|  |
| --- |
|  |
|  | importjava.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; |
|  | import weka.classifiers.Classifier; |
|  | import weka.core.\*; |
|  | import weka.classifiers.trees.J48; |
|  |  |
|  | public final class TraceReceiver\_weka { |
|  | 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 ArrayList attributes; |
|  | private static ArrayList classVal; |
|  | private static Instances dataRaw; |
|  |  |
|  | private static Attribute no; |
|  | private static Attribute time; |
|  | private static Attribute source; |
|  | private static Attribute destination; |
|  | private static Attribute protocol; |
|  | private static Attribute length; |
|  | private static Attribute info; |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  |  |
|  |  |
|  | 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(); |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  | //JavaReceiverInputDStream<String> lines = ssc.socketTextStream( |
|  | // "ec2-35-176-49-16.eu-west-2.compute.amazonaws.com", 31337, StorageLevels.MEMORY\_AND\_DISK\_SER); // SOCKETS |
|  |  |
|  |  |
|  | String path ="/home/mhyark/Documents/uni/fib/tfg/weka-decisiontree.model"; |
|  | cls = null; |
|  | try { |
|  | cls = (J48) SerializationHelper.read(path); |
|  | } catch (Exception ex) { |
|  | //Logger.getLogger(ModelClassifier.class.getName()).log(Level.SEVERE, null, ex); |
|  | System.out.println("\n\n\tERROR LOADING WEKA MODEL\n"); |
|  | } |
|  |  |
|  |  |
|  | no = new Attribute("no"); |
|  | time = new Attribute("time"); |
|  | source = new Attribute("source"); |
|  | destination = new Attribute("destination"); |
|  | protocol = new Attribute("protocol"); |
|  | length = new Attribute("length"); |
|  | info = new Attribute("info"); |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  |  |
|  | // 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(); |
|  | classifyInstance(s); |
|  |  |
|  | }); |
|  | } |
|  |  |
|  | }); |
|  |  |
|  | ssc.start(); |
|  | ssc.awaitTermination(); |
|  | } |
|  |  |
|  |  |
|  | private static void saveInstacene2BD(int no, int time,source, int destination, int protocol, double length, double info) { |
|  |  |
|  | dbm.insertTrace(no,time,source,destination,potocol,lenght,info); |
|  | } |
|  |  |
|  | private static void classifyInstance(String instance) { |
|  | List<String> instance\_fields = Arrays.asList(COMA.split(instance)); |
|  |  |
|  | dataRaw.clear(); |
|  | double[] instanceValue1 = new double[]{Integer.parseInt(instance\_fields.get(0)), Integer.parseInt(instance\_fields.get(1)), Integer.parseInt(instance\_fields.get(2)), Integer.parseInt(instance\_fields.get(3)), Integer.parseInt(instance\_fields.get(4)), Double.parseDouble(instance\_fields.get(5)), Double.parseDouble(instance\_fields.get(6)), Double.parseDouble(instance\_fields.get(7)), Integer.parseInt(instance\_fields.get(8)), Integer.parseInt(instance\_fields.get(9)), Integer.parseInt(instance\_fields.get(10)), Integer.parseInt(instance\_fields.get(11)), Integer.parseInt(instance\_fields.get(12)), Integer.parseInt(instance\_fields.get(13)), Integer.parseInt(instance\_fields.get(14)), Integer.parseInt(instance\_fields.get(15)), 0}; |
|  | dataRaw.add(new DenseInstance(1.0, instanceValue1)); |
|  |  |
|  | String result = null; |
|  | try { |
|  | result = Objects.toString(classVal.get((int)cls.classifyInstance(dataRaw.firstInstance())), null); |
|  | } catch (Exception ex) { |
|  | //Logger.getLogger(ModelClassifier.class.getName()).log(Level.SEVERE, null, ex); |
|  | System.out.println("\n\n\tERROR LOADING WEKA MODEL\n"); |
|  | } |
|  |  |
|  |  |
|  | 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); |
|  | saveInstacene2BD(Integer.parseInt(instance\_fields.get(0)), Integer.parseInt(instance\_fields.get(1)), Integer.parseInt(instance\_fields.get(2)), Integer.parseInt(instance\_fields.get(3)), Integer.parseInt(instance\_fields.get(4)), Double.parseDouble(instance\_fields.get(5)), Double.parseDouble(instance\_fields.get(6)), Double.parseDouble(instance\_fields.get(7)), Integer.parseInt(instance\_fields.get(8)), Integer.parseInt(instance\_fields.get(9)), Integer.parseInt(instance\_fields.get(10)), Integer.parseInt(instance\_fields.get(11)), Integer.parseInt(instance\_fields.get(12)), Integer.parseInt(instance\_fields.get(13)), Integer.parseInt(instance\_fields.get(14)), Integer.parseInt(instance\_fields.get(15)), result, currentTime); |
|  | } |
|  |  |
|  | private static void clearArray() { |
|  | //System.out.println("Clearing Array..."); |
|  | traces\_array = new ArrayList<Trace>(); |
|  | } |
|  | } |