COMP 7780 – Natural Language Processing  
Progress Report: Comprehensive study on OpenNLP  
**Ruchi Ruchi**, **Computer Science**, Spring 2016

**PROJECT GOAL AND SCOPE**

The goal of the project is to study and present a comprehensive NLP software package OpenNLP that implements major steps which includes sentence segmentation, tagging, parsing, semantic processing, etc.

**INTRODUCTION**

OpenNLP is a toolkit based on machine learning which is helpful for processing natural language text. In my project I will be doing the comprehensive study on the Tools and API for Sentence detector, Tokenizer, Name Finder, Part-of-speech Tagger, Chunker and Parser. I will be focusing on understanding of the APIs and implementation of the Training and Evaluation tool for each of the steps of processing natural language processing.

**ABOUT OPENNLP**

**OpenNLP** tool is used for processing natural language text. I have studied the following tasks for my project: Sentence detection, tokenization, parts of speech tagging, chunking and parsing. Implementing each of the steps for Java maven project.

**Sentence Detection**: This tool consider a sentence to be longest white space trimmer character sequence in between two punctuation. It consider the first white space as the starting of the sentence and last white space as the end of the sentence.

Integrated **Sentence Detection** into my project using OpenNLP sentence detection API.

Steps:

1. First step is to load sentence detection model.
2. Instantiated SentenceDetecterME and called sentDetect(model) method.

**Conclusion**:

1. It can detect punctuation character marks at the end of the sentence or not.
2. Sentence detection is done before the text is tokenized that’s the way pre-trained models online are trained.
3. But we can apply sentence detection process for already tokenized text as well.

**Tokenization**: OpenNLP does the segmentation of input character into tokens, they are generally numbers, words and punctuation.

**Tokenizers** can be integrated into application by Tokenizer API.

1. TokenizerModel is loaded first.
2. Instantiate TokenizerME and calling tokenize(model) method.

**Detokenization:** Its opposite of tokenization where the words are organized together to again make sentence. The rules used for detokenizing are: MERGE\_TO\_LEFT: merges token to left side, MERGE\_TO\_RIGHT: merges token to right, RIGHT\_LEFT\_MATCHING: merges the token to the right side on the first occurrence and the left side on the second occurrence.

**Name Finder:** The name finder uses a model. The model is dependent on the language and the entity type it is trained for. The opennlp projects offers a number of pre-trained name finder models which are be trained on various freely available corpora.

The steps of implementing Name finder is first the model should be loaded into memory from disk. We need to instantiate NameFinderME. The input text segmented into documents, sentences and tokens. It tokenizes the word from the text file and then an array of type Spans checks the words present in an array.

**Parts of speech tagging**: Assigning word type to the tokens. The parts of speech tags depending on the token and the context it has been used. OpenNLP tagger predicts the correct parts of speech tag by using a probability model using a tag set.

To embed Pos Tagger, it can be done by using its API. First the pos model is loaded into memory. The PosTaggerME is instantiated. The tag array contains one part-of-speech tag for each token in an input array.

**Chunking**: OpenNLP chuncks the words based on its correlated parts into groups like noun groups, verb groups but does not contains details of the internal structure and the role of those words in main sentence.

After loading the chunker model from the memory. The ChunkerME can be instantiated to perform chunking.

**Parsing**: OpenNLP has parsing API which makes it easy to implement Parsing. The parser model should be loaded first. To instantiate a parser a factory method is used for creating the Parser via the new operator. The factory method will read a type parameter from the model and creates an instance of the corresponding parser implementation.

**WORKING**

I have analyzed OpenNLP by executing and running tool. I worked on Java Maven project and used OpenNLP v1.5 to test sentence detection, tokenization, Name Finding, parts of speech tagging, chunking and parsing for a text file saved in the system. I used the trained model to perform these operations. List of models used:

1. en-sent.bin - Sentence Detection
2. en-token.bin - Tokenization
3. en-ner-person.bin - Name Finding
4. en-pos-maxent.bin - POS tagging
5. en-chunker.bin - Chunking
6. en-parser-chunking.bin – Parsing

**REFERENCES**

https://opennlp.apache.org/