TextAnalytics_Homework1_Final-Copy1

October 11, 2019

0.1 Homework 1

0.1.1 Exercise 1 : Text processing using libraries

Read in Newsgroup dataset

```
In [2]: # WRITE TEXT FILES FROM 5 SUB-FOLDERS THAT WERE PART OF THE NEWSGROUP FOLDER INTO ONE

        import os

        os.chdir("20-newsgroups/")

        import glob

        read_files = glob.glob("*.txt")

        with open("final_text.txt", "wb") as outfile:
            i=0
            for f in read_files:
                i+=1
                if(i==5):
                    break
                with open(f, "rb") as infile:
                    outfile.write(infile.read())

        # READ IN THE FINAL TEXT FILE
        with open("final_text.txt", encoding="utf8", errors='ignore') as file:
            test_text = file.read()
```

```
In [2]: len(test_text)

Out[2]: 13024142
```

0.1.2 Library 1 : NLTK

```
In [3]: #Import necessary libraries
        import datetime
        import string
        import nltk
        from nltk.tokenize import sent_tokenize, word_tokenize, RegexpTokenizer
        from nltk.corpus import stopwords
        from nltk.stem import WordNetLemmatizer
        from nltk.stem.porter import PorterStemmer

In [4]: #nltk.download()
        nltk.download('stopwords')
        nltk.download('punkt')
        SENT_DETECTOR = nltk.data.load('tokenizers/punkt/english.pickle')

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


In [5]: #### Define a function to remove punctuation and get the resulting punctuation-free te

In [6]: def remove_punctuation(text):
            no_punct = "".join([c for c in text if c not in string.punctuation])
            return no_punct

In [7]: now = datetime.datetime.now()
        no_punct = remove_punctuation(test_text)
        print("Took %s"%(datetime.datetime.now()-now))

Took 0:00:01.131350


In [8]: #### Time the function when applied to the original text as well as the punctuation fr

In [9]: now = datetime.datetime.now()
        tokens = nltk.word_tokenize(test_text)
        print("Took %s"%(datetime.datetime.now()-now))

Took 0:00:14.828852


In [10]: print(tokens[:20])

['Newsgroup', ':', 'sci.crypt', 'document_id', ':', '14147', 'From', ':', 'Marc', 'VanHeyninge


In [11]: now = datetime.datetime.now()
         no_punct_tokens = nltk.word_tokenize(no_punct)
         print("Took %s"%(datetime.datetime.now()-now))

Took 0:00:06.731250
```

```
In [12]: print(no_punct_tokens[0:15])

['Newsgroup', 'scicrypt', 'documentid', '14147', 'From', 'Marc', 'VanHeyningen', 'mvanheyncsin
```

```
In [13]: #### Alternately, convert the text to lower case and use regex to remove punctuation

In [14]: now = datetime.datetime.now()
         tokenizer = RegexpTokenizer(r'\w+')
         revised_tokens = tokenizer.tokenize(test_text.lower())
         print("Took %s"%(datetime.datetime.now()-now))

Took 0:00:00.630917
```

```
In [15]: print(revised_tokens[0:50])

['newsgroup', 'sci', 'crypt', 'document_id', '14147', 'from', 'marc', 'vanheyningen', 'mvanhey
```

```
In [16]: #### Remove stop words

In [17]: now = datetime.datetime.now()
         words = [w for w in revised_tokens if w not in stopwords.words('english')]
         print("Took %s"%(datetime.datetime.now()-now))

Took 0:05:11.533861
```

```
In [18]: #### Get POS tags

In [19]: now = datetime.datetime.now()
         tagged = nltk.pos_tag(words)
         print("Took %s"%(datetime.datetime.now()-now))
         print(tagged[0:15])

Took 0:01:07.512428
[('newsgroup', 'JJ'), ('sci', 'NN'), ('crypt', 'VBD'), ('document_id', 'JJ'), ('14147', 'CD'),
```

```
In [20]: #### Identify named entities

In [21]: now = datetime.datetime.now()
         entities = nltk.chunk.ne_chunk(tagged)
         print("Took %s"%(datetime.datetime.now()-now))
         print(entities[0:15])

Took 0:04:09.264478
[('newsgroup', 'JJ'), ('sci', 'NN'), ('crypt', 'VBD'), ('document_id', 'JJ'), ('14147', 'CD'),
In [22]: #### Perform stemming on the text after removal of punctuation and stop words

In [4]: stemmer = PorterStemmer()
        def word_stemmer(text):
            stem_text = " ".join([stemmer.stem(i) for i in text])
            return stem_text

In [24]: now = datetime.datetime.now()
         stemmed_words = word_stemmer(words)
         print("Took %s"%(datetime.datetime.now()-now))
         print(stemmed_words[0:15])

Took 0:00:22.554763
newsgroup sci c


In [25]: print(stemmed_words[0:120])

newsgroup sci crypt document_id 14147 marc vanheyningen mvanheyn cs indiana edu subject ripem 


In [5]: #### Define function that performs all the pre processing steps at once
        def preprocessing_nltk(text):

            now = datetime.datetime.now()

            tokenizer = RegexpTokenizer(r'\w+')
            revised_tokens = tokenizer.tokenize(text.lower())

            words = [w for w in revised_tokens if w not in stopwords.words('english')]

            tagged = nltk.pos_tag(words)

            stemmed_words = word_stemmer(words)

            print("Took %s"%(datetime.datetime.now()-now))

            return [words, tagged, stemmed_words]


In [6]: processed_text = preprocessing_nltk(test_text)
        print("Examples of tokens obtained are : ", processed_text[0][0:25]),
        print("Examples of pos tags obtained are : ", processed_text[1][0:25])
        print("Examples of stemmed tokens obtained are : ", processed_text[2][0:25])

Took 0:05:56.012552
Examples of tokens obtained are :  ['newsgroup', 'sci', 'crypt', 'document_id', '14147', 'marc
Examples of pos tags obtained are :  [('newsgroup', 'JJ'), ('sci', 'NN'), ('crypt', 'VBD'), ('
Examples of stemmed tokens obtained are :  newsgroup sci crypt docum
```

0.1.3 Library 2 : SpaCy

```
In [13]: import re
         import spacy
         from spacy.tokenizer import Tokenizer
         from spacy.lang.en import English
         from spacy.lang.en.stop_words import STOP_WORDS
         nlp = spacy.load("en")
         nlp.max_length = 139844934
```

```
In [14]: def preprocessing_spacy(text):
             now = datetime.datetime.now()

             #  "nlp" Object is used to create documents with linguistic annotations.
             my_doc = nlp(text, disable = ['ner', 'parser'])

             # Create dictionary of tokens and corresponding strings
             token_dict = {}
             for token in my_doc:
                 token_dict[token] = token.text

             # Create list of word tokens after removing punctuation and stopwords
             filtered_sentence =[]

             for token in token_dict:
                 lexeme = nlp.vocab[token_dict[token]]

                 if not token.is_punct | token.is_space:

                     if lexeme.is_stop == False:

                         filtered_sentence.append(token)

             processed_dict = {}

             # Get the processed tokens and corresponding lemmatized forms as well as the POS
             for token in filtered_sentence:
                 processed_dict[token] = [token.lemma_, token.pos_, token.tag_]

             print("Took %s"%(datetime.datetime.now()-now))
             return processed_dict
```

```
In [16]: processed_text = preprocessing_spacy(test_text)
```

```
Took 0:02:24.870070
```

```
In [19]: ### The function defined above returns a token and the associated lemmatized versions
         for x in list(processed_text)[0:25]:
             print ("{},{} ".format(x,  processed_text[x]))
```

```
Newsgroup,['Newsgroup', 'PROPN', 'NNP']
sci.crypt,['sci.crypt', 'PROPN', 'NNP']
document_id,['document_id', 'NUM', 'CD']
14147,['14147', 'NUM', 'CD']
Marc,['Marc', 'PROPN', 'NNP']
VanHeyningen,['VanHeyningen', 'PROPN', 'NNP']
<,['<', 'X', 'XX']
mvanheyn@cs.indiana.edu,['mvanheyn@cs.indiana.edu', 'PROPN', 'NNP']
>,['>', 'X', 'XX']
Subject,['subject', 'NOUN', 'NN']
RIPEM,['RIPEM', 'PROPN', 'NNP']
Frequently,['frequently', 'ADV', 'RB']
Asked,['ask', 'VERB', 'VBD']
Questions,['Questions', 'PROPN', 'NNP']
Archive,['archive', 'ADJ', 'JJ']
ripem,['ripem', 'PROPN', 'NNP']
faq,['faq', 'PROPN', 'NNP']
update,['update', 'NOUN', 'NN']
Sun,['Sun', 'PROPN', 'NNP']
7,['7', 'NUM', 'CD']
Mar,['Mar', 'PROPN', 'NNP']
93,['93', 'NUM', 'CD']
21:00:00,['21:00:00', 'NUM', 'CD']
-0500,['-0500', 'NOUN', 'NN']
POSTING,['posting', 'NOUN', 'NN']
```

0.2 Parallelising Preprocessing

```
In [23]: import multiprocessing
         from multiprocessing import Pool
         from os import listdir
         from os.path import isfile, join
```

```
In [24]: def read_doc(x):
             return str(open(mypath+x,encoding="utf8", errors='ignore').read())

         def read_data(path):
             """ Read data from all docs in a path
             """
             file_names = [f for f in listdir(path) if isfile(join(path, f))]
             file_names = [f for f in file_names if not f.startswith('.')]
             file_names = file_names[0:5]
             print(file_names)
             p = Pool(8)
             texts = p.map(read_doc, file_names)

             p.close()
```

```
            return texts

        mypath = 'Text Analytics/20-newsgroups/'
        space_texts = read_data(mypath)


        # with open("final_text.txt", encoding="utf8", errors='ignore') as file:
        #     test_text = file.read()
```

```
['sci.crypt.txt', 'comp.sys.mac.hardware.txt', 'misc.forsale.txt', 'soc.religion.christian.txt
```

```
In [ ]: count = multiprocessing.cpu_count()
        print(count)
        pool = multiprocessing.Pool(count)
```

```
In [26]: now=datetime.datetime.now()
         result=list(pool.map(preprocessing_nltk, space_texts))
         print("Multiprocessing took %s"%(datetime.datetime.now()-now))
```

```
Multiprocessing took 0:02:32.089033
```

```
In [ ]: ### Using the standard multiprocessing library with spacy results in an error regardin
```

```
In [33]: # now = datetime.datetime.now()
         # result = list(pool.map(preprocessing_spacy,space_texts))
         # print("Took %s"%(datetime.datetime.now()-now))
```

```
In [ ]: ### Hence nlp.pipe() is used instead so that the library works on minibatches of docum
```

```
In [20]: def multiprocessing_spacy(doc):
             # Create dictionary of tokens and corresponding strings
             token_dict = {}
             for token in doc:
                 token_dict[token] = token.text

             # Create list of word tokens after removing punctuation and stopwords
             filtered_sentence =[]

             for token in token_dict:
                 lexeme = nlp.vocab[token_dict[token]]

                 if not token.is_punct | token.is_space:

                     if lexeme.is_stop == False:

                         filtered_sentence.append(token)
```

```
        processed_dict = {}

        # Get the processed tokens and corresponding lemmatized forms as well as the POS
        for token in filtered_sentence:
            processed_dict[token] = [token.lemma_, token.pos_, token.tag_]

        print("Took %s"%(datetime.datetime.now()-now))
        return processed_dict
```

```
In [21]: now=datetime.datetime.now()
         docs = [nlp.pipe(test_text, disable=["ner", "parser"])]
         print("Took %s"%(datetime.datetime.now()-now))
```

```
Took 0:00:00.000106
```

```
In [ ]: now=datetime.datetime.now()
        processed_text = [multiprocessing_spacy(doc) for doc in docs]
        print("Took %s"%(datetime.datetime.now()-now))
```