**GATE**

openNLP is used as a plugin in GATE(General Architecture for Text Engineering). And also as per the GATE documentation on [openNLP plugin](https://gate.ac.uk/sale/tao/splitch23.html#x28-58200023.25) we have to give additional memory to GATE in order to use the OpenNLP PRs comfortably. So using GATE for the entire nlp task for our project won’t be efficient. But we can use GATE only for the model training phase. For that we can use [GATE Teamware](https://gate.ac.uk/sale/tao/splitch25.html) which is a collaborative annotation tool where we can annotate the corpus as a Team in real time.

GATE Teamware is open-source software, released under the GNU Aﬀero General Public Licence version 3. Commercial licences are available from the University of Sheﬃeld. The source code is available from the subversion repository at

<https://gate.svn.sourceforge.net/svnroot/gate/teamware/trunk>

**Apache openNLP**

The detailed documentation of the Apache openNLP is available [**here**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html)

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.

These components include: sentence detector, tokenizer, name finder, document categorizer, part-of-speech tagger, chunker, parser, coreference resolution. Each of these facilities is accessible via its application program interface (API). In addition, a command line interface (CLI) is provided for convenience of experiments and training.

OpenNLP components have similar APIs for the above tasks. Normally, to execute a task, we should provide a model and an input.

**Sentence Detector**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.sentdetect**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.sentdetect)

The OpenNLP Sentence Detector can detect that a punctuation character marks the end of a sentence or not. In this sense a sentence is defined as the longest white space trimmed character sequence between two punctuation marks.

The first and last sentence make an exception to this rule. The first non-whitespace character is assumed to be the beginning of a sentence, and the last non whitespace character is assumed to be a sentence end.

It has its sentence detection tool which is only intended for demonstration and testing. And its API can be used with a sentence model and an input.

There is also the training tool and training API available which is used to train the models. And there is an evaluation tool to analyze the results.

**Tokenization**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.tokenizer**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.tokenizer)

The OpenNLP Tokenizers segment an input character sequence into tokens. Tokens are usually words, punctuation, numbers, etc.

· Whitespace Tokenizer - A whitespace tokenizer, non-whitespace sequences are identified as tokens

· Simple Tokenizer - A character class tokenizer, sequences of the same character class are tokens

· Learnable Tokenizer - A maximum entropy tokenizer, detects token boundaries based on probability model

Tokenizer also has its own tool, API, training tool, Training API and evaluator as the Sentence detector. Here the TokenizerModel is used as the model.

**Detokenizing**

It is simple the opposite of tokenization. The following rules can be assigned to a token:

* MERGE\_TO\_LEFT - Merges the token to the left side.
* MERGE\_TO\_RIGHT - Merges the token to the right side.
* RIGHT\_LEFT\_MATCHING - Merges the token to the right side on first occurrence and to the left side on second occurrence.

**Named Entity Recognition**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.namefind**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.namefind)

The Name Finder can detect named entities and numbers in text. To be able to detect entities the Name Finder needs a model. The model is dependent on the language and entity type it was trained for.

The OpenNLP projects offers a number of pre-trained name finder models which are trained on various freely available corpora. To find names in raw text the text must be segmented into tokens and sentences.

We need to use the training with data in the following format

<START:person> Pierre Vinken <END> , 61 years old , will join the board as a nonexecutive director Nov. 29 .

Mr . <START:person> Vinken <END> is chairman of Elsevier N.V. , the Dutch publishing group .

In our case it would be <START: aspect> XXXXX <END>

- **Custom Feature Generation**

This is the process of taking raw, unstructured data and defining features (i.e. variables) for potential use in our statistical analysis. OpenNLP defines a default feature generation which is used when no custom feature generation is specified. This is required to use the NameFinder API in openNLP.

- **Named Entity Annotation Guidelines**

Annotation guidelines define what should be labeled as an entity. To build a private corpus it’s important to know these guidelines and maybe write a custom one. Here is a list of publicly available annotation guidelines:

* [MUC6](http://cs.nyu.edu/cs/faculty/grishman/NEtask20.book_1.html)
* [MUC7](http://acl.ldc.upenn.edu/muc7/ne_task.html)
* [ACE](http://projects.ldc.upenn.edu/ace/docs/English-Entities-Guidelines_v5.6.1.pdf)
* [CONLL 2002](http://www.cnts.ua.ac.be/conll2002/ner/)
* [CONLL 2003](http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt)

**Document Categorizer**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.doccat**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.doccat)

**Part-of-Speech Tagger**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.postagger**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.postagger)

The Part of Speech Tagger marks tokens with their corresponding word type based on the token itself and the context of the token. A token might have multiple pos tags depending on the token and the context. The OpenNLP POS Tagger uses a probability model to predict the correct pos tag out of the tag set.

As the other component the POS tagger too has its POS tagger tool, API which uses POSModel as the model, Training tool, Training API and evaluation tool.

**Chunker**

[**https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.chunker**](https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.chunker)

Text chunking consists of dividing a text in syntactically correlated parts of words, like noun groups, verb groups, but does not specify their internal structure, nor their role in the main sentence.

This also has all the above facilities.

**Parser**

<https://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.parser>