FREE eBook

LEARNING nltk

Free unaffiliated eBook created from
Stack Overflow contributors.

#nltk

Table of Contents

About ... 1

Chapter 1: Getting started with nltk ... 2

Remarks ... 2

The book ... 2

Versions ... 2

NLTK Version History ... 2

Examples ... 2

With NLTK ... 2

Installation or Setup ... 3

NLTK's download function ... 3

NLTK installation with Conda ... 4

Basic Terms ... 5

Corpus ... 5

Lexicon ... 5

Token ... 5

Chapter 2: Frequency Distributions ... 7

Introduction ... 7

Examples ... 7

Frequency Distribution to Count the Most Common Lexical Categories ... 7

Chapter 3: POS Tagging ... 8

Introduction ... 8

Remarks ... 8

Important points to note ... 8

Examples ... 8

Basic Example ... 8

Chapter 4: Stemming ... 9

Introduction ... 9

Examples ... 9

Porter stemmer ... 9

Chapter 5: Stop Words 11
Introduction 11
Examples 11
Filtering out stop words 11
Chapter 6: Tokenizing 12
Introduction 12
Examples 12
Sentence and word tokenization from user given paragraph 12
Credits 13

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: nltk

It is an unofficial and free nltk ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official nltk.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with nltk

Remarks

NLTK is a leading platform for building **Python** programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

The book

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed .)

Versions

NLTK Version History

Version	Release Date
3.2.4 (**latest**)	2017-05-21
3.2	2016-03-03
3.1	2015-10-15

Examples

With NLTK

You can use NLTK (especially, the `nltk.tokenize` package) to perform sentence boundary detection:

```
import nltk
text = "This is a test. Let's try this sentence boundary detector."
text_output = nltk.tokenize.sent_tokenize(text)
print('text_output: {0}'.format(text_output))
```

Output:

```
text_output: ['This is a test.', "Let's try this sentence boundary detector."]
```

Installation or Setup

NLTK requires `Python` versions **2.7** or **3.4+**.

These instructions consider `python` version - **3.5**

- **Mac/Unix :**

  1. Install NLTK: run `sudo pip install -U nltk`
  2. Install Numpy (optional): run `sudo pip install -U numpy`
  3. Test installation: run `python` then type `import nltk`

  NOTE : For older versions of Python it might be necessary to install setuptools (see http://pypi.python.org/pypi/setuptools) and to install pip (sudo easy_install pip).

- **Windows :**

  These instructions assume that you do not already have Python installed on your machine.

  32-bit binary installation

  1. Install Python 3.5: http://www.python.org/downloads/ (avoid the 64-bit versions)
  2. Install Numpy (optional): http://sourceforge.net/projects/numpy/files/NumPy/ (the version that specifies pythnon3.5)
  3. Install NLTK: http://pypi.python.org/pypi/nltk
  4. Test installation: `Start>Python35`, then type `import nltk`

- **Installing Third-Party Software :**

  Please see: https://github.com/nltk/nltk/wiki/Installing-Third-Party-Software

---

Reference : http://www.nltk.org/install.html

NLTK's download function

You can install NLTK over `pip` (`pip install nltk`).After it is installed, many components will not be present, and you will not be able to use some of NLTK's features.

From your Python shell, run the function `ntlk.download()` to select which additional packages you want to install using UI. Alternatively, you can use `python -m nltk.downloader [package_name].`

- To download all packages available.

```
nltk.download('all')
```

- To download specific package.

```
nltk.download('package-name')
```

- To download all packages of specific folder.

```
import nltk

dwlr = nltk.downloader.Downloader()

# chunkers, corpora, grammars, help, misc,
# models, sentiment, stemmers, taggers, tokenizers
for pkg in dwlr.packages():
    if pkg.subdir== 'taggers':
        dwlr.download(pkg.id)
```

- To download all packages except Corpora Folder.

```
import nltk

dwlr = nltk.downloader.Downloader()

for pkg in dwlr.corpora():
    dwlr._status_cache[pkg.id] = 'installed'

dwlr.download('all')
```

NLTK installation with Conda.

To install NLTK with Continuum's `anaconda` / `conda`.

If you are using Anaconda, most probably nltk would be already downloaded in the root (though you may still need to download various packages manually).

Using `conda`:

```
conda install nltk
```

To upgrade `nltk` using `conda`:

```
conda update nltk
```

With `anaconda`:

If you are using multiple python envriroments in anaconda, first activate the enviroment where you

want to install nltk. You can check the active enviroment using the command

```
conda info --envs
```

The enviroment with the * sign before the directory path is the active one. To change the active enviroment use

```
activate <python_version>
for eg. activate python3.5
```

Now check the list of packages installed in this enviroment using commnad

```
conda list
```

If you dont find 'nltk' in the list, use

```
conda install -c anaconda nltk=3.2.1
```

For further information, you may consult https://anaconda.org/anaconda/nltk.

To install mini-conda a.k.a. `conda`: http://conda.pydata.org/docs/install/quick.html

To install `anaconda`: https://docs.continuum.io/anaconda/install

Basic Terms

Corpus

Body of text, singular. Corpora is the plural of this. Example: A collection of medical journals.

Lexicon

Words and their meanings. Example: English dictionary. Consider, however, that various fields will have different lexicons. For example: To a financial investor, the first meaning for the word "Bull" is someone who is confident about the market, as compared to the common English lexicon, where the first meaning for the word "Bull" is an animal. As such, there is a special lexicon for financial investors, doctors, children, mechanics, and so on.

Token

Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph.

Read Getting started with nltk online: https://riptutorial.com/nltk/topic/4077/getting-started-with-nltk

Chapter 2: Frequency Distributions

Introduction

This topic focuses on the use of the nltk.FreqDist() class.

Examples

Frequency Distribution to Count the Most Common Lexical Categories

NLTK provides the FreqDist class that let's us easily calculate a frequency distribution given a list as input.

Here we are using a list of part of speech tags (POS tags) to see which lexical categories are used the most in the brown corpus.

```
import nltk

brown_tagged = nltk.corpus.brown.tagged_words()
pos_tags = [pos_tag for _,pos_tag in brown_tagged]


fd = nltk.FreqDist(pos_tags)
print(fd.most_common(5))

# Out: [('NN', 152470), ('IN', 120557), ('AT', 97959), ('JJ', 64028), ('.', 60638)]
```

We can see that Nouns are the most common lexical category. Frequency Distributions can be accessed just like dictionaries. So by doing this we can calculate what percentage of the words in the brown corpus are nouns.

```
print(fd['NN'] / len(pos_tags))
# Out: 0.1313
```

Read Frequency Distributions online: https://riptutorial.com/nltk/topic/9318/frequency-distributions

Chapter 3: POS Tagging

Introduction

Part of speech tagging creates **tuples** of words and parts of speech. It labels words in a sentence as nouns, adjectives, verbs,etc. It can also label by tense, and more. These tags mean whatever they meant in your original training data. You are free to invent your own tags in your training data, as long as you are consistent in their usage. Training data generally takes a lot of work to create, so a pre-existing corpus is typically used. These usually use the Penn Treebank and Brown Corpus.

Remarks

Important points to note

- The variable **word** is a list of tokens.
- Even though item **i** in the list **word** is a token, tagging single token will tag each letter of the word.
- nltk.tag.pos_tag_ accept a
    - **list of tokens** -- then separate and tags its elements or
    - **list of string**
- You can not get the tag for one word, instead you can put it within a list.
- POS tag

Examples

Basic Example

```
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
text = 'We saw the yellow dog'
word = word_tokenize(text)
tag1 = nltk.pos_tag(word)
print(tag1)
```

Read POS Tagging online: https://riptutorial.com/nltk/topic/10028/pos-tagging

Chapter 4: Stemming

Introduction

Stemming is a sort of normalizing method. Many variations of words carry the same meaning, other than when tense is involved. The reason why we stem is to shorten the lookup, and normalize sentences. Basically, it is finding the root of words after removing verb and tense part from it. One of the most popular stemming algorithms is the Porter stemmer, which has been around since 1979.

Examples

Porter stemmer

1. Import `PorterStemmer` and initialize

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
ps = PorterStemmer()
```

2. Stem a list of words

```
example_words = ["python","pythoner","pythoning","pythoned","pythonly"]

for w in example_words:
    print(ps.stem(w))
```

Result:

```
python
python
python
python
pythonli
```

3. Stem a sentence after tokenizing it.

```
 new_text = "It is important to by very pythonly while you are pythoning with python. All
pythoners have pythoned poorly at least once."

 word_tokens = word_tokenize(new_text)
 for w in word_tokens:
     print(ps.stem(w))    # Passing word tokens into stem method of Porter Stemmer
```

Result:

```
 It
 is
```

```
import
to
by
veri
pythonli
while
you
are
python
with
python
.
all
python
have
python
poorli
at
least
onc
.
```

Read Stemming online: https://riptutorial.com/nltk/topic/8793/stemming

Chapter 5: Stop Words

Introduction

Stop words are the words which are mostly used as fillers and hardly have any useful meaning. We should avoid these words from taking up space in database or taking up valuable processing time. We can easily make a list of words to be used as stop words and then filter these words from the data we want to process.

Examples

Filtering out stop words

NLTK has by default a bunch of words that it considers to be stop words. It can be accessed via the NLTK corpus with:

```
from nltk.corpus import stopwords
```

To check the list of stop words stored for english language :

```
stop_words = set(stopwords.words("english"))
print(stop_words)
```

Example to incorporate the stop_words set to remove the stop words from a given text:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = "This is a sample sentence, showing off the stop words filtration."
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if not w in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

Read Stop Words online: https://riptutorial.com/nltk/topic/8750/stop-words

Chapter 6: Tokenizing

Introduction

It refers to the splitting of sentences and words from the body of text into sentence tokens or word tokens respectively. It is an essential part of NLP, as many modules work better (or only) with tags. For example, **pos_tag** needs *tags* as input and not the words, to tag them by parts of speech.

Examples

Sentence and word tokenization from user given paragraph

```
from nltk.tokenize import sent_tokenize, word_tokenize
example_text = input("Enter the text:  ")

print("Sentence Tokens:")
print(sent_tokenize(example_text))

print("Word Tokens:")
print(word_tokenize(example_text))
```

Read Tokenizing online: https://riptutorial.com/nltk/topic/8749/tokenizing

Credits

S. No	Chapters	Contributors
1	Getting started with nltk	alvas, Ares, Community, Franck Dernoncourt, hongsy, j_4321, JGreenwell, Mike Driscoll, Preston Hager, Pythonista, RAVI, Sameer Sinha
2	Frequency Distributions	Daniel Palenicek
3	POS Tagging	Sameer Sinha
4	Stemming	hongsy, Sameer Sinha
5	Stop Words	Sameer Sinha
6	Tokenizing	Sameer Sinha