

Using machine learning in python to predict an author's gender

*As a project for Basic Programming for the premaster CSAI/DSBG, Tilburg University*

Group: Warsaw

By Andrew Favia

Gideon Thür

Siebe Albers - u472157

Contents

[Research question 3](#_Toc535230761)

[Objectives of the project 3](#_Toc535230762)

[Stage 1: FTP-lib 4](#_Toc535230763)

[Stage 1.1: Downloading textbook files from the Gutenberg project website, by utilizing the FTPlib library of Python. 4](#_Toc535230764)

[Stage 1.2: cleaning and organizing textbook files from the Gutenberg project website, by utilizing regular expressions 4](#_Toc535230765)

[Stage 2: Scikit-learn 5](#_Toc535230766)

[Stage 3: matplotlib 6](#_Toc535230767)

[Appendixes 7](#_Toc535230768)

[Appendix 1: 7](#_Toc535230769)

[Appendix 2: final version of the FTP script 10](#_Toc535230770)

[appendix 3: cleaning the files 13](#_Toc535230771)

# Research question

Can a machine learning algorithm be created with the scikit learn library in python to predict whether a book has been written by a male or a female with an accuracy of 80 percent?

# Objectives of the project

The project consisted of mainly three sub objectives, where mainly three libraries had to be utilized:

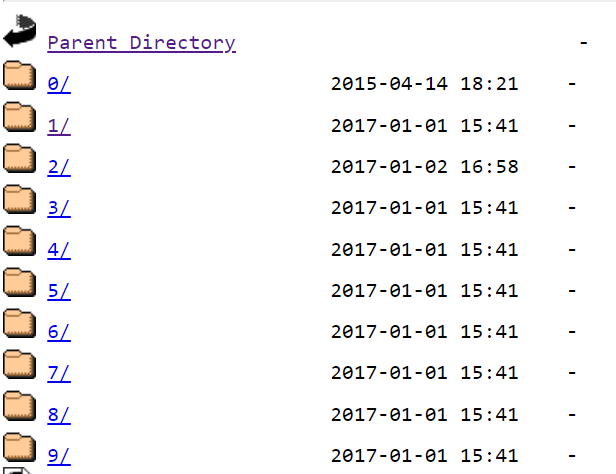
* Stage 1: Downloading and cleaning textbook files from the Gutenberg project website, by utilizing the FTP library.
* Stage 2: Feeding the cleaned book textiles to a supervised learning, by utilizing the sci-kit learn library.
* Stage 3: Graphically representing the results by utilizing the mathplot library

# Stage 1: FTP-library

## Stage 1.1: Downloading textbook files from the Gutenberg project website, by utilizing the FTPlib library of Python.

In order to download books in .txt formats, we utilized the FTP facilitation offered by the following website: <https://www.mirrorservice.org/sites/gutenberg.org/>, The Gutenberg project is a “a volunteer effort to digitize and archive cultural works, to "encourage the creation and distribution of eBooks”. Therefore, the acquisition of the textfiles was legal.

We noticed that there were tens of thousands of ebooks availiable.

These ebooks are located in specific directories (figure 1), in the form of txt.files. We began by creating a script, utilizing the FTP functions to connect to the ftp server, cycle through the directories, and download specific txt. Files (see appendix 1 for the specific script). There were difficulties at first, because the digger() and worm() function that we created (appenix 1) would stop in certain directories. For instance, it would do directories 1/1/1/1/9999 and all those in between, but once it had to go to /2 and do /2/2/2 etc., it would just stop. Conquering problems like these were mainly done by utilizing online sources and trial and error approaches. Eventually, we produced functions that could dig into all directories and download the textfiles, the end product of that script can be found in appendix 2.

## Stage 1.2: cleaning and organizing textbook files from the Gutenberg project website, by utilizing regular expressions

The first objective in this stage was to create a function [name\_builder()] that would return the title and name from the textfile; if the title or author was missing, we deleted it.

We also had to identify the non English books by making use of RE, so we could delete them.

Then we used the identified title and author to save the text file under those two variables in either the female or male folders on our operating system; to identify whether the author was male or female, we utilized list of female and males names downloaded from the internet and simply used if statements to see if their was a match between the two.

Addionally, by making use of RE expressions, we removed meta data from the books that could increase the bias in our machine learning algorithm.

# Stage 2: Scikit-learn

# Stage 3: Seaborn-library

The objective of the third stage was to Graphically represent the results that came together in

second stage. This objective was realized by utilizing the Seaborn library. Initially, we tried using the Matplotlib library. However, after some investigation we found that Seabourn gives us nicer graphs in less lines of code.

The main issues in representing the distributions of our books was getting the grid and scaling of the plots right. In the end we concluded that the function distplot would display the most comprehensive layout. The end result of the script can be seen in **appendix x.**

# Appendixes

## Stage 1 : FTP-library

### Appendix 1:

1. **from** ftplib **import** FTP
3. ftp = FTP('ftp.ibiblio.org')
5. ftp.login()
7. ftp.cwd("pub/docs/books/gutenberg/") #change working directory
9. out\_file = open("C:/Users/Sa/desktop/writingdirs.txt", "w")
11. **print**(ftp.nlst())
13. # =============================================================================
14. #
15. # =============================================================================
16. **def** txtdownload():
17. **for** \_file **in** ftp.nlst():
18. **if** \_file.endswith(".txt"):
19. ftp.retrbinary("RETR") #retrieve a copy of the file

22. **def** getdirs1():
23. list\_dirs = []
24. **for** \_dir **in** ftp.nlst():
25. **if** str(\_dir).isdigit():
26. directory = "pub/docs/books/gutenberg/" + str(\_dir)
27. list\_dirs.append(directory)
29. **return** list\_dirs
31. **def** getdirs2():
32. list\_dirs = []
33. **for** \_dir **in** ftp.nlst():
34. **if** str(\_dir).isdigit():
35. list\_dirs.append(\_dir)
37. **return** list\_dirs
39. # =============================================================================
40. #
41. # =============================================================================
42. **def** digger():
43. # THIS FUNCTION GETS A LIST OF THE FIRST 2 LAYERS OF DIRECTORIES
44. counter = 0
45. list\_of\_dirs = []
46. second\_list = []
47. #uses the getdirs2 function to get the first layer
48. list\_of\_dirs.extend(getdirs2())
50. **print**("LIST: ", list\_of\_dirs)
52. **if** len(list\_of\_dirs) > 0:
53. #first for loop gets the second layer
54. **for** directory **in** list\_of\_dirs:
56. ftp.cwd(directory) #cwd = change working directory
58. second\_list.append(ftp.pwd())
59. **for** \_dir **in** getdirs2():
61. final = str(ftp.pwd()) + "/" + str(\_dir)
63. second\_list.append(final)

66. counter += 1
67. ftp.cwd("..")
69. #once the first two layers are done the worm function starts to dig
70. lista = worm(second\_list)

73. **return** lista
75. # =============================================================================
76. #
77. # =============================================================================
78. **def** worm(\_list):
79. #this function is supposed to dig down all the provided by digger
80. \_list = \_list
81. idx = 0
82. #    idx = len(\_list)
83. final\_list = []
84. lenlist = len(\_list)
86. counter = 0
87. extra = 0
88. **print**("You are in WORM now")
90. **while** counter < 2:
91. #        \_list = list(set(\_list))
92. **try**:
93. **if** len(\_list) > 1:
94. **print**("CICLE HEREEEE")
95. counter += 1
96. **print**(len(\_list[-idx:]))
97. **for** directory **in** \_list:
98. #go in the X directory in the list
99. ftp.cwd(directory) #cwd = change working directory
101. **for** \_dir **in** getdirs2():
102. # for each directory in this new directory
103. # get the name of the new dirs and add them to the
104. # next ones to dig into
105. final = str(ftp.pwd()) + "/" + str(\_dir)
107. **print**(final) #THSI IS THE PRINTED DIR
108. out\_file.writelines(final + "\n")
109. **if** final **not** **in** final\_list:
110. # if these dirs are not in the list of dirs to dig, add them
111. # otherwise skip them
112. final\_list.append(final)
113. \_list.append(final)
114. extra += 1
115. # go back to the previous dir in order to start again
116. ftp.cwd("..")
117. idx = extra
119. **else**:
120. **break**
121. **except**:
122. **continue**
124. **return** final\_list
125. out\_file.close()

128. trial = digger()

131. **print**(trial)

### Appendix 2: final version of the FTP script

|  |
| --- |
|  |
|  | import matplotlib |
|  | import random |
|  | from ftplib import FTP |
|  |  |
|  | # ============================================================================= |
|  | # Define functions |
|  | # ============================================================================= |
|  |  |
|  | def handleDownload(block, fileToWrite): |
|  | # callback handler for the ftp.retrbinary function |
|  | fileToWrite.write(block) |
|  |  |
|  |  |
|  | def txtdownload(): |
|  | for \_file in ftp.nlst(): |
|  | # for each txt file in the directory, create a file with the same name and copy the contents locally. |
|  | if \_file.endswith(".txt"): |
|  | fileToWrite = open(\_file, "wb") |
|  | ftp.retrbinary("RETR %s" % \_file, fileToWrite.write) #retrieve a copy of the file |
|  | print("Writing: %s" % \_file) |
|  | fileToWrite.close() |
|  |  |
|  | def getdirs(): |
|  | list\_dirs = [] |
|  | for \_dir in ftp.nlst(): |
|  | # for all files in the directory, if that whole filename can be converted to a digit, it is a dir. |
|  | if str(\_dir).isdigit(): |
|  | list\_dirs.append(\_dir) |
|  |  |
|  | return list\_dirs |
|  |  |
|  | # ============================================================================= |
|  | # this first section establishes the connection with the Gutenberg's ftp server |
|  | # ============================================================================= |
|  | print("Connecting...") |
|  |  |
|  | ftp = FTP('ftp.ibiblio.org') |
|  |  |
|  | ftp.login() |
|  |  |
|  | print("Connected") |
|  |  |
|  | ftp.cwd("pub/docs/books/gutenberg/") #change working directory |
|  |  |
|  | home = "/pub/docs/books/gutenberg/" |
|  |  |
|  | print("Home directory now") |
|  |  |
|  |  |
|  | out\_file = open("writingdirs.txt", "w") # File where to save the directories for later use. |
|  |  |
|  | # ============================================================================= |
|  | # |
|  | # ============================================================================= |
|  |  |
|  | dirs = [] |
|  |  |
|  | for x in range(1,6): |
|  | # Creates the list of base directories in the ftp Gutenberg server to then dig into. |
|  | # range can go from 1 to 10 in all these 3 layers |
|  | for y in range(0,10): |
|  | for z in range(0,10): |
|  | newdir = "/pub/docs/books/gutenberg/%s/%s/%s" % (x,y,z) |
|  | dirs.append(newdir) |
|  |  |
|  | for \_dir in dirs: |
|  | # for each directory in the ones listed, if it exists, go into it, go one level deeper |
|  | try: |
|  | ftp.cwd(\_dir) |
|  | dirs2 = getdirs() |
|  | for \_dir2 in dirs2: |
|  | try: |
|  | #if these directories exist, write them in the directory list txt file |
|  | print(\_dir2) |
|  | final = str(ftp.pwd()) + "/" + str(\_dir2) |
|  | print(final) |
|  | out\_file.writelines(final + "\n") |
|  | except: |
|  | continue |
|  | ftp.cwd(home) |
|  | except: |
|  | continue |
|  |  |
|  | # close the directory list txt file |
|  | out\_file.close() |
|  |  |
|  | print("Got all directories") |
|  | print("Starting to find and download books now...\n") |
|  |  |
|  | with open("writingdirs.txt", "r") as \_dirs: |
|  | \_dirs = \_dirs.readlines() |
|  |  |
|  | for \_dir in \_dirs: |
|  | #for each directory saved in the txt file, go in that directory and download all the txt files. |
|  | try: |
|  | \_dir = \_dir.rstrip() |
|  | ftp.cwd(\_dir) |
|  | print(ftp.pwd()) |
|  | txtdownload() |
|  | ftp.cwd(home) |
|  | except: |
|  | continue |

### appendix 3: cleaning the files

|  |
| --- |
| import re |
|  | import os |
|  | import filecleaner |
|  |  |
|  | # ============================================================================= |
|  | # Define functions |
|  | # ============================================================================= |
|  |  |
|  | def name\_builder(\_text): |
|  |  |
|  | # find the title and author line and return them. |
|  | title = re.search(r"Title:(.+)",\_text).group(1) |
|  | author = re.search(r"Author:(.+)",\_text).group(1) |
|  |  |
|  | print(title) |
|  | print(author) |
|  | return title,author |
|  |  |
|  | # ============================================================================= |
|  | # |
|  | # ============================================================================= |
|  |  |
|  | # Create the folders for man and women authors if they are not there. |
|  | if "Men" not in os.listdir(): |
|  | os.mkdir("Men") |
|  | if "Women" not in os.listdir(): |
|  | os.mkdir("Women") |
|  |  |
|  | in\_files = os.listdir() |
|  | for \_file in in\_files: |
|  | # find all the books, consider them only if they are in English |
|  | if \_file.endswith(".txt"): |
|  |  |
|  | text= open(\_file , "r",encoding="utf-8",errors="ignore") |
|  | text=text.read() |
|  | if "Language: English" not in text: |
|  | print(\_file) |
|  | in\_files.remove(\_file) |
|  |  |
|  | for \_file in in\_files: |
|  | if \_file.endswith(".txt"): |
|  | # of all the English books, consider only them that provide title and author formatted this way. |
|  | text= open(\_file , "r",encoding="utf-8",errors="ignore") |
|  | text=text.read() |
|  | if "Author:" not in text or "Title:" not in text: |
|  | print(\_file) |
|  | in\_files.remove(\_file) |
|  |  |
|  | # ============================================================================= |
|  | # |
|  | # ============================================================================= |
|  |  |
|  | for \_file in in\_files: |
|  |  |
|  | if \_file.endswith(".txt"): |
|  | # Rename each file that made the selection as "author-book-title.txt" |
|  | text= open(\_file , "r",encoding="utf-8",errors="ignore") |
|  | text= text.read() |
|  | try: |
|  | title, author = name\_builder(text) |
|  | except: |
|  | continue |
|  |  |
|  | file\_name = author + title |
|  | file\_name = file\_name.lstrip() |
|  | file\_name = re.sub("[\. ,\r]+", "-", file\_name) |
|  | file\_name = re.sub("[\. ,\s\r]+", "-", file\_name) |
|  | file\_name = file\_name + ".txt" |
|  | print(file\_name) |
|  |  |
|  | try: |
|  | os.rename(\_file ,file\_name) |
|  | except: |
|  | print(\_file , "this is a double") |
|  | continue |
|  |  |
|  | # ============================================================================= |
|  | # Put books in the Men or Women folder based on author's first name |
|  | # ============================================================================= |
|  |  |
|  | #these are the txt files containing lists of first names. |
|  | with open("male\_names.txt", "r") as malenames: |
|  | malenames = malenames.readlines() |
|  |  |
|  | malenames = [name.lower().rstrip() for name in malenames] |
|  |  |
|  | with open("female\_names.txt", "r") as femalenames: |
|  | femalenames = femalenames.readlines() |
|  |  |
|  | femalenames = [name.lower().rstrip() for name in femalenames] |
|  |  |
|  | nono\_files = ["male\_names.txt","female\_names.txt"] |
|  |  |
|  |  |
|  | for \_file in os.listdir(): |
|  | # for each txt file in the folder, check if the first name is in either name list |
|  | # if so, move the file into the corresponding dir. |
|  | if \_file.endswith(".txt") and \_file not in nono\_files: |
|  | index = \_file.find("-") |
|  | print(\_file[:index]) |
|  | if \_file[:index].lower() in femalenames: |
|  | print("Female! ", \_file[:index]) |
|  | os.rename(\_file,"./Women/"+\_file) |
|  |  |
|  | elif \_file[:index].lower() in malenames: |
|  | print("Male!", \_file[:index]) |
|  | os.rename(\_file,"./Men/"+\_file) |
|  |  |
|  |  |
|  | os.chdir("Women") |
|  |  |
|  | # use the file cleaner to delete the first and last part of the books which usually contain metadata and info |
|  | # that might create bias in the machine. |
|  |  |
|  | for book in os.listdir(): |
|  | try: |
|  | print(book) |
|  | filecleaner.file\_cleaner(book) |
|  | except: |
|  | continue |
|  |  |
|  | os.chdir("..") |
|  |  |
|  | os.chdir("Men") |
|  |  |
|  | for book in os.listdir(): |
|  | try: |
|  | print(book) |
|  |  |
|  | filecleaner.file\_cleaner(book) |
|  | except: |
|  | continue |
|  |  |
|  | os.chdir("..") |

## Stage 2 : Scikit-learn

## Stage 3: Matplot-lib

### Appendix x: Graphically representing the results by utilizing the seaborn library

|  |
| --- |
| import matplotlib.pyplot as plt |
|  | import numpy as np |
|  | import seaborn as sns |
|  |  |
|  |  |
|  | def graphics (randmatrix): |
|  |  |
|  | import random |
|  |  |
|  | num\_rows, num\_columns = randmatrix.shape |
|  |  |
|  | cols = ["r", "w", "blue", "g", "m", "c"] |
|  | sns.set\_style("darkgrid", {"axes.facecolor": ".1", "grid.color": ".8"}) |
|  | sns.set\_context("poster") |
|  |  |
|  | print(num\_rows) |
|  | sns.distplot(randmatrix[0,], hist=False, color=random.choice(cols)) |
|  | plt.title("Frequency of Words") |
|  | plt.xlabel("Words") |
|  | plt.ylabel("Frequencies") |