### QualiMaster
A configurable real-time Data Processing Infrastructure mastering autonomous Quality Adaptation

Grant Agreement No. 619525

#### Deliverable 2.3

<table>
<thead>
<tr>
<th><strong>Work-package</strong></th>
<th>WP2: Scalable, Quality-aware Data Processing Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deliverable</strong></td>
<td>2.3: Scalable, Quality-aware Data Processing Algorithms V2</td>
</tr>
<tr>
<td><strong>Deliverable Leader</strong></td>
<td>LUH</td>
</tr>
<tr>
<td><strong>Quality Assessor</strong></td>
<td>Apostolos Dollas</td>
</tr>
<tr>
<td><strong>Estimation of PM spent</strong></td>
<td>28</td>
</tr>
<tr>
<td><strong>Dissemination level</strong></td>
<td>PU</td>
</tr>
<tr>
<td><strong>Delivery date in Annex I</strong></td>
<td>31.07.2016</td>
</tr>
<tr>
<td><strong>Actual delivery date</strong></td>
<td>01.08.2016</td>
</tr>
<tr>
<td><strong>Revisions</strong></td>
<td>5</td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td>Final</td>
</tr>
<tr>
<td><strong>Keywords:</strong></td>
<td>Algorithms, Stream Processing, Evaluation</td>
</tr>
</tbody>
</table>
Disclaimer

This document contains material, which is under copyright of individual or several Quali-
Master consortium parties, and no copying or distributing, in any form or by any means,
is allowed without the prior written agreement of the owner of the property rights.

The commercial use of any information contained in this document may require a license
from the proprietor of that information.

Neither the QualiMaster consortium as a whole, nor individual parties of the QualiMaster
consortium warrant that the information contained in this document is suitable for use, nor
that the use of the information is free from risk, and accepts no liability for loss or damage
suffered by any person using this information.

This document reflects only the authors’ view. The European Community is not liable for
any use that may be made of the information contained herein.

© 2016 Participants in the QualiMaster Project
List of Authors

<table>
<thead>
<tr>
<th>Partner Acronym</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUH</td>
<td>Tuan Tran, Christoph Hube, Miroslav Shaltev, Tu Nguyen, Andrea Ceroni</td>
</tr>
<tr>
<td>SUH</td>
<td>Cui Qin, Holger Eichelberger</td>
</tr>
<tr>
<td>TSI</td>
<td>Ekaterini Ioannou, Minos Garofalakis, Apostolos Nydriotis, Nick Pavlakis</td>
</tr>
</tbody>
</table>
Contents

List of Authors ........................................................................... 3

Contents ...................................................................................... 5

Executive Summary ..................................................................... 7

1 Introduction .............................................................................. 8
   1.1 Relation to other Work packages and Deliverables ................. 8
   1.2 Outline of the Deliverable .................................................. 10

2 Components for Processing Financial Data Streams ............... 11
   2.1 Mutual Information .......................................................... 11
      2.1.1 Implementation in the QualiMaster Infrastructure ............. 11
      2.1.2 Current Development ................................................ 12
   2.2 Transfer Entropy .............................................................. 13
      2.2.1 Implementation in the QualiMaster Infrastructure ............. 13
   2.3 Dynamic “Hub” computation ............................................. 14
      2.3.1 Implementation in QualiMaster infrastructure ................. 15
   2.4 Distributed similarity indexing ........................................ 15
   2.5 Querying stock correlation graph ..................................... 16
      2.5.1 Modeling Dynamic Stock Correlation Graphs ................ 16
      2.5.2 Implementation in the QualiMaster Infrastructure ............. 18

3 Components for Social Media Analysis .................................. 20
   3.1 Event Detection in Social Media ....................................... 20
      3.1.1 Improvements and Changes within the Event Detection Family 21
      3.1.2 Evaluation .............................................................. 22
   3.2 Mining, Analyzing and Detecting Hashtags carrying Sentiment Information 25
      3.2.1 Related Work .......................................................... 25
      3.2.2 Data Set ................................................................. 26
3.2.3 Approach ................................................................. 26
3.3 Twitter Pre-filtering Component ................................. 28
   3.3.1 Spam Detection .................................................. 29

4 Handling Historical Data ................................. 31
   4.1 Concepts for a Replay Mechanism ...................... 32
   4.2 Architecture of The Replay Mechanism ................ 33

5 Conclusions and Future Work ............................... 36

References ................................................................. 37
Executive summary

This deliverable is the third deliverable of WP2. It reports on the progress of WP2 within the second half of the QualiMaster project. The deliverable describes the progress in developing and evaluating an extended and improved set of algorithmic components for real time processing and analysis of data streams within processing pipelines. Taking the QualiMaster perspective, we are talking about the building blocks for the so called “pipeline playground”, where the expert will be able to choose and combine the algorithms into processing pipelines for financial risk analysis.

The selection of the algorithms for the components described in the previous and in this deliverable was done based on requirements collected in WP1 (Requirement Analysis and Use Case Definitions), which were derived in collaboration with industrial members of the consortium. The selection of algorithms was further refined by the definition of further useful and innovative processing pipelines in close collaboration with the industrial partners, which will drive the QualimAster Applications build on top of the QualiMaster infrastructure. Those pipelines have driven the development of algorithms and the integration of the algorithms into the QualiMaster infrastructure. In the third year of the project special attention has been put on focussing the project effort. Therefore, useful algorithms developed in the first two years of the project have been further improved instead of developing a large number of new algorithms.
1 Introduction

In this Deliverable we describe our activities in the design, development and evaluation of algorithms for the processing of financial and social media streams. This report is only part of the deliverable, which also includes the implemented components.

This deliverable contains an extended and revised set of components based on the components described in D2.2. It includes algorithms for advanced correlation analysis, capturing and querying of dynamic graphs, sentiment analysis, event detection, spam detection as well as components for the re-inspection of past processing results.

The functionality based selection of the algorithm families is derived from the use cases and user requirements defined in WP1 (as documented in D1.1 and D1.2) and from the definition of the QualiMaster Applications as defined in D6.1 and further extended in deliverable D6.3. However, the decision on the specific implementation is mainly based on whether the implementation of the algorithms can scale up with the volume and arrival rate of data in real-time.

The deliverable focuses on the presentation of the algorithms with discussions/evaluations of their quality aspect. A complete experimental evaluation, including the required execution time and scalability, of the processing elements as well as related pipelines will be included in the upcoming deliverable D5.4 and D3.3.

We describe domain specific characteristics of the data processing and evaluate those components with respect to efficiency, quality, and scalability.

1.1 Relation to other Work packages and Deliverables

For the selection and implementation of the components, we base on requirements identified in WP1 and described in deliverables D1.1 and D1.2. We furthermore, closely interacted with WP 6 for aligning the developed components for the processing of financial data streams as well as of social media streams with the consumption of the processing results in the QualiMaster applications. This includes agreements on functionalities, interfaces for result transfer and the discussion of adequate interactive visualizations and options for user-driven adaptations. This interaction is fostered by the joint definition of prioritized processing pipelines including the Focus Pipeline and the Time travel Pipeline. The collaboration with WP3 has been continued for creating algorithm families with hardware and software based implementations of the same functionalities. Some of the considered algorithms have been translated to reconfigurable hardware WP3, which are described in D3.3. Implemented components follow adaptable schema developed in WP4 and described in D4.2 and D4.3. Furthermore, there was a close interaction between the activities for Event detection in WP2 and for Event Prediction in WP4, which are used to discover complementing types of events. Finally, the deployment, integration and evaluation of the developed algorithms are being performed on the QualiMaster infrastructure that is provided by WP5 and described in deliverables D5.1, D5.2, and D5.3.
Addressing Comments from Second Year Review

The activities of WP2 have received some recommendations in the second QualiMaster project Review in February 2016. Most of the reactions to the recommendations are discussed in detail in this deliverable (and in deliverable D3.3 and D4.3) in the context of the respective components. The reactions can be summarized as follows:

- Work between WP2, WP3 and WP4 is now better coordinated by the definition of prioritized processing pipelines, which bring functionalities of all three WPs together. This has proven very helpful in improving coordination and integration into the QualiMaster infrastructure.
  This reacts to the reviewers’ comment: "We strongly recommend ensuring a closer coordination between WP2 and WP3/4 in year 3 of the project."

- Work on event prediction and event detection is (and was) closely coordinated between WP2 and WP4, providing complementing functionality for different use cases. This issue is discussed in more detail in Section 3.1 of this deliverable and in deliverable D4.3.
  This reacts to the reviewers’ comment: "A better coordination between the event detection/prediction activities of this WP and the event detection/prediction activities of WP2 should be ensured."

- The development of processing elements in WP2 and WP3 are frequently discussed together between the teams of WP2 and WP3 at TSI. As our goal is to introduce software- and hardware-implementations for processing elements of the same algorithms, we align existing work on the particular domain as well as the actual techniques developed for the infrastructure and implement common baseline algorithms.
  This reacts to the reviewers’ comment: "WP2 and WP3/WP4 have not been sufficiently coordinated in the second year. ... For instance, both WP2 and WP3 have to deal with a software implementation of Transfer Entropy, but they are working in parallel and are currently at different stages of development/testing."

- The documentation of the follow-up of works started in an early phase of the project have been improved including for example the work on hashtag based sentiment detection and data-stream pre-processing. Furthermore, more components have been and will be integrated into the QualiMaster Infrastructure.
  This reacts to the reviewers’ comment: "The goals of this WP have been slightly volatile. For instance, the activities related to some components have been discontinued (e.g. sentiment analysis-opinionated hashtags) and other new components (e.g. TRBC) are still not integrated into the platform. The next few months should mainly focus on finalizing and integrating current results, rather than keep exploring new opportunities."

- The evaluation of event detection has been extended to lager data sets and is still further extended. Final results will be reported in D2.4.
  This reacts to the reviewers’ comment: "Evaluation of event detection should be further improved."
For the evaluation of the streaming networks in social media, we are currently exploring feasible ways to perform this evaluation. Preliminary experiments (with a game-like environment) have shown that the manual evaluation of dynamic relationship is coupled with a high effort for look up of information even for experts. This reacts to the reviewers’ comment: “Regarding streaming networks in social media, we expect to see the study related to extracted networks as pre-viewed in D2.2. at the end of the project.”

1.2 Outline of the Deliverable

The rest of this deliverable is structure as follows: The deliverable again covers algorithms for processing financial data and algorithms for analyzing social media data streams. These two groups of algorithms are covered in the Section 2 (Components for Processing Financial Data Streams) and Section 3 (Components for Social Web Data Analysis), respectively. In addition, the deliverable also presents a generic component, which is relevant for the processing of all types of data streams, the Replay mechanism (see Section 4). It takes up the need of financial stakeholders of inspecting past development for better understanding current developments. Finally, Section 5 summarizes and concludes this deliverable with insights on future work towards the next deliverable, which is due in December 2016.
2 Components for Processing Financial Data Streams

In this section, we provide and discuss our progress with respect to the development of processing elements for analyzing data streams from the financial domain. More specifically, we describe the status of processing elements for: analysing mutual information (Section 2.1) and transfer entropy (Section 2.2); for dynamic “hub” computation (Section 2.3); for distributed similarity indexing (Section 2.4); and for querying the stock correlation graph (Section 2.5). This section mainly relates to Task 2.2 (Data processing algorithms and data structures for the financial domain) as described in the Description of Work, while the parts for quality estimation and evaluation refer to Task 2.4 and Task 2.5 respectively. Further evaluation results (Task 2.4 and 2.5) for the financial processing elements will be reported in D5.4, since evaluation will be performed in the context of the QualiMaster Infrastructure.

2.1 Mutual Information

Mutual information is a useful metric in time-series analysis as it reveals pair-wise dependencies (both linear and non-linear) between two data vectors. It measures the amount of information that is shared between the two vectors, essentially showing how much knowing one of them reduces uncertainty about the other. Adding the computation of mutual information as a processing element advances the understanding of correlation between two marketplayers by creating a clearer notion of the dependency.

Formally, mutual information of two discrete random variables $X$ and $Y$ is defined as:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p(x) p(y)} \right),$$

where $p(x, y)$ is the joint probability distribution function of $X$ and $Y$, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of $X$ and $Y$ respectively. Note that Mutual Information is a symmetric measure, which means that meaning $I(X; Y) = I(Y; X)$.

More theoretical background related to Mutual information can be found in [1].

2.1.1 Implementation in the QualiMaster Infrastructure

The initial Mutual Information implementation in the QualiMaster infrastructure consisted of partitioning the monitoring of market player pairs among all available nodes, updating the probability density functions (PDFs) in a streaming fashion (upon each arrival) and periodically calculating the Mutual Information index for each pair. The PDF approximation is performed by using histograms and counting the values that “fall” in each bin. The
Mutual Information complexity is quadratic in respect to the number of histogram bins used. More details regarding this approach can be found in deliverable D2.2.

2.1.2 Current Development

Our initial approach, described above and in deliverable D2.2, has a shortcoming in the frequency of computing the Mutual Information index. This index is only calculated at fixed intervals, in order to avoid "paying" a quadratic cost to calculate the Mutual Information for every update in the PDF. Yet, this fact implies that the estimated output value of Mutual Information is delayed to the end of every interval. This observation has inspired an improved streaming version of the existing Mutual Information estimator, that actually updates and outputs the new value of Mutual Information for each input tuple, yet avoiding the quadratic cost per update: Per update this improved approach is only requiring ("paying") a linear cost (with respect to the number of histogram bins).

Using the assumption that the approximation technique for the probability density functions is performed through histogram binning, the streaming version tries to isolate those terms from the original calculation that do not get affected by the new value. Hence, those terms do not need to be included in the calculations. This way, Mutual Information is being monitored in an incremental way (incorporating changes induced by new data, and eliminating influence from old data), instead of being re-constructed from scratch each time.

The main observation underlying this approach is that

\[ p(i) = \frac{C(i)}{N_i}, \]

where \( p(i) \) is the probability density function for vector \( I \), \( C(i) \) is the count of values that end up in a specific cell, and \( N_i \) is the total number of samples observed from vector \( I \). Hence,

\[ p(x) = \frac{C(x)}{N_x}, \quad p(y) = \frac{C(y)}{N_y}, \quad \text{and} \]

\[ p(x, y) = \frac{C(x, y)}{N_{xy}}. \]

Using this observation and some calculations, the Mutual Information can be computed as follows:

\[
I(X; Y) = \frac{1}{N_{xy}} \sum_{y \in Y} \sum_{x \in X} C(x, y) \log \left( \frac{C(x, y)}{C(x) C(y)} \right) - \frac{\log N_{xy}}{O(1)} \sum_{y \in Y} \sum_{x \in X} C(x, y)
\]

\[ \text{O(n) (1 row and 1 column)} \]

\[ \text{O(1) (=previous value+1)} \]

By keeping simple sum statistics in a streaming manner, we only need to update one row and one column \((\sum_{y \in Y} \text{ for given } x \text{ and } \sum_{x \in X} \text{ for given } y)\), instead of the entire 2D matrix. Hence, it is possible to update the Mutual Information index for every input tuple and to
send it to the output as soon as it is updated. This is possible, because each update costs only $O(n)$ time.

This improved approach is a better fit for QualiMaster’s purposes, as it provides the changes immediately to the stakeholder applications coming closer to real-time analysis results. Thus, it is preferred over the first approach and is chosen as the default implementation for Mutual Information in the QualiMaster infrastructure.

### 2.2 Transfer Entropy

Although Mutual Information provides a very useful measure of information sharing between two time series, it is a symmetric metric and, thus, does not provide any directional information. In the financial domain, it is usually very useful to be able to determine not only the amount of shared information between the two vectors, but also the direction of influence, i.e., which one of the two “drives” the other. For example, if there are two market players, namely $A$ and $B$, it is important to know if $A$ is correlated to $B$. However, it is also important (maybe even more important) to know that, for the sake of argument, $A$ affects $B$. This may mean that if $A$’s price rises, then $B$’s price will also rise, whereas the opposite might not hold (i.e., “when $B$’s price rises, $A$’s price will rise too”).

Transfer Entropy is an asymmetrical information transfer measure that can provide the required directivity. Unfortunately, Transfer Entropy is quite demanding with respect to computation resources since it requires a cubic ($O(n^3)$) memory and time complexity. Formally, Transfer Entropy between two discrete random variables $X$ and $Y$ (from $Y$ to $X$) can be defined as:

$$TE_{Y\rightarrow X} = \sum_{x_{n+1}\in X} \sum_{x_n\in X} \sum_{y_n\in Y} p(x_{n+1}, x_n, y_n) \log \left( \frac{p(x_{n+1}|x_n, y_n)}{p(x_{n+1}|x_n)} \right),$$

where $p(x_{n+1}, x_n, y_n)$ is the joint probability, and $p(x_{n+1}|x_n, y_n)$, $p(x_{n+1}|x_n)$ are conditional probabilities.

As in Mutual Information, the main challenge lies in maintaining the probability functions. In the case of histogram approximation, the complexity is related to the number of bins used (as in Mutual Information). For example, if 1000 bins are chosen as the size of the histogram, then the joint probability of a single pair amount to $4Gb$ of memory ($1000^3 * 4$ (number of int for the count of each cell)).

### 2.2.1 Implementation in the QualiMaster Infrastructure

A first version of the Transfer Entropy (TE) estimator has been implemented, in order to be included in the QualiMaster infrastructure. Similar to Mutual Information, the implementation of Transfer Entropy updates the required probability density functions (PDFs) in an on-line fashion and uses them to calculated the TE value for each pair of streams.
It overcomes the memory limitations using a simple observation related to the live input financial data: Market players show a “locality” related to their value movements, which in turn leads to very sparse PDFs (less than 1% of the PDF cells contain any values). This observation allows the implementation to maintain the PDFs in HashMaps instead of using the large matrices, which, due to the sparseness of the data, greatly reduces memory usage and allows for on-line monitoring of the TE value.

### 2.3 Dynamic “Hub” computation

The processing elements for the dynamic “hub” computation is responsible for detecting the most important market players (i.e., market players that have the most influence to others) in an on-line, streaming fashion. This enables financial stakeholders to get a better overview in a possibly large correlation graph and to better understand the structure of this graph.

The implemented approach realizes an algorithm we call DSPM (short for Distributed Stochastic PageRank Maintenance). DSPM maintains a dynamic graph in which the nodes are the market players and the edges are the correlations between the players. Our process tries to estimate the true PageRank value of the nodes (which is normally calculated using the Power Iteration method) by using a Monte Carlo approach involving random walks.

More specifically, it consists of an improvement over an existing Monte Carlo algorithm based on the idea of random walks, called FIP (short for Fast Incremental PageRank). The key idea of FIP is to use reservoir sampling over all stored random walks in order to re-distribute the random walks evenly across the new and old links of the graph. Precisely, for a newly observed edge \((u, v)\) (i.e., node \(u\) now also links to node \(v\)), each of the random walks passing through \(u\) is considered for redirection with a probability of \(\frac{1}{\text{outDeg}(u)}\), with \(\text{outDeg}(u)\) denoting the new out-degree of \(u\). All selected random walks are redirected through \((u, v)\) and the visit counts of the affected nodes are updated accordingly. This is done for all steps in the revoked segment of the random walk. Visit counts are decreased by one, and for all steps in the new segment they are increased by one. It can be shown that the expected visit counts of the nodes maintained by FIP are proportional to the true PageRank scores.

However, FIP has two substantial limitations

- First, it requires that all random walks are stored in order to be able to revert their effect in case of redirection. This requirement can be addressed by a shared memory model, which however severely limits the scalability of the algorithm.

- Second, the algorithm assumes that all nodes in the graph are available during the initialization of the algorithm; only the links between the nodes can change.

DSPM lifts these limitations by only maintaining sampled and aggregate information for the random walks instead of keeping a full trace of them. As a result, the algorithm is
substantially more scalable than FIP, in terms of both memory and network. In DSPM, the nodes of the graph are partitioned to the available servers via consistent hashing. The algorithm maintains the visit counts according to the crawled updates. Since the trace of random walks is not maintained, the algorithm cannot execute the cancellation actions that are necessary for random walk redirections (as in FIP). Instead, updates are handled by initiating a combination of normal and negative random walks. Negative random walks are constructed just like normal random walks, but they decrease by one the counts of the visited nodes, instead of increasing them. It can be shown that the expected visit count at each node after all pending random walks are completed will be equal to the corresponding expected visit count of FIP (and thus also proportional to the PageRank of the node).

2.3.1 Implementation in QualiMaster infrastructure

The particular processing element has already been developed and incorporated into the QualiMaster infrastructure. Currently, it participates in the Dynamic Graph pipeline (Figure 1). This pipeline is responsible for calculating correlations among market players, then using those correlation to create a dynamic graph (where the market players are the nodes, and if there correlation is above a certain threshold, then there is an edge between them) and finally feeding this graph into the Dynamic “hub” computation component to detect the most important market players. A list of the computed most important market players (in descending order of importance) of variable size is created and sent to the client applications. The size of the list can be controlled by the client application, where the marketplayers in the list can be visualized in an adequate way, e.g. by making each node's size reflect its importance.

2.4 Distributed similarity indexing

T-Storm, i.e., our distributed similarity index, was introduced to provide a computational framework that efficiently monitors pairwise correlations among thousands of data streams.
using an approximation technique, a distance measure, and an index to efficiently store the data. Inspired by StatStream [2], T-Storm extends StatStream’s functionality by lifting some of its limitations. In particular, instead of StatStream’s Grid Structure index, which is highly dependent on the Euclidean distance, T-Storm uses a generic and widely used index, that is both efficient and (tunably) accurate for similarity search, namely the LSH index. As a result, T-Storm is able to provide the user with the possibility of using different distance measures that may fit better to their dataset (e.g., cosine, Hamming, etc.), or even allow the user to define their own distance measure. StatStream’s use of DFT coefficients as an approximation technique is also tied to the Euclidean distance. In T-Storm, the approximation technique that can be used is also distance-independent. More details regarding StatStream/T-Storm can be found in the D2.2 deliverable.

2.5 Querying stock correlation graph

Stock correlation graph are a powerful mean for observing, analyzing and predicting stock market dynamics in financial applications. As an example, consider the results generated by the correlation matrix estimator (described in the D2.2 deliverable). The graph nodes are the monitored stock markets and the edges are the detected correlations. Data dynamics cause modifications on the edge values as well as the structure of the graph. These modifications in turn, produce a sequence of graph snapshots. For example, Figure 2.a shows a sample from the data maintained by a financial application over the time. Such developments can be captured by a sequence of graph snapshots. In part b of Figure 2, we see how the data coming from the financial data stream modifies the correlation graph captured by a series of graph snapshots.

A stock correlation graph is used for measuring financial risk and, especially its potential propagation. Systemic risk as an extreme form of financial risk is described as the risk of collapse of a major part of a financial system given the failure of a single, or few, markets. One of the most vital aspect of the proposed measures for systemic risk is that the interconnectedness of financial markets increases the probability of failure contagion. Thus, applying systemic risk measures requires being able to provide node-related, egocentric, analytics over interconnections.

2.5.1 Modeling Dynamic Stock Correlation Graphs

To formally model such volatile graphs, we assume the existence of an infinite set of identifiers \( V \), attributes \( A \), atomic values \( D \), and inner-relationships \( E \). Each instance provides a value \( d_i \) for each attribute \( a_i \) of the corresponding relation. The value \( d_i \) can be either atomic value, time information, or identifier, i.e., \( d_i \in D \cup \mathbb{Z} \cup V \).

A relation \( R \) is a tuple with \( k \) attributes \( \langle a_1, \ldots, a_k \rangle, a_i \in A \). An instance \( r \) is a tuple \( \langle n, t, d_1, \ldots, d_k \rangle \), where \( n \in V \) corresponds to the instance's identifier, \( d_i \in D \) to the attribute values for the particular attribute \( a_i \), and \( t \in \mathbb{Z} \) the time these values appeared for describing the
<table>
<thead>
<tr>
<th>time</th>
<th>stock values</th>
<th>Modifications</th>
<th>correlations</th>
</tr>
</thead>
<tbody>
<tr>
<td>before 1 Oct</td>
<td>(g_0 m_0 y_0 h_0 i_0 a_0)</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>1 Oct</td>
<td>(g_1) (e.g. provider=&quot;NasdaqGS&quot;); (y_1) (e.g., stocks=200)</td>
<td>(\oplus(g, y); \oplus(g, m))</td>
<td></td>
</tr>
<tr>
<td>5 Oct</td>
<td>(y_2; a_1; h_1; m_1)</td>
<td>(\oplus(m, y); \oplus(g, h); \oplus(h, y); \otimes(g, m))</td>
<td></td>
</tr>
<tr>
<td>10 Oct</td>
<td>(\oplus f_1; g_2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 Oct</td>
<td>(\oplus t_1; a_2; i_1)</td>
<td>(\oplus(a, m); \oplus(a, f); \otimes(m, y); \oplus(g, m); \oplus(t, y); \otimes(h, y))</td>
<td></td>
</tr>
<tr>
<td>20 Oct</td>
<td>(g_3; f_2)</td>
<td>(\otimes(g, h); \oplus(g, i); \oplus(f, m))</td>
<td></td>
</tr>
<tr>
<td>25 Oct</td>
<td>(m_2; a_3)</td>
<td>(\oplus(a, g); \otimes(g, m))</td>
<td></td>
</tr>
<tr>
<td>26 Oct</td>
<td>(\otimes f)</td>
<td>(\otimes(f, \ast))</td>
<td></td>
</tr>
<tr>
<td>30 Oct</td>
<td>(t_2; i_2; h_2)</td>
<td>(\oplus(a, t); \oplus(m, t))</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: The evolution of the data maintained by a stock application (listed in fig-a) cause modifications on the nodes as well as the structure of the corresponding stock correlation graph (some of the snapshots are shown in fig-b).

instance with identifier \(n\).

\(R\) may contain various instances with the same identifier but different appearance time, i.e., \(r_i, r_j, \ldots, r_k\) with \(r_i.n=r_j.n=\ldots=r_k.n\) and \(r_i.t<r_j.t<\ldots<r_k.t\). Each relation \(r_i\) that appeared at \(r_i.t\) is valid until another relation with the same identifier appears, i.e., until the appearance of \(r_j\) with \(r_i.n=r_j.n\) and \(r_j.t>r_i.t\).

We consider that the given instances form undirected graphs and we thus need to deal with the evolution of graphs. Formally, we assume that the identifiers in \(V\) correspond to the nodes composing the graphs and therefore the instances in \(R\) provide the information encoded in the nodes. The graph edges correspond to the inner-relationships included in \(E\).

Let \(V_t\) denote all the instances with time equal to \(t\), i.e., \(V_t=\{r_i.n \mid r_i \in R \& r_i.t=t\}\). A graph \(G_t\), also called snapshot, denotes the graph occurring at time \(t\) using \(V_t\). \(G_t\) is given by tuple \(\langle t, V_t, E_t \rangle\), where \(V_t\) provides the nodes with their values and \(E_t \subseteq V_t \times V_t\) the edges.

Given a particular \(V\), we have a sequence of snapshots, i.e., \(G_t, G_{t+1}, \ldots, G_{t+i}, \ldots\). The snapshots illustrate the evolution as time increases, i.e., \(t, t+1, \ldots, t+1+i, \ldots\). We consider the following modifications between two sequential snapshots: (i) addition of a node, denoted as \(\oplus n\); (ii) deletion of a node, denoted as \(\otimes n\); (iii) addition of an edge,
denoted as $\oplus\langle n_i, n_j \rangle$; (iv) deletion of an edge, denoted as $\otimes\langle n_i, n_j \rangle$; and (v) assignment of a new instance for a node, e.g., instance $r_{i+1}$ instead of $r_i$.

Our goal is to efficiently execute queries over the snapshots of the evolving graphs.

For making the querying functionality and the snapshots usable for the QualiMaster stakeholder applications, we have jointly defined a Time-travel pipeline that is shown in Figure 3. The upper part of the pipeline provides analytics over the graph that is dynamically compiled from the correlated financial data. The middle part is responsible for storing snapshots of the graph and answering time-travel queries over these snapshots – a possible query here would be "retrieve the shortest path between node $n_a$ and node $n_b$ from time $t_1$ to time $t_2$". Finally, the lower part provides time-travelling features. For example a user would be able to replay the evolution of the graph and display it in a film-like fashion.

2.5.2 Implementation in the QualiMaster Infrastructure

To enable efficient query processing over the dynamic stock correlation graph, we are currently developing a novel index structure. Our index structure decomposes the data involved in the volatile graph into separate elements that are able to encode both time information as well as the association of nodes with related node-versions and neighboring nodes. The ultimate goal is to reduce the space for storing the information related to the volatile graph and the time needed for processing this information. Our index $I$ is composed by an inverted list $I-N$ for efficiently retrieving node-versions and an interval skip list $I-E$ for detecting edge information of a particular time or period.
The inverted list $I-N$ is a set of tuples, each of which represents a graph node version. The tuple stores information related to the appearance time of the node, updates of the node’s values, as well as modifications of the edges between the node and other nodes. The skip list $I-E$ on the other hand, holds elements that correspond to times in which we notice modifications of the graph edges. Using $I-E$ we can quickly identify which edges are present at which periods.

Combining these two, we get a powerful index that allows us to perform basic and complex operations. For example, using $I-N$ we can rapidly retrieve the period $[t_s, t_e]$ in which a node/node-version is present. Using $I-E$ we can retrieve edges for a period $[t_s, t_e]$. Using both $I-N$ and $I-E$ we can support more complex operations as retrieving snapshots for a time $t_r$ or a period $[t_s, t_e]$, finding shortest, earliest and continuous paths within a period $[t_s, t_e]$ or answering reachability queries within a period of time.

To retrieve a snapshot of the graph for a time $t_r$, we use the $I-N$ to retrieve the nodes that were present before the time $t_r$ along with their version at $t_r$. Then we use $I-E$ to retrieve the edges that were present at $t_r$ and that completes our snapshot. The other option we support is to retrieve a set of snapshots for a period $[t_s, t_e]$. To achieve this, we retrieve a snapshot for the time $t_s$ and then we use basic operations to retrieve nodes and edges that were present at the period $[t_s, t_e]$. The output can either be a set of snapshots (the different graphs that each modification produces) or a snapshot at $t_s$ and a list of deltas, representing the modifications for the specific period.

Processing path queries aims at retrieving various kinds of paths –shortest, earliest, continuous– as well as answering reachability queries within a time period. For example, a path query would request the shortest path between $n_a$ and $n_b$ from time $t_1$ to time $t_2$. The processing is based on the traditional breadth-first search (BFS). We start at one of the end-nodes (e.g., $n_a$) of the path and move to their direct neighbour nodes until we reach the other end-node (e.g., $n_b$). Our algorithm tracks paths from the start node to the visited nodes and the time intervals during which the paths exist. The output is the path that satisfies the request criteria (e.g., shortest, earliest, etc.) and the time constraints of the request (e.g., $[t_s, t_e]$).

In the remaining time of this project, we will work on implementing the necessary query execution mechanisms in order to ensure the correct and efficient execution of the time-travel pipeline.
3 Components for Social Media Analysis

Social media streams and social media data in more general are the second type of data, which is analysed in the QualiMaster project in support of financial risk analysis. The underlying assumption is that social media analysis might provide further signals, e.g. on evolving events, which might be used as an additional input, when assessing financial risk. First methods for social media analysis had already been implemented in the first part of the QualiMaster project and reported in deliverable D2.2.

In this section, we report on the progress on implementing processing elements for analysing social media content. This includes (1) methods for event detection and their evaluation; (2) improved methods for sentiment classification; (3) methods for effective social media stream pre-filtering including spam detection. This section mainly relates to Task 2.3 (Data processing algorithms for Social Web data) as described in the Description of Work, while the parts for quality estimation and evaluation refer to Task 2.4 and Task 2.5 respectively. First results on method evaluation are reported here and will be further detailed in deliverable D2.4 with results on performance as part of the pipelines also being considered in the upcoming deliverable D5.4.

3.1 Event Detection in Social Media

Social media data, such as Twitter data, are an additional source of signals that are relevant for analysing financial risks in QualiMaster. Relevant events and their reflection in Social Media can have an impact on the financial situation either directly (e.g., disasters) or indirectly by influencing the reaction of actors in the financial market. Therefore we have implemented different detection algorithms for the detection of evolving events as well as bursts and other abnormalities within the incoming stream of social media data.

The algorithms that deal with event detection form an algorithm family. An example application of this family can be seen in D6.3, where the Event Detection family is included as a component in the Focus Pipeline, a Storm pipeline for financial risk analysis recommending stocks to re-focus the attention of the user. Within this pipeline the Event Detection component receives tweets from the preprocessing component and forwards any detected events to the keyword extraction component, where the events will be enriched with event describing keywords. Afterwards the Focus Recommendation component will aggregate events and forward them to the user pointing the user to market players involved in the respective events as candidates to include into the list of his/her observed market players.

Furthermore, the Event Detection family is closely related to the Event Prediction family, as introduced in D4.2, and work on both components is coordinated within the L3S team. However, Event Prediction uses the social media stream to predict events that might happen in the future, mainly focusing on events that are announced or expected (and therefore talked about in social media in combination with expected time points). In
contrast, the Event Detection component solely focuses on current and evolving events that can be discovered by observing frequencies. Such events might have an impact on one or several stocks. Event detection algorithms can still be used for the last step of content-based event prediction (see D4.3, Section 3.2.2.2).

In this section, we will first describe important improvements and changes within the Event Detection family compared to D2.2. Afterwards we will show the results of the evaluation of the new Hard Threshold algorithm and the improved Moving Average Based algorithm. Here, we have followed the reviewers’ recommendation to further improve the evaluation by including more market players and covering a larger time frame.

### 3.1.1 Improvements and Changes within the Event Detection Family

In order to meet the requirements of the system and to augment the benefits for the user we conducted some changes to the Event Detection family in general and, more specifically, to the Moving Average Based approach, which serves as the default algorithm within the family. In D2.2 we described an approach that was based on the number of tweets per day containing a fixed list of stocks as chosen by the user. The improved system now covers a much larger and dynamic number of stocks, in order to inform the user about the most important events associated with these stocks. At the same time the user is still able to filter the list of stocks to reduce the number of possible events displayed and to focus on selected market players.

In addition we adjusted the system so that it does not only show events for whole days but for finer granularities even without changing the size of the sliding window. We felt it might be important for users to be informed about an event immediately, since drastic changes in very short time are not uncommon in the financial market.

The frequency for checking for new events is now controlled by the parameter `frequencyOfCheckForEvent`. With this parameter the user can decide, after how many incoming tweets the algorithm should check for new events. The default value is 1, which means that there will be a check after every incoming tweet, not including tweets that have been dismissed in the preprocessing step. A higher value for `frequencyOfCheckForEvent` will lead to a delay, but will improve the efficiency. Usually the algorithms will be able to handle the number of incoming tweets in real-time, so a value of 1 is recommended. Furthermore, event aggregation is no longer a task of the Event Detection system and is handled by a separate component, in order to simplify keyword extraction. In the event aggregation step sequently detected events will be aggregated to one longer lasting event.

### Improved Moving Average Based Event Detection

In the Moving Average Based algorithm as introduced in D2.2 an event was detected when the number of relevant tweets for a stock within a time window exceeded a certain threshold which was computed based on a hard-coded default value (overall average) for
the stock and a percental threshold, which was defined by the user. For example, if the default value for the $AAPL stock was 1000 tweets in one time window and the percental threshold was set to 0.5, than an event was triggered when the number of tweets in one time window was at least equal to 1500.

In the improved approach, we keep the percental threshold. However, we now compute the default value in a dynamic way. The algorithm now offers a defaultAverage parameter to define the start value for all stocks and a includeNewMeasures parameter which defines to what percentage new measures/signals (i.e., new tweets) will alter the overall average value for a stock. The algorithm maintains a map for the overall averages of all observed stocks. The new values will be computed every time a check for events happens. The includeNewMeasures parameter has to be chosen with care. Too low values will lead to a situation where the overall average values for all stocks will stay close to the value defined by the defaultAverage parameter, while too high values will lead to fluctuation. This will make the detection of slowly emerging events difficult. We propose a value of 0.00001 for the includeNewMeasures parameter. The main advantage of the dynamic computation is that the algorithm is now able to adjust to systematic long-time frequency changes for a stock.

Hard Threshold Based Event Detection

With the Hard Threshold Based Event Detection algorithm we included a simple algorithm that is able to detect strong bursts in the tweet stream. While not every financial stock event is represented by a strong burst in the tweet stream, bursts seem to be a reliable way to discover the important stock events. The algorithm is a variation of the Moving Average Based approach. Just as the Moving Average approach it uses a sliding window and a frequencyOfCheckForEvent parameter. The two algorithms differ in the decision process for an event. Instead of a dynamic average and a percental threshold the Hard Threshold Based algorithm simply uses a pre-defined threshold value. Anytime the number of tweets mentioning a certain stock within the time window exceeds this value an event is triggered. The algorithm can either use one common threshold value for all stocks or a threshold value for every stock. The common threshold value will lead to a preference of bigger stocks that are better represented in the Twitter Stream.

3.1.2 Evaluation

In this section we will show the results of the evaluation of the adapted and improved Moving Average Based event detection and the new Hard Threshold Based event detection approaches. The evaluation was conducted on our crawled Twitter dataset covering Tweets from May 2015 until February 2016. The crawling included only tweets that contained at least one of the stock symbols in our list of stock symbols. Overall the dataset contains 9,549,000 tweets. Since we realized that the dataset contains a lot of duplicates in bursts (usually spam), we conducted a simple duplicate detection, removing all tweets
that are at least 70% similar to one of the last 10 tweets. We used the Levenshtein Edit distance for measuring the similarity between tweets.

We decided that an event should be both large enough (in number of tweets) and long enough in order to be of importance. An event that exceeds the threshold only for the time of a small amount of incoming tweets during one day might not be important enough to be considered. Therefore, for both algorithms, we included only events that have been triggered at least 1000 times during a single day. In the evaluation the frequencyOfCheckForEvent is 1, which means that all current events will be triggered after every approaching tweet that has not been removed during the pre-processing step. The aggregation of events happens in the Focus Recommendation component which is not part of this evaluation.

We systematically tried different parameters for both algorithms until the number of events found during the full time period seemed reasonable. In the end we decided for a 1 hour time window. For the Moving Average Based algorithm we used a percentageThreshold of 4.0, meaning that in order to trigger an event the number of tweets within the time window has to be at least 400% larger than the value of the dynamic average. We set the defaultAverage to 80 and the includeNewMeasures parameter to 0.00001. For the Hard Threshold algorithm we set the threshold to 350. Since we wanted to include only well known market players we decided that one threshold value will be enough instead of choosing a threshold for every market player separately.

We decided to include the following five stocks in the evaluation: $AAPL, $FB, $GOOGL, $NFLX, $YHOO. For both algorithms going through the complete dataset took less than 2 hours on a machine with 8 GB memory and 22.20 GHz CPU kernels. We gathered all detected events and sorted them by stock. Since it is hard to find a ground truth for financial stock events on Twitter we decided for a manual evaluation. Therefore, for each event, we conducted a Google query containing the name of the stock, the name of the company and the exact date of the event. For big companies like Apple and Google it is easy to find minor events on many days of the year, so we decided for three criterions in order for an event to be accepted:

1. The event has to be clearly mentioned among the top ten Google entries of the query.
2. The event has to be of clear significance for the whole company, not just for parts of the company or for a specific region.
3. The event has to take place on the exact date or at least one day later (because many events are reported by the news on the next day).

We also manually labeled if the events are scheduled (e.g. a quarterly report) or non-scheduled (e.g. a sudden drop of the stock value) and if the events are directly related to the financial domain (e.g. a stock split) or not directly related to the financial domain (e.g. "Zuckerberg announces birth of his first child"). Non-financial events might still have a strong impact on the stock value.
Tables 1 and 2 show the results for the evaluation of the Moving Average Based (MAB) algorithm and the Hard Threshold (HT) algorithm. The results show that the algorithms perform similarly well with both achieving an overall precision of 0.7 (39 out of 56 detected events are correct events for MAB and 37 out of 53 for HT). Both algorithms perform better at finding financial events compared to non-financial events which is no big surprise given the fact that we were only looking for stock symbols within tweets. Still both algorithms managed to find a small number of non-financial events. The MAB approach performs slightly better at detecting unscheduled events. The most significant difference between the results for both algorithms is the number of events and the precision when it comes to different stocks. For FB, GOOGL and NFLX both algorithms find approximately the same events while for AAPL and YHOO the results are very different. For the YHOO stock the MAB approach detects far more events (24 compared to only 3 with the HT approach) but with a much lower precision (0.54). For the AAPL stock it is the other way around where the MAB finds less events but with a much higher precision. This show that both algorithms perform well for a small number of events per stock but not as good for a high number of events. We assume that some of the peaks in the incoming stream are caused by noise (e.g. spam).

To address this issue we are including a spam detection component in the Focus Pipeline. Furthermore we will work closely together with WP4 to include a component for extraction of frequently co-occuring keywords and adaptive crawling of hashtags. This will help to improve the event detection in two ways: first, it will add further search terms to the source increasing the number of relevant tweets for each stock and, secondly, it will help to validate the correctness of detected events by creating a summarization of the event.

<table>
<thead>
<tr>
<th>Stock</th>
<th>#events</th>
<th>correct</th>
<th>precision</th>
<th>financial</th>
<th>non-financial</th>
<th>scheduled</th>
<th>non-scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAPL</td>
<td>7</td>
<td>6</td>
<td>0.86</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>FB</td>
<td>5</td>
<td>4</td>
<td>0.8</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>GOOGL</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>NFLX</td>
<td>15</td>
<td>11</td>
<td>0.73</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>YHOO</td>
<td>24</td>
<td>13</td>
<td>0.54</td>
<td>10</td>
<td>3</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>all</td>
<td>56</td>
<td>39</td>
<td>0.7</td>
<td>33</td>
<td>6</td>
<td>16</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 1: Evaluation results for Moving Average Based event detection.

<table>
<thead>
<tr>
<th>Stock</th>
<th>#events</th>
<th>correct</th>
<th>precision</th>
<th>financial</th>
<th>non-financial</th>
<th>scheduled</th>
<th>non-scheduled</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAPL</td>
<td>29</td>
<td>15</td>
<td>0.52</td>
<td>13</td>
<td>2</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>FB</td>
<td>5</td>
<td>4</td>
<td>0.8</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>GOOGL</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>NFLX</td>
<td>11</td>
<td>10</td>
<td>0.91</td>
<td>9</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>YHOO</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>all</td>
<td>53</td>
<td>37</td>
<td>0.7</td>
<td>33</td>
<td>4</td>
<td>20</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 2: Evaluation results for Hard Threshold event detection.
3.2 Mining, Analyzing and Detecting Hashtags carrying Sentiment Information

Sentiment Analysis (or Opinion Analysis, as mentioned in Task 2.3 in the Description of Work) is the area, which deals with the problem of assigning opinion labels (e.g., "positive" vs. "negative" vs "neutral") to various types of documents using diverse text-oriented and linguistic features. Nowadays, Twitter represents a major platform for sharing information and opinions about all sorts of topics and events. While a lot of work has focused on tweet-based sentiment assignment (a comprehensive state-of-the-art has been discussed in D2.1, Section 6), various approaches show major limitation mostly due to the ambiguity, the shortage in information that a 140 characters text can deliver, and the platform-specific language often used in Tweets. At the same time, most of the application scenarios nowadays show a need for per entity sentiment aggregation. Despite the large number of studies that focused on sentiment classification on various types of text, the potential of opinionated hashtags associated with the shared content still remains unexplored in the context of sentiment classification.

In the previous deliverable D2.2 (see section 3.2), we described a very preliminary study for analyzing the sentiment exposed in tweets on hashtag level instead of doing this on tweet level. In this preliminary study we have exploited results from tweet-level classification for classifying hashtags according to sentiment. In more detail, we have aggregated the sentiment score of tweets belonging to the same hashtag and have used the aggregated value in order to classify the hashtag.

Meanwhile, we have developed this approach further to make it more effective. The goal of this work is to improve the methods that aim at analyzing and detecting hashtags carrying sentiment information. In contrast to the previous work, we now represent the group of tweets belonging to the same hashtag as a single document which is considered as an atomic instance in training and testing.

In this section, after discussing related work and the data set used in the experiments, we present our extended approach for identifying hashtags carrying opinionated content. Finally, we perform a user annotation study in order to get first insights into the effectiveness of the method for identifying opinionated hashtags. More in depth experiments are currently under preparation.

3.2.1 Related Work

Davidov et al. [3] analyze the performance of classification models for predicting the "sentiment types" of Twitter messages, where the sentiment types are defined as a fixed set of hashtags and smileys. The experimental setup uses 50 Twitter hashtags, each hashtag containing a word suggesting an opinion (e.g., #happy, #crazy, #bored) and 15 mood indicating emoticons as sentiment labels. The final goal here was to predict (assign) previously unseen tweets to one of the hashtags or emoticons class. In another recent work
[4], the authors did not only consider hashtags containing sentiment words but tweet sentiment polarity as well as hashtags co-occurrence in tweets. In their investigation, they focused on three domains in order to address task (1) sentiment polarity of tweets containing the hashtag; (2) hashtags co-occurrence relationship and (3) the literal meaning of hashtags.

In contrast, our work aims at identifying hashtags carrying sentiment information independent from the actual textual content of the hashtag itself.

3.2.2 Data Set

We used the Twitter Stream API, which provides a random sample comprising 1% of all tweets created in the world on a day. In order to form our collection, we collected all messages provided by the API between January and June 2014 together with the available meta-information published in JSON format via the Stream API. This process yielded a collection of 784 million tweets from which 262 million are in English. Subsequently, we automatically selected 113K tweets (from 262 million English) on 10 controversial topics "Obama", "Bush", "Lady Gaga", "Justin Bieber", "Islam", "Lakers", "Youtube", "iPad", "Android" and "Microsoft". Furthermore, we ended up with 17k tweets after applying following steps of Twitter Pre-Filtering component along with other steps

1. Length based and re-tweets removal
2. Duplicate removal using MinHash algorithm
3. Application of tweet spam classifier
4. Restriction to hashtags where the number of tweets is at least 10

Step 1, 2 and 3 yielded in 37k tweets, while step 4 further reduced the number of tweet instances to 17k.

3.2.3 Approach

For the Opinionated Hashtag discovery scenario, we considered all hashtags appearing in at least 10 tweets, which passed through the filtering performed by the Twitter PreFiltering component. In this way, we ended up with 450 hashtags, which were - in the next step - hand-labelled as either positive, negative or objective. This has resulted in three classes of hashtags with 100, 200, and 150 instances, respectively. For the classification, we first have built three types of binary classifiers to separate each sentiment class from the other two classes, i.e., we applied a "one vs. all" (OVA) strategy: positive vs. all, negative vs. all and objective vs. all. The feature vectors were constructed using the tf-idf weights of the terms appearing in twitter messages for every hashtag. While doing this, we also accounted for negations (i.e., if a negation, say, "not", immediately precedes another term, we created a virtual term not). We used balanced sets with equal number of selected
instances from each class. For instance, as 100 tweets are annotated as positive, the positive vs. all classifier was trained with 50 tweets from the positive class and 25 tweets selected from each of the negative and objective classes. In a next step, we applied the one-vs-all trained sentiment models in order to predict the sentiment class of a hashtag. In a nutshell, this meant focusing on predicting the sentiment of hashtags that appear in non "low quality" tweets that are in English and were not automatically generated by diverse applications connected online. For each hashtag appearing in at least five tweets, we predicted the sentiment class from trained SVM binary OVA classifier. Next, we ranked all the hashtags for each class based on the distance from the separating SVM hyperplane of the tweets containing the hashtag.

**Result Inspection**

As we currently have a dataset based on only 150 hashtags, we only did a rough manual inspection of the results, which has shown the potential of the approach. More rigorous evaluation will be done in the next months. Table 3 shows list of positive and negative hashtags which were correctly classified after having manual inspection of results whereas Table 4 shows two example hashtags along with sample tweets as identified by our component for being positively (#beliebers) and negatively opinionated (#ISIS), respectively.

**Table 3: Example Hashtags classified as positive and negative based on their association with negative and positive content.**

<table>
<thead>
<tr>
<th>Positive Hashtags</th>
<th>Negative Hashtags</th>
</tr>
</thead>
<tbody>
<tr>
<td>#beliebers, #ps4, #justin, #surface, #jesus, #jelena, #billgates</td>
<td>#ISIS, #Syria, #Iraq, #Hijab, #Daish, #dawah, #ummah, #drones</td>
</tr>
</tbody>
</table>

**Table 4: Example Hashtags classified as positive and negative based on their association with negative and positive content.**

<table>
<thead>
<tr>
<th>#ISIS</th>
<th>#beliebers</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://t.co/tlbqoncmll">http://t.co/tlbqoncmll</a>. the world is less violent than it ever has been...â€œare you kidding me #obama? #iraq #syria #isis #noamnesty</td>
<td>good luck in 2014 do not forget we love you by: #beliebers #justinbieber #kidraulh @justinbieber 34</td>
</tr>
<tr>
<td>#isis &amp; #assad: #syria is in a revolution not a sectarian civil war <a href="http://t.co/08fuuzctbo">http://t.co/08fuuzctbo</a> #iraq #obama via @clayclai</td>
<td>@justinbieber hello I am biggest fan and I wsnt to meet you very much #beliebers #justinbieber #biggestfan</td>
</tr>
<tr>
<td>#Obama says #ISIS empowerment threatens Jordan, Europe, US #Iraq</td>
<td>when you smile,i smile #justinbieber #beliebers <a href="https://t.co/srmms2jsws">https://t.co/srmms2jsws</a></td>
</tr>
</tbody>
</table>
Next Steps

We are planning to further improve this approach by widening training dataset to at least 1000 hashtags. In this regard, we are working on the HSpam14 dataset, which we have also used in the Twitter pre-Filtering component. Since 9.5 million tweets are already annotated in the HSpam14 dataset, we will get a sufficient number of hashtags for training a classifier and a sufficiently large data set for testing the classifier. Thus we will also have rigorous evaluation in future work as by then we will have at-least 1000 hashtags to evaluate.

3.3 Twitter Pre-filtering Component

As reported in D2.2 (Section 3.1), early in the project, we focused on filtering out non-interpretable content, also referenced as "low-quality" content in the literature [3, 5, 4]. This refers to interpretation by humans as well as to interpretation by machines. The steps used for this purpose are illustrated by figure 4.

![Figure 4: Twitter Stream PreFiltering Component.](image)

This approach works well for filtering out different types of "low-quality" content. However, it has focused on batch pre-processing of data, where duplicate tweets removal technique mainly rely on jaccard similarity algorithm as shown in figure 4. Since our applications mainly target real-time processing and there is also other types of data to be filtered out by pre-filtering, we have revised our approach in two ways: a) we have developed an approach, which works for both real time and batch data, and b) our approach filters out spam [6] tweets along with filtering out re-tweets and duplicate tweets.

In this new approach, tweet level SVM based spam classifier (you are right it should be spam classifier) is implemented on tweets labelled as either spam or not-spam (ham) which can classify a tweet irrespective of application mode either real-time or batch data in order to filter out "low-quality" content where the applied classification approach is described in detail in D2.1 (section 6) for the case of sentiment classification.
3.3.1 Spam Detection

As described above, our aim is to develop a component for filtering out non-interpretable and useless content. Our methods here focus on filtering out the so called "low quality" content. A review of previous literature reveals various works using Twitter pre-processing steps for a wide range of different problem settings [3, 5, 4].

Within this work we focus on a component aimed at pre-filtering the Twitter data that is used by the sentiment and event detection algorithms.

Related Work

In related work, Thomas et al. reported that there is an underground market in Twitter network [7] to influence user perspective either through advertisements or tweets by agents such as mobile application. In their study, they also reported that 77% of spam accounts are identified by Twitter within the first day of creation, and 92% of spam accounts within first three days of creation. The authors also made the observations that 89% of spam accounts have fewer than 10 followers and 17% of spam users exploit hashtags to make their tweets visible in search and trending topics.

Surendra et al. [8] proposed a way to deal with tweet-level spam detection where they mainly focused on hashtags, in order to identify spam tweets and annotate tweets. In this regard they collected 14 million English tweets from trending topics and labeled all these tweets using the following 4 steps:

1. Heuristic-based tweets selection to search for tweets that are likely to be spam
2. Near duplicate cluster based annotation to group similar tweets into clusters and then label the clusters
3. Reliable ham tweets detection to label tweets that are non-spam (also known as ham)
4. EM-based label prediction to predict the labels of the remaining unlabeled tweets using an Expectation Maximization (EM) algorithm.

Table 5 lists a few example tweets from the 14 million dataset.

Data Set

The aforementioned 14 million tweets were available in form of tweets ids. Thus, we used the Twitter Stream API to collect all tweets together with the available meta-information published in JSON format via the Twitter Stream API. This process yielded a collection of about 9.5 million tweets from the 14 million tweets. The remaining 4.5 million tweets were either private or removed and, thus, could not be collected. The resulting dataset contains around 2.5 million ham tweet, while the remaining about 6 million tweets are
Table 5: Example Spam and Ham Tweets from the 14 Million Spam detection data set.

<table>
<thead>
<tr>
<th>Tweet</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>I've collected 12,293 gold coins! <a href="http://t.co/MXyllUOlZa">http://t.co/MXyllUOlZa</a> #android, #androidgames, #gameinsight</td>
<td>Spam</td>
</tr>
<tr>
<td>What would you spend a 30 @wikaniko voucher on? You can win one with TheMiniMesandMe here : <a href="http://t.co/Eh3y9LOCVq">http://t.co/Eh3y9LOCVq</a></td>
<td>Spam</td>
</tr>
<tr>
<td>RT @BenGenMusic23: I love those people you can joke around with and have so much fun with and then have a deep conversation with and it’s n</td>
<td>Ham</td>
</tr>
<tr>
<td>RT @cnnbrk: Chris Kelly of 1990s rap duo Kris Kross has died, said Fulton County, Georgia, medical examiner’s office. <a href="http://t.co/3Wt3twc4ec">http://t.co/3Wt3twc4ec</a></td>
<td>Ham</td>
</tr>
</tbody>
</table>

spam. Furthermore, we used only 2 million tweets in our classification methods where both ham and spam tweets were randomly picked from the set of 9.5 million tweets and this data will be given to SVM for training of tweets spam classifier.

We have to experiment with different SVM classification parameters to evaluate accuracy of this approach and training a classifier on large dataset is quite time expensive. Therefore, we opted for a set 2 million random tweets for every unique set of parameters where training a classifier on 1 million balanced set of spam and ham tweets and then testing a classifier on remaining 1 million tweets took around 24 hours.

**Approach**

For our classification experiments, we have used the same setup as we used in previous tweet level SVM based classification methods described in deliverable D2.1 (see Section 6.1.4). There, is however, one important difference: here we have experimented with an SVM-based binary classifier instead of an one-vs-all classifier, because we are interested in only either ham or spam tweets. In our previous experiments on sentiment classification we had been interested in 3 classes, thus using a one-vs-all classifier.

In order to train the classifiers we randomly split the instances from the target class into two sets reserved for training and testing, and randomly selected an equal number of instances from the two classes for training as well as for testing. In this way, we created balanced training and test sets for each classifier. This is similar to the approach employed by [9] to eliminate the effect of any underlying bias for a particular sentiment class in the data. In the end, a 1 million balanced training set was given to the SVM classifier, in order to train the model. Testing has been done with the 1 million test set. A small sample of the correctly classified test tweets are shown in Table 26.

**Evaluation measures**

In order to evaluate the effectiveness of a spam classifier, which is usually trying to assign previously unseen documents to one out of two classes (e.g., spam vs ham), various eval-
### Table 6: Correctly classified sample tweets from the Test set.

<table>
<thead>
<tr>
<th>Tweet</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT @890203mengjia: Everyone! Vote for T-ara N4 - Jeon Won Diary for</td>
<td>Spam</td>
</tr>
<tr>
<td>Kpop Music Mondays! <a href="http://t.co/n6rbUo9tmX">http://t.co/n6rbUo9tmX</a> via eatyourkimchi</td>
<td></td>
</tr>
<tr>
<td>RT @minsulshippers: 130502 [MBC Radio] The song that Minho picked as</td>
<td>Ham</td>
</tr>
<tr>
<td>&quot;The Music Of My Life” ~ &lt;&lt; Zitten’s ’??’ &gt;&gt; <a href="https://t.co/">https://t.co/</a></td>
<td></td>
</tr>
</tbody>
</table>

Evaluation metrics can be employed. For instance, the Accuracy measure in our classification scenarios refers directly to the accuracy in the given taxonomy, while Precision and Recall refer mostly to the Accuracy vs. Completeness dimensions. In our experiments we use precision and recall for assessing effectiveness. More details on evaluation measures are described in D2.1 (Section 6.1.3).

### Results

Surendra et al. [8] evaluated the quality of the labeled tweets by manually inspecting a randomly selected small sample of 1000 reliable ham tweets out of 14 million tweets. Among them, 32 tweets were incorrectly labeled as ham and the accuracy was 0.968. On the other hand, our selected sample 1 million test set is quite large compared to 1000 tweets, thus results are different than their results which are explained below.

Our quality measures are precision and recall. In more detail, we look at the precision-recall curves as well as the precision-recall break-even points (BEPs) for these curves (i.e. precision/recall at the point where precision equals recall, which is also equal to the F1 measure, the harmonic mean of precision and recall in that case).

The detailed precision-recall curve for the classifier is shown in Figure 5. The main observations are:

1. The classifier provide relatively good precision (at around 0.8) for Recall values around 0.5. This means, nearly half of the spam is found with a low risk of eliminating useful tweets.

2. By trading recall against precision, the classifier can be adapted to the needs of the respective application. For a conservative approach high precision values ensure that no useful content is lost (e.g., prec=0.91 for recall=0.1), whereas combinations such as prec=0.7 for recall=0.6 for the spam-vs-ham classification scenario can be used for finding candidates for spam tweets in large Twitter sets.

### 4 Handling Historical Data

In many situations in financial risk analysis, it is desirable to be able to explore the data and provide the analysis in retrospective manner, in addition to seeing the results of pro-
Figure 5: Precision and Recall for the Spam Filtering Method shown as Precision recall curve.

cessing the current stream data. For example, business analysts might wish to compare the current financial results with the same quarter from the past year or inspect a similar event from the past to better estimate the financial risks. This need for analyzing "historical" processing results is also confirmed in the QualiMaster applications as described in the revised version of deliverable D6.1 and in deliverable D6.3.

In the QualiMaster project two complementing mechanisms for looking into historic data have been realized. The method for inspecting the evolution of dynamic graphs has already been described in Section 2.5 of this deliverable. This mechanism focuses on enabling a deeper analysis of the evolution of dynamic graphs such as the correlation graph. A second, generic mechanism, which we call Replay Mechanism, enables the on-demand replay of older results (result streams) on different levels of granularity. This mechanism is described in more detail below.

4.1 Concepts for a Replay Mechanism

The replay mechanism offers the possibility to replay older result-streams on demand. It is triggered in an interactive way enabling the application user to specify the parameters of the replay such as granularity and time frame, while not interrupting the real-time processing pipelines. The processing should be transparent to users as if they are still in the same pipeline. The ability of combining online and offline result delivery is critical to decision making in financial analysis and for a deeper understanding of current situations.

The replay mechanism imposes some challenges due to the difference in the processing models of stream and batch data. The major functionality of Storm as well as of other stream processing frameworks (Spark Streaming\(^1\), Samza\(^2\)) is to facilitate the ingestion of events in a real-time or near real-time manner. To do so, these frameworks are optimized for high-throughput processing, and tolerate relaxed accurateness guarantees\(^3\). The data

\(^1\)http://spark.apache.org/streaming/
\(^2\)http://samza.apache.org/
\(^3\)Core Apache Storm guarantees that each message is processed at least once. Duplicates due to
cannot be stored over long term because of limited memory capacity, and iterations over data is restricted. Such scenarios are well-handled in batch processing paradigm such as MapReduce [10] or Bulk Synchronous Parallel [11]. However, their disadvantages are the latency of delivering processing results (sometimes can be up to several seconds or minutes, or even hours), which makes online and interactive processing infeasible.

One option to include a replay mechanism into a framework is to use a hybrid processing mode between the streaming and batch right from the beginning by providing the abstraction over the computation. Example frameworks in this direction are SummingBird of Twitter [12] and RADStack [13]. Figure 6 overviews the architecture of SummingBird (elaborate more here). RADStack has a similar machinery, but it relies on an advanced data store to enable broader scenarios of analytics. The problem with the frameworks such as SummingBird and RADStack is that they require to re-write many business logics in new APIs. This is difficult if there are already many advanced analytical modules at hand (some of them might even import third party components which are impractical to refactor). In addition, they do not allow online re-configuration.

4.2 Architecture of The Replay Mechanism

Therefore, a more lightweight approach has been selected in the QualiMaster project, stressing its main focus on stream processing as well as on configuration and adaptation support. Figure 7 illustrates the design of the replay mechanism in the QualiMaster infrastructure.

Our replay mechanism seamlessly integrates with the Storm processing framework, and fully enables the monitoring and adaptation of replay scenarios in real-time. The main idea is that instead of introducing a new module for batch processing over historical data failure are possible
(as, e.g., in Summingbird), we store the real-time results calculated by a pipeline into a repository. The repository supports dynamic aggregation of results.

In the framework the Replay Mechanism is modeled as a special type of a sink, a so-called Replay Sink. With respect to the applications this acts in the same way as a normal sink, just providing a stream of historical results instead of real-time processing results. As this only affects the configuration of the pipeline sinks, developers can adjust existing pipelines with minimum efforts just using a replay sink instead of or in addition to a normal sink.

The only extension needed in the infrastructure are two new modules: the ReplayRecorder and the ReplayStreamer (Figure 7). The ReplayRecorder specifies how the real-time results are stored and compacted into the result repository, and the ReplayStreamer fetches the historical results on demand, makes aggregation if needed, and pushes them to the applications as if they were real-time results. In this way the client-side consumption in the application (e.g. visualization) can be reused. The replay results are pipelined through a special channel and are labeled with identifications to be able to serve multiple replay requests simultaneously.

For each replay request, the replay mechanism works as follows: When the application issues a request, this triggers a special message, which is sent to the listener in the sink (bottom-right part of Figure 7). The listener recognizes the message, delegates it to the adaptation layer and issues a replay command to the coordination layer. The Adaptation Layer can also reject the request due to overload conditions. The coordination layer sends a replay data signal to the replay sink. The coordination layer can also send subse-
quent signals to request a re-configuration, for instance changing the time granularity for aggregation of the historical data. Those parameters will decide about the granularity and the speed, in which the replay will take place. As soon as the replay sink receives one of these signals, it restarts one of its streamers (there will be several such streamers in the pool, each responsible for an individual client replay query), and starts fetching data from the repository. The replay data is sent back to the sink in the same format of the real-time data.

The replay mechanism and its implementation as part of the QualiMaster infrastructure will be described in more detail in deliverable D5.4.

We describe briefly a case study of the integration of the replay mechanism in the Priority pipeline here. There are requirements for the the analytical modules to be able to plug-in into replay mechanism. The results must be timestamped to be stored to the repository, the developer should specify clearly the schema of data in terms of \(<key,value>\) of replay data to facilitate querying over past data later on.
5 Conclusions and Future Work

This deliverable is the third deliverable of WP2. It describes the development state of algorithms and methods in form of pipeline components for scalable and quality-aware real-time data stream processing. The implementations of components from the first phase of the project were further refined and extended and further components have been developed in a demand driven way. In addition further evaluations of components have been performed and planned.

The developed algorithms include processing of financial as well as social web data streams.

Computation of further, more advanced correlation measures such as mutual information (Section 2.1) and transfer entropy (Section 2.2) implemented in a stream-based way will enable deeper analysis of the dependencies between market players and enables innovation applications for financial risk analysis.

Furthermore, in this deliverable we have addressed the need of financial stakeholders to inspect past developments in addition to observing current developments based on real-time data streams. For this purpose, we have developed methods for capturing and querying dynamic graphs (Section 2.5) and for replaying the result of past analysis in a demand driven-way using the Replay mechanism (Section 4).

An advanced component for sentiment analysis (Section 3.2) based on hashtags helps to identify the feelings in the market in different granularity, be it general, player or financial area related. Pre-filtering components (Section 3.3, which can, for example, detect spam in an effective way support focusing on relevant high quality content in the data stream found in Social Media. Finally, event detection (Section 3.1) as identified by advisory board of the project, is important to keep the expert informed about important events related to selected financial areas, or particular market players and can be used to re-focus attention in the market observation.

A special focus of all proposed algorithm and their novelty is in the ability to handle large amount of data in form of high velocity streams and to meet the needs of the QualiMaster applications for financial risk analysis.

The next steps in WP2 will be the further development and optimization of the algorithms and other methods presented in D2.3 We will continue conducting extensive experimental evaluations of the quality, performance and scalability of the developed algorithms.
References


