Twitter Sentiment Analysis using Lambda Architecture

Iacopo Erpichini and Federico Magnolfi

University of Florence

Outline

Sentiment Analysis

What

Sentiment analysis it's the interpretation and classification of emotions within voice and text data using text analysis techniques

Why

It allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback

Sentiment Analysis

What

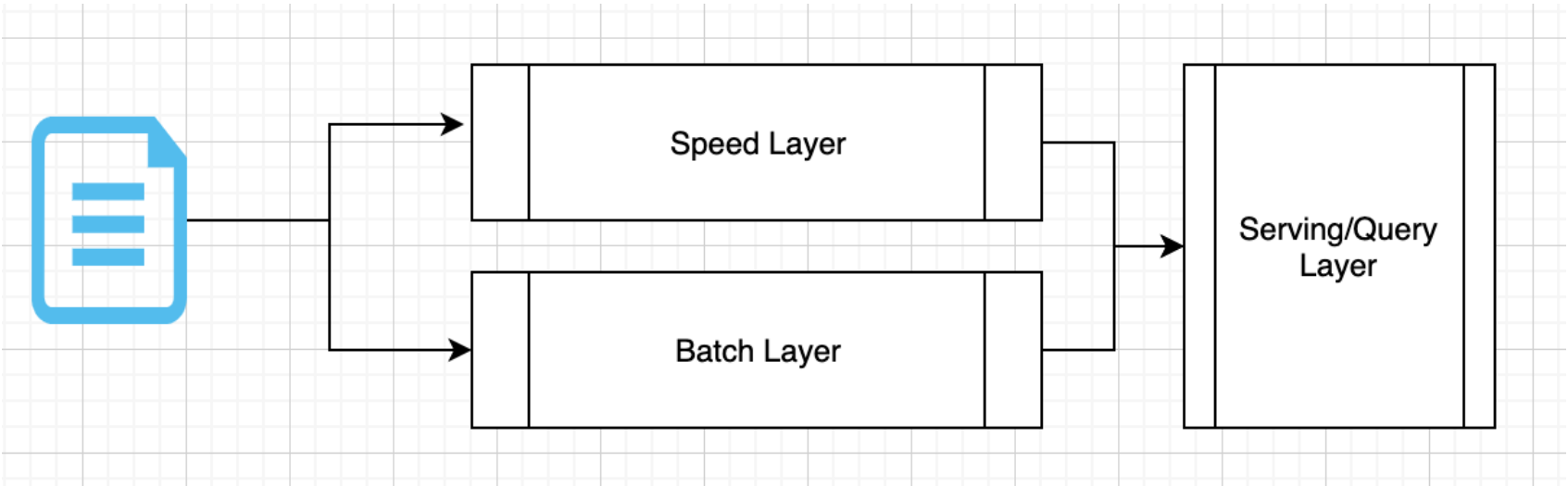
Sentiment analysis it's the interpretation and classification of emotions within voice and text data using text analysis techniques

Why

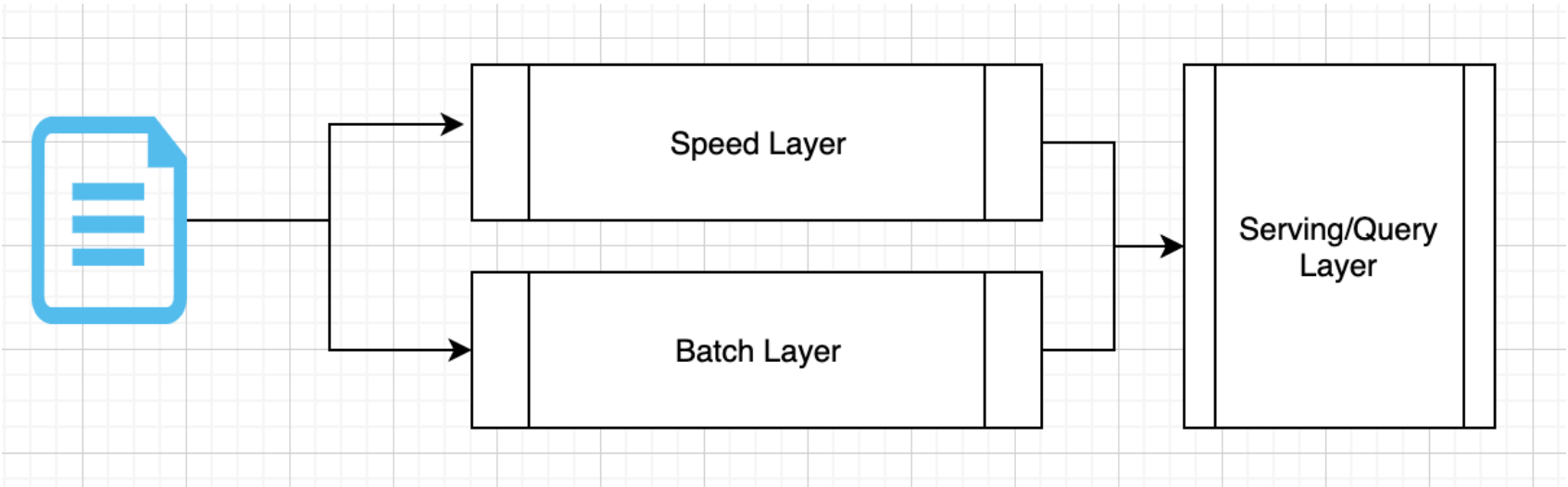
It allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback

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

Lambda Architecture

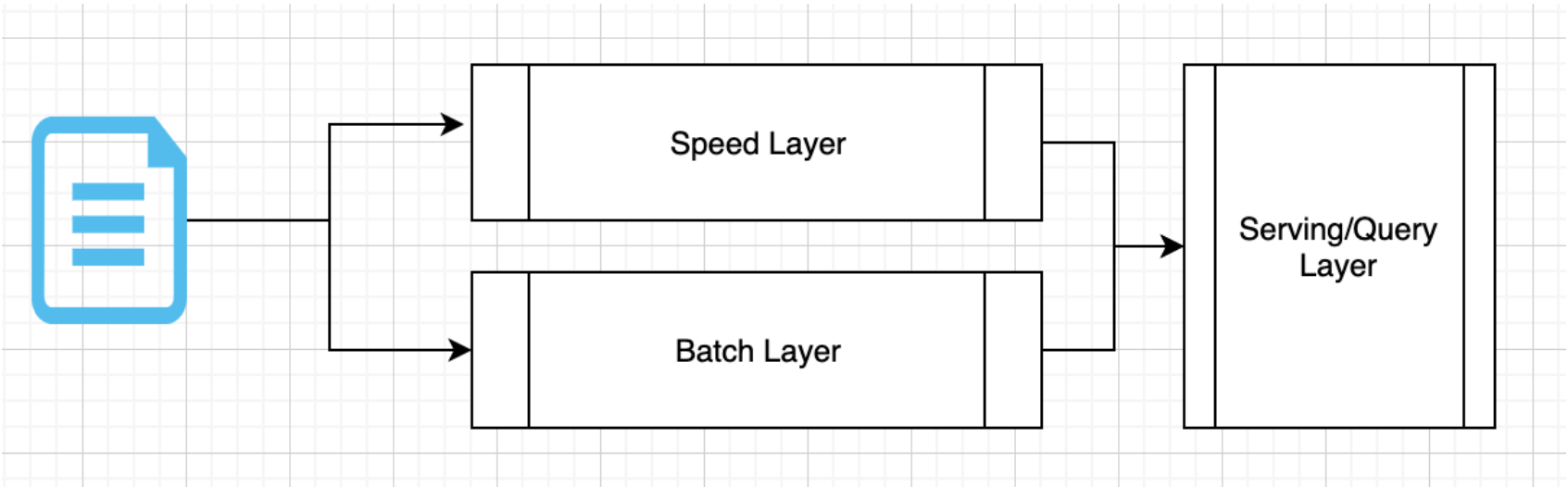


Lambda Architecture



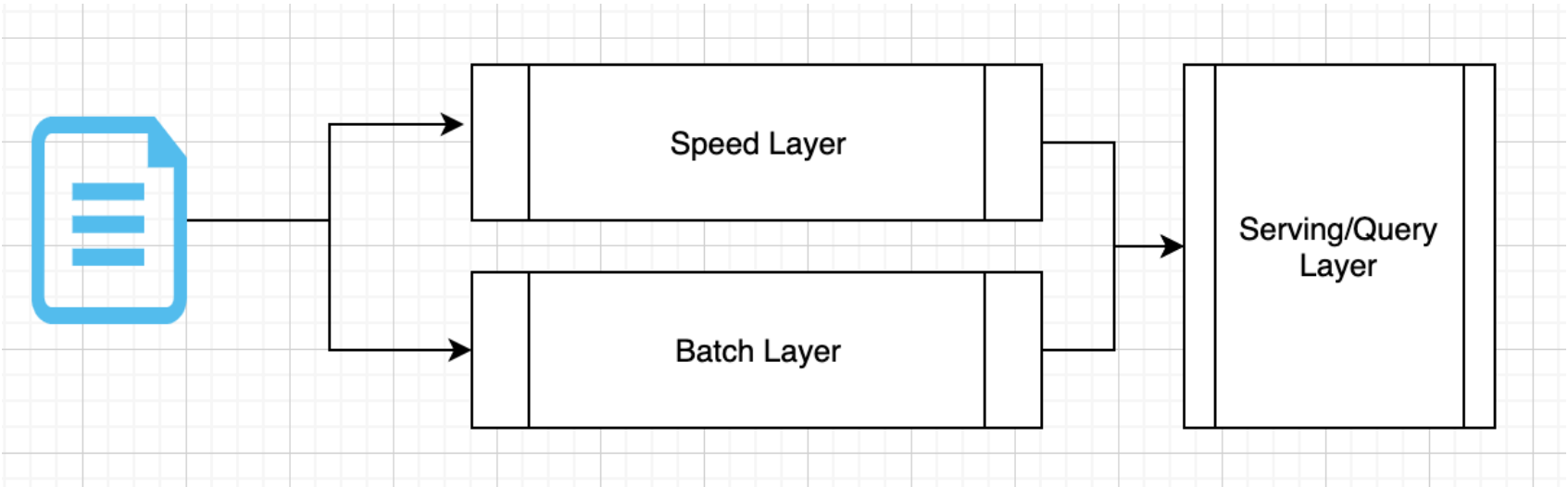
- **Batch Layer**: stores an immutable, constantly growing *master dataset* and compute *arbitrary functions* on that dataset

Lambda Architecture



- **Batch Layer**: stores an immutable, constantly growing *master dataset* and compute *arbitrary functions* on that dataset
- **Speed Layer**: compute similar functions to the batch layer ones but on *recent data*

Lambda Architecture



- **Batch Layer**: stores an immutable, constantly growing *master dataset* and compute *arbitrary functions* on that dataset
- **Speed Layer**: compute similar functions to the batch layer ones but on *recent data*
- **Serving Layer**: answers to *real-time queries* reading batch and speed views

Lambda Architecture

The Lambda Architecture is summarized by these three functions:

- $batchView = function(allData)$
- $realtimeView = function(realtimeView, newData)$
- $query = function(batchView, realtimeView)$

Lambda Architecture

The Lambda Architecture is summarized by these three functions:

- $batchView = function(allData)$
- $realtimeView = function(realtimeView, newData)$
- $query = function(batchView, realtimeView)$

Communication

Communication between layers could happen via the HDFS, or via a distributed database

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

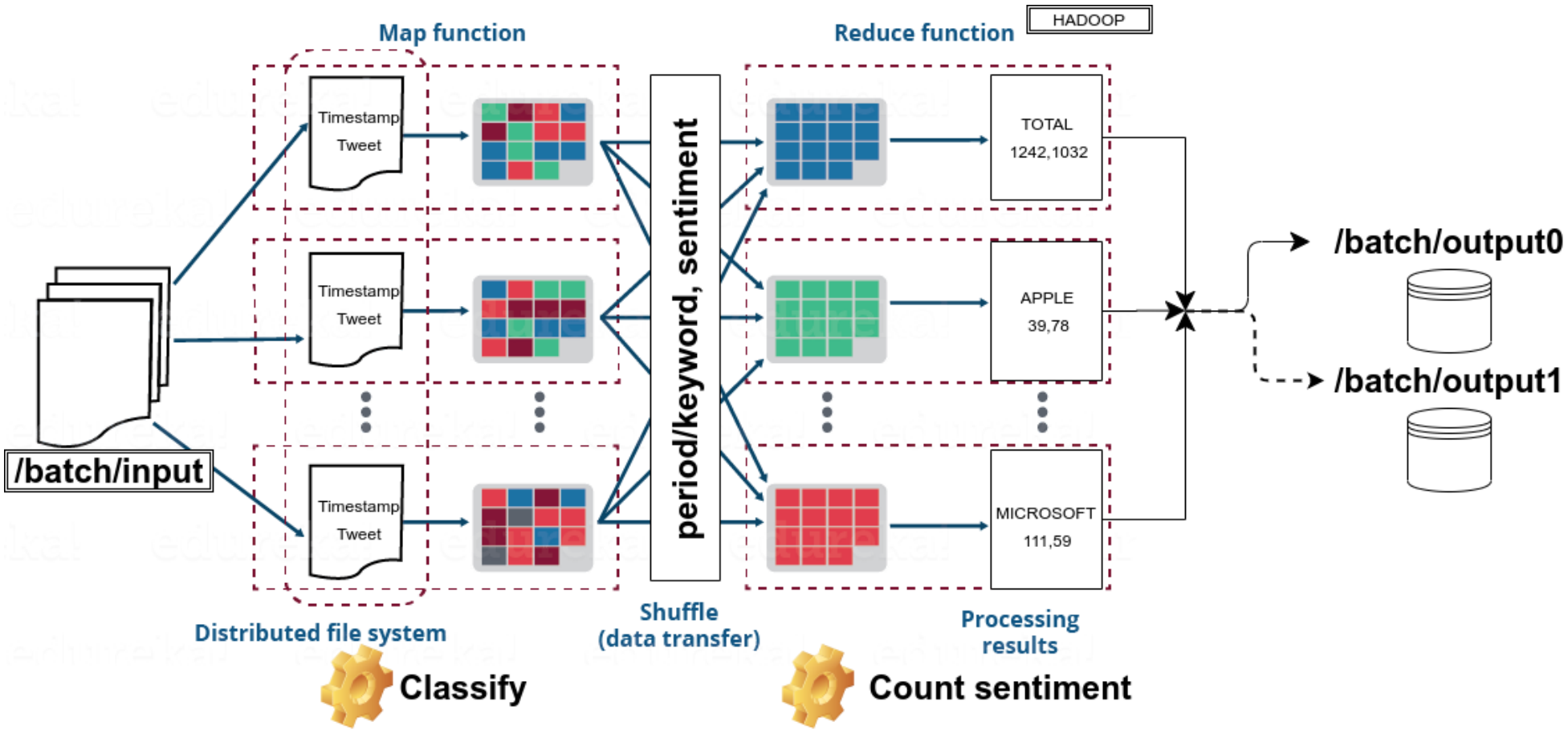
Tweets Generation

The Generator is a continuously-running process that simulates the emission of tweets by users.

- Tweets are sampled from sentiment140
- The generator associates each generated tweet with the current timestamp
- Tweet's timestamp and text are concatenated
- Concatenation is written into `/batch/input` and `/speed/input` HDFS folders

The parameters *maxFileDim* and *maxDeltaTime* control the buffer dimension and the max waiting time before write, respectively.

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

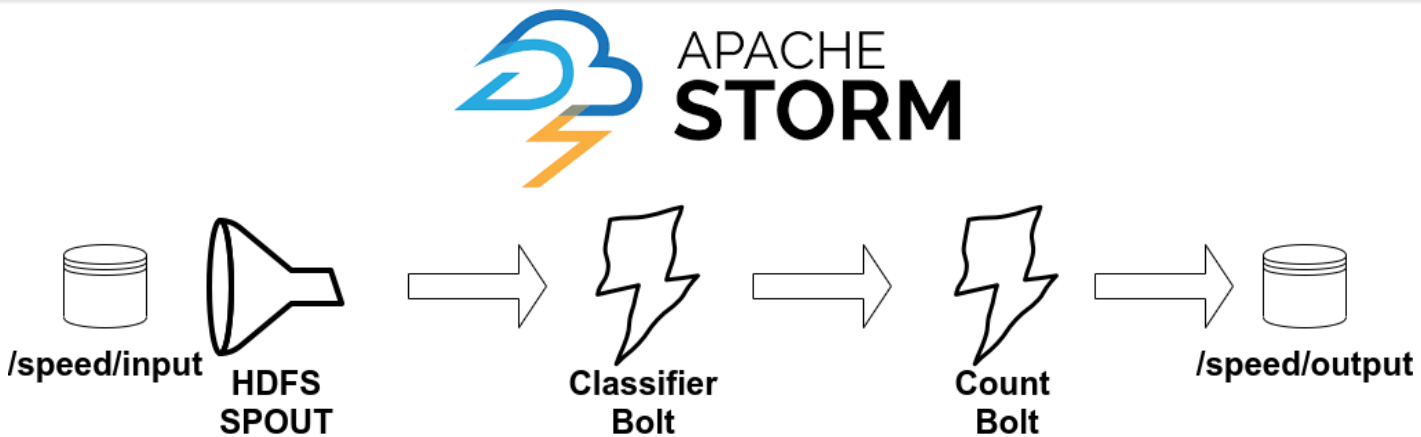
Algorithm 2: Reducer

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)
```

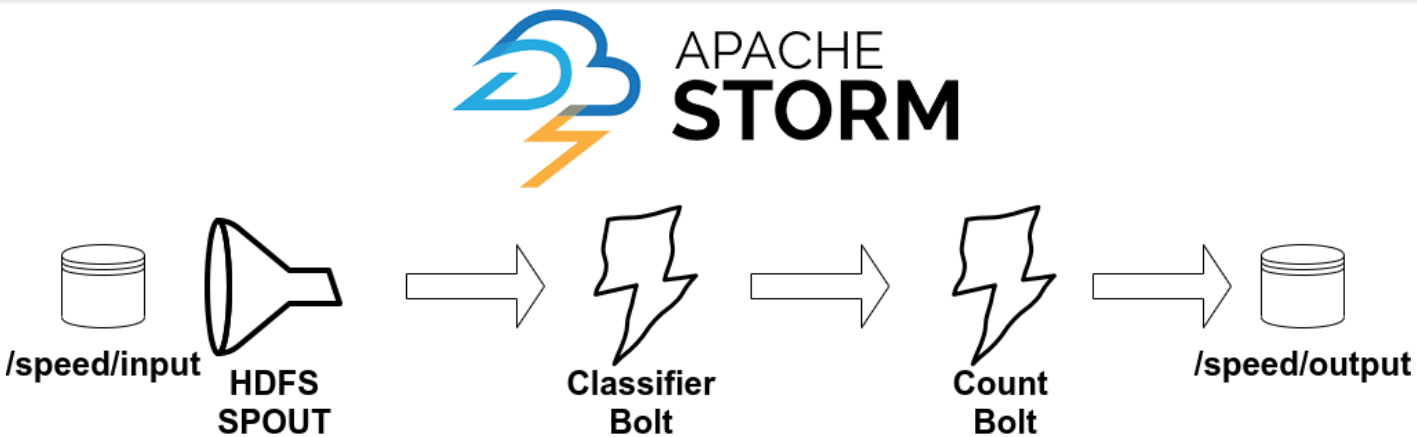
Speed Layer

- Speed layer uses a HdfsSpout for read the data
- It works at the same way of batch layer but with recent data
- Classifier Bolt is analogous to Mapper, Count Bolt is analogous to Reducer



Speed Layer

- Speed layer uses a HdfsSpout for read the data
- It works at the same way of batch layer but with recent data
- Classifier Bolt is analogous to Mapper, Count Bolt is analogous to 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

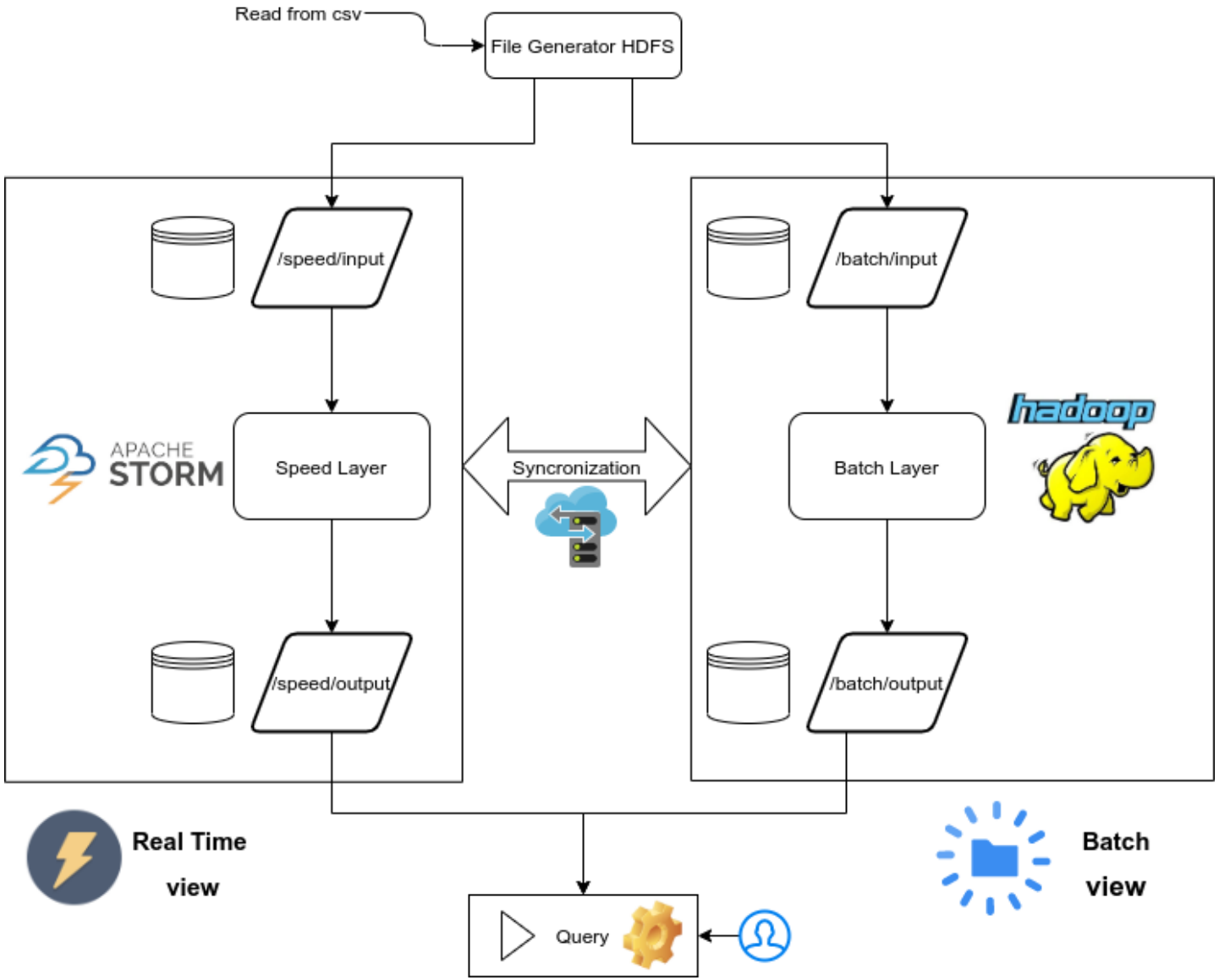
Batch layer

- 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

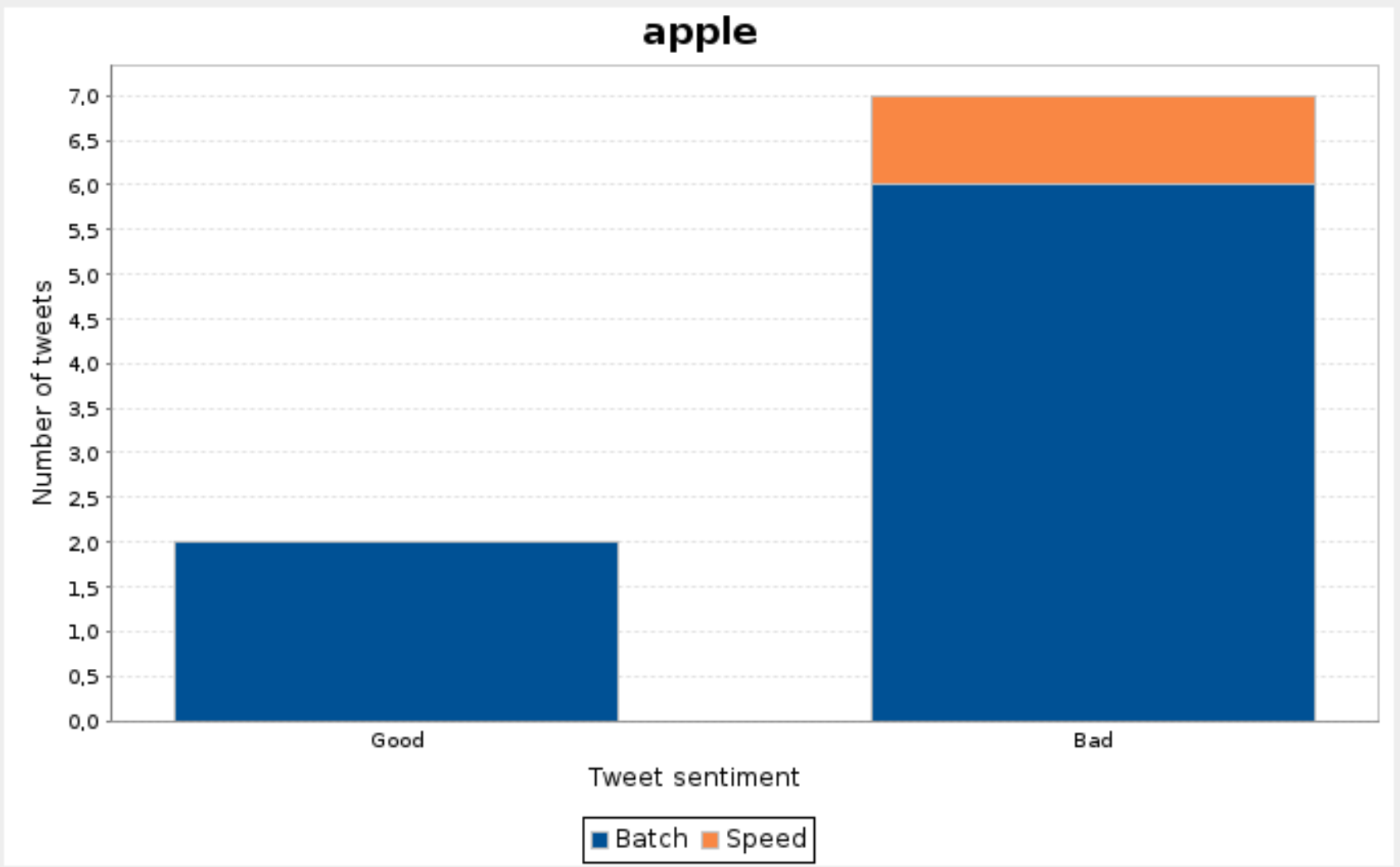
Complete architecture



The Serving Layer reads Batch and Speed views.
Results are provided via a GUI.

- The GUI allows you to specify a **time interval** by entering the start and end date
- The sentiment analysis considers the tweets sent in the chosen interval
- A **real-time summary** is provided for all tweets and for specified keywords

Stacked bar plot



Graphic User Interface

End

Thanks for the attention