Apache OpenNLP

Introduction

The Apache OpenNLP library is an open-source machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as:

- Language detection
- Tokenization
- Sentence segmentation
- Part-of-speech tagging
- Named entity extraction
- Chunking
- Parsing and coreference resolution

These tasks are usually required to build more advanced text processing services.

Features

Apache OpenNLP is built on a component-based architecture which enables one to build a full NLP Pipeline. These components include sentence detector, tokenizer, name finder, document categorizer, part-of-speech tagger, chunker, parser, coreference resolution. These components contain parts which enable one to execute the respective natural language processing task, to train a model and often also to evaluate a model. Apache OpenNLP offers two modes to access these components – application program interface (API) or the command line interface (CLI) mode.

Apache provides pre-trained models that are language specific and can be downloaded and loaded as part of the NLP Pipeline. These models can be used as it is or new models can be built by providing appropriate training parameters and examples.

Let's explore how to use OpenNLP to perform an NLP Tasks. In general, OpenNLP expects a model and an input to perform the task.

Sentence Detection

The sentence detector can break a paragraph into multiple sentences, it can detect whether a punctuation indicates the end of a sentence or not. Below illustration shows how this can be used in CLI and API mode.

- Command line interface - CLI:

  CLI mode is the quickest way to explore the functionality, it is normally used to perform any predetermined manual activities.

  In the screenshot below, we can see that a text file with paragraph was passed as input to SentenceDetector and the component process it and outputs the sentences into another text file. The contents of the files are shown below.

```
(base) apache-opennlp-2.0.0 $ cat paragraph.txt
This is a statement. This is another statement. You can reach me at abcd.efgh@gmail.com. This is the third statement.
(base) apache-opennlp-2.0.0 $
(base) apache-opennlp-2.0.0 $ bin/opennlp SentenceDetector en-sent.bin < paragraph.txt > sentences.txt
Loading Sentence Detector model ... done (0.051s)


Average: 444.4 sent/s
Total: 4 sent
Runtime: 0.009s
Execution time: 0.082 seconds
(base) apache-opennlp-2.0.0 $
(base) apache-opennlp-2.0.0 $ cat sentences.txt
This is a statement.
This is another statement.
You can reach me at abcd.efgh@gmail.com.
This is the third statement.
```

- Application Program Interface - API:

  Open NLP is Java based, so the dependency needs to be added to the java project so that the module can be imported. This can be done either by adding a dependency in pom.xml/gradle or directly importing the jar into the application code.

  Here is an example of how to use it in Java code. The input paragraph is passed as a string to the SentenceDetectorME API, which returns an array of sentences.

```java
public void sentenceDetect() throws IOException {
    String paragraph = "This is a statement. This is another statement."
              + "Now is an abstract word for time, "
              + "that is always flying. And my email address is google@gmail.com.";

            InputStream is = getClass().getResourceAsStream("/models/en-sent.bin");
            SentenceModel model = new SentenceModel(is);

            SentenceDetectorME sdetector = new SentenceDetectorME(model);

            String sentences[] = sdetector.sentDetect(paragraph);

            for(String sentence : sentences) {
                LOGGER.debug(sentence);
            }

}
```

Here is the output for the above code:

```
This is a statement.
This is another statement.Now is an abstract word for time, that is always flying.
And my email address is google@gmail.com.
```

Comparison with NLTK – Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use 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.

Feature	*OpenNLP*	*NLTK*
Programming Language	Java	Python
Open Source	Yes	Yes
Sentence Detection	Yes	Yes
POS Tagging	Yes	Yes
Chunker	Yes	Yes
Tokenization	Yes	Yes
Document Classification	Yes	Yes
Lemmatization	Yes	Yes
Named Entity Finder	Yes	Yes
Stemming	Yes	Yes
API mode	Yes	Yes
CLI mode	Yes	Yes
Pre-trained Models	Yes	Yes
Support for Entropy	Yes	Yes
Support for Perceptron	Yes	Yes
Concordance	No	Yes