ALLDATA 2021

The Seventh International Conference on Big Data, Small Data, Linked Data and Open Data

ISBN: 978-1-61208-842-6

April 18 - 22, 2021

ALLDATA 2021 Editors

Yoshihisa Udagawa, Tokyo University of Information Sciences, Japan

Oana Dini, IARIA, EU/USA

ALLDATA 2021

Forward

The Seventh International Conference on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2021) continued a series of events bridging the concepts and the communities devoted to each of data categories for a better understanding of data semantics and their use, by taking advantage from the development of Semantic Web, Deep Web, Internet, non-SQL and SQL structures, progresses in data processing, and the new tendency for acceptance of open environments.

The volume and the complexity of available information overwhelm human and computing resources. Several approaches, technologies and tools are dealing with different types of data when searching, mining, learning and managing existing and increasingly growing information. From understanding Small data, the academia and industry recently embraced Big data, Linked data, and Open data. Each of these concepts carries specific foundations, algorithms and techniques, and is suitable and successful for different kinds of application. While approaching each concept from a silo point of view allows a better understanding (and potential optimization), no application or service can be developed without considering all data types mentioned above.

We take here the opportunity to warmly thank all the members of the ALLDATA 2021 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to ALLDATA 2021. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions. We also thank the members of the ALLDATA 2021 organizing committee for their help in handling the logistics of this event.

ALLDATA 2021 Chairs

ALLDATA 2021 Steering Committee
Yoshihisa Udagawa, Tokyo University of Information Sciences, Japan
Mamadou H. Diallo, Naval Information Warfare Center (NIWC) Pacific, U.S. Department of Defense, San Diego, CA, USA
Fernando Perales, JOT INTERNET MEDIA, Madrid, Spain

ALLDATA 2021 Advisory Committee
Venkat N. Gudivada, East Carolina University, USA
Sandjai Bhulai, Vrije Universiteit Amsterdam, the Netherlands
Jerzy Grzymala-Busse, University of Kansas, USA
Fabrice Mourlin, Université Paris-Est Créteil Val de Marne, France

ALLDATA 2021 Industry/Research Advisory Committee

Stephane Puechmorel, ENAC, France
Hanmin Jung [정 한 민], Korea Institute of Science and Technology Information, South Korea

ALLDATA 2021 Publicity Chairs
Lorena Parra, Universitat Politecnica de Valencia, Spain
Jose Luis García, Universitat Politecnica de Valencia, Spain

ALLDATA 2021

Committee

Table of Contents

QoS-Aware Self-Adapting Resource Utilisation Framework for Distributed Stream Management Systems

Tarjana Yagnik
Computing, Engineering and Media
De Montfort University
Leicester, United Kingdom
email: tarjana.yagnik@dmu.ac.uk

Feng Chen
Computing, Engineering and Media
De Montfort University
Leicester, United Kingdom
email: fengchen@dmu.ac.uk

Laleh Kasraian
Computing, Engineering and Media
De Montfort University
Leicester, United Kingdom
emai: laleh.kasraian@dmu.ac.uk

***Abstract*— The last decade witnessed plenty of Big Data processing and applications including the utilisation of machine learning algorithms and techniques. Such data need to be analysed under specific Quality of Service (QoS) constraints for certain critical applications. Many frameworks have been proposed for QoS management and resource allocation for the various Distributed Stream Management Systems (DSMS), but lack the capability of dynamic adaptation to fluctuations in input data rates. This paper presents a novel QoS-Aware, Self-Adaptive, Resource Utilisation framework, which utilises instantaneous reactions with proactive actions. This research focuses on the load monitoring and analysis parts of the framework. By applying real-time analytics on performance and QoS metrics, the predictive models can assist in adjusting resource allocation strategies. The experiments were conducted to collect the various metrics and analyse them to reduce their dimensions and identify the most influential ones regarding the QoS and resource allocation schemes.**

***Keywords*—Data stream management; Distributed stream processing; quality of service; resource allocation; prediction; scheduling.**

I. Introduction

During the last decade, a new category of data-intensive systems and applications has emerged and has been recognised by the researchers and industry professionals in the data science field. A data stream is defined as a real-time, continuous, ordered sequence of data items [1].

The systems deployed to manage data streams are called Data Stream Management Systems (DSMS) [2]. A common definition of such system is that it is the system especially developed and assembled to process continuous queries on dynamic and ever-changing data streams. DSMSs are totally different from the traditional Database Management Systems (DBMSs) in that traditional database management systems expect the data to be persistent in the system and the queries to be dynamic while within the DSMSs paradigm it is expected to have dynamic unbounded data streams and the queries are submitted on those streams as persistent queries.

Quality of Service (QoS) [3] is identified as an important attribute of overall performance measure of any system. One of the main challenges within the QoS management of DSMS systems, is how to efficiently and effectively deliver predefined QoS requirements. Within a system that has multiple queries submitted over different data streams, different queries would have totally different QoS requirements and constraints.

The DSMS should have the ability to distribute and allocate physical computing resources between those queries and fulfil the required QoS specification in a fair and square manner. The DSMS utilises a mechanism called *scheduling strategy* to allocate the available resources based on the various queries' QoS requirements.

There are two main issues [4] that have to be dealt with when managing resource allocation of distributed data stream systems and they are:

- The ability of the system to allocate or release computing resources to meet an application workload and specified quality of service requirements; and
- Devising and performing the relevant optimisation actions to alter the system configuration during the runtime to utilise any additional capacity or release previously allocated resources to guarantee the agreed-upon (or specified) end-to-end quality of service levels of the system critical applications.

A framework for QoS-Aware Self-Adaptive Resource Utilisation management of data stream management systems is presented in this paper. QoS is tightly connected between all system components, i.e., each component within the system contributes to the overall quality of service perceived by the system applications. A comprehensive usage model that comprises mixing *instantaneous reactions* with *proactive actions* is incorporated within the proposed framework. Instantaneous reactions include applying real-time analytics on collected performance and QoS metrics of each component of the system (worker profiling) before such data becomes obsolete and loses its value. Proactive actions are the processes of applying predictive models that further assist in decision making and resource allocation planning and scheduling within the system in a real-time streaming environment.

This paper is organised as follows. Section II gives a background and a quick overview of the current research activities and proposed QoS-aware, resource utilisation and scheduling frameworks in the field. Section III introduces the proposed framework and gives an overview of the architecture and the different components within the framework. Section IV explains the various performance metrics and QoS metrics that will be utilised as part of the framework. Section V gives an overview of the Apache Storm, its terminologies and techniques as a case study of the experimental verification of the proposed framework. Section VI details the experiments that have been conducted to collect the various metrics and analyse them to reduce their dimensions and identify the most influential ones. The paper is concluded in Section VII and planned future work is also outlined.

II. Background

Although there have been so many papers and projects that have researched and discussed the various performance aspects in DSMSs [5]–[8], many of those papers had concentrated their attention on the study of the different individual components within the DSMS. There has been an evident lack of studying and investigating the QoS of the whole DSMS system.

A. Resource-Aware and Traffic-Aware Scheduling Schemes

A great deal of research [9]–[13] has been carried over the last decade focusing on scheduling of streaming applications. We present in this section a comparison between the most related resource- and traffic-aware schedulers from different perspectives. Some of the traditional schemes proposed for resource allocation and scheduling [12] rely on measuring a set of performance metrics and consequently make some adjustment to the scheduling strategy or resource allocation schemes based on a comparison with a pre-defined set of constrains and measurements thresholds.

Those schemes suffer from the lack of dynamic adaptability to the real-time and timely-constrained fluctuations in the system workload and data input patterns. Schedulers within DSMSs can be categorised into two classes:

- Static schedulers, such as the default scheduler in Storm [13], where executers are assigned as evenly as possible between all workers and tasks are assigned on a round robin fashion to different task executers. Another type of static schedulers is the resource-aware scheduler called R-Storm [8], proposed by Peng et. al. and implemented in the latest versions of Storm (v2.0 and beyond).
- Dynamic schedulers, where the scheduler plan or strategy is adjusted during run-time and the system configuration is changed to reflect the main scheduler goal or/and quality of service requirements of the system. These schedulers can be classified into the following three categories:
  - Throughput oriented schedulers, as being presented in [15] and [16].
  - Latency-oriented schedulers, as those presented in [17] [18] and
  - Communication reduction schedulers, as the three schedulers presented in [14] [19] [20].

B. Scheduling Optimisation Approaches

The scheduling optimisation approaches within DSMS platforms can be classified based on the main objectives of its scheduling strategy. They are either built based on minimising or maximising CPU utilisation, memory usage, throughput or satisfy certain pre-defined quality-of-service levels. The various data streaming scheduling strategies can be classified based on their scheduling objectives.

1) Minimising Memory Consumption: The memory consumption within a DSMS depends mainly on the size of the operator's buffer in addition to the current internal state of each operator. The first approach is vital to the process of transferring data tuples from each operator to the next one. The second approach is governed by the number of tuples that the operator needs to fulfil its data grouping requirements(join, aggregation, duplicate elimination, etc.). The scheduler prioritise its placement of the operator based on the ability of the operator to reduces the amount of data exchanged and processed within the shortest period of time. This strategy has been the core scheduling strategy in Aurora [21] and STREAM [22] systems and is called Min-Memory and Greedy approaches, respectively.
2) Minimising CPU Utilisation: Reduction in the number of calls to an operator results in the reduction of CPU usage and consequently reduce the overhead associated with the scheduling activities. Examples of such approaches are the Aurora's super-boxes and tuple trains [21] as implemented within the QStream's micro period method [23].
3) Maximising Quality of Service: Maximum QoS metrics are mainly reflected within the DSMS system with their ability to minimise the total output delay and with maximising the system overall throughput. Both methods are considered below:
   - Minimising Delay: This strategy is corner stoned with the well-known First In First-Out (FIFO) allocation strategy. The optimisation objective is fulfilled by pushing or pulling certain data tuples through the network of operators as fast as possible. As a result of such optimisation, the delay time (complete tuple latency) is shortened so it fells within the acceptable range specified by the query submitter or the system application.
   - Maximising Throughput: Whenever a scheduling strategy aims at minimising the overall system CPU utilisation, it automatically enables the system to attain a higher output throughput under a given resource availability.
4) Scheduling for Specified Level of Quality-of-Service: In order to guarantee certain levels of quality of service for critical applications such as health monitoring system, critical infrastructure applications, military command/control applications, it is vital to the proper operation of the system to ensure that such strict QoS levels are always met and provided to such application under various operation conditions. A minimum level of throughput can be guaranteed by certain DSMS platforms so it has to ensure the system will not cross a maximum output delay limits.

C. QoS and Resource-Aware DSMS Frameworks

There are many DSMS that have emerged during the last decade. Apache Storm [13] is one of the most popular systems in this area. It had attracted both industries and researchers' attention from early stages of its development. Aurora [21] and QStream [23] are amongst the main DSMS frameworks that incorporate some levels of QoS assurance from within. A limitation in those systems is that the QoS specifications are provided only for the output streams. Other approaches, such as Borealis [23] and the one proposed by Klein and Lehner [24], manage QoS specifications at the operator level.

D. Self-Adapting Approaches

The approach presented in [25] is a self-adaptive, resource management framework for software services in the Cloud. This approach utilises historical data to build a prediction model to predict the QoS value of a certain management operation. Serhani et. al. [26] proposed a layered-architecture, self-adapting framework that supports end-to-end workflow management with the ability to adapt its configuration and quality specifications and enforcement mechanisms. It uses a declarative, rule-based cloud services orchestration approach to detect event patterns and utilise machine learning algorithms to build a meta-model for monitoring and adaptation of workflow within cloud services. The approach differs from our proposal by focusing into application-based and content-based end-to-end QoS attributes. Cardellinin et. al. [27] proposed a framework that aimed at extending Apache Storm to operate within FOG computing environment. This approach mainly focuses on QoS attributes related to the geographically distributed network environment.

Our approach differs substantially from those approaches by looking deeply into individual component's time-based QoS and performance metrics, analysing those metrics to reduce their dimensions in order to use them to build a dynamic prediction model through incremental learning algorithms along with ensemble learning and abnormality detection.

III. The Proposed Framework

Throughout this paper, a QoS-Aware self-adapting resource allocation framework is proposed. It enables the collection of dynamic performance and QoS metrics for each component within the DSMS submitted query plan or topology. This work contributes to the growing interest in introducing QoS considerations in the data streams domain and helps in supporting further classes of QoS-sensitive streaming applications at scale.

A. Framework Architecture

Figure 1 shows the system architecture of our proposed framework. The framework is composed of four main components: QoS Management, Performance Metrics Management, Incremental Learning Prediction with Abnormality Detection, and the Dynamic Tuning/Scheduler Adapter Module. The following subsections elaborate on the structure, functionality and overall processes involved in each component

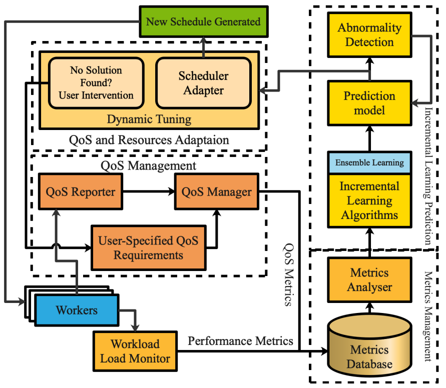


Figure 1. Framework Architectural design.

1) QoS Manager: The QoS Manager stores the various reports that it receives from its reporters located within the various components within the DSMS. For a given QoS constraint, the manager will keep all measurement data concerning the monitored topology component element during the current measurement window span (t) time units and consequently will discard the older data measurements. The QoS management is composed of the following two elements:
   - User QoS Specifications: Each application user should specify the QoS constraints that are needed to be fulfilled during the queries execution or computations over the input streams. This can be provided, through a high-level abstracted QoS interface, as QoS graphs or QoS tables regarding any of the main QoS metrics.
   - System QoS Metrics Collector: The master node within the cluster has global knowledge about each one of the pre-defined QoS requirements. It will inform each worker about where and when QoS metrics measurements should be collected. This will minimise the needed resources to collect such metrics and the computing power needed to analyse them on the fly so the computing overhead of this component is minimised. For every query data operator with user-specified QoS constraints, QoS metrics will be measured once during the configured time window called the measurement interval.
2) Performance Metrics Management: All performance metrics are measured and associated calculation are done over the specified window of time and then the aggregate results will be reported through a metrics collection pipeline to the Metrics Management Module. The module runs certain metrics normalisation, factor and correlation analysis to reduce the data dimensions and identify the most relevant set of metrics that can be fed to the next component which is the Incremental Learning module.
3) Incremental Learning Prediction with Abnormality Detection: The DSMS cluster nodes are affected by fluctuations that may be caused by the abnormal data rates,

network transmission and other factors, or the fact that some of the measured metrics and QoS observations don't fit into the prediction model.

4) Dynamic Tuning and Scheduler Adapter Module: The Scheduler Adapter is the component entrusted in making those decisions. The Scheduler adapter initiates necessary countermeasures in the form of modifications to the run-time scheduling strategy and system global configuration of the deployed cluster and worker node settings, until the specified QoS constraint has been met or there is no additional measures can be made in order to meet such constraint. In this case the result will be reported to the user to decide on the best action to be done (either by adding additional physical resources or relax the QoS specifications for this particular query).

B. Implementation Tools and Techniques

In order to validate the proposed framework and evaluate its performance, the following tools are being used to facilitate the development of a prototype that proves the concept and validate the effectiveness of the proposed framework. This includes the following in-house developed systems, open-source tools and related techniques:

1) Apache STORM.
2) QoS Management Module.
3) Performance Metrics Management Module.
4) WEKA [28] and MOA [29] ML Toolkits.
5) Performance Dynamic Tuning Module.

IV. Data Streams Metrics

The reporting of the metrics can be in the form of gauges, histograms and numeric values. Often these result in multiple metrics being uploaded to the reporting system, such as percentiles for a histogram, or rates for a meter.

A. Performance Metrics:

Throughout our experiments, the Metrics Management module collects the following metrics:

1) Data-Level Metrics: Two main metrics related to the stream data input are:
   - *Data Input Rate*: The rate of input data stream can be controlled by using the Data-Rate metric, by changing the spout parallelism with the Apache Storm cluster. The data rate of a data stream describes how many stream tuples per second occur in this stream.
   - *Data Delay*: Along with the Data Rate, a delay metric is defined as the time interval from a certain tuple arrives at the first input component of the topology and the time it leaves the time it is inserted to the next stream within the system.
2) Query-Plan/Topology Metrics: These metrics are related to the queuing behaviour of the system specially those related to how much time a tuple is setting waiting to be processes by the next component. Sample of the metrics within this category are:
   - *Window-Size* and *Sliding-Step* metrics: Those metrics denote the total number of input and output tuples within a sepecified time measurement window.
   - *Op-Selectivity*: This metric is defined as the total number of data tuples that are emitted relative to the number of data tuples consumed by the topology/query plan operator. For example, if an operator emits 2 tuples as a result of consuming one input tuple, then the Op-Selectivity of that particular operator is 2.
3) Scheduler-Level Metrics: The objectives of a distributed stream management scheduling algorithms is to maximise throughput and system resource utilisation and minimising the latency while trying to meet the user/application requirements and quality of service constraints. The *Uptime metric* measures the total computation time that is consumed by a running Java Virtual Machine process within the worker node.
4) Cluster Level Metrics: Apache storm version 2.2 provides an extensive set of cluster metrics which include the following categories:
   - *Cluster Metrics*: These are metrics that are reported through the nimbus daemon with metrics that report the state of the cluster and its various components.
   - *Supervisor Metrics*: Metrics associated with the supervisor, which launches the workers for a topology.
   - *UI Topology Metrics*: Metrics associated with a topology running in the cluster and reported through a single UI daemon. The metrics can be collected through the extensive REST UI API interface within the DSMS.

B. Quality of Service Metrics:

Within the context of this research, there are several parameters that are able to represent both the query submitter's performance satisfaction, as well as the system's throughput in a variety of scenarios. The experiments are designed in a way that takes into consideration the following most relevant QoS metrics. Those metrics give clear picture and understanding about the effective factors that are taken into consideration in regard to QoS and scheduling optimisation approaches as shown in Table I.

V. Apache Storm

Apache Storm is an open-source, real-time, scalable distributed and fault-tolerant Data Stream Processing System maintained by the Apache Software Foundation. On a physical Storm cluster deployment, there are several types of entities used to execute a topology as shown in Figure 2 and they are:

1) Task: It denotes an instance of a certain topology component (Spout or Bolt) or query plan operator.
2) Executor: which is used to execute one or more tasks related to the same operator.
3) Worker or Work Process: which runs one or more executers on the same topology.

TABLE I
QUALITY-OF-SERVICE (QOS) METRICS

Metric	How to Measure It
Total Throughput	This metric can be measured by calculating the total number of tuples processed within the specified amount of time called measurement window.
Memory Consumption	It is measured by the amount of memory in MB that is consumed by a certain component to execute its functions of the query plan as part of the whole memory consumption of the system.
CPU Utilisation	This metric is defined as the amount of CPU resources consumed by a certain query plan component where it is available on a certain task/executor and is represented by a point system.
Complete Latency	This denotes the average time for a tuple tree to be completely processed from end to end by the topology compoennets.
System Latency	It is measured by the amount of time the system takes to process each tuple or it denotes the time difference between the tuple input and its output from the system.
Processing Latency	This metric is measured by the time the system or any component within the system to process the tuple it receives and output it to the upstream for output or further processing.
Execution Latency	It can be measured by calculating the delay in time experienced by the system when it completely execute the tuple or tuple tree.

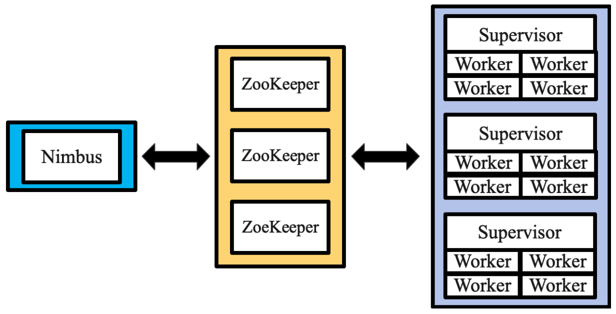


Figure 2. The various componenets and daemons within the Apache Storm Cluster Setup.

4) Worker node: It denotes a physical computational resource or simply a computer instance.
5) Zookeeper: a server application (Daemon) that is responsible for managing the cluster configuration parameters and enable distributed coordination between the different cluster nodes and processes.
6) Nimbus: which is the centralised management “brain” of the cluster and charged with the overall topology execution.
7) Supervisor: It is responsible for starting or terminating worker processes based on the Nimbus assignment and keep coordination with Nimbus for any fault-tolerance mechanisms to implement in case of node-failure

The processing of data streams within Storm is delegated to several types of components within the platform. Each component is responsible for executing simple tasks. The component within the Storm cluster that handles incoming streams of data is called Spout. The Spout function is to pass streams to another component called Bolt. Bolts are usually used to transform streaming data in different ways. The Bolt may store the data in certain form of storage or process it and pass it to another Bolt. The Storm cluster can be viewed as a chain of spouts and bolts arranged in a certain connections layout to form a Direct Acyclic Graph (DAG) called Topology.

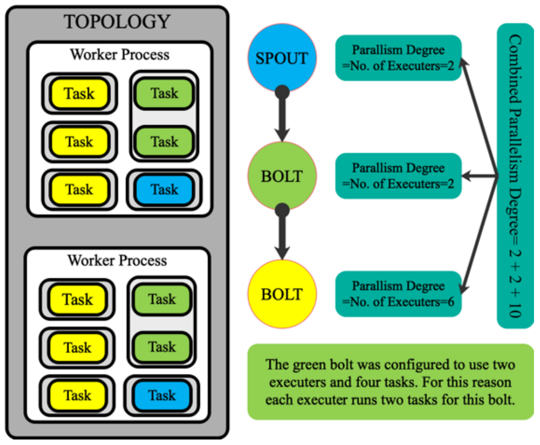

TOPOLOGY
Worker Process
Task
Task
Task
Task
Task
Task
Worker Process
Task
Task
Task
Task
Task
Task
SPOUT
BOLT
BOLT
Parallism Degree =No. of Executers=2
Parallism Degree =No. of Executers=2
Parallism Degree =No. of Executers=6
Combined Parallelism Degree= 2 + 2 + 10
The green bolt was configured to use two executers and four tasks. For this reason each executer runs two tasks for this bolt.

Figure 3. Mapping the logical layout of the WordCountTopology into physical worker nodes, worker tasks and executors within the Storm Cluster.

The illustration in Figure 3 shows how a simple topology would look like in operation within the Apache Storm Data Stream Processing System. One of the widely-used benchmark topologies within Storm is the WordCountTopology shown in Figure 4. This topology has been used to measure the various performance metrics and speed of messaging between spouts and bolts. The topology is composed of the following three simple components:

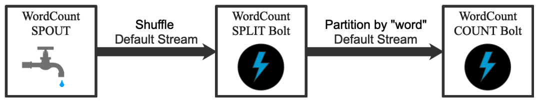


Figure 4. The logical layout of the WordCountTopology.

1) Spout: The *SPOUT* component within a topology will simply emit a stream of data tuples under the name “sentence” as a sequence of string value. To simplify the implementation of the experiment, the input data tuples are retrieved from a static list of sentences that is accessed repeatedly based on the input rate settings of the spout. The spout will repeat a loop of generating and then emitting one tuple for every sentence. In the real-world scenarios, such process is replaced by a data producing mechanism and sentences are received through certain APIs such as Twitter API or through topics producers as used in Apache Kafka or similar data ingestion platforms.
2) Split Bolt: The *SPLIT* bolt subscribes to the “default” stream of sentences emitted by the SPOUT. For each tuple received from the spout in the form of a sentence (stream of strings), it splits the sentence into words, and

emit a tuple for each word to the upward stream of the next connected component of the topology. It uses different stream grouping mechanisms available within Storm to deal with streaming to multiple components or join multiple streams into one main stream.

3) Count Bolt: The *COUNT* bolt subscribes to the output stream (also called default stream) coming from the split bolt. Its main function is to count how many times it has seen a particular word. Every time this bolt receives a tuple from the input stream, it will increment its counters accordingly.

VI. Experiments

In order to establish a solid baseline for our research, several experiments have been conducted using a complete apache storm installation in local mode where all services and daemons are installed in one machine as outlined above. Throughout the set of phase-I experiments, the goal is to measure the performance and QoS Metrics through the deployment of the Metrics Collector Module over the various components of the Apache Storm cluster and individual worker nodes.

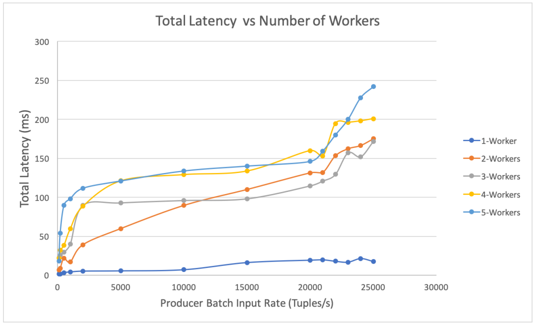


Figure 5. Total Latency of the WordCountTopology running on multiple workers within the same worker node.

1) Storm (In-process) Test Bed: To establish a baseline and investigate the extent of resource utilisation of the experiment test-bed, several throughput tests have been conducted to check how much data the test-bed can consume. Memory-Intensive and CPU-Intensive topologies are being used to generate a rich metrics dataset to be used in the next phase of our project. The experiments were run in local-mode to establish a base line with the setup and configuration of the various components of the Storm platform. The main machine used is a MacBook Pro laptop running Mac OS Catalina version 10.15.7. It is a 2.6 GHz Quad-Core Intel Core-i7 with 16 GB 2133 MHz LPDDR3 memory. The machine run the main Storm daemons (Nimbus, Supervisor and UI) in addition to a zookeeper daemon for coordination and instance management.

A. Preliminary Results and Analysis

1) Metrics Collection: A new metrics system has been introduced in Apache Storm version 2.2. The new system reports the different internal statistics (e.g., acknowledged, failed, emitted, transferred, queue metrics, etc.) as well as a new API for user defined metrics. Metrics related to the network-intensive topology are being used in the cluster setup as part of phase II experiments and will be integrated within the main metrics dataset.
During this experiment, the metrics collection process has focused on two levels:
    a) Topology Level metrics including the number of worker nodes, workers, memory allocation, cpu allocation, executors, tasks, input data rate, input throughput, emitted tuples, complete latency, maximum overall latency, acknowledged and failed tuples.
    b) Component level including the type (Spout/Bolt), executors, tasks, user cpu, system cpu, completed latency, executed tuples, acknowledged and failed tuples, execute latency, operator capacity, process latency, parallelism and congested component.
2) Metrics Measurement: In this set of experiments, topologies that measure the performance of the Apache Storm platform and collect its performance metrics based on the above descriptions and metrics definitions have been utilised. The topologies range from memory-intensive, cpu-intensive and network-intensive topologies.

During the normal operational activities where the available resources are able to handle the fluctuations in the data input rate and the delay/throughput levels are within acceptable

TABLE II
COMPLETE LATENCY FOR NORMAL INPUT

Batch Size	**Number of Workers**				
	1	2	3	4	5
100	1.67	6.87	22.19	21.72	18.22
200	1.40	8.59	25.70	32.07	54.00
500	3.30	21.54	29.47	38.45	89.62
1000	4.16	16.95	39.85	59.79	98.21
2000	5.20	39.01	89.62	88.45	111.5
5000	5.53	59.79	93.65	121.08	120.9
10000	7.08	89.52	96.36	129.08	133.7
15000	16.2	110.98	98.21	133.72	140.6
20000	19.2	131.27	114.7	159.85	146.0
21000	19.7	131.94	120.9	152.71	159.3
22000	17.8	153.23	129.4	194.67	180.5
23000	16.7	162.24	157.2	196.06	200.1
24000	21.3	166.43	152.1	198.76	227.8
25000	17.6	175	171.6	200.79	242.0

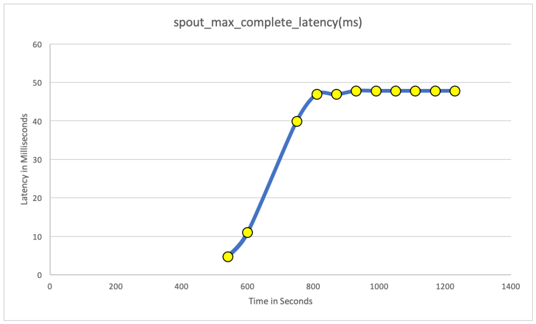


Figure 6. Spout maximum complete latency as reported by the Storm UI.

levels of the topology submitter (user). Figures 5, 6 and 7 show the results of running the topology using normal working conditions and high data rates and how such fluctuations affect the system's QoS.

Running the topology using several workers will introduce traffic between inter-workers and intra-workers which affects the performance and latency of the tuples considerably. Such behaviour is evident when comparing the latency using 1 worker and 2 workers. The latency also has a direct relationship with the way the Storm Scheduler places the physical executors and tasks within the same worker. For example, due to the fact that our topology has only three components, placing the executors and tasks of each component of the topology within the same worker will result in lower latency under high input rates with just 3 workers since the inter-executors traffic will be reduced considerably.

When the data input rate exceeds the computing capacities of the allocated resources where the QoS levels degrade dramatically specially the various latency parameters. This is evident in Figures 8 and 9 where the component execution capacity (defined as the number of executed tuples multiplied by the execution latency/(Observation Window Time)) reaches high levels and the buffers start to be full and tuples are dropped or other mechanisms like back-pressure signals are generated to limit the input rates to the topology data ingestion ports (spouts).

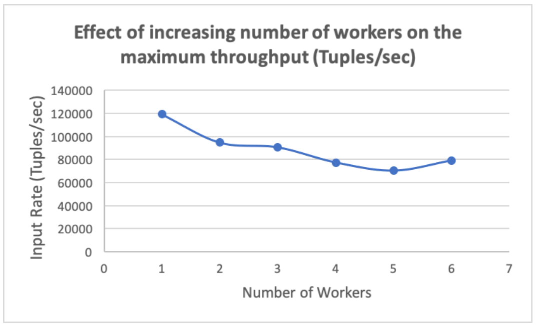


Figure 7. Maximum Input Throughput for the WordCountTopology running on multiple workers within the same worker node.

Maximum Input Throughput for the WordCountTopology running on multiple workers within the same worker node is presented in Table II. During the various runs of our testing topologies, it was observed that the processing power and output rates as well as the various latencies are performing reasonably well when executed in a single worker process within the same worker node. The data input rate to the topology's spout and corresponding worker process was somewhat constrained within a range (under 120k tuples/second).

When the input rate increases, more of the processing power would be used to either (1) drop the tuples instead of processing them or (2) enable the back-pressure techniques recently implemented in the latest versions of Storm to limit the input rate of the incoming data. This can affect the processing rate of the worker and may not be tolerated by certain applications or guaranteed QoS requirements of the application (such as critical infrastructure, military and vital health monitoring systems). A more detailed analysis of the individual worker processes and its components (spouts and bolts) surely will help create better mathematical models and identify bottlenecks and resource starvation or under-use to pinpoint the areas of modifications needed to improve the overall performance of the system.

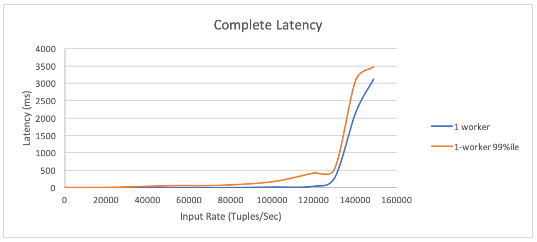
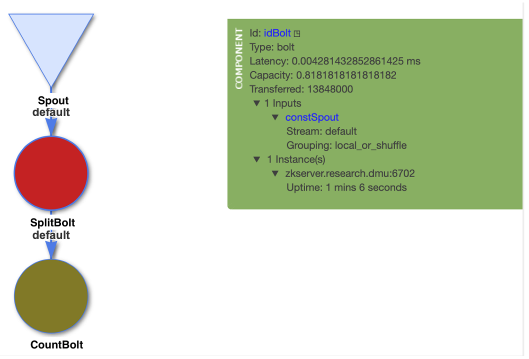


Figure 8. Complete Latency of the Topology as a function of Input Data Rate.



Id: idBolt
Type: bolt
Latency: 0.004281432852861425 ms
Capacity: 0.8181818181818182
Transferred: 13848000
▼ 1 Inputs
▼ constSpout
Stream: default
Grouping: local_or_shuffle
▼ 1 Instance(s)
▼ zkserver.research.dmu:6702
Uptime: 1 mins 6 seconds

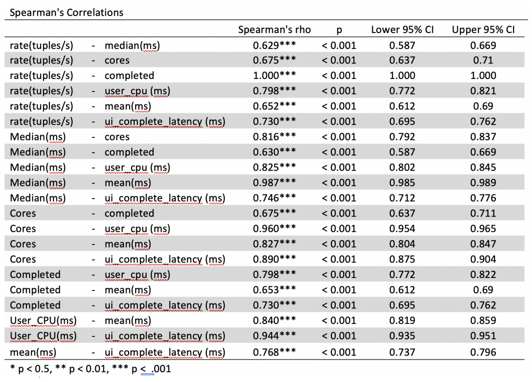
Figure 9. Congestion caused by high input data rates and its effects on the processing capacities of the various topology components.

B. Component Profiling

An improvement to the process is to profile the worker components automatically at runtime. Profiling each individual component (operator) with a topology is time consuming and generally a tedious job that will consume extra "precious" resources from the system and degrade its performance. If the characteristics of the individual component (spout and bolt) changes over time, then it is difficult to depend on the existing processing power, latencies and output throughput to produce suitable resource allocations.

Components can be profiled to monitor the amount of memory, cpu and network bandwidth and then correlate it with the number of tuples being processed in unit time. This will be used to estimate the maximum processing rate the worker can sustain over a time unit so prediction models can be built to anticipate for changes in input rate ahead of time and adjust the resources accordingly.

Figure 9 shows the visualisation of the topology components behaviour when components are congested. Storm deploys a mechanism of back-pressure techniques to limit the input rate in these cases instead of just dropping some of the input tuples.



Spearman's Correlations

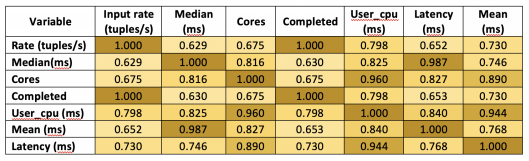
			Spearman's rho	p	Lower 95% CI	Upper 95% CI
rate(tuples/s)	-	median(ms)	0.629***	< 0.001	0.587	0.669
rate(tuples/s)	-	cores	0.675***	< 0.001	0.637	0.71
rate(tuples/s)	-	completed	1.000***	< 0.001	1.000	1.000
rate(tuples/s)	-	user_cpu (ms)	0.798***	< 0.001	0.772	0.821
rate(tuples/s)	-	mean(ms)	0.652***	< 0.001	0.612	0.69
rate(tuples/s)	-	ui_complete_latency (ms)	0.730***	< 0.001	0.695	0.762
Median(ms)	-	cores	0.816***	< 0.001	0.792	0.837
Median(ms)	-	completed	0.630***	< 0.001	0.587	0.669
Median(ms)	-	user_cpu (ms)	0.825***	< 0.001	0.802	0.845
Median(ms)	-	mean(ms)	0.987***	< 0.001	0.985	0.989
Median(ms)	-	ui_complete_latency (ms)	0.746***	< 0.001	0.712	0.776
Cores	-	completed	0.675***	< 0.001	0.637	0.711
Cores	-	user_cpu (ms)	0.960***	< 0.001	0.954	0.965
Cores	-	mean(ms)	0.827***	< 0.001	0.804	0.847
Cores	-	ui_complete_latency (ms)	0.890***	< 0.001	0.875	0.904
Completed	-	user_cpu (ms)	0.798***	< 0.001	0.772	0.822
Completed	-	mean(ms)	0.653***	< 0.001	0.612	0.69
Completed	-	ui_complete_latency (ms)	0.730***	< 0.001	0.695	0.762
User_CPU(ms)	-	mean(ms)	0.840***	< 0.001	0.819	0.859
User_CPU(ms)	-	ui_complete_latency (ms)	0.944***	< 0.001	0.935	0.951
mean(ms)	-	ui_complete_latency (ms)	0.768***	< 0.001	0.737	0.796

* p < 0.5, ** p < 0.01, *** p < .001

Figure 10. Correlation Table of partial Metrics based on the Spearmans Correlation Coefficients with %95 confidence intervals.

C. Correlation Analysis

In this section, a statistical techniques called correlation analysis was deployed in order to evaluate the strength of relationship between the various metrics collected from the different components of the topologies and under variable operational environments. High correlation coefficients mean that two or more variables have a strong relationship with each other. Figure 10 shows the correlations based on Pearson's Coefficient with confidence interval of 95.0%. The significant correlations are flagged with (*).



Variable	Input rate (tuples/s)	Median (ms)	Cores	Completed	User_cpu (ms)	Latency (ms)	Mean (ms)
Rate (tuples/s)	1.000	0.629	0.675	1.000	0.798	0.652	0.730
Median(ms)	0.629	1.000	0.816	0.630	0.825	0.987	0.746
Cores	0.675	0.816	1.000	0.675	0.960	0.827	0.890
Completed	1.000	0.630	0.675	1.000	0.798	0.653	0.730
User_cpu (ms)	0.798	0.825	0.960	0.798	1.000	0.840	0.944
Mean (ms)	0.652	0.987	0.827	0.653	0.840	1.000	0.768
Latency (ms)	0.730	0.746	0.890	0.730	0.944	0.768	1.000

Figure 11. Heatmap representation of the correlation between the metrics.

A heat map presentation of the correlation relationships is presented in Figure 11 and highlights the most relevant metrics that will be used in the future experiments.

VII. Conclusions and Future Work

Throughout this paper, a QoS-aware self-adapting resource utilisation framework has been presented with the aim of achieving the following main two goals:

- well-utilisation of system resources (Memory, CPU and network) by continuously predicting resource usage by online machine learning techniques, and dynamically tuning the related parameter configurations of the DSMS,
- reducing tuple response times and maximising system throughput, and satisfying user-specified QoS demand levels of each stream query application.

The rest of the experiments of this research will be carried out using computing instances from Google Cloud Computing Platform. We utilise this platform to simulate the real environment that Apache Storm and other DSMS operates on in order to fully validate the applicability and performance gains of the proposed framework.

References

[1] X. Li, L. Ma, K. Li, K. Wang, and H. A. Wang, "Adaptive Load Management over Real-Time Data Streams," Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), IEEE, vol. 2, August 2007, pp. 719-725.

[2] Y. Wei, V. Prasad, S. H. Son, and J. A. Stankovic, "Prediction-Based QoS Management for Real-Time Data Stream," In Proceedings of 27th IEEE International Real-Time Systems Symposium, IEEE, December 2006, pp. 344-358.

[3] J. P. Rinne, M. Liljeberg, and J. J. Jouppi, "Quality of service definition for data streams," Nokia Oyj, January 2008. U.S. Patent 7,320,029.

[4] M. D. de Assuncao, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," Journal of Network and Computer Applications, vol. 103, pp.1-17, 2018.

[5] L. Xu, B. Peng, and I. Gupta, "Stela: Enabling stream processing systems to scale-in and scale-out on-demand," In 2016 IEEE International Conference on Cloud Engineering (IC2E), IEEE, April 2016, pp. 22-31.

[6] A. Pagliari, F. Huet, and G. Urvoy-Keller, "NAMB: A Quick and Flexible Stream Processing Application Prototype Generator," In The 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, May 2020, hal-02483008.

[7] A. Muhammad, and M. Aleem, "A3-Storm: topology-, traffic-, and resource-aware storm scheduler for heterogeneous clusters," JOURNAL OF SUPERCOMPUTING, vol. 77, pp. 1059-1093, May 2020.

[8] B. Peng, M. Hosseini, Z. Hong, R. Farivar, and R. Campbell, "R-storm: Resource-aware scheduling in storm," In Proceedings of the 16th Annual Middleware Conference, November 2015, pp. 149-161.

[9] N. Chaudhry, K. Shaw, and M. eds. Abdelguerfi, "Stream data management," Springer Science & Business Media, 2006. vol. 30.

[10] W. Wingerath, N. Ritter, and F. Gessert, "Data Stream Management," In Real-Time and Stream Data Management, Springer, Cham, January 2019, pp. 43-55.

[11] M. Garofalakis, J. Gehrke, and R. Rastogi, "Data stream management: A brave new world," In Data Stream Management, Springer, Berlin, Heidelberg, 2016, pp. 1-9.

[12] H. Isah, T. Abughofa, S. Mahfuz, D. Ajerla, F. Zulkernine, and S. Khan, "A survey of distributed data stream processing frameworks," IEEE Access, vol. 7, 2019, pp.154300-154316.

[13] "Apache Storm," Apache Software Foundation, 2014. [Online]. Available: https://storm.apache.org/about/integrates.html. [Accessed 01 January 2020].

[14] X. Liu, and R. Buyya, "D-storm: Dynamic resource-efficient scheduling of stream processing applications," In 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS) IEEE, December 2017, pp. 485-492.

[15] Y. Wei, V. Prasad, S. H. Son, and J. A. Stankovic, "Prediction-based QoS management for real-time data streams," In 2006 27th IEEE International Real-Time Systems Symposium (RTSS'06), IEEE, December 2006, pp. 344-358.

[16] R. Gopalakrishnan, and G. M. Parulkar, "A framework for QoS guarantees for multimedia applications within an endsystem," In GISI 95, Springer, Berlin, Heidelberg, pp. 43-56, 1995.

[17] M. HoseinyFarahabady, A. Y. Zomaya, and Z. Tari, "QoS-and contention-aware resource provisioning in a stream processing engine," In 2017 IEEE International Conference on Cluster Computing (CLUSTER), September 2017, pp. 137-146.

[18] N. Tantalaki, S. Souravlas, M. Roumeliotis, and S. Katsavounis, "Pipeline-Based linear scheduling of big data streams in the cloud," IEEE Access, vol. 8, pp.117182-117202, June 2020.

[19] T. Buddhika, R. Stern, K. Lindburg, K. Ericson, and S. Pallickara, "Online scheduling and interference alleviation for low-latency, high-throughput processing of data streams," IEEE Transactions on Parallel and Distributed Systems, vol. 28(12), pp.3553-3569, 2017.

[20] D. J. Abadi et al., "Aurora: a new model and architecture for data stream management," the VLDB Journal, vol. 12(2), pp.120-139, 2003.

[21] A. Arasu et al., "Stream: The stanford data stream management system," In Data Stream Management, Springer, Berlin, Heidelberg, 2016, pp. 317-336.

[22] S. Schmidt, H. Berthold, and W. Lehner, "Qstream: Deterministic querying of data streams," In Proceedings of the Thirtieth international conference on Very large data bases, vol. 30, August 2004, pp. 1365-13.

[23] U. Cetintemel et al., “The aurora and borealis stream processing engines,” In Data Stream Management, Springer, Berlin, Heidelberg, pp. 337-359, 2016.
[24] A. Klein, and W. Lehner, “Representing data quality in sensor data streaming environments,” Journal of Data and Information Quality (JDIQ), vol. 1(2), pp.1-28, Sep. 2009.
[25] H. Wang, Y. Ma, X. Zheng, X. Chen, and L. Guo, “Self-adaptive resource management framework for software services in cloud,” In 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), IEEE, 2019, pp. 1528-1529.
[26] M. A. Serhani, H. T. El-Kassabi, K. Shuaib, A. N. Navaz, B. Benatallah, and A. Beheshti, “Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven IoT workflows,” Future Generation Computer Systems, vol. 108, pp. 583-597, 2020.
[27] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli, “Distributed QoS-aware scheduling in Storm,” In Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, Jun. 2015, pp. 344-347.
[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: an update,” ACM SIGKDD explorations newsletter, vol. 11(1), pp.10-18, 2009.
[29] A. Bifet et al., “MOA: a real-time analytics open source framework,” In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Berlin, Heidelberg, Sep. 2011, pp. 617-620.

From Open Data to Linked Open Data

The GIOCOnDa LOD platform

Lorenzo Sommaruga, Nadia Catenazzi, Davide Bertacco, Riccardo Mazza
Department of Innovative Technologies
University of Applied Sciences and Arts of Southern Switzerland (SUPSI)
CH-6962 Lugano, Switzerland
e-mail: lorenzo.sommaruga@supsi.ch
nadia.catenazzi@supsi.ch
davide.bertacco@supsi.ch
riccardo.mazza@supsi.ch

Abstract—**In the context of the European project "GIOCOnDa", this paper describes the conversion process from Open Data to Linked Open Data (LOD) and its implementation in the GIOCOnDa LOD platform. The platform contains a number of conversion configurations that allow different data sources from a variety of Open Data domains to be converted into LOD, without the need of software programming. In addition, the platform is configurable and extensible, as it enables to define mapping configurations for new datasets.**

Keywords-Linked Open Data (LOD); GIOCOnDa; OntoPia.

I. INTRODUCTION

This paper describes the methodology and the conversion process from Open Data to Linked Open Data (LOD) implemented in the GIOCOnDa LOD platform in the context of the EU Interreg GIOCOnDa project ("Integrated and holistic management of the open data life cycle" [1]).

This project, funded by the Interreg V-A Italy-Switzerland Programme, aims to create value by developing information products based on the re-use of public Open Data. The project involves the creation of a federated platform for the publication of Linked Open Data by public administrations. In the GIOCOnDa LOD platform, open data, coming from various sources and in different formats, are converted into homogeneous Linked Open Data according to standard ontologies and published together with their metadata. The platform allows conversion of existing 3* Open Data to 5* Open Data, according to the well-known 5-star deployment scheme [2]: data are formalized in RDF, identified by URI and linked to other datasets.

The project focuses on data from the Insubric area, a cross-border territory and community across Italy and Switzerland. According to the project specifications, data useful in the touristic sector are considered. These include data about museums, accommodation facilities and environmental data. The main data sources currently used include: Regione Lombardia open data portal [3] and ARPA (Regional Agency for the Protection of the Environment) [4] for Italian data; Wikidata, Ticino Turismo [5] and OASI [6] for Swiss data.

The GIOCOnDa LOD platform is mainly oriented to domain and ontology experts, who need to authenticate to operate in the platform to create and modify datasets. A public portal is also available, where the datasets produced in the LOD platform are made accessible. Public administrations can submit new datasets for conversion into LOD.

This paper is structured as follows: in Section 2 the methodology adopted to publish LOD data is described; Section 3 is focused on the process of conversion of Open Data to LOD, one of the main steps of this methodology; finally, Section 4 clarifies how this process is implemented in the GIOCOnDa LOD platform.

II. METHODOLOGY TO PUBLISH LINKED OPEN DATA

The subject of Linked Open Data publishing has been widely discussed in literature (e.g., [7]) and different projects and platforms have been developed to support this process. One of the first significant projects is Lucero and the resulting Tabloid toolkit, which aims to help institutions and developers to publish and consume linked data [8]. Another interesting work, supporting US open government data production and consumption, is the TWC LOGD portal [9]. A workflow for linked open data deployment is defined, consisting of different stages, where the conversion process is automated by using the csv2rdf4lod tool. A more recent initiative is represented by the Italian cultural heritage platform "dati.beniculturali.it", promoted by the Italian Ministry of Culture, which collects and publish standardized and interoperable LOD heterogenous datasets [10].

From a methodological point of view, a number of best practices, recommendations and guidelines have been produced. For example, Bauer and Kaltenböck [11] provide a step-by-step model, highlighting the most important issues that need to be taken into account in LOD publishing; W3C [12] presents best practices designed to facilitate LOD development and delivery; the "Agenzia per l'Italia Digitale" [13] proposes a general methodological approach for the interoperable opening of public data through the LODs. This methodology basically consists of the following steps: selection of dataset, data cleaning, analysis and RDF modelling, enrichment, interlinking, validation and publication.

The approach adopted in GIOCOnDa is in line with the above recommendations and in particular with the AGID guidelines. The selection of datasets was made on the basis of the results of a previous need analysis phase carried out in the project; as a starting point, data about museums, accommodation facilities and environment of the Insubric region are selected.

Concerning data cleaning, it is assumed that the selected datasets are already published as "clean" open data, where a quality check is already accomplished.

Once selected, datasets are deeply analyzed to understand their structure and appropriate ontologies and vocabularies are identified to model them.

In particular, the adopted ontologies are taken from the OntoPia network [14], also presented in [15]. They include for instance the Cultural-ON ontology for museums and the ACCO ontology for accommodations. In the GIOCOnDa LOD platform, data are imported from different sources and converted into the RDF format, according to these standard ontologies. The conversion process is detailed in the next section.

As additional steps, datasets are enriched with metadata and interlinked to other datasets. Metadata are added to the single datasets following the DCAT-AP standard. Interlinks are created to other datasets by identifying alignments and similarities between different datasets. For instance, a museum of the "Regione Lombardia" dataset can be declared "the same as" a museum described in Wikidata. The identification of interlinks is mainly carried out using the Silk software libraries [16].

Finally, datasets are published using Openlink Virtuoso Universal Server [17]; they can be queried through a SPARQL endpoint.

III. THE CONVERSION PROCESS FROM OPEN DATA TO LOD IN THE GIOCONDA PLATFORM

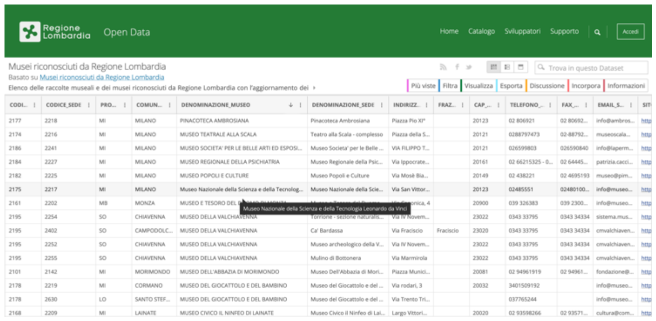
The core of the system lies in the mapping functionality of heterogenous data into linked open data, according to standard ontologies.

This conversion is a complex process that depends on the initial format and on the final standard RDF format. From a literature study it emerges that the most frequently adopted approach is the implementation of ad-hoc middleware. For example, to convert a relational database to LOD, a typical solution is to use declarative languages, such as D2R [18] or R2RML [19] that require ontological and programming skills.

In the GIOCOnDa LOD platform, the complexity of the conversion process is simplified by defining a converter, facilitated by a graphical user interface that an expert can use to configure the conversion. This process can be explained through a simple example: we would like to convert two different datasets about museums into a common interoperable format. The first dataset concerns *Lombard museums* retrieved from the Regione Lombardia portal in JSON format by means of REST APIs [20]. The second is represented by *Tessin Canton museums* retrieved from Wikidata through SPARQL queries.

Figure 1 shows an excerpt of the Lombard museums visualized on the Regione Lombardia portal, while Figure 2 shows an example of a Swiss museum in Wikidata [21].

To be able to configure the mapping from the original to LOD format, the structure of the two museum data sources has to be analyzed and an appropriate ontology selected. In this phase it is important to find the most appropriate ontology to model the domain. The Cultural-ON ontology [22] and its connected ontologies have been chosen because they are representative of the museum domain and can be exploited to support transnational interoperability.

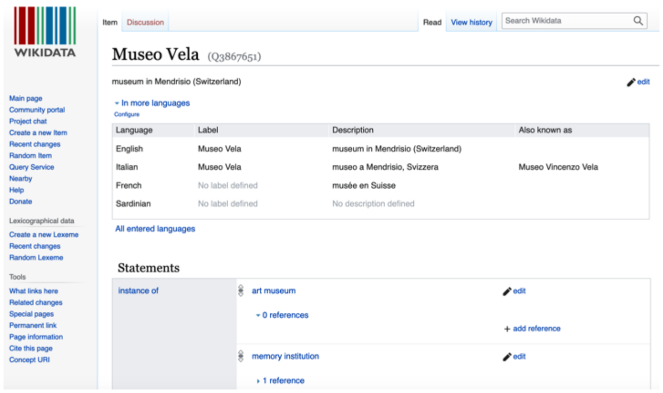


Musei riconosciuti da Regione Lombardia
Basato su Musei riconosciuti da Regione Lombardia
Elenco delle raccolte museali e dei musei riconosciuti da Regione Lombardia con l'aggiornamento dei

CODI...	CODICE_SEDE	PRO...	COMUN...	DENOMINAZIONE_MUSEO	DENOMINAZIONE_SEDE	INDIRIZZ...	FRAZ...	CAP_...	TELEFONO_...	FAX_...	EMAIL_S...	SIT
2177	2218	MI	MILANO	PINACOTECA AMBROSIANA	Pinacoteca Ambrosiana	Piazza Pio XI°		20123	02 806921	02 80692...	info@ambros...	http
2174	2216	MI	MILANO	MUSEO TEATRALE ALLA SCALA	Teatro alla Scala - complesso	Piazza della S...		20121	0288797473	02-88792...	museoscala...	http
2186	2241	MI	MILANO	MUSEO SOCIETA' PER LE BELLE ARTI ED ESPOSI...	Museo Societa' per le Belle ...	VIA FILIPPO T...		20121	026599803	026590840	info@laperm...	http
2184	2227	MI	MILANO	MUSEO REGIONALE DELLA PSICHIATRIA	Museo Regionale della Psic...	Via Ippocrate...		20161	02 66215325 - 0...	02 64445...	patrizia.cacci...	http
2182	2225	MI	MILANO	MUSEO POPOLI E CULTURE	Museo Popoli e Culture	Via Mosè Bia...		20149	02 438221	02 4695193	museo@pim...	http
2175	2217	MI	MILANO	Museo Nazionale della Scienza e della Tecnolog...	Museo Nazionale della Scie...	Via San Vittor...		20123	02485551	02480100...	info@museo...	http
2161	2202	MB	MONZA	MUSEO E TESORO DEL [illegible]	[illegible]	[illegible]onica, 4		20900	039 326383	039 2300...	info@museo...	http
2195	2254	SO	CHIAVENNA	MUSEO DELLA VALCHIAVENNA	Torrione - sezione naturalis...	Via IV Novem...		23022	0343 33795	0343 34334	sistema.mus...	http
2195	2402	SO	CAMPODOLC...	MUSEO DELLA VALCHIAVENNA	Ca' Bardassa	Via Fraciscio	Fraciscio	23020	0343 33795	0343 34334	cmvalchiaven...	http
2195	2252	SO	CHIAVENNA	MUSEO DELLA VALCHIAVENNA	Museo archeologico della V...	Via IV Novem...		23022	0343 33795	0343 34334	cmvalchiaven...	http
2195	2255	SO	CHIAVENNA	MUSEO DELLA VALCHIAVENNA	Mulino di Bottonera	Via Marmirola		23022	0343 33795	0343 34334	cmvalchiaven...	http
2101	2142	MI	MORIMONDO	MUSEO DELL'ABBAZIA DI MORIMONDO	Museo Dell'Abbazia di Mori...	Piazza Munici...		20081	02 94961919	02 94961...	fondazione@...	http
2178	2219	MI	CORMANO	MUSEO DEL GIOCATTOLO E DEL BAMBINO	Museo del Giocattolo e del ...	Via rodari, 3		20032	3401509192		info@museo...	http
2178	2630	LO	SANTO STEF...	MUSEO DEL GIOCATTOLO E DEL BAMBINO	Museo del Giocattolo e del ...	Via Trento Tri...			037765244		info@museo...	http
2168	2209	MI	LAINATE	MUSEO CIVICO IL NINFEO DI LAINATE	Museo Civico il Ninfeo di Lai...	Largo Vittori...		20020	02 93598266	02 93571...	cultura@com...	http

Museo Nazionale della Scienza e della Tecnologia Leonardo da Vinci

Figure 1. Lombard museums from the Regione Lombardia Open Data portal



Museo Vela (Q3867651)

museum in Mendrisio (Switzerland)

Language	Label	Description	Also known as
English	Museo Vela	museum in Mendrisio (Switzerland)	
Italian	Museo Vela	museo a Mendrisio, Svizzera	Museo Vincenzo Vela
French	No label defined	musée en Suisse	
Sardinian	No label defined	No description defined	

Statements

instance of	art museum	0 references
	memory institution	1 reference

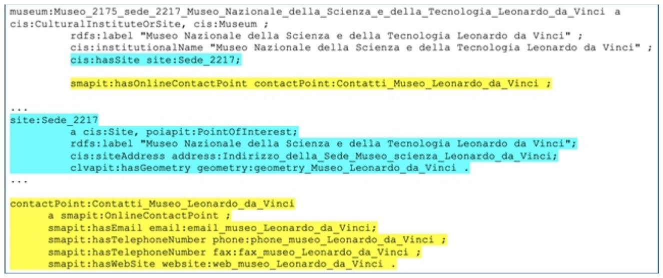
Figure 2. The Swiss *Vela* museum in Wikidata

The next step consists of going through the different descriptive fields of the museum datasets: for instance, each Lombard museum is described in terms of 79 fields, such as *denominazione museo* (name), *telefono* (telephone), *codice sede* (site code) as shown in Figure 1.

For each field, the objective is to find a match with the ontology classes and properties. For example, each museum could be represented as an instance of the c*is:Museum* class of the Cultural-ON ontology, where *cis* is the prefix of the ontology namespace; the *telephone* field can be mapped into a property of a *smapit:OnlineContactPoint* instance of the Social Media / Contact and Internet ontology [23].

Figures 3 and 4 represent some result details of the conversion of a Lombard and a Tessin canton museum, respectively, in RDF Turtle, according to the Cultural-ON and the connected ontologies. In particular, in the excerpts, light blue highlights the *hasSite* relation to the *Site* instance, and yellow highlights the *hasOnlineContactPoint* relation to the *ContactPoint* instance with their respective properties.

It is important to note that two museums, initially available in different formats, are finally described in a common interoperable RDF format. This translation process leads, in this case, to information loss because there is not a full correspondence between the initial format and the ontological one. The ontology contains more classes and properties than the original file format; on the other hand, it is not enough expressive to represent all fields of the original data sources. For instance, the *number of visitors* is not included in the Cultural-ON ontology.



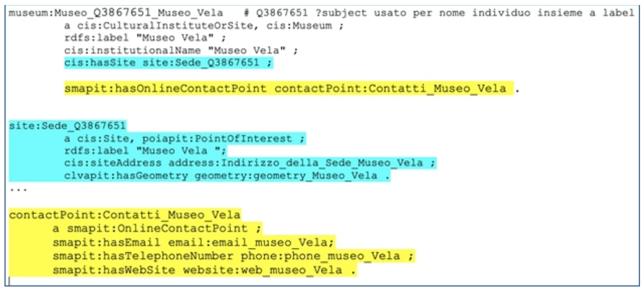
```
museum:Museo_2175_sede_2217_Museo_Nazionale_della_Scienza_e_della_Tecnologia_Leonardo_da_Vinci  a
cis:CulturalInstituteOrSite, cis:Museum ;
         rdfs:label "Museo Nazionale della Scienza e della Tecnologia Leonardo da Vinci" ;
         cis:institutionalName "Museo Nazionale della Scienza e della Tecnologia Leonardo da Vinci" ;
         cis:hasSite site:Sede_2217;

         smapit:hasOnlineContactPoint contactPoint:Contatti_Museo_Leonardo_da_Vinci ;

...
site:Sede_2217
         a cis:Site, poiapit:PointOfInterest;
         rdfs:label "Museo Nazionale della Scienza e della Tecnologia Leonardo da Vinci";
         cis:siteAddress address:Indirizzo_della_Sede_Museo_scienza_Leonardo_da_Vinci;
         clvapit:hasGeometry geometry:geometry_Museo_Leonardo_da_Vinci .
...

contactPoint:Contatti_Museo_Leonardo_da_Vinci
      a smapit:OnlineContactPoint ;
      smapit:hasEmail email:email_museo_Leonardo_da_Vinci;
      smapit:hasTelephoneNumber phone:phone_museo_Leonardo_da_Vinci ;
      smapit:hasTelephoneNumber fax:fax_museo_Leonardo_da_Vinci ;
      smapit:hasWebSite website:web_museo_Leonardo_da_Vinci .
```

Figure 3. Excerpt of the Lombard Museum of Science and Technology in RDF Turtle



```
museum:Museo_Q3867651_Museo_Vela   # Q3867651 ?subject usato per nome individuo insieme a label
        a cis:CulturalInstituteOrSite, cis:Museum ;
        rdfs:label "Museo Vela" ;
        cis:institutionalName "Museo Vela" ;
        cis:hasSite site:Sede_Q3867651 ;

        smapit:hasOnlineContactPoint contactPoint:Contatti_Museo_Vela .


site:Sede_Q3867651
        a cis:Site, poiapit:PointOfInterest ;
        rdfs:label "Museo Vela ";
        cis:siteAddress address:Indirizzo_della_Sede_Museo_Vela ;
        clvapit:hasGeometry geometry:geometry_Museo_Vela .
...

contactPoint:Contatti_Museo_Vela
      a smapit:OnlineContactPoint ;
      smapit:hasEmail email:email_museo_Vela;
      smapit:hasTelephoneNumber phone:phone_museo_Vela ;
      smapit:hasWebSite website:web_museo_Vela .
```

Figure 4. Excerpt of The Swiss Vela museum in RFD Turtle

In the conversion process, the mapping from the initial input data format to the final LOD format would need to be configured for each data source. This requires a deep knowledge of the OWL syntax, and understanding of the classes and datatype/object properties of the selected ontologies.

To simplify the conversion process, an internal vocabulary was created, with the objective to describe in a homogeneous and simple way data coming from different sources, without knowing the details of the ontology and further separate the input from the output. The main advantage of having this vocabulary is to hide the complexity of the ontology in the mapping management. The internal vocabulary is organized in categories, that represent contexts or ontologies; each category contains classes; each class has a number of fields. For instance, to describe museums we have defined the *museum Cultural-ON* category; this category contains classes, such as *museum* and *discipline*, and fields, such as *geographical coordinates*.

Thanks to the internal vocabulary, the conversion process is divided in two steps:

- the conversion from the input data format to the internal vocabulary (*input mapping*)
- the conversion from the internal vocabulary to the ontological LOD format (*output mapping*).

Going back to the museum example, the two datasets, originally described in different formats and with different descriptive fields, are translated by means of the input mapping specifications into a common format, which is described by the internal vocabulary. The resulting datasets are then converted to the LOD format, according to a standard ontology, by means of the output mapping specification. This guarantees standardization and semantic interoperability. Figure 5 illustrates the process.

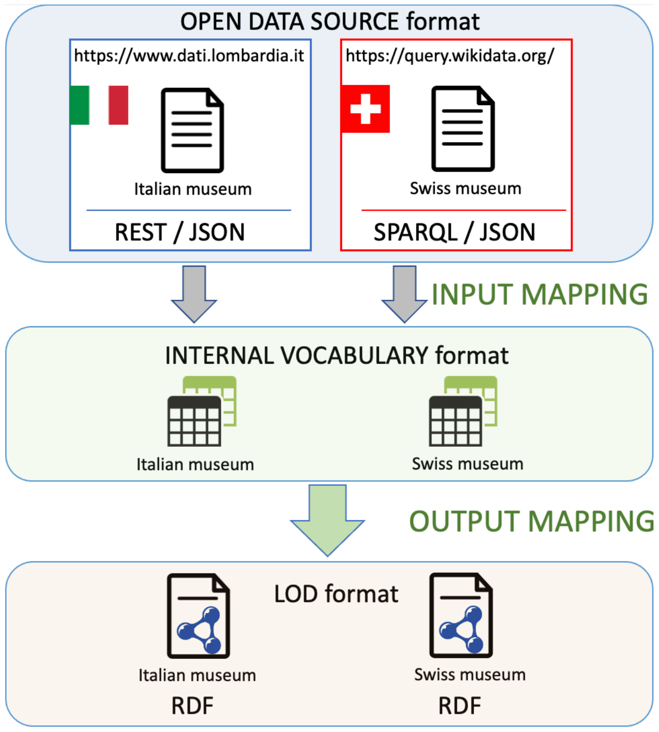


Figure 5. Two step conversion process: museum data example

While it is necessary to configure the input mapping of each imported dataset towards the internal vocabulary, the output mapping of a specific category (e.g., museum) to the corresponding LOD format has to be configured only once. The first step can be accomplished by a user who knows the input format, the domain and the internal vocabulary, the second step requires a deep knowledge of the ontologies and of the OWL language.

This mechanism, to convert Open Data to Linked Open Data based on two independent and configurable steps, is the peculiar feature of GIOCOnDa LOD platform compared to other frameworks, which do not provide the flexibility and dynamic configurability required by the GIOCOnDa project.

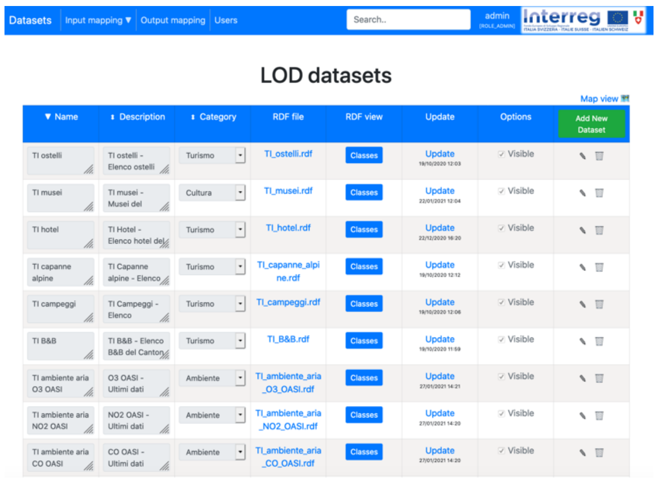
IV. GIOCONDA LOD PLATFORM

The GIOCOnDa LOD platform [24], implemented as a Java based web application, provides different functionalities that enable the publication of LOD datasets starting from open datasets, and their visualization in a catalogue or in a map.

The web app presents a menu consisting of different items: dataset catalogue, input mapping and output mapping.

A. *LOD Datasets*

The catalogue shows the list of the existing datasets, as shown in Figure 6, and enables the creation of a new LOD dataset by converting an existing open dataset on the basis of the input and output mapping configuration. The system supports dataset updates at regular intervals (e.g., for air quality measurements) and propagates the changes to the RDF representation.



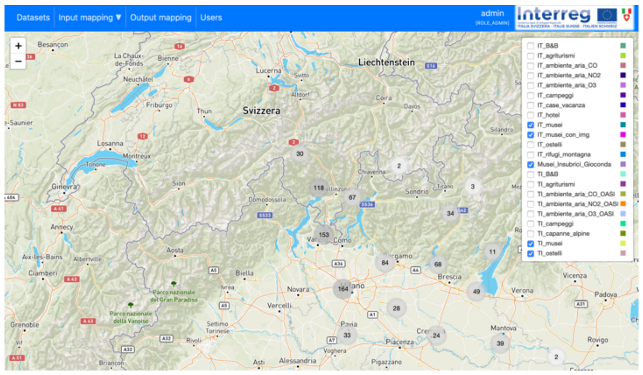
LOD datasets

Map view

▼ Name	⬍ Description	⬍ Category	RDF file	RDF view	Update	Options	Add New Dataset
TI ostelli	TI ostelli - Elenco ostelli	Turismo	TI_ostelli.rdf	Classes	Update 19/10/2020 12:03	✓ Visible	
TI musei	TI musei - Musei del	Cultura	TI_musei.rdf	Classes	Update 22/01/2021 12:04	✓ Visible	
TI hotel	TI Hotel - Elenco hotel del	Turismo	TI_hotel.rdf	Classes	Update 22/12/2020 16:20	✓ Visible	
TI capanne alpine	TI Capanne alpine - Elenco	Turismo	TI_capanne_alpine.rdf	Classes	Update 19/10/2020 12:12	✓ Visible	
TI campeggi	TI Campeggi - Elenco	Turismo	TI_campeggi.rdf	Classes	Update 19/10/2020 12:06	✓ Visible	
TI B&B	TI B&B - Elenco B&B del Canton	Turismo	TI_B&B.rdf	Classes	Update 19/10/2020 11:59	✓ Visible	
TI ambiente aria O3 OASI	O3 OASI - Ultimi dati	Ambiente	TI_ambiente_aria_O3_OASI.rdf	Classes	Update 27/01/2021 14:21	✓ Visible	
TI ambiente aria NO2 OASI	NO2 OASI - Ultimi dati	Ambiente	TI_ambiente_aria_NO2_OASI.rdf	Classes	Update 27/01/2021 14:20	✓ Visible	
TI ambiente aria CO OASI	CO OASI - Ultimi dati	Ambiente	TI_ambiente_aria_CO_OASI.rdf	Classes	Update 27/01/2021 14:20	✓ Visible	

Figure 6. LOD datasets catalogue

By clicking on the "map view" button it is possible to visualize geo-locable data on the map as shown in Figure 7.



☐	IT_B&B	🟩
☐	IT_agriturismi	🟩
☐	IT_ambiente_aria_CO	🟥
☐	IT_ambiente_aria_NO2	🟦
☐	IT_ambiente_aria_O3	🟪
☐	IT_campeggi	🟪
☐	IT_case_vacanza	🟦
☐	IT_hotel	🟪
✓	IT_musei	🟦
✓	IT_musei_con_img	🟪
☐	IT_ostelli	🟫
☐	IT_rifugi_montagna	🟦
✓	Musei_Insubrici_Gioconda	🟪
☐	TI_B&B	🟩
☐	TI_agriturismi	🟪
☐	TI_ambiente_aria_CO_OASI	🟩
☐	TI_ambiente_aria_NO2_OASI	🟧
☐	TI_ambiente_aria_O3_OASI	🟦
☐	TI_campeggi	🟩
☐	TI_capanne_alpine	🟩
✓	TI_musei	🟨
✓	TI_ostelli	🟥

Figure 7. LOD Datasets visualized in the map

About 25 datasets about accommodation, museums and air quality have been boosted to LOD datasets through the GIOCOnDa platform, resulting as published LOD resources complaint to the selected ontologies and vocabularies. Concerning validation, the output mapping process guarantees by design and implementation that the produced datasets are serialized in RDF format conforming to each ontology. A further manual checking has been accomplished on some resources of each typology.

B. *Input Mapping*

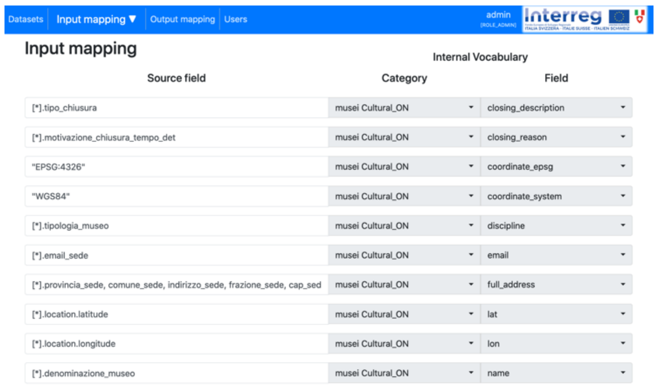
The Input mapping concerns the configuration of the conversion from the input format to the internal vocabulary. Together with the output mapping it enables to configure the conversion from different input data sources to the final LOD format.

In general, the system accepts input data retrieved from a number of sources in different formats, such as JSON, CSV and XML, and using different services, such as Rest APIs, SOAP APIs and SPARQL Queries. For each input format and service the conversion towards the internal vocabulary is configured through the input mapping.

Figure 8 clarifies how the mapping mechanism works: the first column shows the fields of the original data source, the second and third columns concern the internal vocabulary, in particular the second identifies the category and the third the fields.

Categories and fields of the internal vocabulary are predetermined and selected from a drop-down menu, while the input fields must be entered by hand, according to the data structure in use.

In the example shown in Figure 8 we work with fields of the "Cultural-ON museums" category; the "email_sede" field of the initial source, representing the email of the museum site, for example, is mapped into the "email" field of the "museum Cultural-ON" category of the internal vocabulary. In some cases, it is possible to group multiple fields of the data source into a single field of the intermediate vocabulary; for example, to compose the field "full_address", more fields of the input source are used.



Input mapping

	Internal Vocabulary	
Source field	Category	Field
[*].tipo_chiusura	musei Cultural_ON	closing_description
[*].motivazione_chiusura_tempo_det	musei Cultural_ON	closing_reason
"EPSG:4326"	musei Cultural_ON	coordinate_epsg
"WGS84"	musei Cultural_ON	coordinate_system
[*].tipologia_museo	musei Cultural_ON	discipline
[*].email_sede	musei Cultural_ON	email
[*].provincia_sede, comune_sede, indirizzo_sede, frazione_sede, cap_sed	musei Cultural_ON	full_address
[*].location.latitude	musei Cultural_ON	lat
[*].location.longitude	musei Cultural_ON	lon
[*].denominazione_museo	musei Cultural_ON	name

Figure 8. Input mapping

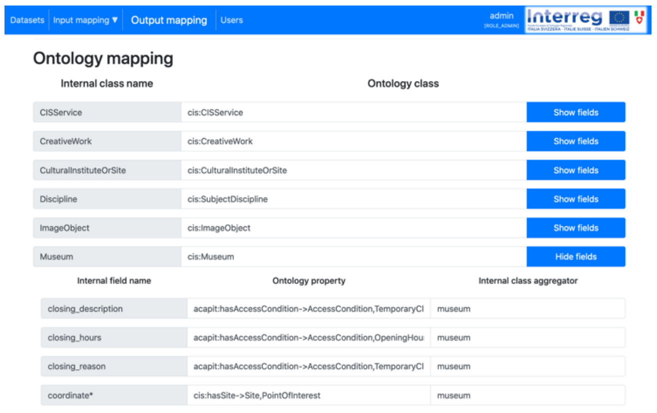
In this mapping, particular attention is dedicated to how the geospatial data are represented. This is essential to guarantee interoperability and efficient sharing of information across different regions and national standards. The Cultural-ON ontology assumes, by default, that spatial data are represented in the geocentric Datum WGS84 and that the coordinates are expressed in terms of latitude and longitude. Therefore, data are transformed in this system when they are imported in the GIOCOnDA platform and appropriate metadata are added to make the Coordinate Reference System (CRS) explicit.

C. *Ouput Mapping*

From the output mapping page, it is possible to create, modify and extend the internal vocabulary and define its mapping to the ontology. This process requires a deep knowledge of ontological concepts and existing reference ontologies. Nevertheless, this mapping has to be done only once for each category by an expert.

As already said, the internal vocabulary consists of several categories, similar to contexts or ontologies; each category contains classes, with associated a number of fields. Examples of categories include museums, addresses, accommodations, etc.

As shown in Figure 9, the output mapping defines the match between internal vocabulary classes and ontology classes, and between fields of the internal vocabulary and object and datatype properties of the ontology; this is visible by activating the “Show fields” button.



Ontology mapping

Internal class name	Ontology class	
CISService	cis:CISService	Show fields
CreativeWork	cis:CreativeWork	Show fields
CulturalInstituteOrSite	cis:CulturalInstituteOrSite	Show fields
Discipline	cis:SubjectDiscipline	Show fields
ImageObject	cis:ImageObject	Show fields
Museum	cis:Museum	Hide fields

Internal field name	Ontology property	Internal class aggregator
closing_description	acapit:hasAccessCondition->AccessCondition,TemporaryCl	museum
closing_hours	acapit:hasAccessCondition->AccessCondition,OpeningHou	museum
closing_reason	acapit:hasAccessCondition->AccessCondition,TemporaryCl	museum
coordinate*	cis:hasSite->Site,PointOfInterest	museum

Figure 9. Output mapping: class/field match

It is worth noting that only categories and fields of the intermediate vocabulary defined in the output mapping can be used in the input mapping (but not classes), providing in this way a simplified version of the data structure for the non-expert user.

A specific interlinking software has also been developed and integrated in the GIOCOnDa LOD platform. In order to boost a dataset to 5* level it is necessary to configure the interlinking and then activate it. The configuration is accomplished in the output mapping page where one or more interlinking files can be associated to each category. This file is generated using the SILK Link Specification Language and contains rules to create cross-reference links towards external datasets, such as Wikidata. The activation takes place in the datasets page, where it is possible to enable or disable interlinking on a specific dataset.

V. Conclusions and Future Works

This paper has presented a platform that facilitates the process of conversion of open data to linked open data, by means of a visual interface, without the need of a specific software programming. Indeed, one of its main advantages is the reduction of complexity of a process that requires deep knowledge of ontologies and programming skills. The complexity of mapping existing data to standard ontologies is one of the major issues preventing a larger diffusion of LOD.

The GIOCOnDa LOD platform contains a number of conversion configurations that allow different data sources to be converted to LOD in different domains. For instance, if a dataset has the same structure of an existing one (for example, a new dataset structured as the Lombard museums), the conversion is very simple, since the input mapping is very similar to an existing one and the output mapping is already defined.

However, the platform is also flexible and extensible, and enables to import and convert other datasets: for example, the conversion to LOD of a new dataset about museums, with a structure that can be mappable to the existing internal vocabulary, only requires the configuration of the input mapping from the initial format to the internal vocabulary, because the output mapping is already configured. More labour-intensive but still possible is to convert a dataset with a new structure, not mappable on the existing internal vocabulary; for example, a new dataset about bike sharing. In this case, it is necessary to find an ontology that models the domain; to extend the internal vocabulary with a new category and define the mapping towards the ontology - this is configured in the output mapping; to define the mapping from the original data format to the internal vocabulary - this is configured in the input mapping.

In addition to the conversion to LOD, another important step of the adopted methodology is the identification and creation of interlinks between datasets. A specific interlinking software has been developed to configure and activate the process of identification of cross-reference links towards external datasets. The integration of the interlinking module in the GIOCOnDa LOD platform enables lifting datasets to 5* level, creating added value through an Extract-Transform-Load (ETL) pipeline. This has been demonstrated for instance in a showcase that presents data about museums of the Insubric region taken both from the GIOCOnDa LOD datasets and from Wikidata.

In spite of the benefits offered by platform to publish LOD datasets (visual interface, configurability, extensibility), it also presents some limitations: the main one is the possibility of information loss during the conversion to LOD if there are fields not represented in the selected ontology. A possible solution is the extension of the selected ontologies with additional fields and the publication of the new version with appropriate documentation.

Finally, the platform development is still in progress and some details need to be fully implemented or considered for future implementation, such as the validation process, inference and interlinking.

ACKNOWLEDGMENT

We acknowledge the European Regional Development Fund, the Tessin Canton and the Swiss Confederation for the financial support provided to the project. We also acknowledge all the partners involved in the project for their contribution to the different project activities.

REFERENCES

[1] GIOCOnDa, Integrated and holistic management of the Open Data life cycle (Gestione integrata e olistica del ciclo di vita degli Open Data) https://interreg-italiasvizzera.eu/progetti/gioconda/ [retrieved: March, 2021].

[2] Berners-Lee, Linked Data, http://www.w3.org/DesignIssues/LinkedData.html, 2009 [retrieved: March, 2021].

[3] Open Data, Regione Lombardia, https://dati.lombardia.it [retrieved: March, 2021].

[4] Arpa Lombardia, https://www.arpalombardia.it/ [retrieved: March, 2021].

[5] Ticino Turismo, https://www.ticino.ch [retrieved: March, 2021].

[6] OASI, Osservatorio Ambientale della Svizzera italiana, https://www.oasi.ti.ch [retrieved: March, 2021].

[7] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool Publishers, 2011.

[8] The Lucero Project, https://lucero-project.info/ [retrieved: March, 2021].

[9] L. Ding, T. Lebo, J. S. Erickson, D. DiFranzo, G. T. Williams, X. Li, J. Michaelis, A. Graves, J. G. Zheng, Z. Shangguan, J. Flores, D. L. McGuinness, J. A. Hendler, "TWC LOGD: A portal for linked open government data ecosystems", Journal of Web Semantics, vol. 9(3), pp. 325-333, 2011, doi: 10.1016/j.websem.2011.06.002.

[10] Open Data e Linked Data, https://www.beniculturali.it/open-data-e-linked-data [retrieved: March, 2021].

[11] B. Florian and K. Martin, Linked Open Data: The Essentials - A Quick Start Guide for Decision Makers. Edition mono/monochrom, Vienna, Austria, 2012, ISBN: 978-3-902796-05-9,

http://www.semantic-web.at/LOD-TheEssentials.pdf [retrieved: March, 2021].

[12] W3C, Best Practices for Publishing Linked Data, W3C Working Group Note 09 January 2014, https://www.w3.org/TR/ld-bp/ [retrieved: March, 2021].

[13] Agenzia per Italia Digitale, Guidelines for semantic interoperability through Linked Open Data (Linee Guida per Interoperabilità semantica attraverso i Linked Open Data), http://www.agid.gov.it/sites/default/files/documentazione_trasparenza/cdc-spc-gdl6-interoperabilitasemopendata_v2.0_0.pdf 2012 [retrieved: March, 2021].

[14] Ontologie e Vocabolari Controllati, https://github.com/italia/daf-ontologie-vocabolari-controllati [retrieved: March, 2021].

[15] G. Lodi, "OntoPiA –The network of ontologies and controlled vocabularies for public admistration (OntoPIA - La rete di ontologie e vocabolari controllati per la pubblica amministrazione)", Open Data Sicilia – raduno annuale, 9/10 novembre 2018, http://ods2018.opendatasicilia.it/presentazioni/Lodi-OntoPiA.pdf, 2018 [retrieved: March, 2021].

[16] Silk, The Linked Data Integration Framework. http://silkframework.org/ [retrieved: March, 2021].

[17] Open Link Virtuoso, https://virtuoso.openlinksw.com [retrieved: March, 2021].

[18] C. Bizer and R. Cyganiak, "D2R Server - Publishing Relational Databases on the Semantic Web", 5th International Semantic Web Conference (ISWC 2006), Athens, USA, November 2006, http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/Bizer-Cyganiak-D2R-Server-ISWC2006.pdf, 2006 [retrieved: March, 2021].

[19] R2ML: RDB to RDF Mapping Language https://www.w3.org/TR/r2rml [retrieved: March, 2021].

[20] Musei riconosciuti da Regione Lombardia https://www.dati.lombardia.it/Cultura/Musei-riconosciuti-da-Regione-Lombardia/3syc-54zf [retrieved: March, 2021].

[21] Museo Vela https://www.wikidata.org/wiki/Q3867651 [retrieved: March, 2021].

[22] Cultural-ON (Cultural ONtology): Cultural Institute/Site and Cultural Event Ontology, https://w3id.org/italia/onto/Cultural-ON [retrieved: March, 2021].

[23] Social Media / Contact and Internet ontology - Italian Application Profile https://w3id.org/italia/onto/SM [retrieved: March, 2021].

[24] GIOCOnDa LOD platform https://gioconda.supsi.ch/ [retrieved: March, 2021].

Drifting and Popularity: A Study of Time Series Analysis of Topics

Muhammad Haseeb UR Rehman Khan
University of Tsukuba
Tsukuba, Japan
s2036048@s.tsukuba.ac.jp

Kei Wakabayashi
University of Tsukuba
Tsukuba, Japan
kwakaba@slis.tsukuba.ac.jp

***Abstract*—Topic modeling is extensively used for the Natural Language Processing (NLP) problems of summarizing, organizing, and understanding large document datasets. Latent Dirichlet Allocation (LDA) is widely used for the collection of topics, whereas Dynamic Topic Model (DTM) is famous for the time-series topic analysis. However, by estimating the number of occurrences of topics in each time slice, we can obtain time-series topic popularity using standard LDA. Therefore, if this can be extracted with LDA, then why do we need DTM which has a very high computation cost? The purpose of this research is to determine, either time-series topic information can be extracted from LDA or we need DTM. Topic drifting and popularity are two fundamental aspects of time-series topic analysis. We conducted experiments with multiple datasets to check the reliability of the information extracted from both models. We used Jensen-Shannon (JS) similarity-based analysis to check for information overlap. We constructed time-series topic popularity graphs for both models from the document-topic distributions and compared the results. Our results show that there is notable DTM topic drifting information in some cases and sometimes no or vague topic drifting. Topic drifting embedded in DTM topics makes this model less favorable for topic popularity analysis. On the other hand, LDA topics with no time transition information provided concrete results of topic popularity.**

Keywords*—*DTM; LDA; Topic Modeling; Time Series Analysis.

I. Introduction

Latent Dirichlet Allocation (LDA) [1] and Dynamic Topic Model (DTM) [2] are widely used topic models that revolutionized the solving of topic modeling-based NLP problems. Situations that need the assistance of topic models often involve time-series document collections, including Twitter posts, news articles, and academic paper archives. By focusing on the nature of time-series, many useful applications can be developed, such as bursty topic detection [3], trend analysis [4][5], topic evolution analysis [6][7][8], topic transition pattern mining [9], etc.

To capture the time-series features of topics, DTM and its related-models [8][10] assume dynamic drift of distributions. Although the DTM-based models appropriately find topics over time, they require expensive computational cost, which can be a critical drawback in some applications. On the other hand, there is a large body of work developing efficient inference algorithms for LDA [11][12][13] because of its simpler architecture compared to DTM. While both models learn and work differently and even give different results, some practitioners and researchers employ LDA instead to analyze the time-series nature to take advantage of its efficiency, and these attempts seem to be successful according to the literature [14].

The question that arises in this background is; if time series topics information can be extracted by using LDA, which is faster than DTM, then why do we need to use DTM? To answer this question this research is conducted with a problem statement "*Can time-series topic information of DTM be extracted from LDA?*" To the best of our knowledge, there have been no studies that extensively compared the information extracted using LDA with that of DTM.

Topic drifting and topic popularity are fundamental time-series information that can be extracted from DTM. Topic drifting is the topic transition over time and popularity is the measure of topic proportion at each time slice. The challenging part in topic transition analysis is that, DTM topic set has a sequential structure whereas LDA topic set has no sequential information at all. To map the unstructured topic set with DTM topics, we used a probability distribution similarity method.

Based on this matching, we analyzed both topic sets, and in this process, we encountered with fragmentation issue, which we will describe later (Figure 1). DTM provides the time evaluation of topics, which means one single DTM topic can shift to a new subject if compared with the initial time's topic subject, whereas LDA topic's theme remains the same because LDA has no time aspect. This shifting in DTM topics is called fragmentation. In this experiment, we found that some DTM topics contain the information of two or more LDA topics; in other words, they have two or more fragmented topics.

We built time-series topic population graphs for topic popularity analysis. There are pros and cons for each model. LDA extracts the focus on the collection of topics, whereas DTM can find connections between different themes and how subjects interchange within the same domain or topic.

Even though DTM has the edge of finding topic transitions over time, mostly constructing only population graphs for LDA topics is enough for time-series analysis [14]. Some specific problems require DTM to extract topic transition despite its high computation cost [15].

The rest of the paper is organized as follows: In the next section, we describe the closely related background research. In Section 3, we present an overview of computing time series topic estimation for LDA topics. Section 4 describe about similarity analysis. Section 5 is about datasets and models used for the experiments. Results are shown in Section 6. At the end, few discussion points are mentioned in Section 7 and conclusion is presented in the last section.

II. Background

D. Koike et al. [3] proposed a method that draws a time-series graph to find the bursty topic detection in Twitter data individually, as well as with correlated news, by using DTM [2]. They extracted 50 topics from a subset of news articles and Twitter about *The London Olympics*. Khan et al. [14] performed a similar type of experiment using LDA. In their experiment, LDA was trained on 1000 topics on hashtag-pooled documents of English tweets. They then created an inference dataset from the same dataset using day-hashtag tweet pooling. In the end, they created time-series graphs of topics that showed the topic popularity, topic burstiness, and interval of bursty topics. In general, [3] and [14] are the same but used different topic models. We want to know why.

Before applying topic modeling, some preprocessing steps are required because documents are messy in general. Applying linguistic preprocessing may be of some help [16]. For Twitter dataset, tweet pooling was used that has been proposed as an intuitive solution [17][18] when models perform poorly because of small document size. Hashtag pooling [19] and day-hashtag [14] were used.

III. Time-Series Topic Estimation by LDA

LDA topics information is organized by time to compare it with DTM topics. LDA assumes a latent topic distribution for each document d denoted by θ_d and a latent topic assignment z_i for each word w_i in a document. The word w_i is drawn from a distribution of words associated to the assigned topic $z_i = k$, which is denoted by ϕ_k. We trained the LDA with multiple datasets without any modification to the LDA machinery. Formally, when we denote a set of documents that we would like to analyze by $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_D\}$, we simply use X as a training dataset for ordinary LDA training. Before the LDA training, we apply a pooling method when we deal with a short text dataset such as Twitter. In that case, each document $\mathbf{x}_d$ consists of multiple text instances (e.g., tweets). We denote the number of instances that are contained in a document $\mathbf{x}_d$ by T_d. If no pooling method is applied, $T_d = 1$ for all documents.

For the inference part that estimates the number of documents for each topic, we take the time information into account. Each document $\mathbf{x}_d$ is associated to a specific time slice, which we denote by $\tau(\mathbf{x}_d)$. Let $X_t = \{\mathbf{x} \in X | \tau(\mathbf{x}) = t\}$, be a set of documents in time slice t. We estimate the topic distribution θ_d for each document in X and calculate the estimated number of documents for each topic k at each time slice t, denoted by N_k^t, using the following equations:

$$N_k^t = \sum_{d:\mathbf{x}_d \in X_t} \theta_{dk} T_d \tag{1}$$

Probability distribution θ is calculated using Dirichlet distribution by applying LDA to the input data.

Given words $\mathbf{x} = w_1, \ldots, w_M$, we estimate the distribution of θ.

$$p(\theta|\mathbf{x}) = \sum_{\mathbf{z}} p(\theta|z) p(\mathbf{z}|\mathbf{x}) \tag{2}$$

where corresponding topics $\mathbf{z} = z_1, \ldots, z_K$, and the summation are over all possible assignments of $\mathbf{z}$. Since summation is analytically intractable, we apply Monte Carlo approximation with only one sample. We obtain a sample $\hat{z}$ from $p(z|x)$ using (collapsed) Gibbs sampling with five iterations. This approximation reduces the equation into the posterior probability of θ given $\hat{z}$.

$$p(\theta|x) \approx p(\theta|\hat{z}) \tag{3}$$

The posterior is a Dirichlet distribution of which the expectation $\hat{\theta}$ is:

$$\hat{\theta_k} = \frac{n_k + \alpha_k}{N + \sum_{k'} \alpha_{k'}} \tag{4}$$

where $n_k = \sum_{i=1}^{N} \delta(\hat{z_i}, k)$, i.e., the number of topic k in $\hat{z}$.

The final step is to estimate the number of documents to make the time-series popularity graphs.

IV. Similarity Analysis of DTM and LDA Topics

The DTM and LDA topics are in the form of word and probability distributions. We extract the top 50 words for all topics so word distribution is $K \times 50$ in LDA and K is the number of topics. Due to very low probability density of lower ranked words, Top 50 words are enough to convey the meaning of a topic. The top words may change in a DTM topic over time, so overall word distribution of a DTM topic varies, but it is $K \times 50$ for one time slice, the same as a LDA topic. To check the relation between topics, we use a widely accepted similarity measure, the Jensen-Shannon (JS) divergence [20].

We apply normalization on both the DTM and LDA topic-word distributions because we consider only top 50 words. We denote the normalized distribution for the kth LDA topic by $\tilde{\phi}_k$ and jth DTM topic at time slice t by $\tilde{\phi}_j^t$. The JS divergence between these distributions is defined as:

$$JSD(\tilde{\phi}_k || \tilde{\phi}_j^t) = \frac{1}{2} D(\tilde{\phi}_k || T_M) + \frac{1}{2} D(\tilde{\phi}_j^t || T_M) \tag{5}$$

where

$$T_M = \frac{1}{2}(\tilde{\phi}_k + \tilde{\phi}_j^t) \tag{6}$$

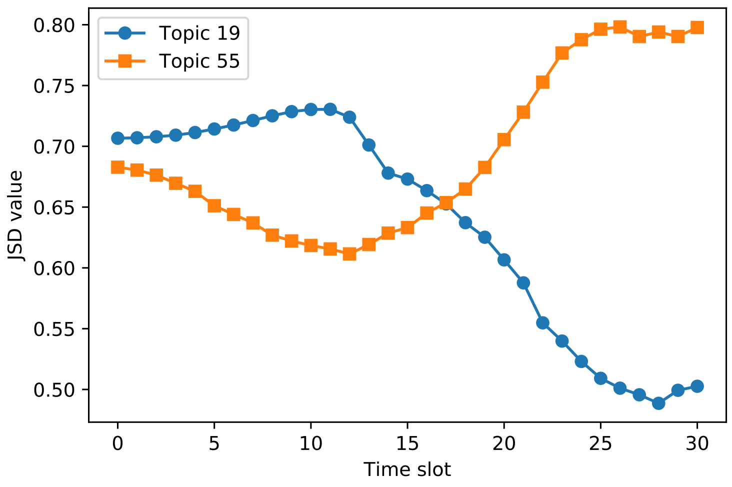
$D(\tilde{\phi}_k || \tilde{\phi}_j^t)$, is the Kullback-Leibler divergence:

$$D_{KL}(\tilde{\phi}_k || \tilde{\phi}_j^t) = \sum_{w \in W} P(w|\tilde{\phi}_k) \log \left(\frac{P(w|\tilde{\phi}_k)}{P(w|\tilde{\phi}_j^t)} \right) \tag{7}$$

A. Matching DTM and LDA topics

JS analysis tells us about the information overlap between DTM and LDA topics and is a good way to confirm either the topics are similar in both models or not. This analysis also illustrates fragmentation. An example is shown in Figure 1, where we see that the sample DTM topic was "Tensor decomposition for signal processing" at start, but later the topic's theme shifted rapidly towards "Tensor decomposition" and "Signal processing" was no longer significant. Whereas "Tensor decomposition" and "Signal Processing" are two different topics in LDA analysis. This phenomenon is called

T=0, (year = 1987)	**T=5, (year = 1992)**	**T=10, (year = 1997)**	**T=15, (year = 2002)**	**T=20, (year = 2007)**	**T=25, (year = 2012)**	**T=30, (year = 2017)**
matrix, vectors, components, component, analysis, principal, signals, signal, matrices, spectral, column, eigenvalues, source, orthogonal, eigenvectors, ...	matrix, component, components, analysis, principal, vectors, signal, source, signals, matrices, pca, sources, spectral, independent, separation, ...	matrix, component, components, analysis, independent, source, signal, sources, separation, principal, ica, signals, pca, blind, matrices, ...	matrix, signal, source, components, analysis, component, sources, signals, ica, matrices, independent, pca, separation, principal, subspace, ...	matrix, matrices, pca, rank, analysis, signal, components, source, columns, component, sources, reconstruction, subspace, ica, vectors, ...	matrix, rank, matrices, norm, low, pca, subspace, tensor, columns, entries, column, principal, singular, decomposition, completion, ...	matrix, rank, matrices, tensor, low, spectral, norm, subspace, decomposition, entries, error, singular, sparse, completion, pca, ...



LDA Topic 19	**LDA Topic 55**
rank, matrices, norm, tensor, entries, decomposition, columns, column, subspace, spectral, singular, completion, privacy, row, svd, ...	noise, signal, components, component, filter, signals, source, filters, coefficients, mixture, noisy, sources, ica, separation, mixtures, ...

Fig. 1. DTM and LDA trained on NeurIPS dataset for $K = 60$: Few words of DTM $\tilde{\phi}_0^t$ at $t = 0, 5, 10, 15, 20, 25, 30$ are shown in the first table. Second table shows few words of LDA $\tilde{\phi_{19}}$ and $\tilde{\phi_{55}}$ respectively, and both curves in the graph are the JS similarity measure of $\tilde{\phi}_0^t$ with LDA $\tilde{\phi_{19}}$ and $\tilde{\phi_{55}}$. This is a graphical representation of two fragmented LDA topics related to one single DTM topic.

fragmentation. By subjective analysis, JS value of 0.7 is selected as threshold value for fragmented topics analysis.

Formally, jth DTM topic is *related* to the kth LDA topic if there exists t such that $JSD(\tilde{\phi}_{L,k}, \tilde{\phi}_{D,j}^t) \leq 0.7$. kth and lth LDA topics are fragmented topics of the jth DTM topic if the jth DTM topic is related to both the kth and lth LDA topic.

B. Topic popularity analysis

The time-series topic popularity, which is the second important information offered by DTM, can be extracted from the LDA topics. After calculating document-topic distribution θ_{dk}, the documents are categorized with the same time-series information as used in DTM. Then, we calculate the estimated number of documents for each topic in a time series manner and construct a graph that is comparable to the DTM topics popularity information.

V. Experiment

The experimental process started with collecting and preparing the datasets. Then, appropriate configurations for DTM and LDA models were selected. After training both models with one dataset at a time, we extracted topics-word distributions and word probability. These word probabilities were used for computing the JS divergence and we made the JS similarity-based graphs. We also plotted population graphs using LDA inference to compare it with the DTM topics.

A. Datasets

Three different datasets were used in this experiment.

NeurIPS: This dataset consists of research papers from the conference of neural information processing systems (NeurIPS formally known as NIPS) from 1987 to 2017 (30 time slices).

Total of 7239 research papers were used. In the preprocessing, we removed stop words, special characters, URLs, and words having only two characters because most two characters words do not have concrete meanings.

Twitter: Tweets2011 [21] dataset consists of more than three million English tweets sampled between January 23 to February 8, 2011 (17 time slices). As the original dataset consists of tweets in many languages, we used the Python library *langdetect* to extract the English tweets. Usually, a tweet is a messy piece of text, so some preprocessing is desirable as the first step in cleaning this data. We therefore removed stop words, usernames, URLs, special characters, and two-letter words. After applying day-hashtag pooling, 408,200 documents were obtained and became part of the training dataset.

News: We use Thompson's dataset [22] consists of 204,135 news articles from 18 American publications. There are 191,530 articles that have date information and also the distribution of articles over the years is sparse. We therefore selected articles from 2016 and 2017, totaling 95,997 and 75,034 respectively. Thus, a total of 171,031 news articles were divided into 24 time slices based on the month-year parameter for DTM training and the inference of LDA. The same preprocessing steps were applied to this dataset as mentioned above for the other datasets.

B. *Models configuration*

LDA with the stochastic variational Bayesian method [23] in Java with number of topics K, 1000 docs per batch, and 1000 iterations was trained with the above-mentioned datasets one at a time.

DTM was implemented using the gensim.model.wrappers with DTM implementation in C and C++. We trained the DTM on three different number of topic configurations with each dataset.

Topics: We selected three values (20, 50, 100) for training time and (30, 60, and 90) for topic drifting and topic popularity as the hyperparameter "number of topics", denoted as K.

VI. Results

This section is divided into multiple sub-sections and each part explains the different aspects of our research.

A. *Training time cost*

As mentioned earlier, the computational cost for DTM is higher than LDA. However, to determine the difference in training time, we conducted a small sub-experiment in which we trained both models with multiple-size datasets and hyperparameter value K. The dataset used for this experiment was the "Twitter" dataset. Preprocessing cleaning and hashtag pooling were applied before training.

Figure 2, shows that increasing the number of documents or the number of topics increase the training time. For small datasets, the training time of DTM was 10X more than LDA and exceeds **"100 times"** for big datasets. Normally in NLP, datasets are relatively bigger in size, therefore we can say that DTM will take around 100X more time for training compared with LDA under the same conditions.

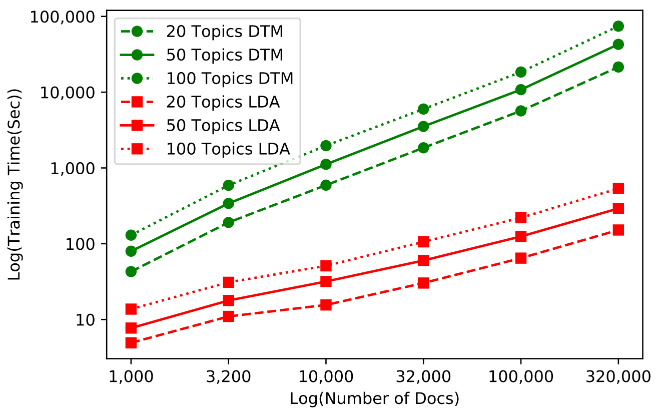


Fig. 2. The graph is in logarithmic scale to fit higher values in the figure. x-axis is document-size value and y-axis is the time in seconds that each model took for training.

B. *Topic drifting*

A single DTM topic consists of topics at each time slot. For clarity, let us call such a time-slice-topic the "focus" of the DTM topic. The focus of a DTM topic changes over time, as shown in the first part of Figure 1, where the focus changed from "Signal Processing" to "Tensor Decomposition" by the end. This is called topic drifting or topic transition. We calculated the total unique vocabularies for each DTM topic. V_s is the vocabulary size, which is the number of unique words that appeared in all time slots normalized topic-word distributions of a single DTM topic. The minimum vocabulary size for any topic was 50. If any topic had V_s close to this number, it means there were few new words in the different time slot topics. In short, the focus of this specific topic remained the same and there was no topic drifting.

In Table I,$K(V_s > 70)$ means the number of DTM topics having a vocabulary size of more than 70. Similarly $K(V_s > 90)$ and $K(V_s > 120)$ mean the number of topics with V_s more than 90 and 120, respectively. These values for the Twitter dataset are very low, which means there were not many new words in the DTM topics and the focus of the topics remained the same over all times. This implies that topic drifting for the Twitter dataset is negligible. And $K(V_s > v)$ values for the DTM trained on NeurIPS and News dataset were relatively high, which implies that there were topic drifting phenomenons.

C. *JS analysis*

To extract the information overlap of the DTM and LDA topics, we computed JS values using (5) for all the datasets in all topic configurations. The JS value is bounded by 0 and 1 for two distributions, where 0 means both distributions are identical and 1 means there is no similarity between both probability distributions. A threshold value of 0.7 was selected

TABLE I
TOPIC DRIFTING: $K(V_s > v)$ VALUES INDICATES THE NUMBER OF DTM TOPICS OF WHICH $V_s > v$.
FRAGMENTATION: RT(RELATED TOPIC) AND FT (FRAGMENTED TOPIC) ARE BASED ON JSD VALUES OF LDA TOPICS WITH DTM TOPICS.

Configuration		Topic Drifting			Fragmentation				
Dataset	**Topics**	$K(V_s > 70)$	$K(V_s > 90)$	$K(V_s > 120)$	RT	FT	F 2	F 3	F 4 & more
NeurIPS	30	13	8	0	17	4	3	1	0
	60	58	56	11	42	16	11	5	0
	90	90	90	90	69	28	25	2	1
Twitter	30	3	1	0	5	0	0	0	0
	60	3	0	0	11	1	1	0	0
	90	1	0	0	14	3	2	1	0
News	30	29	20	5	8	1	1	0	0
	60	57	33	1	24	2	2	0	0
	90	83	37	2	42	4	4	0	0

and any DTM topic distribution having a JS value lower than or equal to this threshold when measured with the LDA topic distributions was part of the related topic "RT", fragmented topic "FT", and others. A summary of this analysis is set forth in Table I under Fragmentation column. "RT" is the total number of DTM topics having a relationship with the LDA topics. "FT", "F2", "F 3", and "F 4 & more" are the number of DTM topics having a JS relationship with two or more LDA topics, only two LDA topics, only three LDA topics, and more than three LDA topics, respectively.

Data shows that a negligible amount of "FT" (fragmented topics) was found for the datasets "News" and "Twitter" because most news articles and tweets are instantaneous responses of some events, and these topics die within short period of time; in other words, we see other tweets and article about other events. Due to this focus shifting behavior of the documents, DTM cannot accurately locate topic transitions over time. That is why very few fragmented topics were found for these datasets. Related topics "RT" are comparatively higher for "News" as compared to "Twitter" dataset because the domain of tweets is huge; it could be anything ranging from personal (My pet is very cute) to political (US president announced a restriction on trade agreement with China), whereas the News articles domain is restricted compared with Twitter. We can therefore have many topics in the Twitter dataset and due to random initial conditions of both DTM and LDA, it is safe to say that both models could come up with different topics. As mentioned, the News dataset domain is restricted so we see high topic overlapping in News dataset.

The domain of the "NeurIPS" documents focused on a few subjects (machine learning, artificial intelligence, computational neuroscience, etc), so related topics' "RT" values are very high compared with other dataset configurations. High fragmented topic "FT" values can be seen for the "NeurIPS" dataset in Table I because research papers tend to follow previous researches or somehow align with previous research papers. That is why we can see a well-defined topic transitions in the DTM topics, as shown in Figure 1.

In all the datasets, increasing the number of topics resulted in an increase in "RT" and "FT" values. Table II shows, if we increase the number of topics in LDA, we get more and more fragmented topics, which means that topics are further divided

TABLE II
FRAGMENTATION BEHAVIOUR WHEN LDA IS TRAINED WITH HIGH NUMBER OF TOPICS: K WAS 30 FOR DTM AND **1000** FOR LDA.

Dataset	**RT**	**FT**	**F 2**	**F 3**	**F 4 & more**
NeurIPS	25	20	3	7	10

into smaller and more focused themes. DTM's computation cost restricts us from increasing the number of topics, so we cannot get the type of topics that we can get from LDA with a very high K hyperparameter.

D. Time dependent topic popularity

For DTM, time-series topic popularity is estimating the number of documents for each topic at each time slice. We can easily construct this information into a self-explanatory graphical representation of topic popularity. For this analysis, we selected the 60 topics "NeurIPS" configuration. Then, γ distributions for the documents were computed. A γ distribution is the probability of each topic for a document. Summation over the topics then, gives us the document estimation. For LDA, model training is done with same configuration. Inference dataset was constructed before extraction of θ_{dk} distributions by coupling date information with documents.

With θ_{dk}'s and X_t's, by using (1) and (2), we got N_k^t and built graphs. To reduce noise effects and to make the graphs smooth, we used the Savitzky-Golay digital filter [24]. These graphs are shown in Figure 3. These graphs show that the time-series topic popularity can be extracted [14] from DTM as well as LDA (Details in Discussion section).

VII. DISCUSSION

Topic drifting and topic popularity are the main aspects of this research and we compared these aspects for DTM and LDA. In this section, we discuss a few important points of both concepts.

Topic drifting: There is no topic drifting for DTM trained on the "Twitter" dataset (Table I), so the only important information which can be extracted from such datasets is the time-series topic popularity which can be extracted using LDA, thus we should avoid the high cost DTM. For the "NeurIPS" dataset, the topic drifting increased with an increase in number

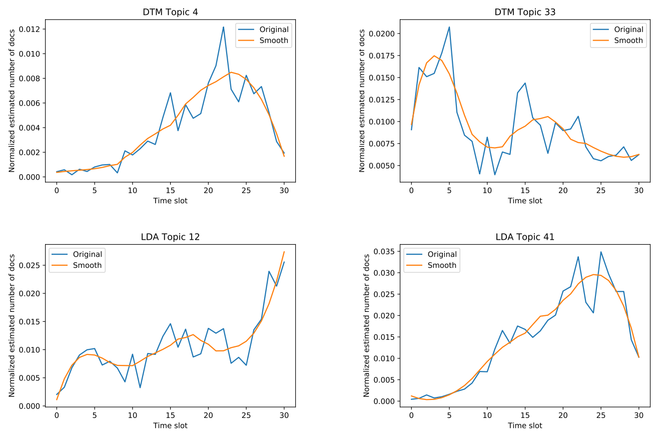


Fig. 3. Two topics from each model are shown here. The horizontal axis shows the time and the vertical axis is the normalized estimated number of documents for these topics.

of topics. All 90 topics have V_s greater than 120 for this dataset (Table I), which means there was high topic drifting which provides rich insights of topic transitions. Therefore, if we want to examine topic drifting in such datasets, DTM is a promising option; however, we must keep in mind that if our goal is topic popularity, then LDA is a far better option. Topic drifting can be experienced for "News" dataset, but the vocabulary size is comparatively low for the higher number of topics. This means that we do have topics drifting with such datasets, but it may not be as effective as we want. More subjective analysis based on problem statement can help to choose better model. An interesting finding is sometimes DTM tends to forcefully find the topic transition. For example, in the 30-topic "News" dataset configuration, topic 29 started with words *(archive, team, collection, sign, projects, machine, contains, lost, websites, wayback)*, but the word distribution at the end was *(travel, airport, flight, trip, passengers, travelers, plane, airlines, united, airline)*. Looking at these distributions, we can say that DTM failed to extract the correct topic drifting for topic 29.

Topic popularity: Once the models are trained, we can extract γ and θ distributions for any document. With θ distributions using the method described in Section 3 for the LDA model, we can construct topic popularity graphs. Similarly, we can construct these graphs for DTM topics using γ distributions. Thus, this information can be extracted using both models. Notably, the topic popularity extracted from DTM is a little vague because DTM topics have topic transition information embedded within the topics. To explain this phenomenon with an example, we manually selected 2 topics from DTM and 2 topics from LDA; DTM topic 4 shown in Figure 3 is *(retrieval, content, query, text, semantic, lda, relevant, word, latent, topics)* at T_0, which is about "Information retrieval from documents" and the word distribution at T_{30} is *(topic, document, lda, word, topics, latent, dirichlet, text, allocation, model)*, which is about "Document analysis with LDA". Similarly, DTM topic 33 was about "Language structure rules" at initial time slots and the theme of the topic was changed to "Question-answer reasoning" around at the end. Therefore, if we are looking for the popularity graph of a topic "Information Retrieval", then LDA topic 12 is a more accurate option. Similarly, LDA topic 41 is more accurate if we want to see the popularity graph of the topic "Variational topic model LDA" because there is no topic drifting in LDA. The word distribution for LDA topic 12 is *(word, language, sequence, recurrent, text, semantic, context, attention, table, embedding)* and for LDA 41 is *(latent, topic, sampling, mixture, gibbs, dirichlet, lda, markov, document, likelihood)*. Because of the topic transition information embedded with DTM topics, DTM is not the best option for time-series topic popularity information.

VIII. Conclusion

In this research, we executed a comprehensive study on the time-series analysis of the popular topic models DTM and LDA. Our research focused on the time-series information of topic drifting and topic popularity. To compare both models, we tried to extract this information from the topic distributions. Multiple datasets along with multiple topic configurations were used for this experiment.
Our findings are:

1) DTM takes 100 times longer to train the model as compared to LDA for large datasets.
2) Topic drifting is a unique property of DTM that is difficult to extract from LDA, but datasets like "Twitter" do not have topic transition information, so applying DTM to such datasets is waste of resources.
3) Time-series topic popularity can be extracted from both models, but it is precise from LDA because DTM has topic transition information embedded in the topics.

Fragmentation of topics was also detected in this process from the datasets focused on one domain, e.g., "NeurIPS", which is another interesting aspect of this research and could be studied in the future. To summarize, topic popularity — common information needed as time series information — should be extracted using LDA because it is faster and provides concrete information. However, if topic drifting is required, then DTM is the only option, although sometimes, it may give inaccurate information.

Based on our findings and research experiment with multiple datasets configurations, we suggest the usage of DTM and LDA in different case scenarios (Table III).

TABLE III
SUGGESTIONS BASED ON FINDINGS

Use LDA for	**Use DTM for**
Twitter with high K	NIPS data with few number of topics
News with high K	News with smaller K
Short duration datasets e.g. Twitter	Long duration docs e.g. NIPS
Extract topic popularity	Extract topic drifting

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet Allocation,” Journal of Machine Learning Research, vol. 3, pp. 993-1022, 2003

[2] D. M. Blei and J.D. Lafferty, “Dynamic topic models,” Proceedings of the 23rd international conference on Machine learning, 2006, pp.113-120

[3] D. Koike, Y. Takahashi, T. Utsuro, M. Yoshioka, and N. Kando, “Time series topic modeling and bursty topic detection of correlated news and twitter,” Proceedings of the Sixth International Joint Conference on Natural Language Processing, 2013, pp.917-921

[4] Noriaki Kawamae, “Trend analysis model: trend consists of temporal words, topics, and timestamps,” Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 317-326

[5] H. Zhang, G. Kim, and P. E Xing, “Dynamic Topic Modeling for Monitoring Market Competition from Online Text and Image Data,” Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp.1425-1434

[6] J. Kalyanam, A. Mantrach, D. Saez-Trumper, H. Vahabi, and G. Lanckriet, “Leveraging Social Context for Modeling Topic Evolution,” Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp.517-526

[7] W. Xie, F. Zhu, J. Jiang, E. Lim, and K. Wang, “TopicSketch: Real-Time Bursty Topic Detection from Twitter,” IEEE Transactions on Knowledge and Data Engineering, vol. 28, pp.2216-2229, 2016

[8] H. Amoualian, M. Clausel, E. Gaussier, and M. R. Amini, “Streaming-LDA: A Copula-based Approach to Modeling Topic Dependencies in Document Streams,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp.695-704

[9] Y. Kim, J. J. Han, and C. Yuan, “TOPTRAC: Topical Trajectory Pattern Mining,” Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp.587-596

[10] A. Acharya, J. Ghosh, Jand M. Zhou, “A Dual Markov Chain Topic Model for Dynamic Environments,” Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp.1099-1108

[11] A. Q. Li, A. Ahmed, S. Ravi, and A. J. Smola, “Reducing the sampling complexity of topic models,” Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 891-900

[12] L. Yut, C. Zhang, Y. Shao, and B. Cui, “LDA* a robust and large-scale topic modeling system,” Proceedings of the VLDB Endowment, 2017, pp.1406-1417

[13] J. Chen, J. Zhu, J. Lu, and S. Liu, “Scalable training of hierarchical topic models,” Proceedings of the VLDB Endowment, 2018, pp.826-839

[14] M. H. U. R. Khan, K. Wakabayashi, and S. Fukuyama, “Events Insights Extraction from Twitter Using LDA and Day-Hashtag Pooling,” Proceedings of the 21st International Conference on Information Integration and Web-based Applications Services, 2019, pp.240-244

[15] M. Huang, M. Zolnoori, J. E. Balls-Berry, T. A. Brockman, C. A. Patten, and L. Yao, “Technological innovations in disease management: text mining US patent data from 1995 to 2017,” Journal of medical Internet research, vol. 21, 2019

[16] B. Han, P. Cook, and T. Baldwin, “Automatically constructing a normalisation dictionary for microblogs,” Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning, 2012, pp. 421-432

[17] L. Hong and B. D. Davison, “Empirical study of topic modeling in twitter,” Proceedings of the first workshop on social media analytics, 2010, pp.80-88

[18] W. X. Zhao, J. Jiang, J. Weng, J. He, E. P. Lim, H. Yan, and X. Li, “Comparing twitter and traditional media using topic models,” European conference on information retrieval, 2011, pp. 338-349

[19] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, “Improving lda topic models for microblogs via tweet pooling and automatic labeling,” Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, 2013, pp. 889-892

[20] J. Lin, “Divergence measures based on the Shannon entropy,” IEEE Transactions on Information theory, vol. 37, pp. 145-151, 1991

[21] TREC, “Tweets2011,” https://trec.nist.gov/data/tweets/, 2011, [Online; accessed 2021.03.15]

[22] A Thompson, “204,135 articles from 18 American publications,” https://components.one/datasets/all-the-news-articles-dataset/, 2018, [Online; accessed 2021.03.15]

[23] D. Mimno, M. Hoffman, and D. Blei, “Sparse stochastic inference for latent Dirichlet allocation,” Proceedings of the 29th International Coference on International Conference on Machine Learning, 2012, pp. 1515–1522

[24] H. William and S. A. Teukolsky, “Savitzky-Golay smoothing filters,” Computers in Physics, vol. 4.6, pp. 669-672, 1990

Analyzing Impact of COVID-19 Pandemic on Global Stock Prices

Yoshihisa Udagawa
Faculty of Informatics, Tokyo University of Information Sciences
Chiba-city, Chiba, Japan
e-mail: yu207233@rsch.tuis.ac.jp

Abstract— **The spread of COVID-19 is making a serious impact on the world economy. As policies to maintain economic activities have been implemented in a timely manner, many stock markets have regained their stock prices to pre-corona levels, although there are still strong opinions that the prices fail to reflect the real economy. This paper describes results of statistical analyses of daily historical data concerning Dow Jones Industrial Average (DJIA), Nasdaq Composite Index (NASDAQ), France Stock Index (CAC), German Stock Index (DAX), Shanghai Composite Index (SSEC) and Nikkei Stock Average (Nikkei). In general, these stock prices plunged for approximately 20 days before the trading day that marked the lowest price, and increased for the next 25 days. Reflecting the fact that the world economies are tightly related to each other, it is confirmed that stock price fluctuations under study generally show the same trend. We propose a technical indicator defined by the difference between a stock price and moving average. The results of our experiments show that the indicator predicts short-term trends with a slight time lag.**

Keywords*— *COVID-19 infection; Technical indicator; Global market comparison; Candlestick chart.

I. INTRODUCTION

More than a year has passed since the first Covid-19 infection was confirmed. But the spread of coronavirus is still continuing and the number of daily infections remains high. The spread of COVID-19 is forecasted to have a significant adverse impact on the global economy. While the global real Gross Domestic Product (GDP) grew by 2.9 percent in 2019, it is predicted that COVID-19 will cause GDP growth to decrease by three percent in 2020 [1]. World stock markets have experienced a large crash in the first quarter of 2020. Concerns about a further plunge of stock prices prevailed over global markets. However, due to the economic policies of each country, stock prices of world markets have turned from falling to rising in Mar. 2020.

It can be easily inferred that there is a difference in the degree of collapse and recovery reflecting the situation in each country. What is missing is a comparison of the stock price fluctuations of world markets from a statistical point of view. In addition, because it is rare for the world stock price indexes to fall all at once and then recover, we come up with the idea of analyzing the history of the world stock price plunge and surge to deepen our understanding of stock price movements. This study discusses a comparison of representative stock indexes of the U.S., European, and Asian markets [2]. Specifically, we focus on Dow Jones Industrial Average (DJIA), Nasdaq Composite Index (NASDAQ), France Stock Index (CAC), German Stock Index (DAX), Shanghai Composite Index (SSEC), and Nikkei Stock Average (Nikkei). Daily historical data are used for the research.

Dimson et al. [3] recommend investing in the U.S. markets rather than emerging ones because of the growth rate of stock prices and the stability of the investment environment. Reference [4] compares profitability of the U.S. and Asian markets by simulations that find buy-timing using a candlestick pattern model consisting of six parameters. The results of the simulation shows that the profitability of the U.S. markets outperforms other markets. However, these studies were carried out before the spread of the COVID-19 coronavirus. There is no guarantee that the situation is the same after the pandemic.

Ngwakwe [5] estimates how COVID-19 infections affected world stock indexes, i.e., Euronext 100, SSEC, DJIA and S&P 500. The results of analyses show that SSEC has resilience to COVID-19 pandemic with profit in stock values during the first fifty days into the pandemic, while the other indexes experience adverse impact from the COVID-19 pandemic with a significant loss at that time period. All stock indexes experience a higher variability or fluctuation of stock market prices.

Verma, et al. [6] statistically analyze the impact of the COVID-19 outbreak on global economic development. The indicators used in the analyses include S&P500 stock index, crude oil, gold, and 20-year treasury bond. They find that S&P500 stock index experiences high uncertainty from Feb. 2019 to Apr. 2020, i.e., the latest month of their research.

This paper aims to analyze stock price fluctuations of approximately 245 trading days before and after the COVID-19 pandemic, and statistically compare degrees of impact on the world markets.

The findings of this research are as follows:

I. The six stock market indexes used for comparison experience bottom prices within a week, specifically four business days from Mar. 18 to 23. After that, they never fall below the bottom prices.
II. The stock price best recovered in NASDAC, followed by Nikkei and DJIA, which is considered to reflect the degree of impact of the COVID-19 spread on each market.
III. It is observed that the trajectory of 25-day average of difference between stock price and 5-day average reverses on the lowest price day. We propose an indicator to predict trend reversal based on the observation.

Experimental results show that the proposed indicator correctly predicts at least a reversal of a stock price decline due to the COVID-19 outbreak.

The remainder of the paper is organized as follows. Section II gives the background of the candlestick chart. Section III shows the result of comparisons of the stock market indexes by the ratio of stock prices to the lowest price caused by the COVID-19 pandemic. Section IV presents the results of comparisons of stock price movements in terms of the average and standard deviation in statistics. Section V describes statistics of candlestick parameters around the lowest price. Section VI proposes and discusses an indicator that can properly predict stock price trend reversals. Section VII concludes the paper with our plans for future work.

II. CANDLESTICK CHART AND PRAMETERS

This section introduces formations of a candlestick chart. The candlestick attributes to be analyzed are identified.

A. *Formation of Candlestick*

As depicted in Figure 1, a daily candlestick is formed with the market's opening, high, low, and closing prices of a specific trading day [7]. The candlestick has a wide part, which is called *real body* representing the range between the opening and closing prices of that day's trading, as shown in Figure 1. The color of the *real body* represents whether the opening price or the closing price is higher. If the price rises, a hollow body is drawn suggesting *bullish* or buying pressure. Otherwise, a filled body is drawn suggesting *bearish* or selling pressure.

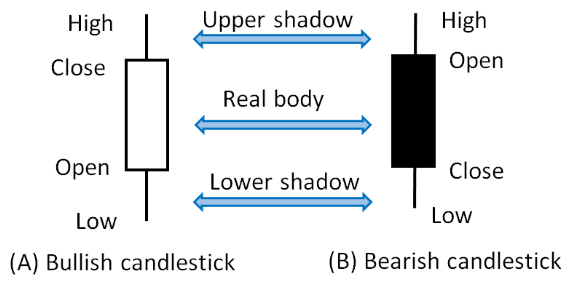


Figure 1. Candlestick formation

The thin lines above and below the body, which are named *shadows*, represent the range of prices traded in a day. The high is marked by the top of the upper shadow and the low by the bottom of the lower shadow.

B. *Candlestick Chart and Parameters*

A candlestick chart is a graph in which candlesticks are arranged in order of market days. It is used as a tool to get information on whether the current price is higher or lower than the historical stock price movements, and what kind of price movements have been made in a certain period of time. Moving averages form a line graph by connecting the average of closing prices over a certain period of time. The line graph is useful to decide whether stock prices are in a rising or falling trend by considering the relative position between the moving average and the current stock price. As for periods of time to compute averages, each country uses its own periods. For example, the short-term average is often calculated for 5 days, the medium-term average is for 25 days, and the long-term average is for 75 days in Japan.

Figure 2 illustrates indicators including averages that formalize a candlestick chart. In accordance with the candlestick chart notation, the following six attributes are used for analysis.

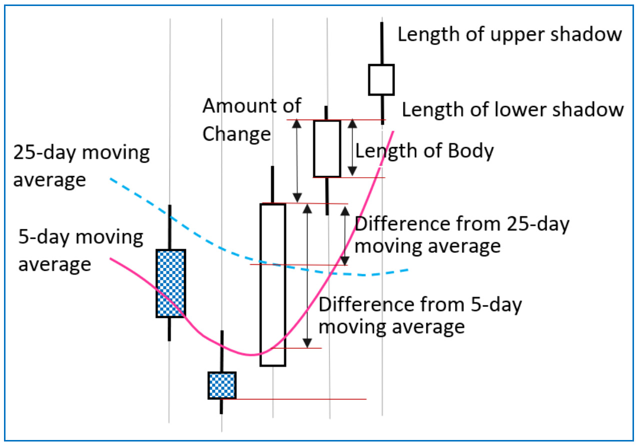


Figure 2. Candlestick chart and its parameters

(1) Amount of stock price change (the difference between a stock's closing price on a trading day and its closing price on the previous trading day)
(2) Length of candlestick body
(3) Length of upper shadow
(4) Length of lower shadow
(5) Difference between the stock price of a trading day and the 5-day moving average
(6) Difference between the stock price of a trading day and the 25-day moving average

III. COMPARISON OF STOCK INDEXES BY RATIO

This section describes the process of data analysis to follow how this research is performed. Stock price fluctuations of the six markets are compared concerning the lowest price during COVID-19 pandemic to understand impacts of COVID-19 on each market.

A. *Data Analysis Process*

Figure 3 overviews the data analysis process in this research that consists of the following operations.

(1) Downloading daily historical data from Web site
(2) Calculating the six attributes mentioned in Section II
(3) Extracting price data around the lowest price during COVID-19 pandemic
(4) Calculating 25-day average and standard deviation of the six attributes

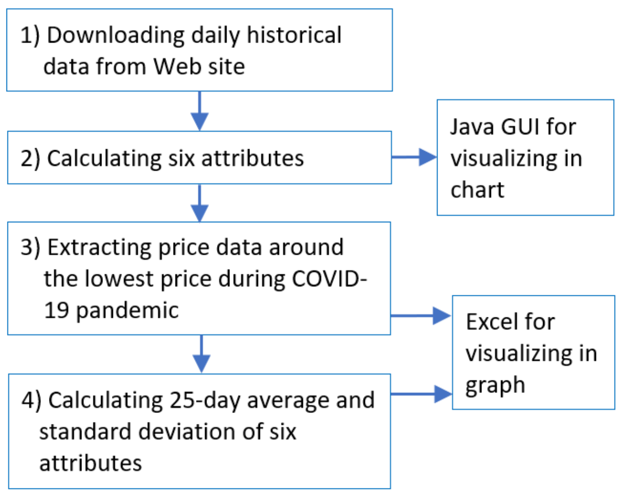

1) Downloading daily historical data from Web site
2) Calculating six attributes
Java GUI for visualizing in chart
3) Extracting price data around the lowest price during COVID-19 pandemic
Excel for visualizing in graph
4) Calculating 25-day average and standard deviation of six attributes

Figure 3. Overviews of data analysis process

Among the sites that provide global stock data, Web site [2] has provided useful daily stock data for more than 10 years in more than 40 markets. All data used in the research are downloaded manually.

Because the daily stock data only consist of close, open, high, low prices, and volume of stock trading, Java programs are developed for performing operations 2) to 4), and visualizing the candlestick chart. Excel is used for visualizing data of 3) and 4) in graphs.

B. *Comparison of Stock Price Fluctuations*

In order to understand overall structures of stock price fluctuations, we compare the stock price ratios to the lowest price that was recorded in Mar. 2020. Let CPr(n) be the closing price of a trading day n, and CPrMin be the lowest closing price recorded in Mar. 2020. The price ratio PrRatio(n) is calculated by the following formula, where CPr(1) represents the closing price of the latest trading day.

$$PrRatio(n) = (CPr(n) - CPrMin) * 100 / CPrMin \qquad (1)$$

Figure 4 is a graph of the price ratio of the six markets' stock indexes for approximately 490 days from 245 days before the lowest price to the latest market day, i.e., Mar. 12, 2021. It shows that all stock indexes keep rising after the lowest price recorded in Mar. 2020.

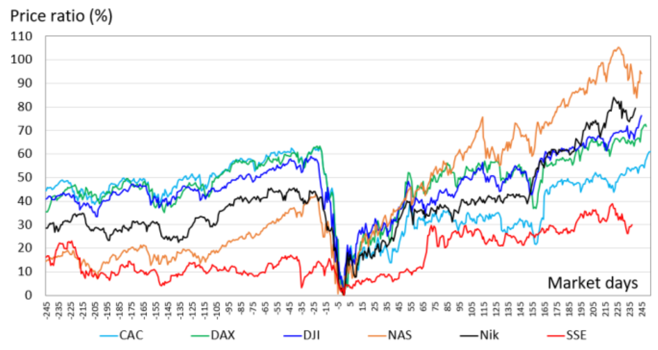


Figure 4. Ratio of stock index compared to the lowest price

Table 1 summarizes the stock price profiles compared to the lowest price. The lowest prices have been recorded from Mar. 18 to Mar. 23 in the six markets under comparison. Since Mar. 22, 2020 is Sunday, they show that the stock trends reversed from downtrend to uptrend in just 4 trading days in the six markets. The degree of plunge is 63.34% in DAX (Germany), followed by 62.76% in CAC (France), and 58.95% in DJIA (US), as shown in the column "highest price of pre-corona." The lowest decline was of 22.95% in SSEC (China).

TABLE I. SUMMARY OF STOCK PRICE INDICATORS COMPARED TO THE LOWEST PRICE

	Day of lowest price	Highest price of pre-corona (%)	Highest price of post-corona (%)	Recovery (%)
CAC40(France)	Mar. 18	62.76	61.03	-1.73
DAX(Germany)	Mar. 18	63.34	72.59	9.25
DowJones(U.S.)	Mar. 23	58.95	76.3	17.35
NASDAQ(U.S.)	Mar. 23	43.09	105.45	62.36
Nikkei(Japan)	Mar. 19	45.49	84.06	38.57
SSEC(China)	Mar. 23	22.95	38.94	15.99

A matter worthy of note is the degree of recovery from the lowest price. NASDAQ achieves the finest recovery of 62.36% rise as the highest price ratio after the corona is 105.45%, and the highest price ratio before the corona is 43.09%. Following NASDAQ, Nikkei (Japan) comes back by 38.57% rise. The slowest recovery is recorded –1.73% in CAC, followed by 9.25% in DAX. From the stock price movements of the two markets, it can be inferred that the impact of COVID-19 pandemic in Europe is larger than the other regions.

IV. COMPARISON BY AVERAGE AND STANDARD DEVIATION

This section presents the results of comparisons of the six market indexes adopted in this research with respect to the average and standard deviation in statistics. In order to compute the meaningful standard deviation, we calculate the average and standard deviation of the past 25 days [8] including the reference trading day. This period is commonly used for the calculation of the six attributes of a candlestick.

A. *Amount of Stock Price Change*

Figure 5 shows a graph of the 25-day moving averages and standard deviations of price changes of each stock market. As seen at the center part of Figure 5, the averages (in solid lines) of stock price changes plummet during approximately 20 days before the lowest trading day, i.e., around Feb. 20 in the five markets excluding SSEC. Stock price averages rally for approximately 25 days after the bottom, i.e., around April 24.

The standard deviations (in dashed lines) over this period, also increase in the range of 3.5 to 6 times of those of the other period reflecting the high volatility of price movements.

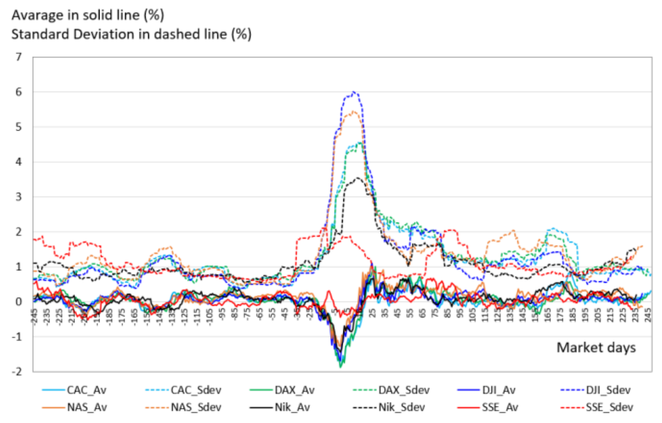


Figure 5. Average and standard deviation of price change

Taking a closer look at the standard deviations, DJIA reaches the maximum of 6.00%, followed by NASDAC of 5.47%, CAC of 4.56%, DAX of 4.54%, Nikkei of 3.53%, and SSEC of 2.14%, which is considered to reflect the strength of the impact on each market. The standard deviations of each market have roughly doubled after the lowest price, i.e., the right part, compared to those before the lowest price, i.e., the left part, which suggests that unstable trading has continued for roughly 180 days after the lowest price day.

B. Length of Candlestick Body

Figure 6 shows the 25-day averages and standard deviations of the candlestick lengths of the six stock markets. Figure 6 shows that the averages of SSEC are positive (plus) during the plunge period (–20 to 0), which means stock prices increase on average within a market day. The 25-day average of candlestick body lengths in NASDAQ is almost zero level during the plunge period. NASDAQ has consistently positive average during the period of stock price recovery (0 to 25), which means that the stock price continued to rise during trading hours. The four other markets experience remarkable negative averages, e.g., –1.39% in CAC, –1.15% in DAX, –0.967% in Nikkei, and –0.83% in DJIA at the lowest, suggesting that colored candlesticks are noticeable.

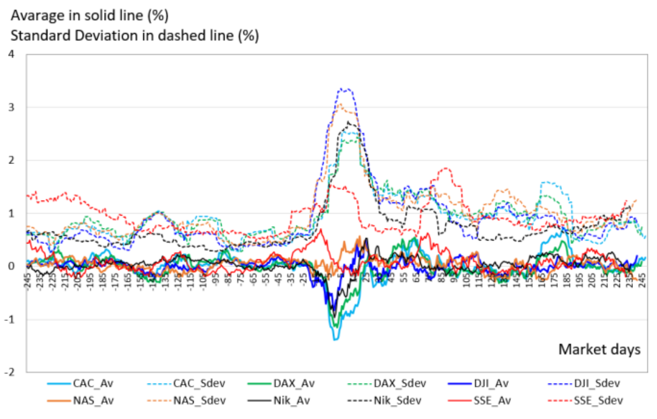


Figure 6. Average and standard deviation of candlestick body length

The largest standard deviation of 3.36% is marked in DJIA, followed by 3.07% in NASDAQ, 2.73% in Nikkei, 2.54% in CAC, 2.44% in DAX, and 1.85% in SSEC. Trends of the average and standard deviation of SSEC are notably different from the other markets.

C. Length of Upper and Lower Shadows

Figures 7 and 8 illustrate the 25-day averages and standard deviations of the lengths of the upper and lower shadows. The two figures look similar, revealing that the lengths of upper shadows are apparently correlated with those of the lower shadows.

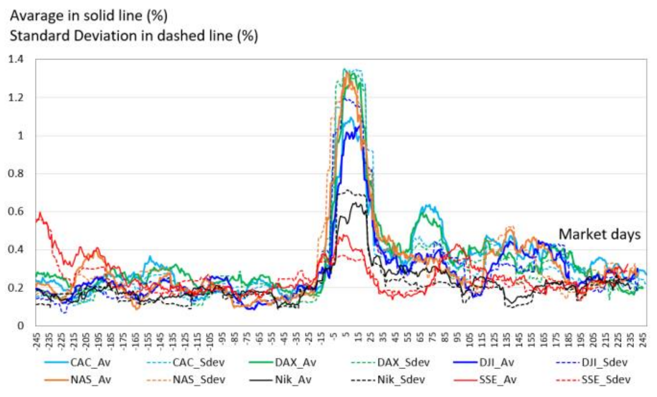


Figure 7. Average and standard deviation of upper shadow length

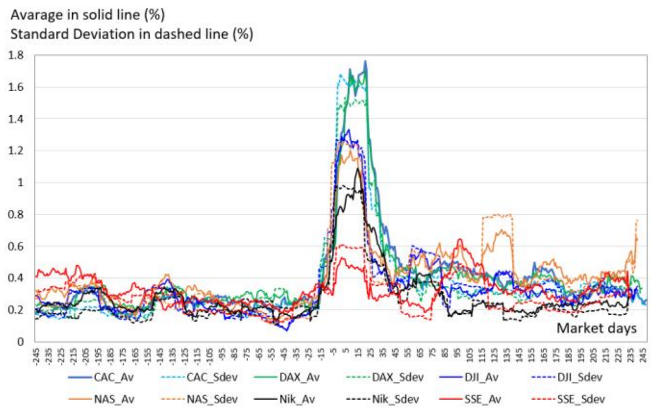


Figure 8. Average and standard deviation of lower shadow length

In the five markets excluding SSEC, the averages and standard deviations of the upper and lower shadows increase sharply during the period from approximately 10 days before the lowest price to approximately next 25 days.

Notably in the European markets of DAX and CAC, both the averages and standard deviations have surged about six times after the lowest price compared to these before the lowest price for the upper shadows, and about eight times for the lower ones. Those surges mean that rough price movements occur during the period.

On the other hand, the Asian market is relatively stable. In SSEC, the averages and standard deviations of shadows are doubled after the lowest price than before for the upper shadows, and are tripled for the lower ones. Nikkei marks about three times higher for the upper shadows, and about five times higher for the lower ones.

Figure 9 is a scatter plot of the lengths of upper and lower shadows in the Nikkei market. It illustrates that there is a strong correlation between the lengths of the upper and lower shadows.

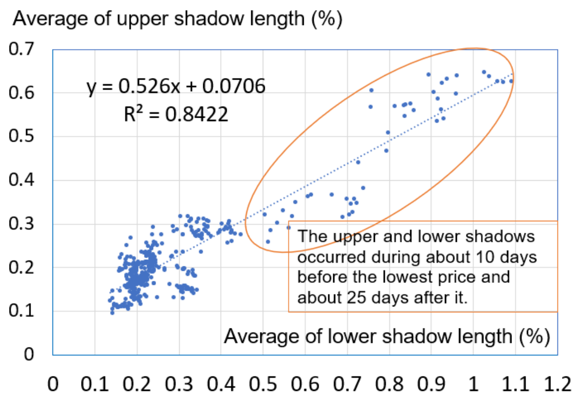


Figure 9. Scatter plot of upper and lower shadow length of Nikkei

The points surrounded by an ellipse correspond to the shadows that occur in the period between approximately 10 days before and approximately 25 days after the lowest price. These points occupy a different portion of Figure 9 from the rest of the points. *R Square* in statistics is 0.8422, which indicates that 84.22% of the upper shadow length can be explained by the lower shadow length, and vice versa.

D. Difference Between Stock Price and 5-day Average

Figure 10 shows the 25-day averages and standard deviations of the "difference between a stock price and 5-day moving average" for the six markets. The 25-day averages of the differences sharply decreased during approximately 20 days before the day that record the lowest, and increases approximately 25 days after the bottom.

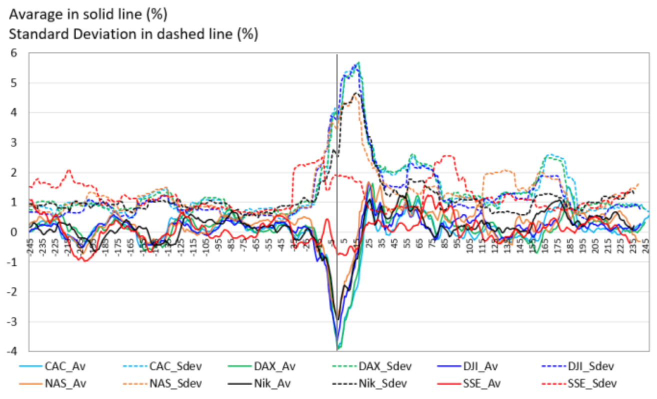


Figure 10. 25-day average and standard deviation of "difference between stock price and 5-day average"

In the five markets excluding SSEC, the day when the 25-day average of the "difference between a stock price and 5-day moving average" reverses the trend from downward to upward roughly coincides with a day when the stock price bottoms out. This is an important finding and detailed analyses are discussed in Section VI. The standard deviations almost reach their maximums during the period.

E. Difference Between Stock Price and 25-day Average

Figure 11 shows the 25-day averages and standard deviations of the "difference between a stock price and 25-day moving average" for the six markets. The graphs have a smooth shape as a whole reflecting the fact that the difference is computed between a stock price and 25-day moving average.

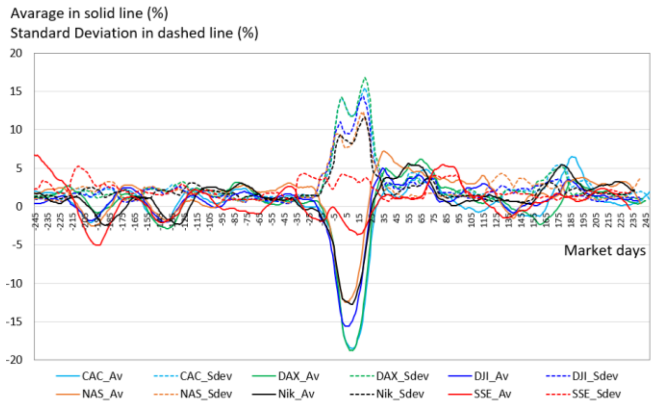


Figure 11. 25-day average and standard deviation of "difference between stock price and 25-day average"

Again, in the five markets excluding SSEC, the averages and standard deviations fluctuate largely during a period between approximately 20 days before the day that recorded the lowest price and the next 25 days.

V. STATISTICS OF PARAMETERS AROUND LOWEST PRICE

The averages and standard deviations of each attribute are examined to find out how the COVID-19 affects each market in 25 days before and after the lowest price day.

A. Amount of Stock Price Change

Figure 12 shows the average and standard deviation of the stock price change. All averages are negative 25 days before the lowest price day and positive after the day.

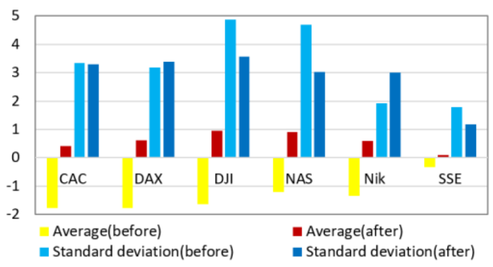


Figure 12. Average and standard deviation of stock price change

The largest average price change is 2.58% in DJIA as the result of the change from –1.63% to 0.95%, followed by 2.39% in DAX, 2.19% in CAC, 2.13% in NASDAQ, 1.94%

in Nikkei, and 0.434% in SSEC. The standard deviations notably decreased in Dow and NASDAQ with suggestion of stabilization of trading.

B. Length of Candlestick Body

Figure 13 shows that the averages of candlestick body lengths reverse the trend from negative to positive before and after the lowest price day in the four markets except for CAC and SSEC. The statistics of trend reversal indicate that there are many opportunities when stock prices rise during trading days after the lowest price day. The average after the day of the lowest price in CAC is –0.1388% for reference purposes. The averages in SSEC keep positive, i.e., 0.112% before the day of the lowest price, and 0.0489% after it.

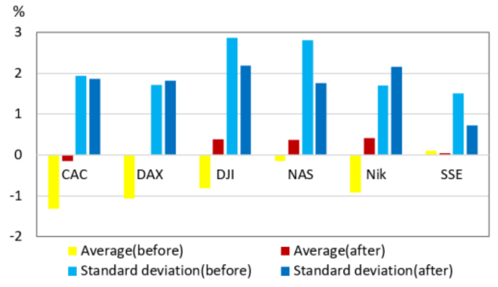


Figure 13. Average and standard deviation of candlestick body lengths

The standard deviations conspicuously decreased in DJI and NASDAQ, which are the same tendency as in the stock price changes.

C. Length of Upper and Lower Shadows

Figures 14 and 15 show the averages and standard deviations of the upper and lower shadows in the six markets. In all markets, the upper shadows have larger averages and standard deviations before the lowest price day than those of after the day.

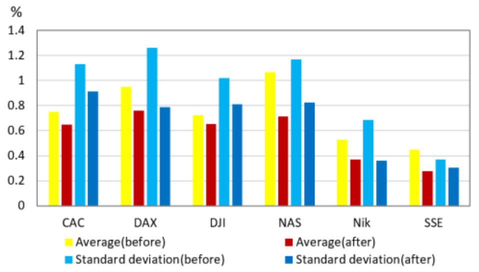


Figure 14. Average and standard deviation of length of upper shadows

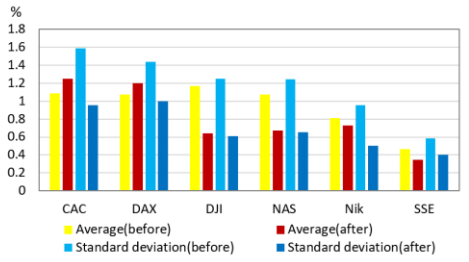


Figure 15. Average and standard deviation of length of lower shadows

The same tendency as the statistics of the upper shadows is observed in those of the lower shadows. However, the averages of the lower shadows increase by approximately 0.3% after the lowest price in CAC and DAX, while they decreased in the other markets.

D. Difference Between Stock Price and 5-day Average

Figure 16 shows the averages and standard deviations of "difference between a stock price and 5-day average." In the five markets excluding SSEC, the averages turn over from –3% or less to approximately 1%. SSEC have far less average change of 0.835% than the other markets, e.g., 3.70% of Nikkei.

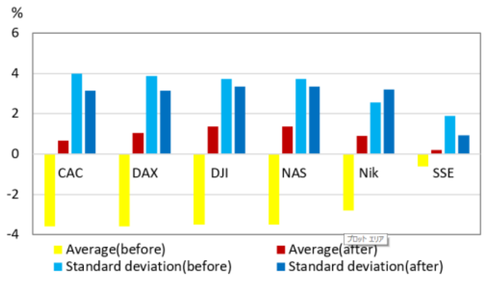


Figure 16. Average and standard deviation of "difference between stock price and 5-day average"

The five markets excluding Nikkei have slight reduction in the standard deviations roughly from 0.4% to 0.9%.

E. Difference Between Stock Price and 25-day Average

Figure 17 shows the averages and standard deviations of "difference between a stock price and 25-day average." The averages in CAC and DAX noticeably remain negative, i.e., –6.05% in CAC and –4.47% in DAX, after the lowest price day, which is deemed to reflect the delay in the recovery of markets.

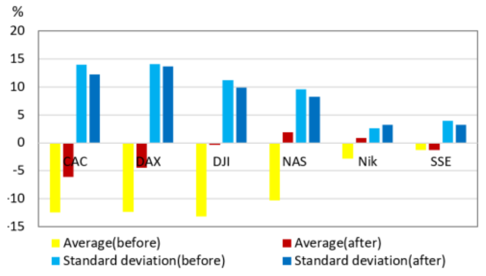


Figure 17. Average and standard deviation of "difference between stock price and 25-day average"

Meanwhile, NASDAQ achieves a positive average of 1.89%, indicating that the stock price steadily recovered compared to the other five markets.

VI. EXAMINING SIGNALS OF TREND REVERSAL

As mentioned in Section IV-D, the 25-day average of "difference between a stock price and 5-day moving average" starts to rise just after the day when the lowest price is recorded. This section discusses that this indicator can properly predict stock price trend reversals. Java programs using Swing packages are developed in order to perform the analysis efficiently.

Let CPr(n) be the closing price of a trading day n, and CPr(1) be the closing price of the latest trading day. CPr(1) represents the current stock price in the case of trading hours.

25-day average of "difference between a stock price and 5-day moving average" is defined by the following formula:

$$\text{CPAvg (n, 25, 5)} = \frac{1}{25}\sum_{j=n}^{j=n+24} \{\, CPr(j) - \frac{1}{5}\sum_{k=j}^{k=j+4} CPr(k) \,\} \qquad (2)$$

Analogously, 5-day average of the difference is defined by the following one:

$$\text{CPAvg (n, 5, 5)} = \frac{1}{5}\sum_{j=n}^{j=n+4} \{\, CPr(j) - \frac{1}{5}\sum_{k=j}^{k=j+4} CPr(k) \,\} \qquad (3)$$

Figure 18 illustrates the candlestick chart for 30 days before and after Mar. 18, 2020 in the DAX market. The magenta line at the bottom of Figure 18 shows CPAvg (n, 5, 5), and the blue line shows CPAvg (n, 25, 5).

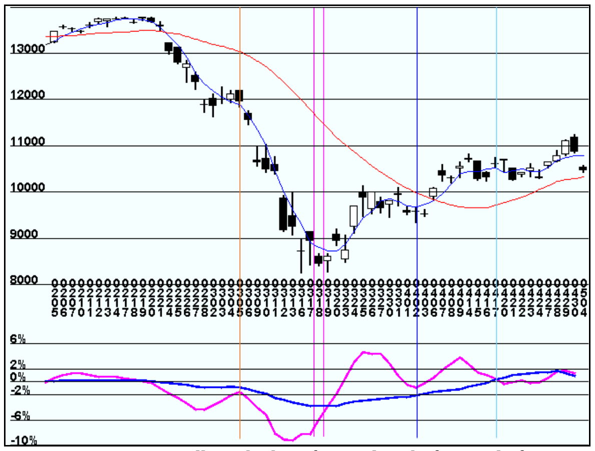


Figure 18. Candlestick chart for 30 days before and after Mar. 18, 2020 in DAX market

The proposed two average lines go across up on Mar. 19, i.e., the next day when the lowest price is recorded. The two lines do not crossover on Mar. 5, which predicts that stock prices will continue to fall. In other words, a short-term recovery from Feb. 28 to Mar. 5 is a "dead cat bounce," i.e., a temporary recovery of stock prices during a prolonged decline period.

The two lines also do not crossover during the decline from Mar. 27 to April 3, which suggests that Apr. 3 is a "buying on the dips" type of opportunity since the difference of the two lines increase on Apr. 3. The magenta line crosses down through the blue line on Apr. 17. However, given that the stock remains above the 25-day average with a margin, the two average lines suggests that we should not take any action for a down trend.

Similar results have been obtained in the other markets. Regarding recovery from the plunge caused by COVID-19, it is confirmed that the proposed two average lines forecast short-term trends properly.

VII. Conclusion and Future Work

This paper describes the results of analyses of stock price fluctuations in European, the U.S. and Asian markets with special focus on the effects of COVID-19 pandemic. In general, thanks to the timely implementation of monetary measures of each country, all stock indexes under study keep rising after the lowest price recorded in Mar. 2020.

As of Mar. 12, 2021, the NASDAQ Composite Index (U.S.) of post-corona recovered 62.36% higher price compared to the highest price of pre-corona, while CAC (France) decreased by –1.73%. Analyses of the average and standard deviation of the six attributes that characterize the candlestick chart reveal that CAC and DAX (Germany) are deemed to experience a larger impact on stock prices than the other markets.

We propose an indicator consisting of a pair of moving averages. The indicator is devised in the process of investigating how the difference between the stock price and 5-day average is related to the reversal of stock trends. The results of experiments using daily stock indexes under study show that the proposed indicator forecasts short-term trends properly with a short time lag, at least as far as the stock price plunge caused by COVID-19 is concerned.

We are planning comparative studies with well-known indicators including MACD (Moving Average Convergence Divergence) and ADX (Average Directional Index) [7] to inspect the effectiveness of the proposed indicator. The study may use daily historical stock data of various global stock markets and individual companies over different periods of time.

References

[1] A. Chudik, K. Mohaddes, M. H. Pesaran, M. Raissi, and A. Rebucci, "Economic consequences of Covid-19: a counterfactual multi-country analysis," Available from: https://voxeu.org/article/economic-consequences-covid-19-multi-country-analysis/ Oct. 2020.
[2] "Major World Market Indices," Available from: https://www.investing.com/indices/major-indices/ Mar. 2021.
[3] E. Dimson, P. Marsh, and M. Staunton, "Should you invest in emerging markets?" London Business School, Available from: https://www.london.edu/ think/emerging-markets/ Apr. 2019.
[4] Y. Udagawa, "Statistical Analysis of Stock Profits to Evaluate Performance of Markets," The Sixth International Conference on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2020) IARIA, pp. 14–21, Feb. 2020.
[5] C. C. Ngwakwe, "Effect of COVID-19 Pandemic on Global Stock Market Values: A Differential Analysis," Acta Universitatis Danubius. Œconomica, No 2, Vol 16, pp. 255-269, 2020.
[6] P. Verma, A. Dumka, A. Bhardwaj, A. Ashok, M. C. Kestwal, and P. Kumar, "A Statistical Analysis of Impact of COVID19 on the Global Economy and Stock Index Returns," Springer Nature journal Computer Science,2:27, pp. 13, Jan. 2021.
[7] "Technical Analysis," Cambridge Univ., Available from: http://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, pp. 1-179, Feb. 2011.
[8] S. Glen, "Statistics How To, T-Distribution/Student's T: Definition, Step by Step Articles," https://www. statisticshowto.com/probability-and-statistics/t-distribution/ 2020.

Distributed Search on a Large Amount of Log Data

Full text index in a Big Data architecture

Fabrice Mourlin, Charif Mahmoudi
Algorithmic, Complexity and Logic Laboratory
UPEC University
Creteil, France
Email : fabrice.mourlin@u-pec.fr, charif.mahmoudi@lacl.fr

Guy Lahlou Djiken
Applied Computer Science Laboratory
Douala University
Douala, Cameroon
Email : ldjiken@fs-univ-douala.cm

Abstract—**Log analysis is the basis for planning maintenance operations. It is used to predict software and hardware failures. Their costs can be high, ranging from activity blocking to data loss, even penalties for late results. In this context, to achieve an even better quality of service, we have built a distributed application for collecting and analyzing software logs. Key properties had to be taken into account such as the number of logs per hour, the variety of formats of these logs or the indexing of information known in these logs. Our approach responds to these properties by using a Big Data cluster and installing a custom indexing engine for software log analysis. Our results show a reduction in software failures and, therefore, better availability of software services under monitoring. This result leads to rethinking software maintenance and reviewing the sizing of our cluster according to the number of monitored applications correlated with the throughput of each one.**

Keywords-Software log; indexing; Big Data; streaming; planning maintenance.

I. INTRODUCTION

Logs are files hosted on the application servers to be monitored, which regularly record their activities such as access to resources, requests being processed, etc. The logs are used to retrieve information on abnormal behavior, alerts, errors and their scheduling, etc. They are full of information, including date of an event, the invoked Web address, the uniform resource location origin, the response code of the page (code 403, code 201, etc.), its payload, etc. The analysis of log files is the evaluation of a set of information recorded from one or more events that have occurred in an application environment. This practice is used to analyze user behavior and identify patterns of behavior, or identify and anticipate incidents. These same techniques are applied to ensure compliance of server behavior with the regulations in place, such as government applications.

Analyzing logs is a challenge and requires tedious work for the Software supervision teams because of the volume. Other features are crucial such as the diversity of types of logs, as well as the proprietary formats, elastic architectures, aggregation of time-stamped data, detection of behavior patterns, etc. Using log analysis software that leverages machine-learning algorithms dramatically reduces the workload on supervisory teams who can focus on value-added tasks. Such log analysis software allows monitoring, aggregation, indexing and analysis of all application and infrastructure log data. The tools such as the ELK (Elasticsearch, Logstash and Kibana) suite software, become a reference in the monitoring domain [1] because of their adaptability and polyvalence.

Log analysis tools provide better visibility into the health and availability of applications using dashboards. This allows software administrators to monitor critical events from a central location. Synthetic situations thus appear where an administrator is able to decide to anticipate a maintenance task in order to ensure continuity of service. For example, many services are written in Java where memory saturation problems cause the need to restart a Java Virtual Machine (JVM). In Big Data on the edge applications, the use of an energy source is often the constraint that leads not to the restart but to the migration of a service, from a network node on another, having still energy resources.

Over time, the use of specific indexing techniques has enriched log file analysis strategies. The structure of these input data is always formatted even though the formats vary. In addition, the use of a data schema provides additional typing which highlights the meaning of these lines of information. It is then useful to separate data storage from data indexing. The search for a pattern of behavior is more effective and prevention becomes better and more reactive.

Log analysis solutions incorporate additional data sources. Thus, machine learning and other analytical techniques push the boundaries of new use cases in application performance management, security intelligence, event management and behavior analysis.

The rest of this paper is organized as follows. Section II describes the works close to our domain. Section III provides a precise description of our use case. Section IV addresses the software architecture of our distributed platform. Section V goes into finer details on our streaming approach, which includes an indexing step. Section VI focuses about on our results and the impact on the maintenance task. The acknowledgement and conclusions close the article.

II. RELATED WORKS

The use of logging has been common practice in IT for many years. Its use for intervention prediction is more recent, but the interest of this approach has quickly become essential in companies and more particularly in any computer system offering services 24 hours a day. Publications have long been available in order to present the broad spectrum of log analysis techniques [2]. In the context of distributed computing systems, Qiang Fu has published some very useful results on behavior anomaly detection from logs [3]. W. Xu's work focuses on the structure of logs and the impact on the analysis strategy [4].

In the more specific context of analysis with prediction, Chinghway Lim's work is based on the use of individual message frequencies to characterize system behavior and the ability to incorporate domain-specific knowledge through user feedback [5]. Jakub Breier follows the same approach based on Apache Hadoop technique to enable processing of large data sets in a parallel way [6].

T. Li and Y. Jiang propose a platform to facilitate the data analytics for system event logs [7]. This work is an end-to-end solution that utilizes advanced data mining techniques to assist log analysts. They apply learning techniques to extract useful information from unstructured raw logs. The parsing technique contains an index management.

Steven Yen published recently a book on the topic of intelligent log analysis using Machine and deep learning [8]. He explains how deep learning implementation can improve the result quality when the data volume achieves a limit. He provides a comparison with a K mean model and two distinct implementations. This work shows that a global solution does not exist and some add-ons are crucial. For instance, the use of an indexing process of log messages could lead to a cost reduction at runtime.

In more constrained fields such as real time, log analysis systems must be able to detect an anomaly in a limited time. Biplob Debnath presents *LogLens* [9] that automates the process of anomaly detection from logs with minimal target system knowledge. *LogLens* presents an extensible index process based on new metrics (term frequency and boost factors). The use of temporal constraint also intervenes in the recognition of behaviour pattern. So, abnormal events are defined as visible in a time window while other events are not. This allows semi-automatic real-time device monitoring.

Aspects remain to be covered such as the use of cross logging in analysis and log indexing. The reason is the separation of storage and indexing. In the previous works, the storage is generally done by the use of relational databases while the indexing uses rather NoSQL (Not only Structured Query Language) databases where the notion of join cannot be easily implemented.

III. USE CASE DESCRIPTION

A. Historic

When doing software monitoring, the first thing we want to get is a reason for each failure, or even the root cause of the problem. The idea is then to automate the creation of an intervention request ticket and to follow up this maintenance operation until the update operation of the service concerned.

Many software programs exist for this need, such as Free Management of Computer Park (GLPI) [10], and new software monitoring needs are appearing in order to improve this incident management by anticipating maintenance operations. The idea is then to reduce the costs of maintenance task, which generally correspond to service interruptions. Even if service replication strategies make it possible to lessen the effects of a failure, it is preferable to anticipate this problem and to research before the event in order to prevent it.

B. Log information

At the heart of log analysis, there is the collection of events, such as the setup of a service, the attempt to connect to the system for example, a configuration request, or variations in CPU or storage, or the trace of an application event (receipt of an order, etc.).

A log entry contains information such as the date and time of the event, on which network node the event occurred, user identification, contextual information (configuration, security) or even the service at the origin of the event.

C. Nominal scenario

The description of our use case is based on our desire to monitor the activity of our information system. This includes several application servers and data management servers, interconnected by a software bus. It enables intelligent message routing between applications and provides a first level of fault tolerance in the event of a service failure.

Our servers provide log files, but also our applications deployed on the servers. Many formatted files are thus written in different directories. To perform a centralized log analysis, a preparatory step consists in moving the files to a dedicated machine. A second step consists in analyzing the data to keep the useful parts on the one hand and to index the key parts on the other hand. This pipeline continues with the use of a statistical model to predict the actions to be planned (Figure 1). Finally, the last step concerns the collection of metrics in order to evaluate the monitoring process.

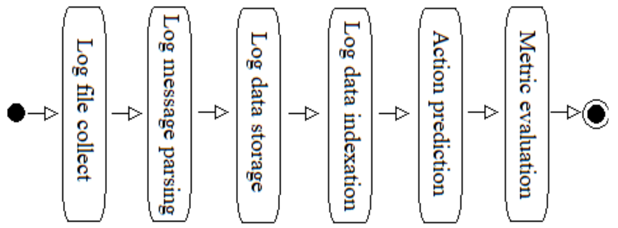


Figure 1. Log file lifecycle.

During our first prototypes, the volume of data processed exceeded 10 MB per hour and it became evident that such a sequential process could not meet our needs. The choice of a Big Data cluster for the processing of such volumes of text is legitimate, especially since this work relates to the monitoring of distributed systems.

IV. SOFTWARE ARCHITECTURE

Log analysis tasks often have strict due dates and data quality is a primary concern in software monitoring activities. This underlines the importance of finely managing the sequencing of tasks on the analysis platform.

The Hadoop ecosystem offers a set of software to process huge data sets. It was originally designed to run on clusters of

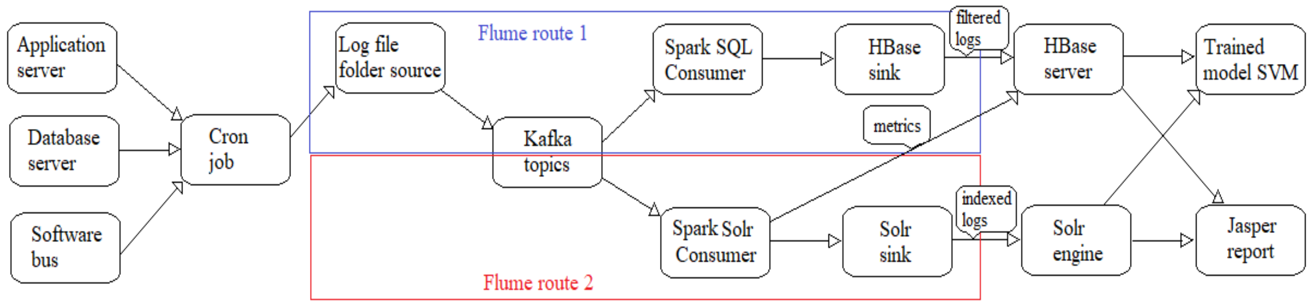


Figure 2. Big Data workflow for log analysis.

physical machines. Distributed analytical frameworks, for example MapReduce, evolve into resource managers that gradually transform Hadoop into a very versatile data operating system. With these frameworks, one can perform a wide range of data manipulation and analysis operations by connecting them to Hadoop Distributed File System (HDFS) as a document storage system [11].

Hadoop is highly scalable; it is easy to add a new service such as a search engine. As it includes the Zookeeper clustering tool, it is able to deploy on a set of nodes a search engine to manage a large volume of text-oriented data.

We have made the choice to use Apache Solr to index, search a large amount of business data, and provide relevant content based on a search query [12].

A. Big Data platform

A part of our work is based on Solr framework 8.1 and the integration with all other components in Hortonworks Data Platform Virtual Machine (HDP VM), such as Apache HBase, Apache Spark, Apache Kafka, in addition to some other open source tools. A part of our work relies on specific configurations of the tools; another part is the development of specific components for customizing the behavior of the Hadoop tools.

Our article outlines our approach and a simplified architecture for analyzing software-generated logs to detect functional-related issues. Our architecture is a batch analytics system analyzing Solr query logs.

The diagram from Figure 2 illustrates the high level of our software architecture.

We use shell scripts to collect log files destined for a remote directory (named "log file folder source"). With a common data ingestion path, the logs go from an Apache Flume source, then to a Kafka channel and are transmitted to a first Spark consumer (named "Spark SQL consumer"). Its essential task is to recognize and process the contents of the file and load them into an SQL table in memory, perform filter operations and put them in common format. Then, the route continues with a backup of these data in HBase tables. The role of this Flume route is to store structured information in a column-oriented database (the blue route in Figure 2).

In parallel, another route has the role of indexing the data from the logs (red route in Figure 2). From the same Kafka source, a second Spark consumer (named "Spark Solr consumer") takes care of data indexing while respecting the Solr schema. The index is updated for the query steps and then we use of a model for the prediction of maintenance tasks.

In this architecture, HBase is a highly reliable data store, supporting disaster recovery and cross-datacenter replication. Solr Cloud is the indexing and search engine. It is completely open and allows us to personalize text analyzes. It allows a close link with HBase database so the schemas used by both tools are designed in a closely related way.

The Jasper Report tool allows us to build a report from data automatically and regularly. Suitable cross tables help to give priorities to software maintenance tasks.

B. Configuration

1) Via operating system

Several elements of this architecture support ad hoc configuration. We have defined specific configuration scripts for routing log files to the "log file folder" directory, source Flume. We use entries in cron tables to ensure regular data collection.

2) Via event streaming-tools.

We have described two Flume routes within our Big Data cluster. Flume configurations correspond to the creation of routing agents so that information reaches the programs that use them.

The Flume and Kafka tools are both event-streaming tools. While their roles are comparable, the developments in these two projects are very different and there are now more Kafka connectors. Thus, the popularity of Apache Kafka is currently higher than that of Flume. We have kept software routes with Flume for event routing, but we define Kafka topics to ensure decorrelation between components. This makes it possible to simplify the management of components, among other things for updates. In addition, the Kafka API allows more controls on the management of messages associated with a topic; for example time management. We have added rules to ensure that a received message is processed within an hour. In that case, we raise an alert and the data is saved in the local file system.

3) Via persistent storage.

We wrote the script for creating tables structured in families of columns to keep the information from the log files. The column families are logical and physical groups of columns. The columns in one family are stored separately from the columns in another family. Because we have data that are not often queried, we assign that data to a separate column family.

Because the column families are stored in separate HFiles, we keep the number of column families as small as possible. We also want to reduce the number of column families to reduce the frequency of mem-store flushes, and the frequency of compactions. Moreover, by using the smallest number of column families possible, we improve the load time and reduce disk consumption.

4) Via indexing engine.

Apache Solr is an open source search engine and Solr index can be considered as an equivalent of a SQL table. A standalone instance maintains several indexes. However, on our Big Data cluster, the Solr installation is also distributed. In that context, we have four shards with a replication rate equals to three. This allows us to distribute operations by reducing blockages due to frequent indexing. We have configured not only the schema, but also the data handlers (*schema.xml* and *solrconfig.xml* files).

Our schema defines the structure of the documents that are indexed into Solr. This means the set of fields that they contain. We also define the datatype of those fields. It configures also how field types are processed during indexing and querying. This allows us to introduce our own parsing strategy via class programming.

C. Component architecture

1) Based on Spark framework.

To implement this architecture, we have developed several components using the Spark framework version 2.4.7. These components are at the heart of Flume routes, so their sequencing is based on the Spark-streaming module. In other words, when log data are available, the scheduler creates micro batches to process these data during a fixed duration window. In order to keep the results of the processing, the components save their results in a HBase database installed on the Hadoop cluster [15].

We have two consumers of the data associated with the Kafka topic. Spark SQL consumer uses the Spark SQL module to store data in an HBase database whose schema is structured in family of columns. The labels of these families of columns are involved in the data schema of the second Spark consumer.

HBase is a database distributed on the nodes of our Hadoop cluster, which allows having a persistence system where the data are highly available because the replicated rate on separate nodes is set to three.

2) Based on Spring Data.

Spark Solr consumer uses the Spring Data and SolrJ library to index the data read from the Kafka topic. It splits the data next to the Solr schema where the description of each type includes a "*docValue*" attribute, which is the name of the HBase column family. For each Solr type, our configuration provides a given analyzer. We have developed some of the analyzers in order to keep richer data than simple raw data from log files. Finally, the semantic additions that we add in our analysis are essential for the evaluation of Solr query. Likewise, we store the calculated metrics in HBase for control.

SolrCloud is deployed on the cluster through the same Zookeeper agents. Thus, the index persistence system is also replicated. We therefore separate the concepts of backup and search via two distinct components. This reduces the blockages related to frequent updates of our HBase database [14].

3) Based on SolrJ library.

At the beginning of our Solr design, we have built our schema based on our data types. Some of them were already defined, but some others are new. In addition, we have implemented new data classes for the new field types. For example, we used *RankFieldType* as a type of some fields in our schema. Then, it becomes a sub class of *FieldType* in our Solr plugin.

We have redesigned Solr filters so that they can be used in our previous setups. Our objective was to standardize the values present in the logs coming from different servers. Indeed, the messages provide information of the form <attribute, value> where the values certainly have units. However, the logs do not always provide the same units for the same attribute calculation. The analysis phase is the place to impose a measurement system in order to be able to compare the results later.

The development pattern proposed by SolrJ is simple because it proposes abstract classes like *TokenFilter* and *TokenFilterFactory* then to build inherited classes. Then we have to build a plugin for Solr and drop it in the technical directory agreed in the installation of the tool [13].

4) Based on Spark-MLlib.

In Artificial Intelligence, Support Vector Machine (SVM) models are a set of supervised learning techniques designed to solve discrimination and regression problems. SVMs have been applied to a large number of fields (bioinformatics, information research, computer vision, finance, etc.) [16]. SVM models are classifiers, which are based on two key ideas, which allow to deal with nonlinear discrimination problems, and to reformulate the ranking problem as a quadratic optimization problem. In our project, SVMs can be used to decide to which class of problem a recognized sample belongs. The weight of these classes if linked to the Solr metrics on these names. This amounts to predicting the value of a variable, which corresponds to an anomaly.

All filtered log entries are potentially useful input data if it is possible that there are correlations between informational messages, warnings, and errors. Sometimes the correlation is strong and therefore critical to maximizing the learning rate. We have built a specific component based on Spark MLlib It supports binary classification with linear SVM. Its linear SVMs algorithm outputs an SVM model [18].

We applied prior processing to the data from our HBase tables before building our decision modeling. These processes are grouped together in a pipeline, which leads to the creation of the SVM model with the configuration of its hyper-parameters such as *weightCol*. Part of the configuration of these parameters comes from metrics calculated by our indexing engine (Figure 2). Once created and tested, the model goes into action to participate in the prediction of incidents. We use a new version of the SVM model builder based on distributed data augmented. This comes from an article written Nguyen, Le and Phung [19].

5) Based on Jasper Report library.

This reporting library allows us to build weekly graphical reports on indexing activity. This information is a help to check the suitability of the SVM model, which supports prediction requests following pattern recognition. The representations are documents in pdf format; we did not automate the impact of this data extraction on the use of our decision-making model.

V. Big Data Streaming

We use Apache Kafka as queue system for our logs. Then we use spark streaming library to read from Kafka topic and process logs on the fly. Spark Streaming is a real-time processing tool that runs on top of the Spark engine. The scheduler exploits all the computation resources of our cluster. Each node runs several executors, which run tasks and keeps data in memory or disk storage across them.

In our program, the Spark context sends all the tasks for the executors to run.

A. *Filtered log strategy*

1) *Asynchronous reading.*

Our component called Spark SQL Consumer contains a Kafka receiver class, which runs an executor as a long-running task. Each receiver is responsible for exactly one input discretized stream (called *DStream*). In the context of the first Flume route, this stream connects the Spark streaming to the external Kafka data source for reading input log data.

Because the log data rate is high, our component reads from Kafka in parallel. Kafka stores the data logs in topics, with each topic consisting of a configurable number of partitions. The number of partitions of a topic is an important key for performance considerations as this number is an upper bound on the consumer parallelism. If a topic has N partitions, then our component can only consume this topic with a maximum of N threads in parallel. In our experiment, the Kafka partition number is set to four.

2) *Normalized form.*

Since log data are collected from a variety of sources, data sets often use different naming conventions for similar informational elements. The Spark SQL Consumer component aims to apply name conventions and a common structure. The ability to correlate the data from different sources is a crucial aspect of log analysis. Using normalization to assign the same terminology to similar aspects can help reduce confusion and error during analysis [17]. This case occurs when log messages contain values with different units or distinct scales. The log files are grouped under topics. We apply transformations depending on the topic the data come from. The filtered logs are cleaned and reorganized and then are ready for an export into an HBase instance.

3) *Stuctured data storage.*

Next step, the Spark SQL Consumer component inserts the cleaned log data into memory data frames backed to a schema. We have defined a mapping between HBase and Spark tables, called Table Catalog. There are two main difficulties of this catalog.

a) The row key definition implies the creation of a specific key generator in our component.

b) The mapping between table column in Spark and the column family and column qualifier in HBase needs a declarative name convention.

The HBase sink exploits the parallelism on the set of Region servers, which are under control of the HBase master. The HBase sink treats both *Put* operation and *Delete* operation in a similar way, and both actions are performed in the executors. The driver Spark generates tasks per region. The tasks are sent to the preferred executors collocated with the region server, and are performed in parallel in the executors to achieve better data locality and concurrency. By the end of an exportation, a timed window of log data is stored into HBase tables.

B. *Index construction and query*

1) *The index pipeline*

The strategy of the Spark Solr Consumer component deals with the ingestion of the log data into Apache Solr for search and query. The pipeline is built with Apache Spark and Apache Spark Solr connector (Figure 3). Spark framework is used for distributed in memory compute, transform and ingest to build the pipeline.

The Apache HBase role is the log storage and the Apache Solr role is the log indexing. Both are configured in cloud mode Multiple Solr servers are easily scaled up by increasing server nodes. The Apache Solr collection, which plays the role of SQL table, is configured with shards. The definition of shard is based on the number of partitions and the replicas rate for fault tolerance ability.

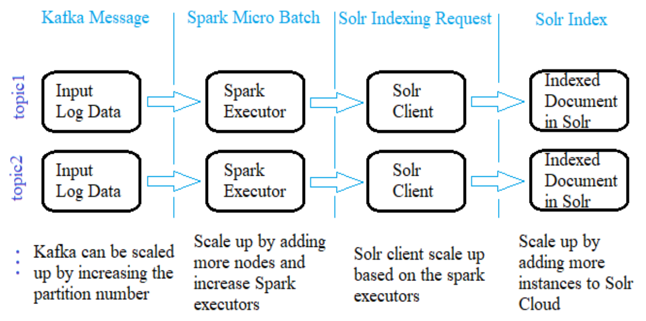

Kafka Message
Spark Micro Batch
Solr Indexing Request
Solr Index
topic1
topic2
Input Log Data
Spark Executor
Solr Client
Indexed Document in Solr
Input Log Data
Spark Executor
Solr Client
Indexed Document in Solr
Kafka can be scaled up by increasing the partition number
Scale up by adding more nodes and increase Spark executors
Solr client scale up based on the spark executors
Scale up by adding more instances to Solr Cloud

Figure 3. Overall high-level architecture of the index pipeline.

The Spark executors run a task, which transforms and enriches each log message (format detection). Then, the Solr client takes the control and sends a REST request to Solr Cloud Engine. Finally, depending on the Solr leader, a shard is updated.

2) *The query process.*

We also use Solr Cloud as a data source Spark when we create our ML model. We send requests from spark ML classes and read results from Solr (with the use of Solr Resilient Distributed Dataset (SolrRDD class). The pre statement of the requests is different from the analysis of the log document. Their configuration follows another analysis process.

With Spark SQL, we expose the results as SQL tables in the Spark session. These data frames are the base of our ML model construction. The metrics called TF (Term Factor) and

IDF (Inverse Document Frequency) are key features for the ML model. We have also used boost factor for customizing the weight of part of log message.

VI. Results and Actions

We have several kinds of results. A part is about our architecture and the capacity to treat log messages over time. Another part is about the classification of log messages. The concepts behind SVM algorithm are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. In our working context, the margin between log patterns is a suitable discriminant.

A. *Data features*

1) *Architecture measurement*

For our tests, we used previously saved log files from 20 days of application server and database server operations. We were interested in the performance of the two Spark consumers, the Spark SQL Consumer and the Spark Solr Consumer.

For Spark SQL Consumer, the volume of data to analyze is 81.7 M rows in HBase. To exploit this data, we used a cluster of eight nodes on which we deployed Spark and HBase. The duration of the tests varies between 24 minutes and 2 hours and 1 minute.

For Spark Solr Consumer, the volume of data indexed is 87.2M rows indexed in about an hour. The number of documents indexed per second is 28k.

We only installed Solr on four nodes with four shards and a replication rate of three. We have seen improved results by increasing the number of Spark partitions (*RangePartitioner*). At runtime for our data set based on a unique log format, the cost of Spark SQL consumer decreases when the partitioning of dataset increases, an illustrated in Figure 4. The X-axis represents the partition number as an integer and the Y-axis represents the time consumed (minute unit). We have to oversize the partitions and the gains are much less interesting.

2) *Model measurement*

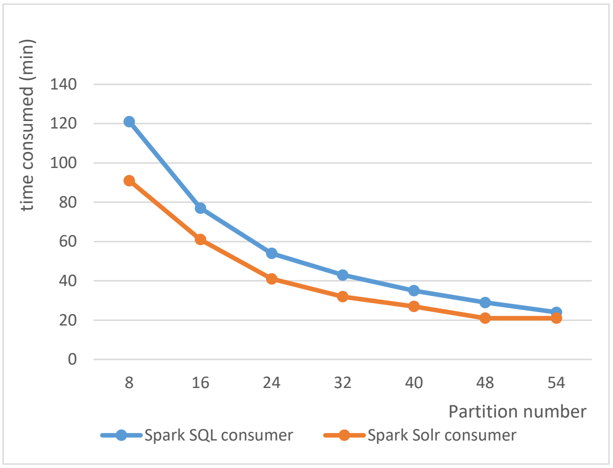


Figure 4. Spark consumer runtime versus number of partitions

SVM offers very high accuracy compared to other classifiers such as logistic regression, and trees. There are several modes of assessment. The first is technical; it is obtained thanks to the framework used for the development (Spark MLlib). The second is more empirical because it relates to the use of this model and the anomaly detection rate on a known dataset.

The analytical expressions of the features precision and recall of retrieved log messages that are relevant to the find are indicated below.

Precision is the fraction of retrieved log messages that are relevant to the find:

$$precision = \frac{|\{relevant\ \log messages\} \cap \{retrieved\ log\ messages\}|}{|\{retrieved\ \log messages\}|}$$

Recall is the fraction of log messages that are relevant to the query that are successfully retrieved:

$$recall = \frac{|\{relevant\ \log messages\} \cap \{retrieved\ log\ messages\}|}{|\{relevant\ \log messages\}|}$$

$$F_\beta = (1+\beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall}$$

In Table 1, we have four classes and for each class we compute three numbers: true positive (tp), false positive (fp) and false negative (fn). For instance, for the third class, we note these numbers tp3, fp3 and fn3. From these values, we compute precision by label, recall by label and F-score by label.

TABLE I. SVM MODEL MEASURES

Class number	Metrics		
	Precision by label	***Recall by label***	***F1 score by label***
0,000000	0.884615	0.920000	0.901961
1,000000	1.000000	1.000000	1.000000
2,000000	0.846154	0.785714	0.814815
3,000000	0.854462	0.7914858	0.842529

Our prediction models are similar to a multiclass classification. We have several possible anomaly classes or labels, and the concept of label-based metrics is useful in our case. Precision is the measure of accuracy on all labels. This is the number of times a class of anomaly has been correctly predicted (true positives) normalized by the number of data points. Label precision takes into account only one class and measures the number of times a specific label has been predicted correctly normalized by the number of times that label appears in the output. The last observations are:

- Weighted precision = 0.917402
- Weighted recall = 0.918033
- Weighted F1 score = 0.917318
- Weighted false positive rate = 0.043919

Our results for four classes are within acceptable ranges of values for the use of the model to be accepted.

The test empirical phase on the SVM model was not extensive enough to be conclusive; however, our results suggest that increasing the number of log patterns deteriorates the performance. In addition, we defined a finite set of log patterns for a targeted anomaly detection approach.

B. Reporting

We have created a custom data source to connect to Apache Solr, therefore, we are able to retrieve data and provide them back in following the *JRDataSource* interface of Jasper Report. With this access point, we have extracted metrics about the document cache and Query result cache. Both give an overview of the Solr activities and is meaningful for the analysts.

We have deployed the CData JDBC Driver on Jasper Reports to provide real-time HBase data access from reports. We have found that running the underlying query and getting the data to our report takes the most time. When we generate many pages per report, there is overhead to send that to the browser.

For the reporting phase, we have developed two report templates based on the use of a JDBC adapter. With system requests, we collect data about the last events (Get, Put, Scan, and Delete). From these HBase view, we have designed the report templates with cross tables. For the storage phase, we compute and display the number of Put events per timed window or grouped over a period.

We periodically updated the data across report runs and export the PDF files to the output repository where a web server manages them.

VII. Conclusion and future work

We have presented our approach on log analysis and maintenance task prediction. We showed how an index engine is crucial for a suitable query engine. We have developed specific plugins for cutomizing the field types of our documents, but also for filtering the information from the log message.

Because indexing and storage are the two sides of our study, we have separated the storage into a Hadoop database. We have stressed the key role of our Spark components, one per data source. The partition management is the key concept for improving the performance of the Spark SQL component. The data storage into data frames during the micro batches is particularly suitable for the management of flows originating from Kafka files. We observed that our approach supported a large volume of logs.

From the filtered logs, we presented the construction of our SVM model based on work from the Center for Pattern Recognition and Data Analytics, Deakin University, (Australia). We were thus able to classify the recognized log patterns into classes of anomalies. This means that we can identify the associated maintenance operations. Finally, to measure the impact of our distributed analysis system, we wanted to automatically build reports based on templates and highlight indexing and storage activity.

Our study also shows the limits that we want to push back, such as the management of log patterns. The use of an AI model is not the guarantee of an optimal result. We want to make more use of indexing metrics to give more weight to some information in the analyzed logs. We are, therefore, thinking of improving the classification model of log data.

A first perspective will be to improve the indexing process based on a custom schema. We think that the use of DisMax query parser could be more suitable in log requests where messages are simple structured sentences. The similarity detection makes DisMax the appropriate query parser for short structured messages.

The log format has a deep impact on the Solr schema definition and about the anomaly detection. We are going to evolve our approach. In the future, we want to extract dynamically the log format instead of the use of a static definition.

We think also about malicious messages, which can perturb the indexing process and introduce bad requests in our prediction step. The challenge needs to manage a set of malicious patterns and the quarantine of some message sequences.

References

[1] J. Andersson and U. Schwickerath, “Anomaly Detection in the Elasticsearch Service,” CERN Openlab Summer Student Report, pp. 1-18, 2019.

[2] T. S. Collett, “A review of well-log analysis techniques used to assess gas-hydrate-bearing reservoirs,” Natural Gas Hydrates : Occurrence, Distribution, and Detection, Geophysical Monograph, The AmericanGeophysical Union, vol. 124, pp. 189-210, 2001.

[3] Q. Fu, J. G. Lou, Y. Wang and J. Li, “Execution anomaly detection in distributed systems through unstructured log analysis,” 2009 ninth IEEE international conference on data mining IEEE, pp. 149-158, 2009.

[4] A. Oliner, A. Ganapathi and W. Xu, “Advances and challenges in log analysis,” Communications of the ACM, vol. 55, no. 2, pp. 55-61, 2012.

[5] C. Lim, N. Singh and S. Yajnik, “A log mining approach to failure analysis of enterprise telephony systems,” 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), IEEE, pp. 398-403, 2008.

[6] J. Breier and J. Branišová, “Anomaly detection from log files using data mining techniques,” In Information Science and Applications, Springer, Berlin, Heidelberg, pp. 449-457, 2015.

[7] Li Tao et al., “FLAP : An end-to-end event log analysis platform for system management,” Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1547-1556, 2017.

[8] S. Yen and M. Moh, “Intelligent log analysis using machine and deep learning,” Machine Learning and Cognitive Science Applications in Cyber Security, IGI Global, pp. 154-189, 2019.

[9] B. Debnath et al., “Loglens : A real-time log analysis system,” 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), IEEE, pp. 1052-1062, 2018.

[10] M. Picquenot and P. Thébault, “GLPI (Free Management of Computer Park) : Installation and configuration of a park management solution and support center,” ENI Editions, 2016.

[11] K. Shvachko, H. Kuang, S. Radia and R. Chansler, “The hadoop distributed file system,” 2010 IEEE 26th symposium

on mass storage systems and technologies (MSST), IEEE, pp. 1-10, 2010.

[12] D. Smiley, E. Pugh, K. Parisa and M. Mitchell, “Apache Solr enterprise search server,” Packt Publishing Ltd, 2005.

[13] J. Kumar, “Apache Solr search patterns,” Packt Publishing Ltd, 2015.

[14] K. Koitzsch, “Advanced Search Techniques with Hadoop, Lucene, and Solr,” Pro Hadoop Data Analytics, Apress, Berkeley, CA, pp. 91-136, 2017.

[15] R. C. Maheshwar and D. Haritha, “Survey on high performance analytics of bigdata with apache spark,” 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), IEEE, pp. 721-725, 2016.

[16] M. F. Ghalwash, D. Ramljak and Z. Obradović, “Early classification of multivariate time series using a hybrid HMM/SVM model,” 2012 IEEE International Conference on Bioinformatics and Biomedicine, IEEE, pp. 1-6, 2012.

[17] F. E. N. G. Changyong et al., “Log-transformation and its implications for data analysis,” Shanghai archives of psychiatry, vol. 26, no. 2, 2014.

[18] M. Assefi, E. Behravesh, G. Liu and A. P. Tafti, “Big data machine learning using apache spark MLlib,” 2017 IEEE International Conference on Big Data (Big Data), IEEE, pp. 3492-3498. 2017.

[19] T. D. Nguyen, V. Nguyen, T. Le and D. Phung, “Distributed data augmented support vector machine on spark,” 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, pp. 498-503, 2016.

Open Research Data in Institutional Repositories

Approach and Practice of Polish Universities

Anna Wałek
University Library
Gdańsk University of Technology
Gdańsk, Poland
e-mail: anna.walek@pg.edu.pl

Dorota Siwecka
Institute of Information and Library Science
University of Wrocław
Wrocław, Poland
e-mail: dorota.siwecka@uwr.edu.pl

***Abstract* — Research data is a type of data that is gaining importance in terms of both scientific value and commercialization potential. Research data is one of the essential parts of the entire data universe. It can be big data, small data, raw or processed data. In order to improve functionality, searchability and indexation, the data is associated with metadata and linking data, allowing it to be linked with other resources, databases, and reference data. Complex calculations are carried out on large amounts of research data generated as part of scientific experiments. An increasing number of universities prepare and implement an Open Access policy, following the latest international guidelines. To meet the conditions of these arrangements, scientific institutions in Poland implement projects to ensure open access to their employees' publications and Open Research Data. This paper presents the results of an analysis of four university repositories in Poland. The study covers, inter alia, the type of published data, database scope, institutions participating in the project implementation, and the possibility of re-use of open data (data sharing format, metadata format, applied ontologies and data schemas, information on linked data, legal licenses).**

Keywords-Open Research Data; Linked Data; institutional repository; data repository.

I. INTRODUCTION

Research data comprises all data produced as a result of scientific research, regardless of whether we are dealing with sciences, natural sciences, social sciences, or humanities. Data can be the results of analyses, measurements, surveys, historical sources, laboratory logs, images or samples, among others. It can take both digital and analog forms. The significance and quality of research data are widely discussed topics in the scientific community and, nowadays, researchers have become increasingly open in sharing and disseminating research findings. The amount of data is continuously growing, and opening research data is not only a good practice, but an obligation. Leading research funding agencies, including the European Commission, implement policies of sharing scientific data associated with publicly financed projects and place increasing emphasis on proper management of the research results they fund. Many guidelines and recommendations in this area have been published (including guidelines from the European Commission, research funding institutions, international organizations and associations). In the big data era, society, business, and research need quality data to evolve. One of the best ways to ensure data quality is managing research data under FAIR principles. FAIR data is data which meets principles of Findability, Accessibility, Interoperability and Reusability [1] (hence the acronym).

Horizon 2020 introduced a policy that obliges grantees to publish research results in the Open Access (OA) model. A pilot of Open Research Data has also been introduced. The announced changes in Horizon Europe will include the need to ensure OA by depositing work in a repository immediately after publication, without any embargo period, leaving copyright with the author, and using open licenses, such as Creative Commons licenses. In the field of research data, the principle of ‘as open as possible, as closed as necessary’ will be maintained, according to which, by default, data should be made available in an open manner. Data can only remain closed in justified situations, with considerable attention being paid to research data management.

On 4 April 2019, the European Parliament adopted a Directive on open data and re-use of public sector information [2]. The document was updated in June 2019. EU member states were given two years to adapt their laws to the new requirements. This means, among other things, that regulations for the sharing of research data by scientific institutions must be in place by 2021. The adoption of the Directive will also affect the Polish legal order, as its implementation will require amendments to the Act of 25th February 2016 on the re-use of public sector information [3]. It is worth paying attention to the fact that easy integration, linking, and re-use of data across silos is enabled by, among others, Linked Open Data (LOD) technology [4].

The EU Directive in points 27 and 28 speaks explicitly about the motives and ways to make research data open:

‘The volume of research data generated is growing exponentially and has potential for re-use beyond the scientific community. In order to be able to address mounting societal challenges efficiently and in a holistic manner, it has become crucial and urgent to be able to access, blend and re-use data from different sources, as well as across sectors and disciplines. Research data includes statistics, results of experiments, measurements, observations resulting from fieldwork, survey results, interview recordings and images. It also includes meta-data, specifications, and other digital objects. (…) Beside open access, commendable efforts are being made to ensure that data management planning becomes a standard scientific practice and to support the dissemination of research data that is findable, accessible, interoperable, and reusable (FAIR principle). (28) For the reasons explained above, it is appropriate to set an obligation on Member States to adopt open access policies with respect to publicly funded research data and ensure that such policies are implemented by all research performing organizations and research funding organizations. (…) certain obligations stemming from this Directive should be extended to research data resulting from scientific research activities subsidized by public funding or co-funded by public and private-sector entities. (…) However, in this context, concerns in relation to privacy, protection of personal data, confidentiality, national security, legitimate commercial interests, such as trade secrets, and to intellectual property rights of third parties should be duly taken into account, according to the principle "as open as possible, as closed as necessary"' [2].

The second section of this paper describes the legal and organizational situation of research data in Poland (based on European Union regulations). It describes the method of calculating the number of repositories in Poland and indicates the reason for selecting those repositories described in the study. The third section describes the method adopted in the study and the individual elements analyzed. This section is divided into the following parts: providers and cooperation; types of provided information; publications and Research Data licenses; repositories and FAIR principles; research Data format; Linked data and semantization; access points; metadata license; used schemas and ontologies. The last section of the paper describes the preliminary conclusions of the research and, above all, the indicated areas that require in-depth analysis and further research. As this publication is an introduction to the research and indicates problems that require analysis, topics have been identified that have not been researched yet, but will be analyzed in the future.

II. Open Research Data at Polish Universities

Based on the Directive mentioned above, the Ministry of Digitalisation of the Republic of Poland prepared in 2020 a draft law on open data and re-use of public sector information [5], which will be introduced in 2021. As a consequence of introducing the Act, guidelines and recommendations for universities will also be presented.

The National Science Centre – NSC (the Polish national funder) also supports the idea of making research data accessible. From 2019, grant applications submitted in NSC competitions must be accompanied by a Data Management Plan (DMP), which specifies, among other things, what research data will be used, produced and made available within the project, as well as how they will be stored.

The academic year 2019/2020 in Poland began with 133 public universities, including 19 universities, 16 technical universities and 13 universities of natural science, economics and pedagogy, 9 medical universities, and approx. 240 non-public universities [6]. According to the statistical data of the Polish Central Statistical Office for 2019, 93,100 academic teachers were employed in various research and teaching positions [7].

Considering the large number of universities in Poland, the number of repositories where their employees can share data is low. The Directory of Open Access Repositories (Open DOAR) database shows 119 open repositories in Poland [8]. After analyzing the list of repositories, it can be concluded that most of the platforms registered there are digital libraries, which are not typical repositories. However, they often contain, in addition to cultural heritage objects, scientific publications and sometimes research data. The content of these databases would require additional separate research. However, considering only the research data repositories registered in the Re3Data.org database, we can see only 8 instances [9]. This is a much better source of information than the Open DOAR, because the repository must meet specific criteria to be registered in that database. Registered repositories are: Copernicus' Heliograph, Open Forest Data, Kujawsko-Pomorska Digital Library (as the only one of the digital libraries), CLARIN-PL, MOST Wiedzy Open Research Data Catalog (Bridge of Data), GEOMIND (no longer available since 2018), RepOD, Polish Platform of Medical Research.

III. The Basis of the Analysis

Repositories chosen for this study had to meet two main criteria: they should provide access to Open Research Data and implement LOD technology to increase accessibility, improve quality and increase the possibility of reusing science resources according to international recommendations. According to a survey concerning Polish LOD projects conducted in January 2021, only four institutional repositories functioning in Polish universities meet the LOD criterion [10]: AZON 2.0 (the Atlas of Open Science Resources) [11], the Bridge of Data (MOST Wiedzy Open Research Data Catalog – Gdańsk University of Technology) [12], RUJ (Repository of Jagiellonian University, Cracow) [13] and PPM (Polish Platform of Medical Research)[14]. Those examples show a diverse approach to providing access to information about publications, research, and Open Data conducted by Polish researchers. Information on the projects comes from the data posted on the official websites of the projects and from questionnaires sent to suppliers between January 21 and 27, 2021.

A. *Providers and cooperation*

The selected examples show that, in the Polish landscape, there are different approaches to creating repositories. We can list here two main types: 1) projects implemented by a single university and 2) projects created cooperatively by more than one institution. In the first group, we can indicate a) projects designed for the university's employees only, e.g. RUJ; b) projects implemented by a single university but open to all research communities in the country, e.g., Bridge of Data, which the Gdańsk University of Technology introduced, but where any interested researcher can create a profile and share information about research conducted and publications. In the second group, we can identify a) projects implemented by "homogeneous" universities, e.g. PPM, which is a result of the cooperation of 8 medical institutions (7 universities with Wrocław Medical University as a leader and 1 Institute of Occupational Health); and b) projects implemented by different kinds of institutions, like AZON 2.0 created by Wrocław University of Science and Technology (leader), the Wrocław University of Environmental and Life Science, Wroclaw Medical University and Systems Research Institutes of Polish Academy of Science. Further analysis of remaining Polish repositories should lead to the creation of a typology that includes them.

B. Types of provided information

Analyzed projects show that these are not only typical institutional repositories. In addition to data on employees' publications and Ph.D. theses, they also provide information about researchers' profiles, conferences, research infrastructure, research projects, research data, and sometimes about inventions (detailed information on the types of provided data are summarized in Table I).

TABLE I. TYPES OF PROVIDED DATA

	AZON 2.0	**Bridge of Data**	**PPM**	**RUJ**
Researchers' profile	+	+	+	–
Publications' metadata	+	+	+	+
Full text of publications	+	+	+*	+
Research data	+	+	+	+
Conferences	conference recordings, news, etc.	conference ranking according to the list of the Polish Minister of Science and Higher Education; conference proceedings are listed in researchers' profiles	+	conference proceedings and conferences organized by Jagiellonian University
Projects/ Grants	–	+	+	+
Institutional publishing output	**?**	+	+ (journals)	Publications of Jagiellonian Library; series publications of Jagiellonian University
Ph.D. thesis	+	+	+	+
Inventions	**?**	+	+ (patents)	–
Research infrastructure	+	+	+	–

* Only those with Open Access and deposited in the repository (ca. 17 % of all publications records in the database).

C. Publications and Research Data licenses

Because all of the analyzed repositories provide access to publications and research data, they must define the licenses under which they are available online. In every case, information about the type of license is given in addition to the metadata. Moreover, the information about the license usually links to the full text of the license on the Internet. The exception is the PPM platform, where clicking the license info directs us to the list with results of all publications deposited in the PPM with the same license. The most common are Creative Commons licenses. If the full text of the publication is available outside of the repository, an appriate link is also given. Most of the suppliers also provide special tutorials for researchers about licenses.

D. Repositories and FAIR principles

As mentioned in the introduction, one of the best ways to ensure data quality is managing research data under FAIR principles. The analyzed repositories in their tutorials for researchers write about FAIR principles [15; 16; 17]. The only repository that doesn't provide such information on the project's website is AZON 2.0. The actual check of whether a given repository's resources meet the FAIR rules is the so-called FAIRification, which consists of the technical review of data using dedicated IT tools [18]. This is an issue that requires a separate study.

E. Research Data format

Research Data are mostly available in many formats, due to the multidisciplinary nature of the repositories. The most common are CSV and PDF. But, for example, the set of formats of the AZON 2.0 repository also includes DOC/X, PPT/X, TIFF, JPG, PNG, GIF, PSD, WRP, STL, OBJ, PTS, PTX, TXT, JSON, MP4, MOV, STR, VTT, STL, AVI, MP3, MD, and many technical formats for datasets. Depositaries are committed to sharing files in open formats and/or providing alternative formats to closed formats.

F. Linked data and semantization

Three of the four analyzed repositories provide information about LOD or semantization of their collection on their websites (AZON 2.0 on the main page [11] after choosing "How to further explore resources"; Bridge of Data

[19]; PPM [20]). Information about LOD implementation in the RUJ project was received in the questionnaire.

All of the repositories implemented linking data within the projects. In AZON 2.0, metadata are connected also to external datasets such as Geonames, Wikidata, plWordnet, General Multilingual Environmental Thesaurus (GEMET), Medical Subject Headings (MeSH), AgroVOC, International Plant Names Index (IPNI), The Plant List, World Flora Online (WFO), Ośrodek Przetwarzania Informacji (OPI), ORCID, Polska Bibliografia Naukowa (PBN), ResearcherID, Scopus. Information about linking data to external datasets in the PPM was not provided.

G. *Access points*

All of the projects provide API for data re-use, and in most cases, this requires filling an access form to receive an access code or token. As to API's data format, there are many variations: the most common are JSON, JSON-LD, RDF, XML, and Turtle. The exception is the PPM platform, which lists all available formats of data and protocols on its website [21].

H. *Metadata license*

Almost none of the projects implemented by Polish institutions provide information about metadata licenses. The Bridge of Data uses the Creative Commons CC-BY license for metadata, but that information is not visible on the project's website. At RUJ, the suppliers didn't apply any license for metadata because, in their opinion, "it was not necessary".

Of course, the issue of a metadata license is strongly related to the applicable law in the country, and this should be the basis for further research.

I. *Used schemas and ontologies*

Creating an extensive database that provides access to diverse content is a big challenge. It requires, for instance, the development of a data entry schema. The analyzed projects mainly created schemas based on other schemas and ontologies already available on the Web. The most common are: Schema.org, datacite, Friend of a Friend (FOAF), Bibliographic Ontology (BIBO), Dublin Core (DC), Bibliographic Framework (BIBFRAME), Metadata Authority Description Schema (MADS), Gemeinsame Normdatei (GND), Simple Knowledge Organization System (SKOS), Functional Requirements for Bibliographic Records (FRBR).

IV. CONCLUSIONS

Research Data Management is a topic that has been discussed for many years and is gaining in importance. The basic premise for appropriate data management is the proper and efficient use of public funds. Considering the changes in scientific communication and the guidelines of research funding agencies, as well as legislative changes, Polish universities will have to create data repositories or use existing platforms for sharing research data. Both the repositories analyzed and the remaining repositories must be carefully analyzed according to specific qualitative criteria, defined at a later stage of the research. The conducted study showed that, among similar initiatives, there are significant differences in the functionality and standards used.

Further research should also cover other issues related to the quality of Polish repositories. These include indexing the repository content in dataset indexing databases, such as Google Dataset Search, DataCite, and Data Citation Index (Web of Science). Another criterion against which repositories will be assessed will be their certification (e.g., Core Trust Seal) and presence on the lists of trusted repositories published by scientific journals and publishers (e.g., Nature Scientific Data Recommended Repositories [22]). Further research should also show in which other data repositories researchers from Poland share their data. Preliminary analyses indicate that these may be repositories with an international scope, such as Zenodo or Figshare, domain repositories and repositories of scientific journal publishers.

REFERENCES

[1] M. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". Sci Data 3, 160018 (2016), doi: 10.1038/sdata.2016.18.

[2] European Union, "Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information (recast)", Official Journal of the European Union, L 172/56, 26.6.2019. [Online]. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32019L1024&from=EN, 2021.02.24.

[3] The Act of February 25, 2016 on the re-use of public sector information, Dziennik Ustaw 2016 poz. 352. Available from: http://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20160000352/U/D20160352Lj.pdf, 2021.02.24.

[4] D. Reynolds, „Linked (Open) Data" in: B.C. Dalsgaard, Data reuse and how metadata can stimulate reuse. A workshop held on Monday 5th of December 2011 at the 7th IDCC conference in Bristol. [Online]. Available from: https://libereurope.eu/wp-content/uploads/2020/11/WGSC_20111205.pdf.

[5] Ministerstwo Cyfryzacji, „Draft act on open data and re-use of public sector information, RCL, 24.08.2020.[Online]. Available from: https://legislacja.rcl.gov.pl/projekt/12337400, 2021.02.24.

[6] S. Zdziebłowski, „On Tuesday, the beginning of the 2019/2020 academic year", Nauka w Polsce, 01.10.2019. [Online].Available from: https://naukawpolsce.pap.pl/aktualnosci/news%2C78787%2Cwe-wtorek-poczatek-roku-akademickiego-20192020.html, 2021.02.24.

[7] Statistic Poland, Higher education and its finances in 2019, Warszawa, Gdańsk 2020. Available from: https://stat.gov.pl/obszary-

tematyczne/edukacja/edukacja/szkolnictwo-wyzsze-i-jego-finanse-w-2019-roku,2,16.html. 2021.02.24.

[8] OpenDOAR, "Browse by country and region: Poland". [Online]. Available from: https://v2.sherpa.ac.uk/view/repository_by_country/Poland.html, 2021.02.24.

[9] Re3data.org. [Online]. Available from: https://www.re3data.org/search?query=&countries[]=POL, 2021.02.24.

[10] The results of the survey were presented during study day *The semantic web and cultural heritage from data convergence to knowledge crossing*, 3 February 2021, GERiiCO & Universitté de Lille [online]. D. Siwecka, "Linked Open Data in the Polish (librarian) landscape".

[11] AZON 2.0, https://zasobynauki.pl/, 2021.02.24.

[12] Bridge of Data, https://mostwiedzy.pl/en/, 2021.02.24.

[13] Repository of Jagiellonian University, https://ruj.uj.edu.pl/xmlui/, 2021.02.24.

[14] Polish Platform of Medical Research, https://ppm.edu.pl/index.seam?lang=en&cid=44836, 2021.02.24.

[15] PPM, Research data in a nutshell. Guide. [Online]. Available from: https://biblioteka.wum.edu.pl/sites/biblioteka.wum.edu.pl/files/poradnik_dane_badawcze_ppm_v7-1.pdf, 2021.02.24.

[16] M. Szufita-Żurawska, Data Management Plan, 2020. [Online]. Available from: https://cdn.mostwiedzy.pl/c0/a3/7d/50/0_202002141036301505451_FME/plan-zarzadzania-danymi-2020.pdf, 2021.02.24.

[17] RUJ, Research Data. [Online]. Available form: https://cawp.uj.edu.pl/documents/102715934/141758336/NCN_zarzadzanie-danymi.pdf/266e134e-418e-4ca0-842f-1ebb47227235, 2021.02.24.

[18] H. Koers, P. Herterich, R. Hooft, M. Gruenpeter, and T. Aalto, "M2.10 Report on basic framework on FAIRness of services" 30th November 2020, Zenodo, doi: 10.5281/zenodo.4292598.

[19] Bridge of Data, 5-star Open Data. [Online]. Available form: https://mostwiedzy.pl/pl/open-data?pageId=1_50711040454643_SPA, 2021.02.24.

[20] PPM: 5* Open Data. [Online]. Available from: https://ppm.edu.pl/fiveStarData.seam, 2021.02.24.

[21] PPM, API. [Online]. Available from: https://ppm.edu.pl/api.seam, 2021.02.24.

[22] "Recommended data repositories", Scientific Data. [Online]. Available from: https://www.nature.com/sdata/policies/repositories, 2021.02.24.

On Compressing Time-Evolving Networks

Sudhindra Gopal Krishna[1], Sridhar Radhakrishnan[1], Michael Nelson[1], Amlan Chatterjee[2] & Chandra Shekaran[3]
[1]School of Computer Science, University of Oklahoma, Norman, OK, {sudhi, sridhar, Michael.A.Nelson-1}@ou.edu
[2]Department of Computer Science, California State University Dominguez Hills, Carson, CA, {achatterjee}@csudh.edu
[3]Department of Computer Science, Loyola University at Chicago, Chicago, IL, {chandra}@cs.luc.edu

***Abstract*—A graph G = (V, E) is an ordered tuple where V is a non-empty set of elements called vertices (nodes), and E is a set of an unordered pair of elements called links (edges), and a time-evolving graph is a change in the states of the edges over time. Extremely large graphs are such graphs that do not fit into the main memory. One way to address the issue is to compress the data for storage. The challenge with compressing data is to allow for queries on the compressed data itself at the time of computation without incurring overhead storage costs. Our previous work on Compressed Binary Trees (CBT), which was shown to be efficient both in time and space, compresses each node and its neighbors (termed as row-by-row compression). This paper first provides encoding to store the arrays in the Compressed Sparse Row (CSR) data structure and extends the encoding to store time-evolving graphs in the form of CSR. The encoding also enables accessing a node without decompressing the entire structure, meaning the data structure is queryable. We have performed an extensive evaluation of our structures on synthetic and real networks. Our evaluations include time/space comparison with both time-evolving compressed binary tree and ck^d data structures, including the querying times.**

***Index Terms*—Compression, Network Science, Time-Evolving, Queries, Graphs**

I. Introduction

Graphs can be used to represent real-world data from a wide variety of domains. The relationships among the data are captured by the characteristics of the graph. For most real-world data, the relationships change over time. This results in the graph evolving from its initial state to the current one. A graph $G = (V, E)$ is represented by a set of vertices V and a set of edges E. For real-world data, the graph G evolves with time and can be statically represented using a series of graphs $G_t = (V_t, E_t)$ where the time t indicates an instant which is spread over a certain interval.

Therefore, a time-evolving graph can be defined as a graph that changes or evolves over time. Consider, as an example, pages on Wikipedia. Each page evolves over time with the addition and deletion of content. The current state of the page is the one that contains the content of the page at present. However, all the edit information is also saved for the page. Using the edit information, the state of the page at previous instants of time can be checked. Hence, having such information preserves the integrity of the page while being open to editing. Now, the information for the Wikipedia pages with the time-evolving data can be represented and stored as graphs. Storing such information is useful for performing various kinds of analyses. For example, one might want to know what changes have occurred to a document from beginning to the current state. Another related query can be, what changes occurred to a document within a certain interval of time; this would be specifically interesting to study if the document represents some current socio-economic or political events. Also relevant would be to know the number of changes that occur to the documents from any point in time to another.

Time-evolving graphs represent data from different domains such as social networks and communication networks. A variety of analyses can be performed on such data based on the availability of the same over time. For example, such graphs can be used to perform descriptive, diagnostic, predictive and prescriptive analytics, among others. Hence, to execute such operations on time-evolving graphs, the data has to be stored. Generally, graphs are stored in either of the three different representations: adjacency matrix, adjacency list and edge list. In adjacency matrix, the graph is represented as a matrix of n^2 elements, where $|V| = n$; edges are represented using 1's and lack of it as 0's. For an adjacency list representation, for each node $v \in V$, a list of adjacent nodes is stored. Finally, for an edge list representation, all edges are stored in a pair format (v_i, v_j) where an edge exists from v_i to v_j; the number of entries in the edge list is $|E|$. However, most real-world graphs are very large in size. Hence, the memory requirements for storing the data are significant. For example, if we consider a time-evolving graph, like Wiki-edits and Yahoo Netflow, the sizes of the data in the edge list format are 5.7 GB and 19 GB, respectively. With such sizes, the data might not fit on the main memory for analysis. Therefore, to store the data for time-evolving graphs and perform computation on the same, the data is required to be compressed.

In this paper, we propose techniques to perform compression on time-evolving graphs. There are normally two methods for compressing the adjacency information for a graph: one is to consider the entire graph together, the other considering portions of the graph at a time.

For our methods, we exploit row-by-row compression for node data separately. Specifically, we utilize two combinations of data structures to store the time and adjacency information. In the first one, the time tree provides information regarding the instants of time the graph evolved; for each node, there is an additional tree to store the adjacency information for each instant in the time tree which is CBT. Rather than storing the entire adjacency information for the nodes, a differential approach is leveraged to reduce the memory requirements. In the second approach, for each time frame the edges are stored in an unsigned bit array and similar to the CSR-CSR compression, the later information is stored in a differential approach to save the memory which is CSR. Our contributions also show that depending on the characteristics of the graph being compressed, using a combination of techniques rather than a single method for the data structures yields better compression.

The rest of the paper is organized as follows. In Section II, we discuss existing techniques for storing and managing time-evolving graph data. We propose our methods for compressing and storing time-evolving graphs in Section III. In Section IV, we examine the various algorithms that provide efficient compression for time-evolving graphs. We report the empirical results and analysis in Section V and the conclusion in Section VI.

II. Related Work

A time-evolving graph can be represented as a sequence of static graphs (snapshots), with each of the snapshots representing the graph at a particular point in time. Since a snapshot can be represented as a 2D matrix, a time-evolving graph can thus be represented as a 3D matrix, also known as a *presence matrix* [1].

In 2009, Chierichetti et al. [2] modified the web compression method developed by Boldi and Vigna's WebGraph [3] called Backlinks Compression (BLC). The compression is based on the social network's property of reciprocity. The compression technique makes use of intrinsic ordering heuristics based on shingles, which improves on the WebGraph format.

Nelson et al. [4] in 2017 introduced a compressed data structure as an indexed array of Compressed Binary Tree (CBT). The data structure eliminates the necessity of intermediate structure to create the compressed binary tree. The data structure also makes use of row-by-row compression which enables faster access to the edge existence, neighbor query and the streaming operation.

In 1976, Compressed Sparse Row (CSR) was first documented by Snay [5], and is one of the most common data structures used for representing a graph. Compressed sparse row is also a row-by-row compression which involves two arrays for the compression of each node. All the information is efficiently packed in the array for the quick traversal of the data structure. The first array shows the degree of each node, and the following array shows the edge incidence to each node. Here, the degree of a node v is the number of edges incident to v, and is denoted as $d(v)$.

In 2016, Caro et al. [6] developed $ck^d - trees$. They define a contact as a quadruplet (u, v, t_i, t_j) and then compress the 4D binary matrix corresponding to the time-evolving graph defined by a set of these contacts. It is done by representing the 4D matrix as a $k^d tree$ and then distinguishing white nodes as those without any contacts, black nodes as ones that only contain contacts, and gray nodes as those that contain only one contact. This work was preceded by Brisaboa et al's $k^2 - trees$ [7] in 2014.

G^* database [8] is a distributed index that solves the space issue of the presence matrix by only storing new versions of an arc when its state changes, i.e. as a log of changes. This is done by storing versions of the vertices as adjacency lists and maintaining pointers to each time frame. If an arc changes in the next frame, a new adjacency list is created for that vertex's arc and a pointer is added to the new frame.

Caro et al. [9] proposed a compressed adjacency log structure based on the *log of events* strategy called EveLogs. It consists of two separated lists per vertex - one for the time frames, and another for representing the arcs related to the event. The time frames are compressed using gap encoding, and the arc list is compressed with a statistical model. Caro et al. [6] shows that query times suffer with scanning the log sequentially.

Ren et al. [10], developed the FVF (Find-Verify-Fix) framework which includes a copy+log compression that also supports shortest-paths and closeness centrality queries. More preliminary work is done in [11] [12], which describe three different methods to index time-evolving graphs based on the *copy+log* strategy.

Two *log of events* strategies, CAS and CET, are proposed in [9] to address the problem of slow query times when processing a log. CAS orders the sequence by vertex and adds a Wavelet Tree [13] data structure to allow for logarithmic time queries. CET orders the sequence by time, and the authors develop a modified Wavelet Tree called Interleaved Wavelet Tree to also allow logarithmic time queries.

In 2014, Brisaboa et al. [14] adapted Compressed Suffix Arrays (CSA) as in [9] for use in temporal graphs (TGCSA) by treating the input sequence as the list of contacts. They use an alphabet consisting of the source/destination vertices and the starting/ending times.

III. Representation of time-evolving graphs

In this paper, we represent the time-evolving graphs based on the neighbors of each node over time. This requires two data structures, the first one stores the time information, which can be represented as a time array, and the second one stores the neighbors of the node at each of the instants given in the time array. Now, the time array can be thought of as a stream of 0 and 1 bits: a 0 indicating no change from the previous instant of time, and a 1 indicating changes in the neighborhood of the node from the previous time stamp. Since the time instants taken into account are finite and relatively small, the size of the time array could be in the range of 10,000 for an example graph. For the same graph, the size of the node array, which would contain the neighborhood information for the specific node over 10,000 time instants, could be 1,000,000,000 elements.

The density of the time array and the node array can be different. Given the time array and the node array are bit arrays, these can be compressed for storage using different methods. In this paper, we propose compressing the binary arrays using one of the two techniques: a) CBT, and b) CSR.

Now, depending on the size, and the sparsity of the graphs, the sizes of the compressed structure vary.

The term bit-packing works on the number of bits required to represent each number. For a given array of unsigned integers, represent each number of the array in bits and store them in as an unsigned bit array. For example, consider an array of unsigned integers 1, 3, 5, 10, 16, and 26. The maximum number of bits required to store each integer is the ceiling of the log of the maximum element in the array. An array location can store a maximum of 32 or 64 bits depending upon the system, and all the bits are stored in a little-endian format. Table I shows all the above numbers stores in a single unsigned bit array location.

TABLE I: THE SINGLE BIT ARRAY NEEDED TO REPRESENT ALL THE INTEGERS.

unsigned int	1	3	5	10	16	26
unsigned bit	00001 00011 00101 01010 10000 11010					

If the entire number does not fit into an array location, a part of the number can be stored in one array location and the rest of the number can be stored in the starting bits of the following memory location. This ensures that there are no unoccupied bits in the bit array. Table II shows the bits carry over for a 32-bit integer.

TABLE II: THE CARRY-OVER BIT ARRAY NEEDED TO REPRESENT ALL THE INTEGERS.

unsigned int	1	3	5	10	16	26	30
unsiged bit	00001 00011 00101 01010 10000 11010 11						
	110 00000 00000 00000 00000 00000 0000						

Algorithm 1 explains the working of the bitPacking method. The algorithm takes in an unsigned integer array, the number of elements in the array, and the number of bits required to represent each number in the array. The variable arraySize indicates the number of array locations needed to store the unsigned bits of all the numbers in the array. Line 7 indicates the start of converting the unsigned integer to the bit representation; m indicates the number of unsigned integers that an array location can accommodate. Lines 9 through 11 convert the unsigned integers to unsigned bits and store it in the array location k. The remaining bits in the array location k are filled by the most significant bits of the next number, as shown in line 13 through line 20.

IV. COMBINING CBT AND CSR

For every input of the time-evolving graphs G, the input is divided as an ordered triplet (u, v, T_τ), where u and v are the nodes which form an edge at time T_τ. If the edge appears again later at another time frame $T_{\tau+i}$, the edge is considered to be deactivated at the time frame. For the CSR-CSR and CBT-CSR combinations, we are assuming the datasets are sorted with respect to the time frames and then sorted by node numbers for each time frame. For the CSR-CBT and CBT-CBT combinations, the datasets are first sorted with respect to source node and then sorted with respect to time frame for each source nodes.

Algorithm 1: Algorithm for bitPacking

```
Input: An unsigned integer array (uArray), number of
       elements (numElements), and the number of
       bits (numBits) required to convert
Output: The converted bit array.
1  begin
2      totalBits = 64;
3      balance = 0;
4      arraySize = $\frac{numberofElements}{numberofBits} * totalBits$;
5      Initialize an unsigned bit array (bArray);
6      k = 0;
7      for $index = 0$ to $(index < numOfElements)$ do
8          m = $availBits/numBits$;
9          for $i = 0$ to $(i < m$ && $(index + i)$
             $< numElements)$ do
10             bArray[k] $| = (uArray[index + i] <<$
                 $((i * numBits) + balance)))$;
11             i++;
12         index $+ = m$;
13         if $index < numElements$ then
14             remBits = $availBits \% numBits$;
15             if $((remBits > 0) \&\& (m <$
                 $numElements))$ then
16                 bArray[k] $| = (uArray[index] <<$
                     $(totalBits - remBits))$;
17                 availBits
                     $= totalBits - (numBits - remBits)$;
18                 k++;
19                 bArray[k] $| =$
                     $(uArray[index] >> remBits)$;
20                 balance $= (numBits - remBits)$;
21                 index++;
22             else
23                 k++;
24                 balance = 0;
25                 availBits = totalBits;
26     return $bArray$;
```

A. CSR-CSR

This is a novel combination. In this algorithm, we compress the graph based on the edges appearing in each time frame. For the time frame T_0, we compress the graph row-by-row in a conventional compressed sparse row format using Algorithm 2. For each row, the first array consists of each node's degree in the time frame T_0, and the second array consists of the upper triangular destination edge id v.

For time frame T_1 *to* T_τ, storing the edge in the conventional CSR format will cost more space as not all the edges from the original start edges change. To overcome this, we encoded all the edges using Algorithm 3. For every time frame T_i, CSR is made up of three arrays, where the first array is the unique source node u involved in the graph's changes, the

Algorithm 2: Algorithm for Compressed Sparse Row at time T_0

```
Input: Unsigned integer array of contacted nodes (v),
       unsigned integer array of degree of each nodes
       (u), maximum degree of the graph ($\delta$(G))
Output: Unsigned bit array
begin
    logD = log2(maxDegree) + 1;
    logN = log2(numNodes) + 1;
    degreeBitArray = bitPacking(degreeArray,
      numNodes, logD);
    csrBitArray = bitPacking(vArray, numEdges,
      logN);
    finalBitArray.append(degreeBitArray);
    finalBitArray.append(csrBitArray);
    return finalBitArray;
```

Algorithm 3: Algorithm for Compressed Sparse Row from time T_1 to T_τ

```
Input: Unsigned integer array of contacted nodes (v),
       unsigned integer array of degree of each nodes
       (u), unsigned integer array of contact nodes (u)
Output: Unsigned bit array
begin
    for $time\ t = 1\ to\ t = \tau - 1$ do
        logU = log2(maxContactNode) + 1;
        logD = log2(maxDegree) + 1;
        logN = log2(maxContactedNode) + 1;
        bitUArray = bitPacking(uArray, uArray.size(),
          logU);
        bitDegreeArray = bitPacking(degreeArray,
          uArray.size(), logD);
        bitVArray = bitPacking(uArray, vArray.size(),
          logN);
        finalBitArray.append(bitUArray);
        finalBitArray.append(bitDegreeArray);
        finalBitArray.append(bitVArray);
    return finalBitArray;
```

second array consists of the degree of the source nodes, and the third array consists of the destination nodes v.

This yields the time complexity of $O(\tau * (n * log(\delta) + mlog(n)))$, where τ is the number of time frames, n is the number of nodes, m is the number of contacts, and δ is the maximum degree of the graph.

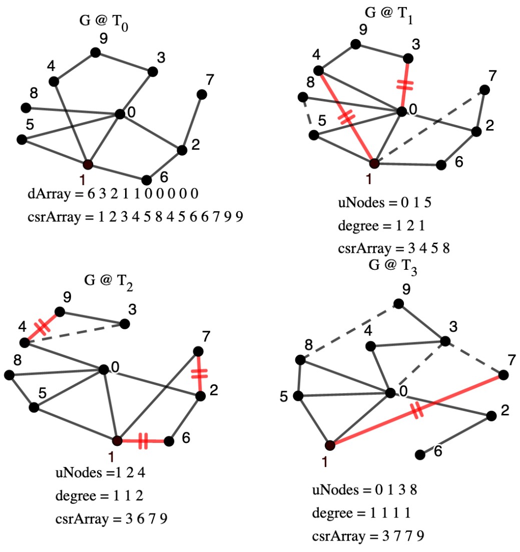
Figure 1 shows the overall structure of the CSR-CSR compression. The dotted line in the graph at each time frame represents the edge being added and the double-crossed red line represents the edge being deleted at the time frame.

B. CBT-CSR

This is a novel combination. In this combination, we compress the first time frame T_0 using the existing CBT algorithm

Algorithm 4: Compressed Binary Tree

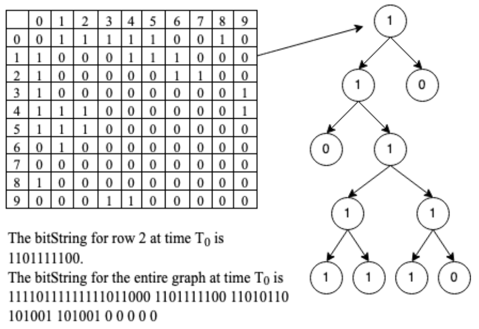
```
Input: An edge's time array, T, of size $\tau$
Output: The compressed edge as a bitstring
begin
    BitString s;
    Node node = root;
    Boolean flip = False;
    Visitor vtr = PreOrderTraversal(node);
    for $i = 0$ to $\tau - 1$ do
        while !node.isLeaf() do
            if (!flip $\wedge$ node.spans(v)) $\vee$
            (flip $\wedge$ !node.spans(v)) then
                s.AppendBit(1);
            else
                s.AppendBit(0);
                vtr.Ignore(node);
            node = vtr.VisitNext(node);
        s.AppendBit($T[i]$)
        if $T[i]$ != flip then
            flip = !flip;
    //fill rest of tree with 0s
    return s;
```



CSR as unsinged char: 01101100010010001000
100001001100001010100010010101110000100110000101010 000100101 100010100
110000101010001 100010001 100100010 0010111000010101000 0000100011000001
1111 1100111011101001

Fig. 1: The structure of the time-evolving CSR

4 [4] as shown in the Figure 2. The input to the algorithm [4] are the edges associated with time frame T_0. For the time frames T_1 *to* T_τ we follow the method used in the CSR-CSR compression, as shown in Figure 1. This yields the time-complexity of $O(\tau * (d(v)log(\delta) + mlog(n)))$.



	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	1	1	0	0	1	0
1	1	0	0	0	1	1	1	0	0	0
2	1	0	0	0	0	0	1	1	0	0
3	1	0	0	0	0	0	0	0	0	1
4	1	1	1	0	0	0	0	0	0	1
5	1	1	1	0	0	0	0	0	0	0
6	0	1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	0	0
9	0	0	0	1	1	0	0	0	0	0

The bitString for row 2 at time T_0 is 1101111100.
The bitString for the entire graph at time T_0 is 11110111111111011000 1101111100 11010110 101001 101001 0 0 0 0 0

Fig. 2: The structure of the time-evolving CBT at time frame T_1

C. CBT-CBT

In this combination, we followed the algorithm mentioned in [15]. The input to this algorithm is first sorted based on the source node u, and for each source node, the data is sorted based on the time T_i. With this type of input comes two arrays for each node, the first being the time frames at which the node has an edge, followed by all the destination node v for each time T_i. Therefore, each node's total number of trees will be the number of time frames for the node u and one tree to represent all the time frames. This yields the time-complexity of $O(\tau * (d(\tau)log(\delta) + mlog(n)))$, where $d(\tau)$ represents the degree of each time-frame.

D. CSR-CBT

This is a novel combination. In this combination, we follow the same input type as CBT-CBT combination, but here we first compress the time array using bit-packing algorithm 1, and to compress the destination edges for each time frame of the source node u, we follow the CBT algorithm 4 [4]. This yields the time-complexity of $O(\tau * (d(v)log(\delta) + mlog(n)))$, where $d(v)$ denotes the degree of each node v.

V. Experimental Results

For our analyses, we have the results of compression size and the time taken to compress from all the combinations using the datasets mentioned in Table III, which is compared with the results of $ck^d - tree$ [6] and CBT [15] as shown in Table IV.

TABLE III: The graph datasets, including the type, number of nodes, number of edges, time frames, and the size of the input file both in .txt and gzip format.

Graphs	Type	Nodes	Edges	Contacts	Time Frames
CommNet	Interval	10000	15940743	19061571	10001
PowerLaw	Interval	1000000	31979927	32280816	1001
Flickr-Days	Incremental	2585570	33140018	33140018	135
Wiki-edits	Point	21504191	561754369	266769613	134075025
Yahoo Netflow	Interval	32904819	122075170	1123508740	58735

If the edges in the graph exist from time $[t_i, t_j)$, then such graphs are called interval graphs. If the edges in the graph appear once and live till the last time-frame, such graphs are referred to as incremental graphs. If the edges appear for a single time-frame, then such graphs are referred to as point graphs.

TABLE IV: The compression size and the time taken to compress each dataset. Please note that ck^d does not allow streaming operations

Graphs	.txt	.txt.gz	CSR-CSR		CSR-CBT	
CommNet	271.6 M	51 M	34 M	10.25 s	16 M	56.19 s
PowerLaw	546.9 M	132 M	80 M	18.94 s	80 M	162.23 s
Flickr-Days	860 M	130 M	107 M	34.16 s	91 M	208.39 s
Wiki-edits	5.7 G	1.8 G	2.0 G	1158 s	1.8 G	2042.88 s
Yahoo Netflow	19 G	4.9 G	4.3 G	1372 s	3.2 G	1770.71 s

Graphs	.txt.gz	CBT-CSR		CBT-CBT		CkD	
CommNet	51 M	16 M	55.80 s	15.9 M	65.5 s	30 M	119 s
PowerLaw	132 M	70 M	141.21 s	73.80 M	149 s	128 M	254 s
Flickr-Days	130 M	82 M	120.015 s	73.8 M	179 s	89 M	235 s
Wiki-edits	1.8 G	2.0 G	1126.85 s	1.4 G	3081s	1.2 G	2059 s
Yahoo Netflow	4.9 G	4.2 G	1874.95 s	2.99 G	3506 s	2.5 G	5471 s

The CommNet graph and the PowerLaw graphs are synthetically generated datasets based on the data avaiable from [6]. CommNet graph simulates short communication between random vertices. PowerLaw graph simulates the powerlaw degree distribution in the graph.

Flickr dataset is an incremental graph [16]. This graph represents the user interaction derived from the Flickr social network for a span of days from 11-02-2006 to 05-18-2007.

Wiki-edits is a bipartite point graph [17]. This graph shows when the user edited an article in Wikipedia. The time is stored in seconds since the creation of Wikipedia.

The last dataset for our analysis is a Yahoo-Netflow graph [18]. This graph is an interval graph, where the data are the interaction between the users and the Yahoo server. The time is measured in seconds and the first occurrence of the data was on 04-29-2008.

All the experiments were run on an Intel(R) Xeon(R) CPU E5520 @ 2.27GHz (16 Cores) with 64 GB of RAM, and the programs are written in GNU C/C++.

The source code for this work is available to download at [19].

A. Compression Results

Table IV shows the compression results for the dataset in Table III over all the combinations and $ck^d - trees$. We have used compression size, time taken to compress each dataset, and querying times as metrics to evaluate.

Table IV clearly shows the space and time tradeoff between the compression results of CSR-CSR and CBT-CBT. While the CBT-CBT consumes 30% less space compared to CSR-CSR, CSR-CSR consumes around 40% less time for the Yahoo-Netflow graph [18]. The combination of CBT-CSR and CSR-CBT has shown similar or better results with datasets with fewer or no change in both compression size and time is taken to compress.

B. Querying Results

For 1000 randomly chosen vertices,

- Neighbor Query: What are the neighbors that exist at time T_i.
- Neighbor Query: What are all the possible neighbors of a node between time interval of T_i through T_j.

TABLE V: THE AVERAGE TIME NEEDED TO QUERY A NODE TO FETCH ALL THE NEIGHBORS AT A GIVEN TIME T_i AND THE TIME INTERVAL T_i THROUGH T_j

Graphs	CSR_Ti (ms)	CBT_Ti (ms)	CKD_Ti (ms)	CSR_Ti_Tj (ms)	CBT_Ti_Tj (ms)	CKD_Ti_Tj (ms)
CommNet	0.78 ± 0.005	1.33 ± 1.44	48.89 ± 11.56	0.93 ± 0.049	1.43 ± 0.47	64.46 ± 0.43
PowerLaw	2.07 ± 0.006	2.99 ± 0.55	374.23 ± 50.72	2.10 ± 0.012	5.70 ± 1.05	374.64 ± 50.66
Flickr-Days	1.31 ± 0.02	11.29 ± 8.52	35.34 ± 10.39	2.08 ± 0.31	38.49 ± 8.34	45.22 ± 5.78
Wiki-edits	0.40 ± 0.007	1.24 ± 1.911	3.0 ± 3.0	0.403 ± 0.001	1.42 ± 2.12	4.39 ± 0.72
Yahoo Netflow	2.19 ± 0.45	43.13 ± 0.263	231.9 ± 82.1	1.51 ± 0.06	51.21 ± 4.67	254 ± 92.06

TABLE VI: THE AVERAGE TIME NEEDED TO QUERY AN EDGE EXISTS BETWEEN TWO NODES AT A GIVEN TIME T_i AND THE TIME INTERVAL T_i THROUGH T_j

Graphs	CSR_Ti (ms)	CBT_Ti (ms)	CKD_Ti (ms)	CSR_Ti_Tj (ms)	CBT_Ti_Tj (ms)	CKD_Ti_Tj (ms)
CommNet	0.78 ± 0.001	0.39 ± 0.66	49.6 ± 3.4	0.82 ± 0.002	0.39 ± 0.55	49.7 ± 0.24
PowerLaw	2.06 ± 0.011	0.64 ± 0.12	216.0 ± 5.3	2.08 ± 0.06	1.6 ± 0.03	226.13 ± 14.38
Flickr-Days	1.31 ± 0.013	4.23 ± 2.81	35.2 ± 1.2	2.21 ± 0.2	5.44 ± 10.11	37.2 ± 2.3
Wiki-edits	0.39 ± 0.08	1.15 ± 0.18	2.62 ± 1.7	0.39 ± 0.008	1.15 ± 0.19	2.98 ± 0.25
Yahoo Netflow	1.38 ± 0.014	30.32 ± 2.36	211.8 ± 89.0	1.52 ± 0.042	31.2 ± 5.37	212.32 ± 71

- Edge Existence: Does an edge exist at time T_i.
- Edge Existence: Does an edge exist between the time interval of T_i through T_j.

From Tables V and VI, we can infer that the CSR takes the least amount of time to query a random node both for edge existence and to fetch all the neighbors.

VI. CONCLUSION

Valuable insights can be gained from the analysis of time-evolving graphs. However, due to the large size of such graphs, the memory requirements are significant and it is a challenge for computing using the main memory. Therefore, in this paper we propose compression techniques for time-evolving graphs.

Our techniques show that a significant reduction in memory requirements can be achieved by exploiting the topological characteristics of graphs, specifically the adjacency information for each node also known as row-by-row compression.

With the help of the characteristics of the graphs, we were also able to combine two identical or different techniques to compress a graph, as proposed in section IV. We also compare our compression results with state of the art compression CBT-CBT and $ck^d - trees$, as shown in Table IV.

We implement our algorithms on real-world datasets and show significant improvements in the time required to query edges or any node's neighbors at a given time over the existing techniques, as shown in the Tables V and VI, thereby showing a clear space/time tradeoff between the compression size and the querying time.

Our future work will focus on exploiting the parallelism in improving the timings for the compression techniques on a wider domain of graphs. Also, we plan use the faster querying on compressed structure to enable us an opportunity to develop graph algorithms with time constraints.

REFERENCES

[1] A. Ferreira and L. Viennot, "A Note on Models, Algorithms, and Data Structures for Dynamic Communication Networks," Research Report RR-4403, INRIA, 2002.

[2] F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan, "On compressing social networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, (New York, NY, USA), p. 219–228, Association for Computing Machinery, 2009.

[3] P. Boldi and S. Vigna, "The Webgraph Framework I: Compression Techniques," in *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, (New York, NY, USA), pp. 595–602, ACM, 2004.

[4] M. Nelson, S. Radhakrishnan, A. Chatterjee, and C. Sekharan, "Queryable Compression on Streaming Social Networks," in *Big Data (Big Data), 2017 IEEE International Conference on*, IEEE BigData '17, IEEE Computer Society, 2017.

[5] R. A. Snay, "Reducing the profile of sparse symmetric matrices," *Bulletin Géodésique*, vol. 50, no. 4, pp. 341–352, 1976.

[6] D. Caro, M. A. Rodriguez, N. R. Brisaboa, and A. Farina, "Compressed kd-tree for temporal graphs," *Knowl. Inf. Syst.*, vol. 49, pp. 553–595, Nov. 2016.

[7] N. R. Brisaboa, S. Ladra, and G. Navarro, "Compact representation of web graphs with extended functionality," *Information Systems*, vol. 39, pp. 152–174, 2014.

[8] A. G. Labouseur, J. Birnbaum, P. W. Olsen, Jr., S. R. Spillane, J. Vijayan, J.-H. Hwang, and W.-S. Han, "The G* Graph Database: Efficiently Managing Large Distributed Dynamic Graphs," *Distrib. Parallel Databases*, vol. 33, pp. 479–514, Dec. 2015.

[9] D. Caro, M. Andrea Rodríguez, and N. R. Brisaboa, "Data structures for temporal graphs based on compact sequence representations," *Inf. Syst.*, vol. 51, pp. 1–26, July 2015.

[10] C. Ren, E. Lo, B. Kao, X. Zhu, and R. Cheng, "On querying historical evolving graph sequences," *PVLDB*, vol. 4, pp. 726–737, 2011.

[11] S. Álvarez-García, N. R. Brisaboa, G. d. Bernardo, and G. Navarro, "Interleaved K2-Tree: Indexing and Navigating Ternary Relations," in *2014 Data Compression Conference*, pp. 342–351, March 2014.

[12] G. D. Bernardo, N. R. Brisaboa, D. Caro, and M. A. Rodríguez, "Compact data structures for temporal graphs," in *2013 Data Compression Conference*, pp. 477–477, March 2013.

[13] R. Grossi, A. Gupta, and J. S. Vitter, "High-order Entropy-compressed Text Indexes," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, (Philadelphia, PA, USA), pp. 841–850, Society for Industrial and Applied Mathematics, 2003.

[14] N. R. Brisaboa, D. Caro, A. Fariña, and M. A. Rodríguez, "A compressed suffix-array strategy for temporal-graph indexing," in *SPIRE*, 2014.

[15] M. Nelson, S. Radhakrishnan, and C. Sekharan, "Queryable Compression on Time-Evolving Social Networks with Streaming," in *Big Data (Big Data), 2018 IEEE International Conference on*, IEEE BigData '18, IEEE Computer Society, 2018.

[16] "http://socialnetworks.mpi-sws.org/data-www2009.html," 03/2020.

[17] "http://konect.uni-koblenz.de/," 03/2020.

[18] "http://webscope.sandbox.yahoo.com/catalog.php?datatype=g," 03/2020.

[19] "https://github.com/sudhigopal/csr_cbt_paper," 02/2021.