Sergi Solís Valdivieso u172769 - Hong-ming Xiang Vico u172801

IRWA Project Part I: Text Processing

Statement

You are provided with a document corpus which is a set of tweets related to Hurricane Ian (tw_hurricane_data.json). You can see an example document in the appendix.
As a first step, you must pre-process the documents by:

- Removing stop words
- Tokenization
- Removing punctuation marks
- Stemming
- and... anything else you think it's needed (bonus point)
-

HINTS:
1. Take into account that for future queries, the final output must return (when present) the following information for each of the selected documents: **Id | Tweet | Username | Date | Hashtags | Likes | Retweets | Url** (here the "Url" means the tweet link).

2. Think about how to handle the hashtags from your pre-processing steps (e.g., removing the "#" from the word), since it may be useful to involve them as separated terms inside the inverted index.

3. Suggested library that may help you in stemming and stop words: **nltk**
Make sure you map the tweet's Ids with the document ids as the document Ids will be considered for the evaluation stage of the project (tweet_document_ids_map.csv).

GitHub Repository

All the code and resources for the project will be submitted to the following repository:
https://github.com/homexiang3/IRWA-2022-u172769-u172801

Code development

Drive connection

The first step in our code development is to import the Google Drive package to access our resource data files.

```
[8] from google.colab import drive
    drive.mount('/content/drive')
```

Import packages

To have a more clean and organized code, we create a code block to import all the packages used in this project.

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from collections import defaultdict
from array import array
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
import collections
import json
import re
from tabulate import tabulate
stemmer = nltk.stem.SnowballStemmer('english')
stopwords = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Some important packages to mention are:

nltk: Bring us built-in functions for stemming and getting stopwords list.
json: Allow us to load and parse data from a JSON file.
re: Allow us to use regular expressions to remove punctuation marks or digits.
tabulate: Provide a simple and elegant data visualization table

Load data

In this section, we take the data file from our Google Drive and store it in a single variable.

In the beginning, we faced some problems loading the data from the JSON file due to our JSON format. However, we found some documentation that allows us to load the JSON file using list comprehension, without modifying the original file.

For checking purposes, we add a print line to see the first tweet and another to see the total number of Tweets. we conclude that the data loaded is correct since the results are successful.

```
[18] # Load file path
     file_name = '/content/drive/Shareddrives/IRWA/PROJECT/data/tw_hurricane_data.json'
     # Use json.loads function with list comprehension to obtain all the tweets
     lines = [json.loads(line) for line in open(file_name,'r')]
     # Print first tweet for checking purposes
     print(lines[0])

     {'created_at': 'Fri Sep 30 18:39:08 +0000 2022', 'id': 1575918182698979328, 'id_str
```

```
[19] # Print total number of tweets
     print("Total number of Tweets: {}".format(len(lines)))

     Total number of Tweets: 4000
```

Attribute selection

Following the statement instructions, we want to keep the attributes **Id | Tweet | Username | Date | Hashtags | Likes | Retweets | Url.**

We decided to create our own class Tweet, with the requested attributes.

```
class Tweet:
  def __init__(self, id, tweet, username, date, hashtags, likes, retweets, url):
    self.id = id
    self.tweet = tweet
    self.username = username
    self.date = date
    self.hashtags = hashtags
    self.likes = likes
    self.retweets = retweets
    self.url = url
  def aslist(self):
        return [self.id, self.tweet, self.username, self.date, self.hashtags, self.likes, self.retweets, self.url]
  def __iter__(self):
        return iter(self.aslist())
```

The first step is to identify which fields correspond to each attribute and assign it to our new class object, for each record on our data. Then we will append to a list of tweets and proceed with the pre-processing step.

In this part, we faced some problems because not all the tweets contain the attribute corresponding to the URL. We decided to put a default value of empty string for those who haven't got the attribute. Also, we implemented an internal for loop to store each one of the hashtags.

```
tweets = []

for i in range(len(lines)):

    hashtags = []
    url = ""

    if 'media' in lines[i]['entities']:
      url = lines[i]['entities']['media'][0]['url']

    for j in range(len(lines[i]['entities']['hashtags'])):
      hashtags.append(lines[i]['entities']['hashtags'][j].get('text'))

    tweets.append(Tweet(lines[i]['id'],
                        lines[i]['full_text'],
                        lines[i]['user']['screen_name'],
                        lines[i]['created_at'],
                        hashtags,
                        lines[i]['favorite_count'],
                        lines[i]['retweet_count'],
                        url))
```

Finally, for checking purposes we print the total number of tweets to see that the number of tweets hasn't been modified.

```
[70] # Print total number of tweets
     print("Total number of Tweets: {}".format(len(tweets)))

     Total number of Tweets: 4000
```

Pre-process

Now that we have our JSON file loaded, and our list of tweets prepared. We can start applying some text processing techniques, following the statement instructions we proceed with stopwords elimination, stemming, tokenization, and removing punctuation marks. As an additional pre-processing step we also add a hashtag, URL, numerical, white spaces, and emoji elimination.

Remove white spaces

We use the join function to eliminate white spaces with text.split() as a unique parameter.

```
# Remove white spaces
def remove_white_space(text):
    return ' '.join(text.split())
```

Remove punctuation marks and hashtags

We use a regex to substitute punctuation marks and hashtags in the string (more precisely we remove everything not considered word or space).

```
# Remove punctuation and hashtags
def remove_punctuation(data):
    return re.sub(r'[^\w\s]', '', data)
```

Remove stopwords

We use the stopwords list to check each of the words, only returning the words that didn't find a match in the list. The stopwords variable was initialized in the import packages section.

```
# Remove stopwords
def remove_stopwords(words):
    return [w for w in words if w.lower() not in stopwords]
```

Remove numbers

We use a regex to substitute numbers in the string.

```
# Remove numbers
def remove_numbers(data):
    return re.sub(r'[0-9]', '', data)
```

Remove URL

For each tokenized word, we use the function startswith to check if they start with "https", using list comprehension.

```
def remove_https(words):
  return [w for w in words if not w.startswith("https") ]
```

Remove emojis

In this function we follow some internet guidelines since emojis use Unicode format and we need to apply some additional steps. We compiled all the possible emojis in *emoj* variable and then eliminate them from the string if any emoji is found.

```
# Remove emojis
def remove_emojis(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return emoj.sub(r'', data)
```

Stemming

We use the built-in stem function for each of the tokenized words to replace them, using list comprehension. The *stemmer* variable was initialized in the import packages section.

```
words = [stemmer.stem(word) for word in words]
```

Pre-process function

In this function we group all the functions previously mentioned.

First of all, we eliminate the newlines in our text and proceed with the full-text elimination functions (emojis, punctuation marks and hashtags, numbers, and white spaces).

Then we tokenize each of the words in the text with the *nltk* function *“word_tokenize”.* Now we could apply our tokenized-word elimination functions (stemming, stopwords, and HTTPS).

Finally, we join the tokenized words and return the text pre-processed.

```
# Preprocess text
def preprocess(text):
    text = text.replace('\\n', '')
    text = remove_emojis(text)
    text = remove_punctuation(text)
    text = remove_numbers(text)
    text = remove_white_space(text)
    words = nltk.tokenize.word_tokenize(text)
    words = [stemmer.stem(word) for word in words]
    words = remove_stopwords(words)
    words = remove_https(words)
    text = " ".join(words)
    return text
```

Results

Now that all our functions have been created, we could pre-process each of the tweets, but first of all, We decided to print the original text with the processed text to check if the pre-processing was done correctly.

Note: We decided to implement the preprocess only in the full text of the tweets since some steps like the number, hashtag, and HTTPS elimination didn't have a sense for other attributes (number or likes and retweets, URL, hashtags).

```
# Compare not pre-processed with processed text
for i in range(10):
    print("====================")
    print("ORIGINAL TEXT")
    print("====================")
    print(tweets[i].tweet)
    print("====================")
    print("PROCESSED TEXT")
    print("====================")
    print(preprocess(tweets[i].tweet))
    print('\n')

for i in range(len(tweets)):
  tweets[i].tweet = preprocess(tweets[i].tweet)
```

In the screenshot below we have two Tweets where we can see all the techniques used. In both, we could see how punctuation marks, hashtags, and stop words were removed, and also how the stemming changed the text.

Additionally, in the first one, we could see the link and the number elimination, and in the second one the emoji (blue heart), was successfully eliminated.

```
====================
ORIGINAL TEXT
====================
So this will keep spinning over us until 7 pm…go away already. #HurricaneIan https://t.co/VROTxNS9rz
====================
PROCESSED TEXT
====================
keep spin us pmgo away alreadi hurricaneian


====================
ORIGINAL TEXT
====================
Our hearts go out to all those affected by #HurricaneIan. We wish everyone on the roads currently braving the conditions safe travels. 💙
====================
PROCESSED TEXT
====================
heart go affect hurricaneian wish everyon road current brave condit safe travel
```

Visualization

In the end, we use *tabulate* to visualize the first 10 tweets with all the requested attributes. We need to create a new list, add the first 10 tweets, define the headers and then print the tabulate function with the next parameters:

1. List of tweets to display
2. Headers of table
3. Table style

```
# Final visualization of tweets
visualization_tweets = []
for i in range(10):
  visualization_tweets.append(tweets[i])


headers = ['ID','TWEET','USERNAME','DATE','HASHTAGS','LIKES', 'RETWEETS', 'URL']
print(tabulate(visualization_tweets, headers=headers, tablefmt='fancy_grid'))
```

After running the code cell, we obtain this table (notice that since the table is large, we separate it in two screenshots).

ID	TWEET
1575918182698979328	keep spin us pmgo away alreadi hurricaneian
1575918151862304768	heart go affect hurricaneian wish everyon road current brave condit safe travel

USERNAME	DATE	HASHTAGS	LIKES	RETWEETS	URL
suzjdean	Fri Sep 30 18:39:08 +0000 2022	['HurricaneIan']	0	0	https://t.co/VROT
lytx	Fri Sep 30 18:39:01 +0000 2022	['HurricaneIan']	0	0	