Tech Review - Apache OpenNLP

Introduction

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. Just like other Apache products, Apache OpenNLP is an open source library available for use free of cost, licensed under Apache software license, version 2.0.

Each of the OpenNLP components are available as CLI and Java APIs. The library is used for Named Entity Recognition, Sentence Detection, POS tagging and Tokenization. In this tech review we will analyze the different CLI options of the library.

Just like any other NLP library, OpenNLP also requires a trained model as input to learn the task and perform the identification, detection, recognition, etc based on the training model. Apache OpenNLP uses Maximum Entropy algorithm to maximise the likelihood/correctness of the task. Unlike most other libraries OpenNLP allows to tune the performance of MaxEnt (Maximum Entropy) training time by assigning higher number of training threads.

Apache OpenNLP is highly backward compatible. Models trained with previous version of OpenNLP work smoothly with the new version of library as it uses the same corpora.

Named Entity Recognition with Apache OpenNLP

The main goal of my analysis was to evaluate the ease and performance of OpenNLP tool for Named Entity Recognition, NER. NER is mechanism to identify and find named entities like people, technologies, location, companies, etc named entities from given text. To start my analysis, I performed **sentence detection** using OpenNLP. Sentence detection is quite challenging due to the ambiguous nature of some languages like Chinese or Urdu. Where there is no space between two words and very little concept of sentence segregation. In case of english language as well, a sentence containing end character, period, is reused for other purposes like email address, website address, etc. To get reliable sentence detection we need to train to provide a trained model as input to OpenNLP CLI to get useful detection from the tool. For my analysis I copied the “en-sent.bin” from Apache OpenNLP repo and performed my analysis using command on test.txt:

$ opennlp SentenceDetector en-sent.bin test.txt output.txt

The above command used “en-sent.bin” as trained model, for sentence detection, test.txt is the file containing sentences with confusing input like “You can contact us at sdfds@aserf.cwd.we or use our live chat on our website www.sdfsd.dds. ”. Apache OpenNLP printed one sentence per line in the output.txt.

OpenNLP also provide an evaluator for every different task. To evaluate Sentence detection I downloaded the data similar to training data to evaluate UTF-8 encoded data as:

$ opennlp SentenceDetectorEvaluator -model en-sent.bin -data en-sent.eval -encoding UTF-8

The output of the above command produced Precision, Recall and F1 measure for the evaluated data. The results of precision and recall were above 0.9 which is quite impressive.

Now that I have the corpus being split into sentences, next I wanted to check the tool for Named Entity Recognition (NER). For NER, first requirement is to tokenize the sentences from above operation. OpenNLP has 2 mostly used options for **tokenization** 1/Simple tokenizer, 2/TokenizerME. TokenizerME is a learnable tokenizer. To generate tokens from "output.txt" of previous operation, I used the following command:

$ opennlp TokenizerME en-token.bin output.txt tokenized_output.txt

Just like "en-sent.bin", I downloaded "en-token.bin" training model from Apache OpenNLP repo. After tokenization activity the data available in tokenized_output.txt has whitespace separated tokens. Now for the main motive of the analysis, I started with **named entity recognition** task. Apache OpenNLP offers number of pre-trained models for NER models which are trained on various freely available corpus. They can be downloaded from Apache OpenNLP repo, I downloaded English language NER trained model named "en-ner-person.bin". The command I used for performing NER is:

$ opennlp TokenNameFinder en-ner-person.bin tokenized_output.txt named_entities.txt

The output of the above command resulted in correct identification of the named entities without relying on case sensitivity. Some words which are names and verbs like Concur, name of the company and an action; were not recognized correctly. For improving the reignition correctness, I did not modify the training model. But Apache OpenNLP allows one to train the name finder model by feeding the training data that contains the entities would like to detect to OpenNLP TokenNameFinderTraining API along with the base training model. For instance, if one would want to train English language name finder model with more words, one can use the training CLI as:

$ opennlp TokenNameFinderTrainer -model en-ner-person.bin -lang en -data en-ner-person.train -encoding UTF-8

With this command, "en-ner-person.bin", the base English language NER training model will be trained with data in "en-ner-person.train". Additionally it is possible to specify the number of iterations, the cutoff and to overwrite all types in the training data with a single type.

Conclusion

For the review I explored exciting capabilities of OpenNLP. By having focused analysis for NER, I was able to do justice to NLP features like Sentence Detection, Tokenization, and NER. From the experience of this review, I can conclude it is very easy to start with OpenNLP due to its CLI option. CLI makes the learning curve shallow. Extensive documentation of Apache OpenNLP helps in getting started with OpenNLP in minutes.

OpenNLP makes training the base models per your domain very easy and quick. OpenNLP also allows to tune the performance as per the configurations. Apache OpenNLP Github contributions suggest the library is going to grow exponentially in coming years and become a common NLP tool in industries.

References

Apache OpenNLP developer guide -
https://opennlp.apache.org/docs/1.9.3/manual/opennlp.html#intro.cli