Python Code for QSS Chapter 5: Discovery

Kosuke Imai, Python code by Jeff Allen

First Printing

Section 5.1: Textual Data

Section 5.1.1: The Disputed Authorship of 'The Federalist Papers'

Importing textual data into a DataFrame

[ ]:
```python
import pandas as pd
import numpy as np
import glob

# Get a list of all txt files in the federalist directory
file_paths = glob.glob('federalist/*.txt')

# Create an empty list
file_contents = []

# Read txt files into the empty list
for file in file_paths:
    # with: open and close file automatically
    # open(file, 'r'): open file in read mode
    # assign opened file to f
    with open(file, 'r') as f:
        file_contents.append(f.read())

# Take a look at the first 100 characters of essay number 10
file_contents[9][:100]
```

[ ]:
```
'AMONG the numerous advantages promised by a well-constructed Union, none \n
deserves to be mor'
```

[ ]:
```python
# Create a data frame with essay number, a placeholder for author, and the text
federalist = pd.DataFrame({'fed_num': np.arange(1,86), 'author': None,
                           'text': file_contents})

# store authorship information
hamilton = ([1] + list(range(6,10)) + list(range(11, 14)) +
            list(range(15, 18)) + list(range(21, 37)) + list(range(59, 62)) +
            list(range(65, 86)))

madison = [10] + [14] + list(range(37, 49)) + [58]
```

```
jay = list(range(2,6)) + [64]

joint = [18, 19, 20] # Madison and Hamilton

# store conditions for authorship
conditions = [
      federalist['fed_num'].isin(hamilton),
      federalist['fed_num'].isin(madison),
      federalist['fed_num'].isin(jay),
      federalist['fed_num'].isin(joint)
]

choices  = ['Hamilton', 'Madison', 'Jay', 'Joint']

# populate the author column; assign 'Disputed' to unassigned essays
federalist['author'] = np.select(conditions, choices, 'Disputed')

federalist
```

[ ]:
```
    fed_num    author                                              text
0         1  Hamilton  AFTER an unequivocal experience of the ineffic…
1         2       Jay  WHEN the people of America reflect that they a…
2         3       Jay  IT IS not a new observation that the people of…
3         4       Jay  MY LAST paper assigned several reasons why the…
4         5       Jay  QUEEN ANNE, in her letter of the 1st July, 170…
..      …       …                                                 …
80       81  Hamilton  LET US now return to the partition of the judi…
81       82  Hamilton  THE erection of a new government, whatever car…
82       83  Hamilton  THE objection to the plan of the convention, w…
83       84  Hamilton  IN THE course of the foregoing review of the C…
84       85  Hamilton  ACCORDING to the formal division of the subjec…

[85 rows x 3 columns]
```

[ ]:
```
federalist['author'].value_counts()
```

[ ]:
```
author
Hamilton    51
Madison     15
Disputed    11
Jay          5
Joint        3
Name: count, dtype: int64
```

Pre-processing textual data

[ ]:
```
import re # regular expressions
import string # string manipulation
import nltk # natural language toolkit

# Pre-process the text using regular expressions, list comprehensions, apply()

# make lower case and remove punctuation
federalist['text_processed'] = (
    federalist['text'].apply(lambda x: "".join(
        [word.lower() for word in x if word not in string.punctuation])
    )
)

federalist[['text', 'text_processed']].head()
```

[ ]:
```
                                                text  \
0  AFTER an unequivocal experience of the ineffic…
1  WHEN the people of America reflect that they a…
2  IT IS not a new observation that the people of…
3  MY LAST paper assigned several reasons why the…
4  QUEEN ANNE, in her letter of the 1st July, 170…

                                      text_processed
0  after an unequivocal experience of the ineffic…
1  when the people of america reflect that they a…
2  it is not a new observation that the people of…
3  my last paper assigned several reasons why the…
4  queen anne in her letter of the 1st july 1706 …
```

[ ]:
```
# download stopwords: only need to run once
# nltk.download('stopwords')

# save and inspect stopwords
stopwords = nltk.corpus.stopwords.words('english')
stopwords[:10]
```

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

[ ]:
```
# instantiate the Porter stemmer to stem the words
ps = nltk.PorterStemmer()

'''
It is more efficient to define a function to apply to the text column than to
use a lambda function for every step.
'''
def preprocess_text(text):
    # make lower case
    text = text.lower()
    # remove punctuation
    text = "".join([word for word in text if word not in string.punctuation])
    # remove numbers
    text = re.sub('[0-9]+', '', text)
    # create a list of individual tokens, removing whitespace
    tokens = re.split('\W+', text)
    # remove stopwords and any empty strings associated with trailing spaces
    tokens = [word for word in tokens if word !='' and word not in stopwords]
    # finally, stem each word
    tokens = [ps.stem(word) for word in tokens]
    return tokens

# apply function to the text column; no need for lambda with a named function
federalist['text_processed'] = federalist['text'].apply(preprocess_text)

federalist[['text', 'text_processed']].head()
```

[ ]:
```
                                                text  \
0  AFTER an unequivocal experience of the ineffic…
1  WHEN the people of America reflect that they a…
2  IT IS not a new observation that the people of…
3  MY LAST paper assigned several reasons why the…
4  QUEEN ANNE, in her letter of the 1st July, 170…

                                      text_processed
0  [unequivoc, experi, ineffici, subsist, feder, …
1  [peopl, america, reflect, call, upon, decid, q…
2  [new, observ, peopl, countri, like, american, …
3  [last, paper, assign, sever, reason, safeti, p…
4  [queen, ann, letter, st, juli, scotch, parliam…
```

[ ]:
```
# each element of the text_processed column is a list of tokens
type(federalist['text_processed'][0])
```

[ ]:
```
list
```

[ ]:
```
# compare the pre-processed text to the original text for essay number 10
federalist['text_processed'][9][:15]
```

[ ]:
```
['among',
 'numer',
 'advantag',
 'promis',
 'wellconstruct',
 'union',
 'none',
 'deserv',
 'accur',
 'develop',
 'tendenc',
 'break',
 'control',
 'violenc',
 'faction']
```

[ ]:
```
federalist['text'][9][:100]
```

[ ]:
```
'AMONG the numerous advantages promised by a well-constructed Union, none \n
deserves to be mor'
```

Section 5.1.2: Document-Term Matrix

[ ]:
```
from sklearn.feature_extraction.text import CountVectorizer

'''
Instantiate the CountVectorizer and pass the preprocess_text function to the
analyzer argument.
'''
count_vect = CountVectorizer(analyzer=preprocess_text)

# transform the text_processed column into a document-term matrix
dtm = count_vect.fit_transform(federalist['text'])

# the dtm is a sparse matrix
type(dtm)
```

[ ]:
```
scipy.sparse._csr.csr_matrix
```

[ ]:
```
# convert the sparse matrix to a dense matrix and store in a DataFrame
dtm_mat = pd.DataFrame(dtm.toarray(),
                       columns=count_vect.get_feature_names_out())

dtm_mat.iloc[:,:10].head()
```

[ ]:
```
   abandon  abat  abb  abet  abhorr  abil  abject  abl  ablest  abolish
0        0     0    0     0       0     0       0    1       0        0
1        0     0    0     0       0     1       0    0       0        0
2        0     0    0     0       0     0       0    2       0        0
3        0     0    0     0       0     0       0    1       1        0
4        0     0    0     0       0     0       0    0       0        0
```

Section 5.1.3: Topic Discovery

In Progress