Natural Language Processing Programming

Tokenization

Tutorial

Introduction (1)

1. Natural Language Processing (NLP) has become an important part of modern systems. It is used extensively in search engines, conversational interfaces, document processors, and so on.
2. Machines can handle structured data well. But when it comes to working with free-form text, they have a hard time.

Introduction (2)

1. The goal of NLP is to develop algorithms that enable computers to understand freeform text and help them understand language.
2. One of the most challenging things about processing freeform natural language is the sheer number of variations. The context plays a very important role in how a particular sentence is understood.

Introduction (3)

1. We immediately use our past knowledge to understand the context and know what the other person is talking about.
2. To address this issue, NLP researchers started developing various applications using machine learning approaches.

Introduction (4)

1. To build such applications, we need to collect a large corpus of text and then train the algorithm to perform various tasks like categorizing text, analyzing sentiments, or modeling topics.
2. These algorithms are trained to detect patterns in input text data and derive insights from it.

Introduction (5)

1. **Corpus -** Body of text, singular. Corpora is the plural of this.

   Example: A collection of medical journals.

2. **Lexicon -** Words and their meanings. 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.

3. **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.

Outline of Tasks

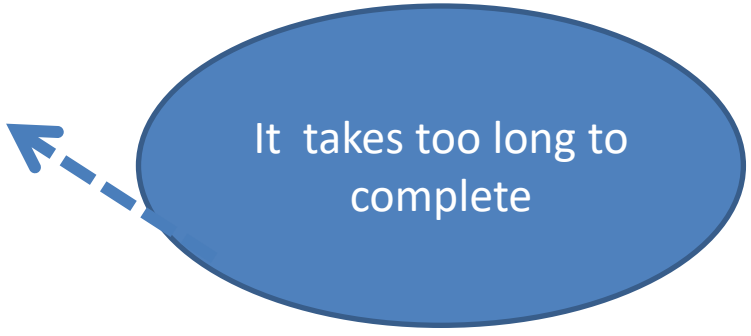
Next, we will study how to install relevant packages .

import nltk

nltk.download()

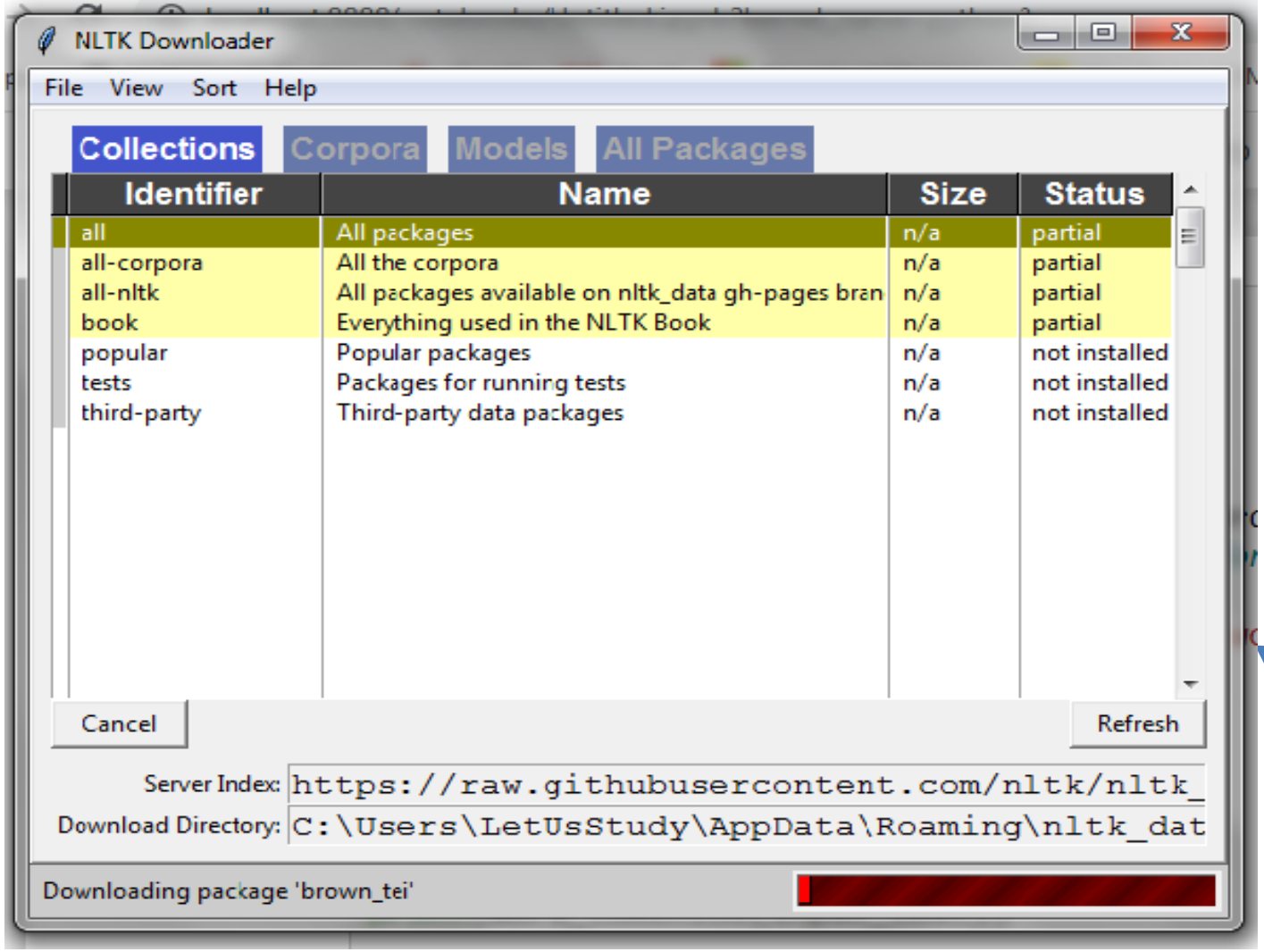
d (for download)

all (for download everything)



Outline of Tasks

Screen shots for downloading window



It takes too long to complete

Outline of Tasks

1. Tokenizing text data.
2. Tokenizing text data Converting words to their base forms using **stemming .**
3. Converting words to their base forms using **lemmatization**

Tokenizer

```
#Examples of Tokenizers

from nltk.tokenize import sent_tokenize, word_tokenize,
WordPunctTokenizer


#Define some input text that will be used for tokenization:

input_text = """Do you know how tokenization works? It's
actually quite interesting!  Let's analyze a couple of
sentences and figure it out."“"
```

Tokenizer

```
# Sentence tokenizer
print("\nSentence tokenizer:")
print(sent_tokenize(input_text))
# Word tokenizer
print("\nWord tokenizer:")
print(word_tokenize(input_text))
# WordPunct tokenizer
print("\nWord punct tokenizer:")
print(WordPunctTokenizer().tokenize(input_text))
```

What is Output ?

STEMMING vs. LEMMATIZATION

1. Problems with Stemming

   Overstemming and Understemming

2. **Examples :**

   (a) Overstemming comes from when too much of a word is cut off. words university, universal, universities, and universe have overstemmed as “univers” which has no meaning.

   (b) the words data and datum to “dat” and “datu.” is called understemming.

STEMMING vs. LEMMATIZATION

LEMMATIZATION

1. lemmatization is a more calculated process. It involves resolving words to their dictionary form. In fact, a lemma of a word is its dictionary or canonical form!
2. **Examples :**

   like resolving is and are to “be”

STEMMING vs. LEMMATIZATION

STEMMING

1. The Porter stemmer is the least in terms of strictness and Lancaster is the strictest.
2. If you closely observe the outputs, you will notice the differences. Stemmers behave differently when it comes to words like possibly or provision. The stemmed outputs that are obtained from the Lancaster stemmer are a bit obfuscated because it reduces the words a lot. At the same time, the algorithm is really fast.
3. A good rule of thumb is to use the Snowball stemmer because it's a good trade off between speed and strictness.

STEMMING

```
#Examples of Stemmers

from nltk.stem import PorterStemmer

from nltk.stem import LancasterStemmer


#create an object of class PorterStemmer

porter = PorterStemmer()

lancaster=LancasterStemmer()
```

STEMMING

```
#proide a word to be stemmed

print("Porter Stemmer")

print(porter.stem("troubling"))

print("Lancaster Stemmer")

print(lancaster.stem("troubling"))
```

What is Output ?

LEMMATIZATION

To implement lemmatization using the following python packages.

1. Wordnet Lemmatizer
2. Spacy Lemmatizer
3. TextBlob
4. CLiPS Pattern
5. Stanford CoreNLP
6. Gensim Lemmatizer
7. TreeTagger

LEMMATIZATION

```
# import these modules
from   nltk.stem    import    WordNetLemmatizer
  lemmatizer = WordNetLemmatizer()
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos ="a"))
```

What is Output ?

LEMMATIZATION

```
import  nltk From  nltk.stem
import  WordNetLemmatizer
# Define the sentence to be lemmatized
sentence = "The striped bats are hanging on their feet for best“
# Tokenize: Split the sentence into words
word_list = nltk.word_tokenize(sentence)
print(word_list)
# Lemmatize list of words and join
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for  w  in  word_list])
print(lemmatized_output)
```

What is Output ?