

WEEK 9 – Final Project

Jaques D'Erasmus Santos Fernando

Abstract

This project is a sentiment analysis about Netflix. The dataset used was a 2000 tweets requested from Twitter, and the Nick Snader's tweet's corpus, that is a collection of nearly 5000 classified tweets labelled as positive, negative, neutral or irrelevant. The research questions was: What is the sentiment about the Netflix; what is the most frequent words on those 2000 tweets; what is the word frequency and the rank of words in a specific frequency interval. In this work, the methodology used was the sentiment analysis using Twitter API and NLTK Naive Bayes Classifier, and the findings are the sentiment about the Netflix and few different perspectives about the word frequency, including a word cloud model, based on the 2000 tweets.

Motivation

The actual project proposes a sentiment analysis about the Netflix platform. This event is an important evaluation to the Netflix company decision makers since the opinion from the users can be considered directed related with its possibility to keep growing their business or, if necessary, review their services to better attend their audience.

To proper address this problem, it will be used the Twitter API to obtain a certain amount of tweets, and the NLTK Naive Bayes Classifier in order to train the Machine Learning Model. It is important to mention that the train dataset is called Nick Sander's corpus, created by Nick Sander, and that contain approximately 5000 of tweets properly labelled. A particularity about this corpus is that it was initially created to be used to train model specifically related with high technological companies as Google, Amazon, Apple, Windows, YouTube and Netflix, for example.

Word frequencies, including a amazing WordCloud view will be presented, as well.

Dataset(s)

Two datasets will be used in this project

1 – A collection of 2000 tweets collected from Twitter platform through the Twitter API system;

2 – Nick Sander's tweets corpus with approximately 5000 classified tweets labelled as positive, negative, neutral or irrelevant. Link to download the corpus https://github.com/zfz/twitter_corpus

Data Preparation and Cleaning

First, it was necessary to remove the duplicates. The Twitter always duplicates few tweets. Also, it was necessary to remove stopwords (words that is undesirable to the analysis, replace and remove URL links link 'www', http or https, replace and remove @ and # symbols, remove emotion icons and tokenize words.

Research Question(s)

What is the sentiment analysis about the Netflix platform services based on its user's opinions on Twitter?

Additionally to this main research question few other secondary questions will also be answered, as for example: What are the top 20 words in the vocabulary from the tweets used on the sentiment analysis? Is the vocabulary of words large or it is a short vocabulary? How a Word Cloud from the tweets looks like, and can this visualization be helpful?

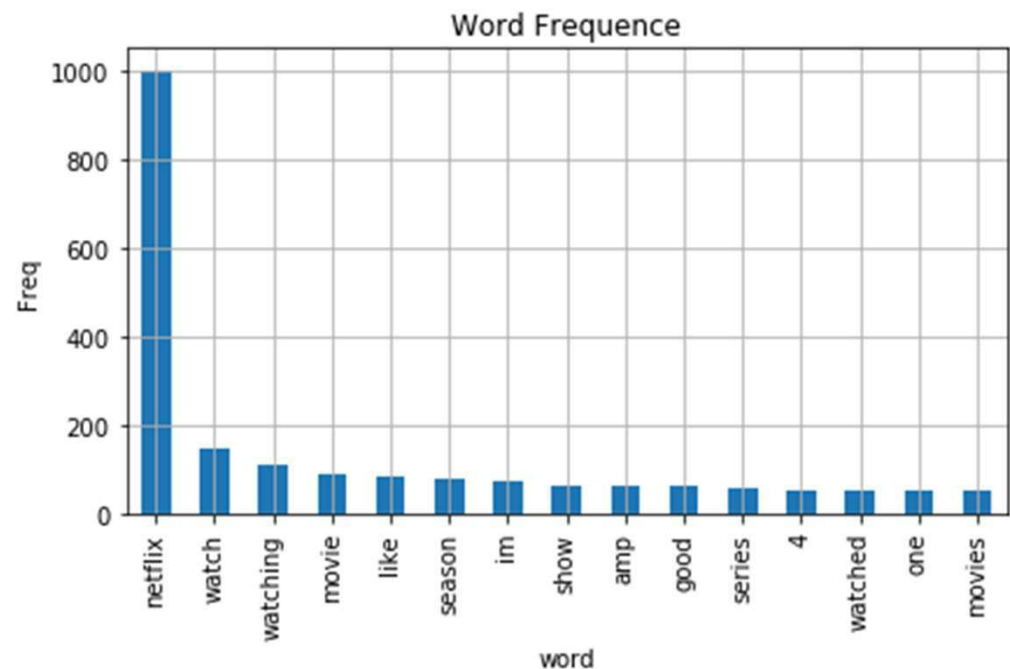
Methods

In order to perform a sentiment analysis about Netflix, it will be used the Twitter API System to obtain the tweets and the NLTK Naive Bayes Classifier to train a Machine Learning Classifier model.

Findings

Word frequency – Top 20 words

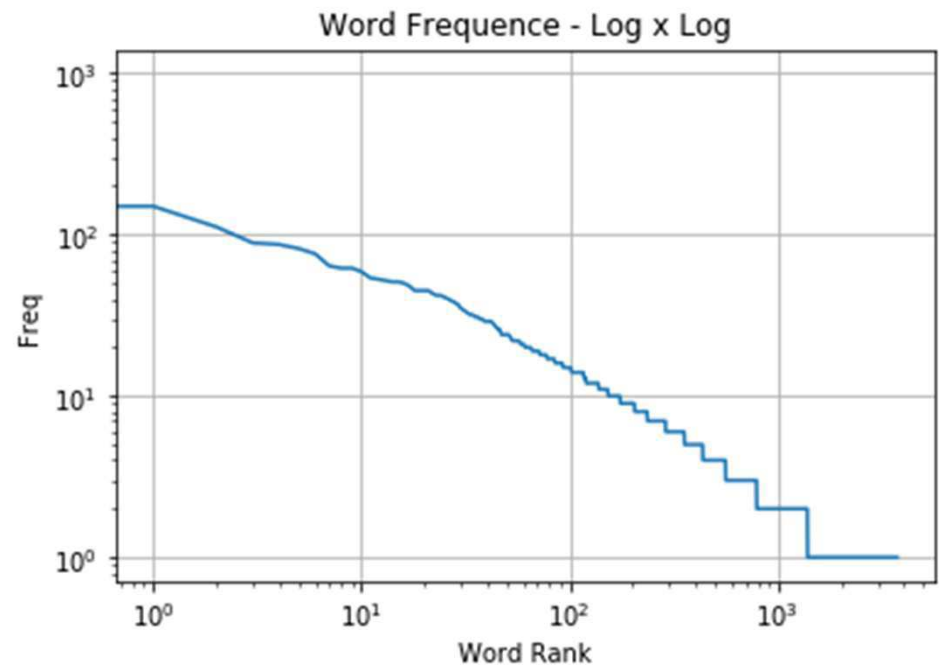
The graph in the right provides a quick and clear visualization about the top 20 most frequent words in the 2000 tweets obtained from the Twitter platform related with Netflix service.



Findings

Word frequency – Log x Log

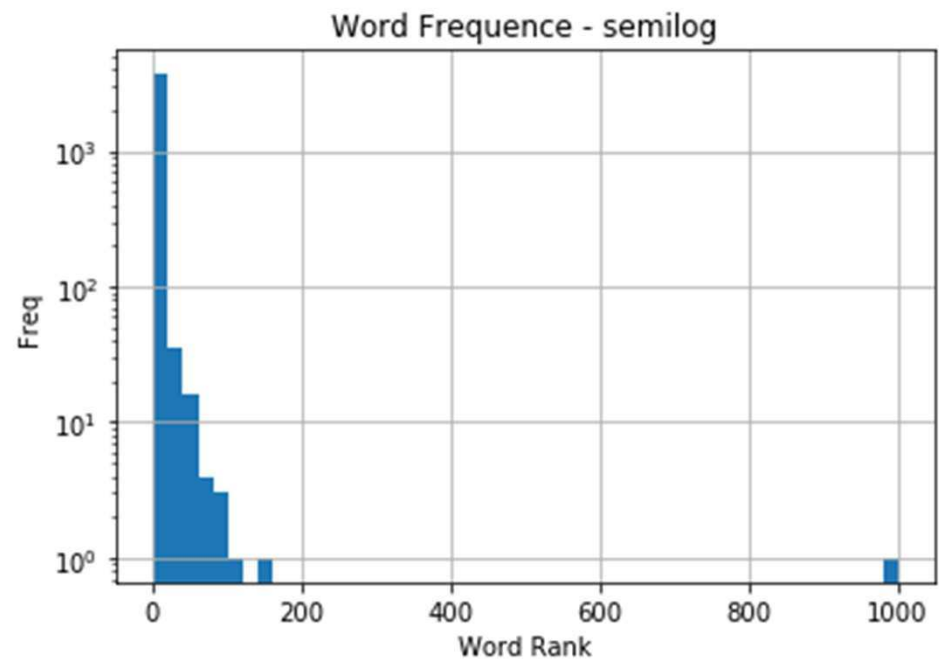
The flat shape of the log x log word frequency graph indicates that there is a large range of words in the tweets, or simply, it has a large vocabulary.



Findings

Word frequency – semiLog

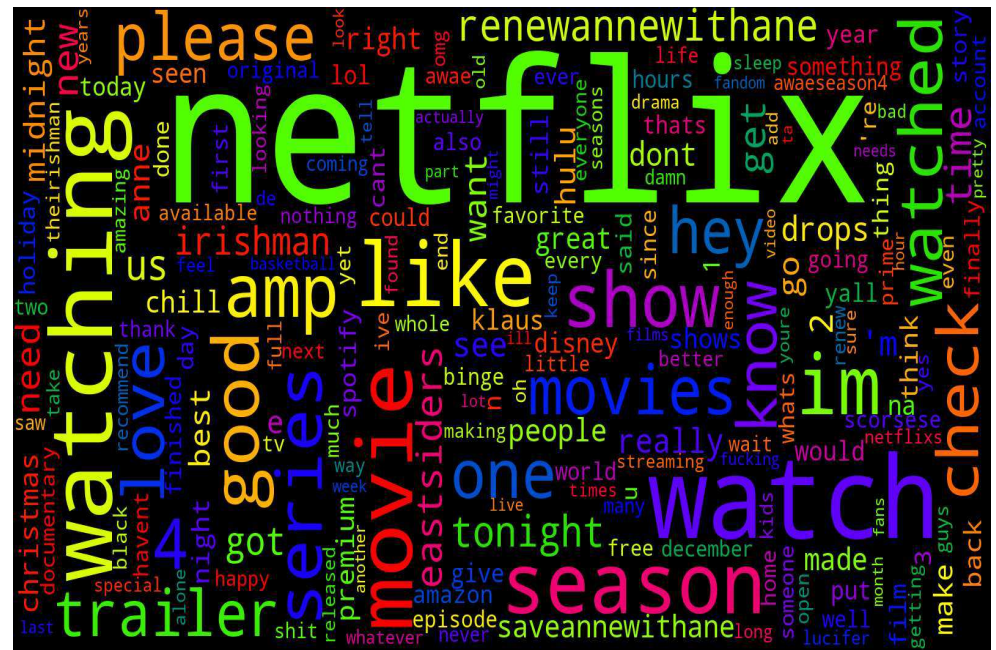
The *semilog* word frequency graph allows us to display how many words have a count in a specific range. In this specific analysis, our range is between 0 and 200 words where the most frequent words in the vocabulary are located.



Findings

Word frequency – Word Cloud

The *Word Cloud* is, in my opinion, one of the most sophisticated and easy to understand, visualization methods for a sentiment analysis. Even not providing the final sentiment result it easily leads to a major comprehension about the analyzed topic.



Findings – Sentiment Analysis Result

```
-----  
SENTIMENT RESULTS USING NAIVE BAYES CLASSIFIER  
-----  
  
Positive Sentiment = 0.15%  
Negative Sentiment = 0.89%  
Neutral Sentiment = 95.57%  
Irrelevant Sentiment = 3.40%
```

First 6 tweets

```
[{'text': "RT @daniballada: Each episode title of #AnneWithAnE is a quote from works of Anne's favorite authors. The seasons  
are dedicated to Charlott...",  
  'label': None},  
{ 'text': '@Granddaddy_Zach On Netflix ?', 'label': None},  
{ 'text': '@CarrieCnh12 Yes she was distracted by her boo #DragonPrince #SaturdayNightSciFi #Netflix',  
  'label': None},  
{ 'text': "Daybreak on Netflix is my type of show.\nIt's a shame it's only 10 episodes and I'm half way through.",  
  'label': None},  
{ 'text': 'RT @6ODWG: watching the blacklist on Netflix https://t.co/1nMcJ50SB8',  
  'label': None},  
{ 'text': "RT @MillennialOfMNL: i recommend watching carole & tuesday on netflix!!! it's about two girls wanting to make  
music in a futuristic world w...",
```

Findings – Sentiment Analysis Result

Naive Bayes Classifier

According to the preview slide the overall sentiment about the Netflix service can be considered **Neutral** (95.57%). A quickly evaluation about the first 6 tweets, also presented above, allows us to understand that, in general, the tweets are expressing a feeling about a movie, or series, or even just doing a comment about their personal mood or what they are watching at that time, what could justify the overall neutral evaluation about specifically Netflix services.

It is important to remember that the model was trained with a corpus specifically created for this purpose, identify the sentiment about the brand and its services. This point will be very important to understand the next extra analysis.

Findings – Sentiment Analysis Result

TextBlob Library

During the project development, I had the opportunity to discover and learn about a great library called TextBlob. TextBlob is an object-oriented NLP text-processing library that is built on the NLTK and pattern NLP libraries and simplifies many of their capabilities (Deitel, 2019).

This section has the object to provide a simplified approach for sentiment analysis but also clarify few particularities that may limit the TextBlob usability.

Findings – Sentiment Analysis Result

TextBlob Library

```
----- PERCENTAGE -----  
42 % Positives  
  
13 % Negatives  
  
44 % Neturals
```

The result of the Sentiment Analysis using the TextBlob library is presented in the left box, where, according it, 42% of the tweets are positive, 13% are negative and 44% neutral.

In fact, this evaluation can be considered correct, but it is important to clarify that **“libraries like TextBlob have pretrained machine learning models for performing sentiment analysis”** (Deital, 2019).

Another important point is that, by default, “TextBlob use a PatternAnalyzer, which uses the same sentiment analysis techniques as in the Pattern library” (Deitel, 2019). That is different to the Naive Bayes Classifier used previously.

Considering those points, it is possible infer that the result from TextBlob might correct about its perception but possible about a general evaluation, not specifically focused on the Netflix services user's sentiment.

Limitations

The main limitation encountered was that the model is limited to analysis sentiment about high tech companies (Amazon, Google, Apple, Windows, YouTube, Netflix, etc) user's satisfaction about those companies' products and service's quality. If used in another context may end up leading to misguided results.

The model can be also limited about the language, since it is designed to capture tweets only in English. An application to translate the tweets if it was not originally in English was tried without success, what can be a point for future improvements.

Conclusions

The Sentiment Analysis using the Naive Bayes Classifier provided a conclusion that 95.57% of the Netflix user's sentiments is neutral. According to this same analysis, 0.15% can be considered positive, 0.89% negative, and 3.40% irrelevant. This result is correlated to the fact that the most part of the tweets was not focused on the Netflix platform services but in general topics, that was identified by the model as neutral, in most of the cases, since the training data was specific related to the products and services.

An additional sentiment evaluation using the TextBlob library provided an entirely different result with 42% of the tweets positive, 13% negative, and 44% neutral. This result can be classified as a misleading result since the TextBlob library was trained with a generic corpus and use a different sentiment analysis technique.

Conclusions

The word frequency analysis provided valuable information and comprehension about the dataset. Based on these analysis, it is possible to infer that there is a large range of words on the dataset and that there are around 200 words concentrating the most frequent words on the entire dataset vocabulary.

Additionally, it is important to mention how helpful was the Word Cloud visualization, creating a quick and easy understand result about the dataset and its word varieties.

Acknowledgements

Thank you Nick Sanders, through Sanders Analytics, for providing an amazing corpus to be used as a training data to the sentiment analysis model.

A special thank you to my permanent Data Science study group composed by old friends who decided to go through this path together. Felipe Solares, Eduardo Passos and Felipe Brandão, that provided valuable feedbacks and insights to this project.

References

Deitel, H., Deitel, P., & Deitel, P. J. (2019). Python for Programmers, First Edition. Retrieved from: <https://learning.oreilly.com/library/view/python-for-programmers/9780135231364/ch11.html>

Total Training. (2017). Machine Learning - Twitter Sentiment Analysis in Python. Retrieved from: <https://learning.oreilly.com/videos/machine-learning/10000LCTWITTE>

Edx Data Science MicroMaster Python for Data Science class notes. (n.d.). Retrieved from: <https://courses.edx.org/courses/course-v1:UCSanDiegoX+DSE200x+3T2019/course/>

Sentiment Analysis using Twitter API and NLTK Naive Bayes Classifier

In this work will be analysed the sentiment of the user about a high tech company as Google, Amazon or Apple.

It is important to have this in mind since the training data that will be used have this purpose. So, the sentiment analyse can be not efficient if the searching term are not related with those high tech companies mentioned above.

Step 1 - Importing libraries

```
In [1]: #General Libraries

import pickle
import os
import tweepy #tweepy is a libraries that work similar as 'twitter' used during the course. I decided
               #to use that since I previously new about that and I feel more comfortable. It is important
               #to mention that the principles behind both libraries are the same

import pandas as pd
import nltk
from textblob import TextBlob #it is necessary install the textblob library if you don't have it yet!
                               #This is a very useful library that will be used during this work in two ways,
                               #as a translator for tweets in different languages and in a simplified
                               #sentiment analysis that will be also presented

nltk.download("punkt")

# if you don't have the wordcloud library installed, one option, if using Anaconda, you can install this
# package with the command below (type exactly as it is below on your Anaconda Prompt):
# conda install -c conda-forge wordcloud

# Visualization Libraries

from operator import itemgetter
from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline

# Preprocess Libraries

import re #module used to work with regular expressions
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\jaque\AppData\Roaming\nltk_data...
```

```
[nltk_data] Package punkt is already up-to-date!
```

Step 2 - Saving my credentials and creating the OAuth object to access tweets on Twitter

```
In [2]: ▶ # This strucre will save your credentials at the first time you exectute it, after that, erase your
# credentials to avoid anyone else to use that
if not os.path.exists('secret_twitter_credentials.pkl'):
    Twitter={}
    Twitter['Consumer Key'] = 'your_cosumer_key_here'
    Twitter['Consumer Secret'] = 'your_cosumer_secret_here'
    Twitter['Access Token'] = 'your_access_token_here'
    Twitter['Access Token Secret'] = 'your_access_token_secret_here'
    with open('secret_twitter_credentials.pkl','wb') as f:
        pickle.dump(Twitter, f)
else:
    Twitter=pickle.load(open('secret_twitter_credentials.pkl','rb'))
```

```
In [3]: ▶ # Authenticating and creating an API object

auth = tweepy.OAuthHandler(Twitter['Consumer Key'], Twitter['Consumer Secret'])
auth.set_access_token(Twitter['Access Token'], Twitter['Access Token Secret'])
api = tweepy.API(auth, wait_on_rate_limit=True,wait_on_rate_limit_notify=True)
```

Step 3 - Getting tweets as test data

```

In [4]: # Function that will access the Tweet and get tweets based on a search term, number of tweets, inicial
# date and final date. This function will return a list of dictionaries where the key is the tweet text
# and the values is a label, in this casa we are considering 'None' in the labor. This format is necessary
# for futures aplication

def get_tweets(word='google',number=100, since='2019-11-01', until='2019-11-28'):

    search_term = str(input('Hi! What are you looking for today? - '))
    num_of_terms = int(input('How many tweets do you want? - '))
    since = input('From what date? ex.: 2019-11-01 - ')
    until = input('Until what date? ex.: 2019-11-28 - ')

    data = []

    tweets = tweepy.Cursor(api.search,
                            #since = f'{since},
                            #until = f'{until},
                            q=search_term,
                            lang='en').items(num_of_terms)

    for tweet in tweets:
        try:
            data.append(tweet.extended_tweet.full_text)
        except:
            data.append(tweet.text)

    new_data = set(data) #converting in a set to remove the duplicates

    new_data_dic = []

    for i in new_data:
        new_data_dic.append({'text':i, 'label':None}) #this struture as a dictionary is necessary for
                                                    #further purposes

    print()

    #Will print a message to the user
    print('Great Job! We got '+str(len(data))+ ' tweets with the term '+word+'!!!')

    return [i for i in new_data_dic]

```



```
In [5]: ▶ # creating a tweets objecting and printing the first 4 tweets
tweets = get_tweets()
tweets[:3]

# IMPORTANT!!!

# The Twitter API have limitations on the total of tweets and how many requests can be made.
# A recommendation is do not request more than 3000 tweets in an interval of 15 min to avoid long waiting
# time.
```

Hi! What are you looking for today? - netflix
How many tweets do you want? - 2000
From what date? ex.: 2019-11-01 - 2019-11-29
Until what date? ex.: 2019-11-28 - 2019-11-30

Great Job! We got 2000 tweets with the term google!!

```
Out[5]: [{'text': "RT @daniballada: Each episode title of #AnneWithAnE is a quote from works of Anne's favorite authors. The seasons are dedicated to Charlott...",
          'label': None},
         {'text': '@Granddaddy_Zach On Netflix ?', 'label': None},
         {'text': '@CarrieCnh12 Yes she was distracted by her boo #DragonPrince #SaturdayNightSciFi #Netflix',
          'label': None}]
```

```
In [6]: ▶ # Checking the final length of the request we can see the total is not the 1000 tweets requested. The
# reason for that is that the tweet api return duplicates and the above function do the job of removing
# those.

len(tweets)
```

```
Out[6]: 1354
```

Step 4 - Getting training data

We'll use the Niek Sander's tweets corpus with ~5000 classified tweets labelled as positive, negative, neutral or irrelevant. Those tweets are related with tech companies like, Apple, Google, Twitter, Youtube, Microsoft and others. Said that, the model that will be created is recommended to a search term related with tech companies. The performance lower if the sentiment analysis is related with another kind of topic.

Getting Nick Sander's tweets corpus!

- access the link https://github.com/zfz/twitter_corpus (https://github.com/zfz/twitter_corpus);
- download the file;
- we will use the **full-corpus.csv** file. This file already contain the tweets texts and it respective labels of *positive*, *negative*, *neutral* or *irrelevant*.

After download the file it is recommend that you explore it using Pandas for a better comprehension

```
In [7]: ▶ # Function to read the nick_sanders_corpus csv file and return a list containing the text and the label
# for each tweet

# reading the nick_sanders_corpus
df = pd.read_csv('nick_sanders_corpus/full_corpus.csv')

#function
def trainingData():
    trainingData = [{'text':row[4], 'label':row[1]} for index,row in df.iterrows()]
    return trainingData

# The function return a list of dict containing the key (tweets text) and the values (the labels)

df.head()
```

Out[7]:

	Topic	Sentiment	TweetId	TweetDate	TweetText
0	apple	positive	126415614616154112	Tue Oct 18 21:53:25 +0000 2011	Now all @Apple has to do is get swype on the i...
1	apple	positive	126404574230740992	Tue Oct 18 21:09:33 +0000 2011	@Apple will be adding more carrier support to ...
2	apple	positive	126402758403305474	Tue Oct 18 21:02:20 +0000 2011	Hilarious @youtube video - guy does a duet wit...
3	apple	positive	126397179614068736	Tue Oct 18 20:40:10 +0000 2011	@RIM you made it too easy for me to switch to ...
4	apple	positive	126395626979196928	Tue Oct 18 20:34:00 +0000 2011	I just realized that the reason I got into twi...

```
In [8]: ▶ training_Data = trainingData()  
training_Data[:3]
```

```
Out[8]: [{ 'text': 'Now all @Apple has to do is get swype on the iphone and it will be crack. Iphone that is',  
          'label': 'positive'},  
        { 'text': '@Apple will be adding more carrier support to the iPhone 4S (just announced)',  
          'label': 'positive'},  
        { 'text': "Hilarious @youtube video - guy does a duet with @apple 's Siri. Pretty much sums up the love aff  
air! http://t.co/8ExbnQjY", (http://t.co/8ExbnQjY),  
          'label': 'positive'}]
```

Step 5 - Preprocessing tweets from test and training data

The preprocess step will use few python tools to work with strings as detailed bellow:

USING .LOWER() STRING FUNCTION

1 - Convert to lower case

USING REGULAR EXPRESSIONS

2 - Replace links with the string 'url'

3 - Replace @ ... with 'at_user'

4 - Replace #word with the word itself

5 - Remove emoticons using ASCII encode and decode

USING NLTK

7 - Tokenize the tweet into words (a list of words)

8 - Remove stopwords (including url and user, RT and '...')

In [9]:  *# Creating a class to preprocess the test and training tweets*

```
class Preprocess:
    def __init__(self):

        lst = ['AT_USER', 'URL', 'rt', '...', "'s", "n't", "`", "'"]
        self._stopwords = set(stopwords.words('english')+list(punctuation)+lst)

    def processTweets(self, tweets):
        #tweets is a list of dict with Keys, 'text' and 'label'
        processedTweets = []
        #this list will be a list of tuple. Each tuple is a tweet which is a list of words and its label

        for tweet in tweets:
            processedTweets.append((self.cleanTweet(tweet['text']),tweet['label']))
            #it will apply the cleanTweet function only to the tweets text

        return processedTweets

    def processTweets_words(self, tweets):

        processedTweets = []
        #this list will be a list of each word in all tweets

        for tweet in tweets:
            processedTweets.append(self.cleanTweet(tweet['text']))
            #it will apply the cleanTweet function only to the tweets text

        return processedTweets

    def cleanTweet(self,tweet):
        #1 - Convert to lower case
        tweet = tweet.lower()

        #2 - Replace links with word 'URL'
        tweet = re.sub('((www\.[^\s]+)|(https?:\/\/[^\s]+))','URL',tweet)

        #3 - Replace @username with 'AT_USER'
        tweet = re.sub('@[^\s]+','AT_USER',tweet)

        #4 - Repalce #word (hashtag) with just the word, witouth the '#' simble
```

```

tweet=re.sub(r'#([\^s]+)',r'\1',tweet)

#5 - Remove emoticons
tweet = tweet.encode('ascii', 'ignore').decode('ascii')

#6 - Tokenizing the tweets
tweet = word_tokenize(tweet)

#7 - Removing stopwords
return [word for word in tweet if word not in self._stopwords]

```

```

In [10]: ► # Instantiating a Preprocess Class called 'tprocessor'
tprocessor = Preprocess()

# Creating an object that will contain the result of the TRAINING DATA after been preprocessed by the
# 'tprocessor' class
cleanedtrainingData = tprocessor.processTweets(training_Data)

# Creating an object that will contain the result of the TEST DATA after been preprocessed by the
# 'tprocessor' class
cleanedtestData = tprocessor.processTweets(tweets)

```

```

In [11]: ► print(cleanedtrainingData[:10])

[(['get', 'swype', 'iphone', 'crack', 'iphone'], 'positive'), (['adding', 'carrier', 'support', 'iphone',
'4s', 'announced'], 'positive'), (['hilarious', 'video', 'guy', 'duet', 'siri', 'pretty', 'much', 'sums',
'love', 'affair'], 'positive'), (['made', 'easy', 'switch', 'iphone', 'see', 'ya'], 'positive'), (['realize
d', 'reason', 'got', 'twitter', 'ios5', 'thanks'], 'positive'), (['"m", 'current', 'user', 'little', 'bit',
'disappointed', 'move'], 'positive'), (['16', 'strangest', 'things', 'siri', 'said', 'far', 'sooo', 'glad',
'gave', 'siri', 'sense', 'humor', 'via'], 'positive'), (['great', 'close', 'personal', 'event', 'tonight',
'regent', 'st', 'store'], 'positive'), (['companies', 'experience', 'best', 'customer', 'service', 'asid
e'], 'positive'), (['apply', 'job', 'hope', 'call', 'lol'], 'positive')]

```

In [12]: `print(cleanedtestData[:10])`

```
[(['episode', 'title', 'annewithane', 'quote', 'works', 'anne', 'favorite', 'authors', 'seasons', 'dedicate
d', 'charlott'], None), (['netflix'], None), (['yes', 'distracted', 'boo', 'dragonprince', 'saturdaynightsc
ifi', 'netflix'], None), (['daybreak', 'netflix', 'type', 'show', 'shame', '10', 'episodes', "'m", 'half',
'way'], None), (['watching', 'blacklist', 'netflix'], None), (['recommend', 'watching', 'carole', 'amp', 't
uesday', 'netflix', 'two', 'girls', 'wanting', 'make', 'music', 'futuristic', 'world', 'w'], None), (['shar
e', 'brings', 'joy', 'chance', 'win', '10,000', 'send', 'us', 'photo', 'could', 'also', 'add', 'daily', 'pr
ize'], None), (['watched', 'klaus', 'netflix', 'fantastic', 'super', 'cute', 'christmassy', 'bad', 'start',
'end', 'november'], None), (['*sweats', 'nervously', 'thinking', 'employees', 'probably', 'know', 'embarras
sing', 'amount', 'times', 'ive', 'watched'], None), (['yay', 'seunoo', 'netflix'], None)]
```

Step 5.1 - Word Frequency Visualizations

- Word Frequency using a bar plot
- Word Frequency using a Word Cloud

In [13]: `# the result of the 'processTweets_words' is a list fo lists, so it will be necessary unify all words in
only one list
words = tprocessor.processTweets_words(tweets)`

```
unif_words =[]
for i in words:
    for j in i:
        unif_words.append(j)

unif_words[:5]
```

Out[13]: ['episode', 'title', 'annewithane', 'quote', 'works']

In [14]: `# word frequency using 'Counter' class from the 'collection' package

from collections import Counter

word_counter = Counter(unif_words)`

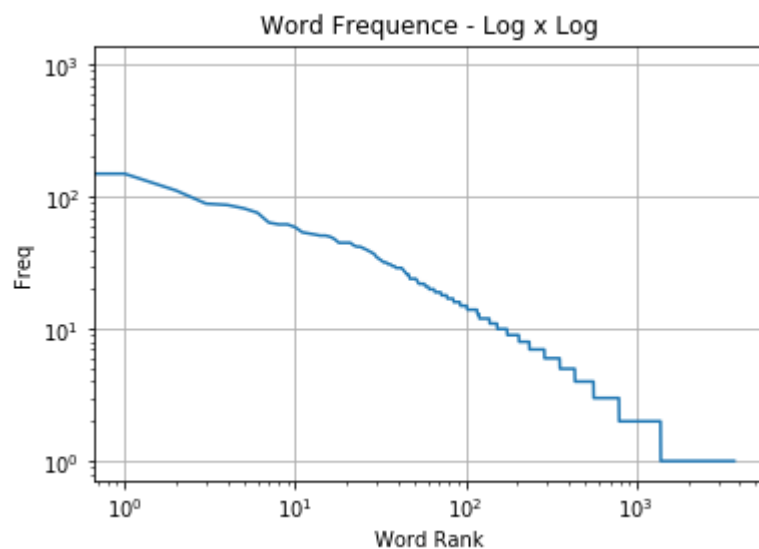
In [15]: `most_common_words = word_counter.most_common()[:15]`

In [16]: `print(most_common_words)`

```
[('netflix', 999), ('watch', 150), ('watching', 112), ('movie', 89), ('like', 87), ('season', 82), ('im', 76), ('show', 64), ('amp', 62), ('good', 62), ('series', 59), ('4', 54), ('watched', 53), ('one', 52), ('movies', 51)]
```

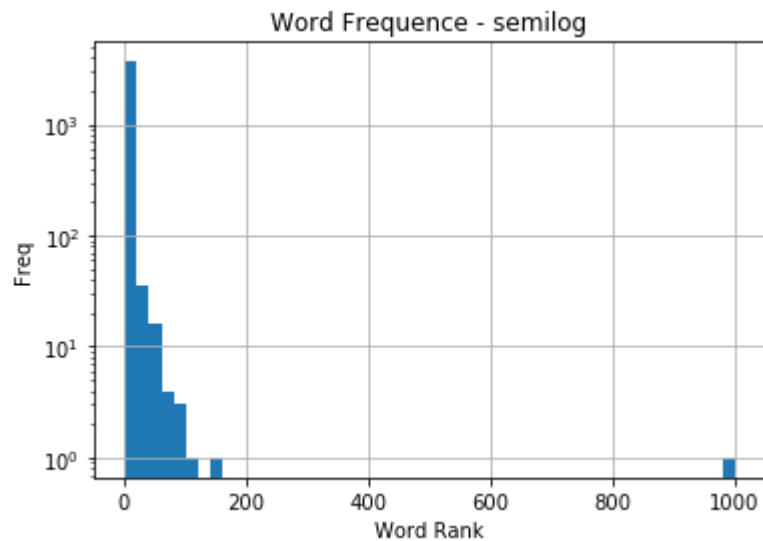
In [17]: `# Word distribution present a large vocabulary`
`sorted_word_counts = sorted(list(word_counter.values()), reverse=True)`

```
plt.loglog(sorted_word_counts)
plt.ylabel("Freq")
plt.xlabel("Word Rank")
plt.title('Word Frequency - Log x Log')
plt.grid()
plt.show()
```



In [18]: `# Word distribution that allowed detect the count of words in a specific range`

```
plt.hist(sorted_word_counts, bins=50, log=True);  
plt.ylabel("Freq")  
plt.xlabel("Word Rank")  
plt.title('Word Frequency - semilog')  
plt.grid()  
plt.show()
```



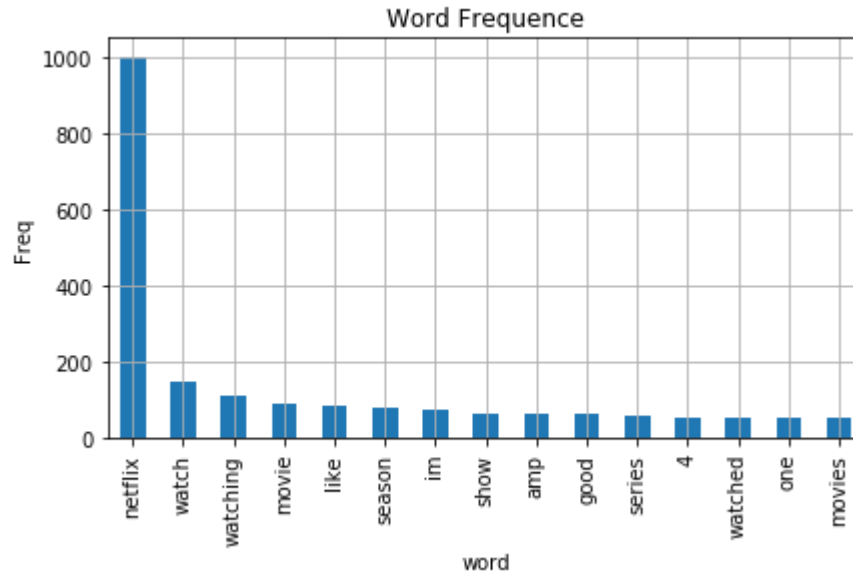

```
In [19]: # Converting the dictionary to a pandas DataFrame  
df = pd.DataFrame(most_common_words, columns=['word', 'freq'])  
  
# sorting the words based on their frequency  
df = df.sort_values(by=['freq'], ascending=False)  
  
# getting the top 20 more frequent words  
top_20 = df.head(20)  
top_20
```

Out[19]:

	word	freq
0	netflix	999
1	watch	150
2	watching	112
3	movie	89
4	like	87
5	season	82
6	im	76
7	show	64
8	amp	62
9	good	62
10	series	59
11	4	54
12	watched	53
13	one	52
14	movies	51

In [20]: `# Bar plotting from the top 20 more frequent words`

```
axes = top_20.plot.bar(x='word',y='freq',legend=False)
plt.gcf().tight_layout()
plt.ylabel('Freq')
plt.grid()
plt.title('Word Frequency')
plt.show()
```



```
In [21]: dic = {}
for item in unif_words:
    dic[item] = dic.get(item, 0) + 1
```

In [22]: ▶ *# Bulding a World Cloud*

```
wordcloud = WordCloud(width=1600, height=900,  
                        prefer_horizontal=0.5,  
                        min_font_size=10,  
                        colormap='prism')  
  
wordcloud = wordcloud.fit_words(dic)  
  
wordcloud = wordcloud.to_file('TrendingTwitter.png')
```

#IMPORTANT!!!

*#The result will create a png file in the folder where the jupyter notebook are running. In order to see the
#result you will need to go check this file*



The NLTK model that will be used is known as NAVIE BAYES CLASSIFICATION

1.1 - Build a vocabulary (list of all unique words in all the tweets in the training data);

2.2 - Represent each tweet with the presence or absence of these words;

Ex.: Given a *vocabulary: {'the','worst','thing','in','the','world'} and a *tweet: {'the','worst','thing'}

This tweet will be represented as a "Feature Vector" (1,1,1,0,0,0) -> indicating that the

first three words from the vocabulary are in the tweet, and there are other 3 words in the vocabulary that are not part of the mentioned tweet.

2.3 - Use NLTK's built in Naive Bayes Classifier to train a Classifier



In [23]: *#Defining the function to create a word Vocabulary or bag_of_words*

```
def wordVocab(cleanedtrainingData):  
    training_features = []  
    for (words, sentiment) in cleanedtrainingData:  
        training_features.extend(words)  
    return list(set(training_features))
```

In [24]: *#Creating the word_features_vocab object*

```
word_features_vocab = wordVocab(cleanedtrainingData)
```

```
In [25]: ▶ # The NLTK library have a function called apply_features that takes a user-defined function to extract
# featrues from training data. In this case the function will be called extract_features_func that will
# take each tweet in the in the training data and repersent it with the presence or absence of a word in
# the vocabulary, as previously explained in the item 2.2 above.

def extract_features_func(tweet):
    tweet_words = set(tweet)
    features = {}
    for word in word_features_vocab:
        features[f'contains {word}'] = (word in tweet_words)
        # this step will creat a dictionary with keys like 'contains word1' and 'contains word2', and values
        # as True or False. The statement that create the True or False return is the '(word in tweet_words)'

    return features
```

```
In [26]: ▶ # Creating the traning_features object

# apply_features will take the extract_features_func defined above, and apply it to each element of
# cleanedtrainingData. it automatically recognize that each of these elements are tuples where the
# first element is the text and the second is the label. The apply_features apply the extract_features_func
# only on the text element.

trainingFeature = nltk.classify.apply_features(extract_features_func,cleanedtrainingData)
```

```
In [27]: ▶ # Creating the classifier objected, trained using the training data features

NBclassifier = nltk.NaiveBayesClassifier.train(trainingFeature)
```

Step 7 - Run the Classifier on the 2000 downloaded tweets

```
In [28]: ▶ sentiment_classifier = [NBclassifier.classify(extract_features_func(tweet[0])) for tweet in cleanedtestData]
```

```

In [29]: print('-'*40)
print('{:^80}'.format('SENTIMENT RESULTS USING NAIVE BAYES CLASSIFIER'))
print('-'*40)
print('')

if sentiment_classifier.count('positive'):
    print('Positive Sentiment = {:.2f}'.format(100*sentiment_classifier.count('positive')/len(sentiment_classifier)))

if sentiment_classifier.count('negative'):
    print('Negative Sentiment = {:.2f}'.format(100*sentiment_classifier.count('negative')/len(sentiment_classifier)))

if sentiment_classifier.count('neutral'):
    print('Neutral Sentiment = {:.2f}'.format(100*sentiment_classifier.count('neutral')/len(sentiment_classifier)))

if sentiment_classifier.count('irrelevant') or sentiment_classifier.count('irrelevant') == 0:
    print('Irrelevant Sentiment = {:.2f}'.format(100*sentiment_classifier.count('irrelevant')/len(sentiment_classifier)))

print('')
print('-'*40)
print('{:^80}'.format('First 10 tweets'))
print('-'*40)
print('')

tweets[0:10]

```

```

=====
                        SENTIMENT RESULTS USING NAIVE BAYES CLASSIFIER
=====

Positive Sentiment = 0.15%
Negative Sentiment = 0.89%
Neutral Sentiment = 95.57%
Irrelevant Sentiment = 3.40%

=====
                        First 10 tweets
=====

```

```

Out[29]: [{'text': "RT @daniballada: Each episode title of #AnneWithAnE is a quote from works of Anne's favorite authors. The seasons are dedicated to Charlott...",
  'label': None},
  {'text': '@Granddaddy_Zach On Netflix ?', 'label': None},
  {'text': '@CarrieCnh12 Yes she was distracted by her boo #DragonPrince #SaturdayNightSciFi #Netflix',
  'label': None},
  {'text': "Daybreak on Netflix is my type of show.\nIt's a shame it's only 10 episodes and I'm half way through.",
  'label': None},
  {'text': 'RT @60DWG: watching the blacklist on Netflix https://t.co/1nMcJ50SB8', (https://t.co/1nMcJ50SB8',)
  'label': None},
  {'text': "RT @MillennialOfMNL: i recommend watching carole & tuesday on netflix!!! it's about two girls wanting to make music in a futuristic world w...",
  'label': None},
  {'text': 'Share what brings you joy for a chance to win $10,000! Send us your photo, and you could also add a daily prize to... https://t.co/x0xa4CCjfr', (https://t.co/x0xa4CCjfr',)
  'label': None},
  {'text': 'We just watched Klaus on Netflix. It was fantastic, super cute, and very Christmassy. Not a bad start for the end of November!',
  'label': None},
  {'text': '*sweats nervously * Thinking about @netflix employees probably know the embarrassing amount of times I've watched @disenchantment',
  'label': None},
  {'text': 'RT @duotheizm: yay seunoo on netflix https://t.co/ZXEzwFkPM1', (https://t.co/ZXEzwFkPM1',)
  'label': None}]

```

A simplified approach using TextBlob library

The object of this section is to present a simple and very interest tool that I could discover on my journey to complete this project. I will also perform the same kind of sentiment analysis in a much more simplefied version using a powerful library called TextBlob.

TextBlob is an object-oriented NLP text-processing library that is built on the NLTK and pattern NLP libraries and simplifies many of their capabilities.

During my exploration to perform this work I had the chance to learn about this library and I strongly recomend that you dedicate a time to read and understand better the main tasks that TextBlob can perform.

In [30]:  *# Libraries*


```
import pickle
import os
import tweepy
from textblob import TextBlob
```

In [31]:  *#Saving/Loading the credations to access the tweet API*

```
# This strucre will save your credentials at the first time you exectute it, after that, erase your
# credentials to avoid anyone else to use that
if not os.path.exists('secret_twitter_credentials.pkl'):
    Twitter={}
    Twitter['Consumer Key'] = 'your_cosumer_key_here'
    Twitter['Consumer Secret'] = 'your_cosumer_secret_here'
    Twitter['Access Token'] = 'your_access_token_here'
    Twitter['Access Token Secret'] = 'your_access_token_secret_here'
    with open('secret_twitter_credentials.pkl','wb') as f:
        pickle.dump(Twitter, f)
else:
    Twitter=pickle.load(open('secret_twitter_credentials.pkl','rb'))

# Authenticating and creating an API object

auth = tweepy.OAuthHandler(Twitter['Consumer Key'], Twitter['Consumer Secret'])
auth.set_access_token(Twitter['Access Token'], Twitter['Access Token Secret'])
api = tweepy.API(auth, wait_on_rate_limit=True,wait_on_rate_limit_notify=True)
```

In [32]:  *# Function that will access the Tweet and get tweets based on a search term, number of tweets, inicial # date and final date. This function will return a list of dictionaries where the key is the tweet text # and the values is a label, in this casa we are considering 'None' in the Labor. This format is necessary # for futures aplication*

```
def get_tweets2(word='google',number=5, since='2019-11-01', until='2019-11-28'):  
  
    search_term = str(input('Hi! What are you looking for today? - '))  
    num_of_terms = int(input('How many tweets do you want? - '))  
    since = input('From what date? ex.: 2019-11-01 - ')  
    until = input('Until what date? ex.: 2019-11-28 - ')  
  
    tweets = tweepy.Cursor(api.search,  
                           #since = f'{since}',  
                           #until = f'{until}',  
                           q=search_term,  
                           lang='en').items(num_of_terms)  
  
    positive = 0  
    negative = 0  
    neutral = 0  
  
    for tweet in tweets:  
        try:  
            analysis = TextBlob(tweet.extended_tweet.full_text)  
        except:  
            analysis = TextBlob(tweet.text)  
  
        if analysis.sentiment[0] > 0.00:  
            positive += 1  
        elif analysis.sentiment[0] < 0.00:  
            negative += 1  
        else:  
            neutral += 1  
  
    print()  
    print('-=' * 3, 'TOTALS', '-=' * 3)  
    print('Total Positives:', positive)  
    print()  
    print('Total Negatives:', negative)
```

```

print()
print('Total Neutrals:', neutral)
print()
print()
print('-=' * 3, 'PERCENTAGE', '-=' * 3)
print(int((positive / num_of_terms) * 100), '% Positives')
print()
print(int((negative / num_of_terms) * 100), '% Negatives')
print()
print(int((neutral / num_of_terms) * 100), '% Neturals')
print()

return positive, negative, neutral

```

In [33]: `tweets2 = get_tweets2()`

```

Hi! What are you looking for today? - netflix
How many tweets do you want? - 2000
From what date? ex.: 2019-11-01 - 2019-11-29
Until what date? ex.: 2019-11-28 - 2019-11-30

```

```

----- TOTALS -----
Total Positives: 847

```

```

Total Negatives: 269

```

```

Total Neutrals: 884

```

```

----- PERCENTAGE -----
42 % Positives

```

```

13 % Negatives

```

```

44 % Neturals

```

Adjusting the TextBlob to perform the sentiment analysis using a Navie Bayes Analyser

By default, a TextBlob and the Sentences and Words you get from it determine sentiment using a PatternAnalyzer, which uses the same sentiment analysis techniques as in the Pattern library. The TextBlob library also comes with a NaiveBayesAnalyzer9 (module textblob.sentiments), **which was trained on a database of movie reviews**.

Considering that we can not re-train the algorithm in the TextBlob, it is important to understand that the performance compromised since the train data was not related with the topic of the search term that will be considered on this analysis, a term related to a tech company for the sentiment analysis

```
In [34]: ▶ from textblob.sentiments import NaiveBayesAnalyzer

# Recommendation!!!

# The sentiment analysis using TextBlob NaiveBayesAnalyzer presented a low performance in terms of speed,
# basically it takes long time to process the analysis. Request a max of 20 tweets in order to be able to
# quickly have a return and evaluate the result. If you have more time and a powerful machine you maybe could
# try a large tweet request.
```

```
In [35]: ▶ def get_tweets3(word='google',number=5, since='2019-11-01', until='2019-11-28'):  
  
    search_term = str(input('Hi! What are you looking for today? - '))  
    num_of_terms = int(input('How many tweets do you want? - '))  
    since = input('From what date? ex.: 2019-11-01 - ')  
    until = input('Until what date? ex.: 2019-11-28 - ')  
  
    tweets = tweepy.Cursor(api.search,  
                            #since = f'{since}',  
                            #until = f'{until}',  
                            q=search_term,  
                            lang='en').items(num_of_terms)  
  
    positive = 0  
    negative = 0  
  
    for tweet in tweets:  
  
        blob = TextBlob(tweet.text, analyzer=NaiveBayesAnalyzer())  
  
        if blob.sentiment.p_pos > blob.sentiment.p_neg:  
            positive += 1  
        else:  
            blob.sentiment.p_pos < blob.sentiment.p_neg  
            negative += 1  
  
    print()  
    print('-=' * 3, 'TOTALS', '-=' * 3)  
    print('Total Positives:', positive)  
    print()  
    print('Total Negatives:', negative)  
    print()  
    print()  
    print('-=' * 3, 'PERCENTAGE', '-=' * 3)  
    print(int((positive / num_of_terms) * 100), '% Positives')  
    print()  
    print(int((negative / num_of_terms) * 100), '% Negatives')  
    print()  
  
    return positive, negative
```

In [36]: `tweets3 = get_tweets3()`

```
Hi! What are you looking for today? - netflix
How many tweets do you want? - 20
From what date? ex.: 2019-11-01 - 2019-11-29
Until what date? ex.: 2019-11-28 - 2019-11-30
```

```
----- TOTALS -----
```

```
Total Positives: 9
```

```
Total Negatives: 11
```

```
----- PERCENTAGE -----
```

```
45 % Positives
```

```
55 % Negatives
```

In []: `tweets3`