Kagan ILTER

Sentiment Analysis with 55.000 Tweets

Springboard Data Science Career Track

Capstone Project-2

Problem Definition

The scope of this project is to predict the sentiment of a short twitter text and categorize them over 3 sentiments (positive, negative, neutral)

Companies/ Organizations	Social Media Influencers	Ecommerce traders	Websites / applications

Data Information

- Dataset is acquired from https://data.world/crowdflower/sentiment-analysis-in-text
- 4 features and 55.000 data points/samples .
- Contains labels for the emotional content (such as happiness, sadness, and anger)
- Target Feature is ‘sentiment’

Data Understanding and Preprocessing

	tweet_id	sentiment	author	content
0	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habit earlier and i started freakin at his part =[
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin on your call...
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
4	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone who has Houston tickets, but no one will.
5	1956968477	worry	xxxPEACHESxxx	Re-pinging @ghostridah14: why didn't you go to prom? BC my bf didn't like my friends
6	1956968487	sadness	ShansBee	I should be sleep, but im not! thinking about an old friend who I want. but he's married now. damn, & he wants me 2! scandalous!

Anger, boredom, hate, worry, sadness : Negative
Happiness, fun, love, surprise, enthusiasm, relief : Positive
Empty, neutral : Neutral

Data Understanding and Preprocessing

- Cleaning the tweet texts :

```
import string

def tweet_cleaner(tweet):

    # To lowercase
    tweet = tweet.lower()

    # Remove HTML special entities (e.g. &amp;)
    tweet = re.sub(r'\&\w*;', '', tweet)

    #Convert @username to "@user"
    tweet = re.sub('@[^\s]+','@user',tweet)

    # Remove whitespace (including new line characters)
    tweet = re.sub(r'\s\s+', ' ', tweet)

    # Remove single space remaining at the front of the tweet.
    tweet = tweet.lstrip(' ')

    # Remove characters beyond Basic Multilingual Plane (BMP) of Unicode:
    tweet = ''.join(c for c in tweet if c <= '\uFFFF')

    # Remove hyperlinks
    tweet = re.sub(r'https?:\/\/.*\/\w*', 'http', tweet)

    # Remove tickers such as USD ($)
    tweet = re.sub(r'\$\w*', '', tweet)

    # Remove hashtags
    tweet = re.sub(r'#\w*', '', tweet)
```

- Removing special characters,
- Punctuations,
- Accented characters,
- Html tags,
- Spaces,
- Tickers
- Hyperlinks
- Usernames
- Stopwords

Data Understanding and Preprocessing

- Stemming and lemmatization:

```
wordnet_lemmatizer=WordNetLemmatizer()

df.groupby('sentiment').content.apply(lambda x: ' '.join(x)).\
apply(lambda x: Counter([wordnet_lemmatizer.lemmatize(a) for a in
                            [k for k in [l for l in [t.lower() for t in word_tokenize(x)]
                                         if l.isalpha()] if k not in stop]]).most_common(5))
```

- Missing Values, outliers, duplicates:

  - No missing values
  - 173 repeating duplicates

Data Understanding and Preprocessing

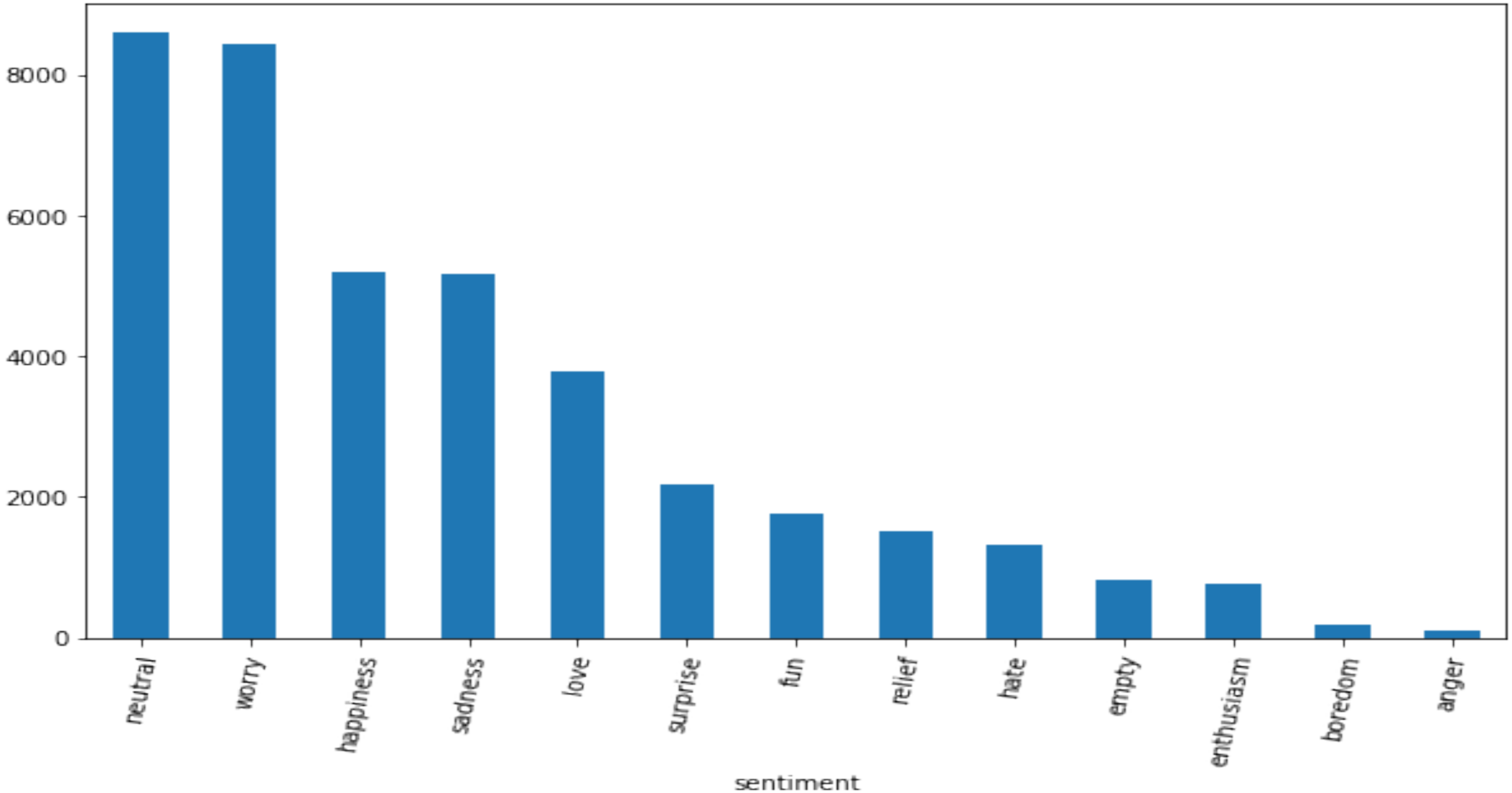
- <u>Bag of Words (Plus n-grams) (CountVectorizing in ScikitLearn):</u>

  - A mathematical model to represent unstructured text (or any other data) as numeric vectors

```
count_vect1 = CountVectorizer(min_df=0.001)
count_vect_train1 = count_vect1.fit_transform(X_train1)
count_vect_train1 = count_vect_train1.toarray()
count_vect_test1 = count_vect1.transform(X_test1)
count_vect_test1 = count_vect_test1.toarray()
```

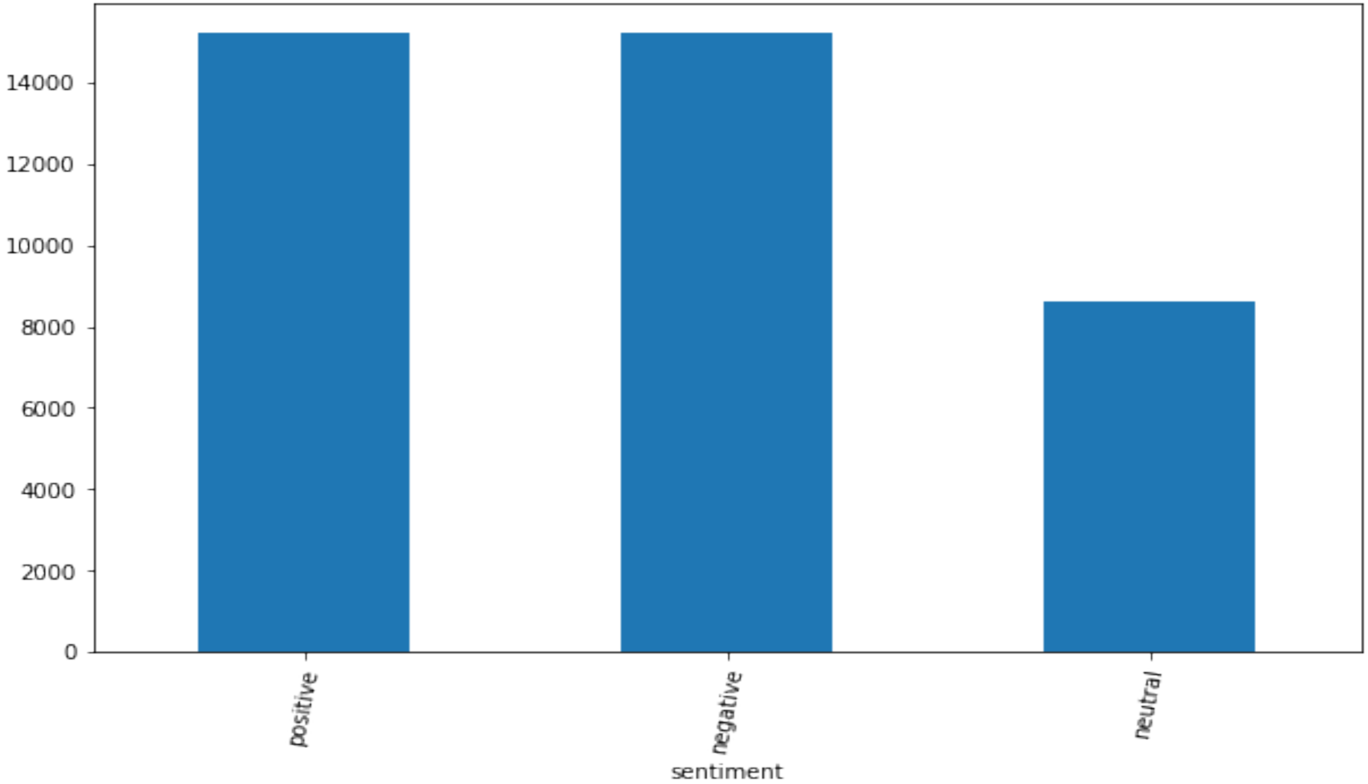
you	youll	your	youre	youtube	youve	yr	yum	yummy	ûï
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Data Visualization



The distribution of emotions in the data set (imbalanced)

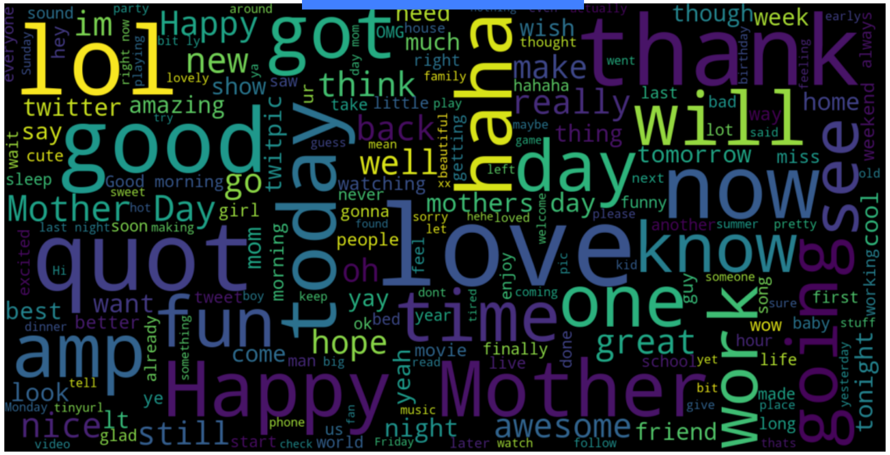
Data Visualization



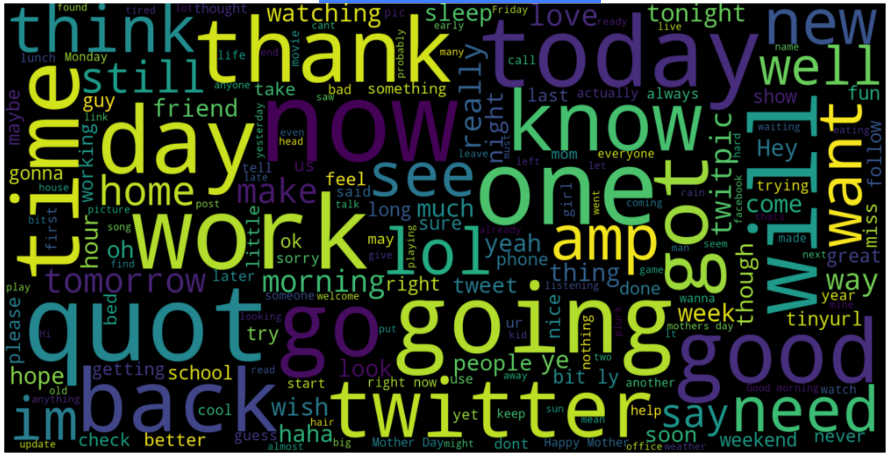
The distribution of 3 sentiments

Data Visualization

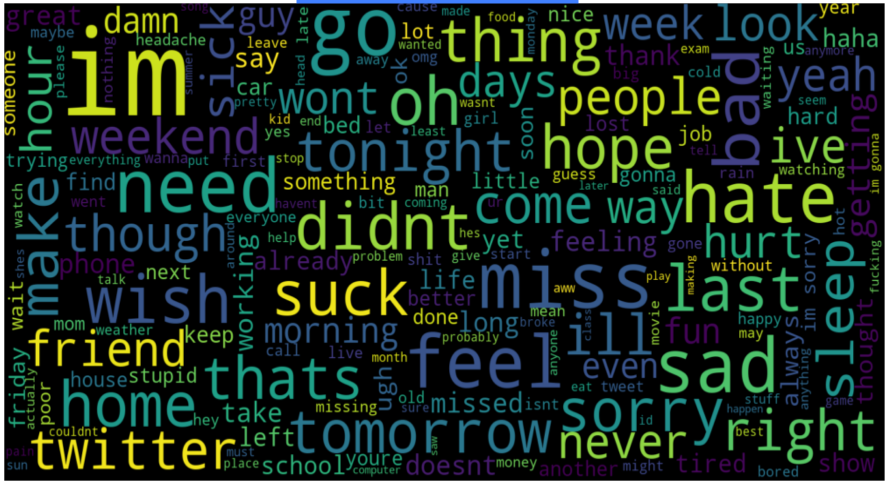
Positive



Neutral



Negative



Data Comparison with NLTK Vader

- NLTK Vader is a parsimonious rule-based model for sentiment analysis of social media text. With Vader, we can compare our dataset's classification with the Vader classification

```
sentiments = []
compounds = []
sid = SentimentIntensityAnalyzer()
for i in dfnew.content:
    sentiment = sid.polarity_scores(i)
    if sentiment['compound']< -0.05:
        sentiments.append('negative')
    elif sentiment['compound']> 0.05:
        sentiments.append('positive')
    else:
        sentiments.append('neutral')
    compounds.append(sentiment)

dfnew['vader'] = pd.DataFrame(sentiments)
dfnew['compound'] = compounds
```

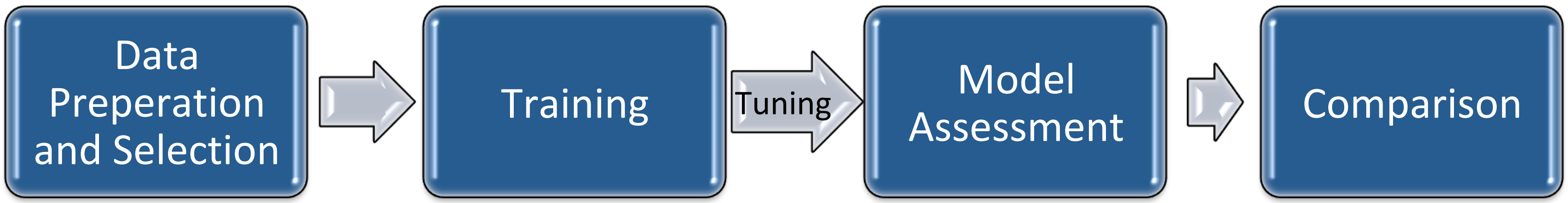
- If the Vader compound result is lower than -0.05, the text is categorized as negative sentiment. Higher than 0.05 is a positive sentiment.

- As a result of Vader comparison, we realized that our dataset's classification is better than Vader's classification.

Predictive Modeling

Machine Learning Models

- Supervised learning multi-class classification



- Subset of whole data: 30%
  - Logistic regression,
  - Naive bayes,
  - Linear svm,
  - Random forest,
  - Gradient boosting,
  - Xgboosting
  - Deep learning

Machine Learning Models

➢ Logistic Regression for 13 emotions

```
logreg_CV1 = LogisticRegression(multi_class='multinomial', solver='newton-cg',class_wei
                              C=1.0,n_jobs=-1, random_state=5)
logreg_CV1.fit(count_vect_train1, y_train1)
y_pred_lr_CV1 = logreg_CV1.predict(count_vect_test1)
print('Accuracy :', metrics.accuracy_score(y_test1, y_pred_lr_CV1))
```

```
Accuracy : 0.2208691193436739
```

➢ Logistic Regression for 3 sentiments

```
logreg_CV = LogisticRegression(random_state=0)
logreg_CV.fit(count_vect_train, y_train)
y_pred_lr_CV = logreg_CV.predict(count_vect_test)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred_lr_CV))
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarni
ng: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this
warning.
  FutureWarning)
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:469: FutureWarni
ng: Default multi_class will be changed to 'auto' in 0.22. Specify the multi_class optio
n to silence this warning.
  "this warning.", FutureWarning)
Accuracy : 0.6158856116185057
```

Machine Learning Models

- Linear SVC with Count Vectorizing

```
Lsvc = LinearSVC()

Lsvc.fit(count_vect_train, y_train)

pred= Lsvc.predict(count_vect_test)

metrics.accuracy_score(y_test, pred)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:929: ConvergenceWarning: Libl
inear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
0.6156610270998653
```

- Naïve Bayes with Count Vectorizing

```
from sklearn.naive_bayes import MultinomialNB

nb_classifier = MultinomialNB()

nb_classifier.fit(count_vect_train, y_train)

pred = nb_classifier.predict(count_vect_test)

metrics.accuracy_score(y_test, pred)
```

```
0.566701602036233
```

Machine Learning Models

- Random Forest with Count Vectorizing

```
from sklearn.ensemble import RandomForestClassifier
rf_CV = RandomForestClassifier(random_state=0, n_jobs=-1, class_weight="balanced", n_est
rf_CV.fit(count_vect_train, y_train)
y_pred_rf_CV = rf_CV.predict(count_vect_test)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred_rf_CV))
```

```
Accuracy : 0.5810001497230124
```

- Gradient Boosting with Count Vectorizing

```
xg_boost_CV = XGBClassifier()
xg_boost_CV.fit(count_vect_train, y_train)
y_pred_xg_boost = xg_boost_CV.predict(count_vect_test)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred_xg_boost))
```

```
Accuracy : 0.5488097020512053
```

Machine Learning Models

➢ Random Forest with Count Vectorizing

```
from sklearn.ensemble import RandomForestClassifier
rf_CV = RandomForestClassifier(random_state=0, n_jobs=-1, class_weight="balanced", n_est
rf_CV.fit(count_vect_train, y_train)
y_pred_rf_CV = rf_CV.predict(count_vect_test)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred_rf_CV))
```

```
Accuracy : 0.5810001497230124
```

➢ Logistic Regression with Tf-idf

```
logreg_TV = LogisticRegression(class_weight='balanced', random_state=5)
logreg_TV.fit(tfidf_vect_train, y_train)
y_pred_lr_TV = logreg_TV.predict(tfidf_vect_test)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred_lr_TV))
print (classification_report(y_test, y_pred_lr_TV))
```

```
Accuracy : 0.618580625842192
              precision    recall  f1-score   support

    negative       0.71      0.72      0.72      6124
     neutral       0.43      0.41      0.42      2917
    positive       0.61      0.62      0.61      4317

    accuracy                           0.62     13358
   macro avg       0.58      0.58      0.58     13358
weighted avg       0.62      0.62      0.62     13358
```

Machine Learning Models

- Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer

text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])

text_clf.fit(X_train, y_train)

print(text_clf.score(X_test, y_test))
```

```
0.5841443329839796
```

Machine Learning Models

- Grid Search

```
grid_search.fit(X_train, y_train)

print('score', grid_search.score(X_test, y_test))
print('-----')

print('GridSearchCV:')
y_pred = grid_search.predict(X_test)

print("Best score: %0.3f" % grid_search.best_score_)
print("Best parameters set:")
best_parameters = grid_search.best_estimator_.get_params()
for param_name in sorted(parameters.keys()):
    print("\t%s: %r" % (param_name, best_parameters[param_name]))
```

```
Fitting 3 folds for each of 64 candidates, totalling 192 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done 192 out of 192 | elapsed:   28.8s finished
score 0.5946998053600838
-----
GridSearchCV:
Best score: 0.593
Best parameters set:
	clf__alpha: 1.0
	vect__max_df: 0.7
	vect__min_df: 10
	vect__stop_words: 'english'
```

Machine Learning Models

- Deep Learning Algorithm

```
import keras
from keras.preprocessing.text import Tokenizer

tokenizer = Tokenizer(num_words=2500,split=' ')
tokenizer.fit_on_texts(dfnew.Cleaned)
```

```
from keras.preprocessing.sequence import pad_sequences

X = tokenizer.texts_to_sequences(dfnew.Cleaned)
X = pad_sequences(X)
print(X)
```

```
[[   0    0    0 ...  106  538   52]
 [   0    0    0 ...    0 1637  207]
 [   0    0    0 ...  669  147  111]
 ...
 [   0    0    0 ...  603  944  479]
 [   0    0    0 ... 1702    2   29]
 [   0    0    0 ...   42   55    4]]
```

```
Epoch 1/5
 - 336s - loss: 0.8706 - acc: 0.5974
Epoch 2/5
 - 333s - loss: 0.7893 - acc: 0.6472
Epoch 3/5
 - 333s - loss: 0.7533 - acc: 0.6618
Epoch 4/5
 - 333s - loss: 0.7174 - acc: 0.6812
Epoch 5/5
 - 333s - loss: 0.6860 - acc: 0.6964
```

Conclusion

- We have chosen a difficult and informal text in order to make it harder to analyze the sentiment (general text, not a review or a sentiment pool data)

- The distribution of the features are not balanced. In order to deal with this problem, we concatenated additional 15 thousands row data set, we limited the categorization, we applied random forest hyper-tuning and feature engineering.

- We used random forest model to balance the distribution and Grid Search with 5-fold cross validation technique to deal with the overfitting problem. Because of the reasons mentioned above, our best score is 0.68 after applying Tf-idf and deep learning algorithm.

- Most important things for an effective sentiment analysis of short social media texts are data preprocessing, feature engineering and choosing the best model

Future Works

- For increasing the accuracy of our model, it is very important to find additional balanced data sets, and applying effective feature engineering techniques.
- Applying Word2vec or Phrase modeling could also improves the model.
- Categorization of sentiments in 2 classes (such as bad or not bad) could also give higher results.
- Run time for these kinds of data with decision tree models and deep learning models is relatively long. Decreasing the run time with more efficient computers or cloud systems could be used for increasing the effectiveness.

Thank You