Notebook

August 1, 2018

Contents

List of Figures

List of Tables

List of Codes

0.0.1 NLP Refresher

```python
text = ["System of the World. By Isaac Newton", "    Snow Crash    . By Neal Stephenson ",
        " AFROFUTURISM. by      Ytasha L. Womack "]
```

```python
strip_whitespace = [string.strip() for string in text]
```

```python
strip_whitespace
```

```
['System of the World. By Isaac Newton',
 'Snow Crash    . By Neal Stephenson',
 'AFROFUTURISM. by      Ytasha L. Womack']
```

```python
strip_whitespace2 = [string.strip() for string in strip_whitespace]
strip_whitespace2
```

```
['System of the World. By Isaac Newton',
 'Snow Crash    . By Neal Stephenson',
 'AFROFUTURISM. by      Ytasha L. Womack']
```

```python
remove_periods = [string.replace(".","") for string in
    ↪ strip_whitespace]
```

```python
remove_periods
```

```
['System of the World By Isaac Newton',
 'Snow Crash    By Neal Stephenson',
 'AFROFUTURISM by      Ytasha L Womack']
```

```python
upper = [string.upper() for string in strip_whitespace]
```

```python
upper
```

```
['SYSTEM OF THE WORLD. BY ISAAC NEWTON',
 'SNOW CRASH   .  BY NEAL STEPHENSON',
 'AFROFUTURISM. BY      YTASHA L. WOMACK']
```

```python
import re
```

```python
xs = [re.sub(r"[a-zA-Z]", "X", string) for string in
    ↪ strip_whitespace]
```

```python
xs
```

```
['XXXXXX XX XXX XXXXX. XX XXXXX XXXXXX',
 'XXXX XXXXX   .  XX XXXX XXXXXXXXXX',
 'XXXXXXXXXXXX. XX      XXXXXX X. XXXXXX']
```

REGEX TUTORIAL

https://www.analyticsvidhya.com/blog/2015/06/regular-expression-python/

0.0.2 Scraping

```python
import requests
from bs4 import BeautifulSoup
```

```python
url = 'https://www.analyticsvidhya.com/blog/2015/06/regular-
    expression-python/'
```

```python
req = requests.get(url)
```

```python
req
```

```
<Response [200]>
```

```
soup = BeautifulSoup(req.text, 'html.parser')
```

```
soup.text[40:50]
```

```
'lar Expres'
```

```
soup.find('h2')
```

```
<h2 class="site-outline">Learn everything about Analytics</h2>
```

```
heads = soup.find_all('h2')
```

```
len(heads)
```

```
6
```

0.0.3 Basic NLP

```
from nltk.tokenize import word_tokenize
import nltk
```

```
nltk.download('punkt')
```

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

```
True
```

```
pgraph = soup.find('p').text
```

```
tokes = word_tokenize(pgraph)
```

```
tokes
```

```
['In',
 'last',
 'few',
 'years',
 ',',
 'there',
 'has',
 'been',
 'a',
 'dramatic',
 'shift',
 'in',
 'usage',
 'of',
 'general',
 'purpose',
 'programming',
 'languages',
 'for',
'data',
'science',
'and',
'machine',
'learning',
'.',
'This',
'was',
'not',
'always',
'the',
'case',
'',
'a',
'decade',
'back',
'this',
'thought',
'would',
'have',
'met',
'a',
'lot',
'of',
'skeptic',
'eyes',
'!']
```

```python
sy = ['.', ',', '!', '-', '?','*', '']
```

```python
for word in tokes:
    if word not in sy:
        print(word)
    else:
        _
```

```
In
last
few
years
there
has
been
a
dramatic
shift
in
usage
of
general
purpose
programming
languages
for
```

```
data
science
and
machine
learning
This
was
not
always
the
case
a
decade
back
this
thought
would
have
met
a
lot
of
skeptic
eyes
```

```python
from nltk.tokenize import sent_tokenize
```

```python
sent_tokenize(pgraph)[0]
```

```
'In last few years, there has been a dramatic shift in usage of
general purpose programming languages for data science and machine
learning.'
```

```python
from nltk.corpus import stopwords
```

```python
stop_words = stopwords.words('english')
```

```python
stop_words[:6]
```

```
['i', 'me', 'my', 'myself', 'we', 'our']
```

```python
[word for word in tokes if word not in stop_words]
```

```
['In',
 'last',
 'years',
 ',',
 'dramatic',
 'shift',
 'usage',
 'general',
 'purpose',
 'programming',
 'languages',
 'data',
 'science',
 'machine',
 'learning',
 '.',
 'This',
 'always',
 'case',
 '',
 'decade',
 'back',
 'thought',
 'would',
 'met',
 'lot',
 'skeptic',
 'eyes',
 '!']
```

```
stop_words[:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "
you're"]
```

```
#stemming
from nltk.stem.porter import PorterStemmer
```

```
porter = PorterStemmer()
```

```
[porter.stem(word) for word in tokes]
```

```
['In',
 'last',
 'few',
 'year',
 ',',
 'there',
 'ha',
 'been',
 'a',
 'dramat',
 'shift',
 'in',
 'usag',
 'of',
 'gener',
 'purpos',
 'program',
 'languag',
 'for',
 'data',
 'scienc',
 'and',
 'machin',
 'learn',
 '.',
 'thi',
 'wa',
 'not',
 'alway',
 'the',
 'case',
 '',
 'a',
 'decad',
 'back',
 'thi',
 'thought',
 'would',
 'have',
 'met',
 'a',
 'lot',
 'of',
 'skeptic',
 'eye',
 '!']
```

```python
from nltk import pos_tag
```

```python
text_tagged = pos_tag(tokes)
```

```python
text_tagged
```

```
[('In', 'IN'),
 ('last', 'JJ'),
 ('few', 'JJ'),
 ('years', 'NNS'),
 (',', ','),
 ('there', 'EX'),
 ('has', 'VBZ'),
 ('been', 'VBN'),
 ('a', 'DT'),
 ('dramatic', 'JJ'),
 ('shift', 'NN'),
 ('in', 'IN'),
 ('usage', 'NN'),
 ('of', 'IN'),
 ('general', 'JJ'),
 ('purpose', 'NN'),
 ('programming', 'NN'),
 ('languages', 'NNS'),
 ('for', 'IN'),
 ('data', 'NNS'),
 ('science', 'NN'),
 ('and', 'CC'),
 ('machine', 'NN'),
 ('learning', 'NN'),
('.', '.'),
('This', 'DT'),
('was', 'VBD'),
('not', 'RB'),
('always', 'RB'),
('the', 'DT'),
('case', 'NN'),
('', 'VBZ'),
('a', 'DT'),
('decade', 'NN'),
('back', 'RB'),
('this', 'DT'),
('thought', 'NN'),
('would', 'MD'),
('have', 'VB'),
('met', 'VBN'),
('a', 'DT'),
('lot', 'NN'),
('of', 'IN'),
('skeptic', 'JJ'),
('eyes', 'NNS'),
('!', '.')]
```

```python
[word for word, tag in text_tagged if tag in ['NN', 'NNS']]
```

```
['years',
 'shift',
 'usage',
 'purpose',
 'programming',
 'languages',
 'data',
 'science',
 'machine',
 'learning',
 'case',
 'decade',
 'thought',
 'lot',
 'eyes']
```

```python
tweets = ["we are more worried about what we can lose than what we
   ↪ feel",
          "it's really cool to say I hate you. But it's not cool to
   ↪ say I love you. Love has a stigma",
          "Instead of doing what you feel you just do what other
   ↪ people think you should do"]
```

```python
tagged_tweets = []
for tweet in tweets:
    tweet_tag = pos_tag(word_tokenize(tweet))
    tagged_tweets.append([tag for word, tag in tweet_tag])
```

```
tagged_tweets[2][:5]
```

```
['RB', 'IN', 'VBG', 'WP', 'PRP']
```

```
from sklearn.preprocessing import MultiLabelBinarizer
```

```
one_hot_multi = MultiLabelBinarizer()
```

```
one_hot_multi.fit_transform(tagged_tweets)
```

```
array([[0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
       [1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0],
       [0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1]])
```

```
one_hot_multi.classes_
```

```
array(['.', 'CC', 'DT', 'IN', 'JJ', 'MD', 'NN', 'NNS', 'PRP', 'RB', '
RBR',
       'TO', 'VB', 'VBG', 'VBP', 'VBZ', 'WP'], dtype=object)
```

0.0.4 CountVectorizer

```
import numpy as np
```

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
text_data = np.array(['I like Cardi B. ', 'Tribeca is a strange
    ↪ place.', ' Germany is where they make volkswagen cars.'])
```

```
count = CountVectorizer()
```

```
bag_of_words = count.fit_transform(text_data)
```

```
count.get_feature_names()
```

```
['cardi',
 'cars',
 'germany',
 'is',
 'like',
 'make',
 'place',
 'strange',
 'they',
 'tribeca',
 'volkswagen',
 'where']
```

```
bag_of_words
```

```
<3x12 sparse matrix of type '<class 'numpy.int64'>'
  with 13 stored elements in Compressed Sparse Row format>
```

```
bag_of_words.toarray()
```

```
array([[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0],
       [0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1]], dtype=int64)
```

```
count.get_feature_names()
```

```
['cardi',
 'cars',
 'germany',
 'is',
 'like',
 'make',
 'place',
 'strange',
 'they',
 'tribeca',
 'volkswagen',
 'where']
```

```
count_2gram = CountVectorizer(ngram_range = (1, 2), stop_words="
    ↪ english",
                              vocabulary=['cardi'])
```

```
bag = count_2gram.fit_transform(text_data)
```

```
bag.toarray()
```

```
array([[1],
       [0],
       [0]])
```

0.0.5 Tfidf

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfidf = TfidfVectorizer()
feature_matrix = tfidf.fit_transform(text_data)
```

```
feature_matrix
```

```
<3x12 sparse matrix of type '<class 'numpy.float64'>'
  with 13 stored elements in Compressed Sparse Row format>
```

```
feature_matrix.toarray()
```

```
array([[0.70710678, 0.        , 0.        , 0.        , 0.70710678,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.40204024, 0.        ,
        0.        , 0.52863461, 0.52863461, 0.        , 0.52863461,
        0.        , 0.        ],
       [0.        , 0.38988801, 0.38988801, 0.29651988, 0.        ,
        0.38988801, 0.        , 0.        , 0.38988801, 0.        ,
        0.38988801, 0.38988801]])
```

```
tfidf.vocabulary_
```

```
{'cardi': 0,
 'cars': 1,
 'germany': 2,
 'is': 3,
 'like': 4,
 'make': 5,
 'place': 6,
 'strange': 7,
 'they': 8,
 'tribeca': 9,
 'volkswagen': 10,
 'where': 11}
```