IBRIDIA: A hybrid solution for processing big logistics data

Mohammed Alshaer, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid, Mohamed Dbouk

▶ **To cite this version:**

Mohammed Alshaer, Yehia Taher, Rafiqul Haque, Mohand-Said Hacid, Mohamed Dbouk. IBRIDIA: A hybrid solution for processing big logistics data. Future Generation Computer Systems, 2019, 97, pp.792-804. 10.1016/j.future.2019.02.044 . hal-02352954

HAL Id: hal-02352954
https://hal.science/hal-02352954

Submitted on 22 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Version of Record: https://www.sciencedirect.com/science/article/pii/S0167739X1830606X
Manuscript_ca32a029a7d2111b605205a4652c56db

IBRIDIA: A Hybrid Solution for Processing Big Logistics Data

Mohammed AlShaer[b,c], Yehia Taher[a], Rafiqul Haque[d], Mohand-Saïd Hacid[b], Mohamed Dbouk[c]

[a] *Université de Versailles Saint-Quentin-en-Yvelines, France*
[b] *Université Claude Bernard Lyon 1, France*
[c] *Lebanese University, Beirut, Lebanon*
[d] *Cognitus, Paris, France*

Abstract

Internet of Things (IoT) is leading to a paradigm shift within the logistics industry. Logistics services providers use sensor technologies such as GPS or telemetry to track and manage their shipment processes. Additionally, they use external data that contain critical information about events such as traffic, accidents, and natural disasters. Correlating data from different sensors and social media and performing analysis in real-time provide opportunities to predict events and prevent unexpected delivery delay at run-time. However, collecting and processing data from heterogeneous sources foster problems due to variety and velocity of data. In addition, processing data in real-time is heavily challenging that it cannot be dealt with using conventional logistics information systems. In this paper, we present a hybrid framework for processing massive volume of data in batch style and real-time. Our framework is built upon Johnson's hierarchical clustering (HCL) algorithm which produces a dendogram that represents different clusters of data objects.

Keywords: Realtime Processing, Clustering, Big Data, Internet of Things, Logistics, Hierarchical Clustering Algorithm

1. Introduction

Lately, with the advent of the Internet of Things (IoT), the operational landscape of the logistics industry is changing. Today, logistics companies

© 2019 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

(such as DHL[1] and FedEx[2]) use various sensors for tracking delivery, maintaining sensitive products, and many other purposes. Sensors assist in tagging and connecting factories, ships, and machines. They also allow handling real-time events. Additionally, connectivity of "things" enables instant communication between devices via the Internet [1]. This hyper-connected ecosystem promises far-reaching payoffs for logistics operators, their business customers, and end-customers [2]. One of the major advantages of IoT-based ecosystem is that it enables connecting the logistics sensors with external sensors such as weather sensors and traffic (GPS) sensors, etc. Furthermore, IoT enables connecting with social media such as Twitter which very often provides important traffic information tweeted by the users. The sensors and social media produce information about events such as accident, weather, natural hazards, heavy road constructions, etc. which are critical to logistics companies. These information can be used to carry out some critical analysis such as predictive analysis to forecast shipment delay or prescriptive analysis to optimize routes to guarantee in-time delivery which increases customer satisfaction and hence guarantees customer retention.

Although many solutions were proposed in the last two decades within logistics domain to tackle various problems, delivery delay remained an open issue. Timely delivery is a huge challenge for logistics companies because sometimes delays are caused by factors outside of anybody's control. Delay has various impacts such as, customer churn or cancellation of orders which eventually leads to huge losses. Therefore, *timely delivery* is critically important to logistics companies.

In recent years, logistics companies have started to investigate how to exploit data in predicting delay. The data driven prediction of delay is gaining popularity. Especially, with the advent of Big Data technologies, the logistics providers are focusing heavily on using streams of events such as accident, high-traffic stemming from external sources such as social media to perform analysis and predict delay in real-time. The real-time prediction of delay enables companies to pro-act such as optimizing route on the fly in (nearly) real-time. We have investigated the requirements of a real-time system which can perform analysis and predict delay. The core requirements are: ability to collect logistics data in real-time from multiple heterogeneous sensors,

[1]http://www.dhl.com/en.html
[2]http://www.fedex.com/us/

social media, and business processes; ability to process data efficiently in real-time or batch-style; a model for analyzing data for predicting the delay; and a model which produces an optimal routing plan to prevent the predicted delay. However, since data is the key element of analysis, efficient processing of data to produce quality dataset is a *sine qua non*. In this paper, we focus on developing a *hybrid* solution which enables efficient processing of data in realtime and batch style. It is worth noting that this paper is an extension of our previous work [3] where we developed batch style data processing. In this paper, we added functionalities that enable real-time processing of data. Realtime processing is strongly required to enable logistics service providers to perform analysis in realtime.

Existing data processing approaches (*e.g.*, techniques or algorithms) are not adequately efficient to process data in real-time. The existing solutions are built on classical data processing techniques. Therefore, conventional logistics information systems are not able to process sensor or social media data in real-time because these data flow with high velocity [4]. Additionally, the traditional data processing approaches are not able to deal schemaless data such as text. In the following, we explain the data variety and velocity challenges:

- **Variety** denotes different types of data models such as structured (*e.g.*, data stored in relational tables) and semistructured (*e.g.*, JSON and XML). Also, data may not have any structure such as text. Processing such a wide variety of data is heavily challenging and conventional information systems are not ready to tackle the variety challenge. Additionally, modern technologies have limited capabilities to tackle variety challenge. For instance, we conducted an experimental study with a solution called "Massive Online Analysis" (MOA[3]) which is an advanced version of one of the most widely used machine learning solution called WEKA[4]. We found that it cannot load tweets (which are texts). The shortcoming of existing solutions leave one very important question to logistics companies: *How to handle different data representations?* Moreover, variety hinders data integration [5, 6, 7]. Data integration is of critical importance within Big Data because it enables correlating events stemming from heterogeneous sources and enables predicting

[3]http://moa.cms.waikato.ac.nz

[4]http://www.cs.waikato.ac.nz/ml/weka/

delay with higher accuracy.

- **Velocity** refers to the speed at which data flow within and across the Web. Unlike the earlier days, data now-a-days is in motion. Millions of records may result in a millisecond. Sensor and social media data move with high speed. There are solutions which can be used to collect high-speed data yet processing them in real-time is heavily challenging.

In our solution called *ProLoD* [3], we focused on variety challenge. In this paper, we extended ProLoD to tackle the velocity challenge by adding functionalities for collecting and processing data streams in real-time. With this extension, our solution named *IBRIDIA* has turned into a hybrid solution that is able to process data in both real-time and batch style. IBRIDIA relies on extended hierarchal clustering algorithm proposed in [8] for processing data streams flowing at high-speed. The core contribution in this paper include the following:

- Developing a data streamer which fetch logistics data streams from sources such as social media and sensors.

- Building a real-time data processing engine that relies on extended agglomerative hierarchical algorithm.

The remainder of the paper is organized as follows. In Section 2, we will present a motivating scenario. In Section 3, we will describe the problem we are trying to solve more clearly. In Section 4, we present the work related to our research. Our previous work *ProLoD* and the extension of *ProLoD* which is the realtime data processor *RePLoD* is discussed in Section 5 by presenting the overall solution *IBRIDIA*. In Section 6, we present briefly the implementation of *IBRIDIA*. We showed the results of several experiments between the two data processing components *ProLoD* and *RePLoD* in Section 7. We conclude the work in Section 8.

2. Motivating Scenario

There are different modes of shipment used by logistics service providers including air cargo, ships, and ground cargo (e.g., Lorries and trucks). A single mode of transportation may not be adequate to deliver goods. Especially, a cross-border long-running shipment may include several modes of

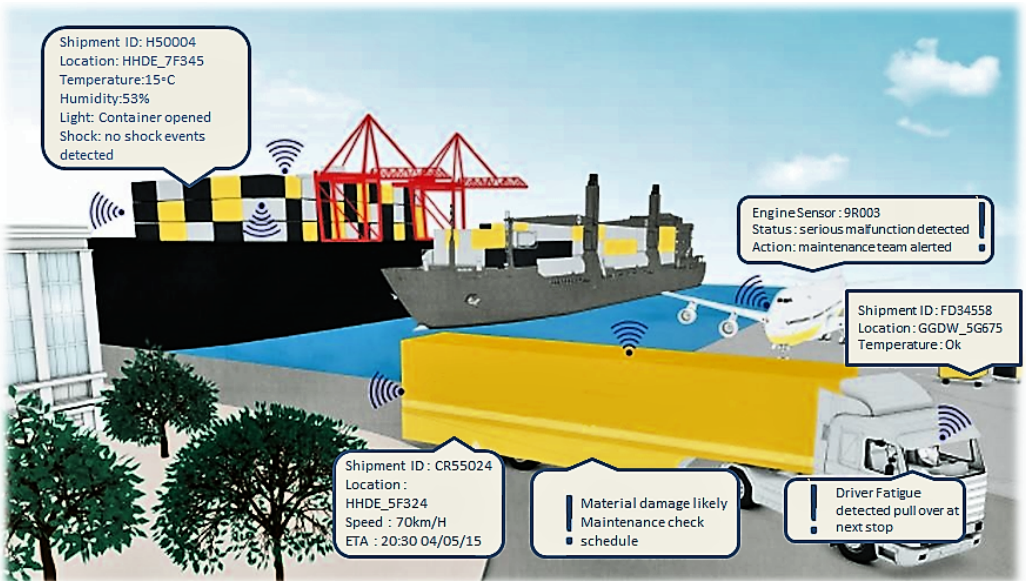


Figure 1: The Multi-modal Logistics System (Source: [9])

transportation. Consider a case where a product manufactured in China will be shipped to different customers located in different cities in the United States; the shipment process has to be multi-modal which means that the process will include lorries, trucks, train, ship or air *etc.* (Figure 1).

The integrated multi-modal logistics processes are prone to encounter various challenges namely delivery delay. For instance, *the shipment could be delayed if clearance at the port is delayed*, even if all other modes of transportation meet pre-defined schedule. Uncertain events such as natural disaster, war, strike, protest may affect one or more of the delivery modes at one or more steps of the integrated logistics processes. Uncertainty is the major challenge concerning such events. Therefore, pro-activeness to the best of our knowledge is a suitable approach which needs continuous streaming of data that contains information of events that may lead to delivery delay. In other words, realtime analysis of data to extract information of events which may lead to delivery delay.

3. Problem Description

There are different challenges involved in an integrated mission-critical logistics process. The predominant challenges reported by experts include the

followings: in-time delivery, cost optimization, efficient management of intermodal transportation, transferring information, Security, and Infrastructure [10, 11, 12].

However, in-time delivery is one of the key performance indicators (KPIs) of logistics services. Delay of a scheduled (expected) delivery increases customer dissatisfaction. In order to prevent delay, logistics service providers heavily rely on automated solutions. Business intelligence is a widely used solution that enables performing different types of cycle time analytics [13] that analyze delay for different combinations of goods, routes, modes and weather condition.

However, this is a reactive approach which performs analysis on historical data. In other words, traditional business intelligence especially, BI&A 1.0 and BI&A 2.0 use only internal data which stem from different information systems and legacy systems [14].

Also, the process mining tool PRoM [15] – a recent tool for mining business processes – lacks the ability to exploit external data. Consequently, the analytics misses important external data such as sensor data (for example, global positioning systems (GPS) data) and social media data (such as Twitter data). The advent of Big Data technologies created wide opportunities to exploit such external data which enhances the predictability of analytics. More specifically, these data are effective to forecast potential delivery delay as they contain important information such as high traffic, weather report, political events such as protest, and other events such as unexpected natural disasters (e.g., Earthquake).

However, collecting, cleaning, filtering, integrating, and storing data from heterogeneous sources is a non-trivial task. Particularly, a seamless integration of unstructured text sourcing from Twitter with structured business process data is not possible by existing logistics solution frameworks. Dong et al. [16], outlined several Big Data integration challenges. Furthermore, there are several techniques and approaches for processing data, however, our investigation suggests that there is a scope to improve these techniques specifically the clustering algorithms.

In this paper, we aim to address the two problems discussed in the above: data preparation and data processing. The goal is to provide efficiently processed data streams which are employed to perform an effective analysis. To that end, we developed IBRIDIA that enables to fetch data from various sources, pre-process and process tasks in real-time.

4. Related Work

Clustering data in real time has drawn a huge research interests in the recent past, especially with an extensive demand of using analytics on streaming data. We choose clustering because we needed to work with unlabeled data which means no model training data set is available and we know nothing about the data previously, thus we needed an unsupervised learning model. A few initiatives have been taken in the field where BIRCH (balanced iterative reducing and clustering using hierarchies) was one of the basic methods for data stream clustering [17]. Essentially, BIRCH introduced micro and macro clustering as two new concepts, and was fabricated to work with traditional data mining techniques but not with voluminous amounts of data sets like data streams. Later, the STREAM algorithm proposed by Guha et al. [18, 19] was an extension of classical K-median and the first algorithm known with the ability to perform clustering on entire data streams.

In [20], Babcock et al. suggested the sliding window model as an extension to STREAM and thus they changed the concept from one single pass over the data, to the concept of receiving data points as a stream and taking into consideration the points that fall within a specific range representing the sliding window. The CluStream framework was suggested by Aggarwal et al. [21] and it was considered effective in handling data streams; it divides the clustering process into two components: online and offline. The former periodically uses micro clusters to store detailed summary statistics, and the latter uses the summary statistics to produce clusters. Later, Aggarwal et al. [22] suggested HPStream that works on data streams high dimensionality reduction by means of data projection prior to clustering. Denstream algorithm was proposed in [23] as an extension for DBSCAN where they combined micro clustering concept to the density based connectivity search. Another density-based extension is the D-Stream proposed in [24]. The proposed solution maps each new data point to a specific grid upon its arrival; the density information is stored and then clustering is applied to the density data grids. Khalilian et al. [25] suggested an improvement for well-known K-Means algorithm. They applied the widely-known divide and conquer method that is capable of clustering objects with high quality and efficiency. Specifically, the solution is suitable for analyzing high dimensional data, but not for realtime data streams. EStream [26] is a data stream clustering technique, which supports five types of evolution in streaming data. They are as

follows: appearance of new cluster, disappearance of an old cluster, split of a large cluster, merging of two similar clusters and changes in the behavior of cluster itself. It uses a decaying cluster structure with a histogram to approximate the streaming data. Although the algorithm has the disadvantage of needing an expert intervention to specify many parameters before it works, its performance is better than HPStream algorithm [27].

In [28], a multi-level unordered sampling technique was suggested to boost the time performance of fuzzy C-means. The technique is double phased. In the first phase, the random sampling is applied to estimate centroids and then fuzzy C-means "FCM" is performed on the full data with the previously initialized centroids. Fuzzy C-means together with probabilistic clustering were then extended to work on huge data sets by the sampling based proposal of Richards and James [29]. In [30], an algorithm called AFCM was suggested to speed up FCM. This is done using lookup table. In [31], the authors proposed several efficient and scalable parallel algorithms for a special purpose architecture description of a modified FCM algorithm known as 2rFCM. A fast FCM algorithm was proposed in [32]. They employed the concept of decreasing the number of distance calculations by checking the membership value for each point.

Furthermore, many machine learning libraries are used to implement the algorithms discussed above. We adopt building a connectivity model (hierarchical clustering algorithm) in our framework. The three main reasons for our decision are the following:

1. No prior knowledge of the nature of the coming data (format, structure, features, etc.)

2. No prior knowledge of how many categories can the data be classified into (number of clusters is unforeseen)

3. The clusters probably will evolve with time (keep changing dynamically, i.e., creating, removing, splitting and merging clusters).

Since we are interested mainly in hierarchical clustering, we studied the machine learning libraries within the scope of this algorithm. Datumbox [33] is a robust framework that provides different functions like Sentiment Analysis, Twitter Sentiment Analysis, Subjectivity Analysis, Topic Classification, Language Detection, Keyword Extraction, Text Extraction and Document Similarity. At low-level, the basic machine learning algorithms such

as K-means, hierarchical clustering, and classification algorithms perform the above functionalities. Although this library is very powerful in handling different data types (categorical, numerical, etc.), it was not implemented to work in the environment of streaming data. Therefore, it can read bulk data. Apache Spark [34] is a fast-general-purpose cluster computing system. It supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLlib for machine learning, GraphX for graph processing, and Spark Streaming, For clustering, Spark offers limited features in particular; it supports few algorithms such as K-Means. The library is missing hierarchical clustering algorithm, which we found suitable for our research project.

Furthermore, the library offers the streaming K-means; it is applicable on numerical data only which is a limitation for Big Data where data variety is major challenge. SPMF [35] offers implementations of 120 data mining algorithms for association rule mining, item set mining, sequential pattern mining, and of course clustering and classification. It works only with numerical data and this was mentioned explicitly in the documentation. The input is a set of vectors containing double values only, a parameter "max-distance" and a distance function. This implies the same limitation that Spark has. To the best of our understanding, this shortcoming is obvious because, the clustering is usually done according to Euclidean or Manhattan distance functions that need numerical data to be applied.

WEKA [36] is a widely-known library. It is an integrated system which consists of a collection of machine learning algorithms for data mining tasks. The algorithms can either be employed directly for a dataset or called from within a Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is a well-suited solution for developing new machine learning schemes. It is very efficient and can be used with big data analytics but needs to work only on data at rest and with a specific file format called ARFF. Recently, an initiative has been taken to extend WEKA to be used for mining data Streams. MOA (Massive Online Analysis) [37] is an open source library for data stream mining. It includes a collection of machine learning algorithms (classification, regression, clustering, outlier detection, and concept drift detection and recommender systems) and tools for evaluation. MOA provides approximately all clustering algorithms for streaming data but it is confined to a specific format just like WEKA (ARFF file only). It enables to generate synthetic data streams and allows users to visualize data clustering in real-time.

Waller and Fawcett [38] underline the importance of data and analytics for supply chain management (SCM). They introduce the term "SCM data science", referring to Big Data Analytics (BDA), as the "application of quantitative and qualitative methods from a variety of disciplines in combination with SCM theory to solve relevant SCM problems and predict outcomes, taking into account data quality and availability issues". Bi and Cochran [39] argue that BDA has been identified as a critical technology to support data acquisition, storage, and analytics in data management systems in modern manufacturing. They attempt to connect IoT and BD to advanced manufacturing information systems to help to streamline the existing bottlenecks through improving forecasting systems. Similarly, Gong et al. [40] argue that a production control system (PCS) can be considered an information-processing organization (IPO).

Discussion

The solutions and tools mentioned in the state of the art provide a variety of machine learning algorithms that can be used for predictive analytics tasks, such as feature selection, parameter optimization and result validation. Many of these systems offer basic visualizations including residual plots, scatter plots and line charts. However, the visualization feature of these systems is limited to presenting the final results; they do not offer any interactive means for manipulation, feature selection or model refinement; instead, these systems often opt to show baseline models or simple statistical measures for result validation, working as more of a black-box system.

SPMF and Spark worked only on numerical data. Additionally, we found that Spark's MLib library does not have an implementation of hierarchical clustering. Datum box had a specific structure for storing and processing data and it lacks the ability to read data by lines. It can handle data as a batch that can be read in one go and thus it was not suitable for real time environments.

WEKA developers extended it into MOA, the version that fits real time processing but it is still very recent and thus had some code bugs that did not allow us to benefit from it[5]. Besides that, MOA is locked into a specific file format which is ARFF and hence it is unable to read other data format. An

[5]We contacted Dr. Albert Bifet the author of the library for assistance because it was only running for a specific number of data points then starts throwing errors but the problem was not solved.

ARFF format needs to have attributes, types, and data explicitly mentioned within the file. Nevertheless, in our case the data streamed and fed into the algorithm do not necessarily have an attribute. Rather, data could be a set of records each of which is made up of different text words. Therefore, we decided to write our own implementation.

In order to fulfill the needs of the logistics sector, we can use social media to do the data enrichment and combine the data from multiple sources to validate and analyze the current situations. But using social media alone, might not be of major interest, there is a need to build a new model that is enriched from social media and the different sensors to predict the delay for the delivery process and suggest improvement according to it. Even, after predicting the delay, more work should be done using advanced analytics for the optimization of the route planing algorithms. The state of art was missing the data integration to forming up the convenient model from the different data sources which have different data presentations (such as text, JSON, XML, audio, video, etc.)

In summary, considering the evaluation of data sources, most of the existing solutions are confined to one data source for analytics and prediction. Additionally, for realtime systems with continuous improvement, the majority of researches used large static historical datasets for their testing while our approach does not depend on historical data alone.

5. IBRIDIA – Solution Overview

In this section, we describe the two main modules of IBRIDIA. In logistics system, we have data generated from internal system for stock, orders, shipments, *etc.* Also, there is a need to collect and analyze data from external sources in realtime especially to monitor the different statuses of the delivery. To address both needs, within IBRIDIA, we developed a batch style data processing engine that we called *ProLod* and a realtime data processing engine that we called *RePLoD*. We explain these two modules in the following subsections. Then, we describe the data processing model that IBRIDIA relies on for both realtime and batch style data processing. Figure 2 depicts a high-level architecture of IBRIDIA.

In IBRIDIA there are four components: *data streamer*, *the storage*, *batch style data processing engine*, and *real-time data processing engine*. The data streamer fetch data from internal and external data sources and ingest them into realtime processing engine and storage. It is worth noting that we used

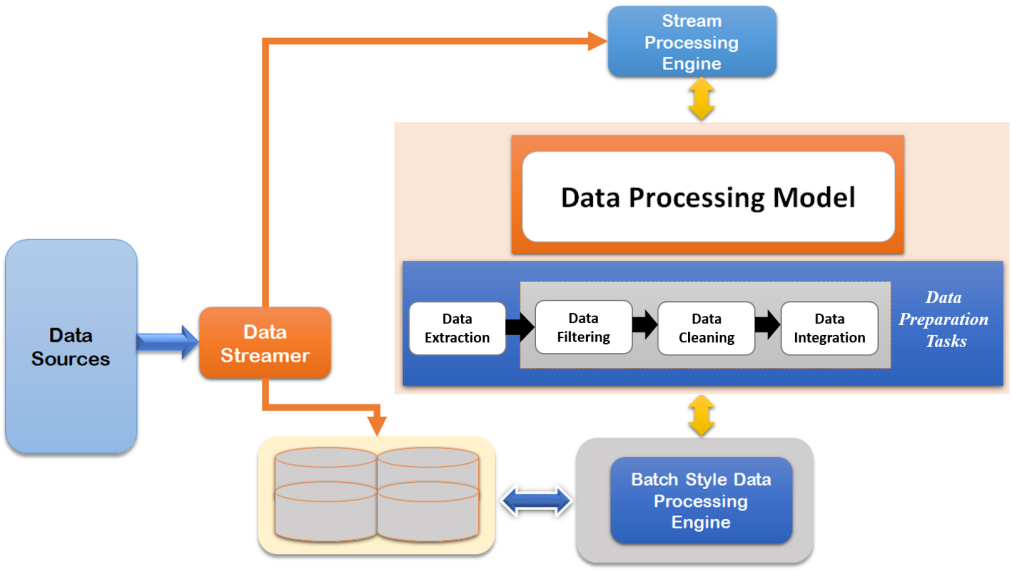


Figure 2: The high level architecture of IBRIDIA

native storage in our previous work, however, we developed Hadoop based scalable storage in IBRIDIA so that a massive scale data can be stored. The batch style data processing engine (ProLoD) *reads/extracts* data from storage and perform data preparation and processing tasks including cleansing, filtering, etc. In case of realtime processing, the streamer sends data directly into the processing engine (RePloD) which carries out processing tasks in realtime. In the following, we provide more detail about the data processing engines.

5.1. ProLod – Batch Style Data Processing Engine

ProLoD represents the batch-style processor for processing of logistics data stemming from multiple heterogeneous sensors (that include vehicle sensor, weather sensor, etc.), logistics applications, microblog (e.g., Twitter), and social media (e.g., Facebook). ProLoD comprises two phases: data preparation phase and processing. The former consists of data extraction (collection), data cleansing, data filtering, data integration, and data storage. In the latter phase, well-prepared data are clustered. Figure 2 shows different functionalities of these phases. ProLoD relies on different machine learning techniques specifically the clustering techniques for data processing. ProLoD includes five components: data extractor, data cleaner, data filter,

data integrator, and data storage for performing data preparation functions. It has a data processor which performs clustering in the second phase.

5.2. RePLoD – Realtime Data Processing Engine

RePLoD represents the core component in our framework for processing the realtime data. As the speed of events from sensors and social media increases, it creates an emerging need for fast processing known as stream processing mechanism. Events may lead to catastrophic consequences if not handled properly in-time. RePloD was designed to add the missing functionalities to the system by adding a convenient way for handling the events generated in realtime. These events are first enqueued into the memory through distributed messaging system. The memory was used instead of the disk because its access speed is faster than the disk by 100,000 times. In this way, we prevent any overwhelming of the receiver and we guarantee fault tolerance in case of any failure. This added feature prevents the loss of any of the data due to their fast generation. Batch processing is not always the right way to do it, sometimes it is important to do the processing on the fly as soon as the events arrive to the servers. These cases can be faced in real world scenarios such as accidents occurring now on roads, bad weather, maintenance of buildings which need to be notified for the driver in realtime to prevent the catastrophic effects due to delay in delivery. These facts carried us to extend the processing behavior to be able to do the required processing in realtime without doing it in batches. RePLoD performs clustering of the events in realtime and gets immediate insights over the processed data.

5.3. Data Preparation Tasks

Both ProLoD and RePloD perform five data preparation tasks using two different approaches namely batch style and realtime respectively. These tasks are explained in the following.

- *Data Extraction*: It is the systematic approach to gather and measure information from a variety of sources to get a complete and accurate picture of an area of interest. The data extractor works with both internal and external sources of data. The internal data sources are typically the information systems used by the users. Consider a user that has an information system consisting of a supply chain management (SCM), a customer relationship management (CRM), a logistics management system, and an account management system (AMS). These systems

produce a large amount of data that are collected by the data extractor. It also fetches data from external sources such as Twitter, traffic sensors, weather sensors, Facebook and other social medias. In addition, IBRIDIA's processing components extract archived sensor data of completed logistics processes. In most of the cases, we found that data extraction from internal sources is more trivial than external ones. Additionally, internal data were transferred faster than the external ones. IBRIDIA can collect structured and unstructured data. For instance, it collects unstructured texts from Twitter, and Facebook and structured business process data from logistics information system.

- *Data Filtering*: It refers to a wide range of strategies or solutions for refining data sets. Datasets are refined into simply what a user (or set of users) needs, without including other data that can be repetitive, irrelevant or even sensitive. ProLoD and RePLoD aim to eliminate all possibilities of data overloading which can increase computational cost and effort during data processing and may jeopardy the analysis regarding accuracy. They collects data that are related to logistics and specifically the data chunks whose hashtags (the words prefixed by #) determine direct and indirect connections with transportation, delivery, logistics, shipment, etc. Consider the term "protest" which may be a political protest or else but can have a great impact on delivery of goods and hence can delay the delivery. However, consider a tweet "the New York stock prices are extremely high today" which will be removed by the data filter because it does not carry any information related to logistics processes.

- *Data Cleaning (i.e. data scrubbing)*: It is the process of detecting and correcting (or removing) corrupted or inaccurate records from a record set, table, or database. ProLoD and RePLoD clean data from all unwanted symbols, numbers, stopping words, hashtags, and any other data items that might lead to noise and cause inaccuracy. Figure 3 shows an example of cleaning Twitter data using ProLoD.

- *Data Integration*: In IBRIDIA, data integration is performed in two steps. In the first step, the data are transformed from source to target serialization format. Currently, the target format is CSV. The second step is merging the transformed data.

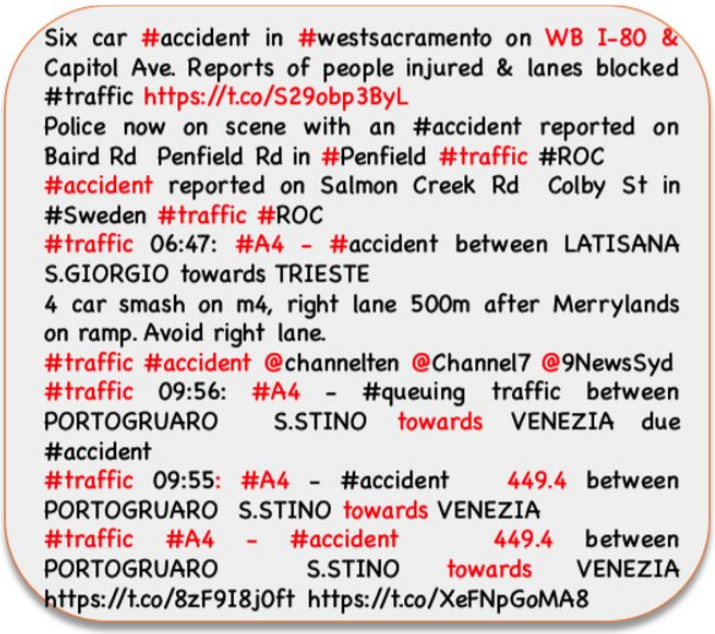


Figure 3: An example of cleaning data with ProLoD.

- *Data Storage*: This step aims to deal with the storage of the integrated datasets. After preparing the integrated datasets, ProLoD and RePLoD store data into the storage.

5.4. Data Processing Model

As mentioned earlier, IBRIDIA relies on the data processing model which we developed in our previous work [3]. We explain the data processing model in this section. Choosing techniques or methods for developing the model is not a trivial job. There is an exhaustive list of techniques available from machine learning, data mining, and statistics. In our case, we considered the nature of data and operation styles to choose the right technique for building data processing model. Our data processing model relies on *unsupervised learning techniques* [41]. Unsupervised learning is a machine learning approach in which a system only receives input (x_1, x_2,..., x_n) without any corresponding (supervised) output (which is also called *labeled output*). *Clustering* and *dimensionality reduction* are the two most well-known unsupervised learning techniques. We choose *clustering* for our model because the objective function is expected to produce a clustered dataset which facil-

itates efficient analysis in prediction of *delivery delay*. Clustering is a process of grouping or segmenting data items that are *similar* between them in a cluster and *dissimilar* to the data items that belong to another cluster [41].

There are different types of cluster models which are grouped into *Connectivity models*, *Centroid model* and *Distribution models*, *Density models*, *Subspace model*, *Group model*, and *Graph-based models* [42]. We are interested in techniques used for building connectivity model which fits to our objective more than the others. *Hierarchical clustering* is a widely used approach for building connectivity model based on distance connectivity between the data items. It is a process of producing a sequence of nested cluster ranging from *singleton clusters* of individual points to an all-inclusive cluster [43]. The hierarchy of the clusters are graphically represented by a dendogram [44]. There are two approaches to develop a hierarchical cluster model:

- *Agglomeration* refers to an approach that start with the points as individual clusters and, at each step, merge the closest pair of clusters. It is also known as *Bottom-Up approach*.

- *Divisive* refers to an approach that starts with one, all-inclusive cluster and, at each step, splits a cluster until only singleton clusters of individual points remain. It is also known as *Top-Down approach*.

We found agglomerative hierarchical clustering approach for our solution because the bottom up approach is more flexible than the others in terms of choosing the number of clusters. The algorithm groups data one by one based on the nearest distance measure of all the pairwise distance between the data points. The distance between the data points is recalculated iteratively. However, the choice of distance to consider for grouping data points is a critical matter. Several methods are available to address this question. These methods – found in [45] – are summarized in the following:

Definition 1. *Single-linkage:* $d(C_i,C_j) = min_{x \in C_i, x\prime \in C_j}\ d(x,x\prime)$*. It is equivalent to the minimum spanning tree algorithm [46]. One can set a threshold and stop clustering once the distance between clusters is above the threshold. Single-linkage tends to produce long and skinny clusters.*

Definition 2. *Complete-linkage:* $d(C_i,C_j) = max_{x \in C_i, x\prime \in C_j}\ d(x,x\prime)$*. Clusters tend to be compact and roughly equal in diameter.*

Definition 3. *Average distance:* $d(C_i, C_j) = \frac{\sum x \in C_i, x\prime \in C_j d(x,x\prime)}{|C_i| \cdot |C_j|}$.

Definition 4. *Wards method* $d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2$ *is the sum of squared Euclidean distance is minimized.*

The iteration is continued by grouping data items until a cluster is formed. As mentioned earlier that the clusters are presented graphically by a *dendogram* which allows to calculate the number of clusters that should be produced, at the end. There are several variants of the agglomerative hierarchical clustering algorithm. Below we present the steps involved in performing an agglomerative hierarchical clustering. Consider a set of data points $\mathcal{S} = (x_1, x_2, x_3, ..., x_n)$ as input. The agglomerative hierarchical clustering algorithm performs the following steps:

- *Step 1*: Disjoint cluster ($\mathcal{C}$) of level $\mathcal{L}(0) = 0$ and sequence number $\mathcal{M} = 0$
- *Step 2*: Calculate the least distance ($\mathcal{D}$) pair of clusters in the current $\mathcal{C}$, say pair $\mathcal{P}(r,\ s)$, according to $\mathcal{D}(r,s) = Min(\ \mathcal{D}(i,\ j))$ where the minimum is over all pairs of clusters in the current clustering
- *Step 3*: Increment the sequence number, $\mathcal{M} = \mathcal{M} + 1$
- *Step 4*: Merge $\mathcal{C}(r)$ and $\mathcal{C}(s) \rightarrow \mathcal{C}\ (z)$ which is a new cluster. Set the level of this clustering to $\mathcal{L}(z) = \mathcal{D}\ (r),(s)$
- *Step 5*: Update the distance matrix Ψ, (delete the rows and columns corresponding to clusters $\mathcal{C}\ (r)$ and $\mathcal{C}\ (s)$ and add a row and column corresponding to $\mathcal{C}\ (z)$. The distance between the new cluster, denoted $(r,\ s)$ and the old cluster(k) is defined as follows: $\mathcal{D}\ ((k),\ (r,\ s)) = Min\ (\mathcal{D}[(k),(r)],\ \mathcal{D}\ ((k),(s)))$
- *Step 6*: Repeat until ONLY one cluster remains.

In [3], we reported several disadvantages of the basic agglomerative clustering algorithm . In particular, undoing is not allowed and the time complexity is $\mathcal{O}(n^2 log\ n)$ where n denotes the number of data points. For a large

dataset, the performance with respect to processing time may not be satisfactory. Based on the type of distance matrix chosen for merging, different algorithms may have one or more of the following drawbacks: (i) sensitivity to noise and outliers, (ii) partitioning a large cluster, (iii) difficulty in handling different sizes of clusters and handling convex shapes. In this algorithm, no objective function is directly minimized. Furthermore, in some cases identifying the correct number of clusters by the dendogram can be very difficult. Therefore, the basic algorithm agglomerative clustering algorithm is not suitable for clustering data. Hence, we choose extended agglomerative hierarchical algorithm proposed in [8]. We intend to use the hamming distance as a measuring criteria in our algorithm, because it can be used as a convenient measuring mechanism for string values which covers most of the unstructured data. Hamming distance measures the minimum number of substitutions required to change one string into the other (the minimum number of *errors* that could have transformed one string into the other). We modified Johnson's Hierarchical Clustering algorithm to become a stream clustering algorithm that supports incremental grouping of text messages according to their similar characteristics directly on the go. The theoretical steps performed by the modified algorithm is based on the theoretical steps of any agglomerative hierarchical clustering algorithm as shown previously. We identify the practical algorithmic steps of the clustering used in our solution IBRIDIA as follows:

- **Step 1**: Read new data streams.
- **Step 2**: Put the *unique* items in the vector format.
- **Step 3**: Fill a matrix of absence and presence of items.
- **Step 4**: Calculate hamming distance.
- **Step 5**: Update the distance matrix.
- **Step 6**: Create Cluster using minimum distance.
- **Step 7**: Repeat **until** only one cluster remains.

In what follows we explain the above steps using an example where we illustrate how IBRIDIA data processing model works. It begins with reading records. Since data is read from the first row, thus the attribute names do not exist; we expressed them here just to make the data set meaningful to the reader.

- Start with each record as a cluster on its own.
- Read new data streams

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01

- A unique item is added in the vector format.
- Fill in the matrix of items absence and presence.

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01
Rec1	1	1	1	1

- Build the similarity matrix using hamming distance. Currently, there is only one record.
  - The algorithm reads new record.

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01
	NetworkManagement	4/1/2017 8:00	NarrowLanes	LYON-06

  - Place the new Unique items.

NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01	4/1/2017 8:00	LYON-06

  - Update the matrix.

	NetworkManagement	1/30/2017 16:47	NarrowLanes	LYON-01	4/1/2017 8:00	LYON-06
Rec1	1	1	1	1	0	0
Rec2	1	0	1	0	1	1

  - Build similarity matrix using hamming distance
    - The Hamming distance can only be calculated between two strings of equal length. String 1: 111100 String 2: 101011.
    - Compare the bits of each string with the other.
    - If they are the same, record a “0” for that bit.
    - If they are different, record a “1” for that bit.
    - Compare each bit in succession and record either “1” or “0” as appropriate.

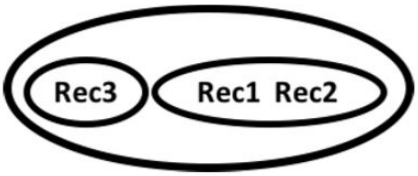
* Add all the ones and zeros in the record together to obtain the Hamming distance. Hamming distance = 0+1+0+1+1+1= 4.

– Update the distance matrix.

	Rec1	Rec2
Rec1	0	4
Rec2	4	0

- Create a cluster with minimum distance.

- The systems read new records and the previous steps are repeated. At the end a new cluster is created.



The iteration stops at this step because the execution loop produces a single cluster and no cluster can be created any further. We discuss the implementation of IBRIDIA in the next section.

6. Implementation of IBRIDIA

We studied various technologies for implementing IBRIDIA. We investigated existing libraries for data extraction, filtering, and transformation. Our goal was to reuse existing ones instead of developing the new ones. Also, we studied machine learning libraries including DatumBox[6], SPMF[7], Massive Online Analysis (MOA), and Spark MLib[8] to implement our data processing model. From our study, we found that existing libraries could not be used to implement our model (discussed in the previous section). Therefore, we

[6] http://www.datumbox.com
[7] http://www.philippe-fournier-viger.com/spmf/
[8] http://spark.apache.org/mllib/

decided to implement the model on our own. For implementation, we used Java language on Eclipse.

To sum up, IBRIDIA is a framework that integrates three APIs for extracting external data from different sources including Twitter API, Facebook API, and Open Weather API. It uses an open source parser. Also, it includes tools for cleaning and transforming incoming data. The prototype of ProLoD is available in GitHub.

We investigated different data processing frameworks including Apache Spark[9], and Apache Storm[10] to develop RePloD module of IBRIDIA. To the best of our understanding Storm is more potential computation system for our ProLoD. It is fast, can process over a million of tuples per second. It is scalable, fault-tolerant, guarantees our data will be processed, and is easy to set up and operate. Storm integrates with the queuing and database technologies we already use. A Storm topology consumes streams of data and processes those streams in arbitrarily complex ways. However, repartitioning the streams between each stage of the running computation is needed.

RePLoD consists of two main components: *data streamer* and *data processor*. We implemented the *Data Streamer* using Apache Kafka[11] and data processing engine using Apache Storm[12].

- *Apache Kafka*: It is a publish-subscribe based fault tolerant messaging system. It is fast and highly scalable distributed messaging technology. It is used in building durable data collection system where high throughput and reliable delivery of messages are critically important. Apache Kafka messaging system is merely a collection of topics split into one or more partitions. A Kafka partition is a linearly ordered sequence of messages, where each message is identified by their index (called as *offset*). All the data in a Kafka cluster is the disjointed union of partitions. Incoming messages are written at the end of a partition and messages are sequentially read by consumers. Durability is provided by replicating messages to different brokers. Apache Kafka provides four different types of APIs. The Producer API allows RePLoD to publish a stream of records to one or more Kafka topics.

[9] `http://spark.apache.org`
[10] `http://storm.apache.org`
[11] Apache Kafka: https://kafka.apache.org
[12] Apache Storm: http://storm.apache.org

The Consumer API allows RePLoD to subscribe to one or more topics and process the stream of records produced to them. The Streams API allows RePLoD to act as a stream processor, consuming an input stream from one or more topics and producing an output stream to one or more output topics, effectively transforming the input streams to output streams. The Connector API allows building and running reusable producers or consumers that connect Kafka topics to existing applications or data systems. For example, a connector to a relational database might capture every change to a table.

- *Apache Storm*: Apache Storm is a distributed realtime computation system. Storm makes it easy to reliably process unbounded streams of data for realtime processing. It is designed to process vast amount of data in a fault-tolerant and horizontal scalable method. It is a streaming data framework that has the capability of highest ingestion rates. Though Storm is stateless, its distributed environment and cluster state is managed by Apache ZooKeeper[13]. It is simple and you can execute all kinds of manipulations on real-time data in parallel. Apache Storm guarantees that every message will be processed through the topology at least once.

  Apache Storm consists of four main components: *tuple* is the main data structure which is a list of ordered elements; *stream* is an unordered sequence of tuples; *spouts* are the sources of stream; *bolts* are logical units. Bolts can perform the operations of filtering, aggregation, joining, interacting with data sources and databases. Bolt receives data and emits to one or more bolts. Spouts and bolts are connected together and they form a topology. Real-time application logic is specified inside Storm topology. In simple words, a topology is a directed graph where vertices are computation and edges are stream of data.

The data processing topology of RePLoD comprises the *data cleaner*, the *data filter*, the *data transformer*, and the *data clustering* component. Figure 4 shows the data processing topology of RePLoD.

The spouts and bolts RePLoD Topology constitute a directed acyclic graph (DAG). Spout is the entry point to the topology used to read the data from Apache Kafka. The Kafka-spout acts as Kafka consumer of the Kafka

[13]https://zookeeper.apache.org

Id	Executors	Tasks	Emitted	Transferred	Complete latency (ms)	Acked	Failed	Error Host	Erro
kafka_spout	2	2	40	40	0.000	0	0		

Showing 1 to 1 of 1 entries

Bolts (All time)

Id	Executors	Tasks	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)	Acked	Failed
realtime-clustering	2	2	0	0	0.015	446.000	20	0.000	0	0
stanford_nlp	2	2	20	20	0.000	13.000	20	0.000	0	0
twitter_analytics	2	2	20	0	0.000	5.000	40	0.000	0	0
twitter_cleaner	2	2	40	40	0.000	4.000	40	1.500	40	0
twitter_filter	2	2	40	40	0.000	2.000	20	6.000	20	0
twitter_transformer	2	2	20	20	0.000	1.000	20	0.000	0	0

Showing 1 to 6 of 6 entries

Figure 4: The Data Processing Topology of RePLoD

topic. The Kafka-spout reads all the messages ingested into the Kafka topic such as "tweets". This spout acts as the only connector between Kafka and Storm but what is interesting is the ability to execute every component of the spout and bolt within multiple executors. Data are inserted into the storm topology which consists of three bolts: `twitterfilter`, `twittercleaner`, and `twitteranalytics`. The `twitterfilter` is used for filtering the attributes of the tweets records. The tweets are consisting of several attributes; however, not all of them are important for the analysis. Therefore, we need to filter the relevant ones to our analysis such as "id, user, description, text, created time, location, etc". Then the simplified records that are emitted from the bolt `twitterfilter` are ingested as input to the next bolt `twittercleaner` which is used for cleaning all the characters that may affect the analysis. The analysis is carried out by the bolt `real-time-clustering` that is built on the real-time clustering algorithm that results in different clusters: merged, splitted or newly created. Once the data is cleaned, they are transformed the texts (e.g, tweets) to csv-like structure using "twitter-transformer" bolt. After the transformation is completed, we extract different named entities to understand the text content and make the analysis able to depend upon the features mentioned in the content. Finally, the clustering is carried out in real-time using `twitteranalytics` bolt which is built on hierarchical clustering algorithm that produces clusters in real-time. The clusters are saved in the disk.

7. Experiments

In this section, we discuss the results produced through experiments with ProLoD and RePLoD. We evaluate the performance of ProLoD and RePLoD over the metric execution time. Given below is the specification of the machine we used for our experiments:

- Processor: 2.40 GHz
- Memory: 4GB
- HDD: 500 GB
- Operating System: Windows 10(64 bit)

We compare the performance of our model with the one implemented by WEKA. Although we tested the performance of SPMF and Spark, unfortunately, we could not compare them to our work since they can only be applied to numerical data. Concerning MOA, we found bugs in it and thus we could not run our model. It allows only ARFF file formats as mentioned before and even though we converted our file to the needed format, it throws multiple exceptions when we tried to read data from an external file.

We implemented two different versions of our model: real-time and Batch style. We tested both versions with a test dataset. RePLoD reads data by records and clusters each incoming record. The clusters are mutable; a cluster may change when a new record is added in the cluster. However, since the algorithm is greedy, the execution time has a positive correlation with number of records, i.e., the execution time increases as the number of records increases. This can be solved using the scalability of the system by inserting more nodes to the cluster for faster processing resources. Table 1 shows the result of our experiment with the realtime version "RePLoD".

Table 1: The result of an experiment with realtime version "RePLoD"

Realtime	1st exec	2nd exec	3rd exec	4th exec	5th exec	average	seconds
8 records	466	426	451	432	420	439	0.439
16 records	3665	2301	2362	2069	2089	2497.2	2.4972
24 records	4363	4311	4491	4386	4135	4337.2	4.3372
32 records	7915	7926	7789	7784	7710	7824.8	7.8248
40 records	12861	12780	12849	12969	12958	12883.4	12.8834

The batch-style ProLoD performs bulk reading and clusters batch data.

The reading and processing occur only once per batch. Table 2 shows the results of batch style version "ProLoD".

Table 2: The result of an experiment with the batch style version of "ProLoD"

Not Realtime	1st exec	2nd exec	3rd exec	4th exec	5th exec	average	seconds
8 records	132	129	136	121	147	133	0.133
16 records	262	286	257	251	286	268.4	0.2684
24 records	465	393	427	393	429	421.4	0.4214
32 records	555	549	617	560	535	563.2	0.5632
40 records	728	719	779	855	726	761.4	0.7614

We compared the performance of both versions. Figure 5 shows the comparison. According to our experiments, we observed that the batch-style version performs better than the real-time version of IBRIDIA. Our study reveals that performance of the real-time version was not satisfactory due to the centralized environment of the experiment consisting of only one node. We believe that performance will improved significantly in a distributed and scalable computation framework with multiple nodes. We believe that technologies which we used for implementation of our solution called Apache Storm has high computational power; hence, it can process a huge number of records within a unit of time *e.g.*, a millisecond. In addition, we assume that the performance of the batch-style version of ProLoD may decrease if the dataset is large. Currently, the dataset is small; therefore, the size of each batch is small and faster in processing (clustering).

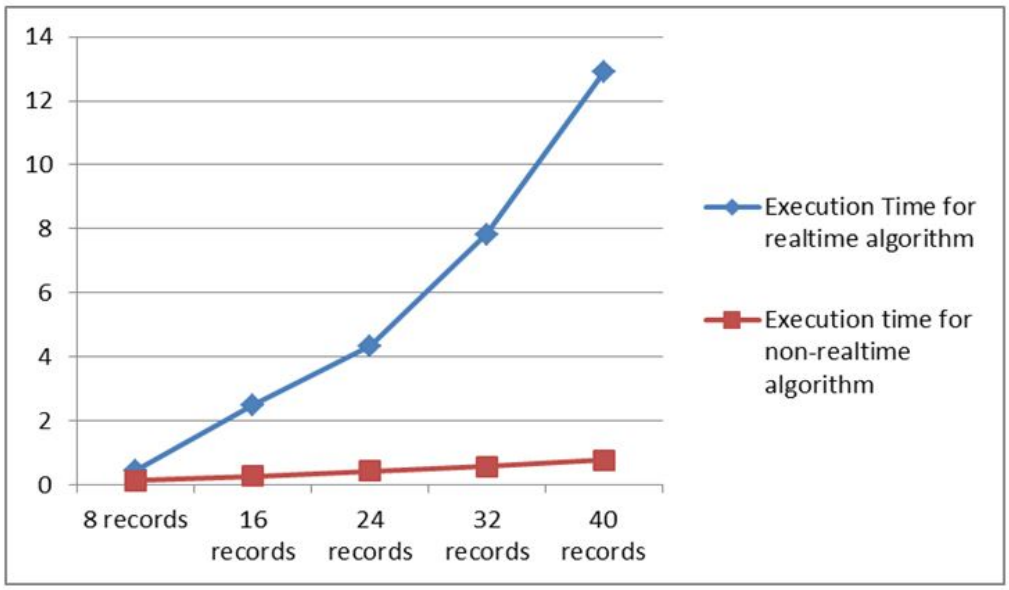


Figure 5: Execution time (in seconds) of RePLoD and ProLoD

We compared our model with the hierarchical clustering algorithm of WEKA. The WEKA algorithm produced 7 clusters with 93 records. Table 3 shows the result which is also visualized in Figure 6 in-which the distribution of the records is also presented as percentage for each cluster separately.

Table 3: The result produced by WEKA Hierarchical Clustering algorithm

Clusters	**Records**
Cluster 1	2 (2%)
Cluster 2	8 (9%)
Cluster 3	37 (40%)
Cluster 4	12 (13%)
Cluster 5	3 (3%)
Cluster 6	4 (4%)
Cluster 7	27 (29%)

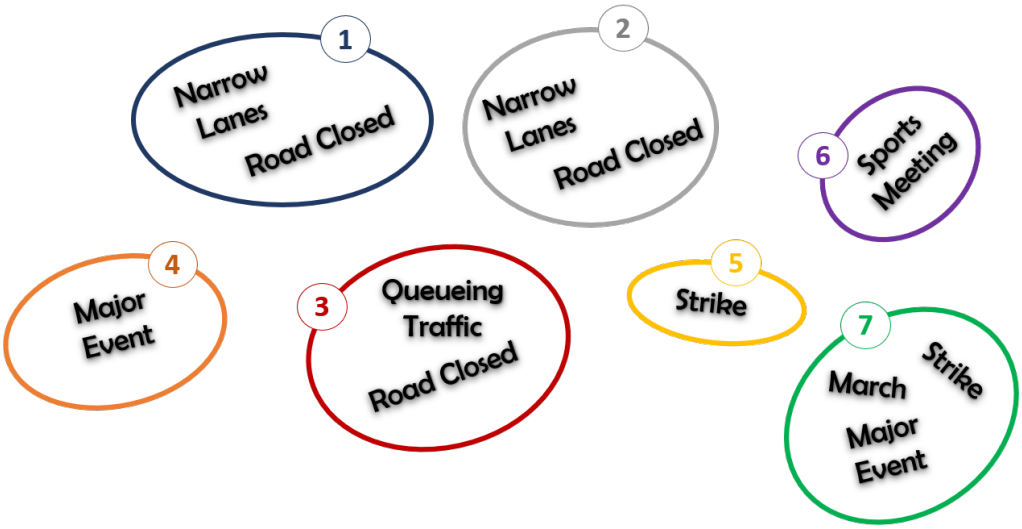


Figure 6: WEKA Text Clustering Results.

Furthermore, we observed that both real-time and batch style versions of ProLoD produced consistent and representable clusters that will assist in exploring data, discovering insights, and supporting predictive analytics when data distribution is observed. Table 4 shows the result which is also shown in Figure 7 in-which the distribution of the records is also presented as percentage for each cluster separately.

Table 4: The result produced by RePLoD Hierarchical Clustering algorithm

Clusters	**Records and Clusters**
Cluster 1	10 (11%)
Cluster 2	31 (33%)
Cluster 3	Cluster 1 and Cluster 2
Cluster 4	50 (54%)
Cluster 5	Cluster 3 and Cluster 4
Cluster 6	2 (2%)
Cluster 7	Cluster 5 and Cluster 6

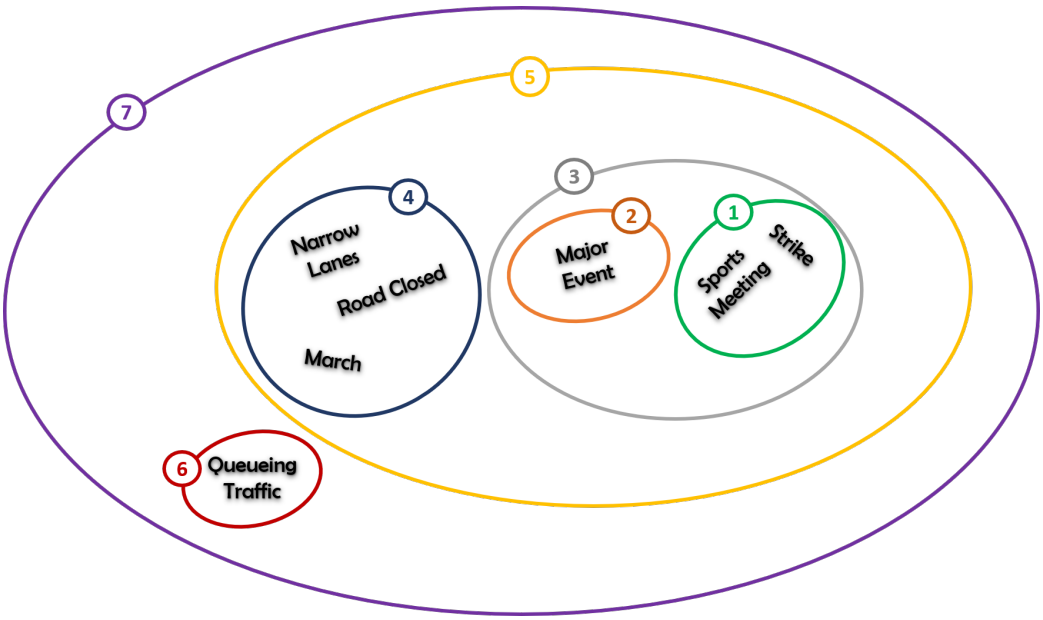


Figure 7: RePLoD Text Clustering Results.

According to our analysis, any labeling process using any of the attributes from the dataset will produce results that are representable and understandable (by representable we meant a reasonable number of clusters produced by the algorithm). Many of the clusters produced by WEKA intersect with other clusters while treated independently. For example cluster 1 and cluster 2 do not need to be represented as two different clusters and cluster 4 and cluster 7 intersect due to the fact that they have a common value called *Major Event* of the attribute *type of event*. According to our observation, the separation of clusters (if applicable) is more effective than clusters with intersection over common values such as *Major Events*.

Some clusters are misclassified and not representable, according to our observation. In Figure 6 some clusters such as *strike*, *sports meeting* etc. do not represent meaningful result. On the other hand, in Figure 7 clusters are more representative in terms of meaningfulness. For instance, an interpretation of data over the *type of event* attribute will produce clusters containing similar records. Similarly, an interpretation of data over the *public event type* and *network management type* attributes will produce groups separating public events (e.g. Activities) from network management events (e.g. narrow lanes, road closed). Such simple and comprehensive visualization will make anal-

ysis readable for the experts. In addition, automated systems can reap the benefits from categorizing data before applying analytics.

Discussion

In this work we considered two quality attributes regarding our solution: *performance* of the system and *accuracy* of our results that is the number of clusters produced by our algorithms. Our observation reveals that the clustering process in RePLoD (which we proposed in this paper for realtime clustering) consume a significant amount of time which was essentially unexpected. We found that the time consumption increases due to the need of continuous listening to new data being fetched. Another reason was the need for applying the clustering process all over again each time a new record is fetched in order to deal with the evolution characteristic of clusters in real time.

Furthermore, the clusters that were produced during our experiments both in RePLoD and ProLoD are better than WEKA's hierarchical clustering algorithm. The results obtained by WEKA were a set of seven clusters placing common records (e.g. cluster 1 and cluster 2) without any meaningful insight; nevertheless, the records corresponding to *Road Closed* are found in three different clusters. On the other hand, our clustering results showed that activities events (major event, sports meeting, and strike) were clustered together, network management events (road closed and narrow lanes) created a cluster, and traffic events (queueing traffic) created a cluster. Therefore, predictive analytics can be applied to such self-explanatory clusters instead of data points.

8. Conclusion

In this paper, we presented a framework called IBRIDIA for processing logistics data. These data are stemming from various internal and external sources within the logistics domain. The proposed framework integrates several data preparation and processing functions. The output of such functions is a clustered dataset. Afterward, the analytical engine exploits this dataset to perform predictive analytics. In order to develop our data processing model, we studied different machine learning algorithms; eventually, we choose Johnson's hierarchical clustering algorithm. However, we modified the algorithm to become a stream clustering algorithm that supports

incremental grouping of text messages according to their similar characteristics directly on the go. We presented the solution architecture of IBRIDIA which consists of components for performing data preparation tasks such as collection, filtering, cleaning, integration, and data storage. It also includes components for processing data and applying some clustering techniques to get self-explanatory groups of data supports predictive analytics for forecasting delivery delay.

Furthermore, we studied various tools to implement our algorithm. We implemented IBRIDIA by adopting both real-time and batch style computational models (ProLoD and RePLoD). Finally, we evaluated our solution with the most widely used machine learning library called WEKA. We reported and discussed the evaluation results. According to our observation, IBRIDIA's hierarchical clustering produced better representative results. However, the real-time version's performance of ProLoD was worse than the batch style - which was due to the centralized environment of experiment consisting of only one node. Thus, we presented both ProLoD as well as our framework RePLoD for real-time collection and processing of logistics data that are stemming from sources such as social media. We tested the framework over a simple use case. Several works are planned for future work.

We plan to introduce time window to the algorithm so that we do not have to store the data more than a specific time limit in order to be processed; this effervescent data store is more suitable for realtime architectures. Using weighted distances other than binary hamming distance with the aim of representing important data point more significantly. Also, we plan to introduce the ability to deal with numerical data as categories because in this article we focused on text data whereas in the big picture numerical data is essential. RePLoD needs to be tested over a real-world use case which will be done in near future. RePLoD will be extended in two aspects: developing a predictive analytics and a prescriptive analytics model to suggest an optimized route plan in order to prevent delay.

9. References

[1] G. Braun, IoT in the Supply Chain - Inbound Logistics, `http://www.inboundlogistics.com/cms/article/iot-in-the-supply-chain/`, 2016. [Online; accessed 04-January-2018].

[2] J. Macaulay, M. Kückelhaus, Internet of Things in Logis-

tics, https://delivering-tomorrow.de/wp-content/uploads/2015/08/DHLTrendReport_Internet_of_things.pdf, 2015.

[3] M. AlShaer, Y. Taher, R. Haque, M.-S. Hacid, M. Dbouk, Prolod: An efficient framework for processing logistics data, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, pp. 698–715.

[4] I. Taxidou, P. Fischer, Realtime analysis of information diffusion in social media, Proc. VLDB Endow. 6 (2013) 1416–1421.

[5] G. Nieva, Integrating heterogeneous data, 2016.

[6] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Generation Computer Systems 56 (2016) 684–700.

[7] E. Mezghani, E. Exposito, K. Drira, M. Da Silveira, C. Pruski, A semantic big data platform for integrating heterogeneous wearable data in healthcare, Journal of medical systems 39 (2015) 185.

[8] R. Power, B. Robinson, D. Ratcliffe, Finding fires with twitter, in: Australasian language technology association workshop, volume 80.

[9] M. Heutger, M. Kückelhaus, K. Zeiler, D. Niezgoda, G. Chung, Self-driving vehicles in logistics, 2014.

[10] LogisticsPlus, Top Logistics Challenges Facing Shippers Today., http://www.logisticsplus.net/top-logistics-challenges-facingshippers-today/, 2015. [Online; accessed 30-March-2016].

[11] D. Morales, Logistics & Transportation Executives Facing Todays Challenges, Seek Solutions Well into the Future., http://www.stantonchase.com/logistics-transportion-executives-\facing-todays-challenges-seek-solutions-well-into-the-future/, 2015. [Online; accessed 30-March-2017].

[12] F. Němec, Distinguished problems of logistics, in: First International Symposium on Business Administration, Challenges For Business Administrators in the new Milleninium, pp. 1–3.

[13] J. Nwaubani, Business intelligence and logistics, in: Proceedings of the 1st Olympus International Conference on Supply Chain, Katerini, Greece.

[14] H. Chen, R. H. Chiang, V. C. Storey, Business intelligence and analytics: From big data to big impact., MIS quarterly 36 (2012).

[15] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, W. M. Van Der Aalst, The prom framework: A new era in process mining tool support., in: ICATPN, volume 3536, Springer, pp. 444–454.

[16] X. L. Dong, D. Srivastava, Big data integration, in: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE, pp. 1245–1248.

[17] Z. T. Birch, an efficient data clustering method for very large databases, in: Proceedings of the 1996 ACM SIGMOD international conference on Management of data (SIGMOD'96).-New York: ACM, pp. 103–114.

[18] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, IEEE transactions on knowledge and data engineering 15 (2003) 515–528.

[19] S. Guha, N. Mishra, Clustering data streams, in: Data Stream Management, Springer, 2016, pp. 169–187.

[20] B. Babcock, M. Datar, R. Motwani, L. O'Callaghan, Maintaining variance and k-medians over data stream windows, in: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, pp. 234–243.

[21] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th international conference on Very large data bases-Volume 29, VLDB Endowment, pp. 81–92.

[22] C. C. Aggarwal, J. Han, J. Wang, S. Y. Philip, On high dimensional projected clustering of data streams, Data Mining and Knowledge Discovery 10 (2005) 251–273.

[23] F. Cao, M. Estert, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: Proceedings of the 2006 SIAM international conference on data mining, SIAM, pp. 328–339.

[24] Y. Chen, L. Tu, Density-based clustering for real-time stream data, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 133–142.

[25] M. Khalilian, F. Z. Boroujeni, N. Mustapha, M. N. Sulaiman, K-means divide and conquer clustering, in: Computer and Automation Engineering, 2009. ICCAE'09. International Conference on, IEEE, pp. 306–309.

[26] K. Udommanetanakit, T. Rakthanmanon, K. Waiyamai, E-stream: Evolution-based technique for stream clustering, Advanced Data Mining and Applications (2007) 605–615.

[27] C. Mahobiya, M. Kumar, Performance comparison of two streaming data clustering algorithms (2014).

[28] T. W. Cheng, D. B. Goldgof, L. O. Hall, Fast fuzzy clustering, Fuzzy sets and systems 93 (1998) 49–56.

[29] R. J. Hathaway, J. C. Bezdek, Extending fuzzy and probabilistic clustering to very large data sets, Computational Statistics & Data Analysis 51 (2006) 215–234.

[30] R. L. Cannon, J. V. Dave, J. C. Bezdek, Efficient implementation of the fuzzy c-means clustering algorithms, IEEE transactions on pattern analysis and machine intelligence (1986) 248–255.

[31] C.-H. Wu, S.-J. Horng, Y.-W. Chen, W.-Y. Lee, Designing scalable and efficient parallel clustering algorithms on arrays with reconfigurable optical buses, Image and Vision Computing 18 (2000) 1033–1043.

[32] M. B. Al-Zoubi, A. Hudaib, B. Al-Shboul, A fast fuzzy clustering algorithm, in: Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, volume 3, pp. 28–32.

[33] V. Vryniotis, DatumBox machine learning framework, `http://www.datumbox.com/`, 2013-2016. [Online; accessed 04-January-2018].

[34] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, The Journal of Machine Learning Research 17 (2016) 1235–1241.

[35] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, V. S. Tseng, Spmf: a java open-source pattern mining library, The Journal of Machine Learning Research 15 (2014) 3389–3393.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: an update, ACM SIGKDD explorations newsletter 11 (2009) 10–18.

[37] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, Journal of Machine Learning Research 11 (2010) 1601–1604.

[38] M. A. Waller, S. E. Fawcett, Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management, Journal of Business Logistics 34 (2013) 77–84.

[39] Z. Bi, D. Cochran, Big data analytics with applications, Journal of Management Analytics 1 (2014) 249–265.

[40] Q. Gong, Y. Yang, S. Wang, Information and decision-making delays in mrp, kanban, and conwip, International Journal of Production Economics 156 (2014) 208–213.

[41] R. S. Michalski, J. G. Carbonell, T. M. Mitchell, Machine learning: An artificial intelligence approach, Springer Science & Business Media, 2013.

[42] V. Estivill-Castro, Why so many clustering algorithms: a position paper, ACM SIGKDD explorations newsletter 4 (2002) 65–75.

[43] K. J. Cios, W. Pedrycz, R. W. Swiniarski, Data mining and knowledge discovery, in: Data mining methods for knowledge discovery, Springer, 1998, pp. 1–26.

[44] T. Galili, dendextend: an r package for visualizing, adjusting and comparing trees of hierarchical clustering, Bioinformatics 31 (2015) 3718–3720.

[45] B. Everitt, S. Landau, M. Leese, Cluster analysis.: Arnold, a member of the hodder headline group, 2001.

[46] R. L. Graham, P. Hell, On the history of the minimum spanning tree problem, Annals of the History of Computing 7 (1985) 43–57.