**Experiment No 11**

**Aim :** Case study to perform Sentiment analysis using Spark Streaming

**Theory**

**Spark Streaming**

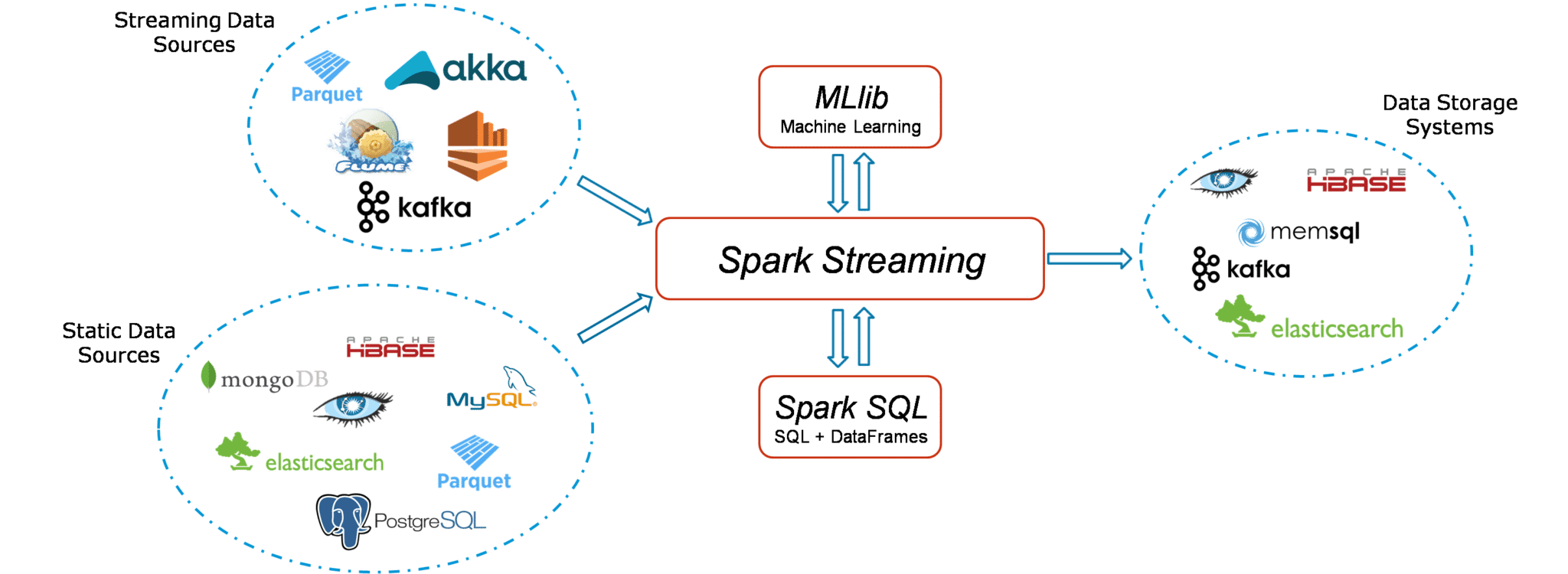
Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark Streaming can be used to stream live data and processing can happen in real time. Spark Streaming’s ever-growing user base consists of household names like Uber, Netflix and Pinterest.

**Why Spark Streaming?**

We can use Spark Streaming to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses. Spark Streaming is used for processing real-time streaming data. It is a useful addition to the core Spark API. Spark Streaming enables high-throughput and fault-tolerant stream processing of live data streams.

**Spark Streaming Features**

1. Scaling: Spark Streaming can easily scale to hundreds of nodes.
2. Speed: It achieves low latency.
3. Fault Tolerance: Spark has the ability to efficiently recover from failures.
4. Integration: Spark integrates with batch and real-time processing.
5. Business Analysis: Spark Streaming is used to track the behavior of customers which can be used in business analysis.



**Using Spark Streaming to perform Sentiment Analysis on Twitter Data**

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

**Applications of Sentiment Analysis**

1. Predict the success of a movie
2. Predict political campaign success
3. Decide whether to invest in a certain company
4. Targeted advertising
5. Review products and services

**Fundamentals Used**

1. Streaming Context

Streaming Context consumes a stream of data in Spark. It registers an Input DStream to produce a Receiver object. It is the main entry point for Spark functionality. Spark provides a number of default implementations of sources like Twitter, Akka Actor and ZeroMQ that are accessible from the context. A StreamingContext object can be created from a SparkContext object. A SparkContext represents the connection to a Spark cluster and can be used to create RDDs, accumulators and broadcast variables on that cluster.

1. DStream

Discretized Stream (DStream) is the basic abstraction provided by Spark Streaming. It is a continuous stream of data. It is received from a data source or a processed data stream generated by transforming the input stream. Internally, a DStream is represented by a continuous series of RDDs and each RDD contains data from a certain interval.

1. Input DStreams

Input DStreams are DStreams representing the stream of input data received from streaming sources. Every input DStream is associated with a Receiver object which receives the data from a source and stores it in Spark’s memory for processing.

1. Output DStreams

Output operations allow DStream’s data to be pushed out to external systems like databases or file systems. Output operations trigger the actual execution of all the DStream transformations.

1. Caching

DStreams allow developers to cache/ persist the stream’s data in memory. This is useful if the data in the DStream will be computed multiple times. This can be done using the persist() method on a DStream. For input streams that receive data over the network (such as Kafka, Flume, Sockets, etc.), the default persistence level is set to replicate the data to two nodes for fault-tolerance.

1. Accumulators

Accumulators are variables that are only added through an associative and commutative operation. They are used to implement counters or sums. Tracking accumulators in the UI can be useful for understanding the progress of running stages. Spark natively supports numeric accumulators. We can create named or unnamed accumulators.

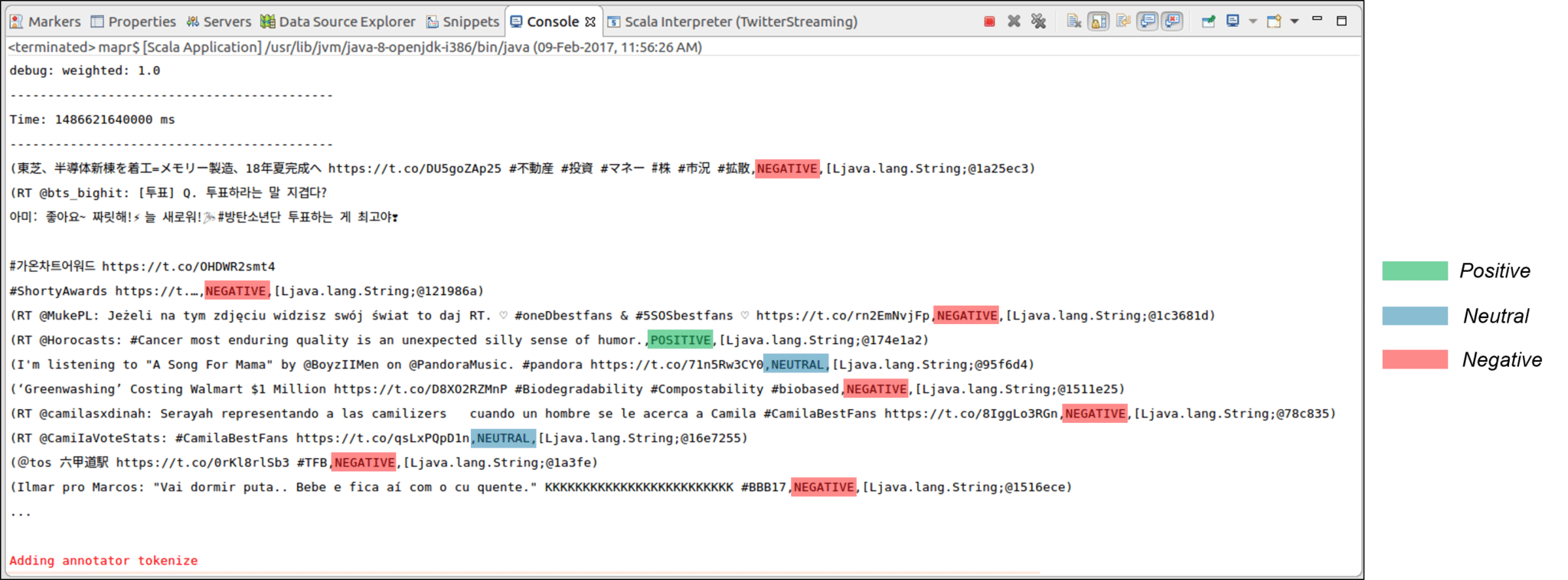
1. Broadcast Variables

Broadcast variables allow the programmer to keep a read-only variable cached on each machine rather than shipping a copy of it with tasks. They can be used to give every node a copy of a large input dataset in an efficient manner. Spark also attempts to distribute broadcast variables using efficient broadcast algorithms to reduce communication cost.

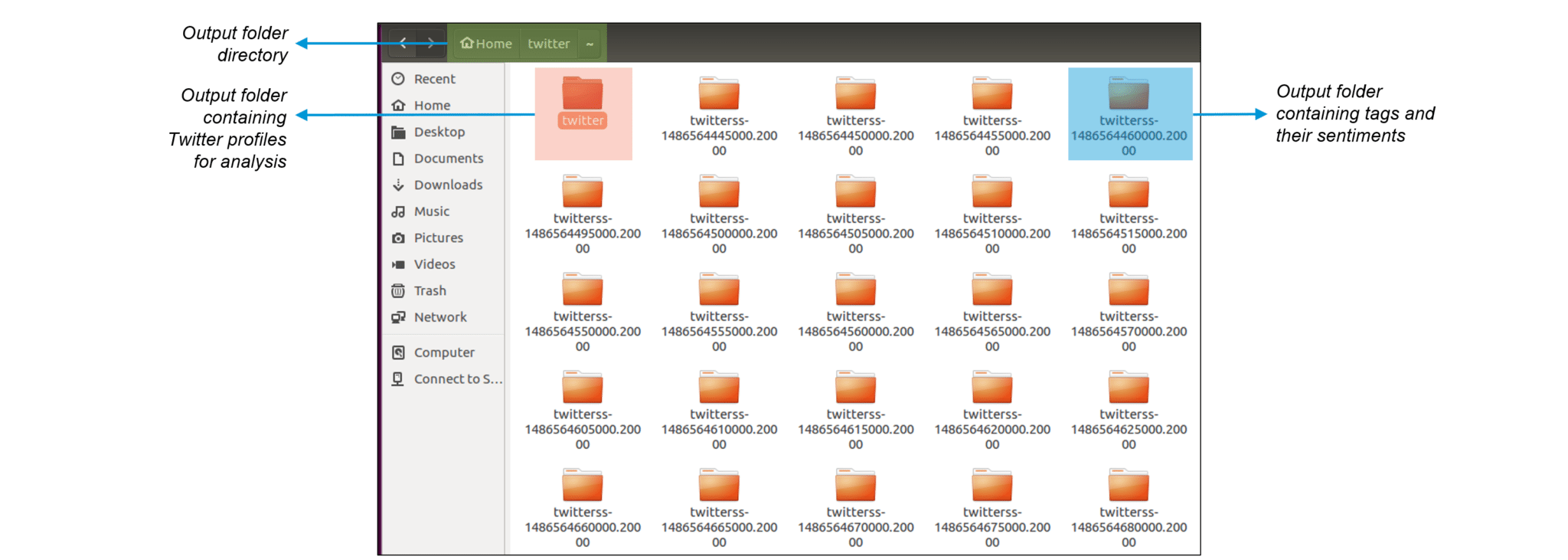
1. Checkpoints

Checkpoints are similar to checkpoints in gaming. They make it run 24/7 and make it resilient to failures unrelated to the application logic.

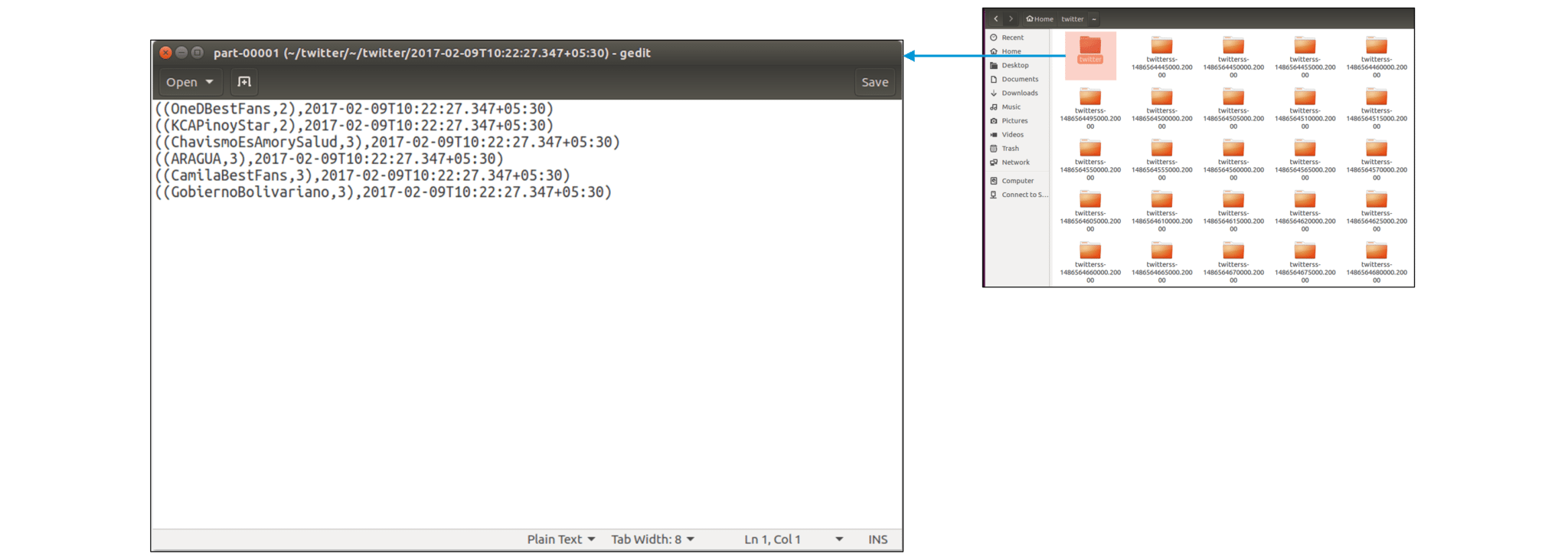
**Results of Sentiment Analysis**



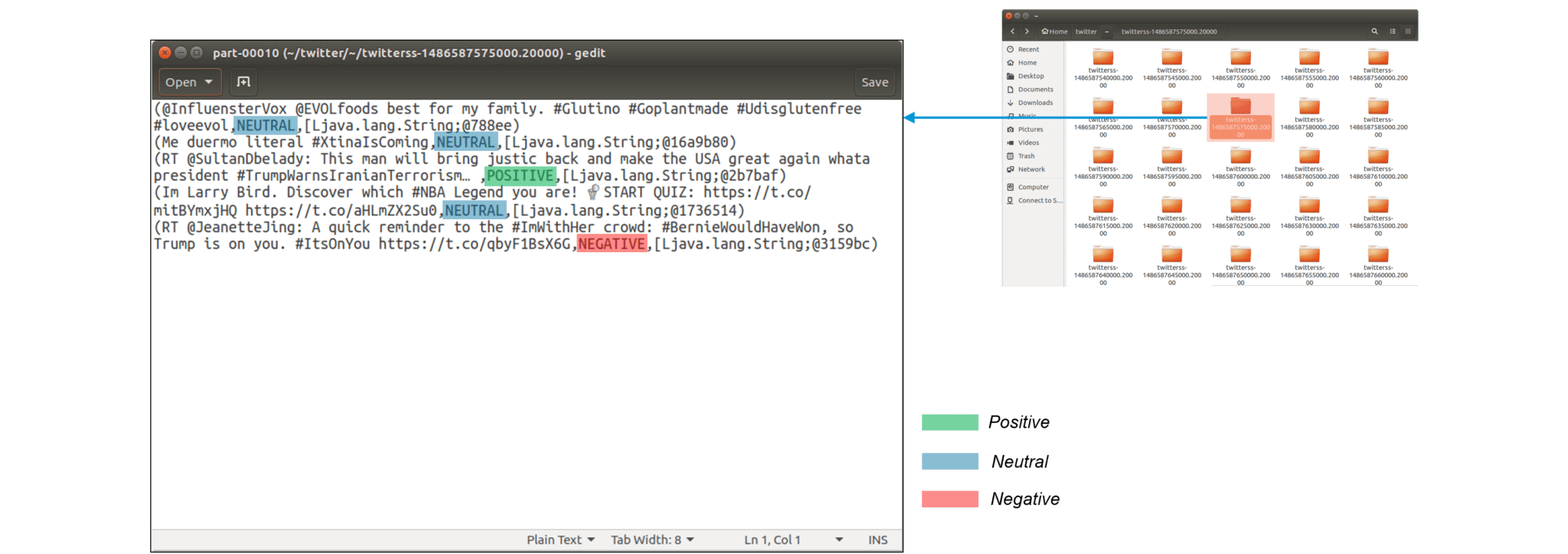
As we can see in the screenshot, all the tweets are categorized into Positive, Neutral and Negative according to the sentiment of the contents of the tweets. The output of the Sentiments of the Tweets is stored into folders and files according to the time they were created. This output can be stored on the local file system or HDFS as necessary. The output directory looks like this:



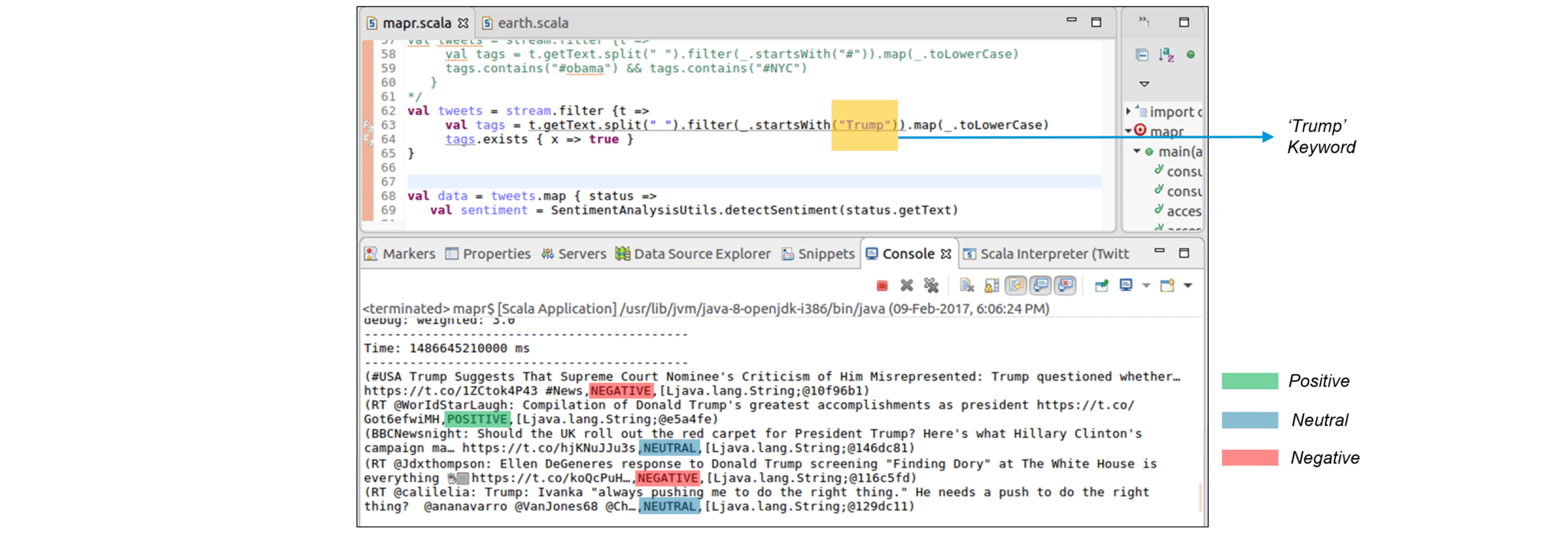
Here, inside the twitter directory, we can find the usernames of the Twitter users along with the timestamp for every tweet as shown below:



Now that we have got the Twitter usernames and timestamp, let us look at the Sentiments and tweets stored in the main directory. Here, every tweet is followed by the sentiment emotion. This Sentiment that is stored is further used for analysing a vast multitude of insights by companies.



Now, let us modify our code a little to get sentiments for specific hashtags (topics). We look at the sentiments associated with the keyword ‘Trump‘.



**Conclusion**

Spark Streaming is a part of Spark API that provides tolerant processing of live data streams. In this experiment we took live streams of data from the twitter API and used the live stream of data for performing Sentiment Analysis on the tweets received from the Twitter API. The Spark Stream used provides fast, scalable and fault tolerant processing of live data streams.