|  |
| --- |
| import org.apache.spark.streaming.{Seconds, StreamingContext}  import org.apache.spark.SparkContext.\_  import org.apache.spark.streaming.twitter.\_  import org.apache.spark.SparkConf  /\*\*  \* Calculates popular hashtags (topics) over sliding 10 and 60 second windows from a Twitter  \* stream. The stream is instantiated with credentials and optionally filters supplied by the  \* command line arguments.  \*  \* Run this on your local machine as  \*  \*/  object TwitterPopularTags {  def main(args: Array[String]) {      if (args.length < 4) {  System.err.println("Usage: TwitterPopularTags <consumer key> <consumer secret> " +  "<access token> <access token secret> [<filters>]")  System.exit(1)  }  StreamingExamples.setStreamingLogLevels()  val Array(consumerKey, consumerSecret, accessToken, accessTokenSecret) = args.take(4)  val filters = args.takeRight(args.length - 4)  // Set the system properties so that Twitter4j library used by twitter stream  // can use them to generat OAuth credentials  System.setProperty("twitter4j.oauth.consumerKey", consumerKey)  System.setProperty("twitter4j.oauth.consumerSecret", consumerSecret)  System.setProperty("twitter4j.oauth.accessToken", accessToken)  System.setProperty("twitter4j.oauth.accessTokenSecret", accessTokenSecret)  val sparkConf = new SparkConf().setAppName("TwitterPopularTags").setMaster("local[2]")  val ssc = new StreamingContext(sparkConf, Seconds(2))  val stream = TwitterUtils.createStream(ssc, None, filters)//Dstream  val hashTags = stream.flatMap(status => status.getText.split(" ").filter(\_.startsWith("#")))  val topCounts60 = hashTags.map((\_, 1)).reduceByKeyAndWindow(\_ + \_, Seconds(60))  .map{case (topic, count) => (count, topic)}  .transform(\_.sortByKey(false))  val topCounts10 = hashTags.map((\_, 1)).reduceByKeyAndWindow(\_ + \_, Seconds(10))  .map{case (topic, count) => (count, topic)}  .transform(\_.sortByKey(false))  // Print popular hashtags  topCounts60.foreachRDD(rdd => {  val topList = rdd.take(10)  println("\nPopular topics in last 60 seconds (%s total):".format(rdd.count()))  topList.foreach{case (count, tag) => println("%s (%s tweets)".format(tag, count))}  })  topCounts10.foreachRDD(rdd => {  val topList = rdd.take(10)  println("\nPopular topics in last 10 seconds (%s total):".format(rdd.count()))  topList.foreach{case (count, tag) => println("%s (%s tweets)".format(tag, count))}  })  ssc.start()  ssc.awaitTermination()  }  } |

|  |
| --- |
| import org.apache.spark.Logging  import org.apache.log4j.{Level, Logger}  /\*\* Utility functions for Spark Streaming examples. \*/  object StreamingExamples extends Logging {  /\*\* Set reasonable logging levels for streaming if the user has not configured log4j. \*/  def setStreamingLogLevels() {  val log4jInitialized = Logger.getRootLogger.getAllAppenders.hasMoreElements  if (!log4jInitialized) {  // We first log something to initialize Spark's default logging, then we override the  // logging level.  logInfo("Setting log level to [WARN] for streaming example." +  " To override add a custom log4j.properties to the classpath.")  Logger.getRootLogger.setLevel(Level.WARN)  }  }  } |

Run the TwitterPopularTags program using below arguments

|  |
| --- |
| 47yiCHhFTzB6PtsTK4hoIMS7M XSDX7w0eqh2X1smO6iOe7EzUsz6cOJhf7Vxd9SzOVBKcZkeLCj 140904884-8iVIRbuIvqJgVGvOSxgImD1NdY1DRHrLhKRGPg7j mp57adaq6LPcVESo7q691rfU83zBeq8uKDep0UnOvdERw narendramodi |

|  |
| --- |
| Usage: TwitterPopularTags <consumer key> <consumer secret> " "<access token> <access token secret> [<filters>] |