Search Medium

Published in ML Research Lab

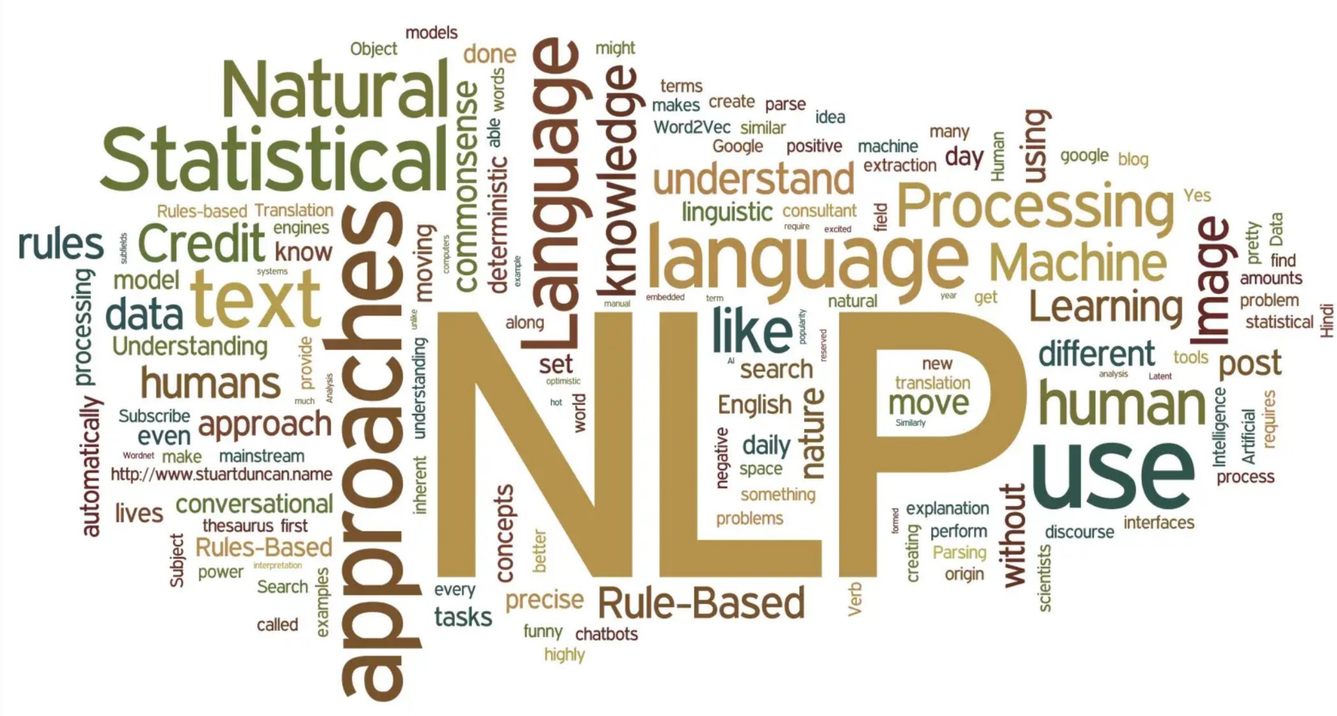
Ashish Patel Follow

Jul 20, 2018 · 6 min read · Listen

Save

Beginner Practical Guide of Natural Language Processing(NLP)

Natural Language Processing Series!!!



Hi Folks, Today I am going to write for beginner who are started learning of NLP. Natural Language Processing is amazing field related to text, speech recolonization.**Natural language processing** (**NLP**) is an area of **computer science** and

artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data.Natural Language Processing (NLP), using Python's NLTK library. NLTK is Python's natural language processing toolkit, one of the most commonly used Python libraries in the NLP world.

1. What is NLP?

Simply put, Natural Language Processing (NLP) is the development of applications or services that understand human language.

Here are some practical examples of natural language processing (NLP), such as speech recognition, speech translation, understanding complete sentences, understanding synonyms of matching words, and generating grammatically correct complete sentences and paragraphs.

This is not all that NLP can do.

2. NLP implementation

Search engines : such as Google, Yahoo, etc. Google search engine knows you are a technician, so it shows technology-related results;

Social network push : For example, Facebook News Feed. If the News Feed algorithm knows that your interest is natural language processing, relevant ads and posts will be displayed.

Speech engine : such as Apple's Siri.

Spam filtering : such as Google spam filter. Unlike ordinary spam filtering, it judges whether it is spam by knowing the deep meaning of the content of the email.

3. NLP library

Here are some open source natural language processing libraries (NLPs):

- Natural language toolkit (NLTK);
- Apache OpenNLP;

- Stanford NLP suite;
- Gate NLP library

The Natural Language Toolkit (NLTK) is the most popular Natural Language Processing Library (NLP), written in Python, and has very strong community support behind it.

NLTK is also very easy to use, in fact, it is the simplest natural language processing (NLP) library.

4. Install NLTK

If you are using Windows/Linux/Mac, you can install NLTK using pip:

```
pip install nltk
```

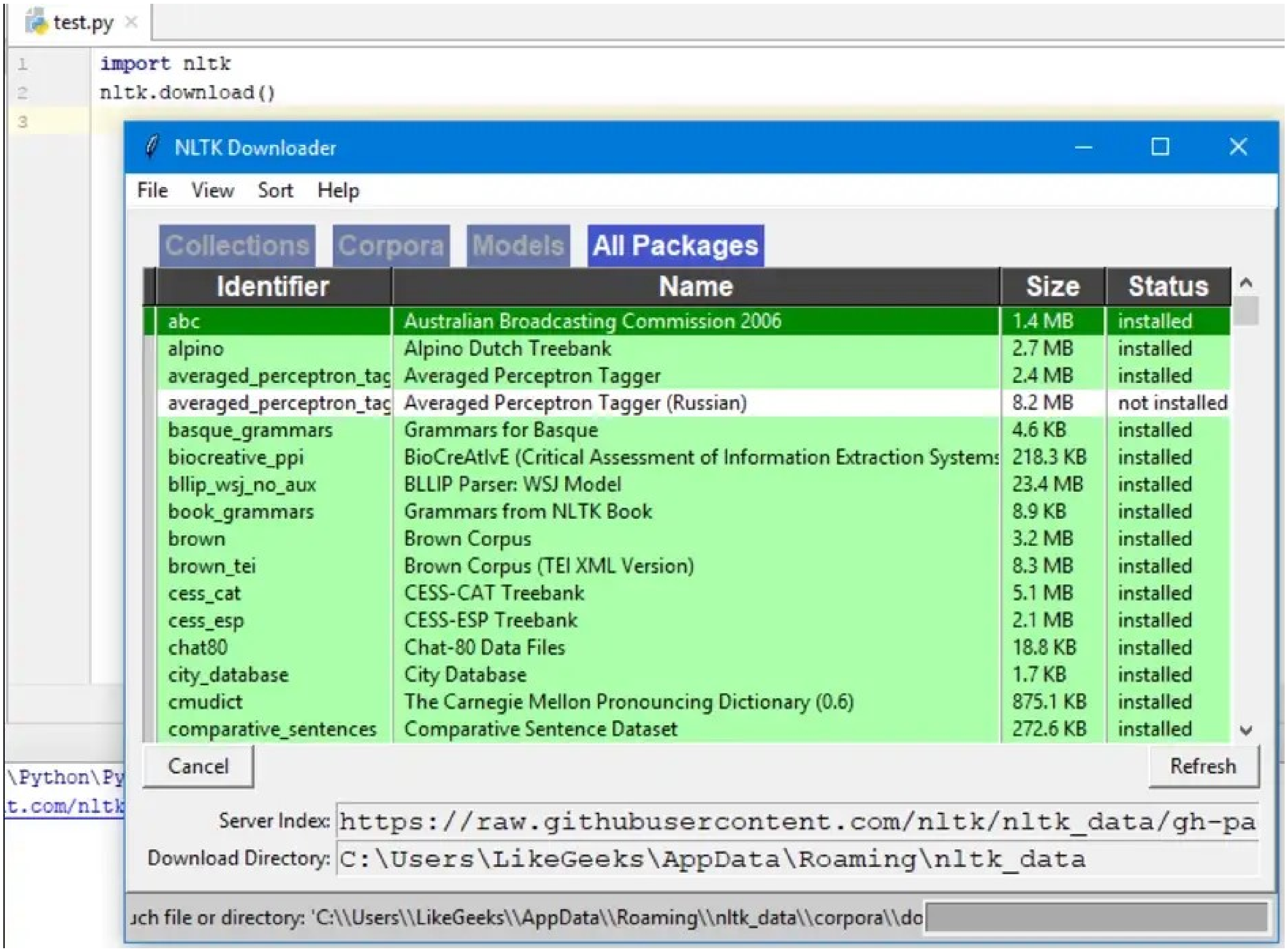
nlp.1py hosted with ❤ by **GitHub** **view raw**

Open the python terminal and import NLTK to check if NLTK is installed correctly and If all goes well, this means that you have successfully installed the NLTK library. To install NLTK for the first time, you need to install the NLTK extension package by running the following code:

```
import nltk

nltk.download()
```

nlp2.py hosted with ❤ by **GitHub** **view raw**

This will bring up the NLTK download window to choose which packages need to be installed:



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

Identifier	Name	Size	Status
abc	Australian Broadcasting Commission 2006	1.4 MB	installed
alpino	Alpino Dutch Treebank	2.7 MB	installed
averaged_perceptron_tag	Averaged Perceptron Tagger	2.4 MB	installed
averaged_perceptron_tag	Averaged Perceptron Tagger (Russian)	8.2 MB	not installed
basque_grammars	Grammars for Basque	4.6 KB	installed
biocreative_ppi	BioCreAtIvE (Critical Assessment of Information Extraction Systems	218.3 KB	installed
bllip_wsj_no_aux	BLLIP Parser: WSJ Model	23.4 MB	installed
book_grammars	Grammars from NLTK Book	8.9 KB	installed
brown	Brown Corpus	3.2 MB	installed
brown_tei	Brown Corpus (TEI XML Version)	8.3 MB	installed
cess_cat	CESS-CAT Treebank	5.1 MB	installed
cess_esp	CESS-ESP Treebank	2.1 MB	installed
chat80	Chat-80 Data Files	18.8 KB	installed
city_database	City Database	1.7 KB	installed
cmudict	The Carnegie Mellon Pronouncing Dictionary (0.6)	875.1 KB	installed
comparative_sentences	Comparative Sentence Dataset	272.6 KB	installed

Server Index: https://raw.githubusercontent.com/nltk/nltk_data/gh-pa

Download Directory: C:\Users\LikeGeeks\AppData\Roaming\nltk_data

ıch file or directory: 'C:\\Users\\LikeGeeks\\AppData\\Roaming\\nltk_data\\corpora\\do

Installed window of NLTK

You can install all the packages because they are small in size, so there is no problem.

5.Use Python Tokenize text

First, we will grab a web page content and then analyze the text to understand the content of the page.

We will use the urllib module to grab the web page:,

```
import urllib.request

response = urllib.request.urlopen('http://php.net/')
html = response.read()
print (html)
```

nlp3.py hosted with ❤ by **GitHub** **view raw**

As you can see from the print results, the results contain many HTML tags that need to be cleaned up.
Then the Beautiful Soup module cleans the text like this:

```
from bs4 import BeautifulSoup

import urllib.request
response = urllib.request.urlopen('http://php.net/')
html = response.read()
soup = BeautifulSoup(html,"html5lib")
# This requires the html5lib module to be installed.
text = soup.get_text(strip=True)
print (text)
```

nlp4.py hosted with ❤ by **GitHub** **view raw**

Now we get a clean text from the crawled page.
Next, convert the text to tokens like this:

```
from bs4 import BeautifulSoup
import urllib.request

response = urllib.request.urlopen('http://php.net/')
html = response.read()
soup = BeautifulSoup(html,"html5lib")
text = soup.get_text(strip=True)
tokens = text.split()
print (tokens)
```

nlp5.py hosted with ❤ by **GitHub** **view raw**

6.Statistical word frequency

The text has been processed, and now the Python NLTK is used to count the frequency distribution of the token.

This can be done by calling the `FreqDist()` methods in NLTK :

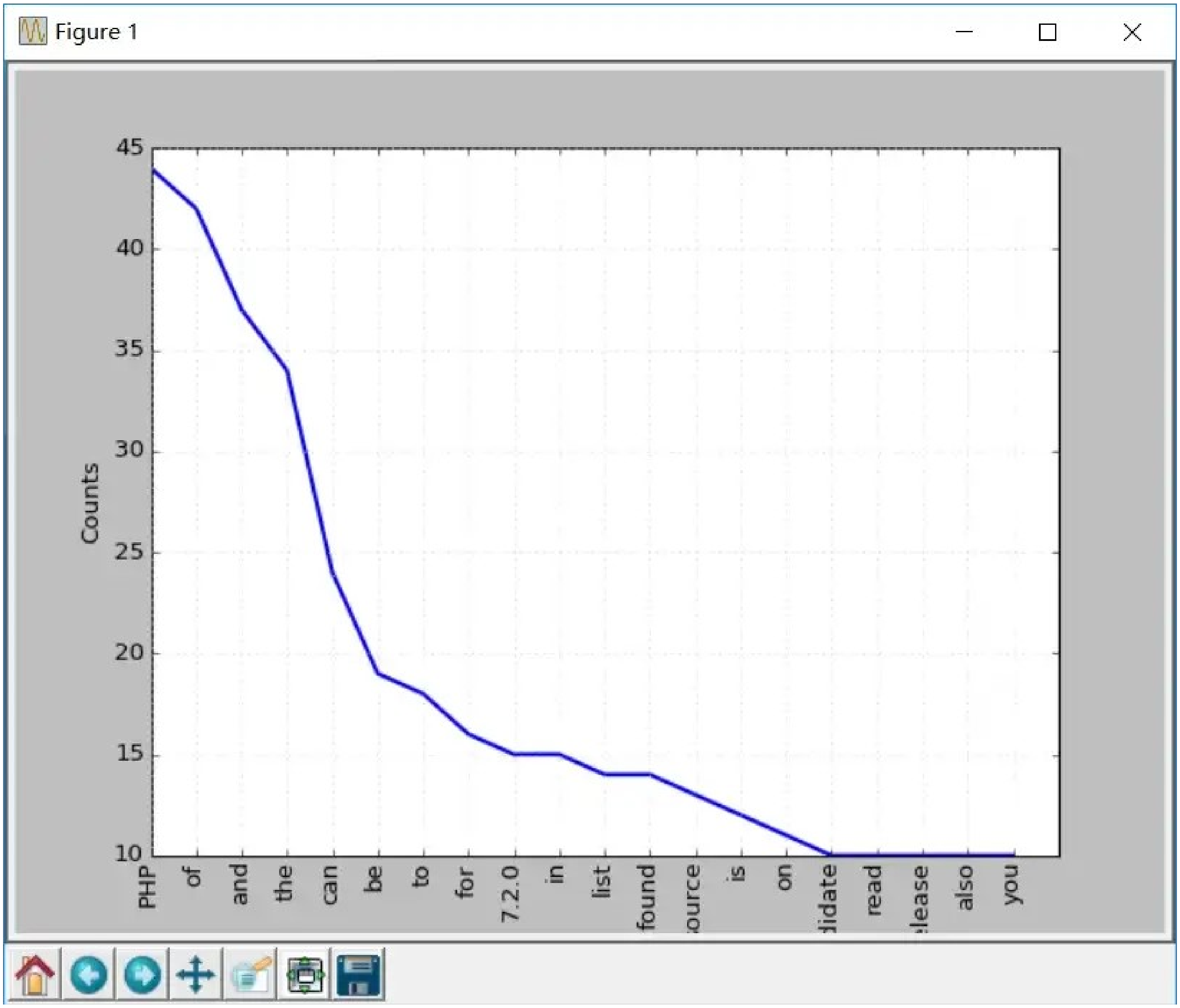
```python
from bs4 import BeautifulSoup
import urllib.request
import nltk

response = urllib.request.urlopen('http://php.net/')
html = response.read()
soup = BeautifulSoup(html,"html5lib")
text = soup.get_text(strip=True)
tokens = text.split()
freq = nltk.FreqDist(tokens)
for key,val in freq.items():
    print (str(key) + ':' + str(val))
```

nlp6.py hosted with ❤ by **GitHub** **view raw**

If you search for output, you can find that the most common token is PHP. You can call a `plot` function to make a frequency distribution map:

```python
freq.plot(20, cumulative=False)
# need to install matplotlib library
```

nlp7.py hosted with ❤ by **GitHub** **view raw**



These words above. For example `of` , `a` , `an` etc., these words are all stop words.

In general, stop words should be removed to prevent them from affecting the results of the analysis.

7.Handling stop words

NLTK comes with a list of stop words in many languages, if you get English stop words:

```python
from nltk.corpus import stopwords

stopwords.words('english')
```

nlp8.py hosted with ❤ by GitHub view raw

Now, modify the code to clear some invalid tokens before drawing:

```python
clean_tokens = list()
sr = stopwords.words('english')
for token in tokens:
    if token not in sr:
        clean_tokens.append(token)
```

nlp9.py hosted with ❤ by GitHub view raw

The final code should look like this:

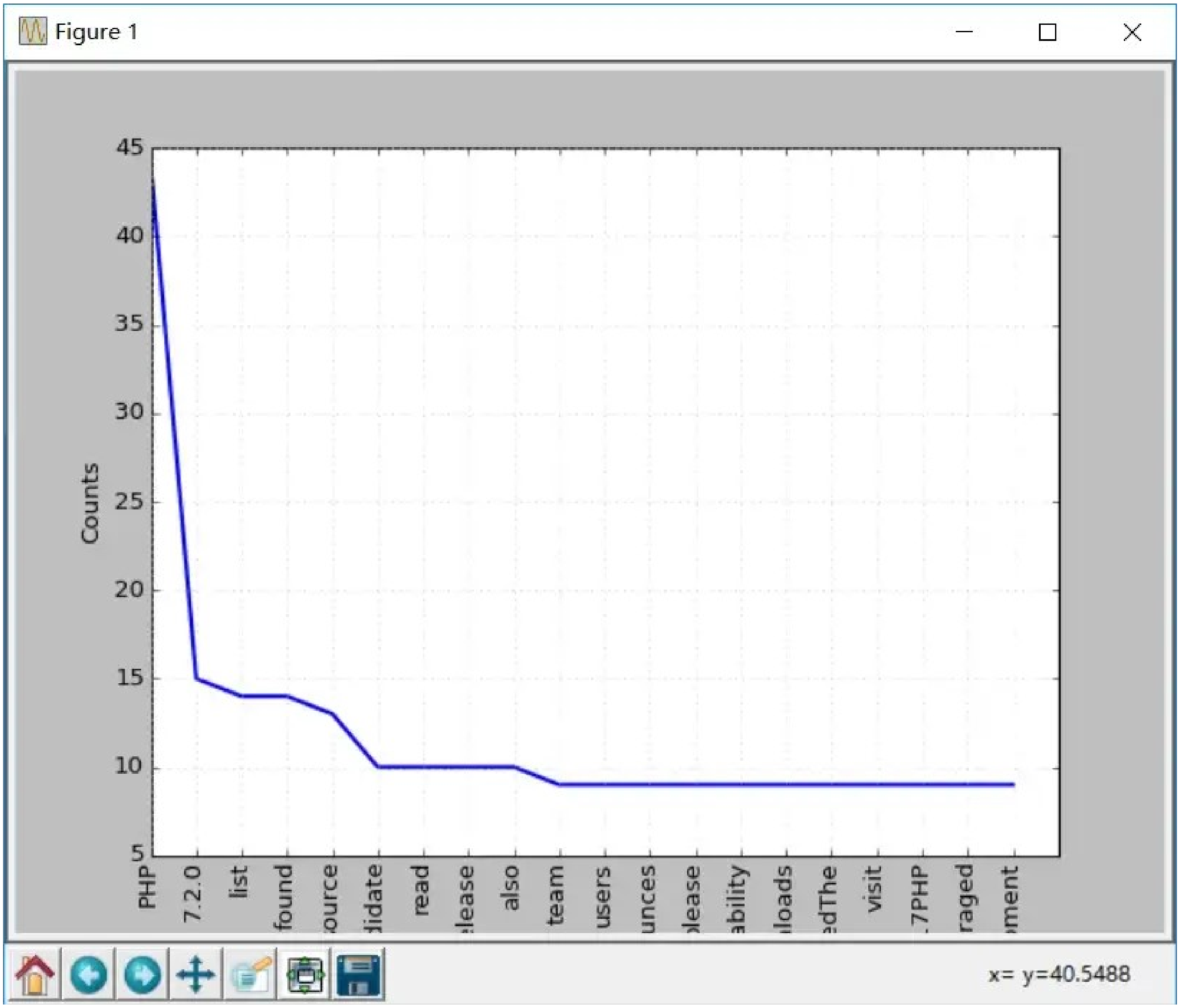
```python
from bs4 import BeautifulSoup
import urllib.request
import nltk
from nltk.corpus import stopwords

response = urllib.request.urlopen('http://php.net/')
html = response.read()
soup = BeautifulSoup(html,"html5lib")
text = soup.get_text(strip=True)
tokens = text.split()
clean_tokens = list()
sr = stopwords.words('english')
for token in tokens:
    if not token in sr:
        clean_tokens.append(token)
freq = nltk.FreqDist(clean_tokens)
for key,val in freq.items():
    print (str(key) + ':' + str(val))
```

nlp10.py hosted with ❤ by GitHub view raw

Now do another word frequency chart, the effect will be better than before, because the stop words are removed:

```python
freq.plot(20, cumulative=False)
# need to install matplotlib library
```

nlp7.py hosted with ❤ by GitHub view raw



8. Use NLTK Tokenize text

Before we used the `split` method to split the text into tokens, now we use NLTK to tokenize the text.

Text cannot be processed before Tokenize, so it is very important to make Tokenize the text. The tokenization process means dividing large parts into widgets.

You can tokenize the paragraph into a sentence, tokenize the sentence into a single word, and NLTK provides the sentence tokenizer and the word tokenizer, respectively.

If there is such a text:

```
Hello Adam, how are you? I hope everything is going well. Today is a good day, see you dude.
```

nlp11.py hosted with ❤ by **GitHub** **view raw**

Use the token tokenizer to tokenize the text into a sentence:

```
from nltk.tokenize import sent_tokenize

mytext = "Hello Adam, how are you? I hope everything is going well. Today is a good day, see you
print(sent_tokenize(mytext))
```

nlp12.py hosted with ❤ by **GitHub** **view raw**

The output is as follows:

```
['Hello Adam, how are you?', 'I hope everything is going well.', 'Today is a good day, see you d
```

nlp13.py hosted with ❤ by **GitHub** **view raw**

This is something you might think, it's too simple, you don't need to use NLTK's tokenizer, you can use regular expressions to split sentences, because each sentence has punctuation and spaces.

Then look at the following text:

```
Hello Mr. Adam, how are you? I hope everything is going well. Today is a good day, see you dude.
```

nlp14.py hosted with ❤ by **GitHub** **view raw**

So if you use punctuation to split, it `Hello Mr` will be considered a sentence if you use NLTK:

```
from nltk.tokenize import sent_tokenize

mytext = "Hello Mr. Adam, how are you? I hope everything is going well. Today is a good day, see
print(sent_tokenize(mytext))

Output:
['Hello Mr. Adam, how are you?', 'I hope everything is going well.', 'Today is a good day, see y
```

nlp15.py hosted with ❤ by **GitHub** **view raw**

This is the correct split.

Next try the word tokenizer:

```
from nltk.tokenize import word_tokenize

mytext = "Hello Mr. Adam, how are you? I hope everything is going well. Today is a good day, see
print(word_tokenize(mytext))

Output:
['Hello', 'Mr.', 'Adam', ',', 'how', 'are', 'you', '?',
 'I', 'hope', 'everything', 'is', 'going', 'well', '.',
 'Today', 'is', 'a', 'good', 'day', ',', 'see', 'you', 'dude', '.']
```

nlp16.py hosted with ❤ by **GitHub** **view raw**

`Mr.` The word has not been separated. NLTK uses the PunktSentenceTokenizer of the punkt module, which is part of NLTK.tokenize. And this tokenizer is trained to work in multiple languages.

9.Non-English Tokenize

Tokenize can specify the language:

```
from nltk.tokenize import sent_tokenize

mytext = "Bonjour M. Adam, comment allez-vous? J'espère que tout va bien. Aujourd'hui est un bon
print(sent_tokenize(mytext,"french"))

Output:
['Bonjour M. Adam, comment allez-vous?', "J'espère que tout va bien.", "Aujourd'hui est un bon j
```

nlp17.py hosted with ❤ by **GitHub** **view raw**

10. Synonym processing

Using the `nltk.download()` installation interface, one of the packages is WordNet.

WordNet is a database built for natural language processing. It includes some synonym groups and some short definitions.

You can get the definition and example of a given word like this:

```
from nltk.corpus import wordnet

syn = wordnet.synsets("pain")
print(syn[0].definition())
print(syn[0].examples())

Output:

a symptom of some physical hurt or disorder
['the patient developed severe pain and distension']
```

nlp18.py hosted with ❤ by **GitHub** **view raw**

WordNet contains a lot of definitions:

```
from nltk.corpus import wordnet

syn = wordnet.synsets("NLP")
print(syn[0].definition())
syn = wordnet.synsets("Python")
print(syn[0].definition())

Output:

the branch of information science that deals with natural language information
large Old World boas
```

nlp19.py hosted with ❤ by **GitHub** **view raw**

You can use WordNet to get synonyms like this:

```python
from nltk.corpus import wordnet

synonyms = []
for syn in wordnet.synsets('Computer'):
    for lemma in syn.lemmas():
        synonyms.append(lemma.name())
print(synonyms)

Output:

['computer', 'computing_machine', 'computing_device', 'data_processor', 'electronic_computer',
```

nlp20.py hosted with ❤ by GitHub view raw

11.Antonym processing

You can also get the antonym in the same way:

```python
from nltk.corpus import wordnet

antonyms = []
for syn in wordnet.synsets("small"):
    for l in syn.lemmas():
        if l.antonyms():
            antonyms.append(l.antonyms()[0].name())
print(antonyms)

Output:
['large', 'big', 'big']
```

nlp21.py hosted with ❤ by GitHub view raw

12.Stem extraction

In **morphological morphology** and **information retrieval**, **stemming** is the process of removing the affixes to get the roots. For example, the working stem is work.

Search engines use this technique when indexing pages, so many people write different versions of the same word.

There are many algorithms to avoid this, the most common being the **Boolean stem algorithm** . NLTK has a class called PorterStemmer, which is the implementation of this algorithm:

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
print(stemmer.stem('working'))
print(stemmer.stem('worked'))

Output:

work
work
```

nlp22.py hosted with ❤ by GitHub view raw

There are other stem extraction algorithms, such as the **Lancaster stem algorithm** .

13.Non-English stem extraction

In addition to English, SnowballStemmer also supports 13 languages.

Supported languages:

```
from nltk.stem import SnowballStemmer

print(SnowballStemmer.languages)

Output:
'danish', 'dutch', 'english', 'finnish', 'french', 'german', 'hungarian', 'italian',
'norwegian', 'porter', 'portuguese', 'romanian', 'russian', 'spanish', 'swedish'
```

nlp23.py hosted with ❤ by GitHub view raw

You can use `SnowballStemmer` a `stem` function of the class to extract non-English words like this:

```
from nltk.stem import SnowballStemmer

french_stemmer = SnowballStemmer('french')

print(french_stemmer.stem("French word"))
```

nlp24.py hosted with ❤ by **GitHub** **view raw**

14. Word variant reduction

A word variant restore is similar to a stem, but the difference is that the result of a variant restore is a real word. Unlike stemming, when you try to extract certain words, it produces similar words:

```
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

print(stemmer.stem('increases'))

Output:

increas
```

nlp25.py hosted with ❤ by **GitHub** **view raw**

Now, if you use NLTK's WordNet to perform a variant restore of the same word, the correct result:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('increases'))

Output:
increase
```

nlp26.py hosted with ❤ by **GitHub** **view raw**

The result may be a synonym or a different word of the same meaning.

Sometimes when you restore a word to a variant, you always get the same word.

This is because the default part of the language is a noun. To get a verb, you can specify it like this:

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize('playing', pos="v"))

Output:
play
```

nlp27.py hosted with ❤ by **GitHub** **view raw**

In fact, this is also a good way to compress text, and finally get the text from the original 50% to 60%.

The result can also be a verb (v), a noun (n), an adjective (a), or an adverb (r):

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize('playing', pos="v"))
print(lemmatizer.lemmatize('playing', pos="n"))
print(lemmatizer.lemmatize('playing', pos="a"))
print(lemmatizer.lemmatize('playing', pos="r"))

Output :
play
playing
playing
playing
```

nlp28.py hosted with ❤ by **GitHub** **view raw**

15. The difference between stems and variants

Observe by the following example:

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
print(stemmer.stem('stones'))
print(stemmer.stem('speaking'))
NLP  ir  Ml Research Lab  o  Text Processing    Naturallanguageprocessing
print(stemmer.stem('jokes'))
Beginners Guide  stem('lisa'))
print(stemmer.stem('purple'))
print('---------------------')
print(lemmatizer.lemmatize('stones'))
print(lemmatizer.lemmatize('speaking'))
print(lemmatizer.lemmatize('bedroom'
print(lemmatizer.lemmatize('jokes'),
print(lemmatizer.lemmatize('lisa'))
print(lemmatizer.lemmatize('purple'))

Output:
stone
speak
bedroom
joke
lisa
purpl
---------------------
stone
speaking
bedroom
joke
lisa
purple
```

587 1

nlp29.py hosted with ❤ by GitHub view ra

Stem extraction does not consider context, which is why stemming is faster and less accurate than variant reduction.

Personally think that variant reduction is better than stem extraction. A word variant restores a real word, even if it is not the same word, synonymous, but at least it is a real word.

If you only care about speed and don't care about accuracy, then you can use stemming.

All the steps discussed in this NLP tutorial are just text preprocessing. In a future article, Python NLTK will be used to implement text analysis.

I have tried to make the article easy to understand. hope it is of help to you.

References:

1. http://www.dataversity.net/natural-language-processing/
2. https://www.wired.com/insights/2014/02/growing-importance-natural-language-processing/