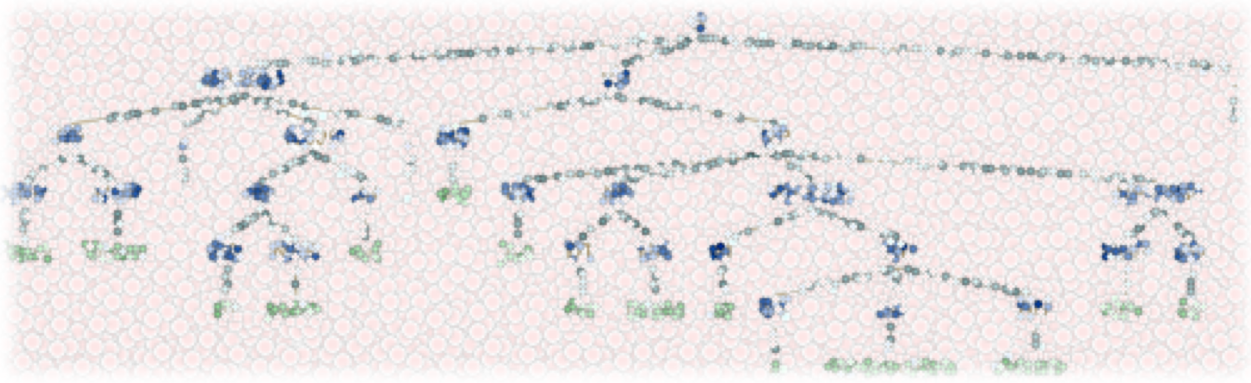
An amazing library to play with natural languages

Natural Language Toolkit - NLTK

Sandeep Kumar | Net-id: kumar64 | CS410 Text Information Systems | University of Illinois, Urbana-Champaign



Picture: Artistic view of a parse tree using NLTK treebank. source: nltk.org

Introduction

Natural Language Toolkit (NLTK) is one of the most commonly used libraries for building solutions using Python programming language to natural language processing challenges e.g. predictive text analysis, email filtering, news summarization, etc., and is a leading open-source platform to work with human language data using Python. NLTK provides user-friendly 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.

History of NLTK

NLTK was designed to overcome the challenges associated with teaching computational linguistics to students. Before NLTK, it was a widespread practice to employ multiple programming languages, where each language provides native data structures and functions that are a good fit for the task at hand. For example, a user might use Prolog for parsing, Perl for corpus processing, and a finite-state toolkit for morphological analysis. This led to a significant focus on learning multiple programming languages. NLTK was developed to streamline and organize the practical components of introductory computational linguistics in a flexible way.

During the design and development of the toolkit, the following criteria shaped the toolkit requirements. The toolkit provides a wide range of functions but it does not cover "Everything" and the toolkit continues to evolve.

- **Ease of Use**: Focus should be on building NLP systems rather than learning to use the toolkit.
- **Consistency**: The toolkit should use consistent data structures and interfaces
- **Extensibility**: The toolkit should easily accommodate new components
- **Documentation**: The toolkit, its data structures, and its implementation all should be thoroughly documented
- **Simplicity**: The toolkit should structure the complexities of building the NLP Systems and not hide them.
- **Modularity:** The interaction between different components of the toolkit should be kept to a minimum, using simple, well-defined interfaces.

NLTK Design

NLTK is implemented as a large collection of minimally interdependent modules, organized into a shallow hierarchy. A set of core modules defines the basic data types that are used throughout the toolkit. The remaining modules are task modules devoted to an individual natural language processing task. For example, the `nltk.parser` module encompasses the task of parsing or deriving the syntactic structure of a sentence; and the

`nltk.tokenizer` module is devoted to the task of tokenizing, or dividing a text into its constituent parts. Table N1 lists the most important and popular NLTK modules.

Language Processing task	NLTK modules	Functionality
Accessing Corpora	nltk.corpus	Standardized interfaces to corpora and lexicons
String Processing	nltk.tokenize, nltk.stem	Tokenizers, sentence tokenizers, stemmers
Collocation discovery	nltk.collocations	t-test, chi-squared, point-wise mutual information
Part-of-speech tagging	nltk.tag	n-gram, backoff, Brill, HMM, TnT
Classification	nltk.classify, nltk.cluster	Decision tree, maximum entropy, naïve Bayes. EM, k-means
Chunking	nltk.chunk	Regular expression, n-gram, named entity
Parsing	nltk.parse	Chart, feature-based, unification, probabilistic, dependency
Semantic interpretation	nltk.sem, nltk.inference	Lambda calculus, first-order logic, model checking
Evaluation metrics	nltk.metrics	Precision, recall, agreement coefficients
Probability and Estimation	nltk.probability	Frequency distributions, smoothed probability distributions
Applications	nltk.app, nltk.chat	Graphical concordance, parsers, WordNet browser, chatbots
Linguistic fieldwork	nltk.toolbox	Manipulate data in SIL Toolbox format

Table N1: Language processing tasks and corresponding NLTK modules with examples of functionality

Toolkit Installation

NLTK requires Python versions 3.5, 3.6, 3.7, or 3.8

Mac/Unix¶

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

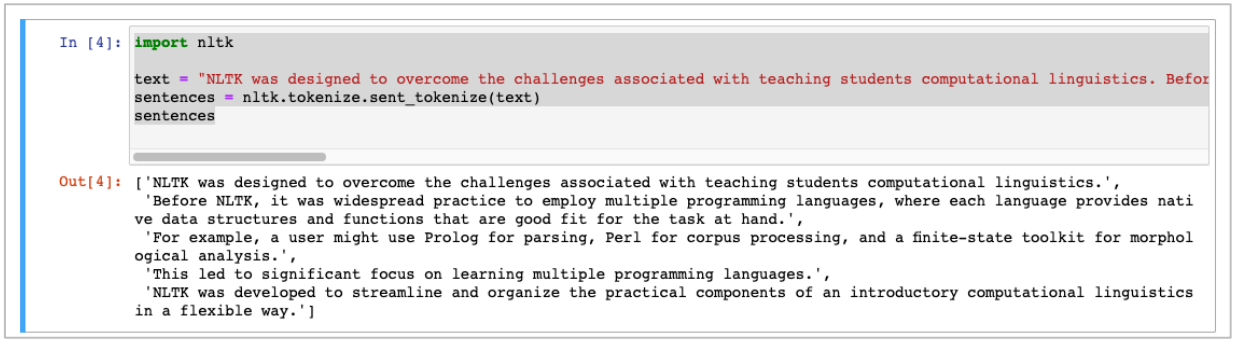
Windows¶

1. Install Python 3.8: http://www.python.org/downloads/ (avoid the 64-bit versions)
2. Install Numpy (optional): https://www.scipy.org/scipylib/download.html
3. Install NLTK: http://pypi.python.org/pypi/nltk
4. Test installation: Start>Python38, then type import nltk

Examples of NLTK modules usage

1. EOS Detection

   The End of Speech (EOS) tagging breaks a text into a collection of meaningful sentences, needed for further processing. Below is an example of EOS detection.



Picture: EOS detection using NLTK in Jupyter notebook. Sample sentence is the first paragraph of this document.

2. Tokenization

   This step operates on individual sentences, splitting them into tokens. Below is an example of tokenization and is a continuation of the previous step.

```
In [5]: tokens = [nltk.tokenize.word_tokenize(s) for s in sentences]
        print(tokens)

[['NLTK', 'was', 'designed', 'to', 'overcome', 'the', 'challenges', 'associated', 'with', 'teaching', 'students', 'co
mputational', 'linguistics', '.'], ['Before', 'NLTK', ',', 'it', 'was', 'widespread', 'practice', 'to', 'employ', 'mu
ltiple', 'programming', 'languages', ',', 'where', 'each', 'language', 'provides', 'native', 'data', 'structures', 'a
nd', 'functions', 'that', 'are', 'good', 'fit', 'for', 'the', 'task', 'at', 'hand', '.'], ['For', 'example', ',',
'a', 'user', 'might', 'use', 'Prolog', 'for', 'parsing', ',', 'Perl', 'for', 'corpus', 'processing', ',', 'and', 'a',
'finite-state', 'toolkit', 'for', 'morphological', 'analysis', '.'], ['This', 'led', 'to', 'significant', 'focus', 'o
n', 'learning', 'multiple', 'programming', 'languages', '.'], ['NLTK', 'was', 'developed', 'to', 'streamline', 'and',
'organize', 'the', 'practical', 'components', 'of', 'an', 'introductory', 'computational', 'linguistics', 'in', 'a',
'flexible', 'way', '.']]
```

Picture: Tokenization using NLTK in Jupyter notebook. Sample sentence is the first paragraph of this document.

3. POS tagging

POS means part-of-speech tagging and is used to assign POS information to the sentence tokens. Example ‘VBD’ indicates a verb, ‘JJ’ indicates an adjective. Below is an example of POS tagging and is a continuation of the previous step.

```
In [8]: PosTokens = [nltk.pos_tag(e) for e in tokens]
        print(PosTokens)

[[('NLTK', 'NNP'), ('was', 'VBD'), ('designed', 'VBN'), ('to', 'TO'), ('overcome', 'VB'), ('the', 'DT'), ('challenge
s', 'NNS'), ('associated', 'VBN'), ('with', 'IN'), ('teaching', 'VBG'), ('students', 'NNS'), ('computational', 'JJ'),
('linguistics', 'NNS'), ('.', '.')], [('Before', 'IN'), ('NLTK', 'NNP'), (',', ','), ('it', 'PRP'), ('was', 'VBD'),
('widespread', 'JJ'), ('practice', 'NN'), ('to', 'TO'), ('employ', 'VB'), ('multiple', 'JJ'), ('programming', 'NN'),
('languages', 'NNS'), (',', ','), ('where', 'WRB'), ('each', 'DT'), ('language', 'NN'), ('provides', 'VBZ'), ('nativ
e', 'JJ'), ('data', 'NNS'), ('structures', 'NNS'), ('and', 'CC'), ('functions', 'NNS'), ('that', 'WDT'), ('are', 'VB
P'), ('good', 'JJ'), ('fit', 'NN'), ('for', 'IN'), ('the', 'DT'), ('task', 'NN'), ('at', 'IN'), ('hand', 'NN'), ('.',
'.')], [('For', 'IN'), ('example', 'NN'), (',', ','), ('a', 'DT'), ('user', 'NN'), ('might', 'MD'), ('use', 'VB'),
('Prolog', 'NNP'), ('for', 'IN'), ('parsing', 'VBG'), (',', ','), ('Perl', 'NNP'), ('for', 'IN'), ('corpus', 'NN'),
('processing', 'NN'), (',', ','), ('and', 'CC'), ('a', 'DT'), ('finite-state', 'JJ'), ('toolkit', 'NN'), ('for', 'I
N'), ('morphological', 'JJ'), ('analysis', 'NN'), ('.', '.')], [('This', 'DT'), ('led', 'VBD'), ('to', 'TO'), ('signi
ficant', 'JJ'), ('focus', 'NN'), ('on', 'IN'), ('learning', 'VBG'), ('multiple', 'JJ'), ('programming', 'NN'), ('lang
uages', 'NNS'), ('.', '.')], [('NLTK', 'NNP'), ('was', 'VBD'), ('developed', 'VBN'), ('to', 'TO'), ('streamline', 'V
B'), ('and', 'CC'), ('organize', 'VB'), ('the', 'DT'), ('practical', 'JJ'), ('components', 'NNS'), ('of', 'IN'), ('a
n', 'DT'), ('introductory', 'JJ'), ('computational', 'JJ'), ('linguistics', 'NNS'), ('in', 'IN'), ('a', 'DT'), ('flex
ible', 'JJ'), ('way', 'NN'), ('.', '.')]]
```

Picture: POS tagging using NLTK in Jupyter notebook. Sample sentence is the first paragraph of this document.

4. Chunking and extraction

Chunking is the process of assembling complex tokens based on tags. NLTK also allows custom grammar for chunking. Extraction is the process to tag the chunks as named entities e.g. people, organizations, locations, etc.

```
In [18]: chunks = nltk.ne_chunk_sents(PosTokens)
         for each in chunks:
             print(each)

(S
  (ORGANIZATION NLTK/NNP)
  was/VBD
  designed/VBN
  to/TO
  overcome/VB
  the/DT
  challenges/NNS
  associated/VBN
  with/IN
  teaching/VBG
  students/NNS
  computational/JJ
  linguistics/NNS
  ./.)
(S
  Before/IN
  (ORGANIZATION NLTK/NNP)
  ,/,
```

Picture: Chunking and extraction using NLTK in Jupyter notebook. Sample sentence is the first paragraph of this document.

User community and contributing to NLTK

NLTK is an open-source project and welcomes any contribution. There are several ways to contribute: users can report bugs, suggest features, or contribute patches on Sourceforge; users can participate in discussions on the nltk-dev mailing list or the NLTK public forums, and users can submit their NLTK-based projects for inclusion in the nltk contribution directory. New code modules that are relevant, substantial, original, and well-documented will be considered for inclusion in the NLTK project. All source code is distributed under the GNU General Public License, and all documentation is distributed under a Creative Commons non-commercial license. Thus, potential contributors can be confident that their work will remain freely available to all.

Contribution platform

Google groups	https://groups.google.com/forum/#!forum/nltk-dev
NLTK repo	https://github.com/nltk/nltk
Corpus repo	https://github.com/nltk/nltk/wiki/Adding-a-Corpus

Conclusion

NLTK provides a simple, extensible, and uniform framework for natural language processing tasks. The toolkit is thoroughly documented, easy to learn, and simple to use and is widely used for academic assignments, demonstrations, and project work. It does not support “everything” but covers a wide range of functionalities and is continuously evolving – thanks to its extensible and modular design and community support.

References

- *https://www.nltk.org*
- *http://www.nltk.org/book/*
- *https://www.researchgate.net/publication/220482883_NLTK_the_Natural_Language_Toolkit*
- *https://www.researchgate.net/publication/220873131_NLTK_The_natural_language_toolkit*
- *Bird, Steven, Edward Loper, and Ewan Klein (2009), Natural Language Processing with Python. O’Reilly Media Inc.*