hw1_text_analytics

October 11, 2019

0.1 Q1: Using nltk, textblob, and spacy for tokenization, stemming, and pos tagging

```
In [41]: import re
         import pandas as pd
         import textblob
         import spacy
         import nltk
         import tarfile
         import time
         from spacy.tokens import Doc
         from spacy.lang.en import English
```

```
In [4]: # data obtained from https://www.kaggle.com/snap/amazon-fine-food-reviews
```

```
In [7]: df = pd.read_csv('~/Downloads/amazon-fine-food-reviews/Reviews.csv')
```

```
In [8]: df.head()
```

```
Out[8]:    Id   ProductId          UserId                      ProfileName  \
        0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
        1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
        2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
        3   4  B000UA0QIQ  A395BORC6FGVXV                             Karl
        4   5  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

           HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
        0                     1                       1      5  1303862400
        1                     0                       0      1  1346976000
        2                     1                       1      4  1219017600
        3                     3                       3      2  1307923200
        4                     0                       0      5  1350777600

                         Summary                                               Text
        0  Good Quality Dog Food  I have bought several of the Vitality canned d...
        1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
        2  "Delight" says it all  This is a confection that has been around a fe...
        3         Cough Medicine  If you are looking for the secret ingredient i...
        4            Great taffy  Great taffy at a great price.  There was a wid...
```

```
In [9]: df.shape

Out[9]: (568454, 10)

In [11]: # create full corpus of all reviews
         corpus = ''
         for row in df.itertuples():
             corpus += row[10]
             corpus += ' '

In [75]: sample = corpus[:100000]

In [76]: # tokenizing in nltk

         start_time = time.time()

         tokens = nltk.tokenize.word_tokenize(sample)

         elapsed_time = time.time()-start_time
         print(len(tokens))
         print(elapsed_time)
         print(tokens[:20])

21858
0.20186400413513184
['I', 'have', 'bought', 'several', 'of', 'the', 'Vitality', 'canned', 'dog', 'food', 'products


In [77]: # tokenizing in textblob

         blob = textblob.TextBlob(sample)

         start_time = time.time()

         tokens = blob.words

         elapsed_time = time.time()-start_time
         print(len(tokens))
         print(elapsed_time)
         print(tokens[:20])

18905
0.5088388919830322
['I', 'have', 'bought', 'several', 'of', 'the', 'Vitality', 'canned', 'dog', 'food', 'products


In [78]: # tokenizing in spacy

         nlp = English()
         # Create a Tokenizer with the default settings for English
         # including punctuation rules and exceptions
         tokenizer = nlp.Defaults.create_tokenizer(nlp)

         start_time = time.time()

         tokens = tokenizer(sample)

         elapsed_time = time.time()-start_time
         print(len(tokens))
         print(elapsed_time)
         print(tokens[:20])
```

```
21563
0.5657451152801514
I have bought several of the Vitality canned dog food products and have found them all to be o
```

```
In [79]: # stemming in nltk

         tokens = nltk.tokenize.word_tokenize(sample)

         ps = nltk.stem.PorterStemmer()

         start_time = time.time()

         stemmed = [ps.stem(word) for word in tokens]

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(stemmed[:20])
```

```
0.3578910827636719
['I', 'have', 'bought', 'sever', 'of', 'the', 'vital', 'can', 'dog', 'food', 'product', 'and',
```

```
In [80]: # stemming in textblob

         tokens = blob.words

         start_time = time.time()

         stemmed = [textblob.Word(word).lemmatize() for word in tokens]

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(stemmed[:20])
```

```
0.14100313186645508
['I', 'have', 'bought', 'several', 'of', 'the', 'Vitality', 'canned', 'dog', 'food', 'product'
```

```
In [81]: # stemming in spacy

         tokens = tokenizer(sample)

         start_time = time.time()

         stemmed = [token.lemma_ for token in tokens]

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(stemmed[:20])
```

```
0.019247055053710938
['I', 'have', 'buy', 'several', 'of', 'the', 'Vitality', 'can', 'dog', 'food', 'product', 'and
```

```
In [85]: # pos tagging in nltk

         tokens = nltk.tokenize.word_tokenize(sample)

         start_time = time.time()

         pos_tags = [nltk.pos_tag(word) for word in tokens]

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(pos_tags[:20])
```

```
7.4419121742248535
[[('I', 'PRP')], [('h', 'VB'), ('a', 'DT'), ('v', 'NN'), ('e', 'NN')], [('b', 'NN'), ('o', 'NN
```

```
In [87]: # pos tagging in textblob

         start_time = time.time()

         pos_tags = blob.pos_tags

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(pos_tags[:20])
```

```
7.295608520507812e-05
[('I', 'PRP'), ('have', 'VBP'), ('bought', 'VBN'), ('several', 'JJ'), ('of', 'IN'), ('the', 'D
```

```
In [88]: # pos tagging in spacy

         nlp = spacy.load("en_core_web_sm")

         tokens = nlp(sample)

         start_time = time.time()

         pos = [token.tag_ for token in tokens]

         elapsed_time = time.time()-start_time

         print(elapsed_time)
         print(pos[:20])
```

```
0.015252828598022461
['PRP', 'VBP', 'VBN', 'JJ', 'IN', 'DT', 'NNP', 'VBN', 'NN', 'NN', 'NNS', 'CC', 'VBP', 'VBN', '
```

0.2 Using regex for finding dates and emails

```
In [230]: # 2.1 Match all emails in text and compile a set of all found email addresses.

          email_re = re.compile(r'[a-zA-Z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}')
          x = email_re.findall('dd@dd.com not_an email email.email.com t3hisisanemail@dac.com'
```

```
In [231]: x
```

```
Out[231]: ['dd@dd.com', 't3hisisanemail@dac.com']
```

```
In [334]: # 2.2 Find all dates in text (e.g. 04/12/2019, April 20th 2019, etc).

          dates_re = re.compile(r'((0?[1-9])|(1[1|2])|jan[a-z]*|feb[a-z]*|mar[a-z]*|apr[a-z]*|
```

```
In [335]: test_dates = '01/1/2019 January/2/2019 dec-12-2020 3/2/2013 13/23/2018 10/23/2018'
```

```
In [336]: print ([x.group(0) for x in dates_re.finditer(test_dates)])
```

```
['01/1/2019', 'January/2/2019', 'dec-12-2020', '3/2/2013', '3/23/2018']
```