

1. Generate the output (changes or transformations in the data) manually when the following tasks are applied on the input text. Show your output in details.

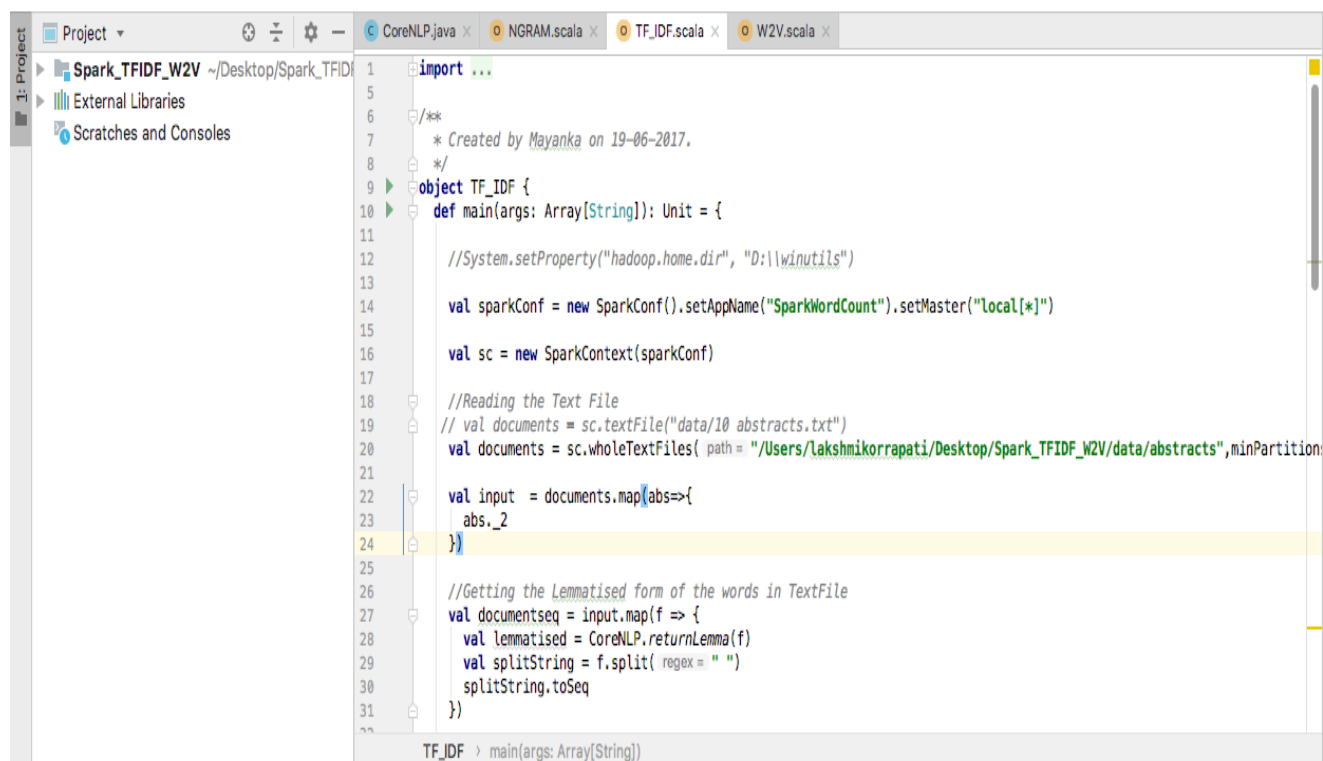
Input: 5 Abstracts saved in separate files

Tasks:

- a. Find out the top TF-IDF words for the above input.
  - b. Find out the top TF-IDF words for the lemmatized input
  - c. Find out the top TF-IDF words for the n-gram based input.
2. Write a simple spark program to read a dataset and find the W2V Synonyms for the Top TF-IDF Words
- a. Try without NLP
  - b. Try with Lemmatization
  - c. Try with NGrams

Compare the results from (a) , (b) and (c)

TF-IDF:



```
1  import ...
2
3  /**
4   * Created by Mayanka on 19-06-2017.
5   */
6
7  object TF_IDF {
8
9    def main(args: Array[String]): Unit = {
10
11      //System.setProperty("hadoop.home.dir", "D:\\winutils")
12
13      val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
14
15      val sc = new SparkContext(sparkConf)
16
17      //Reading the Text File
18      // val documents = sc.textFile("data/10 abstracts.txt")
19      val documents = sc.wholeTextFiles(path = "/Users/lakshmi.korrapati/Desktop/Spark_TFIDF_W2V/data/abstracts", minPartitions = 1)
20
21      val input = documents.map(abs => {
22        abs._2
23      })
24
25      //Getting the Lemmatized form of the words in TextFile
26      val documentSeq = input.map(f => {
27        val lemmatized = CoreNLP.returnLemma(f)
28        val splitString = f.split(regex = " ")
29        splitString.toSeq
30      })
31    }
32  }
```

TF\_IDF > main(args: Array[String])

## Output:

```
Run: TF_IDF x
1 breast cancer , a increase in elastosis be correlate with severity of the disease and age of the patient . elastin-derived peptide -lrp- edp -rrb- be a hallmark of
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
1 adenoid cystic carcinoma of the breast : experience at a tertiary care centre of northern India . Bhutani n1 , Kajal p2 , Singla s3 . author information abstract
```

## W2V for TF-IDF words:

```
Project ▾
  ▸ Spark_TFIDF_W2V ~/Desktop/Spark_TFIDF
  ▸ External Libraries
  ▸ Scratches and Consoles

CoreNLP.java x NGRAM.scala x TF_IDF.scala x W2V.scala x

1 import ...
2
3
4
5
6 /**
7  * Created by Mayanka on 19-06-2017.
8  */
9 object W2V {
10 def main(args: Array[String]): Unit = {
11
12   // System.setProperty("hadoop.home.dir", "D:\\winutils")
13
14   val sparkConf = new SparkConf().setAppName("SparkWordCount").setMaster("local[*]")
15   .set("spark.driver.memory", "6g").set("spark.executor.memory", "6g")
16
17   val sc = new SparkContext(sparkConf)
18
19   val input = sc.textFile( path = "data/sample").map(line => line.split( regex = " ").toSeq)
20
21   val modelFolder = new File( pathname = "myModelPath")
22
23   if (modelFolder.exists()) {
24     val sameModel = Word2VecModel.load(sc, path = "myModelPath")
25     val synonyms = sameModel.findSynonyms( word = "zero", num = 40)
26
27     for ((synonym, cosineSimilarity) <- synonyms) {
28       println(s"$synonym $cosineSimilarity")
29     }
30   }
31   else {
32     val word2vec = new Word2Vec().setMaster("local[*]").setHadoopHome("D:\\winutils").setSparkHome("D:\\spark").setSparkConf(sparkConf).setNumPartitions(1000)
33   }
34 }
35
36 W2V > main(args: Array[String])
```

## Output:

Run: W2V x

▶

⬆

■

⬇

⏸

⏮

📷

🔍

🗑

📄

📌

nine 0.9945034980773926

class 0.9939331412315369

six 0.9934068322181702

eight 0.9930217266082764

late 0.9892733097076416

four 0.989136278629303

international 0.9888012409210205

modern 0.9885989427566528

european 0.9872747659683228

two 0.9852036833763123

votes 0.9833505749702454

civil 0.9820547103081836

theory 0.9796767234802246

front 0.9795065522193909

🖥️ sht shell

📄 Terminal

📄 SBT Console

🔧 Build

🌐 Java Enterprise

▶ 4: Run

⚙ 6: TODO

🌱 Event Log