www.qualiMaster.eu

QualiMaster

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

Grant Agreement No. 619525

Deliverable D2.1

Work-package	WP2: Scalable, Quality-aware Data Processing Methods
Deliverable	D2.1: Approach for Scalable, Quality-aware Data Processing
Deliverable Leader	LUH
Quality Assessor	Ioannis Papaefstathiou (TSI), Apostolos Dollas (TSI)
Estimation of PM spent	12
Dissemination level	PU
Delivery date in Annex I	31.12.2014
Actual delivery date	31.12.2014
Revisions	7
Status	Final
Keywords:	Algorithms, Stream Processing, Social media Analysis

Disclaimer

List of Authors

Partner Acronym	Authors
MAX	Steve Hutt
LUH	Mohammad Alrifai, Claudia Niederée, Sergiu Chelaru, Patrick Siehndel and Sergej Zerr
SUH	Holger Eichelberger
TSI	Ekaterini Ioannou and Minos Garofalakis

Table of Contents

Executive summary

This deliverable is the first deliverable of WP2. It reports on the progress of WP2 in the first year of the QualiMaster project towards developing scalable and quality-aware methods for real-time processing and analysis of data streams. More specifically, the focus of this deliverable is on describing the foundations, the concepts and the set of algorithm families that will be considered during the course of the project. The deliverable describes families of algorithms for generic data stream processing as well as algorithms that are particularly relevant for conducting financial (systemic) risk analysis using real-time information from financial market streams such as stock trading tick data or currency change quotes. The selection of these algorithms was based on a deep analysis of the requirements that resulted in WP1 and in collaboration with the industrial members of the consortium. The selected algorithms will be implemented and optimized for higher scalability and minimum latency.

Furthermore, the deliverable describes a set of methods that will be evaluated, optimized or customized during the course of the project for processing and analyzing (social) web data streams in support of the financial (systemic) risk analysis. The selection of these methods was based on the requirements and on an extensive study of the state-of-the-art in web mining and recent work on extracting meaningful signals from the social web for the detection of emerging events or the prediction of the development in financial markets.

In addition to studying and selecting specific data processing algorithms and methods, we also started implementing and evaluating some of these methods as part of the development of the first real-time stream processing pipeline in QualiMaster. This includes the implementation of the Hayashi-Yoshida Correlation Estimator as well as the SVM and SentiWordNet sentiment classifiers as Apache Storm topologies that are executed and evaluated on a computing cluster.

1 Introduction

One of the key goals of the QualiMaster project, and WP2 in particular, is to analyze, extend, adapt and develop families of scalable data stream processing algorithms to perform real-time analysis over high volume, high velocity and bursty data streams. This includes algorithms for sketching, aggregating, correlating or predicting the evolution of several data streams in order to be used as basic data processing elements in the QualiMaster infrastructure. Building on these basic processing elements, applications can use the QualiMaster infrastructure to conduct more sophisticated analysis on the data streams in real-time. Special focus will be given to the application of the QualiMaster in supporting the financial (systemic) risk analysis as the main application of the QualiMaster project. Furthermore, a set of data and web mining algorithms and methods will be used to harvest the vast amount of user generated data on the web (e.g. in micro-blogging and social media platforms such as Twitter, YouTube or Yahoo News as well online news feeds) in order to extract useful information and signals in support of the analysis that is made on the financial data streams.

The functionality based selection of the algorithm families is derived by the use cases and user requirements defined in WP1 (as documented in D1.1 and D1.2). However, the decision of the selection of specific implementation is mainly based on whether the implementation of the algorithms can be distributed to scale up to high volume and high arrival rate of data in real-time or not. While some algorithms are ready for execution in a distributed computing environment, others either need to be adjusted first or are not suitable for distributed computing at all. In WP2 we target the first two classes of algorithms, and, if required, adjust their implementation in a distributed fashion.

Typically, different implementations (or parameterized variations of the same implementation) of a data processing algorithm or method exist that differ in quality/performance tradeoffs. Our goal in WP2 is to develop families of algorithms that provide the same functionality and share the same interface (in terms of input and output parameters) but differ in quality/performance characteristics in order to provide flexibility to the QualiMaster by enabling different configurations at design time (e.g. according to user requirements) or by automatic adaptation at run time (e.g. in reaction to changes in input load or emerging events). Therefore, in this deliverable we describe the evaluation methodologies and measures for the quality and performance of the data processing families following the taxonomy of quality parameters and the Adaptation model developed in WP4 in collaboration with WP2.

1.1 Relation to other Deliverables

The selection of the algorithms and the foundations to achieve scalability and quality-awareness have been driven by the requirements and use cases of WP1, which are described in deliverables D1.1 and D1.2. Some of the considered algorithms are being already translated to reconfigurable hardware in WP3, which are then described in D3.1. The quality model and quality dimensions applied in this deliverable are the results of a strong collaboration between WP2 and WP4, therefore, this deliverable is also connected to D4.1. Finally, the deployment, integration and execution of the developed algorithms are being performed on the QualiMaster infrastructure that is provided by WP5 and described in deliverables D5.1 and D5.2.

1.2 Contribution to the QM Priority Pipeline

During the first year of the project several data processing algorithms have been studied, analyzed, adjusted or re-implemented to adhere to the real-time processing and scalability

requirements. A special focus have been given to the implementation and experimentation with the algorithms that are part of the *QualiMaster Priority Pipeline* as a proof-of-concept and for testing the integration with the components developed in other work packages. The design of the QM priority pipeline is illustrated in Figure 1. For more detailed description of this pipeline and its integration please refer to deliverable D5.2.

The contribution of WP2 to this pipeline is the distributed implementation of the real-time computation of the correlation matrix from the financial data stream using Hayashi-Yoshida Correlation Estimator (component 3b), the pre-processing and annotation of micro-blogs stream from Twitter (component 5), and the sentiment analysis of micro-blogs using SVM (components 6a) and SentiWordNet (components 6b) as part of Apache Storm topologies. More details on these methods, their implementation and the preliminary results can be found in Sections 4.1.1, 5.2.1, 6.1.4 and 6.1.5, respectively.

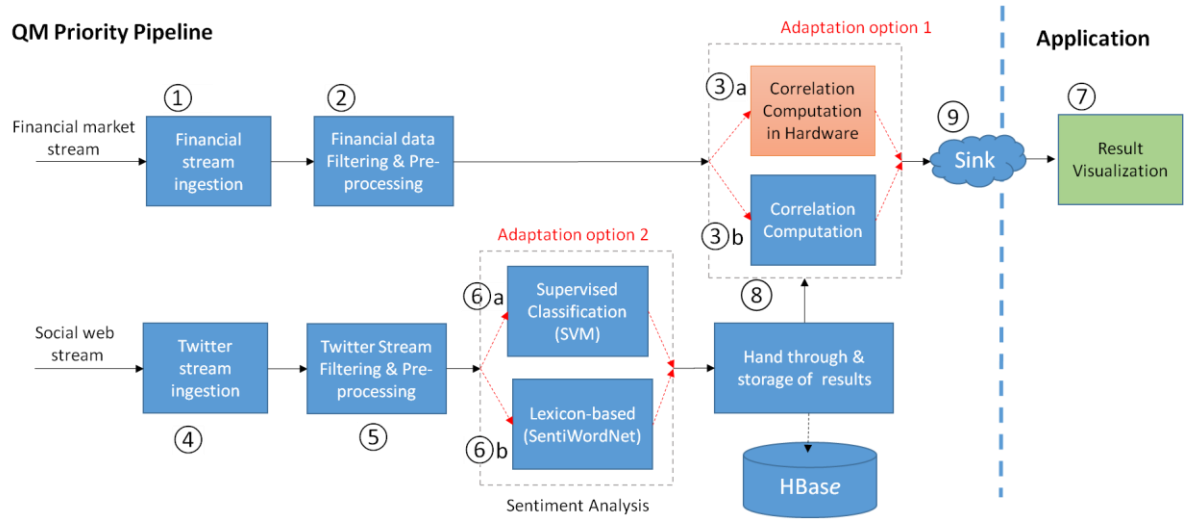

Figure 1: QualiMaster Priority Pipeline

1.3 Outline of the Deliverable

The deliverable is structured as follows. First, Section 2 reviews the requirements and challenges for developing scalable, quality-aware real-time data stream processing algorithms and then describes the QualiMaster approach for addressing them. Section 3 presents three algorithm families for generic data stream processing that will be considered in QualiMaster. These are algorithms that can be re-used in different scenarios and applications. Section 4 then discusses a list of algorithms that have been identified as relevant for the (systemic) risk analysis in the financial domain. In particular, the distributed implementation of the Hayashi-Yoshida Correlation Estimator and its incorporation in the QM priority pipeline is discussed in more details in this section. Next, Section 0 describes some basic methods and tools that are used in QualiMaster for adaptive crawling and pre-processing of textual content from the web. Section 0 gives an overview of the methods that are being considered in QualiMaster for real-time analysis of social web data streams in support of (systemic) risk analysis including sentiment analysis, user and social network analysis, event detection and entity network extraction. Moreover, preliminary results of the initial implementations of some of the proposed methods and the incorporation of them in the QM priority pipeline are also reported in Section 6. Finally, Section 7 summarizes and concludes this deliverable with insights on future work towards the next deliverable, which is due in July 2015.

2 Scalable, Quality-aware Data Processing

This section describes the QualiMaster approach for addressing scalability and quality-awareness issues of data processing algorithms. We start by reviewing the relevant requirements from D1.1 and D1.2 and highlight the key challenges as the motivation for the derived solutions. Next, we discuss the high level concept for achieving scalability and quality requirements.

2.1 Requirements

By analyzing the user requirements and use cases that are described in D1.1 and D1.2, we can summarize the algorithm related requirements as follows. For more details, please, refer to D1.1 and D1.2.

2.1.1 Real-time Processing

Real-time processing of online data streams. This requirement is directly derived from the initial requirements REQ-DS1 and REQ-Q1 in D1.1. It implies the need for designing and developing data stream processing algorithms with a very low latency (possibly in sub-seconds level).

2.1.2 Support for Processing of Historical Data

Although processing of real-time streams is the main focus of the QualiMaster project, it is also required to provide a support for batch processing of historical data as an important part of any analysis or prediction model (see REQ-DS3 in D1.1). This requirement implies providing a batch-processing implementation of the key algorithms to be developed in QualiMaster as well as management mechanisms to provide the input data to the batch processing algorithms in an adequate form. As discussed in D5.1, this is done by the Data Management Layer, which manages the data sources and sinks for the real-time processing. In order to work on actual data in batch processing, the Data Management Layer can be configured to transparently store data received by the data sources, intermediary data produced by the data processing pipelines and output data at the pipeline sinks. In particular, this includes strategies for managing aged data and for coping with the overall data volume. An additional benefit from the use of historical data is that they can be obtained at much lower cost vs. real-time stream data, and that due to our ability to construct experiments in which we can know *a priori* where singular events take place we can feed historical data to the QualiMaster pipeline and in real-time evaluate its ability to detect these singular events.

2.1.3 Scalability

Requirements REQ-DS4, REQ-DS5, REQ-DS6, REQ-DS7 and REQ-Q2 define the minimum scale in terms of the input size that the QualiMaster infrastructure (and its algorithms) has to be able to process. It is required that the coverage of the data sources (e.g. stock markets) is maximized. These requirements also imply that the algorithms need to be capable to process any (possibly unexpected) increase in the numbers of input streams or the arrival rates of tuples.

2.1.4 Quality

It is expected that optimizing algorithms for low latency, and scalability goals often implies some loss in the quality of the output (e.g. approximation vs. exact solutions) and higher cost in terms of computational resources. Furthermore, the use web and social media streams (which are

inherently noisy and have no quality guarantees) require additional effort on the assessment of the quality and reliability of the analysis results that are based on these sources. Requirements REQ-Q3, REQ-Q4 and REQ-Q5 from D1.1 focus on the quality and cost aspects. It is required that the processing algorithms are designed in a way that enables the estimation of the quality of their output as well as the computational cost of producing this output. This should enable QualiMaster to provide the end users with useful information about any possible trade-offs between quality and cost.

2.2 Sources and Challenges of Data Streams

We distinguish between two types of data streams that are considered in the QualiMaster project, which pose different characteristics and thus different challenges (see Figure 2). The two types are: financial data streams and social web data streams.

2.2.1 Data Sources

2.2.1.1 Financial Data Streams

These are online streams of short messages on the trading in the financial markets. Examples include trading of stocks in stock markets, currency exchange quotes, futures on indices and futures on bonds. The messages are typically of a short length and have a well-defined structure. Several streams can be collected from different markets at different levels of granularity. The velocity of the streams is considerably high (i.e. data volume is growing at a high rate) making it very challenging to deal with it in real time.

2.2.1.2 (Social) Web Data Streams

Examples of social web data sources that will be considered in QualiMaster include general-purpose micro-blogging platforms such as Twitter as well as other more specialized financial blogs such as StockTwits and SeekingAlpha, which focus on financial market related topics. QualiMaster will also investigate the exploitation of other social media platforms with rich user-generated content such as YouTube and Wikipedia. In addition to social media platforms, QualiMaster will collect and process news articles from online news RSS feeds. The data collected from the aforementioned sources are typically noisy, semi-structured/unstructured data streams, which originate from heterogeneous sources.

2.2.2 Data Challenges:

The goal of WP2 is to address three key research challenges (see Figure 3) with respect the types and characteristics of the data to be processed, which we describe below.

2.2.2.1 Dealing with high Velocity and high Volume Financial Data Streams:

The challenge here is to develop scalable algorithms for real-time data stream processing. This is particularly crucial for data streams of high velocity (i.e. high arrival rate of new data) such as the financial data streams, which can contains millions of messages per second that report in real-time about trading activities in financial markets (e.g. stock or foreign exchange markets). The algorithms should be re-implemented or adapted to run in a distributed cluster in order to be able to exploit state-of-the-art Big Data technologies such as Apache Hadoop and Storm. In addition, several variations or alternative implementation of the algorithms that differ in quality and

performance should be considered to allow the QualiMaster infrastructure to adapt to (dynamic) changes in the volume and velocity of the streams in real-time.

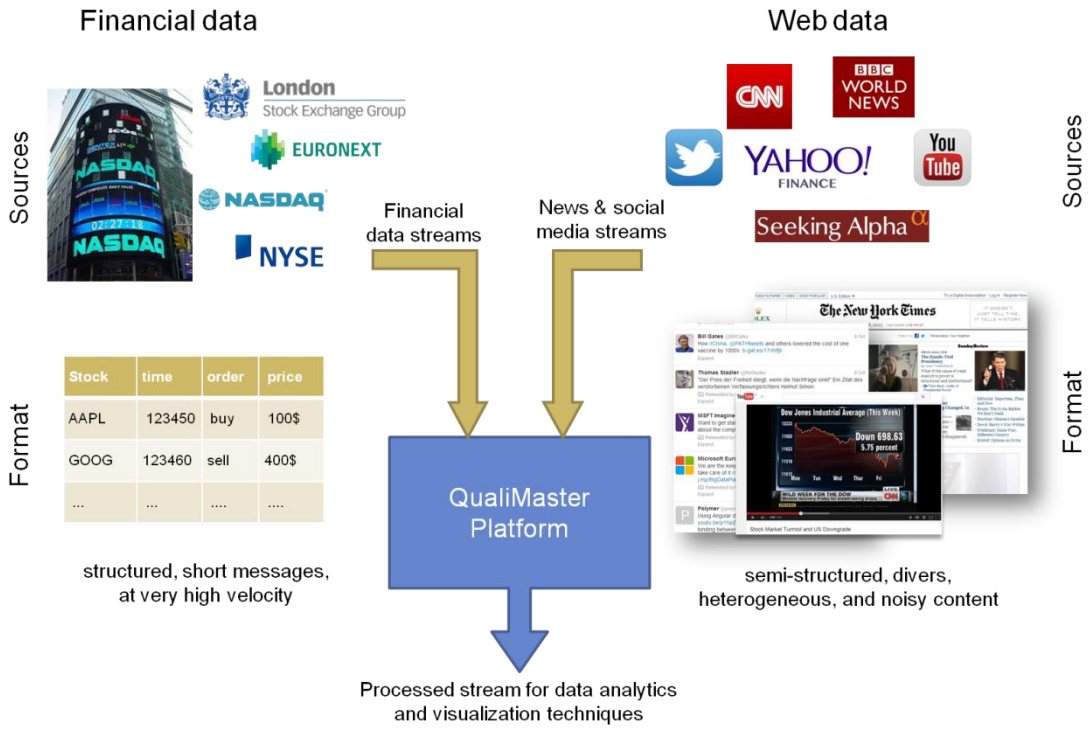

Figure 2: Data stream sources and format

2.2.2.2 Dealing with high Variety and Low Quality Social Web Data Streams:

The challenge here is to cope with the heterogeneity of the sources and the noise in the data in order to be able to extract useful and meaningful signals out of it. This is particularly relevant for social web data streams. The task is transform the unstructured (or semi-structured) and noisy data streams into well-structured and semantically rich streams that can be further exploited by the analytic and visualization tools the same way the structured financial streams are used. Another challenge here is to correctly identify and extract relevant information with respect to the application goals (e.g. in support of systemic risk analysis), which is not straightforward and requires extensive experimentation and investigation of different approaches and strategies.

2.2.2.3 Combining Financial and Social Web Streams for Better Analysis:

It is expected that several signals (e.g. time series) will be extracted from the social web data streams, which might have different semantics depending on the source and the content of the streams. The use of signals from social web streams can help in providing contextual information that complements the analysis that is based on financial streams, and possibly enable developing better prediction models. However, it is expected that the time series produced from the analysis of financial streams will have different scale and granularity than the time series produced from the analysis of web and social streams. The challenge here is to develop models and solutions for combining, aligning and synchronizing these signals with those that are extracted from the financial data streams.

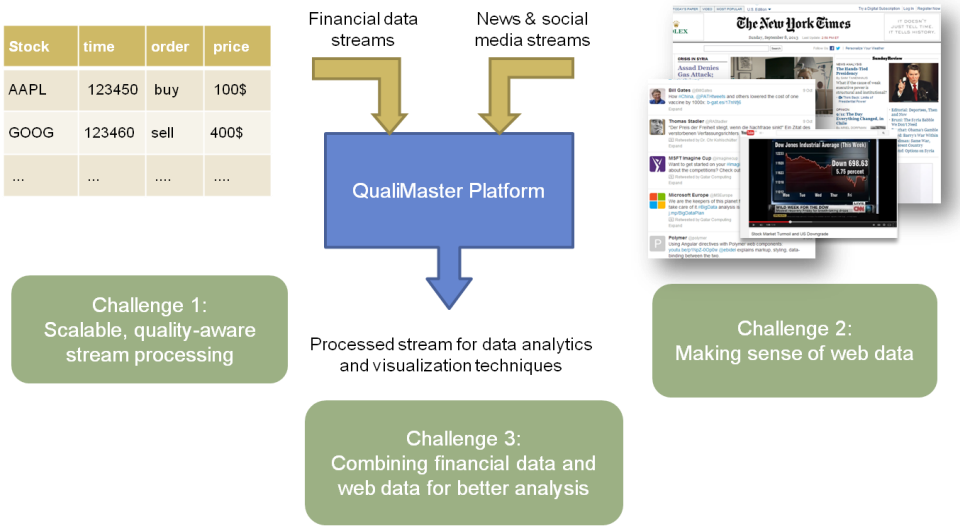

Figure 3: Data challenges

2.3 Scalability Model

The QualiMaster approach addresses the scalability requirement by applying the following four strategies, when applicable: 1) pipelining, 2) distributed and parallel computing, 3) acceleration via re-configurable hardware, and 4) load shedding and sampling. The next sub-sections describe these strategies in more details.

2.3.1 Pipelining

This is a well-known technique in High Performance Architectures which involves breaking down a sequential process into sub operations [18]. The sub-operations are then executed in a chain such that the output of a sub operation becomes the input of the next one. For example, if a function *F* consists of a sequence of sub-functions, for example *a*, *b* and *c*, such that $F(x_i)= a(b(c(x_i)))$, then it is possible to implement *F* using pipelined logic. The decomposition of *F* into three sub-functions is illustrated in Figure 4.

This computation model increases the concurrency and thus the overall throughput as it allows for processing k tuples at a time in a pipeline of k sub operations. For very large numbers of tuples, the speedup resulting from the use of a *k*-stage pipelined implementation is up to k times the time needed by a non-pipelined implementation (asymptotically approaching k in the best case). The maximum speedup is achieved when the computation time for all k sub-functions is equal or comparable.

QualiMaster will investigate the possibility of employing the pipelining strategy when applicable, by decomposing the data processing algorithms into sub-functions or sub-components that can be executed independently in a sequential process.

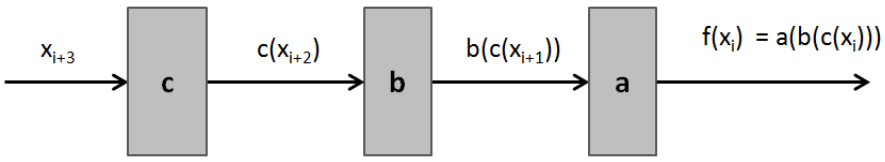

Figure 4: A pipelines Structure (example taken from [18])

2.3.2 Distributed and Parallel Computing

The distributed and parallel computing paradigm has been recently widely accepted and adopted by both academia and industry as a practical solution for solving Big Data issues due to the proliferation of tools and frameworks that enable seamless setup and configuration of clusters of commodity computes. Frameworks such as Zookeeper and Hadoop YARN, for instance, provide the necessary stack of services for managing the computing clusters such as resource management, task coordination, message communication and fault-tolerance keeping the developers focused on the data and processing it. The advantage of this distributed and parallel computing model is that it can easily scale-up to processing larger loads by simply adding more computing and storage nodes to the cluster without the need to change the processing logic. Furthermore, most of the existing frameworks allow for controlling the level of parallelism involved in a data processing job by setting the number of nods, tasks, or threads to be involved in the processing.

2.3.2.1 Stream Processing via Apache Storm

In QualiMaster, we focus mainly on stream processing. Therefore, we have decided to use the Apache Storm framework for stream processing (See D5.1 and D5.2 for more detailed discussion on the other options). There are three main concepts in the Storm model (see http//:storm.apache.org):

- Streams: this is the core abstraction in the Storm model. It is defined as an unbounded sequence of tuples that are processed and created in parallel in a distributed fashion.
- Stream grouping: this concept defines how streams can be partitioned or merged at run time at some of the logical computing nodes (e.g. bolts).
- Spouts: these are the sources of the streams. Typically, a spout reads the stream from external sources and emits its tuples to the Storm application.
- Bolts: these logical processing nodes. The processing of the stream tuples is done in bolts. Bolts receive an input stream (or several streams) and process each tuple and produce an output stream, which can then be sent to the next tuple for further processing.
- Topologies: Storm topologies are graphs of connected spouts and bolts. They define the structure of the real-time processing applications. Topologies are designed to run forever (or until they are stopped by the user).

The distribution and parallelism degree of running a Storm topology can be configured by setting up the number of workers (i.e. physical JVMs) or tasks (i.e. threads per worker).
In QualiMaster, we will implement all the processing algorithms that are considered in the project for generic stream processing (such as sampling or sketching), for financial streams processing (such as computing the correlation matrix or the mutual information) and for web and social streams (such as text annotations or sentiment analysis) as Storm topologies that run on a distributed cluster.

2.3.2.2 Batch Processing via Apache Hadoop MapReduce

For the support of processing historical data we also use the Hadoop MapReduce model in QualiMaster. All algorithms that will be used in the historical data processing use cases will also be implemented as MapReduce jobs for batch processing. For example, the sentiment analysis methods, which will be developed as Storm topology for real-time stream processing will also be implemented as a Hadoop MapReduce job that can be applied on historical offline collection of blogs or micro-blogs.

2.3.3 Acceleration via Re-configurable Hardware

Acceleration by hardware is one of the key strategies of QualiMaster for achieving scalability. This is accomplished by transforming the execution of some of algorithms from CPUs into FPGAs. By translating some of the bottleneck codes or the most frequently used codes into hardware, we aim at gaining significant speedup in the processing of streams. The design and implementation of the FPGA-based algorithms is done in WP3. However, the selection of the algorithms to be translated to hardware is done in strong collaboration between WP2 and WP3. Please, refer to D3.1 for more details on this.

2.3.4 Load Shedding and Sampling

This strategy is useful in situations where the input load suddenly and dramatically increases to levels that cannot be handled by the physical infrastructure. Instead of breaking the infrastructure and stopping the real-time processing of streams it is better to discard some of the input tuples (load shedding) or apply some random or semantic based sampling of the input. Although this can have an impact on the quality (accuracy) of the analysis, it still remains a practical alternative to completely shutting down the real-time analysis (which might have been already running for a while).

2.4 Quality Model

As mentioned earlier, one of the requirements in QualiMaster is to develop quality-aware data processing algorithms. In order to achieve quality-awareness we need a set of quality parameters and their measures for the algorithms. The key quality parameters (or dimensions) we consider in QualiMaster have been defined in deliverable D4.1 as a result of a consortium-internal quality survey based on initial work between WP2 and WP4. In this Section we present a brief overview of these quality dimensions (that are illustrated in Figure 5) and refer the reader to D4.1 for more details on the quality survey, the overall quality taxonomy as well as the propagation of quality parameters through a data processing pipeline.

2.4.1 Quality Dimensions

Figure 5 gives an overview of the taxonomy of the quality dimensions and parameters that are considered in QualiMaster. As the quality dimensions apply to the entire QualiMaster approach, we discuss, below, the parameters and their specific instantiation for algorithms. The quality survey conducted by WP4 indicated a high relevance for measuring and analyzing data confidence, latency, correctness as well as memory consumption characterizing computational cost. For the actual construction of the quality taxonomy and the underlying quality survey, please refer to D4.1. In Sections 3, 4, 0 and 0, when we describe the specific algorithms we will refer back to them with more details.

2.4.1.1 Time Behavior

Time behavior (or timeliness) is the time-related sub-dimension of performance efficiency of [105]. The quality survey of WP4 indicated two quality parameters of particular relevance for WP2, namely latency and throughput, both targeting timeliness.

The aim of the latency parameter is to minimize the overall delay between the arrival of a tuple in the input stream and the emitting of the results of processing this tuple by an algorithm. It is measured in terms of response time, i.e., in time units such as seconds or milliseconds. The algorithms will be optimized for low-latency so that the processing time of a tuple is minimized.

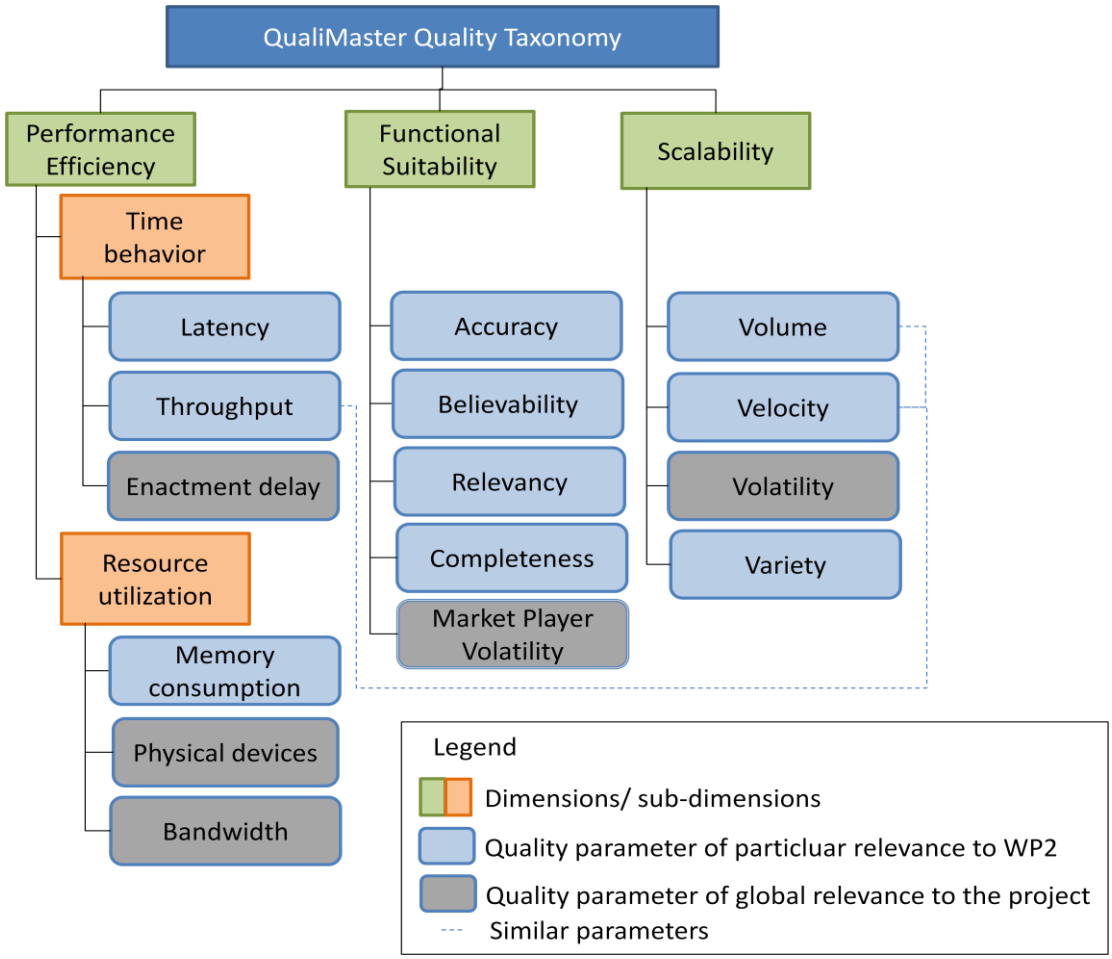

Figure 5: Quality dimensions (from D4.1, based on [105])

There is typically a trade-off between timeliness and other quality dimensions such as resource utilization or quality parameters such as throughput or accuracy.
Throughput measures the amount of items that is processed per unit of time or, related to volume, as the bytes processed per time unit. Obviously, throughput is related to latency, i.e., increased latency reduces the throughput. The aim of QualiMaster is to achieve a high throughput under varying conditions.

2.4.1.2 Resource Utilization

The aim of this parameter is to minimize the consumption of computational resources by an algorithm. The primary measure for this dimension is the memory consumption of algorithms during the design and the implementation. Further, we aim at CPU time as quality parameter, i.e., the actual execution time as opposed to the response time, as a measure for the utilization of the processor(s). However, here the notion of software-based processor utilization and hardware-based processor utilization need to be aligned. Also tools for monitoring the actual resources consumption at run-time will be deployed by the QualiMaster infrastructure (see D5.1, D5.2 and D4.1 for more details on the monitoring mechanisms of QualiMaster). There is usually a trade-off between cost-efficiency and output integrity or timelines. The more accurate or fast the data processing should be, the more resources typically are consumed to satisfy this requirement.

2.4.1.3 Functional Suitability

As discussed in D4.1, functional suitability in QualiMaster in particular the notions of data integrity, and, in WP2 to the quality of the algorithm's output data. It measures the integrity of the output data in several aspects as described below.

- **Data Completeness**:
  Completeness refers to the extent to which data is sufficient in breadth, depth and scope for the task at hand. In QualiMaster we use this property to measure the completeness of the input streams. This can be computed by measuring the coverage of the sources (e.g. in terms of stock markets, market players etc.), the sampling degree that is applied to the streams, and/or the recall of the applied algorithms(e.g. in classification algorithms). The goal is to maximize the completeness of the data as much as possible. However, this implies a trade-off with other quality dimensions such as timeliness and cost efficiency.

- **Data Accuracy**:
  Accuracy measures the extent to which data is correct, reliable and certified free of error. We use this property to measure the accuracy of the data processing and analysis output. The measurement of the data accuracy can vary from an algorithm to another depending on the concrete definition of the accuracy of the specific algorithms. Examples of possible metrics include the confidence or precision value of a prediction or a classification algorithm, or the error rate of an approximation algorithm.

- **Data Relevance**:
  One of the challenges in processing and analyzing social media streams and web sources in general is finding and extracting relevant information with respect to the user query out of the large volume of unstructured or semi-structured and noisy data that comes from a variety of heterogeneous sources. By data relevance we refer to the estimation of the relevance of the processed content (e.g. a blog or a news item) to the user query (e.g. to a specific stock or market player). For example, measuring the similarity degree between the user query and the entities mentioned in an online post gives an estimation of the relevance of that post to the query, which in turn can be used to judge the reliability of the analysis that is done based on the content of this post.

- **Data Believability (Credibility)**:
  The importance and the potential impact of the content of social media streams varies a lot depending on the source. We measure the data credibility based on the estimation of the credibility of the source or the authors of the content. This can be achieved, for example, by manually or automatically evaluating the popularity of a specific source (e.g. blog) or by measuring the influence of users in social networks, and taking this measurement as a basis for filtering the user-generated content in these networks.

2.4.1.4 Scalability:

The aim of this quality dimension is to measure the capability of the algorithm to scale-up to higher volume and/or higher velocity input loads. To achieve higher scalability we deploy the four strategies as described earlier in Section 2.3.

The scalability is measured by the maximum throughput of algorithm in terms of the maximum volume (in bytes) that can be processed per second (volume scalability) or the maximum number of tuples that can be processed per second (velocity scalability).

Scalability can also be characterized by the number of financial and social media streams that can be processed in parallel (called variety in D4.1). It can also be measured in terms of some domain-specific criteria such as, the number of financial markets or stocks that are analyzed simultaneously, or the depth of the market data streams.

Increasing the scalability of an algorithm or a processing pipeline implies the readiness to consume more resources (such as more computing nodes).

2.4.2 Algorithmic Families

Several algorithms can provide the same functionality but differ in the quality parameters (e.g. timeliness and resource consumption). Quite often, the decision on which algorithm is the best always depends on the user requirements. Therefore, in QualiMaster we adopt the concept of algorithm families. An algorithm family is an abstract representation of a class of algorithms that perform the task (i.e. functionality), take the same input and produce the same output. For example, Sentiment Classification of tweets could be seen as an algorithm family whose task is to take a tweet as input and annotate this tweet with one of three sentiment classes: positive, negative, or neutral. There could be several implementations for this family, which we call members. One example member of the Sentiment Classification family is an SVM-based supervised classifier that is trained to detect the right sentiment class of a single tweet. Another possible member is an unsupervised method that doesn't depend on any training data set but still is able to annotate each tweet with a sentiment class. The two family members share the same interface of the algorithm family, and thus can be (logically) swapped at any time in a running application without affecting the functionality of the application. However, these two implementations might differ in terms of quality (e.g. accuracy of the prediction of the right sentiment class or latency) and/or resource consumptions. This variety with respect to quality-performance trade-offs provides at run-time room for flexibility and adaptivity to any possible changes in the input characteristics or in the user needs.

As a rule of thumb, in QualiMaster our aim is to develop several alternative implementations for each algorithm family. Notice that the alternatives can be either completely different implementations (as in the above sentiment classification example) or just variations of the same implementation, which are differed by setting some algorithm parameters differently.

3 Generic Data Stream Processing Algorithms

Efficiently processing data streams is typically achieved using synopses, i.e., data structures that allow representing the most vital characteristics of the provided data while requiring much less space for their storage/maintenance. The idea is that instead of using the original data to perform (approximate) query processing, we now use the generated synopses.

The use of synopses for processing data streams has been shown essential for current information systems, especially since the majority of these systems need to handle massive data collections. This led to the suggestion and development of several families of synopses. The families of synopses that are mostly related to the context and processing needed in the QualiMaster infrastructure are: (i) histograms: data are separated into buckets (i.e., subsets), with each bucket focusing on computing specific statistics for summarize the data placed in the particular bucket; (ii) sketches: correspond to synopses that provide representative trends in the provided data; and (iii) distributed monitoring: focuses on generated synopses in massively-distributed streaming data, which implies that this family needs to also handle the related communication issues.

Trade-Offs Histograms are considered simple to interpret, which makes programmers to prefer them when developing stream processing algorithms. Dealing with high speed streams of data is more effective with sketches and algorithms based on distributed monitoring. However, the typical situation is that each sketch focuses on answering a single type of query, which means that handling many query types will require incorporating a number of sketches. Finally, processing massively-distributed streaming data is more efficiently achieved using algorithms from distributed monitoring.

In the following paragraphs, we discuss the synopses that are mostly related to the processing required by the QualiMaster infrastructure. Reference [2] provides additional information related to existing families of synopses for data stream processing as well as more details for the families discussed in the following paragraphs.

3.1 Histograms

This family of synopses separate the data of the stream into buckets (i.e., subsets), with each bucket focusing on specific statistics that summarize the data placed in the particular bucket. Actually, the statistics in each bucket can be used to (approximately) recreate the data that were summarized in the particular bucket. To process queries using histograms we need to identify the set of buckets that are related to the specific query.

Histograms have been extensively studied and have been already incorporated and used in various database-related systems. For example, they have been used in database query optimizers for more than twenty years. They can be easily used for processing various types of queries. For instance, we can use one-dimensional histograms for approximately answering queries related to range-sum, i.e., retrieve the number of the stream data with values in a given range. To process more general query types, i.e., involving more than one attribute, we must use multi-dimensional histograms, which is a straightforward generalization of one-dimensional histograms.

The introduced techniques for multi-dimensional histograms can require multiple passes over the stream data or only a single pass, with the latter being more efficient. The basic idea of multi-pass techniques is to start from a single bucket which is then separated into more buckets. For example, [15] starts from a bucket that contains data from various dimensions and then chooses a single dimension to further split the bucket. Similar processing is performed by the approach in [16] but now each bucket is separated into two buckets at each step. To reduce the required processing time of multi-pass techniques, the community has introduced a technique that construct histograms in a single pass [17].

3.2 Sketches

During the past years, we have witnessed extensive development for handling streaming data using sketch techniques. These techniques typically create a compact synopsis for the observed streaming data. This synopsis is vastly smaller than the original data and can be use for answering queries over the original stream data. The latter is achieved by updating the synopsis when new data are observed. Note that we use a sketch for a particular set of queries and that queries are answered by executing a procedure over the defined sketch. We could of course have different procedures for the same query type, with each procedure providing different guarantees.

The sketch techniques are considered especially appropriate for streaming data, since for streaming data we need to handle large amounts of data and continually update the synopsis. For this reason, various sketch techniques have been proposed. Each sketch type focuses on a particular query type, e.g., for estimating the frequency of an item in a data set (the so-called frequency queries, that can be answered by Count-min sketches), or for deciding the existence or absence of an item in a data set (membership queries, for which we can use Bloom filters). For processing financial data streams as required by the QualiMaster infrastructure, we will consider the query types and thus the related families of sketch techniques discussed in the following list:

1. **Membership queries** that return whether a given stream contains a particular item X. Handling this type of query is achieved by sketch techniques from the Bloom filter family. Some examples are the original Bloom filters, Counting filters, as well as Bloom filter extensions.

2. **Set cardinality estimation** that return how many distinct items are contained in a given stream, for example Flajolet–Martin Sketches [6]. This functionality is supported by sketch techniques based on Bloom filter extensions.

3. **Count of arrivals over sliding windows** that return how many items arrived in a requested time, for example during the last 10 seconds. The state-of-the-art data structures for supporting these queries are exponential histograms and randomized waves.

4. **Frequency counts** that return many times a particular item X has been contained in a given data stream. These are by far the most frequently requested queries. They can be supported using Count-min sketches [1] and ECM-sketches (ECM-sketches provide the same functionality over a sliding window constraint).

5. **Self-join and inner-product size estimation** that provide the self-join size of the item frequencies in a data stream. This query type is typically used for constructing efficient query plans in relational databases and is typically required for the full implementation of query planners. Such query types can be supported by the Count-min sketches [1], AMS sketches

[4, 5], and ECM-sketches (ECM-sketches provide the same functionality over a sliding window constraint).

3.3 Distributed Monitoring

More recent efforts have concentrated on scaling stream processing techniques to massively-distributed streaming data. In such settings, we not only need to deal with the strict space and time requirements but also with the related communication aspects. Several communication-efficient streaming tools have been recently proposed for handling a number of challenging query tasks. This, for example, includes distributed tracking of simple aggregates [8], quantiles [10], and complex join aggregates [9].

A state-of-the-art approach for monitoring arbitrary functions over distributed data streams is the geometric method [11, 12]. The geometric method addresses the basic problem of monitoring distributed threshold-crossing queries; that is, monitor whether $f(v(o))<\tau$ or $f(v(o))<\tau$, for any arbitrary, possibly complex, non-linear function f() of a global statistics vector v(o) fragmented over N sites, and a fixed threshold τ. The core idea is that, since it is generally impossible to connect the values of f() on the local statistics vectors to the global value f(v(o)), one can employ geometric arguments to monitor the *domain* (rather than the range) of f(). Since this method relies on geometric insights to enable local inference, it drastically reduces the communication cost monitoring of such functions.

Recent work has extended the geometric approach. For instance, the approach introduced in [13] explains how to make the geometric monitoring more efficient using prediction models. Another example is the approach introduced in [14]. This approach combines the geometric approach with AMS sketch estimators and this allows the efficient tracking of a broad class of complex queries over massive, high-dimensional distributed data streams with provable error guarantees.

3.4 Evaluation Methodology

Evaluating the data stream processing algorithms is performed using metrics that are in two categories: (1) generic metrics and (2) family- or algorithm-specific metrics. The former involves the typical metrics used for evaluating the majority of algorithms, such as the required time for processing queries and scalability tests from testing how the processing algorithms scale to large streaming data.

The metrics used for the second category, i.e., family- or algorithm-specific metrics, aim at evaluating the processing incorporated and followed in the particular algorithm(s). For example, a commonly used algorithm-specific metric for distributed monitoring is for the communication cost and this provides the number of messages transmitted in the network across different thresholds for specific network configurations (e.g., a network with 10 sites).

4 Algorithms for Processing Financial Data Streams

There were numerous attempts in the research community to define systemic risk [102, 101, 98, 97]. The general consensus is that in order to identify systemic risk, some sort of network structure has to be recovered from the available market player data. The network should be later analyzed for potential fallbacks of the market player dependency structure. The main difficulty lies in determination of the most suitable measure of symmetrical and asymmetrical dependency different market players.

A symmetrical dependency does not infer a directional relationship, i.e., one stock price drives another stock price. It only says that there is some sort of relationship between two market players, for example there exists a linear correlation of prices. It is not known whether the directionality of the dependency is critical for systemic risk identification. The symmetrical dependency measures are cheaper to compute, and possibly sufficient to define systemic risk. The following two measures are chosen:

1. Hayashi-Yoshida linear correlation coefficient estimator [7] - Linear correlation coefficient provides a measure of linear relationship between two vectors of data. The Hayashi-Yoshida correlation is a cheap-to-compute estimator of the linear correlation coefficient for asynchronous data.
   The Hayashi-Yoshida estimator can offer better accuracy [7, 99] than approaches based on previous step interpolation, linear interpolation or next step interpolation. This should allow for better estimation of linear symmetrical dependence between different market players. This in turn should allow for more accurate systemic risk identification.

2. Mutual Information - In contrary to the Hayashi-Yoshida method presented in bullet 1, above, mutual information can account for non-linear relationships between two data vectors. It is known that financial data often follows heavy tailed distributions, to capture those relationships the dependency measure needs to take into account higher order statistics [97, 96, 104]. These may be critical for identification of systemic risk, yet come at a cost [97]. Computational complexity of mutual information is quadratic. Furthermore, mutual information relies on either model or model-free estimates of data series probability density function estimations, which can introduce substantial noise in the systemic risk calculations. Data synchronization is also necessary for mutual information calculation.

   Despite the potential drawbacks the non-linearity might be crucial for identification of systemic risk and mutual information metric is included in the QualiMaster infrastructure.

Asymmetrical dependency is usually defined as causality, i.e., one stock price is dependent (not solely) on the past prices of a different stock. If the measure can capture directionality a directed graph can be constructed. There is a strong support in literature that asymmetrical dependencies are important for systemic risk identification [103, 72, 60, 59]. The disadvantage is that the directional metrics are more computationally intensive and for analysis they require computation of directionality in both directions. To understand the impact on the computation cost, one can imagine a dependency matrix. The matrix is non-symmetrical and requires N^2 calculations. In the case of non-casual measures the matrix is symmetrical, and only N(N-1)/2 calculations are required.

The following two measures are chosen to include directionality:

3. Hayashi-Yoshida linear cross-correlation estimator [99] - as in 1. Majority of infrastructure required to calculate 1. is reused, making this metric a very attractive choice. The computation of asymmetric relationship between two market players can introduce significant additional cost.

4. Transfer Entropy [100] - Similar to 2 sharing the disadvantages. It measures casualty of two data vectors for a specified lag. It takes into account non-linear relationship between two data vectors. The computational complexity is cubic, becoming a potential bottleneck for a large number of market players.
   The biggest difficulty lies in probability density functions estimation required for transfer entropy calculations. They are three dimensional, and the result is very sensitive to their estimation. As in the case of mutual information, there are multiple possible approaches.

   Despite the potential drawbacks the non-linearity might be crucial for identification of systemic risk and mutual information metric is included in the QualiMaster infrastructure.

In this section, we describe the algorithm families that will be considered for processing data streams with financial data. These families are (i) correlation matrix estimators, (ii) mutual information, and (iii) transfer entropy. During the first year of the project, we have focused on studying and incorporating algorithms from the 1st family, i.e., correlation matrix estimators. We plan to investigate algorithms in the other two families, i.e., mutual information and transfer entropy, in the following months.

Consequently, in this deliverable we provide a detailed discussion on the family focusing on correlation matrix estimations (Section 4.1). This also includes the algorithms corresponding to the particular family. We then present the mutual information and transfer entropy families (Section 4.2, 4.3) and discuss the algorithms we are currently examining to incorporate in the QualiMaster infrastructure.

4.1 Correlation Matrix Estimators

This family of algorithms measures correlations between the requested market stocks. The correlation between two stocks is a value between +1 and −1 inclusive, where 1 illustrates a positive correlation, 0 illustrates no correlation, and −1 illustrates negative correlation. Since QualiMaster will be typically requested to provide the correlation between a collection of market stocks (i.e., S1, S2, …, Sn), the result of the algorithms in this family is a matrix with each cell showing the computed correlation between two specific stocks (i.e., Si and Sj, with i,j<=n).

4.1.1 Hayashi-Yoshida Correlation Estimator

The first estimator we have considered for the QualiMaster infrastructure is the Hayashi-Yoshida Correlation Estimator [7]. The main advantage of this estimator over the similar estimators found in the literature is that it can be applied directly on the series of prices as they are observed in the stream without any additional manipulation.
The Hayashi - Yoshida Estimator estimates the correlation of unsynchronized streams without the need of their synchronization. Let:

$$P_k^1, k = 0, 1, \dots, n$$
$$P_l^2, l = 0, 1, \dots, m$$

two time-series whose correlation we want to calculate. Let $T^{1,i}, T^{2,j}$ be the observation times of P^1, P^2 respectively. The observation intervals:

$$I^i = (T^{i-1}, T^i)$$
$$I^j = (T^{j-1}, T^j)$$

and the respective increment (*delta*) of the time-series for these intervals:

$$\Delta P^1(I^i) = P_i^1 - P_{i-1}^1$$
$$\Delta P^1(J^j) = P_j^1 - P_{j-1}^1$$

The correlation estimator is defined as:

$$\varrho^{HY} := \frac{\sum_{i,j} \Delta P^1(I^i) \Delta P^2(J^j)_{\{I^i \cap J^j \neq \emptyset\}}}{\sqrt{\sum_i \Delta P^1(I^i)^2} \sqrt{\sum_j \Delta P^1(J^j)^2}}$$

That is, the product of any pair of increments $\Delta P^1(I^i)$ and $\Delta P^1(J^j)$ will make a contribution to the nominator summation only when the respective observation intervals I^i and J^j are overlapping.

Since the estimator is not bound by 1 in magnitude, the bounded transformation is applied:

$$\Phi(x) := (x \wedge 1) \vee (-1)$$

therefore

$$\bar{\varrho}^{HY} := \Phi(\varrho^{HY})$$

4.1.1.1 Incorporation in the QualiMaster Infrastructure

We utilize Hayashi-Yoshida estimator to compute the correlation between every pair of the input streams. These pair-wise correlations form the correlation matrix of the input. To compute the correlations we use sliding (time) windows of configurable size (e.g., a sliding window of one hour) and advance (e.g., an advance of 10 minutes). For each symbol (stream) and at each tick, we maintain information relevant to the latest interval of the stream, its delta and its overlap with the intervals of every other stream. Furthermore, we update (when necessary) the estimator's nominator for every pair of symbols in a rolling fashion. The sum of the squared deltas (denominator) is updated as well.

Whenever a window expires (i.e. when window_size millisecs have passed), we calculate the correlation matrix of the input streams simply by dividing the nominator with the appropriate denominator of each symbol pair. Note that at the time of the correlations calculation, the estimator's nominator as well as most of the denominator have already been computed for each stream pair.

After the correlation matrix calculation takes place, we shift the sliding window by advance seconds. For every stream interval that has now expired ---that is, the interval's end is earlier than the window beginning--- we cancel its contribution to the estimator, and we discard all relevant to it information.

In order to fully utilize the computer cluster provided by QualiMaster infrastructure we parallelize the computation of the correlation matrix among every available machine. This is achieved by the following technique: Let h be the count of available cluster nodes and and s the dimension of the correlation matrix (count of symbols). Then, we partition the correlation matrix to blocks of size $b = \left\lceil \frac{s}{h} \right\rceil$ and we assign the calculation of each block to a node, following a round robin fashion while

iterating the diagonals of the correlation matrix (starting from the second one). Since $Corr(A,B) = Corr(B,A)$ and $Corr(A,A) = 1$ it is only logical that we calculate only the upper triangle of the correlation matrix.

4.1.1.2 Example

Provided the nodes h_0, h_1, h_2 ($h = 3$) and symbols $P_i, i = 0,1,\ldots 9$ ($s = 10$), the size of the blocks that we will partition the matrix into is $b = 4$. To partition the matrix, we traverse the matrix diagonals (starting from the second one), we create square blocks of dimension b and assign each block to a node using round robin. The result is shown in the following matrix where the subscript in each cell denotes the node and the superscript denotes the iteration at which the block was assigned to the corresponding node.

	P^0	P^1	P^2	P^3	P^4	P^5	P^6	P^7	P^8	P^9
P^0	-	h_0^0	h_0^0	h_0^0	h_0^0	h_0^1	h_0^1	h_0^1	h_0^1	h_2^1
P^1	-	-	h_0^0	h_0^0	h_0^0	h_0^1	h_0^1	h_0^1	h_0^1	h_2^1
P^2	-	-	-	h_0^0	h_0^0	h_0^1	h_0^1	h_0^1	h_0^1	h_2^1
P^3	-	-	-	-	h_0^0	h_0^1	h_0^1	h_0^1	h_0^1	h_2^1
P^4	-	-	-	-	-	h_1^0	h_1^0	h_1^0	h_1^0	h_1^1
P^5	-	-	-	-	-	-	h_1^0	h_1^0	h_1^0	h_1^1
P^6	-	-	-	-	-	-	-	h_1^0	h_1^0	h_1^1
P^7	-	-	-	-	-	-	-	-	h_1^0	h_1^1
P^8	-	-	-	-	-	-	-	-	-	h_2^0
P^9	-	-	-	-	-	-	-	-	-	-

4.1.1.3 Preliminary Evaluation

To evaluate our implementation we used FOREX symbols data provided by Spring API. The created dataset is a subset of the actual data collection available to QualiMaster, and it comprised of 143 symbols that result in 10153 correlation pairs. The particular dataset was selected by the QualiMaster application partners in order to enable easy and preliminary quality evaluation. We defined the window size to 1 hour and reported the correlation matrix each second. That means that our estimator used tick data from the last hour to calculate and report the correlations every second. Our implementation used Apache Storm framework running on a 3-node computer cluster. The average calculation time for estimating all pair-wise correlations is about 280ms. The correlation results were manually verified with respect to quality by the QualiMaster application partners.

4.1.2 Pearson Correlation Estimator

Pearson coefficient is a measure of linear correlation between two random processes. In contrast to Hayashi-Yoshida estimator, Pearson estimator can only be used with synchronized data; therefore techniques as previous tick interpolation have to be applied to the data before using it.

Let $X_i(t), X_j(t)$two synchronized time-series and $X_k^i = X_i(t_k) - X_i(t_{k-1})$, $\Delta X_k^j = X_j(t_k) - X_j(t_{k-1}), k = 1, 2, \ldots, n$ their values sampled at regular intervals. Then the Pearson correlation coefficient is defined as:

$$\rho_{i,j}^P = \frac{\sum_{k=1}^{n}\left(\Delta X_k^i - \sum_{k=1}^{n}\Delta X_k^i\right)\left(\left(\Delta X_k^j - \sum_{k=1}^{n}\Delta X_k^j\right)\right)}{\sqrt{\sum_{k=1}^{n}\left(\Delta X_k^i - \sum_{k=1}^{n}\Delta X_k^i\right)^2 \sum_{k=1}^{n}\left(\Delta X_k^j - \sum_{k=1}^{n}\Delta X_k^j\right)^2}}$$

The values of the estimator are bounded by -1 and +1 inclusive, where 0 means no correlation, +1 total positive correlation and -1 total negative correlation.

4.1.3 StatStream Correlation Estimator

StatStream [53] implements a data stream monitoring system, for computing a variety of single and multiple stream statistics in one pass with constant time (per input) and bounded memory. Multi-stream statistics such as synchronous as well as time-delayed correlation and vector inner-product are computed in a continuous online fashion. This means that if a statistic holds at time t, that statistic will be reported at time t + v, where v is a constant independent of the size and duration of the stream. For any pair of streams, each pair-wise statistic is computed in an incremental fashion and requires constant time per update. This is done using a Discrete Fourier Transform approximation. The approximation has a small error under natural assumptions. Even when the data streams are monitored over sliding windows, no revisiting of the expiring data streams is needed.

The system distinguishes three time periods:

- timepoint - the smallest unit of time over which the system collects data, e.g. second.
- basic window - a consecutive subsequence of timepoints over which the system maintains a digest incrementally, e.g., a few minutes.
- sliding window - a user-defined consecutive subsequence of basic windows over which the user wants statistics, e.g. an hour. The user might ask, “which pairs of stocks were correlated with a value of over 0.9 for the last hour?”

4.1.3.1 Statistics to Monitor:

Consider the stream $s_i = 1, \ldots, w$. Possible statistics the system can monitor are:

1. Single stream statistics, such as average, standard deviation, best fit slope.
2. Correlation coefficients

$$corr(s,r) = \frac{\frac{1}{w}\sum_{i=1}^{w} s_i r_i - \bar{s}\,\bar{r}}{\sqrt{\sum_{i=1}^{w}(s_i - \bar{s})^2}\sqrt{\sum_{i=1}^{w}(r_i - \bar{r})^2}}$$

3. Autocorrelation: the correlation of the series with itself at an earlier time.
4. Beta: the sensitivity of the values of a stream *s* to the values of another stream *r* (or weighted collection of streams). For example, in financial applications the beta measures the risk of a stock. A stock with a beta of 1.5 to the market index experiences 50 percent more movement in price than the market:

$$beta(s,r) = \frac{\frac{1}{w}\sum_{i=1}^{w} s_i r_i - \bar{s}\,\bar{r}}{\sum_{i=1}^{w}(r_i - \bar{r})^2}$$

4.1.3.2 Maintaining Statistics over Sliding Windows

To compute the statistics over a sliding window, the system maintains a synopsis data structure for the stream to compute the statistics rapidly. The sliding window is equally divided into smaller windows called “basic windows”, in order to facilitate the efficient elimination of old data and the incorporation of new data. It keeps digests for both basic windows and sliding window. For example, the sum over the sliding window is updated as follows:

$$\sum_{new}(s) = \sum_{old}(s) + \sum S[k] - \sum S[0]$$

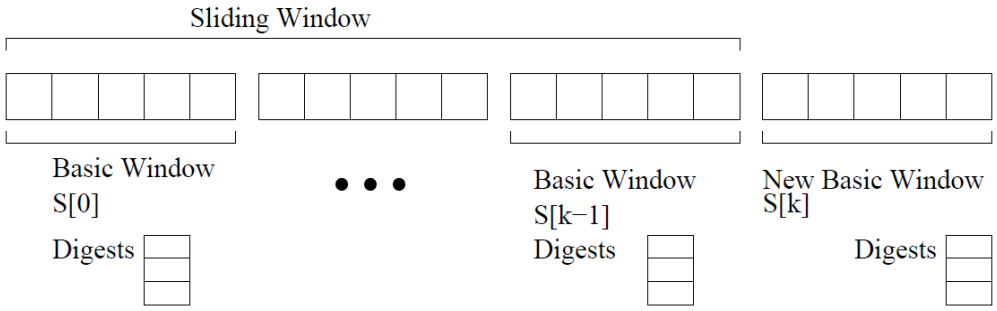

Figure 6: Sliding windows and basic windows (image is from [53])

The size of the basic window is important because it must be possible to report all statistics for basic window i to the user before basic window i+1 completes (at which point it will be necessary to begin computing the statistics for window i+1).

4.1.3.3 Computing Pair-wise Correlations

The efficient identification of highly correlated streams potentially requires the computation of all pairwise correlations and could be proportional to the total number of timepoints in each sliding window times all pairs of streams. The system makes this computation more efficient by (1) using a discrete fourier transform of basic windows to compute the correlation of stream pairs approximately; (2) using a grid data structure to avoid the approximate computation for most pairs.

The system computes the discrete Fourier transformation over each basic window for each stream and then updates the sliding window Fourier approximation by using the existing basic window digests. At this stage, every stream is described by a set of Fourier coefficients. Using DFT properties, the correlation between two streams is shown to be equal to the Euclidean distance of their normalized Fourier coefficients. Hence, the system can use a Grid structure and hash each stream *s* to a certain cell based on a subset of its first few Fourier coefficients. When the input query is to find all streams that are correlated with stream *s* with a correlation higher than a certain threshold *t*, then only streams hashed in neighboring cells in the grid structure need to be examined as they consist of a super set of the true correlated streams (false positives). No false negatives are introduced.

4.1.3.4 Incorporation in the QualiMaster Infrastructure

The StatStream system has been implemented into a distributed Storm architecture in order to utilize the QualiMaster infrastructure. The system can scale depending on the number of available nodes in the cluster. Each stream is hashed to a certain node of the cluster. The load is partitioned

equally among nodes. Each node needs to compute single stream statistics (e.g. moving average, fourier coefficients, etc.) over basic windows as described. Then, it needs to hash each stream to the grid structure based on its Fourier coefficients. The grid structure is currently implemented into one single node for simplicity but can easily be partitioned and scaled based on available nodes.

4.1.4 Correlation Estimator by using the Processing Methodology of StatStream

The StatStream framework can be altered to incorporate different algorithms with similar "philosophy". StatStream itself uses DFT as an approximation and the Grid Structure as the indexing structure. Different combinations of approximation technique/indexing structure (e.g., curve fitting/R-tree index) can be incorporated into the system and replace the existing ones depending on algorithmic needs. Note that StatStream's basic idea can also be used in different use cases and is not restricted in finding pair-wise correlations. Basically all singled-stream statistics can be easily computed in an online fashion using StatStream. Additionally, multi-stream statistics can also be incorporated computed as long as the approximation mechanism and the indexing structure used can be efficient for the computation of the metric the streams are tested against. For example, in the correlation computation setting introduced in Section 4.1.1, the metric to be examined is the pair-wise correlation, which is in turn reduced to the Euclidean distance of the DFT coefficients (approximation mechanism). Hence, a Grid Structure (indexing) is very efficient in this setting as it is an indexing structure optimized against the Euclidean distance metric.

4.1.5 Trade-Offs Between the Algorithms of the Family

A typical method for measuring co-movements between stocks is the Pearson Correlation Estimator (Section 4.1.2). However, as this estimator is influenced by several effects, such as the non-synchronicity of the transactions [3], other estimators are preferred. The Hayashi-Yoshida Correlation Estimator (Section 4.1.1) is one of the estimators preferred instead of Pearson, since it can be applied directly on the series of prices as they are observed in the stream without any additional manipulation.

StatStream (Section 4.1.3) can be used with limited computing resources. It optimizes the processing of query "Given stream id X, which are the streams that have a correlation more than T with X?", which is basically a subset of correlations returned by the other estimators. Note that StatStream a framework that can work with external estimators (Section 4.1.4) and can, thus, be also used for processing the Pearson or Hayashi-Yoshida Correlation Estimator with limited computing resources.

4.2 Mutual Information

Mutual information is frequently used for retrieving the complete dependence structure between two time series (including nonlinear dependence). It basically computes the data shared between the given time series, which (intuitively) can be seen as measuring how much knowing variable X (or variable Y) reduces uncertainty about variable Y (or variable X).

Given two time series X={$x_1, x_2, \dots, x_n$} and Y={$y_1, y_2, \dots, y_n$}, then the mutual information of X and Y is computed by the following formula [57]:

$$I(X;Y) = \int_Y \int_X p_{XY}(x,y) ln \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)} d_x d_y$$

With symbol $p_{XY}(x, y)$ we denote the joint probability density function between X and Y, and symbols $\mathrm{p_X(x)}$ and $\mathrm{p_Y(y)}$ denote the marginal probability density functions.

The main challenge for evaluating the mutual information formula is on computing the probability density function, and more specifically on estimating this probability. To deal with this issue, several methodologies computing such an estimation have been suggested. Some examples (explained in [57]) are the kernel density estimators, the k-nearest neighbors, the Edgeworth approximation of differential entropy, and adaptive partitioning of the XY plane.

For incorporating a mutual information family in the QualiMaster infrastructure we will study existing estimators aiming at computing the probability density function, for example the ones mentioned above such as adaptive partitioning and the kernel density estimators. This is work that we have recently started and plan to progress on it during the following months.

4.3 Transfer Entropy

Transfer entropy quantifies the amount of information transfer from one variable to another variable, and it is considered as a valuable method for detecting the coupling strength between two time series [22]. It is denoted as $\mathrm{T_{Y\rightarrow X}}$, where X and Y are two time series, i.e., X={$\mathrm{x_1, x_2, \dots, x_n}$ } and Y={$\mathrm{y_1, y_2, \dots, y_n}$ }. Given X and Y, then transfer entropy from Y to X, i.e., $T_{Y\rightarrow X}$, is defined as the average information contained in the source Y about the next state of the destination X that was not already contained in the destination's past.

Let us now assume that the two time series X and Y are first-order Markov processes. Then the transfer entropy from Y to X is given by the following formula:

$$T_{Y\rightarrow X} = \sum_{x,y} p(x_t, x_{t-1}, y_{t-1}) log_2(\frac{p(x_t|x_{t-1}, y_{t-1})}{p(x_t|x_{t-1})})$$

As with the computation of the mutual information (Section 4.2), the main challenge when processing the above formula for transfer entropy is estimating the probability density function. To deal with this problem, researchers estimate the probability density function using different methodologies. Such alternative methodologies will become the algorithms that will be included in the QualiMaster infrastructure for the specific family. Some examples of alternative algorithms for estimating the probability density function are introduced in [22].

As explained at the beginning of Section 4, we have recently started investigate algorithms in this family. With respect to algorithms for the transfer entropy family, we are currently studying the kernel density estimation and adaptive partitioning. More specifically, the probability density function in the kernel density estimation is computed by summing individual distributions centered at each data point, with the shape of the individual distributions denoted by a chosen kernel. Adaptive partitioning estimates the probability density function by applying the Darbellay-Vajda partitioning algorithm to transfer entropy estimation.

5 Crawling and Pre-processing of (Social) Web Data Streams

In addition to the financial data streams, the QualiMaster infrastructure will support the processing and analysis of data streams from the web in general and from social media platforms and online news feeds in particular. As discussed earlier in Section 22.2, the goal is to extract useful information and signals from the very noisy and unstructured web data in real-time. More specifically, we want to process and analyze different online streams from social micro-blogging platforms and news feeds to produce high quality and well-structured streams (in the form of time series) that can be further exploited by analytic and visualization tools as illustrated in the example of Figure 7. In this example, the input is unstructured (or semi-structured) and noisy web data streams such as textual content of online news articles, posts on social platforms such as Twitter and YouTube or graph structures of social networks. The output stream in this example is a time series that consists of tuples, where each tuple has a clean and pre-defined structure such as: a stock name, a timestamp, the number of posts containing the stock name and the percentage of positive posts.

This process involves several steps. The first step to crawl and filter the web data streams. The next step is to perform NLP pre-processing and annotation tasks on the textual content of the stream (such as tokenization, Named Entity Recognition etc). Finally, analysis methods are applied on the pre-processed text such as sentiment evaluation (i.e. objectiveness or polarity), event detection or user expertise analysis.

In this section we discuss the first two steps on crawling, filtering and pre-processing input web data streams and in the next section we present and describe in more details the web data analysis methods that will be considered in QualiMaster to support the analysis of financial data streams in general and the systemic risk analysis in particular.

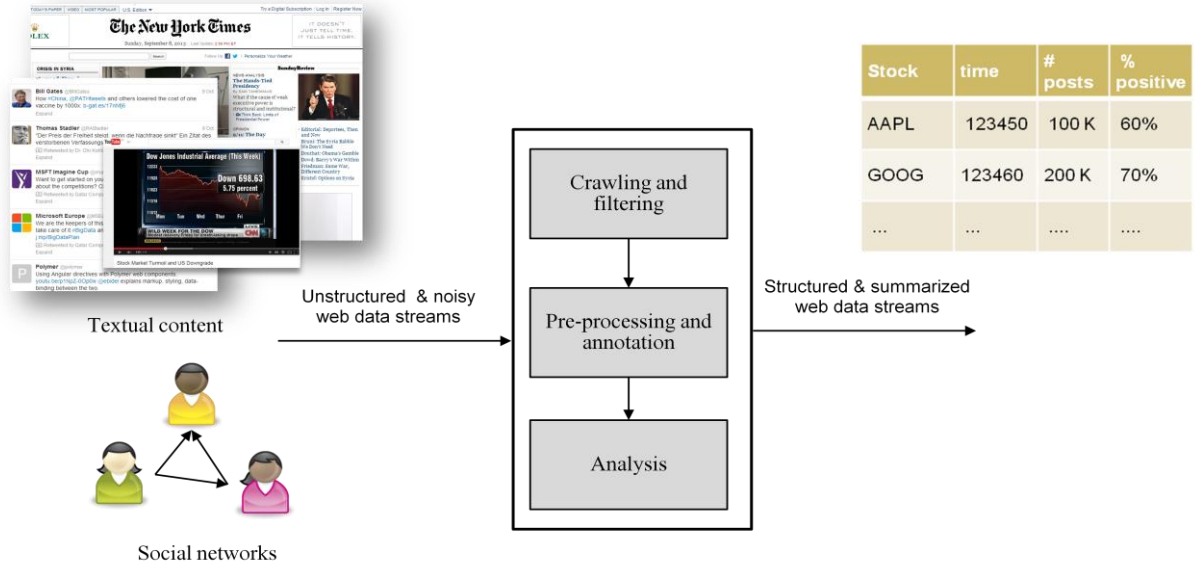

Figure 7: Making sense of Social Web Data

Table 1 gives an overview on the main social web sources that are currently being considered in the project. During the course of the project, we will investigate further sources as well, when possible.

Source	Description
Twitter	The micro-blogging platform Twitter provides a public Streaming API that can be used to connect to and receive the live feed of posts (tweets) in real-time. In QualiMaster we are using the following endpoints: • The filtered stream, which returns public statuses that match one or more filter predicates (https://stream.twitter.com/1.1/statuses/filter.json) • The sample stream, which a 1% random sample of all public statuses (https://stream.twitter.com/1.1/statuses/sample.json) For a detailed overview, please refer to: https://dev.twitter.com/streaming/overview
StockTwits	StockTwits is a financial short message communications platform for the financial and investing community. StockTwits provides a REST API access to their data. http://stocktwits.com/
SeekingAlpha	Seeking Alpha is a financial blog platform for investment research where insights are provided by investors and industry experts directly, such as suggestions for buy/sell stock candidates. It can be accessed via RSS feeds. http://seekingalpha.com/
News RSS feeds	We collect news RSS-feeds from the top international news agencies. The news articles are then fetched and stored locally for later use.
Yahoo News	Yahoo! News is one of the leading news aggregators, which attracts a large number of users who follow news stories around the world and comment on them with over 138 million distinct yearly visitors. http://news.yahoo.com/
YouTube	YouTube is the most popular video sharing site, and an online community centered around user-generated, comprises 60% of the videos watched on-line. In 2014 YouTube reported having over 1 billion unique visitors each month and over 100 hours of videos being uploaded each hour. https://developers.google.com/youtube/analytics/
Yahoo Finance Earnings Calendar	Contains information indicating scheduled events when companies announce their quarterly, annual earnings. Some entries also contain the estimated sentiment in terms of a score (pos/neg). http://biz.yahoo.com/research/earncal/today.html
Wikipedia	Wikichanges is a node.js library for getting an edit stream from the 37 major language Wikipedias. The Wikipedia MediaWiki installations are configured to log changes in specific IRC channels. wikichanges joins all these channels, listens for updates, which it then parses, and sends as JavaScript objects to a callback of your choosing. It is available in Github (https://github.com/edsu/wikichanges). An example application that utilizes Wikichanges to visualize the edits in real-time is Wikistream: http://wikistream.wmflabs.org/

Table 1: Main web data sources that are currently considered in QualiMaster

5.1 Adaptive Crawling

There is a tremendous amount of new data that is being created on the web every day. In addition to online news articles that are published by news agencies, high volumes of user-generated content are being published in blogging and micro-blogging platforms. Moreover, user ratings and comments on content published by organizations or by other users has become a very common and popular behavior in Web 2.0. Many of the popular Web 2.0 platforms (like the ones listed in Table 1) provide public and/or commercial interfaces for querying and downloading content. However, finding relevant content with respect to a given query (e.g. a specific market player or stock symbol) is not trivial. Keyword based search has the limitation that only documents or posts that explicitly mention the keyword are returned, which leads to low recall. The goal of the adaptive crawling is to maximize the completeness (i.e. recall) and the relevancy (i.e. precision) of the collected data. This is similar to the idea of query expansion with relevance feedback in information retrieval.

Figure 8 gives a high level overview on how adaptive crawling is realized in QualiMaster. Staring with a user query (e.g. stock name, or market player name) as a seed, query expansion is achieved in two ways:

1- Based on knowledge bases: one basic approach using DBpedia knowledge base (http://dbpedia.org), for example, to expand the set of query terms could be by making use of the "owl:sameAs" property in order to collect alternative names for a certain company. Additionally we can use properties like "dbpedia-owl:manufacturer of" and get names of products which are produced by a certain company. We are currently investigating this approach and the possibility to construct an initial domain knowledge base that includes information on market players, their organizations, products, stock symbols etc. from DBpedia (see for example the information on Apple Inc. at http://dbpedia.org/page/Apple_Inc. and http://en.wikipedia.org/wiki/Apple_Inc., which includes links to the stock symbol of the company, the stock exchange market, its products etc.) In addition, manual enriched of the initial knowledge base by domain experts will be considered as well.

2- Based on statistics on keyword co-occurrences: This method computes the co-occurrence of terms with the initial user query terms inside the retrieved documents or posts. The most frequently co-occurring terms are then used to expand the query terms. The purpose of this method is to capture new terms that are connected to emerging topics or events, which might not be covered by the user's initial query. For instance, in the case of Twitter stream, this method can detect Hashtags that frequently used in posts that include the user's query terms.

Unlike classical information retrieval, relevance feedback in this case is not based on user input. Instead, we automatically measure the relevance of the collected posts based on their content. The goal of this step is to filter out posts that are noise. A post is considered noise if its content is off topic, e.g. is on arts or sports when the topic is financial markets. Another reason for classifying a post as noise or irrelevant could be the language or writing style of the post, which might be too different than posts that are classified as relevant and on topic. We use ground truth datasets to characterize relevant posts and identify irrelevant ones. Both supervised and unsupervised methods can be applied to classify the posts using the ground truth datasets. Moreover, by estimating the relevance of the retrieved posts we can optimize the selection of the query expansion terms by ranking them based on the number of relevant posts that include each term.

In the rest of this section we describe the initial implementation of adaptive crawling in QualiMaster, in particular, for crawling Twitter public streams using the filter endpoint, which returns only tweets that include a given set of terms.

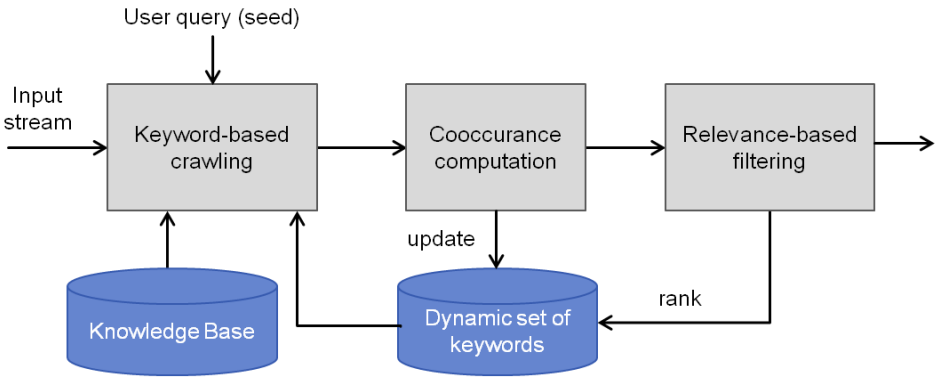

Figure 8: Adaptive crawling of web streams in QualiMaster

5.1.1 Query Expansion for Crawling Twitter Stream

Since early 2013, we continuously collect and archive data from Twitter using its public sample stream API, which provides a random sample (about 1%) of the whole set of tweets posted on the platform in real time. In addition to this collection, we started in 2014 to use the focused crawling approach in order to increase the amount of tweets that are directly relevant for the financial domain using Twitter's public filter stream API. Using this API users can provide a list of filter terms to the API when the connection is established, which will then be used by Twitter interface to filter the tweets based on those terms. As a result, whenever a tweet that includes at least one of the filter terms is posted it will be sent to the receiver in real-time via the established connection.

The terminology used in Twitter is highly dynamic; especially the use of so called Hashtags is very dependent on events. For instance, when a certain event occurs Twitter users agree relatively fast on the use of a certain Hashtag for marking tweets as related to this event.

We performed a series of experiments for analyzing the opportunities and difficulties when working with a dynamic set of keywords for filtering the Twitter stream. The main challenges here are to decide when a new keyword should be selected for the stream filtering and when a keyword should be removed from the stream filter. For our experiments we used sliding windows of different sizes. While a larger window allows the detection of slowly evolving new Hashtags it also reacts slower to fast changes. For our basic setup we decided to add a new keyword or Hashtag if it occurs within a certain percentage of the collected messages. If we increase the window size we have to reduce this percentage because otherwise no new Hashtags will be added to the stream. Selecting the right window size and the correct percentage for a keyword to be added to stream is the most difficult part for an automatic keyword adoption in such a scenario. In our experiments we realized that too small windows are very vulnerable for spammers. If a spammer sends hundreds of messages that contain a keyword we are interested in, within a few seconds our sliding window gets filled with these messages and all other Hashtags that are inside the messages are part of a large percentage of the sliding window. To reduce this problem we tested methods where only distinct messages are considered or only messages from distinct users are taken into account. Alternatively we can increase the window size. With a larger window size it becomes very unlikely that spammers fill up the whole window and due to that the method becomes less vulnerable to spammers.

5.1.2 Relevance-based Filtering of Twitter Stream

When collecting from Twitter data related to certain market players by using the query expansion strategy described in previous section, we always face a tradeoff between the number of tweets we get and the relevance of these tweets to the market players. Therefore, we need to automatically estimate the relevance of the collected tweets in order to balance quantity (i.e. number of retrieved tweets) and quality (i.e. relevance of the retrieved tweets to the user query).

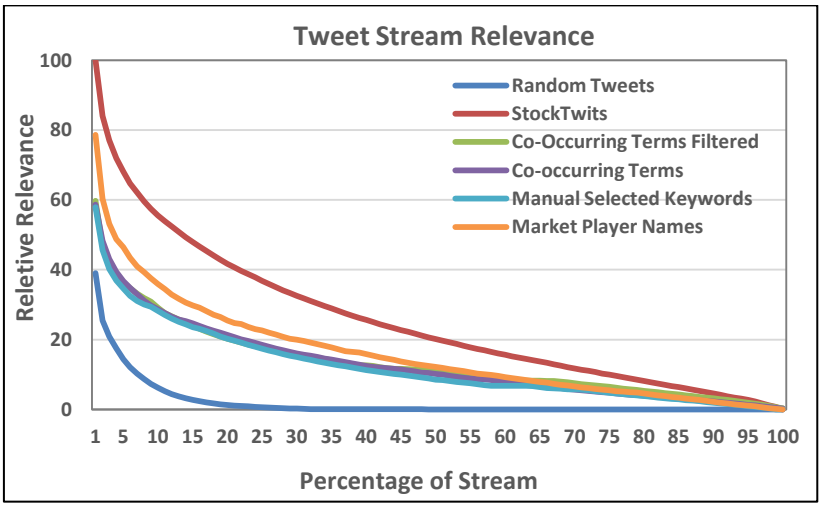

Figure 9: Relevance estimation of collected tweets

Defining whether a tweet is relevant or not is not an easy task. As described earlier, our relevance estimation approach is based on the use of ground truth datasets. Therefore, we collected messages from the StockTwits (http://stocktwits.com) platform using its public API to serve as ground truth for financial relevant tweets. We decided to use this platform because it has similar characteristics to the Twitter platform including its interface and communication style (e.g. the use of Hashtags and Cashtags for stock symbols, the restriction on message lengths, the re-tweeting and user mentioning style etc). An advantage of this platform is that it is dedicated for the posts on topics related to the stock markets. Therefore, our assumption is that a large number of messages posted at this platform are of high relevance to the financial domain. Choosing a large set of messages from this platform allows us to build models which can measure the relevance of a tweet based on its similarity to messages from the ground truth dataset.

For calculating the relevance of the collected tweets we used a Tf/IDF based relevance measure. The results of these experiments are shown in Figure 9. As Ground truth we used a set of 100.000 messages from the StockTwits platform. From these messages we extracted the most relevant keywords using their Tf/IDF values. Using this set of keywords as query we calculated the relevance of different sets of collected tweets. For comparison we used the following strategies:

- Random Tweets: A set of 100.000 randomly collected tweets from the streaming API. These tweets serve as a very simple baseline.
- Manual Selected Keywords: This stream was generated using a set of keywords that are relevant for the stock market. The keywords were selected based on general observation of stock related tweets, such as mentions of exchange markets or stock indices.
- Stock Names: This is a stream using a set of 2.000 market player stock symbols from the NYSE. Examples of market player names are “$AAPL” for Apple or “$AMZN” for Amazon.

- Co-Occurring Terms: This stream is collected using the top 100 co-occurring keywords from the market player based stream without further filtering. Using a stream like this provides a high number of tweets of which many are relevant for the financial sector.

The comparison of the collected tweets shows us that compared to a random sample of tweets all presented methods collect tweets which are of much higher relevance at the cost of recall. For instance, the use of selected keywords that are related to the financial domain leads to higher relevance but lower recall. Selecting tweets that explicitly mention stock names performs best in terms of relevance but worse in terms of volume of the retrieved tweets. The collection of tweets based on co-occurring terms allows the collection of a large amount of tweets with still a relatively high relevance. Table 2 shows the normalized ([1-0] scale) average relevance of the different Twitter filtered streams.

Method	Random Tweets	Manual Selected Keywords	Stock Names	Co-occurring Terms
Relevance	0,0775	0,4792	0,6264	0,513
Tweets per Day	~500 Million	~3.000.000	~20.000	~ 15.000.000

Table 2: The average estimated relevance of tweets based on different filtering strategies

5.1.3 Conclusion

The methods presented in this section allow the collection of different sets of tweets with different quality parameters. Overall the collection of tweets is in most cases a tradeoff between the completeness and the relevance of the collected tweets. Methods like the relevance based filtering will allow us to select parameters that provide a good quality of the collected messages together with an acceptable completeness level.

5.2 Text Pre-processing and Annotation

Extracting relevant information from the collected data and adding additional meta information to the data is one of the first major steps that has to be done in order to give a meaning to the data and using it in the following processing steps. The main components for the information extraction are text pre-processing steps like the tokenization and the Part of speech tagging (POS). Another set of methods is aiming at detecting Named Entities (NER) within the analyzed text. Once entities are detected additional methods can link these entities to external knowledge bases like Wikipedia or DBpedia.

5.2.1 NLP Processing and Named Entity Recognition

One of the available tools for NLP is the IXA pipeline [89] (http://ixa2.si.ehu.es/ixa-pipes/) developed by the IXA NLP Group (http://ixa.si.ehu.es/Ixa), which has been developed in the context of the European projects OpeNER (http://www.opener-project.eu/) and NewsReader (http://www.newsreader-project.eu/). In NewsReader the IXA pipes are used for tokenization, part of speech tagging, named entity recognition and other NLP processes. Many of these processes are also required within the QualiMaster Project.

The tools contain a variety methods that be used in our scenario for the processing of the gathered messages. One of the major advantages for using the IXA pipeline in the QualiMaster project is the modular structure of the tools. The different modules of the framework can be changed so that different Quality parameters can addressed in different situations. Within the framework itself there are also parameters that can be changed in order to adopt the framework. Additionally the data centric architecture makes the different modules independent and allows therefore the use of other toolkits where required. The reported performance values of the tools also sound promising for the use in a real time scenario. The modules use the so called KAF format for the in- and output. This format well defined and allows us to integrate new modules in the pipelines. The methods used by the POS tagging and NERC are maximum Entropy and Perceptron.

At the moment the IXA pipeline offers 5 different pipes for the processing of the data:

- Tokenizer
- POS Tagger & lemmatizer
- NERC
- Constituent parser
- Coreference resolution

This set contains most of the methods we are planning to use when processing the social web data. Beside the integrated tools the IXA pipes include a set of third party tools like Named Entity Disambiguation and Wikification using DBpedia Spotlight. We are planning to compare all of these tools in order to select the best methods or tools for different situations.

In the area of NER a variety of different tools and methods are available. A large set of these tools use Machine learning algorithms for the detection of entities. Additionally some tools use external knowledge bases which in some cases also allow the linking of the entities.

In the area of entity linking we analyzed two different tools and their performances, namely the Illinois Wikifier (http://cogcomp.cs.illinois.edu/page/software_view/Wikifier) and the Wikipedia Miner (http://wikipedia-miner.cms.waikato.ac.nz/). Booth tools allow the linking of detected entities to Wikipedia Pages and due to that the connection of the annotated messages to a large knowledge base. This additional knowledge, which links messages and Wikipedia topics can be used in several scenarios but, compared to standard named entity recognition, requires more memory and computational resources. Both methods use machine learning algorithms for the annotation process. The Illinois Wikifier uses a SVM while Wikipedia miner uses Bagged C4.5 classifier. Detailed descriptions of the two toolkits can be found in [87], [88] and [89].

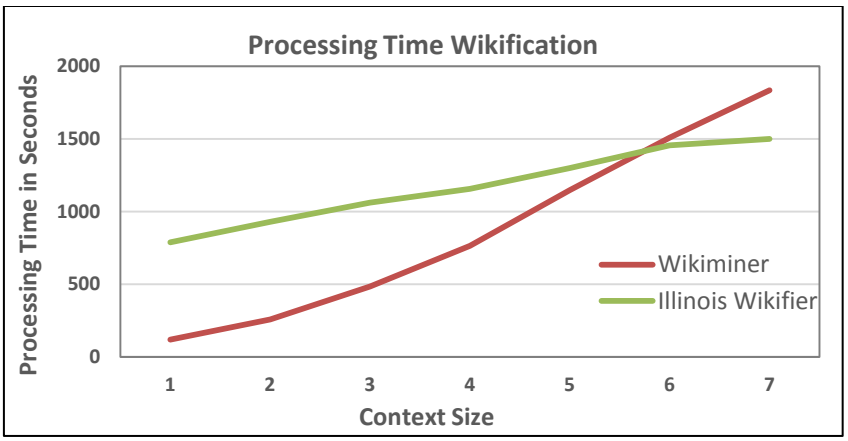

Figure 10: Performance comparison of Wikiminer and Wikifier

We started with a series on experiments that compared the performance of the two tools when the size of the annotated text is changing. For this scenario we choose to select a number between 1 and 7 tweets. For each run we annotated 10000 tweets related to the financial domain. We can see that the Wikipedia miner performs better when the text is shorter but the processing time increases relative strong when more tweets are processed in one run. The reason for processing more than one tweet per run is based on the fact that this context improves the quality the annotations. In the case of the Wikipedia miner the algorithm uses this context for disambiguation and entity linking. A behavior like the one observed in these experiments can lead to an adoption of the pipeline when different texts have to be processed. Beside the processing time also the quality of the annotation is important.

The quality of the produced annotations has been evaluated in [90] and the results are shown in Table 3. These results are based on the annotation of a public available dataset from the Making Sense of Microposts Challenge 2014 (http://www.scc.lancs.ac.uk/microposts2014/). In contrast to long texts the annotation of Microposts is a much harder task because of special language used in the messages.

	F1				Macro AVG			Micro AVG		
Model	**LOC**	**MISC**	**ORG**	**PER**	**P**	**R**	**F1**	**P**	**R**	**F1**
Illinois Wikifier	**0.34**	**0.09**	**0.46**	**0.68**	**0.44**	0.38	**0.39**	**0.63**	0.50	**0.55**
Wikipedia Miner	0.29	0.04	0.29	0.67	0.28	**0.46**	0.32	0.32	**0.57**	0.41

Table 3: The results of the annotation quality assessment reported in [90]

We can see that the Illinois Wikifier outperforms the Wikipedia Miner in most of the categories. On the other hand our experiments showed that the Wikipedia Miner can outperform the Wikifier in terms of processing time. This can lead to adoptions of the pipeline where the Wikipedia Miner is used when low latency and high throughput is more important and the Wikifier when a high accuracy is more important.

5.2.2 Topic Detection

In the area of topic detection we tested two different methods so far. One is LDA the other is a method using the Wikipedia category graph for assigning topics to entities and therefore to documents containing entities. The use of topic detection methods may allow is in the project to base our predictive algorithms only on sets of documents or messages which are related to a given topic, or alternatively to adjust the weight of documents based on their topics. Besides that the information about the topic of messages or documents can serve as feature and show temporal changes in the publishing behavior of certain information sources.

5.2.2.1 Entity-based Topic Detection

For the assignment of topic based on entities we use a background knowledge base namely the Wikipedia category graph or alternatively the category information stored in DBpedia. We can use the categories assigned to the entities and merge these to topics in order to buildtopics based on the entities. The process of merging topics is possible due to the structure of the Wikipedia category graph which contains parent and child relations for categories. Using this structure allows

the merging of very special topics to more general topics up the level of the Wikipedia top categories. With DBpedia we could use similar information the category relations are stored within dcterms:subject and could be used in the same way as in Wikipedia. Additionally we could use attributes like rdf:type which link to knowledge bases like yago (http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/). The use of entities together with external knowledge bases offers an alternative to use of LDA for Topic Detection.

5.2.2.2 Latent Dirichlet Analysis (LDA)

Latent Dirichlet Analysis (LDA) can be employed for uncovering the latent semantics of corpora. Latent Dirichlet allocation identifies a given number of |Z| topics within a corpus. The number z is the most important parameter for LDA because itdetermines the granularity of the resulting topics. In order to find the latent topics, LDA relies on probabilistic modeling. This process can be described as determining a mixture of topics z for each document d in the corpus, i.e., P(z|d), where each topic is described by terms w following another probability distribution, i.e., P(w|z). By applying LDA we are able to represent latent topics as a list of terms with a probability for each term indicating the membership degree for the topic. Furthermore, for each document in our corpus we can determine through topic probabilities P(zj|di) to which topics it belongs and to which degree. The model needs to be trained on an example dataset and can be applied to any document later assigning the probabilities of topics to occur in that document.
An LDA topic is summarizing a set of coherent terms and this can help to detect events [85, 86] by counting their joint bursts as event evidence and aggregating them. Furthermore, it can also be useful for clustering users according their expertise regarding particular topics, or to describe their expertise with a set of coherent terms. Whilst topic detection using an existing LDA model is rather computationally inexpensive and can be performed in real time applications, training of the model is very costly (although parallelizable) in terms of time and hard/software resources, it requires high amount of training documents and can take up to several hours for a good quality LDA model. Training once a day can be suitable for detecting topics that do not change in time very quickly, however a hardware implementation that can be trained much faster would open a promising direction for boosting the process and enable real time adaptation also for the quickly changing topics and streams with high topical diversity.

6 Methods for (Social) Web Data Analysis

This section discusses a set of methods that are being considered in QualiMaster for processing and analyzing web data streams, including both news feeds and social media streams. More specifically, methods for sentiment analysis of (micro) blogs, user and social network analysis, event detection and entity network extraction will be considered in QualiMaster. The selected methods are the results of a study of the state-of-the-art and recent related work that have shown the potential of extracting useful and meaningful signals from web content in general and social media in particular (see [21], [23], [20], [24], [19] for example).

Moreover, a set of use cases that describe the use of the proposed methods in supporting financial risk analysis on the stock markets have been defined in collaboration with WP1 and in light of the user requirements analysis that have been conducted described in D1.1 and D1.2.

Figure 11 gives an example scenario for combining the different methods that will be part of the QualiMaster infrastructure for processing and analyzing web data streams. Two types of input streams are considered in this scenario: streams for social platforms such as Twitter and StockTwits and News RSS feeds. The overall task in this scenario is to monitor the opinions of the users of these social platforms in real-time towards the developments in the financial markets in general or towards specific market players. Therefore, the micro-blogs that are posted by the users of the social platforms are retrieved online by connecting to their public APIs. Next, the textual content of the posts are pre-processed and annotated using NLP tools (as described in Section 5.2). Then the annotated posts are filtered based on user preferences (e.g. focusing on posts of some language or posts originating from some region or country). The input streams can also be filtered based on some events, which are detected in real-time by an Event Detection module that monitors (in parallel) the social media streams and news RSS feeds. Next, an additional filtering step can be applied on the posts based on the author of the post. This step utilizes a so-called User Index that is updated continuously by a User Analysis module. The User Analysis module incrementally builds and maintains profiles of users of social platforms in the User Index in order to model their influence and their expertise based on their posts and their network. Posts can then be filtered based on the user's profile in order to filter out spam and irrelevant content. Finally, the sentiment of the posts is evaluated using sentiment analysis techniques to identify the polarity and opinion of the author towards the target query. The results of this step are then aggregated over time to build a time series that shows the development of the opinions over time.

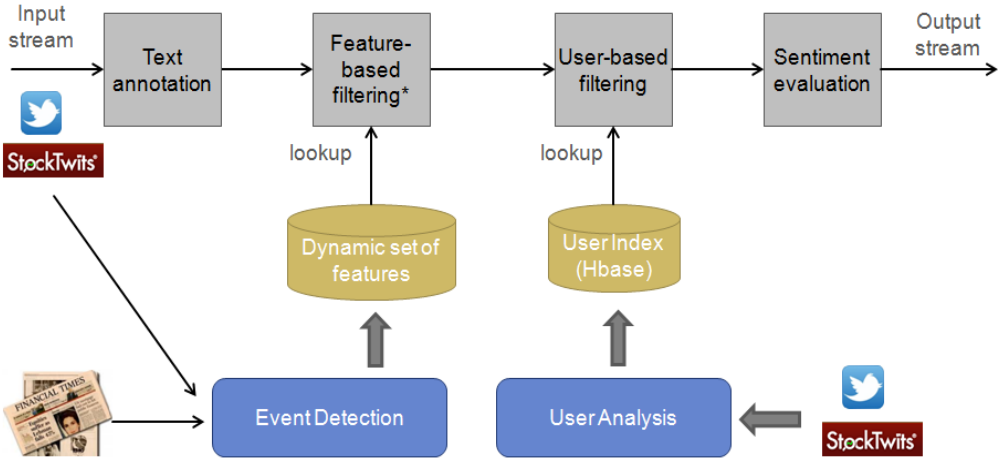

Figure 11: Example scenario for social web and news analysis

The reminder of this section reviews the different processing and analysis methods that are being considered in QualiMaster, describes their use cases, and discusses some initial and preliminary evaluation and investigation results that have been produced in the first year of the project.

6.1 Sentiment Analysis

In recent years, we have witnessed an increasing amount of opinionated text appearing on the Web, with people discussing ideas and political issues, criticizing movies, reviewing books, or elaborating on features of their newly-bought camera. Not surprisingly, this content is not only appreciated by ordinary end users but also by professionals ranging from marketing and advertisement specialists to political strategists. The growing interest and demand for automatic analysis techniques and tools for opinionated digital texts have also fuelled research and led to various approaches in opinion mining and sentiment analysis [68].

Much of the work in this field has focused on the task of **sentiment classification** [28, 29]) which deals with the problem of *automatically* assigning opinion values to documents or topics using various text-oriented and linguistic features. Table 4 presents a couple of examples of objective, positive and negative tweets from the *Sanders Dataset* (http://www.sananalytics.com) where users write their opinions towards the entity *Apple*.

Objective Tweets
@APPLE freaks is Siri available for Iphone 4 yet? #IOS5 needed to include a landscape mode for reminders @Apple
Positive Tweets
Not Bad! @Apple Sells Over 4 Million #IPhones in Debut Weekend - Bloomberg - #smartphone #sm RT @VinodRad loving new technology from @apple iPhone 4s @Apple Safari Reader owns the worldwide web
Negative Tweets
THIS IS WHAT WILL KILL APPLE http://t.co/72Jw4z5c RIP @APPLE @apple why my tunes no go on my iPhone? iPhone lonely without them. silly #iOS5 I just need to exchange a cord at the apple store why do I have to wait for a genius? @apple

Table 4: Examples of objective, positive and negative tweets

In an early work, Pang et al. [28] studied the problem of classifying movie reviews, based on the sentiment expressed into positive or negative using various classification methods. Their results showed that the SVM approach using various textual features provides the highest accuracies for the task of movie reviews sentiment classification. In our recent work [67], results showed that Naïve Bayes and Logistic Regression are usually inferior to SVM classifiers for the query sentiment classification task and that v-SVC (from LIBSVM) performs at best.

A couple of recent works studied the problem of **sentiment classification of micro-blogs** based on the opinions they express. In [62] the authors investigate the performance of supervised methods on Twitter and report that the SVM method using a unigram-based document representation outperforms the Naïve Bayes and MaxEnt approaches. In a more recent work [61],

the results show that, while models trained on textual unigrams perform relatively well, some improvements of up to 4% can be obtained by (1) incorporating features based on prior polarities of words in conjunction with part-of-speech tags and (2) training the models on a significant larger amount of data. Kouloumpis et al. [63] elaborate on the usefulness of linguistic features. The authors claim that, micro-blogging features such as intensifiers, emoticons and abbreviations could improve the classification effectiveness.

Some other papers have tackled the problem of **sentiment classification in the financial domain**. O'Hare et al. [64] attempt to predict the sentiment of financial blogs towards companies and their stocks, with the focus on the usefulness of word, sentence and paragraph level features. Martinez et al. [65] propose using gazetteer lists as sentiment lexicons in order to predict the opinion value of financial news documents. Smailovic et al. [66] propose an active learning approach for estimating the overall sentiment of a Twitter stream in correlation with the stock indexes.

In [26], we developed a framework able to capture **the temporal development of opinions** on politicians. Our experimental studies in the context of the US 2008 election campaign showed that purely data-driven approaches can already capture trends, peaks and switches for public opinion in the political domain. Here we achieved further improvements by utilizing, in addition to blog data, a history of past opinion polls results for learning parameters that capture intrinsic differences between blogosphere and the "real world".

Building on our previous works, in QualiMaster we aim at capturing the real-time development of opinions in social media on certain entities (such as financial market players) or reactions to news events. The Sentiment classification methods from QualiMaster will involve a couple of steps which are briefly described in Figure 12. In a preliminary step, our main focus is on retrieving and filtering stream-based Tweets and Financial Messages. In the second step, we will make use of both supervised and lexicon-based sentiment classification methods in order to assign the documents to either the positive, negative or objective class. Finally, the set of methods will include a sentiment aggregator and time-aware aggregator capable of providing a representative sentiment mood estimator over the input stream.

In QualiMaster, we will make use of both supervised and lexicon based approaches (which we latter discuss in this section) in order to construct the Sentiment Classification components. Our focus will be on adapting and enhancing the effectiveness and efficiency of SVM based approaches together with SentiWordNet in the context of the financial domain. Moreover, our work considers investigating the applicability of other resources in order to deal with the Twitter specific challenges that involve for instance ambiguity, vagueness and slang.

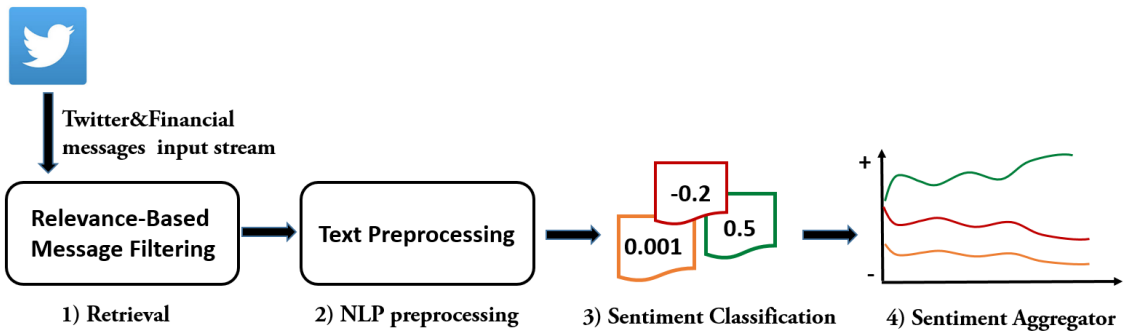

Figure 12: Temporal development of opinions

More specifically, in QualiMaster we are working on implementing the use cases 1, 2 and 3 that are listed below for support of the financial risk analysis application:

Use Case 1	**Monitoring opinions about a market player**
Description	Analyze and monitor development of people opinions about a certain market player over time
Input	● Social media streams ● Features of a market player (name, stock, products etc)
Steps	1. Receive posts from social media streams 2. Select posts about the given market player 3. Perform sentiment analysis on posts
Output	a time series, where each point is a vector of indicators (# of posts, # of positive posts, # of negative posts) along with a timestamp

Use Case 2	**Monitor public mood (investor fear)**
Description	Analyze and monitor the development of general public mood with respect to financial market over time
Input	Social media streams
Steps	1. Receive social media streams 2. Select posts related to financial market and business domain 3. Perform sentiment analysis on posts 4. Consider average users, domain experts and influential users separately
Output	a time series, where each point is a vector of indicators (# of related posts, # of positive posts etc) along with timestamp

Use Case 3	**Monitoring reactions to news events**
Description	Analyze people reactions in social media to a specific event
Input	● Social media streams ● Features of a news event (URL, headline, and/or keyword etc)
Steps	1. Receive social media streams 2. Select posts about the given news event (comments, tweets etc) 3. Perform sentiment analysis on posts
Output	a time series, where each point is a vector of indicators (# of reactions, # of positive etc) along with timestamp

6.1.1 Supervised Approaches for Sentiment Classification

Text classification refers to the supervised machine learning approach in which a model capable to distinguish documents belonging to different *classes* (also known as *categories*) is learned. In order to learn the classification model, a set of labeled *training* documents is required. The learned model can be applied to predict the class labels for a set of *test* documents, for which the class is unknown. Training and test documents, which are given to the classifier, are represented as multidimensional vectors $\vec{d} = (d_1, \dots, d_m)$. These vectors can, for instance, be constructed using *TF* or *TF ·IDF* weights which represent the importance of a term for a document in a specific corpus [30, 31]. In this deliverable, we briefly describe a couple of state-of-the-art text classification approaches which were also frequently used in various works focusing on sentiment classification in Twitter and the financial domain (see for instance [66, 62, 61, 64]) .

Support vector machines (SVMs) were introduced by Vapnik et al. [32, 33] and are considered to be highly accurate methods for a various set of classification tasks [34, 28, 35, 36]. Manning et al. [37] provide a formalization of the SVM classification method in the context of text classification. Given a set of *n* training documents $D = \{(\vec{d}_i, y_i)\}$ with $\vec{d}_i \in R^m$ and $y_i \in \{-1, +1\}$ the corresponding class labels, we make the assumption that the training data is linearly separable. The linear SVM method aims at finding a hyperplane $\vec{w} \cdot \vec{x} + b = 0$ that separates the set of positive training documents from the set of negative documents with a *maximum margin*. Because of its final role on deciding to which class a new document should be assigned to, the separating hyperplane is also known as the *decision hyperplane* [37] or *decision surface* [38].

Traditionally, SVMs have been used for *binary classification* scenarios, but they can be adapted for a multiclass case as well. In our work, we will build binary and multiclass classifiers using the SVM formulation implemented in the LIBSVM package [39]. However, other implementations exist such as the *SMO* variant and the *L2-loss* linear methods implemented in the Weka library [40].

The **Naive Bayes** classification method [37] is a probabilistic approach which makes the “naive” assumption that the term attributes are conditional independent of each other, given the document classes. For a test document *d*, the Naive Bayes approach can estimate the probability of *d* to belong to class c_j , where $j \in \{1, \dots, k\}$ based on: (1) the prior computed conditional probability of each term w_k in *d* to occur in a document of class c_j and (2) the fraction of training documents belonging to class c_j.

The **Logistic Regression** [41] is a statistical and probabilistic classification method which learns a logistic (sigmoid) function in order to predict binary outcomes. In our work, we will build classifiers using the Multinomial Naive Bayes and the Simple Logistic Regression methods from the Weka library.

6.1.2 Lexicon based Sentiment Analysis

Recent work in the area makes use of annotated lexical resources such as SentiWordNet [42] or SentiStrength [43] to detect the sentiment classes (e.g., the former thesaurus is employed in [44]). There are several works that make use of sentiment thesauri for exploratory studies. For instance, in [31] we use SentiWordNet to analyze sentiment in YouTube comments and the relationship between sentiment and comment ratings. In [45] the SentiStrength resource is leveraged for studying sentiments in Yahoo! Answers with respect to temporal and demographic aspects. Vural et al. [46] employs a sentiment thesaurus to guide a focused crawler for discovering opinionated

web content. For our future work in QualiMaster, we plan to explore the applicability of already existing sentiment lexicons in order to enhance the classification modules related to the financial domain. Here, our main focus will be on exploiting SentiWordNet. However, we will also investigate the performance of other possible methods. Moreover, our future work will also investigate relevant methodologies for generating sentiment lexicons specific for the financial domain.

SentiWordNet is an enhanced lexical resource which was built on top of WordNet and can be exploited in various sentiment analysis tasks. WordNet is a thesaurus containing descriptions of terms and semantic relationships between terms (examples are hypernyms: "car" is a subconcept of "vehicle" or synonyms: "car" describes the same concept as "automobile"). WordNet distinguishes between different part-of-speech types (verb, noun, adjective, etc.) A *synset* in WordNet comprises all terms referring to the same concept (e.g. *{car, automobile}*). In SentiWordNet a triple of three *senti values* (*pos, neg, obj*) corresponding to positive, negative, or rather neutral sentiment flavour of a word respectively) are assigned to each WordNet synset (and, thus, to each term in the synset). The sentivalues are in the range of [0, 1] and sum up to 1 for each triple. For instance (*pos, neg, obj*) = (0.875, 0.125, 0) for the term "awesome" or (0.125, 0.75, 0.125) for the term "wrong". Sentivalues were partly created by human assessors and partly automatically assigned using an ensemble of different classifiers.

6.1.3 Evaluation Measures

In order to evaluate the effectiveness of a sentiment classifier, which is usually trying to assign previously unseen documents to one out of two classes (e.g., *positive* vs *negative*, *subjective* vs *objective*), various evaluation metrics can be employed. Within the rest of this subsection we will briefly describe a couple of popular measures used in the literature. Note that these measures are in concordance with the quality dimensions described in Section 2.4.1 (Figure 5). 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.

Let us consider a two-class classification problem, where the learned model can be applied to predict the class labels for a set of *test* documents, for which the class is unknown. For a new previously unseen test document, the task is to assign it to the "positive" or the "negative" class. For each individual prediction, documents can be correctly (true) or incorrectly (false) classified as belonging to the positive or negative class. In this context, we can define the following subsets and use them for defining a couple of standard evaluation measures used in the literature.

T_p = positive instances *correctly* classified as belonging to the positive class
T_n = negative instances *correctly* classified as belonging to the negative class
F_p = negative instances *incorrectly* classified as belonging to the positive class
F_n = positive instances *incorrectly* classified as belonging to the negative class

Accuracy: the fraction of documents correctly classified:

$$\text{Acc} = \frac{Tp + Tn}{Tp + Fp + Fn + Tn}$$

Precision: the fraction of documents correctly classified as belonging to the positive class:

$$P = \frac{Tp}{Tp + Fp}$$

Recall: the fraction of positive documents that are correctly classified:

$$R = \frac{Tp}{Tp + Fn}$$

Precision-Recall Curves: provide a more informative picture for the performance of a text based classifier, taking into account the confidence of the classifier towards assigning an item to the positive or negative class.

Break-Even Points (BEPs): precision/recall at the point where precision equals recall, which is also equal to the F1 measure.

F Measure: represents an evaluation measure computed based on the *weighted harmonic mean* [37] of precision and recall:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}}$$

where $\alpha \in [0, 1]$ is a parameter which can be used to emphasize the importance of either the precision or recall of the classifier. A common version of the F measure is the F_1 score, which gives equal weights to both precision and recall ($\alpha = \frac{1}{2}$).

F_1 Measure (F_1 Score): the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The Break-Even Point and F_1 Score represent a single value measure that provides an overview for the precision recall curves.

Receiver Operating Characteristic (ROC) Curves: describes the True Positive Rate (TPR) vs. the False Positive Rate (FPR) of a classifier, where:

$$TPR = \frac{TP}{TP + FN}$$

and

$$FPR = \frac{FP}{FP + TN}$$

mainly allow to judge classifiers in terms of costs (the number of incorrectly classified negative examples) vs. benefits (the number of correctly classified positive examples), as claimed in the literature [25, 27].

AUC – Area under the ROC Curve: provides a one single value in the interval [0, 1] summarization for the ROC Curve.
Due to the difficulty of the task, there are text classification scenarios which are not feasible for high accuracy/precision scenarios. However, trading recall against precision might lead to applicable results. For instance, given a recall of 0.1, it can be that we can obtain a precision

higher than 0.9. Alternative measures which focus on the True Positives and the amount of correctly identified documents coming from the “positive” class exist.

Despite the common use of ROC curves in the machine learning domain, Precision-Recall curves have been preferred in the area of Information Retrieval, especially for tasks with a skewed distribution for the document classes [27, 69]. Moreover, the P-R curves are claimed to provide a more detailed picture for algorithms, which Davis et al. formalizes under the theorem claiming that “a curve dominates in ROC space if and only if it dominates in PR space” (please see Theorem 3.2 in [27]).

6.1.4 Sentiment Classification in Micro-blogs

In this section, we study the applicability of state-of-the-art sentiment analysis methods for detecting the sentiment classes of tweets. Twitter documents exhibit inherently different characteristics in comparison to classical corpora used for sentiment analysis (i.e., news stories, blogs, product reviews, and comments). In this part, our goal is to exploit features obtained from the pure tweet text, while applying and evaluating the current sentiment detection techniques for this source of data with its unique characteristics. The performance is evaluated on more than 12,000 human annotated tweets coming from three datasets.

Dataset In this part of the work, we aim at building a *general purpose classifier* which tackles the scenario where we do not have any training labeled instances for a specific domain. Therefore, our models are constructed and evaluated using a mixture of three publically available datasets:

1. Sanders-Twitter Sentiment Corpus: contains 5,513 hand sentiment-labelled tweets retrieved for the topics *Apple*, *Google*, *Microsoft* and *Twitter*. The dataset is available at http://www.sananalytics.com.
2. The Dataset used in [49] which is available upon request and contains 5,127 tweets. Each tweet in this collection was labelled as being either *positive*, *negative*, *neutral* or *junk*.
3. SemEval-2013 Dataset: contains manual annotated tweets relevant for a range of topics such as *Steve Jobs*, *Kindle*, *iphone* and *earthquakes*. This data was used for the Message Polarity Classification Task, where the goal was to detect if a message is positive, negative or neutral with respect to the expressed sentiment. The dataset is available for download at http://www.cs.york.ac.uk/semeval-2013/task2/index.php?id=data .

For constructing the *general purpose sentiment* dataset, we sampled from each of the above mentioned datasets, samples of equal sizes for the positive, negative and the neutral classes.
This resulted in a final set of approximately 12,000 tweets, with 4,000 tweets in each sentiment class.

Setup: For our classification experiments, we constructed feature vectors based on the text of the tweets. We constructed multi-dimensional feature vectors using TF-IDF weights of the terms appearing in the tweets. While doing this, we also accounted for negations (i.e., if a negation, say, “not", immediately precedes another term *t*, we created a virtual term *not_t* in a similar way as described in [47]). We employ a state-of-the-art text classification approach, namely the *v*-Support Vector Classification (*v*-SVC) formulation of SVM from LIBSVM [39]. We 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.

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 remaining two classes for training as well as for test sets. In this way, we created balanced training and test sets for each classifier. This is similar to the approach employed by [48] to eliminate the effect of any underlying bias for a particular sentiment class in the data. We repeated the experiments by switching training and test sets and computed the averages for the evaluation metrics. We chose the number of training tweets in such a way that the maximum number of available annotated documents could be used during the training and testing. As approximately 4000 tweets are annotated as positive, the positive vs. all classifier was trained with 2000 tweets from the positive class and 1000 tweets selected from each of the negative and objective classes. The test set was created analogously. For the negative vs. all and objective vs. all classifiers, we trained and evaluated the models with the same amount of tweets.

Results Our quality measures are 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 precisionequals recall, which is also equal to the F1 measure, the harmonic mean of precision and recall in that case). The detailed precision-recall curves for the three classifiers are shown in Figure 13. The main observations are:

- Using the bag-of-words texts, binary *v*-SVC classifiers *positive vs. all*, *negative vs. all* and *objective vs. all* yield BEP values of 0.658, 0.685, and 0.668, respectively.
- All three types of classifiers provide relatively good performance for Recall < 0.4.
- Trading recall against precision leads to applicable results. For instance, we obtain prec=0.876 for recall=0.1, and prec=0.839 for recall=0.2 for the positive-vs-all classification scenario; this is useful for finding candidates for positive-opinionated tweets in large Twitter sets.

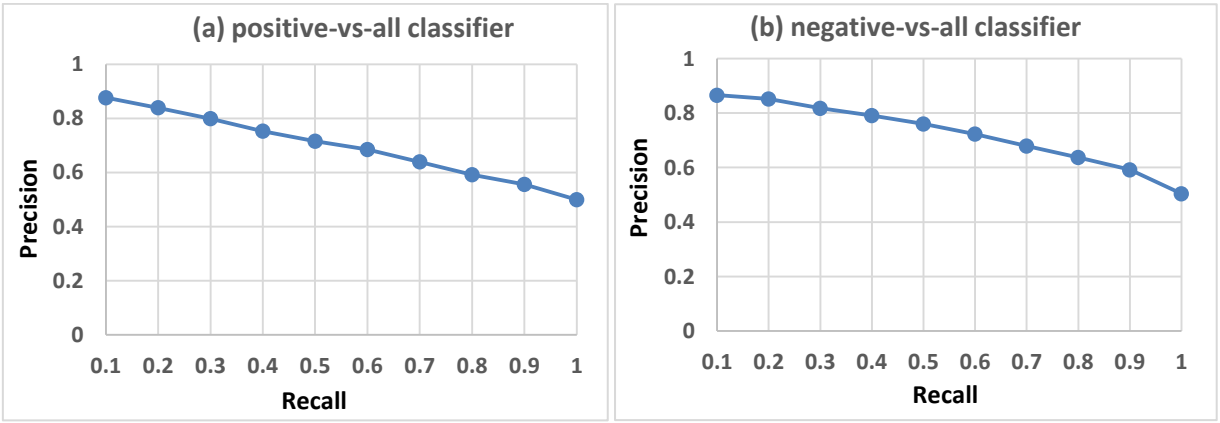

Figure 13: Precision-recall curves and BEPs for (a) positive vs. all, (b) negative vs. all

We also applied the well-known lexicon-based method SentiWordNet in order to predict the sentiment classes of the test Tweets. Here we used the test documents from each of the preceding classification tasks. SentiWordNet yielded accuracy values of 0.649, 0.619 and 0.651 for the test sets employed in positive-, negative-, and objective-vs-all classification experiments. These values are relatively close to the accuracy scores provided by the SVM method, 0.661, 0.680 and 0.676. However, using the unsupervised approach we were able to classify only approximately one third of the messages, as many of the documents do not contain a value in the SentiWordNet's vocabulary list.

6.1.5 Sentiment Classification in Financial Micro-blogs

Preliminary Term Analysis The textual content of the financial microblogs coming from the StockTwits dataset can provide clues on the vocabulary used by the professional users in the dedicated microblogging platform.

As a first step, we conducted a term analysis on the financial microblogs texts to compare the subjective ("bullish" and "bearish" tweets) vs objective ones. We ranked the microblogs terms (after stemming) using the Mutual Information (MI) measure, which mainly quantifies how much the joint distribution of terms deviates from a hypothetical distribution in which terms and sentiment classes are independent of each other. In other works, the MI is also employed for the detection and extraction of opinionated words [50]. In our case, we just exploit this measure for identifying the most probable words appearing in the two distinct sentiment classes (subjective vs objective).

Table 5 shows the top-20 stemmed terms with the highest MI scores for the subjective vs. objective classes. Note that the term lists are quite intuitive. The objective class involves mainly general financial market terms (e.g., open, close, median, list, report), company market names (e.g., *aapl*, *s&p*, *dow jones*, *s&p*). In the other class, many of the terms are *general opinionated terms* (e.g., good, short, long, strong, big), or terms that could be considered *financially opinionated* (e.g., buy, sell, bull, bearish).

Terms for Subjective Financial Microblogs			Terms for Objective Financial Microblogs		
short	share	strong	aftr	averag	energi
sell	money	profit	close	continu	publish
compani	bull	high	trade	drop	aapl
buy	volum	bear	open	point	recent
dump	long	good	rate	block	interest
ebola	cover	peopl	median	nasdaq	event
earn	tomorrow	suit	report	upsid	volatil
pump	bearish	sale	list	qtrs	wk
price	lol	big	s&p	etf	thought
come	million	click	post	correct	dow

Table 5: Top-30 (stemmed) terms w.r.t. MI values for subjective(left) vs. objective(right) category

In this subsection we study the performance of SVM models for predicting the sentiment class of microblogs coming from the financial domain, namely the StockTwits platform.

Here we mainly repeat the "one vs. all" experiments from the previous classification task on a new type of data. Therefore we built three types of binary classifiers to separate each sentiment class from the other two classes. In this case the "positive" microblogs are those labelled as "Bullish", the negative ones the "Bearish" ones and finally the microblogs not containing any such market label the objective ones.

We construct the feature vector of a StockTwit document using the frequencies of the terms occurring in the document, normalized by the number of terms occurring in the document. We removed stopwords and terms occurring just once in the corpus, and applied the stemming method implemented in the Lucene library. Similar to our General Purpose Classification presented in the

previous subsection, we used the *v*-Support Vector Classification (*v*-SVC) formulation of SVM from LIBSVM [39].

Results

Our quality measures are again 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).

The detailed precision-recall curves for two classifiers are shown in Figure 14. The main observations are:

- Using the bag-of-words representation, binary *v*-SVC classifiers *positive vs. all*, *negative vs. all* and *objective vs. all* yield BEP values of 0.664, 0.669, and 0.583, respectively.
- The pos-vs-all and neg-vs-all classifiers provide relatively good performance for Recall = 0.1 (0.755 and 0.781, respectively).
- The BEP for the pos-vs-all classifier is slightly higher for the financial microblogs, while the classification effectiveness is higher for the other two classifiers, *neg-* and *obj vs all*.

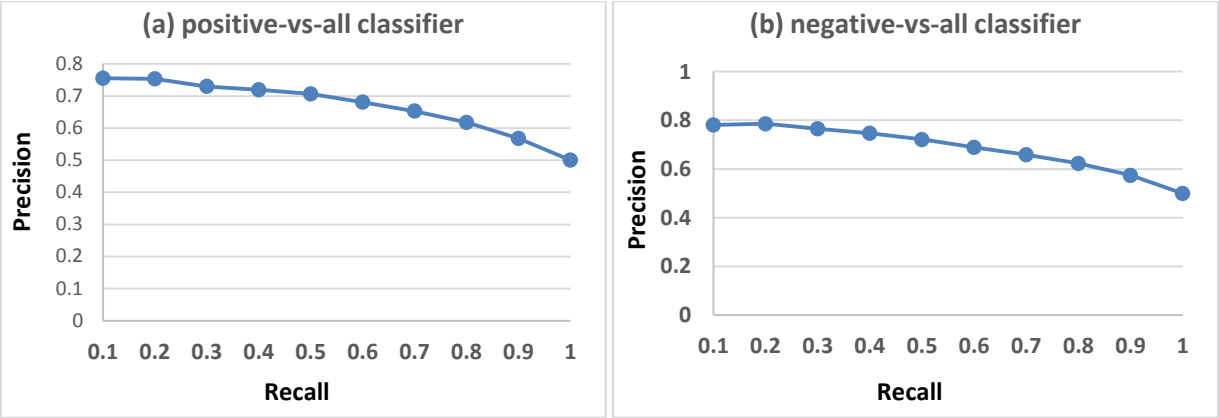

Figure 14: Precision-recall curves and BEPs for (a) positive vs. all, (b) negative vs.

In addition to the SVM supervised approach, we also applied the unsupervised lexicon-based approach for predicting the sentiment classes of the test StockTwits messages. Here we used again the test documents from each of the preceding classification tasks. SentiWordNet yielded accuracy values of 0.612, 0.604 and 0.578 for the test sets employed in positive-, negative-, and objective-vs-all classification experiments. These values are slightly lower then the accuracy scores provided by the SVM method, 0.661, 0.667 and 0.583. Using the unsupervised approach we were able to classify only approximately one third of the messages, due to the fact that many of the documents do not contain a value in the SentiWordNet's vocabulary list.

6.2 User and Social Network Analysis

Estimating the influence of users within a social network and investigating the expertise of users based on the topics they frequently write about can help in estimating the relevance of the content generated by their users for the analysis task in hand. This is particularly useful for filtering the noise and the spam from the vast amount of content that is generated in social media every day. In the scope of the QualiMaster project we want to use this information for estimating the credibility and reliability of the content. For instance, it is important to know who spreads certain news in order to allow a prediction on how this could influence future topics and sentiment towards certain entities in the network. Users with a strong influence in a social network can reach many people with a single message and change their mood towards a certain market player relative quick. Moreover, users with a lot of expertise in specific domain may be the first who report about interesting news in certain areas or about new products and developments from specific market players. The following two use cases will be considered and implemented in QualiMaster towards building a User Index that can be then used to evaluate and/or filter new content posted by users in social networks.

Use Case 4	**Identifying Influential Users in Social Networks**
Description	Analyze user profiles and their contributions in social networks to identify influential users
Input	Social media streams
Steps	1. Receive social media streams 2. Analyze the social network of users (e.g. followers, retweets etc) 3. Estimate the influence of users in their social networks 4. Store the estimated influence scores in the User Index
Output	A User Index that is updated continuously. Each user is assigned an influence score. Further metadata such location, language, etc is also stored in the user profile.

Use Case 5	**Identifying Domain Experts in Social Networks**
Description	Analyze user profiles and their contributions in social networks to identify domain experts
Input	Social media streams
Steps	1. Receive social media streams 2. Analyze the content of user posts in social media 3. Estimate the expertise of users with respect to the financial domain 4. Store the estimated expertise scores in the User Index
Output	A User Index that is updated continuously. Each user is assigned an expertise score (with respect to a topic. e.g. a specific stock or market sector). Further metadata such location, language, etc is also stored in the user profile.

6.2.1 User Expertise Analysis

In this section we describe some initial work that has been conducted in the first year of the project in the context of user expertise analysis. We distinguish here between two approaches that are being considered and evaluated: statistical and probabilistic approach using LDA, and entity-centric approach using Wikipedia.

6.2.1.1 LDA-based Expertise Detection

The work described in this section aims at distinguishing groups of users based on the content they published in different social networks. In the work described here we used data gathered from LinkedIn as a ground truth data for describing the expertise of the analyzed user. This ground truth allows us to analyze how users of different professions make use of social networks and if they use social networks like Twitter to talk about topics related to their professional skills.
Our approaches where developed to analyze the influence and the expertise of a user in real time with high accuracy. However, typically, high accurate methods are complex and therefore could be slower. Therefore, we will develop several alternative methods, where some methods are optimized for accuracy and other methods are optimized for efficiency. One example here is the collection of several tweets for one user in order to generate a complete user profile. Based on a single tweet it is possible to calculate values for the influence of the specific user, based on the number of followers or list counts that the user has, but a complete user model requires more data, so in order to create such a profile the memory consumption of the method increases because the users tweets need to be stored. Additionally the latency of the results increase because the algorithm would have to wait until a certain amount of tweets from the user is collected to generate her profile. We analyzed 3 methods in more detail which are presented in this section.

The first method and the corresponding analysis focus on the differences and commonalities of different groups of users with expertise in different domains across different networks. Many studies exist on general communication and connection practices within these networks. These studies on expertise search suggest the existence of subgroups centered around a particular profession. For this analysis we investigated 94,155 public user profiles.

People who work in similar professions typically share particular skills. Further, if people are asked to indicate their skills, it is expected that the skills they mention vary in granularity. For example, someone working in public relations may indicate skills in social networking and marketing, but also specific skills such as DTP software, writing press releases and time management. It is also known that people from different professions or cultural backgrounds have different practices in how they communicate with one another, the communication mechanisms that they choose and the topics that they discuss [106]. These differences can also be observed on a more private, personal level: programmers are usually more informal than bankers, people working in public relations are typically more active in social media than investors, and pastors will most likely talk about different topics than real-estate agents.

In this section, we investigate differences in communities within self-reported skill networks. We are particularly interested in discovering differences in their communication practices, since these information can help us to detect users that are experts in the domains related to the market players we are investigating in the project. To analyze this we monitor for instance how well a professional community is connected or how often people post updates via Twitter, what are the topics that they talk about, and what is the overall tone or sentiment of these communications?

Based on our analysis we generated an overview on how skills in professional networks are related and categorize these skills into professions. Further, we investigated to what extent

different professions differ from one another in terms of connections, topics, sentiment and shared content.

Dataset

In order to create our dataset, we first collected a set of 94,115 public user profiles from About.me, using the crawling strategy employed by Liu et al. in [107]. About.me is a personal profile site where users can include all their social-web accounts. From each profile, we collected the users' LinkedIn, Twitter and Facebook accounts. For LinkedIn, our crawler gathered the public profile data, including skills and expertise tags, industry, job and number of connections. For each account from Twitter, we gathered the complete user profile with information like number of followers and friends or number of lists the user is in. In addition, we crawled the latest 200 tweets using the Twitter Rest API (https://dev.twitter.com/docs/api). The average number of tweets posted by the users is 5,833, with a median of 1812. This indicates that most of our users are quite active in Twitter. We also had 33 users with more than 100.000 followers, which can be interpreted as already influential. Within QualiMaster we will use Twitter as a source for the social web data, therefore we will not mention the details on the dataset collected from Facebook. The details can be found in [91]. Since the LinkedIn account serves as a source for our topics describing the users, we use for our analysis only Twitter and Facebook profiles that have a corresponding LinkedIn profile, resulting in a final set of 7,740 users. Our datasets are inherently noisy, as they represent human behavior. For example, the skills from LinkedIn are self-reported. Similarly, tweet content and Facebook posts are a mix of - among others - work-related announcements, private updates, and responses to others. However, this noise is reduced by the fact that our analysis is based on a fairly large collection of users.

Subgroups in skills and expertise networks

Different professions are expected to have differences in terms of communication behavior, the way people are connected, the topics they talk about, the resources they use, and the way they express themselves. In order to study topics beyond individual tags and to obtain more context-related information, we additionally employed Latent Dirichlet Allocation (LDA) and modeled each LinkedIn Skills and Expertise tag-based representation of a user as a mixture of latent topics. For this, we used the LDA implementation in the Mallet library. Given a set of term sets (users u_i represented by their Skills and Expertise tags in our case) and the desired number of latent topics, k, LDA outputs the probabilities $P(z_j|u_i)$ that the Skills and Expertise topic z_j is contained (related) in the user profile u_i. In addition, LDA computes term probabilities $P(t_j|z_i)$ for tags t_j ; the terms with the highest probabilities for a latent topic z_i can be used to represent that topic. We empirically chose the number of latent topics as 50 for our LinkedIn dataset. Table 6 shows the top-10 most probable terms for10 selected latent topics, as assigned by the LDA method. In addition, the table contains short topic labels which were manually assigned.

Topic Label	Top-10 Topic Terms
E-commerce-Strategy	marketing media social digital online strategy advertising analytics web management
Entrepreneur-Start-up	business development strategy management start ups strategic entrepreneurship marketing planning
Marketing-Events, Press	communications media marketing relations social management public strategic corporate event
Financial	financial management analysis insurance finance planning business banking accounting risk

Manager-Project Planner	management business project process analysis improvement strategy leadership team planning
Real Estate	real estate homes home buyers sales property residential properties investment
Journalist	journalism editing media writing news radio social style broadcast ap
IPR Person, Legal Analyst	law legal litigation property writing corporate intellectual research contract civil
Sales Manager	sales management business marketing development strategy selling product strategic account
Human Resources, Team Manager	skills problem solving communication team leadership thinking creative people building

Table 6: The manually assigned topic labels and the most probable top-10 terms (assigned by the LDA method) for the 10 selected “Skills and Expertise” topics

Differences in Topics

In order to compare what users of different areas talk about in different networks, we indexed the tweets and Facebook posts of the users into a Solr Index. All the messages were processed through a standard text processing pipeline, in which we removed stop words and used a stemming algorithm. Beside this, we also removed links from the text as we are only interested in the `real words' used by a user. For tweets, we also removed the mentions of other users as well as the hash-symbol from hash tags.

This indexing allows us to compute the cosine similarity between different users and different professions. The similarity is calculated using the Solr *more like this* functionality, which finds documents similar to a given document or a set of documents, based on the terms within the given document. These terms are selected based on their TF/IDF values for the given document, which allows us to obtain a representative set of query terms for each user or profession. For our experiments, we selected the 500 most representative terms that occurred within at least five documents.

The main Question we want to answer here is: Do people use Twitter to talk about their professional skills as described in LinkedIn? If we find a strong correlation between the used vocabularies we can use the gathered terms to build up sets of experts for areas we are interested in. Additionally this experiments allow us to verify the expertise we find inside twitter based on the linkedin profiles of the users.

To answer these question, we built queries based on the terms used for the different skill and expertise groups (professions) from LinkedIn. Using these queries, we computed the score for the tweets or posts of every user. For each profession, we calculated the average score. The result of this computation is a matrix that shows how similar the users from the different professions are to the keywords of these professions. The resulting matrix is shown in Figure 15. In order to make the differences better visible, we normalized the results for every query by dividing it by the maximum score. This ensures that the results are within a [0, 1] interval and are comparable for every LinkedIn profession. The diagonal line shows that most users use Twitter to talk about their professional skills. In Twitter we observed a stronger correlation than in Facebook, which indicates that Twitter is used for *professional* communication to a larger extent than Facebook. These findings confirm the use of twitter as medium to gather professional insides from domain experts. Inside Twitter, we got an average self similarity (between the same profession cluster in LinkedIn and Twitter) of 0.884, while inside Facebook this values decreases to 0.741.

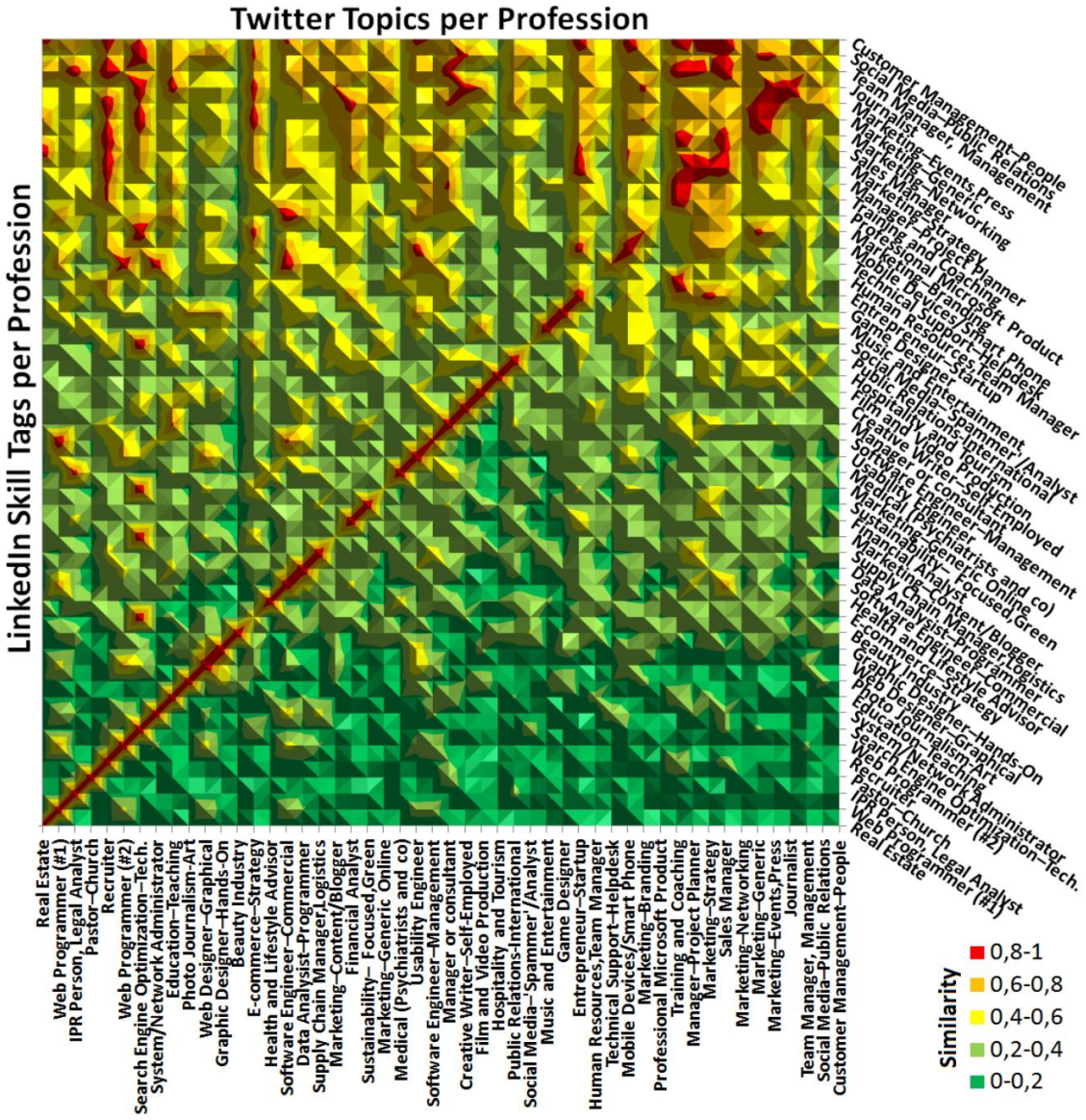

Figure 15: Similarity of skill tags from LinkedIn and terms used in Twitter Similarities are summarized per profession

Differences in Sentiment

The main goal here is to build models which can later help to interpret messages from a user of certain group in the right way. This can be important when looking at the relative sentiment of the messages of specific group, because some groups might be overall positive while others might be on general more negative in their messages. Additionally this can help when two different groups talk about the same product or market player. Considering the average mood inside these groups and normalizing the expressed sentiment can lead to a clearer interpretation of the messages.

In order to calculate the sentiment of messages we made use of the SentiWordNet lexicon. We studied the connection between the users' professions and the sentiment features of tweets and Facebook posts written by these users. SentiWordNet is a lexical resource built on top of WordNet. It contains triples of sentivalues (pos, neg, obj) corresponding to positive, negative or objective sentiment of a word. We restrict our analysis to adjectives, as we observed the highest accuracy in SentiWordNet. Finally, we computed the average positivity, negativity and objectivity over all

tweets and Facebook posts that belong to a profession. This information is of particular interest in the Project because this allows us a better understanding of the sentiment expressed by different users. Based on our analysis we might want to handle a very positive comment from an expert in a technical area different than a positive comment from a sales manager.
The users with skills in computer technical support and data analysis programmers tend to post the most negative messages. Their posts or tweets often offer or request help for problems, i.e., "@user Sounds like a hard drive issue. Either it's hitting bad sectors or the drive has literally slowed down and is having read/write issues". On the other side, users related to human resources, logistics and health, as well as lifestyle advisors post the most positive content in our collection. Some hand-picked examples from Twitter include "Best food moments of 2013 #food http://t.co/JdYO36wVAY", "Kids Eat and Stay Free at the Holiday Inn Washington DC. Bring the entire family for a holiday trip http://t.co/RhYa3zuHgu". We also observed that users tend to be more objective in Twitter than Facebook, particularly for some of the professions. The differences in sentiment between the different skills and expertise groups may reflect that people in some professions are more positive or negative in general, or that they tend to formulate their messages more positively or negatively. Our interpretation, however, is that the differences in sentiment are largely caused by differences in intentions of tweeting. The most positive groups are professions that use social media for selling and promoting items and events; it seems natural that these promotional messages are positive and motivating. On the other hand, the most negative group consists of people who work individually on programming or writing tasks.

Conclusions

With this set of experiments we investigated differences in communication and connection practices between professions, as represented by the skill and expertise groups that we extracted from a representative dataset. In our analysis, we used a combination of exploratory analysis, visualization and interpretation. The aim was to give some insight in the people, professions and conversation topics that constitute these structures. These insights call for separate analysis or treatment of activities within these subgroups, and provide several starting points for new functionality when working with messages from domain experts within twitter. Within the QualiMaster project we see several directions on how to use the gathered insights:

- The connections between Linkedin (which can serve as a ground truth) and twitter may allow us to scan twitter for certain keywords to build sets of experts for certain domain.
- The analysis of the sentiment of different groups can improve the interpretation of messages from certain users.
- The collected data can be used train models for categorizing users into specific expertise groups.

6.2.1.2 Entity-based Expertise Detection

Another way of measuring expertise in certain areas we are currently investigating is based on the occurrences of certain concepts or terms within the posted messages. This method uses the annotations linking to external knowledgebase like Wikipedia described in Section 5.2.2.1. The basic idea behind the approach is that based on the mentioned concepts it is possible to infer the domain a user interested in as well as the expertise the user has in that domain. We investigated how the use of the Wikipedia category graph can help to extract this information from tweets and other textual resources. The results of our experiments have been published in several research paper [92], [93], [94], [95].

The basics approach is based on 3 different steps shown in Figure 16. The extraction of Wikipedia concepts from the posted messages, the categorization of these concepts and the aggregation of the concepts on a user level.

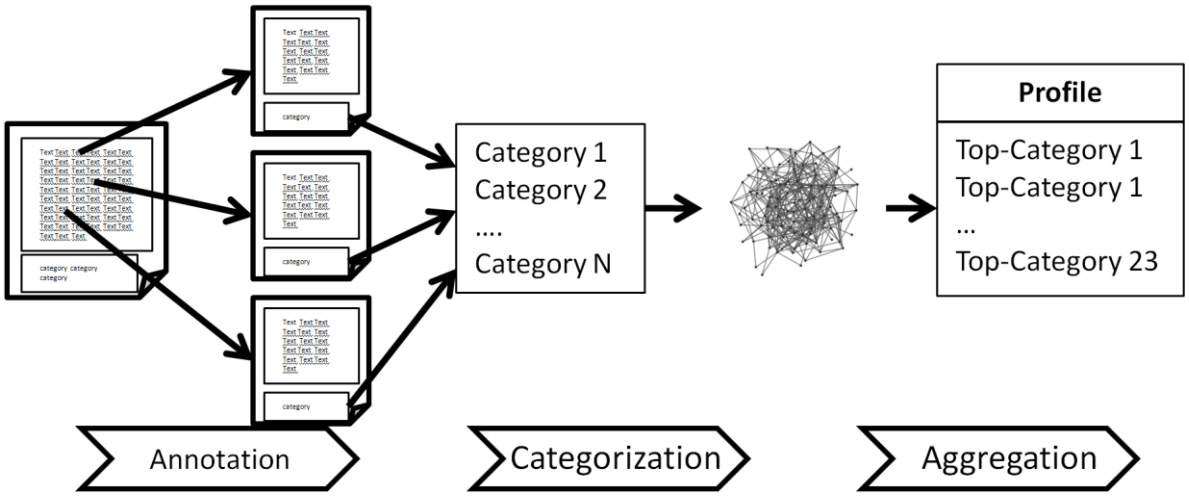

Figure 16: Entity-based expertise detection - overview

In the first step, the **Extraction**, we annotate all tweets of the user using the Wikipedia Miner Toolkit or one of the alternative methods. The tools provide us links to Wikipedia articles. The links discovered by Wikipedia Miner have a similar style to the links which can be found inside a Wikipedia article: Not all words which have a related article in Wikipedia are used a links, but only word which are relevant for the whole topic are used as links. Since tweets are rather short it is relatively hard to define the context, to deal with problem we consider larger sets of tweets from one user, if possible. This is also one of the performance tradeoffs we have to deal with in this approach. When increasing the number of analyzed tweets we also increase the latency and decrease the possible throughput of the method. On the other side we increase the accuracy of the method. The more tweets we analyze for one user the better the profile describes the user.

In the second stage, **Categorization**, we extract the categories of each entity that has been mentioned in the users tweets. For each category, we follow the path of all parent categories, up to the root category or a set of selected categories we are interested in. In some cases, this procedure results in the assignment of several categories to an entity. Since the graph structure of Wikipedia contains in some cases also links to less relevant categories we only follow links to parent categories which distance to the root is shorter or less than the one of the child category. A more detailed description of the method can be found in the published papers. Also in this stage the number of analyzed tweets per user influences the latency and the throughput of the algorithm, since for every detected concept this step has to be repeated. We can pre-compute the relations from article or concept to category so only a simple lookup is required for finding the corresponding categories but storing these relations increases the memory requirements of the algorithm.

For each category a weight is calculated by first defining a value for the detected entity. This value is based on the distance of the entity to the root node. Following the parent categories (which are closer the root category) we divide the weight of each node by the number of siblings categories, resulting in each entity receiving a n-dimensional vector representing the relations to a predefined set of categories. Based on this calculation we give higher scores to mentions which are deeper inside the category graph and are more focused on one topic.

In the final stage, **Aggregation**, we perform a linear aggregation over all of the scores for a tweet in order to generate the final user profile. The generated user profile displays the topics a user talks about as well as the expertise in a certain topic.

Several experiments have shown that an aggregation method like this performs well in different scenarios. In [93] we annotated paragraph from a set of Wikipedia articles and made a user study in order to assess the quality of the annotations. The results are shown in Figure 17. The users rated the quality of the assigned topics on a 5 point Likert scale with an average of 3.9, indicating that most of the assigned topics where very relevant for the given paragraph.

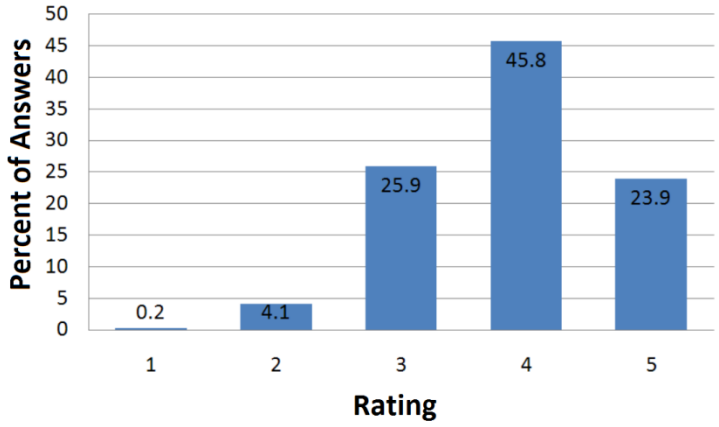

Figure 17: Distribution of user ratings for topic annotations

Besides the overall score we also analyzed how much the results differed between the different topics. We can see in Figure 18 that the distribution over the different topics is very similar, indicating that our method works equally well for all top categories.

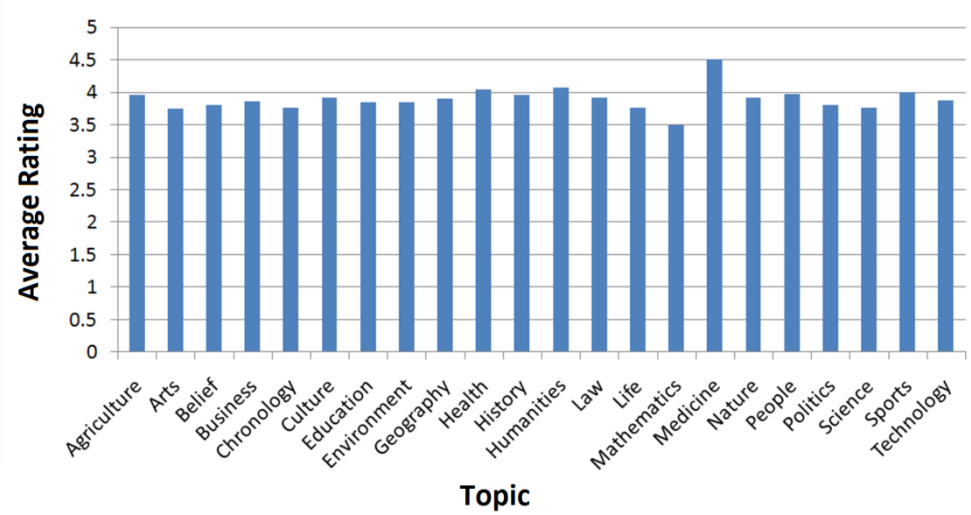

Figure 18: Average user ratings per topic

In [94] we analyzed how the topics a user posts about influence the topics of the links posted by that user. Additionally we analyzed how much the topics of a user change over time, and how much information we can get from a single tweet related to specific keyword.

To analyze this we collected tweets for different topics by filtering the streaming API with a set of related keywords. After this step we collected 200 tweets for every user in our collection and

downloaded the textual content of published links for the message related to the topic. After that we used the proposed method to generate 3 profiles for every user: one based on a single tweet, one based on the 200 tweets, and one based on the textual content of the posted URL. The comparison between the profiles where performed using the cosine similarity between the profile vectors containing the weight for the 25 Wikipedia main categories. The results are shown in Table 7. We see that for most of the analyzed topics we got relatively high similarities showing that also a single tweet can be used for generating a user profile.

	URL Content Single Tweet	URL Content User Tweets	Single Tweet User Tweets
Edward Snowden	0.995	0.968	0.961
Higgs Boson	0.812	0.628	0.496
Iphone 5	0.961	0.698	0.664
Israel Palastinian Talks	0.984	0.884	0.867
Nexus 5	0.968	0.972	0.956
Obamacare	0.983	0.79	0.752
World Music Avards	0.921	0.718	0.614
All topics average	0.946	0.808	0.759

Table 7: Similarity comparison of user profiles generated by different methods

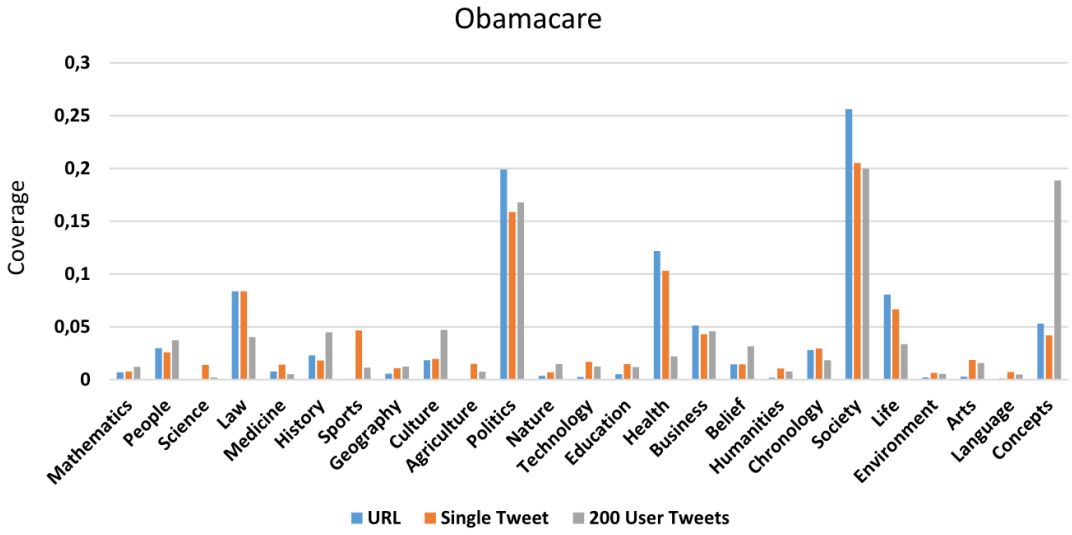

Figure 19: Topic coverage in user profiles who mentioned “Obamacare” in their tweets

Figure 19 shows the coverage of the profiles of the users who mentioned “Obamacare” in one of their tweets. We can see that these users mainly talk about topics like “Society” and “Politcs”. The topic “Health” is only covered in the single tweet related to “Obamacare”. This shows that more tweets are helpful for generating a complete user profile for covering all topics a user talks about and not being influenced too strong by entities mentioned in just one of the users’ tweets.

Table 8 summaries the different attributes of the two presented methods for generating profiles based on the content of the posted messages. Most of the parameters of the LDA and entity based approaches are depended on the number of tweets that are available per user, and therefore on the timeframe which is taken into account for the calculation. While the keyword based approaches offer relative good accuracy we assume, that the entity based approaches can outperform them

because of the availability of additional metadata for the detected entities. This increase of accuracy is depended on additional processing steps like entity detection and linking.

Method	Accuracy	Completeness	Believability	Memory Consumption	Complexity
LDA based Models	medium	Depended on Timeframe	Depended on Timeframe	Depended on Timeframe	Medium
Entity based Models	Medium - High	Depended on Timeframe	Depended on Timeframe	Depended on Timeframe	High

Table 8: Quality and performance comparison of user profile analysis methods

6.2.1.3 Related Work

Our focus on skill and expertise networks fits in the research area of automated expert finding, in which both explicit and implicit information is used for identifying experts in a particular area. Our interest in differences between expertise domains in how people connect and communicate online follows the tradition of social media analysis. Yimam-Seid and Kobsa [108] argue that for the effective use of knowledge in organizations, it is essential to exploit tacit knowledge that is hidden in various forms, including in the people's heads. The authors also separate the need for “information” from the need for `expertise': the need for people who can provide advice, help or feedback - or who can perform a social or organizational role. Their expertise recommender made use of a hand-tailored expertise model. Ghosh et al [109] leveraged social media (Twitter) content for seeking experts on a topic. Their results indicate that endorsement in other users' Twitter Lists (of which the topics need to be extracted) infers a user's expertise more accurately than systems that rely on someone's biography or tweet content.
Guy et al [110] examined indicators for expertise and interest as expressed by users of enterprise social media. The results are based on a large-scale user survey. They separate “expertise” (being knowledgeable or skilled) from “interest”(curiosity, basic knowledge, desire to learn more). As expected, interest and expertise ratings are correlated, with values for interest higher than for expertise. Results indicate that blogs and microblog provide different, more useful, information than communities and forums. The above-mentioned studies suggest that people's skills and expertise can be derived from both explicitly provided lists and from their connections and communication patterns. Kumar et al [111] analyzed the structure and evolution of online social networks. They showed that networks typically have one well-connected core region, but most users are located in one of several more or less isolated communities around it. These communities are typically centered around one central person, and it is unlikely that two isolated communities will merge at some point. Abel et al [112] compared different approaches for extracting professional interests from social media profiles. Results indicate that dedicated tag-based profiles and self-created user profiles are most suitable for this task. Twitter profiles are more diverse but also more noisy; this effect can be reduced by extracting entities from running text. In a follow-up study, Abel et al [113] analyzed the completeness of user profiles in different social media. The outcomes suggest that user profiles in networking services, such as LinkedIn, are more complete than those in services like Twitter. Further, the topics that users talk about differ between channels, but the overlap in topics is higher between services that are used for similar purposes.

6.2.2 User Influence Analysis

The calculation of influence of users within Twitter is another way we can use for ranking and filtering messages from certain user groups. These measures will allow us to distinguish messages from the broad masses in twitter and the influential users which may influence the overall perception of certain topics inside the whole network. These features can also help us sorting out messages from potential spam user. A user posting several messages in a short time with a very low number of followers and retweets has a higher probability of being a spammer than a user which is retweeted several times and has connections to several other users.

When analyzing the influence of a user within Twitter it is a common and easy way to take into account the number of followers this user has in the network or the number of lists the user was added to. A major advantage of this methods is the fact that this information is easily available as it is directly included in the JSON we get with every tweet. This allows us to compute a basic influence score with a very low latency. Also the memory consumption is very low. The basic features we can use for calculating a influence score are the user details we get within one tweet. These numbers are:

- Number of friends
- Number of followers
- Number of Lists the user was added to
- Number of favorites of the user
- Date the account was created
- If the Account is verified
- Number of tweets of the user

In previous works [citPNew1], [citPNew2] the users influence or impact was defined as a function of the users followers, followees (friends) and listings. The formula user by [citPNew1] for calculating the users influence is:

$$S(\phi in, \phi out, \phi list) = ln\left(\frac{(\phi list + \Theta)\,(\phi in + \Theta)^2}{\phi out + \Theta}\right)$$

This formula uses the number of followers and the lists (squared) of the user as a positive feature while the number of friends is used a negative feature.

In terms of Accuracy this formula can be considered as relatively good, but the pure influence of a user does unfortunately not say much about the influence in certain domains or the expertise in these domains. We plan to use this definition of influence as a basic starting value to describe the influence of users where we do not have additional information about the user. This could be the case if it is the first tweets of a user we observe. Once we have collected several tweets from a user we can use the before mentioned method to define the areas the user normally talks about and align this with the influence value.

Another way to calculate the influence of a user is to take into account the retweet rate at which tweets from a user are retweeted within the community. The advantage of a measure like this is that not only the popularity of a user is used to calculate the influence but also the quality of the content of his/her tweets. One way to incorporate the number of retweets to an influence metric would be to use an h-index like measure for defining the influence of a user. A formal definition of this metric would be that a user with N tweets has an h-index of h if h of the users tweets have been retweet at least h times each, and the other ($N - h$) papers have no more than h citations each. This metric would be similar to the h-index used for citations of scientific works. A drawback of this method that it is relative costly to store and collect all past tweets of all relevant users. We

are planning to make experiments with a fixed set of tweets for the given users and measure the correlation to other influence metrics.

Additionally one could also measure the number of mentions a user gets within the community as this also indicates some kind of influence. In contrast to the indegree - outdegree based method mentioned in the beginning these methods can be seen as more accurate with the drawback that more tweets are required to calculate these values, like for the h-index based method.

In the scope of the project we are planning to store a certain amount of tweets and user data. These tweets will be stored in the cluster and will build a database from which user profiles can be inferred and updated when new tweets from the users are collected.

This would allow us to compute more accurate user influence models at the cost of memory consumption and a higher latency. These models will increase their accuracy once enough data is collected, but will perform relative poor with small datasets.

The relevant information could also be collected by using the Twitter seach API, we could collect a set of old tweets for the stored users and build user profiles or calculate their influence score based on these old tweets. A major drawback of this method arises from the API limitation for the Twitter search API. A basic API key is only allowed to make 200 requests of this kind within 15 minutes. Since it is a valid assumption that the expertise and influence of a user is not changing frequently we are still planning to collect tweets for the users we are interested in and store them. This database can then be updated and increased over the time.

Table 9 summaries the attributes of the 3 presented methods for calculating user influence scores.

Method	Accuracy	Completeness	Believability	Memory Consumption	Complexity
Indegree – Outdegree	Low - Medium	High	Medium	Low	Low
Retweet based	Medium - High	High	High	Medium - High	Low - Medium
Mentions based	Medium	High	Medium	Medium - High	Low - Medium

Table 9: Quality and performance comparison of user influence estimation methods

6.2.3 Summary

In this section we present the different methods and algorithms used for building user profiles based on the collected messages from social media streams. The main goal of these methods is to increase the quality of the collected messages and to improve the use of these messages by adding additional metadata regarding the author of the messages. In the scope of the QualiMaster Project we are focusing on real time applications and due to that most of the presented methods are able to work with small set of tweets for each user or even with a single tweet. While other methods in this area make use of the whole social network we are focusing on information regarding each user which are fast and easy to collect. Methods making use of the social graph as a whole are not applicable in our scenario due to limitation of the availability of this information in real time. Methods that make use of the metadata we get with every message are relative easy to compute while offering first insights about the kind of user providing the message. In the QualiMaster project we will make use of all the different methods, since all methods offer advantages which cannot be compensated by other methods.

6.3 Event Detection

Recently, monitoring changes on web has turned to be mandatory for many real- world tasks such as scientific innovations, medicine, technology production, politics, etc. Users have been aware of the importance of detecting events that can be actually anomalous. In the area of business for instance, an early detection of events related to investment can reduce personal costs as well as the cost for society. However, retrieving content related to events of interest is a hard task, requiring searching an ever-increasing amount of data available at different sources and sites through the Web. Therefore, it has been clearly needed to develop event detection algorithms that can support the users in the task of finding events of interest. In this context, a lot of techniques have been worked up to meet the growing demands in several areas and domains, especially in data mining and information retrieval.

Social media data tends to reflect current events with a very short delay and has inspired much research on the reaction to current events and the detection of trends. Sakaki, Okazaki and Matsuo [56] were among the first researchers who have focused on detection of events on Twitter. They have constructed an earthquake reporting system in Japan and have proved in their work, that careful mining of tweets can be used to detect events such as earthquakes. Texts from social media such as Twitter and blogs have been used to predict global social trends [26, 50] and product sales [51], amongst other things. Zheng Liu, et al. have developed in [58] techniques to detect priming events which are defined as influential events triggering abnormal movements over time based on a time series index and evolving document stream.

Using Twitter data, Thelwall et al. [52] found that the occurrence of popular events is linked to increases in negative opinions. Opinion information has also been used to detect significant events. For example, Balog et al. [54] use LiveJournal mood labels to detect events based on times where mood time series has bursts. Similarly, Nguyen et al. [51] use a similar dataset to find the most positive or negative periods of a time period (macro events) as well as local bursts (micro events). Akcora et al. [55] also try to detect changes in public opinion. They track changes in the frequency of word use on social media, based on the idea that during an ongoing event, there are changes to the topics discussed, and therefore also to which words are used.

6.3.1 Event Detection in QualiMaster

The goal of the algorithms in this family is to detect burst or abnormalities in the textual streams such as news articles or user generated content (blogs/tweets). Some web sources provide information about emerging events directly or indirectly. Such source is WikiChanges https://github.com/edsu/wikichanges where Wikipedia edits can be monitored in real time. The assumption thereby is that in case of an event the corresponding Wikipedia also receive a burst of edits. There we can use supplementary information like category structure of the corresponding page, or user edit history to enrich the event with additional data. Other source is Yahoo Earnings Calendar http://biz.yahoo.com/research/earncal/today.html where business events are scheduled such as yearly balance and budget changes that are reported from the companies. Unfortunately Yahoo! does not provide an API to access the data, thus the HTML page needs to be parsed and we created such a parser, adoptable to different HTML templates. We are collecting those scheduled events on daily bases. This way it is theoretically possible to adapt the pipeline parameters pro-actively for processing future events. Such adaptation may be adding or removing keywords for stream filtering, or estimating the necessary throughput increase by judging the impact of the event due to high popularity of a particular stock.

The use cases 6, 7 and 8for event detection in QualiMaster are summarized in the following tables.

Use Case 6	**Events detection in news media**
Description	Detect (breaking) news events and optionally rank them with respect to specific user query
Input	• news feeds • (optionally) a user query
Steps	1. Receive news feeds 2. Detect bursts or stream abnormalities in time as events 3. Acquire context information for the given time period 4. Save the event as time period along with the context information
Output	A set of news to be monitored/further analyzed along with a set of contextual features (e.g. keywords) describing the event. The set of breaking news along with the event description can be used to

Use Case 7	**Events detection from social media**
Description	Detect emerging events in social media streams and optionally rank them with respect to specific user query
Input	• Social media streams • (optionally) a user query
Steps	1. Receive social media streams 2. Detect bursts or stream abnormalities in time as events 3. Acquire context information for the given time period 4. Save the event as time period along with the context information.
Output	A set of contextual features (e.g. keywords) describing an event, which can be used to adapt the monitoring of news feeds or social media streams in order to focus on the emerging event.

Use Case 8	**Events extraction from public calendars**
Description	Extract events descriptions and metadata from public online calendars
Input	• Online public calendars (e.g. Yahoo Earnings Calendar)
Steps	1. Regularly parse the online calendars 2. Extract event metadata including references to entities (e.g. market players) and temporal information (e.g. scheduled date of events) 3. Save the events and the time information along with the metadata
Output	A set of scheduled events along with temporal information and additional metadata including references to entities, which can be used to pro-actively adapt the monitoring of news feeds or social media streams in order to prepare for the upcoming event.

6.3.2 Event Detection Approaches

In order to analyze the posting behavior on Twitter in the area of stock markets we collected tweets related to 2900 different stock names listed at the NYSE. This collection consists of 470.000 Tweets. All of these Tweets have very strong relation to the market player we specified since they contain a keyword in the form of $ followed by the market player abbreviation, for example $AAPL stands for Apple. Figure 20 shows the distribution of tweets over time. We see that during the weekdays more tweets with stock symbols are published compared to weekends.

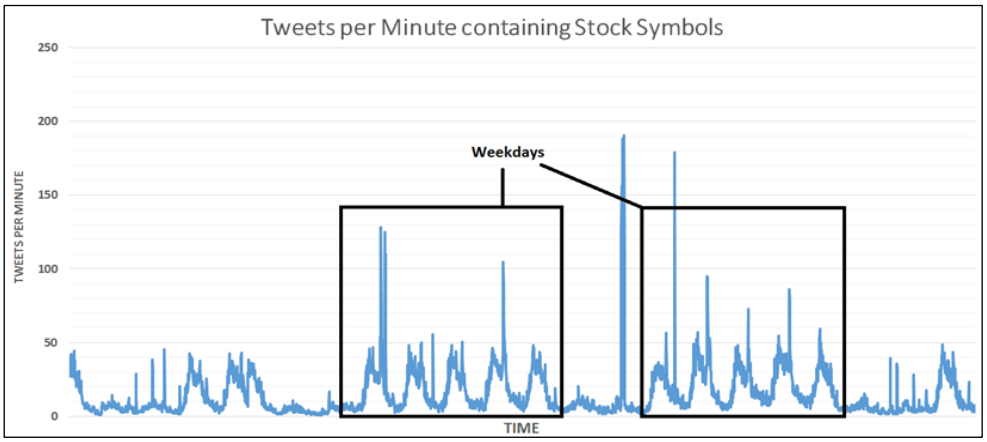

Figure 20: Distribution of tweets mentioning stock names over time

Whilst we plan to exploit different techniques for event detection, simultaneous co-occurrences of a burst or abnormality in different streams, will count as additional evidence for an event.

Social media and news streams are typically characterized by topics that appear, grow in intensity for a period of time, and then fade away. The appearance of such a topic in a document stream joined to certain features rising sharply in frequency as this topic emerges is considered as a burst of activity. There exist a number of basic algorithms with some of them described above. Table 10 depicts and compares the most popular techniques the moving average and the Kleinberg Burst detection algorithm. We plan to evaluate several methods by collecting ground truth and computing precision recall curves.

***Moving average*:** The basic idea of such algorithms is to maintain a moving time window monitoring the live statistics and detect if there is relatively unusual variation in the number of messages as an indication of an event. This number is compared to historical statistics of the average frequency which needs to be collected in advance. Whilst such algorithms are simple, the accuracy is connected to the size of window and an optimal size can usually not be fixed.

***Burst detection*:** Kleinberg has developed through his work a formal approach for modeling bursts, in such a way that they can be efficiently and robustly identified by minimizing a cost function that assumes a set of possible burst states, where it is costly to increase and cheap to decrease a level.

There will be two groups of strategies considered for event detection. The first group of “Content based” algorithms will analyze the textual content of the stream, whereas the second group “Feature based” makes use of other features like user annotations, tags, likes, mentioning frequency.

***Content - based event detection**:* The algorithms in this family typically monitor the stream of data using a sliding time window and detect abnormalities in the features appearance (features such as words, topics or entities) or bursts using the statistics based on the prior experience.

Feature - based event detection: This strategy is very similar to the previous one as the same algorithms. However the features are not extracted from the textual content but rather constructed from the indirect characteristics such as number of reposts of a message, source diversity or likes.

	Method	Complexity	Accuracy	Evaluation
Moving Average	Deviation of feature frequency from the statistical average	Simple	low	Precision recall curves.
Kleinberg Burst detection	Sharply increased occurrence of specific features	Complex	high	Precision recall curves.

Table 10: Comparison of event detection methods

Strategy	Complexity	Accuracy	Completeness
Content - based	High (NER, Topic extraction like LDA or Wikiminer, etc.)	Medium	High
Feature - based	Low	Low	Medium

Table 11: Comparison of event detection strategies

Table 11 compares the two strategies. Content based strategy requires heavy preprocessing such as extracting words and selecting named entities or topics. Precision is still a subject to evaluate, but expected to vary, however the coverage can be expected to be very high given certain accuracy of the concrete algorithm implementation. Feature based event detection does not require a lot of preprocessing as the information is often provided directly through the API's of the data sources, however the correlation of this information and real events is still a subject to evaluate and expected to be low resulting also in probably low coverage. Finally we can use sources like scheduled events which provide highly precise event prediction, but the coverage will depend on the focus of the source. In terms of additional profit, whilst for the content based techniques only terms itself can be used to label and explain identified events, semantics of the features can be exploited in the feature based technique and additional information like user annotations or Wikipedia category structure.

6.3.3 Preliminary Work

In order to perform experiments on the scalability and efficiency of event detection algorithms, we are currently developing a storm pipeline framework which will exploit our datasets of news items and Twitter messages as described. Figure 21 depicts the main components of this pipeline. First RSS items and social messages are collected and stored into HBase in parallel additional files such as HTML-pages with the corresponding news article are stored on Hadoop file system. The items are first filtered within the focused crawling step as described in Section 5.1 and passed over to the Event Detection Family, where the events are detected and the event related information is stored into HBase. Extracted events can further be ranked by relevance to user query e.g. by using

similar methods as described in Section 5.1. The already implemented prototype is currently successfully used to crawl RSS news and scheduled events from Yahoo calendar.

Between June and October 2014 we have collected more than 2 million news articles from top news international agencies such as Reuters, BBC, NYT, CNN, The Guardian as well as Google News. In addition, We collected from Yahoo Finance Calendar around 350,000 scheduled events from ~20,000 companies in the time period of 1999 to 2016 (This number also include rescheduled events).

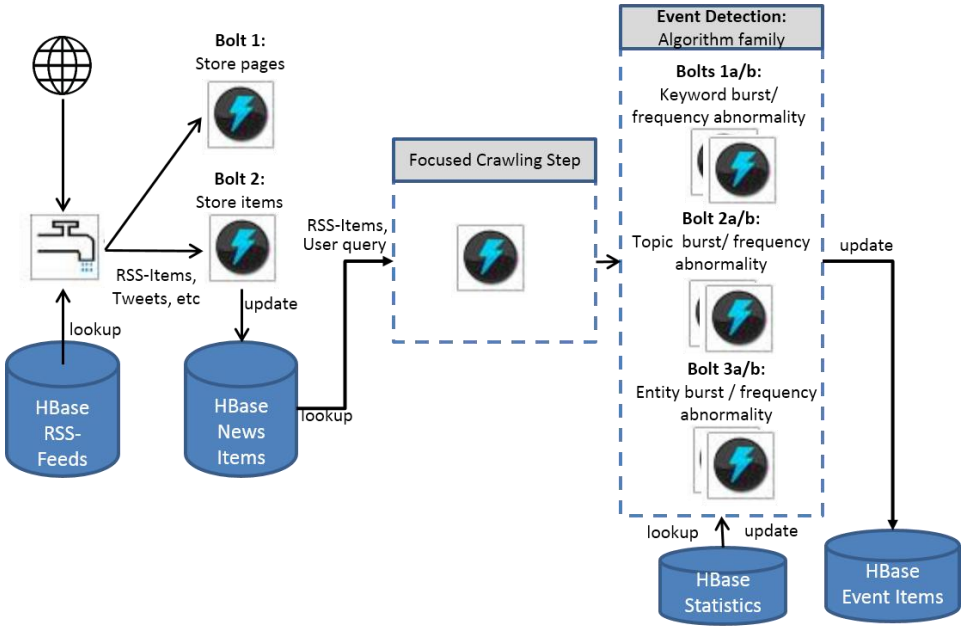

Figure 21: Collection and analysis of news feeds for event detection

6.4 Entity Network Extraction

Large networks of companies are present in many parts of the Web, for instance, in form of co-occurrence in a news article, or tweets as well as web pages. Social networks extracted from these rich information sources can be exploited in many ways and for various purposes. Node centrality measures in networks can help to identify important and influential companies or communities in a variety of business. Furthermore, knowing the topology of social graphs can shed light on the propagation of knowledge and innovations in networks.

There is a plethora of work on social network extraction from text and visual data. In [70] social connections between fictional characters are inferred from dialogs in books; similarly, in [71] a social network was extracted from the narrative of an Ottoman scholar and world traveler. In [73] social connections are constructed from a historical multimedia repository by leveraging co-occurrences of faces in images. Bird et al [74] extract social networks from email corpora. A number of works also explore the extraction of networks using search engine queries. [75] is probably the first of these articles. The authors exploit co-occurrences of names on results pages to identify connections between persons; however, the resulting networks are relatively small and centered on a single person. The POLYPHONET system [76] determines co-occurrence counts in Google queries to identify pairwise connections. In addition, different types of relationships and correlations with topic-related keywords are taken into account. In [77] search engines are used to retrieve connected email addresses. The Flink system [78] focuses on the mining of author

networks and combines web search with information from emails and publications. In [79] co-occurrences of persons in search result snippets are leveraged. There are few attempts to increase efficiency in discovering entity relations on the web [80, 81]; their contribution focuses on snippet clustering and entity disambiguation to reduce the number of pair-wise entity queries. All of the current methods are based on pair-wise entity queries, making them hard for gathering larger graphs. Textual patterns are often used for relationship mining. In [82] a bootstrapping algorithm is employed for iteratively extracting new patterns and semantic relationships. In [83] categorical facts and concept hierarchies are extracted from Web and News Corpora. Cimiano et al [84] leverage pattern-based search engine queries for building ontologies. Although close to our problem, these works tackle the problem of relation mining rather than extracting social networks.

6.4.1 Entity Network Extraction in QualiMaster

In QualiMaster, we plan to investigate the possibility and effectiveness of extracting networks of financial market players from social web data. In particular, we plan to realize the following two use cases:

Use Case 9	**Extracting Entity Networks From News Streams**
Description	Detect co-occurrence of company/stock names in news articles
Input	News article streams
Steps	1. Receive news articles 2. Extract entities co-occuring in articles (company names / stock) 3. Build the network assuming that co-occurring entities are connected through an edge.
Output	Network of entities

Use Case 10	**Extracting Entity Networks From Social Media**
Description	Detect co-occurrence of company/stock names in social media
Input	Social media streams
Steps	1. Receive social media streams 2. Extract entities co-occurring in user posts (company names / stock) 3. Build the network assuming that co-occurring entities are connected through an edge.
Output	Network of entities

To realize the above mentioned use cases, the company or stock name mentions need to be extracted from the text data. To this end we need to evaluate existing NER tools, described earlier regarding their ability to detect such entities. Furthermore, the extracted networks will have to be continuously updated based on new data from online streams to be able to capture the dynamicity in the network structure and its evolution over time and provide several type of analyses as described earlier. Similarly to the event detection scenario, two general extraction techniques with complexity can be exploited, namely content based and feature based techniques.

6.4.2 Preliminary Results

As a proof of concept we extracted a market player network consisting of company abbreviations co-mentioned in our StockTwits dataset (where the co- mentions are stored as features for each posting). Figure 22 depicts a sub-graph of the resulting network. For illustration purposes we only selected top 40 nodes and high weight edges, calculated based on mention frequency. Here we can observe 3 clusters one consisting of IT companies such as Google (GOOG), Facebook (FB), Apple (AAPL), second is related to Medicine and Genetics Hemispherx Biopharma (HEB), NewLink Genetics (NLNK), Lakeland Industries (LAKE) and finally a gold cluster: Gold Trust (GLD), Junior Gold Miners Index (JNUG), Gold Miners (GDX). Furthermore, we can observe that the cluster can have different structure. For example the “medical” cluster is much higher interconnected compared to the “IT” cluster. During the course of the project we will investigate the exploitation of several features that can be extracted from social web and news streams to build more accurate representation of market players networks that reflect the real-world. This work can exploit, for example, the results of the sentiment analysis and event detection methods to use them as additional features (together with co-mentions).
Furthermore, results obtained by this work will be compared and combined with results obtained from the correlation estimation between market players that is computed based on the financial streams in order to evaluate the effectiveness of the analysis on social web data.

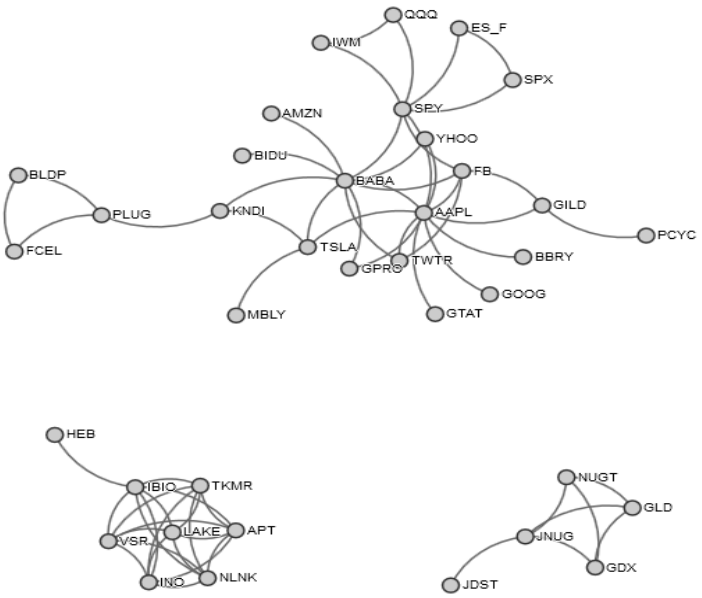

Figure 22: StockTwits graph of co-mentioned stocks

7 Conclusions and Future Work

This deliverable is the first deliverable of WP2. It describes the overall approach for scalable and quality-aware real-time data stream processing algorithm and methods. An overview of several families of algorithms and methods that are being considered in WP2 is provided as a result of an extensive study of the state-of-the-art and in light of the user requirements and use cases defined in WP1.

Different processing algorithms and analysis methods have been presented for the financial data streams as well as for the social web streams. Furthermore, initial and preliminary results of the first implementations or evaluations of the selected methods are also reported in the respective sections. As a proof-of-concept, some of the algorithms have been implemented or adjusted to be executed as Apache Storm (sub) topologies that are then integrated in the QualiMaster priority pipeline in WP5. This includes the Hayashi-Yoshida Correlation Estimator as well as the SVM and SentiWordNet sentiment classifiers.

The next steps in WP2 will be the further development and optimization of the algorithms and methods presented in this deliverable towards the completing the first prototype (version 1), which is deliverable D2.2 (due in July 2015). Extensive experimental evaluations will be conducted to have a better estimation of the quality and performance of each of the individual members of the proposed families. In addition, measurements of the scalability of the developed algorithms using a computing cluster with several computing nodes will be evaluated using real-time streams of financial and web data streams.

Finally, several strategies to combine financial and web data streams in a meaningful way in order to support the financial risk analysis application will be investigated and evaluated.

References

[1] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. J. Algorithms, 55(1):58–75, 2005.

[2] G. Cormode, M. Garofalakis, P. Haas, C. Jermaine: Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. Foundations and Trends in Databases 4(1-3): 1-294, 2012.

[3] V. Mattiussi, M. Tumminello, G. Iori, R. Mantegna: Comparing correlation matrix estimators via Kullback-Leibler divergence. Available at SSRN 1966714, 2011.

[4] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking Join and Self-Join Sizes in Limited Storage. J. Comput. Syst. Sci., 64(3):719–747, 2002.

[5] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. J. Comput. Syst. Sci., 58(1):137–147, 1999.

[6] P. Flajolet and G. N. Martin. Probabilistic Counting Algorithms for Data Base Applications. J. Comput. Syst. Sci., 31(2):182–209, 1985

[7] T. Hayashi and N. Yoshida. On Covariance Estimation of Non-synchronously Observed Diffusion Processes. Bernoulli 11-2, 359-379, 2005.

[8] C. Olston, J. Jiang, and J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. In SIGMOD Conference, pages 563–574, 2003.

[9] G. Cormode and M. Garofalakis. Approximate continuous querying over distributed streams. ACM Trans. Database Syst., 33(2), 2008.

[10] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic Aggregates in a Networked World: Distributed Tracking of Approximate Quantiles. In SIGMOD Conference, pages 25–36, 2005.

[11] D. Keren, I. Sharfman, A. Schuster, and A. Livne. Shape Sensitive Geometric Monitoring. IEEE Trans. Knowl. Data Eng., 24(8):1520–1535, 2012.

[12] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In SIGMOD Conference, pages 301–312, 2006.

[13] N. Giatrakos, A. Deligiannakis, M. N. Garofalakis, I. Sharfman, and A. Schuster. Prediction-based geometric monitoring over distributed data streams. In SIGMOD Conference, pages 265–276, 2012.

[14] M. Garofalakis, D. Keren, and V. Samoladas. Sketch-based Geometric Monitoring of Distributed Stream Queries. In PVLDB 6(10), pages 937-948, 2013.

[15] M. Muralikrishna, D. DeWitt: Equi-Depth Histograms For Estimating Selectivity Factors For Multi-Dimensional Queries. In SIGMOD Conference, pages 28-36, 1988.

[16] V. Poosala, Y. Ioannidis: Selectivity Estimation Without the Attribute Value Independence Assumption. In VLDB, pages 486-495, 1997.

[17] H. Wang, K. Sevcik: A multi-dimensional histogram for selectivity estimation and fast approximate query answering. In CASCOM, pages 328-342, 2003.

[18] Ibbett, Roland N., and Nigel P. Topham. The Architecture of High Performance Computers. Macmillan, 1982.

[19] E. Aramaki, S. Maskawa, and M. Morita. Twitter Catches The Flu: Detecting Influenza Epidemics using Twitter. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP '11, 2011.

[20] J. Bollen and H. Mao. Twitter Mood as a Stock Market Predictor. IEEE Computer, 44(10):91–94, 2011.

[21] R. P. Schumaker and H. Chen. Textual analysis of stock market prediction using breaking financial news: The AZFin text system. ACM Trans. Inf. Syst., 27(2):12:1–12:19, March 2009.

[22] J. Lee, S. Nemati, I. Silva, B. Edwards, J. Butler, A. Malhotra: Transfer entropy estimation and directional coupling change detection in biomedical time series. In Biomedical engineering online 11 (1), pages 1-17, 2012.

[23] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In Proceedings of WWW'2010, 2010.

[24] F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing user modeling on twitter for personalized news recommendations. In Proceedings of the 19th international conference on User modeling, adaption, and personalization, UMAP'11, pages 1–12, Berlin, Heidelberg, 2011. Springer-Verlag.

[25] Harrington, P. 2012. Machine Learning in Action. In Manning Publications Co.

[26] Demartini, G., Siersdorfer, S., Chelaru, S., and Nejdl, W. Analyzing Political Trends in the Blogosphere. In Proceedings of the 5th International Conference on Weblogs and Social Media, ICWSM 2011.

[27] Davis, D., and Goadrich, M. The Relationship Between Precision-Recall and ROC Curves. In Proceedings of the 23rd International Conference on Machine Learning, ICML 2006.

[28] Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10 (2002), EMNLP 2002, Association for Computational Linguistics, pp. 79–86.

[29] Thomas, M., Pang, B., and Lee, L. Get out the vote: determining support or opposition from congressional floor-debate transcripts. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (2006), EMNLP '06, Association for Computational Linguistics, pp. 327–335.

[30] Baeza-Yates, R. A., and Ribeiro-Neto, B. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[31] Siersdorfer, S., Chelaru, S., Nejdl, W., and San Pedro, J. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In Proceedings of the 19th International Conference on World Wide Web (2010), WWW '10, ACM, pp. 891–900.

[32] Vapnik, V. N. The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., 1995.

[33] Cortes, C., and Vapnik, V. Support-vector networks. Mach. Learn. 20, 3 (Sept. 1995), 273–297.

[34] Drucker, H., Wu, D., and Vapnik, V. N. Support vector machines for spam categorization. IEEE TRANSACTIONS ON NEURAL NETWORKS 10, 5 (1999), 1048–1054.

[35] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. Inductive learning algorithms and representations for text categorization. In Proceedings of the seventh international conference on Information and knowledge management (1998),CIKM '98, ACM, pp. 148–155.

[36] Joachims, T. Text categorization with suport vector machines: Learning with many relevant features. In Proceedings of the 10th European Conference on Machine Learning (1998), ECML '98, pp. 137–142.

[37] Manning, C. D., Raghavan, P., and Schütze, H. Introduction to Information Retrieval. Cambridge University Press, 2008.

[38] Aggarwal, C., and Zhai, C. A survey of text classification algorithms. In Mining Text Data, C. C. Aggarwal and C. Zhai, Eds. Springer US, 2012, pp. 163–222.

[39] Chang, C.-C., and Lin, C.-J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2 (2011), 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[40] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: An update. SIGKDD Explor. Newsl. 11, 1 (Nov. 2009), 10–18.

[41] Bishop, C. M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[42] Esuli, A., and Sebastiani, F. Sentiwordnet: A publicly available lexical resource for opinion mining. In In Proceedings of the 5th Conference on Language Resources and Evaluation (2006), LREC '06, pp. 417–422.

[43] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. Sentiment in short strength detection informal text. JASIST 61, 12 (2010), 2544–2558.

[44] Denecke, K. Using sentiwordnet for multilingual sentiment analysis. In ICDE Workshops (2008), pp. 507–512.

[45] Kucuktunc, O., Cambazoglu, B. B., Weber, I., and Ferhatosmanoglu, H. A large-scale sentiment analysis for yahoo! answers. In Proceedings of the 5th ACM International Conference on Web Search and Data Mining (2012), WSDM '12, ACM, pp. 633–642.

[46] Vural, A. G., Cambazoglu, B. B., and Senkul, P. Sentiment-focused web crawling. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management (2012), CIKM '12, ACM, pp. 2020–2024.

[47] Pak, A., and Paroubek, P. Twitter as a corpus for sentiment analysis and opinion mining. In Proceedings of the Seventh International Conference on Language Resources and Evaluation LREC 2010.

[48] Bermingham, A., and Smeaton, A. F. Classifying sentiment in microblogs: is brevity an advantage? In Proceedings of the 19th ACM International Conference on Information and Knowledge Management (2010), CIKM '10, ACM, pp. 1833–1836.

[49] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., Passonneau, R. Sentiment analysis of Twitter data. In Proceedings of the Workshop on Languages in Social Media, LSM 2011, pp. 30-38.

[50] Connor, B.O.; Balasubramanyan, R.; Routledge, B.R.; Smith, N.A. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In Proceedings of the Fourth International AAAI Conference onWeblogs and SocialMedia,Washington, DC, USA, 23–26May 2010; pp. 122–129.

[51] Nguyen, T.; Phung, D.; Adams, B.; Venkatesh, S. Event extraction using behaviors of sentiment signals and burst structure in social media. Knowl. Inf. Syst. 2013, 37, 279–304.

[52] Thelwall,M.; Buckley, K.; Paltoglou, G. Sentiment in Twitter events. J. Am. Soc. Inf. Sci. Technol. 2011, 62, 406–418.

[53] Yunyue Zhu and Dennis Shasha. 2002. StatStream: statistical monitoring of thousands of data streams in real time. In Proceedings of the 28th international conference on Very Large Data Bases (VLDB '02). VLDB Endowment 358-369.

[54] Balog, K.; Mishne, G.; de Rijke, M. Why are They Excited?: Identifying and Explaining Spikes in Blog Mood Levels. In Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations, Trento, Italy, 3–7 April 2006; Association for Computational Linguistics: Stroudsburg, PA, USA, 2006; pp. 207–210.

[55] Akcora, C.G.; Bayir, M.A.; Demirbas, M.; Ferhatosmanoglu, H. Identifying Breakpoints in Public Opinion. In Proceedings of the FirstWorkshop on Social Media Analytics,Washington, DC, USA, 25 July 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 62–66.

[56] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In Proceedings of the 19th international conference on World wide web, WWW '10, pages 851{860, New York, NY, USA, 2010. ACM. URL: http://doi.acm.org/10.1145/1772690. 1772777, doi:10.1145/1772690.1772777.

[57] S. Khan, S. Bandyopadhyay, A. Ganguly, S. Saigal, D. Erickson, V. Protopopescu, G. Ostrouchov: Relative performance of mutual information estimation methods for quantifying the dependence among short and noisy data. Civil and Environmental Engineering Faculty Publication (paper 3), 2007.

[58] Zheng Liu, Di Wu, Yiping Ke, Jeffrey Xu Yu: Detecting Priming News Events. CoRR abs/1201.3458

[59] L. Chollete and C. Ning: Asymmetric Dependence in US Financial Risk Factors?, UiS Working Papers in Economics and Finance, University of Stavanger, 2010.

[60] S. Markose, S. Giansante, M. Gatkowski, A. Shaghaghi: Too Interconnected To Fail: Financial Contagion and Systemic Risk In Network Model of CDS and Other Credit Enhancement Obligations of US Banks, 2009.

[61] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. Sentiment Analysis of Twitter Data. In Proceedings of the Workshop on Languages in Social Media, LSM 2011.

[62] Go, A., Bhayani, R., and Huang, L. Twitter Sentiment Classification using Distant Supervision. In Stanford Technical Report, 2009.

[63] Kouloumpis, E., Wilson, T., and Moore, J. Twitter Sentiment Analysis: The Good the Bad and the OMG! In Proceedings of the International AAAI Conference on Weblogs and Social Media, ICWSM 2011.

[64] O'Hare, N., Davy, M., Bermingham, A., Ferguson, P., Sheridan, P., Gurrin, C., Smeaton, A. Topic-Dependent Sentiment Analysis of Financial Blogs. In TSA'09 - 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion Measurement

[65] Ruiz-Martinez, J.M., Valencia-Garcia, R., and Garcia-Sanchez, F. Semantic-Based Sentiment analysis in financial news. In Workshop on Finance and Economics on the Semantic Web (FEOSW 2012).

[66] Smailovic, J., Grcar, M., Lavrac, N., and Znidarsic, M. Stream-based active learning for sentiment analysis in the financial domain.In Information Sciences Journal, Vol. 285, pp. 181.203, 2014.

[67] Chelaru, S., Altingovde, I.S., Siersdorfer, S., and Nejdl, W. Analyzing, Detecting, and Exploiting Sentiment in Web Queries. ACM Transactions on the Web (Tweb), Vol. 8, pp. 6:16:28, December 2013.

[68] Pang, B., and Lee, L. Opinion mining and sentiment analysis. Foundations Trends Information Retrieval. 2, 1-2 (Jan. 2008).

[69] Bockhorst, J., and Craven, M. Markov networks for detecting overlapping elements in sequence data. In Neural Information Processing Systems 17 (NIPS), MIT Press 2005.

[70] D. K. Elson, N. Dames, and K. R. McKeown. Extracting social networks from literary fiction. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 138–147, Stroudsburg, PA, USA, 2010.

[71] C. Karbeyaz, E. Can, F. Can, and M. Kalpakli. A content-based social network study of evliya ¸Celebis seyahatname-bitlis section. In E. Gelenbe, R. Lent, and G. Sakellari, editors, Computer and Information Sciences II, pages 271–275. Springer London, 2012.

[72] P. Bonaldi, A. Hortacsu, and J. Kastl: An Empirical Analysis of Systemic Risk in the EURO-Zone". Manuscript. Stanford University, 2014.

[73] P. Fraternali, M. Tagliasacchi, M. Melenhorst, J. Novak, I. Micheel, E. Harloff, and J. G. Moron. Building the social graph of the history of European integration - A pipeline for humanist-machine interaction in the digital humanities. In Social Informatics - SocInfo 2013 International Workshops, QMC and HISTOINFORMATICS, Kyoto, Japan, November 25, 2013, Revised Selected Papers, pages 86–99, 2013.

[74] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR '06, pages 137–143, New York, NY, USA, 2006. ACM.

[75] H. A. Kautz, B. Selman, and M. A. Shah. The hidden web. AI Magazine, 18(2):27–36, 1997.

[76] Y. Matsuo, J. Mori, M. Hamasaki, K. Ishida, T. Nishimura, H. Takeda, K. Hasida, and M. Ishizuka. Polyphonet: An advanced social network extraction system from the web. In Proceedings of the 15th International Conference on World Wide Web, WWW '06, pages 397–406, New York, NY, USA, 2006. ACM.

[77] X. Canaleta, P. Ros, A. Vallejo, D. Vernet, and A. Zaballos. A system to extract social networks based on the processing of information obtained from internet. In Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence, pages 283–292, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[78] P. Mika. Flink: Semantic web technology for the extraction and analysis of social networks. Web Semant., 3(2-3):211–223, Oct. 2005.

[79] M. K. M. Nasution and S. A. Noah. Superficial method for extracting social network for academics using web snippets. In Rough Set and Knowledge Technology - 5th International Conference, RSKT 2010, Beijing, China, October 15-17, 2010. Proceedings, pages 483–490, 2010.

[80] J. He, Y. Liu, Q. Tu, C. Yao, and N. Di. Efficient entity relation discovery on web. Journal of Computational Information Systems3, 2:203–213, 2007.

[81] R. Nuray-Turan, Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting web querying for web people search in weps2. In 2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference, 2009.

[82] P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006, 2006.

[83] M. Pasca. Acquisition of categorized named entities for web search. In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, pages 137–145, New York, NY, USA, 2004. ACM.

[84] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In Proceedings of the 13th International Conference on World Wide Web, WWW '04, pages 462–471, New York, NY, USA, 2004. ACM.

[85] Jheser Guzman and Barbara Poblete. 2013. On-line relevant anomaly detection in the Twitter stream: an efficient bursty keyword detection model. In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description (ODD '13). ACM, New York, NY, USA, 31-39. DOI=10.1145/2500853.2500860 http://doi.acm.org/10.1145/2500853.2500860

[86] Chi-Chun Pan and Prasenjit Mitra. 2011. Event detection with spatial latent Dirichlet allocation. In Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries (JCDL '11).

ACM, New York, NY, USA, 349-358. DOI=10.1145/1998076.1998141 http://doi.acm.org/10.1145/1998076.1998141

[87] Milne, D., & Witten, I. H. (2013). An open-source toolkit for mining Wikipedia.Artificial Intelligence, 194, 222-239.

[88] Milne, David, and Ian H. Witten. "Learning to link with wikipedia." Proceedings of the 17th ACM conference on Information and knowledge management. ACM, 2008.

[89] Ratinov, Lev, et al. "Local and global algorithms for disambiguation to wikipedia." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.

[90] Dlugolinský, Štefan, et al. "Combining Named Entity Recognition Methods for Concept Extraction in Microposts." Making Sense of Microposts (# Microposts2014) (2014).

[91] Chelaru, S., Herder, E., Naini, K. D., & Siehndel, P. (2014, September). Recognizing skill networks and their specific communication and connection practices. In Proceedings of the 25th ACM conference on Hypertext and social media (pp. 13-23). ACM.

[92] Siehndel, Patrick, and Ricardo Kawase. "TwikiMe! User profiles that make sense." (2012).

[93] Kawase, Ricardo, Patrick Siehndel, and Bernardo Pereira Nunes. "Supporting Contextualized Information Finding with Automatic Excerpt Categorization."Procedia Computer Science 35 (2014): 551-559.

[94] Siehndel, P., Kawase, R., Herder, E., & Risse, T. Identifying Topic-Related Hyperlinks on Twitter. Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference (ISWC 2014)

[95] Kawase, Ricardo, et al. "Exploiting the wisdom of the crowds for characterizing and connecting heterogeneous resources." Proceedings of the 25th ACM Conference on Hypertext and Social Media, Santiago, Chile. 2014.

[96] S. Straetmans, S. Chaudhry: Tail risk and systemic risk of us and eurozone financial institutions in the wake of the global financial crisis. Working Paper, 2013.

[97] N. Champagnat, M. Deaconu, A. Lejay, N. Navet, S. Boukherouaa: An empirical analysis of heavy-tails behavior of financial data: The case for power laws, 2013.

[98] D. Lautier, F. Raynaud.: Systemic Risk and Complex Systems: A Graph-Theory Analysis: Econophysics of Systemic Risk and Network Dynamics, pages 19-37, 2013.

[99] N. Huth, A. Frédéric: High frequency lead/lag relationships—Empirical facts. Journal of Empirical Finance 26, pages 41-58, 2014.

[100] T. Schreiber: Measuring Information Transfer. Phys. Rev. Lett., pages 461-464, 2000.

[101] M. Billio, M. Getmansky, A. Lo, L. Pelizzon: Econometric measures of connectedness and systemic risk in the finance and insurance sectors. Journal of Financial Economics, volume 104, issue 3, pages 535-559, 2012.

[102] European Central Bank: Recent advances in modeling systemic risk using network analysis (http://www.ecb.europa.eu/pub/pdf/other/modellingsystemicrisk012010en.pdf), 2010.

[103] G. López-Espinoza, A. Moreno, A. Rubia, L. Valderrama: Systemic risk and asymmetric responses in the financial industry. Working Paper, 2012.

[104] B. Bradley, M. Taqqu: Financial risk and heavy tails. Handbook of Heavy-Tailed Distributions in Finance, ST Rachev, pages 35-103, 2003.

[105] ISO, CD 25010.2, Software engineering-Software product Quality Requirements and Evaluation(SQuaRE) Quality model, http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733, 2011

[106] L. Hong, G. Convertino, and E. H. Chi. Language matters in twitter: A large scale study. In IWSM, 2011.

[107] Liu, J., Zhang, F., Song, X., Song, Y. I., Lin, C. Y., & Hon, H. W. (2013, February). What's in a name?: an unsupervised approach to link users across communities. In Proceedings of the sixth ACM international conference on Web search and data mining (pp. 495-504). ACM.

[108] Yimam-Seid, Dawit, and Alfred Kobsa. "Expert-finding systems for organizations: Problem and domain analysis and the DEMOIR approach."Journal of Organizational Computing and Electronic Commerce 13.1 (2003): 1-24.

[109] Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N., & Gummadi, K. (2012, August). Cognos: crowdsourcing search for topic experts in microblogs. InProceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (pp. 575-590). ACM.

[110] Guy, I., Avraham, U., Carmel, D., Ur, S., Jacovi, M., & Ronen, I. (2013, May). Mining expertise and interests from social media. In Proceedings of the 22nd international conference on World Wide Web (pp. 515-526). International World Wide Web Conferences Steering Committee.

[111] Kumar, R., Novak, J., & Tomkins, A. (2010). Structure and evolution of online social networks. In Link mining: models, algorithms, and applications (pp. 337-357). Springer New York.

[112] Abel, F., Herder, E., & Krause, D. (2011). Extraction of professional interests from social web profiles. Proc. UMAP, 34.

[113] Abel, F., Herder, E., Houben, G. J., Henze, N., & Krause, D. (2013). Cross-system user modeling and personalization on the social web. User Modeling and User-Adapted Interaction, 23(2-3), 169-209.