Assignment 6

March 5, 2020

0.1 Book Analysis

This function accepts the a book in the form of a named text file as input. The function develops a number of results useful in analyzing the book. The results include some simple numbers as well as lists and sets created from the book.

The function returns a dictionary including all of the results. Using dictionaries created by the function from different books allows us to compare the books analytically.

[2]:
```
import string
import numpy as np

def bookAnalysis(book):

    inFile = open(book,"r") # Specify source documents

    # Create a base translation table to eliminate punctuation.

    tt = {}

    # Eliminate punctuation
    for char in string.punctuation:
        tt[ord(char)] = 32

    # Eliminate digits
    digits = {'0','1','2','3','4','5','6','7','8','9'}
    for char in digits:
        tt[ord(char)] = 32


    # Initialize empty list to hold lines and set count of lines to 0.

    lines = []
    nlines = 0

    # Iterate through the lines in the source document
```

```
for line in inFile:
    if  len(line) > 1: # Avoid blank lines

        # Add any non ascii characters to translation table

        for char in line:
            if ord(char) > 127: ## last printable ACSII character
                tt[ord(char)] = 32 ## ASCII blank space

        nlines = nlines + 1 # Increment count of lines
        lines.append(line)  # Add current line to list of lines




# Build a list of words from the list of lines

words = []
problem_lines = []
for line in lines:
    lineOK = True
    line = line.lower()
    words_line = line.split()
    words = words + words_line

# Create a dictionary to capture the counts of the unique words in the text.

count_dict = {}
for word in words:
    count_dict[word] = count_dict.get(word,0) + 1

# Get a list of the unique values (counts) in the dictionary.

counts = []
for count in count_dict.values():
    if count not in counts:
        counts.append(count)

# Create a list of the 100 largest values in counts

counts.sort()               # Sort the counts
counts.reverse()            # Convert to descending order
top_counts = counts[:100]
```

```
    # Build a list of lists of words in each place. Note that we have to allow
↪for ties, so there is a list of words in each place.

    place_lists = []
    for i in range(50):
        count = top_counts[i]
        place_list = []
        for key in count_dict.keys():
            if count_dict[key] == count:
                place_list.append(key)
        place_lists.append(place_list)


    unique_words = list(count_dict.keys())

    No_unique_words = len(unique_words)
    Total_no_words = len(words)
    Ratio_unique_total = No_unique_words/Total_no_words
    Ratio_total_unique = Total_no_words/No_unique_words

    lengths = []
    for word in words:
        lengths.append(len(word))

    cnt = list(count_dict.values())
    mean_word_length = np.mean(cnt)

    Out_dict = {}
    Out_dict['unique_words'] = unique_words
    Out_dict['No_unique_words'] = No_unique_words
    Out_dict['Total_no_words'] = Total_no_words
    Out_dict['Ratio_unique_total'] = Ratio_unique_total
    Out_dict["Word_list"] = words
    Out_dict['count_dict'] = count_dict
    Out_dict['Median uses per word'] = np.median(cnt)
    Out_dict["Book"] = book
    Out_dict["place_lists"] = place_lists
    Out_dict["lengths"] = lengths
    Out_dict["mean_word_length"] = np.mean(lengths)
    Out_dict["tt"] = tt
    return(Out_dict)
```

0.2 Task 1

There are four books to analyze.

Two are by Dostoyevsky:

- Crime and Punishment
- Notes from the Underground

Two are by Austen:

- Persuasion
- Pride and Prejudice

Create dictionaries for each of these books. Use the names d1dict, d2dict, a1dict, and a2dict to capture the identity of the author in the names of the dictionaries.

[3]:
```
d1dict = bookAnalysis("Crime and Punishment.txt")
d2dict = bookAnalysis("Notes from the Underground.txt")
a1dict = bookAnalysis("Persuasion.txt")
a2dict = bookAnalysis("Pride and Prejudice.txt")
```

0.3 Task 2

Create sets of words appearing in each of the books by converting the lists of unique words in each of the dictionaries. Label these sets d1words, d2words, a1words, and a2words.

[4]:
```
d1words = set(d1dict['unique_words'])
d2words = set(d2dict['unique_words'])
a1words = set(a1dict['unique_words'])
a2words = set(a2dict['unique_words'])
```

0.4 Task 3

Craete a set too_common as the intersection of all of these sets of words. These words would not be useful in identifying the authorship of an unknown work. Produce a sorted list of the words in this set and display the first 20.

[5]:
```
too_common = a1words & a2words & d1words & d2words
sltoo_common = sorted(too_common)
for w in sltoo_common[:20]:
    print(w)
```

```
(and
(for
a
able
about
about,
above
abroad
absent,
absolute
absolutely
```

```
absurd
absurdity
abuse
abused
accept
accepted
accepting
accompanied
accompanying
```

0.5 Task 4

Create sets d1_sig, d2_sig, a1_sig, a2_sig by removing the words in too_common from the corresponding original sets.

[6]:
```
d1_sig = d1words - too_common
d2_sig = d2words - too_common
a1_sig = a1words - too_common
a2_sig = a2words - too_common
```

0.6 Task 5

Create sets asig and dsig as the intersections of the sets from the individual books. These two set should be useful in idntifying the authorship of a book written by one of these authors. Print the length of each of these sets and their intersection.

[7]:
```
asig = a1_sig & a2_sig
dsig = d1_sig & d2_sig

print(len(dsig))
print(len(asig))
print(len(asig & dsig))
```

```
2295
2988
0
```

0.7 Task 6

Use bookAnalysis to produce UKdict from the "unknown author" file UK2.txt.

[8]:
```
UKdict = bookAnalysis("UK2.txt")
```

0.8 Task 7

Create the set of words in UK2 from the dictionary. Subtract the words in too_common. Then create UKd and UKa by intersecting the UK set with the signature sets of words from our two authors.

[9]:
```
UKwords = set(UKdict['unique_words']) - too_common

UKd = UKwords & dsig
UKa = UKwords & asig
```

0.9 Task 8

Compute the fractions of words from the unknown author found in each each of these intersections. Does this suggest the identity of the author?

[10]:
```
print(len(UKd)/len(UKwords))
print(len(UKa)/len(UKwords))
```

```
0.0635109066627096
0.16938714942869862
```