

# 04\_Exploring Text Data

April 24, 2022

## 0.1 Exploring Text Data

Working with text is generally more challenging than working with numerical data. Hence, any kind of technique that helps in generating an intuition of the existing dataset is welcome. One of the simplest approach to understand any text document or to compare multiple documents can be to compute a frequency table of individual words present in the document/documents and use it to conduct further experiments like: finding top words per document, finding top common words among documents etc.

Let us take the challenge of **Analyzing Sentiments from Twitter data**, so we will focus on how to generate word frequencies and use it to create **Word Clouds** in Python that will help us get a better overall understanding of the dataset.

### 0.1.1 Table of Contents

1. About the Dataset
2. Generating Word Frequency
3. EDA using Word Clouds
4. Why to Preprocess text data?

### 0.1.2 1. About the Dataset

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/tweets.csv', encoding = 'ISO-8859-1')
dataset.head()
```

```
[1]: Unnamed: 0  X                                text \
0          1  1  RT @rsshurjewala: Critical question: Was PayTM ...
1          2  2  RT @Hemant_80: Did you vote on #Demonetization...
2          3  3  RT @roshankar: Former FinSec, RBI Dy Governor,...
3          4  4  RT @ANI_news: Gurugram (Haryana): Post office ...
4          5  5  RT @satishacharya: Reddy Wedding! @mail_today ...

      favorited  favoriteCount  replyToSN      created  truncated \
0      False          0          NaN  2016-11-23 18:40:30      False
1      False          0          NaN  2016-11-23 18:40:29      False
2      False          0          NaN  2016-11-23 18:40:03      False
```

3	False	0	NaN	2016-11-23 18:39:59	False
4	False	0	NaN	2016-11-23 18:39:39	False

	replyToSID	id	replyToUID	\
0	NaN	8.014957e+17	NaN	
1	NaN	8.014957e+17	NaN	
2	NaN	8.014955e+17	NaN	
3	NaN	8.014955e+17	NaN	
4	NaN	8.014954e+17	NaN	

	statusSource	screenName	\
0	<a href="http://twitter.com/download/android" ...	HASHTAGFARZI WAL	
1	<a href="http://twitter.com/download/android" ...	PRAMODKAUSHIK9	
2	<a href="http://twitter.com/download/android" ...	rahulja13034944	
3	<a href="http://twitter.com/download/android" ...	deeptiyvd	
4	<a href="http://cpimharyana.com" rel="nofollow...	CPIMBadli	

	retweetCount	isRetweet	retweeted
0	331	True	False
1	66	True	False
2	12	True	False
3	338	True	False
4	120	True	False

```
[2]: dataset.shape
```

```
[2]: (14940, 16)
```

As can be seen above, **text** column is of interest to us as it contains the tweet. At this point, you don't have to worry about other columns as that will be handled in future sessions. Let's go ahead and inspect some of the tweets.

### 0.1.3 2. Generating Word Frequency

Let's first generate a frequency table of all the words present in all the tweets combined.

```
[3]: 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()

    #Print top 20 words
```

```

word_freq[:20]

return word_freq

word_freq=gen_freq(dataset.text.str)
word_freq

```

```

[3]: RT                11053
    to                 7650
    is                 5152
    in                 4491
    the                4331
    ...
    #News              1
    notes|             1
    https://t.co/EC14oIzdHA 1
    https://t.co/9MjFtLtCtR 1
    https://t.co/hwgqjbqgvG 1
    Length: 19601, dtype: int64

```

### 0.1.4 3. EDA using Word Clouds

Now that you have successfully created a frequency table, you can use that to create multiple **visualizations** in the form of word clouds. Sometimes, the quickest way to understand the context of the text data is using a word cloud of top 100-200 words. Let's see how to create that in Python.

**Note:-** You'll use the WordCloud library of Python. You can install it by -

pip install wordcloud

```
[4]: !pip install wordcloud
```

```

Requirement already satisfied: wordcloud in
c:\users\administrator\anaconda3\lib\site-packages (1.8.1)
Requirement already satisfied: matplotlib in
c:\users\administrator\anaconda3\lib\site-packages (from wordcloud) (3.4.3)
Requirement already satisfied: numpy>=1.6.1 in
c:\users\administrator\anaconda3\lib\site-packages (from wordcloud) (1.20.3)
Requirement already satisfied: pillow in
c:\users\administrator\anaconda3\lib\site-packages (from wordcloud) (8.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\users\administrator\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(3.0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\administrator\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in
c:\users\administrator\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(0.10.0)

```

Requirement already satisfied: python-dateutil>=2.7 in  
c:\users\administrator\anaconda3\lib\site-packages (from matplotlib->wordcloud)  
(2.8.2)

Requirement already satisfied: six in c:\users\administrator\anaconda3\lib\site-  
packages (from cyciler>=0.10->matplotlib->wordcloud) (1.16.0)

```
[5]: #Import libraries
import matplotlib.pyplot as plt
from wordcloud import WordCloud

#Generate word cloud
wc = WordCloud(width=400, height=330, max_words=100,
               background_color='white').generate_from_frequencies(word_freq)

plt.figure(figsize=(12, 8))
plt.imshow(wc, interpolation='bilinear')
plt.axis('off')
plt.show()
```



[6]: WordCloud?

### Few things to Note:-

1. There is noise in the form of “RT” and “&” which can be removed from the word frequency.
2. Stop words like “the”, “in”, “to”, “of” etc. are obviously ranking among the top frequency words but these are just constructs of the English language and are not specific to the people’s tweets.
3. Words like “demonetization” have occurred multiple times. The reason for this is that the current text is not **Normalized** so words like “demonetization”, “Demonetization” etc. are all considered as different words.

The above are some of the problems that we need to address in order to make better visualization. Let’s solve some of the problems!

### Text Cleaning

```
[7]: import re

def clean_text(text):
    #Remove RT
    text = re.sub(r'RT', '', text)

    #Fix &
    text = re.sub(r'&', '', text)

    #Remove punctuations
    text = re.sub(r'[?!.,:;#@-]', '', text)

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

    # remove digits
    text = re.sub(r'\d+', '', text)

    #Fix %
    text = re.sub(r'%', '', text)

    return text
```

The above will solve problems related to RT, & and also the problem of counting same word twice due to case difference. Yet we can do better, let’s remove the common stop words.

**Stop words Removal** WordCloud provides its own stopwords list. You can have a look at it by-

```
[8]: #Import list of stopwords
from wordcloud import STOPWORDS
```

```
print(STOPWORDS)
```

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

Now that you know what all has to be changed to improve our word cloud, let's make some wordclouds. We'll call the previous functions of `clean_text()` and `gen_freq()` to perform cleaning and frequency computation operation respectively and drop the words present in `STOPWORDS` from the `word_freq` dictionary.

```
[9]: text = dataset.text.apply(lambda x: clean_text(x))  
word_freq = gen_freq(text.str)  
word_freq = word_freq.drop(labels=STOPWORDS, errors='ignore')  
  
#Generate word cloud  
wc = WordCloud(width=450, height=330, max_words=200,  
               background_color='white').generate_from_frequencies(word_freq)  
  
plt.figure(figsize=(12, 8))  
plt.imshow(wc, interpolation='bilinear')  
plt.axis('off')  
plt.show()
```

