COMP 9313 Assignment3 Report

Name: Luyao Zhang
Student ID: z5151973

This assignment is relative to Elasticsearch. The aim is to read some data for database and then process the raw data, after processing send them to Elasticsearch Engine. The implement will show below.

First I create a function to process the file inner Dictionary.

```
def readfilename(dir: File): Iterator[File] = {
    val d = dir.listFiles.filter(_.isDirectory)
    val f = dir.listFiles.filter(_.isFile).toIterator
    f ++ d.toIterator.flatMap(readfilename _)

}
```

Then I read the file in a loop, and create a string to store the final result named send_data.

```
for (filename <- files_name){
        var url_elst = "http://localhost:9200/legal_idx/cases/"
        var send_data= "{~id~:~"
        val textFile = filename.toString()
        //read xml file
        val xml = XML.loadFile(textFile)
        val content_1 = (xml \ "name").map(_.text)
        val content_2 = (xml \ "catchphrases" \ "catchphrase").map(_.text)
        val content_3 = (xml \ "sentences" \ "sentence").map(_.text)
        val content_4 = (xml \ "AustLII").map(_.text)
        val textFilename1 = textFile.replace(path,"").replace("/","")
        val textFilename = textFilename1.replace(".xml","")
```

The I load the xml data and process them to string, Sending them to coreNLP server and get the result back. When the response return to us we can prepare to send them to Elasticsearch server. To read the JSON result I use this method:

```
class CC[T] { def unapply(a:Any):Option[T] = Some(a.asInstanceOf[T]) }
            object M extends CC[Map[String, Any]]
            object L extends CC[List[Any]]
            object D extends CC[List[Any]]
            object Q extends CC[Map[String, Any]]
            object B extends CC[String]
            object C extends CC[String]
            val result =  for {
              Some(M(map)) <- List(JSON.parseFull(resStr))
              L(sentences) = map("sentences")
              M(sentence) <- sentences
              D(tokens) = sentence("tokens")
              Q(toke) <- tokens
              B(word) = toke("word")
              C(ner) = toke("ner")
```

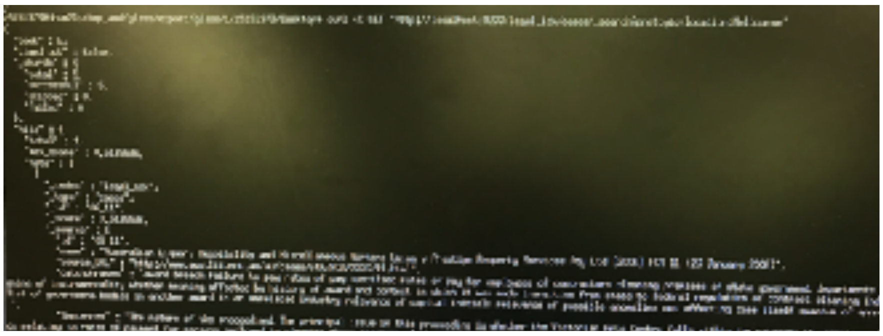
```
        } yield {
            List(word,ner)
        }
```

The final part is send the data to Elasticsearch server. Just use same way as sending data to coreNLP server.

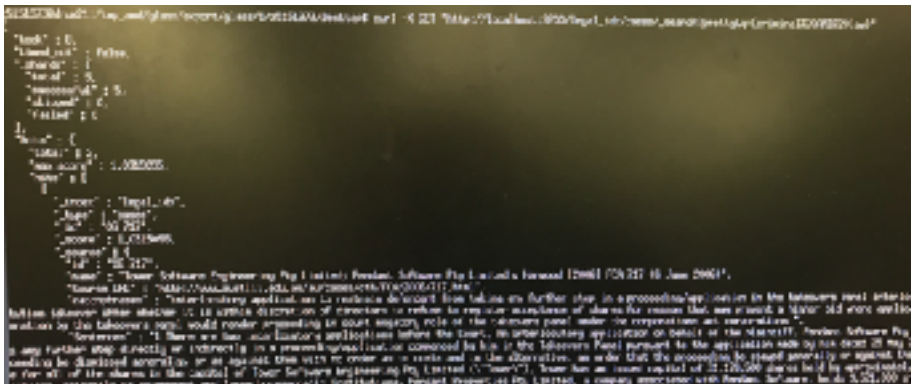
Please use this command to start the program:

spark-submit —packages “org.scalaj:scalaj-http_2.11:2.4.2” —class “CaseIndex” —master local[2] xxxxxxxxx.jar xxxxxxxx

The output when I use curl command in Elasticsearch is:



```
[illegible] curl -X GET "http://localhost:9200/legal_[illegible]/_search?[illegible]"
{
  "took" : [illegible],
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : [illegible],
    "max_score" : [illegible],
    "hits" : [
      {
        "_index" : "legal_[illegible]",
        "_type" : "[illegible]",
        "_id" : "[illegible]",
        "_score" : [illegible],
        "_source" : {
          "id" : "[illegible]",
          "name" : "[illegible]",
          "source_URL" : "http://www.austlii.edu.au/[illegible]",
          "catchphrases" : "[illegible]
[illegible]
          "Sentences" : "[illegible]
[illegible]
```



```
[illegible] curl -X GET "http://localhost:9200/legal_[illegible]/_search?[illegible]"
{
  "took" : [illegible],
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : [illegible],
    "max_score" : [illegible],
    "hits" : [
      {
        "_index" : "legal_[illegible]",
        "_type" : "[illegible]",
        "_id" : "[illegible]",
        "_score" : [illegible],
        "_source" : {
          "id" : "[illegible]",
          "name" : "Tower Software Engineering Pty Limited [illegible]",
          "source_URL" : "http://www.austlii.edu.au/[illegible]",
          "catchphrases" : "[illegible]
[illegible]
          "Sentences" : "[illegible]
[illegible]
```