# Natural Language Processing Toolkit for Python

Mateusz Wyszecki

June 2019

## 1 Introduction

Natural languages are those languages which evolved naturally over the years and are used by humans for everyday communication. They can vary drastically and be extremely complex, therefore proper analysis of natural language oftentimes requires assistance from programs written with computer code, a type of non-natural language. Python is one of the most popular programming languages and its Natural Language Toolkit was designed specifically to analyse natural languages. This documentation provides directions for setting up and using this Python library.

## 2 Software Requirements

To exploit Natural Language Toolkit's full potential, you will need to install the following software:

1. Python (version 3.2 or later)
2. NumPy - a library for Python with support for multidimensional arrays and linear algebra, required for certain probability, tagging, clustering, and classification tasks.
3. Matplotlib (optional) - a 2D plotting library for Python which creates visualizations for the results of linguistic analysis.

## 3 Installation

### 3.1 Mac/Unix

1. Open Command Terminal
2. Install NLTK: `run pip install --user -U nltk`
3. Install Numpy: run `pip install --user -U numpy`
4. Test installation: run `python` then type `import nltk`

If you are using an older version of Python (which is not recommended), it may be necessary to install `setuptools` and `pip` components.

### 3.2 Windows

1. Install Python by navigating to Download page for Windows at python.org
2. Open Windows PowerShell
3. Install NLTK: run `pip install NLTK`
4. Install Numpy: run `pip install Numpy`

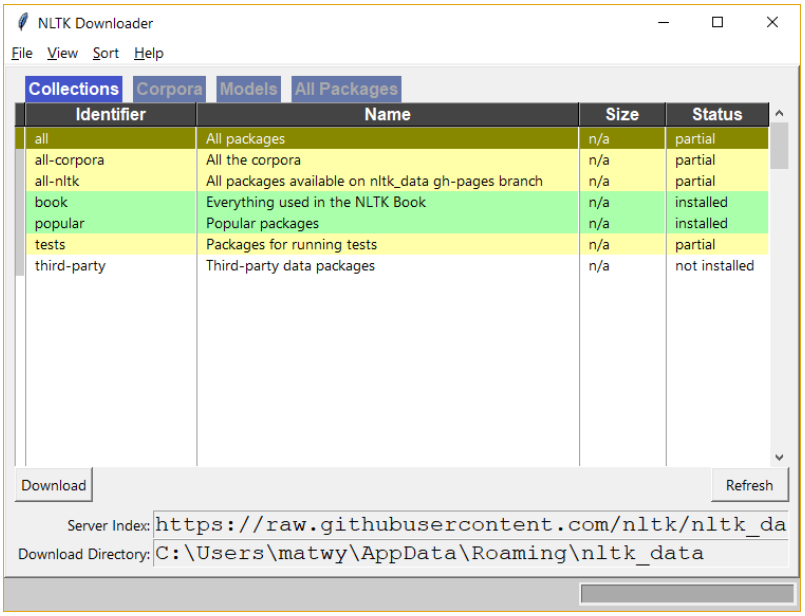
## 4 Acquiring data for testing purposes

In order to use Natural Language Toolkit, you will need a corpus, i.e. a collection of texts for analysis. For testing purposes, creators of NLTK provide a number of corpora which can be freely analysed by all users. If you already have a corpus you wish to examine, skip this section.

To download the corpora, run:

```
>>> import nltk
>>> nltk.download()
```

This command will open the download window. Select "all" to acquire all the additional content proposed by the creators, or simply "all-corpora" to download only the texts for analysis.



# 5 Loading your own Corpus

Loading your own collection of texts is an extremely simple process. You need to:

1. Import the `PlaintextCorpusReader` function
2. Define the path to your folder
3. Define a variable for a given file within the folder

Here is how it is done in Pyhon:

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = '/usr/share/dict'
>>> wordlists = PlaintextCorpusReader(corpus_root, '.*')
>>> wordlists.fileids()
['README', 'connectives', 'propernames', 'web2', 'web2a', 'words']
>>> yourowntext = wordlists.words('connectives')
```

Now, you can use all the functions defined in the next section by simply substituting the word `corpus` for `yourowntext`.

# 6 Analysing texts

Executing the command `from nltk.book import *` loads all text available to the memory of the program.

1. text1: Moby Dick by Herman Melville 1851
2. text2: Sense and Sensibility by Jane Austen 1811
3. text3: The Book of Genesis
4. text4: Inaugural Address Corpus
5. text5: Chat Corpus
6. text6: Monty Python and the Holy Grail
7. text7: Wall Street Journal
8. text8: Personals Corpus
9. text9: The Man Who Was Thursday by G . K . Chesterton 1908

From this point onward, all methods and functions in Python can be used on on those texts.

## 6.1 Concordance

There is a number of ways to search through a text using the NLTK library. One of the most basic and important function is the concordance search, which also provides context for a given word. To execute it, run `corpus.concordance("word")` with the parameter `word` being a word you wish to look for and `corpus` - the text to search through. This is the concordance of a word "monstrous" in `text1`:

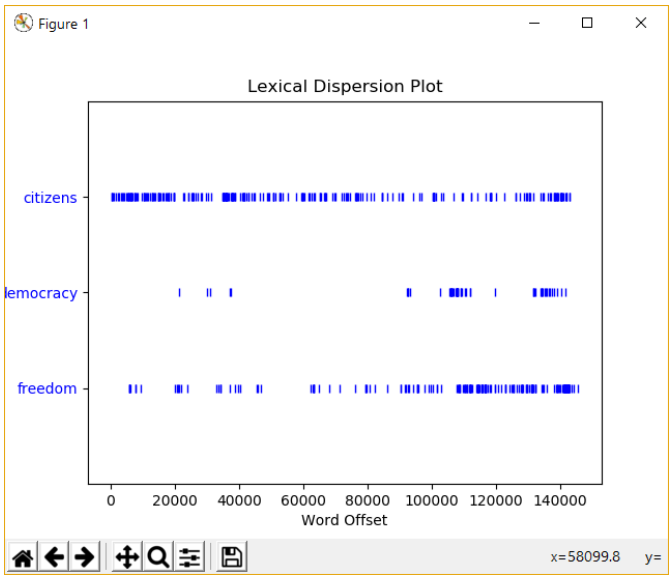
```
>>> text1.concordance("monstrous")
```

```
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the Monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
```

## 6.2 Dispersion Plot

The capabilities of NLTK go beyond simple text-searching. With the use of Matplotlib library, we are able to create complex histograms and visualization of word-placement in a text. To create a dispersion plot showing at which points in text do certain words appear, run: `corpus.dispersion_plot(["word1", "word2"])` It is important to note that dispersion plots can be created for any number of words: to analyse more, simply add additional parameters in the command. Here is an example dispersion plot:

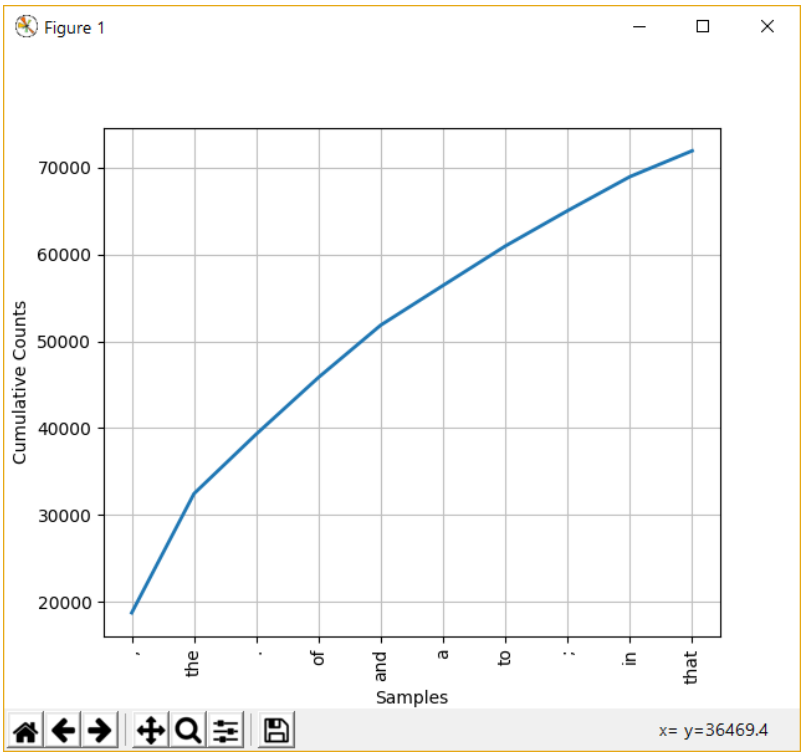
```
>>> text4.dispersion_plot(["citizens", "democracy", "freedom"])
```




## 6.3 Frequency Distributions

Using NLTK, we can easily keep track of frequency of every lexical item in the text. In itself, it is not very informative, but if we select only the most common words and create a frequency plot, it can provide valuable insights into the text. To to so, simply create a frequency variable: `fdist = FreqDist(corpus)` and list all of the words using the `fdist.most_common(num)` method, or create a plot using `fdist.plot(num, cumulative=True)` (as illustrated below), where `num` stands for a number of most frequent words you wish to display.

```
>>> fdist = FreqDist(text1)
>>> fdist.plot(10, cumulative=True)
```




## 6.4 Collocations

Collocations are crucial for studying characteristics of a given text. NLTK offers a very straightforwards way of obtaining them: `corpus.collocations()`

```
>>> text4.collocations()
```

```
United States; fellow citizens; four years; years ago; Federal
Government; General Government; American people; Vice President; Old
World; Almighty God; Fellow citizens; Chief Magistrate; Chief Justice;
God bless; every citizen; Indian tribes; public debt; one another;
foreign nations; political parties
```

## 6.5 Similarity

If you wish to study a particular word instead of the entire text, you may want to use `{corpus.similar("words")}` method, which returns all words that appear in the same context as the word given in the parameters.

```
>>> text2.similar("monstrous")
```

```
very so exceedingly heartily a as good great extremely remarkably
sweet vast amazingly
```