Shanmugapriya T

Postdoctoral Research Associate

Digital Humanities Hub

Department of History

Lancaster University

**An Introduction to NLTK: An Experiment with Indian English Short Stories**

During my PhD tenure in the Digital Humanities Studies Research Group at Indian Institute of Technology Indore in 2015, I looked for text mining on Indian English literature, but couldn’t find one. Even the DH scholars I met from different parts of India during various occasions expressed that their dearth of ‘know-how’ precludes their research in Digital Humanities. Mohanapriya, a research scholar at Bharathiar University said, “[l]ack of infrastructure and proper training in [the] programming language made me give up after facing many attempts” on mining English novels (Shanmugapriya and Menon 2020). The same is the case for many scholars in India. The lack of DH pedagogy resulted in thwarting computer-assisted research in humanities. Applying computational methods on Indian English novels is an underexplored area in the Indian DH spectrum. In this article, I will give a brief introduction to the features of NLTK and endeavoring them on Rabindranath Tagore’s short stories. At the same time, I also acknowledge the lack of digitization is another pivotal issue in Indian literature, but I am not focusing on that in this article. Rather I will use the resource which is available to us. This article is for beginners who do not have knowledge in programming languages but are interested in learning them.

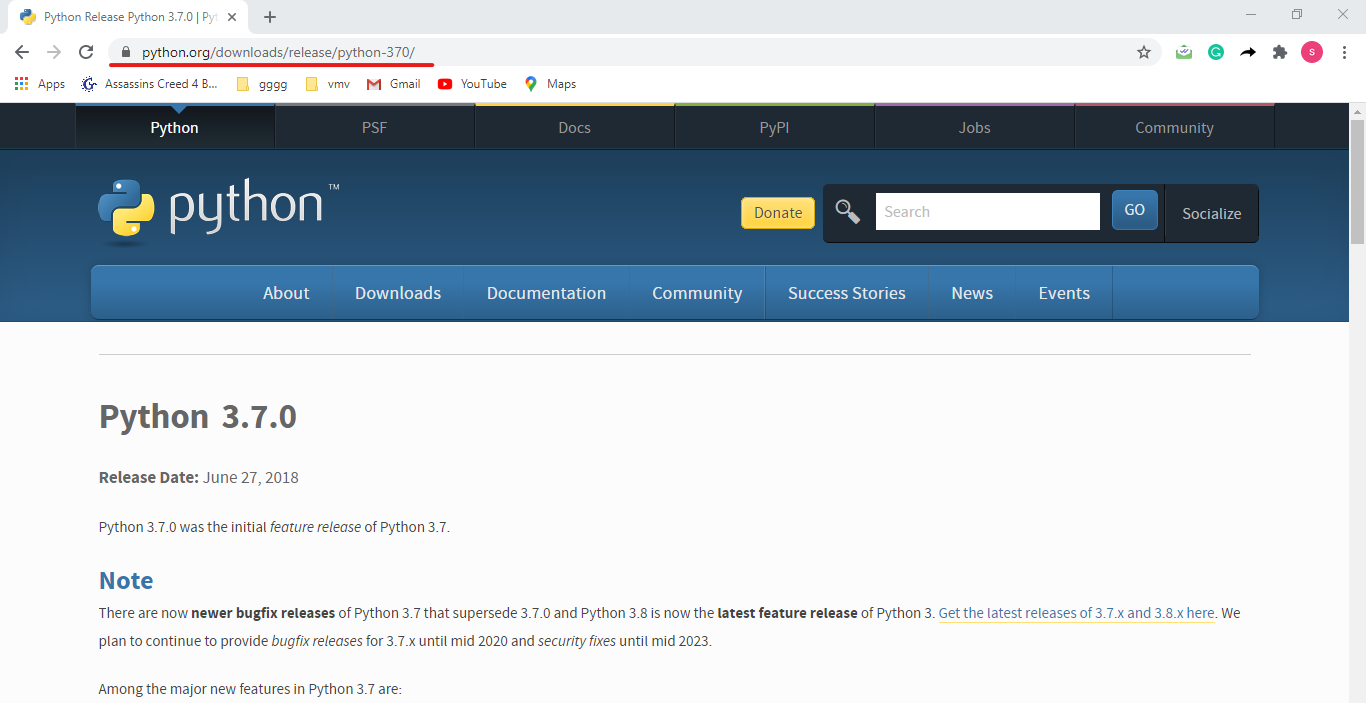
Natural Language Processing (NLP), is a part of linguistics, computer science and artificial intelligence, developed to understand, automate and manipulate human languages. Machine language cannot directly comprehend human language as it has only zeros and ones whereas humans speak and write numerous languages which themselves have a huge number of alphabets, figures and numbers. NLP mediates between human languages and machine languages to help computers read, write, manipulate and interpret data. It is mainly developed for text analysis, speech recognition, named entity recognition (persons, organizations, locations, time, and quantities, etc.), machine translation and Al chat box (automatic online assistant) etc. Natural Language Tool Kits (NLTK) (<https://www.nltk.org/>) is one of the NLP packages created by Steven Bird, Edward Loper and Ewan Klein in the Department of Computer and Information Science at the University of Pennsylvania. It was released in 2001. NLTK is an open-source material and it contains modules, sample data and tutorials. It is well suitable for Windows, Mac OS and Linux. It is the best tool to begin training in text mining for digital humanities students and scholars. Text mining is employed to convert the large text or texts into smaller units to extract information for the humanities inquiry. The best features of NLTK are tokenization, parts of speech tags and many other modules. Python is the best programming language to play with NLTK. In this essay, we will play with a few features such as frequent word analysis for single text and corpus of text files and frequent word analysis based on parts of speech tags. These kinds of text analyses in the digital literary studies are primarily used to study author attribution, stylistic studies, character analysis and gender studies.

Let’s begin with installing Python:

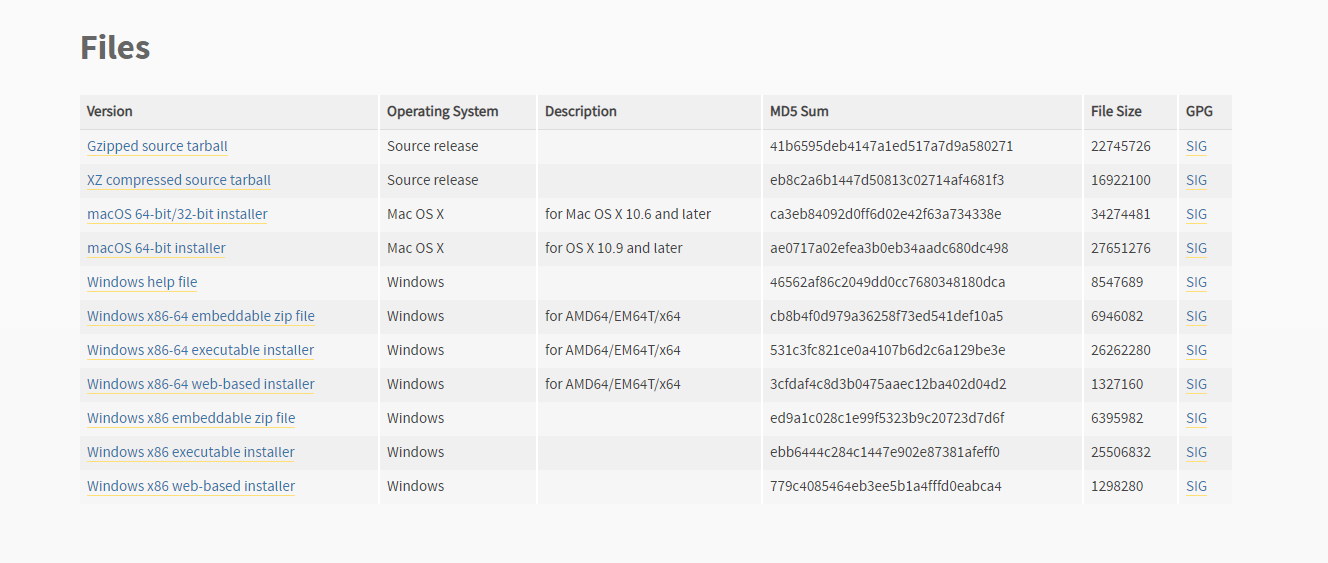
**Downloading and installing Python 3.7.0**

**Step 1:** Go to: <https://www.python.org/downloads/release/python-370/>

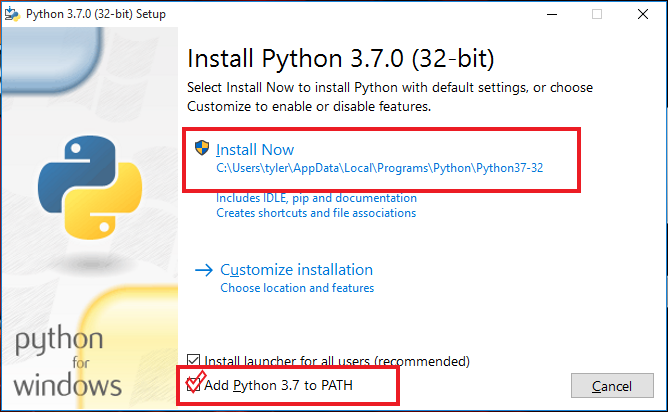
(I use Python 3.7 but if you prefer or already have any other version above 3, that is also fine)



**Step 2**: Download the version according to your operating system and bit



**Step 3**: Please run the python installer which you downloaded and select *Add to* *path* and *Install now* (See screenshot below)

****

Now you must be ready with your Python!

**Basic Python**

I. Data Types:

1. List = []
2. Set = {}
3. Tuple = ()

II. String literals:

1. Words= ‘hello’
2. Sentences—”This is a sentence”
3. Paragraph—””” This is a sentence. This is a sentence.
   * + 1. This is a sentence, This is a sentence”””

III. Comment:

1. Hashtag (# )is used for single line comment out in Python
2. Triple quotes (“ “ “ ” ” ” ) are used for multiple line comments in Python

**Installing NLTK library (https://www.nltk.org/) and other packages**

**Step 1**: Go to your Command Prompt and type *cd* and right click on it and open it through *Run as administrator*

**Step 2:** Type: pip3 install nltk (https://www.nltk.org/book/)

**Step 3**: Other modules which we use for text mining are *requests* and *BeatutifulSoup*. The former is used to make a request to a web page to extract the text and the latter is used to parse the html page. Install these packages as well using pip3, ex. pip3 install BeatutifulSoup

Once the installation is done, you are ready to do coding! Go to *Start* type IDLE and open it !

**Single text analysis**

Import the installed modules into IDLE and also import the *tokenization* and *stopwords* modules from *nltk* package. The former is applied to convert the text into tokens and the latter is deployed to remove the stopwords (such as a, an, the, they etc.) in the text.

**import nltk**

**from nltk.tokenize import RegexpTokenizer**

**from nltk.corpus import stopwords**

**import requests**

**from bs4 import BeautifulSoup**

We import a html link of the text Rabindranath Tagore's *Mashi And Other Stories* and parse the text from the html:

**url = "http://www.gutenberg.org/files/22217/22217-h/22217-h.htm" #get the link of a text**

**get = requests.get(url) #get the url**

**html = get.text # get the text from the url**

**text = BeautifulSoup(html, "html.parser").get\_text() # parsing the html**

The text contains metadata about the author, publisher and Project Gutenberg. We just simply use *find* function to find the primary text as those details will affect our output. The advantage of Project Gutenberg files is that they have tags of "start of this project..." and "end of this project..."for each file which we can as tags to extract the text between these tags.

**extract1 = text.find** **\*\*\* START OF THIS PROJECT GUTENBERG EBOOK MASHI AND OTHER STORIES \*\*\* ")**

**extract2 = text.rfind("End of Project Gutenberg's Stories from Tagore ")**

**#find the index of ending of the text to eliminate the metadata**

**analysis\_part = text[extract1:extract2] # combining them**

Tokenization converts the document into a smaller unit of sentences or words which is known as token. We extract only words in the sentence by using regular expression tokenizer.

**tokenizer = RegexpTokenizer('\w+')**

**tokens = tokenizer.tokenize(analysis\_part) # tokenize the text**

We convert all the tokens into lower case. Then, we remove the stopwords which will create an impact in our analysis as they appear frequently in the text. The below code also demonstrates how to add new stopwords in the existing stopwords list of nltk. Though there are multiple ways to remove stopwords, I used *list* data type and *append* function as they are easy to follow the code.

**words = []**

**for word in tokens:**

**words.append(word.lower()) # convert the entire text into lowercase**

**new\_stopwords = ("could", "would", "also", "us") # add few more words to the list of stopwords**

**stopwords = stopwords.words("english") # call the stopwords from nltk**

**for i in new\_stopwords:**

**stopwords.append(i) # adding new stopwords to the list of existing stopwords**

**words\_list = []**

**for without\_stopwords in words:**

**if without\_stopwords not in stopwords:**

**words\_list.append(without\_stopwords) # applying stopwords**

We use *nltk.FreqDist* function to find the frequent words in the text

**fre\_word\_list = nltk.FreqDist(words\_list) #extracting the frequently appeared words**

**n= 15 # the top 15 frequent words**

**fre\_word\_list.plot(n, color='green')**

**Corpus of text analysis**

Now, we can mine the corpus of texts using little more advanced methods of Python.

1. Install *glob* using pip and import the module
2. Import other necessary modules which we already installed in our previous analysis
3. Asterisk mark will import all plain text files in the corpus
4. Create a corpus of text files and call them using glob
5. Store the *stopwords* of *nltk* in a variable

**import nltk**

**from nltk.tokenize import RegexpTokenizer**

**from nltk.corpus import stopwords**

**import glob**

**corpus = glob.glob("E:\Medium Blog\Text\_mining\\*.txt")**

**stop\_words = set(stopwords.words('english'))**

#### Pre-processing and analysis

We will call the corpus using for loop and then read the texts and convert them into lowercase. We extract the content for analysis, apply stopwords list and tokenization as we did for the single text, but everything should be in the *for loop* as in the below code.

**for i in range(len(corpus)):**

**text\_file = open(corpus[i], "r", encoding = "UTF-8")**

**lines = []**

**lines = text\_file.read().lower()**

**extract1 =lines.find("start of this project")**

**extract2 = lines.rfind("end of this project")**

**lines = lines[extract1:extract2]**

**tokenizer = RegexpTokenizer('\w+') # extracting words**

**tokens = tokenizer.tokenize(lines) # tokenize the text**

**new\_stopwords = ("could", "would", "also", "us") # add few more words to the list of stopwords**

**stop\_words = stopwords.words('english')**

**for i in new\_stopwords:**

**stop\_words.append(i) # adding new stopwords to the list of existing stopwords"""**

**words\_list = [w for w in tokens if not w in stop\_words]**

**filtered\_words = []**

**for w in tokens:**

**if w not in stop\_words:**

**filtered\_words.append(w)**

**fre\_word\_list = nltk.FreqDist(filtered\_words) #extracting frequently appeared words**

**print(fre\_word\_list.most\_common(5)) # check the most common frequent words**

**fre\_word\_list.plot(25) #create a plot for the output**

**pos = nltk.pos\_tag(filtered\_words, tagset = 'universal') # applying parts of speech (pos) tag for further analysis**

**p = []**

**y = ['NOUN'] # change the pos here to store them separately**

**for j in pos:**

**for l in y:**

**if l in j:**

**p.append(j)**

**noun = nltk.FreqDist(p) # check the frequency of each pos**

**noun.plot(20) # creating a plot for pos**

**References:**

**T., Shanmugapriya** and Nirmala Menon. “Infrastructure and Social Interaction: Situated Research Practices in Digital Humanities in India.” *Digital Humanities Quarterly,* Vol. 14, No. 3, 2020.