Google Colab Link: https://drive.google.com/file/d/1R5R_L296vQ5yB8I4-YBkf8Qf_5Vw9xHR/view?usp=sharing

Lab Exercise 7 : Vectorize Scraped Data

In this lab exercise, we will vectorize data scraped from web.

1. Scrape quotes from here.
    a. Extract quotes and authors(optional quotes type) You may limit to scraping only one type of quote from a page.
    b. Save the scraped data to dataframe and save into a .csv file.

In [1]:
```
from bs4 import BeautifulSoup
import re
import pandas as pd
import requests
```

In [2]:
```
url = "https://www.brainyquote.com"
a_c = requests.get(url)
```

In [3]:
```
#print(a_c.content)
bs = BeautifulSoup(a_c.content)
#print(bs.prettify())
```

In [4]:
```
quote_cls = bs.find_all('div', attrs={"class":"bqLn"})
```

In [5]:
```
quote_cls = [q.find_all('a', href=True)[0]['href'] for q in quote_cls] #finding all possible topic links to browse
quote_cls = [url+q for q in quote_cls]
quote_cls[:10]
```

Out[5]:
```
['https://www.brainyquote.com/topics/motivational-quotes',
 'https://www.brainyquote.com/topics/attitude-quotes',
 'https://www.brainyquote.com/topics/beauty-quotes',
```

```
'https://www.brainyquote.com/topics/life-quotes',
'https://www.brainyquote.com/topics/alone-quotes',
'https://www.brainyquote.com/topics/positive-quotes',
'https://www.brainyquote.com/topics/smile-quotes',
'https://www.brainyquote.com/topics/success-quotes',
'https://www.brainyquote.com/topics/inspirational-quotes',
'https://www.brainyquote.com/topics/education-quotes']
```

Extracting Quote, Author and Topic from all Type of Quotes in website

In [6]:
```
def extract_quote_author(topic_url, topic):
    topic_quote_bs = BeautifulSoup(requests.get(topic_url).content)
    topic_quote_cls_list = topic_quote_bs.find_all('div', attrs={"class":"grid-item qb clearfix bqQt"}) #extracting quote and author text
    topic_quote_author_list = [q.text.strip().split("\n\n\n\n") for q in topic_quote_cls_list] #creating list of quote and author
    topic_quote_author_list = [q + [topic] for q in topic_quote_author_list] #adding topic to each quote and author
    return topic_quote_author_list
```

In [7]:
```
all_type_quote_list = []
for topic_url in quote_cls:
    topic = topic_url.split("/")[-1].replace("-quotes","")
    temp_list = extract_quote_author(topic_url, topic)
    all_type_quote_list.extend(temp_list)

all_type_quote_list[:5]
```

Out[7]:
```
[["Good, better, best. Never let it rest. 'Til your good is better and your better is best.",
  'St. Jerome',
  'motivational'],
 ["It always seems impossible until it's done.",
  'Nelson Mandela',
  'motivational'],
 ["If you're going through hell, keep going.",
  'Winston Churchill',
  'motivational'],
 ['Life is 10% what happens to you and 90% how you react to it.',
  'Charles R. Swindoll',
  'motivational'],
 ['When something is important enough, you do it even if the odds are not in your favor.',
```

```
 'Elon Musk',
 'motivational']]
```

In [8]:
```
df_all_quotes = pd.DataFrame(all_type_quote_list, columns = ['quote', 'author','type'])
```

In [9]:
```
df_all_quotes.head()
```

Out[9]:

	quote	author	type
0	Good, better, best. Never let it rest. 'Til yo...	St. Jerome	motivational
1	It always seems impossible until it's done.	Nelson Mandela	motivational
2	If you're going through hell, keep going.	Winston Churchill	motivational
3	Life is 10% what happens to you and 90% how yo...	Charles R. Swindoll	motivational
4	When something is important enough, you do it ...	Elon Musk	motivational

In [10]:
```
df_all_quotes.sample(20)
```

Out[10]:

	quote	author	type
5591	If you have only one smile in you give it to t...	Maya Angelou	maya-angelou
737	Nationalism is an infantile disease. It is the...	Albert Einstein	albert-einstein
3444	My religion is very simple. My religion is kin...	Dalai Lama	dalai-lama
32	By failing to prepare, you are preparing to fail.	Benjamin Franklin	motivational
7203	One is very crazy when in love.	Sigmund Freud	sigmund-freud
3914	Any fool can make a rule, and any fool will mi...	Henry David Thoreau	henry-david-thoreau
382	The happiness of life is made up of minute fra...	Samuel Taylor Coleridge	smile
1666	Our society, including the liberals, must unde...	Vladimir Putin	vladimir-putin
8254	Good night, good night! Parting is such sweet ...	William Shakespeare	william-shakespeare
1684	Russia needs a strong state power and must hav...	Vladimir Putin	vladimir-putin
865	I think life on Earth must be about more than ...	Elon Musk	elon-musk
3912	Live the life you've dreamed.	Henry David Thoreau	henry-david-thoreau
3770	Never give up on what you really want to do. T...	H. Jackson Brown, Jr.	h-jackson-brown-jr

	quote	author	type
8033	I don't look to jump over 7-foot bars: I look ...	Warren Buffett	warren-buffett
4262	It is only too easy to make suggestions and la...	Jawaharlal Nehru	jawaharlal-nehru
1147	Beneficience removes evils, introduces the pra...	Dayananda Saraswati	dayananda-saraswati
2017	They tell us that suicide is the greatest piec...	Arthur Schopenhauer	arthur-schopenhauer
798	Whatever words we utter should be chosen with ...	Buddha	buddha
6562	For everything you have missed, you have gaine...	Ralph Waldo Emerson	ralph-waldo-emerson
6371	For a man to conquer himself is the first and ...	Plato	plato

In [11]:
```
df_all_quotes.to_csv("All_Quotes_Library1.csv", index = False)
```

2. Use scikit-learn and createfollowing text represenationson the dataset:

a. BoWs
b. ngram vectorization
c. tf-idf vectorization

In [12]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [13]:
```
BoW_vectorizer = CountVectorizer(stop_words= 'english')
```

In [14]:
```
list_all_quotes = list(df_all_quotes['quote'])
list_all_quotes[:2]
```

Out[14]:
```
["Good, better, best. Never let it rest. 'Til your good is better and your better is best.",
 "It always seems impossible until it's done."]
```

In [15]:
```
# fit() function will learn a vocabulary from one or more documents.
BoW_vectorizer.fit(list_all_quotes)
```

Out[15]:
```
CountVectorizer(stop_words='english')
```

In [16]:
```
#BoW_vectorizer.vocabulary_
```

In [17]:
```
voc_all_quotes = BoW_vectorizer.get_feature_names()
voc_all_quotes[:10]
```

Out[17]:
```
['000', '10', '100', '101', '10th', '11', '13', '14', '15', '150']
```

In [18]:
```
# transform() function will transform one or more documents to vectors
print(list_all_quotes[:2])
vector = BoW_vectorizer.transform(list_all_quotes)
#print(vector)
print(type(vector))
print(vector.toarray())
```

```
["Good, better, best. Never let it rest. 'Til your good is better and your better is best.", "It always seems impossible until it's done."]
<class 'scipy.sparse.csr.csr_matrix'>
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [19]:
```
df_BoW_vectors = pd.DataFrame(vector.toarray(), columns = voc_all_quotes)
```

Bag of Words Vectorized

In [20]:
```
df_BoW_vectors
```

Out[20]:

	000	10	100	101	10th	11	13	14	15	150	...	zealous	zero	zest	zeus	zlaritable	zlatan	zohan	zone	zoroastrian	zoroastrians
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

	000	10	100	101	10th	11	13	14	15	150	...	zealous	zero	zest	zeus	zlaritable	zlatan	zohan	zone	zoroastrian	zoroastrians
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8516	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8517	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8518	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8519	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8520	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

8521 rows × 10639 columns

Ngrams Vectorized

In [21]:
```
NgramVectorizer = CountVectorizer(stop_words = 'english', ngram_range = (1,3), min_df = 2)
```

In [22]:
```
NgramVectorizer.fit(list_all_quotes)
```

Out[22]:
```
CountVectorizer(min_df=2, ngram_range=(1, 3), stop_words='english')
```

In [23]:
```
Ngram_vectors = NgramVectorizer.get_feature_names()
print(Ngram_vectors[:20])
```

```
['000', '10', '10 000', '10 happens', '10 happens 90', '10 times', '10 times better', '10 years', '100', '10th', '11', '15', '16', '17', '1
8', '1876', '1876 actual', '1876 actual conversation', '1894', '19']
```

In [24]:
```
vector = NgramVectorizer.transform(list_all_quotes)
print(vector.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [25]:
```
df_ngram = pd.DataFrame(data = vector.toarray(), columns = Ngram_vectors)
```

In [26]:
```
df_ngram
```

Out[26]:

	000	10	10 000	10 happens	10 happens 90	10 times	10 times better	10 years	100	10th	...	youth going rest	youtube	zero	zeus	zeus fall	zeus fall luckily	zeus oversees	zeus oversees directs	zlatan	zone
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	1	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8516	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8517	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8518	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8519	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8520	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

8521 rows × 17914 columns

TfidfVectorizer

In [27]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [28]:
```
tf_idf_vectorizer = TfidfVectorizer(stop_words= 'english') #remove stop words
vectorized_text=tf_idf_vectorizer.fit_transform(list_all_quotes)
```

In [29]:
```
tf_idf_vectors = tf_idf_vectorizer.get_feature_names()
tf_idf_vectors[:10]
```

Out[29]:
```
['000', '10', '100', '101', '10th', '11', '13', '14', '15', '150']
```

In [30]:
```
tf_idf_vectorizer.idf_[:10]
```

Out[30]:
```
array([8.25864686, 7.27781761, 7.65251106, 9.35725915, 8.95179405,
       8.44096842, 9.35725915, 9.35725915, 8.66411197, 9.35725915])
```

In [31]:
```
df_tf_idf_vect = pd.DataFrame(data = vectorized_text.toarray(), columns = tf_idf_vectors)
df_tf_idf_vect
```

Out[31]:

	000	10	100	101	10th	11	13	14	15	150	...	zealous	zero	zest	zeus	zlaritable	zlatan	zohan	zone	zoroastrian	zoroastrians
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.481529	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8516	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8517	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8518	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8519	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8520	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8521 rows × 10639 columns

In [32]:
```
#checking new quote against all quotes vector
newDoc = ['life is not a problem to be solved, but a reality to be experienced']
newDocVec = tf_idf_vectorizer.transform(newDoc)
df_new_doc = pd.DataFrame(data = newDocVec.toarray(), columns = tf_idf_vectors)
df_new_doc
```

Out[32]:

	000	10	100	101	10th	11	13	14	15	150	...	zealous	zero	zest	zeus	zlaritable	zlatan	zohan	zone	zoroastrian	zoroastrians

	000	10	100	101	10th	11	13	14	15	150	...	zealous	zero	zest	zeus	zlaritable	zlatan	zohan	zone	zoroastrian	zoroastrians
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1 rows × 10639 columns

In [33]:
```
str_all_quotes = " ".join(list_all_quotes).split(" ")
print(len(set(str_all_quotes)))
```

```
18947
```

In [34]:
```
print(df_new_doc.apply(lambda row: row[row > 0].index, axis=1).values)
```

```
[Index(['experienced', 'life', 'problem', 'reality', 'solved'], dtype='object')]
```