Lab 4: Search Terms 2.0 Pandas

Stuart Harley

Introduction: In this notebook, I am testing my Pandas implementation of the search term lab with a new list of search terms. The search terms come from a script of a South Park episode.

Learning Outcomes:

- Data importing with Numpy and Pandas
- Cleaning data

Importing Libraries

In [1]:
```
from spellchecker import SpellChecker
import pattern.en
import csv
import time
import sys
import pandas as pd
import re
import numpy as np
pd.options.mode.chained_assignment = None # hides a SettingWithCopy warning la
ter on
```

Function splits tokens at %20's and spaces

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

Importing csv data into a pandas dataframe

In [3]:
```
%%time
original_tokens = pd.read_csv('/home/harleys/All-seasons.csv', usecols=["Line"
], dtype=str)
print("bytes in original_tokens: " + str(sys.getsizeof(original_tokens)))
```

```
bytes in original_tokens: 8514555
CPU times: user 87.9 ms, sys: 15.9 ms, total: 104 ms
Wall time: 104 ms
```

Creates a new dataframe containing all tokens: (this splits the tokens at spaces, commas, or %20's.)

In [4]:
```
%%time
tokens = original_tokens["Line"].to_numpy()
all_tokens = pd.DataFrame([elem for singleList in list(map(split_token, tokens
)) for elem in singleList])
all_tokens.columns = ["SearchTerm"]
print("bytes in all_tokens: " + str(sys.getsizeof(all_tokens)))
```

```
bytes in all_tokens: 59262505
CPU times: user 696 ms, sys: 32.1 ms, total: 728 ms
Wall time: 727 ms
```

Adds a column in the dataframe containing the tokens in lower case

In [5]:
```
%%time
all_tokens["Lowercase"] = all_tokens["SearchTerm"].map(lambda token: token.low
er())
print("bytes in lowercase column: " + str(sys.getsizeof(all_tokens["Lowercase"
])))
```

```
bytes in lowercase column: 58999377
CPU times: user 544 ms, sys: 12.1 ms, total: 557 ms
Wall time: 555 ms
```

Creates a new dataframe containing unique tokens and their number of occurances

In [6]:
```
%%time
unique_tokens = all_tokens["Lowercase"].value_counts().to_frame()
unique_tokens = unique_tokens.reset_index()
unique_tokens.columns = ["SearchTerm", "Occurances"]
print("bytes in unique_tokens dataframe: " + str(sys.getsizeof(unique_tokens
)))
```

```
bytes in unique_tokens dataframe: 3789720
CPU times: user 146 ms, sys: 157 µs, total: 146 ms
Wall time: 144 ms
```

Creating an output csv file from the database of the unique tokens and counts

In [7]:
```
unique_tokens.to_csv('/home/harleys/pandas_all_tokens_SP.csv', index=False)
```

Example results of tokens and their number of occurances: In this cell several entries from the unique_tokens dataframe are printed out to illustrate what the csv file looks like.

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

Out[8]:

	SearchTerm	Occurances
0		159314
1	the	24874
2	you	22813
3	to	19510
4	i	17280

Creates a new dataframe containing only the alphabetic tokens and their number of occurances

In [9]:
```
%%time
unique_alpha_tokens = unique_tokens[unique_tokens["SearchTerm"].str.isalpha()]
print("bytes in unique_alpha_tokens dataframe: " + str(sys.getsizeof(unique_alpha_tokens)))
```

```
bytes in unique_alpha_tokens dataframe: 1501615
CPU times: user 26.5 ms, sys: 163 µs, total: 26.6 ms
Wall time: 25.9 ms
```

Spellchecking the alphabetic tokens and adding the possible correct spelling as a new column: This code originally threw a SettingWithCopy warning, but I determined that the code acted as intended so I hid the warning.

In [10]:
```
%%time
spell = SpellChecker(distance=2)
unique_alpha_tokens["CorrectSpelling"] = unique_alpha_tokens["SearchTerm"].map(lambda token: spell.correction(token))
unique_alpha_tokens = unique_alpha_tokens[["SearchTerm", "CorrectSpelling", "Occurances"]]
unique_alpha_tokens = unique_alpha_tokens.reset_index(drop=True)
print("bytes in dataframe columns with tokens and their spellchecked versions: " + str(sys.getsizeof(unique_alpha_tokens["SearchTerm"]) + sys.getsizeof(unique_alpha_tokens["CorrectSpelling"])))
```

```
bytes in dataframe columns with tokens and their spellchecked versions: 2399202
CPU times: user 21min 53s, sys: 38.8 s, total: 22min 31s
Wall time: 22min 32s
```

Creating an output csv file from the dataframe of tokens and their possible correct spellings

In [11]:
```
unique_alpha_tokens.to_csv('/home/harleys/pandas_tokens_and_correct_spellings_SP.csv', index=False, columns=["SearchTerm", "CorrectSpelling"])
```

Example results of tokens and their possible correct spellings: In this cell several entries from the unique_alpha_tokens dataframe are printed out to illustrate what the csv file looks like.

In [12]:
```
unique_alpha_tokens.iloc[0:5, 0:2]
```

Out[12]:

	SearchTerm	CorrectSpelling
0	the	the
1	you	you
2	to	to
3	i	i
4	a	a

Creating a final dataframe of unique spell checked tokens

In [13]:
```
%%time
unique_correct_spelling_tokens = pd.DataFrame(unique_alpha_tokens["CorrectSpel
ling"].unique())
unique_correct_spelling_tokens.columns = ["SearchTerm"]
print("bytes in unique_correct_spelling_tokens: " + str(sys.getsizeof(unique_c
orrect_spelling_tokens)))
```

```
bytes in unique_correct_spelling_tokens: 1065230
CPU times: user 14.3 ms, sys: 3.95 ms, total: 18.3 ms
Wall time: 17.5 ms
```

Creating a final dictionary of tokens: In this cell, the final dictionary 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.

In [14]:
```
%%time
temp_df = unique_alpha_tokens.drop(columns=["SearchTerm"])
final_spelled_dict = {}
row_index = 0
for token in unique_correct_spelling_tokens["SearchTerm"]:
    final_spelled_dict[token] = 0;
for token in temp_df["CorrectSpelling"]:
    final_spelled_dict[token] += temp_df.loc[row_index, "Occurances"]
    row_index += 1
print("bytes in final_spelled_dict: " + str(sys.getsizeof(final_spelled_dict
)))
```

```
bytes in final_spelled_dict: 589928
CPU times: user 127 ms, sys: 3.84 ms, total: 131 ms
Wall time: 129 ms
```

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

In [15]:
```
with open('/home/harleys/pandas_correctly_spelled_tokens_SP.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

- The dataset in this notebook is a file of lines from an episode of south park.
- This file is about a third of the size of the search terms file that we were using previously. However, I don't think there will be many misspelled words in this file so I think it will be considerably quicker. I estimate that the cell running the spell check should complete in about 5 minutes (compared to 24 minutes of the searchterms file).
- The actual runtime for this file was 22.5 minutes.
- My hypothesis was pretty far off. I think that the average word size of this file was less which would mean there are more words per byte which caused me to underestimate. Also, there was greater diversity in the words in the file which also caused me to underestimate since I did not take that into account.
- In conclusion, the size of a file can give you an idea of how long it can take a function to run, but you won't know for sure until you actually test it.