05_Text Classification

0.1 Text Classification

The task is to build a machine learning model to **classify** whether a particular tweet is **hate speech** or **not**.

0.1.1 Table of Contents

0.1.2 1. About the Dataset

The dataset that you are going to use is of **Detecting Hate Speech** in people's tweets. Let's load the dataset using pandas and have a quick look at some sample tweets.

[1]:
```
#Load the dataset
import pandas as pd

dataset = pd.read_csv('data/final_dataset_basicmlmodel.csv')
dataset.head()
```

[1]:
```
   id  label                                              tweet
0   1      0   @user when a father is dysfunctional and is s…
1   2      0  @user @user thanks for #lyft credit i can't us…
2   3      0                                bihday your majesty
3   4      0  #model   i love u take with u all the time in …
4   5      0             factsguide: society now    #motivation
```

Things to note - **label** is the column that contains the target variable or the value that has to be predicted. 1 means it's a hate speech and 0 means it is not. - **tweet** is the column that contains the text of the tweet. This is the main data on which NLP techniques will be applied.

Let's have a close look at some of the tweets.

[2]:
```
for index, tweet in enumerate(dataset["tweet"][10:15]):
    print(index+1,".",tweet)
```

```
1 .  â  #ireland consumer price index (mom) climbed from previous 0.2% to 0.5%
in may   #blog #silver #gold #forex
2 . we are so selfish. #orlando #standwithorlando #pulseshooting
#orlandoshooting #biggerproblems #selfish #heabreaking   #values #love #
3 . i get to see my daddy today!!   #80days #gettingfed
4 . ouch…junior is angryð #got7 #junior #yugyoem   #omg
5 . i am thankful for having a paner. #thankful #positive
```

Note :- Noise present in Tweets

- If you look closely, you'll see that there are many hashtags present in the tweets of the form # symbol followed by text. We particularly don't need the # symbol so we will clean it out.
- Also, there are strange symbols like â and ð in tweet 4. This is actually `unicode` characters that is present in our dataset that we need to get rid of because they don't particularly add anything meaningful.
- There are also numerals and percentages .

0.1.3 2. Data Cleaning

Let's clean up the noise in our dataset.

[3]:
```python
import re

#Clean text from noise
def clean_text(text):
    #Filter to allow only alphabets
    text = re.sub(r'[^a-zA-Z\']', ' ', text)

    #Remove Unicode characters
    text = re.sub(r'[^\x00-\x7F]+', '', text)

    #Convert to lowercase to maintain consistency
    text = text.lower()

    return text
```

[4]:
```python
dataset['clean_text'] = dataset.tweet.apply(lambda x: clean_text(x))
```

0.1.4 3. Feature Engineering

- Feature engineering is the science (and art) of extracting more information from existing data. You are not adding any new data here, but you are actually making the data you already have more useful.
- The machine learning model does not understand text directly, **so we create numerical features that reperesant the underlying text**.
- In this module, you'll deal with very basic NLP based features and as you progress further in the course you'll come across more complex and efficient ways of doing the same.

[5]:
```python
#Exhaustive list of stopwords in the english language. We want to focus less on␣
↪these so at some point will have to filter
STOP_WORDS = ['a', 'about', 'above', 'after', 'again', 'against', 'all',␣
↪'also', 'am', 'an', 'and',
              'any', 'are', "aren't", 'as', 'at', 'be', 'because', 'been',␣
↪'before', 'being', 'below',
              'between', 'both', 'but', 'by', 'can', "can't", 'cannot', 'com',␣
↪'could', "couldn't", 'did',
              "didn't", 'do', 'does', "doesn't", 'doing', "don't", 'down',␣
↪'during', 'each', 'else', 'ever',
              'few', 'for', 'from', 'further', 'get', 'had', "hadn't", 'has',␣
↪"hasn't", 'have', "haven't", 'having',
              'he', "he'd", "he'll", "he's", 'her', 'here', "here's", 'hers',␣
↪'herself', 'him', 'himself', 'his', 'how',
              "how's", 'however', 'http', 'i', "i'd", "i'll", "i'm", "i've",␣
↪'if', 'in', 'into', 'is', "isn't", 'it',
              "it's", 'its', 'itself', 'just', 'k', "let's", 'like', 'me',␣
↪'more', 'most', "mustn't", 'my', 'myself',
              'no', 'nor', 'not', 'of', 'off', 'on', 'once', 'only', 'or',␣
↪'other', 'otherwise', 'ought', 'our', 'ours',
              'ourselves', 'out', 'over', 'own', 'r', 'same', 'shall',␣
↪"shan't", 'she', "she'd", "she'll", "she's",
              'should', "shouldn't", 'since', 'so', 'some', 'such', 'than',␣
↪'that', "that's", 'the', 'their', 'theirs',
              'them', 'themselves', 'then', 'there', "there's", 'these',␣
↪'they', "they'd", "they'll", "they're",
              "they've", 'this', 'those', 'through', 'to', 'too', 'under',␣
↪'until', 'up', 'very', 'was', "wasn't",
              'we', "we'd", "we'll", "we're", "we've", 'were', "weren't",␣
↪'what', "what's", 'when', "when's", 'where',
              "where's", 'which', 'while', 'who', "who's", 'whom', 'why',␣
↪"why's", 'with', "won't", 'would', "wouldn't",
              'www', 'you', "you'd", "you'll", "you're", "you've", 'your',␣
↪'yours', 'yourself', 'yourselves']

#Generate word frequency
def gen_freq(text):
    #Will store the list of words
    word_list = []

    #Loop over all the tweets and extract words into word_list
    for tw_words in text.split():
        word_list.extend(tw_words)

    #Create word frequencies using word_list
    word_freq = pd.Series(word_list).value_counts()
```

```
    #Drop the stopwords during the frequency calculation
    word_freq = word_freq.drop(STOP_WORDS, errors='ignore')

    return word_freq

#Check whether a negation term is present in the text
def any_neg(words):
    for word in words:
        if word in ['n', 'no', 'non', 'not'] or re.search(r"\wn't", word):
            return 1
    else:
        return 0

#Check whether one of the 100 rare words is present in the text
def any_rare(words, rare_100):
    for word in words:
        if word in rare_100:
            return 1
    else:
        return 0

#Check whether prompt words are present
def is_question(words):
    for word in words:
        if word in ['when', 'what', 'how', 'why', 'who']:
            return 1
    else:
        return 0
```

[6]:
```
word_freq = gen_freq(dataset.clean_text.str)
#100 most rare words in the dataset
rare_100 = word_freq[-100:]
#Number of words in a tweet
dataset['word_count'] = dataset.clean_text.str.split().apply(lambda x: len(x))
#Negation present or not
dataset['any_neg'] = dataset.clean_text.str.split().apply(lambda x: any_neg(x))
#Prompt present or not
dataset['is_question'] = dataset.clean_text.str.split().apply(lambda x:␣
↪is_question(x))
#Any of the most 100 rare words present or not
dataset['any_rare'] = dataset.clean_text.str.split().apply(lambda x:␣
↪any_rare(x, rare_100))
#Character count of the tweet
dataset['char_count'] = dataset.clean_text.apply(lambda x: len(x))
```

[7]:
```
#Top 10 common words are
gen_freq(dataset.clean_text.str)[:10]
```

[7]:
```
user      3351
amp        439
love       320
day        254
trump      214
happy      207
will       191
people     186
new        171
u          158
dtype: int64
```

[8]:
```
dataset.head()
```

[8]:
```
   id  label                                              tweet  \
0   1      0   @user when a father is dysfunctional and is s…
1   2      0  @user @user thanks for #lyft credit i can't us…
2   3      0                              bihday your majesty
3   4      0  #model   i love u take with u all the time in …
4   5      0             factsguide: society now    #motivation

                                        clean_text  word_count  any_neg  \
0     user when a father is dysfunctional and is s…          18        0
1    user   user thanks for  lyft credit i can't us…          19        1
2                               bihday your majesty           3        0
3    model   i love u take with u all the time in …          12        0
4              factsguide  society now     motivation           4        0

   is_question  any_rare  char_count
0            1         0         102
1            0         0         122
2            0         0          21
3            0         0          86
4            0         0          39
```

0.1.5 Splitting the dataset into Train-Test split

- The dataset is split into train and test sets so that we can evaluate our model's performance on unseen data.
- The model will only be trained on the `train` set and will make predictions on the `test` set whose data points the model has never seen. This will make sure that we have a proper way to test the model.

This is a pretty regular practice in Machine Learning, don't worry if you are confused. It's just a way of testing your model's performance on unseen data.

[9]:
```
from sklearn.model_selection import train_test_split

X = dataset[['word_count', 'any_neg', 'any_rare', 'char_count', 'is_question']]
y = dataset.label

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
↪random_state=27)
```

0.1.6 4. Train an ML model for Text Classification

Now that the dataset is ready, it is time to train a Machine Learning model on the same. You will be using a **Naive Bayes** classifier from `sklearn` which is a prominent python library used for machine learning.

[10]:
```
from sklearn.naive_bayes import GaussianNB

#Initialize GaussianNB classifier
model = GaussianNB()
#Fit the model on the train dataset
model = model.fit(X_train, y_train)
#Make predictions on the test dataset
pred = model.predict(X_test)
```

0.1.7 5. Evaluate the ML model

It is time to train the model on previously unseen data: **X_test** and **y_test** sets that you previously created. Let's check the accuracy of the model.

[11]:
```
from sklearn.metrics import accuracy_score

print("Accuracy:", accuracy_score(y_test, pred)*100, "%")
```

```
Accuracy: 59.61904761904761 %
```

0.1.8 6. Conclusion

Note: that since we have used very basic NLP features, the classification accuracy and f1 scores aren't that impressive, which can be improved using other calssification algorithms.

[ ]:
```

```