# NLP Assignment 3 Review

Elio Desnoulez #19129131

# Introduction and Goal

- Retrieving data from articles from http://scrapsfromtheloft.com/
- Selecting only the data from 
- Save the retrieved data into local disk with pickle library of Python

# Introduction and Goal

- Clean the collected data by performing the following tasks :
  - Make the text all lowercase
  - Remove the punctuation
  - Remove numerical values
  - Remove common non-sensical text like ‘/n’ and ‘\t’
  - Tokenize the text
  - Remove the english stop words
- Create a document-term matrix using CountVectorizer from sklearn

# Check if the backup exists

```
backup_file = 'downloaded_pages.backup'
if not os.path.exists(backup_file):
    print("Backup file not existing, creating a new one.")
    pages_dict = createBackup(url_list, backup_file)
else:
    print("Backup file existing, importing it.")
    pages_dict = unpickleFile(backup_file)
```

Check if the backup file exists

```
def createBackup(url_list, backup_file):
    pages_dict = {
        url_list[0]: getDivContent(downloadURL(url_list[0])),
        url_list[1]: getDivContent(downloadURL(url_list[1])),
        url_list[2]: getDivContent(downloadURL(url_list[2])),
        url_list[3]: getDivContent(downloadURL(url_list[3])),
        url_list[4]: getDivContent(downloadURL(url_list[4]))
    }
    pickleFile(backup_file, pages_dict)
    print("File", backup_file, "created.")
    return pages_dict
```

Creating a backup file with div content data

# Using Pickle to backup/retrieve data

```
def pickleFile(filename, pages_dict):
    print("Pickling File")
    f = open(filename, 'wb')
    pickle.dump(pages_dict, f)
    f.close()
    print("File pickled")
```

Dump the dictionary in a file to backup it

```
def unpickleFile(filename):
    print("Unpickling File")
    f = open(filename, 'rb')
    new_dict = pickle.load(f)
    f.close()
    print("File unpickled")
    return new_dict
```

Retrieving the dictionary from the backup

# Retrieve data from scrapsfromtheloft site

```
def readFile(filename):
    f = open(filename, "r")
    data = []
    for line in f.readlines():
        line = re.sub(r'[\n]', '', line)
        data.append(line)
    return data
```

Get the URL list

```
def downloadURL(url):
    print('Downloading page :', url)
    res = requests.get(url)
    if res.status_code == 200:
        print('Download Success!')
    elif res.status_code == 404:
        print('Error 404 with URL ', url, ' : Not Found.')
    else:
        print('Error ', res.status_code, 'on URL : ', url)
    return res.content
```

Download the page from URL

# Parse the data from downloaded pages

```
def getDivContent(page):
    soup = BeautifulSoup(page, 'html.parser')
    post_content = soup.find(class_="post-content").find_all('p')
    post_content = ' '.join(map(str, post_content))
    return(post_content)
```

Get the data from the div class post-content
with BeautifulSoup4

# Clean the data retrieved

```
def cleanText(text):
    print("Cleaning Text")
    text = text.lower() #Lowercase Text
    text = re.sub(r'<\/?[\w ="-:;]*>', '', text) #Remove HTML Tags
    text = re.sub(r'-', ' ', text) #Replace - by space
    text = re.sub(r'\d+', '', text) #Remove digits
    text = re.sub(r'[^\w\s\'’]','', text) #Remove punctuation without apostrophe
    text = re.sub(r'[\’’]', '\'', text) #Replace ’ by '
    text = re.sub(r'[\n]', ' ', text) #Replace \n by space
    text = re.sub(r'[\t]', '', text) #Remove \t

    tokenized_text = word_tokenize(text) #Tokenizetext

    filtered_text = []
    stop_words = set(stopwords.words('english'))
    #Remove stopwords
    for i in range(len(tokenized_text)):
        if not tokenized_text[i] in stop_words:
            filtered_text.append(tokenized_text[i])
    print("Done.")
    return filtered_text
```

Cleaning the text, tokenizing it and remove stop words

# Create a matrix with CountVectorizer

```
vec = CountVectorizer(min_df=5,
                      max_df=0.9,
                      ngram_range=(1, 2),
                      token_pattern='(\S+)').fit(corpus)
bag_of_words = vec.transform(corpus)
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
print(words_freq)
print(bag_of_words.toarray())
```

Creating a bag of words representation and a matrix with words frequencies

# Output of the program

```
Backup file not existing, creating a new one.
Downloading page : http://scrapsfromtheloft.com/2017/05/06/louis-ck-oh-my-god-full-transcript/
Download Success!
Downloading page : http://scrapsfromtheloft.com/2019/08/25/bill-maher-trump-save-earth-transcript/
Download Success!
Downloading page : http://scrapsfromtheloft.com/2019/05/22/wanda-sykes-not-normal-transcript/
Download Success!
Downloading page : http://scrapsfromtheloft.com/2019/09/12/george-carlin-dumb-americans-transcript/
Download Success!
Downloading page : http://scrapsfromtheloft.com/2019/08/26/brazil-corruption-amazon-hasan-minhaj/
Download Success!
Pickling File
File pickled
File downloaded_pages.backup created.
Treating URL : http://scrapsfromtheloft.com/2017/05/06/louis-ck-oh-my-god-full-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/08/25/bill-maher-trump-save-earth-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/05/22/wanda-sykes-not-normal-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/09/12/george-carlin-dumb-americans-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/08/26/brazil-corruption-amazon-hasan-minhaj/
Cleaning Text
Done.
```

If the backup file is not existing

```
Backup file existing, importing it.
Unpickling File
File unpickled
Treating URL : http://scrapsfromtheloft.com/2017/05/06/louis-ck-oh-my-god-full-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/08/25/bill-maher-trump-save-earth-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/05/22/wanda-sykes-not-normal-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/09/12/george-carlin-dumb-americans-transcript/
Cleaning Text
Done.
Treating URL : http://scrapsfromtheloft.com/2019/08/26/brazil-corruption-amazon-hasan-minhaj/
Cleaning Text
Done.
```

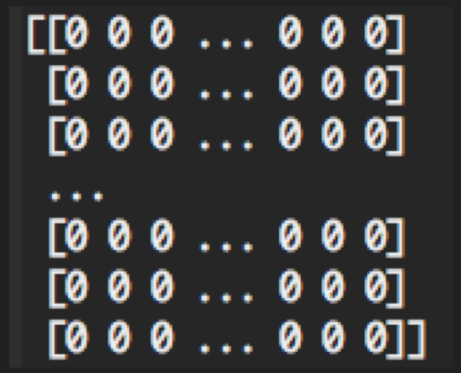
If the backup file is existing

# Output of the program

```
[("'s", 499), ('like', 280), ("n't", 254), ('know', 188), ("'re", 168), ("'m", 134), ('people', 115), ('get', 115), ('one', 100), ('got', 82), ('right', 81), ('
shit', 74), ('yeah', 65), ('go', 61), ('oh', 57), ('na', 54), ('would', 53), ('gon', 52), ('look', 48), ('even', 47), ('want', 46), ('think', 46), ('life', 46),
 ('time', 45), ('fuck', 45), ('good', 44), ('thing', 42), ("'ll", 40), ('see', 39), ('every', 38), ('kids', 38), ('fucking', 38), ('black', 38), ('white', 37),
('bolsonaro', 37), ('really', 36), ('let', 35), ('man', 35), ('come', 34), ('say', 33), ('going', 33), ('way', 33), ('guy', 32), ('little', 31), ('first', 30),
("'cause", 30), ("'ve", 30), ('love', 29), ('need', 29), ('us', 29), ('said', 29), ('president', 29), ('brazil', 29), ('amazon', 29), ('always', 28), ('put', 28
), ('yes', 28), ('take', 28), ('big', 28), ('trump', 28), ('lot', 27), ('well', 26), ('day', 26), ('ever', 26), ('give', 26), ('ca', 26), ('women', 26), ('thing
s', 25), ('uh', 25), ('country', 24), ('two', 24), ('home', 24), ('old', 23), ("'d", 23), ('back', 23), ('could', 23), ('world', 23), ('course', 23), ('another'
, 22), ('tit', 22), ('rainforest', 22), ('indigenous', 22), ('everybody', 21), ('mean', 21), ('bad', 21), ('okay', 21), ('face', 20), ('woman', 19), ('food', 19
), ('anything', 19), ('nobody', 19), ('start', 19), ('something', 19), ('maybe', 19), ('vicks', 19), ('nice', 18), ('around', 18), ('looking', 18), ('everything
', 18), ('hey', 18), ('ai', 18), ('better', 18), ('part', 18), ('older', 18), ('murder', 18), ('sônia', 18), ('kind', 17), ('different', 17), ('live', 17), ('ne
w', 17), ('god', 17), ('ok', 17), ('still', 17), ('land', 17), ('dog', 16), ('reason', 16), ('never', 16), ('sitting', 16), ('happened', 16), ('thought', 16), (
'america', 16), ('saying', 15), ('tell', 15), ('long', 15), ('years', 15), ('whole', 15), ('show', 15), ('last', 15), ('deforestation', 15), ('three', 14), ('se
cond', 14), ('person', 14), ('care', 14), ('stop', 14), ('year', 14), ('worst', 14), ('feel', 14), ('help', 14), ('called', 14), ('playing', 14), ('guajajara',
14), ('thank', 13), ('place', 13), ('money', 13), ('went', 13), ('sorry', 13), ('date', 13), ('earth', 13), ('wife', 13), ('esther', 13), ('jbs', 13), ('many',
12), ('saw', 12), ('school', 12), ('die', 12), ('fat', 12), ('deal', 12), ('eat', 12), ('make', 12), ('ha', 12), ('trying', 12), ('believe', 12), ('getting', 12
), ('guys', 12), ('ass', 12), ('seen', 12), ('tits', 12), ('much', 11), ('looks', 11), ('comes', 11), ('says', 11), ('knew', 11), ('made', 11), ('almost', 11),
('room', 11), ('car', 11), ('history', 11), ('cause', 11), ('dumb', 11), ('wanda', 11), ('normal', 11), ("y'all", 11), ('mm', 11), ('boom', 11), ('hot', 11), ('
hasan', 11), ('hope', 10), ('young', 10), ('toilet', 10), ('kid', 10), ('somebody', 10), ('news', 10), ('happens', 10), ('watching', 10), ('half', 10), ('takes'
, 10), ('best', 10), ('great', 10), ('behind', 10), ('mall', 10), ('stupid', 10), ('jair', 10), ('minhaj', 10), ('york', 9), ('lady', 9), ("'", 9), ('head', 9),
 ('keep', 9), ('goes', 9), ('jesus', 9), ('house', 9), ('stuff', 9), ('pretty', 9), ('fact', 9), ('already', 9), ('minute', 9), ('stay', 9), ('open', 9), ('actu
ally', 9), ('watch', 9), ('play', 9), ('fucked', 9), ('damn', 9), ('motherfuckers', 9), ('american', 9), ('opioids', 9), ('assholes', 9), ('lice', 9), ('walking
', 8), ('wait', 8), ('ta', 8), ('tv', 8), ('must', 8), ('away', 8), ('week', 8), ('coming', 8), ('beautiful', 8), ('nothing', 8), ('asshole', 8), ('smart', 8),
('gets', 8), ('body', 8), ('number', 8), ('dick', 8), ('baby', 8), ('change', 8), ('legal', 8), ('become', 8), ('fight', 8), ('also', 8), ('family', 8), ('ameri
cans', 8), ('malls', 8), ('beef', 8), ('brazilian', 8), ('batista', 8), ('building', 7), ('driving', 7), ('left', 7), ('ends', 7), ('rest', 7), ('daughter', 7),
```

Printing the words occurrences

# Output of the program




Displaying the array