CS410 - Technology Review

Topic: [Tools] NLTK for information extraction

Yi Hao Tan

yihaoht2@illinois.edu

Introduction

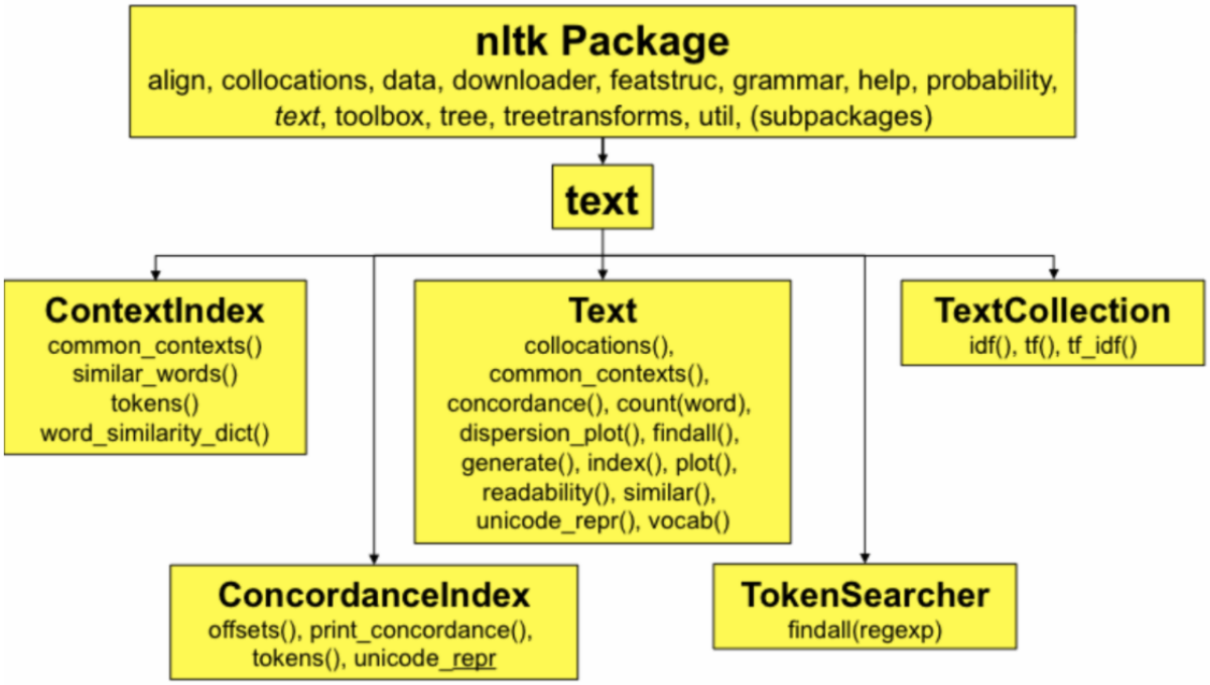
Natural language processing (NLP) focuses on analyzing human generated language data, giving computers the ability to read texts, extract keywords and phrases and understand the intent of the text. This essentially allows computers to communicate with humans and perform language related tasks.

NLP deals with a variety of tasks. Some examples are as follow:

- **Lexical Analysis**: Assign tags such as Verb, Noun and Prep to each tokenized word
- **Syntactic evaluation**: Parse, evaluate and arrange tags to build valid sentence structure
- **Semantic evaluation**: Map the precise meaning to each word to understand content
- **Pragmatic evaluation**: Perform inference on the content to a limited extent

Architecture and usage of NLTK

NLTK, the Natural Language Toolkit, is a suite of open source Python modules for working with human language. Although NLTK has many algorithms, modules and functions, its simple interface results in it widely used as a research tool or a tool for learning. Based on its methods of text analysis, NLTK's functionality is organized into separate modules, resulting in a modular architecture.


nltk Package
align, collocations, data, downloader, featstruc, grammar, help, probability, *text*, toolbox, tree, treetransforms, util, (subpackages)
text
ContextIndex
common_contexts()
similar_words()
tokens()
word_similarity_dict()
Text
collocations(),
common_contexts(),
concordance(), count(word),
dispersion_plot(), findall(),
generate(), index(), plot(),
readability(), similar(),
unicode_repr(), vocab()
TextCollection
idf(), tf(), tf_idf()
ConcordanceIndex
offsets(), print_concordance(),
tokens(), unicode_repr
TokenSearcher
findall(regexp)

Illustrating the organization of 'text' sub-package and its modules. Source: Howard 2016, fig. 3.

NLTK consists of the most common algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition.

Comparison of NLTK against spaCy

Another popular library is spaCy, an open-source python NLP library. It has the ability to handle a large number of text data, and it was intended for developers to build real-world projects. With NLTK and spaCy, you can theoretically accomplish any NLP task, from building chatbots, to search engines and more. The two are the most popular NLP libraries, but are intended for different types of developers. NLTK is generally used as a learning tool, or by researchers to build something from scratch, while spaCy is catered to developers for production usage.

NLTK has a wide array of algorithms, but its numerous libraries present a challenge for developers to keep them up to date. Whereas, spaCy keeps the latest and the best algorithm for a problem in its toolkit, having a smaller scale of updates needed .

As spaCy uses the most up to date and best algorithms, its performance is on par with NLTK. However, the two differ in terms of performance. For sentence tokenization, NLTK is better than spaCy. spaCy's poor performance in sentence tokenization is due to their difference in approach. For each sentence, spaCy constructs a syntactic tree, while NLTK attempts to split text into sentences. Due to this difference in methods, it also differs in speed. NLTK returns results considerably slower than spaCy. Therefore in terms of word tokenization and POS-tagging, spaCy outperforms NLTK.

The two libraries also differ in how they parse inputs. As NLTK is a string processing library, it takes strings as input and returns strings or lists of strings as the output. In contrast, spaCy uses an object-oriented approach, where every function returns objects as the output.

The two also differ in their approaches, spaCy takes a more granular approach while NLTK takes a more holistic approach. spaCy uses a single stemmer and is used more widely to complete real-world tasks for production usage, while NLTK is used to develop complex NLP functions via different stemming libraries.

Conclusion

In theory, both libraries are able to successfully handle a wide range of NLP tasks. In general, NLTK is the recommended tool for beginners and researchers in an academic environment. spaCy is recommended for developing production usage software and applications, as spaCy provides much better tools for that purpose. Ultimately, choosing a NLP library is completely dependent on the problem you want to solve.

References

https://www.nltk.org/
https://spacy.io/
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.100.1089&rep=rep1&type=pdf
https://devopedia.org/natural-language-toolkit
http://www.tulane.edu/~howard/NLP/nlp.html