Working with unstructured texts

October 8, 2019

0.0.1 Environment

```
In [5]: import numpy as np
        import pandas as pd
        PREVIOUS_MAX_ROWS = pd.options.display.max_rows
        pd.options.display.max_rows = 20
        np.random.seed(12345)
        import matplotlib.pyplot as plt
        plt.rc('figure', figsize=(10, 6))
        np.set_printoptions(precision=4, suppress=True)
```

0.0.2 Generate dummy variables

```
In [8]: df=pd.DataFrame({'key':['b','b','a','c','a','b'],
                         'data1':range(6)})
        df  #generates data

Out[8]:   key  data1
        0   b      0
        1   b      1
        2   a      2
        3   c      3
        4   a      4
        5   b      5

In [10]: pd.get_dummies(df['key'])

Out[10]:    a  b  c
         0  0  1  0
         1  0  1  0
         2  1  0  0
         3  0  0  1
         4  1  0  0
         5  0  1  0

In [11]: dummies = pd.get_dummies(df['key'], prefix='key')
         dummies
```

```
Out[11]:    key_a  key_b  key_c
         0      0      1      0
         1      0      1      0
         2      1      0      0
         3      0      0      1
         4      1      0      0
         5      0      1      0
```

```
In [12]: df_with_dummy = df[['data1']].join(dummies)
         df_with_dummy
```

```
Out[12]:    data1  key_a  key_b  key_c
         0      0      0      1      0
         1      1      0      1      0
         2      2      1      0      0
         3      3      0      0      1
         4      4      1      0      0
         5      5      0      1      0
```

0.0.3 String object manipulation

```
In [19]: val='202D Wilson Rd, Pullman, Washington'
         val.split(',')
```

```
Out[19]: ['202D Wilson Rd', ' Pullman', ' Washington']
```

```
In [20]: pieces=[x.strip() for x in val.split(',')]
         pieces
```

```
Out[20]: ['202D Wilson Rd', 'Pullman', 'Washington']
```

```
In [23]: first, second, third=pieces
         'Street:'+first+' City:'+second+' State:'+third
```

```
Out[23]: 'Street:202D Wilson Rd City:Pullman State:Washington'
```

```
In [30]: '::'.join(pieces)
```

```
Out[30]: '202D Wilson Rd::Pullman::Washington'
```

```
In [31]: val
```

```
Out[31]: '202D Wilson Rd, Pullman, Washington'
```

```
In [25]: 'Washington' in val
```

```
Out[25]: True
```

```
In [26]: val.index(',')
```

```
Out[26]: 14
```

```
In [35]: val.find('Wilson') # Wilson starts at 6th character

Out[35]: 5

In [36]: val.count(',')

Out[36]: 2

In [37]: val.replace(',','_')

Out[37]: '202D Wilson Rd_ Pullman_ Washington'

In [39]: val.replace(',','')

Out[39]: '202D Wilson Rd Pullman Washington'
```

0.0.4 Regular Expressions in the string

```
In [43]: import re
         text = "apple    ball\t cat  \tdog"
         re.split('\s+', text)

Out[43]: ['apple', 'ball', 'cat', 'dog']

In [47]: regex=re.compile('\s+') #another way
         regex.split(text)

Out[47]: ['apple', 'ball', 'cat', 'dog']

In [48]: regex.findall(text) #regular text expressions

Out[48]: ['    ', '\t ', '  \t']

In [55]: text = """Dave dave@msn.com
         Steve steve@gmail.com
         Rob rob@gmail.com
         Ryan ryan@yahoo.com
         """
         pattern = r'[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}'
         # re.IGNORECASE makes the regex case-insensitive
         regex = re.compile(pattern, flags=re.IGNORECASE)
         m=regex.search(text)
         m

Out[55]: <re.Match object; span=(5, 17), match='dave@msn.com'>

In [56]: text[m.start():m.end()]

Out[56]: 'dave@msn.com'

In [57]: print(regex.match(text))
```

```
None
```

```
In [58]: print(regex.sub('REDACTED', text))
```

```
Dave REDACTED
Steve REDACTED
Rob REDACTED
Ryan REDACTED
```

```
In [61]: pattern = r'([A-Z0-9._%+-]+)@([A-Z0-9.-]+)\.([A-Z]{2,4})'
         regex = re.compile(pattern, flags=re.IGNORECASE)
         m = regex.match('wesm@bright.net')
         m.groups()
```

```
Out[61]: ('wesm', 'bright', 'net')
```

```
In [62]: regex.findall(text)
```

```
Out[62]: [('dave', 'msn', 'com'),
          ('steve', 'gmail', 'com'),
          ('rob', 'gmail', 'com'),
          ('ryan', 'yahoo', 'com')]
```

```
In [63]: print(regex.sub(r'Username: \1, Domain: \2, Suffix: \3', text))
```

```
Dave Username: dave, Domain: msn, Suffix: com
Steve Username: steve, Domain: gmail, Suffix: com
Rob Username: rob, Domain: gmail, Suffix: com
Ryan Username: ryan, Domain: yahoo, Suffix: com
```

0.0.5 Vectorized string functions using pandas

```
In [64]: data = {'Dave': 'dave@google.com', 'Steve': 'steve@gmail.com',
                 'Rob': 'rob@gmail.com', 'Wes': np.nan}
         data = pd.Series(data)
         data
         data.isnull()
```

```
Out[64]: Dave     False
         Steve    False
         Rob      False
         Wes       True
         dtype: bool
```

```
In [65]: data.str.contains('gmail')
```