An Efficient Framework for Processing and Analyzing Unstructured Text to Discover Delivery Delay and Optimization of Route Planning in Realtime

Mohammad Alshaer

N°d'ordre NNT : **2019LYSE1105**

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
L'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
Informatique et Mathématiques de Lyon (InfoMaths)

Spécialité de doctorat :
Discipline : Informatique

Soutenue publiquement/à huis clos le 13/09/2019, par :
Mohammad ALSHAER

An Efficient Framework for Processing and Analyzing Unstructured Text to Discover Delivery Delay and Optimization of Route Planning in Realtime

---

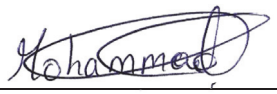
Devant le jury composé de :

Mme. GRIGORI Daniela, Professeure, L'université Paris-Dauphine	Rapporteure
Mme. BOUZEGHOUB Amel, Professeure, Télécom Sud-Paris	Rapporteure
M. MEPHU NGUIFO Engelbert, Professeur, Université d'Auvergne	Président du jury
M. BENABDESLEM Khalid, McF-HDR, Université Lyon 1	Examinateur
Mme. NAJA Hala, Professeure, Université Libanaise	Examinatrice
M. HACID Mohand-Saïd, Professeur, Université Lyon 1	Directeur de thèse
M. DBOUK Mohamed, Professeur, Université Libanaise	Directeur de thèse
M. TAHER Yehia, McF, Université de Versailles	Co-Encadrant de thèse
M. HAQUE AKM Rafiqul, Directeur de recherche, Cognitus	Invité

Declaration of Authorship

I, Mohammad ALSHAER, declare that this thesis titled, "An Efficient Framework for Processing and Analyzing Unstructured Text to Discover Delivery Delay and Optimization of Route Planning in Realtime" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 13/09/2019

"Your present circumstances don't determine where you can go; they merely determine where you start."

Nido Qubein

Abstract

THESIS TITLE

An Efficient Framework for Processing and Analyzing Unstructured Text to Discover Delivery Delay and Optimization of Route Planning in Realtime

by Mohammad ALSHAER

Keywords: Realtime Processing, Clustering, Big Data, Internet of Things, Logistics, Hierarchical Clustering Algorithm, Unstructured Data, Text Analytics.

Internet of Things (IoT) is leading to a paradigm shift within the logistics industry. The advent of IoT has been changing the logistics service management ecosystem. Logistics services providers today use sensor technologies such as GPS or telemetry to collect data in realtime while the delivery is in progress. The realtime collection of data enables the service providers to track and manage their shipment process efficiently. The key advantage of realtime data collection is that it enables logistics service providers to act proactively to prevent outcomes such as delivery delay caused by unexpected/unknown events. Furthermore, the providers today tend to use data stemming from external sources such as Twitter, Facebook, and Waze. Because, these sources provide critical information about events such as traffic, accidents, and natural disasters. Data from such external sources enrich the dataset and add value in analysis. Besides, collecting them in real-time provides an opportunity to use the data for on-the-fly analysis and prevent unexpected outcomes (e.g., such as delivery delay) at run-time.

However, data are collected raw which needs to be processed for effective analysis. Collecting and processing data in real-time is an enormous challenge. The main reason is that data are stemming from heterogeneous sources with a huge speed. The high-speed and data variety fosters challenges to perform complex processing operations such as cleansing, filtering, handling incorrect data, etc. The variety of data – structured, semi-structured, and unstructured – promotes challenges in processing data both in batch-style

and real-time. Different types of data may require performing operations in different techniques. A technical framework that enables the processing of heterogeneous data is heavily challenging and not currently available. In addition, performing data processing operations in real-time is heavily challenging; efficient techniques are required to carry out the operations with high-speed data, which cannot be done using conventional logistics information systems. Therefore, in order to exploit Big Data in logistics service processes, an efficient solution for collecting and processing data in both realtime and batch style is critically important.

In this thesis, we developed and experimented with two data processing solutions: SANA and IBRIDIA. SANA is built on Multinomial Naïve Bayes classifier whereas IBRIDIA relies on Johnson's hierarchical clustering (HCL) algorithm which is hybrid technology that enables data collection and processing in batch style and realtime. SANA is a service-based solution which deals with unstructured data. It serves as a multi-purpose system to extract the relevant events including the context of the event (such as place, location, time, etc.). In addition, it can be used to perform text analysis over the targeted events. IBRIDIA was designed to process unknown data stemming from external sources and cluster them on-the-fly in order to gain knowledge/understanding of data which assists in extracting events that may lead to delivery delay. According to our experiments, both of these approaches show a unique ability to process logistics data. However, SANA is found more promising since the underlying technology (Naïve Bayes classifier) outperformed IBRIDIA from performance measuring perspectives. It is clearly said that SANA was meant to generate a graph knowledge from the events collected immediately in realtime without any need to wait, thus reaching maximum benefit from these events. Whereas, IBRIDIA has an important influence within the logistics domain for identifying the most influential category of events that are affecting the delivery. Unfortunately, in IBRIRDIA, we should wait for a minimum number of events to arrive and always we have a cold start. Due to the fact that we are interested in re-optimizing the route on the fly, we adopted SANA as our data processing framework. We implemented a route optimization application to demonstrate how our solution is used in extracting information of events that may lead to delivery delay and how the routes can be optimized to prevent the delay.

Résumé

TITRE DE THÈSE

Un framework efficace pour le traitement et l'analyse des textes non structurés afin de découvrir les retards de livraison et d'optimiser la planification de routes en temps réel

par Mohammad ALSHAER

Mots clés: Traitement en temps réel, Clustering, Big Data, Internet des objets, Logistique, Algorithme de clustering hiérarchique, Données non structurées, Analyse de texte.

L'Internet des objets, ou IdO (en anglais Internet of Things, ou IoT) conduit à un changement de paradigme du secteur de la logistique. L'avènement de l'IoT a modifié l'écosystème de la gestion des services logistiques. Les fournisseurs de services logistiques utilisent aujourd'hui des technologies de capteurs telles que le GPS ou la télémétrie pour collecter des données en temps réel pendant la livraison. La collecte en temps réel des données permet aux fournisseurs de services de suivre et de gérer efficacement leur processus d'expédition. Le principal avantage de la collecte de données en temps réel est qu'il permet aux fournisseurs de services logistiques d'agir de manière proactive pour éviter des conséquences telles que des retards de livraison dus à des événements imprévus ou inconnus. De plus, les fournisseurs ont aujourd'hui tendance à utiliser des données provenant de sources externes telles que Twitter, Facebook et Waze, parce que ces sources fournissent des informations critiques sur des événements tels que le trafic, les accidents et les catastrophes naturelles. Les données provenant de ces sources externes enrichissent l'ensemble de données et apportent une valeur ajoutée à l'analyse. De plus, leur collecte en temps réel permet d'utiliser les données pour une analyse en temps réel et de prévenir des résultats inattendus (tels que le délai de livraison, par exemple) au moment de l'exécution.

Cependant, les données collectées sont brutes et doivent être traitées pour une analyse efficace. La collecte et le traitement des données en temps réel constituent un énorme défi. La raison principale est que les données proviennent de sources hétérogènes avec une vitesse énorme. La grande vitesse et

la variété des données entraînent des défis pour effectuer des opérations de traitement complexes telles que le nettoyage, le filtrage, le traitement de données incorrectes, etc. La diversité des données - structurées, semi-structurées et non structurées - favorise les défis dans le traitement des données à la fois en mode batch et en temps réel. Parce que, différentes techniques peuvent nécessiter des opérations sur différents types de données. Une structure technique permettant de traiter des données hétérogènes est très difficile et n'est pas disponible actuellement. En outre, l'exécution d'opérations de traitement de données en temps réel est très difficile ; des techniques efficaces sont nécessaires pour effectuer les opérations avec des données à haut débit, ce qui ne peut être fait en utilisant des systèmes d'information logistiques conventionnels. Par conséquent, pour exploiter le Big Data dans les processus de services logistiques, une solution efficace pour la collecte et le traitement des données en temps réel et en mode batch est essentielle.

Dans cette thèse, nous avons développé et expérimenté deux méthodes pour le traitement des données: SANA et IBRIDIA. SANA est basée sur un classificateur multinomial Naïve Bayes, tandis qu'IBRIDIA s'appuie sur l'algorithme de classification hiérarchique (CLH) de Johnson, qui est une technologie hybride permettant la collecte et le traitement de données par lots et en temps réel. SANA est une solution de service qui traite les données non structurées. Cette méthode sert de système polyvalent pour extraire les événements pertinents, y compris le contexte (tel que le lieu, l'emplacement, l'heure, etc.). En outre, il peut être utilisé pour effectuer une analyse de texte sur les événements ciblés. IBRIDIA a été conçu pour traiter des données inconnues provenant de sources externes et les regrouper en temps réel afin d'acquérir une connaissance / compréhension des données permettant d'extraire des événements pouvant entraîner un retard de livraison. Selon nos expériences, ces deux approches montrent une capacité unique à traiter des données logistiques. Cependant, SANA semble plus prometteur puisque la technologie sous-jacente (classificateur Naïve Bayes) a surpassé IBRIDIA du point de vue performance. Il est clairement indiqué que SANA était censé générer une connaissance graphique des événements collectés immédiatement en temps réel sans avoir à attendre, permettant ainsi de tirer le meilleur parti de ces événements. IBRIDIA est importante dans le domaine de la logistique pour identifier la catégorie d'événements la plus influente qui affecte la livraison. Malheureusement, pour IBRIRDIA, nous devrions attendre qu'un nombre minimum d'événements se présente et nous avons toujours un démarrage à froid. Étant donné que nous sommes intéressés par la ré-optimisation de

l'itinéraire en temps réel, nous avons adopté SANA comme cadre de traitement de données. Nous avons implémenté une application d'optimisation d'itinéraire afin de démontrer comment notre solution est utilisée pour extraire des informations d'événements susceptibles d'entraîner un retard de livraison et comment les itinéraires peuvent être optimisés pour éviter ce retard.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors Prof. Mohand-saïd HACID (Université Claude Bernard Lyon 1), Prof. Mohamed DBOUK (Université Libanaise), Dr. Yehia TAHER (UVSQ) and Dr. Rafiqul HAQUE (Cognitus) for the continuous support throughout my PhD. work, for their patience, motivation, and immense knowledge. With their deep experience and knowledge, they have guided me carefully to advance in my PhD. Their suggestions and guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my PhD. study.

Secondly, again, my deepest thankfulness goes to my co-supervisors Dr. Yehia TAHER and Dr. Rafiqul HAQUE for their incredible efforts, not just on my PhD. work, but also facilitating many stages of my life. I would like to thank them as their Master and PhD. student for being a great mentors, always knowing what needs to be done in the best way. Even though Dr. Rafiqul was not officially my supervisor, I would like to thank him from the bottom of my heart for the time that he dedicated for me throughout the years of my PhD. and his invaluable suggestions and contributions.

Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Daniela GRIGORI, Prof. Amel BOUZEGHOUB, Prof. Engelbert MEPHU NGUIFO, Prof. Hala NAJA, and Dr. Khalid BENABDESLEM. It is an honor having you as my jury members and having my work validated by you. I truly appreciate their constructive feedback and the time they spare given their busy schedules.

As I was always a member of ADAM team at the University of Versailles, I want to thank all the members of DAVID laboratory and in particular the members of ADAM team and especially, Prof. Karine Zeitouni, Dr. Béatrice Finance, and Dr. Zoubida Kedad. I had the chance to meet them and to get their feedback on several occasions. They helped in facilitating many things for me during my stay in Versailles.

Even though I am studying in France and away from family, but I was enjoying my moments here and for that, I owe several persons. Actually, I could not survive the life here if not for the true support of my dear friends, who were also my lab mates, and even my neighbors. Almost, we lived the whole time together. Therefore, I would like to deliver my purest acknowledgment

to Ali Masri, Raef Mousheimish, Mohammad Moussawi, Mohammad Rihany, Sara Makki, Alaa Zreik, Hassan Chaitou, Ahmad Ktaish, Alexandros Kontarinis, and others, for being true friends and supporters all the way.

Apart from the lab and the research world, I am grateful for having my friends Ahmad, Hassan, Fatima and many others. We shared unforgettable experiences together and will share even much more in the upcoming years.

My deepest gratitude goes to my precious family, who tried to support me and encourage me in every possible way. I am entirely grateful to my father Bilal and my mother Haifa, not to mention my brothers and sisters, for their tremendous moral and emotional support. I am also grateful to my big family and especially my uncle Dr. Adel and my aunts Sahar and Huda for their genuine encouragement. Everything I have reached is because of your continuous support and love.

Mohammad ALSHAER

Table of Contents

List of Figures

List of Tables

Chapter 1

Chapter One: Introduction

1.1 Context and Motivation

With the advent of the Internet of Things (IoT), lately, the operational landscape of the logistics industry is changing. Today, logistics companies (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 and also handling real-time events. Additionally, the connectivity of "things" enables instant communication between devices via Internet [32]. This hyper-connected ecosystem promises far-reaching payoffs for logistics operators, their business customers, and end customers [152]. One of the major advantages of IoT-based ecosystem is that it enables us to connect the logistics sensors with external sensors such as weather sensors and traffic (GPS) sensors etc. Furthermore, IoT enables us to connect 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. This information can be used to carry out some critical analysis such as predictive analysis for forecasting shipment delay or prescriptive analysis to optimize routes to guarantee on-time delivery which increases customer satisfaction and hence guarantees customer retention.

Although many solutions proposed in the last two decades within the 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

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

has various impacts such as, customer churn or cancellation of orders which eventually leads to loss of revenue. Therefore, *timely delivery* is critically important within logistics companies.

A delivery process consists of a set of steps such as *receive delivery request*, *pick up goods*, *deliver orders to the client* etc. Additionally, the delivery process is constrained with different parameters mainly *delivery time* and *delivery location*. The delivery time is considered as a global constraint that is composed of two temporal constraints: *pick up time* and *vehicle routing time* that are considered as local time constraints. Note that the vehicle routing time refers to the time elapsed from pick up location (i.e., warehouse location) to destination of delivery. The values of global constraints delivery time and location are agreed between the product seller and logistics service provider and assigned at the design-time of the process. Very often the logistics service providers confront a challenge in satisfying the delivery time. The challenge is posed mainly by different uncertain events that may not be possible to foresee at design-time. The reason behind this is that most of these events such *heavy traffic*, *accident*, and *natural disaster* occur while the delivery process is running *i.e.*, during delivery of an order or picking up the order from warehouse. These eventually may lead to the failure of in-time delivery of the order.

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 realtime. The realtime prediction of delay enables companies to pro-act such as optimizing route on the fly in (near) realtime. We have investigated the requirements of a realtime system which can perform analysis and predict delay. The core requirements are: ability to collect logistics data in realtime from multiple heterogeneous sensors, 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 quo non*.

1.2 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 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 1.1).

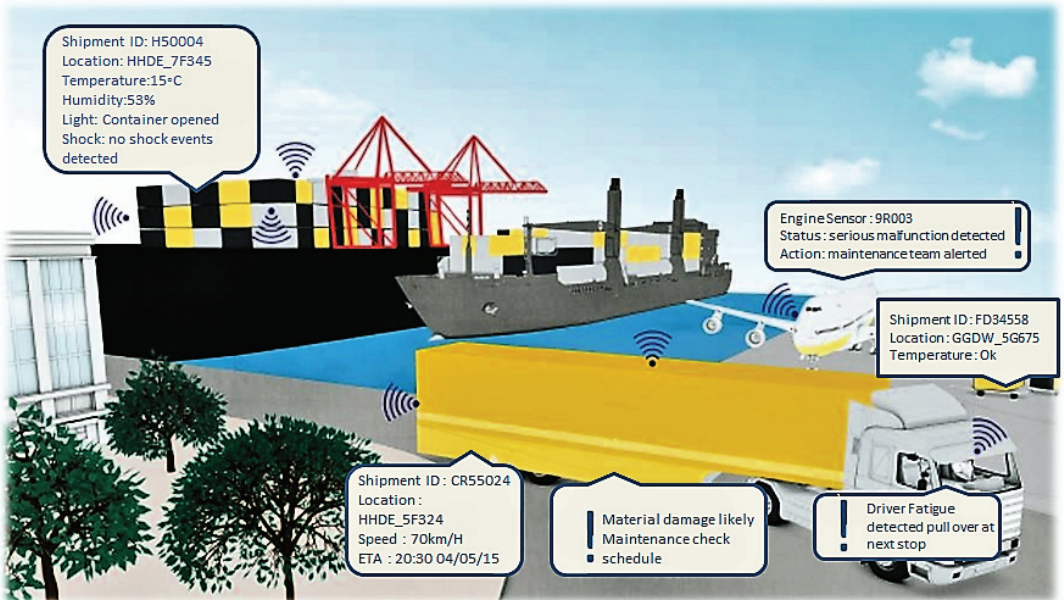


FIGURE 1.1: The Multi-modal Logistics System (Source: [110])

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.

1.3 Problem Description

There are different challenges involved in an integrated mission-critical logistics process. The predominant challenges reported by experts include the following: in-time delivery, cost optimization, efficient management of intermodal transportation, transferring information, Security, and Infrastructure [148, 174, 178].

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 [179] that analyze delay for different combinations of goods, routes, modes and weather conditions.

However, this is a reactive approach that performs an analysis of historical data. In other words, traditional business intelligence especially, BI&A 1.0 and BI&A 2.0 use only internal data that stem from different information systems and legacy systems [44].

Also, the process mining tool PRoM [210] – a recent tool for mining business processes – lacks the ability to exploit external data. Consequently, the analytics miss 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 delays 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, seamless integration of unstructured text sourcing from Twitter with structured business process data is not possible by existing logistics solution frameworks. Dong et al. [66], 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.

To sum up, during our study with literature, we seek for an integrated solution which enables to process data for extracting events that may have an impact on delivering goods. Taking this limitation into account, we formulate our research problem as follows:
"*the existing frameworks do not provide adequate techniques or methods that enable to collect and process heterogeneous data in realtime to extract events that drive the delivery constraints specifically the delivery time. Also, the current frameworks re-engineer the delivery plan dynamically which will help in avoiding the delivery delay.*"

1.4 Research Questions

Based on the problems we discovered, we defined three questions that this research will address. We describe these questions as follows:

- **Research Question 1** : This question concerns collecting heterogeneous data in realtime. Logistics data is collected from different data sources. Integrating diverse data (*e.g.*, unstructured text, audio, and video) is a critical data processing problem. Also, data are presented using different technologies such as json, xml, etc. The question is: *How to handle data variety while collecting them in real-time from different sources?*
- **Research Question 2**: This research question concerns realtime processing of data for extracting relevant events. The collected data need to be processed in realtime and the relevant events must be extracted. Most of these data are unstructured (text-based) that need to be processed in realtime which is a non-trivial task. *How to process data and extract events in realtime efficiently?*
- **Research Question 3**: This research question concerns the dynamic optimization of route planning. Route optimization is an NP-hard problem with no unique solution that can be used. *The major question is how to optimize the route planning*?

1.5 Objectives

Our objective is to build a framework underlying techniques that enable to: (i) collect, process, and analyse unstructured data that move in high speed,

(ii) extract events that may influence the constraints of delivery in logistics processes especially the delivery time, and (iii) prescribe an optimized dynamic plan to react immediately by choosing the optimal solutions.

We aim to focus on two modules that are critical to attaining our objectives. These modules are described below:

- **Data Processing:** In this phase, we collect the data from social media and sensors in realtime. After collecting the data, we process it. The processing is done by integrating the different data collected from the different data sources and then extract the relevant events. We are here defining a new formatted template for the events in order to unify the different data presentation. We translate all the different unstructured text data into meaningful events.
- **Data Analysis:** During this phase, we need to perform the dynamic optimization of route planning. In other ways, re-engineering the delivery plan dynamically to optimize the plan which will help in avoiding the delivery delay.

1.6 Contributions

Following our objectives, we find that there is no standard processing framework for processing the coming data in realtime with the ability to extract the affecting events occurring at the moment. That conclusion was a good motivation to decide on going for an experimental study to find the most adequate processing system to deal with the logistics in-time delivery problem. In this thesis, we developed two data processing solutions: SANA and IBRIDIA for the experimental purpose to choose the right technology for collecting and processing data in realtime.

Our contributions can be presented as follows:

- **SANA:**

  We first tried to tackle a framework that is general-purpose i.e. can deal with any type of data and eventually extract relevant information. We designed this framework to be independent of the domain of interest (e.g. logistics domain). This framework was built as a social media analyzer which we called SANA that can do realtime analytics to understand the sentiment of the users. SANA was published in ICSOC

Workshops 2016 and its refined model in OTM 2016. With SANA, we were able to understand the users' satisfaction on current logistics companies and how they interact to any delivery. This analysis was of major importance for specifying the real problems in logistics and figuring out how the delivery is one of the problems that really affect the satisfaction of customers. From SANA, we were able to recognize more effectively the delivery problem. Besides, we found that it can play the processing role very smoothly due to its different integrated modules not just as a sentiment analyzer, but also as a data processing engine to extract the relevant events in order to re-design the routing of the delivery plan to avoid any probable delay.

The main characteristics that distinguish SANA from existing processing systems include the following: (i) it performs analysis with the entire text of an event, instead of its small fragment. Thus, the semantic along with context (e.g., location, organization, etc.) of the event could be understood and also visualized more comprehensively than the conventional text analytics systems; (ii) SANA is a context-aware solution. Furthermore, SANA performs a lightweight preprocessing with raw data collected from external data sources, e.g., Twitter. None of the existing solutions offer preprocessing functionality. In addition, SANA has graph storage which enables visualizing the results as knowledge graphs that present a comprehensive and detailed view resulted from classification. Additionally, SANA offers a graph-based query interface that enables querying the results to extract finer-grained knowledge. This cannot be done by current solutions. The multi-layered architecture of SANA consists of various components, which are briefly described in the following.

SANA performs a light-weight preprocessing with incoming events. The purposes of preprocessing are two-fold: formulating context of events, and prune unnecessary strings. The former is critical for context extraction and the latter is important for saving memory space. For context formulation, SANA extracts keywords using the n-gram model which is a contiguous sequence of n-character or n-word slice of longer strings or texts. The values of n vary depending on the number of characters or words the users want to extract in each extraction operation. The corpus-based is a widely known approach and used for extracting words, however, we choose n-gram model because the corpus-based

approach is not suitable for extracting keywords realtime. The n-gram model is a probability function, $\mathcal{P}(w_n \mid w_1^{n-1}) \approx \mathcal{P}(w_n \mid w_{n-N+1}^{n-1})$ where, w_1, w^{n-1}, w^n are sequence of words. SANA extracts a pair of words (which are the keywords). The model approximates the probability of a word given all the previous words $\mathcal{P}(w_n \mid w_1^{n-1})$ by using the conditional probability $\mathcal{P}(w_n \mid w_{n-N+1}^{n1})$. SANA formulates contexts using extracted keyword. For instance, consider a tweet *I bought a Christine Davis perfume from Paris, which I found could not satisfy my expectation*, SANA extracts words and formulates contexts by pairing words including *perfume* and *Paris*. In the next step, SANA prunes all unnecessary strings. In order to perform pruning it relies on a training dataset that consists of a set of seed words.

- **IBRIDIA:**

  According to our investigation, well-processed data is of critical importance for efficient analysis. However, existing data processing solutions (e.g., techniques or algorithms) are not adequately efficient to process data in realtime. Our objective at this phase was to address the variety and velocity challenges. We started targeting the variety challenge by developing a module called ProLoD for collecting and processing data in batch style mode which was published in the OTM 2017 international conference. We extend it further by developing a module called RePLoD for collecting and processing data in realtime mode to target the velocity challenge. ProLoD and RePLoD were the main modules of our solution which we called IBRIDIA that was published in Future Generation Computer Systems Journal. Our solution relies on the hierarchical clustering algorithm for processing data in both batch style and realtime.

  IBRIDIA relies on the data processing model which we developed. While choosing the appropriate technique for developing the model, we considered the nature of data and operation styles. Hierarchical clustering was missing in the state of art for realtime analytics. We contributed to this algorithm by adding an extension to be able to cluster the data streams that are arriving in realtime by using an incremental method and since 80% of the data existing today are unstructured data, we concentrate on clustering over text data using the hamming distance for measuring the similarity.

We used this processing algorithm to help us determine the events that have an impact over the delay. We tested our approach and it generated good results, but the limitation was related to the memory bottleneck. As the time passes, the clusters evolve getting more points or creating new clusters and these clusters are based on $(n \times n)$ matrix in memory where all the records and clusters are compared with each other in the memory which leads into a memory bottleneck.

- **Dynamic Route Optimization:**

  Using the adopted data processing framework, we are able to extract the relevant events that may influence the delivery constraints (*e.g.*, vehicle routing time) from the ingested data in realtime. These events are adopted by our dynamic route planning algorithm in order to eliminate or mitigate the delay by prescribing a new optimized plan. This routing algorithm is based on a meta-heuristic approach. It consists of a large neighborhood search that combines elements of simulated annealing and threshold-accepting algorithms. This algorithm is mainly composed of two steps *ruin and create* that are discussed further in section 4.

  Our algorithm will first fill all the possible routes between two different locations. Then, we check if any of the locations mentioned in the processed events fall within the bounding box of the different routes. After that, we check which one of these routes is to be less critically affected by the unexpected events to be selected as the best route by the algorithm. When we detect the best route, we use its distance as the minimum distance between the two provided location and we did this to all locations that need to be visited to find the optimal route. This algorithm was developed to take into consideration the constraints to generate a new route based on the shortest distance and deal with the order in which the delivery plan will consider visiting the customers.

1.7 Research Scope

The scope of this research is more related to the processing and prescriptive analytics research areas. In this scope, we covered the following:

- Processing realtime data: This part was the main focus of our research due to its importance for the analysis. If we do not have the right data, analysis will not show any meaningful information and insights. Due to this fact, we covered different stages of data processing which include parsing, cleaning, filtering, transforming them into well-formatted events sharing the same template and some sort of machine learning algorithms to get the best out of these data.
- Extracting relevant events in realtime: We extract the events and label them as relevant depending on their impact on the delay throughout the route.

The scope of this research did not cover the following research areas:

- Data Storage: We used different data storage technologies, however, it is out of the scope of this research. We could not contribute to the efficiency of the storage and its performance in terms of read and write operations.
- Data Visualization: Despite the fact that we visualize some of the events, it is yet out of the scope as well due to the fact that it needs more research to have the correct visualization which depends on the domain-specific business to show the meaningful results.
- Data Security: We should mention clearly that data security is out of the scope of this research.

1.8 Thesis Structure

This thesis consists of five chapters apart from the introductory Chapter 1.

In Chapter 2, we present the different methods, techniques, and technologies that fall within the scope of our proposed solutions from data collection to data processing and data analysis. This discussion shows the gap between the existing approaches and our approach. An elaborate review of the state of the art shows that it is still missing one framework that can integrate all these different techniques and methods from the different domains.

In Chapter 3, we decided to move forward conducting an experimental study between two different frameworks based on different components. We developed the two frameworks called **SANA** and **IBRIDIA** for processing the logistics data in realtime. SANA uses n-gram with Multinomial Naive Bayes algorithm and context extraction from the information retrieval domain, whereas, IBRIDIA was more focused on an incremental hierarchical clustering algorithm which was missing in the state of the art. In addition, we did some experiments over these two different frameworks and shows that **SANA** and **IBRIDIA** are quite of a different taste. Each has its own taste to extract insight from the data. **SANA** focuses more on the broader area of applications such as customer churn, extracting relevant events that might affect the delivery, etc. On the other hand, **IBRIDIA** was trying to show value from the data that even human being cannot notice easily such as which events can be grouped together, which events are similar to give same sense. This idea of using **IBRIDIA** in some nuclear plants can result in grouping the different events coming immediately into different categories such as critical, non-critical, suspicious, etc.

In Chapter 4, we decided to validate that we did the right choice and the selected framework can provide us with all the required details to do the route planning and the re-routing of the delivery process. We investigated several algorithms in the literature and we discovered that we can enhance the measuring distance function of the preferred algorithm. Thus, we decided that we need to extend the selected framework by a new component that is responsible for planning the whole delivery route and it must be able to adapt the enriched events generated by the selected processing framework to do the route optimization.

We end with Chapter 5 where we sum up our contributions and show how this framework proposes a reliable solution for an improved logistics management system in its different components, specifically the delivery. We discuss the open problems in each solution and show possibilities for future improvements.

Chapter 2

Chapter Two: State of the Art

2.1 Introduction

As mentioned in chapter 1, we tried to address a fully operational system for facilitating the data-driven logistics process by considering any event that might lead to a delivery delay. Conventional logistics systems have several limitations in terms of dealing with unexpected events and make the companies suffer from major losses due to the delay in the delivery that is not handled properly. Nowadays, sensors and social media generate a massive amount of data and some of them concern the different events occurring at the moment including traffic, accidents, weather, parking, terrorist attacks, etc. By integrating the events from the ocean of data, i.e., the world wide web with the internal data from companies we are able to provide the relevant information in-time for analysis.

In order to handle these data properly, we need to address different challenges in the different phases of the system such as data collection, data processing, and data analysis. Collecting data from heterogeneous data sources represents a real challenge from scientific perspectives due to its high complexity. Besides, data velocity creates a bottleneck for conventional systems. Determining the convenient way to handle the coming data is still to be studied extensively in the state of the art. Data processing is yet, another challenging phase that needs to process the coming data in realtime before being stored to any hard drive. Processing data on the fly needs special processing mechanisms that depend on the main memory rather than secondary storage.

2.2 Data Collection Tools and Techniques

2.2.1 Sensor Data Collection

Much research on sensor data and smart data ingestion and fusion have already been studied. Llinas et al. reported an explanation on how to do data ingestion and a basis for sensor ingestion and fusion for future study and research [147]. In order to tackle sensor data ingestion, many frameworks have been proposed; in that sense, we consider the work of Lee et al. in which they presented a peer-to-peer collaboration framework for multi-sensor data fusion in resource-rich radar networks [118]. Most of these frameworks considered exchanging the data among multiple sensors. Among different devices, data cannot be exchanged like in simple sensors. Thus, two types of sensor data processing architectures were discussed by Dolui et al., which are, on-device and on-server data processing architectures [65]. The industrial field adopts mainly the on-server data processing architecture for managing smart devices and products.

Lately, multi-sensor data fusion has received significant importance for both logistics and non-logistics applications. In order to improve the accuracy and the inference of using only one sensor, they introduce data fusion techniques for integrating data from multiple sensors and data from associated databases [216, 217, 105, 129].

While the idea of data fusion is not new, the development of new sensors, advanced processing techniques, and improved processing equipment make the continuous fusion of data incrementally possible [104, 146]. Nowadays, there exist multiple devices for monitoring, tracking and detecting what is going on in the delivery network. Some of these sensors are used for location tracking for example like Global Positioning System (GPS). GPS can collect traffic data in a more efficient way through location tracking. GPS represents an important data source for data collection, especially, if integrated with a geographic information system (GIS) or other map displaying technologies. The data collected from GPS can be utilized to address many traffic issues, such as travel mode detection [220], travel delay measurement [15], and traffic monitoring [109]. Besides GPS sensors, there exist many other sensors that can be used to collect vehicle speeds, vehicle density, traffic flows, and trip times. Some other types of sensors are called on-road sensors such as infrared and microwave detectors which are enhanced to obtain, compute and

transfer traffic data [150].

In the early 1970s, the symbolic processing was found providing an impetus to artificial intelligence [87]. In the same way, recent evolution in computing and sensing has facilitated the ability to emulate in both hardware and software besides the natural data fusion capabilities of organisms through detection and monitoring. Recently, data fusion systems are used for different applications including target tracking, automated recognition of targets, and limited reasoning applications.

Applications for multi-sensor data fusion are spread across different sectors. These applications include monitoring of manufacturing processes, condition-based maintenance of complex machinery, robotics [1], and medical applications. Techniques to integrate or fuse data are discovered from the existence of a diverse set of more traditional disciplines including: digital signal processing, statistical estimation, control theory, artificial intelligence, and classic numerical methods [221, 209, 125]

The fusion of multi-sensor data has significant advantages over using single source data. Combining same-source data can increase the statistical estimation obtained through redundant observations. On the other hand, combining multiple types of sensors data may increase the accuracy with which, an object can be observed, identified and characterized. As an example, a radar can estimates the aircraft's range, but cannot identify the angular direction of the aircraft. However, using an infrared imaging sensor can accurately identify the aircraft's angular direction but it cannot measure the range. If the observations of these two sensors were correctly associated, then the fusion of these sensors data would imply: estimating the location of the aircraft. This produced more accurate results as shown in the fused location estimate. In the same way, identifying multiple object's attributes based on different observations can help in detecting the identity of the object. For example, some studies show that bats use a combination of factors such as size, texture (based on the acoustic signature and kinematic behavior) to identify their prey.

2.2.2 Structured Data Collection

Structured data is also known as traditional data. This type of data is usually stored inside a data warehouse such that it can be ready for analysis.

Structured data is mainly characterized as both highly-organized and easy to digest. Thus, has the advantage of easily being entered, stored, queried and analyzed. At one time, because of the high cost and performance limitations of storage, memory and processing, relational databases and spreadsheets using structured data were the only way to effectively manage data[1]. Often, this type of data is managed using Structured Query Language (SQL). SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS). In order to collect structured data, we need to consider the available tools or techniques that serve this purpose. The point is that these tools must be able to collect the structured data in a continuous, asynchronous, realtime or batched style way. The well-known tools that can fulfill the structured data collection role are: Sqoop, NiFi, Gobblin. These tools will be discussed in detail as follows:

- Apache Sqoop[2]: Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases[3]. It supports transferring a huge amount of data incrementally of a single table or a free form SQL query, saved jobs which can be run multiple times to import updates made to a database since the last import. According to the authors of [12], it enables data imports from external data stores and enterprise data warehouse into Hadoop. It ensures fast performance by parallelizing data transfer and utilizes an optimal system. Sqoop runs on a MapReduce framework on Hadoop -MapReduce will be discussed in the following section-. Sqoop still enables different data formats for data imports. For example, the Avro data format can be used for easier importing of data.

- Apache NiFi[4]: Apache NiFi supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. What is interesting about NiFi is that it includes a web-based user interface that provides the user with access to every single detail about the running processes. NiFi's processors are file oriented and schemaless. This means that every record flowing through the system is represented by a FlowFile. It is crucial for the processor to know the content of the data in order to operate on it. It can run in two main modes

[1] https://www.webopedia.com/TERM/S/structured_data.html
[2] https://sqoop.apache.org/
[3] https://www.predictiveanalyticstoday.com/data-ingestion-tools/
[4] https://nifi.apache.org/

which are: standalone and as a distributed cluster using its own built-in clustering system[5]. It is highly configurable, shows data provenance details and designed for extension. Apache NiFi has a couple of built-in processors for extracting database data into NiFi FlowFile. It has three processors for extracting data from relational databases which are ExecuteSQL, QueryDatabaseTable, and GenerateTableFetch[6].

- Apache Gobblin[7]: Gobblin is a universal ingestion framework developed by LinkedIN. It is a flexible framework for extracting, transforming and loading large volumes of data from a variety of data sources, such as databases, rest APIs, FTP/SFTP servers, etc. onto Hadoop. It is capable of handling both ETL and job schedule pretty well. It also supports auto scalability, fault tolerance, data quality assurance, extensibility, and it provides special sources for JDBC. Besides the continuous ingestion from the database into Hadoop, it provides an efficient approach that involves multiple job types for JDBC ingestion [186]. The handling of the updates is through checking the tuples whose modification timestamps are later than the latest time-stamp pulled by the previous run. Gobblin can run in two modes: standalone mode and distributed mode on the cluster (Runs as MapReduce application, or as elastic Cluster on AWS cloud, etc.).

2.2.3 Web Data Crawling

Significant literature work deals with crawling structured data from the Web. As opposed to ordinary Web crawling of textual content, structured data postures unique difficulties on the organization and performance of crawling and indexing tasks [107]. The work in [107] proposes a pipelined crawling and indexing architecture for the Semantic Web. In [166], the authors propose a learning-based, focused crawling strategy for structured data on the Web. The approach uses an online classifier and performs a bandit-based selection process of URLs, i.e. it determines a trade-off between exploration and exploitation based on the page context and by incorporating feedback from already found metadata. The Web Data Commons [167] activity reuses a Web crawl given by Common Crawl to extract structured data.

[5] https://rcgglobalservices.com/the-best-data-ingestion-tools-for-migrating-to-a-hadoop-data-la
[6] https://www.batchiq.com/database-extract-with-nifi.html
[7] https://gobblin.apache.org/

In [51], the authors provide an investigation on the distribution of structured content for information extraction on the Web. They demonstrate that in order to capture a reasonable amount of the data of a domain, it is significant to respect the long tail of Web sites. On a more broad extension, the authors of [16] estimate that a crawler can achieve a huge segment of the Web pages visited by clients with just after three to five stages from the start page.

Web data crawling is one of the main technologies for collecting data about social relationships, user activities and the contents produced and shared by users. Approaches based on the scraping of HTML pages are able to overcome the APIs usage restrictions, even if they are more complicated to design and implement.

According to [75], one of the first attempts to crawl large Online Social Networks was performed by Mislove et al. [170]. In that paper, the authors focused on platforms like Orkut, Flickr, and LiveJournal. In order to perform crawling for such networks, the approach of [170] recommends to iteratively extracts the list of friends of a user that have not yet been visited and to add these related users to the list of users to visit. According to graph theory, this is similar to perform a Breadth-First-Search (BFS) visit the Social Network graph. The user account from which the BFS begins is frequently called seed node; the BFS ends when the entire graph is visited or, alternatively, a stop criterion is met. The BFS is easy to implement and efficient; it produces accurate results if applied on social graphs which can be modeled as unweighted graphs. Therefore, it has been applied in a large number of studies about the topology and structure of Online Social Networks [43, 223, 88, 234, 36].

As observed in [170], BFS is not an adequate algorithm to be used here due to many limitations. First of all, a crawler can get trapped in a strongly connected component of the social graph. Furthermore, a BFS crawler introduces a bias toward high degree vertices: for instance, a study reported in [133] considered a Facebook sample obtained by means of a BFS crawler and observed that the calculated average degree of vertices was 324 while the actual one was only 94 (i.e., about 3.5 times smaller). Since many structural properties of the graph describing a social network are often correlated with vertex degree, we can conclude that a sample collected with a BFS crawler is far from being representative of the structural properties of a social network.

2.2.4 Social Media Event Collection

In the latest years, Social Web platforms emerged as one of the most pertinent marvels on the Web: these platforms are built around users, allowing them to create a web of links between each other, to share musings, feelings, photographs, travel tips, and so on. In such a scenario, often called Web 2.0 users are moving from passive consumers of contents to active producers.

Social Web platforms provide novel and remarkable research opportunities. The analysis on a large scale of patterns of user interactions provides a unique opportunity for answering questions like: how do people interact and build relationships (e.g., friendship) between each other which may evolve over time [130]? How do novel ideas spread and proliferate through the web of human contacts [24]? How does the natural language evolves through social interactions (e.g., how to do the person expand their vocabulary based on their interactions with other people) [158]?

Besides the previous questions, the analysis of patterns of human interactions in Social Web platforms can generate better business insights: if we are able to understand the dynamic interactions between humans, we are likewise able to determine how clients gather themselves around shared interests. This is an important step for marketing purposes: once clients have been classified into different groups, we can, for example, target each group by a set of advertisements that falls within their interests. Thus, we are sending those advertisements to the group of people that are interested in receiving them. In a similar form, the fabric of social interactions can be used to identify influential users, i.e., those users whose business practices can empower the selection/dismissal of a given product by large masses of users.

Finally, Social Web users regularly make accounts as well as profiles in numerous platforms [206, 53]. Correlating these accounts and profiles is very crucial to understand how the design features and the architecture of a Social Web platform impact on the behavior of a user. In that manner, one may ask whether some features provided by a given platform increase the fitness of users to socialize or they affect the volume of contents generated by a client. Once the co-relations between the features of the platform and the behavior of a user have been clarified, the organizers of that platform can provide unique services to decrease the customer churn (e.g., to avoid users becoming inactive in the platform and migrate to other ones).

In the context above, Web Data Extraction techniques play a key role because their capability of timely gathering large amounts of data from one or more Social Web platforms is an indefeasible tool to analyze human activities. Traditional Web Data Extraction techniques are challenged by new and hard problems both at the technical and scientific level.

We can characterize methods to gather information from a Social Web platform into two principal classes: the first class relies on the use of ad-hoc APIs, usually provided by the Social Web platform itself; the second relies on HTML crawling which we already discussed earlier.

With respect to the first class of methodologies, we call attention to that, today, Social Web platforms offer powerful APIs (frequently accessible in numerous programming languages) permitting to recover in a simple and fast way an extensive variety of information from the platform itself. This information, specifically, respect not just social associations including individuals from the platforms yet additionally the content the users posted and, for example, the labels they connected to name created content.

We can refer to the approach of [134] as an applicable case of how to gather information from a Social Web platform by means of an API. In that paper, the authors introduce the results of the crawling of the entire Twitter platform. The dataset portrayed in [134] composed of 41.7 million user profiles and 1.47 billion social relations; in expansion to gathering information about user relationships, the authors accumulated likewise data on tweets and, by performing a semantic analysis, also on the main topics discussed in these tweets. The last dataset contained 4,262 trending subjects and 106 million tweets.

In addition to collecting data about a social relationship, we may gather the contents produced by users. These contents may vary from platform to another (like photos in Flickr or videos on YouTube) and it is crucial to be tagged by certain labels in order to make the retrieval of these data easier or to increase the insights from the generated contents.

2.3 Data Processing

2.3.1 Data Preparation Techniques

Data preparation is of crucial importance for accurate data analysis. Some important factors in data preparation are reliability, accuracy, understandability, and trustworthiness of the data. Data preparation generates a dataset smaller than the original one, which can significantly improve the efficiency of analysis [237]. The authors of [181] mentioned that 40-60% of the time needed in order to prepare the data to be reliable and accurate as much as possible for big data analysis. Controlling data quality and accuracy in big data applications and especially in realtime has proved to be a challenge. The raw data generally requires data preparation tasks to be prepared for analysis. Data preparation consists of three main tasks which are as follows:

- Data Cleansing: Data cleansing is the first step in data preparation techniques which is used to find the missing values, smooth noise data, recognize outliers and correct inconsistent [9]. That is due to the fact that real-world data is rarely clean and homogeneous. According to [155], most of the data will have some missing values. There could be various reasons for this such as the source system which collects the data might not have collected the values or the values may never have existed. In all cases, it is very important when dealing with Big Data to check out for the missing elements in the data. These missing elements can be handled either by removing a row or replacing a missing value with a more appropriate value. A more appropriate value can be filled in different ways: manually, global constant, attribute mean, attribute mean for all samples belonging to the same class as the given tuple, most probable value using techniques like inference based regression using a decision tree induction or Bayesian formalism. According to our understanding the theory of data cleansing can be formalized as follows:

  Definition 1. *Let Ω be the set of all tuples.*
  Let M be the set of tuples with missing values.
  Let N be the set tuples with noise (mostly, using statistical calculations to measure the noise).
  Let Ω' be the output set of data cleansing on Ω.

$$\Omega \rightarrow \Omega' / \forall x \in \Omega', x \in \Omega, x \notin M \ \& \ x \notin N$$

- Data Filtering: Dealing with the huge volume of Big Data is not always the right way, as it increases the complexity of the algorithms and thus increases the execution time. We need to reach the same meaningful results with minimum complexity and that would happen just through filtering the relevant data out of the overall. Therefore, filtering is another task of the data preparation that can serve in reducing the complexity and focus more on the relevant data for analysis. We can present the data filtering formula as follows:
  Definition 2. *Let Ω be the set of all tuples.*
  Let C be the set of tuples satisfying the condition(s).
  Let Ω' be the output set of data filtering on Ω.

$$\Omega \rightarrow \Omega' / \forall x \in \Omega', x \in \Omega \ \& \ x \in C$$

- Data Integration: In [211], the author mentioned that some of the most interesting studies of data come from combining different data sources. These operations can involve anything from the very straightforward concatenation of two different datasets to more complicated database-style joins and merges that correctly handle any overlaps between the datasets. We presented the formula of data integration in the following:
  Definition 3. *Let A be the set of all tuples from datasource A.*
  Let B be the set of all tuples from datasource B.
  Let $A \bigcup B$ be the output set of data integration of sets A and B.

$$A, B \rightarrow A \cup B / \exists \text{ format } f \text{ where } x \text{ conforms with } f \ \forall x \in A \cup B$$

2.3.2 Realtime Data Processing Systems

Streaming data is a sequence of data tuples that is unbounded in size and generated continuously from possibly a large number of data sources. Streaming data includes a wide variety of data types and formats such as log files created in web servers, online purchases, a player moves in online games, information from social media and telemetry from sensors, and so forth [8]. Because of the continuous data generation, streaming data cannot wait for

[8] Streaming Data, "Webpage," https://aws.amazon.com/streaming-data/.

all the data to be collected like in the batch processing. Data stream processing should be consecutive and incremental as the event occurs in realtime or near realtime without specifying any ending constraints.

Streaming data processing shows challenges in terms of performance, scalability, robustness, and fault-tolerance. Generally, custom coding has been utilized for streaming data processing [205]. But this solution suffers from its inflexibility, high cost of implementation and maintenance, and slow response time to new feature requests. Recently, many modern distributed stream processing frameworks have been developed.

Using these frameworks, developers can easily build their own stream processing applications. These frameworks fall generally into two classes. The first classification, called realtime streaming data processing frameworks, incorporates Apache S4[9], Apache Storm[10], Apache Samza[11] and Apache Flink[12]. Such frameworks process the streaming data on a tuple-by-tuple premise in which each tuple is handled as it arrives. Conversely, the frameworks from the second class, for example, Spark Streaming[13], gather data in certain time intervals and process them in batches. These frameworks are called micro-batch streaming data processing systems. Micro-batch frameworks have a tendency to have higher throughput than ongoing realtime frameworks, especially when using large batches. However, large batches in micro-batch systems require high processing latencies, which prohibit realtime processing in streaming data. Next, we particularly introduce two most frameworks on top of which our realtime stream data processing system performs which are Apache Storm and Spark Streaming.

Apache Storm

Apache Storm aims at providing a framework for realtime stream processing, which additionally accomplishes adaptability and adaptation to internal failure. Similar to Hadoop, Storm can be deployed on a cluster of heterogeneous machines. In a Storm cluster, there are two different kinds of nodes, the master node and the worker nodes known as slaves. The master node runs a daemon process called Nimbus that is responsible for distributing

[9] Apache S4, "Webpage," http://incubator.apache.org/s4/.
[10] Apache Storm, "Webpage," http://storm.incubator.apache.org/.
[11] Apache Samza, "Webpage," http://samza.apache.org/.
[12] Apache Flink, "Webpage," https://flink.apache.org/.
[13] Spark Streaming, "Webpage," http://spark.apache.org/streaming/.

code around the cluster, assigning tasks to machines, and tracking any failure. Each worker node runs a daemon process called Supervisor that handle the work assigned to its machine and manage worker processes based on instructions from Nimbus. All coordination between the master (Nimbus) and the slaves (Supervisors) is done through a Zookeeper cluster, which provides a distributed, open-source coordination service for distributed applications. Furthermore, the Nimbus daemon and Supervisor daemons are fail-fast and stateless; all state is kept in Zookeeper or on a local disk.

A stream in Storm is an unbounded sequence of tuples, which is ingested and processed in parallel in a distributed manner. Each stream is defined with a certain defined schema, as a table in databases. developers design *topologies* in Storm to process streaming data. A topology is a directed acyclic graph of computation in which each node is a primitive provided by Storm to transform data streams.

The execution of each topology is performed by many worker processes that spread across multiple machines in the cluster. There are two fundamental natives in Storm, spouts, and bolts. A spout is a source of streams in topology, it is responsible for ingesting the data from external data sources and emits them into the topology. All processing logic in topologies is performed in bolts, which possibly emit new streams and often cooperate with each other to complete any complex stream transformations. Both spouts and bolts can produce more than one stream. An example of a Storm topology can be found in Figure 2.1.

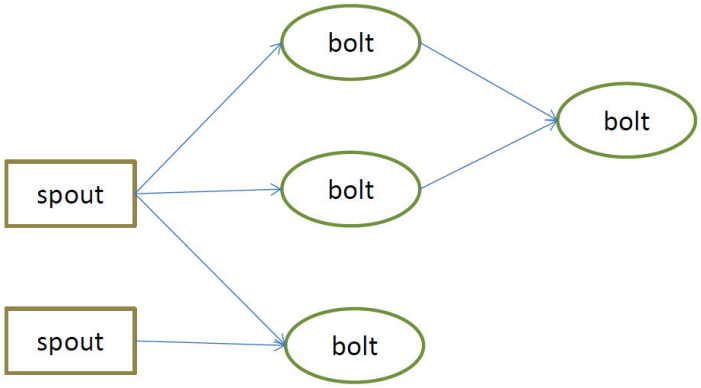


FIGURE 2.1: This shows an example of a Storm topology

Storm guarantees that every spout tuple will be completely processed by the topology with at-least-once semantics [153]. It does this by following the

tree of tuples activated by each spout tuple and deciding when that tree of tuples has been effectively finished through affirmations. On the off chance that Storm neglects to distinguish that a spout tuple has been completely processed within a timeout, at that point, it considers the tuple failed to be delivered and replays it later.

Spark Streaming

Spark Streaming makes it easy to build scalable fault-tolerant streaming applications. Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams[14]. Spark Streaming processes streams of events in a micro-batch manner. The main difference between Spark Streaming and Apache Storm is the fact that Spark Streaming processes tuples in streams in a micro-batch manner, whereas Apache Storm processes tuples in streams one-at-a-time.

To be specific, as appeared in Figure 2.2, Spark Streaming gets input data streams and divides the data into batches, which are then handled by the Spark engine to create the last outcome stream which is likewise in batches. Each batch contains data from a different time interval.

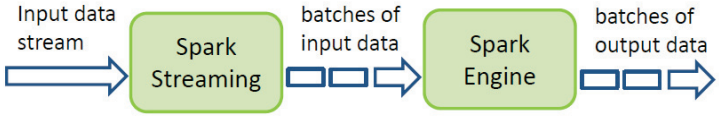


FIGURE 2.2: This shows the Spark Streaming processing example

Spark Streaming provides a high-level abstraction called discretized stream or DStream [235], which reflects a continuous stream of data. DStreams can be made either from input data streams from other data sources or by applying high-level operations on different DStreams. Technically speaking, a DStream is represented as a sequence of Resilient Distributed Datasets (RDDs), and an operation applied on a DStream is mapped to one or more operation on the underlying RDDs. Spark Streaming provides developers a high-level API which abstracts most of the details of DStream operations to facilitate faster and easier usage.

[14]Spark Streaming, "Webpage," http://spark.apache.org/docs/latest/streaming-programming-guide.html.

In order to achieve fault-tolerance that is application independent (such as, system failures, JVM crashes, etc.), Spark Streaming provides a checkpointing mechanism to keep sufficient information periodically stored in well-suited storage (e.g., HDFS) to recover from any failure.

2.3.3 Batch Style Data Processing Systems

During the past decade, numerous large-scale data processing systems have emerged to address the big data challenge. Unlike the conventional DBMS (Database Management System) which runs in a single machine, these systems run in a cluster with a collection of machines (nodes) in a Shared Nothing Architecture [204] where all nodes are connected together and each manages its own resources (local hard disk and local main memory) [180], as shown in Figure 2.3. To achieve parallel processing, these systems divide datasets into partitions distributed into different machines to be available for more efficient analysis.

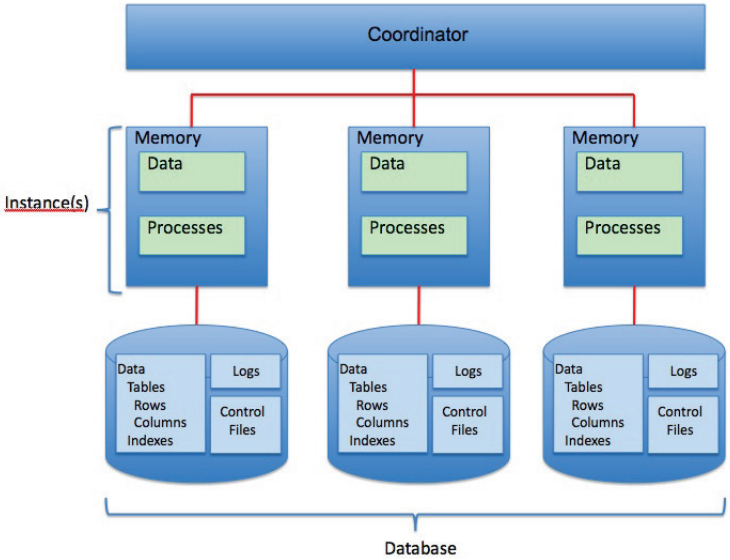


FIGURE 2.3: This shows the shared-nothing architecture

Structured Query Language (SQL) is the standard means of manipulating and querying data in relational databases, though with proprietary extensions among the products. The ease and ubiquity of SQL have even led the creators of many "NoSQL" or non-relational data stores, such as Hadoop, to adopt subsets of SQL or come up with their own SQL-like query languages. Users can specify an analysis task using an SQL query, and the system will

optimize and execute the query. To clarify, we focus on systems for large-scale processing, specifically, the Online Analytical Processing (OLAP) in which the workloads are read-only, as opposed to Online Transaction Processing (OLTP).

Mainly, we can classify these systems into two categories, Parallel DBMSs and SQL-on-Hadoop Systems, based on their storage layers: Parallel DBMSs store their data inside their database instances, while in SQL-on-Hadoop Systems, data is kept in the Hadoop distributed file system.

- **Parallel DBMSs**

  Parallel DBMSs are the most punctual frameworks to make parallel data processing accessible to an extensive variety of users, in which every node in the cluster is a database instance. The research work done by Gamma [62] and Teradata[15] inspired most of these systems. They achieve high performance and scalability by distributing the rows of a relational table across the nodes of the cluster.

  The horizontal distribution of partitions enables SQL operators like selection, aggregation, join and projection to be executed in parallel over the partitions of tables located in different nodes. Many commercial system implementations are available, including Greenplum [16] Netezza [17], Aster nCluster[18] and DB2 Parallel Edition [19], as well as some non-commercial open source projects such as MySQL Cluster[19], Postgres-XC[20] and Stado[21].

  Some different frameworks like Amazon RedShift[22], ParAccel[23], Sybase IQ [154] and Vertica [135], vertically segment tables by collocating entire columns together instead of collocating rows with a horizontal partitioning scheme. When executing user inquiries, such systems can more precisely access the data they need rather than scanning and discarding unwanted data in rows. These column-oriented systems have

[15] Teradata, "Webpage," https://www.teradata.com/.
[16] Greenplum, "Webpage," https://www.greenplum.com/.
[17] Netezza, "Webpage," https://www-01.ibm.com/software/data/netezza/.
[18] Aster Data, "Webpage," http://www.asterdata.com/product/ncluster_cloud.php.
[19] MySQL cluster, "Webpage," https://www.mysql.fr/products/cluster/.
[20] Postgres-XC, "Webpage," https://wiki.postgresql.org/wiki/Postgres-XC.
[21] Stado, "Webpage," https://launchpad.net/stado.
[22] Amazon RedShift, "Webpage," https://aws.amazon.com/redshift/.
[23] ParAccel Analytic Platform, "Webpage," https://www.paraccel.com.

been appeared to utilize CPU, memory and I/O resources better than row-oriented systems in the large-scale data processing.

For parallel DBMSs, data preparation is always an extremely important and time-consuming step. Data cleaning must be performed in advance to guarantee the quality of data. As parallel DBMSs are built on traditional DBMSs, they all require data to be loaded before executing any queries. Each record must be parsed and verified so that data conforms to a well-defined schema. For large amounts of data, this loading procedure may take a few hours, even days, to finish, even with parallel loading across multiple machines.

- **SQL-on-Hadoop Systems**

  One of the most important frameworks in the Big data ecosystem is the MapReduce framework [54]. MapReduce is a framework for processing parallelizable problems across large datasets using a large number of machines (nodes), collectively referred to as a cluster. Processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of the locality of data, processing it near the site is stored in order to minimize communication overhead. The open-source Apache Hadoop implementation of MapReduce has contributed to its widespread usage both in research and industry fields. Hadoop consists of two main components: the Hadoop Distributed File System (HDFS) and MapReduce for distributed processing. Instead of inserting data into the DBMSs, Hadoop can handle data processing with any type of data as long as data is stored in its HDFS.

  Hadoop Basics: The MapReduce programming model [222] consists of two functions: *map(k1; v1)* and *reduce(k2; list(v2))*. Users specify their processing logic by implementing their own map and reduce functions. The Map and Reduce functions of MapReduce are both defined with respect to data structured in (key, value) pairs. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain: *map(k1,v1) => list(k2,v2)*. The Map function is applied in parallel to every pair (keyed by *k1*) in the input dataset. This produces a list of pairs (keyed by *k2*) for each call. After that, the MapReduce framework collects all pairs with the same key (*k2*) from all lists and groups them together, creating one group for each key. The Reduce

function is then applied in parallel to each group, which in turn produces a collection of values in the same domain: *reduce(k2, list (v2)) => list(v3)*. Each Reduce call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list. Thus the MapReduce framework transforms a list of (key, value) pairs into a list of values. HDFS is developed to be resilient to hardware failures and focuses on high throughput of data access. A HDFS cluster adopts a master-slave architecture consisting of a NameNode (the master) and multiple DataNodes (the slaves). The NameNode manages the file system namespace and regulates client access to files, while the DataNodes serve read and write requests from the clients. In HDFS, a file is split into one or more blocks that are replicated to achieve fault tolerance and stored in a set of DataNodes.

SQL query processing over Hadoop has recently attracted many researchers and enterprises, as many enterprise data management tools rely on SQL, and many developers prefer writing high-level SQL scripts rather than writing complex MapReduce programs. As a result, many SQL-on-Hadoop systems were implemented, which all use HDFS as the underlying storage layer. In the following, we present several popular SQL-on-Hadoop systems which are highly adopted by companies.

Hive[24] is the first data warehouse software project built on top of Apache Hadoop for providing an SQL-like interface to query data stored in various databases and file systems that integrate with Hadoop. Queries submitted to Hive are parsed, compiled and optimized to produce a query execution plan. The plan is a Directed Acyclic Graph (DAG) of MapReduce tasks which is either executed through the MapReduce framework or through the Tez framework[25]. Similar to Hadoop, Hive lacks an efficient indexing mechanism. Hence, Hive access data by performing a sequential scan. Hive also supports columnar data organization, typically in the ORC file format, which helps to improve the performance. But transforming data layout as data preparation brings additional cost.

HadoopDB [2] is a combination of the parallel DBMS and Hadoop approaches, planning to achieve the best services from both parallel DBMSs

[24] Apache Hive, "Webpage," http://hive.apache.org/.
[25] Apache Tez, "Webpage," https://tez.apache.org/.

and Hadoop. The core idea is to install a database system on each Hadoop datanode and utilize Hadoop MapReduce to coordinate the execution of these autonomous database systems. To get the most from the advantages of the local DBMSs in query optimization, HadoopDB pushes as much as possible of the query processing work into the local DBMSs. In any case, before performing any query processing, HadoopDB needs to insert data from HDFS to its local DBMSs. This is accomplished by one of its components, the data loader, which also globally repartitions data in light of a predetermined partition key.

Impala raises the bar for SQL query performance on Apache Hadoop while retaining a familiar user experience. Impala [27] is an open-source SQL engine architected completely for the Hadoop data processing environment. As MapReduce focuses more on batch processing rather than interactive queries by users, Hadoop jobs suffer the overhead incurred from task scheduling. Therefore, to reduce latency, Impala avoids using MapReduce and implements its own distributed architecture based on daemon processes that cover all phases of query execution. These daemon processes run on the same machines of the HDFS cluster. Impala also accepts input data in columnar data organization, typically in the Parquet file format[26], that the user needs to create a Parquet table and to load data into the Parquet table.

The MapReduce programming model suffers from many limitations to support some applications that reuse a working set of data across multiple parallel operations, such as iterative machine learning jobs. In this manner, another framework, called Apache Spark[27], is developed for these applications, which likewise gives similar scalability and fault tolerance properties as MapReduce. Spark like other HDFS's dependent frameworks runs on top of HDFS infrastructure. Spark gives a more adaptable dataflow-based execution model that can express an extensive variety of data access and communication patterns, rather than only customizing map and reduce functions as in MapReduce. The main abstraction in Spark is the Resilient Distributed Dataset (RDD), which is a read-only data structure partitioned across a set of nodes. RDDs support many operators, such as *map*, *filter* and *groupByKey*, and

[26] Apache Parquet, "Webpage," http://parquet.incubator.apache.org/.
[27] Apache Spark, "Webpage," http://spark.apache.org/.

enable efficient data to reuse in a wide set of applications by allowing users to save intermediate results in memory. RDDs achieve fault tolerance through a notion of lineage: on the off chance that an RDD segment is lost, Spark can reconstruct this RDD parcel from the data kept in that RDD about how it was derived from different RDDs.

Spark SQL [13] is the SQL processing component in Spark, which contains a DataFrame API that can perform relational operations on both external data sources and Spark's built-in distributed collections. Spark SQL also supports Parquet file format. To reach better processing performance, Spark SQL can also cache entire tables in memory to avoid disk I/O bottleneck, which is similar to the loading procedure in parallel DBMSs.

2.4 Data Analysis

2.4.1 Feature Selection Methods

While collecting data for analysis, we need to deal with the high dimensionality of the data effectively. A large number of high dimensional data has forced fundamentally enormous challenges on existing machine learning methods. Due to the existence of noisy, redundant and irrelevant dimensions, learning algorithms will suffer from high-complexity and thus execute the learning tasks very slowly. Besides, that may affect the interpretability of the model. Feature selection is capable of selecting a small subset of important features that are considered relevant from the original ones by eliminating noisy, irrelevant and redundant features.

In terms of dealing with labeled data, feature selection techniques can be generally grouped into three categories: supervised methods [224, 240, 177, 140], semi-supervised methods [239, 231, 218], and unsupervised methods [35, 68, 232, 114, 145]. The accessibility of labeled data permits supervised feature selection algorithms to viably select discriminative and relevant features to recognize samples from various classes. Some researchers proposed and studied those supervised methods [177, 132]. At the point when some of the data is unlabeled, we can use semi-supervised feature extraction which can exploit both labeled data and unlabeled data. The vast majority of the current semi-supervised feature selection algorithms [239, 45] depend on the

development of the similarity matrix and select those features that best fit the similarity matrix. Unsupervised feature selection is considered as a substantially more difficult issue [68] when the labels used for guiding the search for discriminative features are missing. In order to achieve the objective of feature selection, a few criteria have been proposed to assess feature importance [240, 108].

Feature selection can be classified into three methods based on the different strategies of searching which are filter methods, wrapper methods, and embedded methods. Filter methods select the most discriminative features from the raw data. Commonly, filter methods perform feature selection before performing classification or clustering tasks and it falls into main steps. In the first step, features are ranked depending on predefined criteria. After ranking these features, the features with the highest ranking scores will be selected as a second step. Filter-type methods are highly used in practice especially reliefF [127, 188], F-measurement [64], mRMR [183] and information gain [188]. Wrapper methods use the proposed learning algorithm to assess the features. In [103], the authors used methods based on the Support Vector Machine (SVM) algorithm for Recursive Feature Elimination (RFE) to select the most influencing gene to cancers. Embedding methods perform feature selection in the process of constructing the model.

Supervised feature selection approaches are used for labeled data. Traditional supervised methods such as Fisher Score [67] rank features each independently based on the criterion, which can not consider the correlation among diverse features. Linear Discriminant Analysis (LDA) [78] was proposed to elevate features by maximizing the ratio between both the class scatter and within class scatter. LDA needs to calculate the inverse matrix of within-class scatter, which is not ideal when the number of training samples is smaller than the dimensionality of the data [84]. Thus, LDA suffers from the discussed problem which is known as the small sample size problem. As a resolution to this problem, another approach was proposed which is the Maximum Margin Criterion (MMC) based algorithm in [139]. This latter algorithm uses a linear combination of traces between class scatter and within class scatter in the objective function, in addition to a constraint of the orthogonal weight matrix. All supervised methods suffer from a common limitation which is the sufficient labeled data needed that is known as very hard to obtain in practice. In addition, if the labeled training data are scarce, the efficiency of such supervised methods usually drop dramatically [151].

Semi-supervised feature selection methods can exploit both labeled and unlabeled training data. Semi-supervised methods are able to do feature selection from unlabeled data when there is no sufficient number of labeled data. One of the main methods is the graph Laplacian-based semi-supervised methods in which it considers that most data examples lie on a low-dimensional manifold, such as Semi-supervised Discriminant Analysis (SDA) [34]. In graph Laplacian-based methods, graph Laplacian matrix is drawn to utilize the unlabeled samples. However, due to the time-consuming computation of the graph, they are usually less efficient in handling large-scale data [41]. Therefore, it is fundamental and essential to study unsupervised feature selection methods.

When considering unlabeled data (data missing the label information that is used for guiding the search of discriminative features), unsupervised feature selection methods can be used as a much harder problem [68]. Many research works were done to tackle the relevance of selecting the features by considering a set of criteria. For instance, one commonly used criterion is to choose those features that can best preserve the manifold structure of the original data. Another known method is to identify cluster indicators through clustering algorithms and then changing the problem from unsupervised feature selection to supervised one. This method can be used in two different ways. First, it is done in one unified framework by seeking cluster indicators (considered as pseudo labels) and simultaneously performs the supervised feature selection method. For example, consider the works [232] and [142], the authors integrated nonnegative spectral cluster and structural learning into a connected framework. Second, it seeks for cluster indicators, then removes or selects certain features through performing feature selection, and keeps repeating the previous two steps iteratively until specified criteria are met. The authors in [35] first seeks for obtaining the indicator matrix of data points using spectral analysis, and then use the obtained indicator matrix to perform feature selection like a supervised one.

2.4.2 Event Clustering Algorithms

We are working in the context of the huge amount of data. These data are overwhelming for conventional tools and cannot be handled by traditional analysis techniques. Methods should be enhanced to reach the required scalability to handle it. Some cluster analysis methods have been proposed to

split the data into different clusters. The data clustering methods are unsupervised learning models which means it does not require any labeled data during the training phase. Most of these methods do not need to know the exact number of clusters in advance. The expected output of such methods is dividing a set of data into different multiple clusters. Considering a set of data instances, these instances should be divided by the data clustering method into subsets of data which maximize the intra-subset similarity and inter-subset dissimilarity, where a similarity measure is specified in advance.

Since most data clustering problems have been known as NP-hard problems [93], different approaches have been proposed in previous years. In general, those methods can be categorized into different paradigms, which will be discussed as follows:

- Partitional Clustering

  Data is divided into independent groups such that each data instance is assigned to exactly one group. K-means [203] is one of the well-known classical partitioning methods that applies an iterative enhancement approach with two main steps. The first step is to estimate the means of clusters and select them as centroids, while the second step is to assign data points to their nearest centroids. In practice, people adopt this method due to its computational speed and simplicity [200, 229]. Its main limitation is the vulnerability to its random seeding technique in which the clustering result quality will be affected adversely if the initial seeds are not selected correctly.

- Hierarchical Clustering

  Clusters are created by two main approaches which are known as bottom-up and top-down. Single-linkage clustering [230] is an example of the bottom-up approach in which data points are gradually added together to form clusters. In every step, all pair-wise distances are calculated to identify the minimum. The data points that have the minimum distance are linked together. Such a step is repeated until all data points are grouped together. A hierarchical tree is built to connect all data points at the end. We can cut the tree using the tree depth level in order to form the clusters. A special hierarchical clustering method called "Chameleon" has been discussed to model the data dynamically [124]. It can merge and divide clusters based on the inter-connectivity and closeness concept. If the inter-connectivity and closeness between two

clusters are higher than those within the clusters, then these two clusters are merged into one cluster.

- Density-based Clustering

  Besides the well-known clustering methods, there are different clustering paradigms. Density-based clustering is one of these paradigms. Data is clustered using density-based clustering according to some connectivity and density functions. DBscan [72] is a clustering algorithm that uses density-based notions to build clusters. In order to check whether each data point is a core point or a border point, we can use one of the proposed connectivity functions which are density-reachable and density-connected. DBscan starts visiting the points randomly until all points have been passed through. If the point is a core point, it tries to spread and form a cluster around it. Based on the experimental results, the authors reported its robustness toward finding arbitrarily shaped clusters.

- Grid-based Clustering

  In grid-based clustering, the algorithm divides the data space into multiple portions (grids) at different granularity levels to be clustered independently. CLIQUE [6] can automatically define subspaces with high-density clusters as an example of the grid-based clustering. It does not require any assumption over the data distribution. The experiments show that it could scale well as the number of dimensions increases. Thus, we can consider it an efficient clustering solution for dealing with high-dimensional data.

- Correlation Clustering

  From a document clustering problem, correlating clustering [17] was proposed. This algorithm has a pair-wise similarity function f learned from historical data. The objective of this algorithm to partition the different sets of documents in the best way that correlates the documents with f. For better understanding, let us consider that we have a complete graph of N vertices, where each edge is labeled either with positive $(+)$ or with a negative $(-)$ sign. The output of this clustering is to generate a partition of vertices that agrees with the edge labels. The authors have explained this problem as an NP-complete problem. Hence, they suggested two approximation algorithms to reach partitioning. These algorithms are called Cautious and PTAS. Cautious tries

to minimize the disagreements (number of edges inside clusters plus the number of + edges between clusters). On the other hand, PTAS tries to maximize the agreements (number of + edges inside the clusters plus the number of edges between clusters). It is clear that the ideas of the two methods discussed are the same, that is, to aggregate the vertices which agree with their edge labels).

- Spectral Clustering

  Some clustering approaches may find local minima and need an iterative algorithm to find good clusters starting from different initial clustering points. However, spectral clustering [176, 201, 165] do clustering based on the leading eigenvectors of the matrix derived from a distance matrix which makes it a credible approach. The main idea behind it is the usage of the spectrum of the similarity matrix of the data to reduce the dimensions for performing k-means clustering in fewer dimensions. For further details, you can check the work discussed in [176].

- Gravitational Clustering

  Gravitational clustering was first proposed by Wright [227] as a unique clustering method. In this method, each data instance is represented as a particle within the feature space. The simulation of the movements of the particles was generated by some physical models. Jonatan et al. reported a new gravitational clustering method based on Newton laws of motion [92]. Another version of gravitational clustering more simplified was tackled by Long et al. [149]. In [219], the authors proposed a local shrinking method to move data toward the medians of their k nearest neighbors. A similar method was introduced by Blekas and Lagaris [28] which called Newtonian clustering. Newtonian clustering applies Newton's equations of motion to shrink and separate data, followed by a Gaussian mixture model building. Junlin et al proposed using molecular dynamics-like mechanism for clustering [121].

- Herd Clustering

  A novel clustering method "Herd Clustering (HC)" has been proposed by Wong et al. [226]. The novelty of this method lies in two sides: (1) HC is inspired by the herd behavior in nature, which is a commonly observed phenomenon in the real world including human mobility patterns [182]. (2) HC proves that cluster analysis can be reached in a

non-traditional way by making data alive. Due to these aspects, HC is considered as a very intuitive method and clearly understood for performing well.

- Other Clustering Strategies

  Many clustering methods exist in the literature for many years. A genetic algorithm to search for the centers of the cluster was found in [160]. The authors of [143] discussed a globally incremental approach to k-means. Another novel method called Gaussian parsimonious clustering models was proposed by Celeux et al. in [37]. In order to cluster an arbitrary number of clusters, different distance measures have been implemented into an objective function [82]. A hierarchical agglomerative clustering methodology using symbolic objects has been discussed in [95]. Clustering based on a fuzzy Kohonen network was used by Tsao et al. [25]. A fuzzy c-means algorithm has been implemented and discussed in [228], [241]. In order to reduce the noise, a pruning approach was proposed for the fuzzy c-means algorithm [236]. Recently, many kernel clustering methods have been developed [76]. The authors of [156] reported a fuzzy-rough set application to a microarray. Hu et al. in [115] have used the hierarchical clustering method for active learning. A pleasant approach was considered by Corsini et al. in which they trained a neural network to define dissimilarity measures which are subsequently used in the relational clustering [49]. Clustering over uncertain data was also considered by Gullo et al. in multiple clustering techniques [98, 100, 99]. In the literature, many other works exist; more details can be found in [230, 18, 225].

Clustering on Data Streams

The previously mentioned clustering algorithms assume data are at rest. Nowadays, data are not necessarily at rest. In fact, data can be transferred in streaming form; for example, realtime financial stock market data, video surveillance data for abnormal event detection, and social media data such as Twitter, Facebook, etc. Recently, data keep changing and enriched during the clustering. For extracting insights out of these data, the capability to process the data in a timely manner with minimum memory is fundamental. In light of that, various data stream clustering techniques are proposed. For example, Guha et al. have proposed one of the first-known technique,

STREAM, to tackle the k-median problem on streaming data with constant-factor approximation [97]. In [77], Fisher proposed an incremental clustering technique (COBWEB) to target the hierarchical clustering tree on streaming data. Zhang et al. have proposed an efficient data clustering method for huge datasets [238]. It has linear complexity and single-pass nature, it can also be applied to cluster data streams with a tree data structure, CF Tree [238]. On the other hand, an incremental clustering method (C2ICM) has been proposed to tackle the data stream clustering problems. Specifically, a lower bound for its clustering performance has also been provided [42].

2.4.3 Text Classification Algorithms

Text classification has been extensively considered in various communities, for example, data mining, database, machine learning, and information retrieval. It is also utilized in an immense number of applications in different areas such as image processing, medical diagnosis, document organization, and so on.

Text classification is used to label the text documents by predefined labels [171]. The classification problem is defined as follows: We have a training set $\mathcal{D} = \{d_1, d_2, ..., d_n\}$ of documents, such that each document d_i is labeled with a label ℓ_i from the set $\mathcal{L} = \{\ell_1, \ell_2, ..., \ell_k\}$. The main task in this context is to find a classification model (classifier) $f : \mathcal{D} \rightarrow \mathcal{L}$ which can assign the correct class label ℓ to new document d such that $f(d) = \ell$. The classification was called “hard", if a label is explicitly assigned to the test instance and “soft", if a probability value is assigned to the test instance according to [10]. There are other types of classification which allow assignment of multiple labels [94] to each new document. For a broad review on various classification methods see [67, 117]. Yang et al. assess different categories of text classification algorithms [233]. A large number of classification algorithms have been implemented in different software systems and are freely accessible such as BOW toolkit [163], Mallet [164] and WEKA[28].

- **Naïve Bayes Classifier**

  Probabilistic classifiers have gained a lot of popularity lately and have appeared to perform amazingly well [38, 120, 131, 136, 190]. These probabilistic algorithms make assumptions about how the data (words

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

in records) are produced and propose a probabilistic model in light of these suppositions. At that point, consider using training data to assess the parameters of the model. Bayes rule is used to categorize new examples and select the category that is in all likelihood has created the example [161]. The Naïve Bayes classifier is considered as the easiest and the most broadly used classifier. It models the distribution of documents in each class using a probabilistic model considering the distribution of the different labels are independent of each other -no dependency between the different classes-. Despite the fact that the "Naïve Bayes" assumption is clearly false in many real-world applications, the classifier performs pretty well.

There are two primary models generally used for naïve Bayes classifications [161]. The two models go for finding the posterior probability of a class, based on the distribution of the words in the document. The difference between these two models is, one model takes into account the number of occurrence of the words whereas the other one does not.

1. **Multi-variate Bernoulli Model:**

   In this model, the document is represented by a vector of binary features denoting the presence or absence of the words in the document. Therefore, this model only considers the absence and presence of words in a document neglecting their number of occurrences inside that document. Lewis was the first to present such a model [138].

2. **Multinomial Model:** We fetch the occurrences of words (keywords) in a document by representing the document as a bag of words. Many different implementations of multinomial model have been presented in [122, 162, 171, 202].

An important comparison study between Bernoulli and multinomial models was done by McCallum et al. [161] and concluded that

- If the size of the distinct keywords (vocabulary) used is small, the Bernoulli model may outperform the multinomial model.
- The multinomial model always outperforms Bernoulli model for a large number of distinct keywords (vocabulary sizes), and most of the scenarios perform better than Bernoulli if the size of the vocabulary chosen optimally for both models.

- **Nearest Neighbor Classifier**

  The nearest neighbor classifier is a proximity-based classifier which depends upon measuring the distance to perform the classification. The main idea is that documents that belong to the same class are more likely sharing similar characteristics (values of their features) which means close to each other based on similarity measures. The test document is labeled depending on the classification of a similar document in the training set. If we consider the k-nearest neighbor in the training data set, the approach is called k-nearest neighbor classification and the most common class from these k neighbors is reported as the class label, see [106, 157, 189, 198] for more information and examples.

- **Decision Tree Classifiers**

  A decision tree is a flowchart-like structure in which each internal node represents a condition on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from the root to the leaf represent classification rules. These rules are extracted from the training instances. In other words decision tree [81] recursively partitions the training data set into smaller subdivisions based on a set of tests defined at each node or branch. An instance is classified by beginning at the root node, testing the attribute by this node and moving down the tree branch corresponding to the value of the attribute in the given instance. This process is recursively repeated [171].

  In the case of unstructured text documents, the conditions on the decision tree nodes are normally defined in terms of keywords existing in these documents. For example, a node may be subdivided to its children depending on the presence or absence of a particular keyword or term in the document. Many researchers discussed in details the decision trees [33, 67, 117, 187]. Decision trees have been used in combination with bagging and boosting techniques [80, 195] to enhance the accuracy of the decision tree classification.

- **Support Vector Machines**

  Support Vector Machines (SVM) are supervised learning classification algorithms. SVM have been widely used in text classification problems. SVM are a type of *Linear Classifiers*. In the context of text documents,

Linear classifiers are models that classify the text documents depending on the value of the linear combinations of the documents features. The output of a linear classifier is defined to be $y = \vec{a} \cdot \vec{x} + b$, where $\vec{x} = (x_1, x_2, ..., x_n)$ is the normalized document word frequency vector, $\vec{a} = (a_1, a_2, ..., a_n)$ is vector of coefficients and b is a scalar. We can interpret the predictor $y = \vec{a} \cdot \vec{x} + b$ in the categorical class labels as a separating hyperplane between different classes.

The SVM was first presented in [50, 212]. Support Vector Machines try to find the best possible linear separators between various classes [50, 213]. A single SVM can just separate two classes, a positive class and a negative class [113]. SVM algorithm attempts to find a hyperplane with the maximum distance of ξ (also called margin) from the positive and negative instances. The documents with distance ξ from the hyperplane are called support vectors and specify the actual location of the hyperplane. If the document vectors of the two classes are not linearly separable, a hyperplane is determined such that the least number of document vectors are located on the wrong side.

2.4.4 Information Extraction Techniques

Information extraction (IE) is the process of automatic extraction of structured information from unstructured or semi-structured text. In other words, information extraction can be considered as a limited form of full natural language understanding, where the information we are looking for is known beforehand [113].

For example, consider the following sentence:

"Microsoft was founded by Bill Gates and Paul Allen on April 4, 1975."

We can extract the following information:

FounderOf(Bill Gates, Microsoft)
FounderOf(Paul Allen, Microsoft)
FoundedIn(Microsoft, April - 4 1975)

IE is one of the important processes in text analysis and extensively studied in different research domains such as information retrieval, natural language processing and Web mining.

Information extraction includes two principle methods, which are, named entity recognition and relation extraction. The state of the art in both methods

are statistical learning methods. In the following we briefly explain two main information extraction methods.

- **Named Entity Recognition (NER)**

  A named entity is one or more words that refer to a real-world object, e.g. "Google Inc", "United States", "Barack Obama". The process of named entity recognition is to identify and extract named entities in unstructured text into predefined categories such as a person, organization, location, money, etc. NER is a complicated process that can not be achieved by only doing string matching against a dictionary for several reasons. We can consider two main reasons for that. First, the dictionary is usually incomplete and does not contain all forms of named entities for a certain entity type. Second, named entities are related to the context, for example, **big apple** can be the fruit or the nickname of New York.

  Named entity recognition has many applications such as in question answering [5, 141] and also considered as a preprocessing step in the relation extraction process. Most of the named entity recognition techniques are based on statistical learning methods such as hidden Markov models [26], maximum entropy models [47], support vector machines [116] and conditional random fields [199].

- **Relation Extraction**

  Relation extraction is another essential information extraction process and is the process of searching and finding the semantic relations between entities in text documents. There is a wide range of techniques proposed for relation extraction. The most widely recognized strategy is to consider this process as a classification problem: Given a couple of entities occurring both in the same sentence, how to categorize the relation between two entities into one of the fixed relation types. There is a possibility that relation extends across multiple sentences, but such cases are rare, therefore, most of the existing work has focused on the relation extraction within the sentence. The classification approach for relation extraction have been extensively studied and many research work was done [39, 40, 101, 119, 123].

2.5 Route Planning

Solutions for route planning were extensively studied in transportation networks by Delling et al. [58] and a more recent study was conducted by Bast et al. [20]. We will show briefly the different categories of the proposed solutions in the literature:

- Basic Techniques:

  Dijkstra's algorithm [63] is considered as a standard solution to visiting all the nodes in the shortest path problem. This algorithm was extended in the literature [46, 89, 168, 208] in order to improve the asymptotic complexity from $\mathcal{O}((|V|+|A|)\log|V|)$ to $\mathcal{O}(|A|+|V|\log\min\{|V|,C\})$, where C is the maximum edge cost. Using a bidirectional search [52] proved that it can reduce the search space. This can be achieved by simultaneously running a forward search from a source **s** and a backward search from destination **t**. When the intersection of their search space intersects containing a vertex **x** on the shortest path from **s** to **t**, the algorithm may stop. In practice, using this algorithm in the road networks seems to visit approximately half as many vertices as the unidirectional approach. Due to the fact that the Dijkstra algorithm has some limitations in dealing with negative values, an alternative method for computing shortest paths was developed which is the Bellman-Ford algorithm [23, 79, 173]. Bellman-Ford algorithm competes with Dijkstra's algorithm in some scenarios reaching $\mathcal{O}(|V||A|)$ time in the worst case and often performs faster.

- Goal-Directed Techniques:

  Goal-directed techniques aim to guide the search toward the target by considering only the vertices that are located in the direction of **t**. These techniques achieve better search performance by avoiding redundant traversal of the network. These techniques use either the (geometric) embedding of the road network or properties of the graph itself, such as the structure of shortest path trees toward (compact) regions of the graph. Some examples of goal-directed techniques are: A* search [90, 69, 90], Geometric Containers [214, 215, 31], Arc Flags [111, 137, 172, 61], Precomputed Cluster Distances [159], and Compressed Path Databases [30, 29, 194].

- Separator-Based Techniques:

Planar graphs are graphs that can be embedded in the place, i.e., they can be drawn on the plane in such a way that its edges intersect only their endpoints. They have small (and efficiently-computable) small separators [144]. Road networks are not categorized as complete planar graphs due to the presence of tunnels or overpasses, they are proven to have small separators as well as [59, 71, 193]. Due to this fact, a new category was proposed as separator-based techniques. These techniques use the divide and conquer approach for more effective route planning. In this case, separators partition the large network by decomposing the large search problem into many small search problems. Many works exists to highlight the separator-based techniques including Vector Separators [60, 207, 175], and Arc Separators [128, 57, 56].

- Hierarchical Techniques:

  Hierarchical methods aim to use the inherent hierarchy of road networks. In some scenarios such as long shortest paths, it is enough to consider a small set of important roads, such as highways. Once the query algorithm is far from the source and destination, it just scans the vertices of this subnetwork. Some popular heuristic approaches [70, 112] would adopt input-defined road categories, thus providing no guarantee that it will find the best shortest paths. An overview of early approaches based on this technique was discussed by Fu et al. [83]. In the literature, some solutions were proposed as extensions that rely on hierarchical search idea such as Contraction Hierarchies [126, 86, 191, 14], and Reach [102, 91].

- Bounded-Hop Techniques:

  Bounded-hop techniques are based on precomputing distances between pairs of vertices. Implicitly, it adds "virtual shortcuts" to the graph, allowing queries to return the distance of a virtual path with very few hops. A naïve approach is to use single-hop paths by precomputing the distances among all pairs of vertices u,v in the network. After that, the shortest distance is retrieved through a single table lookup. recently, the PHAST algorithm [61] has made precomputing all-pairs shortest paths feasible, storing all $(|V|^2)$ distances is prohibitive already for medium-sized road networks. Other approaches were found in the literature for considering more hops (two or three) with better trade-offs such as Labeling Algorithms [3, 7, 55], Transit Node Routing [21, 22, 192, 4], and Pruned Highway Labeling [8, 207].

2.6 Gap Analysis/Discussion

Throughout this section, we intensively studied the different methods, techniques, and technologies used in the literature to be able to collect data from heterogeneous data sources, process these data in realtime and analyze them using the different methods from machine learning and route optimization research domains.

Nowadays, the ability to collect social media and sensor data is pretty much mature. These data can be collected and integrated with internal data sources of the logistics system increases the level of understanding of the situations and the status of the delivery and packages. After studying the importance of collecting data from multiple sensors and how together they can provide better observations of the objects and thus more accurate recordings. We deduced the need for data fusion and integrating data from different heterogeneous data sources. More data means more information and thus more accurate analysis and better insight. In addition to the sensor data, we are able to collect the data from the ocean of the data i.e. *the worldwide web*. In order to access the different data available on the web, we need to crawl the different web pages including social media pages that provide restricted APIs.

We discussed several crawling strategies used for fetching structured data on the web. Web data crawling is considered as one of the main technologies that help in having a 360-degree view of the users (e.g. social relationships, user activities, the content produced and shared by users). Crawling large online social networks requires some graph theory methods similar to Breadth-First-Search (BFS) algorithm to make sure all the users are been visited where each user represents a node of the graph. However, it is not always an adequate algorithm to be used due to several limitations. In fact, we noticed from our investigations the importance of collecting data from the web using web crawling techniques into our proposed solution framework.

Recently, a new concept arrived known as social web platforms, these platforms are built around users, linking users together, allowing them to share feelings, photographs, and so on. Sometimes, this new concept is called Web 2.0 where the users are moving from passive consumers of content to active producers. One of the most important information to highlight here is the ability to correlate different accounts on multiple platforms for the same user. Gathering this information from one or more social web platforms to

analyze human activities is of critical importance. Unfortunately, traditional web data extraction techniques are challenged by new and hard problems both at the technical and scientific level. Thus, there is a need to use ad-hoc APIs and fortunately today, social web platforms offer powerful APIs permitting to fetch in a simple and fast way an extensive variety of information from the platform itself. Using different techniques to gather data from different data sources such as sensors, web, and social media is very challenging and what is more challenging is having well-processed data in realtime.

To achieve these well-processed data we tackled the currently existing systems of the two main paradigms for processing any data which are realtime data processing systems and batch-style data processing systems. The main role of realtime data processing systems is to handle the unbounded streams of data which may include a wide variety of data types in realtime. This part is very challenging in terms of performance, scalability, robustness, and fault-tolerance. We investigated several realtime data processing systems including Apache S4, Apache Storm, Apache Samza, Apache Flink and Spark Streaming. Whereas, the main role of batch-style data processing systems is to handle a large amount of data by dividing it into multiple batches and each batch can be processed in parallel by a different machine. The machines involved in such processing runs in a cluster in a *Shared Nothing Architecture*. We presented several batch-style data processing systems including parallel DBMSs, SQL-on-Hadoop systems (e.g., HadoopDB, Impala, Spark SQL). After we extensively investigated the different available systems to adopt the one that best fit to our solution, we adopt one of the most powerful realtime stream processing system called “Apache Storm" to be used for processing the different logistics events in realtime.

The goal of performing stream processing is to have a well-processed data that can let us do the prediction of the delay in realtime. The prediction of the delay is considered within the data analysis section in-which we studied extensively the different methods and techniques that are needed to get the meaningful insights out of the data. These methods and techniques can be classified into different categories such as feature selection methods, event clustering algorithms, text classification algorithms, and information extraction techniques. Using the feature selection methods, we can reduce the high dimensionality of the data effectively by dealing with the noisy, redundant and irrelevant dimensions. Besides, in the event clustering algorithms, we

presented several clustering techniques which are: Partitional Clustering, Hierarchical Clustering, Density-based Clustering, Grid-based Clustering, Correlation Clustering, Spectral Clustering, etc. We deduced that the Hierarchical clustering algorithm has a high potential candidate for obtaining meaningful clusters in realtime. That was deducted due to the maturity of this algorithm and the different approaches in the literature that tried to use such an algorithm, for example, the work proposed by Fisher to do an incremental clustering technique to target the hierarchical clustering tree on streaming data. We also take into consideration the following reasons for selecting such a clustering algorithm:

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).

For the reason that we are processing and analyzing the coming data especially the unstructured text data like social media data, we studied the different text classification algorithms. We focused on the most well-known classifiers in this domain including Naïve Bayes Classifier in its two models: Multi-variate Bernoulli Model and Multinomial Model, Nearest Neighbor Classifier, Decision Tree Classifier, and Support Vector Machines. We decided on using the Naïve Bayes Classifier as this classifier has gained a lot of popularity and has appeared to perform amazingly well as we have mentioned before. More specifically the Multinomial Model of this classifier since we do not have a limited number of keywords, we are gathering the data from social media which contains a huge amount of keywords and thus as explained previously this model outperforms the Bernoulli model for such a case. Due to the fact that we are dealing with mainly unstructured textual data, we intend to study the different text mining techniques to extract the critical information out of these data. Thus, we studied intensively two main techniques used in the state of the art which is Named Entity Recognition (NER) and Relation Extraction. Our main focus was to understand a simple sentence and not multiple sentences and therefore we find NER better technique to be used in our scenario.

Always we tried to place our goal in front of our hands, and thus we intend to study the different route planning techniques used to be able to perform the re-routing of the delivery plan in realtime. The studied techniques are as follows: Dijkstra, Bellman-Ford algorithms, Goal-Directed techniques, Separator-based techniques, Hierarchical techniques, and Bounded-Hop techniques. We used route planning techniques to achieve the prescriptive analysis of the well-processed data after predicting the possibility to run into a delivery delay.

2.7 Conclusion

After discussing our point from the literature review, we noticed several research challenges that we need to tackle in order to reach our objective. These research challenges are mainly because of gathering the huge amount of data with variety and velocity issues. These data cannot anymore be handled by conventional tools and techniques and thus prompts new techniques, methods, and tools to be able to tackle such challenges.

We showed the importance of collecting and integrating data from sensors, social media, and different web pages especially if correlated with the internal data. Besides that, we discussed the different processing modes which are batch-style data processing and realtime data processing. We decided to use one of the most powerful systems which are called “Apache Storm" as it is known for its scalability, fault-tolerant, guarantees your data will be processed and is easy to set up and operate.

We studied different event clustering algorithms and we considered Hierarchical clustering for our solution for two main reasons: its maturity to give meaningful clusters as output and the other one is that it has tried before to do incremental clustering which is an important feature to perform realtime clustering. On the other hand, many text classification algorithms are not able to deal with high dictionary data. As we are collecting data from social media, the size of the dictionary is approximately infinite as everyone writes in his/her own way and thus it was convenient to select Multi-nomial Naïve Bayes Classifier to classify the textual data in realtime. Information retrieval techniques is a natural fit for our solution as we are mainly dealing with unstructured text. NER was better than Relation Extraction for our scenario due to the fact that we are dealing with one sentence at a time.

We noticed from our outcome, the best fit technique, method or tool to be used to fulfill a certain role, but unfortunately for the best of our knowledge we could not find any integrated solution that combines all these different components together to overcome the previously mentioned challenges to do the re-routing of the delivery plan in realtime to avoid any delay.

Chapter 3

Chapter Three: Data Processing Frameworks

Summary

In this section, we will talk about the main contribution of this thesis. This contribution is to find the most convenient solution that can handle the coming events on the fly in order to predict any probable delay through processing and analyzing these events. To find this solution, we developed two data processing frameworks to choose the right framework for collecting and processing data in realtime. These frameworks we called them "SANA" and "IBRIDIA". Analyzing the unstructured data (especially text data) is a fundamental need in the logistics domain to anticipate the running plans. Each record of these data is considered as an event. Each event gives us information about an unpredicted situation happening at the moment. In a real-world scenario, many things can occur at any moment to impact the delivery conditions and lead to delivery delay including traffic, accidents, terrorist attacks, natural disasters, etc. Both frameworks focused more on unstructured textual events. Each relied on different techniques and methods to process these events. For instance, SANA uses n-gram with Multinomial Naïve Bayes algorithm and context extraction from the information retrieval domain, whereas, IBRIDIA was based upon an incremental hierarchical clustering algorithm which was missing in the state of the art.

Throughout this section, we did some experiments over these two different frameworks and shows that SANA and IBRIDIA tackle the processing problem in a different way. Besides that, we will show how SANA outperforms IBRIDIA in the applicability and the execution flexibility. That is due to the

limitations discovered in IBRIDIA, in which it suffers from some issues related to its cold start and the memory bottleneck as time passes. Even though, IBRIDIA was able to show value from the data that even human being cannot notice easily such as which events can be grouped together, which events are similar to be categorized together. Our objective was to tackle the delivery problem that might be affected by the current events which we found SANA more promising approach.

3.1 Functional Design of SANA Framework

The functional design of SANA consists of three main core modules which are text preprocessing, analytical engine and context extraction (shown in Fig. 3.1). In this subsection, we will present these modules in detail.

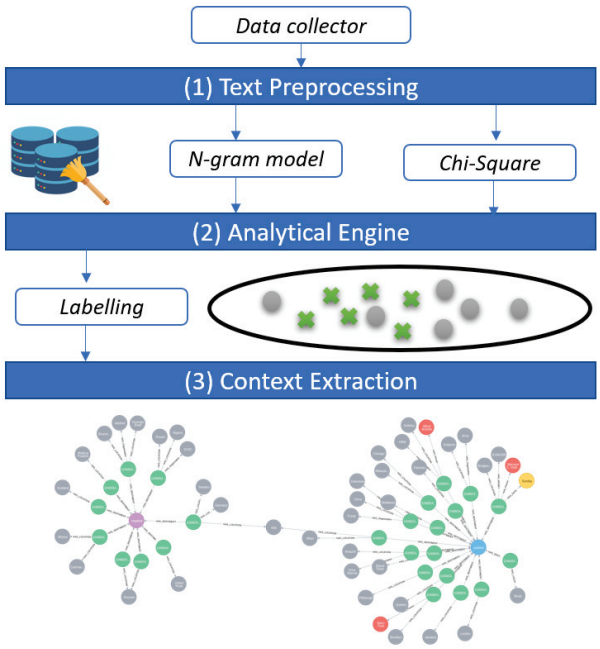


FIGURE 3.1: The principle of SANA

Text Preprocessing

SANA performs a light-weight preprocessing with incoming events. The purposes of preprocessing are two-fold: formulating terms of events, and prune unnecessary strings. The former is critical for feature selection and the latter is important for saving memory space. For terms formulation, SANA

extracts keywords using n-gram model which is a contiguous sequence of n-character or a n-word slice of longer strings or texts. The values of n varies depending on the number of characters or words the users want to extract in each extraction operation. Despite that the corpus-based is a widely known approach and used for extracting words, we choose the n-gram model because the corpus-based approach is not suitable for extracting keywords realtime. The n-gram model is a probability function:

$$\mathcal{P}(w_n \mid w_1^{n-1}) \approx \mathcal{P}(w_n \mid w_{n-N+1}^{n-1})$$

where, w_1, w^{n-1}, w^n are sequence of words.

The n-gram model is a concept found in Natural Language Processing (NLP). In general, the n-gram means a sequence of n words. For instance, we will present some examples as follows:

1. San Francisco (is a 2-gram)
2. The Three Musketeers (is a 3-gram)
3. She stood up quickly (is a 4-gram)

Some of these three n-grams examples are not seen frequently like “She stood up quickly". Such as for example 3 does not occur as often is sentences as the first two examples. Basically, an n-gram model predicts the occurrence of a word based on the occurrence of its $(n-1)$ previous words. So here we are answering the question – how far back in the history of a sequence of words should we go to predict the words that occur frequently together and consider them as terms? For instance, a bigram model $(n = 2)$ predicts the occurrence of a word given only its previous word (as $n - 1 = 1$ in this case). In the same manner, a trigram model ($n = 3$) predicts the occurrence of a word based on its previous two words (as $n - 1 = 2$ in this case). We can also think of the order-n parameters of an n-gram model as constituting the transition matrix of a Markov model the states of which are sequences of $n - 1$ words. As n increases, the accuracy of an n-gram model increases, but the reliability of our parameter estimates, drawn as they must be from a limited training text, decreases.

SANA extracts a pair of words (which are the keywords). The model approximates the probability of a word given all the previous words $\mathcal{P}(w_n \mid w_1^{n-1})$ by using the conditional probability $\mathcal{P}(w_n \mid w_{n-N+1}^{n-1})$. SANA uses the n-gram

models to increase the feature space required for the classification. Instead of taking every single word as a feature, we can double the feature space by taking the 2-grams into account. To highlight more the importance of using n-gram model in text analysis, here is an example “There is no traffic on Avenue des Champs-Élysées": If this example was part of the training data of the negative class and we did not use the n-gram model, each word of the mentioned sentence will be considered as an independent feature and it will predict the class as positive (which is wrong). It can be seen, that if the word “traffic" occurs, the class of the document is not necessarily positive. However, if we used the 2-gram model, we will have “no traffic" as a feature and thus, we will predict that this sentence is negative.

After understanding how the n-gram model works in text classification for feature selection, we intend to use a 3-gram model in our framework for determining more accurately the class of the arriving events. SANA formulates features using the extracted keyword. For instance, consider an event *I bought a Christine Davis perfume from Paris, which I found could not satisfy my expectation*, SANA extracts features by pairing words including *perfume* and *Paris*. In the next step, SANA prunes all unnecessary strings. In order to perform pruning it relies on a training dataset that consists of a set of seed words.

Analytical Engine Model

There are two approaches for text classification: lexicon-based and learning-based. The former uses a dictionary to perform entity-level analysis and the latter extracts features using the learning-based technique. We used a learning-based approach for our text classifier. We used the Multinomial Naïve Bayes classifier (a machine learning technique for supervised learning) alongside Chi-Square (χ^2) feature selection.

The Chi-Square is a feature selection method. This method is used in statistics, to test the independence of two variables. More specifically in feature selection, we use it to check whether the occurrence of a specific keyword and the occurrence of a specific class are independent. In this approach, we need to state the following hypotheses: The null hypothesis states that knowing a certain term in the sentence does not help to predict that class for this sentence. That is, the occurrence of a specific keyword and the occurrence of a specific class are independent:

H_0: the occurrence of a specific keyword and the occurrence of a specific class are independent.

H_a: the occurrence of a specific keyword and the occurrence of a specific class are not independent.

Thus, we estimate the quantity for each term extracted previously by the n-gram model and we rank them by their score according to the following equation:

$$\chi^2(D, w, \mathcal{C}) = \sum_{e_w \in \{1,0\}} \sum_{e_\mathcal{C} \in \{1,0\}} \frac{(N_{e_w e_\mathcal{C}} - E_{e_w e_\mathcal{C}})^2}{E_{e_w e_c}}$$

where $e_w = 1$ (the document contains term w), $e_w = 0$ (the document does not contain w), $e_\mathcal{C} = 1$ (the document is in class $\mathcal{C}$), $e_\mathcal{C} = 0$ (the document is not in class $\mathcal{C}$) and N s are counts of documents that have the values of e_w and $e_\mathcal{C}$. $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of documents.

High scores on χ^2 indicate that the null hypothesis (H_0) of independence should be rejected and thus that the occurrence of the term and class are dependent. If they are dependent then we select the feature for the text classification.

The previous formula can also be written as follows:

$$\chi^2(D, w, \mathcal{C}) = \frac{N(N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01})(N_{11} + N_{10})(N_{10} + N_{00})(N_{01} + N_{00})}$$

When using the Chi-Square method, we should select only a predefined number of features that have a χ^2 test score larger than 10.83 which indicates statistical significance at the 0.001 level.

After selecting the features using the n-gram and Chi-Square method, we are ready to proceed to the Multinomial Naïve Bayes classifier. The Multinomial Naïve Bayes classifier is a specialized model of the Naïve Bayes classifier [197] which is a probabilistic learning model. The Naïve Bayes classifier is a simple probabilistic classifier which is based on Bayes theorem with strong and Naïve independence assumptions. Naïve Bayes classifier is very efficient since it is less computationally intensive (in both CPU and memory) and it requires a small amount of training data. Moreover, the training time with Naive Bayes is significantly smaller as opposed to alternative methods.

The Naïve Bayes classifier assumes that the features used in the classification are independent. In a text classification problem, we will use the words (or terms/tokens) of the document in order to classify it on the appropriate class. We used this machine learning classifier to classify the coming events into two main classes: *positive* (e.g., have an impact over the delay) and *negative* (e.g., have nothing to do with the delay).

Consider an event $\mathcal{X}$, the probability of the event being in class $\mathcal{C}$ (*positive* or *negative*) is computed using the Maximum a Posteriori ($\mathcal{C}_{map}$) function:

$$\mathcal{C}_{map} = \left(\mathcal{P}(\mathcal{C}/\mathcal{X}) = \mathcal{P}(\mathcal{C}) \prod_{i \leq 1 \leq n_d} \mathcal{P}(w_k/\mathcal{C}) \right)$$

where w_k are the words of the event, $\mathcal{C}$ is the set of classes, $\mathcal{P}(\mathcal{C}/\mathcal{X})$ the conditional probability of class $\mathcal{C}$ given event $\mathcal{X}$, $\mathcal{P}(\mathcal{C})$ the prior probability of class $\mathcal{C}$ and $\mathcal{P}(w_k/\mathcal{C})$ is the conditional probability of a word w_k given class $\mathcal{C}$. It can also be interpreted as a measure of how much evidence w_k contributes that $\mathcal{C}$ is the correct class. In other words, in order to find in which class we should classify a new event, we must calculate the product of the probability of each word of the event given a particular class (likelihood), multiplied by the probability of the particular class (prior). Then we calculate the probability by considering all the classes and then we select the one with the highest probability. However, calculating the probability of the product may lead to a float point underflow due to the fact that computers can handle numbers with specific decimal point accuracy. To avoid maximizing the product, SANA adopts a refined classification model which instead of maximizing the product of the probabilities maximizes the sum of their logarithms:

$$\mathcal{C}_{map} = \left(\mathcal{P}(\mathcal{C}/\mathcal{X}) = \log \mathcal{P}(\mathcal{C}) + \prod_{i \leq 1 \leq n_d} \log \mathcal{P}(w_k/\mathcal{C}) \right)$$

Thus, instead of choosing a *positive* or *negative* class with the highest probability, SANA uses the highest logarithmic score.

Besides that, we used the relative frequency of word w_k in documents belonging to class $\mathcal{C}$ for estimating the conditional probability of a word w_k given a class $\mathcal{C}$ as follows:

$$\mathcal{P}(w/\mathcal{C}) = \frac{T_{\mathcal{C}w}}{\sum_{w' \in V} T_{\mathcal{C}w'}}$$

where T represents the count of a word given a class.

One last problem we should address here, what if a particular word does not appear in a particular class, then its conditional probability is equal to 0. If we use the highest logarithmic score, the $log(0)$ is undefined. We were obliged to use add-one or Laplace smoothing by adding 1 to each count to undercome this problem as presented below:

$$\mathcal{P}(w/\mathcal{C}) = \frac{T_{\mathcal{C}w} + 1}{\sum_{w' \in V} (T_{\mathcal{C}w'} + 1)} = \frac{T_{\mathcal{C}w} + 1}{\sum_{w' \in V} (T_{\mathcal{C}w'}) + \mathcal{B}'}$$

where $\mathcal{B}'$ is equal to the number of words contained in the training dataset. Therefore, this classifier takes into account the number of occurrences of word w in training documents from class $\mathcal{C}$.

Both training and testing algorithms (algorithm 1 and algorithm 2) are presented as follows:

Algorithm 1: TrainMultinomialNB($\mathbb{C}$,$\mathbb{D}$)

Input: set of classes $\mathbb{C}$, set of documents $\mathbb{D}$

Result: A set of vocabulary (V), an array of prior to all the classes ($prior$), a double array of conditional probability of each word in V and the different classes ($condprob$)

$V \leftarrow ReadoutVocabulary(\mathbb{D})$;
$N \leftarrow NumberOfDocuments(\mathbb{D})$;
foreach $c \in \mathbb{C}$ **do**
  $N_c \leftarrow NumberOfDocumentsInClass(\mathbb{D}, c)$;
  $prior[c] \leftarrow N_c / N$;
  $text_c \leftarrow ConcatenateAllDocumentsInClass(\mathbb{D}, c)$;
  foreach $w \in V$ **do**
    $T_{cw} \leftarrow NumberOfOccurrencesOfWord(text_c, w)$;
    foreach $w \in V$ **do**
      $$condprob[w][c] \leftarrow \frac{T_{cw} + 1}{\sum_{w' \in V} (T_{cw'} + 1)};$$
    end
  end
end
return $V, prior, condprob$;

Algorithm 2: ApplyMultinomialNB($\mathbb{C}, V, prior, condprob, d$)

$S \leftarrow ReadoutWordsFromDocument(V, d)$;
foreach $c \in \mathbb{C}$ **do**
  $score[c] \leftarrow \log prior[c]$;
  foreach $w \in S$ **do**
    $score[c] += \log condprob[w][c]$;
  end
end
return $arg\ max_{c \in \mathbb{C}}\ score[c]$;

Context Extraction

Unlike conventional sentiment analytics, SANA is context-aware. During the preprocessing phase, the context is formulated by tagging particular keywords. In order to tag correctly these keywords, SANA adopts Named Entity Recognizer (NER). NER labels sequences of words in a text which are the names of things, such as *person*, *organization* and *location* names.

NER is also known as CRFClassifier which means it is based upon the Conditional Random Field (CRF) model. It is a probabilistic graphical model that can be used to model sequential data such as labels of words in a sentence. A CRF is simply a conditional distribution $p(y/x)$ with an associated graphical structure. CRF is a conditional model and thus, dependencies among the input variable x do not need to be explicitly represented. For instance, in natural language processing tasks, features may include neighboring words and words tri-grams, prefixes and suffixes, relations with domain-specific lexicons, etc. The advantage of choosing CRF was allowing both discriminative training and the bi-directional flow of probabilistic information across the sequence. We used the Named Entity Recognition model in SANA to extract the words expressing contexts of events including location, organization, etc. which will help us understand the relevance of these events and anticipate the delivery plan in realtime without the need for humanitarian efforts.

3.2 Functional Design of IBRIDIA framework

The functional design of IBRIDIA is mainly represented by its core model for processing data in a hybrid manner including both batch-style and realtime. In our data processing model, we contributed to improving the current hierarchical clustering algorithms to perform in an incremental fashion to fit realtime processing (as shown in Fig. 3.2 represented by the Incremental Clustering of Events). During this subsection, we will present this model in details and explain it extensively.

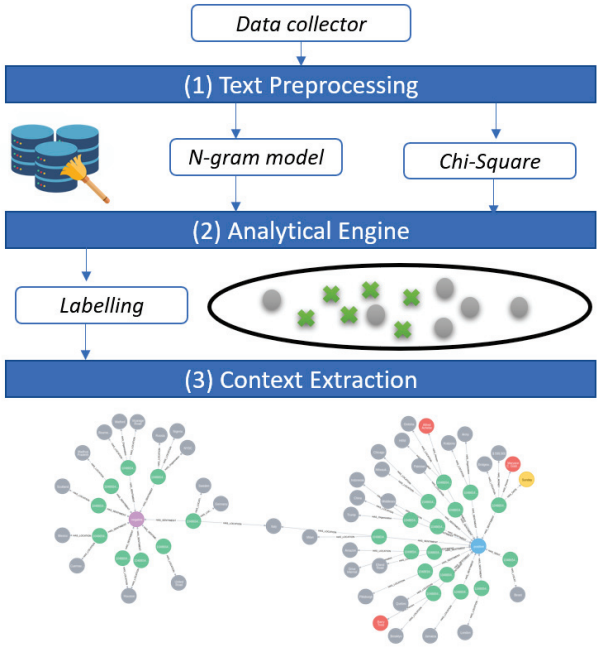


FIGURE 3.2: The principle of IBRIDIA

Data Processing Model of IBRIDIA

As mentioned earlier, IBRIDIA relies on the data processing model which we developed in our previous work [11]. 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 the data processing model. Our data processing model relies on *unsupervised learning techniques* [169]. 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 facilitates 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 [169].

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* [73]. We are interested in techniques used for building connectivity model which fits 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 clusters ranging from *singleton clusters* of individual points to an all-inclusive cluster [48]. The hierarchy of the clusters is graphically represented by a dendrogram [85]. There are two approaches to develop a hierarchical cluster model:

- *Agglomeration* refers to an approach that starts with the points as individual clusters and, at each step, merge the closest pair of clusters. It is also known as the *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 an 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 [74] – are summarized in the following:

Definition 4. *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 [96]. 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 5. *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 6. *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 7. *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 *dendrogram* which allows calculating 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 [11], 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 dendrogram 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 [185]. We intend to use the Hamming distance as a measuring criterion 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 the 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 started by the records (events) to show a real-world example in order to make it meaningful for 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.

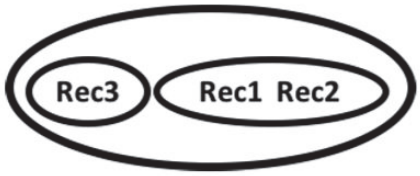
* Compare each bit in succession and record either “1” or “0” as appropriate.
* Add all the ones and zeros in the record together to obtain the Hamming distance. Hamming distance = 0+1+0+1+1+1= 4.

– 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. In the end, a new cluster is created.



The iteration stops at this step when 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.

3.3 Implementation of SANA and IBRIDIA

3.3.1 Preliminaries

- *Apache Kafka*: It is a publish-subscribe based fault-tolerant messaging system. It is a fast and highly scalable distributed messaging technology. It is used in building a 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. 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 a vast amount of data in a fault-tolerant and horizontally scalable method. It is a streaming data framework that has the capability of the highest ingestion rates. Though Storm is stateless, its distributed environment and cluster state is managed by Apache ZooKeeper[1]. 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. Realtime application logic is specified inside Storm topology. In simple words, a topology is a directed graph where vertices are computation and edges are a stream of data.

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

3.3.2 Implementation of SANA

Architecture of SANA

The multi-layered architecture of SANA (shown in Fig. 3.3) consists of various components, which are briefly described in the following.

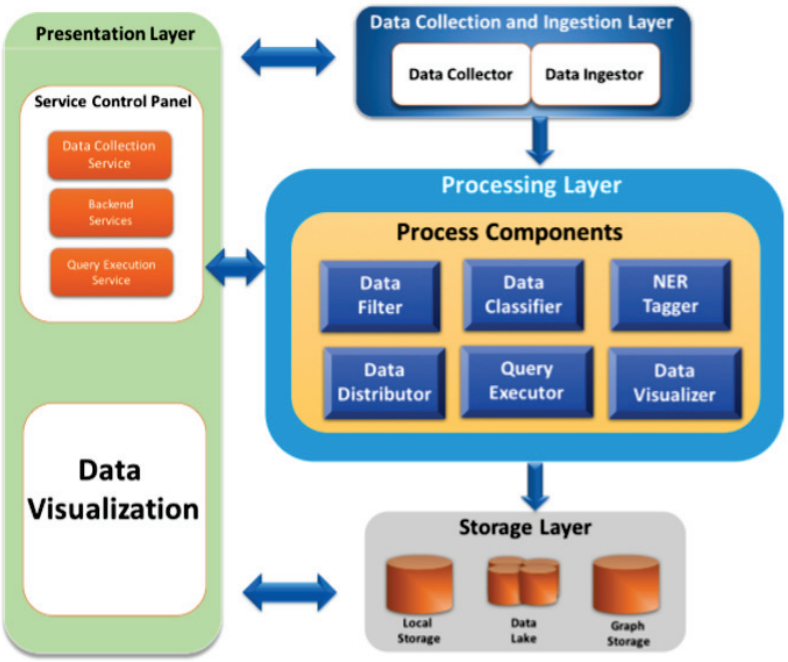


FIGURE 3.3: The Architecture of SANA

Data Collection and Ingestion Layer: This layer contains two components: a data collector and a data Ingestor. The data collector is a client that binds one or many data source APIs that enable access to remote repositories with an authentication check through their public keys. Once the connection is established, the data collector starts fetching data streams (*i.e.*, tweets) in realtime. The data Ingestor consists of two interfaces. The first interface taps data into SANA *data lake* which is a distributed Hadoop cluster, resides in the storage layer. The other interface opens a channel to push the events directly to the data processing components.

Data Processing Layer: The components contained in this layer perform several tasks. The two main tasks are carried out in this layer include *data analysis* and *visualization*. *Data distribution* and *query execution* are two additional tasks performed in this layer. The analysis starts with filtering incoming data. SANA's data filter eliminates unnecessary strings from events and keeps the

core text required for analysis. Also, it allocates an *unique identifier* to each event. Then, the text classifier extracts and classifies the events into positive (have an impact over the delay) and negative (do not have any impact over the delay) classes. We used the multinomial naïve-bayes classifier (a machine learning technique for supervised learning) along with Chi-Square (χ^2) feature selection. The multinomial naïve-bayes classifier is used after being trained with labeled training datasets that are classified into positive and negative classes. The Chi-Square (χ^2) function tests whether the occurrence of a specific string and the occurrence of a specific class is independent. The NER tagger extracts the contexts of classified texts. It labels the sequence of context related strings (*e.g.*, person, location, and organization) in an event. After the classification is done, the data distributor sends the results to a local disk, the data lake (Hadoop cluster), and the graph storage. Queries to find the comprehensive detail of the results is submitted through SANA's query interface.

Data Storage Layer: Two different types of storage is integrated in SANA: *data lake* and *graph storage* where the results are stored. The data lake is a cluster of nodes where data blocks are distributed. SANA adopts data lake to deal with massive-scale data. The graph-based storage of SANA assists in building knowledge graph of classified texts and their contexts.

Presentation Layer: SANA provides a graphical user interface (GUI) which consists of a *control panel* and a text box for *data visualization*. The control panel provides three services. The *data collection service* calls and loads the data collector. The *backend services* call processing servers, the graph database server, the coordination server which maintains configuration information and provides the distributed synchronization service. The *query execution* service calls and loads the query processor. Lastly, the visualization interface loads the data visualizer and visualizes a pie chart that shows the percentage of positive and negative classes through accumulating the classified events in each class.

Technical Details of the Implementation

SANA was developed as a desktop-based solution and software as a service (SaaS) on the cloud. It was developed to perform locally (desktop version) and on the internet (web-based version). SANA was designed as a framework that is capable of launching the full Hadoop eco-system including data acquisition server, processing servers, Hadoop cluster, and graph storage server all at once if they are all installed on the machine. It was also

developed to facilitates the submission of the Apache Storm job from its interface.

We already developed the core of SANA which was the SANA text analytics application (specialized Storm job). This application was developed as a workflow consisting of different processing units contained in a single logical Storm application called “Topology". The topology used in SANA was built using the following processing unit: Text filter, Text classifier, and Text NER which perform three tasks, filtering data, text classification, and context extraction. As soon as SANA starts ingesting data into Apache Kafka -which is the distributed messaging system that is used for ingesting the fast data streams-, The SANA’s text analytics application starts performing the different text filtering operations, classifying and enriching the records until it was visualized in the SANA’s user interface using a pie chart in realtime. SANA provides a plugged-in data source feature that establishes a connection with the data source and provides authentication if required and starts fetching data. These data are ingesting as raw data into SANA’s topology. Fig. 3.4 shows the topology.

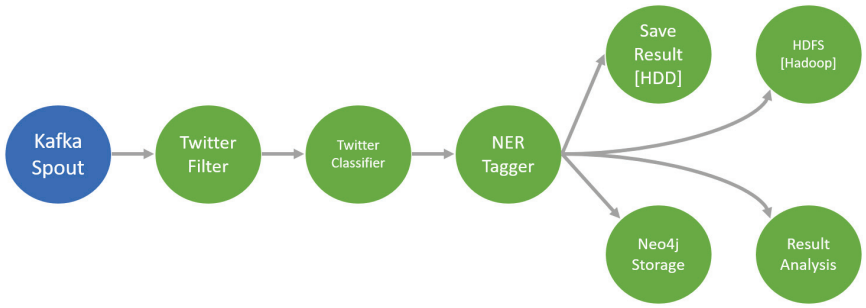


FIGURE 3.4: The topology of SANA

Then, in the next step three tasks are carried out in parallel. First, the events are visualized in a pie chart depending on their classification (positive or negative) which shows the percentages of positive and negative which means the events that have an impact over the delay and the ones that do not have any impact. Fig. 3.5 presents the results produced in multiple timestamps with less than a second difference. These results will help business analysts have better insight over the environment of the delivery because results are updated constantly as the classification is carried out in realtime over the incoming events. Second, we used the Hadoop cluster (data lake) and graph storage server to store the results produced by SANA. Also, the results are stored in a local disk. Third, the knowledge graph – consisting of extracted

classes and their contexts – are visualized by our graph storage. Figure 3.6 shows an example of the knowledge graph.

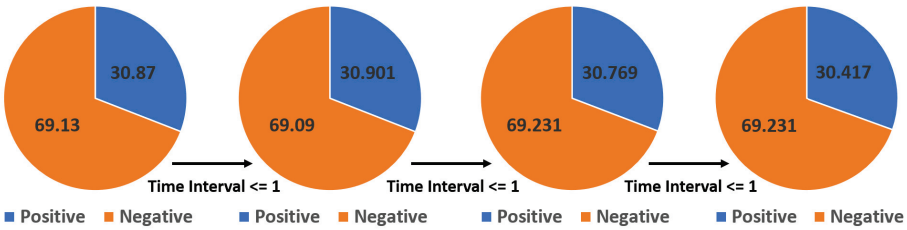


FIGURE 3.5: The percentages of positive and negative classification

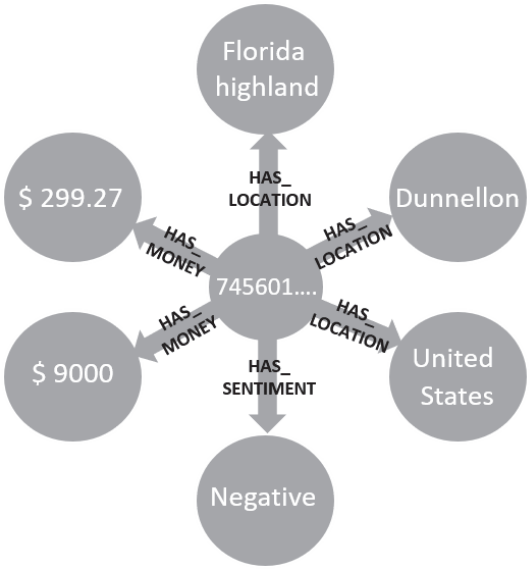


FIGURE 3.6: An example of the knowledge graph

Finally, a user might be interested to perform correlated queries to extract more knowledge from the events. SANA was designed to allow the user to query over the data stored in the graph storage. The user can simply type queries such as, "`match (n) -> 2 with n, count (*) as rel-cnt where rel_cnt > 2 return n.Id n.text Limit 15`". This query, for example, means that return each of these events which contains more than two relations among the nodes that have context and class. Figure 3.7 shows the textual representation of the results of the query.

- {"id": "1047496600078209024", "text": "Eastern Europe was exporting opium over 3,000 years ago, deposits in ancient vase reveal https://t.co/o3f27omC2u", "score": "negative", "probability": "0.611476" , "LOCATION":["Eastern Europe"] }
- {"id": "1047496604268281858", "text": "@BillsQBwatch Yeah, I saw that from Sal this morning. Disappointing on Allen's part there. I can only surmise based... https://t.co/T49Bz2XoLD", "score": "negative", "probability": "0.924243" , "LOCATION":["Sal"] , "TIME":["this morning"] , "PERSON":["Allen"] }
- {"id": "1047496608471048194", "text": "RT @MelodyeDiavian: I miss the old YouTube. The what's on my phone YouTube. The case collection YouTube. The only iconic Youtube. Now we go...", "score": "negative", "probability": "0.987664" }
- {"id": "1047496608454246400", "text": "I liked a @YouTube video https://t.co/XnnO0s9fTl Humpty Dumpty | +More Nursery Rhymes & Kids Songs - Cocomelon (ABCkidTV)", "score": "negative", "probability": "0.878106" , "ORGANIZATION":["Nursery Rhymes & Kids Songs"] }
- {"id": "1047496608454254592", "text": "RT @CarolineLucas: Theresa May sold her post-Brexit immigration system as if it was the policy we've all been waiting for.The reality is...", "score": "negative", "probability": "0.892164" , "PERSON":["Theresa May"] }
- {"id": "1047496612677738496", "text": "RT @Chanharp: Do not miss this concert! You're in for a real treat, #Illinois . If you like #Scottish and #Irish #culture and #music , be...", "score": "positive", "probability": "0.674771" }
- {"id": "1047496616855441409", "text": "RT @tictoc: Thailand's Maya Bay will be shut indefinitely because of damage caused by tourism https://t.co/JXEJreQp2a", "score": "negative", "probability": "0.541900" , "LOCATION":["Thailand", "Maya Bay"] }
- {"id": "1047496621053755392", "text": "RT @JeffreyGoldberg: I hope that every senator reads this @benjaminwittes piece (via @TheAtlantic) https://t.co/O5mNL9c4ei", "score": "negative", "probability": "0.596729" }
- {"id": "1047496621037178881", "text": "RT @JackPosobiec: Wired lets Podesta staffer post hit piece on Posobiec - totally backfires! https://t.co/ksfGRG3dTP", "score": "positive", "probability": "0.684592" , "PERSON":["Podesta"] }
- {"id": "1047496625239646209", "text": "@null BKPM to discuss disaster-resistant infrastructure development at IMF-WB meeting https://t.co/8WHlt5iLqJ", "score": "negative", "probability": "0.526410" }
- {"id": "1047496625239805952", "text": "RT @thisisntjavii: That bitch don't look like she faking.. I'm just sayin https://t.co/bFY7ttDeoy", "score": "negative", "probability": "0.612480" }
- {"id": "1047496625248198656", "text": "RT @roryclewlow: The Spark documentary is out NOW!!!! Made by the brilliant @tompullenphoto @ENTERSHIKARIhttps://t.co/epyKVVLXPj https:...", "score": "positive", "probability": "0.851952" }
- {"id": "1047496629454950400", "text": "It occurs to me that I am America.", "score": "positive", "probability": "0.563792" , "LOCATION":["America"] }
- {"id": "1047496629438353408", "text": "RT @toddstarnes: Kavanaugh drank beer. Mainstream Media declares him unfit for Supreme Court. Obama uses cocaine. Mainstream Media bows d...", "score": "positive", "probability": "0.790674" , "ORGANIZATION":["Supreme Court"] }

FIGURE 3.7: The textual representation of the results of the query

3.3.3 Implementation of IBRIDIA

Architecture of IBRIDIA

In this section, we describe the two main modules of IBRIDIA. In the logistics system, we have data generated from an 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 3.8 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 fetches data from internal and external data sources and ingests them into realtime processing engine and storage. It is worth noting that we used 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

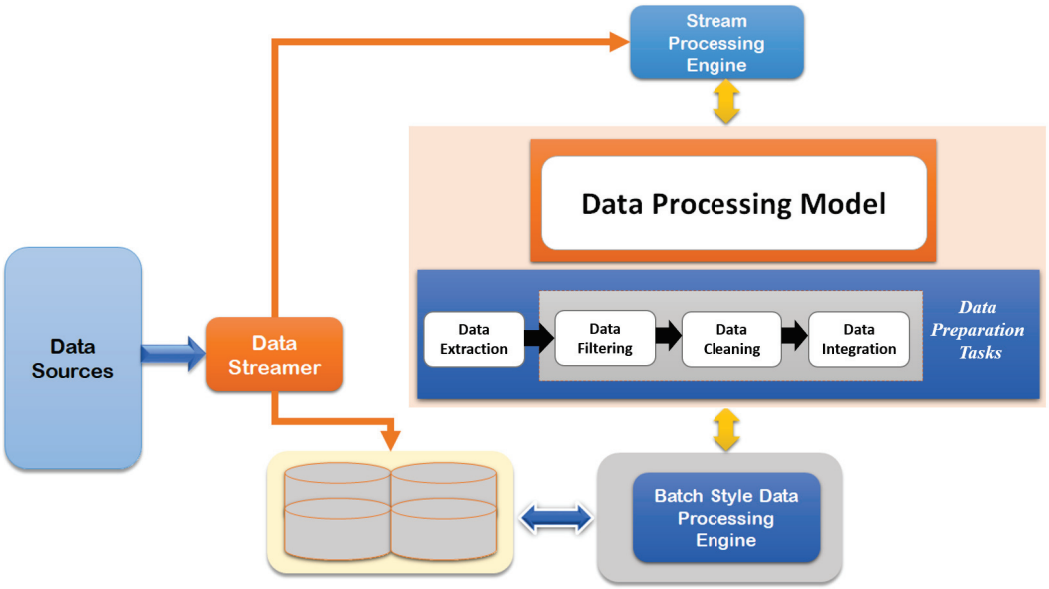


FIGURE 3.8: The high level architecture of IBRIDIA

and perform data preparation and processing tasks including cleansing, filtering, etc. In the 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.

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 is collected by the data extractor. It also fetches data from external sources such as Twitter, traffic sensors, weather sensors, Facebook and other social media. 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 the logistics information system.

- *Data Filtering*: It refers to a wide range of strategies or solutions for refining datasets. 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 collect 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 the 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 3.9 shows an example of cleaning Twitter data using ProLoD.

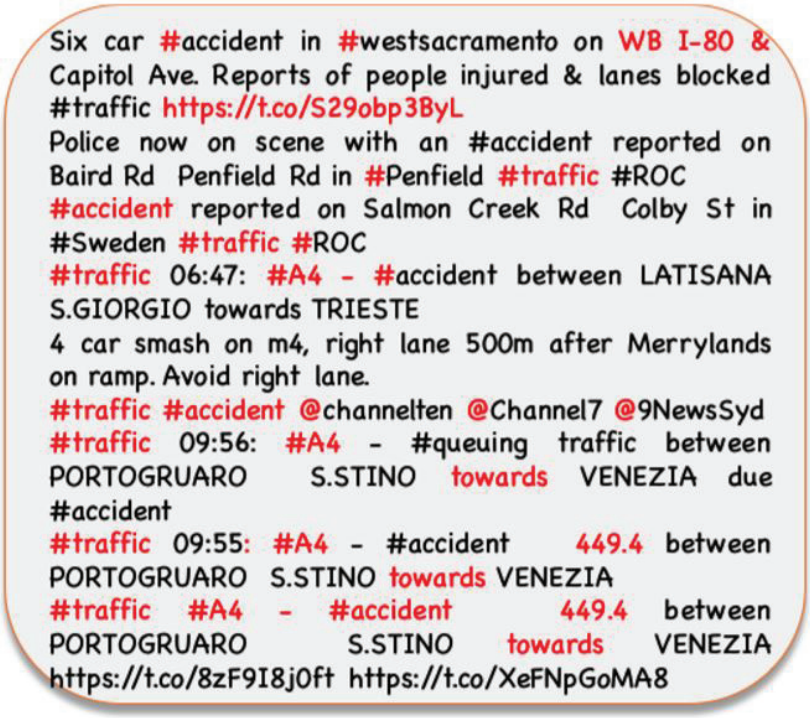


FIGURE 3.9: An example of cleaning data with ProLoD.

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

IBRIDIA Data Processing Modules

- **ProLoD - Batch style Processing Module**

  ProLoD represents the batch-style processor for the processing of logistics data stemming from multiple heterogeneous sensors (that include vehicle sensors, weather sensors, 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 3.8 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 that performs clustering in the second phase.

- **RePLoD - Realtime Processing Module**

  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 mechanisms. 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 the 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 at 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 the clustering of the events in realtime and gets immediate insights over the processed data.

Technical Details of the Implementation

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 new ones. Also, we studied machine learning libraries including DatumBox[2], SPMF[3], Massive Online

[2] http://www.datumbox.com

[3] http://www.philippe-fournier-viger.com/spmf/

Analysis (MOA), and Spark MLib[4] 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 decided to implement the model on our own. For implementation, we used Java language on Eclipse.

Datumbox reads the data from a file stored in CSV format. The data must have attribute names in order to form the data frame that will be used as the data structure in this implementation; it uses a linked hash map. Moreover, the algorithm can handle different data types like numerical, categorical, dates and so on. The reason we did not select Datumbox is its *inability* to read data by lines. The data frame can handle data as a batch that can be read in one go. Changing the data structure for an already implemented library means changing the core of their implementation as if writing one from scratch. Therefore, we decided to investigate Spark's MLib library. However, we found MLib does not have an implementation of hierarchical clustering. Additionally, MLib works only with numerical data.

SPMF is another potential machine learning library but we found that it suffers the same problem as Spark does. It works only with numerical data and this was explicitly mentioned in the documentation. MOA (Massive Online Analysis) was another potential candidate. It is developed by the same team who developed the most popular WEKA machine learning library. However, this library is locked into a specific file format which is ARFF and hence it is unable to read other data formats. An ARFF format needs to have attributes, types, and data explicitly mentioned within the file[5]. Nevertheless, in our case the data – are streamed and fed into the algorithm – does not necessarily have an attribute. Rather, data could be a set of records each of which is made up of different text words.

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[6],

[4] http://spark.apache.org/mllib/

[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.

[6] http://spark.apache.org

and Apache Storm[7] to develop RePloD module of IBRIDIA. To the best of our understanding Storm is a more potential computational system for our solution. It is fast, can process over a million 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[8] and data processing engine using Apache Storm[9].

The data processing topology of RePLoD comprises the *data cleaner*, the *data filter*, the *data transformer*, and the *data clustering* component. Figure 3.10 shows the data processing topology of RePLoD.

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 3.10: The Data Processing Topology of RePLoD

The spouts and bolts of RePLoD's 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 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. As we mentioned earlier, the processing

[7] http://storm.apache.org
[8] Apache Kafka: https://kafka.apache.org
[9] Apache Storm: http://storm.apache.org

logic of RePLoD is capable of processing any type of event. We will consider Twitter data as an example in the following.

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, split 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 realtime. The clusters are saved in the disk.

3.4 SANA Vs. IBRIDIA - An Experimental Study

In this section, we discuss the results produced through experiments with SANA and IBRIDIA. We will start by presenting the environment, datasets and attributes that we based-upon to drive our experiments.

3.4.1 Design of Experiment

System Specification

Given below is the specification of the machine we used for our experiments:

- **Processor:** *Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz (4 CPUs), ~2.9GHz*
- **Memory:** *16GB*

- **SSD:** *500 GB*
- **Operating System:** *Windows 10 (64 bit)*

About Dataset

Throughout our experiments, we used two different data sources, one for realtime processing and the other for batch style processing. The one used for realtime processing is the real data collected from *Twitter Streaming API* which is the main and only API provided by **Twitter** for producing realtime streams of tweets. The second is the real data collected from the sensors taken from Data Grand Lyon[10]. In order to deal with such datasets, it is very important to know their schemas. The schema in details of the Twitter data collected can be found on Twitter official website[11]. The schema of the other datasets will be presented in details as there is no description provided from Data Grand Lyon[12]. We will consider the following schema throughout our experiment:

publiceventtype: of type String - this field represents the type of the public event going on (mainly in the dataset, it will be mentioned as null or Activities).

networkmanagementtype: of type String - this field represents the type of the network management (mainly in the dataset, it will be mentioned as null or NetworkManagement).

observationtime: of type DateTime - this field represents the time when the event occurred.

firstsupplierversiontime: of type DateTime - this field represents the time of the supplier version first installed.

version: of type Integer - this field represents the version number used.

typeofevent: of type String - this field represents the type of the event going on in a more specific way. In this dataset, it will contain different values depending on the **publiceventtype** and **networkmanagementtype**. For example, in case of Networkmanagement, it may take one of the following values: NarrowLanes, RoadClosed, QueueingTraffic, etc. and in case of Activities, it may take one of the following: MajorEvent, Strike, SportsMeeting, etc.

townname: of type String - this field represents the name of the town in which the event occurred.

versiontime: of type DateTime - this field represents the time on the server.

gid: of type Integer - this field represent a unique identifier for each record.

publiccomment: of type String - this field represents the comment of what is going on in

[10] https://data.grandlyon.com

[11] https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object.html

[12] https://data.grandlyon.com/localisation/evfnements-routiers-temps-rfel-de-la-mftropole-de-lyon

details, thus helping in locating the different events in specific roads.

In order to do a valid comparative study between the different frameworks, I intend to take a sample of the dataset that is generated from Twitter to be used by both frameworks for realtime data processing. This sample of the dataset will be used just for testing purposes. The size of this sample data was $\approx 65\ MB$.

This dataset will be the data that drives the main experiments for doing the comparison between the two frameworks: **SANA** and **IBRIDIA**.

Experiment Attributes

The attributes that we considered to compare both systems was mainly the following:

- *Performance/Speed*: is the execution time required for an algorithm to finish its execution.
- *Accuracy*: is the closeness of results of observations to the true values or values accepted as being true.

We considered these attributes for two main reasons. First, as we are working with fast data in a big data environment, the first thing that we would consider is the speed that would reflect the efficiency of the system for real-world scenarios. Second, we considered accuracy because we knew that speed with wrong results is like dreaming without achieving. Accuracy means how correctly the data was processed (clustered or classified). If the data was accurate (correct), the analysis will be accurate which is the main goal behind processing the data.

3.4.2 An Experiment with SANA

We observed in the previous section that SANA is based upon a big data architecture that supports realtime data ingestion and processing. During the processing phase, data will be filtered, classified using multinomial naïve bayes classifier, enriched by the context using named entity recognizer and afterward, the well-processed data is distributed and stored in data lake and graph storage for obtaining a graph knowledge and more persistent storage

mechanisms. These data are then visualized in realtime and questions can be asked by the data scientist on the processed data (queries can run upon the knowledge graph of the processed data). The spouts and bolts of RePLoD's topology constitute a directed acyclic graph (DAG). The existence of these different features together in one framework that is capable of realtime processing is very interesting for getting the best out of the arrived data stream of events.

As we mentioned at the beginning of this section, we are performing our experiments over the *65 MB dataset* by simulating their realtime streaming behavior. The main processing logic of this framework is deployed over Apache Storm. Thus, it is not possible to determine the exact execution time performed for every number of records as each record is processed by different processing units and each has its own set of performance measures. We are going to present some performance measures as statistics of different time intervals for each processing unit of the topology. In order to interpret correctly the different graph visualization of the topology, we should understand a few things. The performance of Storm topology degrades when it cannot ingest data fast enough to keep up with the data source. The velocity of incoming streaming data changes over time. When the data flow of the source exceeds what the topology can process, memory buffers fill up. The topology suffers frequent timeouts and must replay tuples to process them. Using the visual representation of the storm's topology, we can indicate the data bottleneck in our application. These graphs of measuring the performance are interpreted as follows:

- Thicker lines between components denote larger data flows.
- A blue component represents the first component in the topology, such as the Kafka spout below from the RePLoD Topology.
- The color of the other topology components indicates whether the component is exceeding cluster capacity: red components denote a data bottleneck and green components indicate components operating within capacity.

We are going to present the execution of each processing unit of SANA independently within the same time intervals.

- For 1^{st} time interval: The emitted records were 16160 and the transferred were 19080.

TABLE 3.1: The execution measures of the processing units of the topology at time *t1*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.001	0.230	1480	0.000
Neo4jStorage	0	0	0.455	3818.000	20	0.000
ResultAnalysis	0	0	0.023	7.111	1440	6.510
SaveResult	1460	0	0.004	1.278	1440	1.153
twitter-classifier	4340	4340	0.627	46.935	4340	0.000
twitter-filter	4160	4160	0.608	39.221	4340	16.207
twitter-ner	1460	5840	0.691	135.397	1460	0.000

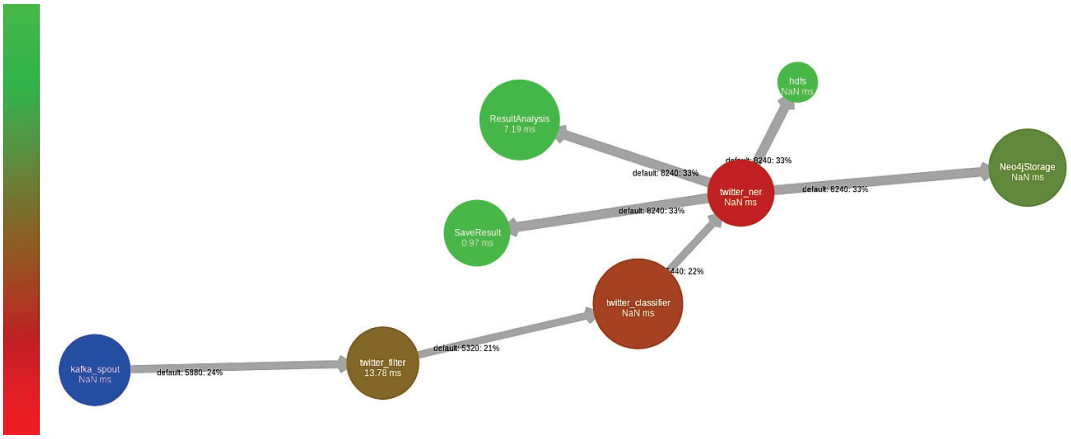


FIGURE 3.11: SANA visualization performance at time *t1*.

- For 2^{nd} time interval: The emitted records were 23840 and the transferred were 29200.

TABLE 3.2: The execution measures of the processing units of the topology at time *t2*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.001	0.143	2660	0.000
Neo4jStorage	0	0	0.893	3722.667	60	0.000
ResultAnalysis	0	0	0.026	7.134	2680	6.366
SaveResult	2680	0	0.005	1.200	2700	0.910
twitter-classifier	6060	6060	0.623	50.709	6040	0.000
twitter-filter	5940	5940	0.420	29.620	6100	12.580
twitter-ner	2680	10720	0.806	127.015	2680	0.000

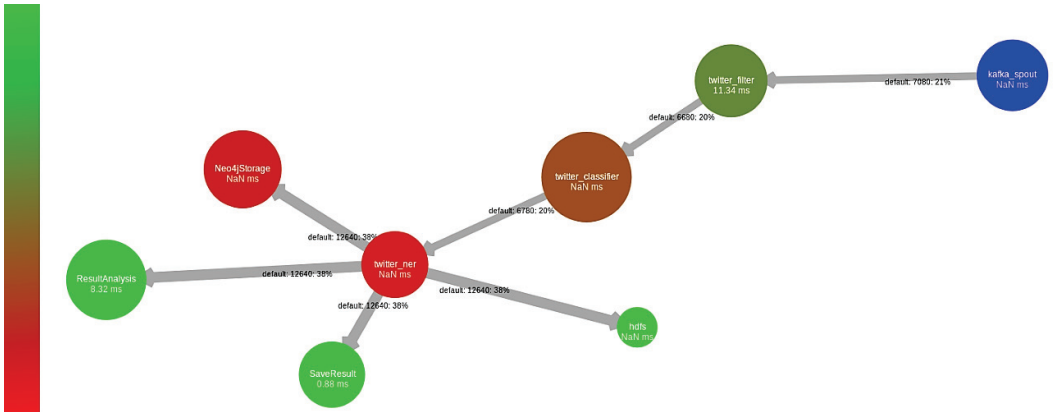


FIGURE 3.12: SANA visualization performance at time *t2*.

- For 3^{rd} time interval: The emitted records were 35840 and the transferred were 44660.

TABLE 3.3: The execution measures of the processing units of the topology at time *t3*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.001	0.230	4000	0.000
Neo4jStorage	0	0	0.715	3609.250	80	0.000
ResultAnalysis	0	0	0.034	7.919	4420	7.666
SaveResult	4440	0	0.005	1.231	4420	2.817
twitter-classifier	8940	8940	0.578	50.419	8920	0.000
twitter-filter	8920	8920	0.263	21.540	8920	9.707
twitter-ner	4420	17680	0.769	129.769	4420	0.000

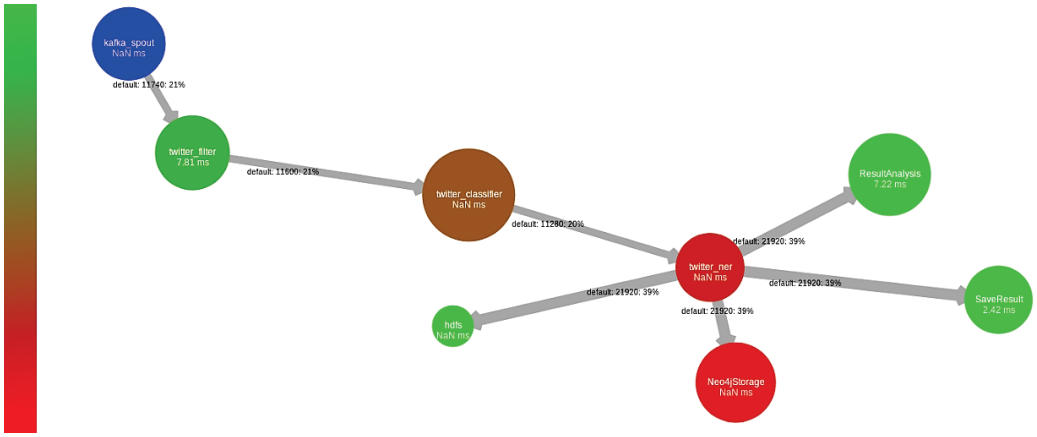


FIGURE 3.13: SANA visualization performance at time *t3*.

- For 4^{th} time interval: The emitted records were 52800 and the transferred were 66320.

TABLE 3.4: The execution measures of the processing units of the topology at time *t4*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.008	0.769	6760	0.000
Neo4jStorage	0	0	0.841	3502.428	140	0.000
ResultAnalysis	0	0	0.029	7.398	6780	6.584
SaveResult	6760	0	0.004	1.112	6760	2.114
twitter-classifier	12920	12920	0.610	52.059	12920	0.000
twitter-filter	13060	13060	0.193	16.282	12980	7.350
twitter-ner	6760	27040	0.825	139.858	6760	0.000

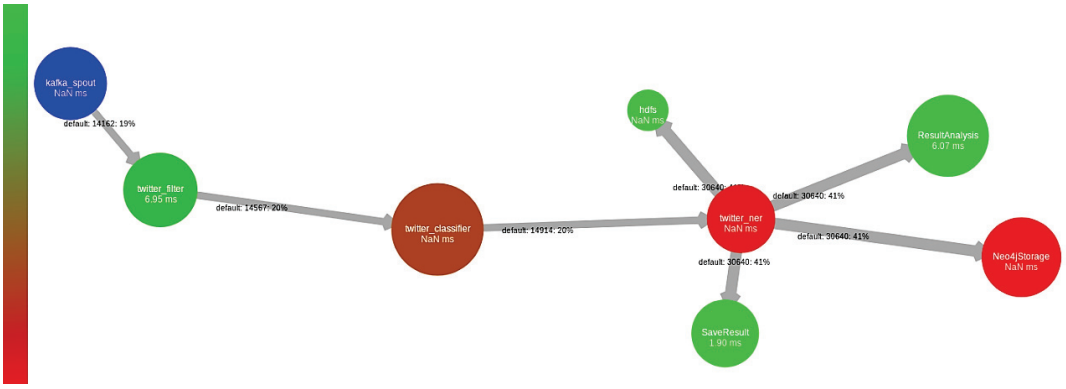


FIGURE 3.14: SANA visualization performance at time *t4*.

- For 5^{th} time interval: The emitted records were 136020 and the transferred were 173560.

TABLE 3.5: The execution measures of the processing units of the topology at time *t5*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.111	3.674	18760	0.000
Neo4jStorage	0	0	0.961	3210.928	280	0.000
ResultAnalysis	0	0	0.033	5.287	18760	4.324
SaveResult	18740	0	0.010	0.943	18740	1.068
twitter-classifier	32800	32800	0.611	35.779	32800	0.000
twitter-filter	32760	32760	0.031	7.921	32760	3.909
twitter-ner	18760	75040	0.955	95.214	18760	0.000

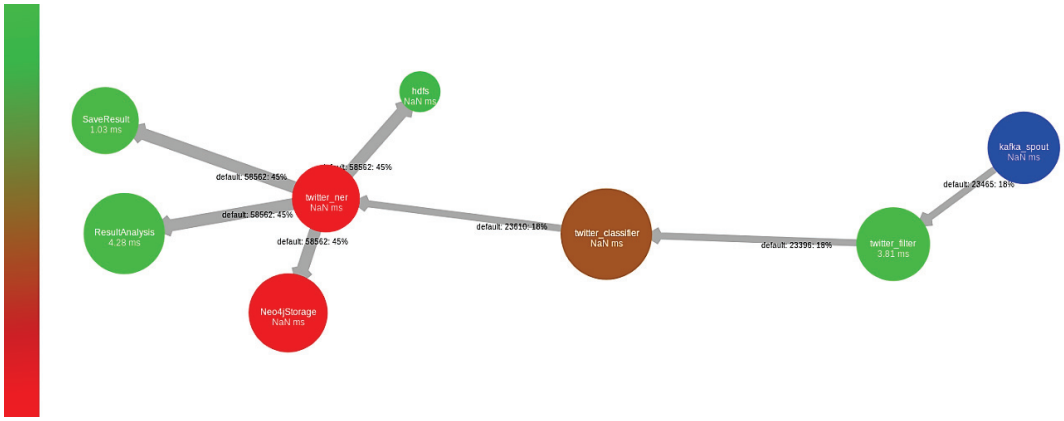


FIGURE 3.15: SANA visualization performance at time *t5*.

- For 6^{th} time interval: The emitted records were 274060 and the transferred were 356040.

TABLE 3.6: The execution measures of the processing units of the topology at time *t6*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
hdfs	0	0	0.026	2.085	40960	0.000
Neo4jStorage	0	0	1.003	3002.160	500	0.000
ResultAnalysis	0	0	0.032	3.861	40960	3.119
SaveResult	40840	0	0.006	0.718	40920	0.688
twitter-classifier	63680	63680	0.299	23.467	63720	0.000
twitter-filter	64060	64060	0.029	4.901	63780	2.469
twitter-ner	40940	163760	0.992	72.694	40940	0.000

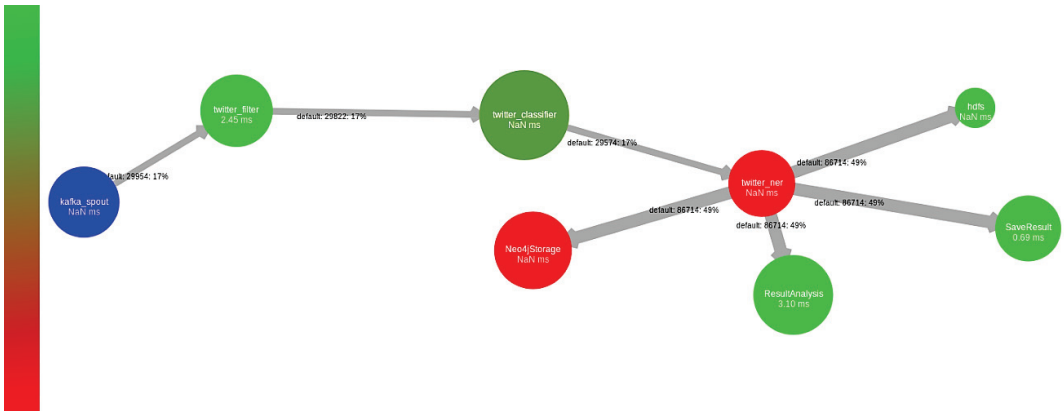


FIGURE 3.16: SANA visualization performance at time *t6*.

The results generated by SANA are written in three different storage systems which are: local disk, Hadoop storage (HDFS) and Neo4j storage (graph storage). The best approach to show the influence and importance of the data is shown as a knowledge graph as shown in figure 3.17.

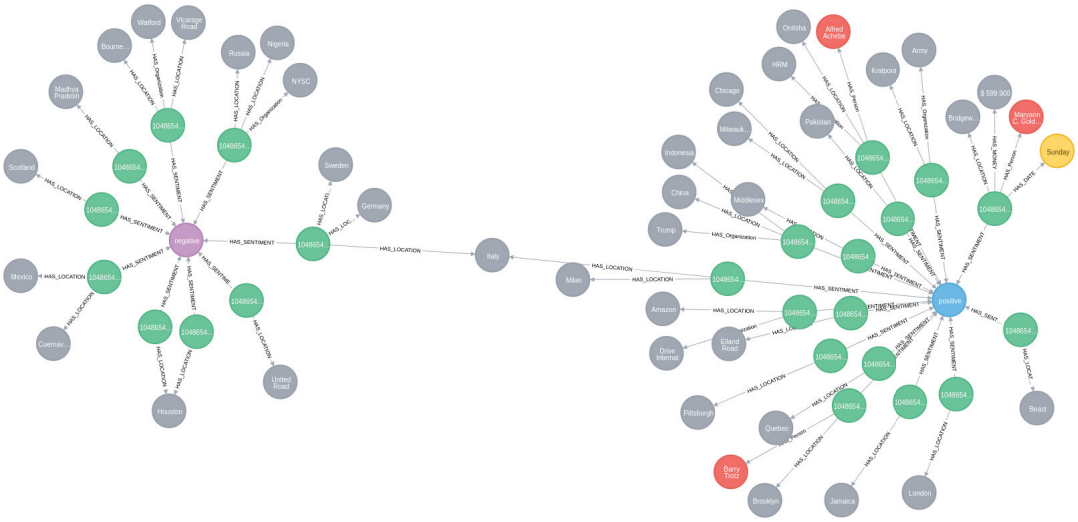


FIGURE 3.17: Graph knowledge of SANA

The obtained knowledge graph is very large so we intend to do some query over the graph storage to get a minimized graph out of the whole records stored by executing the following query:

```
MATCH (tweet)
WHERE (tweet)-[:HAS_LOCATION]->()
RETURN (tweet)-[]-()
LIMIT 25
```

This query is equivalent to saying that I need 25 tweets that have one or more locations with their relationships. For a better understanding of the results returned, we zoomed-in the previous graph in figure 3.17 to obtain two different figures: one concerns the tweets classified as positive and the other for the tweets classified as negative as shown in figure 3.18 and figure 3.19 respectively.

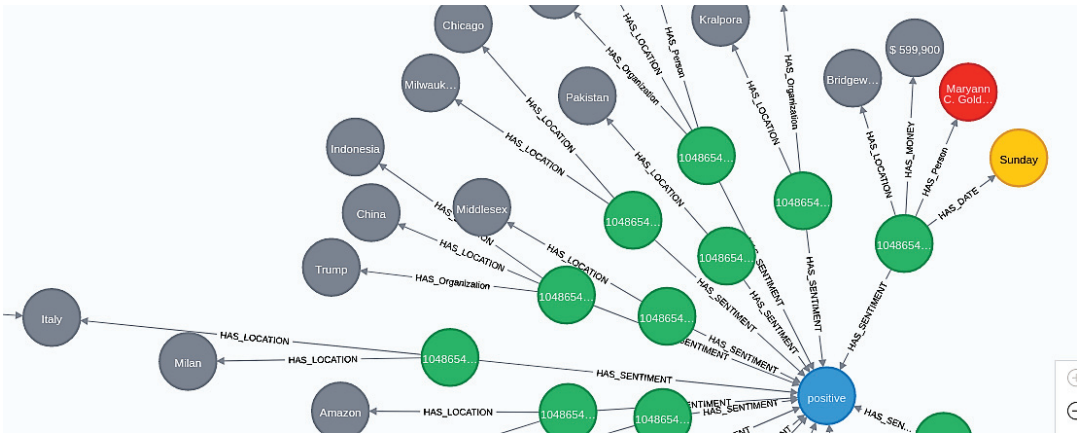


FIGURE 3.18: Graph knowledge of SANA for positive classification

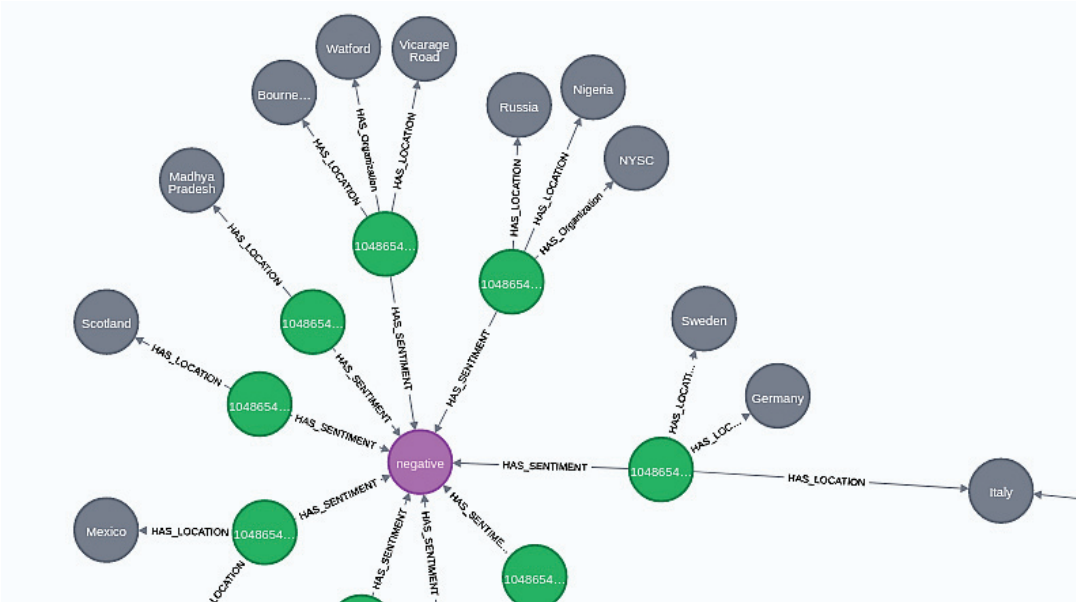


FIGURE 3.19: Graph knowledge of SANA for negative classification

Evaluating the models

During this experiment, we were able to calculate the following contingency table as shown in table 3.7.

TABLE 3.7: Contingency table for measuring the accuracy

false positive	false negative	true positive	true negative
289.5	205.6	819.9	1761.0

The accuracy is one metric for evaluating the machine learning classifier. It is the factor that shows how much of the predictions, our model got right.

$$Accuracy = \frac{Numberofcorrectpredictions}{Totalnumberofpredictions}$$

In another way we can write it as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP means true positive, TN means true negative, FP means false positive and FN means false negative.

Thus, the accuracy of our classifier reached *0.839* which is quite good as we are dealing with unstructured data.

We now need to check the accuracy of the named entity recognizer that was used in SANA to extract the event's context. We were unable to find a way to measure the accuracy and so we depend upon a comparison done between the most two popular tools for recognizing the entities which are Stanford and NLTK. According to their study[13], it is shown clearly how high is the accuracy of Stanford tool reaching *92.23%* whereas NLTK reached *89.71%* at most as shown in figure 3.20. As we used the Stanford implementation in SANA, we are satisfied to say that the accuracy of the context extraction model used is quite high for serving the delivery delay problem efficiently.

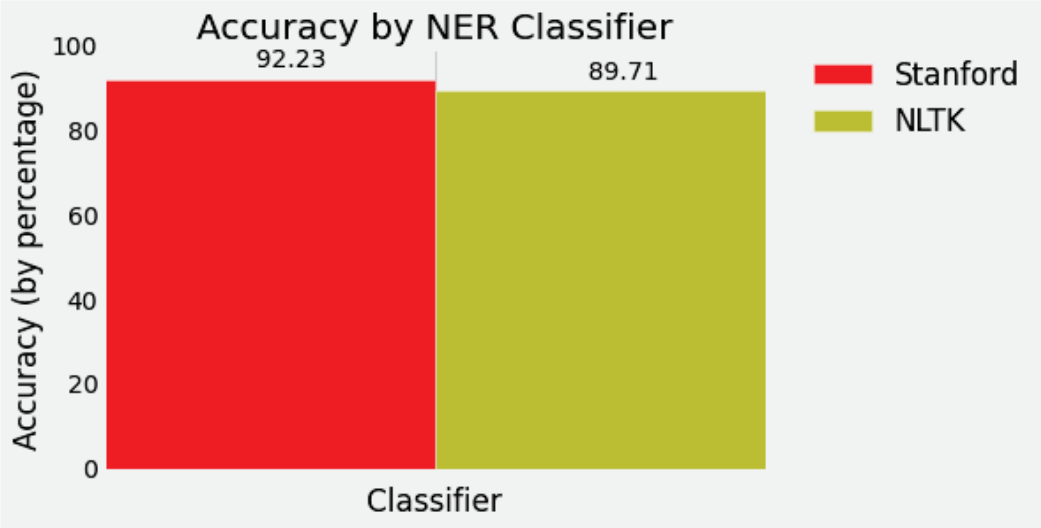


FIGURE 3.20: Comparing accuracy of the top two NER classifiers

Result Interpretation

We can interpret the obtained results using the indications described previously as follows: We can notice that the twitter-ner bolt that is used within this topology for enriching the records by extracting the context is colored red indicating suffering from a bottleneck in all the times which in-fact is not surprising due to the complexity of natural language processing and extracting the entities out of an unstructured text. We can also notice that twitter-classifier is neither running smoothly (see figure 3.15) nor overwhelmed (see figure 3.16) which is predictable since we are running the topology on one machine. The most interesting challenging performance is monitored for graph storage. This Neo4jStorage bolt was performing pretty well within the capacity (see figure 3.11) until the number of fast records reaches a certain threshold (see figure 3.13, it starts to degrade with time resulting in a red bolt

[13] https://pythonprogramming.net/testing-stanford-ner-taggers-for-accuracy/

larger in size. The highest execute latency measure was recorded between 3818.00 (check table 3.1) and 3002.16 (check table 3.6) for the Neo4jStorage bolt, however, this latency is decreasing with time leaving a good impression that it will never get worse. Although the performance of the graph storage seems to be not satisfactory, we can notice how important it is in interpreting the results and linking the different records together building up a knowledge graph as shown in figure 3.6. Besides that, it is not the only storage mechanism available, we are utilizing HDFS and local file storage.

If we look deeply into the results, we notice that all the degradation in the topology are decreasing with time and thus it will keep running normally and generating the results without any problem, e.g., twitter-ner and twitter-classifier. All these processing units are forming the topology (stream processing data-flow) which is running in our case on one machine. If we are going to use it in production, for sure we need to extend the cluster from stand-alone to multi-node cluster gaining more resources and enabling the processing components to scale and compute in parallel within the cluster and thus increasing the performance. From accuracy perspectives, we noticed as well how our classification model and context extraction model reached high accuracy measures. These measures prove that SANA's data processing is quite correct to a certain level and it is what we were looking for to have well-defined data for avoiding any delivery delay.

3.4.3 An Experiment with IBRIDIA

Experiments with Core Algorithm of IBRIDIA
As we mentioned in the previous section, IBRIDIA is a framework that is composed mainly of two different components: ProLoD and RePLoD. ProLoD is the component responsible for performing batch style processing of the data, whereas, RePLoD is responsible for performing the realtime processing of the realtime data streams. ProLod and RePLoD rely on the non-incremental and incremental versions of the algorithm respectively.

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.

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 a number of records i.e., the execution time increases as the number of records increases. However, this can be solved using the scalability of the system by inserting more nodes to the cluster for faster processing resources. We will start first by comparing the execution time of the program from the algorithm level (incremental and non-incremental algorithm). Table 3.8 shows the result of our experiment with the incremental version of the algorithm used in “RePLoD".

TABLE 3.8: The result of an experiment with the incremental version of the algorithm

Incremental	1st exec	2nd exec	3rd exec	4th exec	5th exec	avg	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. The clusters are immutable during clustering. Table 3.9 shows the results of the non-incremental version of the algorithm used in “ProLoD".

TABLE 3.9: The result of an experiment with the non-incremental version of the algorithm

Non-incremental	1st exec	2nd exec	3rd exec	4th exec	5th exec	avg	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 3.21 shows the comparison. According to our experiments, we observed that the batch-style performs better than the realtime version of the algorithm. Our study reveals that the performance of the realtime version was not satisfactory due to the centralized environment of the experiment consisting of only one node. We believe that performance will be improved significantly in a distributed and scalable computation framework with multiple nodes. We believe that technologies that we used for the 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).

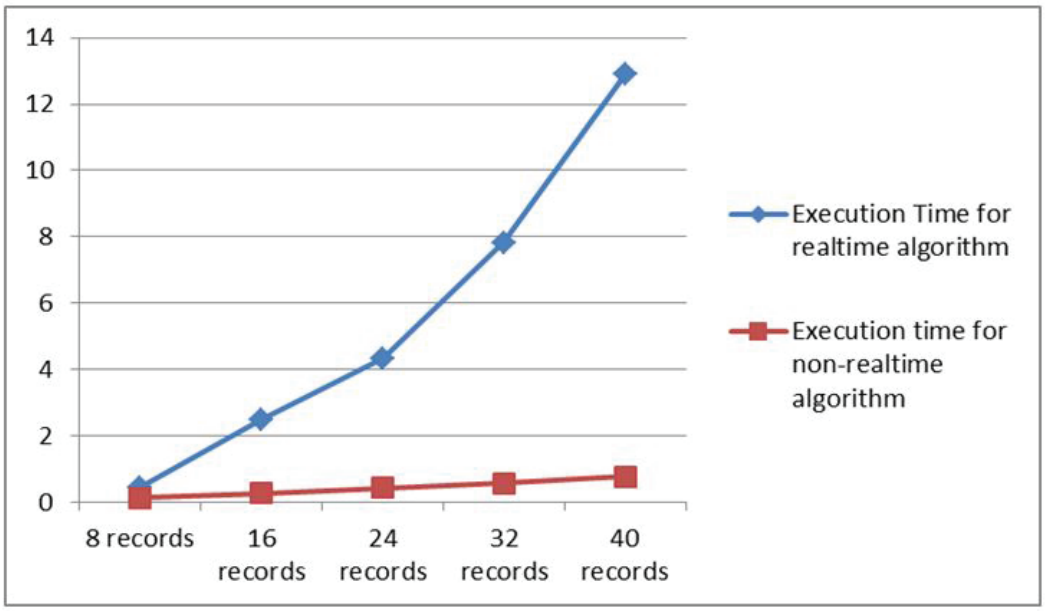


FIGURE 3.21: Execution time (in seconds) of the incremental and non-incremental versions of the algorithm

We compared our model with the hierarchical clustering algorithm of WEKA. The WEKA algorithm produced 7 clusters with 93 records. Table 3.10 shows the result which is also visualized in Figure 3.22 in-which the distribution of the records is also presented as a percentage for each cluster separately.

TABLE 3.10: 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%)

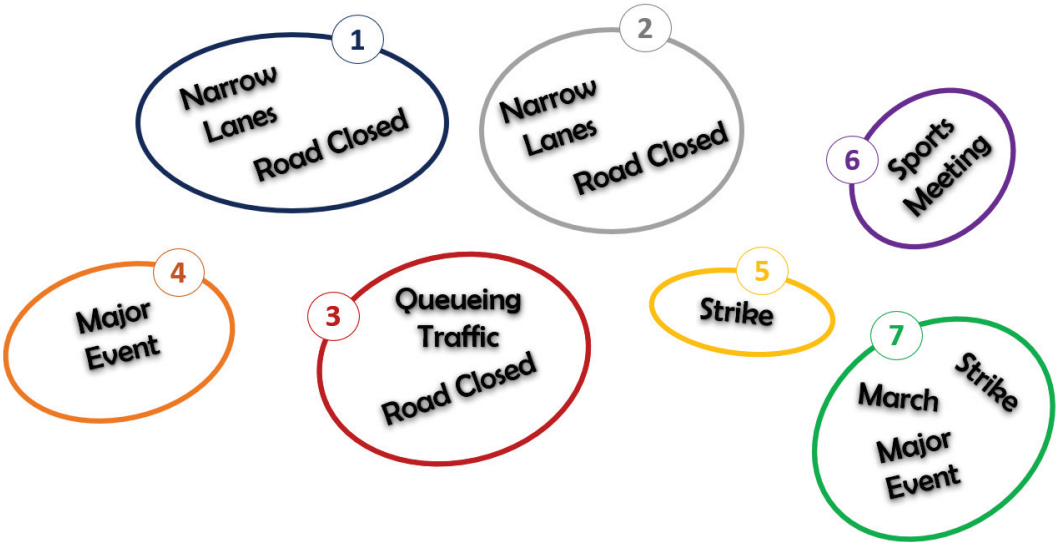


FIGURE 3.22: WEKA Text Clustering Results.

Furthermore, we observed that both realtime and batch style versions of ProLoD produced consistent and representable clusters that will assist in exploring data, discovering insights, and supporting predictive analytics when the data distribution is observed. Table 3.11 shows the result which is also shown in Figure 3.23 in-which the distribution of the records is also presented as a percentage for each cluster separately.

TABLE 3.11: 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

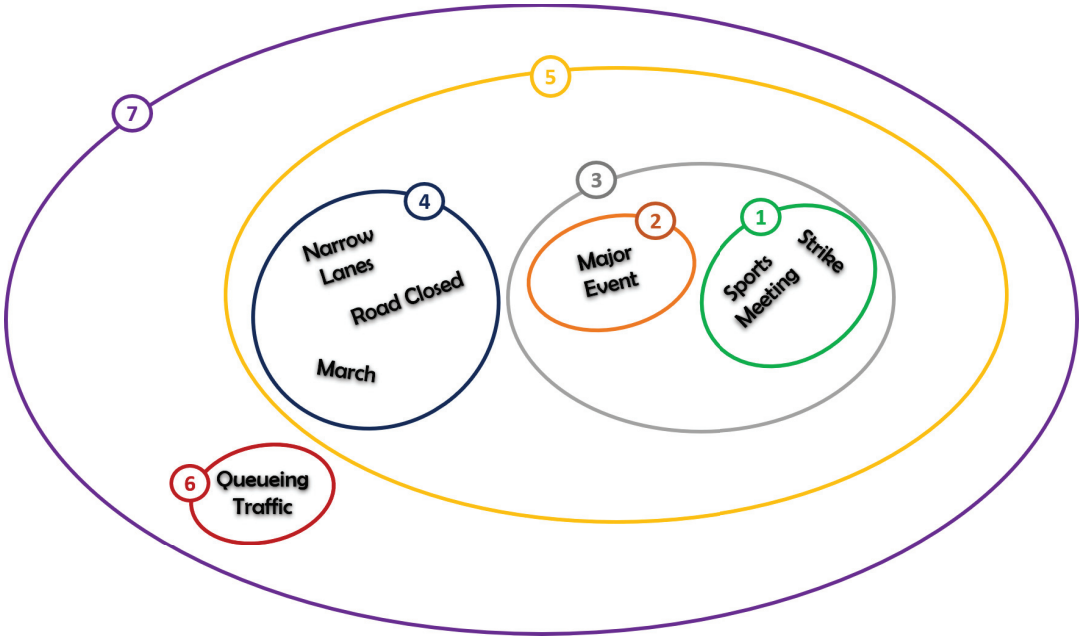


FIGURE 3.23: RePLoD Text Clustering Results.

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 3.22 some clusters such as *strike*, *sports meeting* etc. do not represent meaningful result. On the other hand, in Figure 3.23 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 analysis readable for the experts. In addition, automated systems can reap the benefits of 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 consumes a significant amount of time which was essentially unexpected. We found that the time consumption increases due to the need for 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 realtime.

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.

Experiments with the RePLoD - Realtime Component of IBRIDIA

After we come to the previous conclusion concerning the incremental algorithm, which stated that incremental algorithm performance was not satisfactory. We deployed this algorithm over Apache Storm and make it perform in

a distributed scalable environment.

For the same experimental requirements, we are going to perform our experiments over the *65 MB dataset* by simulating their realtime streaming behavior. As we mentioned earlier, using Apache Storm, it is not easy to determine the exact execution time performed for every number of records as each record is processed by different processing units and each has its own set of performance measures. Thus, we are going to present some performance measures of different time intervals for each processing unit of the topology the same as what we did in SANA. The same interpretation presented above with SANA is valid for RePLoD as well since we are using the same Storm UI settings.

We are going to present the execution of each processing unit of RePLoD independently within the same time intervals.

- For 1^{st} time interval: The emitted records were 26120 and the transferred were 26080.

TABLE 3.12: The execution measures of the processing units of the topology at time *t1*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
realtime-clustering	60	60	1.135	2393.250	80	0.00
stanford-nlp	3080	3080	0.761	57.468	3080	0.00
twitter-analytics	40	0	0.000	0.333	60	0.000
twitter-cleaner	5580	5580	0.267	8.488	5700	9.730
twitter-filter	5920	5920	0.277	11.871	5720	6.584
twitter-transformer	5740	5740	0.029	1.199	5720	0.000

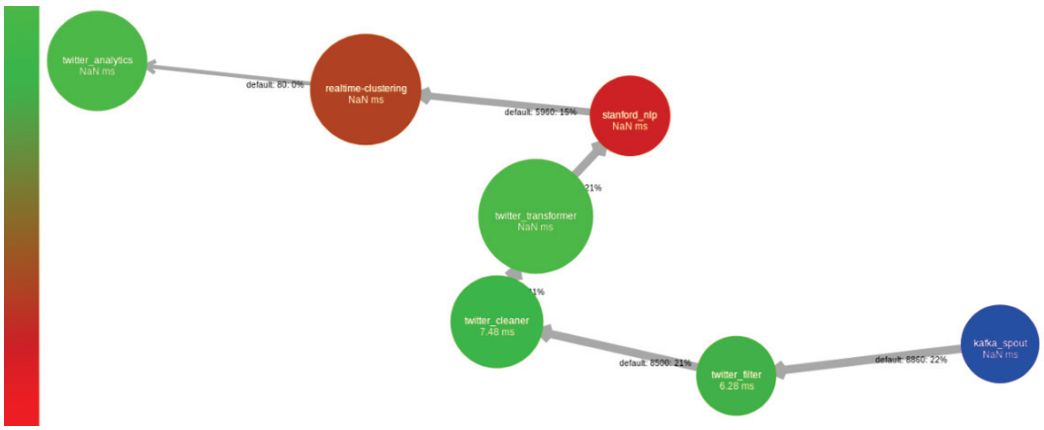


FIGURE 3.24: RePLoD visualization performance at time *t1*.

- For 2^{nd} time interval: The emitted records were 43520 and the transferred were 43460.

TABLE 3.13: The execution measures of the processing units of the topology at time *t2*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
realtime-clustering	80	80	0.585	2393.250	80	0.00
stanford-nlp	6660	6660	0.817	59.806	6700	0.00
twitter-analytics	60	0	0.000	0.333	60	0.000
twitter-cleaner	8940	8940	0.167	8.950	9180	7.057
twitter-filter	9240	9240	0.193	10.373	9160	6.070
twitter-transformer	9180	9180	0.020	0.937	9160	0.000

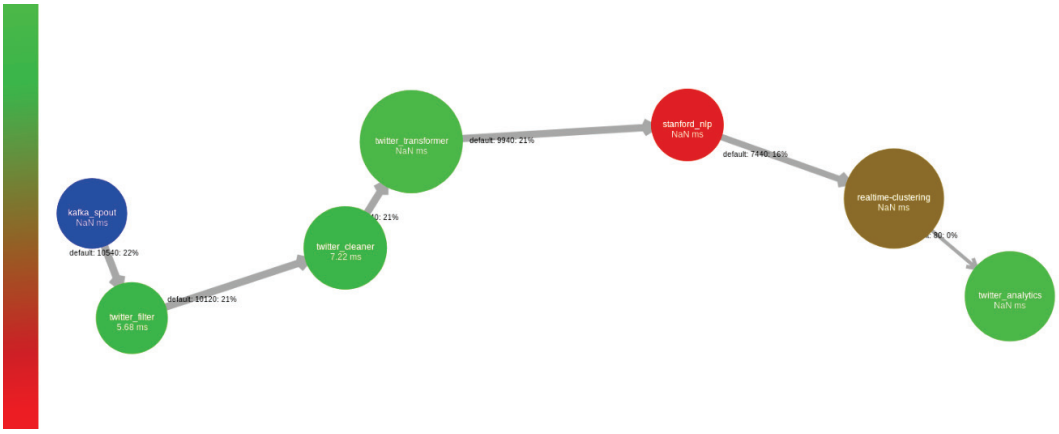


FIGURE 3.25: RePLoD visualization performance at time *t2*.

- For 3^{rd} time interval: The emitted records were 59780 and the transferred were 59720.

TABLE 3.14: The execution measures of the processing units of the topology at time *t3*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
realtime-clustering	80	80	0.429	2393.250	80	0.00
stanford-nlp	8780	8780	0.873	67.836	8800	0.00
twitter-analytics	60	0	0.000	0.333	60	0.000
twitter-cleaner	12600	12600	0.174	8.991	12660	6.876
twitter-filter	12340	12340	0.190	9.795	12660	5.572
twitter-transformer	12680	12680	0.016	0.855	12680	0.000

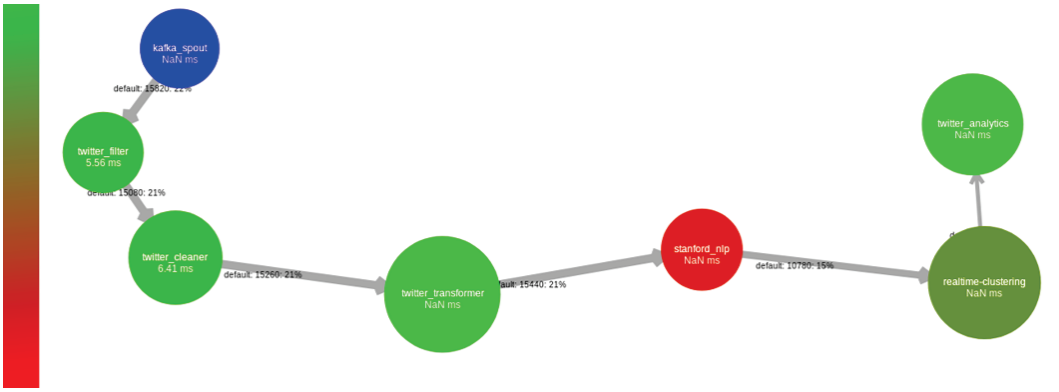


FIGURE 3.26: RePLoD visualization performance at time *t3*.

- For 4^{th} time interval: The emitted records were 118519 and the transferred were 118439.

TABLE 3.15: The execution measures of the processing units of the topology at time *t4*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
realtime-clustering	100	100	0.987	6323.600	100	0.00
stanford-nlp	18060	18060	0.927	60.697	18080	0.00
twitter-analytics	80	0	0.000	0.200	100	0.000
twitter-cleaner	24340	24340	0.148	6.375	25180	4.563
twitter-filter	25160	25160	0.163	7.087	25200	4.205
twitter-transformer	25200	25200	0.023	0.864	25200	0.000

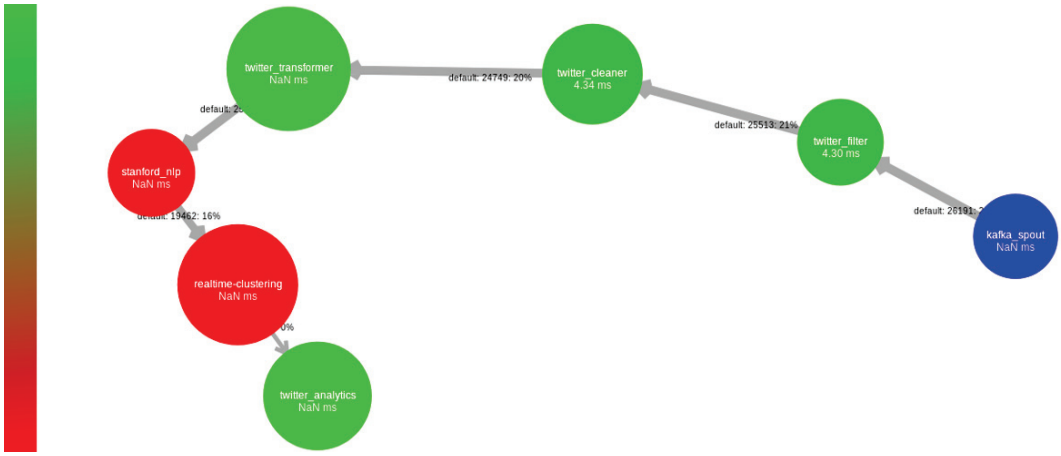


FIGURE 3.27: RePLoD visualization performance at time *t4*.

- For 5^{th} time interval: The emitted records were 160165 and the transferred were 160136.

TABLE 3.16: The execution measures of the processing units of the topology at time *t5*

Id	Emitted	Transferred	Capacity (last 10m)	Execute latency (ms)	Executed	Process latency (ms)
realtime-clustering	200	200	0.000	41550.699	200	0.000
stanford-nlp	199540	199540	1.000	44.201	199520	0.000
twitter-analytics	160	0	0.000	2.273	220	0.000
twitter-cleaner	243220	243220	0.045	2.222	244800	1.824
twitter-filter	246260	246260	0.069	3.092	244780	2.190
twitter-transformer	244800	244800	0.009	0.484	244780	0.000

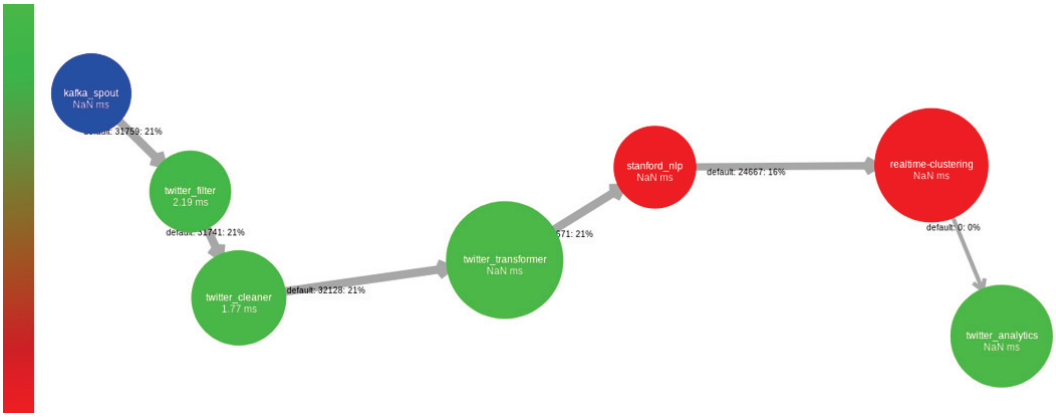


FIGURE 3.28: RePLoD visualization performance at time *t5*.

The results generated are written to the disk on a file of a size larger than 700MB and that is normal due to the appending nature of this file clearly stating all the details and showing how were these clusters evolving and changing.

Result Interpretation

We can interpret the previous results using the indications mentioned above as follows: We can notice that the stanford-nlp bolt that is used within this topology for enriching the records for reaching better clustering results is colored red indicating suffering from a bottleneck. We can clearly notice the more important indication that how the realtime-clustering bolt (our incremental hierarchical algorithm) is performing. At the beginning it was performing pretty much good in-which it was colored green indicating that it was operating within the capacity as we can see in figure 3.26. However, we can clearly notice the high execution latency reaching 2393.25 ms as shown in table 3.14. Then we can notice the changing of the color of the realtime-clustering bolt from green to red as shown in figure 3.27 and 3.28.

Maybe the degradation of the performance of the algorithm as time passes is due to the fact that the algorithm is using the comparative matrix of $(n * n)$ and the time complexity of the algorithm is at least $\mathcal{O}(n^2 log\ n)$ where n is the number of data points which is increasing over time and thus increasing the complexity. Additionally, we are using a stand-alone Storm cluster having fixed resources and this is not the best Big Data scenario in which we can scale and distribute these processing units into different machines and perform them in parallel reaching better performance.

3.4.4 Comparison Bteween SANA and IBRIDIA

Throughout our experiments, we noticed clearly the difference between both solutions. **SANA** was meant to generate a graph knowledge from the events collected immediately in realtime without any need to wait, thus reaching maximum benefit from these events. Whereas, **IBRIDIA** was created more specific to categorize the coming events into different labels. These labels have no meaning if we are going to send them immediately to a sink for analysis. In IBRIDIA, we should wait for a minimum number of events to arrive and always we have a cold start in which we have to wait for the categories to be identified upon arrival of the different events.

Despite the fact that both systems perform quite well in terms of the accuracy perspective, it is clear from the experiments that SANA out-performed IBRIDIA from a performance measuring perspective. Additionally, SANA was designed for general use cases within the logistics domain such as delivery process, identifying the risks within the warehouses (sensor-based monitoring of humans, etc.), customer-churn identification, a ranking of products, etc. In addition, IBRIDIA's clustering algorithm may seem to have an important influence within the logistics domain for general purpose analysis for identifying the most influential category of events that are affecting the delivery. However, in our case, it is more about handling the different events in order to re-optimize the route on the fly. Unfortunately, the value added by the IBRIDIA to help in minimizing delay was not such a significant indicator.

Chapter 4

Chapter Four: Route Planning

4.1 Introduction

In order to deliver the goods from the point of the supplier to the different delivery end-points, we need to do the efficient and convenient route plan that can minimize the cost and increase the revenue as much as possible. This problem is a well-known problem in the research community. Minimizing the cost is not easy, due to the different circumstances that may happen on the routes such as traffic, accidents, bad weather, etc. these conditions if not handled correctly, the delivery will be postponed or canceled which indeed may lead to customer churn and loss in revenue.

In the previous chapter, we mentioned how we were able to handle the data processing of the coming events, which are the conditions of the delivery, and thus they are not unexpected anymore. After figuring out the different unexpected events and having an anticipated overview of the whole delivery plan -thanks to SANA data processing engine-, we need to have an adequate solution over the delivery problem. This solution is not trivial as the vehicle routing problem (VRP) is known as an NP-hard problem that has no single exact solution, instead, multiple non-dominant solutions (alternative solutions). There exist multiple algorithms in the literature to generate solutions for this problem. Some of these algorithms are based on meta-heuristic and heuristic search approaches. Unfortunately, most of these algorithms are using inaccurate measuring mechanism that cannot take into consideration the impact of the events happening through the route. These different algorithms are discussed extensively in the next subsection.

4.2 Design of the Route Planner

- Vehicle Routing Problem (VRP):

  VRP is a well-known problem in the research community, such that we have a number of vehicles (m) that need to serve another number of clients (n) in which we need to consider a certain amount of time to be spent on-site service (t_s). The goal of this problem is to find the set of routes (or the best one if reached) that minimizes the cost (for example the distance or time). The VRP in literature is considered as a static problem, i.e., the delivery routes are generated assuming that there is no new client request arriving or any change in their delivery endpoint. Nowadays, as businesses seek for faster delivery of packages, an extension of the problem was created as a Dynamic Vehicle Routing problem (DVR).

- Route Planning Algorithms:

  Meta-heuristic approach based on ruin and recreate principle:
  This idea was developed by Schrimpf et al. who formulated the *ruin-and-recreate* principle to solve the various Vehicle Routing Problems (VRP). It consists of a large neighborhood search that combines elements of simulated annealing and threshold-accepting algorithms [196]. Essentially, this algorithm is mainly composed of two steps: *ruin* and *recreate*. The *ruin* step is as follows:

  – Starting with an initial solution, it disintegrates parts of the solution leading to

    * a set of jobs that are not served by a vehicle anymore

    * a partial solution containing all other jobs

  and the *recreate* step is based on the partial solution created from the *ruin* step, all jobs that are not served by a vehicle are re-integrated again, which is therefore referred to as *recreation* yielding to a new solution. If the new solution has a certain quality, it is accepted as a new best solution, whereupon a new *ruin-and-recreate* iteration starts. These steps are repeated until a certain termination criterion is met (e.g. computation time, number of iterations, etc.).

The algorithm that is discussed is an extension of the core algorithm described by Schrimpf et al. in [196] with strategies developed from the great work of Pisinger and Ropke [184].

Heuristic Search Approach:
State space search is used to model and solve the problems that their search space is divided into states that compose a particular configuration of the problem. Each of these states represents a configuration of the problem, such as the initial state represents the initial configuration of the problem. In such models, the search is performed from an initial state to a goal state by applying different transformations and actions to reach a solution to the problem. These search algorithms are classified mainly into two categories which are: uninformed and informed search. The uninformed search contains the set of algorithms that do not identify clearly which state to expand next such as Depth First Search, Breadth First Search, etc. However, informed algorithms are those techniques that use heuristics (problem-specific knowledge) that we need to estimate the distance to the goal in order to improve the performance by minimizing the number of visited states. We can divide informed search into two: global search where the exploration is over the whole state space and local search where the exploration is only over a subset of the search space.

Cluster First, Route Second (CFRS):
This algorithm consists of two phases: In the first phase, the authors of this algorithm form clusters of the delivery end-point clients taking into account vehicle capacity. These clusters are then balanced by reassigning clients. In the second phase, the authors create open routes by solving a minimum spanning tree problem (MSTP). They intend to use penalties as a modification of the MSTP solution which facilitates converting infeasible solutions to feasible solutions.

Tabu Search Algorithm (TSA):
The author can use one of the multiple methods including the nearest neighbor heuristic and the K-tree method to generate the starting point of the algorithm (initial solution). This initial solution is the new input for either the nearest neighbor method or an unstringing and stringing procedure to enhance each route. The tabu search algorithm has only two types of moves: 1) an insert move that removes a client from one

route and inserts it onto another, and 2) a swap move that exchanges two clients between two different routes.

Threshold Accepting Algorithms:
Two of the well-known threshold accepting algorithms are: backtracking adaptive threshold accepting (BATA) and list-based threshold accepting (LBTA). Threshold accepting is a deterministic variant of simulated annealing in which a threshold value $\mathcal{T}$ is specified as the upper bound on the maximum allowed value of the objective function (uphill moves can be made). In BATA, $\mathcal{T}$ is allowed to increase during the search. In LBTA, a list of values for $\mathcal{T}$ is used during the search. In these algorithms, two operations are allowed. These operations are 1-1 exchanges and 1-0 exchanges. In 1-1 exchanges, two clients are swapped from either the same or different routes. In 1-0 exchanges, one client is moved from its position in one route to another position in the same or different route. These operations are applied during the local search.

Adaptive Large Neighborhood Search (ALNS):
In ALNS, a feasible solution is generated and then modified. This algorithm is based on destroying and repairing strategy. In each iteration, an algorithm is selected to “destroy" the current solution and another algorithm is selected to “repair" the same solution. For instance, the cheapest possible route is formed by removing clients at random from the solution and reinserting them. Multiple removals and insertion heuristics can be used to diversify and intensify the search. If the solution satisfies the criteria defined by the local search procedure (e.g., simulated annealing), the solution is accepted.

- Discussion:

  We adopted the meta-heuristics approach that is based upon ruin and recreate principle for many reasons including the following:

  - It is more convenient in dealing with complex problems that have many constraints and a discontinue solution space [196].
  - It can be used for solving several classical VRP types as an all-purpose meta-heuristic.
  - It can be computed concurrently in an intuitive way.
  - Basic search strategies can adapt to the complexity of the problem by varying easily between small and large moves.

- It can create new neighborhood structures.
- If the number of search strategies was kept low, it would be simple in structure and easy to understand compared with other approaches and there is a clear distinguishing between *ruin* and *recreate* steps, probably making the checking of the constraints much easier.

4.3 Functional Design

In the previous chapter, we selected the SANA framework as a more convenient solution to prepare the well-processing data. These data are transformed into relevant events that are ready for the prescriptive analysis which will lead to a strategic decision. At first, we were interested to validate that we can rely on the SANA data processing model to make the data ready for analysis. Then, after that, we discovered a space for enhancement in which we noticed that mostly route planning algorithms are just considering an inaccurate distance function to measure the distance between two places. In this matter, we extended SANA by a new layer which we call it “Route Planning Layer". This layer is responsible for planning the whole route of the delivery and it must be able to handle the different events generated by the processing layer and do the route optimization depending on these events. We will explain the extended architecture of SANA to have a better overview of the solution. The architecture of SANA was extended by adding a new layer with three different components as shown in Fig. 4.1.

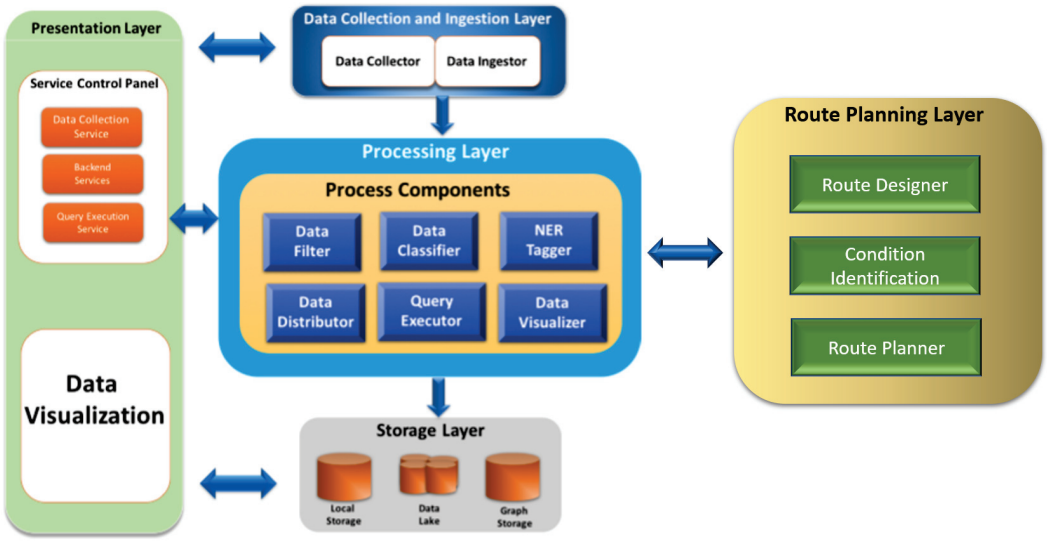


FIGURE 4.1: The Extended Architecture of SANA

The multi-layered architecture of SANA consists of various components that perform in a certain workflow as described in the following.

Data Collection and Ingestion Layer: This layer contains two components: a data collector and a data Ingestor. The data collector is a client that binds one or many data source APIs that enable access to remote repositories with an authentication check through their public keys. Once the connection is established, the data collector starts fetching data streams (*i.e.*, tweets) in realtime. The data Ingestor consists of two interfaces. The first interface taps data into SANA *data lake* which is a distributed Hadoop cluster, resides in the storage layer. The other interface opens a channel to push the events directly to the data processing components.

Data Processing Layer: The components contained in this layer perform several tasks. The two main tasks are carried out in this layer include *data analysis* and *visualization*. *Data distribution* and *query execution* are two additional tasks performed in this layer. The analysis starts with filtering incoming data. SANA's data filter eliminates unnecessary strings from events and keeps the core text required for analysis. Also, it allocates a *unique identifier* to each event. Then, the text classifier extracts and classifies the events into positive (have an impact over the delay) and negative (do not have any impact over the delay) classes. We used the multinomial naïve-bayes classifier (a machine learning technique for supervised learning) along with Chi-Square (χ^2) feature selection. The multinomial naïve-bayes classifier is used after being trained with labeled training datasets that are classified into positive and

negative classes. The Chi-Square (χ^2) function tests whether the occurrence of a specific string and the occurrence of a specific class is independent. The NER tagger extracts the contexts of classified texts. It labels the sequence of context related strings (*e.g.*, person, location, and organization) in an event. After the classification is done, the data distributor sends the results to a local disk, the data lake (Hadoop cluster), and the graph storage. Queries to find the comprehensive detail of the results is submitted through SANA's query interface.

Data Storage Layer: Two different types of storage is integrated in SANA: *data lake* and *graph storage* where the results are stored. The data lake is a cluster of nodes where data blocks are distributed. SANA adopts data lake to deal with massive-scale data. The graph-based storage of SANA assists in building a knowledge graph of classified texts and their contexts.

Presentation Layer: SANA provides a graphical user interface (GUI) which consists of a *control panel* and a text box for *data visualization*. The control panel provides three services. The *data collection service* calls and loads the data collector. The *backend services* call processing servers, the graph database server, the coordination server which maintains configuration information and provides the distributed synchronization service. The *query execution* service calls and loads the query processor. Lastly, the visualization interface loads the data visualizer and visualizes a pie chart that shows the percentage of positive and negative classes through accumulating the classified events in each class.

Route Planning Layer: This layer is the extension that was added to SANA to have the capability of dealing with the re-routing of the delivery plan in realtime. This layer mainly consists of three different components which are *route designer*, *condition identificaiton* and *route planner* (as shown in Fig. 4.2). The problem should be designed first in the route designer component. The design of the problem contains multiple parameters that should be filled with proper values. These parameters are related to all the destinations of each package, the location of departure, etc. In order for this algorithm to be applied in realtime, we have to build a new component that is responsible for detecting and checking the realtime events arriving. This component we called it condition identification. Using this component, we were able to classify the different events into critical (hard constraints) or non-critical (soft constraints). The route planner is where the algorithm actually execute given

the previous components. The route planner is built upon checking the results of condition identification while building any route designed with the route designer. Thus, trying to build a route with minimum cost and guarantees the delivery without delay.

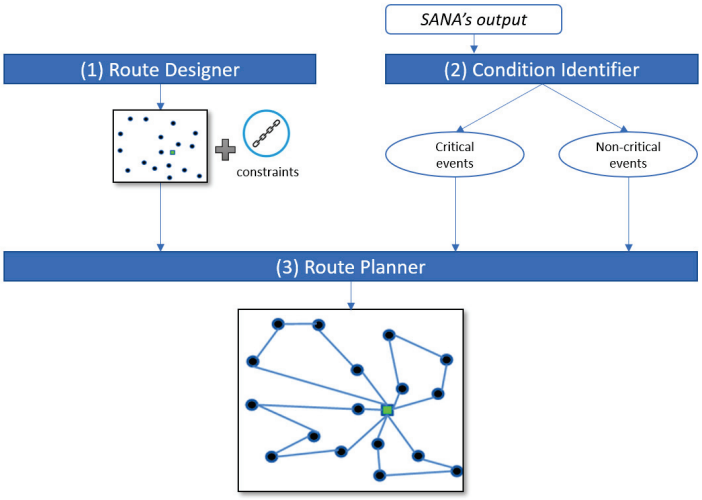


FIGURE 4.2: The principle of Route Planning Layer

In fact, the algorithm is checking the condition identification results every time a new event arrives. The main usage of these events was presented in calculating the distance between two places. As the algorithm naturally tries to minimize the paths for reaching the shortest path, the calculation of the algorithm was based on inaccurate distance functions (such as Manhattan or Euclidean distances). These functions do not represent the real distance to be covered through the roads. We insisted on building a more accurate measuring mechanism, a new distance function that can accurately measure the distance and prevent the violations by selecting different paths. The main purpose of our distance function is to find the route that can be least impacted by occurring events with minimum time:

$$\min \sum C_v \sum T_s$$

where C_v means events that might lead to constraint violation through the route (only we consider soft constraints in such a case) and T_s referred to the time that need to spent on each route.

4.4 Implementation

We were looking forward to solving the dynamic vehicle routing problem that is capable of adapting to the realtime changes in the environment. Thus, we started investigating the different available libraries and tools that are available to implement such a route planning algorithm. The main implementations of the route planning algorithms include the following:

- **jsprit**[1]**:** jsprit is a Java-based, open-source toolkit for solving rich Traveling Salesman Problems(TSP) and Vehicle Routing Problems(VRP). It is lightweight, flexible and easy to use, and based on a single all-purpose meta-heuristic currently solving many vehicle routing problems.
- **Hipster4j**[2]**:** Hipster is a Java-based easy to use, a powerful and flexible type-safe library for heuristic search. It defines search problems relying on a flexible model with generic operators. It enables us to model and solve vehicle routing problems.
- **Open-VRP**[3]**:** Open-VRP is a framework to model and solve multiple vehicle routing problems.
- **OptaPlanner**[4]**:** OptaPlanner is a Java-based lightweight, embeddable planning engine. It can be used to solve the capacitated vehicle routing problem (with time windows).
- **OR-Tools**[5]**:** Google Optimization Tools (OR-Tools) is a fast and portable software suite for solving combinatorial optimization problems, including the VRP. It is an open-source, written in C++ and available through SWIG for Python, Java, and .NET (using Mono on non-Windows platforms). You can compile OR-Tools on Linux, Mac OS X, and Windows (with Visual Studio).
- **OscaR**[6]**:** OscaR, as the abbreviation of the name indicates, it is an Open Source Toolbox for Optimising Logistics and Supply Chain Systems.

After testing and comparing these different available tools and libraries, we selected jsprit as one of the best candidate solutions to solve the delivery

[1] https://github.com/graphhopper/jsprit
[2] http://www.hipster4j.org/
[3] https://github.com/mck-/Open-VRP
[4] https://www.optaplanner.org/
[5] https://developers.google.com/optimization/
[6] https://bitbucket.org/oscarlib/oscar/wiki/Home

problem we are tackling. Besides, it is lightweight, flexible and easy to use and more importantly, it is fit for change and extension.

We considered building our route planning layer using *jsprit* algorithm to facilitates the planning of the delivery process. *Jsprit* is a java-based open source toolkit for solving rich Travelling Salesman Problems (TSP) and Vehicle Routing Problems (VRP). It is based upon the theory studied previously and currently it is capable of solving the following problems: capacitated VRP, multiple depot VRP, VRP with time windows, VRP with backhauls, VRP with pickups and deliveries, VRP with heterogeneous fleet, time-dependent VRP, traveling salesman problem, dial-a-ride Problem, and various combination of these types.

This algorithm was basically built to deal with the static planning and it is not designed to handle the dynamic re-routing in realtime. The advantage of using such an algorithm is that it can fit for change and extension due to a modular design and a comprehensive set of unit and integration tests. Therefore, we insisted on extending this algorithm with several features for performing the dynamic re-planning of the route.

One of the most important features in the accuracy of the algorithm is to select the right distance function. Several distance functions are available for calculating the distance between two points (a and b) for example: Manhattan distance ($|x_a - x_b| + |y_a - y_b|$) and Euclidean distance ($\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$). Unfortunately, these distance functions cannot be used in calculating the distance between two geolocations because they do not represent the roads to get from one geolocation to another. Thus, we were obliged to implement our own distance function based on an online Geolocation API such as Google Direction API[7] and MapQuest Directions API[8]. We used the latter as it offers a larger number of API's requests for free and it has nearly the same accuracy. Besides that, we added another feature which is the ability to determine if an arrived event represents a soft or hard constraint. The prerequisite for building such a distance function is that it requires to have more than one alternative route between two places that will help in selecting the best fitting route's path. To the best of our luck, we found that MapQuest Directions API is capable of generating three different alternative routes between two geolocations. This capability of having 3 different routes between every two geolocations with the ability to determine critical and non-critical events

[7] https://developers.google.com/optimization/routing/google_direction
[8] https://developer.mapquest.com/documentation/directions-api/

affecting the roads, we are getting near to find the solution that may fit real-world scenarios. In that sense, we can calculate accurately if it is possible to *eliminate or minimize the delay* and what the expected time to arrive at a certain destination is.

SANA needed to be extended in order to reach a full framework that can collect, process, analyze, store and visualize. Thus, we extended SANA by adding a new missing processing unit that allows the sending of more enriched data to the route planning layer. This processing unit was added and named **Geo-coding bolt** which is responsible for geocoding the locations extracted from the context and label the arrived events as critical or non-critical events. The process of geocoding was achieved by using another service provided by the MapQuest API which is called MapQuest Geocoding API[9] which enables us to associate latitude and longitude with an associated address. The labeling of the arrived events depends on some keywords/dictionary in-which it will classify the events for example as critical if they contain the terrorist attack and non-critical in case of traffic, accident, road, etc.

As shown in figure 4.3, this processing unit (bolt) was integrated with SANA as a sink to do the previously described tasks and then push these enriched events to the route planning layer which is listening as a server. The server will handle these events as shown in figure 4.4 and then it will add them to one of two lists depending upon their labels. If the event's label was critical, then add the coordinates of the event to the list of critical events (i.e., hard constraints), otherwise, add them to the list of non-critical events (i.e., soft constraints). In the previous figure, just note that we have hidden the other storage bolts (graph storage and Hadoop storage) for experimental purposes.

[9] https://developer.mapquest.com/documentation/geocoding-api/

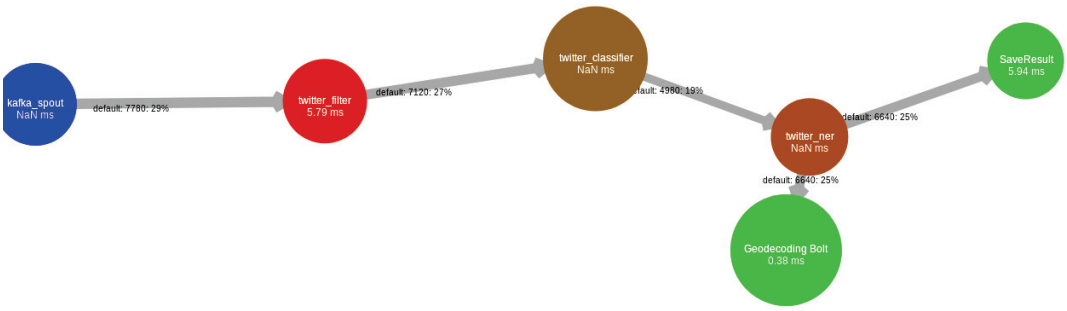


FIGURE 4.3: The topology of SANA after the integration with the route planning component

```
Please select an option:
1: start the server
2: stop the server
3: launch the routing optimization algorithm
4: quit the program
1
Starting server...
Please select an option:
1: start the server
2: stop the server
3: launch the routing optimization algorithm
4: quit the program
Received: 31.400038;49.024101;0
Received: 31.400038;49.024101;0
```

FIGURE 4.4: The route planning events server is listening to the events from SANA to arrive.

During the execution of the route planner, we are calculating the distance between each of the locations to be visited by the truck. Thus, we considered the following technical steps in building our enhanced distance function for more accurate distance measuring:

1. Fill all the possible routes between these two locations (main route and two other alternative routes).
2. Select the best route according to the following:
   (a) For each route, we have the bounding box of that route -bounding box in geographic coordinates is an area defined by a minimum and maximum longitudes and latitudes-.
   (b) Check if any of the coordinates of the lists (critical and non-critical) falls within this bounding box and if so move them to a new list according to their labels.

(c) Check the points of each route if they have any points that are also found in a list of critical events, then we discard this route.

(d) We repeat the previous step for the alternative routes one by one. Note that if all the routes are affected by critical events then there is no way to eliminate or minimize the delivery (we cannot select routes impacted by critical events).

(e) Check if one of the points of the selected route is also found in the list of non-critical, if so, then we go over all the routes that are not critically affected.

(f) Choose the route with the minimum number of non-critical events affecting it.

(g) Return the distance.

These previous steps used in the distance function should be repeated for calculating all the distances of every possibility between the different locations of the route until it generates the result.

Figure 4.5 shows an example of the result generated by the algorithm implemented in the route planning layer depending on the events pushed by the processing model of SANA. The source code of the route planning component can be found on Github at the following link: https://github.com/m-alshaer/DeliveryModel.

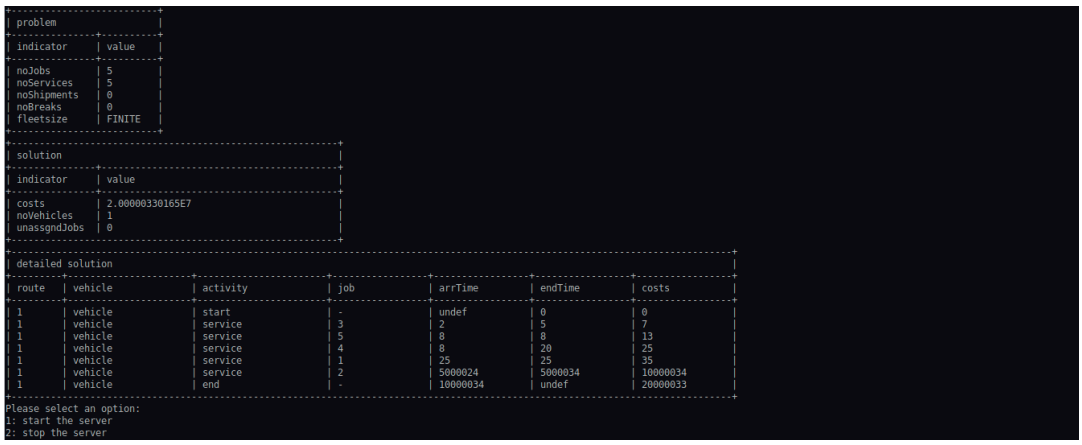


FIGURE 4.5: The result of the route planning algorithm

4.5 Conclusion

According to our implementation and experiments, we deduced that we were able to do the required route planning algorithm depending on the data processed by SANA. Thus, SANA proved to provide the required information for achieving the dynamic routing of the delivery and a convenient candidate for processing logistics data in realtime. The route planning algorithm used in SANA was based upon the *ruin and recreate* principle of the meta-heuristic approach that can be used for solving several classical VRP problems. We noticed that this algorithm, as well as, other algorithms lack the right distance measuring function. Thus, we extended this algorithm with several important features to perform dynamic planning. These features were added to serve one purpose which is the accurate measuring of the distance between the different geolocations. Determining the level of influence of the events over the delivery and utilizing these events in a well-defined distance function to compute and decide more realistically the route plan to be selected. Through our distance function, we guarantee a better distance measuring taking into consideration the current events that might impact the routes.

Chapter 5

Chapter Five: Conclusion and Future Work

5.1 Conclusion

Before the hype of Big Data, businesses were highly dependent on their own data. After that, with the advent of Big Data, they are no longer depending on their internal data alone, however, they seek to enrich it from external data (data that is found on the web, e.g., social media and sensor data). We noticed a big shift from business application perspectives, where they used to design user-specific applications, instead, nowadays, they are adopting data-driven applications. These data-driven applications can dynamically adapt to any changes in the environment by detecting and pro-acting to these changes accordingly to avoid violating any business's constraints.

Throughout this thesis, we described the problem that we are tackling as avoiding the delay in the delivery process of the supply chain management systems. This problem prompts different research questions which can be briefed as follows: 1) handling data variety while collecting data in real-time from different sources; 2) data processing which is considered as one of the most critical issues for analysis especially if we considered the extraction of the relevant events from the coming data in realtime; 3) optimizing the route plan in order to mitigate or eliminate the delay which is known as NP-hard problem with no unique solution to be selected. We intend to seek data processing technology that may deal efficiently with the first two questions. Thus, we did an extensive study to find suitable models, techniques, and methods that may help in achieving our objectives. The study reveals many important factors in terms of approaching the solution such as the importance of collecting data from sensors, social media, and different

web pages especially if correlated with the logistics data. In light of data processing, we have witnessed two modes of data processing batch-style and realtime. Besides, we realized other methods and techniques that can help in processing textual data (unstructured text) such as Multi-nomial Naïve Bayes Classifier, Named Entity Recognizer (NER), Hierarchical clustering algorithm with Hamming distance as a distance function.

Based on our understanding of the existing approaches and techniques, we conduct an experimental study between two different frameworks that we implemented on our own. The expected output of this study was to show which one of these frameworks would fit better to preparing the data for the prescriptive analysis to help to reach our goal in avoiding the violation due to the delay. These frameworks are SANA and IBRIDIA. We contributed to each of these frameworks independently to show their impact in terms of dealing with the well-known research challenges velocity and variety.

The contribution of SANA was its capability of handling fast data with the ability to extract the context of the text and presenting them as entities (e.g. location, organization, etc.). Besides, it was able to classify the coming text and capable of visualizing them as a knowledge graph. We would rather mention that the importance of SANA was not in building a new technique or method, rather it was the integration of different methods and techniques into one unique model that is able to produce well-processed data ready for analysis.

The contribution of IBRIDIA was mainly in the scientific part that was missing in the literature which was modifying 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. After observing the results obtained and validating them with respect to WEKA, our algorithm produced better representative results. It was designed as a hybrid approach that can deal with both data at rest using its module ProLoD and data in motion using its module RePLoD.

From the comparative analysis between these two frameworks, we deduced that SANA out-performed IBRIDIA from performance measuring perspectives. It is clearly said that SANA was meant to generate a graph knowledge from the events collected immediately in realtime without any need to wait, thus reaching maximum benefit from these events. Whereas, IBRIDIA have an important influence within the logistics domain for identifying the most influential category of events that are affecting the delivery. Unfortunately,

in IBRIRDIA, we should wait for a minimum number of events to arrive and always we have a cold start. Due to the fact that we are interested in re-optimizing the route on the fly, we adopted SANA as our data processing framework.

We did not stop on our comparative analysis for selecting our data processing framework, we build a route planning algorithm for validating our framework. In order to build a promising route planning algorithm, we studied again the different route planning techniques used and their implementations to be able to perform the re-routing of the delivery plan in realtime. As to achieve the prescriptive analysis of the plan, we adopted the meta-heuristic approach which was developed by Schrimpf et al based on the *ruin-and-create* principle due to its advantage in dealing with complex routing plans. We discussed in details our implementation and presented the result of applying it in cooperation with SANA and the results were promising for us.

5.2 Future Work

Our work still has a high potential for future improvements. On the data processing level, I believe that there would be more tremendous work to be done. We are exploiting the data and processing them on the fly. However, if the data sources become too many, how many nodes we might need to process all these data arriving at the same moment. In IBRIDIA, we can do some additional modification for the algorithm to determine the exact number of clusters that should be created (i.e. at which level should the algorithm stop so that we really benefit the most from its representative clusters). Besides that, we can seek for the best convenient way to build a predictive model that can use the clustered dataset to produce some rules that may help in detecting certain patterns to have more anticipation over the transportation problems. Concerning SANA, we can test it with different domains and see how it can perform such as military fields and educational sectors in order to help in detecting terrorist attacks and their locations or how to improve the services in the educational sectors to help students and instructors have a better experience.

The major suggested improvement would be around the re-route optimization algorithm as we need to do more benchmarking with other available

solutions and how we can minimize its execution time to get the new plan in less amount of time. I believe that this thesis is just the beginning for helping the logistics sector in building better systems from the scientific point of view to help in minimizing their loss by predicting and avoiding the possible delays.

Appendix A

List of Publications

This appendix contains the list of publications throughout the thesis.

1. Mohammed AlShaer, Yehia Taher, Rafiqul Haque, Mohand-Saïd Hacid, Mohamed Dbouk: IBRIDIA: A hybrid solution for processing big logistics data. Future Generation Comp. Syst. 97: 792-804 (2019)

2. Mohammed AlShaer, Yehia Taher, Rafiqul Haque, Mohand-Saïd Hacid, Mohamed Dbouk: ProLoD: An Efficient Framework for Processing Logistics Data. OTM Conferences (1) 2017: 698-715

3. Yehia Taher, Rafiqul Haque, Mohammed AlShaer, Willem-Jan van den Heuvel, Karine Zeitouni, Renata Mendes de Araujo, Mohand-Saïd Hacid, Mohamed Dbouk: A Service-Based System for Sentiment Analysis and Visualization of Twitter Data in Realtime. ICSOC Workshops 2016: 199-202

4. Yehia Taher, Rafiqul Haque, Mohammed AlShaer, Willem-Jan van den Heuvel, Mohand-Saïd Hacid, Mohamed Dbouk: A Context-Aware Analytics for Processing Tweets and Analysing Sentiment in Realtime (Short Paper). OTM Conferences 2016: 910-917

Bibliography

[1] Mongi A Abidi and Rafael C Gonzalez. *Data fusion in robotics and machine intelligence*. Academic Press Professional, Inc., 1992.

[2] Azza Abouzeid et al. "HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads". In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 922–933.

[3] Ittai Abraham et al. "Hierarchical hub labelings for shortest paths". In: *European Symposium on Algorithms*. Springer. 2012, pp. 24–35.

[4] Ittai Abraham et al. "Highway dimension and provably efficient shortest path algorithms". In: *Journal of the ACM (JACM)* 63.5 (2016), p. 41.

[5] Charu C Aggarwal and ChengXiang Zhai. *Mining text data*. Springer Science & Business Media, 2012.

[6] Rakesh Agrawal et al. *Automatic subspace clustering of high dimensional data for data mining applications*. Vol. 27. 2. ACM, 1998.

[7] Takuya Akiba, Yoichi Iwata, and Yuichi Yoshida. "Fast exact shortest-path distance queries on large networks by pruned landmark labeling". In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM. 2013, pp. 349–360.

[8] Takuya Akiba et al. "Fast shortest-path distance queries on road networks by pruned highway labeling". In: *2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2014, pp. 147–154.

[9] S. ALASADI and W. BHAYA. "Review of data preprocessing techniques in data mining". In: *Journal of Engineering and Applied Sciences* 12.18 (2017), pp. 4102–4107.

[10] Mehdi Allahyari et al. "A brief survey of text mining: Classification, clustering and extraction techniques". In: *arXiv preprint arXiv:1707.02919* (2017).

[11] Mohammad AlShaer et al. "ProLoD: An Efficient Framework for Processing Logistics Data". In: *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer. 2017, pp. 698–715.

[12] SS Aravinth et al. "An efficient HADOOP frameworks SQOOP and ambari for big data processing". In: *International Journal for Innovative Research in Science and Technology* 1.10 (2015), pp. 252–255.

[13] Michael Armbrust et al. "Spark sql: Relational data processing in spark". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, pp. 1383–1394.

[14] Julian Arz, Dennis Luxen, and Peter Sanders. "Transit node routing reconsidered". In: *International Symposium on Experimental Algorithms*. Springer. 2013, pp. 55–66.

[15] C Asensio et al. "GPS-based speed collection method for road traffic noise mapping". In: *Transportation Research Part D: Transport and Environment* 14.5 (2009), pp. 360–366.

[16] Ricardo Baeza-Yates and Carlos Castillo. "Crawling the infinite Web: five levels are enough". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2004, pp. 156–167.

[17] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation clustering". In: *Machine learning* 56.1-3 (2004), pp. 89–113.

[18] Andrea Baraldi and Palma Blonda. "A survey of fuzzy clustering algorithms for pattern recognition. I". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29.6 (1999), pp. 778–785.

[19] Chaitanya Baru and Gilles Fecteau. "An overview of DB2 parallel edition". In: *ACM SIGMOD Record*. Vol. 24. 2. ACM. 1995, pp. 460–462.

[20] Hannah Bast et al. "Route planning in transportation networks". In: *Algorithm engineering*. Springer, 2016, pp. 19–80.

[21] Holger Bast, Stefan Funke, and Domagoj Matijević. "Transit: ultrafast shortest-path queries with linear-time preprocessing". In: *9th DIMACS Implementation Challenge—Shortest Path*. 2006.

[22] Holger Bast et al. "Fast routing in road networks with transit nodes". In: *Science* 316.5824 (2007), pp. 566–566.

[23] Richard Bellman. "On a routing problem". In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90.

[24] Luís MA Bettencourt et al. "The power of a good idea: Quantitative modeling of the spread of ideas from epidemiological models". In: *Physica A: Statistical Mechanics and its Applications* 364 (2006), pp. 513–536.

[25] James C Bezdek, EC-K Tsao, and Nikhil R Pal. "Fuzzy Kohonen clustering networks". In: *Fuzzy Systems, 1992., IEEE International Conference on*. IEEE. 1992, pp. 1035–1043.

[26] Daniel M Bikel et al. "Nymble: a high-performance learning name-finder". In: *Proceedings of the fifth conference on Applied natural language processing*. Association for Computational Linguistics. 1997, pp. 194–201.

[27] MKABV Bittorf et al. "Impala: A modern, open-source SQL engine for Hadoop". In: *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research*. 2015.

[28] Konstantinos Blekas and Isaac E Lagaris. "Newtonian clustering: An approach based on molecular dynamics and global optimization". In: *Pattern Recognition* 40.6 (2007), pp. 1734–1744.

[29] Adi Botea, Daniel Harabor, et al. "Path Planning with Compressed All-Pairs Shortest Paths Data." In: *ICAPS*. 2013, pp. 288–292.

[30] Adi Botea et al. "Ultra-Fast Optimal Pathfinding without Runtime Search." In: *AIIDE*. 2011.

[31] Ulrik Brandes et al. "Travel planning with self-made maps". In: *Workshop on Algorithm Engineering and Experimentation*. Springer. 2001, pp. 132–144.

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

[33] Leo Breiman et al. "347 Notation Index". In: *Classification and Regression Trees*. CRC Press, 1984, pp. 1–17.

[34] Deng Cai, Xiaofei He, and Jiawei Han. "Semi-supervised discriminant analysis". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–7.

[35] Deng Cai, Chiyuan Zhang, and Xiaofei He. "Unsupervised feature selection for multi-cluster data". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2010, pp. 333–342.

[36] Salvatore A Catanese et al. "Crawling facebook for social network analysis purposes". In: *Proceedings of the international conference on web intelligence, mining and semantics*. ACM. 2011, p. 52.

[37] Gilles Celeux and Gérard Govaert. "Gaussian parsimonious clustering models". In: *Pattern recognition* 28.5 (1995), pp. 781–793.

[38] Soumen Chakrabarti et al. "Using taxonomy, discriminants, and signatures for navigating in text databases". In: *VLDB*. Vol. 97. 1997, pp. 446–455.

[39] Yee Seng Chan and Dan Roth. "Exploiting background knowledge for relation extraction". In: *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics. 2010, pp. 152–160.

[40] Yee Seng Chan and Dan Roth. "Exploiting syntactico-semantic structures for relation extraction". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 551–560.

[41] Xiaojun Chang et al. "A Convex Formulation for Semi-Supervised Multi-Label Feature Selection." In: *AAAI*. 2014, pp. 1171–1177.

[42] Moses Charikar et al. "Incremental clustering and dynamic information retrieval". In: *SIAM Journal on Computing* 33.6 (2004), pp. 1417–1440.

[43] Duen Horng Chau et al. "Parallel crawling for online social networks". In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 1283–1284.

[44] Hsinchun Chen, Roger HL Chiang, and Veda C Storey. "Business intelligence and analytics: From big data to big impact." In: *MIS quarterly* 36.4 (2012).

[45] Qiang Cheng, Hongbo Zhou, and Jie Cheng. "The fisher-markov selector: fast selecting maximally separable feature subset for multiclass classification with applications to high-dimensional data". In: *IEEE transactions on pattern analysis and machine intelligence* 33.6 (2011), pp. 1217–1233.

[46] Boris V Cherkassky, Andrew V Goldberg, and Craig Silverstein. "Buckets, heaps, lists, and monotone priority queues". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1326–1346.

[47] Hai Leong Chieu and Hwee Tou Ng. "Named entity recognition: a maximum entropy approach using global information". In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 2002, pp. 1–7.

[48] Krzysztof J Cios, Witold Pedrycz, and Roman W Swiniarski. "Data mining and knowledge discovery". In: *Data mining methods for knowledge discovery*. Springer, 1998, pp. 1–26.

[49] Paolo Corsini, Beatrice Lazzerini, and Francesco Marcelloni. "A fuzzy relational clustering algorithm based on a dissimilarity measure extracted from data". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34.1 (2004), pp. 775–781.

[50] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[51] Nilesh Dalvi, Ashwin Machanavajjhala, and Bo Pang. "An analysis of structured data on the web". In: *Proceedings of the VLDB Endowment* 5.7 (2012), pp. 680–691.

[52] George Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[53] Pasquale De Meo et al. "Finding reliable users and social networks in a social internetworking system". In: *Proceedings of the 2009 International Database Engineering & Applications Symposium*. ACM. 2009, pp. 173–181.

[54] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[55] Daniel Delling, Andrew V Goldberg, and Renato F Werneck. "Hub label compression". In: *International Symposium on Experimental Algorithms*. Springer. 2013, pp. 18–29.

[56] Daniel Delling and Renato F Werneck. "Faster customization of road networks". In: *International Symposium on Experimental Algorithms*. Springer. 2013, pp. 30–42.

[57] Daniel Delling et al. "Customizable route planning in road networks". In: *Transportation Science* 51.2 (2015), pp. 566–591.

[58] Daniel Delling et al. "Engineering route planning algorithms". In: *Algorithmics of large and complex networks*. Springer, 2009, pp. 117–139.

[59] Daniel Delling et al. *Graph partitioning with natural cuts*. US Patent 8,738,559. 2014.

[60] Daniel Delling et al. "High-performance multi-level routing". In: *The Shortest Path Problem: Ninth DIMACS Implementation Challenge* 74 (2009), pp. 73–92.

[61] Daniel Delling et al. "PHAST: Hardware-accelerated shortest path trees". In: *Journal of Parallel and Distributed Computing* 73.7 (2013), pp. 940–952.

[62] David J DeWitt et al. *A High Performance Dataflow Database Machine*. Computer Science Department, University of Wisconsin, 1986.

[63] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[64] Chris Ding and Hanchuan Peng. "Minimum redundancy feature selection from microarray gene expression data". In: *Journal of bioinformatics and computational biology* 3.02 (2005), pp. 185–205.

[65] Koustabh Dolui, Srijani Mukherjee, and Soumya Kanti Datta. "Smart device sensing architectures and applications". In: *Computer Science and Engineering Conference (ICSEC), 2013 International*. IEEE. 2013, pp. 91–96.

[66] Xin Luna Dong and Divesh Srivastava. "Big data integration". In: *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE. 2013, pp. 1245–1248.

[67] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[68] Jennifer G Dy and Carla E Brodley. "Feature selection for unsupervised learning". In: *Journal of machine learning research* 5.Aug (2004), pp. 845–889.

[69] Alexandros Efentakis and Dieter Pfoser. "Optimizing landmark-based routing and preprocessing". In: *Proceedings of the Sixth ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM. 2013, p. 25.

[70] Alexandros Efentakis, Dieter Pfoser, and Agnès Voisard. "Efficient data management in support of shortest-path computation". In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM. 2011, pp. 28–33.

[71] David Eppstein and Michael T Goodrich. "Studying (non-planar) road networks through an algorithmic lens". In: *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. ACM. 2008, p. 16.

[72] Martin Ester et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.

[73] Vladimir Estivill-Castro. "Why so many clustering algorithms: a position paper". In: *ACM SIGKDD explorations newsletter* 4.1 (2002), pp. 65–75.

[74] BS Everitt, S Landau, and M Leese. *Cluster Analysis.: Arnold, A member of the Hodder Headline Group*. 2001.

[75] Emilio Ferrara et al. "Web data extraction, applications and techniques: A survey". In: *Knowledge-based systems* 70 (2014), pp. 301–323.

[76] Maurizio Filippone et al. "A survey of kernel and spectral methods for clustering". In: *Pattern recognition* 41.1 (2008), pp. 176–190.

[77] Douglas Fisher. "Optimization and Simplification of Hierarchical Clusterings." In: *KDD*. 1995, pp. 118–123.

[78] Ronald A Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of eugenics* 7.2 (1936), pp. 179–188.

[79] Lester R Ford Jr. *Network flow theory*. Tech. rep. RAND CORP SANTA MONICA CA, 1956.

[80] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[81] Mark A Friedl and Carla E Brodley. "Decision tree classification of land cover from remotely sensed data". In: *Remote sensing of environment* 61.3 (1997), pp. 399–409.

[82] Hichem Frigui and Raghu Krishnapuram. "Clustering by competitive agglomeration". In: *Pattern recognition* 30.7 (1997), pp. 1109–1119.

[83] Liping Fu, D Sun, and Laurence R Rilett. "Heuristic shortest path algorithms for transportation applications: state of the art". In: *Computers & Operations Research* 33.11 (2006), pp. 3324–3343.

[84] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[85] Tal Galili. "dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering". In: *Bioinformatics* 31.22 (2015), pp. 3718–3720.

[86] Robert Geisberger and Dennis Schieferdecker. "Heuristic contraction hierarchies with approximation guarantee". In: *Third Annual Symposium on Combinatorial Search*. 2010.

[87] Julia Gelfand and Locke Morrisey. *Selective guide to literature on artificial intelligence and expert systems*. American Society for Engineering Education, Engineering Libraries Division, 1992.

[88] Minas Gjoka et al. "Walking in facebook: A case study of unbiased sampling of osns". In: *Infocom, 2010 Proceedings IEEE*. Ieee. 2010, pp. 1–9.

[89] Andrew V Goldberg. "A practical shortest path algorithm with linear expected time". In: *SIAM Journal on Computing* 37.5 (2008), pp. 1637–1655.

[90] Andrew V Goldberg and Chris Harrelson. "Computing the shortest path: A search meets graph theory". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2005, pp. 156–165.

[91] Andrew V Goldberg, Haim Kaplan, and Renato F Werneck. "Reach for A*: Shortest Path Algorithms with Preprocessing." In: *The Shortest Path Problem*. 2006, pp. 93–140.

[92] Jonatan Gomez, Dipankar Dasgupta, and Olfa Nasraoui. "A new gravitational clustering algorithm". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM. 2003, pp. 83–94.

[93] Teofilo F Gonzalez. "On the computational complexity of clustering and related problems". In: *System modeling and optimization*. Springer, 1982, pp. 174–182.

[94] Siddharth Gopal and Yiming Yang. "Multilabel classification with meta-level features". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2010, pp. 315–322.

[95] K Chidananda Gowda and Edwin Diday. "Symbolic clustering using a new dissimilarity measure". In: *pattern recognition* 24.6 (1991), pp. 567–578.

[96] Ronald L Graham and Pavol Hell. "On the history of the minimum spanning tree problem". In: *Annals of the History of Computing* 7.1 (1985), pp. 43–57.

[97] Sudipto Guha et al. "Clustering data streams: Theory and practice". In: *IEEE transactions on knowledge and data engineering* 15.3 (2003), pp. 515–528.

[98] Francesco Gullo, Giovanni Ponti, and Andrea Tagarelli. "Clustering uncertain data via k-medoids". In: *International Conference on Scalable Uncertainty Management*. Springer. 2008, pp. 229–242.

[99] Francesco Gullo and Andrea Tagarelli. "Uncertain centroid based partitional clustering of uncertain data". In: *Proceedings of the VLDB Endowment* 5.7 (2012), pp. 610–621.

[100] Francesco Gullo et al. "A hierarchical algorithm for clustering uncertain data via an information-theoretic approach". In: *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE. 2008, pp. 821–826.

[101] Zhou GuoDong et al. "Exploring various knowledge in relation extraction". In: *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2005, pp. 427–434.

[102] Ronald J Gutman. "Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks." In: *ALENEX/ANALC* 4 (2004), pp. 100–111.

[103] Isabelle Guyon et al. "Gene selection for cancer classification using support vector machines". In: *Machine learning* 46.1-3 (2002), pp. 389–422.

[104] DL Hall and J Llinas. "A challenge for the data fusion community I: Research imperatives for improved processing". In: *Proc. 7th Natl. Symp. on Sensor Fusion*. 1994.

[105] DL Hall and SAH McMullen. *Mathematical techniques in multisensor data fusion. Boston, MA: Artech House*. 1992.

[106] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. "Text categorization using weight adjusted k-nearest neighbor classification". In: *Pacific-asia conference on knowledge discovery and data mining*. Springer. 2001, pp. 53–65.

[107] Andreas Harth, Jürgen Umbrich, and Stefan Decker. "Multicrawler: A pipelined architecture for crawling and indexing semantic web data". In: *International Semantic Web Conference*. Springer. 2006, pp. 258–271.

[108] Xiaofei He, Deng Cai, and Partha Niyogi. "Laplacian score for feature selection". In: *Advances in neural information processing systems*. 2006, pp. 507–514.

[109] Juan C Herrera et al. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment". In: *Transportation Research Part C: Emerging Technologies* 18.4 (2010), pp. 568–583.

[110] M Heutger et al. *Self-driving vehicles in logistics*. 2014.

[111] Moritz Hilger et al. "Fast point-to-point shortest path computations with arc-flags". In: *The Shortest Path Problem: Ninth DIMACS Implementation Challenge* 74 (2009), pp. 41–72.

[112] Petr Hliněnỳ and Ondrej Moriš. "Scope-based route planning". In: *European Symposium on Algorithms*. Springer. 2011, pp. 445–456.

[113] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. "A brief survey of text mining." In: *Ldv Forum*. Vol. 20. 1. Citeseer. 2005, pp. 19–62.

[114] Eduardo R Hruschka et al. "Bayesian feature selection for clustering problems". In: *Journal of Information & Knowledge Management* 5.04 (2006), pp. 315–327.

[115] Weiming Hu et al. "Unsupervised active learning based on hierarchical graph-theoretic clustering". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.5 (2009), pp. 1147–1161.

[116] Hideki Isozaki and Hideto Kazawa. "Efficient support vector classifiers for named entity recognition". In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 2002, pp. 1–7.

[117] Mike James. *Classification algorithms*. Wiley-Interscience, 1985.

[118] Cun Ji et al. "Device data ingestion for industrial big data platforms with a case study". In: *Sensors* 16.3 (2016), p. 279.

[119] Jing Jiang and ChengXiang Zhai. "A systematic exploration of the feature space for relation extraction". In: *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 2007, pp. 113–120.

[120] Thorsten Joachims. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization.* Tech. rep. Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[121] Li Junlin and Fu Hongguang. "Molecular dynamics-like data clustering approach". In: *Pattern Recognition* 44.8 (2011), pp. 1721–1737.

[122] Tom Kalt and WB Croft. *A new probabilistic model of text classification and retrieval*. Tech. rep. Technical Report IR-78, University of Massachusetts Center for Intelligent Information Retrieval, 1996.

[123] Nanda Kambhatla. "Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations". In: *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics. 2004, p. 22.

[124] George Karypis, Eui-Hong Han, and Vipin Kumar. "Chameleon: Hierarchical clustering using dynamic modeling". In: *Computer* 32.8 (1999), pp. 68–75.

[125] O e a Kessler et al. "Functional description of the data fusion process". In: *Office of Naval Technology, Naval Air Development Center, Warminster, PA* 16 (1992).

[126] Tim Kieritz et al. "Distributed time-dependent contraction hierarchies". In: *International Symposium on Experimental Algorithms*. Springer. 2010, pp. 83–93.

[127] Kenji Kira and Larry A Rendell. "A practical approach to feature selection". In: *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 249–256.

[128] Dominik Kirchler, Leo Liberti, and Roberto Wolfler Calvo. "A label correcting algorithm for the shortest path problem on a multi-modal route network". In: *International Symposium on Experimental Algorithms*. Springer. 2012, pp. 236–247.

[129] Lawrence A Klein. "Sensor and data fusion concepts and applications". In: Society of Photo-Optical Instrumentation Engineers (SPIE). 1993.

[130] Jon Kleinberg. "The convergence of social and technological networks". In: *Communications of the ACM* 51.11 (2008), pp. 66–72.

[131] Daphne Koller and Mehran Sahami. *Hierarchically classifying documents using very few words*. Tech. rep. Stanford InfoLab, 1997.

[132] Balaji Krishnapuram et al. "A Bayesian approach to joint feature selection and classifier design". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1105–1111.

[133] Maciej Kurant, Athina Markopoulou, and Patrick Thiran. "On the bias of BFS". In: *arXiv preprint arXiv:1004.1729* (2010).

[134] Haewoon Kwak et al. "What is Twitter, a social network or a news media?" In: *Proceedings of the 19th international conference on World wide web*. AcM. 2010, pp. 591–600.

[135] Andrew Lamb et al. "The vertica analytic database: C-store 7 years later". In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 1790–1801.

[136] Leah S Larkey and W Bruce Croft. "Combining classifiers in text categorization". In: *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1996, pp. 289–297.

[137] Ulrich Lauther. "An Experimental Evaluation of Point-To-Point Shortest Path Calculation on Road Networks with Precalculated Edge-Flags." In: *The Shortest Path Problem*. 2006, pp. 19–40.

[138] David D Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *European conference on machine learning*. Springer. 1998, pp. 4–15.

[139] Haifeng Li, Tao Jiang, and Keshu Zhang. "Efficient and robust feature extraction by maximum margin criterion". In: *Advances in neural information processing systems*. 2004, pp. 97–104.

[140] Jianping Li et al. "Feature selection via least squares support feature machine". In: *International Journal of Information Technology & Decision Making* 6.04 (2007), pp. 671–686.

[141] Xin Li and Dan Roth. "Learning question classifiers". In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 2002, pp. 1–7.

[142] Zechao Li et al. "Clustering-guided sparse structural learning for unsupervised feature selection". In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (2014), pp. 2138–2150.

[143] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. "The global k-means clustering algorithm". In: *Pattern recognition* 36.2 (2003), pp. 451–461.

[144] Richard J Lipton and Robert Endre Tarjan. "A separator theorem for planar graphs". In: *SIAM Journal on Applied Mathematics* 36.2 (1979), pp. 177–189.

[145] Rong Liu, Robert Rallo, and Yoram Cohen. "Unsupervised feature selection using incremental least squares". In: *International Journal of Information Technology & Decision Making* 10.06 (2011), pp. 967–987.

[146] J Llinas and DL Hall. "A challenge for the data fusion community II: Infrastructure imperatives". In: *Proc. 7th Natl. Symp. on Sensor Fusion*. Vol. 16. 1994.

[147] James Llinas and David L Hall. "An introduction to multi-sensor data fusion". In: *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*. Vol. 6. IEEE. 1998, pp. 537–540.

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

[149] Teng Long and Lian-Wen Jin. "A new simplified gravitational clustering method for multi-prototype learning based on minimum classification error training". In: *Advances in Machine Vision, Image Processing, and Pattern Analysis*. Springer, 2006, pp. 168–175.

[150] J Lopes et al. "Traffic and mobility data collection for real-time applications". In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2010, pp. 216–223.

[151] Yong Luo et al. "Vector-Valued Multi-View Semi-Supervsed Learning for Multi-Label Image Classification." In: *AAAI*. 2013, pp. 647–653.

[152] James Macaulay and Markus Kückelhaus. *Internet of Things in Logistics*. https://delivering-tomorrow.de/wp-content/uploads/2015/08/DHLTrendReport_Internet_of_things.pdf. 2015.

[153] James Macaulay and Markus Kückelhaus. *Internet of Things in Logistics*. http://storm.apache.org/releases/2.0.0-SNAPSHOT/Guaranteeing-message-processing.html. 2015.

[154] Roger MacNicol and Blaine French. "Sybase IQ multiplex-designed for analytics". In: *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment. 2004, pp. 1227–1230.

[155] Samir Madhavan. *Mastering Python for Data Science*. Packt Publishing Ltd, 2015.

[156] Pradipta Maji. "Fuzzy–rough supervised attribute clustering algorithm and classification of microarray data". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 41.1 (2011), pp. 222–233.

[157] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *others, Introduction to information retrieval, vol. 1*. 2008.

[158] Adam Mathes. *Folksonomies-cooperative classification and communication through shared metadata*. 2004.

[159] Jens Maue, Peter Sanders, and Domagoj Matijevic. "Goal directed shortest path queries using precomputed cluster distances". In: *International Workshop on Experimental and Efficient Algorithms*. Springer. 2006, pp. 316–327.

[160] Ujjwal Maulik and Sanghamitra Bandyopadhyay. "Genetic algorithm-based clustering technique". In: *Pattern recognition* 33.9 (2000), pp. 1455–1465.

[161] Andrew McCallum, Kamal Nigam, et al. "A comparison of event models for naive bayes text classification". In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Citeseer. 1998, pp. 41–48.

[162] Andrew McCallum et al. "Improving Text Classification by Shrinkage in a Hierarchy of Classes." In: *ICML*. Vol. 98. 1998, pp. 359–367.

[163] Andrew Kachites McCallum. "Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering". In: *http://www. cs. cmu. edu/mccallum/bow/* (1996).

[164] Andrew Kachites McCallum. "Mallet: A machine learning for language toolkit". In: (2002).

[165] Marina Meila and Jianbo Shi. "A random walks view of spectral segmentation". In: (2001).

[166] Robert Meusel, Peter Mika, and Roi Blanco. "Focused crawling for structured data". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM. 2014, pp. 1039–1048.

[167] Robert Meusel, Petar Petrovski, and Christian Bizer. "The webdatacommons microdata, rdfa and microformat dataset series". In: *International Semantic Web Conference*. Springer. 2014, pp. 277–292.

[168] Ulrich Meyer. "Single-source shortest-paths on arbitrary directed graphs in linear average-case time". In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2001, pp. 797–806.

[169] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[170] Alan Mislove et al. "Measurement and analysis of online social networks". In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM. 2007, pp. 29–42.

[171] Tom M Mitchell et al. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[172] Rolf H Möhring et al. "Partitioning graphs to speedup Dijkstra's algorithm". In: *Journal of Experimental Algorithmics (JEA)* 11 (2007), pp. 2–8.

[173] Edward F Moore. "The shortest path through a maze". In: *Proc. Int. Symp. Switching Theory, 1959*. 1959, pp. 285–292.

[174] D.J. 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/. [Online; accessed 30-March-2017]. 2015.

[175] Laurent Flindt Muller and Martin Zachariasen. "Fast and compact oracles for approximate distances in planar graphs". In: *European Symposium on Algorithms*. Springer. 2007, pp. 657–668.

[176] Andrew Y Ng, Michael I Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[177] Feiping Nie et al. "Efficient and robust feature selection via joint $\ell 2$, 1-norms minimization". In: *Advances in neural information processing systems*. 2010, pp. 1813–1821.

[178] FRANTIŠEK Němec. "Distinguished problems of Logistics". In: *First International Symposium on Business Administration, Challenges For Business Administrators in the new Milleninium*, pp. 1–3.

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

[180] Andrew Pavlo et al. "A comparison of approaches to large-scale data analysis". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM. 2009, pp. 165–178.

[181] Paxata. "Ventana research: easing the pain of data preparation". In: *Ventana Research* (2014).

[182] Chengbin Peng et al. "Collective human mobility pattern from taxi trips in urban area". In: *PloS one* 7.4 (2012), e34487.

[183] Hanchuan Peng, Fuhui Long, and Chris Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005), pp. 1226–1238.

[184] David Pisinger and Stefan Ropke. "A general heuristic for vehicle routing problems". In: *Computers & operations research* 34.8 (2007), pp. 2403–2435.

[185] Robert Power, Bella Robinson, and David Ratcliffe. "Finding fires with twitter". In: *Australasian language technology association workshop*. Vol. 80. 2013.

[186] Lin Qiao et al. "Gobblin: Unifying data ingestion for Hadoop". In: *Proceedings of the VLDB Endowment* 8.12 (2015), pp. 1764–1769.

[187] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[188] Laura Elena Raileanu and Kilian Stoffel. "Theoretical comparison between the gini index and information gain criteria". In: *Annals of Mathematics and Artificial Intelligence* 41.1 (2004), pp. 77–93.

[189] Payam Porkar Rezaeiye, Mojtaba Sedigh Fazli, et al. "Use HMM and KNN for classifying corneal data". In: *arXiv preprint arXiv:1401.7486* (2014).

[190] Mehran Sahami et al. "A Bayesian approach to filtering junk e-mail". In: *Learning for Text Categorization: Papers from the 1998 workshop*. Vol. 62. Madison, Wisconsin. 1998, pp. 98–105.

[191] Peter Sanders and Dominik Schultes. "Engineering highway hierarchies". In: *Journal of Experimental Algorithmics (JEA)* 17 (2012), pp. 1–6.

[192] Peter Sanders and Dominik Schultes. "Robust, Almost Constant Time Shortest-Path Queries in Road Networks." In: *The Shortest Path Problem*. 2006, pp. 193–218.

[193] Peter Sanders and Christian Schulz. "Distributed evolutionary graph partitioning". In: *2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2012, pp. 16–29.

[194] Jagan Sankaranarayanan, Houman Alborzi, and Hanan Samet. "Efficient query processing on spatial networks". In: *Proceedings of the 13th annual ACM international workshop on Geographic information systems*. ACM. 2005, pp. 200–209.

[195] Robert E Schapire and Yoram Singer. "BoosTexter: A boosting-based system for text categorization". In: *Machine learning* 39.2-3 (2000), pp. 135–168.

[196] Gerhard Schrimpf et al. "Record breaking optimization results using the ruin and recreate principle". In: *Journal of Computational Physics* 159.2 (2000), pp. 139–171.

[197] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*. Vol. 39. Cambridge University Press, 2008.

[198] Fabrizio Sebastiani. "Machine learning in automated text categorization". In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.

[199] Burr Settles. "Biomedical named entity recognition using conditional random fields and rich feature sets". In: *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*. Association for Computational Linguistics. 2004, pp. 104–107.

[200] Linda G Shapiro and GC Stockman. *Computer Vision: Theory and Applications. 2001*.

[201] Jianbo Shi and Jitendra Malik. "Normalized cuts and image segmentation". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[202] Kamal Nigam t Andrew McCallum St. "Learning to Classify Text from Labeled and Unlabeled Documents". In: (1998).

[203] Hugo Steinhaus. "Sur la division des corp materiels en parties". In: *Bull. Acad. Polon. Sci* 1.804 (1956), p. 801.

[204] Michael Stonebraker. "The case for shared nothing". In: *IEEE Database Eng. Bull.* 9.1 (1986), pp. 4–9.

[205] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. "The 8 requirements of real-time stream processing". In: *ACM Sigmod Record* 34.4 (2005), pp. 42–47.

[206] Martin N Szomszor, Iván Cantador, and Harith Alani. "Correlating user profiles from multiple folksonomies". In: *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. ACM. 2008, pp. 33–42.

[207] Mikkel Thorup. "Compact oracles for reachability and approximate distances in planar digraphs". In: *Journal of the ACM (JACM)* 51.6 (2004), pp. 993–1024.

[208] Mikkel Thorup. "Integer priority queues with decrease key in constant time and the single source shortest paths problem". In: *Journal of Computer and System Sciences* 69.3 (2004), pp. 330–353.

[209] HLV Trees. *Modulation Theory, Part III: Radar-Sonar Signal Processing and Gaussian Signals in Noise*. 2001.

[210] Boudewijn F Van Dongen et al. "The prom framework: A new era in process mining tool support." In: *ICATPN*. Vol. 3536. Springer. 2005, pp. 444–454.

[211] Jake VanderPlas. *Python data science handbook: essential tools for working with data*. " O'Reilly Media, Inc.", 2016.

[212] Vladimir Vapnik. *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[213] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[214] Dorothea Wagner and Thomas Willhalm. "Drawing Graphs to Speed Up Shortest-Path Computations." In: *ALENEX/ANALCO*. 2005, pp. 17–25.

[215] Dorothea Wagner, Thomas Willhalm, and Christos Zaroliagis. "Geometric containers for efficient shortest-path computation". In: *Journal of Experimental Algorithmics (JEA)* 10 (2005), pp. 1–3.

[216] E Waltz. "Data fusion for C3I: A tutorial". In: *Command, Control, Communications Intelligence (C3I) Handbook* (1986), pp. 217–226.

[217] Edward Waltz, James Llinas, et al. *Multisensor data fusion*. Vol. 685. Artech house Boston, 1990.

[218] Pin Wang et al. "Proportional hybrid mechanism for population based feature selection algorithm". In: *International Journal of Information Technology & Decision Making* 16.05 (2017), pp. 1309–1338.

[219] Xiaogang Wang, Weiliang Qiu, and Ruben H Zamar. "CLUES: A non-parametric clustering method based on local shrinking". In: *Computational Statistics & Data Analysis* 52.1 (2007), pp. 286–298.

[220] Xin Wang, Shuxu Zhao, and Liang Dong. "Research and application of traffic visualization based on vehicle GPS big data". In: *International Conference on Intelligent Transportation*. Springer. 2016, pp. 293–302.

[221] Franklin E White. *Data fusion lexicon*. Tech. rep. JOINT DIRECTORS OF LABS WASHINGTON DC, 1991.

[222] Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[223] Christo Wilson et al. "User interactions in social networks and their implications". In: *Proceedings of the 4th ACM European conference on Computer systems*. Acm. 2009, pp. 205–218.

[224] Lior Wolf and Amnon Shashua. "Feature selection for unsupervised and supervised inference: The emergence of sparsity in a weight-based approach". In: *Journal of Machine Learning Research* 6.Nov (2005), pp. 1855–1887.

[225] Ka-Chun Wong et al. "Evolutionary multimodal optimization using the principle of locality". In: *Information Sciences* 194 (2012), pp. 138–170.

[226] Ka-Chun Wong et al. "Herd clustering: A synergistic data clustering approach using collective intelligence". In: *Applied Soft Computing* 23 (2014), pp. 61–75.

[227] William E Wright. "Gravitational clustering". In: *Pattern recognition* 9.3 (1977), pp. 151–166.

[228] Kuo-Lung Wu and Miin-Shen Yang. "Alternative c-means clustering algorithms". In: *Pattern recognition* 35.10 (2002), pp. 2267–2278.

[229] Hui Xiong, Junjie Wu, and Jian Chen. "K-means clustering versus validation measures: a data-distribution perspective". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.2 (2009), pp. 318–331.

[230] Rui Xu and Donald Wunsch. "Survey of clustering algorithms". In: *IEEE Transactions on neural networks* 16.3 (2005), pp. 645–678.

[231] Zenglin Xu et al. "Discriminative semi-supervised feature selection via manifold regularization". In: *IEEE Transactions on Neural networks* 21.7 (2010), pp. 1033–1047.

[232] Yi Yang et al. "l2, 1-norm regularized discriminative feature selection for unsupervised learning". In: *IJCAI proceedings-international joint conference on artificial intelligence*. Vol. 22. 1. 2011, p. 1589.

[233] Yiming Yang and Xin Liu. "A re-examination of text categorization methods". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 42–49.

[234] Shaozhi Ye, Juan Lang, and Shyhtsun Felix Wu. "Crawling online social graphs". In: (2010).

[235] Matei Zaharia et al. "Discretized streams: Fault-tolerant streaming computation at scale". In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM. 2013, pp. 423–438.

[236] Jiang-She Zhang and Yiu-Wing Leung. "Robust clustering by pruning outliers". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 33.6 (2003), pp. 983–998.

[237] Shichao Zhang, Chengqi Zhang, and Qiang Yang. "Data preparation for data mining". In: *Applied artificial intelligence* 17.5-6 (2003), pp. 375–381.

[238] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: an efficient data clustering method for very large databases". In: *ACM Sigmod Record*. Vol. 25. 2. ACM. 1996, pp. 103–114.

[239] Zheng Zhao and Huan Liu. "Semi-supervised feature selection via spectral analysis". In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pp. 641–646.

[240] Zheng Zhao and Huan Liu. "Spectral feature selection for supervised and unsupervised learning". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 1151–1157.

[241] Lin Zhu, Fu-Lai Chung, and Shitong Wang. "Generalized fuzzy c-means clustering algorithm with improved fuzzy partitions". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.3 (2009), pp. 578–591.