A Report

on

“Apache OpenNLP”

“Text Mining Tool”

Submitted by

Shyamal Khachane	**33227**
Neha Jaju	**33224**
Manvi Pandya	**33235**
Gauri Nandkhedkar	**33239**

Under the guidance of

Mrs D.D.Londhe

Department Of Information Technology

Pune Institute of Computer Technology College of Engineering

Sr. No 27, Pune-Satara Road, Dhankawadi, Pune - 411 043.

2019-2020

OPEN NLP™

INTRODUCTION

Apache OpenNLP is an open-source Java library which is used to process natural language text. One can build an efficient text processing service using this library. OpenNLP provides services such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution, etc. These tasks are usually required to build more advanced text processing services. OpenNLP also included maximum entropy and perceptron-based machine learning.

In 2010, OpenNLP entered the Apache incubation. In 2011, Apache OpenNLP 1.5.2 Incubating was released, and in the same year, it graduated as a top-level Apache project and In 2015, OpenNLP was 1.6.0 released.

Overview:

The Apache OpenNLP library contains several components, enabling one to build a full natural language processing pipeline. These components include: sentence detector, tokenizer, name finder, document categorizer, part-of-speech tagger, chunker, parser, coreference resolution. 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. 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.

Purpose

The purpose of the OpenNLP project will be to create a mature toolkit for the above-mentioned tasks. Another is to provide a large number of pre-built models for a variety of languages, as well as the annotated text resources that those models are derived from.

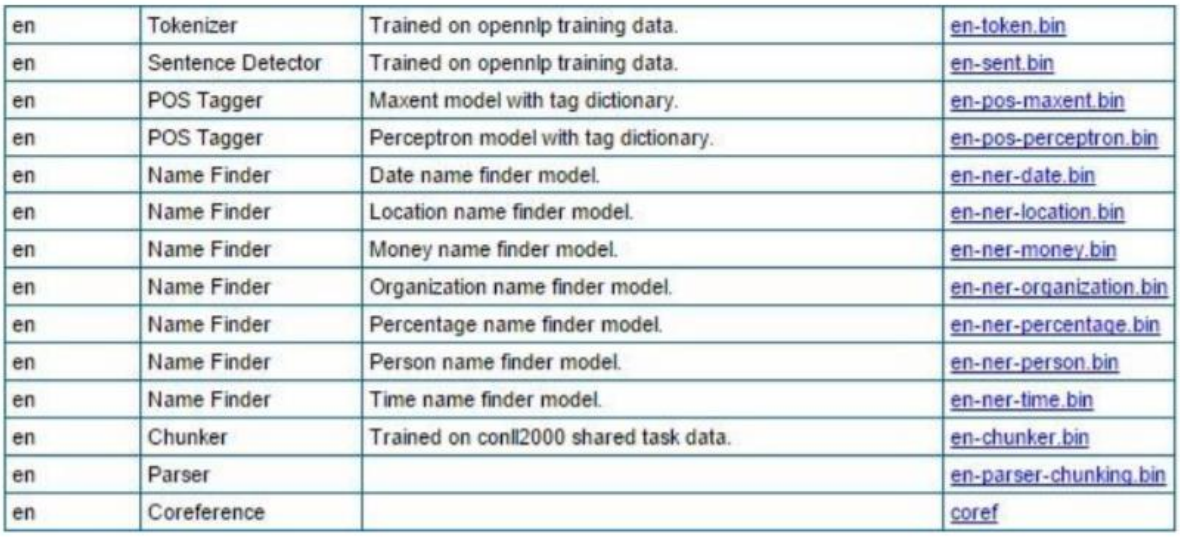
Need:

To perform various NLP tasks, OpenNLP provides a set of predefined models. This set includes models for different languages.

Downloading the models, you can follow the steps given below to download the predefined models provided by OpenNLP.

Step 1: Open the index page of OpenNLP models by clicking the following link: http://opennlp.sourceforge.net/models-1.5/

Step 2: On visiting the given link, you will get to see a list of components of various languages and the links to download them. Here, you can get the list of all the predefined models provided by OpenNLP.



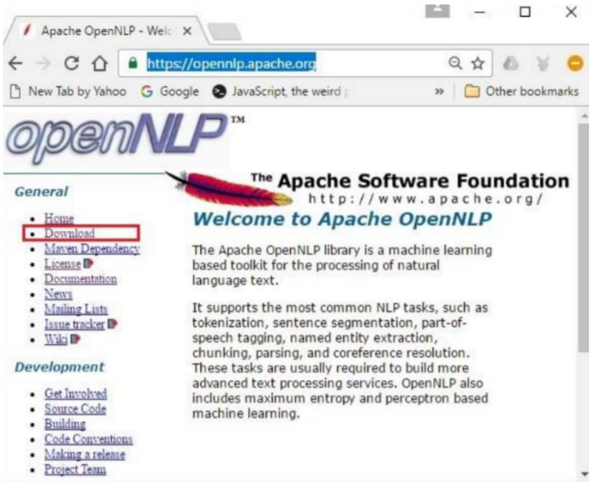
en	Tokenizer	Trained on opennlp training data.	en-token.bin
en	Sentence Detector	Trained on opennlp training data.	en-sent.bin
en	POS Tagger	Maxent model with tag dictionary.	en-pos-maxent.bin
en	POS Tagger	Perceptron model with tag dictionary.	en-pos-perceptron.bin
en	Name Finder	Date name finder model.	en-ner-date.bin
en	Name Finder	Location name finder model.	en-ner-location.bin
en	Name Finder	Money name finder model.	en-ner-money.bin
en	Name Finder	Organization name finder model.	en-ner-organization.bin
en	Name Finder	Percentage name finder model.	en-ner-percentage.bin
en	Name Finder	Person name finder model.	en-ner-person.bin
en	Name Finder	Time name finder model.	en-ner-time.bin
en	Chunker	Trained on conll2000 shared task data.	en-chunker.bin
en	Parser		en-parser-chunking.bin
en	Coreference		coref

Download all these models to the folder C:/OpenNLP_models/, by clicking on their respective links. All these models are language dependent and while using these, you have to make sure that the model language matches with the language of the input text.

Installing OpenNLP

Following are the steps to download Apache OpenNLP library in your system.

Step 1: Open the homepage of Apache OpenNLP by clicking the following link: https://opennlp.apache.org/



General

- Home
- Download
- Maven Dependency
- License
- Documentation
- News
- Mailing Lists
- Issue tracker
- Wiki

Development

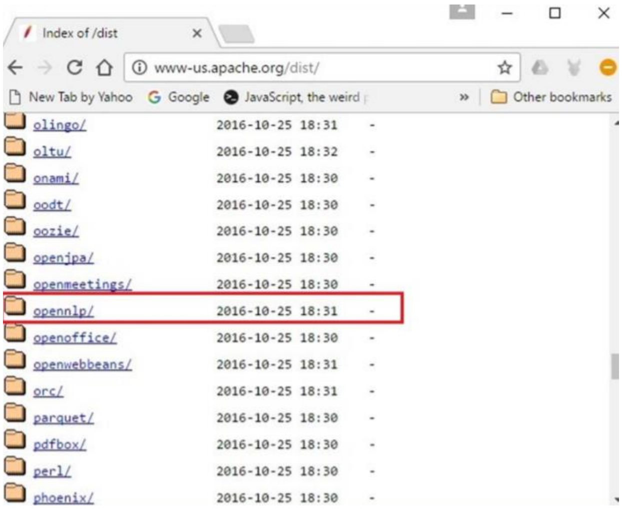
- Get Involved
- Source Code
- Building
- Code Conventions
- Making a release
- Project Team

Welcome to Apache OpenNLP

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text.

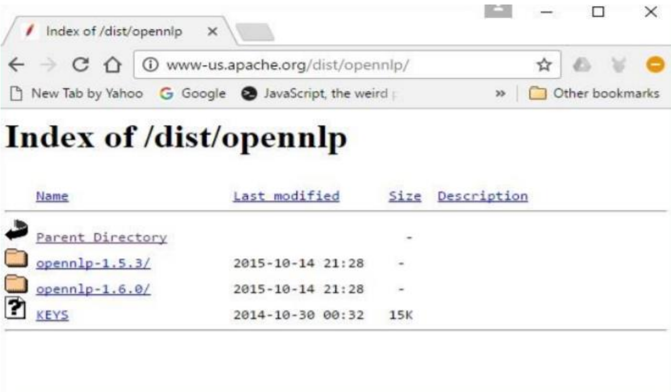
It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. These tasks are usually required to build more advanced text processing services. OpenNLP also includes maximum entropy and perceptron based machine learning.

Step 2: Now, click on the Downloads link. On clicking, you will be directed to a page where you can find various mirrors which will redirect you to the Apache Software Foundation Distribution directory.



```
olingo/          2016-10-25 18:31    -
oltu/            2016-10-25 18:32    -
onami/           2016-10-25 18:30    -
oodt/            2016-10-25 18:30    -
oozie/           2016-10-25 18:30    -
openjpa/         2016-10-25 18:30    -
openmeetings/    2016-10-25 18:30    -
opennlp/         2016-10-25 18:31    -
openoffice/      2016-10-25 18:30    -
openwebbeans/    2016-10-25 18:31    -
orc/             2016-10-25 18:31    -
parquet/         2016-10-25 18:30    -
pdfbox/          2016-10-25 18:30    -
perl/            2016-10-25 18:30    -
phoenix/         2016-10-25 18:30    -
```

Step 3: In this page you can find links to download various Apache distributions. Browse through them and find the OpenNLP distribution and click it.

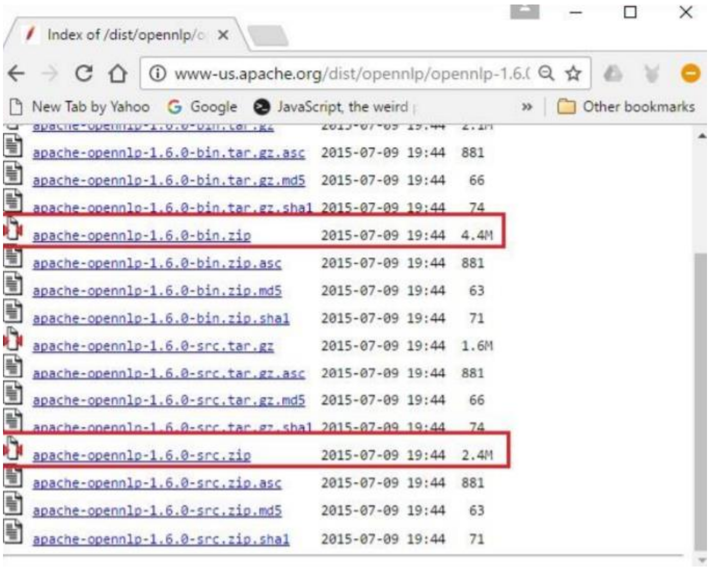


Index of /dist/opennlp

www-us.apache.org/dist/opennlp/

```
Name                       Last modified      Size  Description

Parent Directory                                -
opennlp-1.5.3/             2015-10-14 21:28     -
opennlp-1.6.0/             2015-10-14 21:28     -
KEYS                       2014-10-30 00:32   15K
```

Step 4: On clicking, you will be redirected to the directory where you can see the index of the OpenNLP distribution, as shown below.



Index of /dist/opennlp/o

www-us.apache.org/dist/opennlp/opennlp-1.6.(

```
apache-opennlp-1.6.0-bin.tar.gz        2015-07-09 19:44  2.1M
apache-opennlp-1.6.0-bin.tar.gz.asc    2015-07-09 19:44  881
apache-opennlp-1.6.0-bin.tar.gz.md5    2015-07-09 19:44   66
apache-opennlp-1.6.0-bin.tar.gz.sha1   2015-07-09 19:44   74
apache-opennlp-1.6.0-bin.zip           2015-07-09 19:44  4.4M
apache-opennlp-1.6.0-bin.zip.asc       2015-07-09 19:44  881
apache-opennlp-1.6.0-bin.zip.md5       2015-07-09 19:44   63
apache-opennlp-1.6.0-bin.zip.sha1      2015-07-09 19:44   71
apache-opennlp-1.6.0-src.tar.gz        2015-07-09 19:44  1.6M
apache-opennlp-1.6.0-src.tar.gz.asc    2015-07-09 19:44  881
apache-opennlp-1.6.0-src.tar.gz.md5    2015-07-09 19:44   66
apache-opennlp-1.6.0-src.tar.gz.sha1   2015-07-09 19:44   74
apache-opennlp-1.6.0-src.zip           2015-07-09 19:44  2.4M
apache-opennlp-1.6.0-src.zip.asc       2015-07-09 19:44  881
apache-opennlp-1.6.0-src.zip.md5       2015-07-09 19:44   63
apache-opennlp-1.6.0-src.zip.sha1      2015-07-09 19:44   71
```

Click on the latest version from the available distributions.

Step 5: Each distribution provides Source and Binary files of OpenNLP library in various formats. Download the source and binary files, apache-opennlp-1.6.0-bin.zip and apache-opennlp1.6.0-src.zip (for Windows).

Features:

Following are the notable features of OpenNLP –

- **Named Entity Recognition (NER):** Open NLP supports NER, using which you can extract names of locations, people and things even while processing queries.

- **Summarize:** Using the summarize feature, you can summarize Paragraphs, articles, documents or their collection in NLP.

- **Searching:** In OpenNLP, a given search string or its synonyms can be identified in given text, even though the given word is altered or misspelled.

- **Tagging (POS)**: Tagging in NLP is used to divide the text into various grammatical elements for further analysis.

- **Translation**: In NLP, Translation helps in translating one language into another.

- **Information grouping:** This option in NLP groups the textual information in the content of the document, just like Parts of speech.

- **Natural Language Generation:** It is used for generating information from a database and automating the information reports such as weather analysis or medical reports.

- **Feedback Analysis:** As the name implies, various types of feedbacks from people are collected, regarding the products, by NLP to analyse how well the product is successful in winning their hearts.

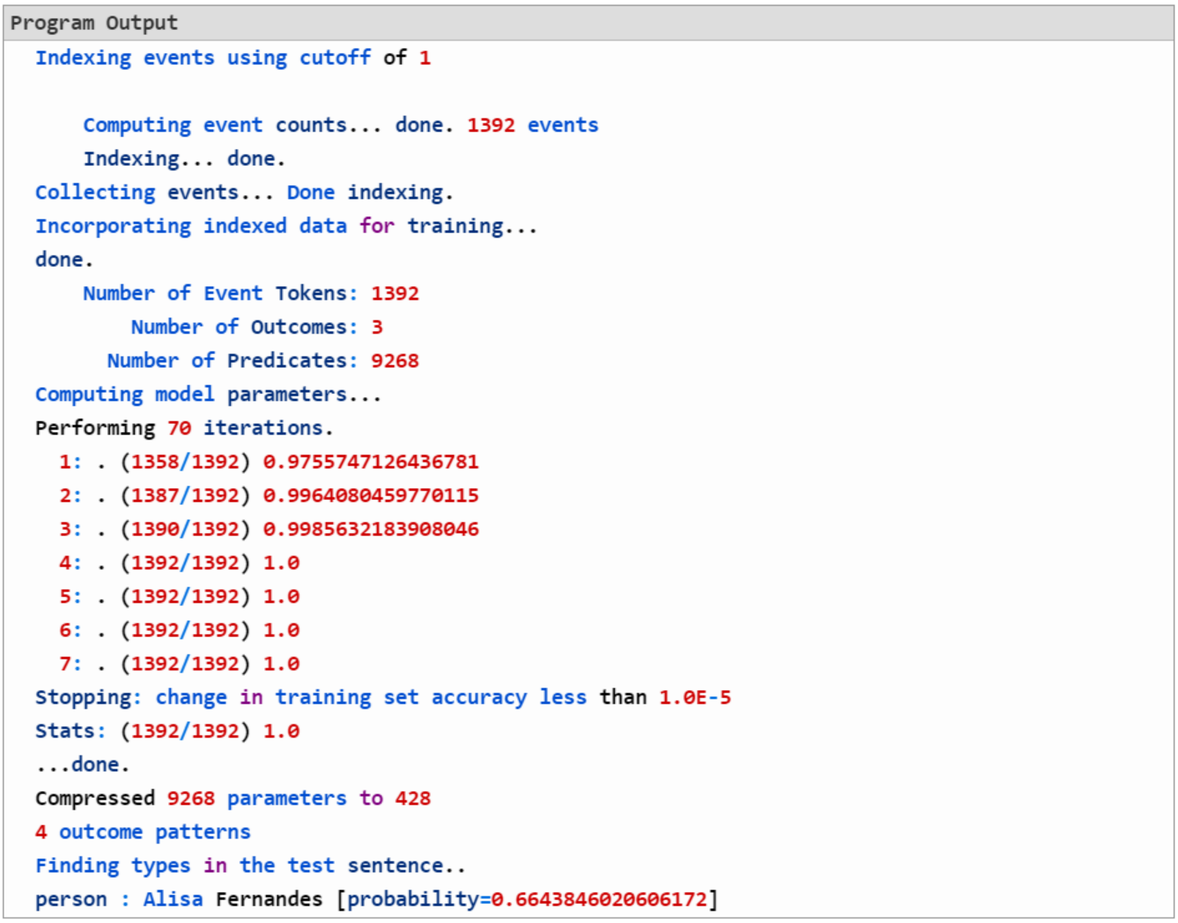
- **Speech recognition:** Though it is difficult to analyse human speech, NLP has some built in features for this requirement.

Services in detail:

1. NER Training in OpenNLP

Following is a step-by-step process in generating a model for custom training data :

- Prepare Training Data
- Read the training data
- Training Parameters.
- Train the model.
- Save the model to a file.
- Test the program.



```
Program Output
 Indexing events using cutoff of 1

     Computing event counts... done. 1392 events
     Indexing... done.
 Collecting events... Done indexing.
 Incorporating indexed data for training...
 done.
     Number of Event Tokens: 1392
         Number of Outcomes: 3
       Number of Predicates: 9268
 Computing model parameters...
 Performing 70 iterations.
   1:  . (1358/1392) 0.9755747126436781
   2:  . (1387/1392) 0.9964080459770115
   3:  . (1390/1392) 0.9985632183908046
   4:  . (1392/1392) 1.0
   5:  . (1392/1392) 1.0
   6:  . (1392/1392) 1.0
   7:  . (1392/1392) 1.0
 Stopping: change in training set accuracy less than 1.0E-5
 Stats: (1392/1392) 1.0
 ...done.
 Compressed 9268 parameters to 428
 4 outcome patterns
 Finding types in the test sentence..
 person : Alisa Fernandes [probability=0.6643846020606172]
```

2. Tokenization

Tokenization is a process of segmenting strings into smaller parts called tokens(say sub-strings). These tokens are usually words, punctuation marks, sequence of digits, and like. An example is shown in the following table :

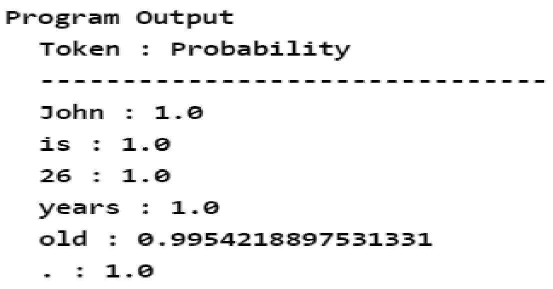
<table>
<tr><td>Input to Tokenizer</td><td>John is 26 years old.</td></tr>
<tr><td>Output of Tokenizer</td><td><table><tr><td>John</td><td>is</td><td>26</td><td>years</td><td>old</td><td>.</td></tr></table></td></tr>
</table>

Tokenizer API in OpenNLP provides following three ways for tokenization

- TokenizerME class loaded with a token model
- WhitespaceTokenizer
- Simple Tokenizer

TokenizerME class loaded with a token model

- Step 1 : Read the pretrained model into a stream.
  InputStream&nbsp;modelIn = new FileInputStream("en-token.bin");
- Step 2 : Read the stream to a Tokenizer model.
  TokenizerModel model = new TokenizerModel(modelIn);
- Step 3 : Initialize the tokenizer with the model.
  TokenizerME tokenizer = new TokenizerME(model);
- Step 4 : Use TokenizerME.tokenize() method to extract the tokens to a String Array.
  String tokens[] = tokenizer.tokenize("John is 26 years old.");
- Step 5 : Use TokenizerME.getTokenProbabilities() to get the probabilities for the segments to be tokens.
  double tokenProbs[] = tokenizer.getTokenProbabilities();
- Step 6 : Finally, print the results.

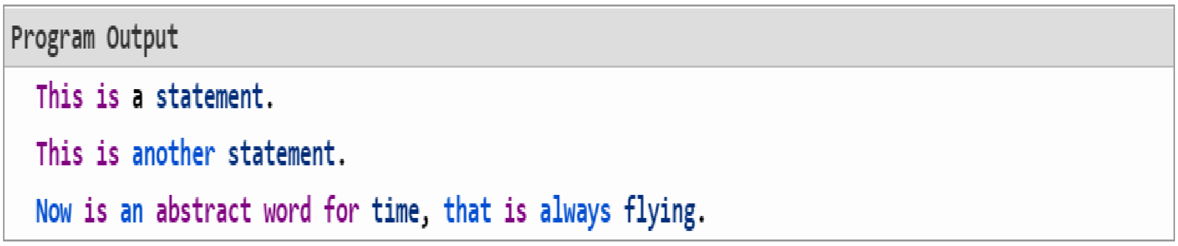


```
Program Output
  Token : Probability
  -------------------------------
  John : 1.0
  is : 1.0
  26 : 1.0
  years : 1.0
  old : 0.9954218897531331
  . : 1.0
```

3. Sentence Detection

Sentence Detection or Sentence Segmentation is a process of finding the start and end of a sentence (in a paragraph). This has to be done often in pre-processing section of most of the use cases, which are trying to be solved using Natural Language Processing techniques.

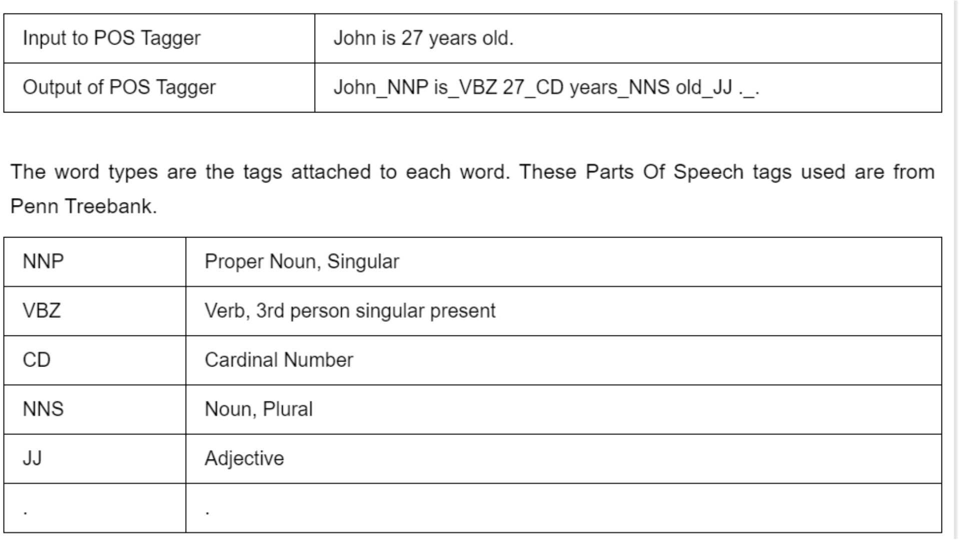
```java
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;

import com.fasterxml.jackson.databind.exc.InvalidFormatException;

import opennlp.tools.sentdetect.SentenceDetectorME;
import opennlp.tools.sentdetect.SentenceModel;

/**
* Sentence Detection Example in openNLP using Java
* @author tutorialkart
*/
public class SentenceDetectExample {

   public static void main(String[] args) {
      try {
         new SentenceDetectExample().sentenceDetect();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   /**
    * This method is used to detect sentences in a paragraph/string
    * @throws InvalidFormatException
    * @throws IOException
    */
   public void sentenceDetect() throws InvalidFormatException, IOException {
      String paragraph = "This is a statement. This is another statement. Now is an abstract word for time, that is always flying.";
      // refer to model file "en-sent,bin", available at link http://opennlp.sourceforge.net/models-1.5/
      InputStream is  =  new  FileInputStream("en-sent.bin");
      SentenceModel model  =  new  SentenceModel(is);

      // feed the model to SentenceDetectorME class
      SentenceDetectorME sdetector  =  new  SentenceDetectorME(model);

      // detect sentences in the paragraph
      String  sentences[]  =  sdetector.sentDetect(paragraph);

      // print the sentences detected, to console
      for(int  i=0;i<sentences.length;i++){
         System.out.println(sentences[i]);
      }
      is.close();
   }
 }
```



Program Output

```
This is a statement.
This is another statement.
Now is an abstract word for time, that is always flying.
```

Parts-of-Speech Tagging



Input to POS Tagger	John is 27 years old.
Output of POS Tagger	John_NNP is_VBZ 27_CD years_NNS old_JJ ._.

The word types are the tags attached to each word. These Parts Of Speech tags used are from Penn Treebank.

NNP	Proper Noun, Singular
VBZ	Verb, 3rd person singular present
CD	Cardinal Number
NNS	Noun, Plural
JJ	Adjective
.	.

Following are the steps to obtain the tags pragmatically in java using apache openNLP

- Step 1 : Tokenize the given input sentence into tokens.

```
String sentence = "John is 27 years old.";
// tokenize the sentence
tokenModelIn = new FileInputStream("en-token.bin");
TokenizerModel tokenModel = new TokenizerModel(tokenModelIn);
Tokenizer tokenizer = new TokenizerME(tokenModel);
String tokens[] = tokenizer.tokenize(sentence);
```

- Step 2 : Read the parts-of-speech maxent model, “en-pos-maxent.bin” into a stream.

```
InputStream posModelIn = new FileInputStream("en-pos-maxent.bin");
```

- Step 3 : Read the stream into parts-of-speech model, POSModel.

```
POSModel posModel = new POSModel(posModelIn);
```

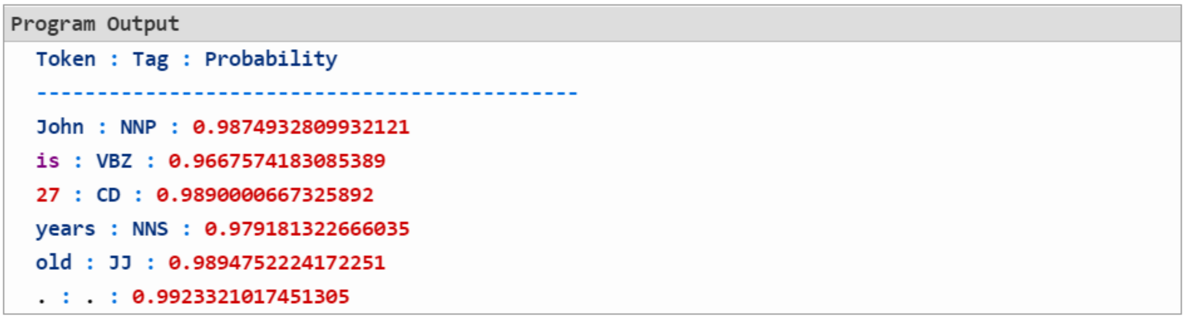
- Step 4 : Load the model into parts-of-speech tagger, POSTaggerME

```
POSTaggerME posTagger = new POSTaggerME(posModel);
```

- Step 5 : Grab the tags using the method POSTaggerME.tag(), and probability for the tag to be given using the method PosTaggerME.probs();

```
String tags[] = posTagger.tag(tokens);
double probs[] = posTagger.probs();
```

- Step 6 : Finally, print what we got, the token, their respective tags and probabilities of the tags.

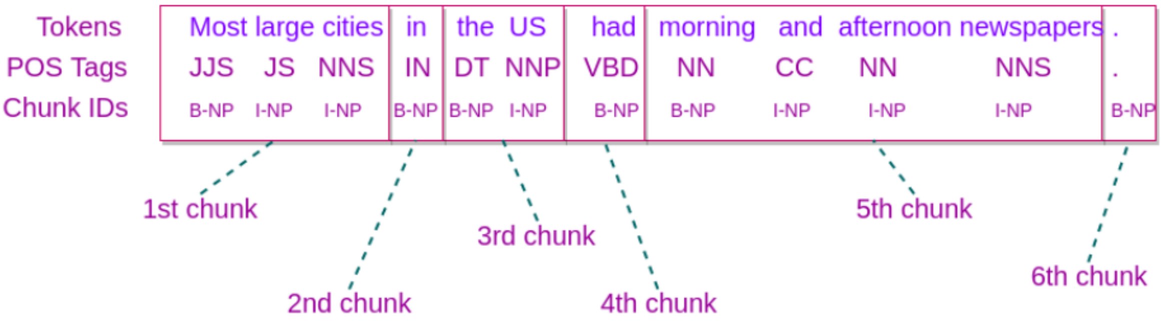


```
Program Output
  Token : Tag : Probability
  ---------------------------------------------
  John : NNP : 0.9874932809932121
  is : VBZ : 0.9667574183085389
  27 : CD : 0.9890000667325892
  years : NNS : 0.979181322666035
  old : JJ : 0.9894752224172251
  . : . : 0.9923321017451305
```

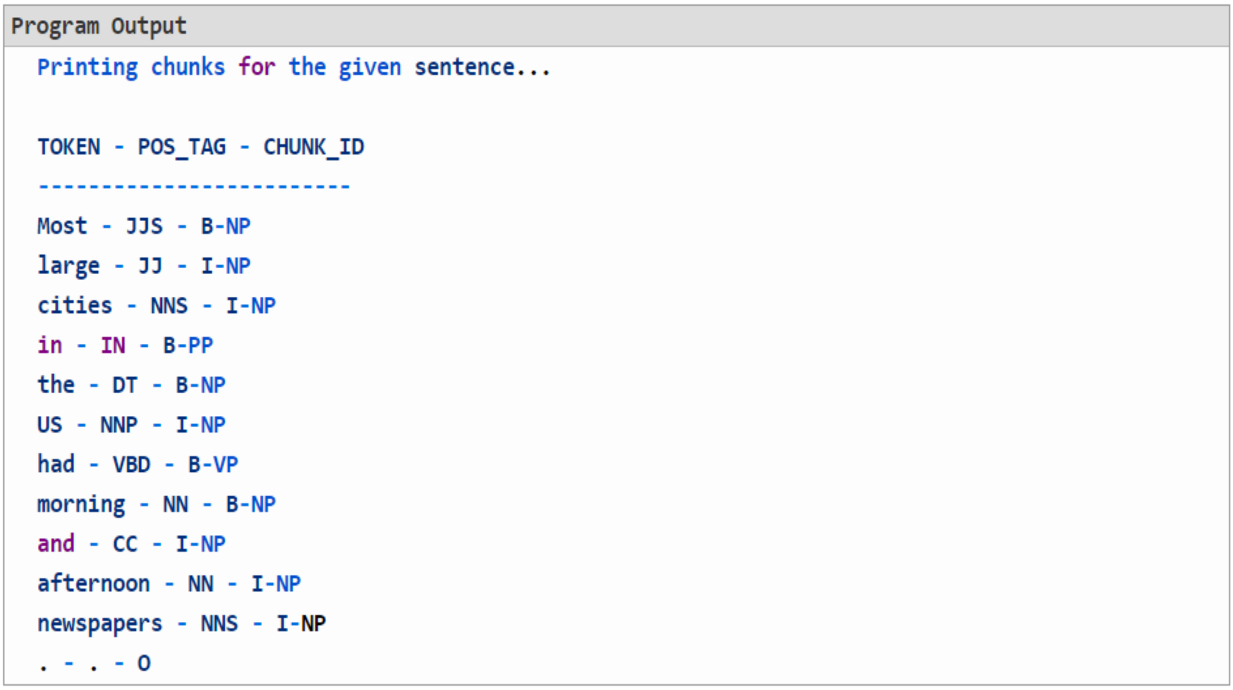
4. Chunker

A chunker breaks the sentence into groups(of words) containing sequential words of sentence, that belong to a noun group, verb group, etc.

Pictorial representation of the test sentence that we are going to divide into chunks is given below :



Chunker Example in Apache OpenNLP



```
Program Output
  Printing chunks for the given sentence...

  TOKEN - POS_TAG - CHUNK_ID
  -------------------------
  Most - JJS - B-NP
  large - JJ - I-NP
  cities - NNS - I-NP
  in - IN - B-PP
  the - DT - B-NP
  US - NNP - I-NP
  had - VBD - B-VP
  morning - NN - B-NP
  and - CC - I-NP
  afternoon - NN - I-NP
  newspapers - NNS - I-NP
  . - . - O
```

- B-   : Represents the start of a chunk
- I-   : Represents the continuation of a chunk

We shall represent the output in a table, and mention the chunks in the last column.

Token	POS Tag	Chunk ID	Chunk
Most	JJS	B-NP	**1st chunk** in the sentence (Noun Phrase)
large	JJ	I-NP	
cities	NNS	I-NP	
in	IN	B-NP	**2nd chunk** in the sentence (Noun Phrase)
the	DT	B-NP	**3rd chunk** in the sentence (Noun Phrase)
US	NNP	I-NP	
had	VBD	B-NP	**4th chunk** in the sentence (Noun Phrase)
morning	NN	B-NP	**5th chunk** in the sentence (Noun Phrase)
and	CC	I-NP	
afternoon	NN	I-NP	
newspapers	NNS	I-NP	
.	.	0	no chunk

Advantages:

- Advanced text-processing services.
- It supports most common NLP tasks.