

Introduction

In this project, I took the data from [Analytics Vidhya Hackathon: Identify the sentiments](#). This is an online competition platform for data science projects and in this NLP project I achieved the position on 12th among thousands of participants.

Example tweet:

```
#fingerprint #Pregnancy Test https://goo.gl/h1MfQV #android #apps  
#beautiful #cute #health #igers #iphoneonly #iphonesia #iphone
```

Cleaned text:

```
fingerprint pregnancy test android aps beautiful cute health igers  
iphoneonly iphonesia iphone
```

hashtags:

```
#fingerprint #Pregnancy #android #apps #beautiful #cute #health #igers  
#iphoneonly #iphonesia #iphone
```

We have 7920 training tweets.

We also have 1953 test tweets without label. (we need to upload the test predictions to get the weighted F1 score.)

Model Comparisons

Notebook	Valid F1	Test F1
b01_sentiment_analysis_modelling_bow_word2vec_tfidf	0.885101	
b02_sentiment_analysis_modelling_tfidf	0.876263	
---	---	---
c01_sentiment_analysis_ktrain	0.88	
c01b_sentiment_analysis_ktrain_neptune	0.924609	0.907575
c01c_sentiment_analysis_ktrain_neptune_hpo	0.917368	0.877973
c02_sentiment_analysis_simpletransformers_wandb_roberta_full_data		
---	---	---
d01_sentiment_analysis_keras_lstm	0.860	0.83785
d02_sentiment_analysis_keras_gru_gbru	0.871895	
---	---	---
e01_sentiment_analysis_small_data_transformers_distilbert_torch	0.9120	

Notebook	Valid F1	Test F1
e02_sentiment_analysis_transformers_distilbert_keras	0.663583	
e03_sentiment_analysis_bert_tf2	0.884748	
e03b_sentiment_analysis_bert_tf2_neptune	0.8787	

Text Analysis and Visualization

- First do data cleaning.
- PCA plot. (dimension reduction)
- Most frequent words for positive and negative sentiment tweets.
- Wordcloud
- Treemap
- kde plots for +ve and -ve sentiments for new added features.
- n-grams

Classical Methods

- We need data cleaning and feature creation.
- Embedding: BoW (CountVectorizer), TF-IDF, Word2Vec
- Algorithms: LogisticRegression, LinearSVC.

	Text Model	Params	Model	Description	F1 Weighted	Time Taken	Accuracy	Precision	Recall	Time Taken Sec
0	Word2Vec	Extra	logregcv	cv=2	0.885101	0.775309	0.775309	0.885101	23.28 sec	23.282527
2	tfidf	Extra	logregcv	cv=2	0.876263	0.772846	0.730864	0.875107	29.80 sec	29.803477
4	tfidf	Extra+Scaling	logregcv	cv=2	0.873737	0.790368	0.688889	0.870769	26.79 sec	26.791343
8	tfidf		logregcv	cv=2	0.855429	0.775000	0.612346	0.849470	2.05 sec	2.048890
9	tfidf		svc	max_iter=200	0.855429	0.795302	0.585185	0.847565	0.07 sec	0.069284

	Text Model	Params	Model	Description	F1 Weighted	Time Taken	Accuracy	Precision	Recall	Time Taken Sec
1	tfidf	Extra	logregcv	cv=2	0.876263	0.772846	0.730864	0.875107	32.41 sec	32.407605
2	tfidf	Extra+Scaling	logregcv	cv=2	0.873737	0.790368	0.688889	0.870769	30.90 sec	30.895569
3	tfidf		svc	max_iter=200	0.855429	0.795302	0.585185	0.847565	0.14 sec	0.135881
0	tfidf		logregcv	cv=2	0.849470	3.18 sec	0.855429	0.775000	0.612346	3.175989
7	tfidf		sgd		0.848485	0.753846	0.604938	0.842669	0.05 sec	0.048415
6	tfidf	n_estimators=1000	rf		0.832071	0.849246	0.417284	0.810180	16.32 sec	16.320531
5	tfidf		bernoulli		0.823232	0.804878	0.407407	0.801164	1.07 sec	1.073401
4	tfidf		gnb		0.796086	0.621302	0.518519	0.789709	1.67 sec	1.666834

Deep Learning: LSTM, GRU

- First we do text processing.
- Prepare data using keras text processing tools `Tokenizer` and `sequence`.

- Keras sequential model using LSTM
- Keras sequential model using GRU

Advanced method: Using module ktrain

- We don't need data cleaning.
- supports 'fasttext' 'nbsvm' 'logreg' 'bigru' 'bert' 'distilbert'.

```
best_so_far = ""
bert lr=2e-5 epochs=5 ngram_range=1 maxlen=300
f1 = 0.908687336005899

n_gram=2 gave worse result
tweet_clean_emoji gave worse result

bert lr=2e-5 epochs=5 ngram_range=1 maxlen=400
f1 = 0.908265806079951

bert lr=2e-5 epochs=5 ngram_range=1 maxlen=300 maincol=tweet_clean
f1=0.877973006703751

""
```

Advanced Method: Using module simpletransformers

```
from simpletransformers.classification import ClassificationModel

model_type = 'xlnet'
model_name = 'xlnet-base-cased'

model = ClassificationModel(model_type, model_name, args=train_args)
model.train_model(df_train, eval_df=None)

test_preds, _ = model.predict(df_test['tweet'].to_numpy())
```