Lab 4: Search Terms 2.0 NumPy

Stuart Harley

Introduction: In lab 3 we created a list of the most popular search terms (tokens) for a given set of search queries. As people are imperfect, they often misspell search terms, so we are using a spell checking library to remove and interpret misspellings to find the actual most popular terms (and not just the most popular and consistently spelled terms). The search terms come from Direct Supply's DSSI eProcurement system.

In this lab, we are doing the same thing but we are using numpy and pandas to manipulate the data. This specific notebook uses Numpy.

Learning Outcomes:

- Data importing with Numpy and Pandas
- Cleaning data

Importing Libraries

In [1]:
```
from spellchecker import SpellChecker
import pattern.en
import time
import sys
import numpy as np
import re
import csv
```

Function splits tokens at %20's and spaces

In [2]:
```
def split_token(string):
    return re.split('%20|\s', string)
```

Function to clean strings: Removes surrounding parenthesis from a string and sets the strings to lower case

In [3]:
```
def clean_str(string):
    if(string.startswith('"')):
        string = string[1:]
    if(string.endswith('"')):
        string = string[:-1]
    string = string.lower()
    return string
```

Importing csv data into a 1D numpy array: I ignore the cases where there is a second word separated by a comma in the csv file since it happens only a small percent of the time. I also ignore the header line.

In [4]:
```
%%time
original_tokens = np.genfromtxt('/home/harleys/searchTerms.csv', delimiter=','
, dtype=str, usecols=(0), skip_header=1)
print("bytes in original_tokens: " + str(sys.getsizeof(original_tokens)))
```

```
bytes in original_tokens: 96
CPU times: user 2.4 s, sys: 200 ms, total: 2.6 s
Wall time: 2.53 s
```

Creates a 1D array of all tokens: (splits the tokens at any %20's and spaces)

In [5]:
```
%%time
split_tokens = np.array([elem for singleList in list(map(split_token, original
_tokens)) for elem in singleList])
print("bytes in split_tokens: " + str(sys.getsizeof(split_tokens)))
```

```
bytes in split_tokens: 199758528
CPU times: user 2.49 s, sys: 204 ms, total: 2.69 s
Wall time: 2.69 s
```

Creates a 1D array of cleaned tokens

In [6]:
```
%%time
cleaned_tokens = np.array(list(map(clean_str, split_tokens)))
print("bytes in cleaned_tokens: " + str(sys.getsizeof(cleaned_tokens)))
```

```
bytes in cleaned_tokens: 199758528
CPU times: user 947 ms, sys: 104 ms, total: 1.05 s
Wall time: 1.05 s
```

Creating a numpy 2D array consolidating same search words and keeping track of the number of times they occur

In [7]:
```
%%time
unique, counts = np.unique(cleaned_tokens, return_counts=True)
tokens_count = np.asarray((unique, counts))
print("bytes in tokens_count: " + str(sys.getsizeof(tokens_count)))
```

```
bytes in tokens_count: 16732464
CPU times: user 4.19 s, sys: 397 ms, total: 4.59 s
Wall time: 700 ms
```

Creating an output csv file from the numpy 2D array of tokens

In [8]:
```
np.savetxt('/home/harleys/numpy_all_tokens.csv', tokens_count.T, delimiter=','
, fmt="%s", header='SearchToken, Occurances')
```

Example results of tokens and their number of occurances: In this cell several entries from the beginning and end of the tokens_count numpy 2D array were printed out to illustrate what the csv file looks like.

In [9]:
```
print(tokens_count.T)
```

```
[['' '543']
 ['"' '1']
 ['%' '15']
 ...
 ['zymox' '1']
 ['zyno' '13']
 ['zyrtec' '352']]
```

Creating a new numpy 2D array with only alphabetic tokens

In [10]:
```
%%time
alphabetic_tokens = np.array(list(filter(lambda token : token.isalpha(), clean
ed_tokens)))
unique_alphabetic_tokens, counts = np.unique(alphabetic_tokens, return_counts=
True)
alphabetic_tokens_count = np.asarray((unique_alphabetic_tokens, counts))
print("bytes in alphabetic_tokens_count: " + str(sys.getsizeof(alphabetic_toke
ns_count)))
```

```
bytes in alphabetic_tokens_count: 1820056
CPU times: user 4.02 s, sys: 407 ms, total: 4.43 s
Wall time: 1.26 s
```

Spellchecking the alphabetic tokens and adding the correct spellings to a new 2D array: The first value is the alphabetic token and the second is the predicted correct spelling.

In [11]:
```
%%time
spell = SpellChecker(distance=2)
correct_spelling_tokens = np.array(list(map(lambda token : spell.correction(to
ken), unique_alphabetic_tokens)))
spellchecked_tokens = np.array((unique_alphabetic_tokens, correct_spelling_tok
ens))
print("bytes in spellchecked_tokens: " + str(sys.getsizeof(spellchecked_tokens
)))
```

```
bytes in spellchecked_tokens: 1560064
CPU times: user 23min 49s, sys: 38.7 s, total: 24min 28s
Wall time: 24min 28s
```

Creating an output csv file from the numpy 2D array of tokens and their correct spellings

In [12]:
```
np.savetxt('/home/harleys/numpy_tokens_and_correct_spellings.csv', spellchecke
d_tokens.T, delimiter=',', fmt="%s", header='SearchToken, PossibleCorrectSpell
ing')
```

Example results of tokens and their possible correct spellings: In this cell several entries from the beginning and end of the spellchecked numpy 2D array were printed out to illustrate what the csv file looks like.

In [13]:
```
print(spellchecked_tokens.T)
```

```
[['a' 'a']
 ['aa' 'aa']
 ['aaa' 'aaa']
 ...
 ['zymox' 'ymor']
 ['zyno' 'zeno']
 ['zyrtec' 'zyrtec']]
```

Creating a final 1D array of unique spell checked tokens

In [14]:
```
%%time
unique_correct_spelling_tokens = np.unique(correct_spelling_tokens)
print("bytes in unique_correct_spelling_tokens: " + str(sys.getsizeof(unique_c
orrect_spelling_tokens)))
```

```
bytes in unique_correct_spelling_tokens: 609144
CPU times: user 3.05 ms, sys: 0 ns, total: 3.05 ms
Wall time: 2.66 ms
```

Creating a final dictionary of tokens: In this cell, the final 2D array of spell checked tokens is created. If a token was misspelled and a correct spelling was found, then the number of occurances of the misspelled word is added to the number of occurances of the correctly spelled word. I could not figure out how to do this with only numpy effectively, so I utilized dictionaries.

In [15]:
```
%%time
temp_dict = dict(zip(correct_spelling_tokens, counts))
final_spelled_dict = {}
for token in unique_correct_spelling_tokens:
    final_spelled_dict[token] = 0;
for key in temp_dict.keys():
    final_spelled_dict[key] += temp_dict[key]
print("bytes in final_spelled_dict: " + str(sys.getsizeof(final_spelled_dict
)))
```

```
bytes in final_spelled_dict: 295008
CPU times: user 14.3 ms, sys: 65 µs, total: 14.4 ms
Wall time: 14.2 ms
```

Creating an output csv file from the dictionary of correctly spelled tokens

In [16]:
```
with open('/home/harleys/numpy_correctly_spelled_tokens.csv', 'w') as file:
    writer = csv.DictWriter(file, fieldnames=["SearchToken", "Occurances"])
    writer.writeheader()
    for key in final_spelled_dict.keys():
        file.write("%s,%s\n"%(key,final_spelled_dict[key]))
```

Conclusion

- Conclusion is contained in the Pandas notebook portion of the lab