RWTH Aachen University | 52056 Aachen | Germany

Chair for
Computer Science 5
Information Systems
RWTH Aachen
University

Semantic Profiling in Data Lake

Master thesis at RWTH Aachen University.
Submitted for the degree of Master of Science (M.Sc.) in Software Systems Engineering.

Submitted by:
Jasim Waheed Ansari
jasim.waheed@rwth-aachen.de
Matriculation number: 350820

Thesis advisors:
Dr. Oya Deniz Beyan, Dr. Michael Cochez, MSc. Naila Karim
Informatik 5, RWTH Aachen University

Supervisor:
Prof. Dr. Stefan Decker

Aachen, February 2018

Version February 13, 2018

Eidesstattliche Versicherung

Waheed Ansari, Jasim	350820
Name, Vorname	Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Semantic Profiling in Data Lake

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 13.02.2018	
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 13.02.2018	
Ort, Datum	Unterschrift

I dedicate this work to my parents and friends for their constant support and encouragement during my course of study. Thank you.

Acknowledgements

Foremost, I would like to express my gratitude to my supervisor Prof. Dr. Stefan Decker to make it possible for me to work in his department.
I am deeply grateful to Dr. Oya Deniz Beyan. Without her constant motivation, optimism and guidance in this work until the very end would not have been possible.
I also owe a great dept of gratitude to Dr. Michael Cochez, for helping me out with research insights and iterative revision of my work.
Last but not the least, I am very thankful to M.Sc. Naila Karim for sharing her knowledge and advice on Big Data and Data Lake's state-of-art.

Abstract

In the Big Data community, Data Lakes have become the de facto standard for storing data. Often, these data lakes are contained within the Hadoop ecosystem, where the actual storage happens on Hadoop Distributed File System(HDFS). The data is then stored in its raw (structured or unstructured) form and whenever an application needs the data, it interprets the raw data. This approach is a schema-on-read in which the interpretation of the data and potential consistency checks happen when the data is read by an application. The biggest challenge for data lake governance is to avoid that it turns in a so-called data swamp. Data swamp occurs due to various reasons. First, there is the data quality aspect, such as noisy or incorrect data. Second, exponential growth of data ingested and lack of schema enforcement. Lastly, the amount of used schemas tends to grow.
In this work we present a new metadata extension to data lake systems by semantic profiling, which attempts to recognize the meaning of the data which is ingested into the Data Lake. The developed tool does not only detect meaning at schema level, but also at the data instance level by employing domain vocabularies and ontologies. With our tool, ingested datasets can easily be mapped to common domain concepts with unique identifiers and the meaning of the data can be discovered by the system. The developed profiling tool will help to produce meaningful summaries of the ingested content and provides opportunities to link relevant data sets ingested using different data schemas. We evaluate our tool by using two cancer genome datasets. We use semantic profiling tool during data ingestion and observe how data sets are tagged and profiled. Our experiments show that Semantic Ingestion is a promising approach for enriching the data sets in a data lake.

Contents

List of Figures

List of Tables

1 Introduction

Data Lakes have become an integral part of the enterprise world in the past decade caused by the exponential growth of data and the increased interest in analysis of unstructured data originating from both batch and streaming sources. It has enabled a whole new system of insight and has shifted the focus to more forward looking reports and probabilistic assessments as opposed to accurate reports based on historical data. Unlike a data warehouses, data lakes support storage of structured, semi-structured, and unstructured (so-called raw) data, with schema-on-read (i.e., the data is only interpreted at the time it is read) [1].

Terizzano et al. [2] defined a data lake as a central data repository containing enormous amounts of raw data, ingested from multiple data sources, that authorized users can readily use for multiple independent analytic activities.

The same study identified the importance of data description as a one of the key challenges in emerging data lakes. Description of data goes beyond describing the structure of the ingested data, it also covers the semantic meaning of data using domain ontologies and vocabularies. Another study has defined Data Lake as a horizontally scalable data store containing structured, unstructured and semi-structured data with data transformation support using frameworks like Apache Hadoop[1] and Apache Spark[2][3]. Extensive metadata annotation using ontologies would make data more meaningful and it could play a vital role in the future data lakes. Other challenges for data lake systems include the lack of informative summaries about the data contents, identification of anomalous data, and generation of data statistics. Semantic description of the data will help to produce meaningful summaries of the ingested content and provides opportunities to link relevant data sets ingested using different data schemas [4].

[1]http://hadoop.apache.org/
[2]http://spark.apache.org/

1.1 Research Questions

The gaps and challenges discussed previously led us to the follow research questions:

- ***RQ1:*** *How can Semantic Web improve the re-usability and discoverability of data ingested in Data Lake?*
  - As Data Lake can host data from 4Vs which will be discussed in section 2.1.1, it is important to have a common language among them for improved data utilization. To enable this common language, Semantic Web can play an important role.
  - Through our literature survey, we identified that one of the most challenging problem of Data Lakes is metadata management. We explore how Semantic Web and vocabularies can be used to solve this problems.
- ***RQ2:*** *What can be a systematic and rigorous approach using semantic-based tools to examine data in Data Lake?*
  - In this work, we introduce a technique which profiles the examined data similar to the idea of data profiling. But our approach in contrast to former techniques will examine beyond the structure and syntax of data. We answered this question in the methodology chapter.
- ***RQ3:*** *How can the proposed technique be beneficial in real-world scenarios?*
  - To answer this question, we have implemented our methodology in a state-of-art Data Lake solution. And evaluated the methodology using health-care dataset. We evaluated the results using different scenarios.

1.2 Research Contribution

In this thesis, we develop and investigate tools for effective metadata management in Data Lakes. In particular, we created algorithms for semantic data profiling using vocabulary services. To improve the re-usability of our work, we integrated our system into Kylo, a data lake system. The proposed solution is not only an automated governed Data Lake, but also reduces the challenge of understanding the data originating from various domains. The main contribution of this thesis is the automatic annotation of data ingested into the data lake system using semantic identifiers. Metadata management is achieved by discovering and adding semantic meaning to the data by tagging both schema and data with domain ontologies and vocabularies.

With our tool, ingested data sets can easily be mapped to common domain concepts with unique identifiers and the meaning of the data can be discovered by the system. Extending semantic annotation from data schema level to data entry level brings the advantage of handling data sets with categorical concepts, named entities or code-able items, e.g., diagnosis and diseases by assigning class labels. The main contribution is divided into two parts.

- An algorithm for extending the metadata annotation in traditional data lakes to semantic ones. These annotations are created automatically.
- This algorithm is integrated into the Big Data ecosystem, in particular it is implemented on top of the Kylo, which is a Data Lake management system. We also used this set-up for our evaluation.

The software developed for this research is available in the github repository[3].

1.3 Thesis Structure

The structure of thesis is as follows:

- **Chapter 2** gives overview on fundamentals of Big Data, Data Lake and Semantic Web. It describes the developments done in respective individual areas. We then dig deeper into the related work that impacts our work overall.
- **Chapter 3** provides explanations on our research methodology
- **Chapter 4** describes our technical contribution in two steps - development of algorithm which is followed by integration into the data lake
- **Chapter 5** describes our experiment criteria, evaluations done on those. We then provide information on challenges and common problems faced.
- **Chapter 6** discusses conclusion and future work.

[3]`https://git.rwth-aachen.de/datalab/SM4DL`

2 Literature Review

It is necessary to provide literature background behind the three technical aspects that together will help us in making solid understanding for the related work.
The three aspects that we will discuss are

- Big Data - overview on big data, characteristics, parameters, tools and technologies, benefits (especially in health care) and challenges

- Data Lake - its concept, capabilities, architecture listing technological drivers behind its success, and challenges

- Semantic Web - discussion on basic components of Semantic Web

2.1 Big Data

As Gartner stated[5], "Information will be the 21st century oil", rightly so, a data-driven era has dawned upon us[6]. This data is generated from various sources such as emails, sensors, audio, images, click-streams, logs, electronic health records etc. at an unprecedented scale. With the burst of this data explosion a.k.a big data, industries, government agencies and scientific journals like *Nature* & *Science* have already geared themselves into ground-breaking study using big data[7][8][9]. One of the areas that has gained strong attraction among big data experts is into health-care. The availability and seamless integration of health-care big data originating from clinical data, claims, costs, R&D data is of high significance. We will investigate through our thesis work's evaluation study how we can unlock value using medical data.

Definition

Among various existing definitions and explanation of big data in literature and keeping in consideration to our thesis work, the following definition can be offered. "The data that is *too big*, *too fast*, *too hard*, or *too uncertain* for traditional tools to store or process." These data can be either structured, unstructured or semi-structured that makes it impossible to manage and process them effectively in a traditional way[10].

Benefits

Not only digital driven organization benefits from applications of big data methodologies, but also traditional businesses can significantly increase their advantages through it. We summed up an overall three benefits broadly from various studies:

- Improved cost reduction: One of the most important benefits is cost reduction[11]. For instance, Mckinsey estimated that big data projects can enable more than $300 billion in savings per year in U.S. health-care[12]. Clinical operations and R&D are the two major areas in health-care that impact significantly to potential savings.
- Better and faster Decision Making: The use of big data doesn't change the urgency of decision making but mitigates it. Large enterprises are seeking faster decision systems using big data and they are indeed finding them[13][14]
- Improved quality of products: Another interesting case of applying big data methodologies is improving the quality of product and customer satisfaction[15].

2.1.1 Characteristics

Our explained characteristics of Big Data are closest to the definition of Gartner's 3V [16]. These are:

- *Volume*: It is the measure of amount of data. These data generated from wide sources are now in TB to XB in comparison to previously existing data.
- *Velocity*: It is the rate at which data is generated, ranging from per-day to per-second.
- *Variety*: It is based on diversity of formats, sources and types of data in structured and unstructured forms[17].

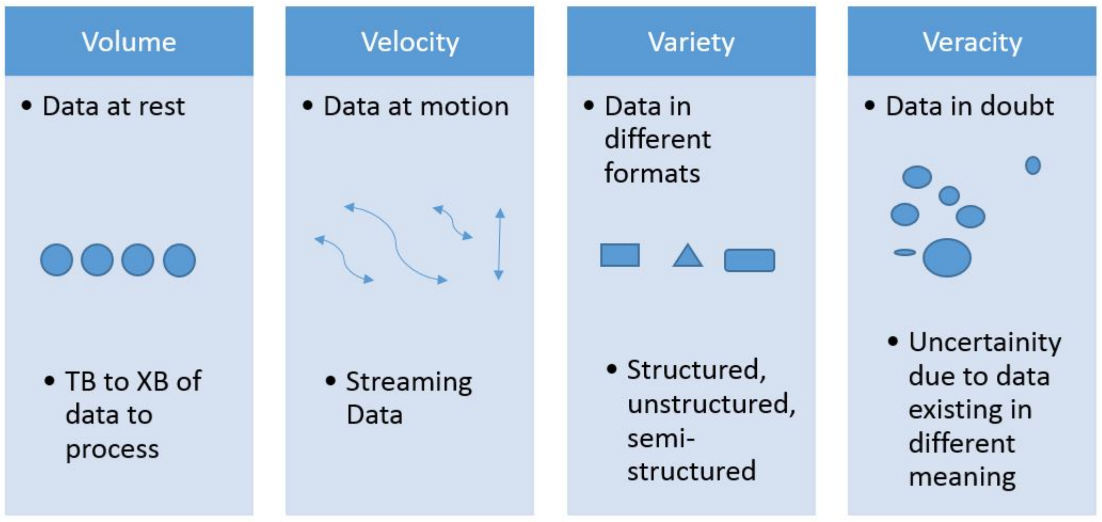

Volume
• Data at rest
• TB to XB of data to process
Velocity
• Data at motion
• Streaming Data
Variety
• Data in different formats
• Structured, unstructured, semi-structured
Veracity
• Data in doubt
• Uncertainity due to data existing in different meaning

Figure 2.1: Big Data - 4Vs

However, to add our thesis interest into this characteristics, we also had to add the fourth V aspect to it. This adds the data quality aspect to the characteristics. This V is also called as *Veracity*

- *Veracity:* It describes how complete and accuracy the data is. It relates to the vernacular “garbage-in, garbage-out” description for data quality issues in existence for a long time[18].

With the above provided characteristics, big data can be can be conceptualized as 4 "Vs" . It is also shown in fig. 2.1

2.1.2 Big Data Processing Types

Apart from the different formats and various kinds of data sources, there exists two ways of processing such big data[19]: batch-oriented processing(favors the Volume aspect of big data) and real-time data processing (favors the Velocity aspect of big data). On the other hand, as there has been increasing need arose to compensate for such high latency batch systems where fresh data is continually coming, a need to adopt hybrid processing style has also been practiced[20][21].

- *Batch-oriented processing* where data stored in disks are processed. These kind of processing are basically based on business events where data is processed in one-go. It is based on periodic style where the assumption is that there will exist a significant delay at each processing step. This processing style favors scenarios in data mining and machine learning where large data sets are meant to be trained to develop accurate models.

- *Real-time processing* where data is stored in memory and its result or logs eventually stores in disks. There exists little or almost-no delay between each processing steps. These are favored in use cases where low response is of utmost importance.
- *Hybrid processing* uses hybrid model also called Lambda Architecture[22] and consists of three architectural principles: robustness(the system is able to adapt to errors); immutability (data is never updated but it is appended with a time-stamp identifier); and re-computation (the result can be recomputed). It is implemented by four level architecture - batch-layer (constantly growing immutable distributed data file-system); speed-layer (deals with low latency data, data is temporary at this layer); serving layer (it loads the batch data as a view for querying); and combination layer (synchronization and result composition).

2.1.3 Big Data Tools & Technologies

Advancement in distributed computing is required in order to handle both data storage and processing. The following section provides an overview of technologies available to analyze big data.

Hadoop

Hadoop is an open source project powered by Apache Source Foundation. Although it consists of many sub-projects to support the Hadoop infrastructure, it mainly consists of[23]:

- *File System (Hadoop Distributed File System)*[24]: It is a fault-tolerant,highly scalable and highly configurable distributed data storage for a Hadoop computing cluster. The idea behind the data stored in HDFS is that its broken down into chunks of blocks either 64Mb or 128Mb and distributed across nodes in Hadoop cluster.
  This HDFS cluster works as a master-slave architecture where master stores the necessary metadata of the slaves that contains the data themselves.

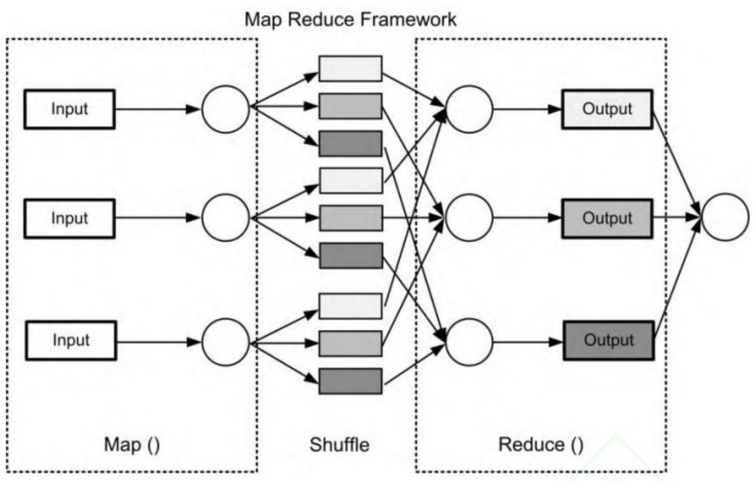


Figure 2.2: Map Reduce Framework[26]

- *Programming Model (Map Reduce)* [25]: It is a software framework that handles the scalable data processing in a reliable and fault-tolerant manner. This processing is two-folds:
  - Map: In this phase, the input is divided into sub-tasks, where each sub-task is then assigned to a mapper function. This mapper function converts into a key-value pair. This key-value pair then undergoes next phase called Shuffling. In this phase, each key-value pair is shuffled across blocks such that similar keys comes under same block.
  - Reduce: In this phase, the shuffled inputs are analyzed, then merged to final output and eventually persisted to HDFS.

  The framework is shown as fig. 2.2

Even though Hadoop eco-system is vast, we decided to list down major modules into respective functional areas that are important in our thesis reference. Table 2.1 summarizes the Hadoop modules and functions.

Constraints of Hadoop

Although Hadoop offers scalable and fault-tolerance features over traditional relational databases, it suffers from the following limitations[28]:

- Duplicate copies of data: To increase the availability of data, replication of data in Hadoop can help. But it also brings inefficiencies along. As Hadoop's

Table 2.1: Hadoop Modules and functions[27]

Function	Module	Description
Data Acquisition	Flume	Data collection from various source to centralized store
	Sqoop	Data import & export between RDBMS and Hadoop
Data Storage	HDFS	Distributed file system
	Hbase	Column-based file store
Data Processing	MapReduce	Distributed computation model
Query & Analysis	Hive	Sql-like query language over HDFS
Management	Zookeeper	Service configuration, consistency management
	Ambari	Cluster management

prominence came due to its offering on commodity hardwares[24]. But, multiple copies of data residing in Hadoop would also lead to performance degradation on cheap devices (for example, low disk I/O).

- Limited SQL support: There are various sub-projects like Hive that offers SQL-like support on Hadoop. This project help in extracting data from Hadoop as a query-able data-warehouse. But unfortunately, project like Hive cannot offer full SQL-like features such as nested sub-queries, transaction support etc[29, 30].
- Performance issues: A large amount of time in processing is spent on task initialization, scheduling, coordination and monitoring. Factors behind these are lack of a query optimizer, intensive disk I/Os.
- Challenging framework to manage: As already explained, Hadoop being a massive framework with different modules, there exists too many performance tuning parameters when deploying Hadoop cluster.

We kept these constraints in mind during our development phase. Hence, we have highlighted the mitigation plan during our architectural consideration in section 4.2.1.

2.1.4 Big Data : Challenges

Deploying big data process or framework to realize any use-case in practice is still challenging. We did literature study to understand these challenges and classified them into three parts[23, 15, 31].

- Technical Challenges: Unlike traditional data processing, Big Data processing life cycle is more complicated due to 4Vs.We describe the technical challenges at each of the step in Big Data processing.
  - Data Acquisition: It is the first step in which following challenges comes in before loading it into data store:
    1. data filtration and reduction: Much of the raw data that can be compressed and filtered down could also lead to potential information loss. For example, there could be a scenario where data marked for filtering might be useful for different kind of analysis.
    2. metadata generation: Once these data are filtered, metadata is generated to improve in understanding source data's origination, type, structure etc. But the problem is maintenance of these metadata that can continuously preserve data quality and accuracy.
  - Data Extraction: The data that is acquired may not be in right format or incompatible with Big Data solution. Therefore, transforming these datasets into useful artifacts gets challenging.
  - Data Integration: Data coming from different sources can be joined together using a common attribute, for instance ID. Performing this stage can be complicated due to difference in data or semantic structure.
  - Data Analysis: This step is dedicated to perform the actual analysis. These data analysis can be as simple as querying the data or as challenging as building a model to extract value.
  - Data Interpretation: Interpreting the results of the analysis can vary from simple tables to graphically visualizing the results using data visualization techniques. Since the same result can be approached from various viewpoints, understanding right assumptions and techniques are challenging for data interpretation.

  Figure 2.3 shows challenges at each big data processing step.
- Management Challenges:
  - Privacy & security issues: Privacy is another important issue in Big Data which is sensitive and is of conceptual, technical and legal importance[23]. For example, in healthcare, electronic health records have laws governing what can be shared and what can be revealed.
    Big data contains sensitive information that needs to be secured against

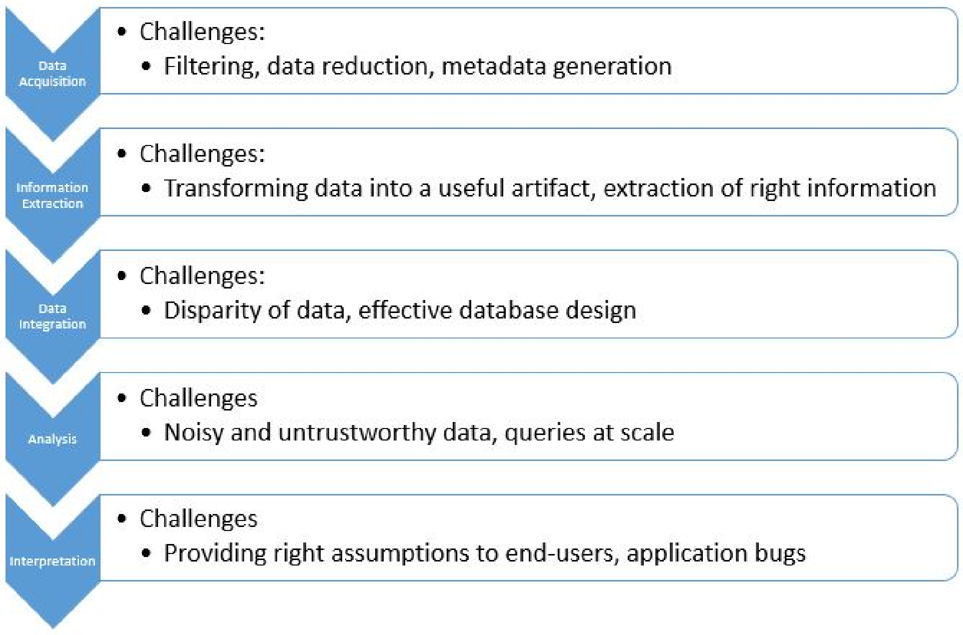

Data Acquisition
• Challenges:
• Filtering, data reduction, metadata generation
Information Extraction
• Challenges:
• Transforming data into a useful artifact, extraction of right information
Data Integration
• Challenges:
• Disparity of data, effective database design
Analysis
• Challenges
• Noisy and untrustworthy data, queries at scale
Interpretation
• Challenges
• Providing right assumptions to end-users, application bugs

Figure 2.3: Challenges: Big Data Processing Steps

unauthorized usage from third parties or illicit users. For instance, data being distributed in nature becomes more vulnerable to attacks over networks.

- Data Governance: As the demand of data continues to grow, the governance infrastructure is of utmost need to maintain the quality of data stored.

• Cost challenges: Big Data processing affects both the cost of resources and performance.
For example, due to the rigorous demands of Big Data on network, storage and server, data transmission incurs significant overhead[23]. There should be consideration given on the best practice on usage of these data.

2.2 Data Lake Concept

Based on challenges listed previously, *data lake* adoption has become popular among many organizations[32, 33]. In our literature survey to find the definition of *data lake*, it was first seen years ago in James Dixon's blog[34]. It is defined as "a method of storing data within a system or repository, in its natural format, that facilitates the collocation of data in various schemata and structural forms, usually object blobs or files"[35]. Unlike traditional data warehouse's rigid schema design or data model, data lake is lenient on standardization and defers modeling, this results in operational insight and data discovery without any bounds[36].

2.2.1 Data Lake: Features

Following are the capabilities offered by data lakes[33]:

- It supports scalable data storage in its native form at a low cost. It gets possible due to no transformation action applied on the data until and unless needed.
- It promises to store wide variety of data types ranging from blobs to traditional DBMS; from multi-structured to multi-media data.
- It leverages the feature of using data when needed. That means, costly data modeling and integration efforts are reduced due this approach. This approach is known as schema-on-read.
- It performs analytics based on single subject area. Since the value is initially hazy, therefore users have to develop particular analytics to use the data.
- Governance policies are setup to identify, reuse and dispose data
- It features provenance of data which track sources, its origin, who changed it, what is the version of change etc.

2.2.2 Architecture

There is no particular base architecture of a data lake. Many enterprises have demonstrated their own proposed architecture. Therefore, we look into the studied architectures and try to find common ground between them and also point out gaps that could help us in making our research foundation stable.
PWC has demonstrated an architecture as a Hadoop data lake as shown in fig. 2.4[36]. However, the architecture tends to be a marketing label for a product which supports Hadoop. Unfortunately, we do not see the full capabilities of data lake through it.
In another work, data lake's architecture is presented as a hybrid system. As shown in fig. 2.5, although data lake performs wider range of capabilities, for example Data Governance, they are still decoupled from data lake[33].

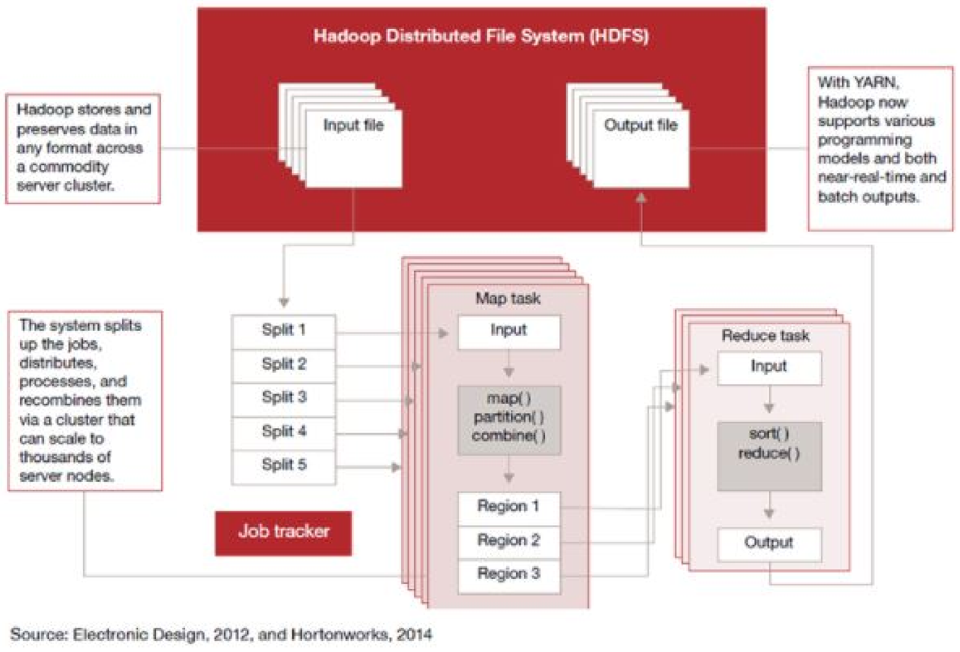

Hadoop Distributed File System (HDFS)
Hadoop stores and preserves data in any format across a commodity server cluster.
Input file
Output file
With YARN, Hadoop now supports various programming models and both near-real-time and batch outputs.
The system splits up the jobs, distributes, processes, and recombines them via a cluster that can scale to thousands of server nodes.
Split 1
Split 2
Split 3
Split 4
Split 5
Job tracker
Map task
Input
map()
partition()
combine()
Region 1
Region 2
Region 3
Reduce task
Input
sort()
reduce()
Output
Source: Electronic Design, 2012, and Hortonworks, 2014

Figure 2.4: PWC's basic Hadoop architecture for scalable data lake infrastructure[36]

Google has designed and implemented architecture called GOODS that helps Google's engineer organize and manage datasets in data lake[37]. Their data lake works in background in a non-intrusive manner, where it continues to collect metadata after data is created, used or updated by different engineers. The architecture is shown as in fig. 2.6
It is worth observing that although the overall goal remains the same i.e. an effective centralized data management system. But their functions and features vary from organizations to organizations.

2.2.3 Challenges in implementing Data Lake

From the technical perspective, deployment and provision tools are available to spin up a Hadoop cluster, but the problem is building an all-round data lake capabilities for Hadoop cluster. Let us list the challenges as follows[33, 32]:

- Ingestion and data integration problems: Typically in Data Lake, schema is applied after the data is stored, also called as *late-binding*[33]. This late-binding creates many problems such as:

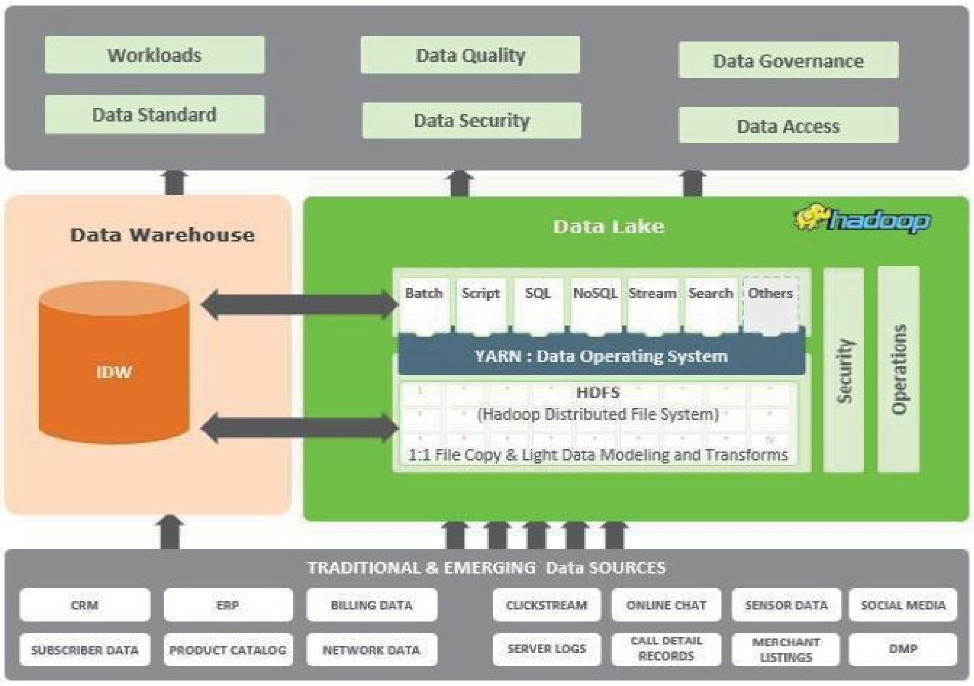
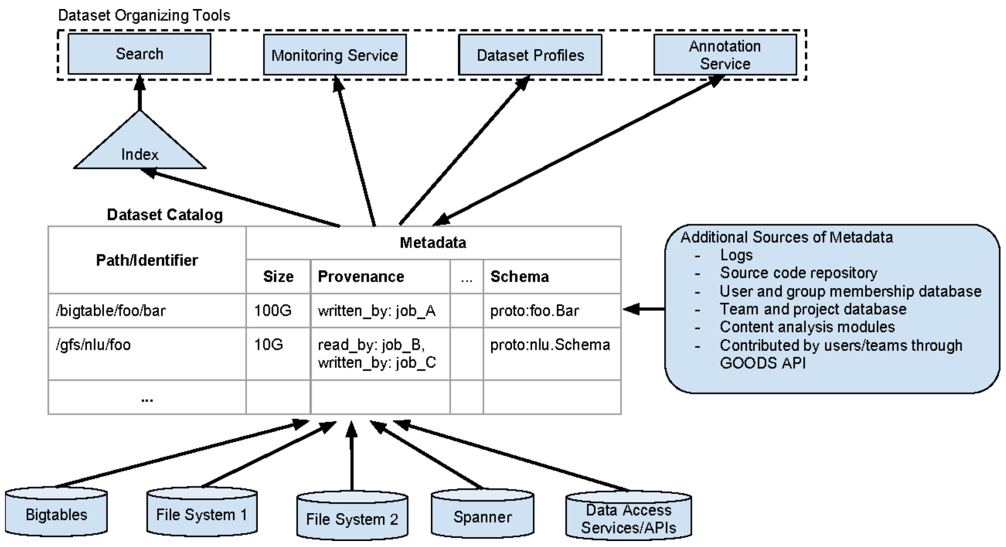


Figure 2.5: Architecture of a hybrid data lake[33]


Dataset Organizing Tools
Search
Monitoring Service
Dataset Profiles
Annotation Service
Index
Dataset Catalog
Additional Sources of Metadata
- Logs
- Source code repository
- User and group membership database
- Team and project database
- Content analysis modules
- Contributed by users/teams through GOODS API
Bigtables
File System 1
File System 2
Spanner
Data Access Services/APIs

Path/Identifier	Metadata			
	Size	Provenance	...	Schema
/bigtable/foo/bar	100G	written_by: job_A		proto:foo.Bar
/gfs/nlu/foo	10G	read_by: job_B, written_by: job_C		proto:nlu.Schema
...				

Figure 2.6: Architecture of GOODs data lake[37]

  - Slow ingestion of data: It is because data transformations are to be done at the end of ingestion process.
  - Data integration problems: Due to lack of pre-defined data structure in data lakes
  - Limitation on reusing the data: Due to lack of modeling on data.

- Data Governance structure needs to have a balance: As Data Governance dictates access policy, privacy and security, there should be a balance between the level of flexibility and cost-effectiveness for Data governance model. For instance, if a rigid structure is followed on an enormous data, it may lead to more cost and time efforts which may unfortunately exceed the value of data.
- Data Swamp problems: If metadata management and governance is not applied with robust controls then data lake can become *data swamps*. These get overwhelming and risky to use since nobody is sure about where is data coming from, reliability of data and how it can be protected[32].

2.2.4 Metadata Management in Data Lakes

In light of problems as described, different organizations came up with solving these issues. EMC [38] discussed about categorization of data into three class such that each has different governance requirements:

- Governed data: Key business data has been understood for ownership, definition, quality and business rules
- Lightly governed data: Key business data has been understood with respect to its definition or ownership but not with respect to data quality
- Ungoverned data: Key business data is only understood from the point of definition and location.

Based on the governance level, IBM proposed an enhanced data lake solution called *data reservoir* which offers built-in management, affordability and governance[39]. There has also been considerable works done among different research communities into metadata management efforts in Data Lakes. In questions regarding improved data traceability of data in data lakes, the authors have demonstrated a reference architecture for provenance in data lakes[40]. In another work, Beheshti et al.[41] presented CoreDB, a data lake service that offers a single REST API to

query, organize and index the growing volume of data and metadata. Alrehamy et al.[42] presented work on Personal Data Lake, a central data lake for analyzing and querying personal data. This work primarily focuses on individual data privacy issues and how it can be tackled in data lakes.
We discussed so far the work done in metadata management in different aspects like governance, provenance, and privacy. But problems like integration and reusability of data in data lake is still not discussed. Thus, it is important that the background of Semantic Web must be highlighted in the next section. This would finally facilitate potential research direction of our work.

2.3 Semantic Web

The principal goal in Semantic Web is to automate the machine processing of web documents to make their meanings explicit[43]. The language for knowledge representation of these web documents are called *Resource Description Framework* (RDF). RDF is a *data modeling language* which represents *resources* and their *relationships* in form of directed labeled graph. *Resources* are the things that is described by RDF expression. *Resources* could be class or concept from a domain of discourse. *Predicates* are another special class of resources that define relationships between resources. Therefore, overall a resource is a class or concept or predicate which is uniquely identified through an identifier called *Universal Resource Identifier*(URI).

2.3.1 Semantic Web Data Formats

There exists different formats for representing the directed labeled graph. These are:

- RDF/XML[1]
- N-Triples[2]
- Turtle [3]
- N-Quads [4]

[1] `https://www.w3.org/TR/rdf-syntax-grammar/`
[2] `https://www.w3.org/TR/n-triples/`
[3] `https://www.w3.org/TR/turtle/`
[4] `https://www.w3.org/TR/n-quads/`

RDF/XML represents the data in traditional XML style, whereas, N-Triples and Turtle later introduced as an improved human readable data format. N-Quads offer additional annotations to be given to RDF graphs such as provenance to indicate the context of data.

2.3.2 Semantic Web Query Languages

To retrieve the RDF data, Semantic Web query languages have been developed. Semantic Query languages are more complex than SQL because of RDF's basic model is graph[44]. Unlike similar data structure style of SQL, graphs can have resources of different types, each type can have different properties and each property can be another resource. Among many implementation of query languages that has been designed, SparQL[5] is the W3C recommendation for RDF.

2.3.3 Ontologies

Researchers from Semantic Web have introduced *Domain modeling languages* which allow creation of semantics i.e. machine-readable information to the contents. These languages help in creation of ontologies. An ontology is an explicit specification of conceptualization[45].
Ontology consists of class and concepts that defines relationships among them as hierarchies or predicates. For example, shown fig. 2.7 an example ontology from health-care domain that models doctor, nurses, patients etc. As it might state that both Doctor and Nurse are subclasses of the Person class and that they are disjoint, i.e., no single person can be a doctor and nurse at the same time. Moreover, each class has describe attributes like title of doctor, name of the nurse etc. Thus, these ontologies help in resolving the ambiguities of the natural language.

2.4 Related Work

Data profiling, big data and the semantic aspects are hardly discussed together in previous literature. A holistic approach of semantic profiling in data lakes to support data stewards, analysts and data scientists is still missing. We identified several

[5]`https://www.w3.org/TR/rdf-sparql-query/`

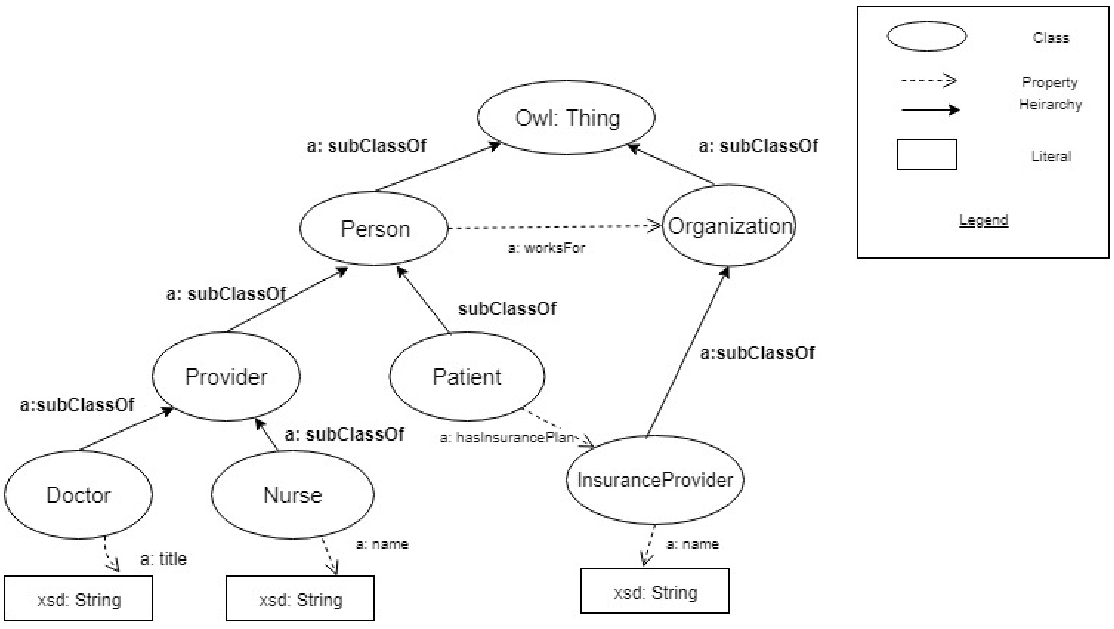


Figure 2.7: Example: Ontology for health-care

related studies which impacted our work. We categorized our literature survey into three parts - Data Profiling; Semantic Annotation; and Semantic Data Lakes

2.4.1 Data Profiling

One of the crucial prerequisite of using data in any application is to understand its data and metadata. Data Profiling is one such prerequisite[46]. "*Data profiling is the process of examining the data available from an existing information source (e.g. a database or a file) and collecting statistics or informative summaries about that data.*"[6] In this section, we discuss some of the work in data profiling in context to semantic web that has been done.

Naumann [47] has demonstrated a renewed focus on traditional data profiling through interactive data profiling, incremental data profiling, and profiling of heterogeneous data. In the latter method, the author has discussed about the scenarios in which profiling of multiple, heterogeneous datasets is needed. The author has explained this technique with three types of scenarios:

- Data profiling techniques uncovering degree of heterogeneity: These hetero-

[6]https://en.wikipedia.org/wiki/Data_profiling

geneity can be divided into 1) syntactical heterogeneity e.g. discovering inconsistent formatting 2) structural heterogeneity e.g. discovery of schema details like type, keys etc. 3) semantical heterogeneity e.g. discovering underlying mismatching meaning of values

- Data profiling for integration: It can require both structural and semantical profiling. This is because apart from schema overlap, the meaning of the values in schema can itself provide further details. And this causes overhead in integration. These data overlap can be identified using methods like ontology modeling, data linkage, and semantic annotation[48, 49, 50].
- Data Profiling unknown data from large volume of sources requires identification of domain of discourse. This is known as topical profiling[47].

Even though the author has provided reflection on our focus of work in the latter method, details on methodologies behind those techniques are still left undiscussed. Also, the data on which the profiling tasks were discussed are not fully described in terms of 4Vs of Big Data.
David McComb in his white-paper has explained Semantic Profiling from the viewpoint of extending profiling feature by using semantic-based tools and ontologies[51]. He has explained the difference between this technique from formers. Firstly, traditional profiling uses different metrics on the data to uncover as many anomalies as one can but there lies stopping point. Whereas, using semantic profiling the user commits to a domain of study and goes as deep as possible. Secondly, results of one dataset can be reused for other datasets as the ontology is the central point for reference. Finally, testable hypotheses can be setup to trigger events when semantic drifts come in new data for testing. It is worth noting that the author has described his approach from the single ontology as a starting point. We believe this approach will create tremendous downfall in performance and also inaccurately capturing semantic drifts if the data source itself a big data. As it is also known that the development of an ontology from scratch is a resource intensive process[44]

Böhm et al. [52] proposed a tool called ProLOD for diving deep into the all-round semantic profiling efforts ranging from clustering correlated datasets together into different domains, inferring schema discovery in each of the clusters, to data level profiling statistics. Although the tool identifies semantic heterogeneity at both intra- and inter-data source level, the focus of that work is limited to Linked Open Data which means data available in RDF. ProLOD also does not explicitly solve the problems of scalability, where the RDF data size would grow to millions or even billions of triples.

Ayman et al. [53] have proposed a framework for profiling similarity between datasets using alignment approach in openML Data Lake. The algorithm that the authors have used is based on an ontology alignment approach. Each ingested dataset (with metadata consisting of different profiled attributes) is converted into an RDF file, these RDF files are then sent to RDF-based ontology engine to detect similarities. The authors have used the PARIS ontology alignment tool because of its effectiveness on large scale ontologies. The gap in these algorithms is that the relationships are largely focused on strategies behind structural matching. There is not enough thought given to also consider synonymous vocabularies of the structured data while matching.

Salem and her co-authors [54] have proposed a profiling method which largely influences our work on data lakes. The main point of interest is the detection of issues in a dataset and proposing an understandable data schema by applying a semantic profiling algorithm. While studying their work, we observed that the core component that does the knowledge management is the data dictionary. They used the support of lexical databases to enhance their data dictionary. This needs the prime focus. It is because the lexical databases might miss many important annotation if the dataset belongs to scientific domain.
We extend their algorithm with entire dataset values and at the same time suit our focus in understanding scientific data sets. Furthermore, we exploit the algorithm's generated artifacts to identify similarities between various ingested data sets.

2.4.2 Semantic Annotation

It is the technique of providing additional information to various concepts (for instance, people, things etc.) in a given text or any other content[50]. Unlike traditional annotations, semantic annotations are usually used by machines to decipher the meaning.
There exists surveys that have been done on Semantic Annotation Platforms(SAPs). The surveys have examined architecture, methods and performances of these tools. F-measures was used to determine annotation effectiveness of these SAP tools. That is, how accurate/inaccurately annotations are generated by SAP. The results showed Armadillo[55], KIM[56] and MUSE[57] as most effective. 1) Armadillo[55] can automatically extract entities from web. It identifies entity based on regular patterns in the web. 2) KIM[56] on the other hand uses repository(SESAME[58]) to

store ontology and knowledge base to process semantic annotation on the document it examines. 3) MUSE is another SAP that performs named entity recognition from various data sources. MUSE platforms place the responsibility on the user for constructing an initial ontology.
These tools work well when the regularities of texts are guaranteed. However, in scientific datasets, these regularities may or may not exist. This is because scientific data are generated through experiments. The metadata can contain axillary information which are not standarized. Thus, the metadata may not be found in generic ontologies like Wordnet or KIMO. Hence, employing these techniques in scientific datasets would not be useful.

2.4.3 Semantic Data Lakes

In this part, we studied work where applications of Semantic Web has been applied in Data Lake.
light of Data Swamp problems, Hai et al.[59] have proposed a metadata managed data lake solution called Constance. It focuses on discovering, extracting, summarizing structural metadata and annotating data & metadata with semantic information. Constance's component that does the semantic metadata management is *Semantic Metadata Matching(SMM)*. It features ontology modeling, data linkage, and semantic annotation. However, much like discussed in other related works, the aspect of data integration is based on structural matching that account to matching on the basis of syntax but not on data's semantics context.
Semantic Data Lakes have also been gaining attrition among clinicians and healthcare exports as these can help in critically linking biological data[60, 61, 62]. However, contextualizing data as a graph model would prove to be a challenge in cases where collection of datasets may be needed in their original format for later use.

3 Research Methodology

In this chapter, we elaborate the methodology to conduct our research. The chapter is broken down as follows:
section 3.1 introduces the motivational case study that helped us to conduct the research.
Although we highlighted Semantic Profiling concept previously in section 2.4.1, it deserves a fresh focus in reference to our work. We discuss the Semantic Profiling concept in section 3.2
In section 3.3, we discuss the proposed framework that works in harmony with data lake and we also briefly discuss about the artifacts that is generated in this work.
Finally, in section 3.4, we formally describe the artifacts with definitions and its algorithm.

3.1 Motivational Case Study

It is a difficult task to analyze scientific datasets for extracting insights. One reason is that the supplied metadata might not be enough to explain every term. The domain of study that we choose is cancer data. Cancer is considered to be one of the most complex disease known to mankind. There exists over 200 cancer types with each having different molecular profiles. There is a huge demand among researchers to help them understand these complex datasets to extract insights which would lead to better diagnosis or even precision medicines for the patients. There has been a large scale technological development in one of the cancer related projects namely The Cancer Genome Atlas (TCGA). TCGA has a wealth of raw data which is in Terabytes [1]. We focus on two of its analytical end points (i.e. Firebrowse[2] and cBioPortal[3]) which are cleansed and normalized datasets of TCGA. Even though these tools have helped clinicians in understanding the results, the

[1]https://portal.gdc.cancer.gov/repository
[2]http://firebrowse.org/
[3]http://www.cbioportal.org/

Table 3.1: Attributes of TCGA data sources cBioPortal and Firebrowse

cBioPortal	Firebrowse
tcga_participant_barcode	*tcga_participant_barcode*
cohort	*cohort*
locus_id	geneId
cytoband	
date	
gene	
all_copy_number	
	gene_expression_log2
	z-score
	sample_type
	Protocol

results are still hard to understand for a wider range of audiences [63]. The properties used (i.e., a simplification of the schema) in these two end points are listed in table 3.1. Now, if we request a ontology library service that contain biomedical terms, we can discover that *locus_id* can also be called as *geneId*. Hence, we could conclude that in addition to tcga_participant_barcode & cohort, which has the same property name in both, also the locus_id in cBioPortal is likely the same as the geneId in Firebrowse. It is critical to have an ontology library with a rich detailed term's information such as its synonymous names, ontology to which it belongs to, etc. Also, it is important to have a programmable API that we can easily integrate in order perform these annotations automatically. It is to be noted that these annotations will not only show interesting insights at schema level but also at the record level. We can observe that schema matching is not only done on attribute's syntax level but also based on attribute's semantics background. As clearly shown from the example of health-care, schema matching requires domain-expertise and these can lead to time-consuming, tedious and error-prone results. Based on this motivation, we have developed an algorithm along with its integration into state-of-art data lake solution. This algorithm attaches the semantic annotation automatically and can help users find correspondence between elements of different schemas.
In the next chapter, we discuss about the development and integration of this algorithm into data lake solution. Ultimately, we will evaluate our solution with respect to three user scenarios and discuss it in Evaluation chapter.

3.2 Semantic Profiling

Semantic profiling is a technique that uses ontologies to gain deeper understanding of the information being stored and manipulated in an existing system [51]. An ontology based approach can be applied to understand the meaning of data and therefore to identify semantic association between data items [64]. Ontologies and domain specific vocabularies describe the semantic meaning of data in a machine readable format. Moreover, they also enable the creation of relations with external knowledge as well as linking internal data sets. There are three core criteria of semantic data profiles, namely 1) the versatility of the features that describe the dataset, 2) domain independence, i.e., whether the generation procedure can be applied to datasets from any domain, and 3) whether the approach is automated [65]. Keeping these criteria in mind, one should choose an ontology library that does the job for the user. In our implementation phase as discussed in chapter 4, we have used biomedical domain as an example. And we explained our rationale behind choosing the most relevant ontology library from this domain.
Our tool interacts with the ontology library to automatically get recommendations for each data element ingested in the data lake. This recommendation is then processed by our algorithm to extend the metadata for each data item, without modifying the data itself. In this manner our approach is automatically adding semantic metadata wrappers for each term. With this idea, we take full opportunity of semantic profiling to make data sets in data lakes discoverable and interpretable.

3.3 Proposed Framework

The framework behind semantic profiling is shown in fig. 3.1. Once the data source is specified in Data Lake, it undergoes set of processes in series. That includes - data extraction, meta-data collection, and data profiling with respect to standardization & validation feature as set by user. At the end of this flow, the ingested data results into valid and invalid profiled tables based on data validation and standardization rules set.
Then comes the semantic profiling step. Our algorithm considers the valid table as input for semantic profiling. Since our data lake can host high volume data, therefore intuition is that using the cluster processing resources appropriately is important. Hence, the data that follows correct structure in terms of syntax and standards should be taken into consideration for semantic profiling. For example, it is a

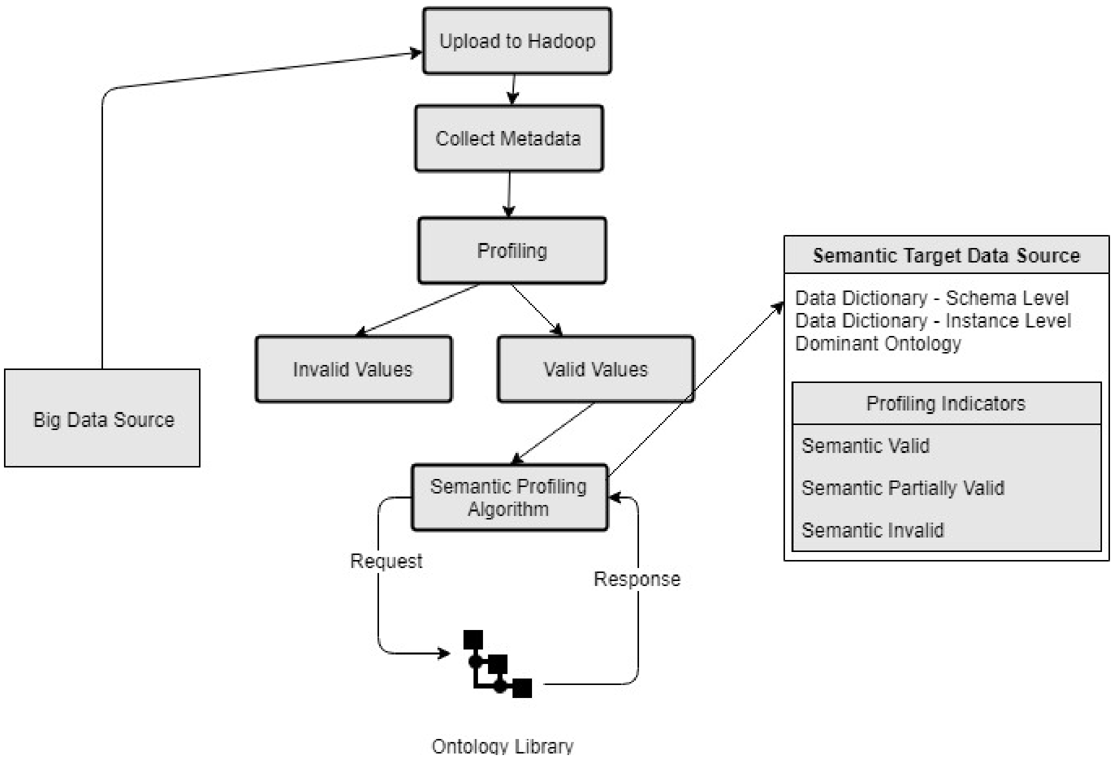


Figure 3.1: Our framework for Semantic Profiling

waste of resources and processing efforts if null values have to undergo annotations and consequently get an empty response from ontology library. Finally, our algorithm uses the top ranked annotation and builds a data dictionary into two parts - schema level and instance level. As the names suggest, schema and instance data dictionary stores the semantic information of schema and data level information respectively. Based on all the identified semantic tags in *Semantic Data Dictionary*, our system develops three profiling *Semantic Indicators*. These are produced via a helper metric called *Dominant Ontology*. The intuition behind these metrics are as follows. First, even though the meaning behind the terms are annotated, it is not a guarantee that there is a full text found by the recommender engine. In cases, partial matches are found whereas sometimes no match as well. Secondly, having an overall domain understanding on each column of the table helps in identifying its semantic structure. To ensure that, we require support of a helper metric dominant ontology for each column. Finally, each ontology stores data's information as preferred name and alternative name. The preferred name of the data is a unique and unambiguous label. Whereas, alternative terms exists as common translations for the data in that domain. We use the preferred name rather than an alternative name to identify meaningful valid values.

3.4 Semantic Profiling Process

In the semantic profiling process, properties of Semantic Data Dictionary and helper property *Dominant Ontology* plays the central role in defining the Semantic Indicators. Therefore, we divide the explanation of each of the property in separate subsections. In the end, we discuss the algorithm.

3.4.1 Semantic Data Dictionary

The Semantic Data Dictionary is a way to represent implicit information about ingested data using ontology library. Semantic Data Dictionary properties contains semantic information of ingested data following specifications as shown in table 3.2. Using these properties, we create data dictionary at two levels - schema and instance level.
Let us take the example of cBioPortal's sample data previously discussed in section 3.1. The sample data source snapshot is given in table 3.3. During fetching

Table 3.2: Semantic Data Dictionary Properties

Column	Description
Label	Name of the column/value of the ingested table
Parsed Label	Recommended result found by the ontology library for the label
Type	If the Parsed Label type is preferred or alternative
Description	Detailed description of the Parsed Label
Synonyms	Collection of synonyms found for the Parsed Label
Ontology Name	Name of the ontology from which this Parsed Label belongs to
Ontology Uri	Uri of the ontology - contains detailed description about the ontology itself

Table 3.3: Data Source: Example

tcga_participant_barcode	cytoband	locus_id	gene	all_copy_number	cohort
TCGA-OR-A5KP	21q21.2	-6309	7SK..	0.537	ACC
TCGA-OR-A5KP	19q13.43	1	A1BG	0.537	ACC
TCGA-GF-A4EO	22q13.2	53947	A4GALT	-0.106	SKCM
TCGA-DK-A6B6	12q13.13	8086	AAAS	-0.015	BLCA

of the data source from the API library, the API documentation may supply metadata information about the source. For example, as shown in table 3.3, the detail metadata information about *tcga_participant_barcode* can be gathered using official documentation of TCGA[4]. However, for scientific data like these, it is important to grasp the idea behind the entire structure and its content details. Hence, Semantic Data Dictionary is of dire need at two levels - schema and instance level.
Using existing ontology library service and keeping our specification in mind, its semantic annotated data dictionary at schema level can be shown as example in table 3.4. As it may be seen, these annotation help us in identifying data source's information with most relevant semantic details that belong to a ontology.
Similarly, we can also generate the instance level dictionary as shown in table 3.6.

3.4.2 Dominant Ontology

As ontology library service facilitates in building the semantic data dictionary, it is still quite a challenge to understand a clear domain knowledge to which each column largely belongs to. This is because sometimes due to semantic ambiguity, ontology library services may not reliably give the trusted recommended ontology.

[4]https://wiki.nci.nih.gov/display/TCGA/TCGA+barcode

Table 3.4: Example: Semantic Data Dictionary - Schema

Label	Parsed Label	Type	Description	Synonyms	Ontology Name	Ontology Uri
tcga_ participant _barcode	TCGA	Synonym	Understand the molecular basis of,cancer..	The Cancer Genome Atlas; TCGA	National Cancer Institute TheSaurus	#NCIT
cytoband	Cytoband	Synonym	A cytologically distinguishable feature of a chromosome..	chromosome band; cytoband;cytological band	Sequence Ontology	#SO
locus_id	Locus Id	Preferred	A unique name or,other identifier of a genetic locus..	Locus name; Locus identifier	EDAM	#EDAM
gene	Gene	Preferred	A functional unit,of heredity which occupies a specific position on a particular chromosome..	gene; Gene; Genes; GENE	National Cancer Institute TheSaurus	#NCIT
all _copy _number	Copy Number	Preferred	The number of,molecules of a particular type on or in a cell ..	Copy Number	National Cancer Institute TheSaurus	#NCIT
cohort	Cohort	Preferred	A group of individuals ..	Cohort; cohort	National Cancer Institute TheSaurus	#NCIT

Table 3.5: Example: Semantic Data Dictionary - Instance

Label	**Parsed Label**	**Type**	**Description**	**Synonyms**	**Ontology Name**	**Ontology Uri**
21q21.2	21q21.2	Preferred	A chromosome band present on 21 q		National Cancer Institute TheSaurus	#NCIT
19q13.43	19q13	Preferred	A chromosome band present on 19 q		National Cancer Institute TheSaurus	#NCIT
22q13.2	22q13.2	Preferred	A chromosome band present on 22q		National Cancer Institute TheSaurus	#NCIT
12q13.13	12q13.13	Preferred	A chromosome band present on 12q		National Cancer Institute TheSaurus	#NCIT
A1BG	A1BG	Preferred	Protein coding gene..		Gene Expression Ontology	#GEXO
A4GALT	A4GALT	Preferred	CD77 synthase blood group..		Ontology of Genes and Genome	#OGG
AAAS	AAAS	Preferred	Protein coding gene..		Gene Expression Ontology	#GEXO

Table 3.6: Example: Semantic Data Dictionary - Instance

Label	Parsed Label	Type	Description	Synonyms	Ontology Name	Ontology Uri
21q21.2	21q21	Preferred	A chromosome band present on 21 q		National Cancer Institute TheSaurus	#NCIT
19q13.43	19q13	Preferred	A chromosome band present on 19 q		National Cancer Institute TheSaurus	#NCIT
22q13.2	22q13.2	Preferred	A chromosome band present on 22q		National Cancer Institute TheSaurus	#NCIT
12q13.13	12q13.13	Preferred	A chromosome band present on 12q		National Cancer Institute TheSaurus	#NCIT
A1BG	A1BG	Preferred	Protein coding gene..		Gene Expression Ontology	#GEXO
A4GALT	A4GALT	Preferred	CD77 synthase blood group..		Ontology of Genes and Genomes	#OGG
AAAS	AAAS	Preferred	Protein coding gene..		Gene Expression Ontology	#GEXO

Table 3.7: Dominant Ontology : Example

Column	Dominant Ontology
tcga_participant_barcode	
cytoband	NCIT
locus_id	
gene	GEXO
all_copy_number	
cohort	

For instance, Paris is a name of city and can also name of a person.
In our example, as shown for column *gene*'s SDD instance level information in table 3.6. Majority of its instance i.e. *A1BG* and *AAAS* described it as a protein coding gene since both belong to *GEXO* ontology. In other words, it should be a common recurring ontology for respective column which we call it as Dominant Ontology. The Dominant Ontology for our example is in table 3.7

3.4.3 Semantic Indicators

Based upon SDD and Dominant Ontology developed, we build three types of indicators for each values.

- *Semantic Valid*: The value is semantic valid if it meet *all* of the following properties:

  - full text is found from the ontology library service
  - the text is used as preferred name by that ontology
  - the text's ontology is same as the dominant ontology

  Example: AAAS shown in table 3.8 is semantic valid because - Parsed label is same as Label value given in table 3.6; it is Preferred type; and it belongs to Dominant Ontology which is GEXO.

- *Semantic Partially Valid*: The value is semantic partially valid if it meet *either* of the following properties:

  - partial text is found from the ontology library service
  - the text is used as an alternative name by that ontology

Table 3.8: Semantic Indicator: Example

Indicator Type	**Values**
Semantic Valid	21q21.2, 22q13.2,12q13.13,A1BG,AAAS
Semantic Partially Valid	19q13.43
Semantic Invalid	A4GALT,-6309,1,53947,8086

and

- the text's ontology is same as the dominant ontology

Example: 19q13.43 shown in table 3.8 is semantic partially valid because Parsed label is subset of Label value given in table 3.6; and it belongs to Dominant Ontology which is GEXO.

- *Semantic Invalid*: The value is semantic invalid if it meets *either* of the property:
  - the text's ontology is not same as the dominant ontology
  - text that is not found from ontology library service
  - text belongs to columns that are of numeric types

Example: A4GALT shown in table 3.8 is semantic invalid because it does not belong to Dominant Ontology.
It is important to note - even though alternative or synonymous names belong to dominant ontology, they are marked as *Semantic Partially Valid*. This is because that in our study of an ontology library service, the authors claimed that synonyms are not reliable[66].

3.4.4 Defintions

Let us first give notations and formalize definition used in the pseudo-code of the semantic profiling algorithm.
Let Semantic Data Dictionary at schema level called DD_{schema} and at instance level called $DD_{instance}$. Each column C_i (i=1..n) belonging to the input source S contains set of values v_j (j=j..m). These values v_j has data type such as String, Numeric.
Semantic Data Dictionary attributes are denoted as : Label(L), Parsed Label (PL), Type(T), Description(D), Synonyms(S), Ontology Name(O_{name}), Ontology Uri(O_{uri}). Where:

- L accepts C_i or v_j
- $PL(v_j)$ indicates coverage result found by the ontology library for the value v_j
- $T(v_j)$ indicates if the type found by the ontology library for the value v_j is preferred or synonymous
- $D(v_j)$ denotes description of the coverage result found by the ontology library for the value v_j
- $O_{name}(v_j)$ indicates ontology name from which this coverage result belongs to
- $O_{uri}(v_j)$ indicates the ontology Uri for the coverage result

Schema Level Data Dictionary is denoted as DD_{schema}. Instance Level Data Dictionary is denoted as $DD_{instance}$.

Definition 1: Dominant Ontology (Dom): It is the most recurring O_{uri} found among all values v_j belonging to each column C_i.

Definition 2: Semantic Valid Values(FV): A value v_j is fully semantic valid value iff:

$$v_{\text{j}} = PL(v_{\text{j}}) \wedge O_{\text{uri}}(v_{\text{j}}) \in Dom \wedge T(v_{\text{j}}) = Preferred$$

Definition 3: Semantic Partially Valid Values(PV): A value v_j is semantic partially valid value iff:

$$PL(v_{\text{j}}) \subset v_{\text{j}} \vee T(v_{\text{j}}) = Synonym \wedge O_{\text{uri}}(v_{\text{j}}) \in Dom$$

Definition 4: Semantic Invalid Values(I): A value v_j is semantic invalid value iff:

$$O_{\text{uri}}(v_{\text{j}}) \notin Dom \vee v_{\text{j}} \notin DD_{\text{instance}} \vee DataType(v_{\text{j}}) \in \{\mathbb{R}\}$$

Algorithm

Given the valid profiled table T, the algorithm returns DD_{schema}, $DD_{instance}$, Dom, and Semantic Indicators FV, PV, I.

The function **generateSchema** takes table as an input and creates a schema out of it. This returns a data structure containing name of the columns and their respective data types.
The role of **createSemanticDataDictionary** is to create Data Dictionary at two levels: 1) $\text{DD}_{\text{schema}}$ is created first by accepting column names as input 2) $\text{DD}_{\text{instance}}$ is developed by iterating through all the distinct values v for each column
The **generateDominantOntology** function takes $\text{DD}_{\text{instance}}$ as an input and develops Dominant Ontology Dom table as described in Definition 1.
The **generateSemanticIndicator** function accepts the valid profiled table T, $\text{DD}_{\text{instance}}$ and Dom as input and generated the indicators FV, PV, and I according to Definition 2, 3, 4 respectively.

Algorithm 1: Semantic Profiling Algorithm

```
1  function ($T$);
   Input  : Table $T$
   Output: $DD_{\text{schema}}$,
           $DD_{\text{instance}}$,
           $Dom$,
           $FV$,
           $PV$,
           $I$
2  C:=generateSchema(T)
3  for each $C_i$ from C do i=1..n
4  $DD_{\text{schema}}$ := createSemanticDataDictionary($C_i$)
5  for each $v_j$ from $C_i$ do j=1..m
6  $DD_{\text{instance}}$ = createSemanticDataDictionary($v_j$)
7  end for;
8  end for
9  $dom$:=generateDominantOntology($DD_{\text{instance}}$)
10 $FV$, $PV$, $I$:= generateSemanticIndicators($T$, $DD_{\text{instance}}$, $dom$)
11 return $DD_{\text{schema}}$,$DD_{\text{instance}}$,$Dom$,$FV$,$PV$,$I$;
```

4 Realization of Concept

In this chapter, we discuss the following: justification to use the right ontology service that suit our research method; algorithm implementation where we briefly discuss about the technology we used to implement the algorithm; and ultimately discuss about its integration into state-of-art data lake.

4.1 Selection of appropriate Ontology Library

Ontology library provides a mean to access and reuse existing ontology and vocabulary resources. There are few available ontology libraries that support data discovery. We have studied the following ontology library services:

- **The OBO Foundry:** It is a *domain-specific* crowd-source ontology directory focuses on collection of well-documented biomedical ontologies that supposed to work in inter-operable fashion[67]. The ontology collection hosts over 100 ontologies. It is established over time where users submit ontologies. These ontologies are reviewed according to an editorial process defining which ontologies become part of that collection. One of the major drawbacks is lack of browsing ontologies or ranking of relevant ontologies.
- **Ontology Lookup Service:**It is a *domain-specific* bank of biomedical ontologies. It hosts nearly 80 ontologies. Its main goal is to provide query access and browsing through ontological sets. The querying can be within and across ontology(s)[68]. The results are shown in document-centric format have no ranking or order.
- **Linked Open Vocabulary(LOV):**It is a *generic* ontology library service, used or supported by datasets in Linked Data Cloud[69]. Their browsing scope is limited to ontological terms or ontologies themselves. There is no support for text search. To rank the results, it uses the term *popularity* in Linked Data Cloud and in LOV framework.

- **BiOSS:** It is another crowd-source *domain-specific* ontology system used specifically for biomedical ontologies[70]. Based on keyword provided by the user, it ranks ontology(s) using the following criterion (i) the input coverage; (ii) the semantic richness of the ontology for the input;(iii) the acceptance of the ontology. The drawbacks are its criterion limitation; lack of regular updates with new ontologies.
- **BioPortal:** It is a *domain-specific* ontology library to publish and explore over 400 biomedical ontologies [71][66]. Their greatest offering is ontology recommendation service which uses extensive thoughtful criterion. Furthermore, its availability as REST Web services in different language gave it a strong contender as a chosen candidate.

Selected Ontology

To meet the three criteria and adhere to the definition of semantic profiling as mentioned in section 3.2 , we used *BioPortal API* vocabulary service for the semantic annotation of datasets.

The recommendation algorithm takes two tuning factors to rank the ontologies, as follows:

- Evaluation criteria: To evaluate the relevance of ontology, there are following four criteria as weight that are taken together into consideration:
  - Coverage: To what extend does the ontology cover the input data?
  - Acceptance: How accepted the ontology is within the community?
  - Detail Knowledge: What is the level of detail the ontology carries for input data?
  - Specialization: How specialized the ontology is to domain of input data?

  these criterion are weighed as probabilistic value between 0.0 and 1.0 such that sum of all weight is 1.0
- Input data: In what mode does the term need to be searched in BioPortal? i.e. as *list of words* or *keywords*.
  For example, if the input data is "Lung Cancer" and the mode is marked as *list of words*, the ontology would look for relevant ontology that may contain

"Lung" and "Cancer" whereas *keywords* mode would crawl for "Lung Cancer" term.

In our implementation, we use the default setting of the recommender system where each evaluation criteria is marked as 0.25 and use a list of words as input. This is due to the following:

- Equal weight distribution helps in giving equal considerations to all factors in retrieving the appropriate ontology.
- Unlike keyword mode, list of words mode generates more coverage on the data to be annotated

4.2 Algorithm Implementation

Our algorithm's implementation is written as a JAVA program which takes input as the valid values resulting from profile, timestamp value at each ingestion is marked, Bioportal API Key which can be found[1]. The output results in Data Dictionary at schema & level respectively, Dominant Ontology, and three inductors for semantic profiling (i) Fully Valid, (ii)Partially Valid and (iii) Invalid.

If BioPortal API Key used is invalid, it logs the error and sends the bug report to the client. If it is valid, it connects to the profile table T and gets its schema. DD_{schema}, it is enriched for each attribute with values from BioPortal. We use the following semantic information as specification provided in Semantic Data Dictionary- Label, Parsed Label, Type, Description, Synonyms, Ontology Name, Ontology URI. It is also to be noted that the Ontology URI are actually HTTP schemes that lead to a document-centric result of the ontology. $DD_{instance}$ is enriched by taking 2d list containing record name and frequency of each record occurring per column. We proceed with the rest of the code with the definitions mentioned as *Definition 1,2,3,4*

4.2.1 Architectural consideration and implementation

Our algorithm could be plugged in any Data Lake system. The implementation decision to use a Hadoop based Data Lake is based on its increasing use in Data

[1] `https://bioportal.bioontology.org/login`

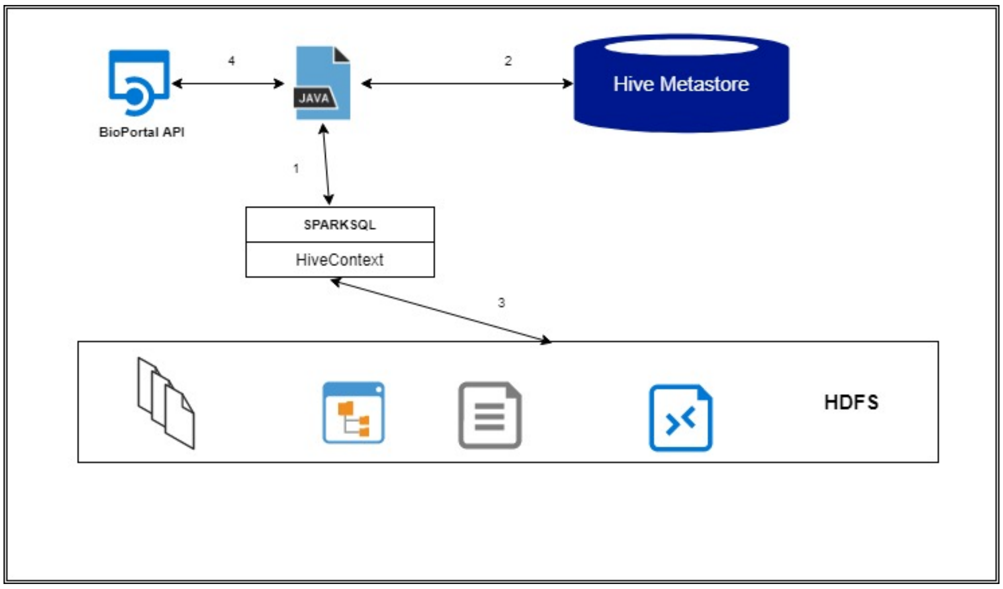


Figure 4.1: High level implementation architecture

Lakes. In order to realize our algorithm, the first challenge was to find solution for performance constraint in Hadoop as discussed in section 2.1 . This is because our algorithm requires hefty operations on the table stored in HDFS.
Therefore, the decision was to move away from Hadoop's disk-based processing Map-Reduce to a platform which does in-memory computation. Hence, we decided to use and run Spark[2] over Hadoop to achieve fast data processing. It uses Resilient Distributed Dataset (RDD) data structure. This helps Spark to transparently store data in-memory and send to disk only when needed. That is how we will be able to successfully minimize high disk latency for our algorithm.
The architecture of our implementation is shown in fig. 4.1. The table that we intend to use for algorithm implementation is stored in HDFS. Our JAVA program uses SparkSQL module of Spark library that reads the metadata about the valid profiled table in Hive Metastore. The Hive Metastore returns the destined metadata information and then HiveContext method accesses the data.
Once data is accessed as an RDD, our algorithm continue the semantic enrichment process using BioPortal API, as explained in section 3.4.4.

[2]`https://spark.apache.org/`

4.3 Integration of algorithm in Data Lake

4.3.1 The Hadoop Stack & Kylo

Kylo is a big data lake system developed by Think Big[3] . It runs on top of Hadoop distribution stack offered by Hortonworks Data Platform [4]. This Hadoop distribution stack hosts pre-built Hadoop services.
Kylo uses the graph based data flow system Nifi[5] for data routing and transformation. Kylo uses a terminology called *template* for creating a reusable end-to-end data ingestion flow. Following the reusable approach of Kylo, we integrated our module in Kylo, by creating a customized ingestion template. Therefore any ingestion flow that is designed using our customized template will have the extended vocabulary based semantic annotation as an inherent feature.

4.3.2 Integration Method

We use Kylo's template feature to build the flow. These templates are categorized with respect to Kylo framework into two parts - *flow template* and *reusable template*. *Flow template* contains set of initial data source connectors, and other components in this flow where Kylo can inject common metadata configuration through the wizard, and connection port which is meant to be connected to a re-usable flow. *Reusable flow*, on the other hand, initializes a single running instance of the flow that can support multiple connected feed and it connects through port defined in flow template. It is to be noted that the entire data routing design for templates are done in Apache Nifi. Each block in data routing templates play unique roles and these blocks in Nifi's terminology are known as Processors. Once design is completed, they are exported as a file or via Kylo's inter-operable feature pulls the template into its environment for full flow initialization.

In this work, we designed an extension of standard ingestion using both flow template and reusable template to fully meet the research goal. We named it *semantic ingest template*
In semantic ingest template, data come from HTTP endpoints as well as file-systems, we have integrated them in one flow. Furthermore, we added a new component

[3]`https://www.thinkbiganalytics.com/`
[4]`https://hortonworks.com/products/data-platforms/hdp/`
[5]`https://nifi.apache.org/`

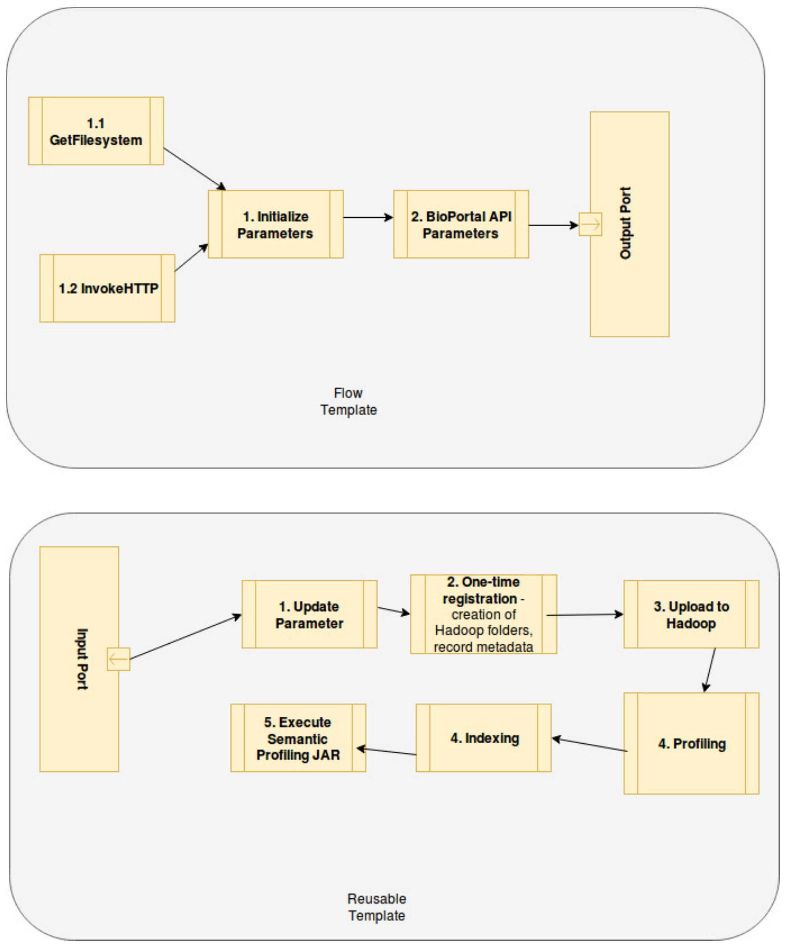
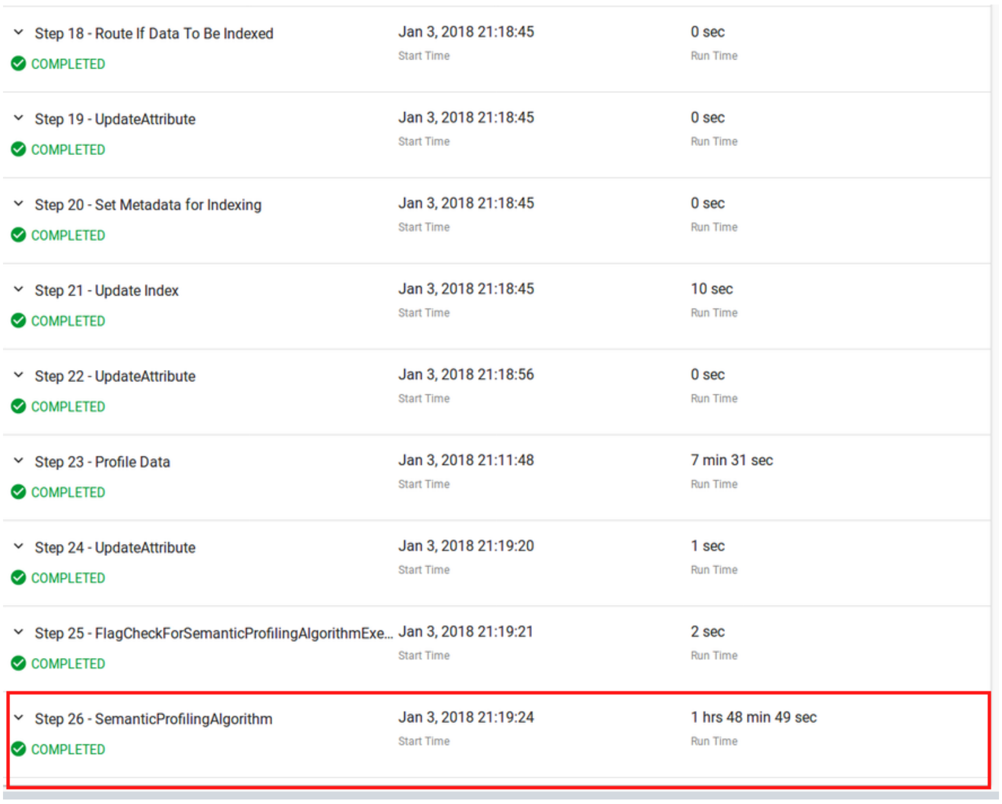


Figure 4.2: Templates - Feed Template and Reusable Template for semantic profiling

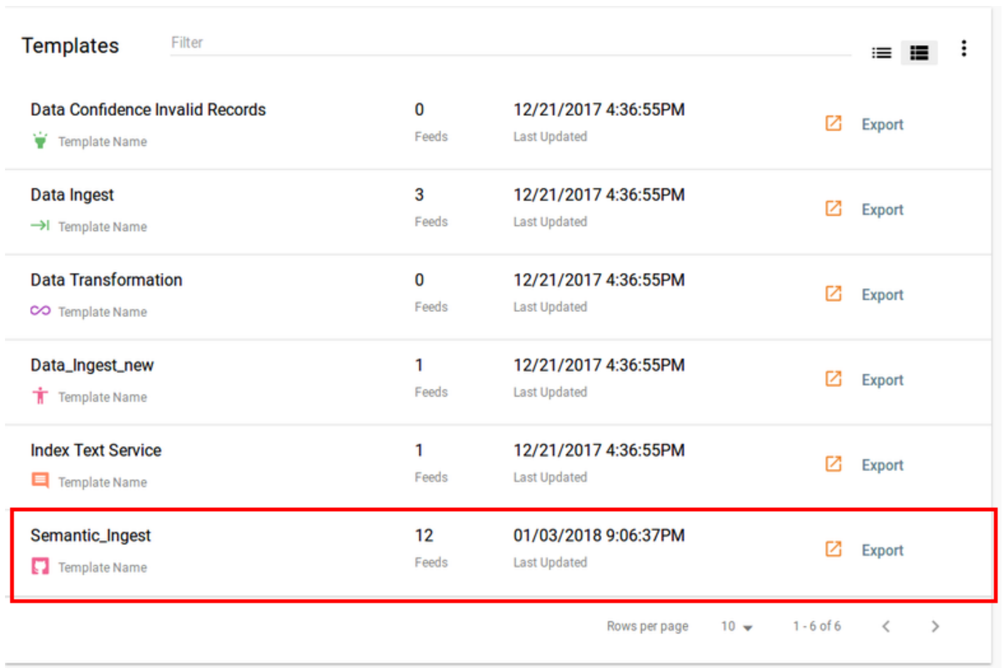
which accepts the API Key from BioPortal. Whereas, in the reusable template, we extend the flow using a processor called *ExecuteStreamProcessor*, which invokes and executes our Semantic Profiling Algorithm stored as a JAR file. A high level overview of the two templates described is shown in fig. 4.2:

During/after execution of results the users can track the ETL provenance, it provides job execution information in detail at each step as shown in fig. 4.3. In the fig. 4.3, it shows that the *SemanticProfilingAlgorithm* step has been completely executed. Furthermore, in the fig. 4.4 we can see all our registered templates. Our registered Semantic Ingestion Template can be seen as *SemanticIngest* in the figure.



Step 18 - Route If Data To Be Indexed ✓ COMPLETED	Jan 3, 2018 21:18:45 Start Time	0 sec Run Time
Step 19 - UpdateAttribute ✓ COMPLETED	Jan 3, 2018 21:18:45 Start Time	0 sec Run Time
Step 20 - Set Metadata for Indexing ✓ COMPLETED	Jan 3, 2018 21:18:45 Start Time	0 sec Run Time
Step 21 - Update Index ✓ COMPLETED	Jan 3, 2018 21:18:45 Start Time	10 sec Run Time
Step 22 - UpdateAttribute ✓ COMPLETED	Jan 3, 2018 21:18:56 Start Time	0 sec Run Time
Step 23 - Profile Data ✓ COMPLETED	Jan 3, 2018 21:11:48 Start Time	7 min 31 sec Run Time
Step 24 - UpdateAttribute ✓ COMPLETED	Jan 3, 2018 21:19:20 Start Time	1 sec Run Time
Step 25 - FlagCheckForSemanticProfilingAlgorithmExe... ✓ COMPLETED	Jan 3, 2018 21:19:21 Start Time	2 sec Run Time
Step 26 - SemanticProfilingAlgorithm ✓ COMPLETED	Jan 3, 2018 21:19:24 Start Time	1 hrs 48 min 49 sec Run Time

Figure 4.3: Provenance showing job execution of each stage



Templates

Template Name	Feeds	Last Updated	
Data Confidence Invalid Records	0	12/21/2017 4:36:55PM	Export
Data Ingest	3	12/21/2017 4:36:55PM	Export
Data Transformation	0	12/21/2017 4:36:55PM	Export
Data_Ingest_new	1	12/21/2017 4:36:55PM	Export
Index Text Service	1	12/21/2017 4:36:55PM	Export
Semantic_Ingest	12	01/03/2018 9:06:37PM	Export

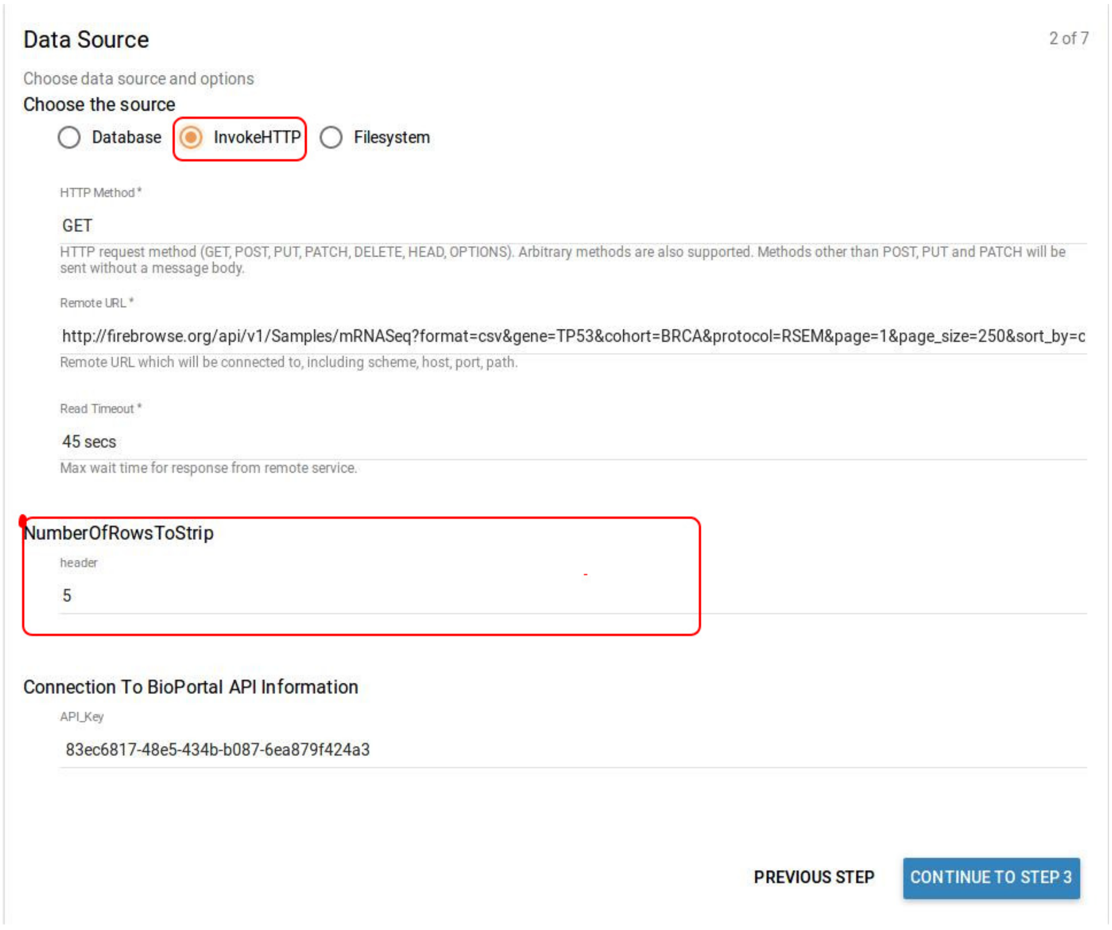
Rows per page 10 1 - 6 of 6

Figure 4.4: Template list registered in Kylo

4.4 Kylo's Feature Enhancement using Ingestion Template

Using Kylo's promise of high extensibility, we have implemented two additional features to guide user in ingesting different types of data sources having varied content into Kylo as shown fig. 4.5.

- Currently, Kylo only offers data ingestion using either Database or File-system. We have also integrated HTTP end-points to this feature.
- For SQL abstraction over semi-structured/unstructured data in Hadoop, it is important to slash header details. These header details can also cover numerous rows. However, Kylo assists only one header row to slash. In many cases like ours in evaluation study, data source header spans over multiple rows, it is important to give user a flexibility to slash variable number of rows.



Data Source

2 of 7

Choose data source and options

Choose the source

Database [✓ InvokeHTTP] Filesystem

HTTP Method *

GET

HTTP request method (GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS). Arbitrary methods are also supported. Methods other than POST, PUT and PATCH will be sent without a message body.

Remote URL *

http://firebrowse.org/api/v1/Samples/mRNASeq?format=csv&gene=TP53&cohort=BRCA&protocol=RSEM&page=1&page_size=250&sort_by=c

Remote URL which will be connected to, including scheme, host, port, path.

Read Timeout *

45 secs

Max wait time for response from remote service.

NumberOfRowsToStrip

header

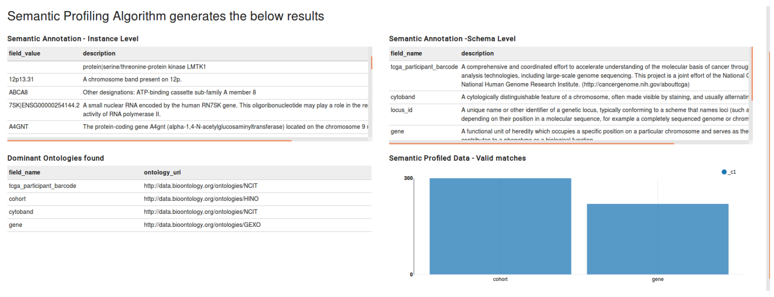
5

Connection To BioPortal API Information

API_Key

83ec6817-48e5-434b-b087-6ea879f424a3

Figure 4.5: Miscellaneous Implementation



Semantic Profiling Algorithm generates the below results

Semantic Annotation - Instance Level

field_value	description
	protein|serine/threonine-protein kinase LMTK1
12p13.31	A chromosome band present on 12p.
ABCA8	Other designations: ATP-binding cassette sub-family A member 8
7SK|ENSG00000254144.2	A small nuclear RNA encoded by the human RN7SK gene. This oligoribonucleotide may play a role in the re activity of RNA polymerase II.
A4GNT	The protein-coding gene A4gnt (alpha-1,4-N-acetylglucosaminyltransferase) located on the chromosome 9

Semantic Annotation -Schema Level

field_name	description
tcga_participant_barcode	A comprehensive and coordinated effort to accelerate understanding of the molecular basis of cancer throug analysis technologies, including large-scale genome sequencing. This project is a joint effort of the National C National Human Genome Research Institute. (http://cancergenome.nih.gov/abouttcga)
cytoband	A cytologically distinguishable feature of a chromosome, often made visible by staining, and usually alternatin
locus_id	A unique name or other identifier of a genetic locus, typically conforming to a scheme that names loci (such a depending on their position in a molecular sequence, for example a completely sequenced genome or chrom
gene	A functional unit of heredity which occupies a specific position on a particular chromosome and serves as the contributes to a phenotype or a biological function

Dominant Ontologies found

field_name	ontology_uri
tcga_participant_barcode	http://data.bioontology.org/ontologies/NCIT
cohort	http://data.bioontology.org/ontologies/HINO
cytoband	http://data.bioontology.org/ontologies/NCIT
gene	http://data.bioontology.org/ontologies/GEXO

Semantic Profiled Data - Valid matches

Figure 4.6: Dashboard in Zeppelin

4.4.1 Integration with Hadoop Stack

As Kylo works on top of Hadoop stack, it brings wide range of opportunities to deploy Hadoop-compatible applications ranging from machine learning tools like Mahout[6] to visualization service like Zeppelin [7].

To demonstrate our algorithm in terms of big data processing lifecycle, we demonstrate our last phase i.e. data interpretation in form of visualization. Thus, we present the sample dashboard shown in fig. 4.6

[6]`https://mahout.apache.org/`

[7]`https://zeppelin.apache.org/`

5 Evaluation

We ran experiments on a single node cluster in Virtual Machine. This Virtual Machine(VM) hosts pre-configured Data Lake instance with Hadoop, Spark, and NIFI. The system configuration of the VM that we set is 11GB RAM and 4 virtual CPU core.
We evaluate our tool with data sets from The Cancer Genome Atlas(TCGA)[1]. In order to demonstrate multiple data sources, two different TCGA data sets are ingested from two different portals, namely cBioPortal[2] and Firebrowse[3]. We ingested heterogeneous data in different forms i.e. as HTTP source and from filesystem into Hadoop for semantic annotation [63]. Through our evaluation, we observed that the developed semantic profiling tool can be utilized into two aspects -
a) Schema Level: recognizing if schema is enriched with annotation and then identify any sort of relationships existing across different datasets
b) Data Instance Level: if schema fails to enrich with annotation then recognize relationships across different datasets using semantic indicators at instance level.
For the Schema Level we performed two schema alignment evaluation based on attributes that are syntactically different and partially different respectively:

- Evaluation 1: Syntactically different - We ran both of the data sources S1 and S2 as HTTP request from Firebrowse with different analyses as shown in table 5.1. After running the algorithm, based on Data Dictionary at Schema level, we identify that geneId generates three synonyms
[*GeneID,Locus_ID,EntrezGene_ID*]
whose description field is "Indicates the accession number for a Gene within the Entrez Gene database." and ontology belongs to National Cancer Institute Thesaurus. We can infer that *geneId* and *locus_id* are semantically similar although both the attributes in syntax term are dissimilar.
- Evaluation 2: Syntactically partial different- We ran data sources S1 as Filesystem and S2 as HTTP request from Bioportal and Firebrowse respectively as

[1]`https://cancergenome.nih.gov/`
[2]`http://www.cbioportal.org/`
[3]`http://firebrowse.org/`

Table 5.1: Evaluation 1

S1(Firebrowse)	S2(Firebrowse)
cohort	gene_expression_log2
cytoband	z-score
date	cohort
gene	sample_type
locus_id	Protocol
tcga_participant_barcode	*geneId*
all_copy_number	tcga_participant_barcode

shown in table 5.2. After running the algorithm, based on Data Dictionary at Schema level, we identify that *entrez_gene_id* generates synonyms [*Gene identifier (NCBI)Gene identifier (NCBI),NCBI gene ID,Entrez gene ID,NCBI geneid,Gene identifier (Entrez)*] whereas *geneId* generates three synonyms [*GeneID,Locus_ID,EntrezGene_ID*]. We can infer that *geneId* and *locus_id* are semantically similar although both the attributes in syntax term are partially different.

For the Data Instance Level we performed another evaluation. In this case Data Dictionary at schema level failed to find any common attribute based on synonym lists as shown in table 5.3.

- Evaluation 3: Similar datasets but no common structure - Data Dictionary at schema level identifies *cohort*'s synonym list as *Cohort* and on the other hand *type_of_cancer_id* finds no synonym lists.
  Therefore, we now proceed to look for any hints at instance level i.e. we check dominant ontology for both, we discover that for both the attributes for both tables the dominant ontology found to be is *http://data.bioontology.org/ontologies/NCIT*. This ontology is National Cancer Institute Thesaurus, which covers cancer research domains.
  We drilled down for further analysis based on semantic indicators. Upon further examination through semantic indicator results, we find that both the dataset describe the same background, which is cancer acronyms with description. Hence, even though we have different schema structure with no common synonyms as well, we still ended up examining at instance level. It turned out the data are fully similar to each other.
  It is important to point out that there were no results found for semantic valid

Table 5.2: Evaluation 2

S1(cBioPortal)	S2(Firebrowse)
entrez_gene_id	gene_expression_log2
gene_symbol	z-score
case_id	cohort
Sequencing center	sample_type
Mutation status	Protocol
mutation_type	*geneId*
validation_status	tcga_participant_barcode
amino_acid_change	
functional_impact_score	
xvar_link	
xvar_link_pdb	
xvar_link_msa	
chr	
start_position	
end_position	

Table 5.3: Evaluation 3

S1(cBioPortal)	S2(Firebrowse)
type_of_cancer_id	*cohort*
name	*description*

values. It is because data contains generic domain information instead of biological details. Semantic Data Dictionary at instance level showed either partial coverage or full coverage but of synonymous type from the contained ontology. Therefore, we had to examine only semantic partially valid values. Before examining, we decided run F-measure tests on them to check how precise and robust the partial results are. For Source S1 that contained 500 records with execution time of nearly two hours, F-measure showed 68% whereas S2 with 50 records having execution time less than 5 minutes resulted in 83.1%.

The complete end-to-end execution time for the different evaluation cases are as shown in table 5.4.

Table 5.4: Execution Time of different evaluation cases

Evaluations	**S1**	**S2**
Evaluation 1	30 min. 26 sec.	8 min. 5 sec.
Evaluation 2	1 hrs. 0 min. 5 sec.	8 min. 5 sec.
Evaluation 3	1 hrs. 58 min. 12 sec.	4 min. 33 sec.

5.1 Discussion

Through extensive study on ingestion of different types of data with large sample of data content, we were successfully able to get insightful revelation on schema matching through our technique. There are couple of points that we would like to discuss regarding our evaluation and reflect upon our algorithm.
It is interesting to add to our results that BioPortal also helps in finding generic as well as cross-domain semantic information. For instance, data source S2 shown in table 5.1 has *z-score* attribute which belongs to mathematical background. This is evident from explanation in Data Dictionary. Through this evaluation study, BioPortal integration into data lake can vastly help clinicians and data scientists in their study. But it is also important to add that we could rely on BioPortal in semantic discovery of data containing non-biomedical information. For example, clinical datasets.
In case 1 & 2, the data contain biological specific information, our semantic data dictionary at schema level is sufficient enough to guide in finding the common attribute based on preferred and synonymous names.
In case 3, we evaluated our algorithm on attributes with generic names. But unlike previous cases, results found from generic long texts such as *type_of_cancer_id* or short texts like *name* or *description* were either ambiguous or only partial match. This lack of complete semantic dictionary information at schema level led us to study the results of instance level data dictionary and its artifacts. Although the dominant ontology for both the tables generated was *NCIT*, there exists no semantic valid values. The results of semantic partial valid values were either synonymous names or partial text found. It is not always possible that using these synonymous names or partial results alone one could guarantee the semantic on the data. Thus, lack of semantic valid values may provide incorrect observation about the overall domain understanding of the data source.
Based on F-measure test as described in case 3, we found 68% for larger generic domain dataset, this motivates us to do further evaluation on large data sets based on datasets containing clinical information about the patients. This would help us in identifying improvement points on our algorithm.

Currently from our technique, in the election of choosing the *Dominant Ontology* when two or more Ontology URI has the same frequency, Dominant Ontology is chosen based on lexical order[4].
We executed the results on different volumes of data ranging from 37 to 500 records. The execution time of end-to-end semantic profiling in data lake varied from 8 minutes to as high as 2 hours. Even though the performance can be significantly improved by tuning parameters in Hadoop and Spark. But the true performance of Data Lake comes into picture when it is deployed on a multi-node cluster.

5.1.1 Challenges and Best Practices

We will discuss challenges that we faced during the implementation and proposed best practices for them. It is important to highlight these as they will help in the future course of evaluation.

- Cluster Management – Once the algorithm is production-ready it is important to do resource super-visioning and cluster health check up.
  Best practice would to make sure to constantly monitor and delete old logs from Hadoop, Mapreduce, Nifi etc. if not needed.
- Development: We used Maven[5] as a tool to build all the relevant libraries for our algorithm's implementation. We encountered various issues like dependency version conflicts etc. Best practice would be to check the Hadoop and Spark version in the data lake solution before deploying our algorithm into it.
- Integration: During integration phase, we encountered problems like Out of Memory errors in Nifi. Best practice is choosing appropriate processors from vast array of similar kind of processors to do the right job.
- Scheduling: Scheduling of data ingestion into Kylo comes with two options - CRON and timer based. One shall make sure to use Timer option carefully. We noticed that scheduler starts every time cluster restarts. It is irrespective of the timer value set i.e daily or weekly or hourly. // Best practice would be to use CRON expression for schedulers.
- Ontology Service availability - We have noticed during one of the execution phase of semantic annotation, the BioPortal service was unavailable.

[4]https://en.wikipedia.org/wiki/Lexicographical_order
[5]https://maven.apache.org/

It is important to make sure to choose the ontology library service that has the high availability.

6 Conclusion & Future Work

6.1 Conclusion

In context to data lake problems - data swamps, we have proposed a semantic profiling approach for data lakes in this thesis.
In order to do so we first delved into research problems of data lake from the perspective of semantic profiling. We clarified questions like - what uniquely characterizes them? what are the works that has been done in this area? has there been past research study done in data lake for this? how can survey study on various semantic annotation tools contribute to our research focus? From these studies, we distilled core activity that can be achieved using semantic profiling. Also, how can this algorithm be extended to suit many ontologies rather than using one ontology in evolutionary fashion.
The algorithm identifies data alignments using both schema and instances. To present our formalized algorithm with a research claim, we briefed on a motivational example. This described how health-care experts are facing challenges of using data lake if the biological information is not properly managed. We kept this domain throughout our work as an example.
With this domain, we realized the methodology using state-of-art data lake. Before realization of concept, we described the rationale behind selecting the appropriate ontology service for the domain. We chose BioPortal service upon serious considerations based on semantic profiling criterion in this case.
The proposed algorithms is integrated into the Kylo data lake system and is evaluated using the Cancer Genome dataset.
The experiments showed identification of semantic relationships between datasets, even when they are totally syntactically different. The detailed meta-data information would give bio-informaticians and data scientists a better understanding of the dataset. To re-emphasize, this approach is extensible for other data domains as long as relevant ontology service that can meet the properties of Semantic Data Dictionary.

6.2 Future Work

We have identified following areas for the future work.

- Adding user control for parameter tuning of ontology library in Data Lake: During setting up semantic ingestion feed in data lake, our algorithm currently do not expose the parameters of ontology library. These parameters are currently set as default values which are already discussed in section 4.1. As evident from the evaluation, this lack of end-user control on the parameters has produced rigid results. However, if we allow the end user to understand the data first and then accordingly change these parameters, the results can be significantly improved.
  For example, if a dataset needs to be semantically annotated and which contains a lot of multi-words, the text-mode can be selected as *keywords* from user interface of data lake.
  Furthermore, if user's central goal for semantic annotation is to find term matching, the *coverage* weight can be increased and the other corresponding weights can be kept smaller. Or, user is interested in finding popular ontology then *acceptance* can be kept higher. Whereas, if user's goal is to find semantic richness then *detail* weight can have a higher value than others. Lastly, if user's focus is to find annotations containing detailed information of sub-domains or classes containing the text then *specialization* is set higher than others.
- Improving Dominant Ontology metric: Finding the dominant ontology can be challenging for large volume data. The following strategy that could improve this metric.
  - The election process of choosing dominant ontology can be set and stopped as soon as the ontology URI recurred reached the 50% of the data size. This improvement can help in avoiding processing large datasets.
  - Apply caching strategies in the algorithm: For cases like above discussed, it is a good idea to apply caching strategies. As we deployed our algorithm using Spark, methods[1] like *cache()* and *persist()* should be considered on the Spark's parallel data structures called RDDs.

[1]https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html

- Improve algorithm with multiple ontology recommender systems: In the evaluation, we observed that semantic annotations can have ambiguous meanings if domain-specific ontology libraries cannot find cross-domain or generic meta-information. This could affect the overall domain understanding of the data source. Thus, through future work, algorithm can be improved by integrating generic library like Linked Open Vocabulary [69] on top of domain-specific library. This can annotate generic data if domain-specific library fails to annotate or returns partial results.
  Due to Kylo's integrated provenance feature, end-user can ultimately trace and identify the concerned semantic annotations from ontology service.

- Integrate an ontology alignment engine for analyzing relationships between datasets: We manually discovered relationship between different datasets using SDD in our work. But, once these SDD become Big Data themselves, inferring schema and instance relationships gets difficult. Therefore, these relationship inference can be handled using ontology alignment engine.
  One of the future works is to convert the generated SDDs into respective RDF N-triple ontology which represents SDDs' schema and its instance. We provide input of these N-triple ontology to ontology alignment engines like PARIS [72] or COMA++ [73]. These engines produces a measure which indicates a value. Which is, higher the measure more similar are the N-triple ontologies. This in turn would mean more similar two different SDDs. From this, we can infer the measure of similarity between two ingested datasets in data lake.

Bibliography

[1] *White paper: The EDW Lives On The Beating Heart of the Data Lake*. 2017.

[2] Ignacio G Terrizzano et al. "Data Wrangling: The Challenging journey from the Wild to the Lake." In: *CIDR*. 2015.

[3] Isuru Suriarachchi and Beth Plale. *Provenance as Essential Infrastructure for Data Lakes [Preprint, forthcoming in IPAW 2016]*.

[4] Philip Russom. *White Paper: Data Lakes Purposes, Practices, Patterns, and Platforms*. 2017. URL: `https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper2/tdwi-data-lakes-108964.pdf`.

[5] *Gartner Says Worldwide Enterprise IT Spending to Reach $2.7 Trillion in 2012*. `https://www.gartner.com/newsroom/id/1824919`. Accessed: 2018-01-21.

[6] James Manyika et al. "Big data: The next frontier for innovation, competition, and productivity". In: (2011).

[7] Menno Mostert et al. "Big Data in medical research and EU data protection law: challenges to the consent or anonymise approach". In: *European Journal of Human Genetics* 24.7 (2016), pp. 1096–1096. DOI: `10.1038/ejhg.2016.71`. URL: `https://doi.org/10.1038/ejhg.2016.71`.

[8] *Turning data into new business models*. `https://www.bundesregierung.de/Content/EN/Artikel/2017/06_en/2017-06-13-merkel-digital-gipfel_en.html`. Accessed: 2018-01-21.

[9] M. J. Khoury and J. P. A. Ioannidis. "Big data meets public health". In: *Science* 346.6213 (2014), pp. 1054–1055. DOI: `10.1126/science.aaa2709`. URL: `https://doi.org/10.1126/science.aaa2709`.

[10] Natalia Miloslavskaya et al. "Information security maintenance issues for big security-related data". In: *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE. 2014, pp. 361–366.

[11] Marcos D. Assunção et al. “Big Data computing and clouds: Trends and future directions”. In: *Journal of Parallel and Distributed Computing* 79-80 (2015), pp. 3–15. DOI: 10.1016/j.jpdc.2014.08.003. URL: https://doi.org/10.1016/j.jpdc.2014.08.003.

[12] Wullianallur Raghupathi and Viju Raghupathi. “Big data analytics in healthcare: promise and potential”. In: *Health Information Science and Systems* 2.1 (2014). DOI: 10.1186/2047-2501-2-3. URL: https://doi.org/10.1186/2047-2501-2-3.

[13] *IBM and Hortonworks: Accelerating data-driven decision making*. http://www.ibmbigdatahub.com/blog/ibm-and-hortonworks-accelerating-data-driven-decision-making. Accessed: 2018-01-28.

[14] *The Deciding Factor: Big Data & Decision Making*. https://www.capgemini.com/wp-content/uploads/2017/07/The_Deciding_Factor__Big_Data___Decision_Making.pdf. Accessed: 2018-01-28.

[15] Bala M. Balachandran and Shivika Prasad. “Challenges and Benefits of Deploying Big Data Analytics in the Cloud for Business Intelligence”. In: 112 (Dec. 2017), pp. 1112–1122.

[16] *IT Glossary Big Data*. https://www.gartner.com/it-glossary/big-data. Accessed: 2018-02-5.

[17] Julie Frizzo-Barker et al. “An empirical study of the rise of big data in business scholarship”. In: *International Journal of Information Management* 36.3 (2016), pp. 403–413.

[18] Wo L Chang. *NIST Big Data Interoperability Framework: Volume 1, Definitions*. Tech. rep. 2015.

[19] Kostas Kolomvatsos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades. “An efficient time optimized scheme for progressive analytics in big data”. In: *Big Data Research* 2.4 (2015), pp. 155–165.

[20] Mariam Kiran et al. “Lambda architecture for cost-effective batch and speed big data processing”. In: *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015. DOI: 10.1109/bigdata.2015.7364082. URL: https://doi.org/10.1109/bigdata.2015.7364082.

[21] *Lambda Architecture for Big Data*. https://dzone.com/articles/lambda-architecture-big-data. Accessed: 2018-01-27.

[22] Natalia Miloslavskaya and Alexander Tolstoy. “Big Data, Fast Data and Data Lake Concepts”. In: *Procedia Computer Science* 88 (2016), pp. 300–305. DOI: 10.1016/j.procs.2016.07.439. URL: https://doi.org/10.1016/j.procs.2016.07.439.

[23] A. Katal, M. Wazid, and R. H. Goudar. “Big data: Issues, challenges, tools and Good practices”. In: *2013 Sixth International Conference on Contemporary Computing (IC3)*. 2013, pp. 404–409. DOI: 10.1109/IC3.2013.6612229.

[24] Konstantin Shvachko et al. “The hadoop distributed file system”. In: *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. Ieee. 2010, pp. 1–10.

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

[26] Aftab Chandio, Nikos Tziritas, and Cheng-Zhong Xu. “Big-Data Processing Techniques and Their Challenges in Transport Domain”. In: 13 (Feb. 2015), pp. 50–59.

[27] H. Hu et al. “Toward Scalable Systems for Big Data Analytics: A Technology Tutorial”. In: *IEEE Access* 2 (2014), pp. 652–687. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2332453.

[28] *Hadoop’s Limitations for Big Data Analytics*. https://www.whitepapers.em360tech.com/wp-content/files_mf/1360922634PARACCEL1.pdf. Accessed: 2018-02-08.

[29] *Data Access Subqueries*. https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.6.0/bk_data-access/content/hive-013-feature-subqueries-in-where-clauses.html. Accessed: 2018-02-08.

[30] *Hive Transactions*. https://cwiki.apache.org/confluence/display/Hive/Hive+Transactions. Accessed: 2018-02-08.

[31] Abhay Kumar Bhadani and Dhanya Jothimani. “Big Data: Challenges, Opportunities, and Realities”. In: (2016).

[32] Mandy Chessell et al. “Governing and managing big data for analytics and decision makers”. In: *IBM Redguides for Business Leaders* (2014).

[33] Huang Fang. “Managing data lakes in big data era: What’s a data lake and why has it became popular in data management ecosystem”. In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 2015. DOI: 10.1109/cyber.2015.7288049. URL: https://doi.org/10.1109/cyber.2015.7288049.

[34] *Pentaho, Hadoop, and Data Lakes*. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Accessed: 2018-01-28.

[35] *Data lake*. https://en.wikipedia.org/wiki/Data_lake. Accessed: 2018-01-28.

[36] Brian Stein and Alan Morrison. "The enterprise data lake: Better integration and deeper analytics". In: *PwC Technology Forecast: Rethinking integration* 1 (2014), pp. 1–9.

[37] Alon Y Halevy et al. "Managing Google's data lake: an overview of the Goods system." In: *IEEE Data Eng. Bull.* 39.3 (2016), pp. 5–14.

[38] *Data Lake, Data Reservoir, Data Dump... Blah, Blah, Blah...* https://infocus.dellemc.com/william_schmarzo/data-lake-data-reservoir-data-dumpblah-blah-blah/. Accessed: 2018-02-10.

[39] *Data Lakes: The biggest big data challenges*. http://analytics-magazine.org/data-lakes-biggest-big-data-challenges/. Accessed: 2018-01-28.

[40] Isuru Suriarachchi and Beth Plale. "Crossing analytics systems: A case for integrated provenance in data lakes". In: *e-Science (e-Science), 2016 IEEE 12th International Conference on*. IEEE. 2016, pp. 349–354.

[41] Amin Beheshti et al. "Coredb: a data lake service". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. 2017, pp. 2451–2454.

[42] Coral Walker and Hassan Alrehamy. "Personal data lake with data gravity pull". In: *Big Data and Cloud Computing (BDCloud), 2015 IEEE Fifth International Conference on*. IEEE. 2015, pp. 160–167.

[43] S. Staab and R. Studer. "Handbook on ontologies". In: *Science & Business Media. (cited on pages 1, 5, and 19*. 2013.

[44] Anila Sahar Butt. "Ontology Search: Finding the Right Ontologies on the Web". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: ACM, 2015, pp. 487–491. ISBN: 978-1-4503-3473-0. DOI: 10.1145/2740908.2741753. URL: http://doi.acm.org/10.1145/2740908.2741753.

[45] Thomas R Gruber. "A translation approach to portable ontology specifications". In: *Knowledge acquisition* 5.2 (1993), pp. 199–220.

[46] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. "Data profiling: A tutorial". In: *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM. 2017, pp. 1747–1751.

[47] Felix Naumann. "Data Profiling Revisited". In: *SIGMOD Rec.* 42.4 (Feb. 2014), pp. 40–49. ISSN: 0163-5808. DOI: 10.1145/2590989.2590995. URL: http://doi.acm.org/10.1145/2590989.2590995.

[48] WY Zhang and JW Yin. "Exploring Semantic Web technologies for ontology-based modeling in collaborative engineering design". In: *The International Journal of Advanced Manufacturing Technology* 36.9-10 (2008), pp. 833–843.

[49] Peter Christen. *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[50] *What is Semantic Annotation?* https://ontotext.com/knowledgehub/fundamentals/semantic-annotation/. Accessed: 2018-02-5.

[51] *White Paper: Semantic Profiling*. https://www.semanticarts.com/articles/semantics-and-ontologies/semantic-profiling/. Accessed: 2018-01-10.

[52] C. Böhm et al. "Profiling linked open data with ProLOD". In: *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*. 2010, pp. 175–178. DOI: 10.1109/ICDEW.2010.5452762.

[53] A. Alserafi et al. "Towards Information Profiling: Data Lake Content Metadata Management". In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 178–185. DOI: 10.1109/ICDMW.2016.0033.

[54] Aïcha Ben Salem, Faouzi Boufares, and Sebastiao Correia. "Semantic Recognition of a Data Structure in Big-Data". In: *Journal of Computer and Communications* 02.09 (2014), pp. 93–102. DOI: 10.4236/jcc.2014.29013. URL: https://doi.org/10.4236/jcc.2014.29013.

[55] Alexiei Dingli, Fabio Ciravegna, and Yorick Wilks. "Automatic semantic annotation using unsupervised information extraction and integration". In: *Proceedings of SemAnnot 2003 Workshop*. 2003.

[56] Borislav Popov et al. "KIM–semantic annotation platform". In: *International Semantic Web Conference*. Springer. 2003, pp. 834–849.

[57] Diana Maynard. "Multi-source and multilingual information extraction". In: *Expert Update* 6.3 (2003), pp. 11–16.

[58] Jeen Broekstra, Arjohn Kampman, and Frank Van Harmelen. “Sesame: A generic architecture for storing and querying rdf and rdf schema”. In: *International semantic web conference*. Springer. 2002, pp. 54–68.

[59] Rihan Hai, Sandra Geisler, and Christoph Quix. “Constance: An Intelligent Data Lake System”. In: *Proceedings of the 2016 International Conference on Management of Data*. SIGMOD ’16. San Francisco, California, USA: ACM, 2016, pp. 2097–2100. ISBN: 978-1-4503-3531-7. DOI: 10.1145/2882903.2899389. URL: http://doi.acm.org/10.1145/2882903.2899389.

[60] *Whitepaper: Anzo Smart Data Lake*. https://info.cambridgesemantics.com/smart-data-lake-whitepaper-download. Accessed: 2018-01-28.

[61] *Hadoop and AllegroGraph The Semantic Data Lake for Analytics*. https://allegrograph.com/hadoop-and-allegrograph-the-semantic-data-lake-for-analytics/. Accessed: 2018-01-28.

[62] *Semantic data lake architecture in healthcare and beyond*. http://www.zdnet.com/article/semantic-data-lakes-architecture-in-healthcare-and-beyond/. Accessed: 2018-01-28.

[63] Kyle Chang and et al. “The Cancer Genome Atlas Pan-Cancer analysis project”. In: *Nature Genetics* 45.10 (2013), pp. 1113–1120. DOI: 10.1038/ng.2764. URL: https://doi.org/10.1038/ng.2764.

[64] Muhammad Shoaib and Amna Basharat. “Ontology based knowledge representation and semantic profiling in personalized semantic social networking framework”. In: *2010 3rd International Conference on Computer Science and Information Technology*. IEEE, 2010. DOI: 10.1109/iccsit.2010.5564449. URL: https://doi.org/10.1109/iccsit.2010.5564449.

[65] Francois Scharffe Mohamed Ben Ellefi Zohra Bellahsene and Konstantin Todorov. “Towards Semantic Dataset Profiling”. In: 2014.

[66] Marcos Martínez-Romero et al. “NCBO Ontology Recommender 2.0: an enhanced approach for biomedical ontology recommendation”. In: *Journal of Biomedical Semantics* 8.1 (2017). DOI: 10.1186/s13326-017-0128-y. URL: https://doi.org/10.1186/s13326-017-0128-y.

[67] Barry Smith et al. “The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration”. In: *Nature Biotechnology* 25.11 (2007), pp. 1251–1255. DOI: 10.1038/nbt1346. URL: https://doi.org/10.1038/nbt1346.

[68] Richard G Côté et al. “The Ontology Lookup Service, a lightweight cross-platform tool for controlled vocabulary queries”. In: *BMC bioinformatics* 7.1 (2006), p. 97.

[69] Pierre-Yves Vandenbussche et al. “Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web”. In: *Semantic Web* 8.3 (2016). Ed. by Michel DumontierEditor, 437–452. ISSN: 2210-4968. DOI: 10.3233/SW-160213. URL: http://doi.org/10.3233/SW-160213.

[70] Marcos Martínez-Romero et al. “BiOSS: A system for biomedical ontology selection”. In: *Computer methods and programs in biomedicine* 114.1 (2014), pp. 125–140.

[71] N. F. Noy et al. “BioPortal: ontologies and integrated data resources at the click of a mouse”. In: *Nucleic Acids Research* 37.Web Server (2009), W170–W173. DOI: 10.1093/nar/gkp440. URL: https://doi.org/10.1093/nar/gkp440.

[72] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. “PARIS: Probabilistic Alignment of Relations, Instances, and Schema”. In: *Proc. VLDB Endow.* 5.3 (Nov. 2011), pp. 157–168. ISSN: 2150-8097. DOI: 10.14778/2078331.2078332. URL: http://dx.doi.org/10.14778/2078331.2078332.

[73] Hong-Hai Do and Erhard Rahm. “COMA—a system for flexible combination of schema matching approaches”. In: *VLDB’02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier. 2002, pp. 610–621.

Abbreviations

$HDFS$	Hadoop Distributed File System
$SM4DL$	Semantic Metadata for Data Lake
$R\&D$	Research and Development
SDD	Semantic Data Dictionary
SAP	Semantic Annotation Platform