Extensible Metadata Management Framework for Personal Data Lake

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Hassan H. Alrehamy

June 2018

Cardiff University

School of Computer Science & Informatics

Declaration

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed ……………………………………………… (candidate) Date ……………………………

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of …………………………(insert MCh, MD, MPhil, PhD etc, as appropriate)

Signed ……………………………………………… (candidate) Date ……………………………

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed ……………………………………………… (candidate) Date ……………………………

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ……………………………………………… (candidate) Date ……………………………

STATEMENT 4: PREVIOUSLY APPROVED BAR ON ACCESS

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loans **after expiry of a bar on access previously approved by the Academic Standards & Quality Committee.**

Signed ……………………………………………… (candidate) Date ……………………………

Abstract

Common Internet users today are inundated with a deluge of diverse data being generated and siloed in a variety of digital services, applications, and a growing body of personal computing devices as we enter the era of the Internet of Things. Alongside potential privacy compromises, users are facing increasing difficulties in managing their data and are losing control over it. There appears to be a de facto agreement in business and scientific fields that there is critical new value and interesting insight that can be attained by users from analysing their own data, if only it can be freed from its silos and combined with other data in meaningful ways. This thesis takes the point of view that users should have an easy-to-use modern personal data management solution that enables them to centralise and efficiently manage their data by themselves, under their full control, for their best interests, with minimum time and efforts. In that direction, we describe the basic architecture of a management solution that is designed based on solid theoretical foundations and state of the art big data technologies. This solution (called *Personal Data Lake - PDL*) collects the data of a user from a plurality of heterogeneous personal data sources and stores it into a highly-scalable schema-less storage repository. To simplify the user-experience of PDL, we propose a novel extensible metadata management framework (MMF) that: (i) annotates heterogeneous data with rich lineage and semantic metadata, (ii) exploits the garnered metadata for automating data management workflows in PDL – with extensive focus on *data integration*, and (iii) facilitates the use and reuse of the stored data for various purposes by querying it on the metadata level either directly by the user or through third party personal analytics services.

We first show how the proposed MMF is positioned in PDL architecture, and then describe its principal components. Specifically, we introduce a simple yet effective lineage manager for tracking the provenance of personal data in PDL. We then introduce an ontology-based data integration component called *SemLinker* which comprises two new algorithms; the first concerns generating graph-based representations to express the native schemas of (semi) structured personal data, and the second algorithm metamodels the extracted representations to a common extensible ontology. SemLinker outputs are utilised by MMF to generate user-tailored unified views that are optimised for querying heterogeneous personal data through low-level SPARQL or high-level SQL-like queries. Next, we introduce an unsupervised automatic keyphrase extraction algorithm called *SemCluster* that specialises in extracting thematically important keyphrases from unstructured data, and associating each keyphrase with ontological information drawn from an extensible WordNet-based ontology. SemCluster outputs serve as semantic metadata and are utilised by MMF to annotate unstructured contents in PDL, thus enabling various management functionalities such as relationship discovery and semantic search. Finally, we describe how MMF can be utilised to perform holistic integration of personal data and jointly querying it in native representations.

Acknowledgement

First and foremost, I am immeasurably indebted to my supervisor **Dr. Coral Walker**, for her fundamental role in my doctoral research. Dr. Walker provided me with every bit of patient guidance, encouragement, and advice that I needed over the course of my Ph.D. journey. She gave me the freedom to do whatever I wanted, whilst continued to actively care about my work and promptly respond to my questions and enquiries. In addition to our scientific collaboration, I immensely value the personal rapport that Dr. Walker and I have forged over the past four years. I am quite sure it is impossible to find a better supervisor.

I would like to express my profound sense of reverence and sincere appreciation to many parties in Iraq and United Kingdom. My true gratitude goes to **Babylon University** and **Iraqi Cultural Attaché** for offering me the opportunity to pursue my Ph.D. with continuous funding, support, and encouragement. Equal gratitude goes to all the staff members (both teaching and non-teaching) in the **School of Computer Science & Informatics** at **Cardiff University**, for their constant presence and the positive learning atmosphere without which this thesis would not exist. I would also like to express my heartfelt thanks to the anonymous reviewers of my publications for the donation of their expertise and effort in providing valuable comments and suggestions. My special thanks go to **UKCI17's organising committee** for offering me a wonderful opportunity to communicate an important part of my thesis, and for selecting *SemCluster* as a high-quality conference paper.

Ali Ibn Abi Talib once said: "*There are two kinds of people; those who seek but cannot find, and those who found but still want more*". While giant service providers continue to develop cutting-edge technologies to derive more value from our personal data, we – the common Internet users – are left behind straggling to seek but not always find the right information among the large volumes of our fragmented personal data. I dedicate this work to us.

Contents

List of Publications

The work introduced in this thesis has been disseminated in the following publications:

- Alrehamy H, Walker C. SemLinker: automating big data integration for casual users. Journal of Big Data, Springer. 2018;5(1):14.
- Alrehamy H, Walker C. Exploiting extensible background knowledge for clustering-based automatic keyphrase extraction. Soft Computing, Springer. 2018;22(21):16.
- Alrehamy H, Walker C. Personal data lake with data gravity pull. In: IEEE Fifth International Conference on Big Data and Cloud Computing (BDCloud), IEEE;2015. p.160-167.
- Alrehamy H, Walker C. SemCluster: unsupervised automatic keyphrase extraction using affinity propagation. In: Advances in Computational Intelligence Systems. UKCI 2017, Springer;2017. p.222-235.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

3Vs Volume, Variety, Velocity

ACID Database Atomicity, Consistency, Isolation, Durability

API Application Programming Interface

AKE Automatic Keyphrase Extraction

ASA Automatic Semantic Annotation

BDI Big Data Integration

CRUD Create, Read, Update Delete operations

DCMI Dublin Core Metadata Initiative vocabulary

DSAPI Data Source API

DSRM Design Science Research Methodology

DW Data Warehouse

ELT Extract, Load, Transform Paradigm

ESA Explicit Semantic Analysis

ETL Extract, Transform, Load Paradigm

GUI Graphical User Interface

HAR Human Activity Recognition

HDL Hadoop-based Data Lake

IAC Ingestion Agents Container

IDF Inverse Document Frequency

IE Information Extraction

IR Information Retrieval

LDA Latent Dirichlet Allocation

LSA Latent Semantic Analysis

ML Machine Learning

MMF Metadata Management Framework

NER Named Entity Recognition

NLP Natural Language Processing

OWL Web Ontology Language

PDL Personal Data Lake

PDLSF Personal Data Lake Serialisation Format

PIM Personal Information Management

QoS Quality of Service

RDF Resource Description Framework

RDFS RDF schema vocabulary

REST Representational State Transfer

TF Term Frequency

VTSA Virtual Transformation of Source Attribute

WordNet Lexical Database WordNet

WSD Word Sense Disambiguation

Chapter 1 Introduction

1.1 Background

Before the Internet era, people had their data kept in personal computers, stored on local hard drives, and in a few network-shared directories. With the Internet boom and the growing use of early online digital services like emailing and blogging, common Internet users started to have partial amounts of their data hosted in autonomous machines on the web (e.g. mail servers, web servers, etc.), and managing this data was relatively an easy task. In the years that followed, people living and working environments became greatly enriched with affordable personal computing devices and various kinds of digital services that are tailored to the user needs and preferences in different contexts; social media, online banking, personal and collaborative communications, e-commerce, cloud, to name only a few examples. This technology evolution upsurge in daily life and social organisation has made it more convenient for common users to transform the traditional way of performing daily activities into simple interactions with digital services. All of a sudden, significant amounts of users' data are distributed everywhere [1], including: the documents they create (presentations, spreadsheets, publications), the messages they exchange (SMS, IM, Email), the multimedia they share (social posts, photos, videos, music), the financial transactions they make (product purchases, billing invoices, ticket bookings), and so forth, much of this data is usually generated and stored using third party digital services offered by distinct entities in the digital service sector, which are generally called *service providers*. In addition to being distributed, the data that users generate and keep using multiple personal devices and different digital services became growing rapidly over time in terms of *volume* and *variety*. Such growth is further accelerated by the typical user behaviour of actively seeking out and consuming new data from private and public sources, and hoarding the data which the users may consider *personal* for many purported reasons [2,3,4]. In fact, a minute's reflection on modern Information seeking and foraging studies will reveal that common users tend to store large amounts of personal data for future use and reuse [1,5,6]. As a result, those users who choose to manage their data and take advantages of it, must do so in parallel across different services and applications [52], and should have sufficient knowledge and management resources to handle large amounts of diversified personal data.

Recently, it was revealed that managing personal data is becoming an increasingly difficult task for common users as the economic value of this kind of data in the digital industry is amplifying [8], and to the extent that it is often branded as "digital oil" [9]. An important question promptly arises here is that: "*how the management challenges of personal data are related to its value?*". In 2006, an influential study [11] reported that the personal data collected from users through multiple third-party services was enough to recognise social patterns, infer relationships, identify important geolocations, and model organisational rhythms. Soon after, it was clear that third party services, with their capability of gathering intimate and contextual information from users, entail immense increment in the production of personal data, and this kind of data is exponentially growing in depth and breadth forming a new era of empirical analysis with potential opportunities for businesses in the digital industry. Realizing this trend, dominant service providers began promoting data-incentive services (e.g. Google Drive, Facebook Creators, Apple iCloud, Amazon Alexa) that are deeply integrated with users' personal devices to expand providers' reach into the private storage spaces of users [11], and to migrate various data collections that are individually managed by the user to "central" data-housing repositories called *service platforms* [12]. This paradigm of data centralisation, and backed by the economic convenience of data transmission, storage capacity, and computational power, enables service providers to amass unprecedented amounts of personal data collected from large number of users, and analyse them using state of the art data mining and machine learning tools to glean valuable insights that magnify understanding of users' social behaviours and personality patterns, thus deriving revenues through utilising these insights to optimise business intelligence and consumer analytics models, or by licensing them to other third parties.

While the ultimate implications of the centralisation paradigm on common users are not fully examined yet [11], one promptly apparent consequence is that; users are increasingly losing control over their data [12] and are unable to take advantages of it. The main reasons for this limitation are personal data *fragmentation* and *isolation*.

1.2 Personal data problems

In a sense, personal data represents a comprehensive “black box” that records its owner's life in varying degrees of detail across different contexts [13]. By organising this box, the owner can infer knowledge and obtain valuable insights about her life and daily activities. As with all other data kinds, personal data may have certain characteristics, which in turn may impose multiple challenges on access, management, and usage, especially if we expect personal data to be multi-modelled, error-prone, and with severe fragmentation and isolation, which must be dealt with before personal data can be ready to be taken advantage of. If we properly understand the main problems of personal data, then we can devise appropriate solutions to overcome them and unlock the potential value of data through personal data management which, in short, concerns data collection, storage, organisation, maintenance, and usage [1]. Personal data management offers benefits to content-based search, contextual information retrieval, reporting and summarization. The opportunities that efficiently *managed* personal data may offer are based on the observation that a black box of the user’s life provides a detailed context about the user (e.g., who is the user, where is her location, where she has been recently, what is she doing now, and with whom, etc.) which can be readily leveraged to design useful tools for activating value creation. To illustrate this observation, consider the following example; by processing the user’s financial data (e.g. purchase transactions history), geospatial data (e.g. GPS data), and by assuming the user’s current location is near a market which she frequently shops in, then a sophisticated context-aware application would process such data and automatically infers that she is at risk of going over credit limit in the next purchase, thus it can intervene either by warning the user or by suggesting an alternative payment methods to avoid the overdraft fees imposed by the credit card issuer. Many context-aware applications have been proposed in the past [14,15,16], but their inability to uniformly access and query user’s personal data as an organised black box, has impeded the progress of these applications

[15]. A detailed discussion about contextual applications in the personal domain is covered in [13].

The first step in studying personal data problems is to understand how a data piece (datum) can be distinguished as *personal*, and in which *sense*. Unfortunately, there exists no agreed-upon technical definition of personal data, this is because it has evolved so quickly and disorderly that such a universally accepted statement denoting its exact meaning not yet exists, instead there are multiple statements in the field of data management that try to capture its traits in specific and restricted contexts. One practical way to conceive the abstract nature of personal data is through its legal definition by international data protection laws, since governmental articulations in general are readily understood and widely accepted. The *EPDP Directive*[1] defines personal data as follows:

> "*Personal data shall mean any information relating to an identified or identifiable natural person ('data subject'); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.*".
>
> (EPDP, 95/46/EC Art. 2/A)

The definition emphasizes that personal data is any information in the world, whether physical or digital, that can be linked to a person in some way. There are different ways through which data are linkable; "full name" is obviously personal data that is directly linked to a person. Linking might be obvious by combining autonomous identification elements, such as physical characteristics, pseudonyms, banking details, occupations, addresses, etc. There also exist many types of data that sound un-linkable to a given person at the first glance, but a closer look will promptly reveal important linking. Consider the following dataset examples; Underground service information, postcode dictionary, and hotel guest reviews in London. These datasets are decisively unrelated to a person living

[1] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with respect to the processing of personal data and its free movement. URL: http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:31995L0046

in Washington. However, once that person decides to visit London, linking to the datasets becomes obvious and compelling.

The legal definition of personal data is deliberately made broad for legislative purposes. Consequently, we need to focus on an alternative but relative conception from a technical point of view. Henceforth, we understand personal data as "the data over which a person has some interest or control, currently or in the future, in order to negotiate the person's environment and organise life activities". Such conception is much more in tune with an intuitive understanding of what data means to a person in the course of daily activities, and as one would expect, it might be in compliance, to a great extent, with the legal definition of the data from which the person is identifiable, but may also comprise data of which the person owns, but from which the person cannot be identified at all – as illustrated in the former example. With this technical but informal abstraction, personal data may span a vast range of domains, and therefore scoping all its types into a widely agreed upon convincing definition is on one hand an extremely difficult task, but on the other hand is an enabler of scientific progression as Ronda et. al. [17] suggest; the level of consensus shown by a scientific community on a definition of a concept can be used as a measure of the progress of a discipline.

Researchers in personal data management field have suggested different statements to discern a rough boundary between what data is personal and what is not [1,11,18]. The work introduced in [1] is an important progress in this regard as it builds upon an intensive review of (personal) information literature. Jones [1] defines six senses in which data may be recognised as personal. Table 1.1 presents each sense, its relatedness (linking) to the user, the production type of the data falls under that sense, and multiple general examples. Basically, there are two types of personal data production; *active* and *passive*. Active data production refers to the data that is purposefully self-generated to negotiate one's environment and to perform daily activities, whereas passive data production refers to the data that is generated on the one's behalf through by-product service interactions. Text messages, photos, and blogs are examples of actively produced data, whereas IoT sensing data, search engine logs, and credit history are examples of passively produced data.

It is noteworthy to mention that personal data senses are inclusive, and by combination they make up the whole personal data of a user. It should also be borne in mind that the notion of personal data is the usual term in Europe whereas it is known as "personal information" or "personally identifiable information" in the United States. Likewise, the terms "personal data management" and "personal information management" are identical [18]. Therefore these synonymous terms are interchangeably used throughout this thesis.

Table 1.1 – Personal data senses

Sense	Production	Description
Owned/Controlled by user	Active/Passive	Data stored on user's personal devices, cloud spaces, online accounts, and any other service platforms, like documents, files, multimedia.
About user	Active/Passive	Data factorizing the user's identity; such as official paperwork, medical records, search engine queries, product orders, and credit history.
Directed to user	Passive	Data communicated *to* the user; such as phone calls, emails, instant messages, social and collaborative invitations, promotion ads, and vouchers.
Sent by user	Active	Data communicated *by* the user; such as phone calls, emails, private (instant) messages, blogs, social media posts, GPS check-ins, and research publications.
Experienced by user	Passive	Data already utilised by the user; such as website pages, books in the library, on-demand streaming content like online courses, TV, and Radio shows.
Relevant to user	Passive	Data interesting to the user for future use; such as the next hotel booking, vacation destination, job to apply for, and house to buy.

1.2.1 The 3Vs nature

As our understanding now is in tune with the notion of personal data, its possible senses, and its basic terminology, our next step is to examine the nature of personal data from *Big Data* perspective. Such examination can take place by comparing the main characteristics of big data with those exhibited by the total data of a common user. The benefit here is: if personal data conforms to big data characteristics, then there are certain advantages that big data management technologies could bring to enable both effective personal data management and analytics on a personal rather than on the enterprise level (i.e. personal analytics [19] and personal informatics [20]).

Big data is a ubiquitous term to describe any voluminous amount of structured, semi-structured and unstructured data that is growing and moving so fast that it becomes extremely difficult to store, manage, or process using conventional data management technologies. In most enterprise scenarios, big data is believed to have three main characteristics [21]: *Volume*, *Variety*, *Velocity*, (3Vs). Volume refers to the amount of data, variety refers to the heterogeneous sources from which the data is produced and gathered, and velocity refers to the unpredicted shifting patterns and changes in the data over time. There is a surging interest in big data by many parties in the digital industry and academia, this is because of the immense value and unlimited opportunities that can be unlocked by mining and cross-referencing information from diverse data sources. Big data is also becoming a daunting problem due to management challenges resulting from the continuous expansion of all its aspects over time [22], rather than just the sheer amount of data to be managed. The term *Big Data Analytics* [23] refers to the quintessential technologies that the user must possess to harness and utilise the data at high 3Vs scale for creating value. Not only does this make big data management and storage vastly different from normal or structured data that most people are accustomed to handling, but it also means the users now require powerful integrated solutions for making big data scenarios usable and applicable. Currently, there exist many enterprise-level big data management systems and tools that can be adopted to collect, integrate, manage, fuse, and analyse big data efficiently and effectively. Before discussing the benefits of the big data ecosystem to the personal data management, we initially need to run an end-to-end comparison between big data and personal data in terms of the 3Vs model.

One characteristic which makes personal data a big data application and poses both challenges and opportunities in its management is the *variety* of personal data sources, which may include: social, entertainment, work-related, consumption-related, freely and by-license accessible, data services, furthermore, they may also include by-product online tracing, physiological monitoring, and pervasive monitoring (IoT) services. Personal data sources are monumentally varied, and the data generated from them is so diverse that we simply cannot enumerate its types. To take a quick look, consider a typical smartphone that is often equipped with the following constituents (services):

- **GSM, LTE, Wi-Fi, and Bluetooth**. The data generated from these constituents include: cellular network tracking, LTE roaming information, home and work network-shared directories, printers, and scanners, Wi-Fi location data, nearby Bluetooth devices, Wi-Fi networks and signal strengths, Internet usage metering, power consumption information.
- **Always-on Radio GPS**. The data generated from this constituent include geographic positioning logging, tracking, and navigation.
- **Accelerometer, Gyroscope Barometer, Thermometer, Magnetometer, Infrared.** The data generated from these constituents include real-time gyroscopic orientation tracking streams, real-time accelerometer movement tracking streams, basic and advance physiological monitoring data like temperature and galvanic skin readings, heart rate tracking, and environmental sensory data.
- **Video camera and microphone** (or several). The data generated from these constituents are multimedia files, live streaming, visual and audio communications.
- **Internet-connected services** for entertainment, communication, and to help users manage their daily lives.

Common users carry this bundle of services with them wherever they go, and generate data almost all day. To derive value from these services, the common user must be able to merge and combine various data collected from such services to form a holistic consolidated view through which variety is normalised or eliminated.

The second characteristic we examine is *volume*. Personal data sources constantly generate structured, semistructured, and unstructured data on a per-person basis. the fate of these data may be one of the following: (i) actively stored on local hard drives, (ii) actively/ passively hosted in remote storage locations, i.e. online accounts and third-party service platforms, or (iii) disposed due to lack of applicability, i.e. the common user cannot derive any benefit from the data, neither currently nor in future, because of insufficient knowledge and resources to manage it. For example, nowadays only small groups of data enthusiasts in the society (e.g. life-loggers [13], self-quantifiers [19]) are recording their physiological sensory data and deriving insights out of it, whereas the remaining population ironically turn off their sensory services to save battery charge. Although a one's personal data on average has not yet reached huge volume (i.e. below petabytes), we believe that such volume is constantly scaling up, and at some point in the foreseen future will conform to the big data volume aspect. This observation is predicated on multiple recent Internet usage reports; here are some statistics: a common user, on average, has 3.64 personal devices connected to the Internet [24], interacts with 30 third-party services monthly using these devices [25], uses with 9 services on daily basis to generate personal data [26], has 100 online personal accounts in third-party service platforms [26], uses 7 of them regularly to exchange personal data with others [25], and these statistics are doubling every 18 months.

The final characteristic of the 3Vs model is *velocity*. Technically, common users cannot control the physical details of their personal data being passively generated by (third-party) services, consequently, they cannot veto any subtle shifting changes in its patterns and representations (i.e. schema, structure, and semantics), and their only available choice is adapting to any emerging changes propagated by a service of interest.

Based on the above comparison, we can reasonably regard the total amount of one's personal data as a scaled-down version of big data, that is, the version of a single person in contrast to the large body of an enterprise organisation. During this comparison, we observe the characteristics of personal data are problematic. A modern personal data management solution must account for the 3Vs model, but it should extensively focus on variety with effective accountability. This emphasis has been delineated since the early age of big data, for instance, a specialist survey based on interviews conducted with 20 major big data firms in 2012 concluded the following [28]: "*The survey indicates companies are focused on the variety of data, not its volume. The most important goal and potential reward of Big Data initiatives is the ability to analyse diverse data sources and new data types, not managing very large datasets*".

1.2.2 Fragmentation

For a common user, the daily interaction with an arbitrary digital service entails generating a certain amount of personal data every day which exponentially grows over time, resulting in a large and complex data collection (or dataset) that is stored and managed by the provider of that service. For example, consider a third party email service for regular email messages exchange, Fisher et. al.[5] report that common users, on average, end up having 28000 messages archived within their email accounts and stored on the platform of the email service provider. There are many reasons that motivate (or even compel) users to delegate the responsibility of storing and managing their personal data to third party service providers, mainly including:

- The provided service requires intermediate data exchange medium. For example, a social media service provider like Twitter retains the user's data (tweets) into a central platform in order to share it with the user's friends and followers.
- The provided service is fundamentally developed to assimilate the personal data of its users. For example, a cloud-based storage service emphasizes migrating user's data (documents and files) to a central service platform in order to provide support functionalities like backup and recovery, location-independent accessibility, and data synchronization between multiple personal devices.
- Users do not have the sufficient knowledge or resources to self-host and manage large amounts of their personal data.

As users expand the set of their favourite services over time, their personal data becomes no longer co-existing into single storage space but rather increasingly scattered across various applications running on personal devices, as well as autonomous service platforms exclusively controlled by the third parties – this phenomenon is called personal data *fragmentation*. Fragmentation is very problematic on the long term [52] as it hinders users' ability to understand *what* personal data they have, *where* it is stored, and *how* it can be accessed and utilised to meet personal information needs. Due to fragmentation, users may not be able to list all service platforms hosting their personal data, furthermore, essential data management functionalities may become increasingly difficult undertakings without simultaneous accessibility to all fragmented personal data, regardless of its hosting platforms, from a single access point. The following questions are simple examples to illustrate the effect of fragmentation on common users:

- *I know I have this message, but was it received via my personal Email, work Email, Facebook, Telegram, or Viber?*
- *I know I have this photo, but was it posted on Facebook, Twitter, Instagram, or saved on my hard drive?*
- *I know I have this document but was it saved as an email attachment, or in Dropbox, Google Drive, OneDrive, or on my hard drive?*

Unfortunately, many of the services mentioned above may act as data silos since they do not offer content search functionality [29], instead the user may have to browse the personal data hosted in the platform of each service using the provided GUIs in order to find specific data entities (e.g., messages, photos). Nevertheless, without a software agent that simultaneously accesses these platforms and performs global search over them, then finding a specific data entity may be as hard as finding a needle in a haystack; the user should list all the platforms that are likely to store the required data entity and then goes platform-by-platform trying to find it.

An intuitive solution to the fragmentation problem is to reverse or at least reduce the paradigm of personal data centralisation by empowering common users with a personal data management solution that allows to retrospectively harvest (i.e. extract and collect) fragmented personal data from various service platforms, and any other data sources for the matter, and accumulating it into a central storage space that is fully controlled by the user [30,31]. With the rise of the REST architectural style for webservices [32], providers now have a flexible mechanism to facilitate retrospective sharing of data with its owners; that is RESTful application programming interfaces (API). The API offered by a service provider is often a well-documented publicly accessible endpoint in the platform of the service and it enables flexible data collection by authenticated users over web protocols like HTTP either manually or through software agents. API-based data collection has been utilised in many quantitative research projects that involve analysing huge amounts of big data collected from service platforms to develop predictive models [33,34]. In qualitative research, APIs can be exploited to harvest unstructured text archives of communication patterns on social media for close-up analysis [35]. APIs have been also proposed as personal data collection trajectories for common users [36]. A user may deploy one or more agents that connect to the APIs of different service platforms of interest, to retrieve personal data from each platform and accumulate it into a central storage space that is controlled by the user. This workflow presents a practical way for user-centric data centralisation as opposite to the service provider-centric centralisation paradigm described earlier. However, API-based automated data collection is only part of the solution to the fragmentation problem. There are many service providers that do not offer API-based data retrieval, offer only partial retrieval, or act as data silios that permit retrieval strictly through ad hoc service-specific data collection adapters (agents). The reasons service providers lock users' data inside their platforms or complicate its programmatic access and retrieval may be technical, or even legal, but mostly economic, as they do not want to relinquish the economic advantage of holding users' personal data. An effective solution to the fragmentation problem should involve collecting personal data using traditional and non-traditional means.

1.2.3 Isolation

Another latent problem arises from users' reliance on a variety of third-party services is *isolation*. Basically, the delegation of personal data storage responsibility to a service provider necessitates relinquishing control over *how* the data is represented in the platform of that provider. Service providers may store and handle incoming and outgoing data using different models and representations[2] that are essential for optimising their internal data processing and management workflows. If all providers adopted a single unified data representation for all the data they are dealing with, then mashing up and utilising fragmented personal data coming from scores of platforms and sources would be achieved in straightforward manner, nonetheless performing management functionalities like global search and integrated analysis could be relatively easy, for instance, to answer the question "*how much milk I consume monthly*?", the user may formulate a "simple" query targeting

[2] A data entity may have certain underlying physical details including: physical format, native schema (if any), and structure (i.e., structured, semistructured, or unstructured). Throughout this thesis, we use the abridged term "representation" to refer to the combination of these details.

historical purchase data (e.g. receipts) to directly count all the records which contain purchased milk quantities. It is widely accepted fact that convincing the entire world to use singular data representation is fruitless effort since the degree of diversity and continued independent evolution of service platforms and data sources practically guarantees that data representation singularity will never happen [37], in fact, this is the main reason behind inventing the Semantic Web [38]. Consequently, accumulating collections of fragmented personal data with different representations into single storage location imparts isolation; the stored contents may be regarded as disparate datasets and information icelands with heterogeneities on the syntactic, semantic, and structural levels, which prevent their use in any possible scenario.

Isolation impedes users' ability to uniformly access fragmented personal data, from a single access point, to perform management functionalities. To tackle this impediment, the user must perform a kind of *integration*; a data management task seeks to provide unified view over the data collected from multiple sources so that they may look as if coming from a single well-managed source [39]. Providing unified views in practice involves designing a global (mediated) representation that captures the user's data requirements, followed by a manual construction of relationships, which are called *mappings* [39], between the native representations of the data under integration and the global representation. This overall approach can provide high-quality integrations but at a high cost and tends to be unsuitable in areas where data is coming from a plurality of frequently changing sources, and where users are willing to tolerate a less than perfect integration.

The notion of integration is often viewed as a spectrum, at one end all fragmented data conform to a single agreed-upon representation, at the other end all the native presentations of fragmented data are heterogeneous as they are independently designed and maintained by autonomous sources, and in between these ends lies various integration solutions and approaches. Such spectrum may indicate the degree to which management and analysis functionalities can be performed across fragmented personal data, with higher degrees of integration providing more sophisticated functionalities. Integration is known to be an extremely complex task that incurs tedious supervision efforts by skilled users for *schema matching, schema mapping, semantic and structural reconciliation, and data quality* [40]. Not all common users are data specialists, this necessitates the need for an adaptive data management solution wherein "automatic" integration workflows can be invoked on isolated personal data collected from autonomous sources and exist with different representations that the user knows very little about, but at the same time, demands high degree of data homogeneity in order to mush up, utilise, and take full advantages of it.

1.3 A personal data management solution

In today's digital age, common Internet users are immersed with an overwhelming number of third party services and other data sources, tempted by service providers to generate more and more data and relinquish control over it in exchange of useful and mostly "free" functionalities. Accordingly, common users are unable to gain control over their personal data, finding the right information within it, or readily deriving any value out of it [12,23,36]. Nevertheless, because users cannot determine *who* is accessing their data, *when*, *how*, and *why* [11,12], they are incapable of preventing privacy-abandoning third parties from cross-referencing their fragmented intimate information using malicious methods [11,41] (e.g. linking attacks[3]). In this thesis, we take the point of view that the cycle of organisational activities for personal data management [1] can provide the necessary means to assume back the control over one's personal data and to facilitate taking full advantages of one's black box [13], not only to empower oneself with valuable insights for enhancing life and daily activities, but also to elevate, to a certain extent, the privacy of one's data. To this end, we describe the design and the core components of a big personal data management platform called *personal data lake* (PDL) [44] that supports common users in managing the sum of their personal data. PDL is based on the *data lake* concept [42]; a widely used data management technology in the big data ecosystem [43]. It enables the user to bring together and enrich fragmented personal data in raw forms, remedying its main inherent problems, and reducing the efforts required to smooth its recurrent use in various scenarios. PDL has complete access to user's personal data pouring anywhere in the cyber world, whether hosted by third party service platforms, personal computing devices, pervasive electronics, or any current or future active/passive data

[3] Linking attack may also refer to the interdisciplinary term *jigsaw identification.*

sources that would produce data which can be linked to the user in any of the senses listed in Table 1.1. PDL fully supports automatic and manual retrospective collection of personal data from local or remote sources in batched, streaming, or near-streaming modes, as per the requirements and needs of the user. Such lake permanently stores and archives personal data into a highly-scalable central repository of unified schema-less data storage, thus offering a current state of the art solution to the fragmentation problem discussed in section 1.2.2. In the heart of PDL lies the most important architectural part, which is *metadata management framework* (MMF). MMF is the de facto management solution that orchestrates personal data storage, organisation, usage, maintenance, as well as privacy. MMF annotates personal data with machine-readable metadata to describe its lineage, important facets, underlying attributes, and relationships with other data within PDL. The final product of MMF is rich contextual metadata that are utilised for:

- Taming the heterogeneities of fragmented personal data and eliminating any inherent isolations that stem from its centralisation into PDL. Thus, relieving the user from the tedious efforts required for data organisation and maintenance.
- Generating comprehensive consolidated views over the centralised data that are defined by a user-tailored formalism in term of a coherent Metamodel. Thus enabling simultaneous and uniform access to a wide variety of personal data and conveniently querying it on the view (metadata) level.
- Enabling the user to experience new functionalities with personal data which once were prevented by fragmentation and isolation like semantic search, relationships discovery and data preparation for personal analytics and informatics.
- Allowing the user to control the accessibility of the stored data based on privacy settings with varying levels of actionable details.

The user has complete control over PDL and its underlying MMF, and retains the ability to retrieve its metadata, extend it with new (meta)models, modify the views materialized through it, and share it and/or its views with any third-party data consumer for gaining benefits as perceived by the user and without relying on external tools.

1.4 Thesis aims and objectives

Treating personal data as little big data, and dealing with its inherent 3Vs model lies at the heart of the big data problem. The personal data of an average common user is usually voluminous and exists in a plurality of autonomous silos with incompatible and rapidly changing representations, while more new silos and representations are expected to appear in the future due to the continuous emergence of renovated third-party services and new personal data sources (e.g. IoT sensors). It is fair to say that the 3Vs of personal data contribute to the 3Vs of global big data, and solving the problem at the personal scale can lead to a possible solution on the global scale. The main aim of this thesis is to exploit the data lake as a flat space for collecting and storing large amounts of diverse personal data without compromising its native representations. A data lake by definition is a complex enterprise solution that is intended for use by highly skilled data scientists and not common users. Therefore, the objective of the work presented in this thesis is to materialize the necessary set of capabilities for automating the lake's data management workflow, thus giving common users the incentive to take the responsibility of collecting, organising, and analysing their personal data by their own, within a highly supportive environment, under their full control, with minimum efforts and very limited technical skills. Accordingly, we focus on proposing methods and algorithms for isolating the user from the complex details of the downstream management tasks and activities in PDL, particularly:

- Metadata extraction and annotation.
- Schema matching and mapping.
- Semantic and structural heterogeneity reconciliation.
- Schema evolution handling.
- Holistic data integration.

The proposed methods and algorithms heavily rely on the metadata technology, and they inclusively form an extensible metadata management framework that enables the PDL user to efficiently integrate, query, and analyse, with minimal time and effort, large amounts of heterogeneous raw data centralised in PDL whilst preserving their native schemas, formats, and structures intact.

1.5 Research methodology

Design science is an outcome based research methodology that offers specific guidelines for designing and evaluating purposeful innovative artefacts to solve a special problem in a particular domain [45]. An outcome artefact is perceived to be knowledge containing, this knowledge ranges from novel methods, algorithms, and tools, to assumptions about the context wherein the artefact is intended to function. To form a novel contribution, the research outcome must either solve a problem which is not solved yet or provide a better solution for it. In this thesis, the design science research methodology (DSRM) is regarded as an optimal methodology to follow for fulfilling the research aims described in section 1.4. Accordingly, we adopt the DSRM introduced by Peffers et. al. [45]. Figure 1.1 depicts the steps that compose our adopted DSRM, and each step is explained and related to the thesis chapters as follows:

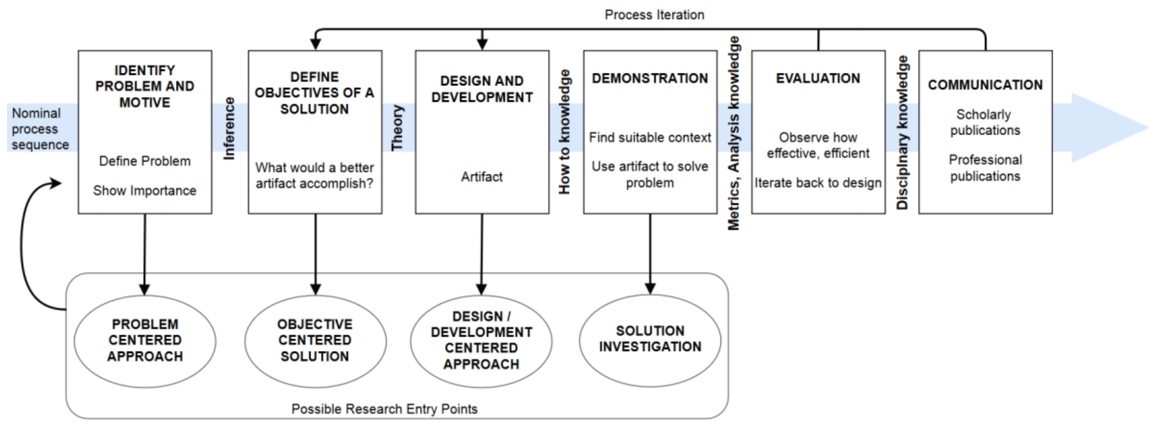

Process Iteration
Nominal process sequence
IDENTIFY PROBLEM AND MOTIVE
Define Problem
Show Importance
Inference
DEFINE OBJECTIVES OF A SOLUTION
What would a better artifact accomplish?
Theory
DESIGN AND DEVELOPMENT
Artifact
How to knowledge
DEMONSTRATION
Find suitable context
Use artifact to solve problem
Metrics, Analysis knowledge
EVALUATION
Observe how effective, efficient
Iterate back to design
Disciplinary knowledge
COMMUNICATION
Scholarly publications
Professional publications
PROBLEM CENTERED APPROACH
OBJECTIVE CENTERED SOLUTION
DESIGN / DEVELOPMENT CENTERED APPROACH
SOLUTION INVESTIGATION
Possible Research Entry Points

Figure 1.1– The adopted design science research methodology process model

1. Problem Identification and Motivation: This step involves critical thinking of the research problem and modelling strategies to justify the value of designing a solution to solve that problem. In the first phase of this step, our aim is to identify the gaps in the related literature. In the second phase, the research aims are carefully articulated as presented in section 1.4. The third phase requires the selection of the data sources and the tools for developing the solution. The last step involves research planning by dividing the research problem into two sub-problems, namely:

- Structured and semistructured personal data management.
- Unstructured personal data management.

In last the phase we also identify the required means to solve each sub-problem.

2. Objectives of the Solution: This step requires knowledge of the state of the problem, its current solutions, and their efficacy. The problem definition in the previous step is used to propose the objectives of the solution. Our research problem is general in many fields, whose objective is the management of raw data scattered across a variety of data sources, and isolated due to severe heterogeneity in its native representations. However this problem in the context of our research exhibits a fundamental difference in that its solution has specific objectives that are necessary for the success of a bigger solution (i.e. PDL). These objectives are:

- The solution is data lake-oriented; by definition, it must preserve the native representations of the centralised data to sustain its potential future value, and should be efficiently tractable to the 3Vs model.
- The solution is user-centric; it must be automated to ease utilisation by unskilled common users, and, should be flexibly extensible to support customization according to user's changing needs and preferences over time.

3. Design and Development: This step aims to design and develop a solution for the defined problem. The details of this step are covered in chapters 3, 4, and 5. The design of the proposed framework and its integration in PDL architecture is covered in Chapter 3. The design of the component responsible for structured and semistructured personal data management is presented in Chapter 4, and the design of the component responsible for unstructured data management is presented In Chapter 5.

4. Demonstration: This step involves using the developed framework in suitable contexts. In this thesis, multiple showcases are discussed in Chapters 6 using a total of 16 real-world datasets. The chosen data is regarded as personal data of an imaginary person. To demonstrate the utility and robustness of the proposed framework and its components in solving real-world problems, five main demonstrations are introduced. The first involves integrating and querying heterogeneous sensory data collected from multiple sensors embedded in wearable devices. In the second, we simulate a scenario of making data-driven hotel and restaurant booking decisions in the city of London. In the third and fourth, we demonstrate how MMF automatically extracts keyphrases from free-text documents, and annotating them with ontological information. Finally, we present an experimental scenario to demonstrate the holistic integration capability of MMF over public art data collected from a museum by the user in efforts to create new useful knowledge through back-story information cross referencing.

5. Evaluation: This step observes and measures how well the proposed framework resembles an optimised solution to the defined problem. It involves assessing the effectiveness/efficiency of the components in the proposed framework compared to their current state of the art counterparts available in the relevant literature. In DSRM, once framework components are developed, researchers start a thorough testing process for each component. In this thesis, we righteously evaluate the main components of the proposed framework in Chapter 6. The evaluations aim to assess the efficiency of each component, and its effectiveness compared to the chosen state of the art approaches.

6. Communication: The main thesis contributions have been published as peer-reviewed scholarly publications. Four papers have been disseminated from the presented work: two are published in high-quality journals, and the other two are published in conferences relevant to our research's core topics. The papers are given in the list of publications section.

1.6 Thesis contributions

The main contributions of this thesis are as follows:

- We present a novel extensible MMF for handling personal data management and simplifying, to large extent, PDL user-experience for common individuals with no or limited data management skills.
- We introduce a basic lineage manager as the first principal component in MMF to track the provenance of personal data in PDL and maintain its privacy settings.

- We propose SemLinker, an ontology-based data integration system as the second principal component of MMF. SemLinker comprises two novel algorithms; the first constructs graph-based representations to express the native schemas of (semi) structured data collected from a variety of local and remote data sources. The second concerns partial metamodeling of such representations to an extensible general-purpose and widely-used ontology in big data applications.
- We propose a novel approach to handle schema evolutions that stem from data velocity, more precisely, the rapid and frequent changes that may emerge in the source schema and format of (semi)structured data collected from a data source.
- We introduce SemCluster, a novel domain-agnostic clustering-based algorithm for the task of unsupervised automatic keyphrase extraction. The proposed algorithm exploits extensible background knowledge for extracting thematically important keyphrases from input unstructured data, and associates each keyphrase with fine-grained semantics drawn from an extensible common ontology.
- We propose a SemCluster-based approach for automatically annotating unstructured textual data with rich semantic metadata, then we show the usefulness of this approach in facilitating functionalities like relationship discovery between, and semantic search over, heterogeneous unstructured contents.
- We show that by combining SemLinker and SemCluster metadata, MMF can offer attractive management functionalities over PDL stored data on metadata level without requiring any physical transformations, thus preserving the native representations of the data and facilitating its use and reuse in various informatics and analytics applications while maintaining its potential value longevity.

1.7 Thesis outline

In this thesis, we investigate and evaluate automated solutions to the problem of data management in data lakes. We present techniques that are related to three main topics: metadata management, data integration, and unsupervised keyphrase extraction. In particular, we examine new and exciting methods and algorithms in order to introduce an extensible metadata management framework for PDL, which can be generalised in the future for reusability in other data lake systems. The outline of this thesis is as follows:

Chapter 2 discusses the background and main foundations of the data lake concept. The chapter starts by illustrating the difference between ETL [46] and ELT [47] paradigms, then explores the feasibility of ELT in solving the problems of big personal elaborated in section 1.2. We then review the data lake literature and discuss the importance of metadata in managing raw data. The chapter explores various metadata management approaches and frameworks that have been proposed for different data lake systems and architectures. Finally, the general limitations of existing works as well as their fundamental drawbacks are discussed from the perspective of our research objectives.

Chapter 3 introduces an overview of the proposed MMF and its architectural integration in PDL. The chapter begins by discussing the empirical feasibility of Hadoop ecosystem for developing a big personal data management system. Then an overview of PDL's ELT workflow is introduced by means of two processing pipelines; storage and usage. To illustrate how MMF is managing these pipelines in an automated fashion, we describe each layer in PDL, including the rationale behind its design, its constituting components, and their relations with MMF components as pipelining inputs or outputs.

Chapter 4 presents SemLinker; our proposed solution to the problem of (semi)structured personal data management in PDL. In this chapter, we review a core process in data management: data integration, and discuss the main challenges and requirements that must be met to automate this process in data lakes and particularly PDL. The chapter then introduces SemLinker architecture, its layers, and their underlying components. We first describe SemLinker's underlying algorithms and then introduce a new automatic approach for handling raw data schema evolutions. Next, we discuss how SemLinker annotates ingested personal data with formal schema metadata, and how the output annotations are exploited for generating "partial" unified views which can be queried in a uniform way by common users and their gravity-enabled analytics applications.

Chapter 5 presents SemCluster; our proposed solution to the problem of unstructured personal data management in PDL. This chapter starts by elaborating a core process in unstructured data management; semantic annotation. The chapter then discusses keyphrase extraction as a potential solution for automating the semantic annotation of unstructured documents. The existing work in the keyphrase extraction literature is thoroughly reviewed in order to understand its research gaps and limitations. Next, we present SemCluster, a new algorithm for unsupervised automatic keyphrase extraction that exploits background information to identify thematically important phrases in input documents, and to annotate these phrases with fine-grained ontological information. SemCluster algorithm adopts an extensible ontology (WordNet [49]) as an internal knowledge source, which can be further extended by integrating extra knowledge sources (e.g. DBPedia [50], BabelNet [51]) for incorporating more background information to improve the extraction precision and the overall annotation performance. In this regard we introduce a new knowledge integration approach to materialize personalised extensibility based on the needs and preferences of the PDL user. Finally, we discuss how SemCluster is implemented as MMF component to manage unstructured textual data through annotation-based semantic representations.

Chapter 6 presents the empirical evaluations of the methods and algorithms proposed in this thesis. As per the adopted research methodology, we focus on evaluating the effectiveness and efficiency of MMF core components: SemLinker and SemCluster. We start by testing the effectiveness of SemLinker using various datasets. To explore the accuracy of the component's underlying algorithms, we first conduct an experiment of integrating 3 heterogeneous datasets and preparing them for utilisation by an analytics application plugged in PDL. We also run a second experiment for integrating 8 large heterogeneous datasets collected from different domains, with the aim of integrating them and generating a unified view that is directly queried to obtain insights for supporting a decision-making process regarding hotel and restaurant bookings. To evaluate the efficiency and simplicity of use, we compare the overall performance of SemLinker against another recent integration approach [52]. Next, we evaluate the effectiveness and efficiency of SemCluster. We start by validating the algorithm's effectiveness using two datasets that are popular in NLP literature. We compare the overall performance of SemCluster under optimised settings – in terms of Precision, Recall, and F-Measure – against multiple leading algorithms in the keyphrase extraction literature, namely, TextRank [53], ExpandRank [54], and KeyCluster [55]. We further evaluate the effect of background information on SemCluster's efficiency using different knowledgebase integration settings. Finally, we conduct a demonstrative experiment to show utility and effectiveness of the proposed MMF in bridging data over structural heterogeneity using 3 heterogeneous gallery datasets.

Chapter 7 presents the conclusions drawn from this research, and discusses the possible research directions which our future work might take.

Chapter 2 Literature Review

As part of the followed DSRM methodology, it is critical to understand the state of the data management problem in data lake [45]. Chapter 2 introduces a comprehensive review to establish the depth and breadth of the existing body of work in data lake literature. The presented discussions not only help in understanding the cost implications of adopting an existing metadata-based solution for PDL, but also to pave the way for introducing the proposed MMF and delineating how it advances on the current state of the art research. Though this chapter mainly focuses on exploring various lake-oriented metadata-based approaches and frameworks, additional reviews related to downstream management tasks are covered in Chapters 4 and 5. Sections 4.1 and 4.2 explore the research works related to (semi)structured data integration. Section 5.1, 5.2, and 5.3 review the literature of unstructured data management with extensive focus on automatic keyphrase extraction.

To maintain the consistency of our discussions, we compile a short terminology that we use throughout this thesis chapters, and as follows:

- “data entity” is a piece of raw data that is extracted and ingested from an external data source. Generally, this term may refer to an object, record, file, text blob, and so forth.
- “usage workload” is the amount of processing required to meet a particular information need given a collection of raw data in the form of datasets, individual data entities, or a mixture of both. Data querying, integrated analysis, reporting, and summarization, are general examples of usage workloads.
- “input source” is a private/public data source that actively or passively generates raw data entities which can be collected by a third party data collection agent. Input sources may range from social media, e-commerce applications, email clients, cloud services, to (ubiquitous) applications that run on personal devices.
- “data consumer” is a third-party agent that can run usage workloads on data entities stored with a data management system. Data consumers may be human or machine agents.

2.1 Data lake

Historically, data warehouse (DW) has long been considered by the digital industry as a unique data management solution for centralising raw data, remedying its isolations, and preparing it for querying and analysis to deliver accurate and timely information that supports decision making and business intelligence processes [46,56,57]. DW is a structured management solution that collocates fragmented data and makes it available for use to end users based on a data processing paradigm called ETL (*Extract-Transform-Load*) [46]. The first step in this paradigm (Extract) concerns pulling raw data from a set of heterogeneous input sources on a regular cadence. An intrinsic part of this step is to rigorously validate, using a predefined list of validation rules, whether the data extracted from each input source has the "expected" representation details. If any data fails validation, it will be either fully or partially rejected. The second step (*Transform*) concerns treating the heterogeneities of the collected data by passing them through a series of transformation rules, called *staging functions*, in order to structure it based on a predefined "unified" schema, before ETL can proceed to the next step. Transformation details may differ from one DW implementation to another but generally involves cleaning [58], consolidation [59], and modelling [60]. The last step (*Load*) involves arranging the transformed data into hierarchical groups, called *analysis dimensions*, and storing them into a central database. Beyond this point, the processed data becomes highly structured with enforced unified schema that can be directly utilised by usage workloads on different aggregate levels.

With the emergence of big data, ETL paradigm has been shown to pose too many inherent constraints on data storage and usage in a DW environment. First, ETL can process only the types of data that are specified in the analysis requirements during DW design time, whereas any other types are discarded [61,58]. Empirically, it is not always the case that the all the types of data required for analysis should to be known in advance. For example, the increasing use of services and applications by everyone produces large volumes of personal data that, at first level, cannot be linked with any analysis requirements as the potential wealth of information in the data is not preliminarily known, or not enough explored. In these cases, such types of data cannot be stored in DW, instead they are held elsewhere, thus, elevating fragmentation. Secondly, the monolithic set of transformations and their related integration routines applied on data inputs before the loading step may lead to two types of losses: losing the future value of data due to the physical modifications applied on its native representations, and losing parts of the actual data contents that do not meet the specified validation and staging criteria, even though such data might be of potential value in future analysis workloads. Fundamentally, ETL's imposed criteria allow examining only a predefined subset of the data attributes, therefore only pre-determined analysis questions can be answered in DW [42]. This limitation prevents the end users from working on datasets in their native representations; users may have their own ideas about how they want to use the stored data. As result, each user may need to individually examine a dataset before devising a target data model or engineering data transformation routines for performing a particular usage workload. In response to this stance, a new concept has emerged in the data management landscape: data lake (or *datahub*). DL is initially coined by James Dixon [42] as a theoretical methodology to address two important problems; one is old, and one is new. The old problem pertains to data fragmentation; rather than having dozens of independently managed datasets scattered across a multitude of platforms, it would be very convenient, and in fact much cheaper, to centralise these datasets into scalable flat storage space. The new problem pertains to big data initiatives; a big data scenario usually requires large amounts of varied data, and the data is so varied that it is not clear what it is, when it arrives, and how frequently it changes over time. Imposing constraints on the storage of such data entails constraining its future use and value mining. Although the popularity of DL has substantially grown in business and scientific fields, yet there is no formal consensus on a technical definition for it. Simply put: DL is as a data management architecture enabled by massive storage repository principally based on low-cost infrastructure to facilitate storing huge amounts of raw data, that for the most part, have a potential value which yet to be explored in the future [43].

ETL's schema enforcement on input data before its storage in DW is overwhelmingly characterized as *schema-on-write* [47]. In stark contrast, the DL concept originates from a new premise called *ELT (Extract-Load-Transform)* [47], which emphasizes extracting raw data from multiplicity of heterogeneous and unknown input sources, directly loading them into scalable unified repository in "untransformed state", and deferring all upfront data processing efforts to a later stage in the data's lifecycle inside DL. ELT reflects significant shifting from traditional ETL, as it allows preserving the native representations of data upon storage by postponing the T step until the data is absolutely required for a particular usage workload, thus, no schema enforcement is required prior to L step, and all incoming data is accepted, without rejecting any. This idea of "load data now and deal with it later" is usually characterized as *schema-on-read* [47], and it enables DL to offer empirical advantages that cannot be gained using any other management solutions. Among these advantages, we are mainly interested in the following:

- Loading big data is really fast undertaking and can be readily automated, as users no longer need to define standard data models, design rules for handling cleaning, validating, or aggregating data upon its input.
- The no-schema approach relaxes data capturing restrictions and enables complete storage agility, this becomes very attractive when the user requires centralising a variety of voluminous big data with unknown schemas and with rapidly evolving structures. Data of an arbitrary type can be directly "dumped" in the lake.
- DL gives end users more power to explore big data in their own way with extreme flexibility to impose ad hoc structures and transformations on the data as needed, consequently, the user can ask questions that the stored data might hold answers for, not just the type of questions that can be mentally realized during data storage, but also new questions which were not thought of at that stage.

The above advantages are not without costs. There are doubts and concerns about the possibility of data becoming incomprehensible due to the absence of unified schema or similar means of data interpretation, and that the ample accumulation of incomprehensible raw data could cause DL to drift into a *data swamp*. The analogy here suggests that a lake is somehow neater and more orderly than a swamp, but the only reason it appears so is due to the complexities hidden below the surface. In a swamp, some of the complexities are clearly visible: the data consists of disparate datasets and isolated information silos. This intuition is well established in DL literature. Gartner recently published a report to outline the potential pitfalls in adopting DL as an enterprise-wide big data management solution [62]. The report portrays DL as a "catch-all" repository where storing raw data is easy but pulling it out – or even making sense of it – is very difficult undertaking due to DL's inherent lack of data *governance* and *quality*. Governance is defined as the policies and procedures required to ensure proactive and effective data use [63]. Data quality is defined as the state of completeness, validity, consistency, timeliness and accuracy that makes data suitable for a specific use [64]. Dekker [65] states that data is of high quality "*if they are fit for their intended uses in operations, decision making and planning*".

Current state of the art governance and quality techniques rely on the existence of a unified schema to define how the stored data can be systematically discovered, accessed, and used, and what quality and integrity constraints are imposed on its usage [66]. By relinquishing unified schema enforcement on data upon its ingestion, DL becomes a natural architecture for agilely capturing and storing raw data with a plethora of native schemas – ranging from relational, self-describing (e.g. CSV, JSON, XML), and schema-less (e.g. free-text and multimedia). The downside of the offered agility is that: data in its native representations is rarely immediately available for consumption since the end users cannot run usage workloads on data with severe discrepancies, instead, they must wait until application-specific schemas are defined. On one hand, retrieving raw data without known schemas entails propagating large amounts of heterogeneous, inconsistent, or irrelevant data to the end user [67]. On the other hand, attempting to *govern* data upon its retrieval from DL and before its use is very difficult undertaking [68,69], for instance, only limited number of quality rules (e.g. *denial constraint* [70]) can be defined without the need for schema information. The work in [71] further extends these concerns by arguing that the schema-on-read does not only enhance data accessibility and agility but also relieves the DL's administrators from any upfront processing burdens and squarely placing them on the shoulders of end users in very problematic way from management perspective. Schema-on-write approach exists for a strong reason: data with an enforced schema can be readily understood by end users, such understanding is important for querying and utilising it [71].

To compensate for the lack of schema-on-write enforcement, the DL concept emphasizes the following simple assumptions [42,62]:

- The user can recognise the contextual bias of how data was pulled from its sources.
- The user can accurately identify the correct data in DL predicated on its structural characteristics.

- The user knows how to reconcile the semantic and structural heterogeneities of DL data and can readily integrate it during usage workloads.

Recent studies (e.g. [62],[71], and [72]) strongly disagree with the above assumptions on the basis of their negative impact on DL's applicability in the real world. For instance, Gartner analysed DL adoption in the business domain and delineated the following:

> "*While many assumptions may be true for users working with data, such as data scientists, the majority of business users lack this level of sophistication or support from operational information governance routines. Developing or acquiring these skills or obtaining such support on an individual basis, is both time-consuming and expensive, or impossible.*".
>
> [62]

Overall, this review provides an accentuated insight into the impracticality of utilising DL stored data by unskilled end users. In any usage scenario, the user will want to retrieve particular raw data from the lake (full datasets, or subsets of data entities), and customize how the usage workload at hand can be executed on the data as efficiently as possible. Before such an endeavour can take place, the user should first conduct an exploratory analysis to identify relevant useful data. Once data is spotted, the user will need to determine its utility and detect any anomalies that would require pre-processing (i.e. quality) [58,69]. Unless the user already understands the data, they would need to gather information about important facets of the data by asking questions like:

- *Where did the data come from, is the source of origin reliable?*
- *How old is the data, are its implicit facts outdated?*
- *What changes occurred in the data, is the degree of its fidelity trustworthy?*
- *What is the contextual meaning of the data, is it really relevant?*
- *What is the representation of the data, is it accessible and retrievable?*
- *How to combine the data with other data that have been found earlier?*

Finding answers to this kind of questions requires a priori knowledge regarding the lake's data, which is typically provided by means of the *metadata* technology. In the next sections we review this technology from multiple viewpoints and describe its roles in simplifying the user-experience for DL systems.

2.2 Metadata management

From a general viewpoint, metadata is popularized as "*data about data*", or "*information about data*" [73]. This definition is broad; more specific definitions have been provided in the literature, perhaps the most comprehensive of which is the one from [74]: "*Metadata is structured data about an object that supports functions associated with the designated object*". This notion implies the systematic organisation of the raw data based on metadata specifications, or *functions*. Metadata functions may include discovery, tracking (lineage), storage and archiving, organisation and management, privacy, query and retrieval of data throughout its lifecycle [75]. Researchers draw on metadata functions to create *typologies* categorizing desiderata metadata [76]. Enterprise DW typologies may include technical and business, and on finer graining level, process and operational metadata [77,78]. Different typologies might also be adopted in other fields, for example, digital library and information retrieval fields use descriptive, structural and administrative metadata [79], database design field may use structural and guidance metadata [77,79]. Beyond labelling and categorization, the types of metadata are meant to connect to the lifecycle of the data entity being represented or tracked. Metadata types can collectively be thought of as "value-added language" [73] that serves as an integrated layer in any data management system, which if appropriately placed and made accessible, to humans and machines, can act as eloquent language to enable the interplay between data entities stored in the system, and a particular activity over them, such as access, linking, analysis, among other similar directives.

While the metadata application is manifold [73], with various typologies coming from a wide range of research fields, in the context of this thesis we focus on two frequently adopted metadata in DL literature: *lineage* (*provenance*) and *semantic* metadata. Buneman et al. [80] define lineage in the context of databases as essential management information that specifies the origins of data and the processes by which it arrived at the database premises. Likewise, Simmhan et al. [81] define lineage as one kind of metadata that tracks the steps by which the data was extracted from its sources and made available for use a management system. Semantic metadata serves as management information to describe the meaning of its associated data, and its physical and operational attributes (i.e. schema attributes, physical format type, structure information) [82,83]. Both metadata types may offer sufficient governance information about the DL stored data, which consumers can exploit to measure the quality of data and determine how it can be (pre)processed, integrated, and utilised [68,58].

Enterprise Hadoop-based data lakes (HDL), like Hortonworks[4] and Cloudera[5], are salient exemplars of a full-fledged DL system that regards lineage metadata as an enabler of data provenance, governance, and usage [71]. The core architecture of a typical HDL usually consists of: data acquisition system for collecting raw data from local/remote input sources; Hadoop File System (HDFS) [84], scalable distributed flat repository for schema-less data storage; Hadoop MapReduce [85], a software framework for processing huge amounts of raw data stored in HDFS (or similar repositories such as HBase[6]); and a basic metadata repository for managing the chain of custody and tracking the lineage of the ingested data. Technically, HDL is a flat architecture for storing huge amounts of data in untransformed state. Each data entity is associated with a unique identifier (key) and lineage metadata. To prepare the data for a particular usage workload, the consumer should conduct exploratory analysis on the lineage supplementary information associated with the data stored in HDL to determine its sources, subsequently, the consumer may have to go through the tedious task of carefully studying the documentation offered by each source (e.g. database catalogues, API documentation web pages, software manuals, etc.), and adapting suitable tools to understand the semantics of the data and its provided schema(s) [52]. Such task is known by many terms, the most frequent of which is *data stewardship*, and it enables the consumer to prepare the data for utilisation through MapReduce jobs. Essentially, a MapReduce job is a two-step process that involves: *map* which concerns retrieving data from the lake as a set of key-value pairs, and *reduce* which concerns shuffling, sorting, and listing the retrieved keys into groups, where the data in each group

[4] https://hortonworks.com
[5] https://www.cloudera.com
[6] https://hbase.apache.org

shares commonalities based on user's defined criteria. Particular groups may be selected for further processing, and the contained data may be further pre-processed through transformation using external ETL-like tools or ad hoc scripts before the actual usage workload can take place.

It is well-established that metadata can play active roles in reducing manual stewardship efforts. When metadata is managed efficiently, it can significantly improve multiple downstream tasks in the data processing pipeline, like schema modelling, data integration, data retrieval, and querying [77]. However, in contrast to lineage, semantic metadata may require a dedicated process to govern the necessary activities for ensuring its consistency and availability, including metadata extraction (or creation), metamodeling, annotation, retention, retrieval, and maintenance. The process of managing these activities is called *metadata management*, and it is problematic in dynamic environments. The work in [87] discusses some of the main essentials and challenges in this regard. Technically, in order for a system to exploit the full potential of metadata technology, it must be capable of creating, and associating data with appropriate lineage and semantic artefacts that are stored where they can be easily accessed and queried [73], indexed for consistent availability [82,88], and nonetheless, persistently kept up to date over time [60,71,81,82], Associations between the data and its relevant metadata artefacts should be accurate and comprehensive in capturing and reflecting as much information about the data as possible [82,89,90]. They should also conform to specific metadata standard(s), and made available in an appropriate machine-readable format [52,74,79] so they can be readily and uniformly accessed and processed, or even shared with third parties whenever needed [44].

An analogue comparative for metadata management is the management of reference and bibliographic data (i.e. card catalogue) in brick-and-mortar libraries. Librarians often manually add high-quality information to describe the library materials, such as their storage and origin (i.e. lineage), what the information in them is, and how to find them quickly (i.e. semantics). Catalogues are usually organised using a suitable management system (e.g. Dewey decimal classification) to simplify the access and use of their contents by librarians and visitors. Applying a similar procedure in a DL environment would be sufficient practice to deliver effective metadata management, however, such practice may also be impaired by obstacles pertaining to the difference between the two comparatives.

It is obvious that *manually* annotating big data with appropriate metadata artefacts is laborious and time-consuming, especially if the size of data is huge, which is the most popular characteristic of big data. In order to exploit metadata for big data management and usage, DL administrators should create it in an easy-to-understand format and utilise it to annotate everything in the lake, starting from highly structured data records, down to rudimentary flat files (e.g. unstructured textual contents), otherwise unannotated data will be invisible to consumers. The work in [82] portrays metadata (especially semantic type) as a necessary representation of whatever data it is associated with. Like any other kinds of representations, its *raison detre* is to summarize and reduce the actual content of its associated data to a very condensed easy-to-manipulate form. One important property of such representation is that: it is a collection of purposive artefacts that have associated purposes, and criteria of selection and summarization, to reflect those purposes. For example, the artefacts used to represent a review in product reviews dataset might be very different from those used to represent an artwork in a gallery dataset, this leads us directly to the need for various types of semantics which may have to be drawn from multiple semantics providers (i.e. ontologies [91]). Because there is no single ontology that can satisfy all purposes, an abundance of ontologies should be adopted to create appropriate metadata artefacts for representing a variety of types of data stored in the lake [71,92], while at the same time, providing ways to heuristically *connect* the adopted ontologies [93,94]. Without such connections, data consumers cannot utilise disparate metadata for integrating and utilising disparate data.

Manual metadata creation and annotation express a strategic shift of time and attention from data consumers to administrators. Such shift involves both advantages and expensive requirements. On one hand, the metadata artefacts created by humans are usually more precise than those produced through mechanical processes [82], and precisely crafted metadata can greatly enhance the quality of data and simplify its usage in various downstream data management tasks. On the other hand, administrators, as human metadata managers, must have concrete technical background regarding semantic technologies and metadata management activities, for instance, a DL administrator should:

- Know precisely what artefacts to be chosen for annotating the raw data at hand.
- Recognise the well-established ontologies available in different fields, and the semantic tractability of each ontology.
- Understand the appropriate techniques (e.g., *semantic integration* [94]) to establish connections between the adopted ontologies, which are necessary for inference over ontology-federated data representations.
- Comprehend the nature of data consumers to produce metadata artefacts that are compatible with their data needs and requirements.

2.3 Data lake metadata management

There exists a good deal of work in DL literature that focuses on automating the core activities of metadata management under the user's supervision. The common goal of existing approaches and frameworks is to dramatically reduce the costs of metadata management within DL environments by offering the necessary means to simplify the activities of metadata creation, extraction, annotation, querying. *Data profiling* [95] is the simplest approach for creating metadata by obtaining statistics and descriptive summaries about the raw data gathered from a given input source. Data profiling was originally introduced to support DW MMFs in assessing data quality and identifying its anomalies during ETL extraction and transformation steps [46]. Instance-level data profiling involves applying statistical operations on the data contents, such as the computations of minimum, maximum, mean, mode, percentile, standard deviation, frequency, variation, aggregates like count and sum, as well as applying rules to determine the length, discrete values, uniqueness, occurrence of null, typical string patterns, and abstract type recognition [58]. Schema-Level (semantic) profiling involves capturing relationships between raw data by determining its structural and syntactic similarities using advanced techniques such as clustering and semantic integration. Bohm et al. [96] proposed a tool called ProLOD++[7] for diving deep into the all-around semantic profiling efforts ranging from clustering correlated datasets and inferring schema in each resulting cluster, to instance- level

[7] https://hpi.de/naumann/projects/data-profiling-and-analytics/prolod.html

profiling statistics. The focus of this work is limited to Open Linked Data [97] that is strictly available in *Resource Description Format* (RDF) [98] format. Alserafi et al. [99] proposed a framework for profiling similarity, both syntactic and semantic, between datasets. This framework is based on an ontology alignment approach. Each ingested dataset – with metadata consisting of different profiled attributes – is converted into an RDF file. Profiling files are then sent to an RDF-based ontology engine to detect their similarities and record them as metadata. The framework uses PARIS ontology alignment [100] due to its robust performance on high scale RDF data. Salem et al.[69] introduce an algorithm to govern big data and describe the quality of large datasets by detecting the issues of each dataset and generating semantic metadata for its contents using a semantics-based profiling algorithm. Ansari et al. [101] extend the former work and introduce a generic framework for semi-supervised statistic and semantic profiling in any DL. The framework implements profiling algorithm that accepts external background knowledge about the dataset under concern, and its aim is to annotate the attributes of the dataset with these semantics, and the final output is a set of semantic artefacts that is stored into Hive [102], and can be directly queried by consumers. *Revelytix Loom* [103] is a similar but more advanced framework that is designed as an integrated layer in enterprise HDLs for data profiling and governance, and audit trail. The core of Loom is an extensible metadata repository for managing business and technical metadata, including lineage, for describing the data in HDL as well as the surrounding systems. Loom's active scan process automates the creation of metadata for HDL stored data. It crawls HDFS to discover and introspect new files. Loom's metadata repository is implemented using Hive; administrators can directly perform CRUD operations on the stored metadata, whereas data consumers can only read the metadata during usage workloads by executing Hive-compatible queries. Atlas [104] is a governance and metadata management initiative from Hortonworks[8] for HDLs. This Apache framework automates the creation of metadata and defines the relationship between the data stored in the lake. It focuses on lineage more than semantic metadata, however additional components can be added to its architecture in order to extend the core foundational services – an example of Atlas extensibility is given in section 4.2. Atlas enables the user to import or define ontological business-oriented metadata, and

[8] Atlas was initially started by Hortonworks and then taken over by Apache as a top-level project.

to export metadata to third-party systems. It also allows the user to create and store security metadata that specifies how data consumers can interact with the stored data, with column and row level masking based on cell values, cell attributes, and the preferences of the user. Regarding usage, Atlas offers (i) text-based search feature to locate relevant data and audit events across HDL quickly and accurately, (ii) visually browsing the lineage of datasets, thus allowing users to drill down into basic, security, and provenance-related metadata. Atlas's metadata repository is a graph database that is implemented using Titan[9], with multiple options to support a variety of stores for persisting the metadata graph, including: Berkeley DB[10], Apache HBase and Apache Cassandra[11]. It also contains an additional repository in the form of index store for indexing data and metadata using Elastic Search [105]. For adding metadata to Atlas's repository, libraries that are referred to as "hooks" can be programmatically called from various external systems such as Hive, Nifi, Falcon[12] and Sqoop[13]. A hook captures metadata about data and events in the respective systems and propagate them to Atlas, which in turn consumes the forwarded metadata and updates the metadata repository. Any updates applied on the metadata repository, whether via hooks or APIs, are propagated from Atlas to downstream systems as events. Systems like Apache Ranger consume these propagated messages and allow administrators to act upon them (e.g. configuring governance policies, access control, etc.). Similar full-fledged, but commercial MMFs are also available as HDL-specific solutions, for example, Cloudera Navigator[14], as part of ClouderaEnterprise, enables the user to add and manage metadata about the data stored in a target HDL, and offers critical metadata-based data discovery and continuous optimisation of lineage and governance enforcement services to the HDL user.

Shifting to current state of the art scientific research; Ground Metadata [106] is an on-going research project aims to provide vendor-neutral (generic) management approach for collecting contextual information in big data applications. Information may be collected

[9] http://titan.thinkaurelius.com
[10] http://www.oracle.com/us/products/database/berkeley-db/index.html
[11] https://cassandra.apache.org
[12] https://falcon.apache.org
[13] https://sqoop.apache.org
[14] https://www.cloudera.com/products/product-components/cloudera-navigator.html

through big data classification it, tracking the data follow with hosting application, and monitoring who is using the data, when, how, and why. Contextual information is stored as metadata in a SQL backend and made available for various purposes, such as data model-specific interpretation, reproducibility, interoperability, and governance. Ground Metadata implementation is publicly available[15] but without any literature to explain its architecture or empirical performance benchmarking. CLAMS[68] is a system to discover and enforce expressive integrity constraints (i.e. denial constraints) from large amounts of lake data using very limited schema information given in the form of as RDF triples. The system is built for HDL, it executes SPARQL queries over the schema of each dataset stored in HDFS for detecting its issues and recording them as quality metadata. CLAMS metadata can be devised by end users as heuristics to determine how queries should be formulated. The system is optimised to operate on distributed Spark[16] for performance gains and scalability support, therefore, it can handle huge amounts of data compared to relational denial constraint algorithms. Datamaran [107] is a metadata extraction tool that scans semi-structured log-like datasets stored within a DL, and extracts their structures as schema metadata that can be used for data annotation. The extracted schema metadata can be also enriched by extended semantic artefacts to improve data quality. Datamaran operates as an unsupervised tool; it does not require prior knowledge for processing the datasets under concern. It automatically identifies field and record boundaries and separates the structured parts from the unstructured noise or formatting to tease apart multiple structures from within a dataset. Datamaran has been shown to be able of extracting structured relational metadata from semi-structured log datasets at scale and with high accuracy [107]. GEMMS [90] is another MMF that extracts and manages metadata about the data stored in a DL system called Constance [108]. This MMF aims to generate and annotate user's personal data in life science field by modelling it according to a common model defined by means of an ontology. Kayak [109] is a generic framework for managing DL contents through data staging and profiling. An extended discussion regarding Apache Atlas, GEMMS, and Kayak, among other relevant approaches, is given in section 4.2.

[15] https://github.com/ground-context/ground
[16] https://spark.apache.org

2.4 Existing work limitations

Mainstream frameworks and tools in the enterprise are generally more biased towards lineage than semantic metadata in terms of management, whereas their counterparts in the scientific field take the opposite trajectory. Existing body of work expects significant involvement of the DL user in the metadata management process for supervising its downstream tasks, commonly creation and annotation, in efforts to ensure the consistency and freshness of the metadata repository [108,110,111]. In Section 2.2, we listed four requirements that a DL user must meet in order to fulfil metadata management supervisory roles. These requirements may become costly when the DL user is considered "common user", which is the case of PDL. Based on the review presented in this chapter, we draw the following observations:

- MMFs currently put into practice focus on metadata creation, annotation, and usage but ignore another vital activity; *maintenance*. The reliance on metadata, especially for usage, emphasizes the need for high-quality metadata [65,68]. When data *velocity* comes into play, the quality of metadata may be drastically affected [112], as the representation details of the data can rapidly and/or frequently change due to schema evolutions and other external dynamics imposed by input sources. Velocity is not problematic when the end user is skilful and expected to be actively involved in the management process. However, when the attention is shifting towards automating the metadata management process, then addressing velocity through automated maintenance becomes a paramount effort.
- Most current state of the art MMFs are application-specific, designed to meet particular requirements, and tightly coupled with specific architectures [108]. Even when an MMF is assumed generic, it still requires a lot of efforts to facilitate its adaptation, customization, and interoperability in other architectures. To illustrate, we readily notice that Atlas is built for HDLs, and its adoption in another DL type necessitates extended adoption of other Hadoop projects, such as Apache Kafka[17], which is used by Atlas as a notification server to exchange metadata over hooks

[17] https://kafka.apache.org

[103]. Adopting CLAMS entails extended adoption of HDFS, Spark, and Hive. Similarly, GEMMS is proposed as generic MMF, but described as a solution that can only fit Constance architecture and its object-oriented model [108].

- The metadata management mindset in DL research is largely in accordance with traditional data cataloguing practices. An ideal metadata management solution should be data-driven and derive from context, such that, metadata – as purposive representations of data – should not only be intended for answering common questions (who, what, when, where and why) and performing integrations for data with basic heterogeneities, but rather supporting advanced inter-play functions over data to facilitate deep analytics over data with severe heterogeneities which require holistic integration to squeeze more value [110]. Providing unified views over integrated structured, semistructured, and unstructured data altogether is an extremely complex requirement. Such complexity is further elevated by the absence of schemas due to replacing ETL with ELT in DL architectures, as well as the fact that unstructured data does not naturally sustain any schemas that can be extracted and represented through metadata artefacts. That said, a sophisticated process is required for discovering the schemas of unknown (semi) structured data as well as building some sort of schemas for unstructured data before other metadata management activities can take place. Such process should also take into account sustaining the native state of data towards preserving its future value in DL – compared to other management approaches that emphasize transforming the structures of the (semi) structured data (e.g. DW) or even converting unstructured data into structural entities (e.g. personal knowledge bases [36,113]). These complexities plausibly justify why all the discussed tools tend to specialise in handling either (semi) structured or unstructured data.

2.5 Summary

Although DW is an important data management solution for addressing the fragmentation and isolation problems, the technical specification of its underlying processing workflow is problematic. ETL tends to exhibit brittle performance in the presence of the 3Vs model.

Enforcing predefined unified schema upon data storage (on-write) entails a partial loss in its value and restrictions on its future analysis. DL is a more recent data management solution that is specifically designed to address the 3Vs model complexities. The DL underlying workflow (ELT) entails inexplicable data storage due to the lack of an inherent mechanism for organising the stored data and orchestrating its usage. Lack of governance (and quality) promotes data isolations and threatens to shift DL into data swamp if not effectively addressed. Therefore, different approaches have been proposed in the enterprise and scientific fields to bring order to DL through heavy reliance on metadata. The *lingua franca* of these approaches is to materialize data governance and quality largely through lineage and semantic metadata. Lineage artefacts are used to track the stored data. Semantic artefacts are used to facilitate data metamodeling with respect to predefined ontologies, towards reconciling the semantic and structural heterogeneities of the stored data and enhancing its discovery, accessibility, retrieval, and usage. Although metadata is regarded in the DL literature as quintessential technology, the current state of the art approaches suffer several management limitations which are discussed in Section 2.4, and further elaborated in sections 4.2 and 5.1.

Chapter 3 Personal Data Lake Architecture

As presented in section 1.6, the main deliverable of this thesis is a framework (MMF) for automating data management tasks in PDL through metadata fabrics. The focal aim of this chapter is to describe the architecture of MMF and its position as a central layer in PDL. In order to explain the constituting components of MMF and their roles in PDL processing workflow, it is important to explore PDL architecture as a hosting platform. This is because reviewing each architectural layer, the rationale behind its design, and its constituting components, can greatly help in understanding *how* does MMF obtain personal data inputs, *how* are they processed in ways that satisfy the requirements of an automated management solution, and *how* a hosting platform can be optimised to exploit the full potential of MMF – this is particularly important in the cases of reusability in other relevant systems.

3.1 A Hadoop-independent data lake system

At present, the DL concept is tightly linked to Hadoop's ecosystem of Apache projects [43]. Any organisation that seeks to develop a DL based enterprise-wide data management solution normally adopts Apache Hadoop technologies for two reasons: cost-effectiveness, and technological feasibility. Basically, Hadoop is a collection of open source software utilities that have been invented to offer new ways of storing and processing big data at limitless scale; in fact, no data is too big to be handled by an HDL. Hadoop does not require expensive hardware, instead, it leverages on benefits from distributed parallel processing for handling huge amounts of data. It can run on any number of cheap commodity servers to store and process data with low-cost and high scalability according to changing needs. Such infrastructure can easily address the 3Vs model of big data, and this is the main reason for its wide adoption by the major enterprise organisations and service providers[18].

[18] List of100 major organizations using Hadoop: https://wiki.apache.org/hadoop/PoweredBy

There is no formal specification (e.g. reference architecture) to describe a standard process for building an HDL, instead, Apache ecosystem offers a vast range of utilities, each with distinct characteristics to meet certain design requirements. Generally, the architecture of a mainstream HDL consists of four layers: *ingestion*, *storage*, *management*, and *access*. Each layer can be developed by drawing and mixing multiple specialised Apache technologies. It is important to note that the HDL architectures currently put in practice consist of too many Apache technologies[19]. A recent report estimates that, on average, 20 different technologies[20] are needed to implement full-fledged HDL, with each technology uses different language, manages different data types, supports particular design purposes, etc.

Following the thesis aims set out in section 1.4, PDL is developed independently from Hadoop ecosystem. Basically, if we utilise the technologies adopted in current state of art HDLs as fundamental building blocks in PDL architecture, then our final solution is certainly not usable by common users due to escalating complexities relating to user-experience. It is impossible to have a common user with broad and deep expertise across multiple Apache technologies, nonetheless, even if the user has such brilliance, she still needs to fulfil management roles that are normally played by a team of experts in an enterprise organisation. This observation emphasizes that any Hadoop-based PDL solution would be regarded as modern PIM solution that could efficiently account for the problems of personal data, but at an expensive cost: difficulty of use. A centralistic viewpoint here is the ideological difference between DL and PIM: while the former is specifically designed for skilful *objective-in-nature* users (e.g., data scientists, analysts, wranglers, etc.) who look for advanced technologies to derive more value from internal and external data in the business domain, the latter is designed for the personal domain, wherein unskilled *subjective-in-nature* users look for easy ways to conveniently store, manage, and retrieve their own data [1], and are unwilling to invest time and effort in learning how to use complex tools to do so. To situate our viewpoint on scientific ground, we reviewed several previous PIM studies, and found that common users may encounter difficulties even in using PIM systems with basic functionalities (e.g., organising files [114], emails

[19] List of all Hadoop projects: https://hadoopecosystemtable.github.io
[20] https://hortonworks.com/ecosystems/

[115], bookmarks [1], etc.). Common users also tend to avoid investing effort or time in learning more about their systems, as the derived value is marginal, and eventually the persisting difficulties affect their satisfaction with the offered user-experience in a negative way. Boardman et al. [7] among the early researchers in PIM field to highlight this finding; in their words:

> "*Since PIM is an ongoing and often repetitive everyday activity, we found that even relatively minor bugbears can build up and have a negative impact on productivity and/or user experience.*".
>
> ([7], p.3)

The adoption of DL concept in the personal domain necessitates taking careful design and development considerations for a DL-based PIM solution like PDL, perhaps the most important among which is the extensive focus on simplifying the user-experience which such solution should be obligated to deliver whilst maintaining the robustness of the offered functionalities. In the context of our thesis, such viewpoint has greatly influenced our research work as will be demonstrated in subsequent chapters. Many studies in the DL literature seem to be in agreement with our viewpoint, for instance, the work in [108] proposes Hadoop-independent DL architecture for managing the personal data of common users in life science field. Similarly, the work in [116] introduces ad hoc architecture for their proposed DL without reliance on Hadoop.

3.2 Personal data lake architecture overview

PDL architecture follows a design pattern called *layered pattern* [117], which emphasizes separating the components of a large architecture into similar-functionality groups called *layers*. Such separation of concerns allows to flexibly encapsulate the complexity of the workflow of a given system architecture, and facilitates higher degrees of reusability, maintenance, and scalability [118]. For example, it is possible to deploy a new, update an existing, or maintain a malfunctioning component in one layer without affecting other layers within the same architecture. It is also possible to configure the security levels of various deployed components, such as isolating sensitive components in the architecture core from the outside world whereas allowing other components to be accessible under appropriate access permissions.

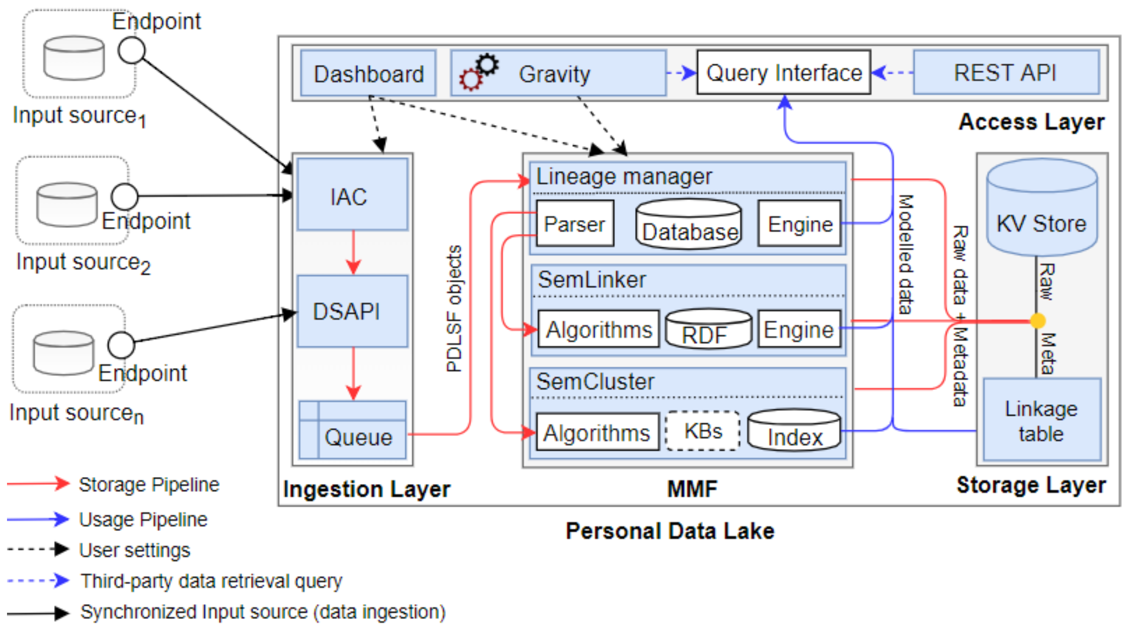


Figure 3.1– Overview of personal data lake architecture

Figure 3.1. depicts overview of PDL architecture; it consists of four layers that correspond to the main activities of PIM organisational cycle defined by Jones [1], namely: *ingestion layer* for personal data acquisition (and creation), *metadata management layer* for personal data organisation and maintenance, *storage layer* for personal data archiving *and* storage; and *access layer* for personal data use and sharing. Each layer in PDL interacts with its peers through *processing pipelines*. A pipeline is a predefined sequence of processing components where the output of one component (e.g. raw data, metadata, business logic) is moved as input to the next inter- or intra- layer component until reaching the last component in the sequence. The underlying ELT of any DL is rigorously reviewed in section 2.1. In following, we define two pipelines that constitute PDL's ELT.

Storage pipeline: *Extract-Manage-Load* activities sequence for processing input raw data. The first activity concerns ingesting personal raw data from input sources through the ingestion layer. Ingested data is dispatched to the management layer in the second activity for metadata processing (creation, annotation, maintenance, storage). The final activity involves dispatching the raw data and information about its associated metadata to the storage layer for permanent archiving.

Usage pipeline: *Load-Transform-Use* processing activities for utilising the stored data. The first activity concerns receiving an input query formulated either directly by the user or more commonly through a third-party service plugged in the access layer. The input query is dispatched to MMF for compilation and execution. MMF is responsible for loading all the raw data stored in the storage layer which is relevant to the query's requirements. In the second activity, the loaded data is organised based on its associated metadata (*views*), with possible transformations that are called *virtual transformations*, and returned to the access layer as query results. The final activity involves the direct use of the results, or processing it before it can be used to serve the query issuer's requirements.

In the remainder of this chapter, we construe each layer in PDL architecture and its relations with the proposed MMF to illustrate how the latter is handling the ELT specified above.

3.3 Ingestion layer

Data ingestion is an input process that concerns controlling the flow of personal raw data from its original sources to the lake. Technically, there are two issues to consider in personal data ingestion: PDL must support the end user in ingesting raw data from a range of target input sources regardless of their "offered" data collection mechanisms. Though service providers commonly serve data collection via RESTful APIs [119], not all personal data sources offer the same means, instead they may offer collection through application-specific protocols (e.g. SOAP, SPARQL, SMTP, etc.), or not exposing any explicit means at all (e.g. PC software applications, smartphone built-in services) [36]. Secondly, PDL should relax ingestion constraints to offer unlimited data capturing potential, which entails that the ingestion process should be comprehensive in collecting data from input sources regardless of their data generation modes (active/passive), the native representations of their offered data (schema, format, structure type), and its velocity over time.

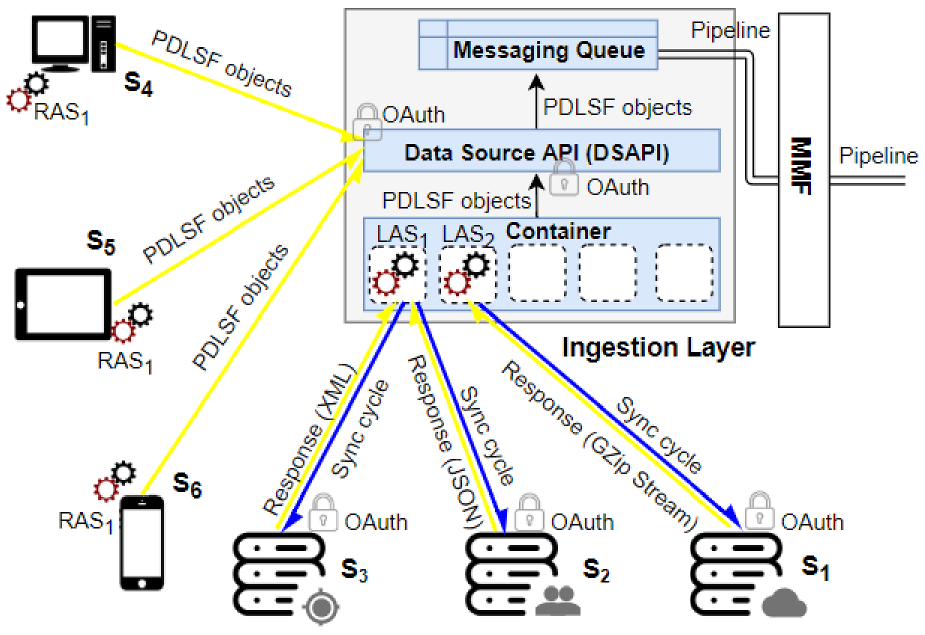


Figure 3.2– Overview of ingestion layer

PDL contains a layer called *ingestion layer* that serves as the entry point of the architecture as depicted in Figure 3.1. This layer is designed with extensive focus on addressing the aforementioned ingestion issues. The main functionality of ingestion layer is to automate the process of personal data ingestion and to help the PDL user in keeping the stored data up to date by *synchronizing* PDL with all the data sources which are of interest to the user. As depicted in Figure 3.2, ingestion layer consists of three components: public RESTful web API called *Data Source API* (*DSAPI*) which accepts data from the outside world as new inputs, *Ingestion Agents Container (IAC)*, a plugging-enabled platform for running third-party software agents that specialise in collecting data from target input sources, and *Messaging Queue*, a data persistence component to temporarily hold newly ingested data before the storage pipeline becomes ready for processing it.

3.3.1 Data ingestion approach

The underlying workflow of ingestion layer is conceptually based on data synchronization, a general management process concerns establishing and maintaining the consistency of data between two or more parties [120]. Synchronization is a fundamental process in many research areas, such as personal and enterprise data collection [36], mobile data backup

[120], and sensory data management [121]. Generally, there are two kinds of data synchronization: directional and bi-directional. In directional synchronization, a party A monitors data changes in a target party B over time so that any data not existing in A is propagated from A to B. In bi-directional synchronization two parties are monitoring the data changes in each other, so that any data not existing in one party is propagated to the other. The workflow of ingestion layer follows directional synchronization, such that, the ingestion layer exploits software agents to monitor target data sources, when an agent detects new personal data on a target source under monitoring, it issues a data collection request to that source, and upon the request's approval, the agent connects to the endpoint of the target and transfers any new data updates to the PDL premises in the form of payloads that are posted to DSAPI and eventually deposited in the messaging queue. There are two methods for implementing directional synchronization between PDL and a target input source which are *local agent-based synchronization* (LAS) and *remote agent-based synchronization* (RAS). In following we describe the details of each method.

Local agent-based synchronization

The LAS method solves the problem of personal data extraction from input sources with different kinds of endpoints through *plugging*, a feature that enables the user to plug one or more software agents (called *LAS Agents*) in the IAC component to automate data ingestion from particular input source(s). A LAS agent is technically an open source executable code (e.g. class, library, package, etc) that is independently developed and distributed by a third-party developer. LAS agent operates inside IAC as a one-way adapter between DSAPI and the endpoint of a target data source, its role is to establish an end-to-end communication channel over which PDL initializes and respectively maintains data synchronization with the target source. The workflow of LAS agents is as follows:

- Upon deployment in IAC, the LAS agent loads a set of user-managed configuration settings that describe the agent's ingestion behaviour, including: what target sources the agent is allowed to connect to, how to connect to each target source, and what personal data it should be extracted and propagated to PDL.

- The LAS agent establishes a connection with the remote endpoint offered by the target source at hand, and satisfies all the networking and security requirements imposed by it (e.g. authentication and authorization, permissible end point requests etc.).
- Upon successful connection with the target's endpoint, the LAS agent accesses user's space hosted in that target source (e.g. social media account, cloud storage folder, etc.), and extracts any personal data that has been actively/passively generated by the user on the platform of that source.
- The LAS agent propagates all the extracted personal data over the established communication channel to PDL. Data propagation is performed as authorized API post calls to DSAPI over HTTP. Once newly extracted payloads are posted by the LAS agent to DSAPI, the latter ingests and respectively stores it in the messaging queue component for later processing by the metadata management layer.
- Beyond this point, PDL is said to be *synchronized* with a target data source, and thereafter it is the agent's responsibility to maintain this synchronization over time by triggering periodic synchronization *cycles*. Each cycle involves repeating the above steps in order to propagate any new data generated by the target input source, and which have not been extracted in the previous cycle.

Remote agent-based synchronization (RAS)

LAS agents automate data ingestion from input sources with dedicated endpoints, however not all input sources offer such lavishness, and this is where RAS method comes into play. RAS relies on executable code units called *RAS agents* to ingest data using source-specific (ad hoc) personal data collection means. In contrast to LAS, a RAS agent is not locally plugged in IAC platform, but rather is designed as an installable software program that runs as a background process in the platform that runs a target input source of interest to the user. A RAS agent acts as: (i) virtual endpoint for input sources operating on personal devices which commonly do not expose explicit endpoints, such as ubiquitous sensors and applications locked by their vendors (e.g. Skype[21]), and (ii) one-way synchronization

[21] Skype is a third-party service that acts as data silo by locking in user data with no programmatic access point, however, its backend is SQLite database that is persisted locally on the device hosting it, hence, synchronizing it with PDL is possible using purpose-built RAS agent that can directly access the database and scan newly added data.

adapter between the operating system of a personal device and DSAPI. RAS workflow is similar to the LAS counterpart that is explained earlier but with a single difference, the execution environment of the agent.

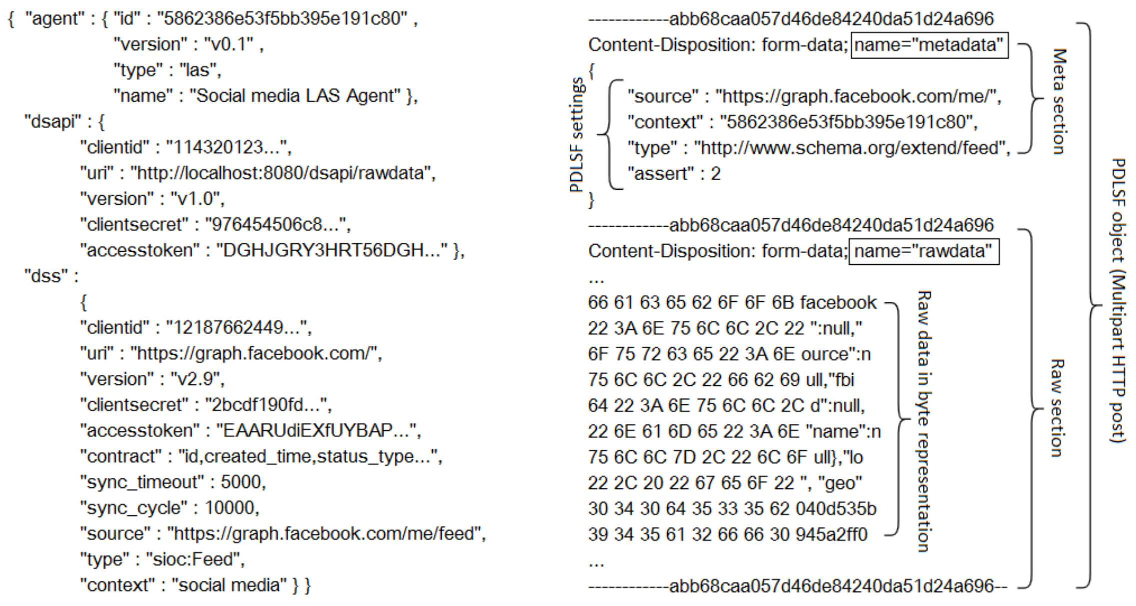
Figure 3.2 depicts an example of directional synchronization between PDL and six target sources denoted as $\{S_1, S_2, S_3, S_4, S_5, S_6\}$. The sources S_1, S_2, S_3 are third-party services independently run on remote platforms managed by different providers. These services are monitored by two LAS agents that run inside IAC with request/response communication channels to collect personal data through the public API offered by each service platform, for example S_1 offers API that allows the agent LAS$_2$ to collect data in the form of GZIP payload, similarly S_2 offers API that allows the agent LAS$_1$ to collect data in the form of XML payload. Figure 3.2 also depicts another three target sources, namely S_4, S_5, S_6, which are applications running on different personal devices (e.g. PC, smart phone, smart tablet, etc.), each contains an internal RAS agent that pushes new personal contents directly to DSAPI in the form of payloads serialised in a format called PDLSF (see section 3.3.2 for more details). It is not necessary to have a LAS/RAS agent that corresponds to a single input source in one-to-one fashion, instead the same agent may be used to synchronize PDL with multiple target sources (i.e. one-to-many). For example, Figure 3.2 depicts two LAS agents that are plugged in IAC; namely LAS$_1$ and LAS$_2$. The former synchronizes PDL with two social media services (S_1,S_2) whereas the latter synchronizes PDL with a cloud-based storage service (S_3). Analogously, the agent RAS$_1$ is used to synchronize PDL with S_4, S_5, S_6.

3.3.2 Metadata-added data ingestion management

Filling DL with raw data from a variety of input sources is not an easy task, this may explain why Hadoop ecosystem offers several ingestion-specific Apache projects for fitting different data collection requirements. LAS/RAS methods greatly simplify the filling process in PDL by relying on specialised agents to isolate the common user from the technical details of the data ingestion. In order to add[22] new input source to PDL, the

[22] Throughout this thesis, we refer to the procedure of synchronizing PDL with a new input data source of interest to the user as *source addition*.

user must select an ingestion agent that is compatible with that source, configure it with appropriate settings, and finally deploy it, whether in the IAC (LAS) or on a personal device (RAS).



```
{  "agent" : { "id" : "5862386e53f5bb395e191c80" ,
               "version" : "v0.1" ,
               "type" : "las",
               "name" : "Social media LAS Agent" },
   "dsapi" : {
          "clientid" : "114320123...",
          "uri" : "http://localhost:8080/dsapi/rawdata",
          "version" : "v1.0",
          "clientsecret" : "976454506c8...",
          "accesstoken" : "DGHJGRY3HRT56DGH..." },
   "dss" :
          {
          "clientid" : "12187662449...",
          "uri" : "https://graph.facebook.com/",
          "version" : "v2.9",
          "clientsecret" : "2bcdf190fd...",
          "accesstoken" : "EAARUdiEXfUYBAP...",
          "contract" : "id,created_time,status_type...",
          "sync_timeout" : 5000,
          "sync_cycle" : 10000,
          "source" : "https://graph.facebook.com/me/feed",
          "type" : "sioc:Feed",
          "context" : "social media" } }
```

```
-----------abb68caa057d46de84240da51d24a696
Content-Disposition: form-data; name="metadata"
{
   "source" : "https://graph.facebook.com/me/",
   "context" : "5862386e53f5bb395e191c80",
   "type" : "http://www.schema.org/extend/feed",
   "assert" : 2
}
-----------abb68caa057d46de84240da51d24a696
Content-Disposition: form-data; name="rawdata"
...
66 61 63 65 62 6F 6F 6B facebook
22 3A 6E 75 6C 6C 2C 22 ":null,"
6F 75 72 63 65 22 3A 6E ource":n
75 6C 6C 2C 22 66 62 69 ull,"fbi
64 22 3A 6E 75 6C 6C 2C d":null,
22 6E 61 6D 65 22 3A 6E "name":n
75 6C 6C 7D 2C 22 6C 6F ull},"lo
22 2C 20 22 67 65 6F 22 ", "geo"
30 34 30 64 35 33 35 62 040d535b
39 34 35 61 32 66 66 30 945a2ff0
...
-----------abb68caa057d46de84240da51d24a696--
```

PDLSF settings
Meta section
Raw data in byte representation
Raw section
PDLSF object (Multipart HTTP post)

a. Ingestion settings sample b. Ingested data in PDLSF format

Figure 3.3– PDLSF object propagated by LAS agent to DSAPI

There are two kinds of settings associated with each ingestion agent in PDL environment: *technical* and *PDLSF* settings. Technical settings are similar to the technical metadata heavily used in DWs, but with less verbose structuration, and their main utility is to govern how the ingestion agent connects to the endpoint of a target data source, and how the synchronization cycling should be established and maintained. From finer grained perspective, technical settings are divided into: *security*, and *synchronization*. It is well-known fact that most service providers offer endpoints that are wall-gardened with a lightweight security mechanism called OAuth [122] to protect their platforms from malicious access attempts by unauthorized third parties. DSAPI implements OAuth for the same purpose (see Figure 3.2). When a data source S_i offers an explicit OAuth-enabled endpoint, then a compatible ingestion agent would certainly be LAS, thereby as an adapter between a pair of endpoints it requires two separate sets of security settings: one for delegated access to S_i endpoint, and another for authorized access to PDL's DSAPI. In contrary, if the data source is a service running on personal devices then the agent is likely to be RAS, therefore only one set of security settings is required for connecting with PDL's DSAPI. A set of Security settings includes: *endpoint address* (URI), *secret key*, and *access token* [122]. A set of Synchronization settings include: *cycle interval* (sync cycle) which is an integer number specifying the time period between two synchronization cycles, and *connection timeout* which is an integer number specifying the time that the agent should take for the endpoint to send a response payload. Figure 3.3.a shows an example of security and synchronization and settings for a LAS agent that is configured to synchronize a PDL instance with a Facebook account over Facebook Graph API [256].

Common users usually have very abstract idea about the "types" of their personal data residing in a source of their interest. For example, it is a common sense for the user to assume that social media services deal with posts, mailing services mostly deal with email messages, and so on. If such abstractions are interpreted by means of semantic information (or concepts) that can be drawn from a formal ontology, denoted as O, then they can be easily shared with PDL. For instance, given the concepts `feed` and `email` from O, the former is an obvious choice to convoy the type of the data ingested from a social media service, and latter for data ingested from a mailing service. To support type conception sharing, we define our own serialisation format, called *personal data lake serialisation format* (PDLSF), that every ingestion agent must follow for submitting personal raw data with basic type concept to DSAPI. The description of PDLSF format settings, their permissible values, and the responsibility of value assignments for any LAS/RAS agent are listed in Table 3.2. In a synchronization cycle, the agent renders the raw data extracted from an input source as *PDLSF object* before submitting it to DSAPI. A PDLSF object is technically an HTTP raw message that follows a formal specification called *HTTP Multipart Messaging* [123]. A PDLSF object consists of two consequent sections: *meta* and *raw*. The meta section holds a JSON object that consists of the four attributes as listed in Table 3.1. The corresponding values of *source* and *type* are manually specified by the user upon the deployment of the agent in IAC (i.e. source addition). The remaining attributes are set by the developer of the agent, and can be overridden by the user as desirable. The raw section holds the data payload extracted by the agent from that target data source with native representation that is preserved intact. Figure 3.3.b illustrates an example for serialising personal raw data collected from Facebook API and serialised in

PDLSF format based on the PDLSF settings defined in Figure 3.3.a. The depicted PDLSF object here can be directly pushed to PDL's DSAPI over the established communication channel between PDL and Facebook. During our experiments with PDL, we observed that the conception of various types of structured and semistructured raw data can be readily abstracted using ontological concepts. Conversely, such undertaking for unstructured data is relatively difficult, therefore we relax the DSAPI constraints by permitting LAS/RAS agents to submit PDLSF objects that contain unstructured raw data in the raw section (by means of *assert*), and which contain *type* attribute in the meta section with missing value (null), to DSAPI. Consequently, the responsibility of abstracting the type of unstructured data is placed elsewhere in the storage pipeline (see Chapter 5).

Table 3.1 – PDLSF metadata settings

Setting	Type	Tracking	User-defined	Functionality
Source	URI	Semantic	Yes	To distinguish a target source from other sources in PDL.
Type	URI	Lineage	Yes	To express type of data in the form of concept drawn from O.
Context	Hash	Lineage	No	To represent the unique identifier of the ingestion agent.
Assert	Numeric	Semantic	No	To specify the structure type of input source data.

3.4 Metadata management layer

As ingested raw data accumulates in PDL over time, usage interposition tasks like data discovery, identification, tracing, and querying, increasingly become difficult to perform without an automated user-assistance management approach. This problem is not specific to PDL; we explored several studies that share the same concerns in section 2.2, that is: DL's natural lack of data governance and quality risks turning it into useless data swamp. As depicted in Figure 3.1, at the heart of PDL architecture lies the *metadata management layer* that is responsible for data management through formal metadata fabrics. This layer is resembled by our proposed MMF and in the following subsections we explore its main foundations, architecture, and workflow.

3.4.1 Metadata in personal data lake context

MMF adopts two types of metadata for managing the personal data stored within PDL: *lineage* (or provenance [81]), and *semantic* metadata. The term “lineage” can be viewed as a concatenation of two parts [80]: *line* and *age*. The first part involves maintaining information to describe the input source of the raw data at hand, how the data is collected from that source, and when [124,125]. The second part involves maintaining information to specify who accessed the data, what changes have been applied to it during its lifecycle, who made such changes, and using what tools [125]. In PDL environment, lineage metadata is largely useful for exploratory analysis during usage workloads such as identifying all the data extracted from certain input source(s); ingested through specific ingestion agents; collected on specific dates and times, and so on (see section 2.2). In this work, we further extend the functionality of lineage metadata to provide basic but effective privacy measure for controlling data accessibility and protecting user privacy.

Semantic metadata on the other hand is a significant source of prior knowledge for supporting data usage workloads. The common user, as well as authorized data consumers, may consult MMF to retrieve the necessary semantic metadata artefacts to understand PDL data and to determine how it can be efficiently exploited for creating new knowledge and deriving value [58,68]. Another approach is to directly operate on the metadata level through query formulation and delegate MMF to automatically process input queries, then retrieve and combine the results of each query to present them to the end user (human or machine agent) in an easy-to-understand form. The first step for materializing such approaches is to maintain machine-readable metadata artefacts that serve the following purposes:

- Formally describing raw data from conceptual point of view.
- Formally describing the physical and operational attributes of the raw data.

Regarding the first purpose, semantic metadata aims to add *contextual meaning* to raw data [82], independently of its native representations [71]. The contextual meanings expressed by formal artefacts provide a characterization of the conceptual elements by which the data consumer can understand what the data under concern is about, and how it is relevant to a particular information need [83]. Semantic metadata can be expressed in a wide range of languages; from natural to formal directive/descriptive languages, and with a vast range of vocabularies; from primitive – based on a set of agreed upon terms – to complex ones – with agreed upon taxonomies and thesauri. The simplest form of attaching semantic metadata to raw data is via *tagging*. Adding a concept to the PDLSF setting “type” in ingestion agents is an empirical example of metadata tagging. Regarding the second purpose, semantic metadata can collectively describe the physical representations of personal data including:

Data Schema: semantic artefacts can adequately express the contextual meaning of the elements (attributes) constituting the physical (or logical) schema of the raw data from conceptual point of view, for instance, specifying the semantic meaning of a column name in relational data, key in JSON object, tag identifier in XML fragment, keyphrase in a document, and so on. Schema-oriented metadata may also describe the datatype of a schema element (e.g. string, integer, float, Boolean, etc.).

Data Structure & Format: semantic artefacts can also express how the raw data is structured and formatted, for instance, specifying whether the data is relational, CSV, JSON, XML, free text, binary, photo, video etc. they may also describe the structural organisation of raw data, for instance, whether a given key-value pair in a JSON object is an attribute, object, array, nested object, etc., whether the naming type of a tag in an XML fragment is item, list, array, and so forth.

In the context of this thesis, we refer to representation-oriented semantic artefacts as *schema metadata*. An important perspective of metadata management is *precision*. Precisely produced metadata artefacts enable data consumers to efficiently determine what data should be excluded or included during usage workloads, whether for reasons of irrelevance, inappropriateness, or even redundancy. That said, the less precise semantic metadata, the vaguer it becomes, and intuitively, the more precise semantic metadata, the more “value-added” it offers. The degree of precision may drastically affect MMF utility; when semantic metadata is too general, the common user may find it difficult to determine the relevance of particular data in PDL, on the other extreme, too specific semantic metadata may degrade the overall performance of data access and retrieval. Ontologies are one of the most common means to specify the structure and modelling of raw data. The concept of ontology is "*formal, explicit specifications of shared conceptualizations*" [91]. *Conceptualization* refers to the conceptual modelling of some phenomenon in the world (personal data in this case) and involves identifying the relevant concepts of that phenomenon, the types of these concepts, the constraints on their use, and the relationships between them. *Shared* refers to consensual knowledge expressed by the ontology, i.e., it should not be understood by the common user only, but rather, accepted by a large group of other consumers. *Formal* refers to the fact that the ontology should be machine-readable. Not all ontologies have the same degree of formality, neither do they include all the components that could be expressed with formal languages, such as concept hierarchies, formal axioms, disjoint and exhaustive decompositions of concepts, etc. Due to formalism variance, ontologies may be broadly classified as [126]: *lightweight* or *heavyweight*. MMF adopts both kinds of ontologies for managing different types of data (see sections 4.5 and 5.4). MMF generates metadata artefacts that are always drawn from an ontology, and utilises them to annotate any raw data ingested by PDL.

3.4.2 Metamodeling approach

A metadata management solution that operates within PDL environment should be capable of handling metadata in a generic way for two reasons: (i) input raw data is likely coming a variety of data sources and with severe representation heterogeneities, and (ii) the maintained metadata may be utilised by multiple parties, with each party (i.e. the PDL user, third-party consumers) interacting with the metadata has its own characteristics and operates on the managed data for specific purposes. Accordingly, our proposed MMF has to apprehend a generic metamodel for metamodeling a constantly expanding variety of raw data accumulating in PDL over time, and it needs to be extensible for accommodating various information needs imposed by all the parties interacting with PDL. Nevertheless, such metamodel should offer formal and unified access to the data on the metadata level. In order to meet such requirements, one can follow a metamodel design standard. There are many standards proposed in the data management literature. Common Warehouse

Metamodel (CWM) [127] is a well-established specification for metadata modelling in DWs, it defines a standard for data exchange between DW and other systems in distributed heterogeneous environments. CWM extensively focuses on the metadata relating to DW. Since our focus is on DL-oriented metamodeling, we observed that there are some more metadata that need to be modelled in order to support user-assistance data management, which CWM does not cover. For example, CWM does not support annotating the native schemas of raw data with semantic (or schema) metadata due to the effect of schema-on-write approach followed by the ETL paradigm. CWM also cannot be easily extended through adding new external elements, which contradicts the requirements discussed earlier. Another approach is to follow SM4AM [128], which proposes a generic and extensible method to define and model metadata artefacts for user-assistance data management with extensive support for analysis workloads. SM4AM offers semantic awareness by leveraging RDF formalisation to express the metamodel. Therefore, the derived metadata artefacts are formal, machine-readable, and interoperable by third parties. In SM4AM, the metamodel level is used as the unified formalism of the metadata that is generated for data coming from heterogeneous systems. For each system (personal data source in the context of this work), the metamodel facilitates instantiating specific model to reflect the physical representation of the source's offered data. The initiated model may vary depending on the representation of the data and the details of its source. However, the metamodel is common over all the initiated models. It is noteworthy to mentation that SM4AM uses the concepts of *dictionary* and *type*. To offer additional explicit details for the user regarding specific data model, SM4AM captures finer-grained information about type conceptinions and maintains it in *type* – similar to PDLSF *type* setting. Different sources may have different specifications and purposes. A dictionary defines the set of potential methods, algorithms, or metadata types that are applicable to the source's data depending on various physical details (i.e. type, structure, schema complexity, etc.). We believe that SM4AM is an excellent specification to follow and the only drawback here is that: it is specifically designed for business domain, therefore we are bound to adopt its specification only as a guideline for the design of MMF's metamodel rather than reusing it.

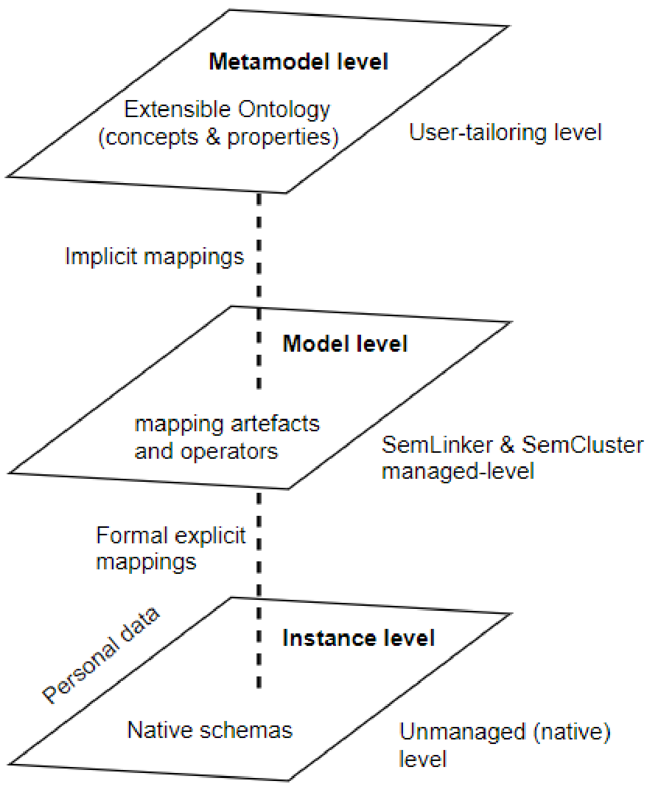


Figure 3.4– Overview on the three-level metamodel of MMF

MMF defines an abstraction of three modelling levels: metamodel level, model level, and instance level, as depicted in Figure 3.4. By ignoring the implementation details for now, the metamodel level consists of two generic extensible ontologies, each offers flexible interoperability across wide range of data sources and types, and is maintained by a specific MMF component (see sections 4.4 and 5.4.1). The metamodel level is designed to capture all the metadata types discussed in section 3.4.1. It is extensible, and allows the user to add particular elements that conform to the conceptualization of a given ontology as per evolving needs and requirements over time. The metamodel level is also designed to support personal data querying on the metadata level, though it can be easily extended to support other tasks afterwards. The next level is the model level which consists of a set of models corresponding to the representations of different personal data stored in PDL. A model in MMF is constituted by a set of *formal mapping* artefacts that aim to map various aspects of the data under concern to their appropriate counterparts in metamodel (ontological concepts and properties), each artefact can be either a direct mapping, or an operator-based mapping assertion (e.g. virtual transformation – see sections 4.5.4).

Mappings are formal because they are expressed using RDF to support semantic inference among other operations. Two components of MMF, called *SemLinker* and *SemCluster*, are responsible for creating and maintaining models for any ingested raw data in fully automatic fashion. Instance level is the lowest level and it is resembled by the native representations of the personal data stored within PDL in their raw forms. A model at the instance-level consists of physical schema elements, where each element is mapped to its reflection counterpart (typically ontological property) in metamodel level. Not all the elements in an instance level model must be mapped. This aspect offers more flexibility in modelling raw data even if its native schemas are not full-fledged or known in advance, we refer to incomplete modelling as *partial unified viewing*. Chapters 4 and 5 cover the implementation details of MMF's metamodel and the overall metamodeling approach, furthermore, Chapter 6 presents several scenarios to delineate the utility of this metamodel in real world.

3.4.3 MMF architecture overview and workflow

The MMF architecture consists of three components: *Lineage Manager*, *SemLinker*, and *SemCluster*. In following we give an overview of each component and its roles in MMF.

Lineage manager

The lineage manager is the MMF component that is responsible for lineage metadata management in PDL. From operational viewpoint, this component is a simple data flow system that controls PDL's data storage and usage pipelines, and records various lineage information regarding the flowing of personal data in each pipeline. Lineage manager consists of three components; *PDLSF parser*, *lineage database*, and *query engine*. PDLSF parser consists of a collection of internal parsers that specialise in deserialising data that exists in various physical data formats, ranging from standard to semi-standard ones. Among these parsers, *PDLSF deserialiser* which converts input PDLSF objects into their original HTTP Multipart form (i.e. meta and raw sections), and *Apache Tika* [129]; a toolkit that detects and extracts metadata and unstructured raw contents embedded in hundreds of formats[23] (e.g., PPT, XLS, PDF, etc.). The predecessor of PDLSF parser in the storage pipeline is the messaging queue component (see Figure 3.1). PDLSF parser pulls PDLSF objects from the messaging queue in FIFO fashion. Each pulled object is deserialised by invoking PDLSF deserialiser, then parsed by Tika to collect the metadata artefacts composed by the ingestion agent in its meta section, and to extract the raw data held in its raw section. During parsing, the PDLSF parser assigns an immutable hash key to the data entity at hand, which represents its global unique identifier during its life cycle inside PDL. Various lineage information about the data entity may be collected during parsing, including: its input source, its ingestion agent, processing timestamp, and any standard meta information associated with it (e.g., title, size, copyrights, etc.). Extracted lineage information is stored in the lineage database. Finally, the data entity is pushed to the next components in MMF for metadata processing.

Lineage manager offers three default personal data access control settings: *public*, *private*, and *custom*. *public* indicates that the data and its associated metadata are freely accessible by any third-party consumer via access layer. *private* setting indicates that data and metadata are solely accessible by the PDL owner. *custom* setting enables the owner to grant/revoke access of third-party consumers on an individual basis. Configuring access control settings may be applied on the data source level or entity level. This enables the owner to grant appropriate access to all the data ingested from a single input source, and blocking access to specific entities among them, or vice versa. By default, ingested data without user-defined access control is treated as private data. Security settings are stored in the lineage databases. The query engine offers the PDL owner, and other data consumers, SQL querying service to consult the lineage metadata during exploratory analysis. The query engine permits read-only (Select) queries on the lineage database, and all security settings are only accessible by the owner. Any data entity, that is loaded from the storage layer and returned to the access layer as a result of query execution over the metamodel of MMF, is reviewed by the lineage manager to determine whether the query issuer has the right permission to access and process it.

[23] https://tika.apache.org

SemLinker and SemCluster

In distributed computing environments, designing a comprehensive solution for managing heterogeneous big data using a single underlying management strategy is an impractical undertaking [111] due to the severe heterogeneity arises from big data structural representation types (structured, semistructured, and unstructured) [130]. Each structure type may impose specific requirements that the management solution must meet using a compatible management strategy, for instance, a data integration approach, that operates on the schema level of (semi)structured data, may be capable of remedying isolation issues pertaining to the structural heterogeneities, however such approach may rapidly become useless when operating on unstructured data due to the lack of explicit schemas. In order to address the structural heterogeneity of big personal data, we utilise two strategies that are resembled in MMF as two components; SemLinker and SemCluster. Each component can be viewed as a complete metadata-based data management system that operates on specific structuration; the former specialises in managing structured and semistructured data, whereas the latter specialises in managing unstructured data. As depicted in Figure 3.1, the lineage manager is the predecessor of SemLinker and SemCluster in both processing pipelines, and represents the entry point to each component in terms of data/query inputs. When the lineage manager finishes processing a newly ingested PDLSF object, the contained data entity and its associated metadata are dispatched to the specialised MMF component for semantic metadata processing (i.e. creation, annotation, and maintenance). Lineage manager selects the right component depending on the value specified in the *assert* attribute inside the meta section of the PDLSF object at hand (see Table 3.1). As listed in Table 3.2, the data entity is structured when *assert*=1, semistructured when *assert*=2, and for both cases, SemLinker is selected as the right component to process the data entity. When *assert*=3 the entity is unstructured text (e.g. PDF, word processing, etc.) and in this case the entity is dispatched to SemCluster. Finally, when *assert*=4 the entity is of another unstructured type (e.g. multimedia, binary file), and in this case the entity is directly dispatched to the storage layer without further metadata processing (see MMF limitations in section 7.2). A data entity dispatched to SemLinker or SemCluster passes through exhaustive metadata management workflows which are construed in Chapters 4 and 5. Both components are predecessors to the storage layer components.

Table 3.2 – Metadata processing applicability based on structure types

Structure Type	*assert*	Lineage M.	SemLinker	SemCluster
Structured	1	√	√	X
Semistructured	2	√	√	X
Unstructured (text)	3	√	X	√
Unstructured (generic)	4	√	X	X

3.5 Storage layer

Storage layer represents the "backend" of PDL architecture and its main functionality is to store personal raw data and information about its associated metadata. To deliver attractive user-experience, PDL needs to make sure that its user can access personal data quickly and conveniently, a paramount effort emphasized by *Memex* seven decades ago [131]. Essentially, accessibility can be greatly enhanced in a data management solution when the adopted storage approach is optimisation-oriented; this concerns not only *where* to store the data, but also how it should be stored to facilitate later efficient retrieval. Optimising data storage contributes, to a large extent, in improving the performance of MMF and enabling it to cope with surges in data demand imposed by different use scenarios within PDL. The first step towards delivering such optimisation objective is to carefully consider how the backend is designed and implemented. An intuitive option is to build from scratch a database system that is specifically designed to meet PDL empirical requirements, however, the trade-off of such option is forfeiting several advantages that could be gained from reusing an existing databases system, the most obvious of which is the fact that a mature software with proven efficacy is probably more reliable and requires less development and maintenance efforts. With this purely practical consideration, there are several criteria questions a one should regard for undoubtedly choosing the right database for PDL, and as follows:

- *What kind of data model should the database support?*
- *What kind of query language does the database offer?*
- *How flexible the database scales up in future?*
- *How reliable the database for housing the entire black box of the user*?

Multiple other criteria that relate to distributed computing are often considered in the development of enterprise DL systems, mainly including: maintainability, availability, and cluster-based shredding. In this work, any criteria beyond the listed question are discarded, since unlike enterprise DLs, PDL is certainly designed for individual users and is expected to run on single machines not distributed clusters.

3.5.1 Database selection

Until recently, relational databases have been always considered as the perfect backend for almost all data management systems, mainly due to the effectiveness of their ACID properties [132]. However, the emergence of big data movement has rendered this kind of databases as obsolete in modern data management systems [133,134]. First, the data size has increased tremendously, relational databases find it very challenging to handle huge data volumes, and addressing this issue is only feasible through vertical scalability which entails incurring more computational hardware power. Secondly, the majority of big data comes in semistructured and unstructured formats, whereas relational databases are designed to accommodate structured data (e.g. transactional, sensory, and financial). The necessity for finding effective alternatives has led to the emergence of NoSQL databases. The term NoSQL[24] was first introduced in 2009 to describe non-relational databases like Google BigTable[25] and since then it is widely adopted in enterprise and academia. The notion "NoSQL" is not really accurate because NoSQL databases rarely fully drop the relational model [135].

In spite of being a recurrent theme in big data research, NoSQL covers a vastly broad spectrum of very distinct database systems [133], their common characteristic is trading ACID in exchange of relaxed storage constraints, optimised data read and write, flexible

[24] This term was originally suggested in 1988 to describe databases that did not use SQL interfaces [136].
[25] https://cloud.google.com/bigtable

horizontal scalability, beside other performance gains [136] that are not related to our work (e.g. availability, distribution). Researchers in the database literature tried to categorize NoSQL databases and identify their kinds based on the design architectures and goals they support. As a result, they suggested grouping the databases of different vendors into four broad categories [133,136]: *Document* based, *Column* based, *Graph* based, and *Key-value* (KV) based. Table 3.3 lists a general comparison between the general properties of each category. Currently, there are scores of NoSQL databases, and each may exhibit optimised performance in particular big data scenario [133,135,136]. To find the "right" database for PDL, we need to base our selection on solid performance benchmarking that is verified by multiple studies which exhibit interests similar to ours. The studies in [137-142] introduce empirical performance evaluations for a multitudes of NoSQL databases that belong to different categories, using YCSB [143], a benchmark platform offered by Yahoo! for evaluating databases performance, particularly NoSQL.

Table 3.3 – Types of NoSQL databases

Category	Model Description	Model Aspects	
		Pros	Cons
Document	Every item is stored as a pair of key, and a complex data structure called *document*.	Unknown data storage; Data nesting.	Slow CRUD and I/O; Lack of join queries
Column	Items are stored in columns instead of rows.	Huge storage; Flexible scalability;	Undefined data usage pattern.
Graph	Items are stored as nodes in network and edges between them to represent their relationships.	Flexible typed relationships; Advance querying capabilities.	Limited scalability; Limited applicability.
KV	Every item is stored as an attribute name (key) together with its corresponding value.	Unknown data storage; extremely fast CRUD;	Basic querying capabilities; Lack of rich indexing

These studies share the same conclusion: Redis [144] is significantly superior in terms of data read/write performances and generic applicability, compared to almost all current state of the art NoSQL databases. The covered experiments indicate that Redis is so fast that it requires overall 1.52 seconds to read, write, and update 600K data items in three subsequent workloads [140,141]. The reason of performance superiority is mainly attributed to the underlying design of Redis as an optimised in-memory KV database, such design is typically regarded by the database community as a trade-off, because it entails volatile storage mechanism; any memory interruption will lead to losing the stored data. However, Redis is equipped with built-in snapshotting and journaling approaches to conveniently persist the stored data on the disk, thus overcoming the volatility issue. Redis is pure KV database; it fully drops the relational model and embraces representation-agnostic model, thus, any possible data item can be associated with a unique key and stored as key-value pair without further assumptions. This schema-less storage style is the main reason for its applicability in any possible big data scenario. Beyond experimental evaluations, Redis is highly reliable and scalable open source software that is currently adopted by scores of dominant service providers[26] in the digital industry for different uses, including: full-fledged database, cache server, or messaging queue. For instance, Twitter uses Redis as its main cache server and has scaled its storage capacity up to 105 terabytes. Based on these characteristics, we believe Redis is well suited to meet our criteria specifications better than any other current state of the art NoSQL database.

Existing HDLs favour document and column database types (e.g. Cassandra[27], HBase[28], MongoDB [146]) over other types. Document and column based databases offer schema-less storage and encompass sophisticated querying engines that allow the user to formulate queries ranging from low-level MapReduce based scripts, and database-specific (e.g. MongoDB document querying [146]), high-level SQL-like (e.g. Hive [102]) and Flow-based (e.g. Pig [145]). Graph databases are known to scale poorly, unless running on distributed infrastructure. KV based databases typically offer limited querying capabilities, for example, Redis can only be queried through low-level *get* and *set* -like commands[29], whereas more sophisticated querying can be implemented through application-specific wrappers that would translate high-level query formulations to low-level Redis commands.

[26] List of prominent service providers using Redis. https://redis.io/topics/whos-using-redis
[27] https://cassandra.apache.org
[28] https://hbase.apache.org
[29] List of Redis commands. https://redis.io/commands

It is obvious that the efforts of developing and maintaining a system can be largely reduced if its chosen database offers sophisticated querying capabilities, however, this also means the system's potential may become greatly pinned by these capabilities, and future replacement of the database may be relentlessly tedious undertaking from redesign and data migration viewpoints. Currently, there is a growing argument for designing systems as *database-agnostic*, a term describes the capacity of a system to function normally with any given database rather than being customized for a particular vendor's database [11,148]. We believe that such agnosticism, combined with a strong emphasis on schema-less storage model, may become a design reference in data lake research. An unavoidable fact is that hardware and software are dramatically changing over time, as with the databases built on top of them, this section already introduced an empirical example of the rapid shifting from relational to non-relational storage model in the booming era of big data. Analogously, database agnosticism is important for PDL, since there may be future need to accommodate variety of different data requirements for uses constantly increasing as data streams become more numerous, personal data becomes larger and more varying over time, and data querying and sharing functionalities become tasked with increasingly demanding usage workloads. By taking these considerations into account, the "right" database (e.g. Redis) which might make sense to run as the main backend of PDL today, may become unsuitable to accommodate user's increased storage capacity and querying requirements in future. Henceforth, we implement PDL's backend as database-agnostic so that PDL can accept any possible KV database. In doing so, MMF, as the only layer in PDL architecture that directly accesses the backend (see Figure 3.1) needs only low-level interactions to read/write data whilst abandoning any other high-level capabilities offered by the chosen database, including sophisticated querying. In Chapters 4 and 5 we explain how MMF compensates the loss of capabilities due to agnosticism with native, robust, and effective alternatives.

3.5.2 Metadata and data storage approach

In PDL, metadata storage is *federated* and raw data storage is *centralistic*, such that, the metadata manged by a particular MMF component (the lineage manager, SemLinker, or SemCluster) is stored in an internal metadata repository maintained by that component, whereas ingested raw data is stored in the central backend. PDL's backend consists of two components; *unified repository* for data storage and *linkage table* for *metadata information* storage. Regardless of the adopted database system, the underlying structure of each backend component is a hash table that consists of a scalable set of entries. Like any other hash table, a hash entry here consists of two fields: *key* and *value*. The key field stores a unique identifier, and the value field stores a content associated with that identifier. Figure 3.5 depicts an overview of the PDL backend.

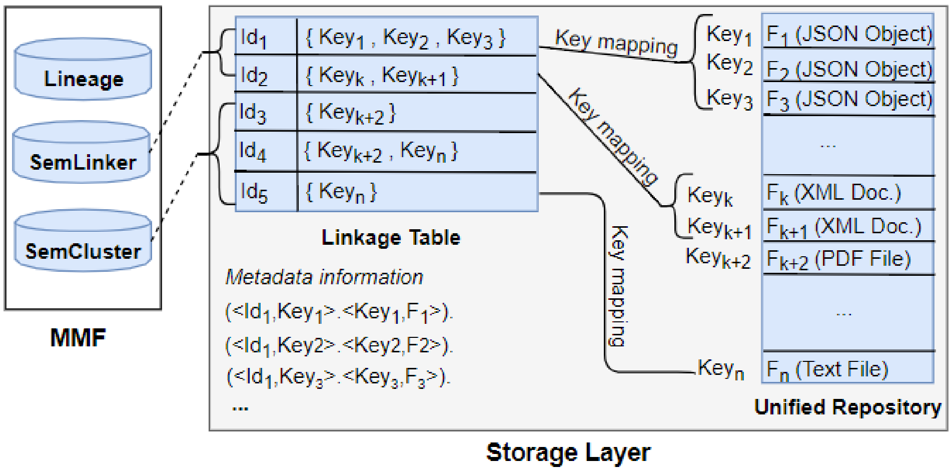


Figure 3.5– Overview of PDL backend and MMF-managed storage approach

As indicated in section 3.4.3, the lineage manager generates a unique identifier, denoted as key_F for the data entity, denoted as F, that is extracted from a newly ingested PDLSF object. Regardless of the processing component, when the metadata processing of F is finalized inside MMF, it becomes associated with at least one metadata record which also has a unique identifier, denoted as Id_i, that is issued by the respective MMF component. Therefore, the pair $\langle Id_i, key_F \rangle$ indicates that there is an association between the entity F whose identifier is key_F and the metadata record Id_i, such pair is called *metadata information*. Metadata information is similar to primary-foreign key relationships in traditional RDBMS, but instead of linking between data entities in separate data tables, metadata information explicitly emphasize the associations between the raw data stored in the unified repository and its corresponding metadata records that are stored in different metadata repositories. MMF controls how each data entity F and its metadata information are stored in PDL's backend through a two step-process, and as follows:

Step-1 (Data storage): MMF creates the pair $\langle key_F, F\rangle$ that corresponds to the unique identifier of F and its raw content. This pair is passed to the storage layer for permanent storage. Given Redis as the physical implementation of the storage layer, the pair $\langle key_F, F\rangle$ is added as a new entry in the unified repository by executing the following command: `[HSET keyF F]`. *HSET* is a Redis-specific command that adds the entity F in the entry whose key is key_F. If such entry does not exist in the unified repository– which is the usual case – a new entry with the key key_F is first created before F can be added as its field value. If key_F already exists then its corresponding field value is overwritten.

Step-2 (Metadata storage): In this step, MMF creates the necessary metadata information for the raw data entities at hand and stores them in the linkage table. Given a data entity F and the set of metadata records $\mathfrak{m} \neq \emptyset$ that is generated for F during its metadata processing, MMF creates the pair $\langle Id_i, key_F\rangle$ for each $Id_i \in \mathfrak{m}$ and passes it to the storage layer. Given Redis as the physical implementation of the storage layer, a pair $\langle Id_i, key_F\rangle$ is added in linkage table by executing the command `[SADD Idi keyF]`. *SADD* is a Redis-specific command that "appends" key_F in the entry whose key is Id_i. If such an entry does not exist in the linkage table then a new entry with the latter as its key is created before the former is added as field value.

As listed in Table 3.2, an ingested data entity may be processed by more than one MMF component, and the last processing component becomes the sole authority responsible for storing the set of its associated metadata information ($\mathfrak{m}$). When F is structured or semistructured then SemLinker annotates it with a set of schema metadata artefacts that inherently share a single unique identifier, that is, the URI of the current source schema version – see section 4.5.2. Accordingly, the set $\mathfrak{m}$ associated with F consists of a single pair $\langle Id_i, key_F\rangle$ where $Id_i = URI$ and key_F is the unique identifier of F (i.e. $|\mathfrak{m}| = 1$). In contrast, when F is unstructured then SemCluster annotates it with a set of metadata artefacts, where each artefact is the *seed* of a keyphrase extracted from the raw content of F during metadata processing – see section 5.4 for more details, and Table 5.2 for an empirical example. This means, if k keyphrases are extracted from F then $|\mathfrak{m}| = k$ and

Step-2 is executed k times; for each execution step a pair $\langle Id_i, key_F \rangle$, $Id_i = seed$, $Id_i \in$ $\mathfrak{m}$ is passed as new input to the storage layer.

Unlike the unified repository, each entry in the linkage table may consist of a pair of single key Id_i and a non-empty set $\{key_j, key_{j+1}, \dots, key_n\}$ stored in its value field, where each key represents the unique identifier of a data entity F that is already stored in the unified repository. Having keys of various data entities grouped by a single metadata key is an obvious indication of inherent relationship. For example, an entry Id_i in the linkage, $Id_i = \{key_j, key_{j+1}, \dots, key_n\}$, that is managed by SemCluster, indicates that the unstructured documents $\prod_{i=j}^{n} D_i$ share the same seed. When the PDL user searches for unstructured documents that contain the seed Id_i, then MMF directly retrieves the documents $\prod_{i=j}^{n} D_i$ from the unified repository and returns them to the user as correct search results. It is noteworthy to mention that the time complexity of the retrieval operation for $\prod_{i=j}^{n} D_i$ is O(1) given that the underlying implementation of the unified repository and linkage table is a giant in-memory hash table (i.e. Redis). Analogously, it is intuitive to assume that all the (semi)structured data entities ingested from the same input data source share the same schema metadata's unique identifier, thereby, the same storage and retrieval optimisation thus described is applicable. To illustrate, let the set $\{key_1, key_2, \dots, key_n\}$ be the unique identifiers of all the entities ingested from an input source S, whose schema metadata's unique identifier is Id_S, then an input query to SemLinker concerning the retrieval of all these entities is directly executed on a single entry in the linkage table, that is Id_S, and the result is retrieving $\prod_{i=1}^{n} F_i$ entities from the unified repository in O(1) complexity similar to the previous case.

Figure 3.5 depicts an overview of the storage approach. Here, multiple heterogeneous data entities are stored with their associated unique identifiers as key/value pairs in the unified repository. Additionally, the metadata information of these entities is stored as key/value pairs in linkage table with links to external metadata repositories maintained by SemLinker and SemCluster. The figure also illustrates how the MMF interprets the stored data in both components as pairs of key/values pairs. For example, the data entities F_1 and F_2 can be retrieved as the following subset:

$$\{\ \langle\langle Id_1, key_1\rangle, \langle key_1, F_1\rangle\rangle, \langle\langle Id_1, key_2\rangle, \langle key_2, F_2\rangle\rangle\ \}$$

Such subset indicates that F_1 and F_2 share an inherent relationship that is defined by the metadata information Id_1; this relationship can be interpretable by the MMF component which managed these entities during their metadata processing. It important to emphasize that F_1 and F_2 are stored in untransformed state, and are intended to remain so during their lifecycle inside PDL since MMF allows their processing only on the metadata level, (i.e. Id_1).

The simplicity of the storage layer architecture, backed by the flexibility of schema-on-read approach, and the MMF-managed data storage, are major enablers of database agnosticism; the backend development requirements are reduced to three main criteria: scalability, schema-less KV storage, and disk persistence to avoid any volatility trade-off. The decision of selecting Redis as physical implementation of PDL's backend is largely influenced by its empirical capacity in boosting MMF's performance during data storage and retrieval. By leveraging Redis, MMF obtains the following features: the maximum number of key/value pairs that can be stored in PDL is 2^{64}-1 on 64x machine [144], this means PDL storage limitation is 18.446 Exabyte. The maximum size allowed for storing a data entity within a single entry in the unified repository is 512 MB [144]. A single metadata schema' unique identifier can group up to 2^{32}-1 data entity keys which is far more than a common may possibly need. As illustrated above, MMF requires O(1) time complexity to retrieve metadata information from the linkage table; it requires O(1) time complexity to find any raw data in the unified repository; it requires 0.59 seconds to load 600K entities of small size [140,141], and relatively longer time if the target entities of considerably large size.

3.6 Access layer

The access layer has the responsibility of providing convenient access to the underlying layers of PDL on the access patterns expected. Generally, we can broadly categorize the access layer to be performing either a *pull* or a *push* of certain artefacts with respect to the serving layer. At a high level, the push access refers to the outflow of settings, queries, or source code from the access layer wherein these artefacts are pushed out to other layers in PDL architecture. The pull access refers to the outflow of artefacts from the serving layer wherein the access layer pulls settings, metadata, or modelled data. The access layer consists of four components which collectively represent the "frontend" of PDL as depicted in Figure 3.1. These components are: *Dashboard*, *Gravity*, *Query Interface*, and a RESTful API for data consumption. In following we give a brief overview of each component.

Dashboard

The dashboard is a configuration-focused component that provides important services for the PDL user to control the settings of various architectural components through user-friendly GUIs. In following, we list three kinds of GUIs that are relevant to the discussions in this thesis:

- **IAC GUI:** This GUI is connected with IAC component in the ingestion layer and it enables the PDL user to install, upgrade, or uninstall ingestion agents in IAC, and to configure the security, synchronization and PDLSF settings associated with each deployed LAS agent.
- **Privacy GUI**: This GUI is connected with the lineage database in MMF to help the user in configuring the access control settings of the stored personal data as specified in section 3.4.3.
- **Extensibility GUI:** This GUI is connected with SemLinker and SemCluster in MMF to help the user in adding, upgrading, or removing third-party plugins related to the underlying workflows of these components. For example, this GUI can be used to add new knowledge source for extending the semantic coverage of Sem-Cluster's ontology (see section 5.4.1), or adding new schema matcher to the set of plugins in SemLinker (see section 4.5.3).
- **Gravity GUI:** This GUI is the fronted of the *gravity* concept implementation in PDL, and it aims to enable the user in installing, upgrading, or uninstalling third-party gravity-enabled services.

Gravity

In 2011, a blog published by McCrory [148] introduced a discussion about a qualitative characteristic of data that was referred to as *Data Gravity*. In the years that followed, this characteristic has resonated in the digital industry. Data gravity is a metaphor describing the economics of data; it is better to keep the data where it is and not to be exchanged between the systems in distributed computing environments, no matter how big or small the amount of data may be. McCrory compared the cost of moving data, and found that gravity is cost-efficient approach among other rising advantages. Consequently, it was stated that data must have something comparable to a gravitational pull which pulls services and applications to it instead of the other way around. The work in [149] reports three approaches to bring computation to data and mitigate data gravity: agent platforms, code mobility, and Fog Computing. In this thesis, we leverage on code mobility to embrace data gravity in PDL architecture for gaining two important advantages:

- Simplifying personal data utilisation by relieving the PDL user from formulating complex queries during usage workloads; gravity-enabled services can be pulled from third parties and plugged in PDL's access layer to write and execute queries on the behalf of the user.
- Enabling the PDL user to share personal data with third-party consumers without privacy compromises. By authorizing gravity-enabled services to operate in the access layer, a personal data consumer would be able to collect interesting information without having the data to leave the PDL premises.

To materialize these advantages, we design a code execution platform in the access layer called *Gravity*. This component enables the user to plug in "trusted" business logic in the form of compiled source code (e.g. dynamic libraries, packages) that can formulate queries and submit them to MMF for execution on PDL metadata and data repositories. The query results are either directly presented to the user, or further processed for mining patterns, insights, or creating new knowledge. Another approach is to allow remote data consumers connecting to PDL, submitting gravity-enabled services, which the user can review, and upon granting permission for each service, it can execute as usage workload which enables the consumer to process PDL data and collect the processing results back to its platform. In gravity-powered scenarios, MMF operates as a central point of information exchange that offers the necessary capabilities to gravity-enabled services for querying highly managed personal data on the metadata level in terms of a well-organised metamodel and formal querying means. To illustrate how this works, consider a user having central heating system that operates on oil, and which is equipped in the user's house or office. The user may put oil consumption data in PDL and mark it as publicly available. An energy company would request to connect to the user's PDL and submit a gravity-enabled analytics service. Upon user's approval, this service could submit queries targeting heating data, which are executed by MMF and the results are returned to the energy company to to analyse them and generate personalised oil delivery offers to the user. In this example, it can be readily observed that the heating data does not have to leave PDL, instead, the data demand of the energy company is satisfied as a workload operating locally PDL. Similar ideas have existed in the personal data management literature for some time (e.g. [150,151,152]), however they are typically discussed from theoretical perspective due to the lack of the necessary technologies for constructing efficiently-managed black boxes.

Query interface

Query interface component represents the frontend of the query engines implemented in the MMF component. It is a simple command line for submitting queries to a particular query engine, and for displaying the query results. In sections 4.6, 5.5.2, 6.1, and 6.3, we discuss how the queries submitted via the query interface can be systematically executed on the metadata repositories to load the required data from the storage layer.

RESTful API

Besides gravity, PDL also implements a RESTful API endpoint for data consumption as an additional support for the user to share personal data with consumers who do not support gravity. Similar to DSAPI, the access layer's API is secured with OAuth and accepts only data requests that are signed with access tokens issues by PDL.

3.7 Summary

In this chapter, we presented an overview of PDL architecture and discussed each of its layers with extensive focus on the metadata management layer. The introduction of the chapter examined the possibility of implementing PDL using building blocks from Hadoop ecosystem. Our examination revealed that a Hadoop based PDL implementation is feasible but impracticable in the context of the personal domain due to the expensive technical complexities imposed on the usability of the system by common users. Then, we described how personal data can be pulled from input data sources and ingested inside the system. Multiple opinions indicate that annotating raw data with metadata upon its arrival, lineage particularly, is a very important undertaking for maintaining effective data provenance. A useful discussion on this matter is given in [178]. Realizing this, we introduced an agent-based ingestion approach that automates the extraction step in PDL's ELT and annotates ingested personal data with basic but useful metadata information. Next, we discussed our proposed MMF; the metadata types it operates on (lineage, and semantic), the design of its SM4AM-based metamodel, the components constituting its architecture (the lineage manager, SemLinker, and SemCluster), and the role of each component in controlling the storage and usage pipelines underlying the loading step in ELT. Next, we described the storage approach of raw data and its associated metadata information in the storage layer. Although many interesting database options are available, we explored the usefulness of database agnosticism as part of our effort to deliver agnostic solution for managing big personal data. Our discussions also covered how MMF controls the data and metadata storage independently from the adopted database details. Finally, we elaborated on PDL's access layer and how the data gravity concept is situated in this layer to simplify the lake's usability.

Chapter 4 (Semi)structured Data Management

The lineage manager in MMF maintains the necessary means to enable plain technical access to structured, semistructured, and unstructured raw data residing in untransformed state within PDL's unified repository. However, such accessibility can be only the first step in creating our envisioned analytics-enabled personal black box. To make finding and querying personal data in PDL both possible and practical, common users and third-party data consumers must not be confronted with the semantic and structural heterogeneities of raw data, instead they should be provided integrated homogeneous *views* that abstract from these heterogeneities. In this chapter, we explore SemLinker as a data integration system for automating structured and semistructured data management in PDL. To begin with, we first describe the driving challenges and requirements behind SemLinker proposal, next, we give a brief review of relevant studies and approaches. Then we introduce an overview of the theoretical foundations upon which SemLinker was designed and construe its architecture and participation in MMF workflow.

4.1 Research challenges and requirements

The main goal of *data integration* is to synthesize heterogeneous raw data collected from autonomous input sources into unified views that can be exploited by end users to utilise the data as if it is coming from a single well-managed input source, without requiring knowledge about where the data is stored and how it is natively modelled and structured [39,153]. The common assumption in DL research regarding data integration is that once the raw data is annotated with appropriate semantic metadata, it can be readily integrated using an integration system [71][43][52][110][111]. Though this assumption is plausible, yet it is impractical from the perspective of big data dynamics. Existing data integration systems are inherently complex and they require tedious manual interactions [110][183], for instance, an enterprise HDL system would rely on professionals and experts (e.g. data scientist [43], steward [52], or wrangler [71]) to play active roles in the data integration workflow, mainly to supervise the following tasks:

> **Schema Matching:** a task concerns identifying the similarity between the schema elements of two or more heterogeneous data entities coming from different input sources [154]. This task can be automated when the data is annotated with precise schema metadata artefacts. As indicated by Jarke et al. [40], the input sources of any DL system often do not expose full-fledged schemas. When the native schema is not completely known in advance, precisely annotating its elements with appropriate metadata to reflect their underlying semantics can become problematic in a way that prevents automating schema matching.
>
> **Schema Mapping:** while schema matching produces correspondences that state only the similarities between schema elements, schema mapping is a task concerns specifying these correspondences in a formal expression that can be interpreted as constraints over the data or as rules to guide the transformation of data during its utilisation [39][92].

The literature of other DL systems imposes requirements similar to those of HDL albeit details of such requirement are not explicitly specified – we expand on this in the next section. Essentially, PDL is designed for ordinary people and has no highly trained and skilled IT personnel to physically handle the above tasks. Equipping PDL with an efficient and easy-to-use data integration solution is therefore crucial for isolating common users from the technical integration details. Any data integration solution for PDL faces the following challenges:

The first challenge is extreme heterogeneity; it arises as an implication of dealing with various types of raw data collected from a large number of unrelated input sources [110]. The data sources of a DL system, even in the same domain (e.g. enterprise), can be very heterogeneous regarding how their data is structured, labelled and described (e.g., naming conventions for JSON keys, XML tags, or CSV headers), exhibiting considerable variety even for data with substantially similar attributes [112]. The reconciliation of semantic and structural heterogeneities in raw data is a critical preparatory step for storing and retrieving the data quickly and cost-efficiently, and for aligning the data from different sources so that all types relevant to a single analysis requirement can be combined and analysed together. Manually handling heterogeneity reconciliations would pose a huge burden on PDL users. Despite efforts in the fields of semantic web and data integration for automating the reconciliation process [93,160,161], existing approaches, most of which require optimised parameter tuning and expertise-based configurations to cope with the heterogeneities of data [112], cannot be "directly" adopted in PDL.

The second challenge is *schema evolution* [17,48], which refers to the problem of handling unexpected changes in the schema and structure of the data ingested. Big data is often subject to frequent schema evolutions, which would cause query executions over the unified views to crash if not dealt with [162]. Handling schema evolutions is a non-trivial task, and the common practice normally involves employing skilled manpower. Schema evolution has been a known problem in the database community for the last three decades [163] and has become frequent and extensive in the era of big data, yet it has not been addressed effectively [52,93,110].

Besides the above challenges, we believe that the simplicity offered by the data ingestion approach described in Chapter 3 can incentivize common users to synchronize PDL with a large number of input sources as a natural consequence of the personal data conception discussed in section 1.2. This belief is largely predicated on the observation that the information needs and requirements of users may change to cope with data *discovery* [I37], *foraging* [2], *orienteering* [156], *searching* [157], among other related behaviours in which common users purposefully seek new information or serendipitously encounter it in the course of their daily activities, therefore they are highly likely to continue adding new input sources to PDL over time to meet these requirements. Classical integration solutions are designed to handle few input sources and are generally intolerant to the continuous addition of sources [158,159]. In most cases, adding new input sources to the system implies a manual revision of the integration pipeline to ensure the integration validity [92].

To address these issues we propose SemLinker, an ontology-based data integration system as principal component of MMF. SemLinker adopts an automatic approach that only operates on the schema metadata level without involving physical transformation on the data during integration, therefore it preserves PDL stored data in its native schemas, formats, and structures, while, at the same time, allowing the data to be integrated and utilised. A critical factor in the design of SemLinker is the "ease" in which data can be queried and analysed by PDL users to incentivize them deriving value from their own data with minimum time and efforts. Therefore, the research requirements that are met in SemLinker are as follows:

- Addition and removal of input data sources with no impact on existing integrations.
- Flexibility of incorporating integrated (semi)structured data with unstructured data (e.g. free-text documents, images, and other complex data).
- Adaptability to the changing information needs and requirements of users, that is, allowing the common user to flexibly personalise unified views over PDL stored data, whilst persisting seamless integration of data upon any view changes.
- Handling schema evolutions in input data sources, that is, keeping all past data automatically integrated with newly ingested data that exist with evolving changes on the schema, format, or structure level. Additionally, providing detailed timeline describing the schema evolutions of any input source for data provenance purposes.
- Formal accessibility and rapid query executions over large volumes of diverse raw data ingested from heterogeneous input data sources and stored in untransformed state.

4.2 Related work

The classical approach to data integration can be viewed as a staging architecture that consists of two parts [164]: *wrappers* and *mediators*. A wrapper wraps the data of an input source and speaks the dialect of its physical (or *local*) schema [92]. A *mediator* maintains a mediated schema and understands how *mappings* between the mediated and local schemas are corresponding. The mediator may follow a *virtual* or *materialized* approach. In the former, data is physically residing in its original sources [165], whereas in the latter, data is pulled from its sources and centralised into a central storage space [166] – which is the case of PDL. The mediator facilitates formulating queries in terms of the mediated schema [39]. An input query submitted to the mediator is processed through *reformulation procedure* before it can be executed on the integrated data. Query reformulation involves translating the input query into sub-queries in terms of the local schemas of the integrated sources and passing them to wrappers, which accept the sub-queries, execute them on the sources, extract the answers, and finally send them as query results back to the mediator. Wrappers may also apply transformations on the retrieved data in order to return the query results into a unified representation [40]. Mappings are important in any data integration approach; they express the correspondence between the mediated schema and the local schemas of the integrated sources [154], this enables the mediator to accurately translate an input query into downstream sub-queries. Early research efforts attempted to explicitly formalise mappings, the work in [167] gives an overview of these efforts. When mappings are formally crafted and described by specifications, they can serve as a basis for modelling (or metamodeling) based data integration, where the wrapper and mediator code could be automatically generated from a given specification. Collet et al. [168], and Singh et al. [169] were the first to use description logics [170] for expressing relationships between sources on the model level. Following these studies, a large body of research has been undertaken, the work in [92] surveys many notable contributions in this regard.

Lenzerini introduces in [39] a theoretical framework for integrating a set of heterogeneous data sources based on the schema metadata associated with their local schemas. The framework's integration workflow follows the mediation architecture and thus maintains mediated schema, and formal mappings between the mediated and the local schemas. The concept of ontology was proposed as an efficient description tool for expressing the mediated schema and is called global schema (or global ontology). The global schema materializes integrated unified views over the data collected from the integrated sources by means of mappings. On a formal side, the framework defines two semantic perspectives for mapping formalisation: *Global-As-View* (GAV) and *Local-As-View* (LAV). In a GAV integration scenario, any single element g of the global schema is defined as a view on the local schemas of the integrated sources, i.e., $q_s \sim g$. This reflects the classical integration of the mediation architecture. In a LAV integration scenario, an element s of the local schema is defined as a view on the global schema, i.e., $s \sim q_g$. This reflects the integrated data as a partial description of the real world uniform which the sources capture. Such reflection may be incomplete, erroneous, or even inconsistent, which gives LAV greater flexibility than GAV in integrating real world data. From unified view perspective, the symbol q represents an input query either over the global ontology and down to the local schemas (GAV), or vice versa (i.e. LAV). In description logic terms, the mapping ~ is a relationship that may resemble $=$, $\subseteq$, and so forth. For example, the query G: q_s~ q_g means that the result set of the query q_s is equivalent to (or a subset of) the result set of the query q_g for all the valid input sources participating in the integration at hand. It is well-known that query rewriting in GAV approach is easy and fast in most scenarios [92,155,171,172]; any input query over the global schema needs just to be unfolded in such a way that the elements of the global schema in the query formulation are substituted with the corresponding query q_s over the local schemas of the integrated data. The downside here is that: GAV is very sensitive to schema evolutions, changes in the local schema of a particular data under integration entails immediate reactions to revise the global schema as well as its mappings, otherwise unfolded queries on a modified local schema may crash due to schema mismatching issues. In contrast, query rewriting in LAV is more expressive and allows to perform vital operations during reformulation (e.g. grouping, constraint enforcements, etc.). The down side of LAV approach is that query rewriting is moderately slow and requires reasoning [171]. Several algorithms have been proposed to overcome these issues [90]. There are also suggestions to combine both GAV and LAV into unified architecture to benefit from the merits of each approach. An example of such proposals is the Global-Local-As-View (GLAV) approach described in [172].

Many current state of the art ontology-based data integration systems follow Lenzerini framework [39] to integrate (semi)structured data that are collected from heterogeneous data sources [52, 173,174,175]. Although these systems may deliver effective and efficient data integration performance in many use cases, they typically require continuous human intervention to supervise the task of discovering mappings between the global ontology and the local schemas [110, 161, 176], which is a laborious and time-consuming itself as it requires expertise in schema matching techniques. Furthermore, these systems favour static integration workflows, where any changes in the global ontology or the metadata of the integrated local schemas imply a high degree of manual efforts to (re)configure the mappings [52, 161,183].

With the increasing popularity of DL in the big data landscape, metadata is becoming of immense importance for BDI research [177], and embedding model-based integration into MMFs is currently an active research topic. In Chapter 2 we described GEMMS [90] as MMF for a DL system called Constance. GEMMS implements a model-based incremental approach for interactive personal data integration in life science fields. The first step in this approach is extracting schema metadata from the ingested data. Metadata extraction here is manual task that requires user's attention. The authors propose a semiautomatic tool to help the user in that regard. Once data is associated with appropriate metadata, it is modelled to a common metamodel that is a simple variant of the Entity-Relationship (ER) model, where data entities are resembled as objects and their attributes as object properties. Although the approach is theoretically capable of reconciling semantic heterogeneities between the stored data, and tolerating the volume and variety aspects of the 3Vs model, its architecture suffers multiple drawbacks: first, it reconciles structural heterogeneities through physical data transformations to fit ER modelling, which implies altering the native schemas and structures of the data and posing constraints on the ingestion process; secondly, it is very sensitive to emerging changes in the raw data schemas, therefore the velocity aspect can quickly crash the modelling process; thirdly, the GEMMS literature does not describe how the integrated data can be systematically accessed and queried. Kayak [109], a generic framework for managing data lake content through metadata-based data preparation and wrangling, is a case similar to GEMMS. Although it promises integration and querying capabilities, the underlying integration process and modelling are not clearly described. The work in [282] introduces a generic approach that allows the user to extract the local schema of the data ingested from any input source connected with DL. This approach utilises a schema management tool called Darwin [179] for scanning the data stored in DL, and in cases where the data exists with multiple local schema versions due to historical evolutions, each schema version is extracted, and eventually, all the versions are displayed to the user. The approach enables the user to define a global ontology and draw ontological information (concepts and properties) to annotate the schema elements in each version, thus offering an easy solution to reconcile the heterogeneities originate from the changes between any given schema and its previous version. Although this approach focuses on the problem of schema evolution, it is not equipped with data integration capability, but rather integrations can be performed manually by investigating similarities between various schema elements annotated with the same ontological information and record them as mappings for later use. Furthermore, schema detection and handling is not performed in real-time (i.e. upon ingestion) instead the data must first be stored in DL backend in good amount in order to begin schema analysis procedures, this means, the data consumers operating on the schema level will face query crashes and schema mismatching issues until the new schema changes are propagated in the system. Apache Atlas [104] allows the user to provide global schemas in the form business-oriented ontologies which then can be utilised to annotate the HDL stored data with semantic metadata drawn from these ontologies to describe their physical schemas. Such annotations are stored in the MMF's repository. For integrating data, the user must hire a suitable integration system, and consult the metadata repository for guiding the integration system and specifying how data should be integrated. Atlas provides basic support for the schema evolution problem. The user can integrate Apache Avro [180] in Atlas in order to detect schema evolutions and react to them based on sets of rules provided by the user in the form of *avro files*[30] and through a process called schema resolution.

In [52], Nadal et al. propose a metadata-driven system for integrating heterogeneous JSON and XML data in DL systems and governing local schema evolutions. This system follows Lenzerini's framework and is based on a BDI-oriented Metamodel (ontology) that consists of two levels: top level abstract abstraction that is expressed by Web Ontology Language (OWL) [184], and low level collection of RDF triples describing the local schemas of the data sources under integration. The top level of the BDI ontology offers unified views over the local schemas in the low level. The system offers an automatic algorithm that extracts an RDF representation of the physical (local) schema of a newly added data source after examining a few samples of its offered (JSON/XML) data. An extracted representation is expressed as a set of RDF triples and stored in the low level. A data specialist, called steward, is responsible of providing mappings between the RDF representations in the low level and their corresponding counterparts in the high level. If the physical schema of a particular data source evolves, the steward is notified, and a manual remapping then takes

[30] http://avro.apache.org/docs/current/spec.html#Schema+Resolution

place to ensure the consistency of the correspondences between the top and low levels. The system allows the user to formulate SQL-like queries over the top level of the integration ontology. Each input query is unfolded by rewriting it into internal queries that are executed on the system's backend(s). The integration ontology also allows the user to flexibly handle evolutions in the local schema of a given data source by maintaining multiple schemas that correspond to different versions of the local schema extracted from that source.

The shortcoming of existing BDI solutions for DL is that they inherently exhibit the same drawbacks found in traditional data integration, such that, raw data (meta) modeling remains an expensive task that requires expert user supervision [110, 181], furthermore, the schema evolution problem and its impact on data access, processing, integration, and analysis in a DL system is often overlooked and its suggested solutions largely remain manual [52, 182,183]. Rahm states in [110] that most BDI proposals are limited to a few data sources, and analytics over a large volume of heterogeneous data ingested from various autonomous data sources is only possible with the availability of a holistic data integration solution that: (i) should be fully automatic or require only minimal manual interaction, and (ii) should make it easy to add and use additional data sources and automatically deal with frequent changes in these sources (i.e. velocity). SemLinker, as a data integration system, shares many features and functionalities with other solutions. However, as a solution for PDL whose users are typically casual and unskilled, it needs to be in agreement with Rahm's automation proposal and isolates its users from the technical details imposed by the integration process, thus meeting all the requirements listed in section 4.1. Our proposed automations are implemented in the following operations:

- Management of semantic and schematic metadata; extraction, annotation, storage, and maintenance.
- Management of mappings between the system's global ontology and the metadata denoting the local schemas of the data sources added to PDL. In other words, automating schema matching, schema mapping, and metamodeling tasks.
- Management of schema evolutions on the local schema level, and automatic responding to views adjustments on the global ontology level.

SemLinker supports personal data analytics in PDL by accepting direct queries over its metadata repository. Thus, management functionalities over personal data such as summarization, analysis, and insight discovery (informatics) can be readily performed.

4.3 SemLinker architecture overview

The architecture of SemLinker consists of *Global Schema Layer* (corresponding to the metamodel level), *Local Schemata Layer* (corresponding to the model level), and the relationships between these layers (corresponding to formal mappings over instance level). The global schema layer consists of the global schema ($\mathcal{G}$), and the query engine for formulating queries over $\mathcal{G}$. The local schemata layer consists of the schemas repository (S), and schema metadata extraction, and mapping and management components. As an ontology-based integration system, SemLinker is conceptually based on the theoretical framework proposed by Lenzerini [39]; we formally define the system as follows:

Definition 4.1: *SemLinker is a triple $\langle \mathcal{G}, S, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global schema, S is a set of local schemas corresponding to n data sources, $S = \{S_1, S_2, \dots, S_n\}$, and $\mathcal{M}$ is a set of mapping assertions, such that, for each S_i there is a set of mappings between g and S_i, $g \in \mathcal{G}$, $1 \leq i \leq n$, in the form: $p \rightarrow a$, where attribute $a \in S_i$ and property $p \in g$.*

The system's global schema $\mathcal{G}$ is modeled as a global ontology and is described using OWL. It extracts and maintains machine-readable metadata that describes the physical schema details of each data source connected with the PDL, and specific semantics about its available data, we refer to such metadata as *local schema*. Any local schema in PDL is described using RDF and is stored in the schemas repository S. SemLinker is responsible for automatically mapping the local schema S_i of the data ingested from the data source i to a semantically corresponding concept in the global ontology $\mathcal{G}$ in the GAV formulation approach. Accordingly, mappings provide a metadata model that enables SemLinker to systematically annotate the ingested data, and allows the user to pose queries over $\mathcal{G}$ which serves as a GAV abstraction layer over S and its associated raw data. With this metamodeling in place, the stored data are integrated on the metadata level; no manual effort is required to reconcile the heterogeneities in the physical schemas and structures of the raw data.

Figure 4.1 depicts a high-level overview of SemLinker architecture. The figure illustrates the flow of ingested (semi)structured raw data from the ingestion layer, to the lineage manager, then to SemLinker, and finally to the storage layer. When SemLinker finishes meta-processing the input raw data entity at hand, the metadata outputs (if any) are stored in the local schemata layer, whereas the data entity in its raw form as well as its associated "metadata information" are dispatched to the storage layer to be stored in the "unified repository" and the "linkage table" respectively (see section 3.5.3 for full details). The data consumers can interact with SemLinker metadata through PDL's access layer. The details of meta- processing and metadata usage workflows concerning SemLinker to be construed in the following sections.

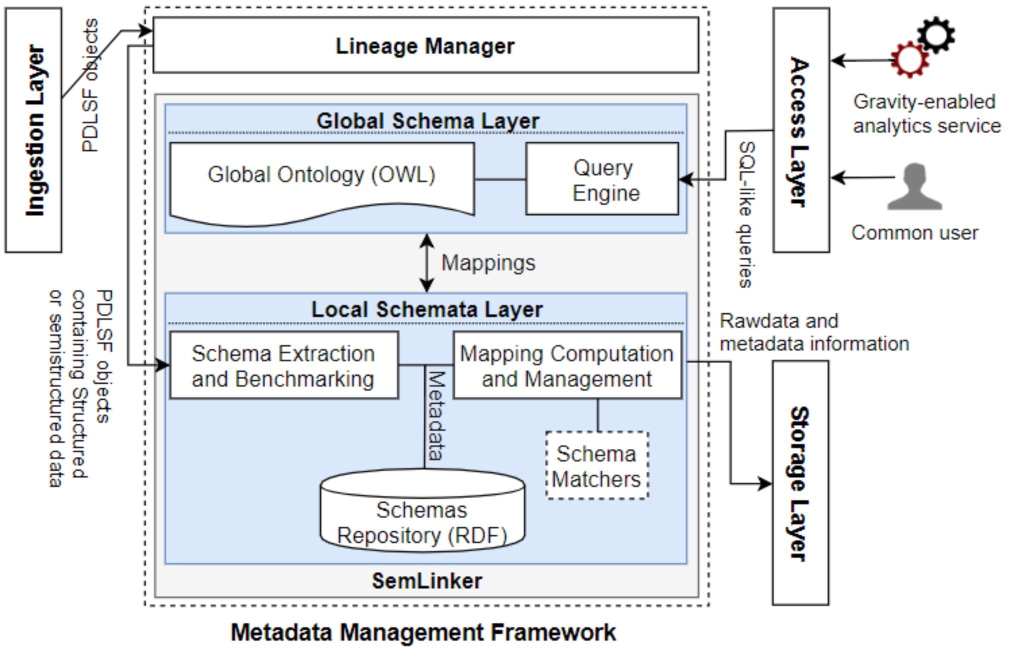
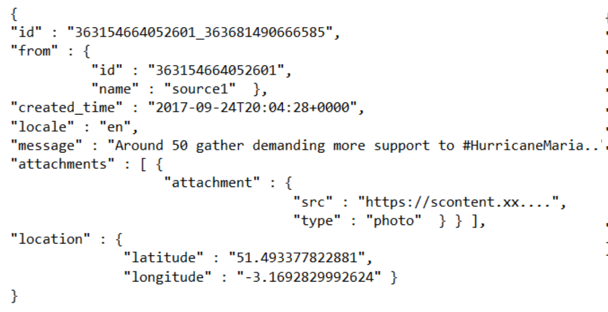


Figure 4.1– Overview of SemLinker architecture

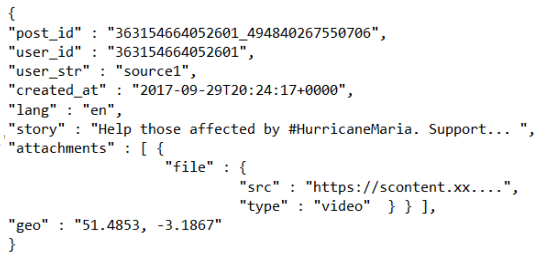
Here we introduce a personal data example comparable to a real-life scenario to give a realistic view of the challenges that a BDI system like SemLinker must meet. Figure 4.2 lists four personal data entities representing social media feeds posted by the PDL user on Facebook and Twitter and ingested by the PDL through the available API of each source

(Facebook Graph API [265], and Twitter Streaming API [266]) with evolved schemas. Although these entities exist in self-describing formats and contain abstract schema metadata implicitly encoded in JSON keys and XML tags, they suffer semantic and structural heterogeneities, even for the instances ingested from the same data source. For example, the JSON keys in Facebook data entities (Figure 4.2a and Figure 4.2b) are expressed with different strings and exist in different structures (see "*location*" and "*geo*" keys). Similarly, Twitter data (Figure 4.2c and Figure 4.2d) also exist in different data formats. The example serves as a reference point for later sections on SemLinker discussions.



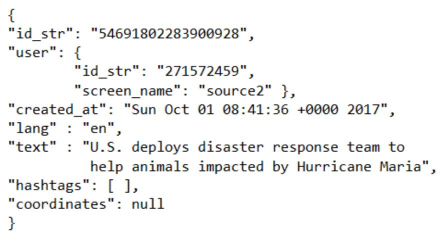
```
{
"id" : "363154664052601_363681490666585",
"from" : {
          "id" : "363154664052601",
          "name" : "source1"  },
"created_time" : "2017-09-24T20:04:28+0000",
"locale" : "en",
"message" : "Around 50 gather demanding more support to #HurricaneMaria..'
"attachments" : [ {
                   "attachment" : {
                                    "src" : "https://scontent.xx....",
                                    "type" : "photo"  } } ],
"location" : {
              "latitude" : "51.493377822881",
              "longitude" : "-3.1692829992624" }
}
```

(a) Facebook data in schema v.2.9



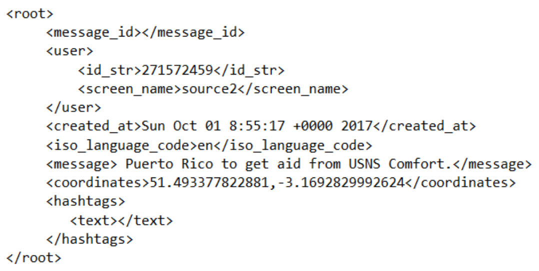
```
{
"post_id" : "363154664052601_494840267550706",
"user_id" : "363154664052601",
"user_str" : "source1",
"created_at" : "2017-09-29T20:24:17+0000",
"lang" : "en",
"story" : "Help those affected by #HurricaneMaria. Support... ",
"attachments" : [ {
                    "file" : {
                               "src" : "https://scontent.xx....",
                               "type" : "video"  } } ],
"geo" : "51.4853, -3.1867"
}
```

(b) Facebook data in schema v.3.0



```
{
"id_str": "546918022839009 28",
"user": {
        "id_str": "271572459",
        "screen_name": "source2" },
"created_at": "Sun Oct 01 08:41:36 +0000 2017",
"lang" : "en",
"text" : "U.S. deploys disaster response team to
          help animals impacted by Hurricane Maria",
"hashtags": [ ],
"coordinates": null
}
```

(c) Twitter data in schema v.1.1



```
<root>
     <message_id></message_id>
     <user>
         <id_str>271572459</id_str>
         <screen_name>source2</screen_name>
     </user>
     <created_at>Sun Oct 01 8:55:17 +0000 2017</created_at>
     <iso_language_code>en</iso_language_code>
     <message> Puerto Rico to get aid from USNS Comfort.</message>
     <coordinates>51.493377822881,-3.1692829992624</coordinates>
     <hashtags>
        <text></text>
     </hashtags>
</root>
```

(c) Twitter data in schema v.1.2

Figure 4.2– Raw data from two social media platforms (Facebook and Twitter)

4.4 Global schema layer

The global ontology $\mathcal{G}$ serves two purposes: tagging input sources upon their addition (synchronization) in PDL to reflect the *type* semantic information of their personal generated data, and to form an indispensable basis in the form of query-able format-agnostic unified views that allows executing uniform queries over the raw data ingested from each synchronized input source. An ideal global ontology is a comprehensive and standardized ontology that provides semantic coverage and interoperability across a vast range of domains [177]. For this reason, we initiate $\mathcal{G}$ as an OWL implementation[31] of SCHEMA [48]. SCHEMA is a lightweight and well-curated vocabulary that consists of abstract concepts common across many domains and is used as a backbone schema for annotation in many large-scale knowledgebase projects, such as *Wikipedia*[32], *DBPedia*[33], and *Google Knowledge Graph*[34]. Such initiation is beneficial in supporting the semantic interoperability between a multitude of data sources that possibly exist in different domains. The disadvantage however is that SCHEMA abstract concepts can be too generic and require more specificity to support concise metamodeling and integration.

To balance between the conceptual abstraction and the semantic specificity, we enable $\mathcal{G}$ *extensibility.* The elements of $\mathcal{G}$ may be extended by adding new properties to the current set of properties of a concept $g \in \mathcal{G}$, to increase its coverage over elements in local schemas at the local schemata layer. g may also be extended by adding a new subordinate concept to it. `rdf:type` and `rdfs:subClassOf` are used for importing new and more specific concepts. To comply with $\mathcal{G}'s$ structure, the newly added concept must be associated with a set of properties (declared using $\mathcal{G}$*:hasProperty*) and each property is of a certain primitive data type that is strictly reused from the XSD vocabulary [185] (declared using $\mathcal{G}$*:hasDatatype*). Figure 4.3 depicts an example of extending *SocialMediaPosting*, a concept in $\mathcal{G}$, with `Feed`, a more specific concept imported from the SIOC vocabulary [186]. The extension taking place is to support a unified view over data ingested from social media data sources. The extended concept `Feed` is linked to `SocialMediaPosting` using `rdfs:subClassOf`, and is described by a set of properties imported from the DCMI [187] and WGS84 [188] vocabularies. The required RDF data to implement such an extension are automatically generated by SemLinker and are added to the $\mathcal{G}$ ontology.

[31] https://github.com/schemaorg/schemaorg
[32] https://meta.wikimedia.org/wiki/Wikidata/Notes/Schema.org_and_Wikidata
[33] https://wiki.dbpedia.org
[34] https://developers.google.com/knowledge-graph/

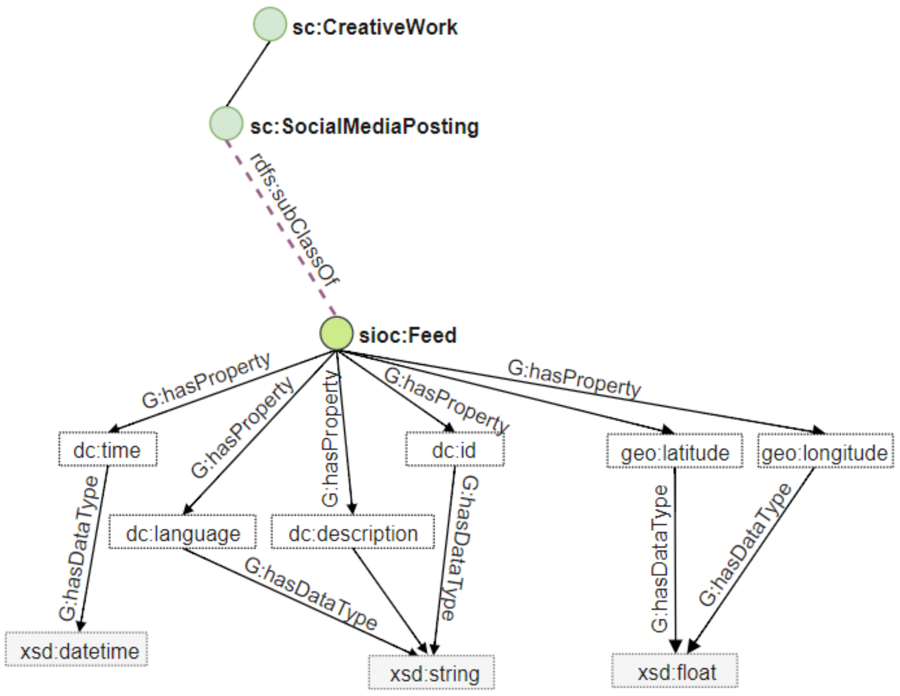


Figure 4.3– Example of concept extension in $\mathcal{G}$

4.5 Local schemata layer

The schemas repository S is a principal component in the local schemata layer that stores the set of local schemas corresponding to different input sources added to SemLinker over time. Each local schema is stored in S as a data graph that contains machine-readable metadata in the form of RDF triples which describe the physical schema details of the data ingested from each input source synchronized with PDL, and how various elements in each schema are corresponding to $\mathcal{G}$, i.e. the schema mappings. In this section we explore how local schemas in SemLinker are automatically, extracted, mapped to $\mathcal{G}$, stored in S, maintained over time, and used for generating unified views.

4.5.1 Automatic data source addition

SemLinker, as an MMF component, is physically isolated from the ingestion layer, and its single point of input is the lineage manager (see Figure 4.1). Such separation of concern enables the latter to forward raw data with associated lineage metadata that are necessary for optimising the metadata management and storage processes.

Every ingested (semi)structured data entity is dispatched to SemLinker with the following artefacts: *key*, *source*, *type*, and *mime*. As indicated earlier, *source* specifies the unique identifier of the input source i (in the form of URI) from which the data entity at hand is ingested. *type* specifies the global concept g,$g \in \mathcal{G}$ representing the type abstraction of i's data. Both artefacts are originally obtained from the user during the deployment of the ingestion agent concerning i (see section 3.3.4). *key* represents the unique identifier that is automatically generated by the lineage manager and permanently associated with the entity during its lifecycle inside PDL. *mime* specifies the physical format of the entity (e.g. JSON, XML, CSV, etc.) that is discovered by the lineage manager during PDLSF parsing. For each received raw data entity, SemLinker checks whether the associated *source* already exists in the schemas repository, if not, then this is an indication that the data source i, is newly added input source to PDL. In this case, SemLinker initializes a new empty RDF graph that represents i's local schema metadata, denoted as S_i. Subsequently, it tags S_i with the concept g in *type*, so that it reflects the underlying type semantics of the data typically ingested from i. Local schema tagging is normally modeled as an RDF triple, and follows the pattern:

$$\langle S_i \; M{:}\, isInstanceOf \;\, g \rangle$$

For example, if i is Facebook, and g is `Feed` then the RDF interpretation of the local schema tagging is asserted as

$$\langle S_{Facebook} \; M{:}\, isInstanceOf \;\, \mathcal{G}{:}\, Feed \rangle$$

To this end, i is automatically added to SemLinker, and its local schema S_i is to be further processed by downstream algorithms in the system's workflow to maintain appropriate semantic and schema metadata that will always be utilised to automatically annotate any data entities ingested from i in the future. Metadata management post-processing in SemLinker involves dispatching the processed data entity to the storage layer with two identifiers: *key*, and an internal identifier specifies the current *source schema* of i (see next section). The storage of the entity and identifiers in the storage layer is construed in section 3.5.3.

The lineage metadata added to SemLinker inputs offers the following advantages:

- Enabling the user to add new input sources and plug new ingestion agents in the ingestion layer without the need to notify SemLinker.
- Allowing the system to ignore the technical details of the ingestion agents plugged in the ingestion layer by treating input sources as Linked Data [97] resources with fixed URIs. This is useful in cases where the user replaces the ingestion agent concerning certain input source with another agent and without the need to notify SemLinker about such change due to its autonomy from the ingestion environment.
- Providing all the necessary input information to the downstream algorithms in SemLinker's workflow in a disciplined way, rather than relying on manual inputs, which may be erroneous or time-consuming for the PDL user.

4.5.2 Local schema redefined

The physical schema of any input source is subject to changes and updates [162,190,282]. In the example depicted in Figure 4.2, schema evolution is observed at both the semantic level (data attribute renaming, e.g. *"message"*$\longmapsto$*"story"*, and *"text"*$\longmapsto$ *"message"*), and the structural level (data format changes, e.g. JSON$\longmapsto$XML, and attribute changes, e.g. casting the JSON object "*location*" in Figure 4.2a into the simple attribute "*geo"* in Figure 4.2b). SemLinker takes a novel, automatic approach to handle the schema evolution problem. In this approach, the RDF representation of the local schema of an input source is regarded as *dynamic*. It contains a changeable set of subgraphs, each of which represents an evolving version of the schema and is called a *source schema*. A *Schema Extraction Algorithm* is used to extract source schemas automatically, and a *Mapping Computation Algorithm* is responsible for mapping them to the global ontology. A formal definition of the local schema of a dynamic feature is given below.

Definition 4.2 (Local Schema): *The Local schema $S_i \in S$ is a dynamic set of source schemas corresponding to m physical schema evolutions in the data ingested from the* *data source i,* $S_i = \{S_{i1}, S_{i2}, \dots, S_{im}\}$*. For each* $S_{ij} \in S_i$*,* $1 \leq j \leq m$*, there is a set of mapping assertions* $\mathcal{M}$ *between* S_{ij} *and* $g \in \mathcal{G}$ *of the form:* $p \rightarrow a$*, where attribute* $a \in S_{ij}$ *and property* $p \in g$*.*

The system accepts a data entity that is ingested from its source's through an endpoint which is typically associated with a release version. Analysis of the data entity's physical schema is needed to obtain its source schema S_{ij}, where i is the input source's URI, and j is i's endpoint release version. SemLinker (fully/partially) then maps S_{ij} to the tagging concept g in the global ontology, stores it in the underlying graph of S_i, and uses it to integrate i's data with other raw data stored in PDL. Furthermore, S_{ij} is regarded as a *benchmark* and is used to run schema checks on any new data entities ingested from i. A schema check may fail, and when the number of failures reaches a predefined threshold, the system infers that data source i has released its endpoint with a newer version. Consequently, a new evolution has occurred in the physical schema of i's data, and the system must augment the local schema S_i by constructing a new source schema, say S_{ik}, that is also mapped to g and added to S_i, so that S_{ik} is utilised to integrate any new data entities ingested from i with the release version k, meanwhile utilising S_{ij} to maintain backward integration support (compatibility) for the data entities that have already been ingested from i with the release version j. The procedure for augmenting the local schemas upon schema evolutions in the endpoints of their data sources is automatically repeated to keep up-to-date metadata about the physical schema of the data entities ingested from each data source.

4.5.3 Automatic schema metadata extraction

The Schema Extraction algorithm automatically extracts source schemas from data entities (see Algorithm 1). It takes as input a data entity F ingested from a data source i, with release version j, and a *mime* string specifying the format type of F. F is assumed to conform to a known format specification [189], and its structure consists of a mix of flat and complex attributes, each of which has a *label* and a *value.*

Table 4.1 – RDF schema representation vocabulary ($\mathcal{S}$)

Element	Type		Details
	Element	Semantics	
$\mathcal{S}$:`Attribute`	Class	Type	Describes flat attribute, e.g., JSON key-value pair, single tabular column, XML tag.
$\mathcal{S}$:`hasAttribute`	Property	Relation	Links flat attribute with its semantic type class.
$\mathcal{S}$:`Object`	Class	Type	Describes complex, e.g., JSON object, primary or CSV header column, etc.
$\mathcal{S}$:`hasObject`	Property	Relation	Links complex attribute with its semantic type class.
$\mathcal{S}$:`Collection`	Class	Type	Describes enumerator ranging from list, ordered list, array, collection, etc.
$\mathcal{S}$:`hasCollection`	Property	Relation	Links enumerator attribute with its semantic type class
$\mathcal{S}$:`hasFormat`	Property	Functional	Internally assigned attribute reflects data format, e.g. JSON, XML, CSV, etc.
$\mathcal{S}$:`isComposedBy`	Property	Functional	Internally assigned attribute to control virtual transformation.
$\mathcal{S}$:`isDecomposedFrom`	Property	Functional	Internally assigned attribute to control virtual transformation.

The algorithm operates on the structure level of F and extracts its RDF representation S_{ij} that consists of nodes and relationships between them. Each node in S_{ij} describes a specific element (attribute) in the physical schema of F and is associated with three constructs: *Identifier*, *Semantic Type*, and *Relation*. The algorithm assigns a value to each node and constructs *Identifier* using the URI of the input source and the release version j as base values. *Semantic Type* specifies the semantic class of the node, and its value can be any of the concepts listed in Table 4.1. *Relation* refers to a relation between a pair of nodes, and it can be any of the properties listed in Table 4.1.

```
1   function InitializeGraph(F, i , j , mime)
2     root ← i + '/' + j
3     format ← root + '/' + mime
4     $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(format, $rdf{:}type$ , $\mathcal{S}{:}Attribute$ )
5     $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(root, $\mathcal{S}{:}hasFormat,$ format)
6     GenerateGraph(F , Root)
7   end
8   function GenerateGraph(F, ParentId)
9     foreach (label,value) ∈ $F$ do
10      NodeId ← ParentId + '/' + label
11      NodeType ← Type(value)
12      $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(NodeId , $rdf{:}type$ , NodeType)
13      if NodeType = $\mathcal{S}{:}Attribute$ then
14        $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(ParentId , $\mathcal{S}{:}hasAttribute$ , NodeId)
15      else
16        if NodeType = $\mathcal{S}{:}Object$ then
17          $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(ParentId, $\mathcal{S}{:}hasObject$ , NodeId)
18          GenerateGraph(value,NodeId)
19        else
20          if NodeType = $\mathcal{S}{:}Collection$ then
21            $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(ParentId, $\mathcal{S}{:}hasCollection$ , NodeId)
22            GenerateGraph(value,NodeId)
23      end if
24    end foreach
25  end
```

Algorithm (1)– Schema extraction algorithm

The algorithm has two procedures: *InitializeGraph()* and *GenerateGraph()*. The first starts with specifying the given URI and the release version j as the root of S_{ij} (line 2); the auxiliary function *ToRDF()* adds format attribute (the input *mime* corresponds to `format`) to S_{ij} as one of its nodes (lines 3 and 4); *ToRDF()* then specifies the relationship (*:hasFormat*) between the format node and its parent node (line *5*). At this point, the source schema S_{ij} is initiated. The procedure then invokes *GenerateGraph()* and passes F and the root of S_{ij} to it. *GenerateGraph()* constructs S_{ij} through a series of iterations and recursive calls over the physical schema of F. In each call, the procedure takes a label-value pair from F and *parentId* (the URI) as new input, creates a node in S_{ij} corresponding to the passed label, and links the node to its parent node (*parentId*). The initialization and linking of any node is modeled as the RDF triples *(NodeId rdf:type S:Type)* and *(ParentNodeId S:relation NodeId),* respectively (line 12). Next *ToRDF()*, based on the type of the node, appends these triples to S_{ij} (lines 6-15).

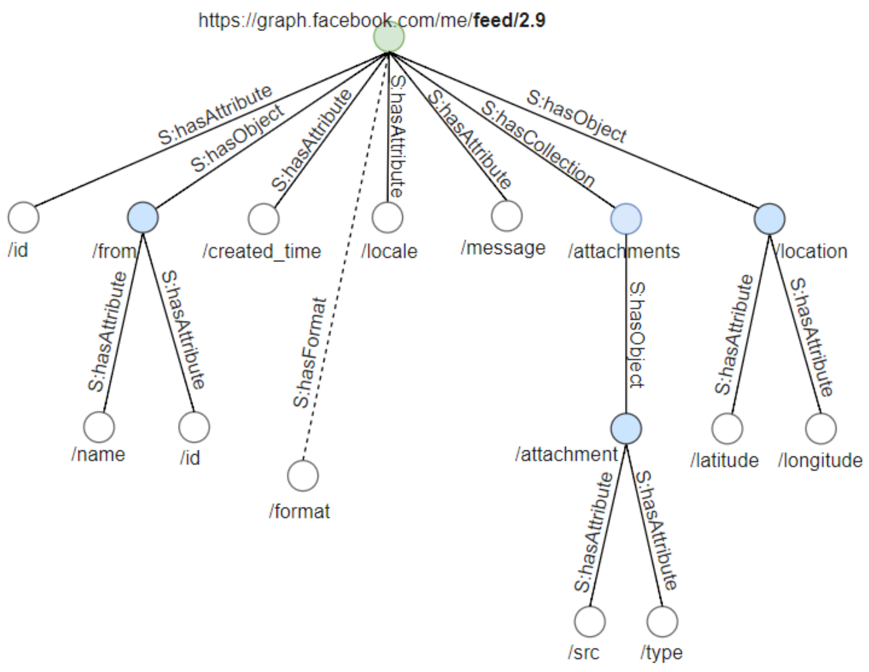


Figure 4.4– A source schema extracted using the schema extraction algorithm

Depending on the complexity of F's structure, a label-value pair may represent a flat attribute in F (e.g. "id" key in Figure 4.2a), in which case, the node type obtained from the auxiliary function *Type()* is *$\mathcal{S}$:Attribute*, and the corresponding node is linked to its parent node using *$\mathcal{S}$:hasAttribute* relation, and the algorithm moves to the next label-value pair. Conversely, the current label-value pair may correspond to a complex attribute (e.g. "location" embedded object in Figure 4.2a). In this case, the type obtained from *Type()* is either $\mathcal{S}$:`Collection` or $\mathcal{S}$:`Object`, and the node is linked to its parent node using one of the relations $\mathcal{S}$*:hasCollection* or $\mathcal{S}$*:hasObject*, and subsequently the node's identifier and value are passed to the recursive procedure *GenerateGraph()*.

Figure 4.4, as an example, depicts the graph-based representation of the source schema extracted from the data sample given in Figure 4.2a. The first node in the graph is created as a leaf node because the first label-value (JSON key "*id"*) is a simple attribute in F. Its identifier is Facebook relative URI[35]. A node maybe embedded in another node, such is the case with the node labelled '*latitude'*, which serves as one of the flat attributes of the object '*location'*.

4.5.4 Mapping computation and management

Once a source schema is constructed, it needs to be mapped to the global ontology. A mapping is a relationship specifying how an element structured under one schema (i.e., the source schema) corresponds to an equivalent element structured under the global ontology (i.e., $\mathcal{G}$) [39]. Mappings may be discovered either *implicitly* or *explicitly*. In SemLinker, because the global concepts of $\mathcal{G}$ are predefined independently from the input sources, it is likely that a source schema is semantically *incompatible* with the concepts of $\mathcal{G}$, and therefore no implicit mappings can be directly discovered between a source schema S_{ij} and a tagging concept g. Typically, computing mappings between a source schema and a tagging concept involves specifying the semantic types of the source schema elements, i.e., labelling each schema element with a semantically equivalent property in the tagging concept [93]. However, semantic labelling alone is not sufficient [83], and a precise mapping computation process requires an extra step that specifies how the elements of a source schema should be organised in accordance with the structure of its tagging concept so that the two constructs become semantically compatible and ready for mapping. This 'extra step' is often missed in systems that automate mappings discovery [161,176,181, 191,192] and is commonly expected to be dealt with manually [83]. SemLinker uses a two-step mapping approach that not only does the explicit mappings, but also performs the

[35] https://graph.facebook.com/me/feed/2.9/id

'extra step' automatically. The two steps are Schema Matching (SM) and Virtual Transformation of Source Attribute (VTSA).

Mapping Algorithm

The mapping algorithm (see List 2) takes as inputs a tagging concept g, a source schema S_{ij}, and a threshold t. It takes two steps, SM and VTSA, to compute mappings between properties and source attributes. Mappings are established as RDF triples, where each mapping triple has the pattern *(p M:mapsTo a),* $p \in g$, $a \in S_{ij}$. Such modeling offers the flexibility of allowing multiple source attributes of multiple source schemas to be mapped to a single property. The source attributes mapped to the same property are considered semantically equivalent between themselves, so a unified view over them can be automatically represented by the property.

Revisiting the example in Figure 4.2, we see that the Twitter data source is tagged with the concept `Feed`. With the mappings specified below, "*text*" in $S_{Twitter,v1.1}$ (Figure 4.2c) is regarded as semantically equivalent to "*message*" in $S_{Twitter,v1.2}$ (Figure 4.2d).

$$\langle Feed\colon description\ M\colon mapsTo\ S_{Twitter,v1.1}\colon text\ \rangle$$
$$\langle Feed\colon description\ M\colon mapsTo\ S_{Twitter,v1.2}\colon message\rangle$$

Such mappings allow SemLinker to automatically reconcile heterogeneous attributes from different source schemas of the same data source, and the reconciliation can be further obtained by a SPARQL query with the pattern $\langle Feed\colon description\ M\colon mapsTo\ ?\,a\rangle$. In our running example, the result $?\,a = \{S_{Twitter,v1.1}\colon text,\ S_{Twitter,v1.2}\colon message\}$ allows an analysis process to access the values of both data attributes from both versions, $S_{Twitter,v1.1}$ and $S_{Twitter,v1.2}$. This can also be applied to unify semantically equivalent attributes in the source schemas of different data sources as long as they are tagged with the same concept. In our example, if we have both Facebook and Twitter data sources tagged with the same concept `Feed`, then "message" in $S_{Facebook,v2.9}$, "story" in $S_{Facebook,v3.0}$, "text" in $S_{Twitter,v.1.1}$, and "message" in $S_{Twitter,v1.2}$ are all regarded as equivalent.

Schema Matching

For each property p (line 2), the mapping algorithm invokes function *Matcher*() to find the attribute in S_{ij} that is semantically equivalent with p (line 3). *Matcher()* is an interface function that passes the matching task to an external schema matcher that is plugged in the system. Any *appropriate* schema matching approach may be plugged in SemLinker to implement the computation logic underlying the interface *Matcher()*. In section 6.1.2, we discuss multiple relevant approaches to serve that purpose, and evaluate their performance as an integral part of Algorithm (2). Generally, for an attribute a, the matcher computes a score that quantifies the semantic correspondence between a and p. If the score is larger than the threshold t, a and p are regarded as semantically equivalent. When there is more than one property equivalent to the same source attribute, or more than one source attribute equivalent with the same property, the algorithm, before a mapping is established, adjusts the structure of S_{ij} using VTSA. *Matcher()* returns a data structure containing two collection constructs, A and P; while A holds zero or more source attributes, P holds one or more properties. The algorithm decides its next step according to what is returned in the A and P constructs.

- If $A = \emptyset$ (line 4), no match is found and the algorithm proceeds to the next p.

- If $|A| = 1$ and $|P| = 1$ (line 5), one matching attribute a of the source schema is found, so the algorithm establishes a mapping between p and a (line 6).

- If $|A| > 1$ and $|P| = 1$ (line 8), an operation called *Composition* is performed on the attributes of A before establishing mappings (lines 9-17).

- If $|A| = 1$ and $|P| > 1$ (line 18), an operation called *Decomposition* is performed on the attribute a stored in A before establishing mappings (lines 19-28). After the operation, P is skipped from g using the auxiliary function *Skip()* for optimisation purposes (line 29).

While typically information regarding the concept g is abundant, information regarding a specific input S_{ij} is often inadequate [193], especially given that PDL input source often do not expose full-fledged schemas (see section 4.2). When a situation like this arises, SemLinker uses matchers from third parties to handle schema matching tasks. Matchers are classified into three groups, schema-level, instance-level, and hybrid matchers [194]. Schema-level matchers utilise the information available in input schemas to find matches between schema elements. Instance-level matchers use statistics, metadata, or trained classifiers to decide if the values of two schema elements match. Hybrid matchers combine both mechanisms to determine match candidates. Schema matching approaches are constantly evolving, and often they apply other techniques such as dictionaries, thesauri, and user-provided match or mismatch information [193]. After every single property is examined, and mappings between g and S_{ij} are established, the underlying RDF data of the newly constructed S_{ij} are added into the local schema S_i (line 33).

```
1  function ComputeMap($g$ , $S_{ij}$ , $t$)
2    foreach $p \in g$ do
3      $result \leftarrow$ Matcher($p, g, S_{ij}, t$)
4      if $result(A) \neq \emptyset$ then
5        if $result(|A|) = 1$ and $result(|P|) = 1$ then
6          $S_{ij} \leftarrow S_{ij} \cup$ ToRDF($p, M{:}MapsTo, result(A[0])$)
7        else
8          if $result(|A|) > 1$ and $result(|P|) = 1$ then
9            parent $\leftarrow$ Parent($result(A[0])$)
10           newNode $\leftarrow$ parent + '/' + Label($p$)
11           $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(newNode, $rdf{:}Type, S{:}Attribute$)
12           $S_{ij} \leftarrow S_{ij} \cup$ ToRDF(parent, $S{:}hasAttribute$, newNode)
13           $S_{ij} \leftarrow S_{ij} \cup$ ToRDF($p, M{:}MapsTo$, newNode)
14           for $h = 0$ upto $result(|A|) - 1$ do
15             DeleteRelation(parent, $S_{ij}$, $result(A[h])$)
16             $S_{ij} \leftarrow S_{ij} \cup$ ToRDF($result(A[h]), S{:}isComposedBy$, newNode)
17           end for
```

```
18          else
19            if Type($result(A[0])$) $\neq \mathcal{S}: Object$ then
20              UpdateType($S_{ij}$, $result(A[0])$, $\mathcal{S}: Object$)
21            end if
22            for $h$ = 0 upto $result(|P|) - 1$ do
23              newNode ← $result(A[0])$ + '/' + Label($result(P[h])$)
24              $S_{ij} \leftarrow S_{ij}$ ∪ ToRDF(newNode,$rdf: type$, $\mathcal{S}: Attribute$)
25              $S_{ij} \leftarrow S_{ij}$ ∪ ToRDF($result(A[0])$, $\mathcal{S}: hasAttribute$,newNode)
26              $S_{ij} \leftarrow S_{ij}$ ∪ ToRDF(newNode, $\mathcal{S}: isDecomposedForm$,$result(A[0])$)
27              $S_{ij} \leftarrow S_{ij}$ ∪ ToRDF($result(P[h])$, $\mathcal{S}: M: mapsTo$,newNode)
28            end for
29            Skip($result(P)$)
30          end if
31        end if
32      end for
33    $S_i \leftarrow S_i \cup S_{ij}$
34  End
```

Algorithm (2)– Mapping algorithm

Virtual Transformation of Source Attribute

In Figure 4.3, `latitude` and `longitude`, the two properties in the concept `Feed`, correspond *directly* to the flat attributes of the embedded object labelled "*location*" in Figure 4.2a, but correspond *indirectly* to the flat attribute labelled "*geo*" in Figure 4.2b. The relationships between `latitude` and `longitude` and their indirect corresponding source attribute "*geo*", though apparent, can semantically hold only if "*geo*" is transformed into two new source attributes, i.e., $\text{"}geo\text{"} \longrightarrow \langle \text{"}latitude\text{"}, \text{"}londitude\text{"} \rangle$, or vice versa. To preserve the structure of the raw data stored in the lake, we adopt two virtual transformation operations, *Composition* μ and *Decomposition* γ, to work on the schema of the raw data rather than on the data themselves. The virtual transformation operations are based on [172, 195], and they allow SemLinker to virtually map an attribute in a source schema to a property in the global ontology.

Definition 4.3 (Composition μ): *Given a set of source attributes A,* $A = \{a_1, a_2, , \dots, a_k\}$*,* $A \subseteq S_{ij}, S_{ij} \in S_i, 1 < k \le n, n = |S_{ij}|$*, the composition operator* μ_{A,a_μ} *composes A into a single virtual attribute* a_μ*.*

The mapping algorithm uses the condition ($|A| > 1$, and $|P| = 1$) as the heuristic rule to compose a subset of source attributes $A, A == \{a_1, a_2, , \dots, a_k\}$ as a single new attribute a_μ and adds it to the S_{ij}. Since a_μ is a new source attribute, it must be initialized in the same manner as other source attributes. Two types of mappings are established to activate the composition transformation. Mapping $p_x \rightarrow a_\mu$ is performed by adding the RDF triple $\langle p_x\ M{:}\ mapsTo\ a_\mu \rangle$ to S_{ij} (line 13); mapping $A \rightarrow a_\mu$ is performed by adding a set of RDF triples, each following the pattern $\langle a_y\ S{:}\ isComposedBy\ a_\mu \rangle$ (lines 14-17, see Table 4.1). Since a_μ is a virtual attribute that has no physical implementation, its data values are dynamically constructed when queried.

Definition 4.4 (Decomposition γ): *Given an attribute* $a_y \in S_{ij}, S_{ij} \in S_i$*, the decomposition operator* γ_{a_y, A_γ} *decomposes the attribute* a_y *into a set of virtual attributes* $A_\gamma, where\ \ A_\gamma = \{a_{\gamma 1} a_{\gamma 2}, , \dots, a_{\gamma k}\}, k > 1$.

When $|A| = 1$, and $|P| > 1$ (line 18) a decomposition operation takes place to decompose a source attribute a_y into a set of new virtual attributes A_γ, and adds the set to S_{ij}. In the operation, a_y is modeled as the parent node of the new virtual attributes (lines 23-25). Similar to composition, the algorithm establishes two types of mappings to activate the decomposition transformation. Mapping $p_x \rightarrow a_{\gamma i}$ is materialized through the RDF triple $\langle p_x\ M{:}\ mapsTo\ a_{\gamma i} \rangle$, and mapping $a_{\gamma i} \rightarrow a_y$ is materialized through the RDF triple $\langle a_{\gamma i}\ S{:}\ isDecomposedFrom\ a_y \rangle$ (see Table 4.1). Since A_γ is a set of virtual attributes that have no actual implementation, the value of each attribute $a_{\gamma i}$ in A_γ must be dynamically constructed whenever needed.

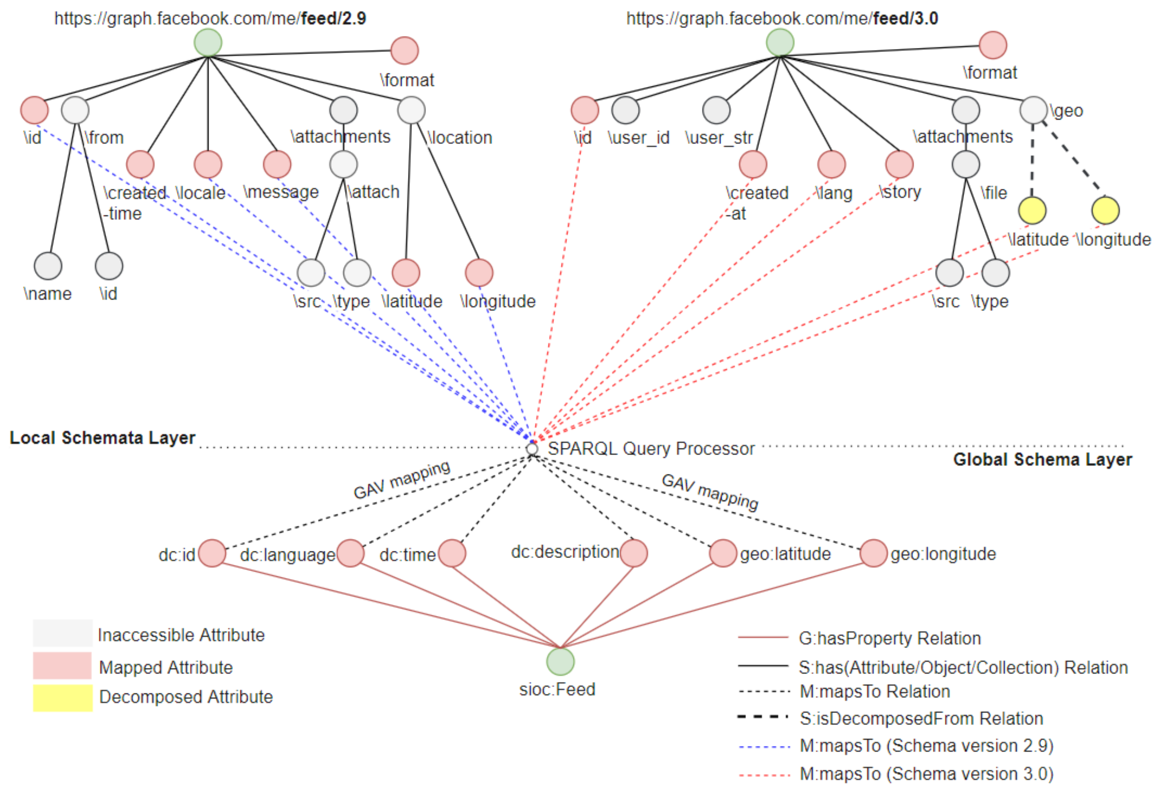


Figure 4.5– Mappings between the concept `Feed` and two source schemas

4.5.5 Partial unified views

A mediation-based integration approach typically emphasizes *rigorous* mapping between the global schema and the local schemas of the data sources under integration. While the assumptions behind such strict emphasis are construed in [183], the expected outcome is a set of one or more comprehensive unified views that effectively cover all the physical representation details of the integrated data. An implicit requirement in such approach is the prior knowledge about the local schemas of the data sources participating in the integration workflow. In contrast, as indicated in section 4.1, the input sources of a DL system rarely expose full-fledged schemas. This may justify why mediation-based integration approaches frequently break down inside a DL system [183], and similar dynamic data management environments for that matter. The dataspace research [196] has managed to overcome the strict modelling and rigorous mapping requirements by adopting an incremental integration approach in *pay-as-you-go* fashion [197]. For a given dataset in a dataspace system, the user may manually map only subset of its local schema elements, towards generating *partial* unified view that can be further enhanced when the user discovers more mappings over time, thus covering more physical details of the dataset's schema. In SemLinker, we adopt a similar but automated approach to generate unified views that are "partial". Technically, SemLinker harvests as much schema information as possible from a given input source through the schema extraction algorithm. It then attempts to correctly map as many elements in the schema of that source as possible to the tagging concept, through the mapping computation algorithm and by exploiting the data profiling capabilities offered by the matcher plugin, based on the information obtained from the former algorithm. The final product is a set of mappings represent partial modelling that can be directly utilised to rewrite user queries targeting the data of that source, among other data in PDL.

A partial unified view over the local schema of a given input source is fundamentally defined by the global concept – and its constituting global properties – tagging that source's data. Since the common user has full control on the global ontology $\mathcal{G}$, then the details of any concept $g \in \mathcal{G}$ can be readily modified in order to adjust the scope of the corresponding view over all the data tagged with this particular concept. Adjustments may involve replacing g with more specific/general concept of g', adding to or removing from its set of properties, and so forth. When any updates occur in $\mathcal{G}$, SemLinker automatically repeats its integration workflow (i.e. algorithms invocation) to reconstruct the mappings of the local schemas corresponding to g according to the newly applied modifications. Partial unified views can be regarded as user-tailored that may be personalised according to the emerging needs and requirements of the user regarding specific data in PDL. The underlying mappings of a partial unified view are static GAV mappings. Overcoming the sensitivity problem of GAV modelling, which we covered in section 4.2, is inherently achieved through local schema versioning, such that, since any changes in the local schemas are interpreted as new schema versions (i.e. definition 4.2), this enables the query engine component to rapidly and adequately rewrite and unfold input queries without encountering any schema mismatching problems; an input query over data ingested from particular evolving source is perfectly unfolded over the old source schema, and separately unfolded on another source schema that corresponds to the evolution of that source. Furthermore, as s shielding measure to protect the query execution process, SemLinker's query engine permits query-based accessibility to mapped schema elements only whereas unmapped elements are labelled as *inaccessible* elements and are hidden from the query issuer.

Figure 4.5 depicts a sample of two local schema versions and their mappings to properties of a tagging concept in the global ontology. The source schemas are extracted from Facebook data samples given in Figure 4.2a and Figure 4.2b using the schema extraction algorithm, and the mappings are computed using the mapping algorithm. In the figure, red circles indicate normal source attributes mapped to the equivalent properties in straightforward schema matching operations. The source attribute '*geo*' in the source schema *https://graph.facebook.com/me/feed/3.0* is marked by a white circle to indicate that decomposition has taken place, and '*geo'* is decomposed into virtual source attributes, namely, "*latitude*" and "*longitude*". The virtual source attributes (yellow circles) are also mapped to their corresponding global properties "*geo:latitude*" and "*geo:longitude*". Here, not all source attributes are mapped to properties of the tagging concept. Some source attributes (grey circles) are *inaccessible*, such as "*attachments*" of the second local schema. An inaccessible attribute, without an equivalent property in the global ontology, cannot be accessed through queries.

4.6 Querying

The Query engine in the global schema layer (see Figure 4.1) provides querying services over SemLinker's metadata. The engine serves two purposes inside MMF: (i) to provide an SQL abstraction for formulating SQL-like queries targeting the unified views over structured and semistructured personal raw data stored in PDL, and (ii) to compile, translate, and execute SQL-like queries that are input by the user or third-party services to SemLinker via PDL's access layer. The query engine accepts a successfully compiled input SQL-like query, converts it into a *relevance query* and an *unfolding query*, both of which are internal SPARQL queries that are automatically formulated by the engine based on analysing the input query statement(s), and as follows:

- A relevance query is a SPARQL SELECT query derived from the input query based on the concepts embedded in its clause formulation, and its execution over $\mathcal{G}$ returns all conceptually relevant local schemas.
- An unfolding query is a SPARQL SELECT query that is derived from the input query based on the ontological elements (global concepts and properties) embedded in the latter. Any unfolding query is iteratively executed on the underlying RDF graphs of the relevant local schemas that result from the relevance query execution. The result of the iterative execution is a list of source attributes that correspond to properties of the concepts specified in the query.

Once all the unfolded queries are executed, all the source attributes can be identified, and at this point, we have all the relevant metadata information regarding the query. The last phase of the query execution is to load data values that correspond to each identified source attribute PDL's unified repository. The loaded data values are assembled into a list of results and returned back to the SQL-like query issuer (the user or third-party service).

Here is a simple query scenario. Suppose the PDL user (the same user of the example in Figure 4.2) is interested in retrieving all social feeds, and their associated geolocation information (if any) that are stored in the unified repository, after a specified date (e.g., 1/7/2018), and so she writes the following query in the access layer's query interface and submits it to SemLinker.

```
SELECT
Feed.description,Feed.latitude,Feed.longitude

FROM sioc:Feed Feed WHERE

ParseTime(Feed.date,"dd/mm/yyyy")> "1/7/2018"
```

Upon submission, the access layer dispatches the query to SemLinker's query engine. The engine compiles it, and based on the concept (`sioc:Feed`) specified in the FROM clause of the query, it forms the following relevance query:

```
SELECT ?s WHERE { ?s M:isInstanceOf Feed .}
```

This relevance query is executed on the global ontology. A successful execution returns a view that is constituted by all local schemas in the local schemas repository that are tagged with the concept `Feed`. In our case (assuming Facebook and Twitter are the only local schemas tagged with this concept), $S_{Facebook}, S_{Twitter}$ are returned. Next, the query engine unfolds the input query and generates the following unfolded query, executing it iteratively on the RDF graphs of each local schema it has found, i.e., $S_{Facebook}, S_{Twitter}$.

```
SELECT ?a1 ?a2 ?a3 WHERE
{  Feed:description M:mapsTo ?a1 .
   Feed:latitude M:mapsTo ?a2 .
   Feed:longitude M:mapsTo ?a3 . }
```

Table 4.2 lists the view returned from executing the above unfolded SPARQL query. From the table, we see two matching local schemas, each with two source schemas, and their attributes corresponding to the properties specified in the original input query. Two virtual source attributes from decomposition, "*latitude*" and "*longitude*", are among the source attributes returned. Once the necessary metadata information for meeting the input query's data requirements is obtained, the query engine retrieves and parses the corresponding data entities and loads all data values matching the specific source attributes after performing the conditional filtering. The technical details of data retrieval from the storage layer (linkage table and unified repository) are covered earlier in section 3.4.

Since PDL stores raw data in its native representations, SemLinker's query engine may require invoking multiple parsers during raw data retrieval workloads. In our running example, the source schemas $S_{Facebook,v2.9}$, $S_{Facebook,v3.0}$, $S_{Twitter,v1.1}$ use JSON format, whereas the source schema $S_{Twitter,v1.2}$ uses XML format. The query engine automatically infers the appropriate parser for parsing the data entities annotated with each among these schemas. Inference is performed by querying the underlying RDF graph of each schema, more precisely, the *format* metadata artefact added to the source schema graph by the schema extraction algorithm.

```
SELECT ?x WHERE {  $S_{Facebook,v2.9}$ S:hasFormat ?x .}
SELECT ?x WHERE {  $S_{Twitter,v1.2}$ S:hasFormat ?x .}
```

For instance, by executing the above internal SPARQL queries, the engine can infer that the data entities annotated with $S_{Facebook,v2.9}$ exist with "application/json" format, whereas their counterparts annotated with $S_{Twitter,v1.2}$ exist in "application/xml". The query engine may call JsonCPP and PugiXML (see section 3.2) for rapid raw data parsing to access the values correspond to the attributes which reflect the global properties specified in the input SQL-like query.

Table 4.2 – Schema metadata results (view) of SQL-like query execution

Local S.	Source S.	Query Properties	Result Attributes
$\boldsymbol{S_{Facebook}}$	$S_{Facebook,v2.9}$	description, latitude, longitude	message, latitude, longitude
$\boldsymbol{S_{Facebook}}$	$S_{Facebook,v3.0}$	description, latitude, longitude	story, geo(latitude,longitude)
$\boldsymbol{S_{Twitter}}$	$S_{Twitter,v1.1}$	description, latitude, longitude	text, coordinates(latitude,longitude)
$\boldsymbol{S_{Twitter}}$	$S_{Twitter,v1.2}$	description, latitude, longitude	message, coordinates(latitude,longitude)

4.7 Summary

In this chapter, we presented SemLinker as an ontology-based data integration system that automates multiple tedious and time-consuming tasks in the data management workflow of PDL. SemLinker is the central component of MMF for managing (semi)structured personal data. It enables PDL users to accelerate time-to-value analytics on heterogeneous personal data by offering capabilities of data integration, processing, and querying through conceptual representation of physical schemas regarding a widely used global ontology. The system is based on two algorithms that automate the overall management process of personal data in PDL through metadata fabrics. The first algorithm constructs formal RDF representations that reflect the physical schemas of input raw data. The second algorithm metamodels these representations to the global ontology by discovering formal GAV mappings that are computed using pluggable profiling-based schema matcher that is designed by reusing two important current state of the art schema matching approaches. SemLinker uses the algorithms' outputs as metadata artefacts to annotate data inputs and to generate queryable unified views over them.

SemLinker was introduced to the academic community in a research paper [86]. To the best of our knowledge, it is the first domain-agnostic data integration system that offers self-adapting capabilities to integrate a variety big personal data with frequently evolving schemas based on solid theoretical foundations, without modifying the integrated data through physical transformations or similar intermediate modelling techniques. However, the system is not without limitations. It focuses on managing structured and semistructured personal data only, hence it is by no means a holistic data integration solution when unstructured data comes in the picture. In the next chapter, we introduce a principled solution to overcome this limitation.

Chapter 5 Unstructured Data Management

It is clear that there are large bodies of personal data in unstructured form, and managing them is unavoidable if we are pursuing a complete metadata management solution for PDL. In Chapter 4, we introduced an effective MMF component for managing personal data with explicitly-defined or self-describing structures, however, such component may quickly become futile when operating on data with no structure at all. In this chapter, we address the problem of unstructured data management by first proposing an unsupervised keyphrase extraction algorithm and then adopting it as a principal MMF component to automatically annotate unstructured personal data with formal metadata drawn from an extensible ontology, towards creating semantic representations that are utilised to organise the storage and usage of this kind of data and to facilitate its seamless integration with data of other structure kinds within PDL.

5.1 Introduction

Currently, structured and semistructured data sources can be viewed as islands in a sea of unstructured data [130]. Several studies estimate that 80-90% of the world's data is in unstructured from [198,199,200], and a significant portion of which is created and shared by common users as freeform text [201]. A comprehensive data management solution should be capable of creating bridges between structured, semistructured, and unstructured data, for seamless integration and smooth querying [201,202]. One important application of such bridging is to exploit unstructured contents as sources of strategic information for obtaining insights about the "backstory" of certain (semi)structured raw data pieces during analysis workloads or vice versa – see section 6.3 as an empirical example. However, data bridging over structure heterogeneity is by no means an easy task. The main challenge of unstructured data lies in its freeform nature. It – unlike structured and semistructured kinds – cannot be fitted into neat and explicitly defined representations [203], and with the absence of such representations, core management activities like access and retrieval become problematic [130,204]. The early research efforts for bridging the gap between unstructured text[36] and information retrieval (IR) [205] extensively focused on the bag-of-words paradigm. In recent years, a good deal of work attempted to go beyond this paradigm with the goal of constructing representations for unstructured text through *semantic annotations*. Basically, semantic annotation refers to the process of attaching metadata artefacts (e.g. keywords, ontology classes, etc.) to text segments as an enabler of information access and retrieval [206]. Various methods have been proposed in the literature for annotating textual documents, most of which employ Information Extraction (IE) techniques to automatically recognise instances of concepts (e.g. lexical chaining [207] and wikification [208]), topics (e.g. LSA [209], LDA [210], ESA [211]), events (e.g. event detection [212]), entities (e.g. named entity recognition [213,214]), relations (e.g. relation mining [214,215]), or combination of these (e.g. keyphrase extraction [53,55]), inside a given document. From operational perspective, existing methods are deployed either as automatic or semi-automatic applications where the user can inspect, and if necessary corrects the generated annotations. Automatic methods are preferred when the volume of data is too large to make human post-annotation practicable – which is the case of PDL. From general perspective, current state of the art methods are not flexible enough to support adaptability, domain tractability, and maintainability for being used in cost-saving and domain-agnostic scenarios [202][204][206]. To support managing unstructured text and bridging it with other raw data within a dynamic environment, an optimised semantic annotation method is needed, which should effectively address the following requirements:

- **Precision**: in section 3.4.1, we explored the importance of precise metadata creation and annotation. The heavy reliance on semantic annotations as means to enable unstructured data management entails strong correlation between the efficiency of the offered management functionalities and the precision of the adopted annotation method.

[36] In this work, we view unstructured data as any data entity comprises primarily textual content without predefined structure or metadata describing that content. Such understanding may be plausible but certainly incomprehensive as there are too many other forms of unstructured data (e.g. multimedia). In chapter 7, we discuss an extended approach for supporting other unstructured data forms as future work direction.

- **Domain-agnosticism**: a wide variety of unstructured documents may be ingested by PDL, hence the adopted method should exhibit effective robustness across various domains rather than targeting specific ones.
- **Supervision**: one of the main goals of MMF is the reduction of manual management efforts for users. A supervised or semi-supervised method typically requires expensive training corpora of ground truth data [216]. Such requirement inherently challenges our research goal from operational perspective: common users do not have the knowledge or the sufficient resources to deliver high-quality training corpora, and going domain agnostic entails the need for multiple training datasets to train different ML models that correspond to each domain of interest to the user. Thus it is critical to adopt an unsupervised method where no training of whatsoever is required.
- **Personalisation**: while existing methods typically annotate the text constituents of an input document with general semantic classes, the adopted method should offer finer annotation granularity for faster and convenient information access and retrieval whenever applicable. For instance, annotating a mention of the user's sibling inside a personal email message with the class `Brother` (or `Sister`) can contribute in boosting the efficiency of retrieving that message from a large text base than if it would otherwise be annotated with the generic class `Person`[37].

To this end, we introduce an automatic semantic annotation (ASA) method for addressing the above requirements. The method is fundamentally based on a new *automatic keyphrase extraction* (AKE) algorithm, and it aims to model any unstructured document in PDL into *semantic representation* that is defined by a set of thematically important disambiguated text constituents (e.g. concepts, instances, named entities) and their associated conceptual meanings defined by a non-empty set of classes drawn from an extensible ontology. The proposed method is implemented as an MMF component, and it can be directly queried for retrieving the information which the PDL user is looking for, and present it at the right level of detail.

[37] For example, most current state of the art NER taggers annotate name mentions with `Person`. See the following cases: http://nlp.stanford.edu:8080/ner/ and http://text-processing.com/demo/tag/

In the remainder of this chapter, we briefly review AKE literature, present SemCluster algorithm, and describe its implementation as MMF component.

5.2 Automatic keyphrase extraction

Keyphrases are single or multi-word expressions that describe the essential content of a free-text document. AKE [218] is an NLP task that concerns analysing the content of an input document and automatically identifying and extracting keyphrases that correspond to the document's theme. Existing AKE work can be broadly divided into two lines of research [221,222,223]: supervised and unsupervised. A supervised AKE approach typically treats the extraction task as a classification problem, in which, a classifier is trained on a large corpus of documents that are annotated with "correct" keyphrases by human experts, and the result is a machine learning model which then can be used for discriminating keyphrases from non-keyphrases in unseen documents. Various text features and classification algorithms have been applied in supervised AKE, (e.g. [218,219,220,224,225]. Supervised approaches perform AKE with promising results [221,226], however they contradict the requirements listed in section 5.1, such that, a supervised AKE approach requires substantial amount of manually annotated training data which is very expensive requirement for common users and may lead to inconsistencies in heterogeneous processing environments that demand cross-domain tractability [44]. For instance, a classifier trained on features of labelled keyphrases that belong to a particular domain (e.g. scientific papers) may exhibit poor performance when applied on documents from another domain (e.g. news articles).

Unsupervised AKE overcomes the critical challenges of training data and domain bias by casting the extraction task as a ranking problem. The typical workflow here is to select a particular set of terms from the input document, then by applying some ranking strategy, top ranked terms are taken as keyphrases. Generally, an unsupervised AKE approach can either be graph-based or statistics-based [222]. In the former, the input document is modelled as a graph where each node represents a term in the document, and the edge between a pair of nodes resembles a relevance relation (e.g. co-occurrence). Subsequently, a centrality measure (e.g. PageRank [227]) is applied on the graph to rank each node based on its incoming and outgoing edges. Finally, the top-k ranked nodes are selected as keyphrases. In the latter, the terms are ranked based on their associated statistical information, such as TF, IDF, or statistical distances, and then the top-k ranked terms are either selected as keyphrases, or utilised as heuristics that can be further exploited to search for better candidates.

Compared with other NLP tasks, unsupervised AKE approaches struggle to achieve better results [221]. AKE requires not only local statistical information about the terms contained in the document but also extensive background knowledge to capture the relations between them [221]. Many recent approaches suggest utilising external knowledge sources (e.g., WordNet [49]) to obtain rich relation information about terms during AKE [228,229]. Although these approaches demonstrate improved extraction performance in some cases, their utilised knowledge sources are not consistent enough to supply background information in arbitrary domains, therefore a term representative of the document's theme may be disregarded simply because the knowledge source does not maintain information about it –we refer to this issue as *coverage limitation*. For example, Figure 5.1 depicts a text segment drawn from DUC-2001 news dataset [54]; the term "Ben Johnson" is important because it refers to the name of the athlete who this news article is about, hence the term is selected as a valid keyphrase by human curators. A typical WordNet-based unsupervised AKE approach, such as SemGraph [229], would disregard this term because WordNet has no entry matching "Ben Johnson".

Another issue in existing unsupervised approaches is their heavy reliance on statistical information to capture the statistical relations between terms, thereby failing to account for latent semantic relations. In a graph-based approach, if two representative terms are not co-occurring within a predefined window, then no occurrence edge will be established between their donating nodes, and their ranks in the graph decrease. Similarly, statistics-based approaches treat terms solely as statistical elements, therefore a term of high frequency is typically ranked higher than an infrequent but semantically important term. Due to this *semantics loss*, a term representative of the document theme is not guaranteed to be among its top-ranked candidates if it occurs infrequently [222], which also means a top-ranked candidate of statistical importance may not be suitable to be a representative of the document [55]. In our running example, most state of the art AKE approaches fail to identify “dash” as a representative term because of its distance from other co-occurring terms such as “*Olympics*” and its infrequent occurrence in the text, thereby the phrase “100-meter dash” is not regarded as a valid keyphrase. On the other hand, the term “Olympics” appears frequently, so it is not surprising that most AKE approaches select “Olympics", “Olympic Games”, and “Olympic movement” as valid keyphrases, without considering that “Olympic movement” is not identified by the human curators as a valid keyphrase, because it has no immediate semantic relevance to the document theme, for instance, many knowledge bases[38,39,40] map this term to “Organisation” and not “Sport” domain.

Canadian **Ben Johnson** left the **Olympics** today “in a complete state of shock,” accused of cheating with drugs in the world’s fastest **100-meter dash** and stripped of his **gold medal**. The prize went to American **Carl Lewis**. Many athletes accepted the accusation that Johnson used a muscle-building but dangerous and illegal **anabolic steroid** called **Stanozolol** as confirmation of what they said they know has been going on in track and field. Two tests of Johnson’s urine sample proved positive and his denials of **drug use** were rejected today. “This is a blow for the Olympic Games and the Olympic movement,” said International Olympic Committee President Juan Antonio Samaranch.

Figure 5.1– A text segment from #AP880927-0089 in DUC-2001 dataset[41]

Building an AKE-based ASA method to support unstructured data management in PDL intuitively necessitates the need for an efficient underlying AKE algorithm that can overcome the aforementioned challenges. Accordingly, we propose SemCluster, an unsupervised clustering-based algorithm that extracts high-quality keyphrases from free-text documents in any domain. SemCluster first extracts a particular set of terms from an input unstructured document and then performs clustering on them so that *similar* terms are grouped within the same cluster based on their latent lexical and semantic relations. Each resulting cluster may implicitly correspond to a topic in the document. Terms that are close to the centroids of *specific* clusters are selected as *seeds* and used to search for

[38] http://www.babelnet.org
[39] http://www.conceptnet.io
[40] http://lookup.dbpedia.org/api
[41] Phrases in bold are gold standard keyphrases that are assigned by expert curators as given in [54].

candidate phrases that are representative of the main theme of a document. Finally, candidate phrases are refined and the resulting candidates are chosen as *appropriate* keyphrases.

SemCluster makes use of knowledge extensibility in order to address the aforementioned unsupervised AKE challenges. SemCluster adopts WordNet as a default knowledge source to obtain background semantic information about the terms within the document. The semantic coverage of WordNet can be flexibly extended by integrating any number of additional generic, specialised, or personalised knowledge sources, so that when the semantic information of an arbitrary term is not present in WordNet, it may be available in the integrated source(s). For example, by integrating WordNet with DBPedia [50], SemCluster can obtain rich semantics about "Ben Johnson" from DBPedia, even though this term has no matching entry in WordNet. With the availability of rich semantics, SemCluster can readily capture the latent semantic relations between terms, and adequately rank each term based on its semantic importance in the context of the document and its relevance to the underlying theme. In the example presented in Figure 5.1, despite the infrequent occurrence of the term "*dash*" and its distance from statistically relevant terms, SemCluster assigns it a high rank due to its semantic *closeness* to "Olympics", "Ben Johnson", and "Carl Lewis".

5.3 Related work

Most early studies on unsupervised[42] methods for keyphrase extraction focus on using the local information within input documents. The simplest approach uses the TF criterion [235] for choosing frequently occurring phrases as keyphrases. More sophisticated methods incorporate additional statistical and linguistic information. Barker et al. [238] suggests extracting noun phrases from the document and ranks them based on the length, frequency, and the head-noun frequency of each phrase. Munoz [239] proposes an unsupervised algorithm that is based on adaptive resonance theory to identify bi-gram

[42] In this thesis, we do not consider any AKE approaches that demand ML training to improve terms ranking, such as word embedding based approaches.

keyphrases, though keyphrases intuitively vary in length. El-Beltagy et al. [236] propose KP-Miner, a state of the art $TF.IDF$ based approach that operates on n-gram phrases, and only phrases that do not contain a stop word or punctuation mark, occur for the first time within the first m words of the document, and have a frequency greater than a threshold determined by the document length, are selected as candidate phrases. Subsequently, candidates are ranked using a modified $TF.IDF$ model that incorporates a boosting factor aimed at reducing the bias towards single-word candidates. KP-Miner suffers two main drawbacks: it treats phrases as solely statistical elements in the document, and it ignores the fact that based on recent studies [248] 15% of keyphrases contain stop words.

Graph-based AKE is another major stream of AKE research [210]. Mihalcea et al. [53] propose TextRank, the first approach to rank candidate keyphrases based on the co-occurrence links between words. TextRank uses a sliding window technique to construct the word graph of an input document. The sliding window moves from the first word to the last word in the document, and words that co-occur within a window $m \geq 2$ are connected by an edge in the graph. The approach then applies PageRank on the graph to rank nodes through voting [227], such that a node with more in- and out- edges has more probability of being top-ranked. However, because important words with low frequency are often ranked low (i.e. semantic loss), TextRank performs AKE with poor accuracy. Numerous methods have been proposed in the literature to compensate the semantic loss of TextRank. Among these, Danesh et al. [232] present a hybrid (statistics- and graph-based) approach that computes an initial weight for each phrase based on its $TF.IDF$ score and the position of its first occurrence in the document. Then the phrases, together with their weights, are modeled as a graph and their weights are recomputed using a centrality measure to produce the final ranking of phrases. Wan et al. [54] introduce an extension of TextRank that incorporates the co-occurrence information from a set of neighbour documents to weight the edges between words in the graph-based representation of the input document. The algorithm uses the Cosine Similarity measure to retrieve documents from a large document corpus that are topically related to the input document. The retrieved documents contribute in identifying and ranking the phrases that correspond to the topics in the document. However, the retrieval of topic-related documents from large corpora is very expensive. Wang et. al. [228] extends TextRank by incorporating background semantic information form WordNet for weighting the nodes in the graph. Then PageRank is used to compute the top-k ranked nodes. Similarly, Martinez-Romo et al. [229] use information from WordNet to enrich the semantic relationships between words in the word graph. Though the performance of the methods using Wordnet has improved greatly, as indicated in the introduction section, WordNet is limited in terms of its semantic coverage and is not a panacea.

Clustering-based studies are another family of unsupervised AKE [222]. Bracewell et al. [237] present a method for extracting noun phrases from a document and grouping them into clusters based on their shared noun terms. The resulting clusters are ranked based on noun term frequencies, and the top-k ranked clusters are selected as keyphrases. Liu et al.[55] introduce a similar clustering-based algorithm, called KeyCluster, that extracts noun terms from an input document, groups them into clusters based on their semantic relatedness, then selects the phrases from the document body which contain one or more cluster centroids and which follow a certain linguistic pattern. Finally, the selected phrases are regarded as output keyphrases. KeyCluster adopts Wikipedia as an external knowledge source to capture the relatedness between noun terms. The basic idea here is to consider each Wikipedia article as a concept, then the semantic meaning of a term is represented as a weighted vector of Wikipedia concepts, of which the values are the term's TFIDF within corresponding Wikipedia articles. Accordingly, a similarity metric can be used to capture the relatedness between two terms based on their conceptual vectors. The work in [55] gives three options to implement the similarity measure, which are Cosine similarity, Euclidean distance, Point-wise Mutual Information and Normalised Google Similarity Distance [230]. The output of pair-wise similarity computations is a similarity matrix that is clustered to obtain the clusters' centroids that are necessary for selecting candidate phrases. The authors of KeyCluster choose three clustering algorithms for the clustering task, which are: Hierarchal Clustering, Spectral Clustering, and Affinity Propagation. KeyCluster has been shown to outperform many prominent AKE methods, however early clustering-based methods in general cannot guarantee that all generated clusters are sufficient to cover the document theme, and selecting the centroid of a topically unimportant cluster as a heuristic to identify and extract keyphrases yields erroneous outputs. More recent studies propose to incorporate topic analysis in the AKE task to ensure that output keyphrases have strong association with the document's main theme from a topical viewpoint. In the topical clustering-based method [231,233,234], terms are grouped into clusters using an appropriate clustering algorithm, and the method proceeds to conduct topic analysis using a probabilistic topical model, such as Latent Dirichlet Allocation [210], in order to extract all latent topics T in the document. The importance of each term is computed as the sum of its scores in each topic $T_i \in T$, weighted by the probability of T_i. Hence a term that belongs to an important topic T_i is weighted more heavily than a term that belongs to a less important topic T_j. Although topical clustering-based methods improve significantly their AKE performance, they essentially suffer empirical challenges related to the topic analysis process. For instance, when applied to new domains, LDA and similar models induce high computational complexity and require hyperparameter (re)tuning, which is a non-trivial task in domain-agnostic text processing applications, given that the user is "common".

SemCluster is based on an extensive literature review and through learning the advantages and disadvantages of other approaches. It adopts an approach that extracts n-gram terms and named entities instead of single words (similar to KP-Miner) and relies greatly on background knowledge sources (similar to SGRank, SemGraph, ExpandRank, and KeyCluster). However, because of the coverage limitation problem that would arise if it was based on a sole knowledge source, SemCluster is designed to allow extensibility of its knowledge base by integrating with other knowledge sources.

5.4 SemCluster overview

Given an extensible background knowledge source that is modelled as the ontology O, for an input unstructured document D, SemCluster performs the algorithmic steps depicted in Figure 5.2 to extract a set of keyphrases that are most representative of the document's them. The following subsections cover the full details of each step.

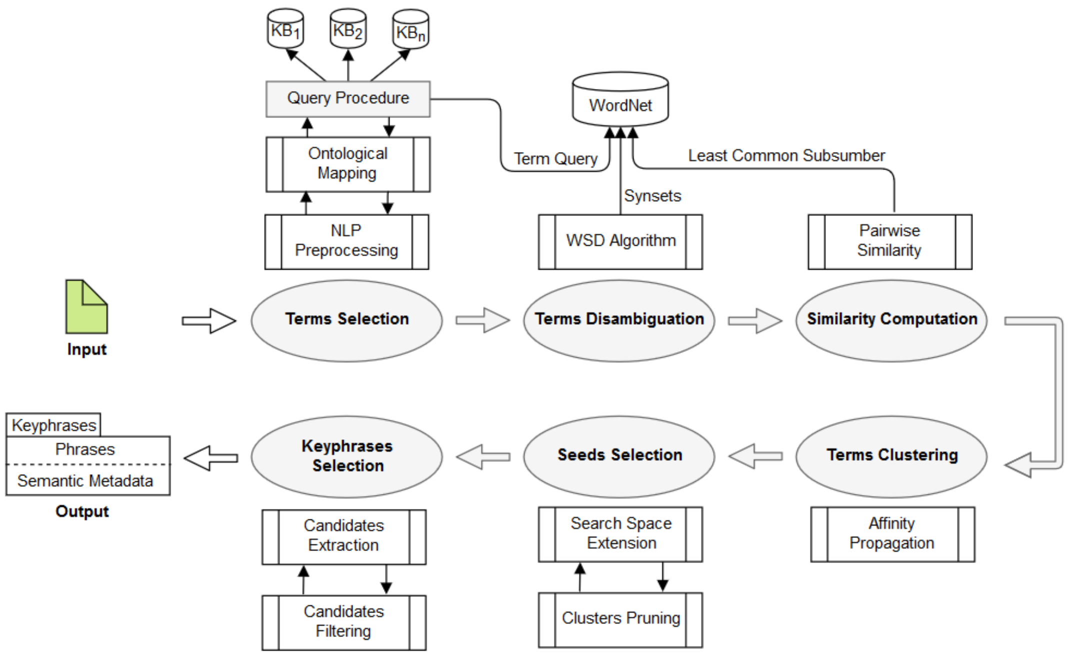


Figure 5.2– SemCluster algorithm steps

5.4.1 Candidate terms selection

The first step in SemCluster is the selection of candidate terms, it is aimed at extracting from the content of D a general set of terms, where each term is associated with background semantic information. The step begins with pre-processing D by applying the following NLP tasks: *tokenization*, *sentence boundary detection*, *part-of-speech* (POS) *tagging*, and *shallow parsing* (chunking). Penn Treebank notion [240] is adopted for POS tagging and chunking. The aim of chunking is to group words into chunks based on their discrete grammatical meanings. Many NLP studies have shown that almost all keyphrases assigned by expert curators are typically embedded in noun phrases (i.e. NP chunks) [55,220,237,238]. SemCluster considers only NP chunks to find keyphrases, and detects and extracts terms in each NP chunk based on their POS annotations. We allow the selection of n-gram terms (where $0 < n \leq 5$) using the POS patterns listed in Table 5.1. $\mathcal{N}$ denotes *Noun*, a word tagged as a singular noun (NN) or plural noun (NNS). $\mathcal{C}$ denotes *Compound Noun*, a sequence of words starting either with an adjective (JJ) or noun (both NN and NNS). E denotes an *Entity*, a sequence of words of singular proper nouns (NNP) or plural proper nouns ($NNPS$) with at most one stop-word ($\mathcal{S}$): *the* at the beginning, or *of* in the middle. Each term extracted using these patterns is mapped into SemCluster's ontology O, and depending on the mapping result, a term is regarded either as a *candidate term* or *miscellaneous*. When a term does not map to any entries in the ontology, it is decomposed into smaller constituents to be mapped again. The terms that fail to find matches even after being reduced to smaller constituents are discarded.

SemCluster uses WordNet as its ontology O. WordNet is a widely used lexical database. It comprises four lexical networks [49]: Nouns, Verbs, Adjectives, and Adverbs. In SemCluster we use only the Nouns network. WordNet groups nouns of equivalent meanings into *synsets*[43]. A synset consists of a list of synonyms and a short definition called a *gloss*. Synsets are connected to other semantically relevant synsets by means of semantic relations. Noun synsets are organised using hyponym/hypernym (Is-A), and meronym/holonym (Part-Of) relationships, providing a hierarchical tree-like structure that can be directly modeled as an ontology.

Table 5.1 – POS patterns for n-gram extraction from NPs

Extraction pattern	Extraction examples
$\mathcal{N} = (\mathrm{NN}\|\mathrm{NNS})$	dash/NN, prize/NN, drugs/NNS
$\mathcal{C} = (\mathrm{JJ}) * (\mathrm{NN}\|\mathrm{NNS}) +$	anabolic/JJ steroid/NN, gold/NN medal/NN, urine/JJ sample/NN
$\mathrm{E} = (\mathrm{NNP}\|\mathrm{NNPS}) * (\mathcal{S}) * (\mathrm{NNP}\|\mathrm{NNPS}) +$	Stanozolol/ NNP, Ben/NNP Johnson/NNP, Olympics/NNPS

Background knowledge extensibility

In practice, no knowledge base is comprehensive, and neither is WordNet. The Nouns network contains a large but limited number of English nouns collected nearly two decades ago, and therefore, WordNet does not support newly emerging nouns, or new meanings of already existing nouns. Relying solely on WordNet as the only background knowledge

[43] The semantics terminology in NLP field is slightly different from data management (e.g. chapter 4). To maintain the consistency of this thesis discussions, it should be borne in mind that the following terms are identical: *synset*, *external synset*, (ontological) *concept*, *class*, and *type class*.

source leads to the background knowledge coverage limitation as discussed earlier. To overcome this we design a procedure for extending WordNet coverage by integrating external knowledge bases that utilise ontology-based schemas for structuring their internal information, such as DBPedia, BabelNet [51], Yago [241], or any *ad hoc* (specialised or personalised) knowledge bases.

```
Input:
    $t_i \in D, KB_x \in \{KB\}$
Output:
    $H = \{(e,s)_1, (e,s)_2, \dots, (e,s)_n\}$ : The set of entries matching $t_i$ and their
    corresponding WordNet synsets.
Procedure:
    If $t_i$ found in $KB_x$ then
      Retrieve all entries $E$ matching $t_i$, $E = \{e_1, e_2, \dots, e_n\}, E \subset KB_x$
      For each entry $e_j$ in $E$:
        Retrieve all type classes $C$ of $e_j$, $C = \{c_1, c_2, \dots, c_m\}$
        For each $c_h$ in $C$:
          If $c_h$ is the deepest type class in $KB_x$ schema ontology then
            Select $c_h$ as hypernym of $e_j$
            Find the equivalent synset $s_i$ of $c_h$
            Assign $s_i$ as hypernym of $e_j$
            Add the pair $(e_i, s_j)$ to $H$
      Return $H$
    Else
      Return $\emptyset$
```

Algorithm (3)– Extensible background knowledge querying

The workflow of our proposed procedure is as follows: given an external knowledge base, denoted as KB_x, its schema is modeled as an ontology, and each entry in KB_x is assigned one or more ontological concepts called a *type class*. To perform a meaningful integration, the schema ontology of KB_x is horizontally aligned with O by mapping each type class to its semantically equivalent synset. To prevent conceptual ambiguity, ontological alignments are performed as one-to-one mappings, such that, each type class in the KB_x schema ontology is mapped to exactly one synset in O. During the selection of candidate terms, an n-gram that is extracted from the pre-processed content of D and that cannot be mapped to WordNet, is queried against the integrated knowledge base(s), denoted as $\{KB\}$, using the procedure depicted in Algorithm (3). Given the knowledge base $KB_x \in \{KB\}$, whose schema ontology is properly aligned with O, SemCluster queries KB_x with the n-gram t_i. If there are entries in KB_x matching t_i, then each matched entry is retrieved from KB_x and is considered as an external contextual meaning (or *sense*) of t_i. All the type classes associated with external senses of t_i in KB_x are mapped into their corresponding synsets in O and are considered as hypernyms of t_i. The synset that corresponds to the deepest type class in the schema ontology of KB_x is considered the correct hypernym of the external sense. With this construct, we allow SemCluster to dynamically generate appropriate senses for the terms that are absent in WordNet, or even expand the set of synsets for an existing term.

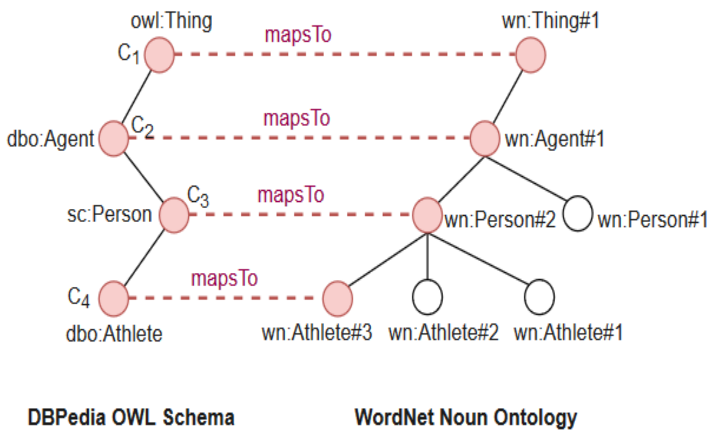


Figure 5.3– A fragment of ontological WordNet-DBPedia alignment

To illustrate with a real-world example, we consider extending O with DBPedia (i.e. $KB_{DBPedia}$) and aligning the type classes in its schema ontology[44] with their equivalent WordNet synsets. For example, the type class `dbo:Athlete` in $KB_{DBPedia}$ is directly mapped to `wn:Athlete#n1` in WordNet, `dbo:MusicFestival` is mapped to its equivalent synset `wn:Fete#n2`. Revisiting the news article example depicted in Figure 5.1, we see that the term "Ben Johnson" has no entries in WordNet but five entries in $KB_{DBPedia}$. SemCluster generates five new external senses for "Ben Johnson", each reflecting one entry in $KB_{DBPedia}$. The third sense in particular, "*Ben Johnson*

[44] DBPedia schema ontology is available at http://mappings.dbpedia.org/server/ontology/classes/

(Sprinter)”[45], is associated with four classes as depicted in Figure 5.3: `owl:Thing`, `dbo:Agent`, `sc:Person`, `dbo:Athlete`. According to the querying algorithm, the deepest among the four classes, `dbo:Athlete`, becomes the hypernym of the third sense and is referred to as `wn:Athlete#n1`. After mapping each extracted term against the extended ontology $O^{\dagger}$, only a subset of the terms are selected as candidate terms. We denote the set of the candidate terms as T_D. Due to the pattern-based method of term extraction, especially when D contains informal text, T_D may contain noisy terms that can adversely affect similarity computation and clustering performance. Noisy terms are nouns with no semantic value (e.g. “one”,”someone”, etc.). To identify and remove noisy terms, SemCluster maps each term in T_D to an internal list that contains the most frequent noisy terms in the English language, and any term found in the list is removed from T_D.

5.4.2 Candidate terms disambiguation

A consequence of obtaining semantic background information about candidate terms is that each term in T_D may be associated with one or more contextual meanings (or *senses*), whether local or external. Prior to semantic similarity computation, SemCluster must identify the correct sense of each term in T_D. Word Sense Disambiguation (WSD) is an NLP task that gives machines the ability to computationally determine which sense of a term is activated by its use in a particular context. WSD approaches are generally divided into three categories [242]: supervised, unsupervised, and knowledge-based. SemCluster employs the SenseRelate-TargetWord method [243] for term sense disambiguation. The algorithm is WordNet-based and is implemented in WordNet::Similarity[46], a popular package in computational linguistics. The SenseRelate-TargetWord method takes one *target* candidate term as input and outputs a WordNet synset as the disambiguated sense of the target candidate term, based on information about the target as well as a few other candidate terms surrounding the target. The surrounding candidate terms are called the *context window*. Let t_i be a target candidate term, $t_i \in T_D$, and the context window size be N, and the set of surrounding candidate terms in the context window be $W,\ W =$

[45] http://dbpedia.org/page/Ben_Johnson_(sprinter)
[46] http://wn-similarity.sourceforge.net

$\{w_1, w_2, \dots, w_N\}$, where, if $|W| < N$, then $N = |W|$. Since t_i is deemed to be associated with a set of one or more senses, we denote this set by $Sense(t_i) = \{s_{i1}, s_{i2}, \dots, s_{im}\}$. For each sense s_{ij}, we obtain not only its synonyms list and gloss from WordNet, but also the synonym lists and glosses of other synsets that are related to s_{ij} via the following set of semantic relations:

$$\{Hypernym, Hyponym, Meronym, Holonym\}$$

The goal of the SenseRelate-TargetWord algorithm is to find the synset responsible for s_{ij} whose synonyms and gloss content maximizes the string-based overlap score with each w_k in the context window.

5.4.3 Candidate terms similarity computation

After disambiguating all the candidate terms in T_D, each $t_i \in T_D$ becomes associated with the following information: POS tag, position in the document, and a pointer linking t_i with its correctly disambiguated WordNet synset s_i. In this step, SemCluster computes the pairwise semantic similarity between each pair of terms in T_D based on their synset pointers. There exist many measures to quantify the similarity between two synsets, and these measures are broadly divided into three main categories [244]: path-length based, information content based, and feature based. Unlike the other two, path-length measures offer greater flexibility in computing the similarity between synsets based on SemCluster's extensible ontology. The WuPalmer measure [245] is a prominent path-length measure to compute semantic similarity between two synsets s_i, s_j by finding the shortest path between them relative to the deepest common parent synset, i.e. the *Least Common Subsumer* (LCS). The similarity $S(s_i, s_j)$ is quantified by counting the nodes in the shortest path between each synset and the LCS in O. The measure is defined as follows:

$$\mathrm{S}(s_i, s_j) = \frac{2d}{L_{si} + L_{sj} + 2d} \tag{5.1}$$

where d is the depth of *LCS* from the root node, L_{si} is the path length from s_i to LCS, and L_{sj} is the path length from s_j to *LCS*. In this work, we modify the WuPalmer metric to capture extra semantic similarity between s_i and s_j. Path length measures in general, and

WuPalmer in particular, focus on measuring the semantic similarity between a pair of synsets s_i and s_j by exploiting the explicit semantic relations existing between them. However, WordNet does not cover all possible relations that may exist between synsets. For example, there is no direct link between "`wn:Bush#n4`" and `wn:President#n2`, although they are clearly related if they co-occur in a document. To capture explicit, as well as implicit, semantic similarities using WuPalmer, we extend its mathematical notion as follows:

$$\mathrm{S}(s_i, s_j) = \frac{2d + Overlap(C(s_i), C(s_j))}{L_{si} + L_{sj} + 2d + Overlap(C(s_i), C(s_j))} \tag{5.2}$$

where $C(s_i), C(s_j)$ are functions that retrieve s_i and s_j information from WordNet in string format, and $Overlap(C(s_i), C(s_j))$ is a function that measures the string-based overlap between $C(s_i)$ and $C(s_j)$. Let $Synonyms(s_i)$ be a function that retrieves all the words in the synonyms list of the synset s_i, $Gloss(s_i)$ be a function that retrieves the definition of s_i, $Related(s_i)$ be a function that retrieves the synonyms and glosses of all synsets connected directly to s_i via the relation set:

$$\{Hypernym, Hyponym, Meronym, Holonym\}$$

Then $C(s_i)$ is defined as follows:

$$C(s_i) = Synonyms(s_i) \cup Gloss(s_i) \cup Related(s_i) \tag{5.3}$$

where $\cup$ is the string concatenation function. $Overlap(C(s_i), C(s_j))$ finds the maximum number of words shared in the output of $C(s_i)$ and $C(s_j)$ normalised by the natural logarithm to prevent too much effect of implicit semantic similarity on the WuPalmer explicit semantic similarity measurement. Thus, we define *overlap* as follows:

$$Overlap(C(s_i), C(s_j)) = \log\left(C(s_i) \cap C(s_j) + 1\right) \tag{5.4}$$

The extended WuPalmer measure is used to compute the pairwise similarities between each pair of terms in T_D, and the result is a complete adjacency similarity matrix of size

$|T_D|^2$ denoted as $\mathcal{A}$. Once we have produced $\mathcal{A}$, we move on to the next step – clustering T_D based on $\mathcal{A}$.

5.4.4 Candidate terms clustering

There are many state of the art clustering algorithms to efficiently cluster the adjacency matrix $\mathcal{A}$ resulting from the previous step. Affinity Propagation (AP) [247] has been proposed as a powerful technique for exemplar learning by passing messages between nodes. It is reported to find clusters with much lower error compared with other algorithms [246]. In addition, AP does not require specifying the number of desirable clusters in advance as clustering is fully data-driven. Both advantages are extremely important for SemCluster to support fully AKE, and hence, AP is adopted as SemCluster's underlying clustering algorithm. The input to AP is the matrix $\mathcal{A}$. The set T_D is modeled as a graph. An edge exists between two candidate terms t_i and t_j, $t_i, t_j \in T_D$, if $\mathrm{S}(t_i, t_j) > 0$, and the weight of the edge is given by the cell $\mathcal{A}[i][j]$. Initially, all the nodes are viewed as exemplars, and after a large number of real-valued information messages have been transmitted along the edges of the graph, a relevant set of exemplars and corresponding clusters are identified.

In AP terms, the similarity metric $\mathrm{S}(t_i, t_j)$ indicates how much t_j is suitable as an exemplar of t_i. In SemCluster, $\mathrm{S}(t_i, t_j) = \mathcal{A}[i][j], i \neq j$. If there is no heuristic knowledge, self-similarities are called *preferences*, and are taken as constant values. The preference $P(t_i) = \mathrm{S}(t_i, t_i)$ represents the *a priori* suitability of the term t_i to serve as an exemplar. In SemCluster, preferences are computed using the *median*. AP computes two kinds of messages exchanged between nodes: *responsibility* and *availability*. A responsibility message, denoted by $\mathrm{r}(t_i, t_j)$, is sent from node t_i to node t_j, and reflects the accumulated evidence for how well-suited t_j is to serve as the exemplar of t_i. An availability message, denoted as $\mathrm{a}(t_i, t_j)$, is sent from t_j to t_i, and reflects the accumulated evidence for how well-suited it would be for t_i to choose t_j as its exemplar. At the beginning, all availabilities are initialized to zero, i.e., for each $\mathrm{a}(t_i, t_j) = 0$; then responsibility and availability messages are updated using equations (5,6) [247].

$$\mathrm{r}(t_i, t_j) = s(t_i, t_j) - max_{j' \neq j}\{\, a(t_i, t_{j'}) + s(t_i, t_{j'})\} \tag{5.5}$$

$$\mathrm{a}(t_i, t_j) = \begin{cases} \min\left\{ 0\,, r(t_j, t_j) + \sum\limits_{i' \neq i,j} \max\{0, r(t_{i'}, t_j)\} \right\} & , i \neq j \\ \sum\limits_{i' \neq i} \max\{0, r(t_{i'}, t_j)\} & , i = j \end{cases} \tag{5.6}$$

The responsibility and availability messages are updated iteratively for m iterations, and a dumping factor, denoted by $\lambda, \lambda \in [0,1]$, is added to both types of messages in order to avoid *numerical oscillations* [247], as depicted in equations 5.7 and 5.8.

$$R_{m+1} = (1 - \lambda)R_m + \lambda R_{m-1} \tag{5.7}$$

$$A_{m+1} = (1 - \lambda)Y_m + \lambda A_{m-1} \tag{5.8}$$

where R is the responsibility matrix, $R = [r(t_i, t_j)]$, and A is the availability matrix, $A = [a(t_i, t_j)]$. AP continues updating $\mathrm{r}(t_i, t_j)$ and $\mathrm{a}(t_i, t_j)$ until they remain constant for a specified number of iterations, and then both types of messages are combined to discover the exemplar candidate terms in T_D, specified as follows:

$$\varepsilon_j \leftarrow arg_{1 \leq j \leq N} \max[r(t_i, t_j) + a(t_i, t_j)] \quad , where\ N = |T_D| \tag{5.9}$$

where ε_j is a term in T_D and is regarded as an exemplar of term t_i. Eventually, every term in T_D is annotated with its exemplar. The number of clusters, and other clustering information, are directly obtained by grouping terms based on their shared exemplars. At start-up, we allow the set T_D to be *redundant* in order to incorporate not only the semantic and lexical information of each term T_D but also the influence of its frequency information on the clustering results, such that, if the term t_i is highly frequent in the document, its frequency can be a reason to qualify as an exemplar on the condition that t_i is always allocated the same WordNet synset s_{ij} in all its occurrences in D.

5.4.5 Seeds selection

Typically, clustering-based AKE approaches use the centroids of clusters as seeds [55,238,222], and any phrase in D containing one or more centroids is chosen as a keyphrase. From our empirical observation, we suggest that direct selection of centroids resulting from the adopted clustering algorithm may lead to poor keyphrase extraction recall and/or precision, due to the following reasons:

Theme-independent seed selection

Clustering-based methods assign equal importance to all cluster centroids [222,234]. Thus, a phrase containing a centroid of an unimportant cluster is ranked exactly equivalent to a phrase containing a centroid of an extremely important cluster relative to the document theme [234]. Consequently, there is no guarantee that the extracted keyphrases are the best representative phrases. Our solution to this is to discard irrelevant or marginally related clusters and keep the most relevant ones. The solution is largely based on the observation that clusters that sufficiently cover the document theme tend to be semantically more related to each other than irrelevant or marginally related clusters. Regarding AP, the exemplar is the best representative of its cluster's semantics. Therefore, we assess the average of semantic relatedness strength of each exemplar against all other exemplars, and any cluster whose exemplar exhibits *weak* semantic relatedness is removed. Let C_D be the set of clusters resulting from clustering T_D, $C_D = \{C_1, C_2, \dots, C_N\}$, where $N = |C_D|$. For each cluster C_i, we compute its exemplar's average semantic relatedness, $Ave(\varepsilon_i)$, as follows:

$$Ave(\varepsilon_i) = \frac{\sum_{i \neq j} SR(\varepsilon_i, \varepsilon_j)}{N-1}, N > 1 \tag{5.10}$$

Here $SR(\varepsilon_i, \varepsilon_j)$ is a metric to quantify the semantic relatedness between the exemplars of two clusters C_i, C_j. Each cluster C_i is ranked based on its exemplar average score and is removed from C_D if its average score, $Ave(\varepsilon_i)$, is below the average of all clusters. $SR(\varepsilon_i, \varepsilon_j)$ is concerned with measuring the relatedness between ε_i and ε_j rather than their latent semantic similarity. For instance, the terms “drug” and “Olympics” are not similar, but, because of their tendency to co-occur together (“drug use” appears frequently in “Olympics” themes), they are judged semantically related. To quantify such relatedness in an unsupervised cross-domain environment, we expand SemCluster to take advantage of Wikipedia, the largest and fastest growing knowledge base. There are a number of approaches that measure semantic relatedness by exploiting Wikipedia. Explicit Semantic Analysis [211] is one of the most accurate Wikipedia-based measures that, to an extent, comes close to the accuracy of a human [249], and, hence, is employed by SemCluster to compute the relatedness of exemplars.

Search space restriction

Relying solely on the centroids of clusters may lead to restricting the search space for finding the best representative phrases in a given document and, consequently, may result in degrading keyphrase extraction recall and/or precision. Suppose we have a valid keyphrase containing a term t_j that is semantically close to a centroid term ε_i. The phrase will not be selected as a candidate keyphrase simply because t_j is not a centroid. This may explain why the Spectral Clustering algorithm outperforms AP in KeyCluster experiments – the former allows multiple terms close to a cluster centroid to be chosen as seeds and accordingly, extends the keyphrase search space. Taking advantage of this observation, SemCluster expands the selection of seeds from AP clustering in a fashion similar to that of spectral clustering. Let $C_D^{'}$ be the final set of clusters resulting from clustering T_D using AP after centroid relatedness average ranking, where $C_D^{'} \subseteq C_D, C_D^{'} = \{C_1, C_2, \ldots, C_k\}$. For each cluster $C_i, i \leq k$, we select its exemplar ε_i as a seed. We regard each member t_j in C_i ($t_j \neq \varepsilon_i$) as an additional seed if $S(\varepsilon_i, t_j) \geq \tau$, where $S(\varepsilon_i, t_j)$ is the computed score stored in $\mathcal{A}$ from the previous step (see section 5.4.3), and τ is a predefined distance threshold specifying how semantically close t_j should be to the centroid ε_i in order to qualify as a seed. We repeat this procedure for all the clusters in $C_D^{'}$ to obtain a set of appropriate seeds from the extended search space.

5.4.6 Candidate phrases extraction and keyphrases selection

After the selection of the seeds, each chunk NP_i in D is scanned by SemCluster. Any sequence of words in NP_i is regarded as a *candidate phrase* if it satisfies the following conditions: (i) it contains a seed, and (ii) it matches any of the following POS-based extraction rules:

- NP_i contains a seed extracted using an E-pattern
- If NP_i contains a seed extracted using a $\mathcal{C}$-pattern, then two cases are considered: if the seed starts with (JJ), then the sequence matching the pattern $(\mathcal{C})*(\text{NN}|\text{NNS})+$ is extracted from NP_i; if the seed starts with (NN), then the sequence matching pattern $(\text{JJ})*(\mathcal{C})+$ is extracted from NP_i)
- If NP_i contains a seed extracted using a $\mathcal{N}$-pattern, then the sequence matches pattern $(\text{JJ})*(\mathcal{N})+$ is extracted from NP_i

Once all NP chunks have been scanned and processed, the step proceeds to the next phase – refining the set of extracted candidate phrases. The refining phase starts by pruning redundant candidate phrases. Two or more candidate phrases may be semantically equivalent but exist in different forms. They may be synonymous phrases, for example, in Figure 5.1, both “Olympics” and “Olympic Games” belong to the synonyms list of the same WordNet synset, or adjective-synonymous phrases. For example, in the Wikipedia article about “Bernard Madoff”[47], there are many candidate phrases which share the same representative seed “fraud”, such as “financial fraud”, “gigantic fraud”, “massive fraud”, and in this case, we keep the first occurring candidate phrase and remove the others. There is also the case of subphrases, as in the example of “Johnson” and “Ben Johnson”. Both phrases contain “Johnson”, so we keep the longer phrase, which is more specific, and discard the shorter one.

By default, refined candidate phrases are selected as appropriate keyphrases for the input document D. However, for documents with moderate content size, the set of output keyphrases may be relatively large, which would affect the algorithm’s performance. To

[47] https://en.wikipedia.org/wiki/Bernard_Madoff

overcome this drawback, we adopt an empirically effective heuristic from [236], where the position that a given candidate term first occurs in a lengthy document in significant in two ways: (i) it is likely that the keyphrase of more importance appears *sooner* in the document than others, and (ii) after a certain location in the document, candidate phrases that appear for the first time, are highly unlikely to be keyphrases. Based on such an empirical heuristic, for a lengthy input document, we predefine a window of size k, and any candidate phrase that occurs beyond a window starting from the first word up to the k^{th} word is disregarded.

5.5 SemCluster component overview

5.5.1 Semantic representations

While the mean of AKE is to extract important keyphrases from an input document, the justified end is to annotate the document with the extracted keyphrases towards modelling its content into a condensed representation that is easier to handle and process. ASA follows a similar approach, but it derives semantic metadata from an input document to model its content into machine-readable semantic representation. Both approaches play vital roles in many management tasks such as document summarization [54], relationships discovery (e.g. classification [250], clustering [251]), and retrieval [252]. SemCluster combines AKE and ASE approaches to offer dual interconnected representations for modelling and metamodeling the contents of input documents on two levels of detail: literal and conceptual.

Keyphrase appropriateness

Tomokiyo et al. [253] suggest that an *appropriate* keyphrase is a semantically and syntactically correct phrase without any unnecessary words, and propose a measure called *phraseness* for quantifying the appropriateness of keyphrases. Similarly, Liu et al. [55] suggest that keyphrases should be *understandable* to humans to qualify as appropriate keyphrases. The authors give an example that the phrase “machine learning” is appropriate whereas the phrase “machine learned” is not. Existing approaches in NLP literature typically reduce the inflectional forms – and sometimes derivationally related forms – of keyphrases to their common bases (*stems*) using a *text stemming* algorithm [217]. An example of stemming is reducing the phrase “international libraries” to “intern librari”. Stemming can dramatically improve string matching based operations [262], however, it drastically degrades the appropriateness of phrases for humans. For example, it is difficult for a user to understand that “intern librari” refers to international libraries. Based on our experiments with SemCluster, we observe that any output keyphrase is appropriate and fully understandable to humans. This is because candidate phrases are initially extracted from an input document using a set of NLP patterns that encode generally accepted linguistic knowledge/feature assumptions [220] (see section 5.4.1). Nevertheless, instead of stemming, SemCluster reduces the constituents of a given keyphrases to its syntactically-correct linguistic base (*lemmas*) using an internal WordNet-based lemmatization approach.

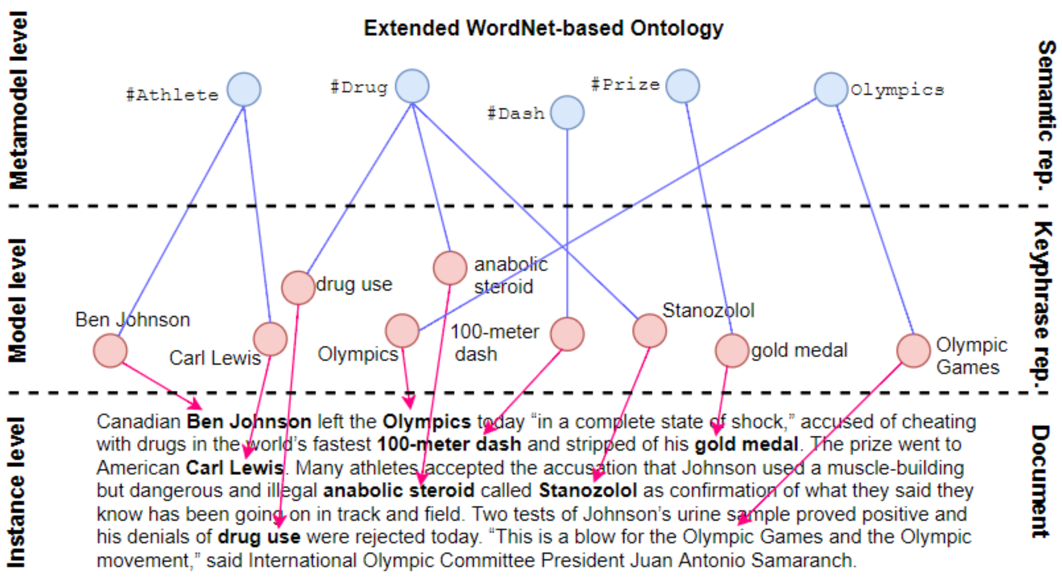
Table 5.2 – SemCluster extraction results from the article depicted in Figure 5.1

Candidate Phrase	Centroid	Seed	Valid
NNP/Ben NNP/Johnson	wn:athlete#1	wn:#9820263	Yes
NNPS/Olympics	wn:olympics#1	wn:#7457126	Yes
JJ/100-meter NN/dash	wn:prize#1	wn:#7469043	Yes
NN/gold NN/medal	wn:prize#1	wn:#3444942	Yes
NNP/Carl NNP/Lewis	wn:athlete#1	wn:#11131135	Yes
NNP/Johnson	wn:athlete#1	wn:#9820263	No
JJ/anabolic NN/steroid	wn:drug#1	wn:#15111116	Yes
JJ/urine NN/sample	wn:olympics#1	wn:#6026635	Yes
NNP/Stanozolol	wn:drug#1	wn:#3247620	Yes
NN/drug NN/use	wn:drug#1	wn:#3247620	Yes
JJ/Olympic NNPS/Games	wn:olympics#1	wn:#7457126	No

To illustrate this, consider applying SemCluster on the news article depicted in Figure 5.1. The results are listed Table 5.2. It can be readily observed that all the keyphrases are human-readable and without any unnecessary words. Generally, keyphrase appropriateness is not important when AKE is an intermediate task in the workflow of an NLP process (e.g. plagiarism detection). However, such importance surges in information retrieval scenarios that involve direct human interaction (e.g. visualization). For example, consider a third-party service plugged in PDL's access layer and it aims to offer a GUI for navigating through the keyphrases maintained by SemCluster. The user could click on a keyphrase to display all its associated documents, or even expanding the results by loading *similar* keyphrases and choosing the most suitable ones to refine/expand the results. The common user here should be able to read each keyphrase in its correct linguistic form.

Semantic metadata

Unlike existing AKE approaches, SemCluster outputs keyphrases that are automatically associated with two types of machine-readable metadata as presented in Table 5.2, which are: (i) the WordNet synset of the seed embedded in the keyphrase, and (ii) the WordNet synset of the cluster's centroid to which the seed belongs. This kind of semantic metadata offers a natural way to *unambiguously* link keyphrases within the same document, or even across independent documents, Furthermore it offers benefits to multiple tasks such as content aggregation and recommendation [255,256], and automatic content relationship identification [254]. In fact the semantic metadata artefacts generated using SemCluster are equivalent to those generated using a traditional ASA tool [257,258] but with higher accuracy, granularity, and richer modelling potential. We can revisit Table 5.2 to illustrate an empirical application of the semantics-based linking: the candidate keyphrases "anabolic steroid", "Stanozolol", and "drug use" are grouped together as they share the common annotation concept `wn:drug#1` from WordNet. Similarly, the keyphrases "Ben Johnson" and "Carl Lewis" are grouped based on the common concept `wn:athlete#1`. Figure 5.4. depicts the semantics-based linking of the keyphrases listed in Table 5.2. The middle layer of the artefacts in the figure (pink circles) reflects the literal representation over the document, whilst the highest layer in the figure (blue circles) reflects the semantic representation over the keyphrases as well as the textual content of the input document. From MMF metamodeling perspective, both layers represent the model, and metamodel levels over the document. The model layer offers common management functionalities (e.g., keyword-based search, browsing, and so forth), whereas the metamodel layer offers far richer functionalities, one among which is semantic-aware content retrieval (i.e. cross-document semantic linking). Consider a semantic search query for retrieving SemCluster-processed documents that contain the search term "Ethylestrenol", (or similar sport-doping drugs[48]). SemCluster recognises this term as a text entity during the processing of documents as discussed in section 5.4.1. Although the term is not available in WordNet, it is maintained as an entry with the labelled type class `Drug` in DBPedia[49].



Canadian **Ben Johnson** left the **Olympics** today "in a complete state of shock," accused of cheating with drugs in the world's fastest **100-meter dash** and stripped of his **gold medal**. The prize went to American **Carl Lewis**. Many athletes accepted the accusation that Johnson used a muscle-building but dangerous and illegal **anabolic steroid** called **Stanozolol** as confirmation of what they said they know has been going on in track and field. Two tests of Johnson's urine sample proved positive and his denials of **drug use** were rejected today. "This is a blow for the Olympic Games and the Olympic movement," said International Olympic Committee President Juan Antonio Samaranch.

Figure 5.4– Unstructured data metamodeling in SemCluster

During retrieval, all the documents associated with this search term are returned as matched query results. These results could be further enriched by adding the news article in Figure 5.1. Though the article does not contain the input search term, it is annotated with the concept `wn:Drug#1` which is the WordNet hypernym for the keyphrases "Ethylestrenol" and "Stanozolol", nevertheless, the gloss definitions of these keyphrases indicate them as *anabolic steroid* substances, and such indication may serve as a strong contextual clue to justify adding the article to the search results.

[48] https://en.wikipedia.org/wiki/List_of_drugs_banned_by_WADA
[49] http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?QueryString=Ethylestrenol

5.5.2 Storage and querying

SemCluster is implemented as an MMF component with a dedicated architecture that offers management functionalities to the common user over the unstructured personal data stored in PDL. Figure 5.5 depicts an overview of this architecture, it consists of three components: SemCluster algorithm, WordNet-based query engine, and the annotations repository. Section 6.2.2 describes the building blocks for a full implementation of SemCluster algorithm. The query engine is a simple semantic search engine that accepts keyphrases and/or WordNet-defined concepts as input search terms, and it retrieves all the documents that are associated with an input search term, with optional query expansion based on the WuPalmer measure given in equation 5.1. The measure expands an input concept based on a specified criteria, which can be either WordNet relation, or semantic similarity score. Annotations repository represents the backend of SemCluster, and it uses a compound index approach where different types of metadata artefacts are stored in separate *direct* or *inverted* indexes similar to Mimir tool in the GATE project [259]. A direct index is key-value table that stores the unique identifier of an input unstructured document in the key field and the sequence of keyphrases/concepts assigned to the document by SemCluster algorithm in value field. In contrast, an inverted table stores a keyphrase in the key field and the identifiers of all the documents wherein the keyphrase occurs in the value field. Note that the metadata information stored in the linkage table is conceptually an inverted index.

Similar to SemLinker, SemCluster, as an MMF component, is physically isolated from the ingestion layer and its single point of input is the lineage manager (see Figure 5.5). Such separation of concern enables the lineage manager to forward raw data to SemCluster with associated lineage information that is necessary to optimise the metadata management and storage tasks. Ingested unstructured data is dispatched to SemCluster with only *key* information, since materializing the by-concept type abstraction for an entity is empirically difficult for common users – compared to structured and semistructured data (section 3.3.2). Because SemCluster does not maintain an internal data parser, input raw data is expected to be in bare textual form, therefore the lineage manager tackles this issue during PDLSF deserialisation stage. Upon receiving the content of an ingested unstructured data entity, SemCluster component invokes its underlying algorithm and the output is a set of keyphrases and their associated metadata artefacts that are stored in the direct and inverted indexes underlying the annotations repository. When metadata processing finishes, the processed data entity and its associated metadata information are dispatched to storage layer and stored using the approach specified in section 3.5.2.

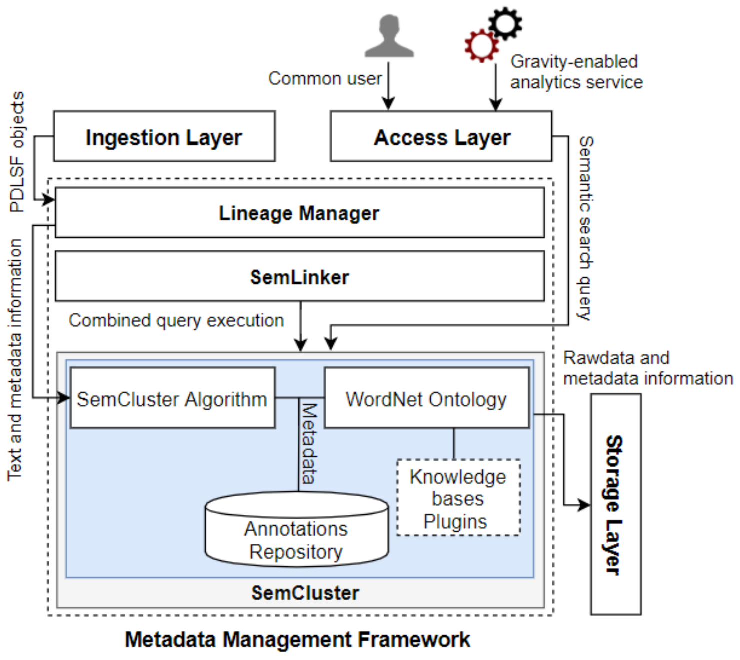


Figure 5.5– Overview of SemCluster component architecture

SemCluster can be queried to retrieve any unstructured personal data stored in PDL based on its associated representations. A SemCluster-specific input query is entered in the query interface within the access layer by the user, a third party service, or SemLinker, and then it is passed to SemCluster for execution over the annotations repository. In following, we describe three kinds of SemCluster-specific query formulations:

1. Keyphrase-based Selection

A keyphrase-based selection query follows the below syntax:

SELECT *key* **FROM** SemCluster **WHERE** <keyphrase> **IN** { $(phrase_0, function_0)$, $(phrase_1, function_1)$, …, $(phrase_n, function_n)$ }

The query modifier key specifies retrieving the unique identifier of each document that matches the selection criteria in the `IN` clause. The internal keyword `<keyphrase>` instructs the query engine to target keyphrase-related indexes only. A selection criteria is a non-empty set of ordered pairs in the form $(phrase, function)$ where $phrase$ is an input search term, and $function$ is a predefined inline function, or null. The following query is an example of a keyphrase-based selection formulation:

```
SELECT key FROM SemCluster WHERE
<keyphrase> IN {("Ben Johnson", NULL)}
```

This query aims to retrieve any textual documents stored in PDL that contain the keyphrase "Ben Johnson". The $function$ part here omitted using the null value.

2. Concept-based Selection

A concept-based query follows the below syntax:

SELECT key **FROM** `SemCluster` **WHERE** `<concept>` **IN**
$\{(concept_0, function_0), (concept_1, function_1), \ldots, (concept_n, function_n)\}$

Similar to the former kind, key specifies retrieving the unique identifier of each document that is annotated with a concept which matches the selection criteria in the `IN` clause. The internal keyword `<concept>` instructs the query engine to target the concept-related indexes only. A selection criteria is a non-empty set of ordered pairs in the form $(concept, function)$ where $concept$ is an input concept drawn from SemCluster's ontology, and $function$ is a predefined inline function, or null. The following query is an example of a concept-based selection formulation:

```
SELECT key FROM SemCluster WHERE
<concept> IN {("wn:Athlete#1", NULL)}
```

This query aims to retrieve any textual documents stored in PDL that are annotated with the first sense of the WordNet concept `athlete`. The $function$ part here omitted using the null value.

3. Combined Selection

The third kind of SemCluster query formulations is called the combined selection and it refers to composing queries by mixing keyphrase-based and concept-based formulations. Composition can be achieved by using the logical operators AND and OR. The following query is an example of such composition:

```
SELECT key FROM SemCluster WHERE
<keyphrase> IN {("drug use", NULL)}
OR <concept> IN {("wn:Drug#1", WuP(0.9))}
```

This query aims to retrieve any documents that contain the keyphrase "drug use" *or* are annotated with WordNet concept `wn:Drug#1`. The query involves passing the selection criteria with an internal function that instructs SemCluster to load all the documents that are annotated with a concept whose semantic similarity with `wn:Drug#1` is 0.9 or higher based on the similarity computation of the WuPalmer measure. Another combined selection query example is depicted below and it has a similar requirement to the former query, but it instructs SemCluster to load any documents that are annotated with a concept related to the input concept through the semantic relationship `hyponym` and with a semantic radius of one edge inside WordNet ontology.

```
SELECT key FROM SemCluster WHERE
<keyphrase> IN {("drug use", NULL)}
AND <concept> IN {("wn:Drug#1", Hyponym(1))}
```

5.6 Summary

In this chapter, we presented SemCluster, the third MMF component that is responsible for unstructured data management in PDL. The main purpose of this component is to overcome the fundamental limitation of SemLinker by adopting an automatic annotation method for enriching unstructured documents with data-driven metadata and materializing explicitly defined representations over them. This component is based on a generic unsupervised clustering-based AKE algorithm that is designed to address the requirements stated in section 5.1. It incorporates extensible background knowledge to identify and extract semantically important terms from an input document, clusters them, and identifies thematically important seeds that are then used to search for representative phrases, and from which appropriate keyphrases are selected. Unlike other unsupervised AKE algorithms, SemCluster outputs are appropriate (understandable) keyphrases that are automatically annotated with formal ontological classes. SemCluster was reported in the computational intelligence literature by two research papers [260,261]. To the best of our knowledge, it is the first approach that dynamically incorporates extensible background knowledge in the extraction task from a collection of integrated knowledge bases. This feature may have multiple applications, one important among which is incorporating personal knowledge sources that are audited by the common user for personalising the AKE task over intimate textual contents for faster retrieval and more convenient integration with data of other structure types.

The presented component models the semantic annotations of unstructured documents into machine-readable semantic representations that can be further utilised for offering various functionalities over the documents. In the context of this research, we are mainly interested in semantic search and bridging-based holistic integration with (semi)structured data for meeting the requirements of various usage workloads that demand data integration across structural heterogeneities.

Chapter 6 Empirical Evaluation

In previous chapters, we explored the proposed MMF, its integration in PDL (Chapter 3), and its principal components: the lineage manager (Chapter 3), SemLinker (Chapter 4), and SemCluster (Chapter 5). While the first component is a traditional data tracking system, the latter two are advanced metadata management systems that are based on several new algorithms and techniques developed during our doctoral research. The aim of this chapter is to evaluate and subsequently validate the performances of SemLinker and SemCluster as part of our adopted DSRM methodology. In this regard, we conduct an interesting set of experiments to rigorously assess the efficiency and effectiveness of SemLinker and SemCluster using disciplined assessment measures. Then, we carry out an experiment to illustrate the utility of MMF in PDL and its relative ease of use for the PDL user. This experiment demonstrates an empirical example for holistic data integration by first combining structured and semistructured raw data and then utilising independent unstructured textual data as backstory information to uncover interesting relationships in the integrated data, thus addressing the recommendations discussed in [1] and [183] which are reviewed in sections 4.1 and 5.1.

Although the experiments presented in this chapter may largely fall in the personal domain, we are not using any personal data that is linkable to a real person, but rather general purpose and publicly available data. Besides the copyright and privacy issues that may stem from the former case, we prefer the latter kind of data to: (i) demonstrate how MMF components can generalise over different data types while taking into account that even general purpose data may be regarded as personal when it can be linked to a data subject under any of Jones' defined senses [1] (see Table 1.1), and (ii) to ensure the accuracy of the evaluation results, particularly for performance benchmarking with the comparative approaches that have been previously evaluated using the same data.

6.1 SemLinker evaluation

The purpose of this section is centered upon the following question: *can SemLinker serve as a standard solution for effectively integrating frequently-changing large-scale heterogeneous datasets and facilitating query-based value creation, while ensuring performance robustness*?”. Although this question addresses the essence of the research efforts presented in Chapter 4, we are also interested in comparing the user-experience of SemLinker against a recent approach [52] that is closely related to our work. To answer this question and to validate the methods covered in Chapter 4, a prototype of SemLinker was developed as a full proof of concept in order to be evaluated in two use cases with real-world data drawn from multiple domains; in the first, we examine the accuracy of SemLinker’s mapping computations on data with substantial heterogeneities, and in the second, we investigate the system’s integration effectiveness and runtime functional complexity on heterogeneous data with frequent and rapid schema evolutions.

6.1.1 Evaluation data

We chose 11 publicly available datasets as the evaluation data in SemLinker experiments (see Table 6.1). These datasets exhibit a high degree of heterogeneity. Each dataset consists of a moderately large number of data entities, and each entity may be of a schema with a different release version. The first 3 datasets – HAR-1, HAR-2, and HAR-3 – contain sensor streams (accelerometer and gyroscope) that are generated by personal devices (smartphones and smartwatches) worn by human subjects, and were collected during the human activity recognition experiments described in [48, 49]. The evaluation data also includes six social media datasets independently collected from several service platforms as shown in Table 6.1. Each among these datasets exists with more than one evolved schema and it may either contain social media posts or a combination of user opinions, reviews, and ratings of popular business establishments (e.g. hotels, restaurants, pubs) in the city of London. The last two datasets are public data published by multiple UK government agencies; the first lists geospatial as well as other details about business establishments in UK, and the second lists the geospatial information associated with every postcode in UK.

Table 6.1 – SemLinker evaluation datasets

Source	Domain	Format	#Attributes	#Items	#Evolutions
HAR-1 [263]	Scientific	CSV	10	3540962	0
HAR-2 [263]	Scientific	CSV	10	3205431	0
HAR-3 [264]	Scientific	CSV	4	200471	0
Facebook [265]	Social	JSON	17	19770	4
Twitter [266]	Social	JSON	19	169000	2
Foursquare [267]	Social	JSON	17	15712	2
Flickr [268]	Social	XML	10	20000	3
TripAdvisor [269]	Social	Spread S.	13	19998	4
Tourpedia [270]	Social	JSON	7	115732	3
EnglandPubs [271]	Public	CSV	9	51566	4
OpenPostCode [272]	Public	CSV	7	2525575	1

6.1.2 Evaluation setup

For simulating personal data ingestion, the actual content of each dataset is split into raw data entities that are serialised into PDLSF objects and temporarily stored in the messaging queue component of PDL's ingestion layer. RabbitMQ [273], an open source[50] message queue software, was used to implement the messaging queue component. We implemented a lineage manager prototype and configured it to automatically pull PDLSF objects from the queue whenever it is idle. The meta section of a pulled object holds the appropriate `source` URI which reflects the unique identifier of its dataset (see section 4.5.1). Our chosen `type` concepts are as follows: we used `sc:Review` to tag "TripAdvisor" and "Tourpedia" datasets, `sc:LocalBusiness` for "EnglandPubs" dataset, and finally

[50] https://www.rabbitmq.com/

`sc:PostalAddress` to tag "OpenPostCode" dataset. Furthermore, $\mathcal{G}$ was extended to include more specific concepts. The concept `sc:SensorReading` is an extension of `sc:Dataset` and was used to tag "HAR-1", "HAR-2", and "HAR-3" datasets. The concept `sioc:Feed` is extenstion of `sc:SocialMediaPosting` and was used to tag "Facebook", "Twitter", "Flickr", and "Foursquare" datasets. We also performed other extensions on the property level of the used concepts, mainly to add the global properties `geo:latitude` and `geo:longitude` whenever they are missing in the above tagging concept, as these properties are heavily utilised in the second experiment. Full information about $\mathcal{G}$ and the extensions used in our evaluations are made publically available in the GitHub repository of SemLinker [274].

One of our main evaluation goals is to investigate the accuracy of SemLinker's mapping computations between the schema of each dataset and its tagging concept. Three schema matcher plugins are used to implement the function Matcher() in the mapping algorithm (see Algorithm 2, line 3). The first two, *SemanticTyper* [93] and *AgreementMaker* [192], are open source schema matching approaches[51,52]. SemanticTyper is an instance-level schema matcher that collects statistical information about data based on their types and decides if two schema elements match. AgreementMaker comprises multiple internal automatic matchers that are grouped into three layers. Each layer uses a different representation and similarity comparison measure, with the third layer being a combination of the other two. For AgreementMaker, because PDL lacks a priori knowledge regarding the native schemas of the ingested data, we use only the first layer which represents features of schema elements (labels, comments, instances, etc) in TF.IDF vectors and computes their similarities using the Cosine Metric or a similar string-based similarty measures (e.g., Edit Distance). The third plugin, SemMatcher, is the system's default matcher, and it is currently implemented as a combination of AgreementMaker, the schema-level matcher, and SemanticTyper, the instance-level matcher, therefore it can be regarded as a linguistics-based approach that measures the similarity between two input schema elements based on their syntactic similarities and the overlapping between their

[51] https://github.com/agreementmaker/agreementmaker
[52] https://github.com/tknandu/SemanticLabelingRepo

textual descriptions that are retrieved from an external schema dictionary [193]. In Chapter 7, we discuss our intended future work to search for a more sophisticated approach (e.g. ML-based) to implement this plugin.

The accuracy of SemLinker output mapping computation is measured by comparing the system's output mappings against *gold standard* mappings using the following equation:

$$Acc(S_{ij}, g) = \frac{M_{SemLinker}(S_{ij}, g)}{M_{Gold}(S_{ij}, g)} \tag{6.1}$$

Where $M_{SemLinker}(S_{ij}, g)$ is the number of correct mappings – between the source schema version j of the dataset i and the global tagging concept g – that are automatically computed by SemLinker, and the total number of gold standard mappings $M_{Gold}(S_{ij}, g)$ between the same constructs. Unfortunately, the evaluation data are utilised beyond their intended application, thereby, we could not find publicly available mappings for the provided schemas. To overcome this technical limitation, we use a specialised schema management tool called Karma[53]. This tool is an open source research project with full details that are covered in the specified reference. To obtain gold standard mappings, we input 100 samples that are taken from each dataset with their default as well as evolved schemas as depicted in Table 6.1. Karma is semiautomatic, thus it requires manual guidance during computations. The final mapping results are manually supervised and therefore we regard them as our gold standard. Besides obtaining SemLinker's mapping accuracy on individual schemas of each dataset using equation 6.1, we are also interested in obtaining the overall accuracy score of the mapping computations for the default schema as well as its subsequent evolved versions for each dataset. To obtain this, we formulate the following equation:

$$Ave(S_i, g) = \frac{\sum_{j=1}^{N} Acc(S_{ij}, g)}{N} \tag{6.2}$$

Where *N* is the total number of evolutions (release versions) of the physical schema of i.

[53] https://github.com/usc-isi-i2/Web-Karma

6.1.3 Automatic mapping management evaluation

The evaluation was run three times, each using a different schema matcher. Figure 6.1 displays the comparison results of the overall precision score when using different schema matchers. The results clearly indicate that the accuracy of a computed mapping is very much determined by the adopted schema matcher plugin. The system's own matcher, SemMatcher, outperforms the other two matchers on most of the datasets. SemanticTyper successfully captures correct matches wherever AgreementMaker fails, and this, to an extent, explains why SemMatcher, which combines the best features of the matchers, gets almost full scores on 6 of the datasets. The fact that SemMatcher is also linguistics-based suggests that, for social and public datasets, by providing proper schema-level linguistic information (e.g., meaningful labels), schema matching can achieve a better precision.

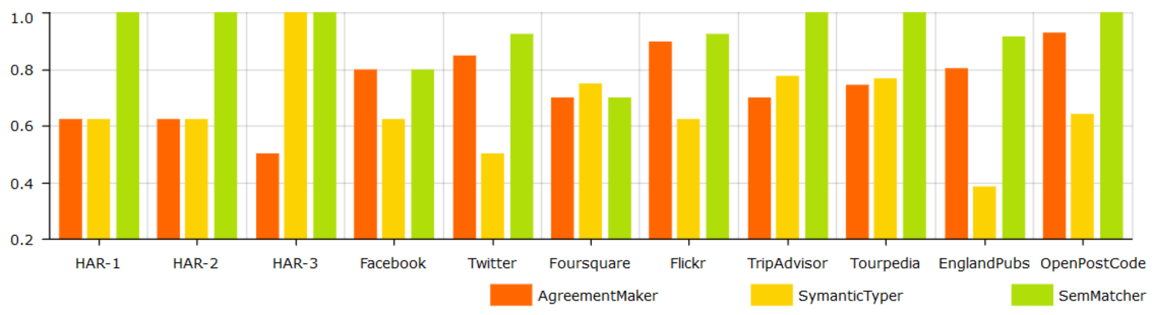


Figure 6.1–The overall mapping precision scores of the adopted matchers

6.1.4 Functional efficiency and query complexity evaluation

To evaluate the functional efficiency of SemLinker in integrating big data with frequently changing schemas and the time complexity of executing queries, we compare SemLinker with a similar integration-oriented and ontology-based prototype system that is used in the SUPERSEDE project and is discussed in [52] (we refer to this as the BDI Ontology system). The BDI Ontology system prototype is implemented using a MongoDB [146] database backend to store JSON data, and SQL to store CSV and XML data. The downside of using the BDI Ontology system is immediately apparent as substantial effort (including manual interactions) is required to maintain its global ontology and to manage the source attributes found in the data collected from data sources. Each schema evolution also requires manual (re)mappings. Two scenarios are used in the evaluation:

Scenario 1 (involving datasets HAR-1, HAR-2, and HAR-3)

It is assumed that user wants to retrieve the gyroscope readings ingested from gyroscope sensors and passes them to a specialised HAR application plugged in the access layer for running HAR analysis workload. To meet this requirement, the following SQL-like query is formulated and executed by SemLinker's query engine:

```
SELECT Sensor.reading

FROM sc:SensorReading Sensor

WHERE Sensor.sensor-type = "gyroscope"
```

A gyroscope reading, such as [0.0041656494, -0.0132751465, 0.006164551], consists of values corresponding to the x, y and z axes. The global concept `SensorReading`, which has one property, `reading`, has been used to tag all three HAR datasets. This apparently simple gyroscope data has some complexity: the readings in HAR-2 are expressed by three separate attributes, *X*, *Y*, and *Z*, whereas the readings in HAR-3 are expressed by only one attribute. However, before the query execution takes place, such heterogeneity problem is already solved when HAR-2 reading were ingested and the schema mappings computed. During mapping, the source schema, $S_{HAR-2,V1.0}$, was virtually transformed into the virtual source attribute "reading". Consequently, the above input query is sufficient to retrieve the required data without any extra pre- or post-processing steps. Regarding the comparative BDI Ontology system, since there is no automatic solution for structural heterogeneities in the source schemas, it is impossible to directly execute the above query. We should either transform the data so that HAR-2 and HAR-3 share the same physical structure, or tag them with different concepts and query them separately.

Scenario 2 (involving social and public datasets**)**:

It is assumed we are interested in local businesses in London such as hotels, restaurants, and pubs, and would like to know their full address (including postcode), and reviews and ratings about them. We may also apply sentiment analysis to gauge the polarity in the comments that are retrieved.

```
SELECT
sc:LocalBusiness.name, Review.reviewRating,
Sentiment(Review.reviewBody),sc:PostalAddress.postcode
FROM sc:Review Review
JOIN sc:LocalBusiness ON
sc:LocalBusiness.name = Normalise(Review.name)
JOIN sc:PostalAddress ON
sc:PostalAddress.latitude =
Radius(sc:LocalBusiness.latitude,5)
AND sc:PostalAddress.longitude =
Radius(sc:LocalBusiness.longitude,7)
WHERE Review.about IN
(WordNet("hotel",1),WordNet("restaurant",1))
AND Review.location = "london"
```

The above query may seem to be complicated. The raw data relevant to the query exists in different formats (structural heterogeneities), with multiple semantic contexts (semantic heterogeneities), and contains instance-level discrepancies (syntactic heterogeneities). While SemLinker handles the first and second kinds of heterogeneities by metamodeling the data after its ingestion by PDL, its query engine enables plugging predefined inline functions that can fulfill certain relevant tasks to resolve any syntactic heterogeneities. To illustrate, there are defects in the actual contents of the relevant datasets, such as a postcode missing from the reviewed business, or some geolocation is inaccurate, or the name of a business may have different spelling (e.g, using *"&"* for "*and*" or "*65*" for "*sixty-five*", and so on). If such problems are though of in advance, as in our case, customized inline functions may be designed and imported prior to the query formulation to deal with these situations at time of query execution. Here we use *Sentiment(string)* to produce a polarity representing the user opinion (i.e., positive, negative, or neutral), *Normalise(string)* to normalise the business names, *Radius(float, integer)* to generate *x* values around an input spatial coordinate, and *WordNet(string,int)* (a WordNet-based function) to retrieve all the possible synonyms and hyponyms of an input string. In addition, there is also the complication regarding schema evolution that has already been dealt with by SemLinker. Once all elements are in place, the user can retrieve the desired information using the query formulation depicted above.

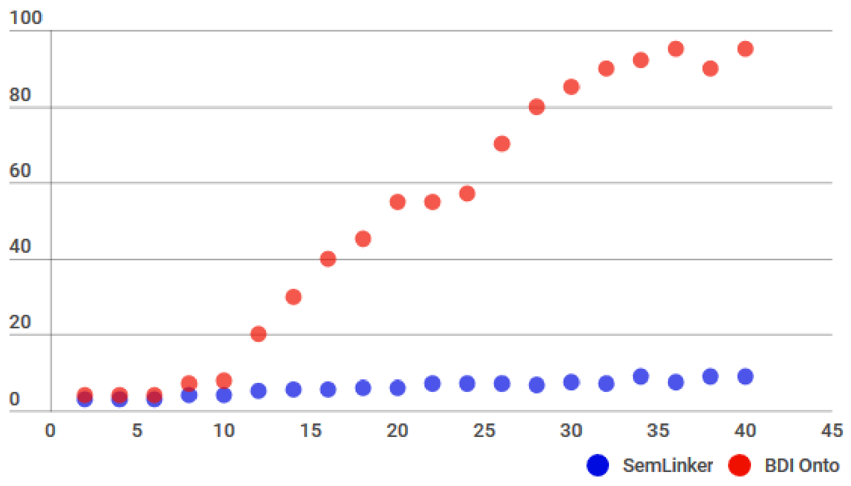


Figure 6.2– Real-time query execution performance comparison

For both scenarios, we ran 20 queries targeting raw data in the range 0-800K data instances on both SemLinker and the BDI Ontology prototype system, and we measure and compare their query execution times. The recorded time for each query includes input query translation, query unfolding, and data retrieval from the backend. Figure 6.2 presents the runtime benchmark data recorded for the query executions of each system. We observe that when the number of datasets is small, the difference in the execution times for the two systems is insignificant, but when the retrieved data are moderately large, SemLinker significantly outperforms the BDI Ontology system. For example, SemLinker requires 8 seconds on average to retrieve and integrate 40K review results, whereas the BDI ontology system requires 96 seconds on average to perform the same task. SemLinker's significant improvements are mainly due to the following reasons:

1. Since SemLinker fully supports the storage, integration, and querying of raw data regardless of its formats and structures, any high-performance key-value store can

be adopted as backend in our experiments. Based on our discussions in section 3.5, we adopt in-memory RDF triple store for housing metadata (i.e. the schemas repository), and Redis server to store raw data (unified repository) and its associated metadata information (linkage table). Compared to the data access and query execution overheads imposed on the BDI Ontology system due to reliance on advanced database systems (i.e. MongoDB and SQL – see [52]), SemLinker's storage technologies are conceptually RAM-based big hash tables with extremely fast data access and retrieval complexities – O(1).

2. In SemLinker, the source schemas of each dataset are modelled as subgraphs grouped into one RDF graph (i.e., the local schema) whose graph context is defined by a single URI reflecting the input source's identifier (see section 4.5.1). In contrast, each source schema in the BDI Ontology system is treated as a separate RDF graph. As expected, SemLinker, which executes its internal SPARQL queries on a single graph for each dataset, is much faster than the BDI Ontology system, which executes its queries on several graphs for each data set. SemLinker's metadata storage-optimised approach, and supported by O(1) retrieval complexity, significantly contributes in boosting the query rewriting and unfolding. In fact, the intermediate SPARQL query processor (see Figure 4.5) adds negligible overheads on the overall SQL-like query execution process.

Housing the schemas repository S in the physical memory may arise performance concern in terms of resource consumption, given the size growth of S overtime is affected by: (i) the continuous addition of new RDF triples reflecting the local schemas of newly added input sources to PDL, along with their associated mappings to the global ontology, and (ii) the continuous addition of new RDF triples reflecting new source schemas generated by SemLinker as result of automatic reactions to schema evolutions (velocity). To investigate this concern, we downloaded 245 datasets from *UCI Machine Learning Repository*[54] and 190 datasets from Kaggle[55]. The datasets belong to various domains with varying sizes and physical schemas. Subsequently, we applied SemLinker on the data collection to generate schema metadata and store it in our volatile triple store underlying S. A Windows

[54] https://archive.ics.uci.edu/ml/index.php
[55] https://www.kaggle.com/datasets

server with 128GB- RAM and E3-1230-V2-3.30GHz-CPU was used in this investigation. We dumped the store's content on the hard disk and found that the total size of S was ~4MB. It is important to note that the size of S, denoted as $|S|$, is solely dependent on SemLinker's RDF data outputs due to the two reasons given above. Let the growth of $|S|$ be denoted as $\triangle$, then $|S|$ after any invocation of Algorithm (1) and Algorithm (2) becomes $|S| \times \triangle$. For example, if we simulate 5 evolutions in the physical schemas of all the evaluation datasets, then then $|S|$ is expected to be ~19MB. Figure 6.3 depicts the overall growth in $|S|$ after applying schematic changes in the physical schemas of the datasets. We have also measured the size shrinkage resulted from the deletion of the source schemas v.5.0 and v.4.0 from each dataset and found that $|S|$ roughly shrunk back to its original size. This empirical finding indicates that it is practically safe to adopt an in-memory RDF triple store as a metadata repository for SemLinker without any technical issues on the long term.

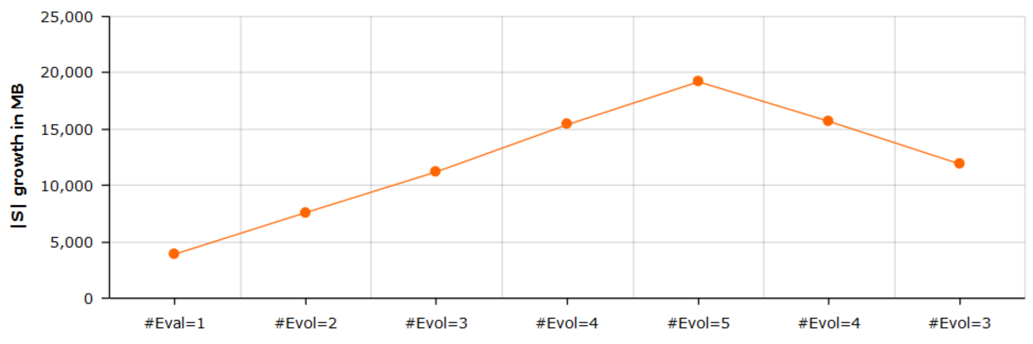


Figure 6.3– Schemas repository size growth correlation with schema evolutions

6.2 SemCluster evaluation

To evaluate SemCluster, two experiments are conducted using two evaluation datasets, and the results are reported in this section. In the first experiment, we examine the impact of SemCluster parameter settings on the keyphrase extraction performance, and provide guidelines for optimal parameter setting in two popular domains. In the second experiment, SemCluster is compared with multiple AKE methods in terms of precision, recall, and F-measure of the reported keyphrases.

6.2.1 Evaluation data and metrics

Two frequently used datasets in AKE literature are chosen as the evaluation datasets: Inspec[56] [220], and DUC-2001[57]. Both datasets consist of free-text documents with manually assigned keyphrases and differ in length and domain (see Table.6.2), and therefore are appropriate to test the robustness of SemCluster AKE performance over documents that belong to different domains.

The Inspec dataset is a collection of abstracts of scientific papers from the Inspec database, consisting of 2000 abstracts. Each abstract is represented by three files: *.abstr, .contr* and *.uncontr*. The file *.abstr* contains the abstract content; *.contr* contains keyphrases restricted to a specific dictionary; and *.uncontr* contains keyphrases freely assigned according to the personal judgements of human curators. In Hulth's work [220], the evaluated AKE method was supervised, and the dataset was split into three partitions: 1000 abstracts for training, 500 for validation, and 500 for testing. TextRank, and KeyCluster are unsupervised methods, and thus only the test partition was used in their evaluations. Since SemCluster is also unsupervised, we adopt a similar approach and use only the test partition to provide a precise comparison with the other AKE methods mentioned. As listed in Table 6.2, the average length of each abstract ($\boldsymbol{W/D}$) is 121.824, and the average number of keyphrases assigned to each abstract ($\boldsymbol{KP/D}$) is 9.826. However, since the manual assignment of keyphrases is uncontrolled, not all the keyphrases in a particular .uncontr file necessarily occur in the corresponding .abstr file. Instead, any phrases regarded by the human curators as suitable are stored in .uncontr as valid keyphrases. For the purpose of this evaluation, we programmatically[58] scan each .uncontr file and filter out any keyphrases that do not occur in the corresponding .abstr file. A similar preprocessing practice has been applied to the dataset during the experimental evaluations of TextRank, ExpandRank [54], and KeyCluster as well as many others. After processing the dataset, the average number of assigned keyphrases ($\boldsymbol{eKP/D}$) drops to 7.726.

[56] https://github.com/alrehamy/SemCluster/data/inspec
[57] http://www-nlpir.nist.gov/projects/duc/guidelines/2001.html
[58] Datasets statistics are calculated using the code at: https://github.com/alrehamy/SemCluster/data/stats

The DUC-2001 dataset is a collection of news articles retrieved from TREC-9, originally consisting of 309 articles with one duplicate (d05a\FBIS-41815 with d05a\FBIS-41815~). The dataset was originally published as a benchmark for document summarization tasks, and [54] have used human curators to manually annotate each article with 10 keyphrases in order to evaluate the ExpandRank algorithm. The Kappa statistic of inter-agreement between the curators regarding manual keyphrase assignments was 0.7, and assignment conflicts were resolved by discussions, and therefore, the $\boldsymbol{KP/D}$ dropped to 8.08. Each article is represented as a .txt file and consists of multiple HTML tags. In our evaluation, we only consider the textual content in text tags (i.e. <text> … </text>).

Table 6.2 – SemCluster evaluation datasets.

Name	Domain	$\#D$	W/D	KP/D	eKP/D
Inspec	Scientific	500	121.824	9.826	7.726
DUC	News	308	740	8.080	-

D: document, W: word, KP: manually assigned keyphrase, eKP: KP exists in the text.

As mentioned earlier, the metrics used for all SemCluster evaluations are *Precision (P)*, *Recall (R)*, and *F-measure (F)*, which are defined as follows:

$$P = \frac{KP_{correct}}{KP_{extracted}}, \qquad R = \frac{KP_{correct}}{KP_{gold}}, \qquad F = 2 \times \frac{P \times R}{P + R} \qquad (6.3)$$

where $KP_{correct}$ is the number of correct keyphrases extracted by SemCluster, $KP_{extract}$ is the total number of keyphrases extracted, and KP_{gold} is the total number of keyphrases manually assigned by human curators, which in our case, are considered the gold standard. An output phrase extracted from a given input document is regarded as a valid keyphrase if it is identical to, semantically equivalent to, or is a sub-phrase of, a gold standard keyphrase manually assigned to the document in any given dataset.

6.2.2 Evaluation setup

SemCluster prerequisites

In the first step of SemCluster, we adopt OpenNLP[59], an opensource and publicly available NLP library, for text pre-processing. The content of an input document is tokenized using a rule-based tokenizer, whereas sentence boundary detection, POS tagging, and chunking are performed using a Maximum Entropy sequence labelling algorithm that utilises large machine learning models trained on corpora in multiple domains. The default background knowledge source of SemCluster is WordNet[60] v.3.1. We perform slight modifications on the *data.noun* and *index.noun* files to accommodate our needs, including re-indexing the original byte-based synsets' indices for faster access, POS-tagging the tokens in each synset's gloss, filtering out any token that is not tagged as a noun or adjective, and lemmatizing the result gloss to improve string-based operations during disambiguation and similarity computations of terms (see sections 5.4.2 and 5.4.3).

To support SemCluster with rich and tractable background information, we adopt two external knowledge bases to reinforce the semantic coverage of WordNet: DBPedia, and BabelNet. For DBPedia integration, its schema ontology is aligned with the WordNet ontology using the alignment algorithm described in section 5.4.1, and the alignment results are made publicly available[61]. For computational efficiency, we adopt a lookup-server[62] that allows DBPedia to be run in a local mode. BabelNet is a lexicalized semantic network that combines and interlinks knowledge facts extracted from many online resources [51], providing unified access to them[63]. Similar to the structure of WordNet, a noun phrase in BabelNet may have one or more synsets, with each synset consisting of a short definition that is often extracted from Wikipedia, and a list of one or more type classes that are expressed as concepts and linked with the noun phrase using an Is-A relationship. Unlike DBPedia, BabelNet utilises WordNet directly as its schema ontology, which makes its integration in SemCluster a straightforward undertaking. Finally, we use

[59] http://opennlp.apache.org
[60] WordNet v.3.1 is available at https://wordnet.princeton.edu/wordnet/download
[61] https://github.com/alrehamy/SemCluster/extensions/dbpedia/alignment
[62] https://github.com/dbpedia/lookup
[63] http://babelnet.org/download

EasyESA[64] as a local server for measuring the relatedness between cluster centroids using the Wikipedia-based ESA metric (see equation 5.10).

Architectural Integration

Similar to SemLinker (section 6.1.2), we implemented SemCluster as full concept of proof system and integrated it as MMF component in PDL. The lineage manager dispatches the actual content D (e.g., pdf file, word document, text file, etc.) of any unstructured PDLSF object pulled from RabbitMQ, and without associated lineage metadata (see section 3.3.4) except its unique identifier. The lineage manager handles the extraction of the object's raw content by utilising Apache Tike or other internal parsers for non-traditional file formats that are not natively supported by Tika.

Comparative methods and parameter setting

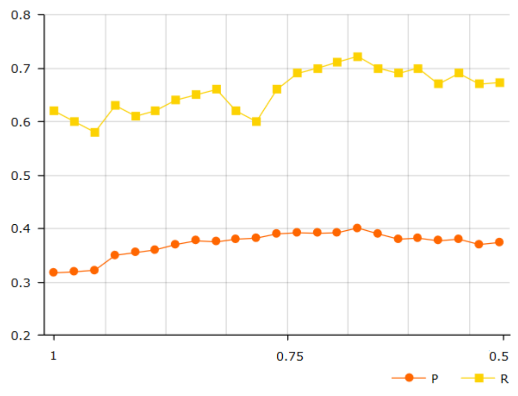
Three unsupervised AKE methods relevant to the SemCluster workflow are selected for comparative evaluation: TextRank, ExpandRank, and KeyCluster. TextRank is a graph-based method that computes the importance scores of candidate words using only *local* structure information embedded in the word graph of the document; ExpandRank is also a graph-based method that exploits an *external* textual neighbourhood in addition to the local structure information of the document's word graph to enhance co-occurrence relations between graph nodes; KeyCluster is a cluster-based method that exploits Wikipedia as an external background knowledge source to capture the semantic relations between candidate terms and compute their pairwise similarities. As discussed in section 5.3, the underlying clustering algorithm of KeyCluster can implemented using any of the following algorithms: Hierarchal Clustering (HC), Spectral Clustering (SC), and Affinity Propagation (AP). Due to the poor performance of HC reported in [55], we evaluate KeyCluster based on only SC and AP implementations.

During the test, only the best results under the best possible parameter settings, if any, for a given method are considered. As shown in Table 6.2, the $\boldsymbol{eKP/D}$ of each dataset is less than 10, therefore we set the co-occurrence window in ExpandRank to 10, whereas for

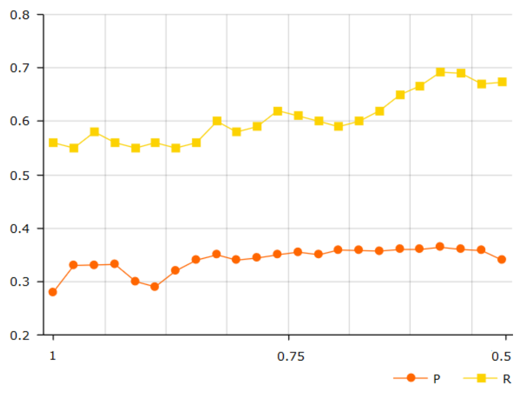
[64] http://treo.deri.ie/easyesa

TextRank, the co-occurrence window size is set to 2 for Inspec, and 5 for DUC-2001. The PageRank dumping factor is a constant value that is used to balance the probability of a random walk from a given node to a random node in the graph. Setting this factor to 0.85 has been shown to be the best empirical setting not only in web surfing [227], but also in keyphrase extraction [53]. For ExpandRank, we set the number of neighbor documents to 5, because for this setting ExpandRank obtains the highest F score. The setting for KeyCluster-SC is that, m, the predefined number of clusters, is $m = \frac{2}{3}n$, where $n = |\boldsymbol{D}|$. For KeyCluster-AP, the maximum number of iterations is set to 1000, the propagation damping factor is set to 0.9, and the clustering preference is computed using $mean$, which has been shown to outperform other preference functions in KeyCluster experiments. Although SemCluster performs AKE in fully automatic mode, it requires general tuning for a set of parameters, which are: (i) WSD context windows size N (section 5.4.2), (ii) AP algorithm parameters (section 5.4.4), (iii) distance threshold τ (section 5.4.5), and (iv) window size k (section 5.4.6). From empirical observation, SemCluster performs the best possible WSD when N=10. However, when N<10, WSD performance degrades, whereas N>10 has no discernable influence on the task. The default tuning of AP parameters is as follows: m is set to 500, and λ is set to 0.9 similar to that of KeyCluster.

As indicated earlier, AP iteratively computes responsibilities and availabilities, and the execution terminates only if decisions for the exemplars and the cluster boundaries are unchanged for *convit* iterations. For computational efficiency, we set *convit* to 50. The custom tuning of AP parameters has no influence on the clustering results regardless of the dataset used during evaluation or its domain, because the input similarities are always positive and in the range [0,1]. Unlike KeyCluster, we choose the $median$ function as SemCluster's clustering preference, to ensure that SemCluster performs clustering with higher granularity (i.e. a larger number of clusters) so that unimportant terms with weak inter-cluster relations can be automatically allocated in unimportant clusters, and hence easily identified and pruned from the clustering results using Equation (10). As shown in Table 2, the $\boldsymbol{W}/\boldsymbol{D}$ of Inspec abstracts is very low, and therefore we set $k = |\boldsymbol{D}|$. Conversely, the $\boldsymbol{W}/\boldsymbol{D}$ of DUC-2001 articles is relatively high, and therefore we set $k = 400$ [236], and, if $k > |\boldsymbol{D}|$ then $k = |\boldsymbol{D}|$.



(a) P/R of runs on Inspec dataset.



(b) P/R of runs on DUC-2001 dataset.

Figure 6.4– τ impact on SemCluster performance using settings in $0.5 \leq \tau < 1$.

The distance threshold τ has a direct influence on SemCluster's performance, such that, when τ=1, only centroids of clusters are chosen as seeds to identify and extract candidate keyphrases; when $\tau = 0$, all the terms in T_D (except those belonging to the pruned clusters) are selected as seeds, and hence most NP chunks in $\boldsymbol{D}$ are chosen as keyphrases. Given that the pairwise semantic similarity score 0.5 is the least extent to which two terms can be judged similar on scale from 0 (dissimilarity) to 1 (identicality) [275], then τ can be assigned any value in the range $0.5 \leq \tau < 1$. Estimating the optimal value of τ is a hyperparameter optimisation problem that can be readily solved either by multiple trials or by employing a dedicated optimisation search algorithm such as Random Search [276]. In this work, we design a sampling-based procedure to infer the best τ setting: from each evaluation dataset we select 100 random documents as inputs to SemCluster, select different τ settings starting from $\tau = 0.99$ and gradually decrease it in the series $\tau_{i+1} = \tau_i - 0.01$, testing the precision and recall of SemCluster's output from each run using the value τ_{i+1}.

The results of our sampling-based trials are plotted in Figure 6.4 for both datasets. As depicted in Figure 6.4a, the precision and recall scores are very low when $\tau > 0.8$, and this is because a relatively large number of important candidate terms are not close enough to their cluster centroids in order to qualify as seeds, and consequently, many valid keyphrases are not identified by SemCluster as construed in section 5.4.5. However, when $\tau < 0.8$, the performance gradually improves as semantically important terms start being qualified as seeds, which contributes towards improving the total number and the quality of extracted keyphrases. SemCluster's best performance (P=0.401, R=0.742) is achieved when $\tau = 0.665$. Similarly, Figure 6.4b depicts SemCluster's performance for the DUC-2001 dataset using different τ settings. A prominent performance improvement is achieved when $\tau < 0.8$, and continues to gradually improve until $\tau = 0.59$, where the best performance (P=0.364, R=0.692) is realized.

Table 6.3 – Performance comparison of SemCluster and other algorithms

Methods	Inspec			DUC-2001		
	P	R	F	P	R	F
TextRank	0.312	0.431	0.362	0.189	0.391	0.127
ExpandRank	0.344	0.471	0.398	0.288	0.354	0.317
KeyClsuter$_{SC}$	0.350	0.660	0.457	0.256	0.529	0.345
KeyCluster$_{AP}$	0.330	0.697	0.448	0.239	0.538	0.331
SemCluster	**0.401**	**0.742**	**0.520**	**0.364**	**0.692**	**0.477**

Bold values indicate the best result for each dataset

6.2.3 Performance comparison and results

Using Inspec and DUC-2001, we compare SemCluster's performance with the methods described in the previous section. Table 6.3 presents the evaluation results of each evaluation dataset in terms of the precision, recall, and F-measure of the extracted keyphrases.

The results show that, for both datasets, SemCluster outperforms the compared methods on the recall of correct keyphrases and the precision of the extracted keyphrases. Comparing with KeyCluster-SC, which has the second-best performance, SemCluster achieves F-measure improvements of ~14% and ~38%, respectively. Although both SemCluster and KeyCluster-AP utilise the same clustering algorithm, the former outperforms the latter with F-measure improvements of ~16% and ~44%, respectively. To the best of our knowledge, SemCluster's F-measure scores of 0.520 and 0.477 are the highest among current state-of-the-art unsupervised cluster-based methods.

The main contributors to the significant improvements in the F measure in SemCluster can be summarized as follows:

1. Given sufficient background knowledge, we extract n-grams from the input document's content as potential candidates, including successfully mapped noun phrases and proper named entities (see Table 5.1), while other state of the art approaches typically extract single words only, causing many potentially important candidates to be either eliminated early or to become semantically inadequate during the selection of terms. For example, instead of selecting "third world" as a candidate term (which is a compound noun manually assigned as a keyphrase for the article *AP880926-0203/* DUC-2001), all comparative methods extract the words "third" and "world" separately. The drawback of n-gram terms selection, however, is that it may lead to keyphrase overgeneration [222]. SemCluster overcomes this issue by eliminating semantically irrelevant candidates during cluster pruning, as discussed in section 5.4.5, thus boosting SemCluster's recall.
2. Although the background knowledge obtained from relevant documents used in ExpandRank, and the vector representation of terms based on Wikipedia articles used in KeyCluster, contribute to enhancing their F scores compared with TextRank, SemCluster's extensible background knowledge is more effective. This is because SemCluster clusters candidate terms based on their latent semantic relations rather than frequency and co-occurrence statistics, and also obtains thematically representative seeds even if they occur infrequently in the input document to improve the keyphrase extraction precision.
3. We observe that expanding seeds with τ equal to 0.665 and 0.59 for Inspec and DUC-2001, respectively, allows SemCluster to extract keyphrases that match the gold standard keyphrases, while KeyCluster fails to identify them because their corresponding seeds often do not qualify as cluster centroids and are thus eliminated from the clustering results. This accounts for the significant improvements in the recall and precision of SemCluster, compared with both implementations of KeyCluster.

It is also noteworthy that SemCluster is more computationally efficient than the other methods, especially KeyCluster. Due to its reliance on WordNet, SemCluster loads the WordNet ontology and any related ontology alignments into its physical memory (WordNet noun files and external ontology mapping files require ~22MB) so that accessing the semantics of a term in $\boldsymbol{D}$ requires O(1) time. Because of this, our method performs AKE with significant improvements in computational complexity compared with other methods. For example, KeyCluster requires ~5M Wikipedia articles to be crawled in order to construct the Wikipedia-based conceptual vector for each term in $\boldsymbol{D}$ during the pairwise similarity computation of terms. Furthermore, Wikipedia crawling is correlated with the length of the input document, whereas SemCluster accesses Wikipedia only for computing the relatedness averaging for cluster centroids, which we have observed often requires less than 15 centroids in the evaluation.

Table 6.4 – Comparison of SemCluster$_{DBP}$ and SemCluster$_{DBP,BN}$.

Versions	Inspec			DUC-2001		
	P	R	F	P	R	F
SemCluster$_{DBP}$	0.392	0.721	0.507	**0.371**	0.639	0.475
SemCluster$_{DBP,BN}$	**0.401**	**0.742**	**0.520**	0.364	**0.692**	**0.477**

* Bold values indicate the best result for each dataset

One of the main contributions of SemCluster is the way that background knowledge extensibility is leveraged to overcome knowledge and semantic losses. To evaluate the impact of knowledge extensibility on SemCluster performance, we produced two implementations of SemCluster. In the first implementation, denoted as SemCluster$_{DBP}$, we extend WordNet using DBPedia only, and in the second version, denoted as SemCluster$_{DBP,BN}$, WordNet is extended with DBPedia as well as BabelNet. Table 6.4 presents a performance comparison between these implementations using the same evaluation datasets and settings described above. These empirical results indicate that SemCluster$_{DBP,BN}$ outperforms SemCluster$_{DBP}$ in all the metrics except for the precision metric on DUC-2001. Although the improvements in SemCluster$_{DBP,BN}$ performance are not significant, they provide empirical evidence that background knowledge extensibility can enhance the AKE performance of the unsupervised clustering-based method.

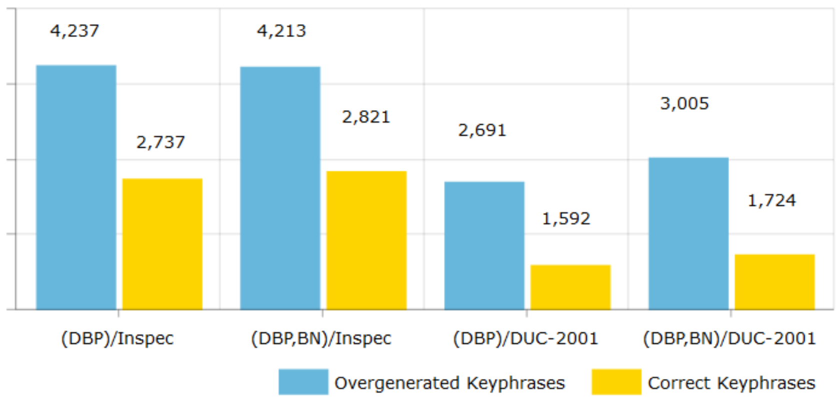


Figure 6.5– Influence of background knowledge on the keyphrase extraction result

As depicted in Figure 6.5, it can be readily seen that both SemCluster implementations perform AKE more efficiently on Inspec than DUC-2001. This performance aspect is also shared by all the comparator methods as presented in Table 6.3. In SemCluster, this may be explained as follows: (i) as presented in Table 6.2, Inspec documents are shorter than their DUC-2001 counterparts, such that, for any given document $\boldsymbol{D}$, $k = |\boldsymbol{D}|$, whereas for DUC-2001 documents, $k = 400$, and accordingly, valid keyphrases that occur outside the window k are eliminated in the early steps of the SemCluster workflow, leading to degraded recall; (ii) Inspec documents contain more scientific and technical noun phrases than DUC-2001, many of which have matching entries in BabelNet, and therefore are picked by SemCluster as valid keyphrases, thus boosting both implementation's F scores. In contrast, a news article in DUC-2001 often contains more named entities that tend to have high semantic similarities due to infrequent topic shifting or changing [277], consequently leading to the over generation of keyphrases and degraded recall.

6.3 Holistic data integration

In this section, we demonstrate the utility of the proposed MMF in supporting holistic-integration over heterogeneous personal data within PDL environment. The demonstration is conducted as a multi-step analysis workload over a collection of datasets with severe structural heterogeneities in the art domain.

Table 6.5 – Integration experimental datasets

Name	Format	#Attributes	#Size	Structure
Tate-Art [278]	CSV	20	69203	Structured
Tate-Artist [279]	JSON	4	3533	Semistructured
Tate-Text [280]	Text	0	850	Unstructured

6.3.1 Experimental data

The data used in this experiment consists of the datasets listed in Table 6.5, the first and second are publicly available data offered by *Tate*[65], a UK institution that houses British and international modern and contemporary artefacts. The dataset Tate-Art is a CSV file that consists of a collection of records, each record stores a set of attributes describing an artwork that is displayed on Tate website. Although we designate this dataset as structured in Table 6.5, it is not neatly authored since many CSV records contain missing or erroneous field values, and in an empirical sense the dataset may be regarded as semistructured. The Tate-Artist dataset consists of JSON documents, each document describes the personal details of the artists who crafted the artworks listed in Tate-Art. The Tate-Text dataset is a collection of unstructured documents, each document contains a textual content that briefly describes an artwork in Tate-Art. This dataset contains 850 documents collected from Tate website via web crawling. The website offers descriptions for almost all the artworks, however we crawled only those with free license to ensure using the crawled data under appropriate copyrights. It is important to note that the filename of a document is identical to the title of its corresponding artwork in Tate-Artwork.

[65] http://www.tate.org.uk

6.3.2 Experimental setup

The experimental setup involves using the implementations discussed in sections 6.1.2 and 6.2.2 for ingesting raw data in PDLSF format, storing it in RabbitMQ, parsing it by the lineage manager, dispatching it to the appropriate metadata-processing component in MMF, and finally storing it in Redis server. The same parameter settings used earlier for SemLinker and SemCluster are adopted in this experiment. Regarding data ingestion in PDL, each dataset is split into raw data entities that are serialised in PDLSF format and stored in the messaging queue component. The settings for the attributes in the meta section of a PDLSF object are listed in Table 6.6. To illustrate, a PDLSF object that belong to Tate-Art holds the URI *http://www.tate.org.uk/art* in the *source* attribute of its meta section, and is tagged with the concept `sc:VisualArtwork` to reflect the *type* abstraction of its raw content relative to Schema.org. As listed in Table 6.5, Tate-Art is structured, therefore all its PDLSF objects hold the value 1 in their *assert* meta attribute. In a real-world scenario, the attributes listed in Table 6.6 are expected to be provided by the ingestion agent, and are automatically obtained by the lineage manager during PDLSF parsing as indicated in section 3.4.3. Given a PDLSF object that belongs to Tate-Art (or Tate-Artist), it is normally processed by SemLinker, which initiates the workflow covered in Chapter 4 to generate a source schema that reflects the physical schema of the dataset which the input PDLSF object belongs to. For example, applying Algorithms (1) and (2) on Tate-Art entails one-to-one mappings between the physical schema of this dataset and its tagging concept `sc:VisualArtwork`. A fragment of the mapping results is depicted in Figure 6.6.

Table 6.6 – Meta section settings for Tate data

Dataset	*source*	*type*	*context*	*assert*
Tate-Art	www.tate.org.uk/art	`sc:VisualArtwork`	--	1
Tate-Artist	www.tate.org.uk /artist	`sc:Artist`	--	2
Tate-Text	www.tate.org.uk /art-text	`Null`	--	3

Likewise, a PDLSF object that belongs to Tate-Text holds URI *http://www.tate.org.uk/art-text* in the *source* attribute of its meta section. The *type* attribute is set to null since the dataset is unstructured (see section 3.3.2). Essentially, the value 3 in the *assert* attribute for any object belongs to Tate-Text implies that the lineage manager should dispatch the object to SemCluster as bare. After processing and storing Tate datasets in PDL, the query interface is used to submit input queries with three requirements: (i) structured and semistructured data retrieval, (ii) unstructured data retrieval, and (iii) cross-referencing based insight mining.

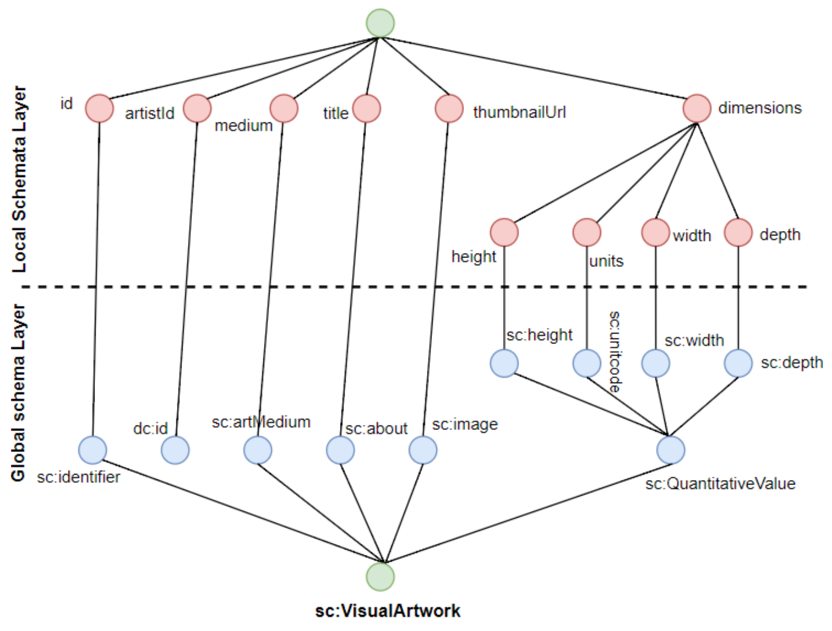


Figure 6.6– The metamodeling of Tate-Art source schema to `sc:VisualArtwork`

6.3.3 Experiment Scenario

Initially, it is assumed that we are interested in associating each artwork in Tate-Art with the personal details of its creator(s) from Tate-Artist. To familiarize ourselves with the elements of the unified view that is produced by SemLinker for each among these datasets, we need to submit the following exploratory queries to SemLinker's query engine through the query interface in PDL's access layer:

```
SELECT * FROM sc:VisualArtwork WHERE
sc:VisualArtwork.root =
"http://www.tate.org.uk/art"
```

And subsequently

```
SELECT * FROM sc:Artist WHERE
sc:Artist.root =
"http://www.tate.org.uk/artist"
```

The first query aims to retrieve the unified view over Tate-Art which represents the metamodeling of its local schema to the concept `sc:VisualArtwork`. Likewise, the second query retrieves the unified view reflecting the metamodeling of the local schema of Tate-Artist to its tagging concept `sc:Artist`. For example, executing the first query yields loading all the records of Tate-Art that are stored in PDL with their native schema – which is depicted in the high level of Figure 6.6 (pink circles) – however, the records are returned as query results in a source schema that is organized according to the ontological structure of the tagging concept – which is depicted in the low level of Figure 6.6 (blue circles). By playing the role of a PDL user, or even a third party data consumer, we assume no prior knowledge regarding the access, storage, or retrieval details of Tate data, therefore executing these queries enables us to learn the necessary information for posing more complex queries. With such information at hand, one can readily formulate the following query to automatically integrate the datasets by joining their unified views.

```
SELECT
sc:VisualArtwork.identifier, sc:VisualArtwork.about,
sc:VisualArtwork.dateCreated,
sc:VisualArtwork.image, sc:Artist.id,
sc:Artist.name, sc:Artist.gender, sc:Artist.url
FROM sc:VisualArtwork
```

```
JOIN sc:Artist
ON sc:Artist.id = sc:VisualArtwork.id
AND sc:Artist.root= "http://www.tate.org.uk/artist"
WHERE sc:VisualArtwork.root =
"http://www.tate.org.uk/art"
```

Here the attention is tuned to specific attributes of the data, namely:

- The identifier of the artwork: `identifier`⟶ id.
- The name of the artwork: `about`⟶ $title$.
- The creation date of the artwork: `dateCreated`⟶ $dateText$.
- The thumbnail image of the artwork: `image`⟶ $thumbnailUrl$.
- The identifier of the artist(s) who created the artwork: `id`⟶ id.
- The name of the artist: `name`⟶ $name$
- The gender of the artist: `geneder`⟶ $geneder$.
- The photo of the artist: `url`⟶ url.

A pair `property`⟶ $attribute$ is an interpretation of a formal GAV mapping from a property in the tagging concept (view) to its semantically corresponding attribute in the physical schema of the meta-modelled data. Such pair is equivalent to the RDP mapping triple $\langle property\ M{:}\,mapsTo\ attribute\rangle$. One important advantage of the above query is that the retuned results may uncover obvious relationships between the data of Tate-Art and Tate-Artist, for instance, what artworks are created by a given artist (i.e. grouping). Therefore, we are able to jointly query heterogeneous raw data that exists in different schemas and formats (CSV and JSON), and process them on the metadata level without the need to change their representations on the physical level. However, such querying is incapable of revealing any *hidden* relationships that might exist between the integrated datasets. By exploiting the holistic integration feature offered by the proposed MMF, the next requirement is to mine hidden relationships through cross-referencing. To meet this requirement, one would plug the Tate-Text dataset as backstory information in the current usage workload. An interesting observation in SemCluster's processed data is that: the phrase *Tulane University* is extracted as keyphrase from multiple unstructured documents.

To test the applicability of this keyphrase as backstory information, one can formulate the following query:

```
SELECT
sc:VisualArtwork.identifier, sc:VisualArtwork.about,
sc:VisualArtwork.dateCreated, sc:VisualArtwork.image,
sc:Artist.id, sc:Artist.name, sc:Artist.gender,
sc:Artist.url
FROM sc:VisualArtwork
JOIN sc:Artist
ON sc:Artist.id = sc:VisualArtwork.id
AND sc:Artist.root= ″http://www.tate.org.uk/artist″
WHERE sc:VisualArtwork.root =
‶http://www.tate.org.uk/art″
AND sc:VisualArtwork.about IN
 #SemCluster:(
  SELECT Lineage.filename(key) FROM SemCluster
  WHERE <keyphrase> IN {(‶Tulane University″, NULL)}
 )
```

From a technical viewpoint, this query is identical to its former counterpart but with one difference: there is a new SQL-like filtering statement added in the *Where* clause. When SemLinker detects the syntax `#SemCluster:(…)` during the compilation of this query, it understands that the query formulation inside the brackets is meant to be executed by SemCluster, accordingly, its execution is postponed by SemLinker's query engine until the rest of the input query is executed and the query results are loaded from the metadata repository, the linkage table, and the unified repository. Once the results are ready, the query formulation inside `#SemCluster:(…)` is sent from SemLinker to SemCluster for execution, and its results are sent back to SemLinker , which are then used to filtering the loaded data.

Table 6.7 – Documents associated with "Tulane University" in Tate-Text dataset

Key (Lineage)	Filename	Contains
6B86B273FF	From the Freud Museum	Tulane University
D4735E3A26	Trestle Trains	Tulane University
4E07408562	Elevated Smoke	Tulane University
4B227777D4	Burial at Sea	Tulane University
EF2D127DE3	Diamond Express	Tulane University
E7F6C01176	Cubie Smoke	Tulane University

To illustrate this workflow, SemLinker executes the entire formulation of the above query (except for part `#SemCluster:(…))` to integrate the data of Tate-Art and Tate-Artist. Subsequently, the SemCluster-specific formulation is submitted to SemCluster in order to find all the documents that are annotated with the keyphrase *Tulane University*. The underlying information need here is to load the value associated with the *filename* attribute (lineage metadata) of each matched document. As indicated in section 6.3.2, the filename of each document in the Tate-Text dataset is identical to the *title* attribute in the Tate-Art dataset. These values can be exploited to filter the Tate-Art/Tate-Artist integrated data. Table 6.7 lists the results of executing the SemCluster-specific query on Tate-Text data. The first column (*key*) represents the unique identifiers assigned to a document by the lineage manager during its ingestion, the second column (*Filename*) lists the original filename of the document, and the third column (*contains*) indicates that the document contains the keyphrase specified in the input query. SemLinker utilises these results to refine the data loaded from executing the original input query. The refined results are depicted in Figure 6.7. In this visualization figure[66], we found that 6 artworks in Tate-Art are associated with the keyphrase *Tulane University*, 5 among which were crafted by *William Crutshfield*, and the last was crafted by *Susan Hiller*. This finding indicates that

[66] We designed a program to draw a visualization of the query results to simplify their understanding.

there is a hidden relationship between these artists. By examining the documents listed in Table 6.7, we learned that both artists studied at Tulane University and this is why their artworks seem to be related. Based on this experiment, it is empirically evident that querying over well-managed and holistically integrated raw data is effective and can lead to reveal hidden patterns in the queried data towards generating new knowledge and deriving value.

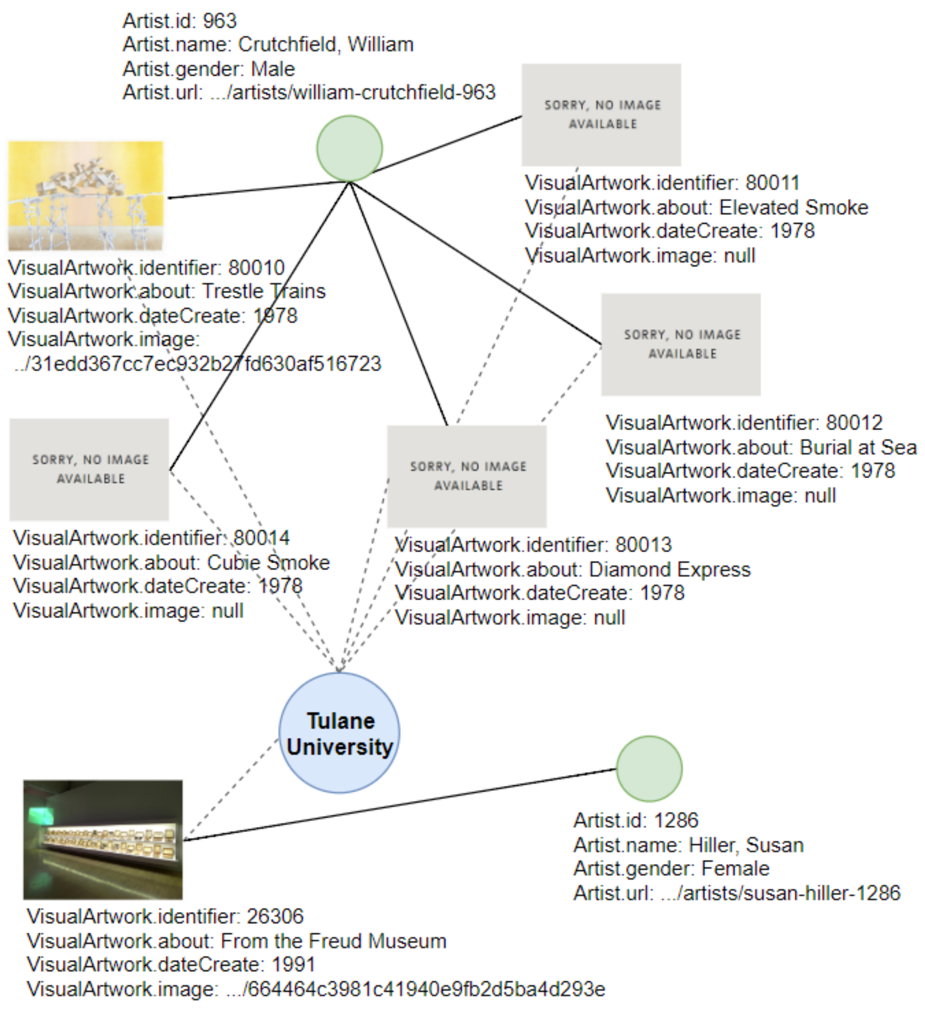

Artist.id: 963
Artist.name: Crutchfield, William
Artist.gender: Male
Artist.url: .../artists/william-crutchfield-963
SORRY, NO IMAGE AVAILABLE
VisualArtwork.identifier: 80011
VisualArtwork.about: Elevated Smoke
VisualArtwork.dateCreate: 1978
VisualArtwork.image: null
VisualArtwork.identifier: 80010
VisualArtwork.about: Trestle Trains
VisualArtwork.dateCreate: 1978
VisualArtwork.image: ../31edd367cc7ec932b27fd630af516723
SORRY, NO IMAGE AVAILABLE
VisualArtwork.identifier: 80012
VisualArtwork.about: Burial at Sea
VisualArtwork.dateCreate: 1978
VisualArtwork.image: null
SORRY, NO IMAGE AVAILABLE
VisualArtwork.identifier: 80014
VisualArtwork.about: Cubie Smoke
VisualArtwork.dateCreate: 1978
VisualArtwork.image: null
SORRY, NO IMAGE AVAILABLE
VisualArtwork.identifier: 80013
VisualArtwork.about: Diamond Express
VisualArtwork.dateCreate: 1978
VisualArtwork.image: null
Tulane University
Artist.id: 1286
Artist.name: Hiller, Susan
Artist.gender: Female
Artist.url: .../artists/susan-hiller-1286
VisualArtwork.identifier: 26306
VisualArtwork.about: From the Freud Museum
VisualArtwork.dateCreate: 1991
VisualArtwork.image: .../664464c3981c41940e9fb2d5ba4d293e

Figure 6.7– Visualization of the holistic integration querying results

6.4 Summary

The main methods and algorithms introduced in this thesis are empirically validated and evaluated, using proof of concept implementations and a multitude of real-world datasets. We conducted two experiments using 11 datasets to evaluate SemLinker. The first tested the robustness of automatic schema mapping computation on 3 semantically similar but schematically heterogeneous datasets, and using one tagging concept that is drawn from Schema.org. The second experiment tested the integration and querying performance of SemLinker against a current state of the art method and using 8 heterogeneous datasets. During this experiment, we simulated evolutions in the physical schemas of the integrated data to examine the tractability of the proposed system in handling data velocity. The results obtained from both experiments not only validate the integration effectiveness and functional efficiency of the system, but also indicate that the performance is robust and promising. SemLinker stores output metadata in an "in-memory" triple store, this aspect requires empirical feasibility test. We applied SemLinker on a collection of 435 datasets and recorded the initial memory consumption rate and the consumption increment after each simulated evolution wave in the physical schemas of the datasets. The results indicate that the memory consumption is insignificant and it is practical to implement SemLinker's metadata repository using any in-memory store.

Next, we conducted one main experiment to validate SemCluster and evaluate multiple aspects of its performance, namely: keyphrase extraction recall, precision, F1-measure, tractability across domains, and the effect of background knowledge on the keyphrase extraction task. Popular datasets in NLP literature were used in the experiment. The performance of SemCluster was compared against 3 AKE methods under the best possible parameter settings. The results indicate that SemCluster outperforms the comparative methods with significant performance improvements. Finally, we conduct a casual experiment to demonstrate the utility of MMF in performing holistic integration over semantically similar but structurally heterogeneous datasets. We showed the usefulness of the proposed MMF as a foundation for querying heterogeneous personal raw data and discovering new interesting insights from it through cross-referencing.

Chapter 7 Conclusions and Future Work

There is an evident need for common users to have their own self-controlled personal spaces in a response to the pressing problems of digital autonomy and the asymmetry of power between the user and the large-scale service providers (and any other third-party personal data consumers for that matter). In [11], the authors discuss the trending paradigm of provider-centric data centralisation and its implications on those users who want to manage their own data and derive value from it. In [281], the authors argue that the missing governance of personal data markets threatens to undermine the common user's trust in data sharing practice, given that data sharing underlies not only a series of valuable public services, but also the whole digital industry. Motivated by these as well as similar ideas, we designed PDL as a highly suggestive mean for putting the common user at the centre of personal data management, and market. PDL enables its user to accumulate large amounts of a potentially useful personal data in a central space, which the user has full control on, and which serves as a platform for value creation and fair information exchange with third-party data consumers, to serve various gainful purposes as conceived by the user. The PDL solution naturally solves the known fragmentation problem of personal data, albeit the centralised data within it may quickly exhibit qualitative characteristics that are identical to the 3Vs model of big data, but on a smaller scale; the scale of an individual. Common users obviously do not have sufficient knowledge to deal with the 3Vs characteristics or the inherent isolations stem from the heterogeneity of personal data, and even if they do, it is still a tedious and time consuming manual undertaking. In this thesis, we argue that equipping PDL with an automated solution for meta data management can address all the problems of personal data. To verify this argument, we propose a novel extensible MMF that operates on the metadata level of the personal data to orchestrate its storage and usage within PDL environment, thus enabling the PDL user to integrate, query, and analyse personal data in its native representations and with minimum time and efforts.

7.1 Conclusions

This thesis introduces an extensible MMF to annotate (semi)structured and unstructured personal raw data, that are collected from a plurality of heterogeneous data sources, with lineage and semantic metadata artefacts, which can then be exploited to model the data based on a formal metamodel. Throughout this research we designed, implemented, and evaluated multiple algorithms and techniques for automating many downstream tasks in the metadata and data management workflows with a focal aim of isolating the PDL user from the complex technical details typically imposed by these tasks. We showed that such automation enables the user to focus on knowledge discovery and value creation through formulating and executing formal queries over MMF's metamodel.

The proposed MMF consists of three solutions for data provenance; (semi)structured data management, and unstructured data management. The first solution, lineage manager, is a data tracking system that focuses on recording information about the line, age, and privacy of the data in PDL environment. This system controls who can access the stored data, and on what levels of accessibility, thus it provides agile but effective mechanism to protect the user privacy and prevent unauthorized access to the stored data. The second solution, called SemLinker, is an ontology-based data integration system that follows Lenzerini theoretical integration framework [39]. It consists of (i) an extensible OWL ontology that serves as a global schema, (ii) an RDF-based metadata repository to store local schemas that correspond to the physical schemas of the (semi)structured personal data in PDL, and (iii) formal mapping specifications that define the correspondences between the global and the local schemas. SemLinker focuses on automating data integration and offering partial unified views over PDL data on the metadata level. Such views can be readily qureied by analysis workloads. During our initial experiments with SemLinker, we observed that the velocity of personal data is faster, in terms of schema evolution, than what we anticipated at an early stage of our research. For example, the schema of Facebook Graph API has evolved three times over the last year (v.2.8, v.2.9, v.3.0). This provides the following insight: an efficient integration system that operates in a DL system needs to effectively address many challenges, one compelling among which is schema evolution. Tackling this challenge requires self-adaptability to external dynamics imposed by the integrated data sources, and with lack of such feature then automating the data integration workflow may become a near-impossible undertaking. Accordingly, we redefine the “local schema” as a collection of source schemas, each among which corresponds to the physical schema of the given source at a certain point in its schema evolution timeline. This definition enables SemLinker to readily automate the integration process, but it introduces a new challenge; there are always structural and/or semantic differences between the current schema version and its previous counterpart, otherwise evolution would not take place. It is critical for an automated integration system to identify and reconcile such heterogeneities to prevent the tasks operating on the schema from crashing. The fact that raw data must be stored by a DL system in untransformed state complicates this challenge as we are forced to come up with a solution to the heterogeneity reconciliation without applying physical transformations on the data. To address such a solution, we introduce the idea of virtual transformations on the metadata level of the data. In Chapter 6, we evaluated all the proposed techniques in SemLinker to validate them and to examine the overall system performance. Two experiments are conducted to serve these requirements. The evaluation results not only validate the automated integration effectiveness and efficiency but also indicate that the performance is robust and promising. Query execution is really fast, in fact, SemLinker is ~10 times faster than a comparative recently introduced method. The metamodeling approach of SemLinker and its flexibility in supporting schema-less storage are major contributors to the performance improvements.

The third solution, called SemCluster, is an ontology-based AKE-based ASA system for automatically annotating unstructured textual data with semantic data-driven metadata. The system consists of: (i) an extensible global WordNet-based ontology that serves as a formal metamodel, (ii) a repository of keyphrases and their associated metadata artefacts, which serves as a collection of semantic representations over the textual data, and (iii) formal mappings between the semantic representations and the global ontology, which are WordNet-defined relationships. The underlying algorithm of the system is an AKE six-step algorithm that automatically extracts thematically important keyphrases from the body of an input document, and associates each extracted keyphrase with fine-grained ontological information drawn from the global ontology. The presented algorithm is the first of its kind, it dynamically incorporates external background information in the AKE task, which is obtained from generic, specialised, or personalised knowledge sources that are integrated with WordNet through agile alignment on the schema level, and which can be systematically queried using an internal knowledge querying algorithm. The ASA performance of the system is solely dependent on its underlying AKE algorithm, therefore, we evaluated SemCluster using datasets of different size and domain, against three comparative methods. The evaluation results indicate that SemCluster outperforms the compared methods across all the evaluation metrics. This empirical finding verifies the findings of Liu et. al. [55] that unsupervised clustering-based AKE methods can be effective and robust, even across multiple domains. Background knowledge plays vital role in improving the AKE, however, we needed to empirically verify this observation. Accordingly, we conducted an experiment to compare between the performances of two implementations of SemCluster, in the first we plugged only one external knowledge source (DBPedia), and the second we added another source (BabelNet). The F1-measure scores indicate that there is performance improvement in the second implementation, but it is not significant due to technical issues related to the domains of the experimental data.

To demonstrate the usefulness of MMF and its roles in simplifying the usability of PDL, we conducted an experiment where the metadata of all MMF components are combined and jointly queried to perform a personal analytics workload over holistically integrated heterogeneous personal data. The goal of this experiment was to reveal hidden insights in 3 datasets that suffer severe structural heterogeneity. The scenario of the experiment involved extracting lineage information from the datasets by the lineage manager, then applying SemLinker on 2 (semi)structured datasets to generate partial unified views over their native representations, and applying SemCluster on a collection of unstructured text documents to construct keyphrase-metadata representation for each document, and finally querying MMF with a combined SQL-like query that utilises the keyphrases of the unstructured documents as backstory information about particular entities in the integrated data. Using relatively simple query formulations, it was applicable to discover obvious and hidden relationships between the structured and semistructured datasets. Overall, the evaluations presented in Chapter 6 demonstrate remarkable potential for the PDL user to understand, manage, and utilise personal raw data through MMF-based queries.

Although most of our discussions tie MMF with PDL, the framework is in fact loosely coupled with PDL architecture in two ways: first, MMF is not concerned with the workflow details of the ingestion layer since it only requires pulling raw data entities from a queue (e.g. the messaging queue component). Secondly, MMF is not concerned with the implementation details of the storage layer, it expects the PDL backend to be a giant hash table that could be modelled into two logical components: the unified repository and the linkage table. MMF handles data storage by passing the metadata-processed data entities to the backend in the form of key-value pairs of raw data and their associated metadata information. Furthermore, MMF introduces querying capability by means of an ontology-based metamodel for query formulation, and internal engines to execute input queries on the metadata level of the stored data regardless of its native schemas, structures, or formats. By eliminating the need for external support to data storage and querying, MMF offers a natural support to database agnosticism in DLs. Based on these architectural independence characteristics, and motivated by the experimental results elaborated above, we plan to examine the applicability of MMF in different DL architectures. Our initial aim is to integrate the framework as a management solution for a full-fledged Hadoop DL system and study its empirical effectiveness and efficiency.

The results of this work support the claim that the use of lineage and semantic metadata for the purpose of data management can compensate the inherent lack of data governance and quality in the data lake concept. However, these kinds of metadata are not sufficient enough as an effective mechanism against potential security risks. PDL preserves all the personal data of the user in a central space that is controlled by the user, not a third party. Currently, MMF enables the PDL user to specify basic metadata-based access control for various kinds of personal data as described in section 3.4.3. A data consumer who approaches PDL for mining useful information must request access to the stored data from the user. If the request sounds invasive, the user can simply reject it. However, the potential risks arise upon requests approval. A nefarious third party with permissible access to a particular kind of personal data could formulate MMF-based queries that elicit more data pieces than the PDL user intends to disclose, for instance, by allowing a third party geolocation service to access GPS history data, it may conduct pattern mining to infer the user’s current location without the consent of that user. Accordingly, equipping PDL with safeguards against information breaches through using and mining personal data is critical and has to evolve in parallel with considerations about the requirements of each third party data consumer and the user's awareness of those requirements, for example, an effective safeguard should allow the service in the former example to retrieve GPS data up to a certain point in time, thus preventing it from predicting the current location of the user.

Another security risk in the user-centric data centralisation paradigm lies in the safety of the personal data in the long term. PDL collects a variety of raw data from different data sources, many among which are not necessarily guaranteed to provide risk-free data, for instance, a RAS agent running on a device infected with a malicious software (e.g. Trojan, Ransomware, etc.) may propagate infected files to the PDL, which once are situated within the PDL premises, they can contaminate other preserved files, leading to irrevocable corruption of particular personal data that may be regarded of high importance to the user. Therefore, PDL should also be equipped with a mechanism that scans any ingested raw data entities and filters out those with potential security bugs.

7.2 Future work directions

Future work could take many directions. In the following subsections, we summarize the pressing ones.

7.2.1 Simplifying query formulation

Gartner [283] outlined personal analytics in its *hype cycle* of emerging technologies. The firm believes this technology is promising and may attract competitive advantages in the foreseen future. We share the same expectation and further argue that as the size and convergence of personal data increasingly become a norm, the common users' need to derive personal insights and patterns from their data on their own will keep augmenting. PDL can serve as single point of centralisation for all the personal data of the common user, whilst MMF is the PDL-compatible framework that can be exploited to discern insights in personal habits, patterns, and motives in the course of the user's daily activities and across various contexts including: work, health, finance, leisure, emotions, and so forth. The main enabler for such services is "querying". The presented MMF components require constructing SQL-like queries, mostly based on a priori knowledge of the concepts and properties associated with the metamodel which is used to manage the personal data and over which such queries are executed. Although from a technical perspective MMF components have been shown to hide considerable data management complexities, they still require a higher level of query formulation abstraction for allowing the common user to run personal analytics workloads in PDL without the need to write verbose queries, which the user may not understand or does not have the experience to expand in order to conduct advanced workloads. We set two future directions to overcome this limitation: first, we plan to build an interactive graphical query designer tool as the frontend of the query interface currently available in the access layer. Such tool provides both a graphical query design GUI and a text-based query design GUI for creating queries to retrieve meta-modelled personal data from PDL. The first GUI would be used to interactively build an SQL-like query and view the results for different source types like PDL backend, the lineage database, SemLinker metadata repository, and SemCluster annotations repository. The second GUI would be used to specify multiple statements, complex query or command syntax, and expression-based queries similar to those shown in Chapter 6. The second direction is to exploit the gravity feature of PDL. Our aim here is to develop a multitude of general-purpose gravity-enabled services to cover analytics which any PDL user might need, and distribute them as compiled code which would be easily downloaded by the user and plugged in the access layer. In this context, a service would either tune itself with the MMF's metamodel in an adaptive way to formulate queries on the behalf of the user and based on their information needs, or it contains predefined query formulation patterns which the user would directly trigger without the need to be concerned with their technical details. By reviewing the digital society (i.e., blogs, forums, news), we prioritise building gravity-enabled services for manipulating integrated social media data coming from multiple social services, such as graphical search and analysis (sentiment analysis, infographs), services for analysing integrated health data coming from smartphones and watches, services for mining relationships between unstructured textual documents, and services for geolocation informatics.

7.2.2 Intrinsic tagging and plugging in SemLinker

To the best of our knowledge, SemLinker is the first domain-agnostic integration system that offers self-adapting capabilities to automatically integrate raw data with frequently evolving schemas based on solid theoretical foundations as explained in Chapter 4. However, though in many aspects SemLinker can be regarded as an automatic system, it still has two vital tasks that need to be dealt with manually: data source tagging and selecting a schema matcher plugin. Using machine learning based approaches to label data sources with ontological concepts automatically, and thus relieving users of the burden of manual data source tagging during the installation of ingestion agents, is one of our future research goals. As for schema matcher selection, though the performance of SemMatcher in the evaluation is promising, we intend to extend it by combining a number of other matching approaches, so that it offers good matching solution for schemas of various characteristics, reducing the need for users to resort to other schema matchers.

7.2.3 Similarity and WSD in SemCluster

Although SemCluster exhibits better performance than other approaches, there is still room for improvement. In an experiment on a collection of mixed documents from Inspec and DUC-2001, we replaced the WuPalmer measure with the Jiang-Conrath metric [284] and used Babelfy [285] for the WSD task. An improvement in F1-measure compared with that of SemCluster$_{DBP,BN}$ was observed, however, its computational efficiency significantly decreased because Babelfy is available only as an online service. This suggests a potential enhancement to SemCluster, particularly by improving its semantic similarity metric and WSD algorithm. We are also interested in extending WordNet with more personalised knowledge sources and study their impact on performance using personal documents with greater length and domain variance (e.g. emails, health records, microblogs, etc.) than the currently used datasets.

7.2.4 Extending multimedia support

At the end of this doctoral research, a pressing idea in MMF development was extending its metadata management capabilities to support unstructured multimedia data. At present, MMF provides limited support for these kinds of data in the form of lineage recording. However, we plan to develop a fourth integral component in MMF, called SemMedia, to handle the metadata processing of different image and video files. With lineage manager, SemLinker, SemCluster, and SemMedia in place, MMF would be able to handle a wide spectrum of personal data, hence greatly benefiting the PDL user by extending the support of storing and managing new personal data types, and offering multiple choices for backstory information support during personal informatics and analytics, compared to the current text-based support only.

References

[1] Jones W. Keeping found things found: The study and practice of personal information management. Morgan Kaufmann. 2010.

[2] Pirolli P. Information foraging theory: Adaptive interaction with information. Oxford University Press. 2007.

[3] Mai J. Looking for information: A survey of research on information seeking, needs, and behavior. Emerald Group Publishing. 2016.

[4] Whittaker S. Personal information management: from information consumption to curation. Annual review of information science and technology. 2011;45(1):1-62.

[5] Fisher D, Brush A, Gleave E, Smith M. Revisiting Whittaker & Sidner's "Email Overload": Ten years later. In: Proceedings of the 20th Anniversary ACM Conference on Computer Supported Cooperative Work; 2006. p.309–312.

[6] Aula A, Natalie J, Mika K. Information search and re-access strategies of experienced web users. In: Proceedings of the 14th international conference on World Wide Web. ACM;2005. p.583-592.

[7] Boardman R, Spence R, Sasse M. Too many hierarchies? The daily struggle for control of the workspace. In: Proceedings of HCI international, vol. 1; 2003. p.616-620.

[8] Klaus S, Marcus A, Oyola J, Hoffman W, Luzi M. Personal data: The emergence of a new asset class. In: An Initiative of the World Economic Forum; 2011.

[9] Lazer D, Pentland A, Adamic L, Aral S, Barabasi A, Brewer D, Christakis N. Life in the network: the coming age of computational social science. New York:Science; 2009:323(5915):721-723.

[10] Eagle N, Pentland A. Reality mining: sensing complex social systems. Personal and ubiquitous computing 10. 2006;4:255-268.

[11] Van Kleek M, OHara K. The future of social is personal: The potential of the personal data store. In: Social Collective Intelligence. Cham:Springer;2014. p.125-158.

[12] Narayanan A, Toubiana V, Barocas S, Nissenbaum H, Boneh D. A critical look at decentralized personal data architectures. arXiv preprint;2012. arXiv:1202.4503.

[13] Gurrin C, Smeaton A, Doherty A. Lifelogging: Personal big data. Foundations and Trends in Information Retrieval. 2014;8(1):1-125.

[14] Kaasinen E. User needs for location-aware mobile services. Personal and ubiquitous computing. 2003;7(1):70-79.

[15] Dey K. Understanding and using context. Personal and ubiquitous computing. 2001;5(1):4-7.

[16] Bettini C, Brdiczka O, Henricksen K, Indulska J, Nicklas D, Ranganathan A, Riboni D. A survey of context modelling and reasoning techniques. Pervasive and Mobile Computing. 2010;6(2):161-180.

[17] Ronda G, Guerras L. Dynamics of the evolution of the strategy concept 1962–2008: a co-word analysis. Strategic Management Journal. 2012;33(3):162-188.

[18] Schwartz P, Solove D. Reconciling Personal Information in the United States and European Union. California Law Review. 2014;102(1):877.

[19] Choe K, Lee N, Lee B, Pratt W, Kientz J. Understanding quantified-selfers' practices in collecting and exploring personal data. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems. New York:ACM;2014. p.1143-1152.

[20] Li I, Dey A, Forlizzi J. A stage-based model of personal informatics systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York:ACM;2010. p.557-566.

[21] De Mauro A, Greco M, Grimaldi M. "What is big data? A consensual definition and a review of key research topics. In: AIP conference proceedings. AIP 1644(1);2015. pp.97-104.

[22] Siddiqa A, Hashem I, Yaqoob I, Marjani M, Shamshirband S, Gani A, Fariza N. A survey of big data management: Taxonomy and state-of-the-art. Journal of Network and Computer Applications. 2016;71(C):151-166.

[23] Russom P. Big data analytics. TDWI best practices report, fourth quarter. 2011;19(4):1-34.

[24] Buckle C. Digital consumers own 3.64 connected devices, Global Web Index.https://blog.globalwebindex.net/chart-of-the-day/digital-consumers-own-3-64-connected-devices/. Accessed 4 Feb 2018.

[25] Thompson E. Spotlight on Consumer App Usage, Annie App. http://files.appannie.com.s3.amazonaws.com/reports/1705_Report_Consumer_App_Usage_EN.pdf. Accessed 4 Feb 2018.

[26] Bras L. Online Overload - It's Worse Than You Thought, Dashlane. https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/. Accessed 4 Feb 2018.

[27] Jason Mander, The social media trends shaping 2018, Global Web Index. https://www.globalwebindex.net/reports/social. Accessed 4 Feb 2018.

[28] NewVantage Partners. Big Data Executive Survey: Themes and Trends. http://newvantage.com/wp-content/uploads/2012/12/NVP-Big-Data-Survey-Themes-Trends.pdf. Accessed 4 Feb 2018.

[29] Chaudhry A, Crowcroft J, Howard H, Madhavapeddy A, Mortier R, Haddadi H, McAuley D. Personal data: thinking inside the box. In: Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives. Aarhus University Press;2015. p.29-32.

[30] Heath T, Bizer C. Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology. 2011;1(1):1–136.

[31] Karger D, Jones W. Data unification in personal information management. Communications of the ACM. 2006;49(1):77–82.

[32] Pautasso C, Zimmermann O, Leymann F. Restful Web Services vs. "Big" Web Services: Making The Right Architectural Decision. In: WWW;2008. p.805–814.

[33] Boyd D, Crawford K. Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon. Information, communication & society. 2012;15(5):662-679.

[34] Rogers R. Digital methods. MIT press. 2013.

[35] Vicente C, Assent I, Jensen C. Effective privacy-preserving online route planning. In: Mobile Data Management (MDM), 2011 12th IEEE International Conference on, IEEE;2011. p.119-128.

[36] Montoya D, Tanon T, Abiteboul S, Senellart P, Suchanek F. Thymeflow, An Open-Source Personal Knowledge Base System. Doctoral dissertation. 2016.

[37] Bannon L, Bodker S. Constructing common information spaces. In: Proceedings of the Fifth European Conference on Computer Supported Cooperative Work. Berlin:Springer;1997. p.81–96.

[38] Berners-Lee T, Hendler J, Lassila O. The semantic web. Scientific american. 2001:284(5):34-43.

[39] Lenzerini M. Data integration: A theoretical perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.New York: ACM;2002. p.233-246.

[40] Jarke M, Quix C. On Warehouses, Lakes, and Spaces: The Changing Role of Conceptual Modeling for Data Integration. In: Conceptual Modeling Perspectives. Cham:Springer;2017. pp.231-245.

[41] Sowmya Y, Nagaratna M. Parallelizing K-Anonymity Algorithm for Privacy Preserving Knowledge Discovery from Big Data. International Journal of Applied Engineering Research. 2016;11(2):1314-1321.

[42] Dixon J. Pentaho, hadoop, and data lakes, James Dixon Blog. https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/. Accessed 25 Feb 2018.

[43] Huang F. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In: Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference on, IEEE;2015 p.820-824.

[44] Walker C, Alrehamy H. Personal data lake with data gravity pull. In: 2015 IEEE fifth international conference on Big data and cloud computing (BDCloud);2015. p.160-167.

[45] Peffers K, Tuunanen T, Rothenberger M, Chatterjee S. A design science research methodology for information systems research. Journal of management information systems. 2007;24(3):45-77.

[46] Kimball R, Ross M. The data warehouse toolkit: the complete guide to dimensional modeling. Net York:Wiley. 2011.

[47] Hurwitz J, Nugent A, Halper F, Kaufman M. Big data for dummies. New York:Wiley. 2013.

[48] Ramanathan V, Brickley D, Macbeth S. Schema.org: evolution of structured data on the web. Commun ACM. 2016;59(16):44–51.

[49] Miller G. WordNet: a lexical database for English. Communications of the ACM. 1995:38(11):39-41.

[50] Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes P, Hellmann S. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web. 2015;6(2):167-195.

[51] Navigli R, Ponzetto S. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence. 2012;193:217-250.

[52] Nadal S, Romero O, Abelló A, Vassiliadis P, Vansummeren S. An integration-oriented ontology to govern evolution in big data ecosystems. EDBT/ICDT workshops; 2017.

[53] Mihalcea R, Tarau P. Textrank: Bringing order into text. In: Proceedings of the 2004 conference on empirical methods in natural language processing;2004.

[54] Wan X, Xiao J. Single Document Keyphrase Extraction Using Neighborhood Knowledge. In: AAAI, volume 8;2008. pp. 855-860.

[55] Liu Z, Li P, Zheng Y, Sun M. Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1, Association for Computational Linguistics;2009. P. 257-266.

[56] Turban E, Sharda R, Aronson J, King D. Business intelligence: A managerial approach. Upper Saddle River, NJ:Pearson Prentice Hall. 2008.

[57] Abai Z, Yahaya J, Deraman A. User requirement analysis in data warehouse design: a review. Procedia Technology. 2013;11:801-806.

[58] Rahm E, Do H. Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin. 2000;23(4):3-13.

[59] Calvanese D, De Giacomo, G, Lenzerini M, Nardi D., Rosati R. Data integration in data warehousing. International Journal of Cooperative Information Systems. 2001;10(3):237-271.

[60] Inmon W, O'Neil B, Fryman L. Business metadata: Capturing enterprise knowledge. Morgan Kaufmann. 2010.

[61] Vassiliadis P. A survey of Extract–transform–Load technology. International Journal of Data Warehousing and Mining (IJDWM). 2009;5(3):1-27.

[62] Rivera J, Meulen R. Gartner says beware of the data lake fallacy. Gartner Inc. http://www.gartner.com/newsroom/id/2809117. Accessed 4 Feb 2018.

[63] Gwen T. Alpha males and data disasters: The case for data governance. Brass Cannon Press. 2006.

[64] Keith G. Principles of Data Management: Facilitating Information Sharing Second Edition. BCS. 2013.

[65] Dekkers M, Loutas N, De Keyzer M, Goedertier S. Open data and metadata quality. 2013.

[66] Ilyas I, Chu X. Trends in cleaning relational data: Consistency and deduplication. Foundations and Trends in Databases. 2015;5(4):281-393.

[67] Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. ACM Sigmod record. 1997;26(1):65-74.

[68] Mina F, Roatis A, Ilyas I, Hoffmann H, Chu X. CLAMS: bringing quality to Data Lakes. In: Proceedings of the 2016 International Conference on Management of Data, New York:ACM;2016 p.2089-2092.

[69] Salem AB, Boufares F, Correia S. Semantic Recognition of a Data Structure in Big Data. Journal of Computer and Communications. 2014;2(09):93.

[70] Chu X, Ilyas I, Papotti P. Discovering denial constraints. Proceedings of the VLDB Endowment. 2013;6(13):1498-509.

[71] Terrizzano I, Schwarz P, Roth M, Colino J. Data Wrangling: The Challenging Journey from the Wild to the Lake. In: CIDR;2015.

[72] Khine P, Wang Z. Data lake: a new ideology in big data era. In: ITM Web of Conferences 2018 (Vol. 17), EDP Sciences;2018.

[73] Greenberg J. Big Metadata, Smart Metadata, and Metadata Capital: Toward Greater Synergy Between Data Science and Metadata. Journal of Data and Information Science. 2017;2(3):19-36.

[74] Greenberg J. Metadata and the world wide web. Encyclopedia of library and information science. 2003;3:1876-1888.

[75] UK Data Archive. Research data lifecycle. http://www.data-archive.ac.uk/create-manage/life-cycle. Accessed 4 Feb 2018.

[76] Méndez E, Hooland SV. Metadata typology and metadata uses. In: Handbook of Metadata, Semantics and Ontologies;2014. pp.9-39.

[77] Dong R, Su F, Yang S, Xu L, Cheng X, Chen W. Design and application on metadata management for information supply chain. In: the 16th International Symposium on Communications and Information Technologies (ISCIT), Washington, DC: IEEE Computer Society Press;2016. p.393–396.

[78] Shankaranarayanan G, & Even A. The metadata enigma. Communications of the ACM. 2006;49(2):88–94.

[79] Zeng M, Qin J. Metadata. New York: Neal-Schuman Publishers. 2016.

[80] Buneman S, Tan W. Why and Where: A Characterization of Data Provenance. In: ICDT;2001. p.316-330.

[81] Simmhan L, Plale B, Gannon D. A survey of data provenance in e-science. ACM Sigmod Record. 2005;34(3):31-36.

[82] Haase K. Context for semantic metadata. In: Proceedings of the 12th annual ACM international conference on Multimedia. New York:ACM;2004. p.204-211.

[83] Taheriyan M, Knoblock C, Szekely P, Ambite J. A scalable approach to learn semantic models of structured sources. In: semantic computing (ICSC), IEEE international conference on. IEEE;2014. p. 183-190.

[84] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium;2010. p.1-10.

[85] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the ACM. 2008;51(1):107-13.

[86] Alrehamy H, Walker C. SemLinker: automating big data integration for casual users. Journal of Big Data, Springer. 2018;5(1):14.

[87] Zhu WD, Alon T, Arkus G, Duran R, Haber M, Liebke R, Morreale Jr F, Roth I, Sumano A. Metadata Management with IBM InfoSphere Information Server. IBM Redbooks. 2011.

[88] Lanter P. Design Of A Lineage-Based MetaData Base For GIS. In: Cartography and Geographic Information Systems;1991. pp.255-261.

[89] Stöhr T, Müller R, Rahm E. An integrative and uniform model for metadata management in data warehousing environments. In: Proceedings of the International Workshop on Design and Management of Data Warehouses. Heidelberg:Germany;1999.

[90] Quix C, Hai R, Vatov I. Metadata extraction and management in data lakes With GEMMS. Complex Systems Informatics and Modeling Quarterly. 2016;9(16):67–83.

[91] Gruber T. A translation approach to portable ontology specifications. Knowledge acquisition. 1993;5(2):199-220.

[92] Jarke M, Jeusfeld M, Quix C. Data-centric intelligent information integration—from concepts to automation. Journal of Intelligent Information Systems. 2014;43(3):437-462.

[93] Ramnandan S, Mittal A, Knoblock C, Szekely P. Assigning semantic labels to data sources. In: European semantic web conference. Cham:Springer;2015.

[94] Noy N. Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record. 2004;33(4):65-70.

[95] Naumann F. Data profiling revisited. ACM SIGMOD Record. 2014;42(4):40-9.

[96] Böhm C, Naumann F, Abedjan Z, Fenz D, Grütze T, Hefenbrock D, Pohl M, Sonnabend D. Profiling linked open data with ProLOD. In: Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference, IEEE;2010. p.175-178.

[97] Bizer C, Heath T, Berners-Lee T. Linked data: the story so far. In: Semantic services, interoperability and web applications: Emerging concepts, IGI Global;2011. p.205-227.

[98] Klyne G, Carroll JJ. Resource description framework (RDF): Concepts and abstract syntax. 2006.

[99] Alserafi A, Abelló A, Romero O, Calders T. Towards information profiling: data lake content metadata management. In: Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on, IEEE;2016. p.178-185.

[100] Suchanek FM, Abiteboul S, Senellart P. Paris: Probabilistic alignment of relations, instances, and schema. Proceedings of the VLDB Endowment. 2011;5(3):157-68.

[101] Ansari JW, Karim N, Decker S, Cochez M, Beyan O. Extending Data Lake Metadata Management by Semantic Profiling. 2018.

[102] Huai Y, Chauhan A, Gates A, Hagleitner G, Hanson EN, O'Malley O, Pandey J, Yuan Y, Lee R, Zhang X. Major technical advancements in apache hive. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. New York:ACM;2014. pp. 1235-1246).

[103] Revelytix. Revelytix Loom and Hortonworks Data Platform. Revelytix Loom whitepapr. https://hortonworks.com/wp-content/uploads/2013/01/Revelytix-Hortonworks_Reference_Architecture-080820131.pdf. Accessed 4 Feb 2018.

[104] Apache. Atlas Technical User Guide. Apache Atlas white paper. https://atlas.apache.org/0.7.1-incubating/AtlasTechnicalUserGuide.pdf. Accessed 4 Feb 2018.

[105] Gormley C, Tong Z. Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine. O'Reilly. 2015.

[106] Hellerstein J, Sreekanti V, Gonzalez J, Dalton J, Dey A, Nag S, Ramachandran K. Ground: A Data Context Service. In: CIDR;2017.

[107] Gao Y, Huang S, Parameswaran A. Navigating the Data Lake with Datamaran: Automatically Extracting Structure from Log Datasets. In: Proceedings of the 2018 International Conference on Management of Data. New York:ACM;2018. p. 943-958.

[108] Hai R, Geisler S, Quix C. Constance: An intelligent data lake system. In: Proceedings of the 2016 International Conference on Management of Data. New York:ACM;2016. p.2097-2100.

[109] Maccioni A, Torlone R. Crossing the finish line faster when paddling the data lake with kayak. Proc VLDB Endowment. 2017;10(12):1853.

[110] Rahm E. The case for holistic data integration. In: East European conference on advances in databases and information systems. Berlin:Springer;2016.

[111] Jagadish HV, Gehrke J, Labrinidis A, Papakonstantinou Y, Patel J, Ramakrishnan R, Shahabi C. Big data and its technical challenges. Commun ACM. 2014;57(14):86–94.

[112] Abelló A. Big data design. In: Proceedings of the ACM eighteenth international workshop on data warehousing and OLAP. New York:ACM;2015.

[113] Cai Y, Dong X, Halevy A, Liu J, Madhavan J. Personal information management with SEMEX. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. New York:ACM;2005 p.921-923.

[114] Barreau D, Nardi BA. Finding and reminding: file organization from the desktop. ACM SigChi Bulletin. 1995;27(3):39-43.

[115] Whittaker S, Sidner C. Email overload: exploring personal information management of email. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground. Vancouver:ACM Press;1996. p.276–283.

[116] Sankaranarayanan H, Lalchandani J. Passenger reviews reference architecture using big data lakes. In: Cloud Computing, Data Science & Engineering-Confluence. IEEE;2017. p.204-209.

[117] Martin F. Patterns of enterprise application architecture. Addison-Wesley Longman Publishing. 2002.

[118] Richards M. Software architecture patterns. O'Reilly Media. 2015.

[119] Lomborg S, Bechmann A. Using APIs for data collection on social media. The Information Society. 2014;30(4):256-265.

[120] Choi M, Cho E, Park D, Moon C, Baik D. A database synchronization algorithm for mobile devices. IEEE Transactions on Consumer Electronics. 2010;56(2).

[121] Cui Y, Lai Z, Wang X, Dai N. QuickSync: Improving synchronization efficiency for mobile cloud storage services. IEEE Transactions on Mobile Computing 2017;16(2):3513-3526.

[122] Jones, M, Hardt D. The oauth 2.0 authorization framework: Bearer token usage. No. RFC 6750. 2012.

[123] Fielding R, Gettys J, Mogul J, Frystyk H, Masinter L, Leach P, Berners-Lee T. Hypertext transfer protocol HTTP/1.1. No. RFC 2616. 1999.

[124] Ikeda R, Widom J. Data lineage: A survey. Stanford InfoLab. 2009.

[125] Cui Y, Widom J, Wiener J. Tracing the lineage of view data in a warehousing environment. ACM Transactions on Database Systems. 2000;25(2):179-227.

[126] Gómez AP, Corcho O. Ontology languages for the semantic web. IEEE Intelligent systems. 2002;17(1): 54-60.

[127] Object Management Group: Common warehouse metamodel specification 1.1. http://www.omg.org/spec/CWM/1.1/PDF/. Accessed 24 Feb 2018.

[128] Varga J, Romero O, Pedersen T, Thomsen C. SM4AM: A semantic metamodel for analytical metadata. In: Proceedings of the 17th International Workshop on Data Warehousing and OLAP, DOLAP;2014. p.57-66.

[129] Mattmann C, Zitting J. Tika in action. Manning Publications. 2011.

[130] Mansuri I, Sarawagi S. Integrating unstructured data into relational databases. In: Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on, IEEE;2006. p.29-29.

[131] Bush V. As we may think. The atlantic monthly. 1945;176(1):101-108.

[132] Gray J, Reuter A. Transaction processing: concepts and techniques. Elsevier. 1992.

[133] Floratou A, Teletia N, DeWitt DJ, Patel JM, Zhang D. Can the elephants handle the nosql onslaught? Proc VLDB Endowment. 2012;5(12):1712–1723.

[134] Leavitt N. Will nosql databases live up to their promise?. Computer 2010;43(2):12–14.

[135] Schram A, Anderson KM. MySQL to NoSQL: data modeling challenges in supporting scalability. In: Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity. New York:ACM;2012 p.191-202.

[136] Hecht R, Jablonski S. Nosql evaluation. In: International Conference on Cloud and Service Computing. IEEE;2011 pp336–41.

[137] Kashyap S, Zamwar S, Bhavsar T, Singh S. Benchmarking and analysis of nosql technologies. Int J Emerg Technol Adv Eng. (2013);3:422–426

[138] Nelubin D, Engber B. Ultra-High Performance NoSQL Benchmarking: Analyzing Durability and Performance Tradeoffs. Thumbtack Technology. White Paper. 2013.

[139] Abramova V, Bernardino J Nosql databases: Mongodb vs cassandra. In: Proceedings of the International Conference on Computer Science and Software Engineering. New York:ACM:2013. pp.14–22.

[140] Abramova V, Bernardino J, Furtado P. Which nosql database? a performance overview. Open Journal of Databases (OJDB). 2014;1(2):17-24.

[141] Abramova V, Bernardino J, Furtado P. Experimental evaluation of NoSQL databases. International Journal of Database Management Systems. 2014;6(3):1.

[142] Abubakar Y, Adeyi TS, Auta IG. Performance evaluation of NoSQL systems using YCSB in a resource austere environment. Performance Evaluation. 2014;7(8):23-27.

[143] Cooper BF, Silberstein A, Tam E, Ramakrishnan R, Sears R. Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM symposium on Cloud computing. New York:ACM;2010 p.143-154.

[144] Carlson L. Redis in action. Manning Publications. 2013.

[145] Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. New York:ACM;2008. pp.1099-1110.

[146] Banker K. MongoDB in action. Manning Publications. 2011.

[147] N. Chohan, C. Bunch, C. Krintz, and Y. Nomura. Database-agnostic transaction support for

cloud infrastructures. In Cloud Computing (CLOUD), 2011 IEEE International Conference on, IEEE;2011. p.692–699.

[148] McCory D. Data Gravity – in the Clouds. https://blog.mccrory.me/2010/12/07/data-gravity-in-the-clouds/. Accessed 24 Feb 2018.

[149] Fritsch J. Functional Programming Languages in Computing Clouds. Doctoral dissertation, Cardiff University. 2016.

[150] Montjoye Y, Shmueli E, Wang S, Pentland A. openpds: Protecting the privacy of metadata through safeanswers. PloS one. 2014;9(7).

[151] Ng I. Engineering a Market for Personal Data: The Hub-of-all-Things (HAT), A Briefing Paper. WMG Service Systems Research Group Working Paper Series. 2014.

[152] Mortier R, Zhao J, Crowcroft J, Wang L, Li Q, Haddadi H, Amar Y, Crabtree A, Colley J, Lodge T, Brown T. Personal Data Management with the Databox: What's Inside the Box?. In: Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking. New York:ACM;2016. p.49-54.

[153] Halevy A, Rajaraman A, Ordille J. Data integration: the teenage years. In: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment;2006. p.9-16.

[154] Rahm E, Bernstein PA. A survey of approaches to automatic schema matching. the VLDB Journal. 2001;10(4):334-50.

[155] Dong XL, Srivastava D. Big data integration. In: 2013 IEEE 29th international conference on data engineering (ICDE); 2013.

[156] Teevan J, Alvarado C, Ackerman M, Karger D. The perfect search engine is not enough: a study of orienteering behavior in directed search. In: Proceedings of the SIGCHI conference on Human factors in computing systems. New York:ACM;2004. p.415–422.

[157] Vakkari P. Task-based information searching. Annual review of information science and technology, 2003;37(1):413–464.

[158] Brodie M. Data integration at scale: From relational data integration to information ecosystems. In: Proc. 24th IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA), IEEE;2010 pp.2–3.

[159] Haas L. Beauty and the beast: The theory and practice of information integration. In: T. Schwentick & D. Suciu (Eds.), ICDT, lecture notes in computer science. Barcelona:Springer;2007. pp.28–43.

[160] Shvaiko P, Euzenat J. Ontology matching: state of the art and future challenges. IEEE Trans Knowl Data Eng. 2013;25(1):158.

[161] Peukert E, Eberius J, Rahm E. A self-configuring schema matching system. In: 2012 IEEE 28th international conference on data engineering (ICDE); 2012.

[162] Curino C, Moon H, Deutsch A, Zaniolo C. Automating the database schema evolution process. VLDB J. 2013;22(13):73–98.

[163] Andany J, Léonard M, Palisser C. Management of schema evolution in databases. In: VLDB;1991. p.161–70.

[164] Ullman J. Information integration using logical views. In: F.N. Afrati and P. Kolaitis, editors, Proceedings of the 6th International Conference on Database Theory (ICDT'97), volume 1186 of Lecture Notes in Computer Science;1997. p.19–40.

[165] Ceri S, Pelagatti G. Distributed databases: principles and systems. McGraw-Hill.1984.

[166] Zhou G, Hull R, King R. Generating data integration mediators that use materialization. Journal of Intelligent Information Systems. 1996;6(2-3):199-221.

[167] Wiederhold G Intelligent integration of information. InIntelligent Integration of Information. Boston:Springer;1996. p.193-203

[168] Collet C, Huhns MN, Shen WM. Resource integration using a large knowledge base in Carnot. Computer. 1991;24(12):55-62.

[169] Singh MP, Cannata PE, Huhns MN, Jacobs N, Ksiezyk T, Ong K, Sheth AP, Tomlinson C, Woelk D. The Carnot heterogeneous database project: Implemented applications. Distributed and Parallel Databases. 1997;5(2):207-25.

[170] Nardi D, Brachman RJ. An introduction to description logics. Description logic handbook. 2003;1:40.

[171] Halevy AY, Ives ZG, Madhavan J, Mork P, Suciu D, Tatarinov I. The piazza peer data management system. IEEE Transactions on Knowledge and Data Engineering. 2004 Jul;16(7):787-98.

[172] Xu L, Embley D. Combining the best of global-as-view and local-as-view for data integration. ISTA. 2004;48:123–36.

[173] Giese M, Soylu A, Vega-Gorgojo G, Waaler A, Haase P, Jiménez-Ruiz E, Lanti D. Optique: zooming in on big data. Computer. 2015;48(15):60–7.

[174] Calvanese D, Cogrel B, Komla-Ebri B, Kontchakov R, Lanti D, Rezk M, Rodriguez-Muro M, Xiao G. Ontop: answering SPARQL queries over relational databases. Semantic Web. 2017;8(17):471–87.

[175] Marcos M, Maldonado J, Martínez-Salvador B, Boscá D, Robles M. Interoperability of clinical decision-support systems and electronic health records using archetypes: a case study in clinical trial eligibility. J Biomed Inform. 2013;46(4):676–89.

[176] Cate B, Dalmau V, Kolaitis P. Learning schema mappings. ACM Trans Database Syst (TODS). 2013;38(13):28.

[177] Varga J, Romero O, Pedersen T, Thomsen C. Towards next generation BI systems: the analytical metadata challenge. In: International conference on data warehousing and knowledge discovery, vol. 8646. Cham: Springer; 2014. p.89–101.

[178] Chilvers A. Managing long-term access to digital data objects: a metadata approach. Doctoral Thesis, Loughborough University. 2000.

[179] Störl U, Müller D, Klettke M, Scherzinger S. Enabling Efficient Agile Software Development of NoSQL-backed Applications. Datenbanksysteme für Business, Technologie und Web;2017.

[180] Apache Avro. https://avro.apache.org/. Accessed 25 Dec 2017.

[181] Reis D, Cesar J, Pruski C, Reynaud-Delaître C. State-of-the-art on mapping maintenance and challenges towards a fully automatic approach. Expert Syst Appl. 2015;42(15):1465–78.

[182] Scherzinger S, Cerqueus T, Cunha de Almeida E. Controvol: a framework for controlled schema evolution in nosql application development. In: 2015 IEEE 31st international conference on data engineering (ICDE);2015. p.1464–7.

[183] Golshan B, Halevy A, Mihaila G, Tan WC. Data integration: After the teenage years. InProceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, ACM;2017 p.101-106.

[184] McGuinness D, Van Harmelen F. OWL web ontology language overview. W3C Recommen. 2004;1010(4).

[185] XSD Vocabulary. https://www.w3.org/TR/xmlschema11-1/. Accessed 25 Dec 2017.

[186] SIOC Vocabulary. http://rdfs.org/sioc/spec/. Accessed 25 Dec 2017.

[187] DCMI Vocabulary. http://dublincore.org. Accessed 25 Dec 2017.

[188] WGS84 Vocabulary. https://www.w3.org/2003/01/geo/. Accessed 25 Dec 2017.

[189] Media types listing by the internet assigned numbers authority. https://www.iana.org/assignments/media-types/media-types.xhtml. Accessed 25 De 2017.

[190] Wang S, Keivanloo I, Zou Y. How do developers react to restful API evolution? In: International conference on service-oriented computing. Berlin: Springer; 2014. p. 245–59.

[191] Shen W, Wang J, Han J. Entity linking with a knowledge base: Issues, techniques, and solutions. IEEE Trans Knowl Data Eng. 2015;27(15):443–60.

[192] Cruz I, Antonelli F, Stroe C. AgreementMaker: efficient matching for large real-world schemas and ontologies. Proc VLDB Endowment. 2009;2(9):1586–9.

[193] Madhavan J, Bernstein P, Doan A, Halevy A. Corpus-based schema matching. In: Proceedings 21st international conference on ICDE 2005 data engineering; 2005. p. 57–68.

[194] Bernstein A, Madhavan J, Rahm E. Generic schema matching, ten years later. Proc VLDB Endowment. 2011;4(11):695–701.

[195] Fagin R, Kolaitis P, Popa L, Tan W. Schema mapping evolution through composition and inversion. In: Schema matching and mapping. Berlin: Springer; 2011. p. 191–222.

[196] Belhajjame K, Paton NW, Embury SM, Fernandes AA, Hedeler C. Incrementally improving dataspaces based on user feedback. Information Systems. 2013;38(5):656-87.

[197] Das Sarma A, Dong X, Halevy A. Bootstrapping pay-as-you-go data integration systems. InProceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM:2008. p.861-874.

[198] Gandomi A, Haider M. Beyond the hype: Big data concepts, methods, and analytics. International Journal of Information Management. 2015;35(2):137-144.

[199] Gantz J, Reinsel D. Extracting value from chaos. IDC iview. 2011;1142(2011):1-2.

[200] Dobre C, Xhafa F. Intelligent services for big data science. Future Generation Computer Systems. 2014;37:267-81.

[201] Chen H, Chiang RH, Storey VC. Business intelligence and analytics: from big data to big impact. MIS quarterly. 2012:1165-88.

[202] Kulkarni S, Singh A, Ramakrishnan G, Chakrabarti S. Collective annotation of Wikipedia entities in web text. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining;2009 p.457-466.

[203] Buneman P, Davidson SB, Suciu D. Programming constructs for unstructured data. IRCS Technical Reports Series.1995;1:121.

[204] Adrian B, Hees J, Elst L, Dengel A. iDocument: using ontologies for extracting and annotating information from unstructured text. In: Annual Conference on Artificial Intelligence, Heidelberg:Springer;2009. p.249-256.

[205] Cunningham H, Tablan V, Roberts I, Greenwood M, Aswani N. Information extraction and semantic annotation for multi-paradigm information management. In: Current Challenges in Patent Information Retrieval, Heidelberg:Springer;2011. p.307-327

[206] Cunningham H, Maynard D, Bontcheva K, Tablan V. A framework and graphical development environment for robust NLP tools and applications. In: ACL;2002 p. 168-175.

[207] Silber H, McCoy K. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. Computational Linguistics. 2002;28(4):487-496.

[208] Mihalcea R, Csomai A. Wikify!: linking documents to encyclopedic knowledge. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, New York:ACM;2007 p. 233-242.

[209] Sahlgren M. The distributional hypothesis, Ital. J. Linguist. 2008;20(1):33–54.

[210] Blei D, Ng A, Jordan M. Latent Dirichlet allocation, J. Mach. Learn. Res. 2003;(3):993–1022.

[211] Gabrilovich E, Markovitch S. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: IJCAI;2007. p.1606–1611.

[212] Ritter A, Clark S, Etzioni O. Named entity recognition in tweets: an experimental study. In: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics;2011. p.1524-1534.

[213] Leaman R, Lu Z. TaggerOne: joint named entity recognition and normalization with semi-Markov Models. Bioinformatics. 2016;32(8):2839-2846.

[214] Ren X, El-Kishky A, Wang C, Tao F, Voss C, Han J. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York;2015. p.995-1004.

[215] Shi C, Li Y, Zhang J, Sun Y, Philip S. A survey of heterogeneous information network analysis. IEEE Transactions on Knowledge and Data Engineering. 2017;29(1):17-37.

[216] Meng F, Morioka c. Automating the generation of lexical patterns for processing free text in clinical documents. Journal of the American Medical Informatics Association.2015;22(5):980-986.

[217] Jivani AG. A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl. 2011;2(6):1930-8.

[218] Turney P. Learning algorithms for keyphrase extraction. Information Retrieval. 2000;2(4):303-336

[219] Witten I, Paynter G, Frank E, Gutwin C, Nevill-Manning C. KEA: Practical automatic keyphrase extraction." In: Proceedings of the fourth ACM conference on Digital libraries, ACM;1999. p.254-255.

[220] Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 conference on Empirical methods in natural language processing, ACL;2003. p.216-223.

[221] Hasan K, Ng V. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, ACL;2010. p.365-373.

[222] Hasan K, Ng N. Automatic Keyphrase Extraction: A Survey of the State of the Art. In: ACL (1);2014. p.1262-1273.

[223] Siddiqi S, Sharan A. Keyword and keyphrase extraction techniques: a literature review. International Journal of Computer Applications. 2015;109(2).

[224] Sterckx L, Demeester T, Develder C, Caragea C. Supervised keyphrase extraction as positive unlabeled learning. In: EMNLP2016, the Conference on Empirical Methods in Natural Language Processing;2016. p.1-6.

[225] Zhang Q, Wang Y, Gong Y, Huang X. Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter. In: EMNLP;2016. p.836-845.

[226] Kim S, Medelyan O, Kan M, Baldwin T. Automatic keyphrase extraction from scientific articles. Language Resources and Evaluation, Springer. 2013:47(3):723–742.

[227] Page, L, Brin S, Motwani R, Winograd T. The PageRank citation ranking: Bringing order to the web. Stanford InfoLab. 1999.

[228] Wang J, Liu J, Wang C. Keyword extraction based on pagerank. In: Advances in Knowledge Discovery and Data Mining, Springer;2007. p.857-864.

[229] Martinez-Romo J, Araujo L, Fernandez A. SemGraph: Extracting keyphrases following a novel semantic graph-based approach. Journal of the Association for Information Science and Technology. 2016:67(1):71-82.

[230] Cilibrasi R, Vitanyi P. The google similarity distance. IEEE Transactions on knowledge and data engineering. 2007:19(3):370–383.

[231] Pasquier C. Task 5: Single document keyphrase extraction using sentence clustering and Latent Dirichlet Allocation. In: Proceedings of the 5th international workshop on semantic evaluation, ACL;2010. p.154-157.

[232] Danesh S, Sumner T, Martin J. SGRank: Combining Statistical and Graphical Methods to Improve the State of the Art in Unsupervised Keyphrase Extraction." In: Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics;2015. p.117-126

[233] Danilevsky M, Wang C, Desai N, Ren X, Guo J, Han J. Automatic construction and ranking of topical keyphrases on collections of short documents. In: Proceedings of the 2014 SIAM International Conference on Data Mining, SIAM;2014. p.398–406.

[234] Liu Z, Huang W, Zheng Y, and Sun M. Automatic keyphrase extraction via topic decomposition. In: Proceedings of the 2010 conference on empirical methods in natural language processing, ACL;2010 p.366-376.

[235] Sparck K. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation. 1972:28(1):11-21.

[236] El-Beltagy S, Rafea A. KP-Miner: A keyphrase extraction system for English and Arabic documents. Information Systems. 2009;34(1):132-144.

[237] Bracewell D, Ren F, Kuroiwa S. Multilingual single document keyword extraction for information retrieval. In: Proceedings of the 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, Wuhan;2005. p.517-522.

[238] Barker K, Cornacchia N. Using noun phrase heads to extract document keyphrases. InConference of the Canadian Society for Computational Studies of Intelligence. Heidelberg:Springer;2000. p.40-52.

[239] Munoz A. Compound key word generation from document databases using a hierarchical clustering ART model. Intelligent Data Analysis. 1997;1(4):25-48

[240] Clark A, Fox C, Lappin S. The handbook of computational linguistics and natural language processing. John Wiley & Sons Inc. 2013.

[241] Hoffart J, Suchanek F, Berberich K, Weikum G. YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence. 2013;194:28–61.

[242] Navigli R. Word Sense Disambiguation: A Survey. ACM Computing Surveys. 2009;41(2):1-69.

[243] Patwardhan S, Banerjee S, Pedersen T. SenseRelate::TargetWord: a generalized framework for word sense disambiguation. In: Proceedings of the ACL 2005 on Interactive Poster and Demonstration Sessions;2005. p.73-76.

[244] Meng L, Huang R, Gu J. A review of semantic similarity measures in WordNet. International Journal of Hybrid Information Technology. 2013;6(1):1-12.

[245] Wu Z, Palmer M. Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, ACL;1994. pp 133-138.

[246] Guan R, Shi X, Marchese M, Yang C, Liang Y. Text clustering with seeds affinity propagation. IEEE Transactions on Knowledge and Data Engineering. 2011;23(4):627-637.

[247] Frey B, Dueck D. Clustering by passing messages between data points. Science. 2007;315(5814):972-976.

[248] Le T, Le N, Shimazu A. Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases. In: Australasian Joint Conference on Artificial Intelligence, Springer International Publishing;2016. p.665-671.

[249] Witten I, Milne D. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI press;2008. p.25-30.

[250] Androutsopoulos I, Koutsias J, Chandrinos K, Spyropoulos C. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval;2000. p.160-167.

[251] Hammouda K, Matute D, and Kamel M. Corephrase: Keyphrase extraction for document clustering. In: MLDM;2005. p.265–274.

[252] Qiu M, Li, Y, Jiang J. Query-oriented keyphrase extraction. In: Information Retrieval Technology;2012. p.64-75.

[253] Tomokiyo T, Hurst, M. A language model approach to keyphrase extraction. In: Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment, Volume 18, ACL;2003. p.33-40.

[254] Xu Z, Wei X, Luo X, Liu Y, Mei L, Hu C, Chen L. Knowle: a semantic link network based system for organizing large scale online news events. Future Generation Computer Systems. 2015;43:40-50.

[255] Yang S, Lu W, Yang D, Li X, Wu C, Wei B. KeyphraseDS: Automatic generation of survey by exploiting keyphrase information. Neurocomputing. 2017;224(2017):58-70

[256] Nguyen Q, Nguyen T, Cao T. Semantic-based recommendation method for sport news aggregation system. In: International Conference on Research and Practical Issues of Enterprise Information Systems, Springer;2016. p.32-47.

[257] Erdmann M, Maedche A, Schnurr H, Staab S. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. In: Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content, ACL;2000. p.79-85.

[258] Giannopoulos G, Bikakis N, Dalamagas T, Sellis T. GoNTogle: a tool for semantic annotation and search. In: Extended Semantic Web Conference, Springer;2010. p.376-380.

[259] Tablan V, Bontcheva K, Roberts I, Cunningham H. Mímir: An open-source semantic search framework for interactive information seeking and discovery. Web Semantics: Science, Services and Agents on the World Wide Web. 2015;30:52-68.

[260] Alrehamy H, Walker C. Exploiting extensible background knowledge for clustering-based automatic keyphrase extraction. Soft Computing, Springer. 2018; (accepted).

[261] Alrehamy H, Walker C. SemCluster: Unsupervised Automatic Keyphrase Extraction Using Affinity Propagation. In: Advances in Computational Intelligence Systems. UKCI 2017, Springer;2017. p.222-235.

[262] Jivani AG. A comparative study of stemming algorithms. Int. J. Comp. Tech. Appl. 2011;2(6):1930-8.

[263] Human Activity Recognition Dataset (HAR 1,2). https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition. Accessed 25 Dec 2017.

[264] Human Activity Recognition Dataset (HAR 3). https://hal.archives-ouvertes.fr/hal-01586802. Accessed 25 Dec 2017.

[265] Facebook Open Graph API. https://graph.facebook.com. Accessed 25 Dec 2017.

[266] Twitter Data Streaming API. https://api.twitter.com. Accessed 25 Dec 2017.

[267] Foursquare API. https://api.foursquare.com/v2/. Accessed 25 Dec 2017.

[268] Flickr API. https://api.flickr.com/services/rest/. Accessed 25 Dec 2017.

[269] London Restaurants Reviews Dataset. https://www.kaggle.com/PromptCloudHQ/londonbased-restaurants-reviews-on-tripadvisor. Accessed 25 Dec 2017.

[270] Tourpedia API. http://tour-pedia.org/api/. Accessed 25 Dec 2017.

[271] United Kingdom Government open datasets, the food standards agency, food safety and food hygiene ratings dataset. http://ratings.food.gov.uk/open-data/. Accessed 25 Dec 2017.

[272] United Kingdom Postal Codes Dataset. https://www.getthedata.com/open-postcode-geo. Accessed 25 Dec 2017.

[273] Videla A, Williams J. RabbitMQ in action: distributed messaging for everyone. Manning Publications. 2012.

[274] SemLinker Experimental Evaluation Setup. https://github.com/alrehamy/SemLinker_Evaluation. Accessed 25 Dec 2017.

[275] Tversky A. Features of similarity. Psychological review. 1977;84(4):327-352.

[276] Bergstra J, Bengio, Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research. 2012;13(2):281-305.

[277] Allan J. Topic detection and tracking: event-based information organization. Vol. 12. Springer Science & Business Media. 2012.

[278] Tate Gallery Artists Public Dataset. https://www.kaggle.com/rtatman/the-tate-collection/data. Accessed 25 Dec 2017.

[279] Tate Gallery Artworks Public Dataset. https://github.com/tategallery/collection. Accessed 25 Dec 2017.

[280] Tate Artwork description dataset. https://github.com/alrehamy/tate-experiment. Accessed 25 Dec 2017.

[281] A. Novotny and S. Spiekermann. Personal information markets and privacy: a new model to solve the controversy, pages 102–120. IOS Press, 2013.

[282] Klettke M, Awolin H, Störl U, Müller D, Scherzinger S. Uncovering the evolution history of data lakes. In: Big Data (Big Data), 2017 IEEE International Conference on, IEEE;2017. p.2462-2471.

[283] Gartner’s 2016 Hype Cycle for Emerging Technologies Identifies Three Key Trends That Organizations Must Track to Gain Competitive Advantage. http://www.gartner.com/newsroom/id/3412017. 4 Feb 2018.

[284] Jiang J, Conrath D. Semantic similarity based on corpus statistics and lexical taxonomy. arXiv: preprint cmp-lg/9709008. 1997.

[285] Moro, A, Cecconi F, Navigli R. Multilingual word sense disambiguation and entity linking for everybody. In: Proceedings of the 2014 International Conference on Posters & Demonstrations Track-Volume 1272;2014. p 25-28.