Big Data Programming Extra Credit 2

Shayideep Sangam

Code:

```
/*
 * this has the code for the lecture, and also the code for homework 10
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.StringTokenizer;

import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function;
import org.apache.spark.broadcast.Broadcast;
import org.apache.spark.ml.classification.NaiveBayes;
import org.apache.spark.ml.classification.NaiveBayesModel;
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator;
import org.apache.spark.ml.feature.CountVectorizer;
import org.apache.spark.ml.feature.CountVectorizerModel;
import org.apache.spark.ml.feature.StopWordsRemover;
import org.apache.spark.ml.feature.Tokenizer;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.RowFactory;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.types.DataTypes;
import org.apache.spark.sql.types.StructField;
import org.apache.spark.sql.types.StructType;

import scala.Tuple2;

public class SparkNaiveBayes {

	private static final String LABEL_SEPARATOR = "|";
	private static final String TRAINING_URI =
"C:/Users/dell/Downloads/NB_training_doc*.txt";
	private static final String CATEGORIES =
"C:/Users/dell/Downloads/NB_categories.txt";
	private static final String TESTING_URI =
"C:/Users/dell/Downloads/NB_test_doc*.txt";

	public static void main(String[] args) throws IOException {

		// initializing spark
		SparkSession spark =
SparkSession.builder().config("spark.master","local[*]").getOrCreate();
		JavaSparkContext sc = new JavaSparkContext(spark.sparkContext());
```

```
            sc.setLogLevel("WARN");

            // read the categories file that maps text categories to numerical ones
            HashMap<String, Integer> categories = getCategoryMap(CATEGORIES);
            final Broadcast<HashMap<String, Integer>> allCategories =
sc.broadcast(categories);

            // read the training documents
            JavaPairRDD<String,String> documents = sc.wholeTextFiles(TRAINING_URI);
            //
System.out.println(documents.take((int)documents.count()).toString());
            JavaPairRDD<String,String> documents1 = sc.wholeTextFiles(TESTING_URI);
            // each training document starts with the label
            // get the label, and change it to an integer
            JavaPairRDD<String, Tuple2<Integer,String>> trainingDocs =
documents.mapValues( new Function<String,Tuple2<Integer,String>>() {
                  public Tuple2<Integer,String> call(String line) throws Exception
{
                        if ( line == null || line.length() == 0 ) return null;
                        if ( line.indexOf(LABEL_SEPARATOR) < 0 )  return null;
                        String label = line.substring(0,
line.indexOf(LABEL_SEPARATOR));
                        if ( allCategories.getValue().containsKey(label) == false
) {
                              // missing label
                              return null;
                        }
                        String content =
line.substring(line.indexOf(LABEL_SEPARATOR)+1);
                        return new
Tuple2(allCategories.getValue().get(label),content);
                  }
            });

      System.out.println(trainingDocs.take((int)trainingDocs.count()).toString());
            JavaPairRDD<String, Tuple2<Integer,String>> testDocs =
documents1.mapValues( new Function<String,Tuple2<Integer,String>>() {
                  public Tuple2<Integer,String> call(String line) throws Exception
{
                        return new Tuple2(0,line);
                  }
            });
            // create a dataframe for training documents
            StructType docSchema = new StructType(
                  new StructField[] {
                        DataTypes.createStructField("label",
DataTypes.IntegerType, false),
                        DataTypes.createStructField("text", DataTypes.StringType,
false)
                  }
            );
            Dataset<Row> trainingSet = spark.createDataFrame(
                  trainingDocs.map( new Function<Tuple2<String,
Tuple2<Integer,String>>, Row> () {
                        @Override
```

```
                        public Row call(Tuple2<String, Tuple2<Integer,String>>
record) {
                              return RowFactory.create(record._2()._1(),
record._2()._2());
                        }
                  } ), docSchema);
            // trainingSet.show(false);
            Dataset<Row> testSet = spark.createDataFrame(
                        testDocs.map( new Function<Tuple2<String,
Tuple2<Integer,String>>, Row> () {
                              @Override
                              public Row call(Tuple2<String,
Tuple2<Integer,String>> record) {
                                    return RowFactory.create(record._2()._1(),
record._2()._2());
                              }
                        } ), docSchema);
            // tokenize the training set
            Tokenizer tokenizer = new
Tokenizer().setInputCol("text").setOutputCol("words");
            Dataset<Row> trainingSetTokenized = tokenizer.transform(trainingSet);
            // trainingSetTokenized.show(false);
            Dataset<Row> testSetTokenized = tokenizer.transform(testSet);
            // remove stopwords etc, can use Stanford NLP library if needed
            StopWordsRemover remover = new
StopWordsRemover().setInputCol("words").setOutputCol("filtered");
            Dataset<Row> trainingSetStopWordsRemoved =
remover.transform(trainingSetTokenized);
            //trainingSetStopWordsRemoved.show(false);
            Dataset<Row> testSetStopWordsRemoved =
remover.transform(testSetTokenized);
            // fit a CountVectorizerModel from the corpus
            CountVectorizer vectorizer = new
CountVectorizer().setInputCol("filtered").setOutputCol("features");
            CountVectorizerModel cvm = vectorizer.fit(trainingSetStopWordsRemoved);
            CountVectorizerModel cvr = vectorizer.fit(testSetStopWordsRemoved);
            System.out.println("vocab size = " + cvm.vocabulary().length);
            for (int i = 0; i < cvm.vocabulary().length; i ++ ) {
                  System.out.print(cvm.vocabulary()[i] + "(" + i + ") ");
            }
            System.out.println();
            Dataset<Row> featurizedTrainingSet =
cvm.transform(trainingSetStopWordsRemoved);
            System.out.println("===> final featured training set");
            featurizedTrainingSet.show(true);
            Dataset<Row> featurizedTestSet = cvm.transform(testSetStopWordsRemoved);
            System.out.println("===> final featured testing set");
            featurizedTestSet.show(true);
            /* create LabelledPoint
            JavaRDD<LabeledPoint> labelledJavaRDD =
                  featurizedTrainingSet.select("label",
"features").toJavaRDD().map(new Function<Row, LabeledPoint>() {
                        @Override
                        public LabeledPoint call(Row row) throws Exception {
```

```
                                LabeledPoint labeledPoint = new LabeledPoint(new
Double(row.get(0).toString()),
                                                        (Vector)row.get(1));
                            return labeledPoint;
                        }
                    }
            );

      System.out.println(labelledJavaRDD.take((int)labelledJavaRDD.count()).toString
());
            */

            // create naive bayes model and train it
            NaiveBayes nb = new NaiveBayes();
            NaiveBayesModel model = nb.fit(featurizedTrainingSet.select("label",
"features"));
            // NaiveBayesModel model =
nb.train(featurizedTrainingSet.select("label", "features"));

            // study the model
            System.out.println("model.getFeaturesCol() = " +
model.getFeaturesCol());
            System.out.println("model.getLabelCol() = " + model.getLabelCol());
            System.out.println("model.getModelType() = " + model.getModelType());
            System.out.println("model.getPredictionCol() = " +
model.getPredictionCol());
            System.out.println("model.getProbabilityCol() = " +
model.getProbabilityCol());
            System.out.println("model.getRawPredictionCol() = " +
model.getRawPredictionCol());
            System.out.println("model.numFeatures() = " + model.numFeatures());

            // extra credit homework goes here
            // end of extra credit homework
            Dataset<Row> predictions = model.transform(featurizedTestSet);
            predictions.show();

            MulticlassClassificationEvaluator evaluator = new
MulticlassClassificationEvaluator()
                            .setPredictionCol("prediction")
                            .setMetricName("accuracy");
                           double accuracy = evaluator.evaluate(predictions);
                           System.out.println("Test set accuracy = " + accuracy);
            allCategories.unpersist();
            allCategories.destroy();
            sc.close();
      }

      private static HashMap getCategoryMap(String filePath) {

            HashMap<String, Integer> categories = new HashMap<String,Integer>();
            BufferedReader br = null;

            try {
                  br = new BufferedReader(new FileReader(CATEGORIES));
```

```
                String line = br.readLine();

                while (line != null) {

                        StringTokenizer st = new StringTokenizer(line);
                        String categoryText = st.nextToken();
                        Integer categoryIndex = new Integer(st.nextToken());
                        categories.put(categoryText, categoryIndex);

                        line = br.readLine();

                }

        } catch(Exception e) { // handle it the way you want
                System.out.println(e.getMessage());
        } finally {
                if ( br != null ) {
                        try {
                                br.close();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
        }

        return categories;
    }

}
```

OUTPUT:

```
<terminated> SparkNaiveBayes [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (Nov 25, 2018, 12:21:58 AM)
+-----+--------------------+--------------------+--------------------+--------------------+
|label|                text|               words|            filtered|            features|
+-----+--------------------+--------------------+--------------------+--------------------+
|    0|Chinese Beijing C...|[chinese, beijing...|[chinese, beijing...| (6,[0,2],[2.0,1.0])|
|    0|Chinese Chinese S...|[chinese, chinese...|[chinese, chinese...| (6,[0,4],[2.0,1.0])|
|    0|       Chinese Macao|    [chinese, macao]|    [chinese, macao]| (6,[0,5],[1.0,1.0])|
|    1| Tokyo Japan Chinese|[tokyo, japan, ch...|[tokyo, japan, ch...|(6,[0,1,3],[1.0,1...|
+-----+--------------------+--------------------+--------------------+--------------------+

===> final featured testing set
+-----+--------------------+--------------------+--------------------+--------------------+
|label|                text|               words|            filtered|            features|
+-----+--------------------+--------------------+--------------------+--------------------+
|    0|Chinese Chinese C...|[chinese, chinese...|[chinese, chinese...|(6,[0,1,3],[3.0,1...|
|    0|Chinese Chinese C...|[chinese, chinese...|[chinese, chinese...|       (6,[0],[3.0])|
|    0|Tokyo Tokyo Tokyo...|[tokyo, tokyo, to...|[tokyo, tokyo, to...| (6,[2,3],[1.0,3.0])|
+-----+--------------------+--------------------+--------------------+--------------------+

18/11/25 00:22:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
18/11/25 00:22:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
model.getFeaturesCol() = features
model.getLabelCol() = label
model.getModelType() = multinomial
model.getPredictionCol() = prediction
model.getProbabilityCol() = probability
model.getRawPredictionCol() = rawPrediction
model.numFeatures() = 6
+-----+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+
|label|                text|               words|            filtered|            features|       rawPrediction|         probability|prediction|
+-----+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+
|    0|Chinese Chinese C...|[chinese, chinese...|[chinese, chinese...|(6,[0,1,3],[3.0,1...|[-8.2254733485002...|[0.59713120479585...|       0.0|
|    0|Chinese Chinese C...|[chinese, chinese...|[chinese, chinese...|       (6,[0],[3.0])|[-2.9473586892697...|[0.93483733080028...|       0.0|
|    0|Tokyo Tokyo Tokyo...|[tokyo, tokyo, to...|[tokyo, tokyo, to...| (6,[2,3],[1.0,3.0])|[-10.268547246009...|[0.07867566822155...|       1.0|
+-----+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+

Test set accuracy = 0.6666666666666666
```