Theory: Overview of NLTK

11 minutes 0 / 5 problems solved

NLTK (short for the *Natural Language Toolkit*) is a Python library for natural language processing (NLP). It provides several modules for various language-related tasks, including part-of-speech tagging, syntactic parsing, text classification, named-entity recognition, etc. The library includes a lot of datasets and pre-trained models that are available for free. It is designed to support NLP researchers and learners in the field. Apart from its practical application, NLTK is also good for getting acquainted with the techniques used in computational linguistics.

§1. Installation

To begin working with NLTK, we need to install it first. It can be easily done with pip:

```
pip install nltk
```

Now, If we want to use it, we just need to import it at the beginning of our program:

```
import nltk
```

Once you have installed the library, you may also want to download some **external datasets and models** that are not included in the initial NLTK set, but still are the part of the toolkit. The datasets include, for instance, collections of classic literary works, samples of web conversations, movie reviews, as well as various lexical resources like sets of synonyms. As for the models, NLTK provides several grammars, the pre-trained word2vec model which allows you to find out relations between words, pre-trained models for sentiment analysis, and so forth. The whole list is available on the official NLTK site (NLTK Data). To obtain the resources, you need to use `download()`:

```
nltk.download()
```

Calling the method without any arguments should open the NLTK Downloader window. You can select the required data manually there. To obtain the entire collection, you need to choose 'all' in the Collections tab. Alternatively, you can simply type `'all'` as the function argument, which will also result in getting the entire set of resources:

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

Any single package or collection of packages in NLTK can be downloaded in the same way. Their IDs are the arguments of `nltk.download()`, as in the example above.

§2. Advantages and Disadvantages

Earlier, we have noticed that NLTK, owing to its mainly academic nature, is a great starting point of studying NLP. The documentation is clear, easy to follow, and comes with a lot of usage examples. We would like to mention some other advantages of the toolkit:

- NLTK is well suited for NLP tasks,
- All of its external resources are easily accessible and all the models are trained on reliable datasets,
- The textual resources are often supplied with annotations.

However, there are some restrictions on the NLTK functionality.

- For some tasks, NLTK is not a good choice,

Current topic:

Overview of NLTK

Topic depends on:

✓ Invoking a function Stage 1 16

✗ Natural language processing

✗ Pip Stage 1

Topic is required for:

Tokenization

Table of contents:

↑ Overview of NLTK

§1. Installation

§2. Advantages and Disadvantages

§3. Possible applications of NLTK

§4. Summary

Feedback & Comments

- Its built-in models are not very powerful (but they are still good as a starting point),
- Even though the library supports some standard machine learning techniques, it does not provide any tools for neural networks training.

To sum up, NLTK is a perfect tool for a quick analysis of your data; it can also help with pre-processing for some of your further tasks.

§3. Possible applications of NLTK

Now, we are going to take a quick look at the most common applications of NLTK. Most of them will be discussed in detail later, so just the basics. All the modules we refer to below are listed in the table:

Language processing task	NLTK modules
String processing	tokenize, stem
Accessing corpora	corpus
Collocation discovery	collocations
Part-of-speech tagging	tag
Syntactic analysis	chunk, parse
Machine learning	classify, cluster
Evaluation metrics	metrics
Probability and estimation	probability

Let's start with pre-processing. There are several procedures that usually take place before any further data processing takes place. The first one is **tokenization**. It is the task of breaking some raw textual data into smaller units (words, phrases, or any other entities that can be considered meaningful). The second step is either **lemmatization** or **stemming**. Roughly speaking, both these techniques amount to word normalization and reducing various word forms to accepted base or root ones. The difference between them will be discussed a bit later. NLTK has special modules for these procedures: `nltk.tokenize` for the first one, and `nltk.stem` for lemmatization and stemming.

Some additional pre-processing may take place, removing some of the high-frequency words from our data, for example. This is quite often done as those words have no value for such tasks as text classification or topic modeling. NLTK contains wordlists of commonly used words for a number of languages. Such words are called **stopwords** and can be found in `nltk.corpus.stopwords`. With the help of the same `corpus` module, you can get access to other corpora NLTK provides.

The library can be successfully used for other specific tasks, for example, **collocation discovery**. Collocations are two or more words that appear frequently together ('best friend', 'make breakfast', 'save time', etc.). Such phrases can be extracted utilizing several metrics accessed from `nltk.collocations`.

The other task, which can be performed using this library, is **part-of-speech tagging**. The annotation is done using the pre-trained model included in NLTK. It also has tools for **chunking**, the procedure is closely related to part-

of-speech tagging. It aims to identify syntactically related constituents of sentences, such as noun phrases. Unlike plain part-of-speech tagging, chunking provides limited insight into the syntactic structure of a given text. **Parsing** is designed to assist you with deep analysis of the text syntactic organization. NLTK contains the module which allows you to produce tree representations of the inner structures of sentences.

The library also contains modules for **text classification and clustering**, providing some basic machine learning techniques. To evaluate the performance of your NLP tasks, you can use the **metrics** provided in NLTK.

What is more, NLTK has tools for **statistical counting**. Most of them are included in the `FreqDist` class of the `nltk.probability` package. For example, you can learn about word frequency distributions in your text.

§4. Summary

In this topic, we have learned how to install the library and download its external resources, what are the advantages and disadvantages of the library, and outlined some of the modules that can be used for natural language processing tasks. Of course, NLTK provides much more possibilities and you can explore them by looking at the NLTK documentation.