

FastText

Loading the Data

```
!gdown --id 15LVlv7K31SzBs_IyZE_md0u6qEg6rby6
!unzip dataminingmtl782.zip
```

Importing Required Libraries

```
!pip install -q wordcloud
import wordcloud

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.stem.wordnet import WordNetLemmatizer
import pickle
import cv2
import re
from pylab import rcParams
from matplotlib import rc
from nltk.tokenize import word_tokenize
from textblob import TextBlob
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix
from tqdm import tqdm
import torch
from torch import nn, optim
from torch.utils import data
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
sns.set(style = 'whitegrid', palette = 'muted', font_scale = 1.2)
HAPPY_COLOURS_PALETTE = ['#01BEFE', '#FFDD00', '#FF7D00', '#FF006D', '#ADFF02', '#8F00FF']
sns.set_palette(sns.color_palette(HAPPY_COLOURS_PALETTE))
rcParams['figure.figsize'] = 8,8
RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
torch.manual_seed(RANDOM_SEED)
```

Loading the file Data-frames

```
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
train_df
```

Setting Hyper Parameters

```
IMG_SCALING = (3, 3)
BATCH_SIZE = 48
GAUSSIAN_NOISE = 0.1
N_CLASSES = 3
dim = (192,192)
MAIN_PATH = ''
```

Exploratory Data Analysis and Text pre-processing

```
print(train_df['image id'].duplicated().any()) # Check if there are any duplicated images in the data
train_df['label'].value_counts().plot(kind = 'barh')# Plot the counts of each label
train_df['label_num'].value_counts().plot(kind = 'barh') # Plot the counts of each label_num
```

```
sns.countplot(train_df.label_num)
plt.xlabel('Sentiment')
```

```
train_df['text'] = train_df['text'].apply(lambda x:x.lower())
test_df['text'] = test_df['text'].apply(lambda x: x.lower())# Convert the text in lower case
print(train_df[train_df['text']=='#name?']) # Check if there are any empty text images in the dataset
print(test_df[test_df['text'] == '#name?'])
train_df['text']
```

```
train_df.isnull().sum() #Check number of null entries in the dataframe
```

```
train_df.drop(858,inplace=True)# Remove the corrupted Image
train_df[ train_df['image id']=='image_6357.jpg' ]
```

```
num = 858#Check that the corrupted image has been removed
train_df['label'].iloc[num]
img = cv2.imread(MAIN_PATH +'train_images/train_images/'+train_df['image id'].iloc[num])
plt.figure(figsize = (12*1.2,8*1.2))
plt.imshow(img)
plt.axis('off')
```



```

sample = re.sub(r"http\S+", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+.COM", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+.com", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+.com", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+.COM", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+ COM", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+ com", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+ com", "", sample)
sample = re.sub(r"[a-zA-Z0-9_@]+ COM", "", sample)
return sample
# string = re.sub(r'^https?:\/\/\.[^\r\n]*', '', string, flags=re.MULTILINE) # WILL REMOVE
HYPERLINKS!!!!
return string.strip().lower()

train_df['text'] = train_df['text'].apply(clean_str) # Should be very very clean

```

```

import nltk
nltk.download('punkt')
def TOK(text):
    return word_tokenize(text.lower())
train_df['tokens'] = train_df.text.apply(TOK)

train_df

```

```

test_df['tokens'] = test_df.text.apply(TOK)

test_df

```

```

nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = [word.lower() for word in stopwords.words('english')]
def remove_stopwords(tokens):
    ans = []
    for tok in tokens:
        if tok in stop_words:
            continue
        pas = True
        for s in '@!`~\'\"\\*&^%$#-=[.,<>?':
            if s in tok:
                pas = False
                break
        if not pas:
            continue
        if 'meme' in tok:
            continue
        ans.append(tok)
    return ans
# print(train_df.tokens.iloc[599],remove_stopwords(train_df.tokens.iloc[599]))

```

```

train_df.tokens = train_df.tokens.apply(remove_stopwords)
test_df.tokens = test_df.tokens.apply(remove_stopwords)

```

```
def Tokens_to_text(tokens):
    s = ''
    for token in tokens:
        s+=token+' '
    return s.lower().strip()
def maxLen(df):
    x = 0
    for i, tokens in enumerate(df.tokens):
        x = max(x, len(tokens))
    return x
max_words = maxLen(train_df)
```

```
for i, text in enumerate(train_df.text):
    print(i, text)
```

```
train_df.text = train_df.tokens.apply(Tokens_to_text)
test_df.text = test_df.tokens.apply(Tokens_to_text)
```

```
for i, text in enumerate(train_df.text):
    print(i, text)
```

Before processing vs after processing

Before: 3...2...1..dont do it timmy!

After: dont do it timmy

Before: a vote for trump is a vote for putin 2009 www.protectourelections.com

After a vote for trump is a vote for putin 2009

Before: "i start where the last man left off." thomas edison visit: www.cettechnology.com/memes
for more quotes @ techsolmarketing.com - free for use without modification

After: i start where the last man left off thomas edison visit: memes for more quotes @ free for use without modification

Converting text into FastText format

```
# fastText format
def Convert_to_fast_text(df, target):
    fp = open(target, "w")
    for i, text in enumerate(df.text):
        fp.write("__label__"+df.label.iloc[i]+' '+text+'\n')
    fp.close()
    return
Convert_to_fast_text(train_df, 'train.txt')
#Convert_to_fast_text(test_df, 'test.txt')
```

Loading FastText

```
! pip install fasttext
import fasttext
```

Training the model

```
model = fasttext.train_supervised('train.txt')
print(model.labels)
```

Making predictions

```
fp = open('predictions.txt', 'w')
for i in range(len(test_df)):
    fp.write(str(model.predict(test_df.text.iloc[i]))+" "+str(test_df.label.iloc[10])+'\n')
fp.close()
```

```
%cat predictions.txt
```

```
def getPredfromfasttext(Filename):
    fp = open(Filename)
    y_pred = []
    for line in fp:
        if '__label__hilarious' in line:
            y_pred+= [1]
        elif '__label__troll' in line:
            y_pred+= [2]
        else:
            y_pred+= [0]
    return y_pred
y_pred = getPredfromfasttext('predictions.txt')
```

```
y_pred
```

Making Submission

```
submit = pd.read_csv('Sample_submission.csv')
submit.label_num = y_pred
submit.to_csv("Submission.csv", index = False)
```