Real or Not?

NLP with Disaster Tweets

Team 8

Yan Sun

Chao Ma

Ruonan Ren

Overview

01 Use Cases

02 Goals

03 Methodology

04 TF-IDF

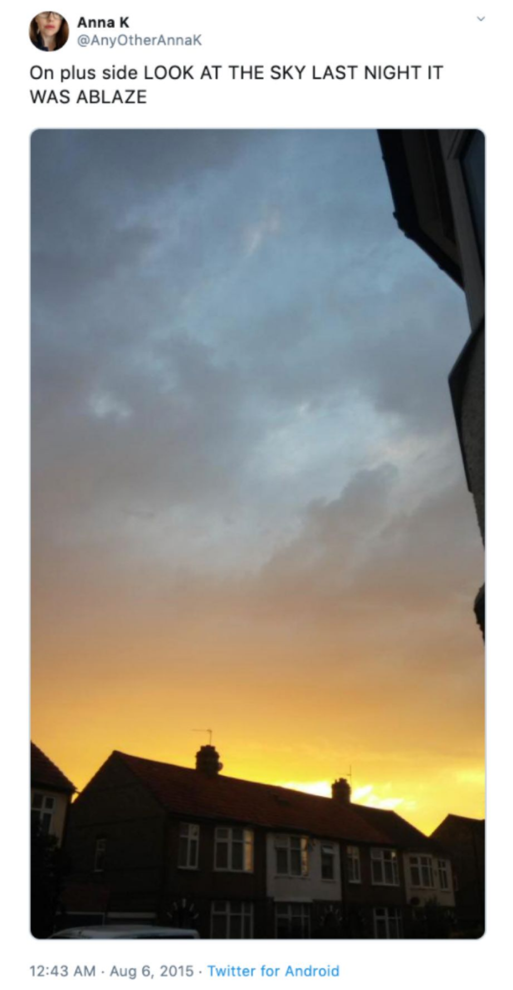
05 Classification Algorithms

06 Project Results

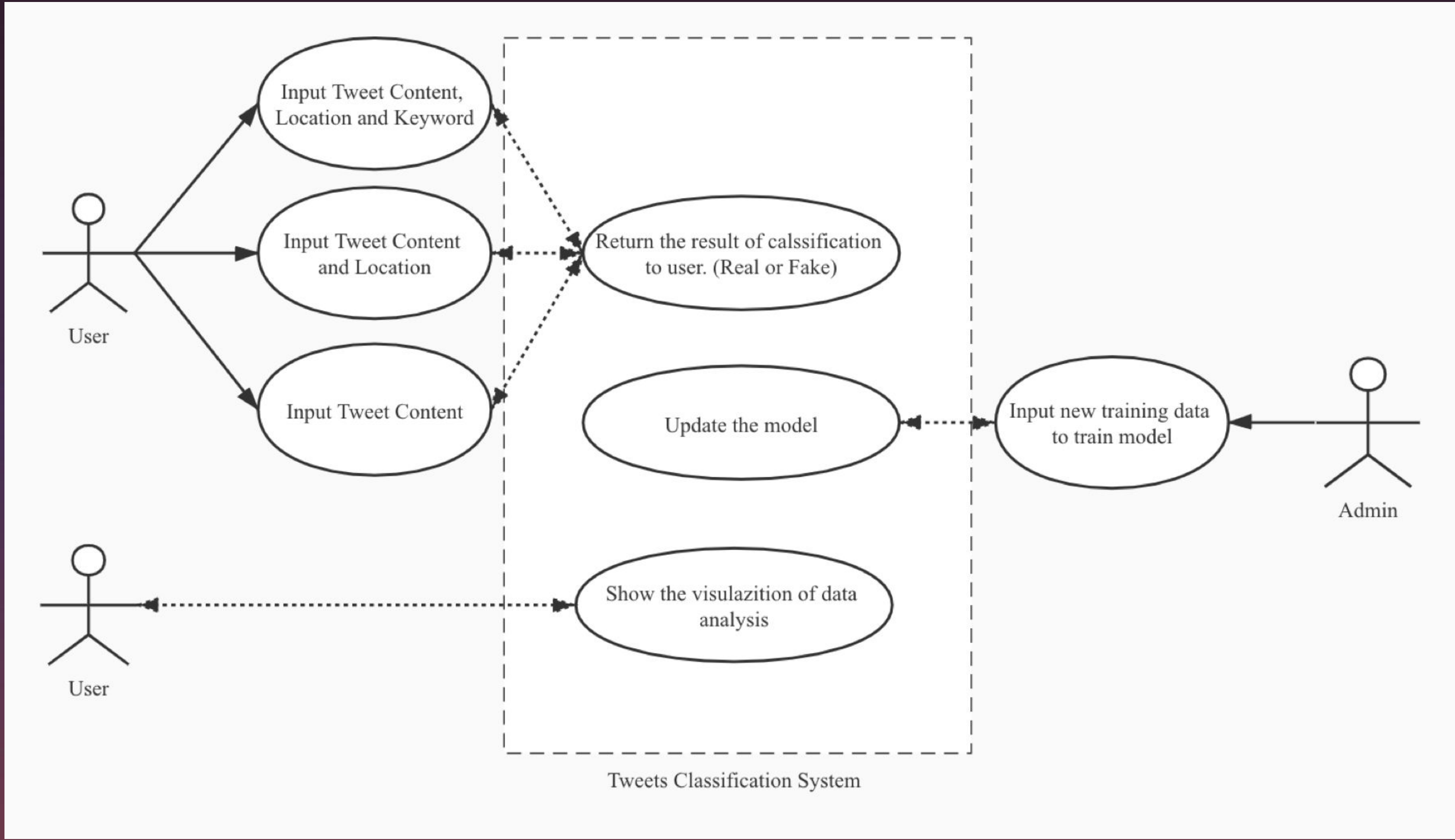
07 Acceptance Criteria Review

01 Background

Twitter has become an important communication channel in times of emergency.
The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programatically monitoring Twitter (i.e. disaster relief organizations and news agencies).
But, it's not always clear whether a person's words are actually announcing a disaster.

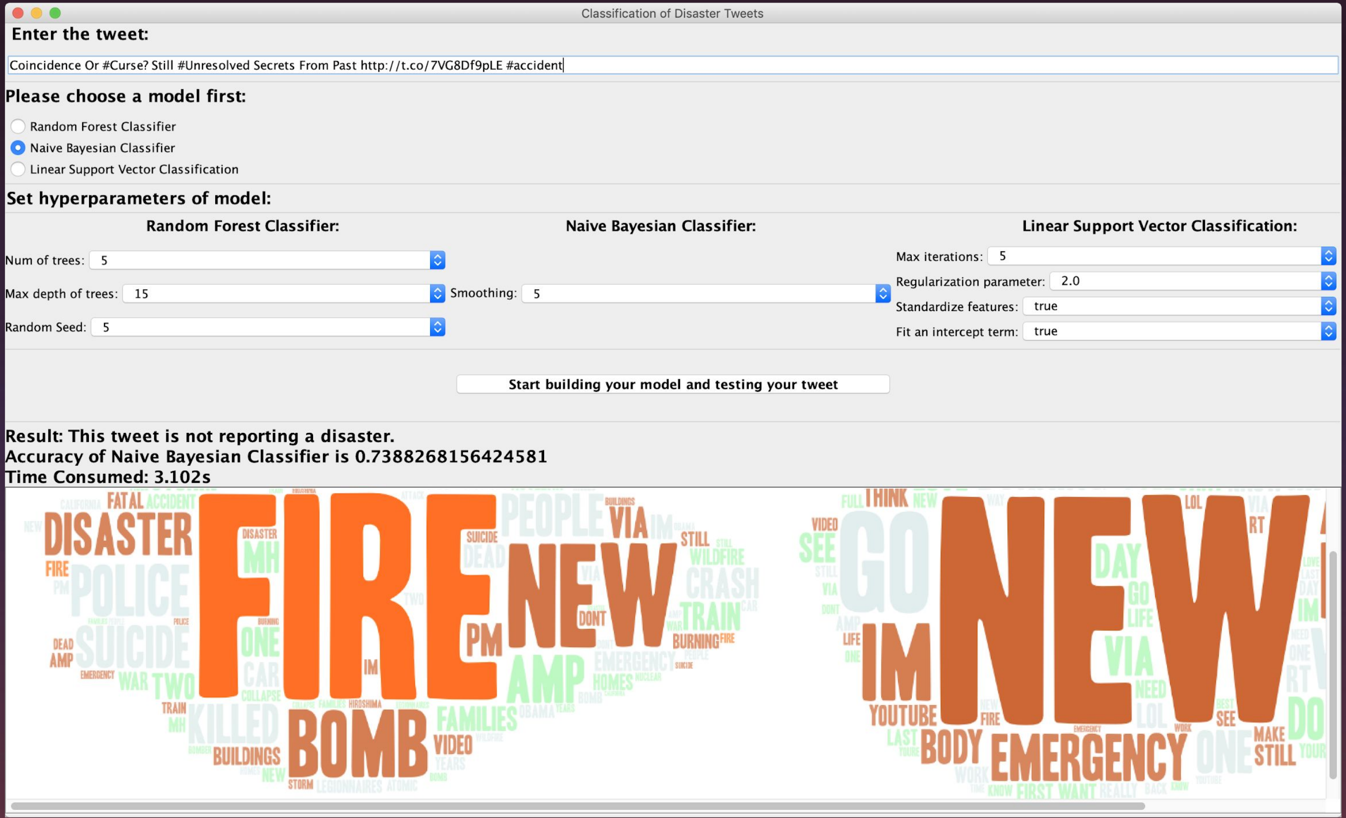


01 Uses Cases Review



01

UI Design

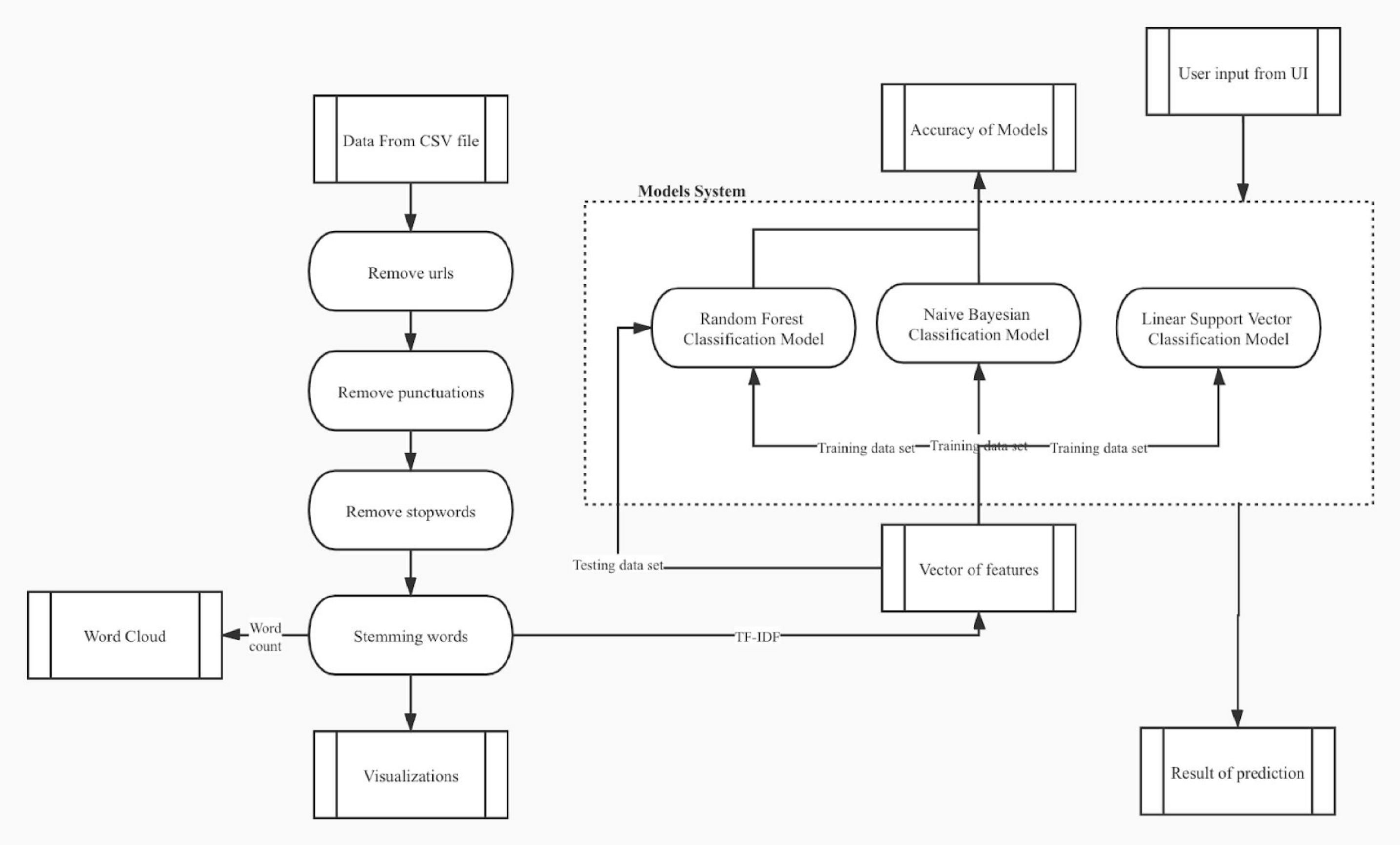

Classification of Disaster Tweets
Enter the tweet:
Coincidence Or #Curse? Still #Unresolved Secrets From Past http://t.co/7VG8Df9pLE #accident
Please choose a model first:
Random Forest Classifier
Naive Bayesian Classifier
Linear Support Vector Classification
Set hyperparameters of model:
Random Forest Classifier:
Num of trees: 5
Max depth of trees: 15
Random Seed: 5
Naive Bayesian Classifier:
Smoothing: 5
Linear Support Vector Classification:
Max iterations: 5
Regularization parameter: 2.0
Standardize features: true
Fit an intercept term: true
Start building your model and testing your tweet
Result: This tweet is not reporting a disaster.
Accuracy of Naive Bayesian Classifier is 0.7388268156424581
Time Consumed: 3.102s

03

Goals

1. Create a reactive page to detect fake news on twitter.
2. Create a reactive page to analyze the characteristics of fake tweets.
3. Get a well trained model for fake tweets prediction.

02 Methodology

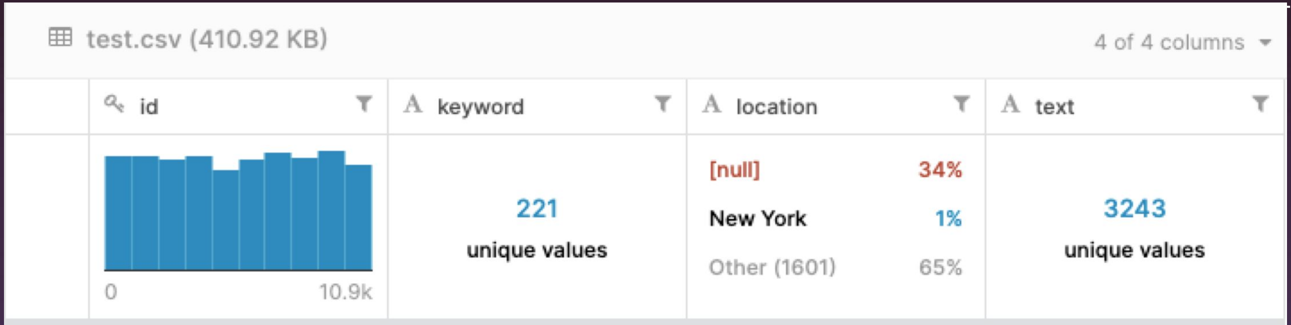


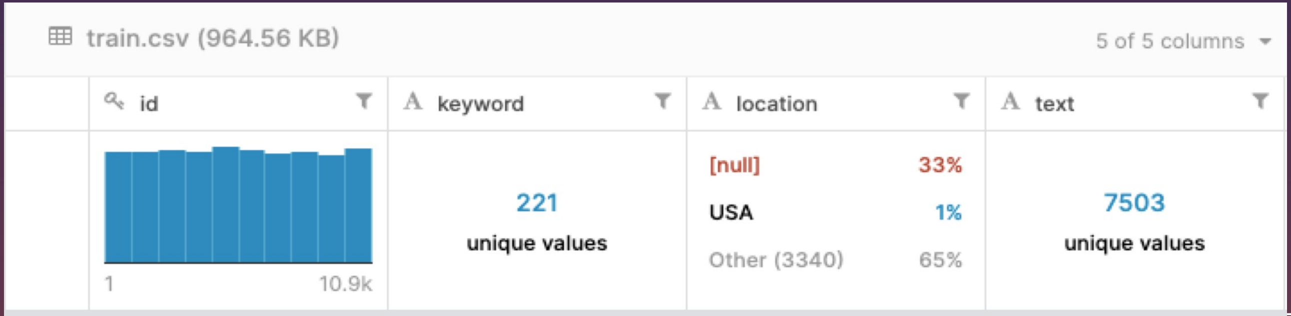
03

Methodology

- Used Zeppelin for running code line by line
- Extracted features of tweets (nature language) by TF-IDF
- Implemented 3 classification algorithms:
    - Random Forest Classifier
    - Naive Bayesian Classifier
    - Linear Support Vector Classifier
- Applied functions from Spark MLlib, Spark SQL, Spark RDD
- Designed and implemented UI by Scala Swing
- Visualized data by Vegas

01 Data Sources





Data come form Kaggle competition

Data magnitude is more than 10,000 rows

03

TF-IDF

Used to extract features and find keywords

- TF(t,d) is the number of times that term t appears in document d
- DF(t,D) is the number of documents that contains term t
- |D| is the total number of documents in the corpus

$$IDF(t, D) = \log \frac{|D| + 1}{DF(t, D) + 1},$$

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D).$$

03

NLP

- Use Tokenizer to tokenize the tweets
- Use StopWordsRemover to remove the stop words

  Such as “ I, am, what, have, is, are….. ”
- Use HashingTF and IDF to vectorize the words
- Then we got featured data from nature language

03

Classification Algorithm

In order to classify the tweets according to if it is a real disaster tweets, we implemented three different classification algorithms:

- Random Forest Classifier
- Naive Bayesian Classifier
- Linear Support Vector Classifier

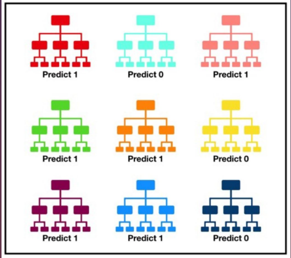
03

Random Forest Classifier

- A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.
- Given a training set X = x1, ..., xn with responses Y = y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples
- For b = 1, ..., B:
  - Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
  - Train a classification or regression tree fb on Xb, Yb.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B} (f_b(x') - \hat{f})^2}{B-1}}.$$



03

Naive Bayesian Classifier

- Another assumption made here is that all the predictors have an equal effect on the outcome.
- X represents the features , y represents the label

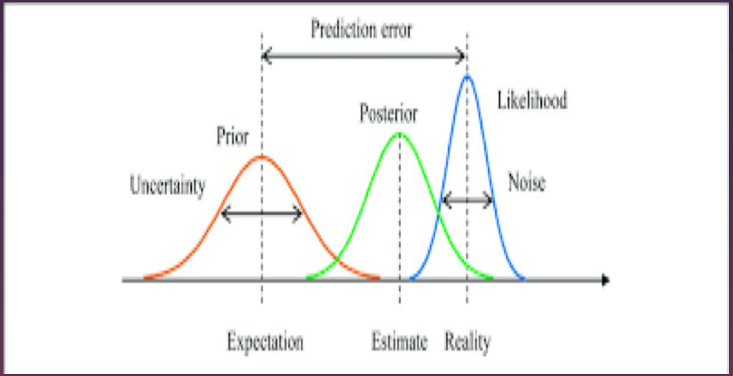
$$X = (x_1, x_2, x_3, ..., x_n)$$

- According to Bayesian Theory

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

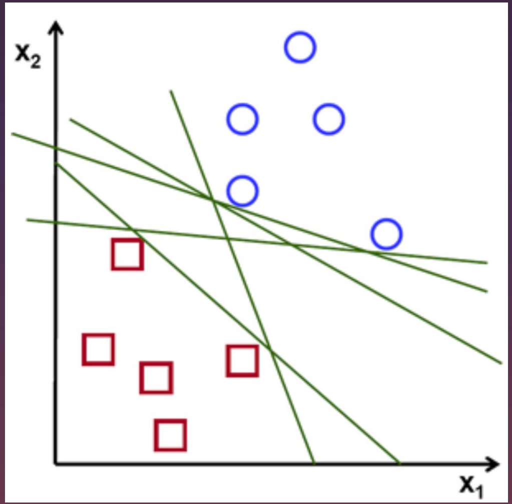
$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

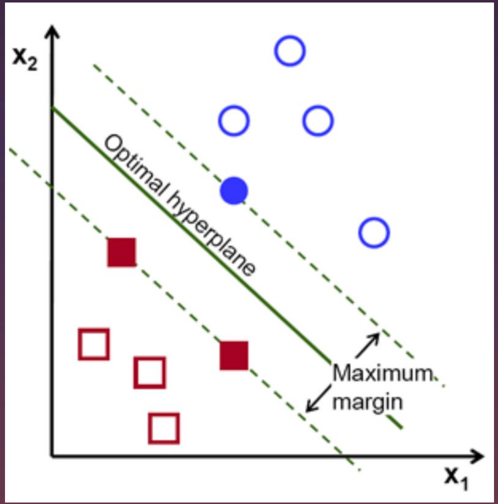
$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

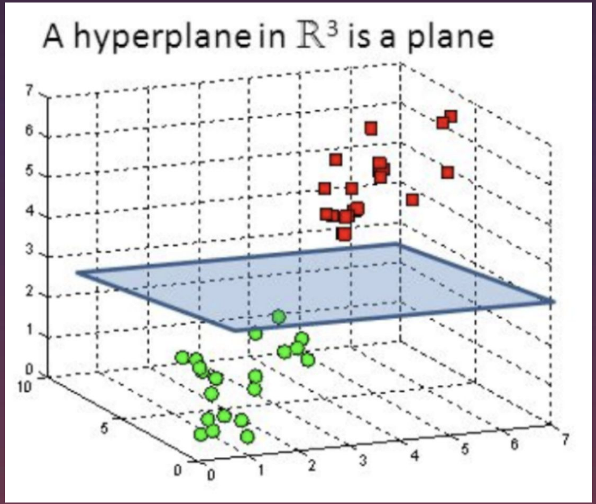


03 Linear Support Vector Classifier

- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.







03

Linear Support Vector Classifier

- In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

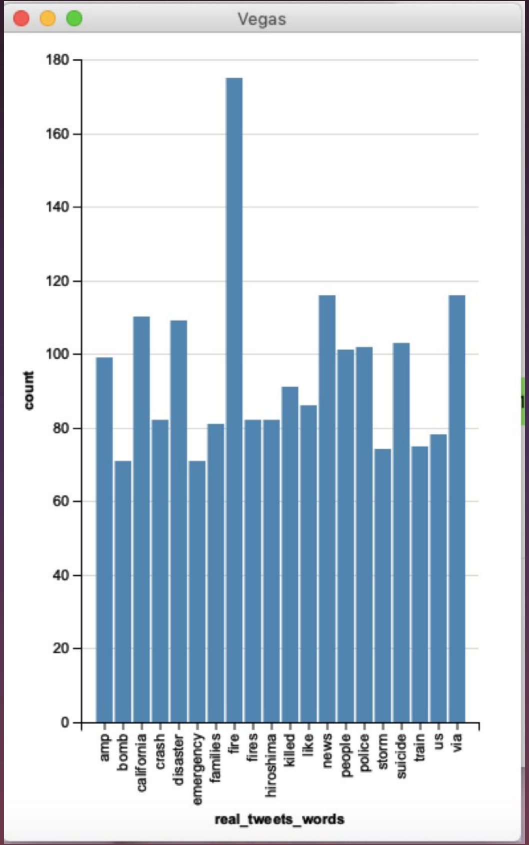
- Take partial derivatives with respect to the weights to find the gradients
- Update gradient (no misclassification)
- Update gradient (with misclassification)

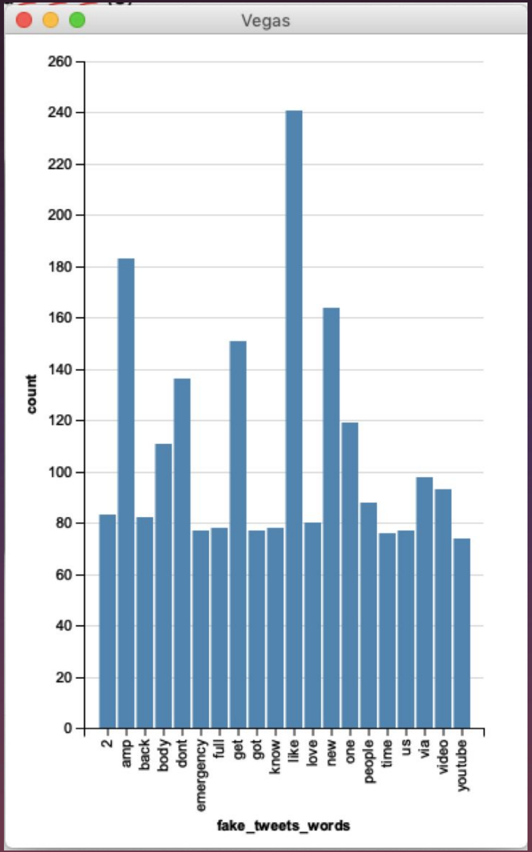
$$\frac{\delta}{\delta w_k} \big(1 - y_i \langle x_i, w \rangle\big)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

$$w = w - \alpha \cdot (2\lambda w)$$

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

03 Visualization of Keyword Extraction





03

Real Disaster Tweets

03

Fake Disaster Tweets

03

Evaluation of models

Model	Accuracy
Random Forest Classifier	0.7456170505328291
Naive Bayesian Classifier	0.8380480905233381
Linear Support Vector Classifier	0.8553220806062694

03

Acceptance Criteria Review

As a user, I am able to input Disaster Tweet content, location and keyword to get the prediction if the tweet is fake:

- The prediction accuracy for complete input data should be over 70%
- The time to respond should be under 5 seconds

As a user, I am able to show the visualization of data analysis:

- The time to respond should be under 5 seconds

Thank You