

Stemming and Lemmatization in Python

datacamp.com/community/tutorials/stemming-lemmatization-python



GAIN THE CAREER-BUILDING PYTHON
SKILLS YOU NEED WITH DATA CAMP

Start Learning Now

Stemming and Lemmatization are **Text Normalization** (or sometimes called **Word Normalization**) techniques in the field of **Natural Language Processing** that are used to prepare text, words, and documents for further processing. Stemming and Lemmatization have been studied, and algorithms have been developed in Computer Science since the 1960's. In this tutorial you will learn about Stemming and Lemmatization in a practical approach covering the background, some famous algorithms, applications of Stemming and Lemmatization, and how to stem and lemmatize words, sentences and documents using the Python `nltk` package which is the **Natural Language Tool Kit** package provided by Python for Natural Language Processing tasks.

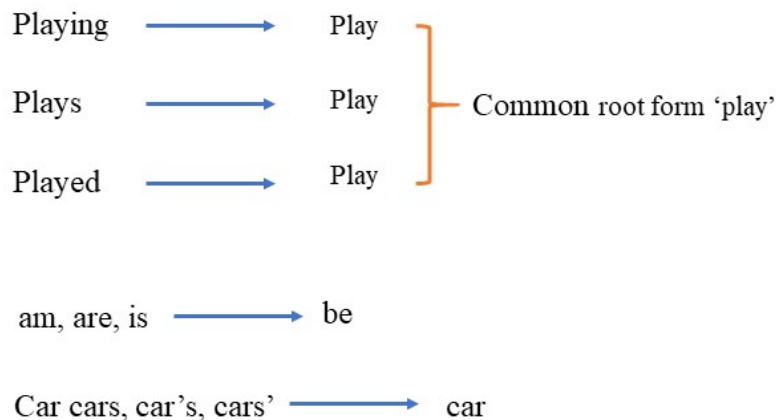
1. Background
2. Stemming with Python `nltk` package
 - What is Python `nltk` Package
3. Lemmatization with Python `nltk` package
4. Applications of Stemming and Lemmatization

Background

Languages we speak and write are made up of several words often derived from one another. When a language contains words that are derived from another word as their use in the speech changes is called **Inflected Language**.

"In grammar, inflection is the modification of a word to express different grammatical categories such as tense, case, voice, aspect, person, number, gender, and mood. An inflection expresses one or more grammatical categories with a **prefix, suffix or infix**, or another internal modification such as a vowel change" [Wikipedia]

The degree of inflection may be higher or lower in a language. As you have read the definition of inflection with respect to grammar, you can understand that an inflected word(s) will have a common root form. Let's look at a few examples,



Using above mapping a sentence could be normalized as follows:

the boy's cars are different colors → the boy car be differ color



Above examples must have helped you understand the concept of normalization of text, although normalization of text is not restricted to only written document but to speech as well. Stemming and Lemmatization helps us to achieve the root forms (sometimes called synonyms in search context) of inflected (derived) words. *Stemming is different to Lemmatization in the approach it uses to produce root forms of words and the word produced.*

Stemming and Lemmatization are widely used in **tagging systems, indexing, SEOs, Web search results, and information retrieval**. For example, searching for *fish* on Google will also result in *fishes, fishing* as *fish* is the stem of both words. Later in this tutorial, you will go through some of the significant uses of Stemming and Lemmatization in applications.

Stemming with Python `nltk` package

"Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language."

Stem (root) is the part of the word to which you add inflectional (changing/deriving) affixes such as (-ed,-ize, -s,-de,mis). So stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.

Information: Removing suffixes from a word is called **Suffix Stripping**

What is Python `nltk` package?

Natural Language Tool Kit (NLTK) is a Python library to make programs that work with natural language. It provides a user-friendly interface to datasets that are over 50 corpora and lexical resources such as WordNet Word repository. The library can perform different operations such as tokenizing, stemming, classification, parsing, tagging, and semantic reasoning.

The latest version is **NLTK 3.3**. It can be used by students, researchers, and industrialists. It is an Open Source and free library. It is available for **Windows, Mac OS, and Linux**.

Installing python `nltk`

NLTK requires Python versions **2.7, 3.4, 3.5, or 3.6**. You can install `nltk` using *pip installer* if it is not installed in your Python installation. To test the installation:

- open your Python IDE or the CLI interface (whichever you use normally)
- Type `import nltk` and press enter if no message of missing nltk is shown then `nltk` is installed on your computer.

Mac Os/LINX Installation

run `sudo pip install -U nltk` to install `nltk` on Mac or Linux

Windows

run `pip install nltk` on your cmd.exe bash to install `nltk` on Windows.

Now after installation, you can use the `nltk` library for Stemming and Lemmatization using Python.

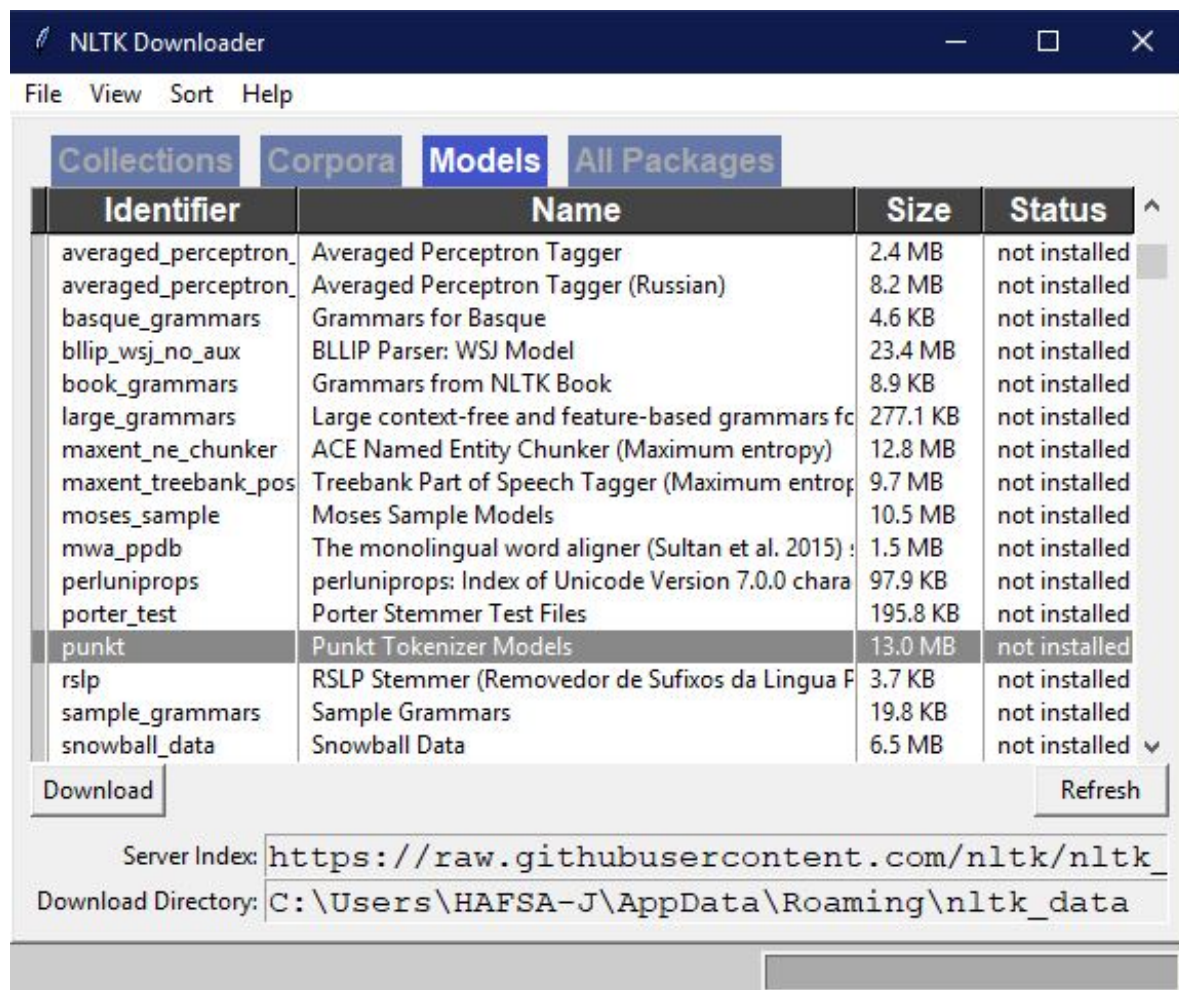
After installation, `nltk` also provides test datasets to work within Natural Language Processing. You can download it by using the following commands in Python:

```
#import the nltk package
import nltk
#call the nltk downloader
nltk.download()
```

showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

True

`nltk.download()` will call a Graphical Window will appear to choose different corpus and datasets to choose from.



Click on **Models** tab and select **punkt** and click **Download**. You will need this model later in this tutorial.

Stemming Algorithms and Code

There are English and Non-English Stemmers available in **nltk** package.

A computer program or subroutine that stems word may be called a stemming program, stemming algorithm, or stemmer.

This tutorial will see different stemmers available in different languages in Python **nltk**. For the English language, you can choose between **PorterStamner** or **LancasterStamner**, PorterStemmer being the oldest one originally developed in 1979. LancasterStemmer was developed in 1990 and uses a more aggressive approach than Porter Stemming Algorithm. Let's try out the PorterStemmer to stem words, and along with it you will see how it is stemming the words. This tutorial will not go deep into the algorithm of the Porter Stemmer and LancasterStemmer also known as (Paice-Husk Stemmer), but you will see their advantages and disadvantages.

Stemming Words

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
```

`nltk.stem` is a package that performs stemming using different classes. PorterStemmer is one of the classes, so we import it using the above line of code.

```
#create an object of class PorterStemmer
porter = PorterStemmer()
lancaster=LancasterStemmer()
#provide a word to be stemmed
print("Porter Stemmer")
print(porter.stem("cats"))
print(porter.stem("trouble"))
print(porter.stem("troubling"))
print(porter.stem("troubled"))
print("Lancaster Stemmer")
print(lancaster.stem("cats"))
print(lancaster.stem("trouble"))
print(lancaster.stem("troubling"))
print(lancaster.stem("troubled"))
```

```
Porter Stemmer
cat
troubl
troubl
troubl
Lancaster Stemmer
cat
troubl
troubl
troubl
```

PorterStemmer uses **Suffix Stripping** to produce stems. Notice how the PorterStemmer is giving the root (stem) of the word "cats" by simply removing the 's' after cat. This is a suffix added to cat to make it plural. But if you look at 'trouble', 'troubling' and 'troubled' they are stemmed to 'trouble' because ***PorterStemmer algorithm does not follow linguistics rather a set of 05 rules for different cases that are applied in phases (step by step) to generate stems***. This is the reason why PorterStemmer does not often generate stems that are actual English words. It does not keep a lookup table for actual stems of the word but applies algorithmic rules to generate stems. It uses the rules to decide whether it is wise to strip a suffix. One can generate its own set of rules for any language that is why Python `nltk` introduced `SnowballStemmers` that are used to create non-English Stemmers!

So Why use it? **PorterStemmer is known for its simplicity and speed.** It is commonly useful in **Information Retrieval** Environments known as **IR Environments** for fast recall and fetching of search queries. In a typical IR, environment documents are represented as vectors of words or terms. Words having the same stem will have a similar meaning. For example,

```
CONNECT
CONNECTIONS-----> CONNECT
CONNECTED-----> CONNECT
```

CONNECTING-----> CONNECT

CONNECTION-----> CONNECT

Try out the following in your Python environment:

```
#A list of words to be stemmed
word_list = ["friend", "friendship", "friends",
"friendships", "stabil", "destabilize", "misunderstanding", "railroad", "moonlight", "football"]
print("{0:20}{1:20}{2:20}".format("Word", "Porter Stemmer", "Lancaster Stemmer"))
for word in word_list:
    print("{0:20}{1:20}{2:20}".format(word, porter.stem(word), lancaster.stem(word)))
```

Word	Porter Stemmer	Lancaster Stemmer
friend	friend	friend
friendship	friendship	friend
friends	friend	friend
friendships	friendship	friend
stabil	stabil	stabl
destabilize	destabil	dest
misunderstanding	misunderstand	misunderstand
railroad	railroad	railroad
moonlight	moonlight	moonlight
football	footbal	footbal

The **LancasterStemmer (Paice-Husk stemmer)** is an iterative algorithm with rules saved externally. One table containing about 120 rules indexed by the last letter of a suffix. On each iteration, it tries to find an applicable rule by the last character of the word. Each rule specifies either a deletion or replacement of an ending. If there is no such rule, it terminates. It also terminates if a word starts with a vowel and there are only two letters left or if a word starts with a consonant and there are only three characters left. Otherwise, the rule is applied, and the process repeats.

LancasterStemmer is simple, but heavy stemming due to iterations and over-stemming may occur. Over-stemming causes the stems to be not linguistic, or they may have no meaning.

For example, in above code **destabilized** is stemmed to **dest** in LancasterStemmer whereas, using PorterStemmer **destabl**. LancasterStemmer produces an even shorter stem than porter because of iterations and over-stemming is occurred.

Stemming Sentences

You can stem sentences and documents using **nltk** stemmers. You can stem sentences as follows:

```
sentence="Pythoners are very intelligent and work very pythonly and now they are pythoning their way to success."
porter.stem(sentence)
```

```
'pythoners are very intelligent and work very pythonly and now they are pythoning their way to success.'
```

As you see the stemmer sees the entire sentence as a word, so it returns it as it is. We need to stem each word in the sentence and return a combined sentence. To separate the sentence into words, you can use **tokenizer**. The `nltk` tokenizer separates the sentence into words as follows. You can create a function and just pass the sentence to the function, and it will give you the stemmed sentence.

```
from nltk.tokenize import sent_tokenize, word_tokenize
def stemSentence(sentence):
    token_words=word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)

x=stemSentence(sentence)
print(x)
```

python are veri intellig and work veri pythonli and now they are python their way to success .

Stemming a document

You can write your own function that can stem documents. Here is one way to stem a document using Python filing:

1. Take a document as the input.
2. Read the document line by line
3. Tokenize the line
4. Stem the words
5. Output the stemmed words (print on screen or write to a file)
6. Repeat step 2 to step 5 until it is to the end of the document.

Let's do some coding! Open a file, any text file. I have a text file named '*data-science-wiki.txt*' in a folder named '*Stemming and Lemmatization*' in my working directory of the Python Notebook. You have to provide your complete file path in `open()` command of Python if it stored in any other directory. You can learn about reading and writing files in Python in detail [here](#).

```
file=open("Stemming and Lemmatization\data-science-wiki.txt")
file.read()
```


'Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured,[1][2] similar to data mining. \nData science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data.[3] It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science. \nTuring award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.\n\nIn 2012, when Harvard Business Review called it "The Sexiest Job of the 21st Century",[6] the term "data science" became a buzzword. It is now often used interchangeably with earlier concepts like business analytics,[7] business intelligence, predictive modeling, and statistics. In many cases, earlier approaches and solutions are now simply rebranded as "data science" to be more attractive, which can cause the term to become "dilute[d] beyond usefulness."While many university programs now offer a data science degree, there exists no consensus on a definition or suitable curriculum contents.To its discredit, however, many data-science and big-data projects fail to deliver useful results, often as a result of poor management and utilization of resources. '

You can see the content of the file using the `.read()` method altogether. You can maintain the lines in a file in a Python list using `.readlines()` . You can then use the list to access each line and tokenize and stem the selected line.

```
file=open("Stemming and Lemmatization\data-science-wiki.txt")
my_lines_list=file.readlines()
my_lines_list
```

['Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured,[1][2] similar to data mining. \n',

'Data science is a "concept to unify statistics, data analysis, machine learning and their related methods" in order to "understand and analyze actual phenomena" with data.[3] It employs techniques and theories drawn from many fields within the context of mathematics, statistics, information science, and computer science. \n',

'Turing award winner Jim Gray imagined data science as a "fourth paradigm" of science (empirical, theoretical, computational and now data-driven) and asserted that "everything about science is changing because of the impact of information technology" and the data deluge.\n',

'Data Science is now often used interchangeably with earlier concepts like business analytics,[7] business intelligence, predictive modeling, and statistics. In many cases, earlier approaches and solutions are now simply rebranded as "data science" to be more attractive, which can cause the term to become "dilute[d] beyond usefulness."While many university programs now offer a data science degree, there exists no consensus on a definition or suitable curriculum contents.To its discredit, however, many data-science and big-data projects fail to deliver useful results, often as a result of poor management and utilization of resources. ']

You can now access each line and use the tokenize 'stemSentence()' function you created before to tokenize and stem the line.


```
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer
```

```
porter=PorterStemmer()
```

```
def stemSentence(sentence):
    token_words=word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(porter.stem(word))
    stem_sentence.append(" ")
    return "".join(stem_sentence)
```

```
print(my_lines_list[0])
print("Stemmed sentence")
x=stemSentence(my_lines_list[0])
print(x)
```

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from data in various forms, both structured and unstructured,[1][2] similar to data mining.

Stemmed sentence

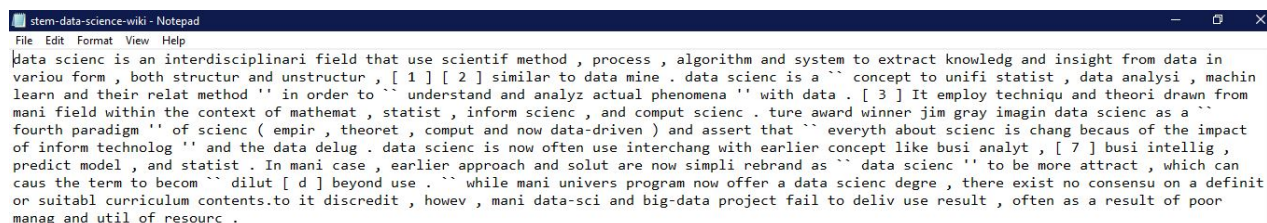
data scienc is an interdisziplinari field that use scientif method , process , algorithm and system to extract knowledg and insight from data in variou form , both structur and unstructur , [1] [2] similar to data mine .

You can save the stemmed sentence to a text file using Python `writelines()` function. Make a list first to store all the stemmed sentences and simply write the list to the file using `writelines()` .

```
stem_file=open("Stemming and Lemmatization\stem-data-science-wiki.txt",mode="a+",
encoding="utf-8")
for line in my_lines_list:
    stem_sentence=stemSentence(line)
    stem_file.write(stem_sentence)

stem_file.close()
```

The text file created will be as follows:



NLTK Corpus and Lexical Resources

In this section of the tutorial, you will learn about the NLTK corpora and how to use it.

You can use the **NLTK Text Corpora** which is a vast repository for a large body of text called as a **Corpus** which can be used while you are working with Natural Language Processing (NLP) with Python. There are many different types of corpora available that you can use with varying types of projects, for example, a selection of free electronic books, web and chat text and news documents on different genres. [Here](#) you can see how to work with different corpora available in the Python NLTK package, and the useful code is also provided that you can use in your projects.

If you have not worked with NLP before in Python, it is likely that you don't have any copora installed on your machine. You can run the `nltk.download()` command and use the downloader to install desired corpora in the `corpora` tab.

Note: Change the server address in the NLTK downloader by clicking on `File->Change server address` and paste http://nltk.org/nltk_data/ in the server address text box; otherwise you may not be able to download the corpora.

Here is an example of how you can use a corpora and stem that document:

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

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
showing info http://nltk.org/nltk_data/
```

True

```
nltk.corpus.gutenberg.fileids()
```

```
['austen-emma.txt',
'austen-persuasion.txt',
'austen-sense.txt',
'bible-kjv.txt',
'blake-poems.txt',
'bryant-stories.txt',
'burgess-busterbrown.txt',
'carroll-alice.txt',
'chesterton-ball.txt',
'chesterton-brown.txt',
'chesterton-thursday.txt',
'edgeworth-parents.txt',
'melville-moby_dick.txt',
'milton-paradise.txt',
'shakespeare-caesar.txt',
'shakespeare-hamlet.txt',
'shakespeare-macbeth.txt',
'whitman-leaves.txt']
```

You can use any of the above text file for stemming. Try it out like below:

```
text_file=nltk.corpus.gutenberg.words('melville-moby_dick.txt')
my_lines_list=[]
for line in text_file:
    my_lines_list.append(line)
my_lines_list
```

[['',

'Moby','Dick','by','Herman','Melville','1851',''],'ETYMOLOGY',:,'(','Supplied','by','a','Late','Consumptive',
-
,',threadbare','in','coat',:,'heart',:,'body',:,'and','brain',:,'I','see','him','now',:,'He','was','ever','dustin
,',fish','is','to','be','called','in','our','tongue','leaving','out',:,'through','ignorance',:,'the','letter','H',:,'w
-
,',HACKLUYT',:,'','WHALE',:,'...','Sw',:,'and','Dan',:,'HVAL',:,'This','animal','is','named','from','roundnes
-
,',WEBSTER',:,'','S','DICTIONARY',:,'','WHALE',:,'...','It','is','more','immediately','from','the','Dut',:,'and','
,',IAN',:,'','to','roll',:,'','to','wallow',:,'','--
,',RICHARDSON',:,'','S','DICTIONARY','KETOS',:,'','GREEK',:,'','CETUS',:,'','LATIN',:,'','WHOEL',:,'','ANGLO',-
,',SAXON',:,'','HVALT',:,'','DANISH',:,'','WAL',:,'','DUTCH',:,'','HWAL',:,'','SWEDISH',:,'','WHALE',:,'','ICELANDIC',:,'
,',NUEE',:,'','NUEE',:,'','FEGEE',:,'','PEKEE',:,'','NUEE',-
,',NUEE',:,'','ERROMANGOAN',:,'','EXTRACTS',(','Supplied','by','a','Sub',:,'','Sub',-
,',Librarian',).',',It','will','be','seen','that','this','mere','painstaking','burrower','and','grub',-
,',worm','of','a','poor','devil','of','a','Sub',-
,',Sub','appears','to','have','gone','through','the','long','Vaticans','and','street',-
,',stalls','of','the','earth',:,'','picking','up','whatever','random','allusions','to','whales','he','could','anyways
,',piggledy','whale','statements',:,'','however','authentic',:,'','in','these','extracts',:,'','for','veritable','gospel
,',Sub',:,'','whose','commentator','I','am',:,'','Thou','belongest','to','that','hopeless',:,'','sallow','tribe','which
,',strong',:,'','but','with','whom','one','sometimes','loves','to','sit',:,'','and','feel','poor',-
,',devilish',:,'','too',:,'','and','grow','convivial','upon','tears',:,'','and','say','to','them','bluntly',:,'','with','full','e
-',',Give','it','up',:,'','Sub',-
,',Subs',!,'','For','by','how','much','the','more','pains','ye','take','to','please','the','world',:,'','by','so','much'
,',mast','with','your','hearts',:,'','for','your','friends','who','have','gone','before','are','clearing','out','the','s
,',storied','heavens',:,'','and','making','refugees','of','long',-
,',pampered','Gabriel',:,'','Michael',:,'','and','Raphael',:,'','against','your','coming',:,'','Here','ye','strike','but',
-
,',there',:,'','ye','shall','strike','unsplinterable','glasses',!,'','EXTRACTS',:,'','And','God','created','great','w
-
,',GENESIS',:,'','','Leviathan','maketh','a','path','to','shine','after','him',:,'','One','would','think','the','deep'
-',',JOB',:,'','','Now','the','Lord','had','prepared','a','great','fish','to','swallow','up','Jonah',:,'','--
,',JONAH',:,'','','There','go','the','ships',:,'','there','is','that','Leviathan','whom','thou','hast','made','to','play
-
,',PSALMS',:,'','','In','that','day',:,'','the','Lord','with','his','sore',:,'','and','great',:,'','and','strong','sword',:,'','s
-
,',ISAIAH',:,'','','And','what','thing','soever','besides','cometh','within','the','chaos','of','this','monster',:,'','s',
-
,',HOLLAND',:,'','S','PLUTARCH',:,'','S','MORALS',:,'','','The','Indian','Sea','breedeth','the','most','and','the'
-
,',HOLLAND',:,'','S','PLINY',:,'','','Scarcely','had','we','proceeded','two','days','on','the','sea',:,'','when','abo
,',mouthed',:,'','raising','the','waves','on','all','sides',:,'','and','beating','the','sea','before','him','into','a','fo
-
,',TOOKE',:,'','S','LUCIAN',:,'','','THE','TRUE','HISTORY',:,'','','He','visited','this','country','also','with','a','vi
,',whales',:,'','which','had','bones','of','very','great','value','for','their','teeth',:,'','of','which','he','brought','s
,',eight',:,'','some','fifty','yards','long',:,'','He',...]

You can read the lines and save the lines in a Python list like above and use the list for stemming like demonstrated in the section above.

Non-English Stemmers

Python `nltk` provides not only two English stemmers: **PorterStemmer** and **LancasterStemmer** but also a lot of non-English stemmers as part of **SnowballStemmers, ISRIStemmer, RSLPStemmer**. Python NLTK included SnowballStemmers as a language to create to create non-English stemmers. One can program one's own language stemmer using snowball. Currently, it supports the following languages:

SnowballStemmers

- Danish
- Dutch
- English
- French
- German
- Hungarian
- Italian
- Norwegian
- Porter
- Portuguese
- Romanian
- Russian
- Spanish
- Swedish

ISRIStemmer is an Arabic stemmer and **RSLPStemmer** is stemmer for the Portuguese Language.

In this section, you will learn how to use SnowballStemmer, and then you can further incorporate what you learned in the above sections to make detailed code.

```
from nltk.stem.snowball import SnowballStemmer
```

```
englishStemmer=SnowballStemmer("english")  
englishStemmer.stem("having")
```

```
'have'
```

You can also tell the stemmer to ignore stop-words.

Stop words: Stop Words are words which do not contain important significance to be used in Search Queries. Usually, these words are filtered out from search queries because they return a vast amount of unnecessary information. Each programming language will give its own list of stop words to use. Mostly they are words that are commonly used in the English language such as 'as, the, be, are' etc.

You also need to download the `stopwords corpus` from using `nltk.download()` like you downloaded the `gutenberg` corpora above. Otherwise, a **Resource not found error** will be given.

```
englishStemmer2=SnowballStemmer("english", ignore_stopwords=True)
englishStemmer2.stem("having")
```

showing info http://nltk.org/nltk_data/

'having'

You can see, that before using `ignore_stopwords=True` *having* was stemmed to *have* but after using it, it is ignored by the stemmer.

```
spanishStemmer=SnowballStemmer("spanish", ignore_stopwords=True)
spanishStemmer.stem("Corriendo")
```

'corr'

This is all about Stemming in Python using NLTK Package. You will now learn about **Lemmatization** in the next section.

Lemmatization with Python `nltk` package

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called **Lemma**. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

For example, *runs*, *running*, *ran* are all forms of the word *run*, therefore *run* is the lemma of all these words. Because lemmatization returns an actual word of the language, it is used where it is necessary to get valid words.

Python NLTK provides **WordNet Lemmatizer** that uses the WordNet Database to lookup lemmas of words.

Note: Download the WordNet corpora from NLTK downloader before using the WordNet Lemmatizer

```

import nltk
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

sentence = "He was running and eating at same time. He has bad habit of swimming after playing long hours in the Sun."
punctuations="?!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)

sentence_words
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))

```

Word	Lemma
He	He
was	wa
running	running
and	and
eating	eating
at	at
same	same
time	time
He	He
has	ha
bad	bad
habit	habit
of	of
swimming	swimming
after	after
playing	playing
long	long
hours	hour
in	in
the	the
Sun	Sun

In the above output, you must be wondering that no actual root form has been given for any word, this is because they are given without context. You need to provide the context in which you want to lemmatize that is the parts-of-speech (POS). This is done by giving the value for `pos` parameter in `wordnet_lemmatizer.lemmatize` .

```

for word in sentence_words:
    print("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word, pos="v")))

```


He	He
was	be
running	run
and	and
eating	eat
at	at
same	same
time	time
He	He
has	have
bad	bad
habit	habit
of	of
swimming	swim
after	after
playing	play
long	long
hours	hours
in	in
the	the
Sun	Sun

Applications of Stemming and Lemmatization

Stemming and Lemmatization are itself form of NLP and widely used in Text mining. Text Mining is the process of analysis of texts written in natural language and extract high-quality information from text. It involves looking for interesting patterns in the text or to extract data from the text to be inserted into a database. Text mining tasks include **text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities)**. Developers have to prepare text using lexical analysis, POS (Parts-of-speech) tagging, stemming and other Natural Language Processing techniques to gain useful information from text.

Information Retrieval (IR) Environments:

It is useful to use stemming and lemmatization to map documents to common topics and display search results by indexing when documents are increasing to mind-boggling numbers. **Query Expansion** is a term used in Search Environments which refers to that when a user inputs a query. It is used to expand or enhance the query to match additional documents.

Stemming has been used in Query systems such as Web Search Engines, but due to problems of under-stemming and over-stemming it's effectiveness in returning correct results have been found limited. For example, a person searching for 'marketing' may not be pleased with results that will show 'markets' and not marketing. But Stemming may be found useful in other languages and using different algorithms for stemming may result in better outputs. Google search adopted stemming in 2003.

Sentiment Analysis

Sentiment Analysis is the analysis of people's reviews and comments about something. It is widely used for analysis of product on online retail shops. Stemming and Lemmatization is used as part of the text-preparation process before it is analyzed.

Document Clustering

Document clustering (or text clustering) is the application of cluster analysis to textual documents. It has applications in an automatic document organization, topic extraction, and fast information retrieval or filtering. Examples of document clustering include web document clustering for search engines. Before Clustering methods are applied document is prepared through tokenization, removal of stop words and then *Stemming and Lemmatization* to reduce the number of tokens that carry out the same information and hence speed up the whole process. After this pre-processing, features are calculated by calculating the frequency of all tokens and then clustering methods are applied.

Stemming or lemmatization?

After going through the entire tutorial, you may be asking yourself when should I use Stemming and when should I use Lemmatization? The answer itself is in whatever you have learned from this tutorial. You have seen the following points:

- Stemming and Lemmatization both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word.
- Stemming follows an algorithm with steps to perform on the words which makes it faster. Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming. You also had to define a parts-of-speech to obtain the correct lemma.

So when to use what! The above points show that if speed is focused then stemming should be used since lemmatizers scan a corpus which consumed time and processing. It depends on the application you are working on that decides if stemmers should be used or lemmatizers. If you are building a language application in which language is important you should use lemmatization as it uses a corpus to match root forms.

That's the end of the tutorial! In this tutorial, you learned about NLP, Python NLTK package, how to use the package and how to use the lexical resources in Python. You learned about Stemming, Lemmatization, their applications and how you can use them in Python NLP applications. Happy Learning!

If you would like to learn more about Natural Language Processing in Python, take DataCamp's [Natural Language Processing Fundamentals in Python](#) course.

References