CS 6350

ASSIGNMENT \_\_\_3\_\_\_\_\_\_\_\_\_

Names of students in your group:

Qi Gao (qxg150130)

Number of free late days used: \_\_\_\_0\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

Q1:

Reference:

<https://stackoverflow.com/questions/41918826/how-to-order-my-tuple-of-spark-results-descending-order-using-value>

<https://stackoverflow.com/questions/44718878/how-to-extract-a-part-of-string-in-rdd>

<https://blog.csdn.net/u013007900/article/details/79307948>

<https://blog.csdn.net/u013063153/article/details/53381873>

Q2:

Reference:

<http://text2vec.org/similarity.html#cosine_similarity_with_tf-idf>

<https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/>

Q3:

Code:

1. **import** java.io.FileWriter
3. **import** org.apache.spark.{SparkConf, SparkContext}
5. **import** scala.collection.mutable
6. **import** org.apache.spark.mllib.clustering.LDA
7. **import** org.apache.spark.mllib.linalg.{Vector, Vectors}
8. **import** org.apache.spark.rdd.RDD
9. **import** org.apache.spark.ml.feature.{RegexTokenizer, StopWordsRemover}
11. object TopicModeling {
12. def main(args: Array[String]): Unit = {
14. //Create a SparkContext to initialize Spark
15. val conf = **new** SparkConf()
16. conf.setMaster("local")
17. conf.setAppName("TopicModeling")
18. val sc = **new** SparkContext(conf)
19. val book: RDD[String] = sc.textFile(args(0))
21. val stopWords = StopWordsRemover.loadDefaultStopWords("english").toSet
22. val tokenized: RDD[Seq[String]] =
23. book.map(\_.toLowerCase.split("\\s")).map(\_.filter(\_.length > 3).filter(token => !stopWords.contains(token)).filter(\_.forall(java.lang.Character.isLetter)))


27. val num\_counts: Array[(String, Long)] =
28. tokenized.flatMap(\_.map(\_ -> 1L)).reduceByKey(\_ + \_).collect().sortBy(-\_.\_2)
30. val stopWords\_count = 20
31. val vocabs: Array[String] =
32. num\_counts.takeRight(num\_counts.size - stopWords\_count).map(\_.\_1)
33. val vocab: Map[String, Int] = vocabs.zipWithIndex.toMap
35. val docs: RDD[(Long, Vector)] =
36. tokenized.zipWithIndex.map { **case** (tokens, id) =>
37. val counts = **new** mutable.HashMap[Int, Double]()
38. tokens.foreach { term =>
39. **if** (vocab.contains(term)) {
40. val idx = vocab(term)
41. counts(idx) = counts.getOrElse(idx, 0.0) + 1.0
42. }
43. }
44. (id, Vectors.sparse(vocab.size, counts.toSeq))
45. }

48. val num\_topics = 5
49. val lda = **new** LDA().setK(num\_topics).setMaxIterations(10)
51. val ldaModel = lda.run(docs)
53. var str=""
54. val topicIndices = ldaModel.describeTopics(maxTermsPerTopic = 10)
55. topicIndices.foreach { **case** (terms, termWeights) =>
56. str=str+"TOPIC is below:\n"
57. terms.zip(termWeights).foreach { **case** (term, weight) =>
58. str=str+s"${vocabs(term.toInt)}\t$weight\n"
59. }
60. str=str+"\n"
61. }
62. sc.parallelize(Seq(str)).saveAsTextFile(args(1))
63. sc.stop()
64. }
66. }
67. /\*
68. val fw = new FileWriter("test.txt", true) ;
69. fw.write("This line appended to file!") ;
70. fw.close()
71. \*/

Reference:

<https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/>