Twitter Sentiment Analysis using Lambda Architecture

Iacopo Erpichini and Federico Magnolfi

University of Florence

Outline

Sentiment Analysis

What is *Sentiment Analysis*

Interpretation and classification of **emotions** within text data using text analysis techniques

Why *Sentiment Analysis*

Companies can understand if their products are **appreciated** by customers

Sentiment Analysis

What is *Sentiment Analysis*

Interpretation and classification of **emotions** within text data using text analysis techniques

Why *Sentiment Analysis*

Companies can understand if their products are **appreciated** by customers

To do *SA* you need to analyze a large amount of data

Tweets Sentiment Analysis

Objective

- Analyze sentiments for specific **keywords**
- Analyze sentiments in different **periods**

Tweets Sentiment Analysis

Objective

- Analyze sentiments for specific **keywords**
- Analyze sentiments in different **periods**

Example keyword: Apple, Facebook, Google...
The granularity of periods can be: per hour, per day, per year...

Tweets Sentiment Analysis

Objective

- Analyze sentiments for specific **keywords**
- Analyze sentiments in different **periods**

Example keyword: Apple, Facebook, Google...
The granularity of periods can be: per hour, per day, per year...

Solution

Each pair <`period, keyword`> must have its own counters

Tweets Classification with LingPipe

sentiment140 dataset

This dataset contains **1.6 Million tweets**. Each tweet is annotated with the the **sentiment** (0 = negative, 4 = positive).

Classification

- We trained a **LingPipe classifier** on *sentiment140* dataset
- We saved the model so that it can be loaded to classify tweets
- The classifier has a training accuracy of $\sim 80\%$, but this wasn't the focus of this work

Tweets Generation

The **Generator** simulates the emission of tweets by users

- **Tweets** are sampled from **sentiment140 dataset**
- Each generated tweet is associated with **current timestamp**
- Concatenate tweet's timestamp and text $\rightarrow$ **line** of a file
- **Files** put into `/batch/input` & `/speed/input` *HDFS* folders

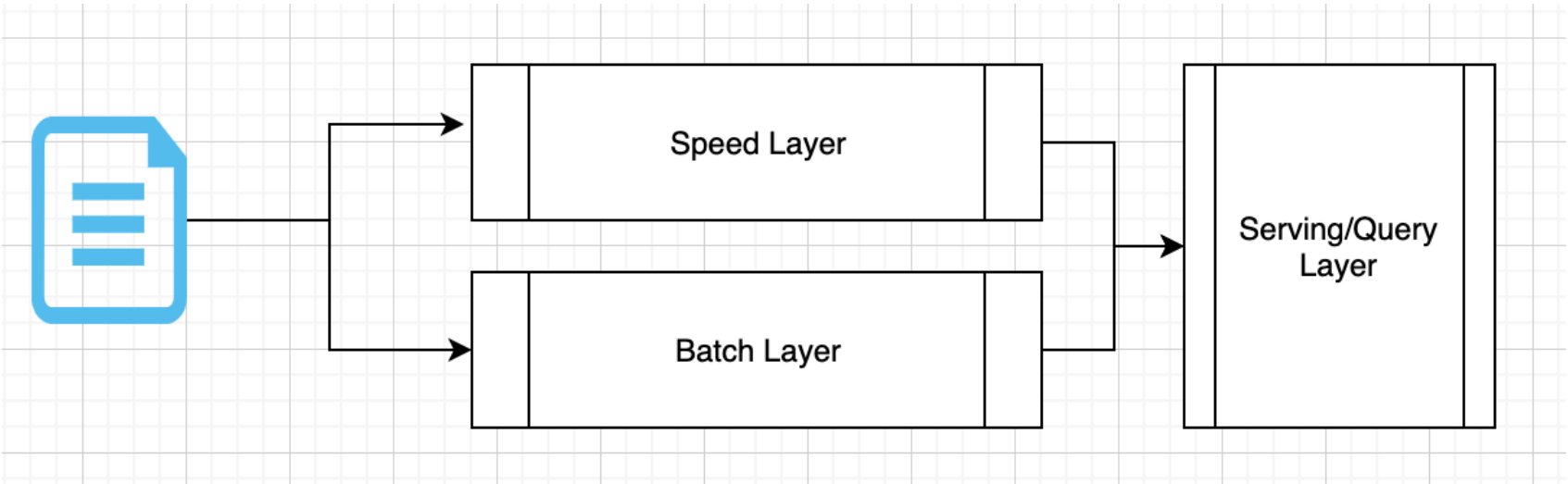
Tweets Generation

The **Generator** simulates the emission of tweets by users

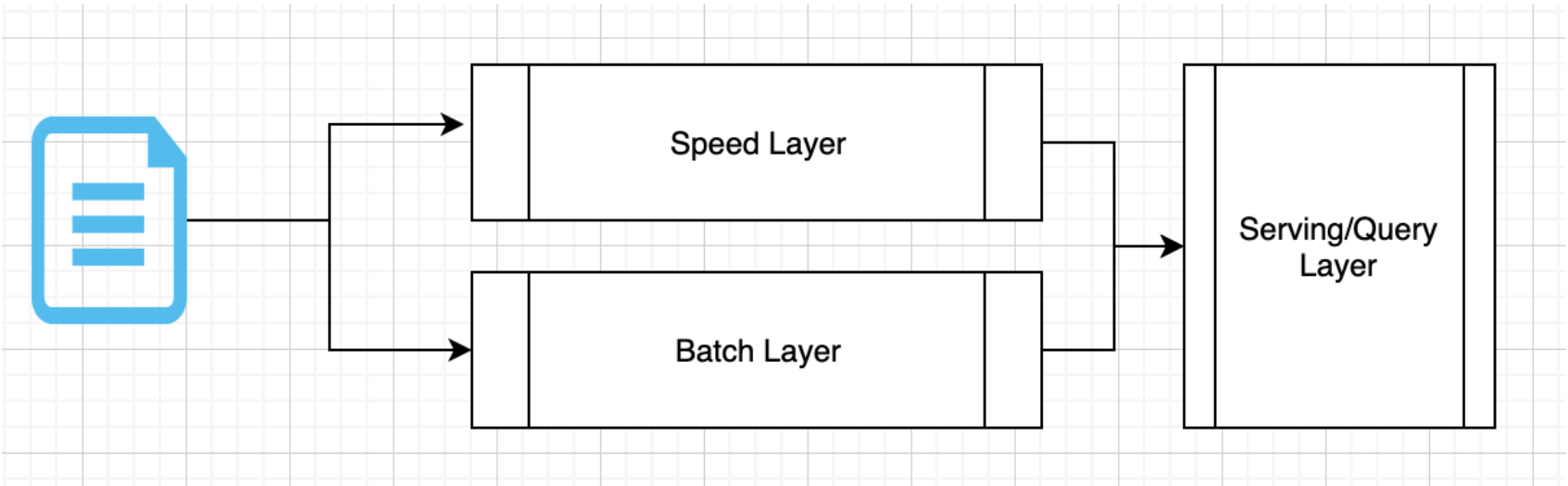
- **Tweets** are sampled from **sentiment140 dataset**
- Each generated tweet is associated with **current timestamp**
- Concatenate tweet's timestamp and text → **line** of a file
- **Files** put into `/batch/input` & `/speed/input` *HDFS* folders

Some parameters control **dimension** of the files

Lambda Architecture

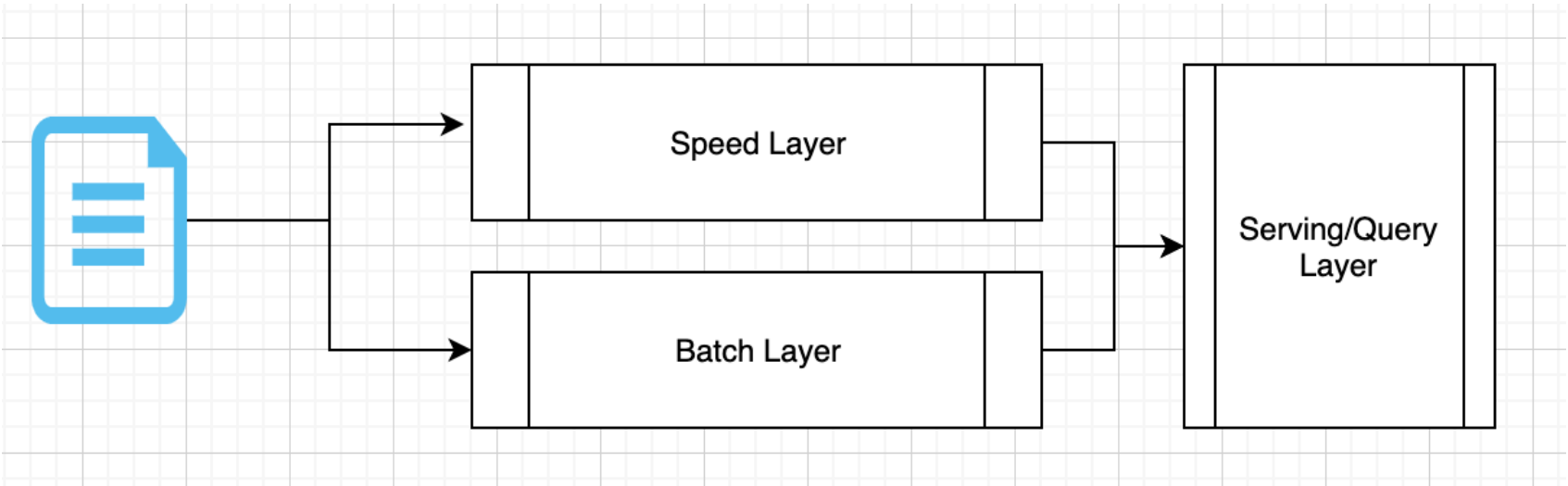


Lambda Architecture



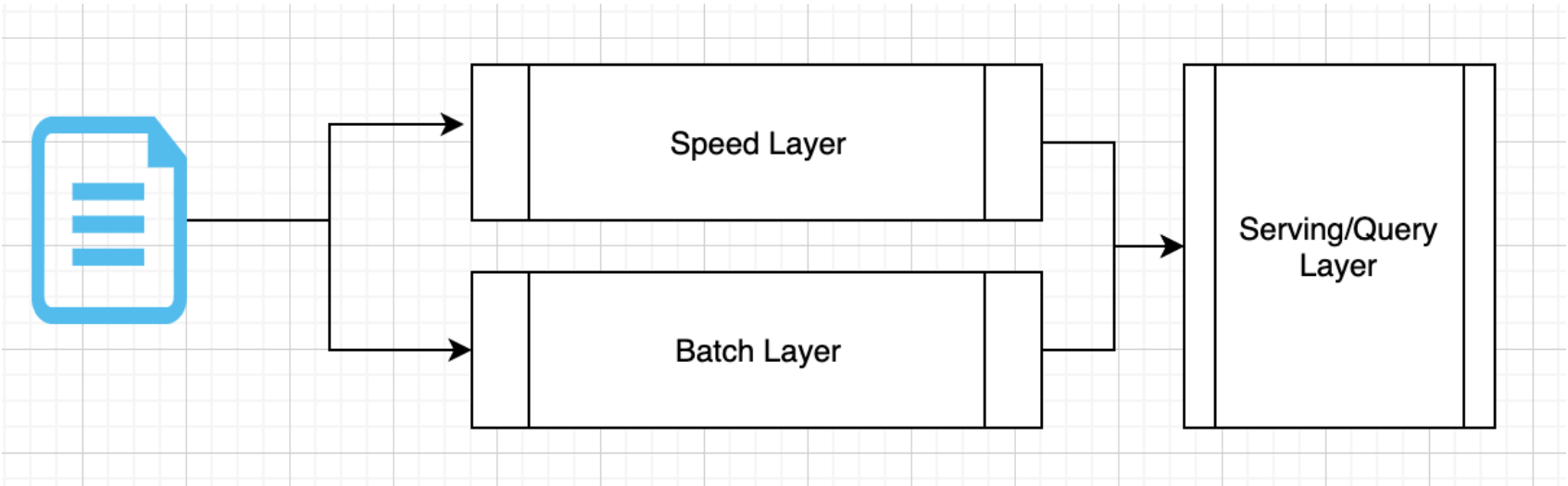
- **Batch Layer**: creates a view from the *master dataset*

Lambda Architecture



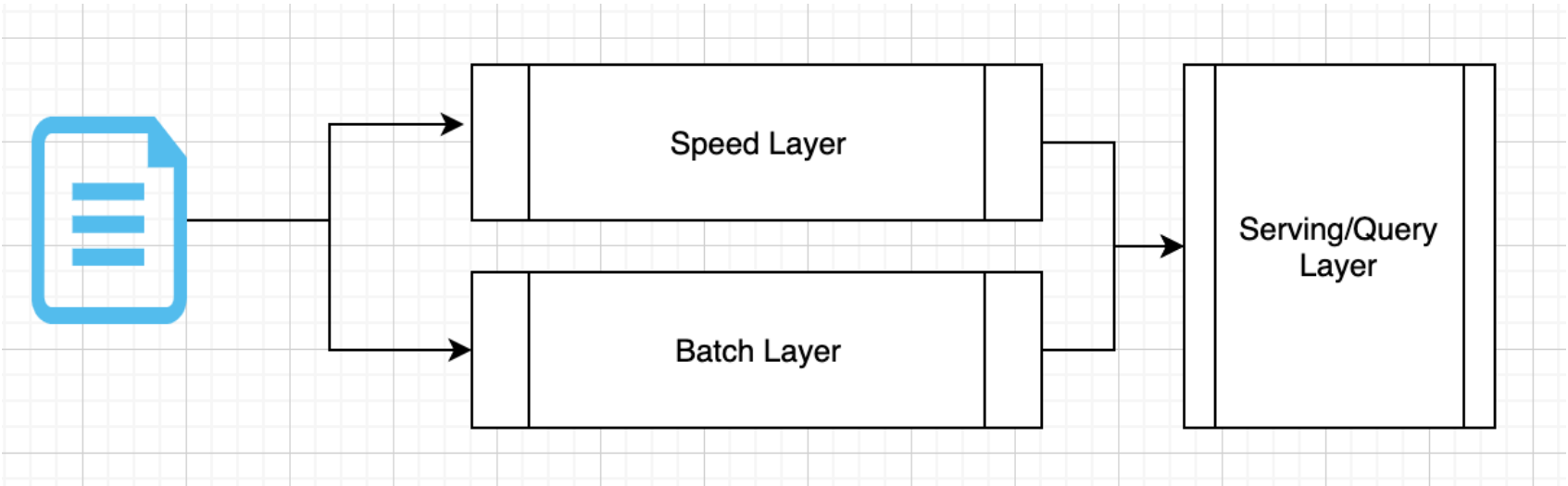
- **Batch Layer**: creates a view from the *master dataset*
- **Speed Layer**: creates a view from *recent data*

Lambda Architecture



- **Batch Layer**: creates a view from the *master dataset*
- **Speed Layer**: creates a view from *recent data*
- **Query Layer**: answers to *real-time queries* reading both views

Lambda Architecture

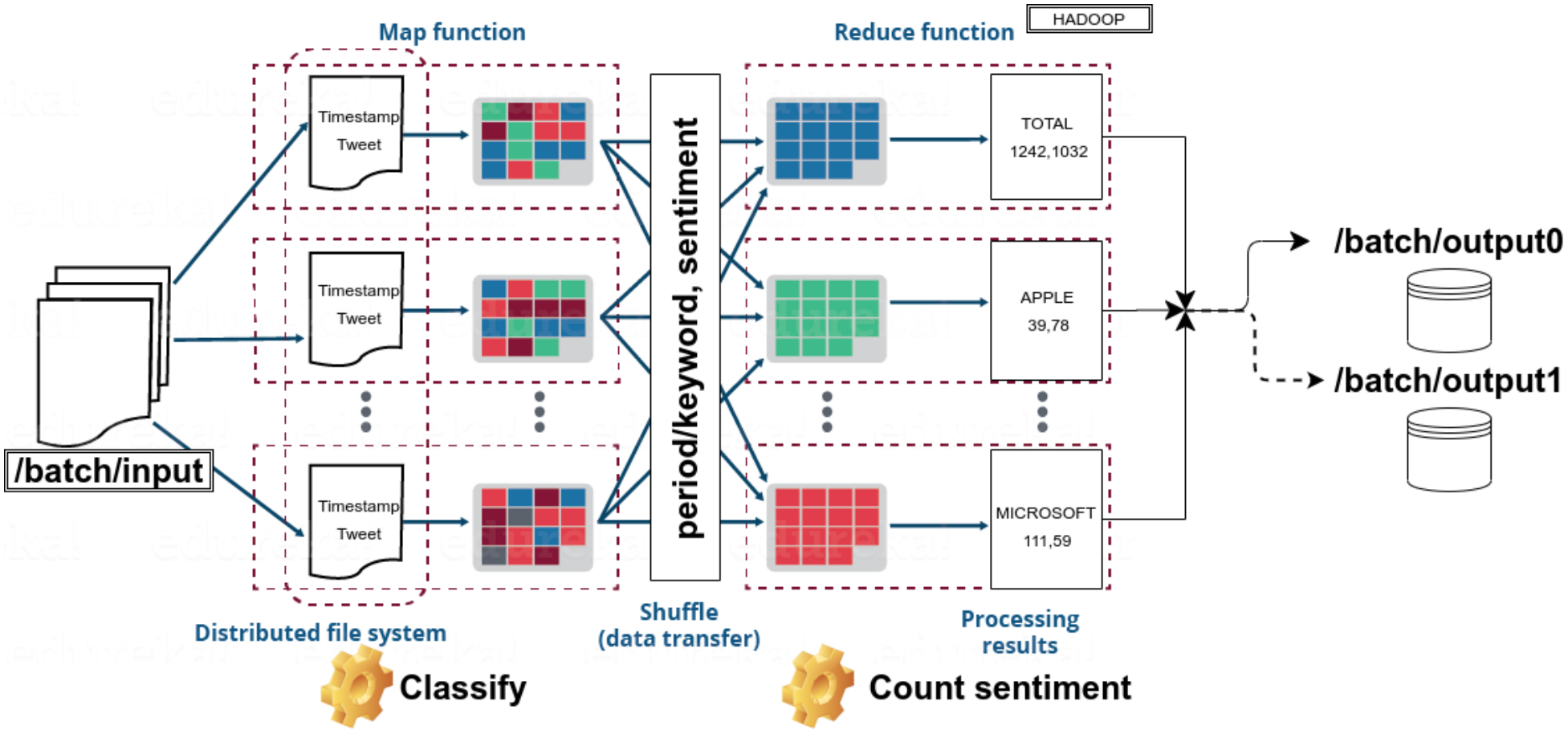


- **Batch Layer**: creates a view from the *master dataset*
- **Speed Layer**: creates a view from *recent data*
- **Query Layer**: answers to *real-time queries* reading both views

Communication between layers

Could happen via the HDFS, or via a distributed database

Batch Layer Schema



Mapper pseudo-code

Algorithm 1: Mapper

Input : tweet
Output: <period/keyword, sentiment>*

```
1 tweetTimestamp, tweetText = tweet.split(",")
2 classifier = loadClassifier()
3 sentiment = classifier.evaluate(tweetText)
4 for word in tweetText.split(" ") do
5     outputKey = tweetTimestamp + "/" + word
6     context.write(outputKey, sentiment)
7 end
8 period = getPeriod(tweetTimestamp)
9 context.write(period + "/total", sentiment)
```

Reducer pseudo-code

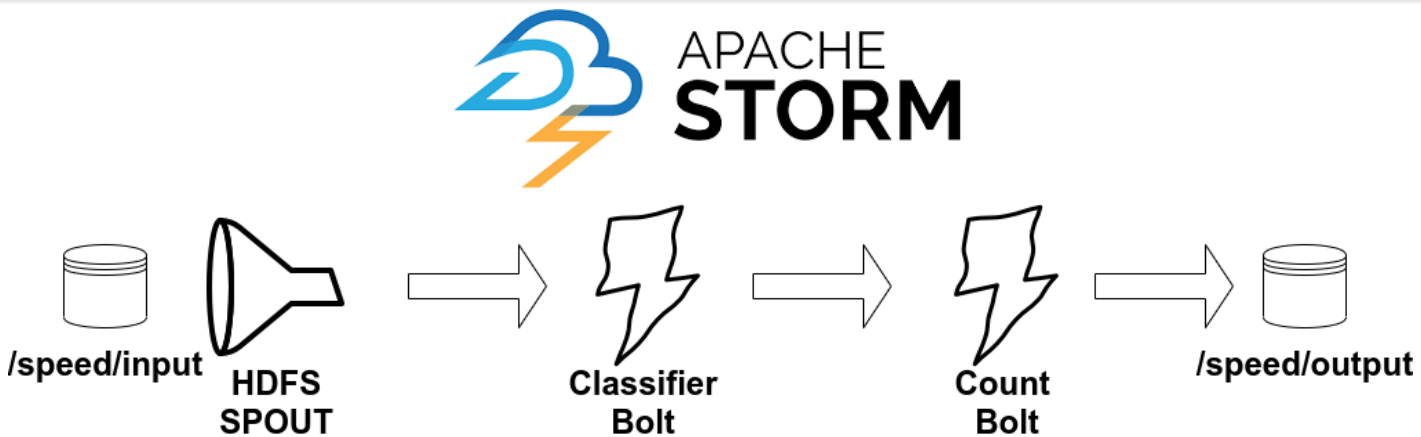
Algorithm 2: Reducer

Input : <key, sentiments>
Output: <key, stats>

```
1  numGood = 0
2  numBad = 0
3  for sentiment in sentiments do
4      if sentiment == 0 then
5          numBad++
6      else
7          numGood++
8  end
9  stats = str(numGood) + "," + str(numBad)
10 context.write(key, stats)
```

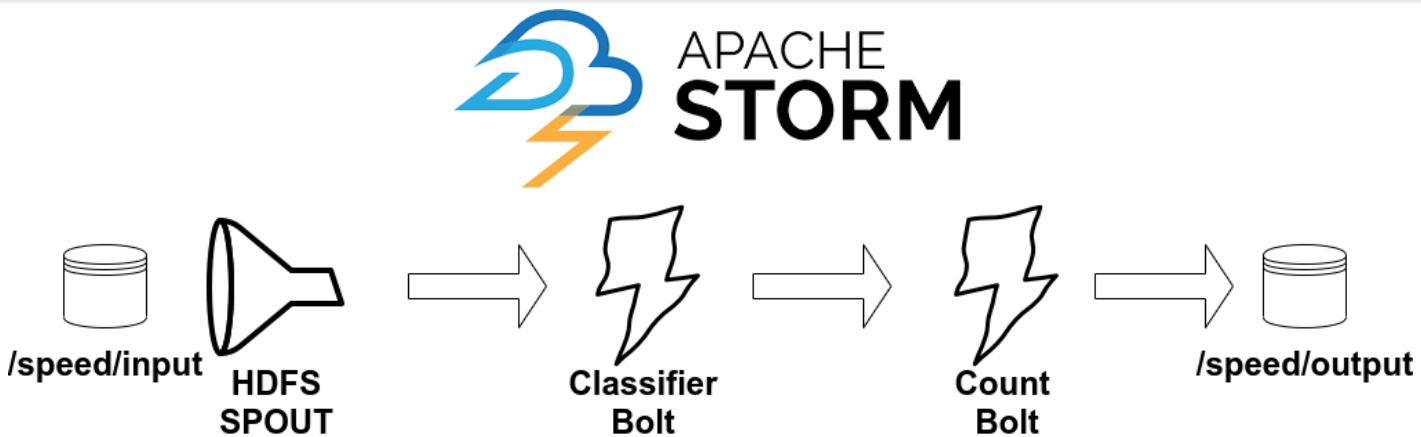
Speed Layer

- The **HdfsSpout** inserts input data into the *topology*
- The *topology* is similar to the MapReduce pattern
- **Classifier Bolt** $\leftrightarrow$ Mapper
- **Count Bolt** $\leftrightarrow$ Reducer



Speed Layer

- The **HdfsSpout** inserts input data into the *topology*
- The *topology* is similar to the MapReduce pattern
- **Classifier Bolt** $\leftrightarrow$ Mapper
- **Count Bolt** $\leftrightarrow$ Reducer



Counters have to be **reset** when Batch Layer finishes computation: therefore, a **synchronization** is required

Ping Pong schema

Generated tweets can be:

1. **Not yet seen** by the batch layer
2. **In processing** by the batch layer for the first time
3. **Already processed** by the batch layer

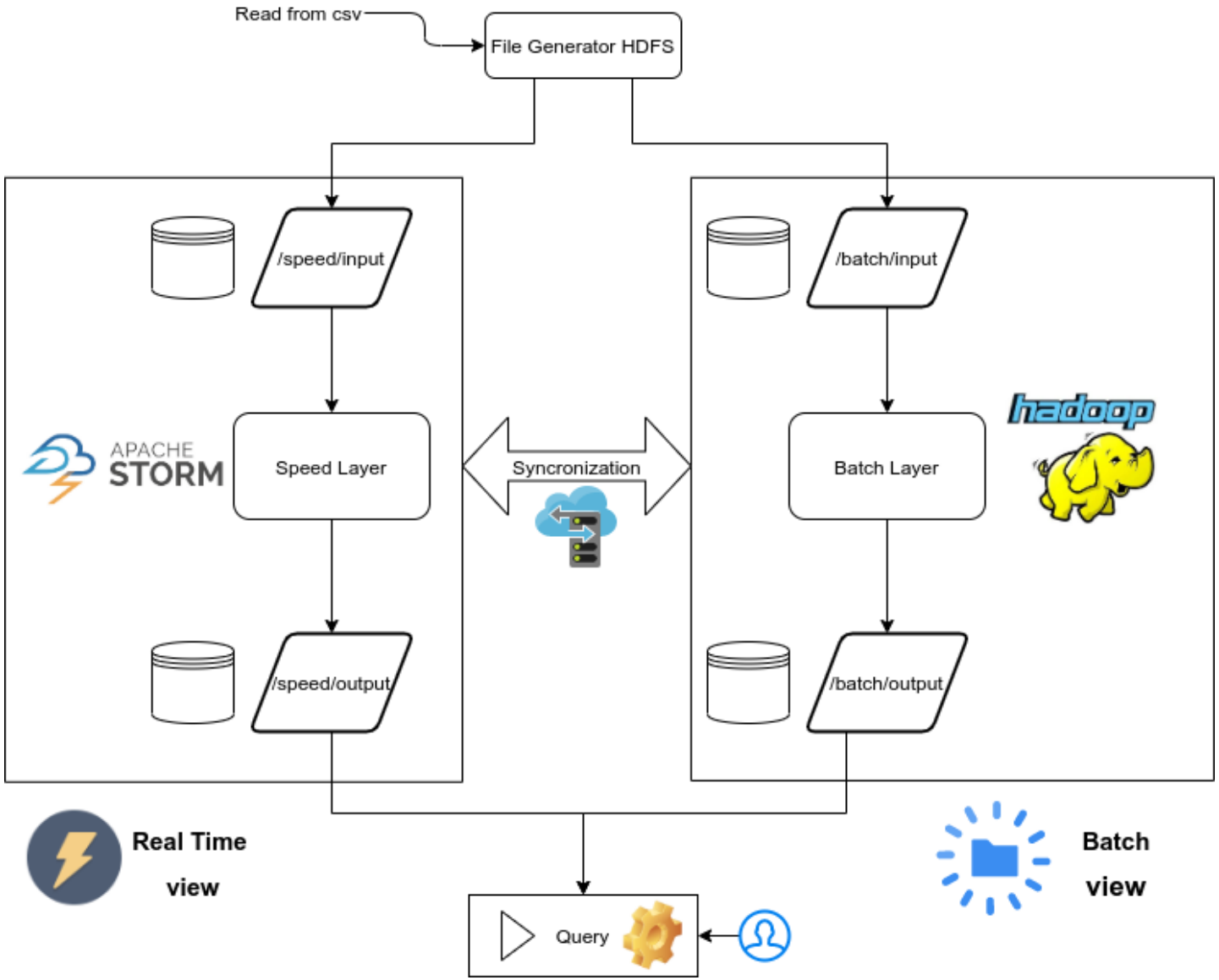
Batch layer

- writes last output folder in `/sync`
- writes processed and inProgress timestamps in `/sync`

Speed layer

- reads inProgress timestamp from `/sync`
- counts tweet after inProgress timestamp only
- different output folders, depending by inProgress timestamp

Complete architecture



Query

Query polling iteration

- Reads info about folders from `/sync` on *HDFS*
- Reads **last folder** tweets stats from *Batch View*
- Reads **after processed** and **after inProgress** tweets stats from *Speed View*

Query

Query polling iteration

- Reads info about folders from `/sync` on *HDFS*
- Reads **last folder** tweets stats from *Batch View*
- Reads **after processed** and **after inProgress** tweets stats from *Speed View*

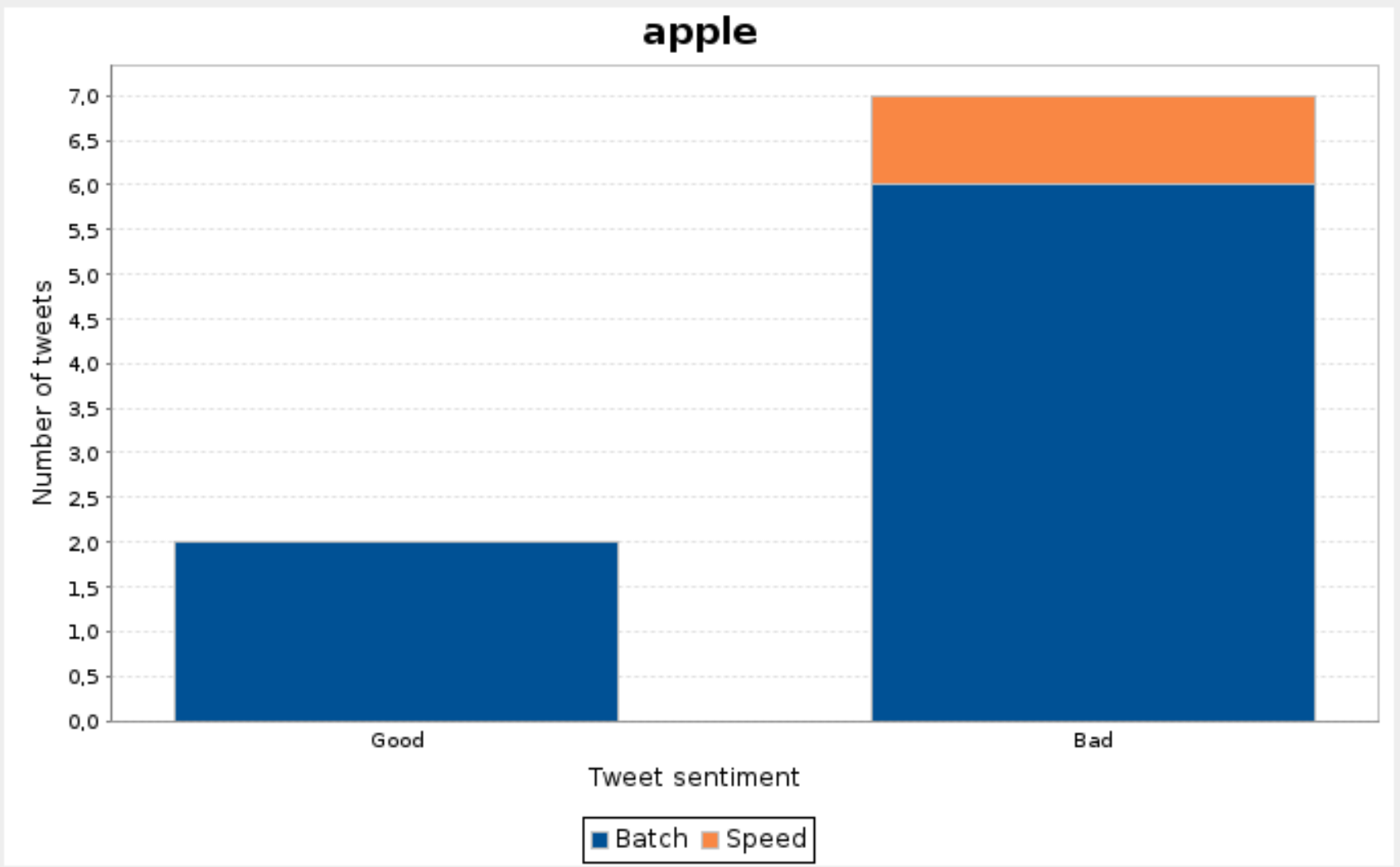
Combined stats are displayed via a Graphical User Interface

Query GUI

Graphical User Interface

- **Time interval** specifiable by entering the start and end date
- Sentiment analysis of tweets sent in the chosen interval
- **Real-time summary** for all tweets and for specified keywords

Stacked bar plot



Graphical User Interface

End

Thanks for the attention