|  |
| --- |
|  |
|  |  |
|  | package cat.mhyark.uni.tfg;  import java.util.\*; |
|  | import java.util.regex.Pattern; |
|  |  |
|  | import org.apache.kafka.clients.consumer.ConsumerRecord; |
|  | import org.apache.kafka.common.serialization.StringDeserializer; |
|  | import org.apache.spark.api.java.JavaRDD; |
|  | import org.apache.spark.api.java.function.ForeachFunction; |
|  | import org.apache.spark.sql.\*; |
|  |  |
|  | import org.apache.spark.SparkConf; |
|  | import org.apache.spark.api.java.StorageLevels; |
|  | import org.apache.spark.streaming.Durations; |
|  | import org.apache.spark.streaming.api.java.JavaInputDStream; |
|  | import org.apache.spark.streaming.api.java.JavaReceiverInputDStream; |
|  | import org.apache.spark.streaming.api.java.JavaStreamingContext; |
|  | import org.apache.spark.streaming.kafka010.ConsumerStrategies; |
|  | import org.apache.spark.streaming.kafka010.KafkaUtils; |
|  | import org.apache.spark.streaming.kafka010.LocationStrategies; |
|  |  |
|  | public final class TraceReceiver { |
|  | private static final Pattern COMA = Pattern.compile(","); |
|  |  |
|  | private static ArrayList<Trace> traces\_array = new ArrayList<Trace>(); |
|  |  |
|  | private static SQLContext sqc; |
|  | private static JavaStreamingContext ssc; |
|  |  |
|  | private static TreeClassifier tc; |
|  |  |
|  | private static DBManager dbm; |
|  |  |
|  | private static long timeA; |
|  | private static long timeB; |
|  | private static long timeC; |
|  | private static long timeD; |
|  |  |
|  | public static void main(String[] args) throws Exception { |
|  |  |
|  |  |
|  | if (args.length < 1) { |
|  | System.err.println("Usage: appname <brokerlist:XXXX>"); |
|  | System.exit(1); |
|  | }   |  | | --- | | System.out.println(args[0]); | |  |  | |  | // Create the context with a 5 second batch size | |  | SparkConf sparkConf = new SparkConf().setAppName("FlowInspector"); | |  | ssc = new JavaStreamingContext(sparkConf, Durations.seconds(5)); | |  |  | |  | //SQLContext sqc = new SQLContext(SparkSession.builder().getOrCreate()); | |  | sqc = new SQLContext(SparkSession.builder().getOrCreate()); | |  |  | |  | tc = new TreeClassifier(sqc); | |  |  | |  | dbm = new DBManager(); | |
|  |  |
|  |  |
|  |
|  |
|  |
|  |
|  |  |
|  |  |
|  | System.out.println(args[0]); |
|  |  |
|  | // Create the context with a 5 second batch size |
|  | SparkConf sparkConf = new SparkConf().setAppName("FlowInspector"); |
|  | ssc = new JavaStreamingContext(sparkConf, Durations.seconds(5)); |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  | //JavaReceiverInputDStream<String> lines = ssc.socketTextStream( |
|  | // "ec2-35-176-49-16.eu-west-2.compute.amazonaws.com", 31337, StorageLevels.MEMORY\_AND\_DISK\_SER); // SOCKETS |
|  |  |
|  |  |
|  | // KAFKA!!!! |
|  | Map<String, Object> kafkaParams = new HashMap<>(); |
|  | kafkaParams.put("bootstrap.servers", args[0]);//"10.10.1.2:9092"); |
|  | kafkaParams.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); |
|  | kafkaParams.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); |
|  | kafkaParams.put("group.id", "use\_a\_separate\_group\_id\_for\_each\_stream"); |
|  | kafkaParams.put("auto.offset.reset", "latest"); |
|  | kafkaParams.put("enable.auto.commit", false); |
|  |  |
|  | Collection<String> topics = Arrays.asList("flowinspector"); |
|  |  |
|  | final JavaInputDStream<ConsumerRecord<String, String>> lines = |
|  | KafkaUtils.createDirectStream( |
|  | ssc, |
|  | LocationStrategies.PreferConsistent(), |
|  | ConsumerStrategies.<String, String>Subscribe(topics, kafkaParams) |
|  | ); |
|  |  |
|  |  |
|  | lines.foreachRDD((JavaRDD<ConsumerRecord<String, String>> rdd) -> { |
|  | //lines.foreachRDD((JavaRDD<String> rdd) -> { // SOCKETS |
|  |  |
|  | if (!rdd.isEmpty()) { |
|  |  |
|  | clearArray(); |
|  |  |
|  | Trace t = new Trace(); |
|  | //rdd.foreach((String s) -> { // SOCKETS |
|  | rdd.foreach((ConsumerRecord<String, String> cr) -> { |
|  | String s = cr.value(); |
|  | System.out.println("\n\nFOREACH-STRING COMMING!!!\n\n"); |
|  |  |
|  | List<String> a = Arrays.asList(COMA.split(s)); |
|  |
|  |
|  |
|  |
|  | //Trace t = new Trace();  t.setNo(Integer.parseInt(a.get(0))); |
|  |  |
|  | t.setTime(Integer.parseDouble(a.get(1))); |
|  | t.setSource(Integer.parseString(a.get(2))); |
|  | t.setDestination(Integer.parseString(a.get(3))); |
|  | t.setProtocol(Integer.parseString(a.get(4))); |
|  | t.setLength(Integer.parseInt(a.get(5))); |
|  | t.setInfo(Double.parseString(a.get(6))); |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  | addTrace(t); |
|  |  |
|  | }); |
|  |  |
|  | Dataset ds = createDF(); |
|  | Dataset<Row> d = tc.predict\_realtime(ds); |
|  |  |
|  |  |
|  | java.util.Date dt = new java.util.Date(); |
|  | java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS"); |
|  | String currentTime = sdf.format(dt); |
|  |  |
|  |  |
|  | d.foreach((ForeachFunction<Row>) row -> |
|  | dbm.insertTrace( row.getInt(7), // no |
|  | row.getInt(8), // time |
|  | row.getInt(9), // source |
|  | row.getInt(10), // destination |
|  | row.getInt(11), // protocol |
|  | row.getDouble(12), // lenght  row.getDouble(13), // Info |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  | row.getString(14), // application |
|  | Double.toString(row.getDouble(15)), // prediction |
|  | currentTime, |
|  | row.getString(16)) // predictedLabel |
|  | //System.out.print(row.schema()) |
|  | ); |
|  | } |
|  |  |
|  | }); |
|  |  |
|  | ssc.start(); |
|  | ssc.awaitTermination(); |
|  | } |
|  |  |
|  | private static void clearArray() { |
|  | //System.out.println("Clearing Array..."); |
|  | traces\_array = new ArrayList<Trace>(); |
|  | } |
|  |  |
|  | private static void addTrace(Trace t) { |
|  | //System.out.println("Adding new trace..."); |
|  | traces\_array.add(t); |
|  | //System.out.println("Size of Array is: " + traces\_array.size()); |
|  | } |
|  |  |
|  | private static Dataset createDF() { |
|  | //System.out.println("Creating DataFrame with Array of size: " + traces\_array.size()); |
|  | Dataset ds = sqc.createDataFrame(traces\_array, Trace.class); |
|  | return ds; |
|  | } |
|  | } |