DOKUZ EYLÜL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

E-BOOK ANALYSIS AND REPRESENTATION

Assignment Report

by
R.Hakan Cankul

January 2020
İZMİR

Contents

1 INTRODUCTION

In this project, we developed an e-book downloader with word frequency counters. The project basically, consists of 5 part: finding correct book from query, downloading pdf as txt file, reading txt with correct pre-processings, frequency calculation, and displaying. I personally believe, the most different topics that I implemented differently are using very smart and rare library for pdf to text editor, using regular expressions for scraping and pre-processing and using google infrastructure for finding correct books. About all 3 strategy, the details will be given at the below.

2 METHODOLOGY

2.1 Structure of Your Project

In this project following libraries is used, re, nltk, request, bs4, webbrowser, numpy, pandas and pdftotext. Except for pdftotext all of the python libraries is widely used. The request, bs4 libraries is used for finding correct pdf file within the web. The pandas and numpy is used for more aesthetic representation of words. The re library is (regular expressions) used for pre-processing of words. The nltk library is used for removing stopwords and counting frequencies of words. Lastly, the pdftotext library is used for saving pdf file as text file.

As data structures, python list, python string, ntlk freqlist and pandas dataframe have been used. The lists and strings are convertible structures in natural language processing, especially in tokenization. After, saving all book as a single string, we used freqlist to store frequencies. Lastly, convert it to pandas for displaying results.

The finding correct pdf upon web and removing stopwords are the most critical algorithms that I developed. For clarity, simplicity and time performance freqdist and pdftotext.PDF method have been used, besides less important built-in methods like str.lower, request.get or dataframe.to_string.

The algorithm for webscraping:

To use google infrastructure, the following pseudo-code have been used which you can find in source code more detailed way.

SEARCH KEY WORDS = [‘wikibooks’, ‘wikisource’]
→ Form a query for google like google.com/search?q+ keyword + bookname.
→ Get the resulting pages (that page would be regular google search page)
→ Get first link in the above page (that will give you, the first link that google suggest) (functions as I’m feeling Lucky button)
→ Handle redirect links
→ Get the page url, it supposed to be wikipedia website.
→ Parse the above url to check if it is wikibooks or not by regex.
→ If it is wikibooks, find the button that redirects PDF. And save that pdf url for downloading and saving as text.
→ Otherwise, check for next keyword.

The algorithm for removing StopWords.

→ download nltk stopwords for all language.
→ open the book that you downloaded and saved, readlines.
→ merge all of the sentences as a single string.
→ lower the string and delete newlines (\n)
→ create regex tokenizer to parse intended words
→ check if these words are in stopword list or not.
→ if not, save it as a new array, convert it into string again.
→ put that string into nltk.FreqDist method.

2.2 Encountered Problems and Solutions

The most crucial 3 problem that I encounter were, finding correct book, converting pdf to text and displaying. To solve how to find correct book, I used basic google search method which I explained above. The rest of two solutions are below.

The PDF's are like solid structures which is very hard to parse, manipulate or extract information. Therefore, I spent most of my time to how save them as string. First, sub-problem was about the characters. Some string consist unreadable characters for the machine. Therefore, I used utf-8 encoding at many places in my code. The second problem is extracting only alphabetical information. I tried some libraries until the pdftotext that is developed by some student. I am aware of that library is very rare to implement therefore, I also put downloading links into source code. In case, it is forbidden to use, or tester could not download and test this library, I put some manual tests at the below of the code that you can easily check rest of the parts functions correctly. I took full responsibility, if any point reduction occurs about that part.

The second problem was displaying. The nltk.freqdist data structure needs some investigation to use by un-familiar student. Therefore, I imported two different libraries too, pandas and numpy which are essentials of data science. For getting common or distict words were easy with freqdist data structure because it makes us enable to just substract and add these lists. However, for better visualization I had to convert them into pandas.DataFrame at the end of the code.

2.3 Improvements

It is asked to do in assignment paper but I asked user's two extra different questions. How many book will you select 1 or 2? and How many words you would like to display at the end of the execution.

Also, for ease to tester, I direct the pdftotext github page for downloading. For getting many errors I also used try and except block. Lastly, after downloading english stopwords, I cached them into an array, so code execution will be better at time performance.

Last improvement was regular expression. Most of the words need many pre-process to get valuable information. In this project numbers and signs considered non-valuable information by my decision. Therefore, I used regex for parse and extract the one solid words. Thanks to the fact that regex are super fast, it could be considered as improvement.

3 EXPERIMENTATION

I personally experiment/test the provided two books Non-Programmer's Tutorial for Python 2.6 and Non-Programmer's Tutorial for Python 3 in wikibooks. Even if you could download, pdftotext or not, additional tests could give you proper results. The code will work on both one book or two book. The distinct words and common words also executes successfully. For downloading pdftotext, I also add some breach y/n to direct you to browser or built-in tests.

4 CONCLUSION

I believe I learned the web scraping in detail. Also, learning regex for data processing and nltk toolbox was an extra for me. In addition, most of the computer engineers use the tools that others develop. The pdftotext could be considered the hard library to implement therefore, to merge my project with it, was surely gave me experience on large project later.

APPENDIX A: CODE

Due to mass amonth of comments reading the source code from appendix must be hard. Please check .py file for readability.

```
import requests,bs4,webbrowser,sys,time
import re
import nltk

from nltk.corpus import stopwords # For removing stopwords
nltk.download('stopwords') # If you did not download stopwords before, it will.
from nltk.tokenize import RegexpTokenizer # Just tokenizer for wide usage
import numpy as np
import pandas as pd

TEST = False  # This global variable exist for built-in tests or user based test.
# 0 for user based. 1 for built-in tests.

#   /$$   /$$           /$$                                  /$$$$$$                  /$$                 /$$
#  | $$  | $$          | $$                                 /$$__  $$                | $$                | $$
#  | $$  | $$  /$$$$$$ | $$   /$$  /$$$$$$  /$$$$$$$       | $$  \__/  /$$$$$$  /$$$$$$$  /$$$$$$  /$$$$$$$  /$$$$$$  /$$$$$$$
#  | $$$$$$$$ |____  $$| $$  /$$/ |____  $$| $$__  $$      | $$       |____  $$| $$__  $$ |____  $$| $$__  $$ /$$__  $$| $$__  $$
#  | $$__  $$  /$$$$$$$| $$$$$$/   /$$$$$$$| $$  \ $$      | $$        /$$$$$$$| $$  \ $$  /$$$$$$$| $$  \ $$| $$$$$$$$| $$  \ $$
#  | $$  | $$ /$$__  $$| $$_  $$  /$$__  $$| $$  | $$      | $$    $$ /$$__  $$| $$  | $$ /$$__  $$| $$  | $$| $$_____/| $$  | $$
#  | $$  | $$|  $$$$$$$| $$ \  $$|  $$$$$$$| $$  | $$      |  $$$$$$/|  $$$$$$$| $$  | $$|  $$$$$$$| $$  | $$|  $$$$$$$| $$  | $$
#  |__/  |__/ \_______/|__/  \__/ \_______/|__/  |__/       \______/  \_______/|__/  |__/ \_______/|__/  |__/ \_______/|__/  |__/

numberOfBook = int(input("How many books will you download (1 or 2): "))
# To choose how many books will I display.
print()
bookname1 = input("Book1: ")

# CAREFUL!!
# The best way to find e-books on internet is basically surfing on google with spefic keywords
# Below keywords is for your searching query. In this homework just wikibook and wikisource.
# Assume you want to download Hamlet in wikibook, then you need to search wikibook hamlet in google.
searchKeywords = ['wikibook','wikisource']

# Below you search in google respectively in wikibook and wikisource. If the book is not find in wikibooks
# Then, it will try wikisource.
for keyword in searchKeywords:
    # var: searching guery in google however, it directs you to re-direct page.
    var = requests.get(r'http://www.google.com/search?q='+keyword+' '+bookname1+'&btnI')
    # This page is about where to direct.
```

```
40.     redirect = var.url
41.     raw = requests.get(redirect)
42.     soup = bs4.BeautifulSoup(raw.text, "html.parser")
43.     # You basically parse all 'a' symbols because it shows html urls.
44.     # We use find because, first result would be most relevant link.
45.     # Think of I'm feeling lucky button.
46.     lucky = soup.find('a').text
47. 
48.     # You lucky would be like https://en.wikibooks.org/wiki/hamlet
49.     # To test if it is in wikibook or not we simply use regex.
50.     # \.[\w]\. is finds all words between two dot. in this case only '.wiki
    books.'
51.     # [1:-1] deletes first and last character .wikibooks. => wikibooks
52.     x = re.findall(r'\.[\w]+\.',lucky)[0][1:-1]
53.     # If the link in wikibooks then simply break. Otherwise, loop again for
     wikisource.
54.     if(x == 'wikibooks'):
55.         break
56. 
57. # After getting lucky link we need to parse the wikibook webpage.
58. # To find pdf page, we need to find pdf url.
59. 
60. raw_page = requests.get(lucky)
61. wiki_page = raw_page.content
62. 
63. # urf-
    8 for just taking appoirate character. \x08 like chars is out of context
64. soup2 = bs4.BeautifulSoup(wiki_page.decode('utf-8'), "html.parser")
65. 
66. # I manually inspect the website and pdf link is within the html statement

67. #  <a class='internal' href='PDF_LINK' >PDF Version</a>
68. book_url = soup2.find_all('a', {"class": "internal"})[0]['href'][2:]
69. book_url = 'https://' + book_url
70. 
71. # VOLAA ! We found pdf of book. Webscraping part is done.
72. print("The url of the book is:",book_url)
73. 
74. # The next thing to do, save the pdf as txt. Therefore, I used external lib
    rary called pdftotext.
75. # It will take pdf from above url and save it with the same name that you q
    ueried above.
76. try:
77.     import pdftotext
78.     # Load your PDF
79.     with open(book_url, "rb") as f:
80.         pdf = pdftotext.PDF(f)
81. 
82.     # Save all text to a txt file.
83.     with open(bookname1 +'.txt', 'w') as f:
84.         f.write("\n\n".join(pdf))
85. except:
86.     # This is just an error message because I am aware of pdftotext is not
    widely used.
87.     # Therefore, I put links to download, if may take a while.
88.     # If you do not want to install and trust me, I provide extra 2 .txt fi
    le for built-in tests.
89.     # You can simply press 'n' to go manual tests otherwise, it will open d
    ownload page.
90.     print()
91.     print("Looks like you did not download required library pdftotext. Plea
    se install it from below link")
92.     print("https://github.com/jalan/pdftotext")
```

```
93.     print("Installing can take a while.")
94.     print("Or in the below, you can try the tests with pre-
    installed books which the developer provides")
95.     x = input("To install please enter 'y', to forward tests please enter '
    n': ")
96.     if(x == "y"):
97.         try:
98.             import webbrowser
99.             webbrowser.open("https://github.com/jalan/pdftotext")
100.                exit()
101.            except:
102.                exit()
103.        elif(x == "n"):
104.            TEST = True
105.            print('\nTEST STARTED...\n')
106.        else:
107.            print("Incorrect button. Shutting Down.")
108.            exit()
109.
110.
111.    # If you choose 2 book and after first book all libraries are correc
    tly installed.
112.    # We can take second book with simply the same stradegy.
113.    if(numberOfBook == "2"):
114.        bookname2 = input("Book2: ")
115.        searchKeywords = ['wikibook','wikisource']
116.
117.        for keyword in searchKeywords:
118.            var = requests.get(r'http://www.google.com/search?q='+keywor
    d+' '+bookname2+'&btnI')
119.            redirect = var.url
120.            raw = requests.get(redirect)
121.            soup = bs4.BeautifulSoup(raw.text, "html.parser")
122.            lucky = soup.find('a').text
123.            x = re.findall(r'\.[\w]+\.',lucky)[0][1:-1]
124.            if(x == 'wikibooks'):
125.                break
126.        raw_page = requests.get(lucky)
127.        wiki_page = raw_page.content
128.        soup2 = bs4.BeautifulSoup(wiki_page.decode('utf-
    8'), "html.parser")
129.
130.        book_url = soup2.find_all('a', {"class": "internal"})[0]['href']
    [2:]
131.        book_url = 'https://' + book_url
132.
133.        # Load your PDF
134.        with open(book_url, "rb") as f:
135.            pdf = pdftotext.PDF(f)
136.
137.        # Save all text to a txt file.
138.        with open(bookname2 +'.txt', 'w') as f:
139.            f.write("\n\n".join(pdf))
140.
141.    # If you downloaded the books without trouble, in the current folder
    ,
142.    # there should be THEBOOKNAMEYOUENTERED.txt. Then we needed to parse
     all words.
143.    if(TEST == False):
144.        if(numberOfBook == 1):
145.            # If only one book is selected... Simply open it in read mod
    e utf-8.
146.            book1 = open(bookname1 + '.txt','r',encoding=('utf-8'))
```

```
147.                b1 = book1.readlines()
148.                book1.close()
149.                # First we stringfy all sentences in list array.
150.                # We need to lower all words and delete \n (new lines)
151.                # "Hello World!\n" => "hello world"
152.                bookstring1 = " ".join(b1).lower().replace("\n", "")
153.                text = bookstring1
154.
155.                # we are using Regular Expression Tokenizer for personally,
    most accurate preprocesing
156.                # For comparing words we needed to eliminate signs (+, -
    , ') or numbers (3.6, 5) etc.
157.                # Also, we need to count each of cat, cat's, non-
    cat as one.
158.                # That is why we use regex with [A-Za-
    z]+ which will take each word.
159.                tokenizer = RegexpTokenizer(r'[A-Za-z]+')
160.                text_tokens = tokenizer.tokenize(text)
161.
162.                # For performance, we cached stopwords one.
163.                cachedStopwords = stopwords.words('english')
164.                # Below, we shrink all statement block in below, into one li
    ne.
165.                # tokens_without_sw = []
166.                # for word in text_tokens:
167.                #   if(word not in cachedStopwords):
168.                #       tokens_without_sw.append(word)
169.                tokens_without_sw = [word for word in text_tokens if not wor
    d in cachedStopwords]
170.
171.                # Then we convert list to string and found frequency distrib
    utions
172.                text1 = " ".join(tokens_without_sw)
173.                fdist1 = nltk.FreqDist(tokens_without_sw)
174.
175.                # we get how many words you want to display
176.                n = int(input("How many word you want to display: "))
177.
178.                # Some manipulations for more aestatic table
179.                df = pd.DataFrame(fdist1.most_common(n))
180.                df = df.rename(columns={0: "WORD", 1: "FREQ_1"})
181.
182.                df['NO'] = np.array(range(1,n+1))
183.                df = df[['NO','WORD','FREQ_1']]
184.
185.                # VOLAA !!
186.                print(df.to_string(index=False))
187.
188.            if(numberOfBook == 2):
189.                book1 = open(bookname1 + '.txt','r',encoding=('utf-8'))
190.                b1 = book1.readlines()
191.                book1.close()
192.                bookstring1 = " ".join(b1).lower().replace("\n", "")
193.                text = bookstring1
194.                tokenizer = RegexpTokenizer(r'[A-Za-z]+')
195.                text_tokens = tokenizer.tokenize(text)
196.
197.                cachedStopwords = stopwords.words('english')
198.                tokens_without_sw = [word for word in text_tokens if not wor
    d in cachedStopwords]
199.
200.                text1 = " ".join(tokens_without_sw)
201.                fdist1 = nltk.FreqDist(tokens_without_sw)
```

```
                book2 = open(bookname2 + '.txt','r',encoding=('utf-8'))
                b2 = book2.readlines()
                book2.close()

                bookstring2 = " ".join(b2).lower().replace("\n", "")

                text2 = bookstring2
                text_tokens2 = tokenizer.tokenize(text2)

                tokens_without_sw2 = [word for word in text_tokens2 if not w
    ord in cachedStopwords]
                text2 = " ".join(tokens_without_sw2)
                fdist2 = nltk.FreqDist(tokens_without_sw2)
                n = int(input("How many word you want to display: "))
                df = pd.DataFrame(fdist1.most_common(n))
                df2 = pd.DataFrame(fdist2.most_common(n))

                # Thanks to freqdist, If we have 2 booko to compare,
                # we can easily subsract or add them to find common or disti
    nc words.
                df3 = pd.DataFrame((fdist1 + fdist2).most_common(n))
                df4 = pd.DataFrame((fdist1 - fdist2).most_common(n))
                df5 = pd.DataFrame((fdist2 - fdist1).most_common(n))

                # Rest of them is also table manipulation.
                df = pd.DataFrame(fdist1.most_common(n))
                df = df.rename(columns={0: "WORD", 1: "FREQ_1"})

                df['NO'] = np.array(range(1,n+1))
                df = df[['NO','WORD','FREQ_1']]
                print(df.to_string(index=False))

                print("\n")
                df2 = pd.DataFrame(fdist2.most_common(n))
                df2 = df2.rename(columns={0: "WORD", 1: "FREQ_2"})

                df2['NO'] = np.array(range(1,n+1))
                df2 = df2[['NO','WORD','FREQ_2']]
                print(df2.to_string(index=False))

                print("\n")

                df = pd.DataFrame(fdist1.most_common(n))
                df2 = pd.DataFrame(fdist2.most_common(n))
                df3 = pd.DataFrame((fdist1 + fdist2).most_common(n))


                d = pd.DataFrame()
                d['NO'] = np.array(range(1,n+1))
                d['WORD'] = df3[0]
                d['FREQ_1'] = df[1]
                d['FREQ_2'] = df2[1]
                d['FREQ_SUM'] = df3[1]
                print(d.to_string(index=False))
                print("\n")

                df4 = df4.rename(columns={0: "WORD", 1: "FREQ_1"})
                df4['NO'] = np.array(range(1,n+1))
                df4 = df4[['NO','WORD','FREQ_1']]
                print(df4.to_string(index=False))
                print("\n")

                df5 = df5.rename(columns={0: "WORD", 1: "FREQ_2"})
```

```
                df5['NO'] = np.array(range(1,n+1))
                df5 = df5[['NO','WORD','FREQ_2']]
                print(df5.to_string(index=False))
                print("\n")

        #$$$$$$$$\ $$$$$$$$\  $$$$$$\ $$$$$$$$\
        #\__$$  __|$$  _____|$$  __$$\\__$$  __|
        #   $$ |   $$ |      $$ /  \__|  $$ |
        #   $$ |   $$$$$\    \$$$$$$\    $$ |
        #   $$ |   $$  __|    \____$$\   $$ |
        #   $$ |   $$ |      $$\   $$ |  $$ |
        #   $$ |   $$$$$$$$\ \$$$$$$  |  $$ |
        #   \__|   \________| \______/   \__|

        ## WARNING!! ##

        # These part is identically same with above if statement so any more comment is not necessary.
        # If you press n above, these statements will be executed.
        if(TEST == True):
            if(numberOfBook == 1):
                print("BOOK 1: Non-Programmer's Tutorial for Python 2.6")
                book1 = open('Non-Programmer\'s_Tutorial_for_Python_2.6.txt','r',encoding=('utf-8'))
                b1 = book1.readlines()
                book1.close()
                bookstring1 = " ".join(b1).lower().replace("\n", "")
                text = bookstring1
                tokenizer = RegexpTokenizer(r'[A-Za-z]+')
                text_tokens = tokenizer.tokenize(text)

                cachedStopwords = stopwords.words('english')
                tokens_without_sw = [word for word in text_tokens if not word in cachedStopwords]

                text1 = " ".join(tokens_without_sw)
                fdist1 = nltk.FreqDist(tokens_without_sw)
                n = int(input("How many word you want to display: "))

                df = pd.DataFrame(fdist1.most_common(n))
                df = df.rename(columns={0: "WORD", 1: "FREQ_1"})

                df['NO'] = np.array(range(1,n+1))
                df = df[['NO','WORD','FREQ_1']]
                print(df.to_string(index=False))

            if(numberOfBook == 2):
                print("BOOK 1: Non-Programmer's Tutorial for Python 2.6")
                print("BOOK 2: Non-Programmer's Tutorial for Python 3")
                book1 = open('Non-Programmer\'s_Tutorial_for_Python_2.6.txt','r',encoding=('utf-8'))
                b1 = book1.readlines()
                book1.close()
                bookstring1 = " ".join(b1).lower().replace("\n", "")
                text = bookstring1
                tokenizer = RegexpTokenizer(r'[A-Za-z]+')
                text_tokens = tokenizer.tokenize(text)

                cachedStopwords = stopwords.words('english')
                tokens_without_sw = [word for word in text_tokens if not word in cachedStopwords]

                text1 = " ".join(tokens_without_sw)
```

```
                fdist1 = nltk.FreqDist(tokens_without_sw)

                book2 = open('Non-Programmer\'s_Tutorial_for_Python_3.txt','r',encoding=('utf-8'))
                b2 = book2.readlines()
                book2.close()

                bookstring2 = " ".join(b2).lower().replace("\n", "")

                text2 = bookstring2
                text_tokens2 = tokenizer.tokenize(text2)

                tokens_without_sw2 = [word for word in text_tokens2 if not word in cachedStopwords]
                text2 = " ".join(tokens_without_sw2)
                fdist2 = nltk.FreqDist(tokens_without_sw2)
                n = int(input("How many word you want to display: "))
                df = pd.DataFrame(fdist1.most_common(n))
                df2 = pd.DataFrame(fdist2.most_common(n))
                df3 = pd.DataFrame((fdist1 + fdist2).most_common(n))
                df4 = pd.DataFrame((fdist1 - fdist2).most_common(n))
                df5 = pd.DataFrame((fdist2 - fdist1).most_common(n))

                df = pd.DataFrame(fdist1.most_common(n))
                df = df.rename(columns={0: "WORD", 1: "FREQ_1"})

                df['NO'] = np.array(range(1,n+1))
                df = df[['NO','WORD','FREQ_1']]
                #print(df.to_string(index=False))

                print("\n")
                df2 = pd.DataFrame(fdist2.most_common(n))
                df2 = df2.rename(columns={0: "WORD", 1: "FREQ_2"})

                df2['NO'] = np.array(range(1,n+1))
                df2 = df2[['NO','WORD','FREQ_2']]
                #print(df2.to_string(index=False))

                print("\n")

                df = pd.DataFrame(fdist1.most_common(n))
                df2 = pd.DataFrame(fdist2.most_common(n))
                df3 = pd.DataFrame((fdist1 + fdist2).most_common(n))


                d = pd.DataFrame()
                d['NO'] = np.array(range(1,n+1))
                d['WORD'] = df3[0]
                d['FREQ_1'] = df[1]
                d['FREQ_2'] = df2[1]
                d['FREQ_SUM'] = df3[1]
                print("COMMON WORDS")
                print(d.to_string(index=False))
                print("\n")

                print("BOOK 1: Non-Programmer's Tutorial for Python 2.6")
                print("DISTINCT WORDS")
                df4 = df4.rename(columns={0: "WORD", 1: "FREQ_1"})
                df4['NO'] = np.array(range(1,n+1))
                df4 = df4[['NO','WORD','FREQ_1']]
                print(df4.to_string(index=False))
                print("\n")

```

```
382.                 print("BOOK 2: Non-Programmer's Tutorial for Python 3")
383.                 print("DISTINCT WORDS")
384.                 df5 = df5.rename(columns={0: "WORD", 1: "FREQ_2"})
385.                 df5['NO'] = np.array(range(1,n+1))
386.                 df5 = df5[['NO','WORD','FREQ_2']]
387.                 print(df5.to_string(index=False))
388.
389.
390.         print("\n\nProgram Finished.")
```

APPENDIX B: SCREENSHOTS OF YOUR USE CASES

```
TEST STARTED...

BOOK 1: Non-Programmer's Tutorial for Python 2.6
How many word you want to display: 20
 NO      WORD  FREQ_1
  1     print     517
  2    number     282
  3   program     231
  4   license     228
  5      name     222
  6         b     198
  7     first     190
  8    python     176
  9      line     155
 10      list     151
 11      work     149
 12     input     147
 13      http     141
 14     index     139
 15     value     134
 16       org     130
 17      user     127
 18  function     120
 19   version     111
 20       use     107
```

```
BOOK 1: Non-Programmer's Tutorial for Python 3
How many word you want to display: 30
 NO        WORD  FREQ_1
  1       print     572
  2      number     288
  3      python     279
  4        name     228
  5           b     181
  6    tutorial     177
  7     program     172
  8        list     159
  9         non     156
 10  programmer     155
 11        line     153
 12   wikibooks     152
 13       index     144
 14       value     133
 15       input     132
 16     numbers     129
 17    function     121
 18       first     112
 19         org     104
 20        true      99
 21        menu      98
 22        item      96
 23        next      94
 24        type      93
 25      string      89
 26     version      87
 27       https      86
 28        file      86
 29           w      82
 30          en      75
```

```
BOOK 1: Non-Programmer's Tutorial for Python 2.6
BOOK 2: Non-Programmer's Tutorial for Python 3
How many word you want to display: 10



COMMON WORDS
 NO     WORD  FREQ_1  FREQ_2  FREQ_SUM
  1    print     517     572      1089
  2   number     282     288       570
  3   python     231     279       455
  4     name     228     228       450
  5  program     222     181       403
  6        b     198     177       379
  7     list     190     172       310
  8     line     176     159       308
  9    first     155     156       302
 10    index     151     155       283


BOOK 1: Non-Programmer's Tutorial for Python 2.6
DISTINCT WORDS
 NO      WORD  FREQ_1
  1   license     222
  2      work     134
  3      http     115
  4      user      92
  5  document      84
  6     first      78
  7       may      73
  8       raw      66
  9      copy      63
 10   section      61


BOOK 2: Non-Programmer's Tutorial for Python 3
DISTINCT WORDS
 NO        WORD  FREQ_2
  1    tutorial     147
  2  programmer     145
  3         non     138
  4      python     103
  5       https      86
  6         ope      74
  7         sur      74
  8   wikibooks      67
  9       print      55
 10       float      28
```

WARNING:

The difference in numbers could be result of the source of the pdf, pdf to text converter or preprocessing steps (RegExp Tokenizer in this case.) .

REFERENCES

1. Online Word Counter. (n.d.). CountWordsFree. Retrieved December 22, 2020, from https://countwordsfree.com

2. Wikibooks. (n.d.). Retrieved December 22, 2020, from https://en.wikibooks.org/wiki/Main_Page

3. Wikibooks contributors. (2020a). Non-Programmer's Tutorial for Python 3—Wikibooks, open books for an open world. Wikibooks, The Free Textbook Project. https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_3

4. Wikibooks contributors. (2020b). Non-Programmer's Tutorial for Python 2.6—Wikibooks, open books for an open world. Wikibooks, The Free Textbook Project. https://en.wikibooks.org/wiki/Non-Programmer%27s_Tutorial_for_Python_2.6
5. *Jalan/pdftotext*.(2021, January 17). GitHub. https://github.com/jalan/pdftotext
6. Steven Bird, Ewan Klein, and Edward Loper (2009).

Natural Language Processing with Python. O'Reilly Media Inc.

http://nltk.org/book

7. *Bs4*. (2021, January 17). PyPI. https://pypi.org/project/bs4/