GROUP C
Assignment no.16

1. Problem Statement: Study and implementation of research paper in Multidisciplinary NLP using open source tool

2. Objective:
1. To apply algorithmic strategies while solving problems
2. To develop time and space efficient algorithms

3. Theory:
Natural Language Processing:
NLP is strongly concerned with the interactions between computer and human languages.
GOAL:
To get computers to perform useful tasks involving human language.

Tasks:
1. Enabling human-machine communication,
2. Improving human-human communication and
3. Useful processing of text or speech

OpenNLP:
The Apache OpenNLP library is a machine learning based toolkit for processing of natural language text. It includes a sentence detector, a tokenizer, a name finder, a parts-of-speech (POS) tagger, a chunker, and a parser. It has very good APIs that can be easily integrated with a Java program. However, the documentation contains unupdated information.

Download Jar Files and Set Up Environment
Before starting the examples, you need to download the jar files required and add to your project build path. The jar files required are loaced at "apache-opennlp-1.5.3-bin.zip", the download page looks like the following:

File	Signatures
apache-opennlp-1.5.3-bin.tar.gz	md5 sha1 asc
apache-opennlp-1.5.3-bin.zip	md5 sha1 asc
apache-opennlp-1.5.3-src.tar.gz	md5 sha1 asc
apache-opennlp-1.5.3-src.zip	md5 sha1 asc

Unzip the .zip file and copy the 4 jar files in the "lib" directory to your project. In addition, you will need to download some model files later based on what you want to do.

1. Sentence Detector
Sentence detector is for detecting sentence boundaries. Given the following paragraph:
Hi. How are you? This is Mike.
sentence detector returns an array of strings. In this case, the array has two elements as below.
Hi. How are you?
This is Mike.

Example Code:
public static void SentenceDetect() **throws** InvalidFormatException,

```
                IOException {
        String paragraph = "Hi. How are you? This is Mike.";

        // always start with a model, a model is learned from training data
        InputStream is = new FileInputStream("en-sent.bin");
        SentenceModel model = new SentenceModel(is);
        SentenceDetectorME sdetector = new SentenceDetectorME(model);

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

        System.out.println(sentences[0]);
        System.out.println(sentences[1]);
        is.close();
}
```

2. Tokenizer

Tokens are usually words which are separated by space, but there are exceptions. For example, "isn't" gets split into "is" and "n't, since it is a a brief format of "is not". Our sentence is separated into the following tokens:

```
Hi
.
How
are
you
?
This
is
Mike
.
```

Example Code:

```
public static void Tokenize() throws InvalidFormatException, IOException {
        InputStream is = new FileInputStream("en-token.bin");

        TokenizerModel model = new TokenizerModel(is);

        Tokenizer tokenizer = new TokenizerME(model);

        String tokens[] = tokenizer.tokenize("Hi. How are you? This is Mike.");

        for (String a : tokens)
                System.out.println(a);

        is.close();
}
```

3. Name Finder

By its name, name finder just finds names in the context. Check out the following example to see what name finder can do. It accepts an array of strings, and find the names inside.

Example Code:

```
public static void findName() throws IOException {
        InputStream is = new FileInputStream("en-ner-person.bin");

        TokenNameFinderModel model = new TokenNameFinderModel(is);
        is.close();

        NameFinderME nameFinder = new NameFinderME(model);

        String []sentence = new String[]{
                    "Mike",
                    "Smith",
                    "is",
                    "a",
                    "good",
                    "person"
                    };

                Span nameSpans[] = nameFinder.find(sentence);

                for(Span s: nameSpans)
                        System.out.println(s.toString());
}
```

4. POS Tagger

Hi._NNP How_WRB are_VBP you?_JJ This_DT is_VBZ Mike._NNP

Example Code:

```
public static void POSTag() throws IOException {
        POSModel model = new POSModelLoader()
                .load(new File("en-pos-maxent.bin"));
        PerformanceMonitor perfMon = new PerformanceMonitor(System.err, "sent");
        POSTaggerME tagger = new POSTaggerME(model);

        String input = "Hi. How are you? This is Mike.";
        ObjectStream<String> lineStream = new PlainTextByLineStream(
                        new StringReader(input));

        perfMon.start();
        String line;
        while ((line = lineStream.read()) != null) {

                String whitespaceTokenizerLine[] = WhitespaceTokenizer.INSTANCE
                                .tokenize(line);
                String[] tags = tagger.tag(whitespaceTokenizerLine);

                POSSample sample = new POSSample(whitespaceTokenizerLine, tags);
                System.out.println(sample.toString());
```

```
            perfMon.incrementCounter();
        }
        perfMon.stopAndPrintFinalResult();
}
```

5. Chunker

Chunker may not be a concern for some users, but it is worth to mention it here. What chunker does is to partition a sentence to a set of chunks by using the tokens generated by tokenizer.
I don't have a good example to show why chunker is very useful, but you can replace the sentence with your own sentences in the following example code, and try to generate something you like.

Example Code:

```
public static void chunk() throws IOException {
        POSModel model = new POSModelLoader()
                        .load(new File("en-pos-maxent.bin"));
        PerformanceMonitor perfMon = new PerformanceMonitor(System.err, "sent");
        POSTaggerME tagger = new POSTaggerME(model);

        String input = "Hi. How are you? This is Mike.";
        ObjectStream<String> lineStream = new PlainTextByLineStream(
                        new StringReader(input));

        perfMon.start();
        String line;
        String whitespaceTokenizerLine[] = null;

        String[] tags = null;
        while ((line = lineStream.read()) != null) {
                whitespaceTokenizerLine = WhitespaceTokenizer.INSTANCE
                                .tokenize(line);
                tags = tagger.tag(whitespaceTokenizerLine);

                POSSample sample = new POSSample(whitespaceTokenizerLine, tags);
                System.out.println(sample.toString());
                        perfMon.incrementCounter();
        }
        perfMon.stopAndPrintFinalResult();

        // chunker
        InputStream is = new FileInputStream("en-chunker.bin");
        ChunkerModel cModel = new ChunkerModel(is);

        ChunkerME chunkerME = new ChunkerME(cModel);
        String result[] = chunkerME.chunk(whitespaceTokenizerLine, tags);

        for (String s : result)
                System.out.println(s);
```

```
        Span[] span = chunkerME.chunkAsSpans(whitespaceTokenizerLine, tags);
        for (Span s : span)
                System.out.println(s.toString());
}
```

6. Parser

Given this sentence: "Programcreek is a very huge and useful website.", parser can return the following:

```
(TOP
        (S
                (NP
                        (NN Programcreek)
                )
                (VP
                        (VBZ is)
                        (NP
                                (DT a)
                                (ADJP
                                        (RB very)
                                        (JJ huge)
                                        (CC and)
                                        (JJ useful)
                                )
                        )
                )
                (. website.)
        )
)
```

Example Code:

```
public static void Parse() throws InvalidFormatException, IOException {

        InputStream is = new FileInputStream("en-parser-chunking.bin");

        ParserModel model = new ParserModel(is);

        Parser parser = ParserFactory.create(model);

        String sentence = "Programcreek is a very huge and useful website.";
        Parse topParses[] = ParserTool.parseLine(sentence, parser, 1);

        for (Parse p : topParses)
                p.show();

        is.close();

        }
```

4. Conclusion:

Hence we have studied and learnt implementation of an Open Source tool for natural language

processing.

5. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	ACHIEVED(√)
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness, embedded security and NLP.	
To solve problems for multicore or distributed, concurrent/Parallel environments	√