

Usecase

- <https://mapr.com/blog/real-time-analysis-popular-uber-locations-spark-structured-streaming-machine-learning-kafka-and-mapr-db/>

Usecase – Twitter sentiment analysis



Trending Topics can be used to create campaigns and attract larger audience.

Sentiment Analytics helps in crisis management, service adjusting and target marketing.

- ❑ Sentiment refers to the emotion behind a social media mention online.
- ❑ Sentiment Analysis is categorising the tweets related to particular topic and performing data mining using Sentiment Automation Analytics Tools.
- ❑ We will be performing Twitter Sentiment Analysis as our Use Case for Spark Streaming.

FACEBOOK	TWITTER
<p>TRENDING</p> <ul style="list-style-type: none">➤ Kourtney Kardashian: Kourtney Kardashian Confirms: 'I'm Pregnant'➤ Miss USA: Miss Nevada Nia Sanchez crowned as 63rd Miss USA➤ Sir Mix-a-Lot: Sir Mix-a-Lot Takes 'Baby Got Back' Classical With Beatz...	<p>Trends ▾ Change</p> <ul style="list-style-type: none">#XboxE3 PromotedRik MayallThe Young Ones#E32014#FireSideTieOnSaleNowPhantom DustThe DivisionRise of the Tomb Raider#TonyAwards#SoundCloud

Figure: Facebook And Twitter Trending Topics

Problem statement



Problem Statement

To design a Twitter Sentiment Analysis System where we populate real time sentiments for crisis management, service adjusting and target marketing

Sentiment Analysis is used to:

- Predict the success of a movie
- Predict political campaign success
- Decide whether to invest in a certain company
- Targeted advertising
- Review products and services

Sentiment analysis for Nike



Figure: Twitter Sentiment Analysis For Nike

Sentiment analysis for Adidas



Figure: Twitter Sentiment Analysis For Adidas

Importing packages

```
//Import the necessary packages into the Spark Program
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.SparkContext.
import org.apache.spark.streaming.twitter.
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext.
import org.apache.spark.
import org.apache.spark.rdd.
import org.apache.spark.rdd.RDD
import org.apache.spark.SparkContext.
import org.apache.spark.sql
import org.apache.spark.storage.StorageLevel
import scala.io.Source
import scala.collection.mutable.HashMap
import java.io.File
```

Twitter token authorization

```
object mapr {

  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()
    //Passing our Twitter keys and tokens as arguments for authorization
    val Array(consumerKey, consumerSecret, accessToken,
    accessTokenSecret) = args.take(4)
    val filters = args.takeRight(args.length - 4)
  }
}
```

Dstream transformation

```
// Set the system properties so that Twitter4j library used by twitter stream
// Use them to generate 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("Sentiments").setMaster("local[2]")
val ssc = new StreamingContext(sparkConf, Seconds(5))
val stream = TwitterUtils.createStream(ssc, None, filters)

//Input DStream transformation using flatMap
val tags = stream.flatMap { status =>
status.getHashtagEntities.map(_.getText)}
```

Generating tweet data

```
//RDD transformation using sortBy and then map function
tags.countByValue()
    .foreachRDD { rdd =>
    val now = org.joda.time.DateTime.now()
    rdd
        .sortBy(_.value)
        .map(x => (x, now))
//Saving our output at ~/twitter/ directory
    .saveAsTextFile(s"~/twitter/$now")
}

//DStream transformation using filter and map functions
val tweets = stream.filter {t =>
    val tags = t.getText.split(" ")
        .filter(_.startsWith("#"))
        .map(_.toLowerCase)
    tags.exists { x => true }
}
```

Extracting sentiments

```
val data = tweets.map { status =>
    val sentiment = SentimentAnalysisUtils.detectSentiment(status.getText)
    val tagss = status.getHashtagEntities.map(_.getText.toLowerCase)
    (status.getText, sentiment.toString, tagss.toString())
}

data.print()
//Saving our output at ~/ with filenames starting like twitterss
data.saveAsTextFiles("~/twitterss", "20000")

ssc.start()
ssc.awaitTermination()
}
```