Apache OpenNLP and NLTK

shyam3
CS 410: Text Information Systems
October 30, 2020

The purpose of this technology review is to discuss Apache OpenNLP and NLTK and compare the two.

Apache OpenNLP is a toolkit used for natural language processing. It can be used directly through a CLI and is also available as an API in Java. It provides many useful tools such as document categorizing, language detection, and POS tagging which will be discussed in more detail later on.

NLTK is also a toolkit used for natural language processing. It is a library built specifically for Python. A nice thing about it is that it comes with many trained models and corpora. It also has many of the same tools that Apache OpenNLP provides.

Both NLTK and Apache OpenNLP are free and open source, which is great if you want to dive into their implementation code or if you want to get involved in contributing yourself. They also include many more useful tools than those mentioned in this review, so I would recommend looking into their documentation (linked below) for more insight.

For document categorization, OpenNLP encapsulates a maxent model and uses it for classification in it's DoccatModel object. NLTK gives you more flexibility by not encapsulating the underlying model. You can use a decision tree model, a Naive Bayes model, or a few others, including a maxent model, from the nltk.classify library to achieve this. So depending on your level of expertise, you may prefer to use OpenNLP to abstract all of the classification model details away, or you may want to use NLTK to get much more control over it.

Language detection another function that OpenNLP and NLTK both support. OpenNLP has a LanguageDetectorModel object which can be trained and used to get the language of a given piece of text. Similar to document categorization, NLTK can use the nltk.classify library, specifically the textcat module for language detection. They both provide pretty similar functionality in this case.

For POS tagging, OpenNLP has a default tagger that is ready for use without any training required by the user with the POSModel and POSTaggerME objects. It also gives the user the ability to train a POS tagger with their own data if desired. NLTK has its nltk.tag library which provides a few different kinds of POS taggers, such as nltk.tag.crf (CRFSuite), nltk.tag.hunpos (HunPos open-source POS tagger), nltk.tag.senna (Senna), and nltk.tag.tnt (Statistical TnT POS

tagger). NLTK gives you access to some interesting taggers that you may choose to experiment with.

Trying to decide which to use for your project heavily depends on your use case. As mentioned above, they each support a different language so whichever language you prefer should help you decide. Depending on which parts of the toolkits you need to use, and your expertise level with these techniques, your preferences may also change.

References:

OpenNLP documentation: https://opennlp.apache.org/docs/1.9.3/manual/opennlp.html
NLTK documentation: https://www.nltk.org/api/nltk.html