Halil AKSU

MS: University of Southern California, Operations Research

Twitter Sentiment Analysis

July 2nd, 2018

- Personel Background:
- 16 years of experience on management and analysis
- Additional background on international relations and public affairs
- Valuable experience on logistics and personnel management
- BS degree on Industrial Engineering (graduated 3rd out of 242)(3.84)
- MS degree on Operations Research (3.79)
- MA degree on Leadership and Management (graduated 3rd place)
- Numerous presentations before various VIP audience

Twitter Sentiment Analysis

Data Science Career Track Capstone Project, February 05th 2018 Cohort

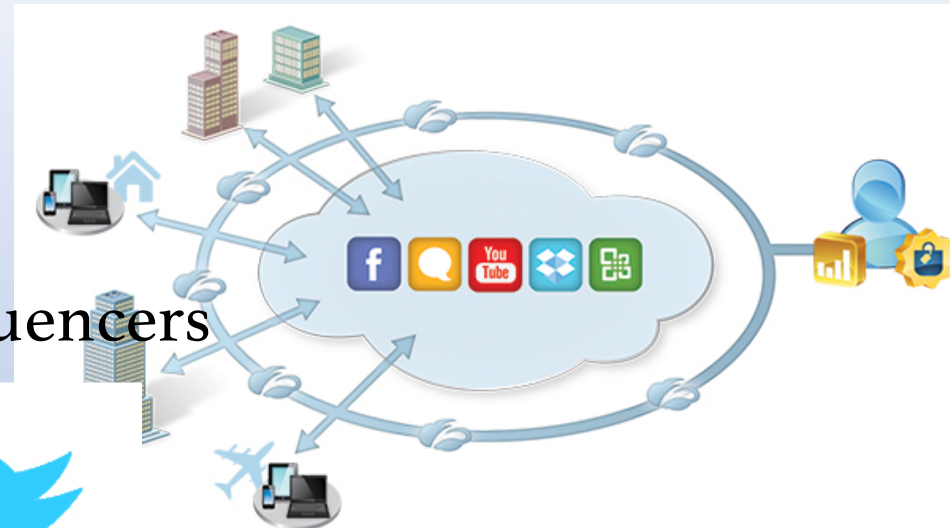
Springboard

Problem Introduction
Who might care?

Business/Organizations

Social Media Influencers

Corporate Network



Data Set :

- From Data World
- 4 features and 60000 data points/ observations (can be added more)
- Target Feature is 'sentiment'

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69

model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	dateCreated	nrOfPictures	postalCode	lastSeen
golf	150000	0	benzin	volkswagen	NaN	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
NaN	125000	5	diesel	audi	ja	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
grand	125000	8	diesel	jeep	NaN	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:46
golf	150000	6	benzin	volkswagen	nein	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
fabia	90000	7	diesel	skoda	nein	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

Data Pre-Processing:

Removing special characters:

Special characters and symbols which are usually non alphanumeric characters often add to the extra noise in unstructured text. More than often, simple regular expressions (regexes) can be used to achieve this.

Data Cleaning

▷Special character

Unicode emotions	☺, ♥…
Symbol icon	☏, ✉…
Currency symbol	€, £, $...

▷Utilize regular expressions to clean data

Tweet URL

(^|\\s*)http(\\S+)?(\\s*|$)

ube playlist
mie Riepe

Filter out non-(letters, space, punctuation, digit)

(\\p{L}+)|(\\p{Z}+)|
(\\p{Punct}+)|(\\p{Digit}+)

ng ♥ ✉

Data Pre-Processing:

Stemming and lemmatization:
Word stems are usually the base form of possible words that can be created by attaching affixes like prefixes and suffixes to the stem to create new words. This is known as inflection. The reverse process of obtaining the base form of a word is known as stemming. A simple example are the words WATCHES, WATCHING, and WATCHED. They have the word root stem WATCH as the base form. Lemmatization is very similar to stemming, where we remove word affixes to get to the base form of a word. However the base form in this case is known as the root word but not the root stem. The difference being that the root word is always a lexicographically correct word (present in the dictionary) but the root stem may not be so.

Stemming vs. Lemmatization

- **Stemming:**
  - Set of rules for removing characters from words
  - Increased recall at the expense of precision
  - Example EN rule: Remove trailing "ing" or "al"
- **Lemmatization:**
  - Complex set of approaches for producing the dictionary form of a word
  - Increased recall without hurting precision
  - Uses context to disambiguate candidates

Data Pre-Processing:

Stemming vs. Lemmatization

- English: "I have spoken at several conferences"
- Stemming:

org.apache.solr.analysis.EnglishPorterFilterFactory {protected=p

term position	1	2	3	4	5	6
term text	i	have	spoken	at	sever	conferences.
term type	word	word	word	word	word	word
source start,end	0,1	2,6	7,13	14,16	17,24	25,37
payload						

- Lemmatization:

com.basistech.rlp.solr.RLPTokenizerFactory

term position	1	2	3	5	6
term text	i	have	spoken	several	conferences
			speak		conference
term type	word	word	word	word	word
			lemma		lemma
source start,end	0,1	2,6	7,13	17,24	25,36
			7,13		25,36
payload					

Data Pre-Processing:

Removing accented characters:

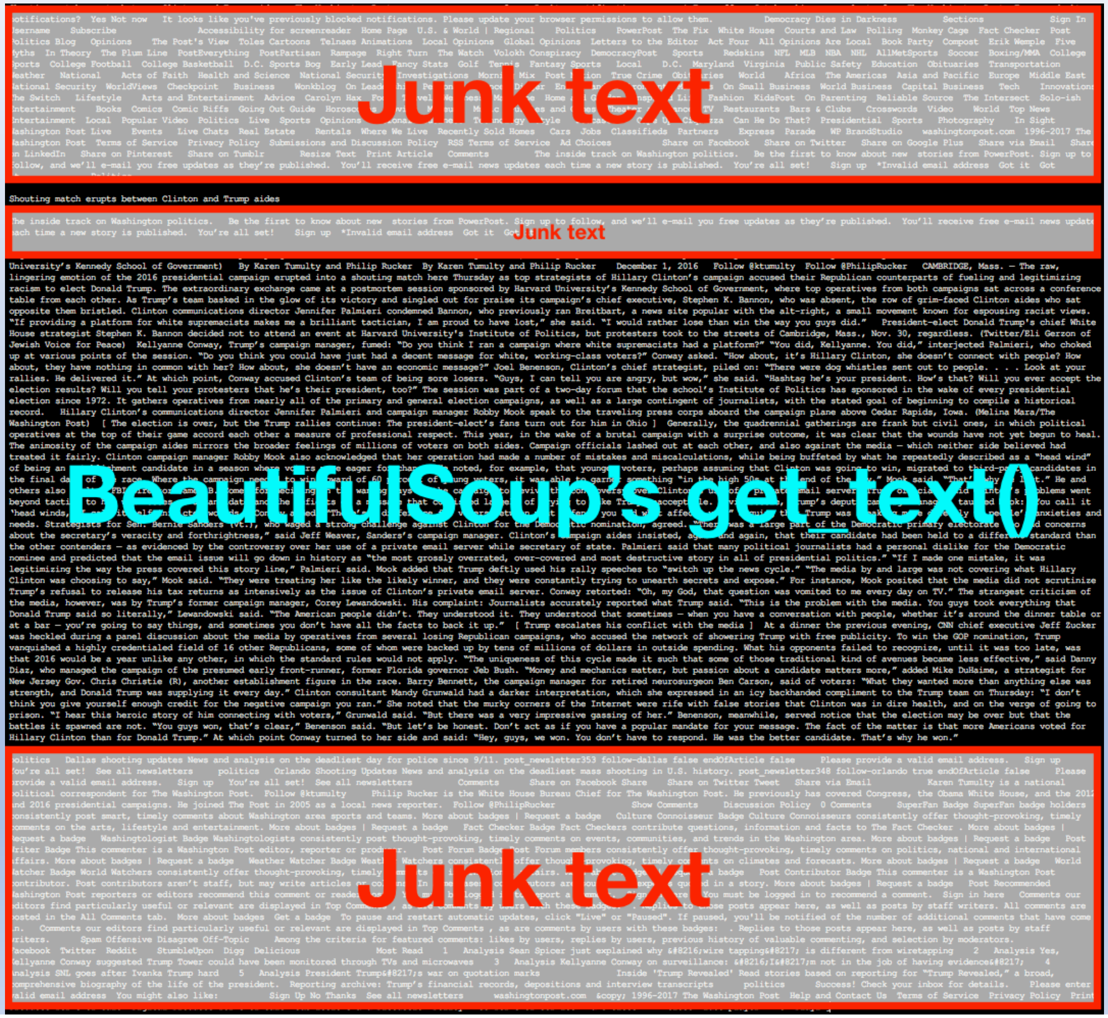
In any text corpus, especially if you are dealing with the English language, often you might be dealing with accented characters\letters. Hence we need to make sure that these characters are converted and standardized into ASCII characters. A simple example would be converting é to e.

```
Tous les éléments
présents
sur le site et le site en
lui-même sont protégés
par
le droit d'auteur et sont
la propriété du studio
Atelier Manufacture.
Toute
reproduction ou
publication
est interdite sans
l'autorisation du studio
```

Data Pre-Processing:

Removing HTML Tags:

Our text often contains unnecessary content like HTML tags, which do not add much value when analyzing text. The BeautifulSoup library does an excellent job in providing necessary functions for this.



Data Pre-Processing:

Removing punctuation

(removing extra white spacing)

Python 3.5.2 Shell

File Edit Shell Debug Options Window Help

```
>>> # define punctuation
>>> punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
>>> # take input from the user
>>> my_str = input("Enter a string: ")
Enter a string: hello..... This is a encripted (@#$@$,,.$) keyword.
>>> # remove punctuation from the string
>>> no_punct = ""
>>> for char in my_str:
        if char not in punctuations:
                no_punct = no_punct + char


>>> # display the unpunctuated string
>>> print(no_punct)
hello This is a encripted  keyword
>>> 
```

Ln: 35 Col: 4

Data Pre-Processing:

Text Lower Casing

```
# Initialize new list
words = []


# Loop through list tokens and make lower case
for word in tokens:
    words.append(word.lower())


# Print several items from list as sanity check
words[:8]
```

```
['the', 'project', 'gutenberg', 'ebook', 'of', 'the', 'adventu
res', 'of']
```

Data Pre-Processing:

Removing stopwords

Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

Bag of Words

Bag of Words is the vector space representational model for unstructured text. It is simply a mathematical model to represent unstructured text (or any other data) as numeric vectors, such that each dimension of the vector is a specific feature/attribute. The bag of words model represents each text document as a numeric vector where each dimension is a specific word from the corpus and the value could be its frequency in the document, occurrence (denoted by 1 or 0) or even weighted values.

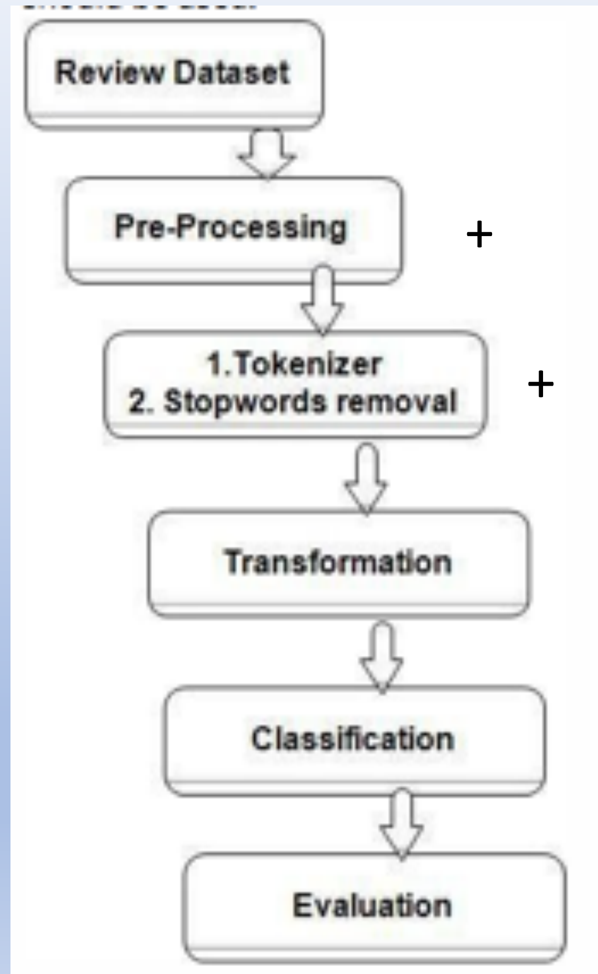
zikihekai	zillaman	zillygrl	zimbabwe	zimmer	zimmermann	zinc	zincroof	zindelayentl	zinedistro	zing
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

TF-IDF Vectorizer

tfidf(w, D) is the TF-IDF score for word w in document D. The term tf(w, D) represents the term frequency of the word w in document D, which can be obtained from the Bag of Words model. The term idf(w, D) is the inverse document frequency for the term w, which can be computed as the log transform of the total number of documents in the corpus C divided by the document frequency of the word w, which is basically the frequency of documents in the corpus where the word w occurs.

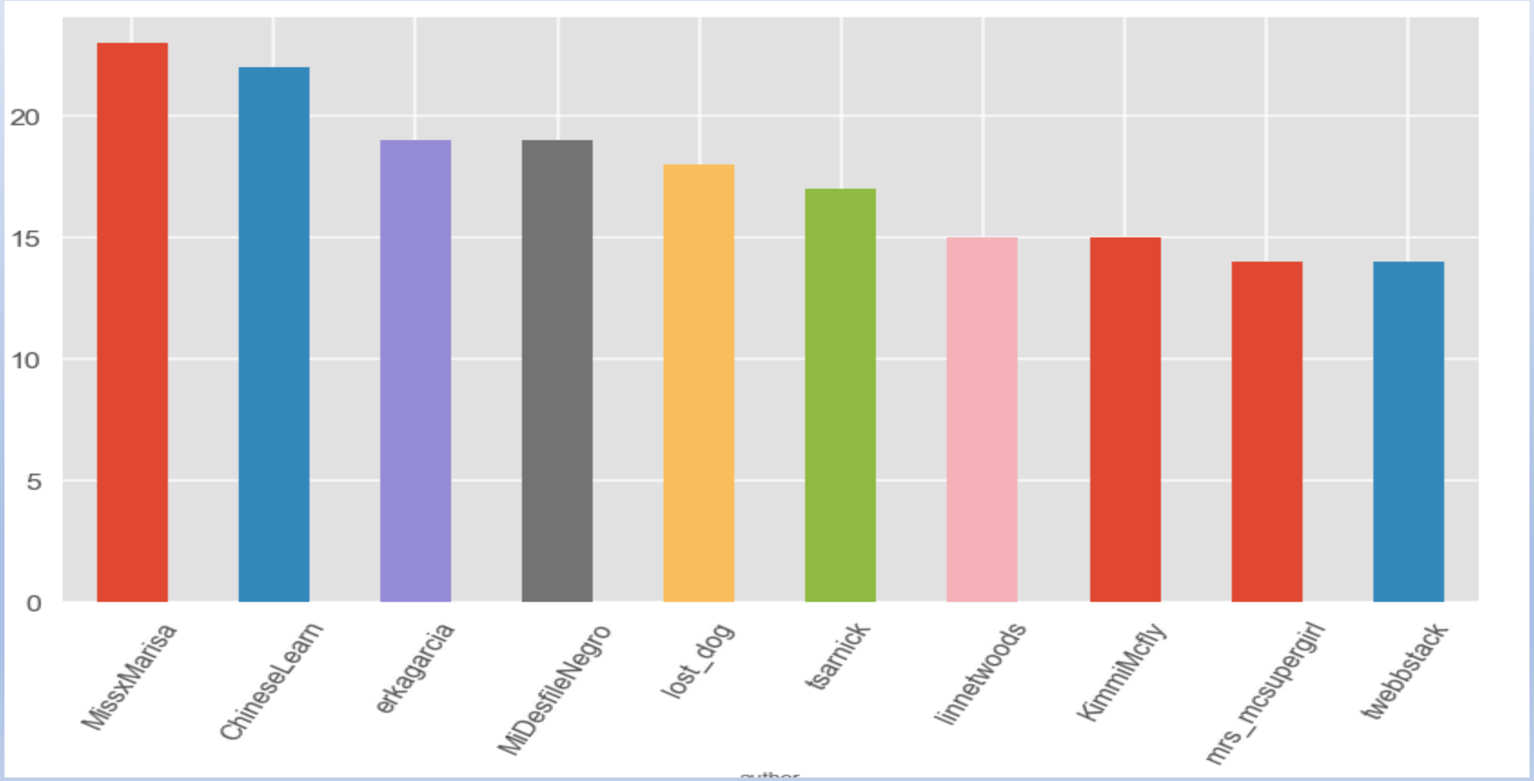
grl	absolves	absoulutely	absoulutley	absoute	abstractg	abstraction	absurd	abt	abueltia	abundantly	abuse	abusing	abusive	abuzz	aby	abzquine	ac
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

NLP Cycle:


Review Dataset
Pre-Processing
+
1.Tokenizer
2. Stopwords removal
+
Transformation
Classification
Evaluation

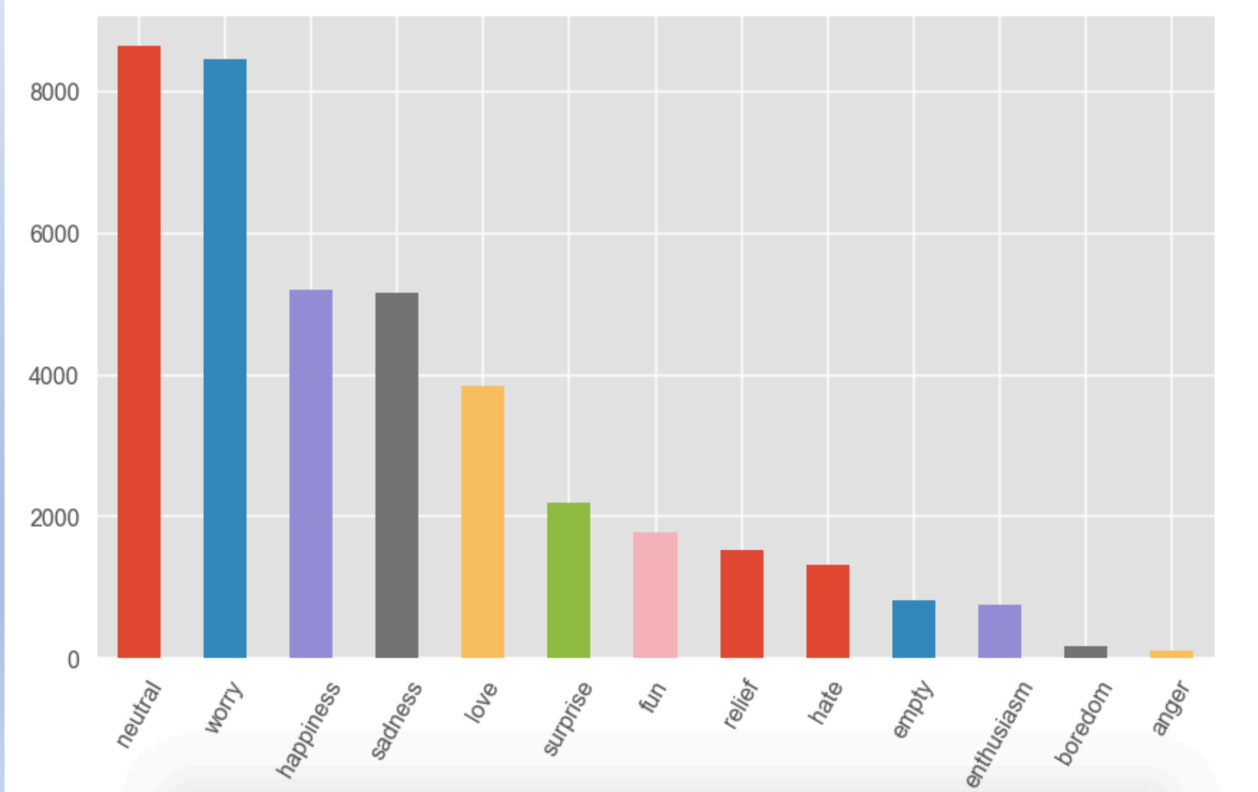
Data Visualization:

Here is the number of tweets who tweeted most in order. The highest number of tweets is 27. This doesn't do serious impact on the data.



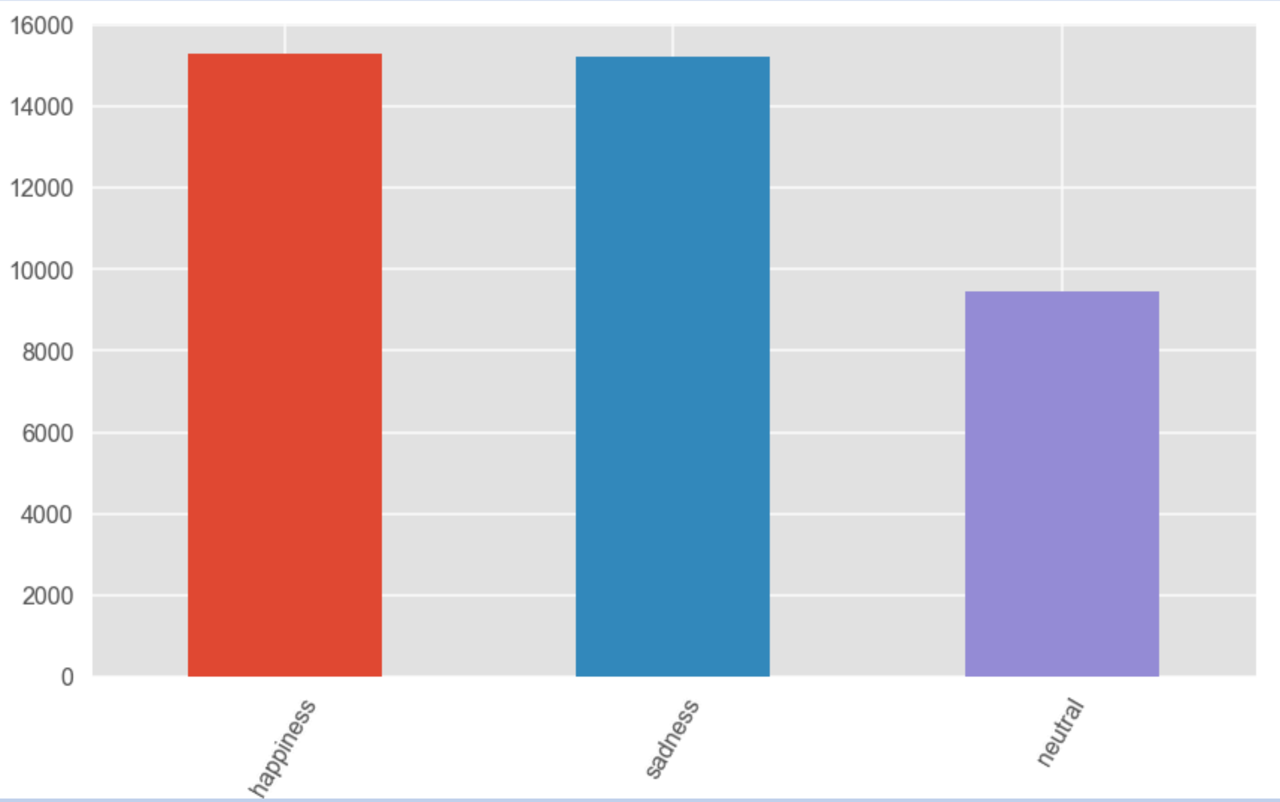
Data Visualization:

This is the number of tweets on specific feelings. There are 13 labels at first.



Data Visualization:

Because it is hard to get a score above 45%, then I shrank the no of labels to 3 as happiness, sadness and neutral.



Data Visualization:

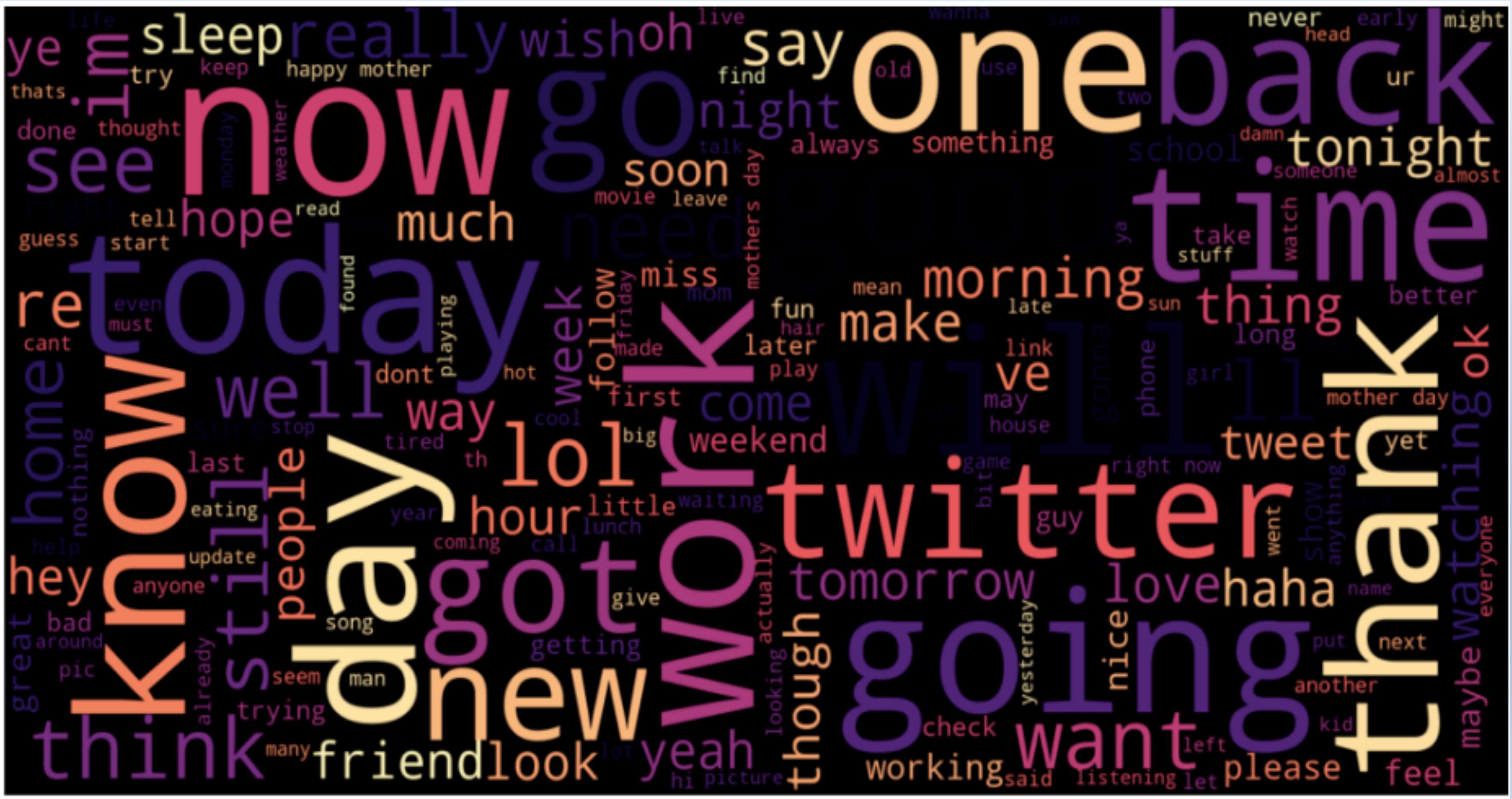
Word Cloud : Happy

Data Visualization:

Word Cloud : Sad

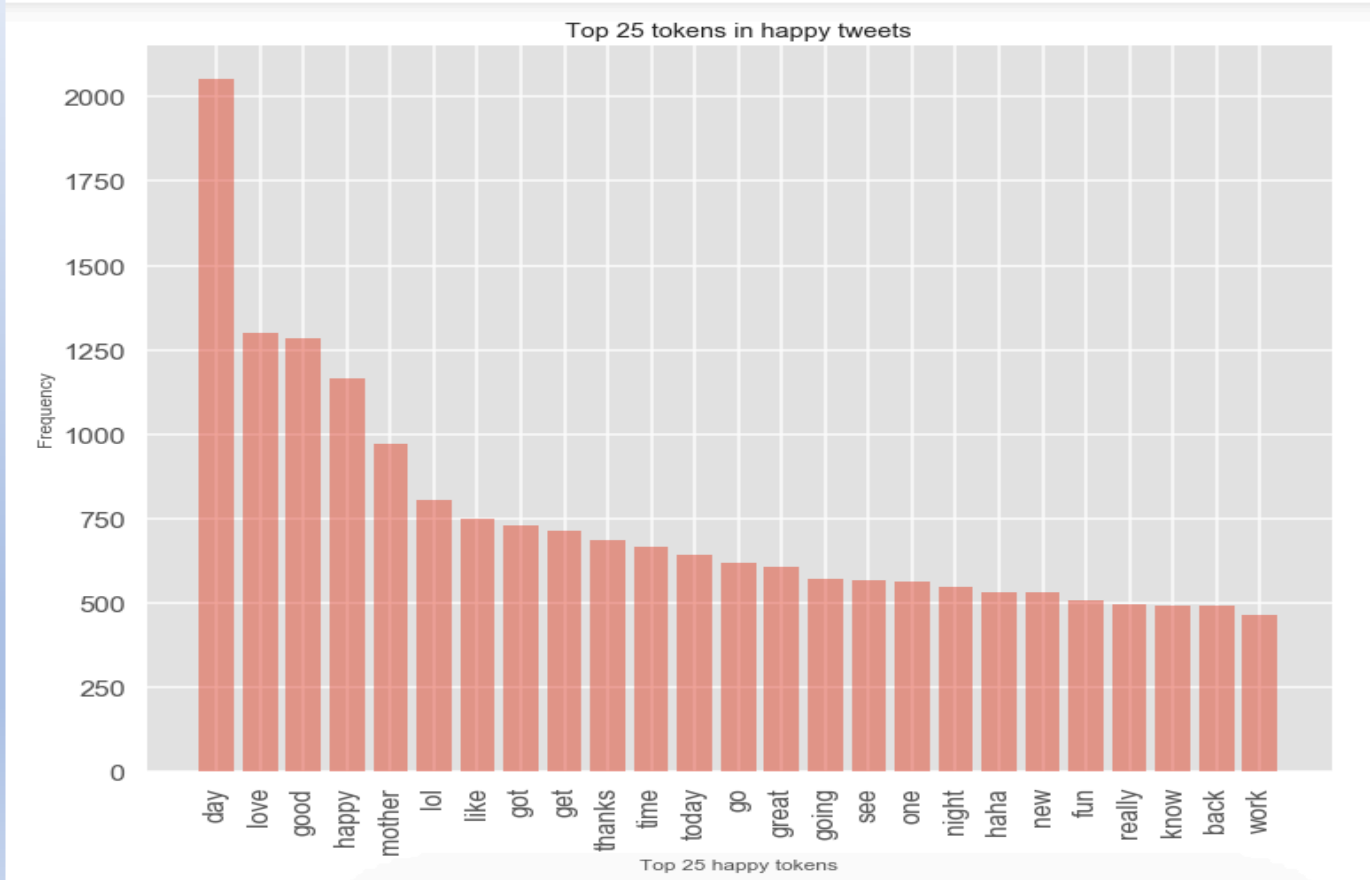
Data Visualization:

Word Cloud : Neutral



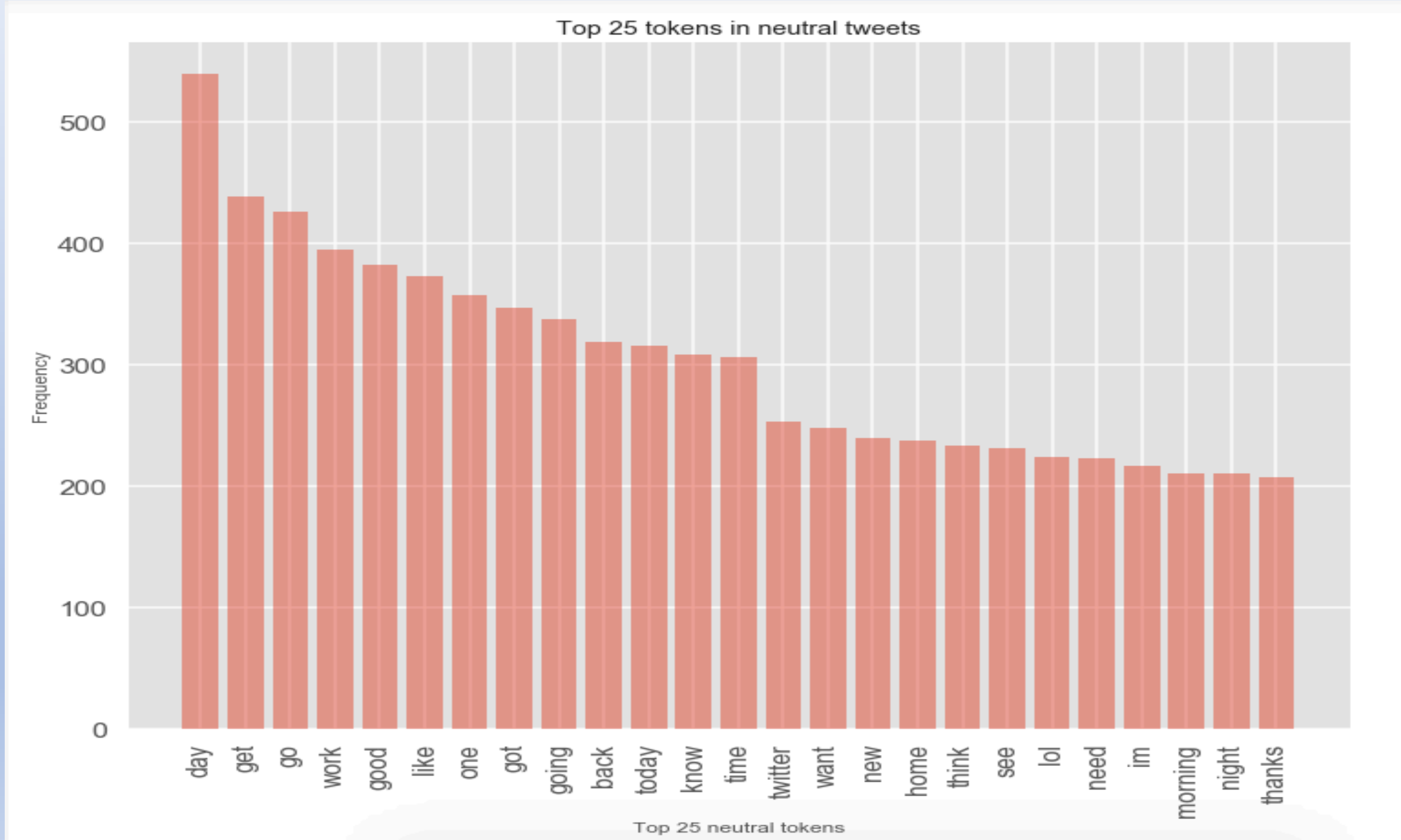
Data Visualization:

These are the top 25 tokens in happy tweets:



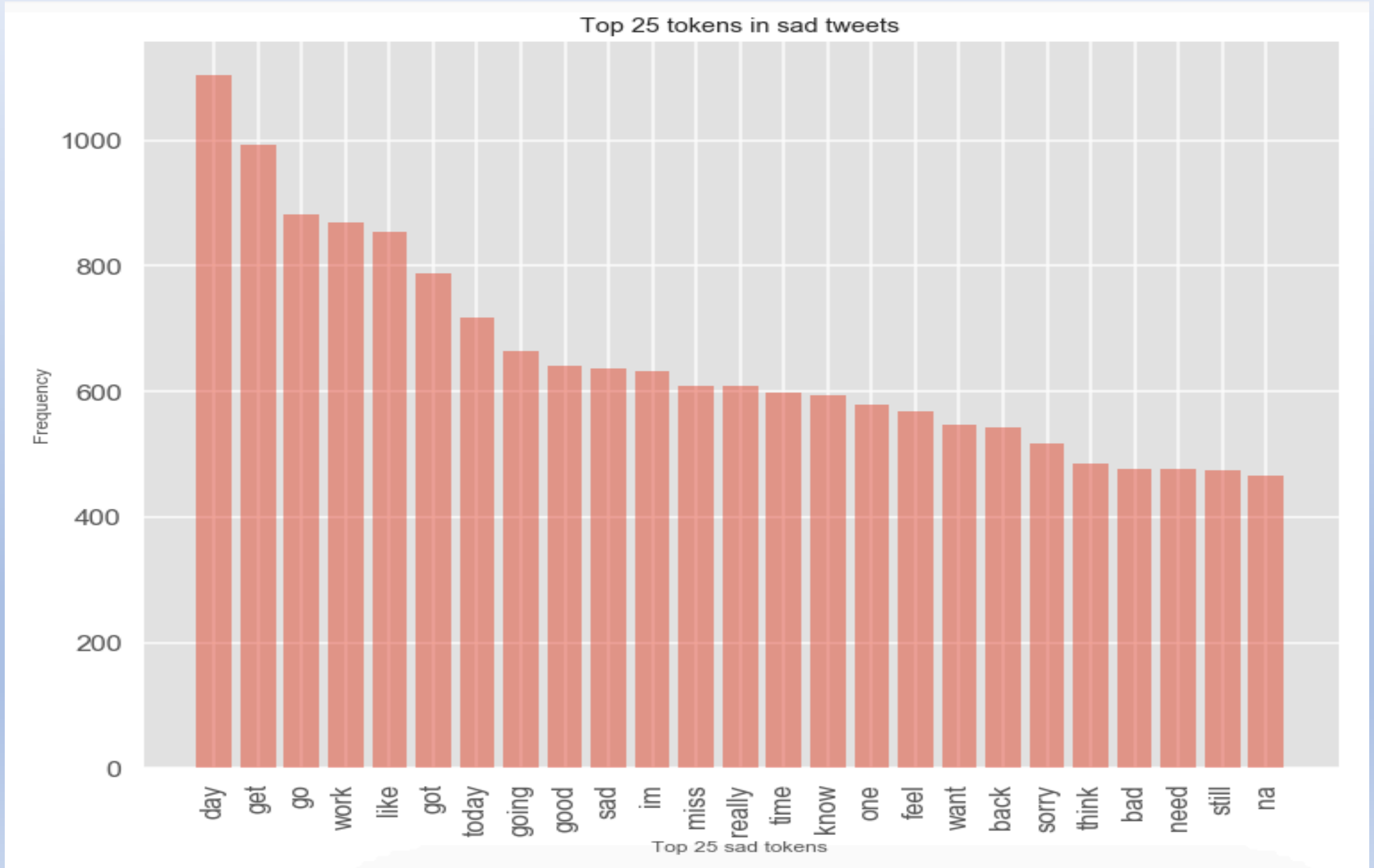
Data Visualization:

These are the top 25 tokens in sad tweets:



Data Visualization:

These are the top 25 tokens in neutral tweets:



Predictive Modeling

Modeling Overview

Type: Supervised learning

Multi Label Classification

Tools: Python's scikit learn(main), Deep Learning Model with Keras

Model Assumptions, Limitations and Disclaimers

- Assume that all tweets are independent
- Used the data only from 2016 (Future Work: Should be expanded)
- Utilized CountVectorizer, TF-IDF Vectorizer and Hash-Vectorizer
- Tried topic modelling, but there is much to do for future work.

Classification:

Logistic Regression

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

logreg = LogisticRegression(random_state=0)

logreg.fit(cv_train, y_train)

pred = logreg.predict(cv_test)

metrics.accuracy_score(pred, y_test)
```

```
0.6065333333333334
```

Classification:

Linear SVM

Linear SVC

```
from sklearn.svm import LinearSVC

Lsvc = LinearSVC()

Lsvc.fit(cv_train, y_train)

pred= Lsvc.predict(cv_test)

metrics.accuracy_score(y_test, pred)
```

```
0.5749333333333333
```

Classification:

Naïve-Bayes

Naive Bayes

```
]: from sklearn.naive_bayes import MultinomialNB

   nb_classifier = MultinomialNB()

   nb_classifier.fit(cv_train, y_train)

   pred = nb_classifier.predict(cv_test)

   metrics.accuracy_score(y_test, pred)

]: 0.6098
```

Classification:

With Tf-IDF Vectorizer, different models

```
Validation result for Logistic Regression features
accuracy score: 62.22%
Validation result for Linear SVC features
accuracy score: 59.30%
Validation result for LinearSVC with L1-based feature selection features
accuracy score: 59.80%
Validation result for Multinomial NB features
accuracy score: 61.57%
Validation result for Bernoulli NB features
accuracy score: 61.41%
Validation result for Ridge Classifier features
accuracy score: 60.56%
Validation result for AdaBoost features
accuracy score: 55.87%
Validation result for Perceptron features
accuracy score: 54.71%
Validation result for Passive-Aggresive features
accuracy score: 56.85%
Validation result for Nearest Centroid features
accuracy score: 53.91%
```

Classification:

Deep Learning Algorithm (Basic)

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Embedding, LSTM

model = Sequential()
model.add(Embedding(2500,128,input_length=X.shape[1],dropout=0.2))
model.add(LSTM(300, dropout_U=0.2,dropout_W=0.2))
model.add(Dense(3,activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer='adam',metrics=['accuracy'])
```

```
model.fit(X_tr ,y_tr, epochs=5,verbose=2,batch_size=32)
```

```
Epoch 1/5
 - 118s - loss: 0.9404 - acc: 0.5573
Epoch 2/5
 - 119s - loss: 0.8653 - acc: 0.6136
Epoch 3/5
 - 118s - loss: 0.8350 - acc: 0.6314
Epoch 4/5
 - 119s - loss: 0.8003 - acc: 0.6504
Epoch 5/5
 - 119s - loss: 0.7635 - acc: 0.6690
```

Classification:

Topic Modelling with LDA

Topic Modeling

In [85]:

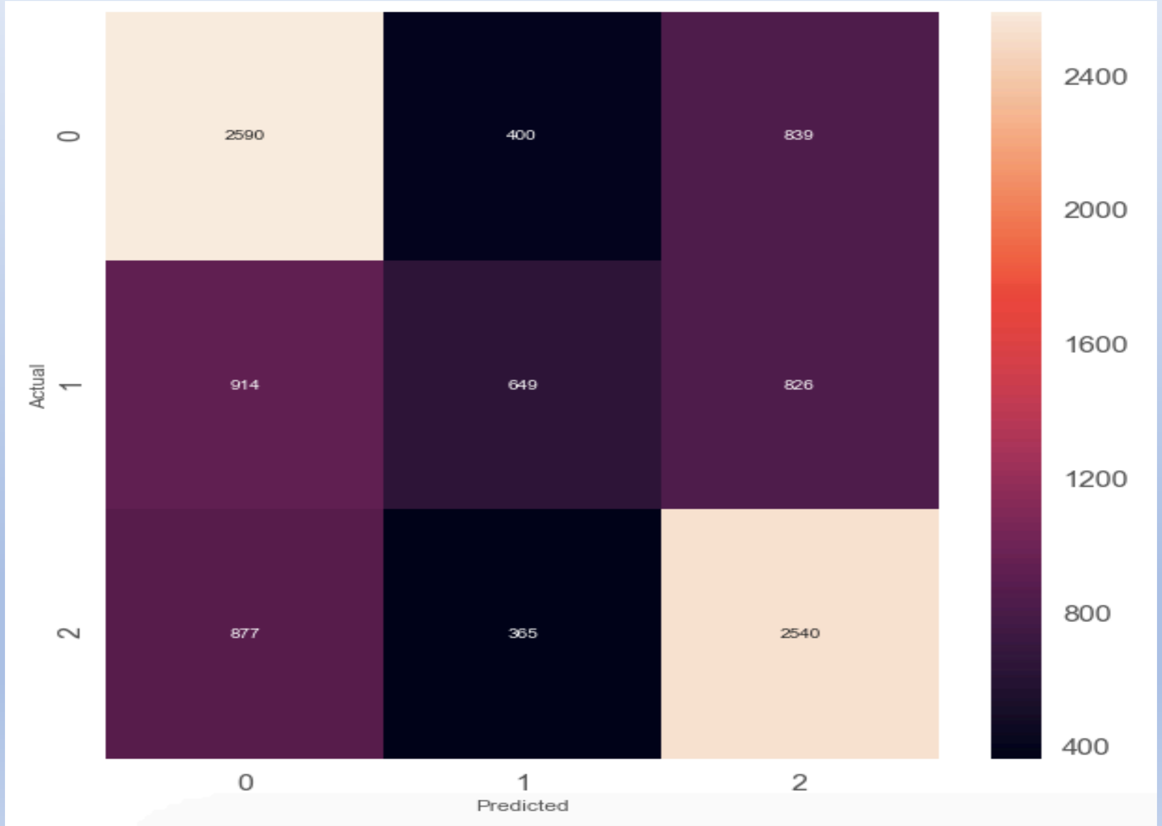
```
from sklearn.decomposition import LatentDirichletAllocation

lda = LatentDirichletAllocation(n_topics=3, max_iter=100, random_state=0)
dt_train = lda.fit_transform(cv_train)
dt_test = lda.transform(cv_test)

features = pd.DataFrame(dt_train, columns=['T1', 'T2', 'T3'])
features
```

Out[85]:

	T1	T2	T3
0	0.041684	0.243554	0.714762
1	0.925560	0.037359	0.037080
2	0.925418	0.037301	0.037282
3	0.026618	0.762926	0.210455
4	0.048124	0.049001	0.902875
5	0.830821	0.085682	0.083498
6	0.028106	0.943740	0.028154
7	0.033676	0.932924	0.033400
8	0.024049	0.101810	0.874142
9	0.066777	0.066699	0.866524
10	0.288042	0.651026	0.060932

Classification:

Confusion Matrix



More Ideas to Improve Model in Future

- Extract more data with Cloud Platform of more powerful Computer
- Use other Algorithms/Models to get better results (Word2Vec, Doc2Vec, Topic Modelling, Phrase Modelling etc).

Conclusions

- Tried Bag of Words and TFIDF with different models
- Out of various Models, DNN provided the best result: 63 %
- With more ideas, the model can improve in the future

Thank you!

Halil Aksu

Email: halilaksu79@gmail.com

https://www.linkedin.com/in/halil-aksu-38bab6129/

https://github.com/CoderModer79