존 모델과 BERT 학습방식 비교]

예) I accessed the \_\_\_\_ account

I accessed the 'bank' (기존모델)

I accessed the 'bank' account (BERT)

### 트윗 의미 구분을 통한 BERT 성능측정

#### Kaggle 통한 성능 검증이 제일 용이하다고 판단

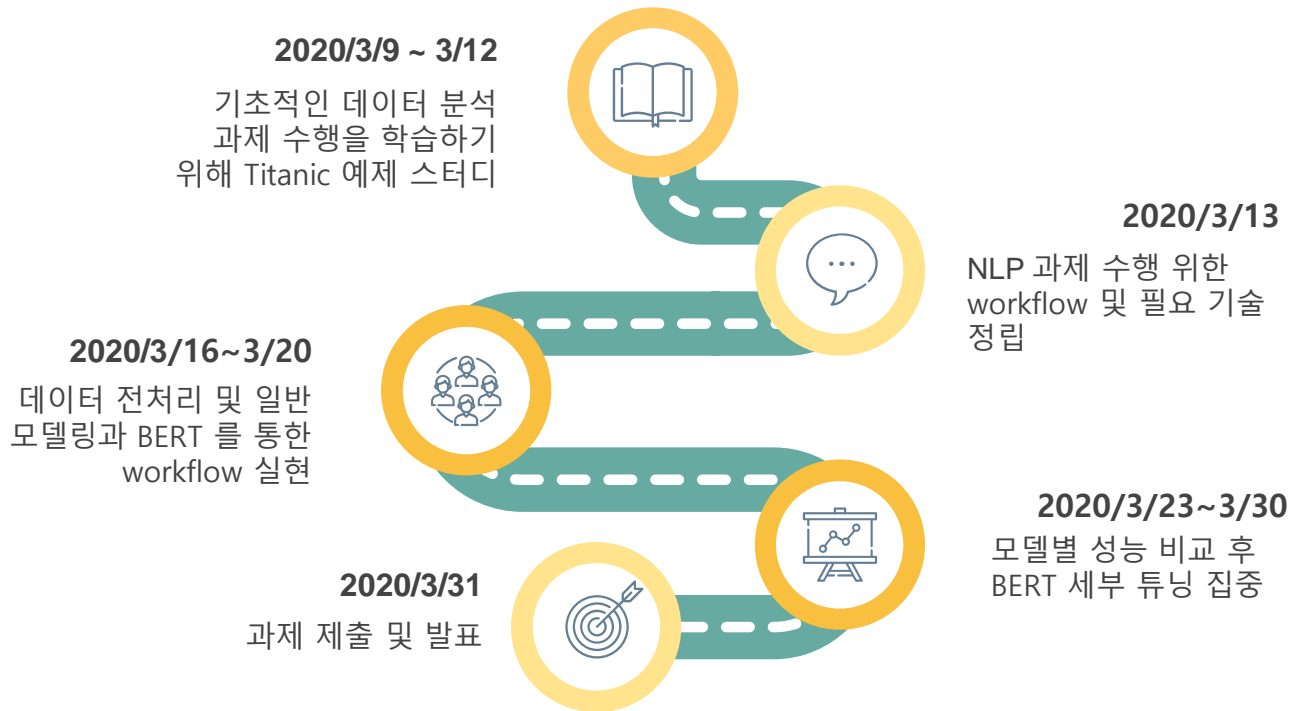
- 데이터수집에 대한 애로사항 없이 BERT 의 성능 측정에 집중가능
- 트위터 내용 중 재해 관련된 트윗을 학습하여 예측하는 Competition 선정해 수행

#### [Competition 개요]

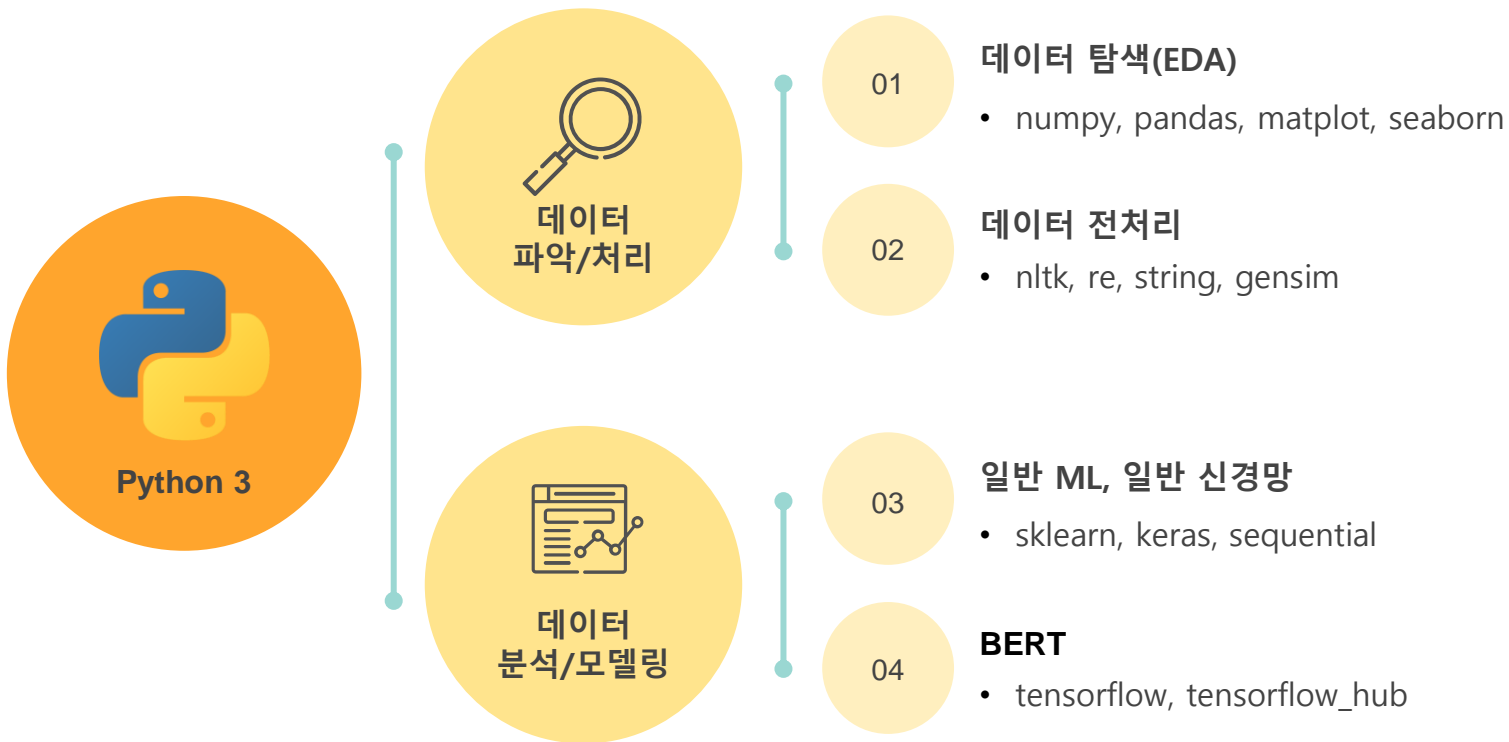
True True False True True



## 프로젝트 Timeline

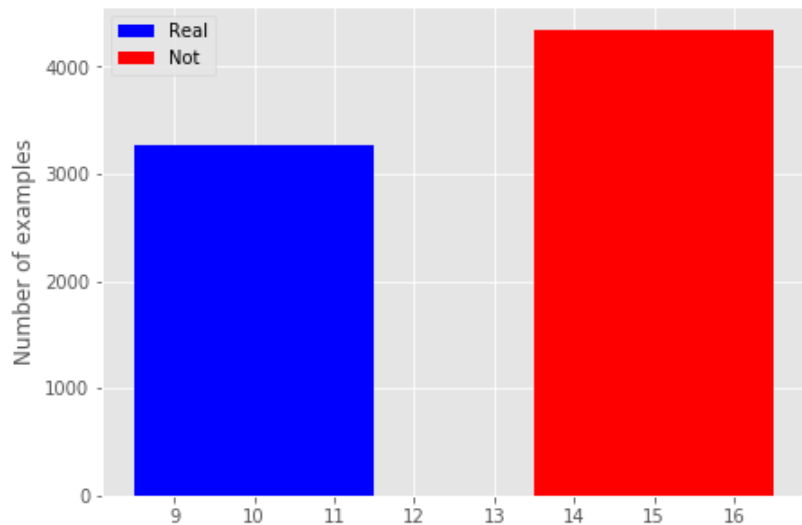


## 프로젝트 활용기술



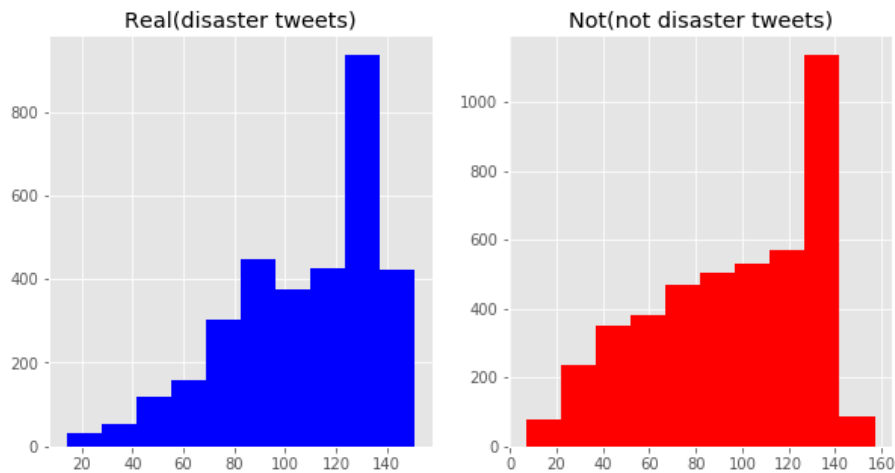
## 데이터 탐색(1/4): 데이터 파악

훈련데이터 트윗 분류현황



7,613개 Target 값 비중은 3:4 로 비등한 수준임

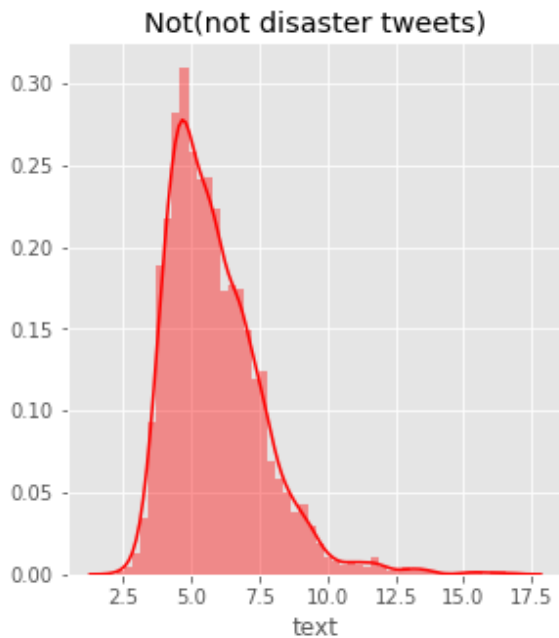
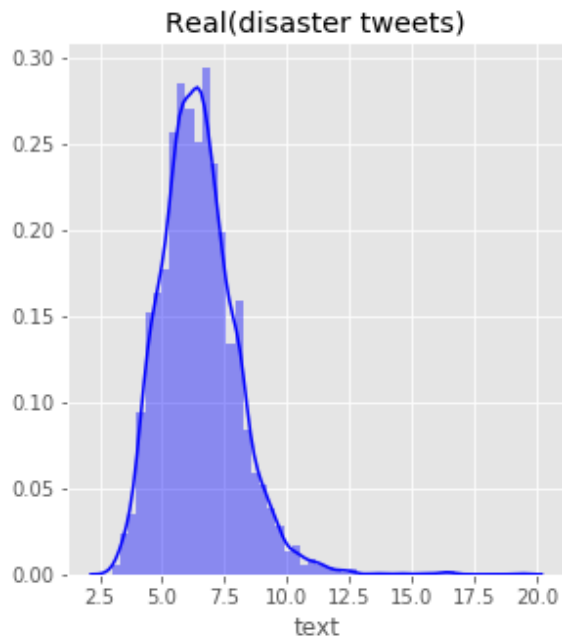
훈련데이터 트윗 글자수 분포



단어 길이 분포 유사함

## 데이터 탐색(2/4): 길이 파악

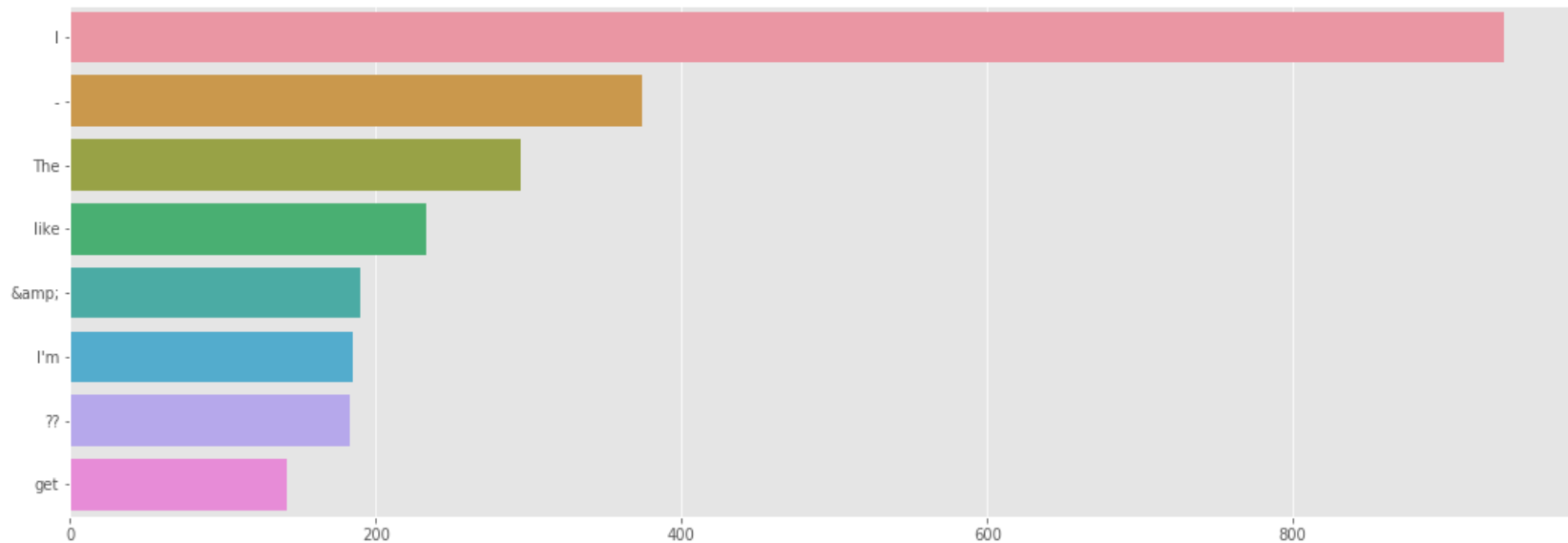
훈련데이터 트윗 평균 단어수



트윗당 단어 갯수 분포 유사함

## 데이터 탐색(3/4): stopwords, 특수문자 분포 파악

### 훈련데이터 내 stopwords, 특수문자 분포 현황

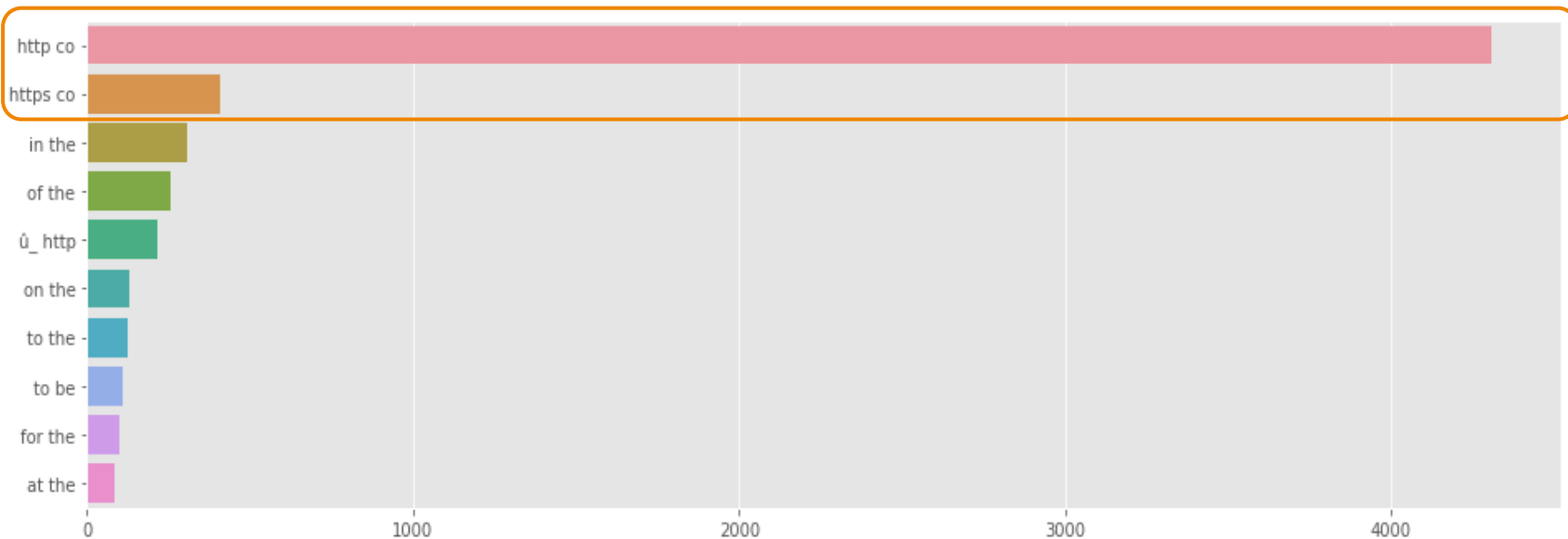


대명사, 관사, 문법부호, HTML 태그 및 줄임말 표현이 존재하므로 필터링 필요



## 데이터 탐색(4/4): n-gram 분석

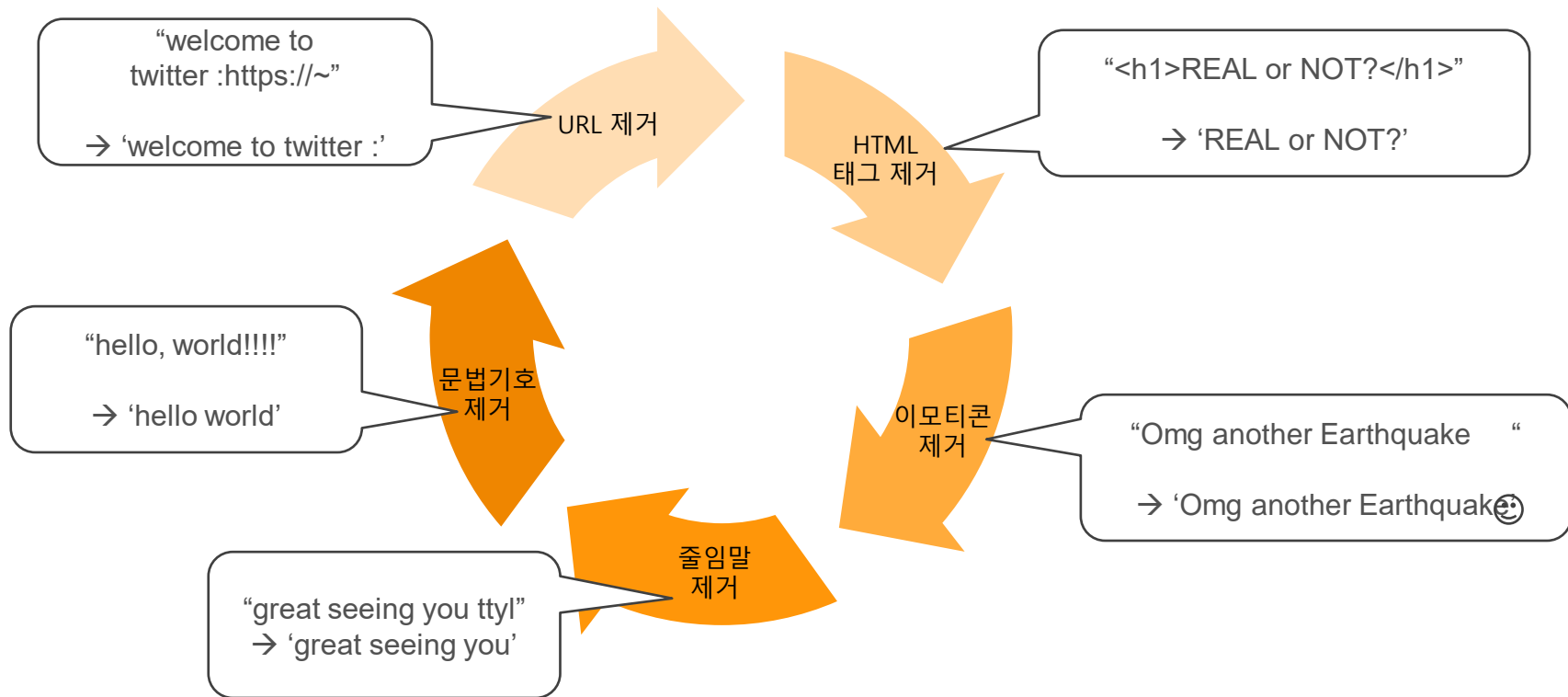
훈련데이터 내 두 단어로 구성되어 의미가 불명확한 문자열 분포 현황



url 구성 표현이 존재하여 일련의 단어 전부 필터링 필요

## 데이터 정제(1/4): Pre-processing

### 필터링 대상 선정



## 데이터 정제(2/4): Pre-processing

### 필터링 대상 선정

```
def remove_URL(text):  
    url = re.compile(r'https?://\S+|www\.\S+')  
    return url.sub(r'', text)
```

URL 제거

```
def remove_html(text):  
    html=re.compile(r'<.*?>')  
    return html.sub(r'', text)
```

HTML  
태그 제거

```
def remove_punct(text):  
    table=string.maketrans('', '', string.punctuation)  
    return text.translate(table)
```

문법기호  
제거

이모티콘  
제거

```
abbreviations = {  
    '$': 'dollar ',  
    '€': 'euro ',  
    '30+': 'for adults only',  
    'a.m.': 'before midday',  
    'a3': 'anytime anywhere anytime',
```

```
def convert_abbrev(word):  
    return abbreviations[word.lower()] if word.lower() in abbreviations.keys() else word
```

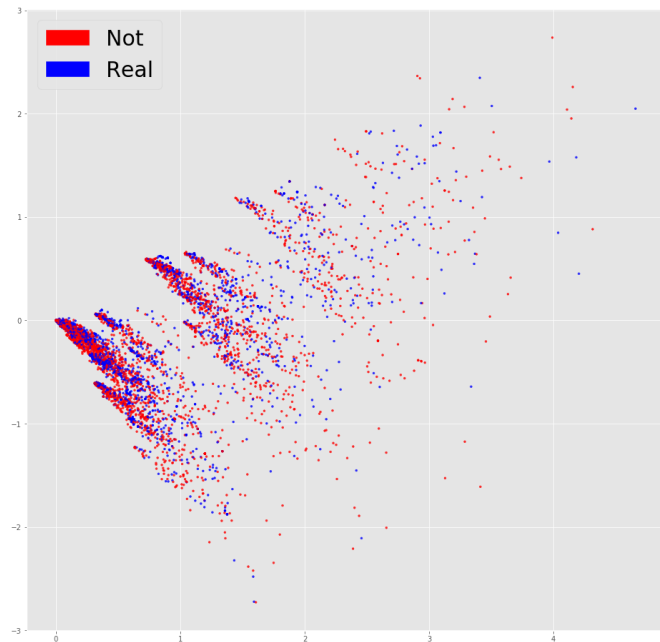
```
def convert_abbrev_in_text(text):  
    tokens = word_tokenize(text)  
    tokens = [convert_abbrev(word) for word in tokens]  
    text = ' '.join(tokens)  
    return text
```

임말  
제거

```
def remove_emoji(text):  
    emoji_pattern = re.compile("["  
        u"\U0001F600-\U0001F64F" # emoticons  
        u"\U0001F300-\U0001F5FF" # symbols & pictographs  
        u"\U0001F680-\U0001F6FF" # transport & map symbols  
        u"\U0001F1E0-\U0001F1FF" # flags (iOS)  
        u"\U00002702-\U000027B0"  
        u"\U000024C2-\U0001F251"  
    ]+", flags=re.UNICODE)  
    return emoji_pattern.sub(r'', text)
```

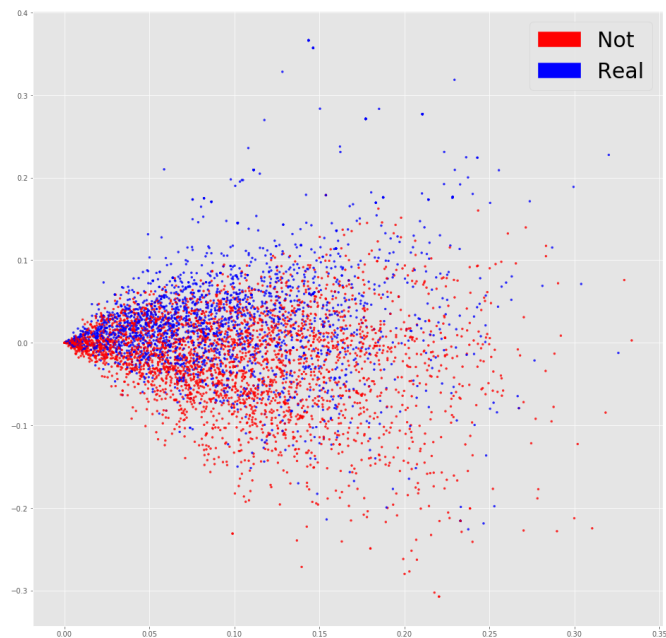
## 데이터 정제(3/4): 차원 축소

CountVectorization



분포가 고르지 않아 단어간 유기성 낮음

TF-IDF



빈도수에 따른 영향력을 측정해 반영

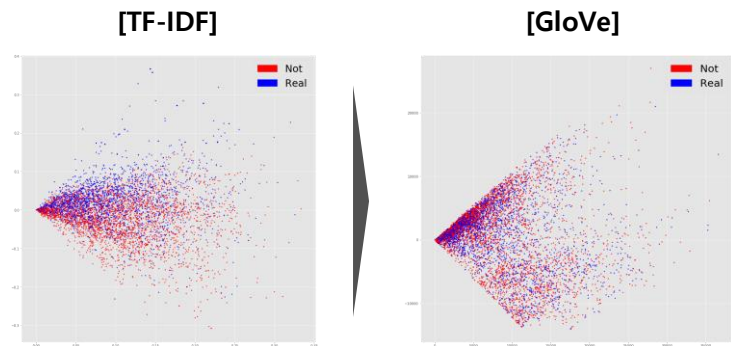
## 데이터 정제(4/4): GloVe 적용

### GloVe 개요

**GloVe 는 전체 단어를 분석하는 일에 최적화됨**

- GloVe 는 Global Vectors for Word Representation 의 약자임
- 비지도 학습을 통해 문자열 데이터의 벡터화를 수행
- Word2vec 대비 유추 작업 성능은 떨어지나 단어 전체를 고려한 분류 작업 성능이 뛰어남
- 영문 위키피디아 DB 를 통해 사전학습 후 훈련데이터 학습 진행

### GloVe 적용 결과



[기초 RNN 통한 두 데이터 적합도 비교]

차원축소 방식	TF-IDF & LSTM	GloVe & LSTM
Accurcay	0.5708	<b>0.7810</b>
Val_accuracy	0.5686	<b>0.8050</b>

## 데이터 분석: 일반 ML 모델링 (Logistic Regression)

### Logistic Regression 개요

#### 범주형 결과값의 분류를 위한 회귀분석

- 독립변수의 선형 결합으로 종속변수를 설명하는 점에서는 선형 회귀분석과 유사
- 다만 일반 선형 회귀와 달리 결과가 특정 분류로 나누어짐
- 분류값의 성공 확률을 종속변수로 정의 후 로짓변환으로 구현

$$(a) \text{ odds ratio} = \frac{p(y=1|x)}{1-p(y=1|x)}$$

$$(b) \text{ logit}(p) = \log \frac{p}{1-p}$$

$$(c) p_i = \text{logit}^{-1}(\beta \cdot \mathbf{X}_i) = \frac{1}{1 + e^{-\beta \cdot \mathbf{X}_i}}$$

### 적용 결과

#### [Logistic Regression\_코드]

```
lr_clf = LogisticRegression(max_iter=150,  
                             penalty='l2',  
                             solver='lbfgs',  
                             random_state=0)  
lr_clf.fit(X_train_onehot, y_train)  
lr_pred = lr_clf.predict(X_val_onehot)
```

#### [Logistic Regression\_요약]

F-measurement	0.6537
Accuracy	0.7931
Val_accuracy	0.7214

## 데이터 분석: 기초 Sequential 모델링 (GloVe + Keras)

### Keras 개요

#### Keras 의 Sequential 모델링을 통해 신경망 baseline 모델 생성

- Keras 는 Sequential 과 Functional 모델 두  
분류로 구성됨
- 전자는 전통적인 신경망 구조로 간단한  
코딩으로 모델 구현 가능
- 후자는 다중 입/출력값 및 레이어 재활용  
가능함
- 단어를 단일 벡터로 치환시켰으므로  
Sequential 모델링을 통한 단순 RNN 적용
- GloVe 통해 사전학습된 데이터를 입력

### 적용 결과

#### [Sequential\_코드]

```
model=Sequential()

embedding=Embedding(num_words,100,embeddings_initializer=Constant(embedding_matrix),
                    input_length=MAX_LEN,trainable=False)

model.add(embedding)
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

optimizer=Adam(learning_rate=3e-4)

model.compile(loss='binary_crossentropy',optimizer=optimizer,metrics=['accuracy'])
```

#### [Sequential\_요약]

F-measurement	0.7494
Accuracy	0.7883
Val_accuracy	0.8109

## 데이터 분석: BERT 모델링 - Baseline

### BERT 구현

#### 논문에 공표된 BERT 권장 사양 적용

- Pooling 없이 CLS 토큰(BERT 유형) 단일 적용
- Dense layer (FC layer) 없이 BERT 마지막 층에 반응함수(Sigmoid) 추가
- 학습률, epochs, optimizer 및 batch 크기 고정
  - 학습률 :  $2e - 5 \sim 5e - 5$
  - Epochs = 3
  - Adam
  - Batch 크기 = 16
    - 원래 32 이나 Kaggle Notebook 사양 (GPU 15.2GB)에 맞추어 16으로 축소 조정

### 적용 결과

#### [BERT\_코드]

```
# Baseline 하이퍼파라미터
random_state_split = 7
Dropout_num = 0
learning_rate = 3e-5
valid = 0.2
epochs_num = 3
batch_size_num = 16
target_corrected = False
target_big_corrected = False

checkpoint = ModelCheckpoint('model_BERT.h5', monitor='val_loss', save_best_only=True)

train_history = model_BERT.fit(
    train_input, train_labels,
    validation_split = valid,
    epochs = epochs_num, # recommended 3-5 epochs
    callbacks=[checkpoint],
    batch_size = batch_size_num
)
```

#### [BERT\_Baseline\_요약]

F-measurement	0.8442
Accuracy	0.8772
Val_accuracy	0.8339



## 1차 학습 결과

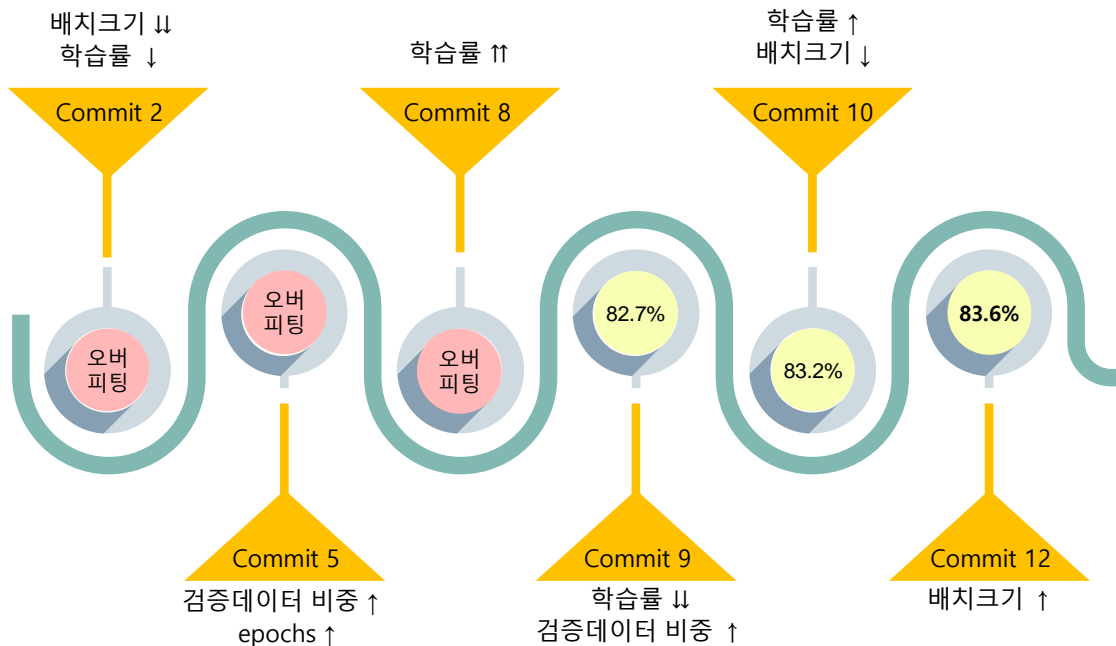
### Summary

모델명	Logistic Regression	GloVe (Sequential)	BERT
F-measurement	0.6537	0.7494	<b>0.8442</b>
Accurcay	0.7931	0.7883	<b>0.8772</b>
Val_accuracy	0.7214	0.8109	<b>0.8339</b>

별도의 튜닝을 하지 않음에도 불구하고 BERT 의 교차검증 정확도가 제일 높음

## 데이터 분석: BERT 모델링 – 세부 튜닝

### 세부 튜닝 과정



적정 epochs 및 검증데이터 비중 결정하여 오버피팅 제어 후, 학습률 미세 조정으로 성능 향상

# 참고자료

## 데이터 탐색 관련

- [N-gram](#)
- [NLP 임베딩 작동원리 및 필요성](#)
- [SVD](#)
- [PCA\(1\)](#)
- [PCA\(2\)](#)
- [LSA](#)
- [Word2vec vs GloVe](#)

## 기타 자료

- [GloVe 기초](#)
- [약어 처리](#)
- [토픽 모델링](#)
- [NLP 기초\(1\)](#)
- [NLP 기초\(2\)](#)
- [벤치마크 평가지표](#)
- [Confusion Matrix](#)
- [Checkpoint 기법](#)

## 기계학습 관련

- [로지스틱 회귀\(1\)](#)
- [로지스틱 회귀\(2\)](#)
- [Keras 기초](#)
- [Keras 임베딩](#)
- [Keras API](#)

## BERT 관련

- [BERT 개념\(1\)](#)
- [BERT 개념\(2\)](#)
- [BERT 개념\(3\)](#)
- [BERT 기초\(1\)](#)
- [BERT 기초\(2\)](#)
- [BERT 기초\(3\)](#)
- [BERT 기초\(4\)](#)
- [BERT 기초\(5\)](#)
- [BERT 초기사양](#)
- [BERT K-Fold](#)
- [BERT 튜닝](#)
- [How to Fine-Tune BERT for Text Classification?\(코넬대학교, 2019\)](#)
- [BERT 적정 batch 크기](#)