DEVELOPING A SEMANTIC FRAMEWORK FOR HEALTHCARE INFORMATION INTEROPERABILITY

A dissertation submitted to
Kent State University in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Mehmet Aydar

December 2015

Dissertation written by

Mehmet Aydar

B.E., Bahcesehir University, 2005

MTEC, Kent State University, 2008

Ph.D., Kent State University, 2015

Approved by

________________________________, Chair, Doctoral Dissertation Committee
Austin Melton

________________________________, Members, Doctoral Dissertation Committee
Angela Guercio

____
Ye Zhao

____
Alan Brandyberry

____
Helen Piontkivska

Accepted by

________________________________, Chair, Department of Computer Science
Javed I. Khan

________________________________, Dean, College of Arts and Sciences
James L. Blank

TABLE OF CONTENTS

LIST OF FIGURES vii

LIST OF TABLES viii

Acknowledgements ix

Dedication x

1 Introduction 2

1.1 Interoperability in Healthcare 2

1.1.1 Standards vs. Translations 4

1.1.2 RDF as a Common Healthcare Information Representation 4

1.1.3 Summary Graph 5

1.1.4 Graph Node Similarity Metric 6

1.1.5 Instance Matching 7

1.2 Dissertation Overview and Contributions 8

2 Background and Related Work 11

2.1 Data, Information and Knowledge 11

2.2 Semantic Web 12

2.2.1 Resource Description Framework (RDF) 13

Definition (RDF Graph Data) 15

2.2.2 RDF Schema 15

2.2.3 Ontology 15

Web Ontology Language (OWL) 16

2.2.4 Linked Data 16

2.3 Data Mapping 18

2.4 Data Translation 23

2.5 Entity Similarity 24

2.5.1 Jaccard Similarity Measure 24

2.5.2 RoleSim Similarity Measure 25

2.6 Graph Summarization 28

2.6.1 Definition (Summary Graph) 28

2.6.2 Summary Graph Generation Methods 28

2.7 Healthcare Data Interoperability Initiatives 30

3 Translation of Instance Data using RDF and Structured Mapping Definitions . 32

3.1 Translation of Instance Data 34

3.2 Metadata Repository 34

3.3 Mapping Schema 36

3.4 Mapping Interface 39

3.5 Translation Engine 40

3.6 Conclusion 41

4 Pairwise Nodes Similarity in RDF Graphs 44

4.1 Contributions and Outline 45

4.2 Input Graph Data 46

4.3 Methods 46

4.4 Computation of Entity Similarity 47

4.4.1 IRI Nodes Similarity 47

4.4.2 Literal Node Similarity 51

4.4.3 Descriptor Importance and Automatic Detection of Noise Labels . 52

Detection of Noise Descriptors 54

4.5 The Algorithm 54

4.6 Related Work 57

4.7 Conclusion 60

5 Building Summary Graphs of RDF Data in the Semantic Web 61

5.1 Contributions and Outline 62

5.2 Methods 63

5.3 The Algorithm 64

5.4 Generation of the Classes and Properties 64

5.5 Class Relation Stability Metric 65

5.6 Data Dictionary Extraction 67

5.7 Evaluation 68

5.8 Related Work 75

5.9 Conclusion 76

6 RinsMatch: a suggestion-based instance matching system in RDF Graphs . . . 77

6.1 Contribution and Outline 78

6.2 Instance Matching in Healthcare Data 79

6.3 RDF Entity Similarity for Instance Matching 80

6.4 User Interaction 80

6.5 Evaluation 83

6.6 Related Work 83

6.7 Conclusion 86

7 Conclusion 88

BIBLIOGRAPHY 92

LIST OF FIGURES

1 An example of RDF triples illustrating resources from diverse domains [87] 14
2 Example Ontologies and their Mappings [37] 22
3 RoleSim(u,v) based on similarity of their neighbors [54] 27
4 Translation of instance data 35
5 An example of data dictionary elements stored in the metadata server . . 36
6 A mapping example showing how selected data dictionary elements are mapped 37
7 A figure showing the RDF representation of mapping between sample data dictionary elements 38
8 A screen shot of the mapping interface. 40
9 Example Graph for Graph Equivalence matching 49
10 Sample graph demonstrating two nodes 58
11 A figure consisting of different types of entities and elements belonging to the class types. 74
12 An excerpt from the generated summary graph. 75
13 Instance matching process 82

LIST OF TABLES

1 A Sample of RDF Triples from Each Dataset 70

2 An Excerpt from Dynamically Assigned Weights of Descriptors 71

3 Evaluation Results 73

Acknowledgements

I wish to thank my committee members who were more than generous with their expertise and precious time. A special thanks to Dr. Austin Melton, my advisor for his countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process. Thank you Dr. Ye Zhao, Dr. Angela Guercio, Dr. Alan Brandyberry, and Dr. Dr. Helen Piontkivska for agreeing to serve on my committee.

I would like to acknowledge and thank the Kent State University Semantic Web Research Group (SWRG) members, the Cleveland Clinic Cardiology Application Group members, and the Semantic Web Health Care and Life Sciences (HCLS) Interest group members for their helpful feedback.

Finally I would like to thank Dr. Ruoming Jin for his help and guidance during the study and Dr. Victor E. Lee for sharing RoleSim similarity measure.

I dedicate my dissertation work to the hero of my life, my dear and loving father, Abdullah Aydar for his encouragement, inspiration and endless support. I will always appreciate all he has done for me in my whole life.

1

CHAPTER 1

Introduction

It is estimated that up to 7000 unique dialects are spoken around the world [14]. Despite the fact that individuals speak distinctive dialects, they discover routes to communicate by either agreeing on the same dialect, or by utilizing a translator. This aligns with the definition of "interoperability." Broadly speaking, interoperability is a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal [46]. The term was initially defined for information technology or systems engineering services to allow for information exchange [2]. The concept of interoperability plays a pivotal role in our daily lives. Individuals, organizations and governments have a tendency to agree on standards to make products more understandable and usable by diverse communities. For instance, the World Wide Web(www) [101] is a large interoperable network of documents whose standards are defined by the World Wide Web Consortium(W3C) [88].

1.1 Interoperability in Healthcare

Interoperability in healthcare is stated as the ability of health information systems to work together within and across organizational boundaries in order to advance the effective delivery of healthcare for individuals and communities [47]. In 2009 The Health Information Technology for Economic and Clinical Health Act(HITECH) authorized incentive payments to clinicians and hospitals when they implement electronic health record

(EHR) [45] systems to achieve Meaningful Use of healthcare data. [19]. Meaningful Use is specified by the following 3 components [41]: (1) use of certified EHR, (2) use of certified EHR technology for electronic exchange of health information and (3) use of certified EHR technology to submit clinical quality measures. The components 2 and 3 require interoperability between EHRs and that is where the Meaningful Use is delayed because different EHRs do not talk to each other. According to a review [64] published in 2014, only 10% of ambulatory practices and 30% of hospitals are participating in operational health information exchange efforts. The lack of interoperability has a significant economic impact on the healthcare industry. The West Health Institute(WHI) testified to U.S. Congress, and released an estimate that system and device interoperability could save over $30 billion a year in the U.S. healthcare system alone [44].

Achieving interoperability in healthcare is a significantly challenging process. The current healthcare information technology (IT) environment breeds incredibly complex data ecosystems. Clinical data exist in multiple layers and forms ranging from heterogeneous structured, semi-structured, and unstructured data captured in enterprise-wide electronic medical record and billing systems, through a wide variety of departmental, study, and lab-based registries and databases, to the massive quantities of device-specific data generated by an ever increasing number of instruments used to evaluate, monitor, and treat patients. In many cases pertinent patient records are collected in multiple systems, often supplied by competing manufacturers with diverse data formats. This causes inefficiencies in data interoperability, as data retrieval from disparate data sources is time consuming and costly. Also different formats of data create barriers in exchanging health information, because physicians are not able to interpret the exchanged data if they are not well acquainted with the source data. The same holds true for the computer software systems which may focus on different algorithmic purposes including but not limited to mapping, translation, validation and search.

1.1.1 Standards vs. Translations

Interoperability can be accomplished by adopting standards and/or translations between different standards. There exist many different standards in healthcare, each is developed to fulfill different purposes. In [22], the authors express the barriers in implementing universal standards: complexity of the standards, diverse use cases and evolution of the standards. These hurdles cause the standardization efforts to take significant amounts of time, and the systems relying on the standards also need to be updated along with the standardization. Consequently a proficient route for translation between different data models is more practical for information interoperability. Given the fact that it is unrealistic to have one universal standard that fits all the use cases, is it conceivable to implement standards in information translation? We think the answer is "yes." Our work is based on the technical side of healthcare information translation. We propose translation oriented methods, metrics, algorithms and frameworks that assist in healthcare information interoperability.

1.1.2 RDF as a Common Healthcare Information Representation

Is it possible to depict the healthcare data entities having diverse formats in a single flexible form? In this sense, RDF (Resource Description Framework) [57] gained notoriety in recent years, because it is schemaless and is a commonly acknowledged framework by the Semantic Web [17] community. It is also proposed as a universal healthcare information representation in [23]. RDF makes statements about resources in the form of (subject, predicate, object) expressions, which are known as triples. In this work we utilize RDF representation of healthcare data. Both instance data and data models which are used to organize the instance data are represented in RDF. Healthcare data in different systems are linked through the relations in RDF triples, ending up with an extensive connected graph.

1.1.3 Summary Graph

An intermediate data structure, that describes the metadata or the summary of the dataset, is often needed for faster computational processing of the graph representation of healthcare data. The summary data structure comprises of the fundamental type classes and their associations, with each type class representing the type of a collection of data entities. Such a data structure is also represented in RDF that forms a summary graph. The flexibility of RDF allows the metadata of the related data entities to be linked. In this case, the summary graph is linked to the RDF representation of the dataset and utilized for data interoperability. For instance, if a physician wants to retrieve the lab records of a cohort [94], the entities that link to the lab type class are filtered, making use of the type class associations. A summary graph also reduces the size of the input dataset for various computer algorithms. For instance, an instance matching algorithm utilizes the higher level type classes, and compares the instances only belonging to the same type class. Also the associations of the type classes are leveraged for intelligent exploration of graph representation of the healthcare dataset, such that a filtering algorithm retrieves the data elements having a specific relation to a specific type class. In this work, we present an effective algorithm for generating a summary graph from an RDF dataset and we test the algorithm on a healthcare dataset.

The relations between different type classes are represented as predicates in the summary graph. However since in this work, the summary graph is auto-generated, it is error prone. The degree of confidence of type class relations in the summary graph needs to be known by the physician or by the interpreter algorithm for more precise results. For instance, in a cardiac surgery related data set, if an “Event” type class has relations to “Vascular Procedure” and “Cardiac Valve Replacement” type classes, it makes more sense for the interpreter to know how likely an instance in the “Event” class has a relation to the “Vascular Procedure” or to the “Cardiac Valve Replacement” class. Therefore in this work, the relations between the type classes are generated along with a degree of confidence, which we call the stability measure. The stability measure is computed based on the actual number of the data entities in the relations between the type classes in the dataset.

1.1.4 Graph Node Similarity Metric

Summarizing RDF data requires grouping similar entities in the same type class. But how should we characterize the similarity of healthcare data elements? When healthcare data elements are represented in RDF, the similarity can be examined as graph node similarity. Therefore, the challenge is to define the factors that affect the similarity of a pair of RDF nodes. There have been numerous studies investigating the similarity measures in various fields. Some studies concentrated on graph neighborhood-based similarity measures as in [53, 59], while others focused on comparing instances based on properties and roles [49, 90]. An RDF entity is described through its relations to other entities and the collection of literals associated with it in the lexical form. The descriptors are represented as predicates and literal nodes in RDF. In this work we introduce a graph node pairs similarity metric, that utilizes the graph nodes' descriptors, namely, common predicates of the RDF nodes and common string literal words referenced by the nodes, augmented with the node neighborhood similarity.

Each descriptor of a data entity may have a different impact in similarity calculation. For instance in a dataset related to heart surgery, the word "cardiac" is less important than the word "valve," since "valve" describes more about the nature of the surgery while "cardiac" is a more general term in the dataset. On the other hand the importance of "valve" is not the same for all the entities, such that it is more important for a surgical procedure related data entity while it is less important in a cardiac valve anatomy pathology related data entity. As a consequence; each descriptor has a different importance weight, and the weight differs per each data entity. So how should we identify appropriate metrics for generating the descriptor weights in the RDF representation of healthcare data entities? In this work we propose an importance weighting metric which has a similar notion to the term frequency-inverse document frequency(tf-idf) [61, 80], based on the frequency of the descriptor in the referenced data entity and in the corpus.

1.1.5 Instance Matching

Healthcare data exist in different systems with diverse data models. RDF representation of healthcare data contains both the instance data and the concepts belonging to the data models of the instance data. Since different systems may have substantial amounts of redundancy, the RDF graph may have duplicate nodes both in data instances level and data model concepts level. Therefore, an efficient instance matching technique utilized for RDF graphs covers both de-duplication and data concepts matching. Once detected, the redundant nodes can be merged, reducing the size of the input graph. Also matching the data concepts belonging to separate data models assists in the information translation process, because translation requires data models level mapping. The instance matching technique needs to be automated due to the size of the healthcare data. Nevertheless automated instance matching techniques are error prone. Consequently, a technique that permits user interactions yields more precise results. In this work we present a semi-automatic instance matching technique which is based on a graph node similarity measure with the inputs from the users and utilized for the RDF representation of healthcare data.

1.2 Dissertation Overview and Contributions

This dissertation makes the following contributions. First, we propose a system for translation of healthcare instance data, based on structured mapping definitions and using RDF as a common information representation to achieve semantic interoperability between different data models. The main components of the framework are a metadata repository that stores the data dictionary of the data sources, a user-friendly interface to view and manage the metadata and the mappings between different data elements, and a translation engine that translates the source RDF data with regards to the target data model. The framework allows the domain experts to define mappings between different data elements and utilizes the mappings to translate the data models from one format to another.

Second, we introduce a pairwise graph nodes similarity metric that utilizes the Jaccard index with the common relations of the healthcare data entities and common string literal words referenced by the healthcare data entities and augmented with data entity neighbors similarity. The precision of the similarity metric is enhanced by incorporating the auto-generated importance weights of the healthcare data entity relations and the string literals referenced by the healthcare data entities in the RDF representation of the dataset.

Third, we provide a novel machine learning algorithm that automatically generates a summary graph structure from the RDF representation of healthcare dataset, and we propose that the summary graph can further be utilized for interoperability purposes. In the summary graph, similar entities are classified as the same type class, and the relations between the type classes are created with regards to their members' relations. Furthermore, a degree of confidence measure is added for the relations between the type classes in the summary graph.

Fourth, we present a suggestion based semi-automatic instance matching system and we test it on the RDF representation of a healthcare dataset. The system utilizes the graph node similarity metric introduced in our second contribution, and it presents similar node pairs to the user for possible instance matching. Based on the user feedback, it merges the matched nodes and suggests more matching pairs depending on the common relations and neighbors of the already matched nodes. The process runs in iterations until the similarity algorithm infers no new matching candidate pairs and there is no more feedback from the user. We propose that the instance matching technique could be leveraged for mapping between separate healthcare data models, as the data model concepts are also incorporated in RDF. Mappings contribute in the information translation process, because translation requires data models level mapping.

The dissertation is organized as follows. Chapter 2 gives background information on Semantic Web and existing work. Chapter 3 introduces a framework for translation of healthcare instance data, based on structural mapping definitions and RDF representation of healthcare data. Chapter 4 introduces a pairwise graph node similarity metric and its algorithm. Chapter 5 introduces an automatic summary graph structure generation approach utilizing the similarity metric introduced in chapter 4. Chapter 6 introduces RinsMatch, a suggestion-based instance matching system in RDF Graphs. We offer concluding remarks in Chapter 7.

CHAPTER 2

Background and Related Work

This chapter provides background information regarding Semantic Web technologies and prior work relevant to our work in healthcare data interoperability.

2.1 Data, Information and Knowledge

In [48] the author described data, information and knowledge as follows: data is the facts and a description of the World, information is captured data and knowledge, and knowledge is a personal map/model of the World. In this sense, when the facts are captured in structured format they become information. The interconnection between the data and information elements constitutes the knowledge.

In a clinical domain, data are the facts about the patients, diseases, results of received care, etc. For instance, patient demographic information such as name, race and date of birth is data. They are true whether this is recorded somewhere or not. Information is the way that the data is captured in some format in order that people or machines can understand and access it at different times. Therefore any sort of database is information as long as it can be accessible and understood. For instance, when a nurse captures the demographics data of a patient on a hospital visit they become information as they are accessible by other people or machines. In this case the birthday record of a patient is information but the actual birth date of the patient is data. The patient record (information) could be transferred to other data sources or it could be modified. But modifying the patient information does not change the data (actual birth day of the patient).

The knowledge is based on the data, information and other knowledge. A processing engine bases its decisions from the knowledge that it has. The human processing engine, the brain, stores the knowledge inter-connectedly and makes decisions based on its own knowledge.

2.2 Semantic Web

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [17]"

– Tim Berners-Lee

Following the definition proposed by Tim Berners-Lee, Semantic Web [99] is not a new Web but an extension of the current Web. The fundamental thought is to describe the data on the Web with machine interpretable metadata in order that machines can process the information available on the Web. Semantic Web technologies are heavily influenced by Artificial Intelligence, by languages that developed from efforts to represent documents in a universal way, and by the network protocols that the World Wide Web is built upon.

Each object on Semantic Web is identified with an identifier called URI (Uniform Resource Identifier) [16]. A uniform resource locator (URL) [89] is a subclass of URI that identifies a Web page. URI and URL support only ASCII encoding. On the other hand, IRI (International Resource Identifier) [31] is a generalization of the URI that fully supports international characters. In this regard, every absolute URI and URL is also an IRI, however not every IRI is a URI.

Semantic Web does not only contain Web pages; in fact, the objects on Semantic Web can be any type of resource such as Web pages, images, cities, people, events, etc. The combination of identifiers that can refer to resources or concepts on distributed machines, a universal way to retrieve data over a network, and the unambiguous interpretation of content are the primary reasons why Semantic Web technologies address a variety of problems associated with traditional databases: integration, translation, querying, and interpretation.

2.2.1 Resource Description Framework (RDF)

Central to Semantic Web technologies is a general purpose language, the Resource Description Framework (RDF) [57], for representing information in Semantic Web in a way that the meaning (or semantics) is unambiguous to a machine or software process. RDF describes resources through statements in the form of (subject, predicate, object) expressions which are known as triples. Every triple describes an entity or a relationship between two entities [31].

The subject in an RDF triple is either an Internationalized Resource Identifier(IRI) or a blank node, the predicate is an IRI, and the object is either an IRI, a literal or a blank node. The subjects and objects of triples in the RDF graph form RDF nodes. Each RDF node that corresponds to a unique RDF entity is represented with a unique IRI, and the values such as strings, numbers and dates are represented by literal nodes. A literal node can consist of two or three elements: a lexical form, a datatype IRI and a

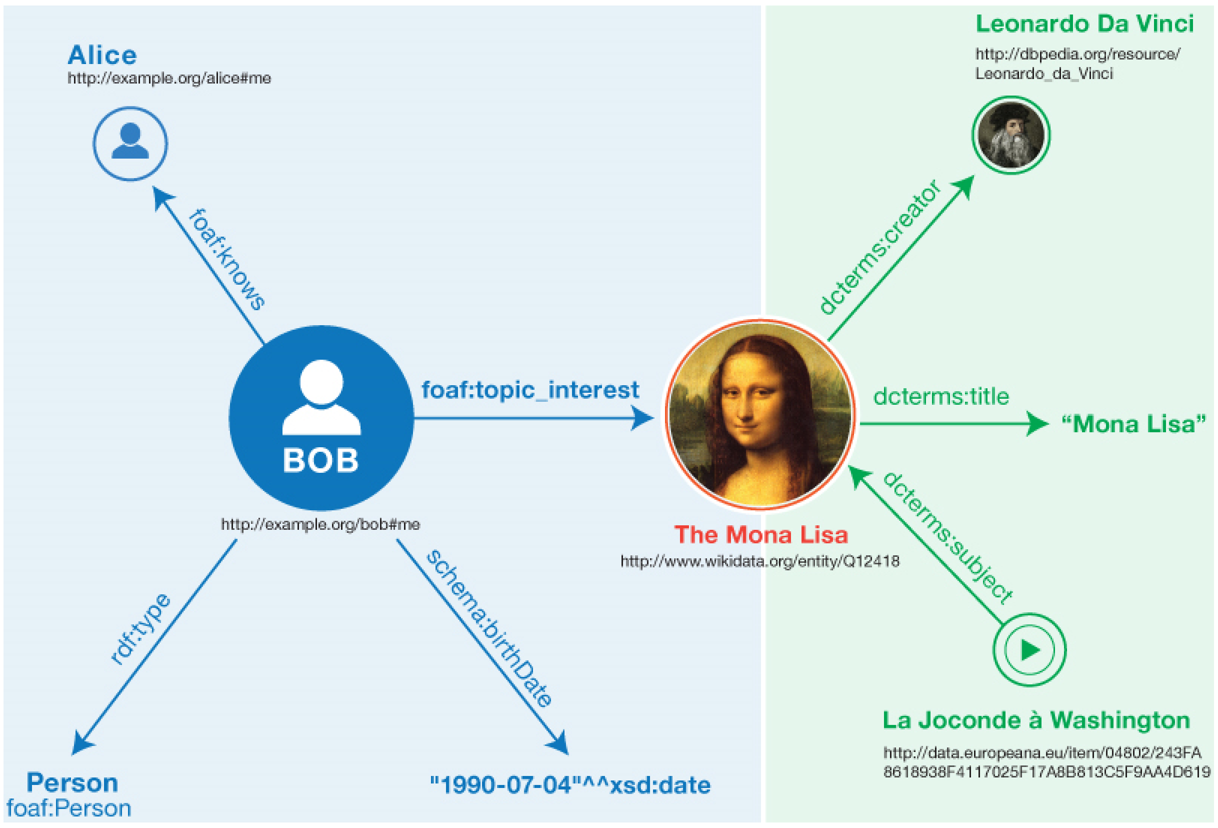

Alice
http://example.org/alice#me
foaf:knows
BOB
http://example.org/bob#me
foaf:topic_interest
rdf:type
schema:birthDate
Person
foaf:Person
"1990-07-04"^^xsd:date
Leonardo Da Vinci
http://dbpedia.org/resource/Leonardo_da_Vinci
dcterms:creator
dcterms:title
"Mona Lisa"
The Mona Lisa
http://www.wikidata.org/entity/Q12418
dcterms:subject
La Joconde à Washington
http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D619

Figure 1: An example of RDF triples illustrating resources from diverse domains [87]

language tag. The language tag in a literal node is included if and only if the datatype IRI of the literal node corresponds to rdf:langString [24]. A predicate in an RDF triple is also called a property of the RDF subject node. A predicate can be one of two types: a DatatypeProperty where the subject of the triple is an IRI and the object of the triple is a literal, or an ObjectProperty where both the subject and object of the triple are IRIs. Each object of a subject node is called a neighbor of that subject node.

Figure1 (copied from [87]) shows an example of RDF triples, illustrating the resources from diverse domains and the way they are linked through named relations.

The flexibility of RDF data model enables data interchange, merging datasets with different data models and evolution of the data models over time. The linked structure of RDF triples allows resources belonging to different datasets to be linked with a named relation. The collection of triples with the linking structures constitutes a directed and labeled graph, where the edges represent the named relations that have semantically unambiguous meanings that can be utilized by the software algorithms for intelligent exploration of the graph-structured data.

Definition (RDF Graph Data)

Let a directed labeled graph $G(V, L, E)$ be the RDF data such that V is the finite set of vertices; L denotes the finite set of edge labels; and E is the set of edges of the form $e = l(u, v)$, with u, $v \in$ V, $l \in L$ and $e \in E$. Note that an edge $l(u, v)$ represents the RDF triple (u, l, v).

2.2.2 RDF Schema

RDF is schemaless which means that the RDF data is not tied to a particular schema. However, schemas for RDF can exist and the RDF resources represented in RDF can have a structure given by the schema. As a matter of fact, the W3C specified RDF Schema (RDFS) [24] standard to provide a data-modelling vocabulary for RDF resources is an extension of the basic RDF vocabulary. RDF schema provides features to describe RDF classes and properties; define hierarchies; constraint the values for the domain and range of properties, etc.

2.2.3 Ontology

The meaning of ontology slightly differs in different contexts. In the context of Computer Science, a frequently cited explanation of an ontology is given by Gruber [42]: "an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members)." RDF Schema(RDFS) is a formal language to define an ontology to be used for a specific domain.

A formal definition of ontology is given in [55] as a pair $O = (S, A)$, where S is the ontology signature which describes the vocabulary (the named terms) of an ontology, and A is a set of ontological axioms (statements that say what is true in the domain) specifying the interpretation of the vocabulary in some domain of discourse.

Web Ontology Language (OWL)

The Web Ontology Language (OWL) is another formal language for authoring ontologies. It extends RDFS with more complex features, enabling applications to process the content of the information and facilitating greater data interoperability. OWL is compatible with RDFS. In addition, it provides additional features like property characteristics, property restrictions, ontology mapping(such as equivalence between classes, properties and individuals) and complex class definitions [27].

2.2.4 Linked Data

As the Web has grown exponentially, it has not only become the universal platform of interconnected documents but also the linked information space for data. Data published in the Web have been traditionally in the forms of HTML tables, CSV or XML. Due to the lack of expressivity of entity relations in HTML documents and the lack of explicit semantics and relational structures in the published data, the actual semantics of data in the Web is currently being underutilized for cognitive computation purposes.

With continuing development of Semantic Web technologies, there has in recent years been significant progress including explicit semantics with data in the Web as an ever-growing number of organizations adopt Semantic Web technologies. Publishing data on the Web in a standard model by using a standard methodology and interlinking the data available on the Web using Semantic Web technologies provides a Web of data that applications can access to and utilize through semantic queries.

The collection of inter-linked datasets on the Web is also referred to as Linked Data. In [18] the authors defined Linked Data as the: "data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets." Historically, the term "Linked Data" was used in 2006 by Tim Berners-Lee, the inventor of the World Wide Web, who outlined the rules for publishing data on the Web to make the Web of data a reality, as follows [15]:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards.
- Include links to other URIs, so that they can discover more things.

Consequently, in short amount of time many organizations started to publish open data using the data standards of the World Wide Web Consortium. As of 2015, the structured data available in the Semantic Web have been rapidly increasing with the contribution of Linked Open Data along with several other Semantic Web projects which incorporate many datasets associated with different domains such as publications, life sciences, media, social Web, geography, etc. For instance, DBpedia [9] and FreeBase [21] made available the content of Wikipedia in RDF and incorporated links to other open datasets, e.g., to GeoNames [3], a geographical database which covers over eight million place names in RDF model. Concurrently, the number of Web pages published in RDFa (Resource Description Framework in Attributes) [5], a W3C recommendation that adds semantics to web page content by embedding a set of a attribute-level extensions, and microformats [56], conventions for embedding semantic markup to identify specific resources in web pages, has increased. The applications that consumes a dataset on the Linked Open Data can exploit the extra knowledge available through links to the other datasets. While publishing open data initiatives have made available thousands of general purpose datasets and many domain-specific data sources in the Resource Description Framework (RDF) data model, there is still a long way to go as Linked Open Data is still a small portion of the information available on the Web.

2.3 Data Mapping

Data mapping [96] is the process of creating the linkages and relations between data elements of distinct data models. Data mapping creates connections between different data elements. The connectivity of the data elements increases data interoperability and data reusability while reducing the redundancy. Also it is a key step for data integration tasks such as data transformation, data lineage analysis, discovery of new data details within connected data sources, consolidation of multiple data sources into a single data source, etc. Furthermore data mapping is needed for standardization of the data. For instance, healthcare institutions often need to map their local data to an accepted medical standard such as ICD-9 [40] or SNOMED CT [71] to be able to share their data with other medical facilities. There are different techniques for data mapping.

In manual data mapping the mapping between different data elements is being done manually by the users. The users can document the mapping in a variety of ways using hard programming codes, XSLT [28] transforms or by using graphical mapping tools that automatically generate data transformation programs. There is a variety of graphical tools for data mappings. Some tools work by drawing a line between different data elements while others use some kind of string similarity algorithms to automatically connect the source and destination. These kinds of graphical tools are common for ETL (Extract-Transform-Load) [1] based applications. They create automatic data transformation programs in different languages such as SQL, XSLT, Java or C++ depending on the application environment to support transforming data from one form to another.

A data-driven mapping technique uses heuristics and statistics to automatically discover mappings and transformations between different data sets by evaluating the actual data values in the data sources [96]. The discovered transformation logic could be arithmetic, concatenations, case statements, etc. The main advantage of a data-driven technique over the manual data mapping is that it reduces the human labor. But on the other hand the results may not be as accurate as manual data mapping.

Synonyms-driven mapping relies on a metadata registry to auto map the data elements based on data element names and their synonyms. The metadata registry contains the synonyms of the data element names. For example auto connecting a data element named “gender” to another data element named “sex” is possible if both these data element names are listed as synonyms in the metadata registry. A synonyms-driven mapping can generate matches between data schema elements but will not auto generate the data transformation logic.

The Semantic Web technology is not in its mature phase yet. However its ambitious vision aims to take the current state of the data mapping techniques to a more automatic process based on Semantic Web languages such as Resource Description Framework (RDF), the Web Ontology Language (OWL) [8], etc. The Semantic Web technology is based on connected data elements. Ontology which is a part of the Semantic Web technology consists of mapped data concepts which are mapped to actual data elements in RDF and give meaning to the data that can be leveraged for the mapping process. The idea of “Linked Data” of Semantic Web builds on a standardized metadata publishing mechanism which will accelerate the development of more automatic data mapping tools.

In essence, the task of linking is connecting semantically related concepts from multiple data sources. Over the years, this problem has been studied by the research community as the task of ontology matching, ontology alignment or more commonly ontology mapping. Although these concepts are frequently used interchangeably, they have slight differences. Mapping takes place after matching the related concepts first, and the alignment process refers to identifying semantically equivalent classes from the data. For our purposes, we will be using the term ontology mapping as finding the matching concepts and aligning them with other semantically relevant concepts and classes of concepts. The matching concepts do not necessarily have to be identical or equivalent. They might also be hierarchically related with subset or superset relations.

Ontology mapping is an essential task in achieving semantic interoperability on the Semantic Web. As the amount of publicly available heterogenic data on the Semantic Web grows continually, the applications require using more and more ontologies every day. Considering the scale and distribution of the Semantic Web resources, it is no longer possible to maintain one ontology that covers the whole spectrum. Hence, many ontologies are being created and used by many different applications.

In this context, ontology mapping can also be thought of as a semantic interoperability layer which allows applications to query information across multiple resources. Over the years, the ontology mapping problem has been researched by the many diverse communities for various reasons including data translation, ontology merging, data integration, query answering, etc.

In [83], the problem of ontology mapping is described as: "given two ontologies O_1 and O_2, mapping one ontology with another means that for each concept (node) in ontology O_1, try to find a corresponding concept (node), which has same or similar semantics, in ontology O_2 and vice versa." In other words, the process of ontology mapping is identifying the semantic relationships between two related concepts belonging to different ontologies.

A mathematical definition of ontology mapping is given in [55] as morphism (a structure-preserving map from one mathematical structure to another [97]) of ontological signatures. Given two ontologies $O_1 = (S_1, A_1)$ and $O_2 = (S_2, A_2)$, a total ontology mapping from O_1 to O_2 is a morphism function $f : S_1 \to S_2$ of ontological signatures, such that, $A_2 \models f(A_1)$, i.e., all interpretations that satisfy O_2 axioms also satisfy O_1's translated axioms. In reality, a total mapping between two ontologies may not always

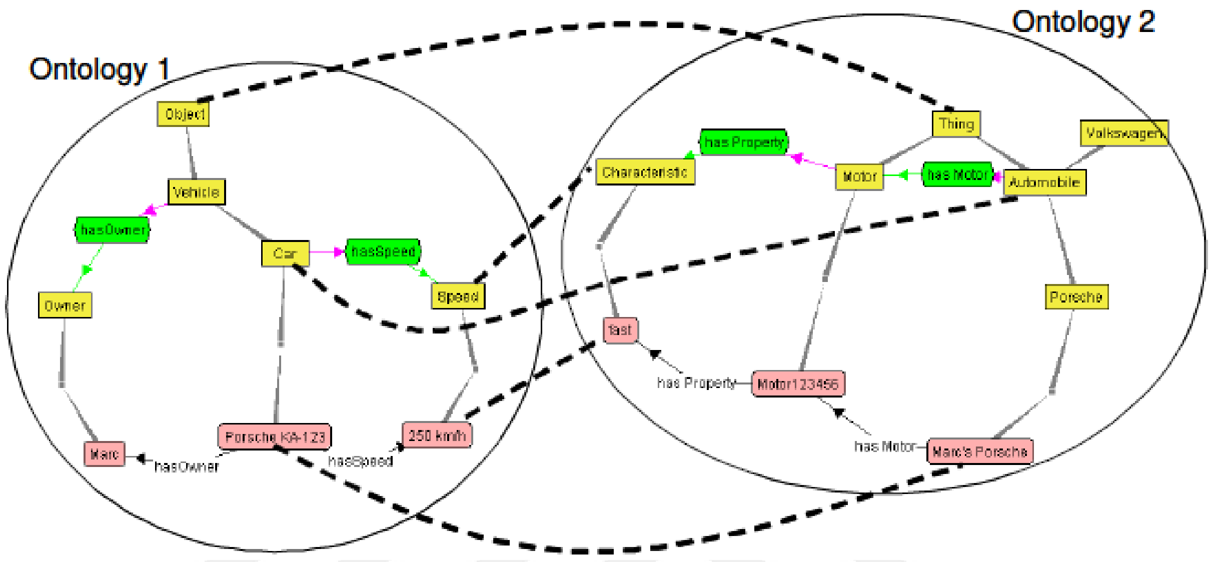


Figure 2: Example Ontologies and their Mappings [37]

exists. Therefore, a weaker notion of ontology mapping is introduced as a partial ontology mapping from O_1 to O_2 if there exists a sub-ontology $O'_1 = (S'_1, A'_1)$, i.e., $S'_1 \in S_1$ and $A'_1 \in A_1$, such that there is a total mapping from O'_1 to O_2.

Figure 2 (taken from [37]) illustrates a mapping between two given ontologies describing the domain of car retailing. A reasonable mapping between the two ontologies is given by the dashed lines in the figure.

In the literature, there has been a lot of research done in the subject of ontology mapping in diverse communities from different perspectives. As a result, the terminology used for defining the problem varied such as matching, alignment, merging, articulation, fusion, integration, morphism, etc. Consequently, many tools and methods have developed in the field. Some surveys on the current state-of-the-art have tried to categorize the approaches and describe the terminologies [39] [55].

In some other studies, the authors have addressed the problem in relation to ontology design and integration i.e., [66, 69, 70]. However, they focus on heuristics to match ontology elements and do not utilize the information in the data instance level. These approaches do not use machine learning. Still others advocated implementing machine learning techniques [20, 91, 102] in their frameworks for learning to combine matching measures and selection strategies.

Moving forward a few years, we see a large number of related works under the topic of Ontology Alignment [33, 35, 51, 52, 74, 75]. Some of them are focusing on mapping using an upper ontology [33, 51, 52]. An upper ontology is a top-level ontology [100] that describes very general concepts that are the same across all knowledge domains. Other related works are concentrating on aligning ontologies based on set theory approaches that utilize existing links on instances [74, 75].

Parundekar, et al. [74, 75] proposed an approach for aligning ontologies in Linked Open Data by discovering concept coverings and linkage using set theory ideas. They consider the union of disjoint concepts for alignment process. Although our approach has some commonalities, we don't rely on existence of owl:sameAs links between instances as it is the case in their approach. Another difference is that we use machine learning algorithms for finding concepts and their hierarchy.

2.4 Data Translation

Data translation [93] is converting a set of data values from the data format of a source data system into the data format of a destination data system. Data mapping from the source data system to the destination data system is a key step for data translation. The programming code of the actual data translation program is based on the data mapping between the source and destination data systems.

2.5 Entity Similarity

2.5.1 Jaccard Similarity Measure

The Jaccard similarity coefficient also known as the Jaccard index is a statistical measure that is used for comparing similarity and diversity of sample sets, and it is defined as the size of the intersection divided by the size of the union of the sample sets [50]. According to the Jaccard index if A and B are subject nodes in a dataset, then their similarity, $J(A, B)$, is formulated as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

For instance, the Jaccard similarity measure of two sets can be illustrated in a simple example as follows. Given three sets A, B, C such that:

$A = \{apple, banana, orange\}$,

$B = \{banana, orange, peach\}$,

$C = \{peach, fig, cherry\}$

The Jaccard index $J(set1, set2)$ between two sets, $set1$ and $set2$, is calculated as follows:

$J(A, B) = 2/4 = 0.5$,

$J(A, C) = 0/6 = 0$,

$J(B, C) = 1/5 = 0.2$

Therefore, A and B are the most similar sets while A and C are the least similar sets.

2.5.2 RoleSim Similarity Measure

RoleSim is a similarity metric for graph nodes based on the maximal matching of neighborhood pairs and a simple iterative computational method.

Given a graph $G = (V, E)$, RoleSim measures the similarity of each node pair in V based on their neighborhood similarities with the intuition that "two nodes are similar if they relate to similar objects" [54] :

$$RoleSim(u,v) = (1-\beta) \times max_{M \in Mm(u,v)} \frac{\sum\limits_{(x,y)\in M} RoleSim(x,y)}{N_u + N_v - |M|} + \beta \quad (2)$$

$RoleSim(u,v)$ denotes the similarity of the nodes $u, v \in V$. The definition of RoleSim is recursive; i.e., $RoleSim(x,y)$ is calculated the same way as $RoleSim(u,v)$. $N(u)$ and $N(v)$ denote their respective sets of neighborhoods and N_u and N_v denote their respective degrees, i.e., $N_u = |N(u)|$ and $N_v = |N(v)|$.

We define M to be a set of ordered pairs (x,y) where $x \in N(u)$ and $y \in N(v)$ such that there does not exist $(x',y') \in M$, s.t. $x = x'$ or $y = y'$, and furthermore, M is maximal in that no more ordered pairs may be added to M and keep the constraint above. $Mm(u,v)$ is the set of all such M's. $Mm(u,v)$ is a set of sets.

As shown in the formula, the $RoleSim(u,v)$ depends on their neighborhood similarities. Instead of using the mean of neighborhood similarities, the maximal matching is used. Figure 3 (taken from [54]) shows how the maximal matching works to calculate the similarity measure between the nodes u and v. The nodes $(x1, x2, x3)$ are the neighbors of u and the nodes $(y1, y2, y3, y4)$ are the neighbors of v. The cell values are the similarities of each (x,y) pair. M is a maximal subset of $N(u) \times N(v)$ such that no element of $N(u)$ appears more than once as a first coordinate and no element of $N(v)$ appears more than once as a second coordinate of an ordered pair in M. Thus, $|M| = min(N_u, N_v)$. The maximal matching ensures that the total value of selected cells has the maximum

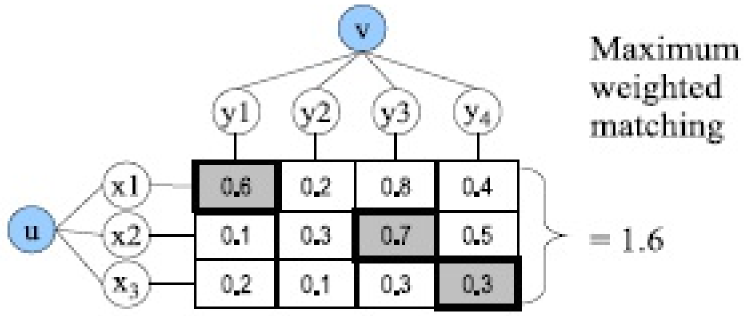


Figure 3: RoleSim(u,v) based on similarity of their neighbors [54]

possible value. The maximal matching value, $M(u, v)$, is calculated as

$$M(u, v) = max_{M \in Mm(u,v)} \frac{\sum\limits_{(x,y)\in M} RoleSim(x, y)}{Max(N_u, N_v)} \tag{3}$$

The parameter β is a decay (damping) factor [98], $0 < \beta < 1$. It is similar to the one used in [72], so that the influence of neighbors decreases with distance which dampens the recursive effect. In [72], the value of β was originally set to 0.15. We provide more details about how the value of β is chosen in section 5.7.

The input graph in RoleSim is not a labeled graph. This makes it hard to directly apply the RoleSim measure in an RDF data set since the RDF triples contain labeled edges. In our work we utilize the RoleSim measure in conjunction with the Jaccard index to calculate the similarity of two graph nodes. The $RoleSim(u, v)$ measure is used when there may be multiple neighbors for a common property reached from the node pairs u and v, where u and v are the nodes in the input RDF graph.

2.6 Graph Summarization

A summary graph of an RDF data graph is the RDF graph such that each node in the summary graph is a subset of the original graph nodes of the same type and each property in the summary graph is a subset of the original graph nodes properties.

2.6.1 Definition (Summary Graph)

Let $G = (V, L, E)$ be an RDF graph as defined in section 2.2.1. We define a summary graph as $G' = (V', L', E')$, such that V' contains equivalence classes of V. E' and L' are, respectively, the sets of edges and labels in the graph G'. As we will see, $L' \subset L$, and the elements of E' are defined by the elements in the equivalence classes in V' and the edges in E.

2.6.2 Summary Graph Generation Methods

There exist several methods to obtain a summary graph: (1) A summary graph can be obtained from the dataset ontology, if the dataset is already tied to an ontology. (2) Another way to obtain the summary graph is to locate the type triples in the dataset and to organize the type classes and relations accordingly, if the data set is published using a standard vocabulary [36]. (3) Or the summary graph can be built automatically by inferring the class types based on the similarity of the RDF nodes. Our graph summarization approach is based on method 3 as explained in details in chapter 5.

The summary graph data structure has been used for various purposes in different studies. For instance in [86] the authors utilized the summary graph for keyword search on RDF graphs. However they assumed that there is already a summary graph data structure in place, which may not always be correct in real world datasets.

In [36] the authors used an index structure similar to a summary graph (called type system in their study) to generate benchmark data from a real world dataset with complete control over both the structuredness and the size of the generated data. In their study, the level of structuredness of a dataset D with respect to a type T in the summary graph is determined by how well the instance data in D conform to type T. In the study, instead of assuming that there is already a summary graph in place they propose an algorithm to auto generate the summary graph with the assumption that a standardized and known vocabulary are used in the input RDF data set. Their summary graph generation algorithm uses the following steps:

- Scan the input RDF dataset and look for the triples whose property is rdf:type [24], extract t as the object of the triple.
- Then extract the subjects that has a type T.
- $P(T)$ = union of all the properties of the subjects above, where the type T refers to a class in the summary graph and $P(T)$ is the union of all the properties used in T.

In this sense, their summary graph generation algorithm aligns with the method 2, as opposed to our work in which we infer the summary graph classes based on the similarity of the RDF nodes.

2.7 Healthcare Data Interoperability Initiatives

Healthcare information interoperability requires the necessary tools, an efficient roadmap and a strong commitment from relevant communities. Many studies have been suggested for this purpose. In this section we review the selected studies regarding healthcare data interoperability.

The Yosemite Project [23] suggests an ambitious roadmap for healthcare information interoperability on a global scale, by using RDF as a universal information representation and creating a hub for crowd-sourcing translation rules. They provide general translation rules, metadata including source and target data models, translation rules language (such as SPARQL/SPIN N3, Java, Python, etc.), dependencies, test data with validation, license, maintainer and usage metrics. They envision a translation hub that allows downloading the translation rules as needed. Comparing to the Yosemite Project, our work is more specifically focused on leveraging entity similarities for auto-classification, generating mapping between entities and exploiting the structured mapping definitions in healthcare information translation.

3M Health Information Systems [4] created a public use version of the 3M Healthcare Data Dictionary (HDD), a medical terminology server that has been in operational use at the US Department of Defense (DoD) and other health care organizations. The project is named as HDD Access [58]. It is an application that contains a controlled medical vocabulary stored in a relational database with application programming interface (API) runtime services that other applications can call. In this sense their work is a standardization effort, as opposed to our work in which we concentrate on translation.

In [65], the authors presented FURTHeR, a semantic framework intended to run federated queries across disparate sources. The infrastructure incorporates a terminology server, a metadata repository and translation services for translating the semantic queries to run on disparate data sources. The infrastructure enables semantic interoperability via semantic search. Our work does not utilize a standardized terminology server. In our work, the information interoperability is achieved by translation of instance data with structured mapping definitions.

SemanticDB is a semi-structured patient data repository system developed within the Cleveland Clinic that leverages several advances in XML processing and Semantic Web technology standards for the purpose of outcomes research [32]. The SemanticDB patient data registry stores the current and historical data for a patient population which is specifically selected for clinical research in cardiology domain. SemanticDB contains a data production pipeline that translates the pertinent patient data to generate customized views of the repository content for statistical analysis and reporting. The translation in SemanticDB relies on the hard coded mapping definitions implemented in the transformation code, and in contrast to our work their mappings are not stored in structured format. However, in our work we utilize the SemanticDB patient data registry as a test dataset for testing various algorithms presented in this work.

CHAPTER 3

Translation of Instance Data using RDF and Structured Mapping Definitions

In this chapter, we present a translation oriented framework for healthcare information interoperability. The main ideas and sections explained in this chapter are part of a published research study [12]. The idea behind the system is achieving semantic interoperability between diverse healthcare data models, by empowering translation of instance data based on structured mapping definitions, and using RDF as a common information representation. We have developed a framework that allows the domain experts to define linkages between different data elements and utilizes the mappings to translate the data models from one format to another.

The clinical data in a healthcare IT environment can be incredibly complex. The data is generated in different systems through diverse work flows. Therefore, often the related patient data is stored in multiple systems, supplied by different manufacturers with diverse data formats. This not only causes inefficiencies for the physicians as data retrieval from disparate data sources is time consuming, wasteful and costly, but also creates barriers in exchanging health information. Users may not be able to understand the exchanged data if they are not competent with the data format. The same holds true for the machines to process the data for various algorithmic purposes including but not limited to mapping, translation, validation, search, etc.

Interoperability can be achieved by adopting standards and translations between different standards. There exist many different standards in healthcare, each having its own uses and advantages. It is unrealistic to have one universal standard that fits all the use cases since there are significant barriers in implementing universal standards, as explained in [22]: complexity of the standards, diverse use cases and evolution of the standards cause the standardization efforts to take significant amounts of time, and the systems relying on the standards also need to be updated along with the standardization. Therefore an efficient way of translation between different data models is needed. This work focuses on the technical side of healthcare information translation.

Since each data model can be in a different format, adopting a common information representation model is unavoidable for the translation. RDF (Resource Description Framework) [57] gained popularity in recent years, since it is schemaless and is a commonly accepted information representation model by the Semantic Web community. RDF is also proposed as a universal healthcare information representation in [23]. In our work, we represent healthcare data in RDF model. Instance data along with its data model is represented in RDF. And different datasets are linked through the predicates in RDF triples which form an extensive and connected graph.

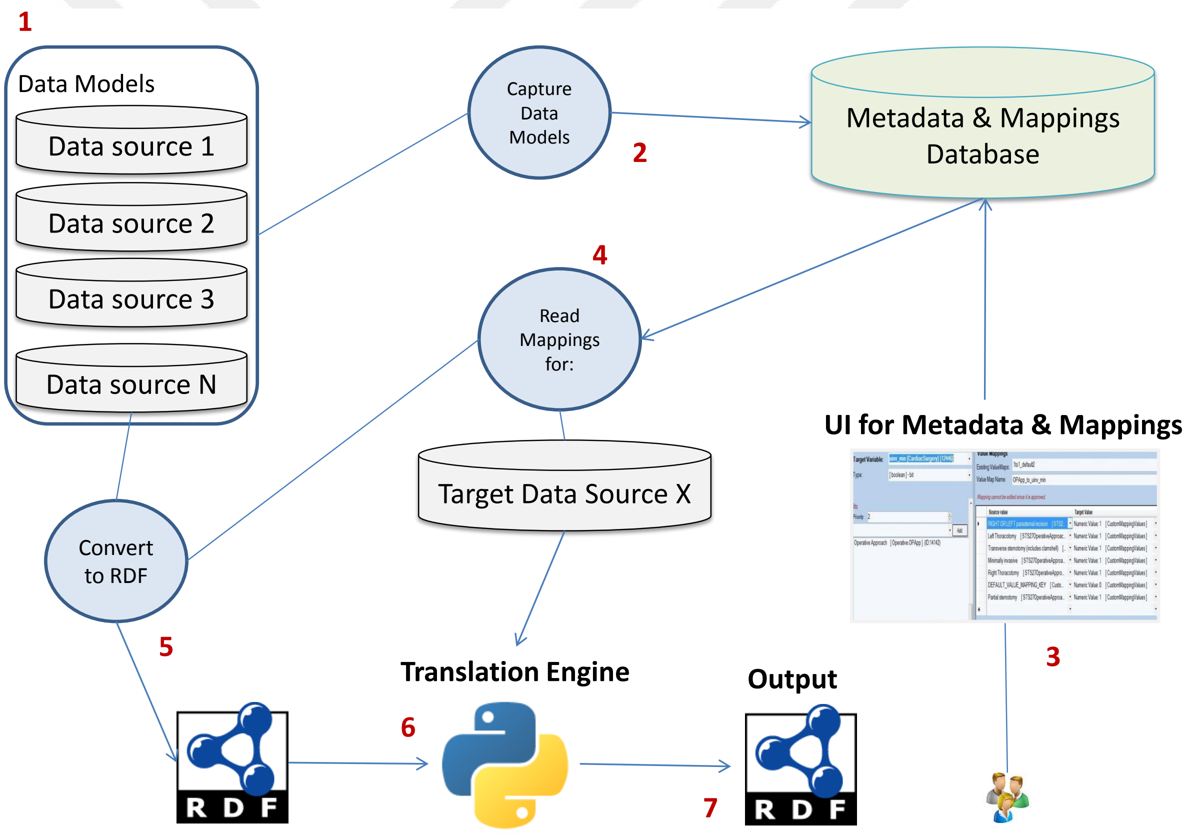
Translation requires mappings between the sets of data elements in the different data models. In our work, we present a mapping schema to enable capturing the mappings between different data elements in structured and machine readable format. Capturing the mappings between the data models in a machine readable format enables auto-generation of the translation code up to some degree.

3.1 Translation of Instance Data

Figure 4 illustrates the main components of the translation framework and their interactions as proposed in this work. The red labels help show how the translation components work together. The metadata for each data source (1) are captured and saved in the metadata repository or database (2). Further, the data managers define the linkages (mappings) between the data elements using a user-friendly interface, and these linkages are also saved in the metadata repository (3). In preparation for a translation, the defined mappings from the source data model to the target data model are retrieved (4). The source instance data are represented or lifted to RDF model (5). Then the translation engine uses the source RDF data, the target data model metadata, and the mappings from the source data model to the target data model to translate the source instance data into target data (6). The translation engine produces the results, and these results are conveyed to the RDF representation of the target data model (7). More details about the components are given in the following sections.

3.2 Metadata Repository

For each of the data sources, a data dictionary is captured and stored in a metadata repository [95]. In our work a data dictionary incorporates data model details for classes, concepts, concept values, and their associations to each other. For instance, the metadata of a dataset in relational schema [34] includes tables, table fields, pre-defined possible field values for the fields, as well as the relations between different metadata elements. And when represented in the metadata repository, the tables are represented as classes, table fields are represented as concepts and pre-defined possible field values are represented as



Target Variable: uinv_min [CardiacSurgery] [12446]
Type: [boolean] - bit
Priority: 2
Add
Operative Approach [Operative.OPApp] (ID:14142)

Value Mappings
Existing ValueMaps: 1to1_default2
Value Map Name: OPApp_to_uinv_min
Mapping cannot be edited since it is approved.

Source value	Target Value
RIGHT OR LEFT parasternal incision [STS2...	Numeric Value: 1 [CustomMappingValues]
Left Thoracotomy [STS27OperativeApproac...	Numeric Value: 1 [CustomMappingValues]
Transverse sternotomy (includes clamshell) [...	Numeric Value: 1 [CustomMappingValues]
Minimally invasive [STS27OperativeApproa...	Numeric Value: 1 [CustomMappingValues]
Right Thoracotomy [STS27OperativeAppro...	Numeric Value: 1 [CustomMappingValues]
DEFAULT_VALUE_MAPPING_KEY [Custo...	Numeric Value: 0 [CustomMappingValues]
Partial sternotomy [STS27OperativeApproa...	Numeric Value: 1 [CustomMappingValues]

Figure 4: Translation of instance data

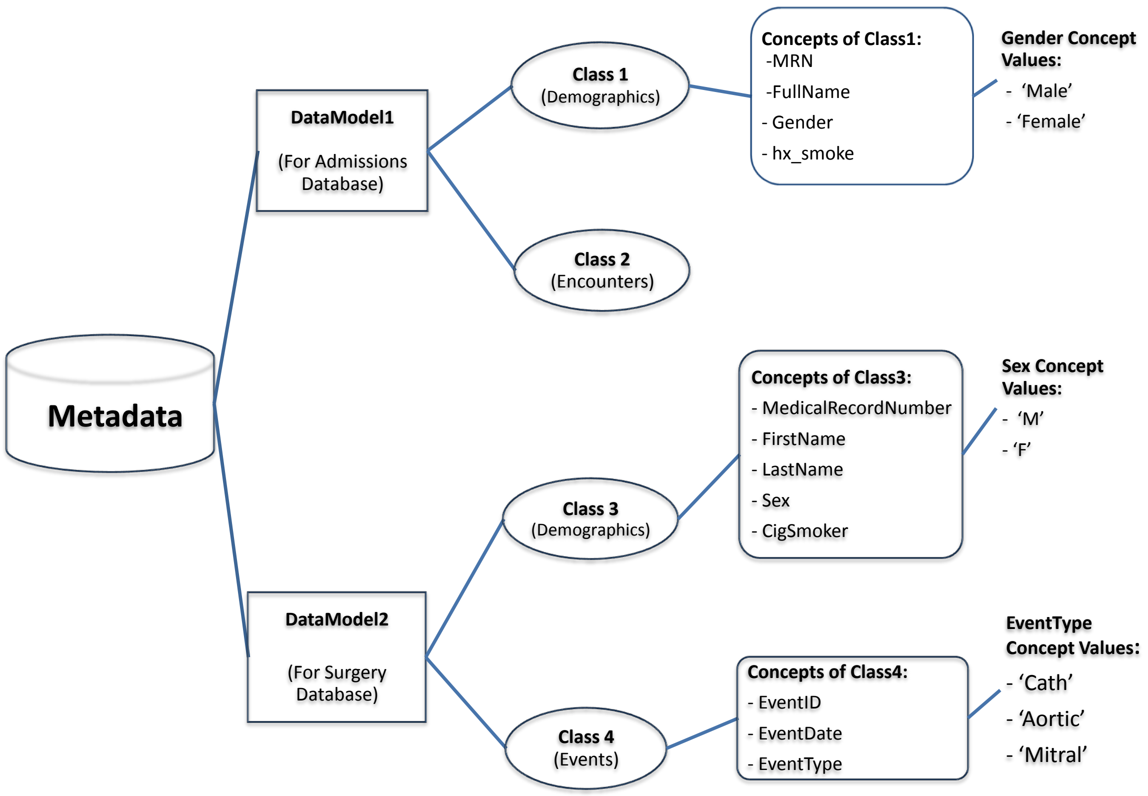


Figure 5: An example of data dictionary elements stored in the metadata server concept values.

Figure 5 shows the structure of the main data dictionary elements stored in the metadata server with sample data models, classes, concepts and value sets.

3.3 Mapping Schema

The metadata repository also includes a mapping schema. The goal is to enable storing the mappings between different data models in a structured format. The mapping schema is designed so that a target field can be mapped to from multiple candidate source fields that belong to one or more candidate source data models with field-level

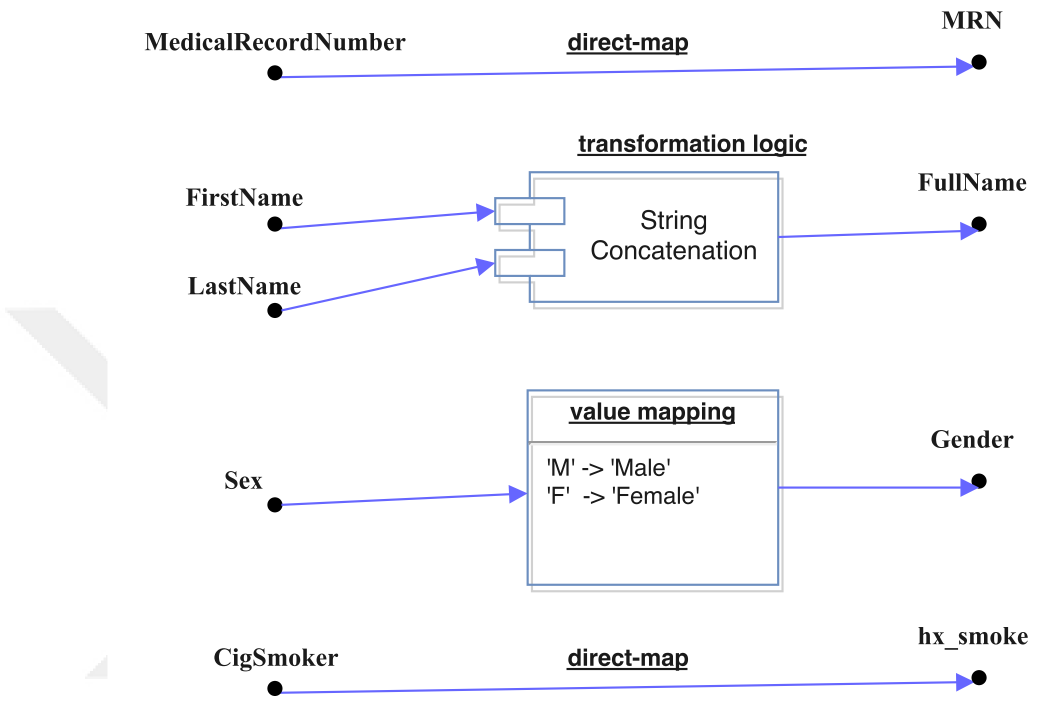


Figure 6: A mapping example showing how selected data dictionary elements are mapped

priorities, and a value set translation from source values to target values, and a pre-defined derivation (transformation) logic from source fields to the target field.

Figure 6 illustrates how the mappings are characterized between concepts. The figure explicitly shows the mappings from the concepts of "Class 3" to the concepts of "Class 1" of figure 5. The concepts and classes shown in the figure belong to the example data models defined in figure 5. As it seen in figure 6, the concept of "MRN" is mapped to from the concept of "MedicalRecordNumber" and the concept of "hx_smoke" is mapped to from the concept of "CigSmoker." The mapping to the concept of "FullName" involves a derivation logic, as it is mapped to from 2 different concepts ("FirstName," "LastName") and the translation engine needs to know the derivation logic. In this case the derivation logic is a string concatenation method. The "Gender" concept is mapped to from the

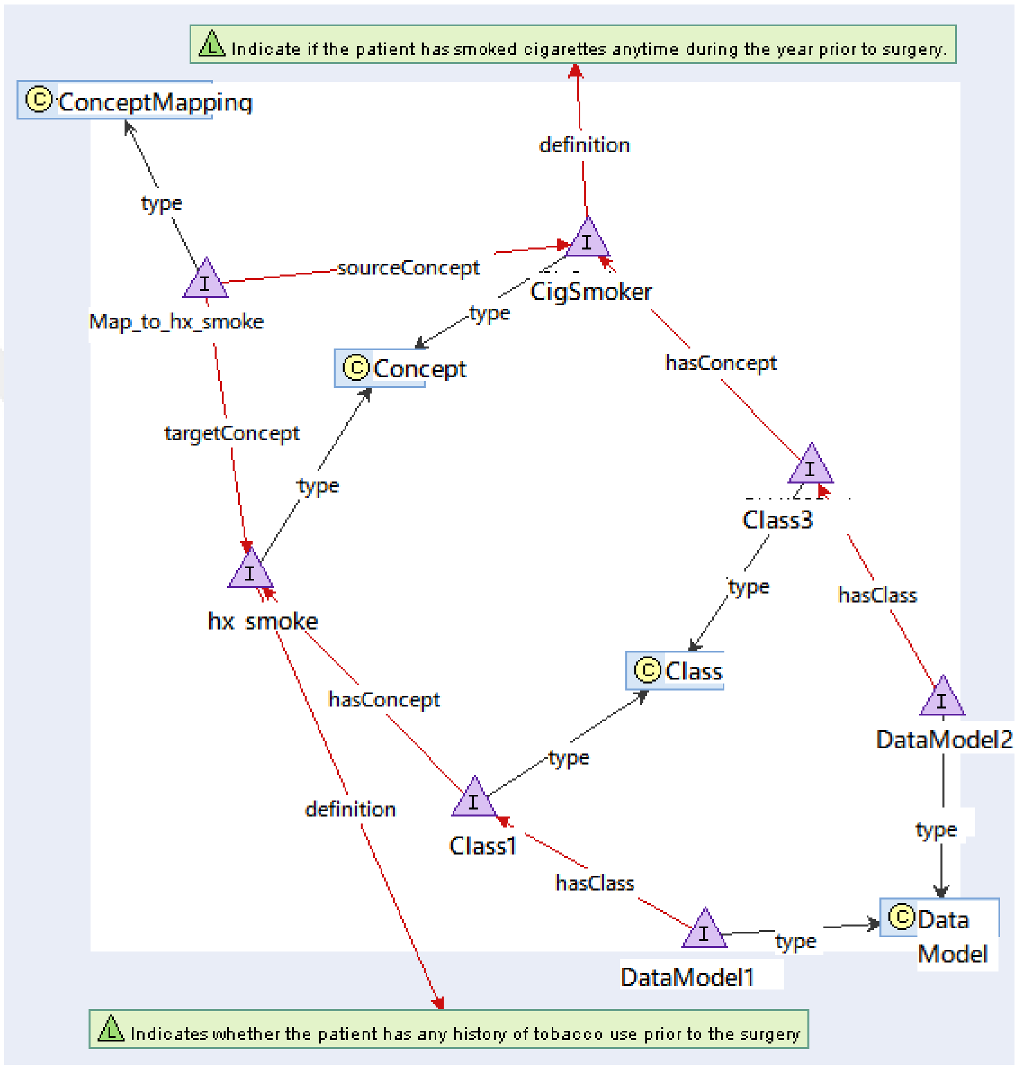


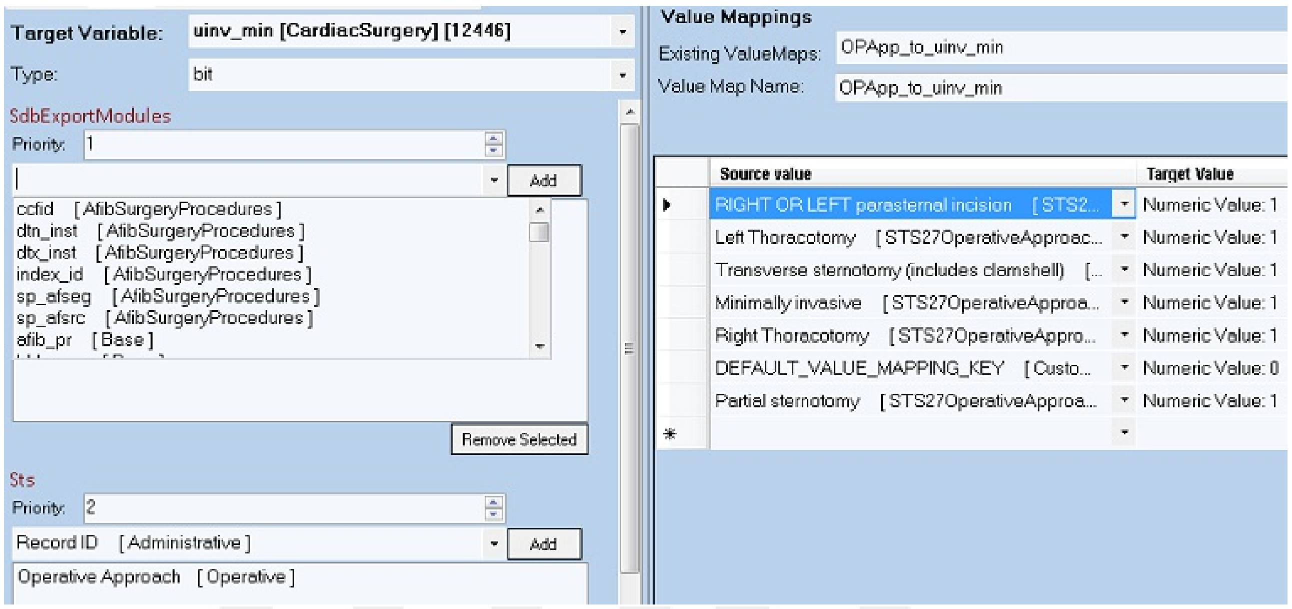
Figure 7: A figure showing the RDF representation of mapping between sample data dictionary elements

"Sex" concept. However the pre-defined values for "Gender" concept and "Sex" concept are different. Therefore the mappings additionally need to be defined on the concept values level. As seen in the figure, the value "M" mapped to "Male" while the value "F" is mapped to "Female."

In this work, data dictionary and mappings between the data dictionary elements are represented in RDF model. By doing this we are able create linkages between isolated data models. Figure 7 depicts a visualization of an excerpt from the RDF graph, that includes the mappings between the “hx_smoke” and “CigSmoker” concepts along with the concepts’ data models details. The data dictionary elements shown in the figure belong to the example data models defined in figure 5.

3.4 Mapping Interface

In a clinical domain, data concept knowledge and concept relations are known by the data domain experts. But the data is stored in physical data sources which are managed by the IT specialists. The IT specialists often do not know the meaning of the data stored in the data sources. They refer to data on the underlying data source schema level while the data domain experts or the end users refer to the data on data concept level. For example a doctor performs a procedure for a patient to cure a specific disease. The disease itself is a data concept. The doctor has the knowledge of the disease concept. But he/she does not necessarily have the knowledge of the data schema field where the disease concept is being stored. IT specialists have the knowledge of the metadata details of the data schema field but they often do not know the actual meaning of the data concept being stored in the data schema field. This causes a communication gap between the domain experts and the IT specialists in defining the mappings between different data concepts. Traditionally the mappings between different data concepts are defined by the data managers. However the data managers rely on the IT specialists to locate the data concepts in the physical data sources and create the linkages between the concepts. Therefore in this work, a user-friendly interface is presented to be utilized by the data managers. The interface allows viewing the data dictionaries, managing the concept definitions and defining mappings between different data elements. It uses



Target Variable: uinv_min [CardiacSurgery] [12446]

Type: bit

SdbExportModules

Priority: 1

ccfid [AfibSurgeryProcedures]
dtn_inst [AfibSurgeryProcedures]
dtx_inst [AfibSurgeryProcedures]
index_id [AfibSurgeryProcedures]
sp_afseg [AfibSurgeryProcedures]
sp_afsrc [AfibSurgeryProcedures]
afib_pr [Base]

Sts

Priority: 2

Record ID [Administrative]

Operative Approach [Operative]

Value Mappings

Existing ValueMaps: OPApp_to_uinv_min

Value Map Name: OPApp_to_uinv_min

Source value	Target Value
RIGHT OR LEFT parasternal incision [STS2...	Numeric Value: 1
Left Thoracotomy [STS27OperativeApproac...	Numeric Value: 1
Transverse sternotomy (includes clamshell) [...	Numeric Value: 1
Minimally invasive [STS27OperativeApproa...	Numeric Value: 1
Right Thoracotomy [STS27OperativeAppro...	Numeric Value: 1
DEFAULT_VALUE_MAPPING_KEY [Custo...	Numeric Value: 0
Partial sternotomy [STS27OperativeApproa...	Numeric Value: 1

Figure 8: A screen shot of the mapping interface.

the metadata repository as the backend database. The mappings defined by the data managers are stored back in the metadata repository within the mapping schema which is a machine readable format. In addition, the interface also has a test module that lets the data managers write and execute tests to validate their mappings.

Figure 8 shows a screen shot of the interface. In the figure the left side contains form elements to select the target and source concepts while the right side contains the form elements for identifying the value set conversion.

3.5 Translation Engine

The flexibility of RDF allows different schemata and models to be converted, represented and connected via RDF. In this work, the translation engine performs the translation on the RDF representation of the source data elements. The conversion from a source data format to the RDF representation differs for each data model, i.e., a different RDF data conversion routine is executed for each data format. The conversion from source instance data to the RDF representation happens on the fly during the translation process; only the specific source instance data, which is required by the mapping, is converted.

The usage of a metadata repository provides valuable benefits in data interoperability. Capturing the mapping knowledge in a structured format provides connectivity between disparate data elements, and it enables an automatic generation of the information translation code to some extent. This leads to more transparency in the translation process; e.g., in case of any data error it helps to track the source of the problem by checking the documented data mappings.

Data value translation enables the translation of a concept term in one data code system into an equivalent concept term in another data code system. The associations in the documented value mappings are used to discover the concept term conversion. Also, in many cases a single target data field is mapped to from more than one source data field. In this case the translation engine needs to know the priority of the source fields and/or a derivation logic to translate from multiple source fields. The priorities and the derivation logic can be defined by the data managers using the mapping interface. The translation engine is then able to generate the translation code by utilizing the defined priorities and the derivation logic.

3.6 Conclusion

In this chapter we presented a translation oriented framework to achieve interoperability between different healthcare data models. It alleviates the lack of interoperability solutions based on translation of RDF representation of instance data and structured mapping definitions between the data concepts belonging to different data models. We utilized a metadata repository to store the data dictionaries, which incorporate data model details for classes, concepts, concept values, and their associations to each other. We also introduced a user-friendly interface that permits the data managers to view and manage the data dictionary elements and the mappings between them.

In this chapter we assumed that there is already a defined data dictionary in place for each data source. This assumption is correct for some data sources, e.g., the data dictionary of a relational database can be captured automatically using tools like SchemaSpy [30]. However, capturing the data dictionary from a dataset that is already represented in RDF model can be a challenging task due to the flexibility of the RDF data model not imposing constraints on the schema. In chapter 4 we introduce a proficient pairwise graph node similarity metric. Following with chapter 5 we present a summary graph generation method based on the pairwise graph node similarities introduced in chapter 4 and we clarify how the summary graph can be exploited to capture the data dictionary elements from RDF representation of the instance data.

Translation of instance data requires data models level mapping, which means the data concepts belonging to the source and target data sources need to be mapped accordingly. The interface introduced in this chapter lets the data manager characterize the mappings between healthcare data concepts belonging to different data models. However manually defining the mappings requires significant amounts of time due to the size of the healthcare data. In chapter 6, we introduce a semi-automatic instance matching technique in RDF graphs that leverages the pairwise graph node similarities introduced in chapter 4 and we describe how the instance matching technique introduced in chapter

6 can be utilized for semi-automatically generating the mappings between different data models.

CHAPTER 4

Pairwise Nodes Similarity in RDF Graphs

In this chapter, we introduce an efficient algorithm to compute the similarity between different healthcare data elements. The main ideas and sections explained in this chapter are part of the published research studies [10,13]. Since we represent the healthcare data in an RDF model, the computation of entity similarity is studied as a pairwise RDF graph node similarity problem. Computation of entity similarity is essential for numerous interoperability tasks. For instance, automatic classification algorithms group similar entities together which requires an entity similarity metric. Also, instance matching and concept mapping techniques utilize entity similarities to link the same or similar real-world objects.

The algorithm is based on an efficient graph node similarity metric which is called RNodeSim (RDF Node Similarity). An RDF entity is described through a set of predicates, the collection of literal neighboring nodes that it references and the neighbor nodes with which it interacts. We utilize the common descriptors within the Jaccard measure context when calculating the similarity of an RDF node pair, along with the similarities of their neighbors.

Each descriptor of an RDF node may have a different impact in the similarity calculation, in other words each descriptor has a different importance weight. Therefore in this work, we present an importance weighting metric for the descriptors of the RDF nodes and we enhance the similarity metric by incorporating the auto-generated importance weights of the descriptors. The algorithm runs in multiple iterations as the similarity of the neighbor nodes is not known in advance. The iterations continue until the pair similarity results converge.

4.1 Contributions and Outline

In this chapter, we investigate the problem of calculating entity similarity in RDF graphs. Our approach is to use graph locality and neighborhood similarity within the Jaccard measure context. We treat the properties of the entities as the dimensions when measuring the similarity of two subject entities. Our main contributions:

- We provide a novel RDF node pair similarity metric that utilizes common descriptors.
- We auto-generate the importance weight of each descriptor, and we apply the weights in the pairwise similarity calculation.
- We add a string similarity measure when two graph nodes have literal type neighbors.
- We enhance the RDF node pairs' similarity by taking into account the similarity of neighbors in the RDF graph.

The rest of the chapter is organized as follows. We define the input data graph. Then, we discuss the proposed methods. The subsequent section presents our approach for computation of entity similarity and the algorithm in detail. Finally, we review the related work, and follow with our conclusion.

4.2 Input Graph Data

The input data is an RDF graph $G(V, L, E)$ as defined in section 2.2.1.

4.3 Methods

The intuition in our graph node pair similarity computation is that the nodes that have similar edges to similar neighbors tend themselves to be similar nodes. To achieve our goal, we define the similarity of two nodes in the original graph using a function similar to the Jaccard index [50] in conjunction with RoleSim Similarity [54].

The Jaccard similarity measure can be applied to graph data to find the similarity of the entities by considering the graph nodes as set names or labels and the predicates between the nodes as set elements. However, it is clear that the Jaccard similarity only considers the properties of the entities when determining the similarity of the entities; it does not take into account the similarity of neighboring entities. By a neighbor of an entity, we mean another entity which is "connected" by a property. In our work, given an input RDF graph $G(V, L, E)$, entity v is connected to entity u, i.e., v is a neighbor of u, if there is a label $l \in L$ such that $l(u, v) \in E$. In other words, objects are neighbors of their subjects for a collection of RDF triples. Thus, an entity and its neighbor are connected and related by a property.

Intuitively, when we are trying to decide how similar two entities are, it is, of course, important to compare the properties of the two entities. However, the properties by themselves tell only part of the story. More can be told by comparing the neighboring entities which are related by similar properties. Thus, by utilizing the neighborhood similarity, we may produce better results than by using only the Jaccard similarity.

For this reason, we use the RoleSim similarity measure to determine the similarity of the interacting references based on the way they interact with their neighborhood nodes. In other words, two nodes or entities tend to have the same role when they interact with equivalent sets of neighbors.

4.4 Computation of Entity Similarity

Our premise is that similar nodes tend to have similar properties and interact with similar neighbor nodes, which are either IRIs or literals.

4.4.1 IRI Nodes Similarity

In this work, the subjects of the triples determine the sets in which we are interested. We think of the subjects as being the names or labels for the sets. More exactly, the subject of each triple determines a node of a new graph which is itself a set, and the elements of each set are the predicates of the triples whose subject is the name or label of the set. The objects of the triples, which may themselves be names or labels of sets, determine the neighbors of the subject sets. Hence, we use the subject nodes in the original graph to designate sets and their properties in the original graph are the elements in these sets. We calculate the Jaccard similarity between two nodes u and v by noting that $|u \cap v|$ is the number of properties that the subject nodes u and v have in common while $|u \cup v|$ is the number of properties in the union of the subject nodes u and v.

While the Jaccard index gives a good initial similarity between two nodes, it can be improved since it does not take the importance of neighborhood similarities into consideration. To improve our similarity calculations, the Jaccard similarity of the two nodes is augmented with the similarity of their neighbors. However, we do not compare all neighbors to all neighbors. We compare neighbors selectively. Let's assume we are trying to determine the similarity of two nodes u and v. Further, we assume that s is a neighbor of u, and t is a neighbor of v. We determine the similarity of s and t only if there is a common property which connects u to s and also connects v to t. Then we apply the RoleSim measure in conjunction with the Jaccard index to calculate the similarity of two graph nodes augmented with similarities of their neighbors.

We note, however, that the input data for the generic version of the RoleSim measure is not a labeled graph. We extend the RoleSim similarity to labeled graphs since RDF triples contain labeled edges. In this work, the $RoleSim(u, v)$ measure is used when there may be multiple neighbors with a commonly labeled edge reached from the node pairs u and v, where u and v are subject nodes in the input graph.

In the lists below, for $1 \leq i \leq |L|$, l_i is a label for an edge, i.e., $l_i \in L$. When $1 \leq h \leq |L|$ and if i and h are not equal, then l_i and l_h are different labels, i.e., l_i and l_h are different properties. $[x_i]$ and $[y_i]$ are the sets of nodes which are related to u and v, respectively, by property l_i. $[x_i]$ and/or $[y_j]$ may for some properties l_L be the empty set.

$l_1(u,[x_1]), l_2(u,[x_2]), ...l_j(u,[x_{|L|}]) \in E$

$l_1(v,[y_1]), l_2(v,[y_2]), ...l_j(v,[y_{|L|}]) \in E.$

For each $l_L \in L$, then for each $z \in [x_i]$, $l_i(u,z) \in E$ and for each $w \in [y_j]$, $l_i(v,w) \in E$.

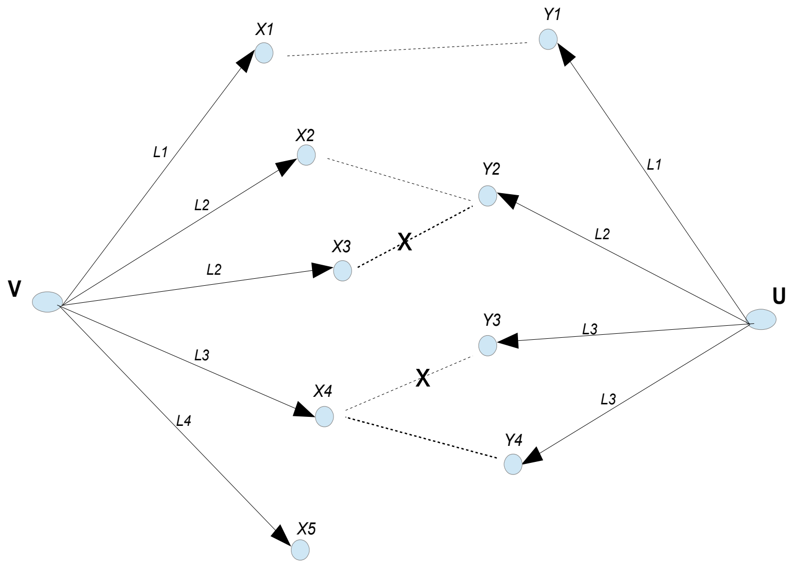


Figure 9: Example Graph for Graph Equivalence matching

Then by using the Jaccard index in conjunction with the RoleSim measure on the common properties, their similarity can be calculated as

$$
\begin{aligned}
RNodeSim(u,v)^k =& (1-\beta) \\
&\times \frac{1}{|u \cup v|} \\
&\times \Big(\sum_{j \in (u \cap v)} max_{M \in Mm^j(u,v)} \Big(\frac{\sum_{(x,y)\in M} RNodeSim(x,y)^{k-1}}{N_u^j + N_v^j - |M|} \Big)\Big) \\
&+\beta
\end{aligned} \tag{4}
$$

where k is the iteration number, such that, if $k = 3$ then $RNodeSim(u, v)^k$ denotes to the similarity of the node pair (u, v) at the third iteration and $RNodeSim(u, v)^{k-1}$ denotes to the similarity of the node pair (u, v) by the end of the second iteration. Also, $N^j_{(u)}$ and $N^j_{(v)}$ denote their respective neighborhoods that are reached by a common edge. $x \in N^j_{(u)}$ and $y \in N^j_{(v)}$, and N^j_u and N^j_v denote their respective degree connected by jth common edge. Said differently, $N^j_{(u)}$ is the cardinality of $[x_j]$, and $N^j_{(v)}$ is the cardinality of $[y_j]$.

We define M to be a set of ordered pairs (x, y) where $x \in N^j_{(u)}$ and $y \in N^j_{(v)}$ such that there does not exist $(x', y') \in M$, s.t. $x = x'$ or $y = y'$, and furthermore, M is maximal in that no more ordered pairs may be added to M and keep the constraint above. $Mm^j(u, v)$ is the set of all such M's. $Mm^j(u, v)$ is a set of sets.

By a "maximal nonrepeating matching," we mean that we form as many pairs as we can from the elements in $N^j_{(u)}$ and $N^j_{(v)}$ with the restriction that no element in either $N^j_{(u)}$ and $N^j_{(v)}$ may be used in more than one ordered pair.

4.4.2 Literal Node Similarity

As stated in the previous section, our IRI nodes similarity measure is based on calculating the similarity of entities by utilizing the predicates of the IRI nodes. Our premise is that similar nodes tend to have similar properties and interact with similar neighbor nodes, which are either IRIs or literals. Though it is challenging to infer the semantics of literal nodes, an effective literal node similarity metric is needed for calculating the similarity of pairs of IRI nodes when some of the neighbors of the IRI nodes are literal nodes. We think that incorporating literals when computing the similarity of pairs can be beneficial when identifying similar entities, particularly in datasets where the entities are commonly described using literals. Thus, we want to take the similarity of literal neighbor nodes into account when doing similarity calculations.

While incorporating literals in the computation of the similarity of IRI node pairs, we are assuming that all the literals are in the same language, as the same literals may have totally different meanings in different languages. Thus there is only one value for the rdf:langString component of the literal nodes if present. In this work, we disregard the third component of rdf:langString if present, which means we work only with the lexical form and the data type URI component of a literal node. Both the lexical form and the data type URI component clearly impact the similarity of a pair of literal nodes. Thus, they indirectly impact the similarity of IRI nodes in the calculation of neighborhood similarity, and their impacts need to be weighted. Since it is meaningless to calculate the similarity of literal nodes when data types are different, we only give weight to the data type factor when the two data types are equal. For the lexical form components of the literal nodes, we use a string similarity technique based on common words within the two lexical forms along with their auto-generated importance weights. More details about the importance weights will be given in the following section.

4.4.3 Descriptor Importance and Automatic Detection of Noise Labels

An IRI node is described through its predicates and the collection of literal neighboring nodes in the lexical form. We call these the descriptors of the IRI nodes. The similarity of two IRI nodes is calculated from their descriptor similarities including the similarities of their neighbors. The accuracy of pairwise graph node similarity is often impacted by the weight of a property associated with the graph nodes when the nodes are object nodes, or with the weight of a string literal word referenced by the graph nodes when the nodes are literal type. Each descriptor may have a different impact on an IRI node. Identifying appropriate metrics for generating weights for the IRI descriptors to be utilized in the pairwise graph nodes similarities is a formidable yet significant task.

In this chapter, we investigate the factors that can impact the weight of a descriptor. We propose an approach for generating the importance weights of the IRI node descriptors automatically. Our approach is based on two premises: (1) the weight of a descriptor may differ for each IRI for which it is a descriptor and (2) the weight increases proportionally by the number of times a descriptor appears in the reference IRI, but it is offset by the frequency of the descriptor in the entire RDF dataset. It is a similar notion to the term frequency-inverse document frequency (tf-idf) [61, 80], a commonly used technique in information retrieval, indicating that some words may be important in some documents but not as important in other documents. More exactly, the importance of a word in a document increases by its frequency in the document but its importance decreases by its frequency in the corpus [76]. We apply the tf-idf concept to the properties and nodes in RDF graphs to compute the weight of properties. Given an RDF graph $G = (V, L, E)$ as defined in section 2.2.1, tf-idf is calculated as follows:

$$tf - idf(p, u, G) = tf(p, u) \times idf(p, G). \tag{5}$$

where the term frequency (tf) [61] is related to the frequency of a proposition p with respect to a graph subject node u. More exactly, when $u \in V$ and $p \in L$, then

$$f(p, u) = |\{v \in V : p(u, v) \in E\}|. \tag{6}$$

Equivalently, $f(p, u)$ is the number of RDF triples with subject u and property p.

To define $tf(p, u)$, it is helpful to have a notation for the set of all properties with subject u. Thus, for $u \in V$, $L(u) = \{q \in L : \exists v \in V \text{ with } q(u, v) \in E\}$. Then

$$tf(p, u) = \frac{f(p, u)}{\sum_{q \in L(u)} f(q, u)}. \tag{7}$$

The inverse document frequency (idf) [80] represents the frequency of a property usage across all graph nodes, and it is defined as

$$idf(p, G) = \ln \frac{|V|}{|\{u \in V : p \in L(u)\}|}. \tag{8}$$

We apply a similar approach to calculate the weight of word importance in literal nodes, which can consist of a set of words. A value for a DatatypeProperty is a string literal. The weight of word importance depends upon the source subject node, the frequency of the word within the triple collection for each subject node, and the frequency of the word within the entire data set.

Detection of Noise Descriptors

We calculate the descriptor importance and assign weights depending on the degree of distinctiveness of a descriptor describing an entity. With descriptor distinctiveness, we mean the uniqueness of a descriptor in describing the key characteristics of an entity type. For instance, if a descriptor is specific to an entity type, it is a distinguishing character of the type from other types. When a descriptor exists in all entity types, its quality of being distinctive is low. The noise descriptors tend to be common for a majority of entities if not for all entities. For instance, in the LUBM [43] dataset described in section 5.7, the name property is a common property for all subject nodes in the dataset. Thus, the name property is a noise descriptor.

By increasing importance weights of descriptors with a higher degree of distinctiveness, we reduce the importance of noise descriptors automatically. As a result, the noise descriptors have significantly less impact on the overall similarity measures.

4.5 The Algorithm

Augmenting the neighborhood similarities adds extra complexity to the algorithm because only immediate neighbors are known in advance. Immediate neighbors' neighbors or similarities depend on two-hop neighbors' similarities; two-hop neighbors' similarities depend on three-hop neighbors' similarities, and so on. Hence the algorithm has to run in multiple iterations until the results converge. To perform these calculations, we propose an iterative algorithm which is similar to the PageRank algorithm [72]. At initiation, the similarity of all node pairs is set to 1 if they share a common property and 0 otherwise. Thus,

$\forall(u, v \in V):$

$(S(u,v) = 1) \rightarrow (|u \cap v| > 0)$ and,

$(S(u,v) = 0) \rightarrow (|u \cap v| = 0)$

The algorithm runs iteratively until the similarity measure for each pair converges. In the worst case, this technique requires n^2 passes if all of the nodes in the graph have a common edge with every other node in the graph. In reality, this situation is very rare. In fact, a node pair only has to be processed if the nodes share a common property. Hence, the similarity computation of the algorithm runs in the $n(logn)k$ time in average, where k is a constant number of iterations. However, the overall cost remains n^2 due to the generation of common pairs and the similarity matrix of entities.

Below are the steps of the algorithm:

- Sort the triples according to their edge labels
- Process the sorted triples to extract the pairs of nodes which have at least one common edge label
- Keep running the similarity metric above in a loop until the similarity measures converge

The parameter β is a decay (damping) factor [98], $0 < \beta < 1$. It is similar to the one

Algorithm 1 $SimMeasure(G(V, L, E), \beta, MaxIter)$

```
$\forall pair(u, v) \in V(S(u, v) \leftarrow 0)$
$H \leftarrow \emptyset$
$T \leftarrow Sort(E)$ by $l, u, v$ s.t. $l(u, v) \in E$
**for each** distinct $pair(u, v)$ from $T$ **do**
    **if** $\exists(l_1(u, x)$ and $l_2(v, y))$ s.t. $l_1 = l_2$ **then**
        $S(u, v) \leftarrow 1$
        $P(u, v) \leftarrow (u, v, L^j, N^j(u), N^j(v))$ where $u, v \in V$ and $L^j \in L$ is the list of common labels between $u$ and $v$
        $H \leftarrow H \cup P(u, v)$
    **end if**
**end for**
$S_{previous} \leftarrow \emptyset$
$converged \leftarrow 0$
$count \leftarrow 0$
**while** $converged = 0$ and $count < MaxIter$ **do**
    **for each**$((u, v, L^j, N^j(u), N^j(v)) \in H)$ **do**
        $S(u, v) = RNodeSim(u, v)^{count}$
        $converged = IsConverged(S, S_{previous})$
        $S_{previous} \leftarrow S$
        $count \leftarrow count + 1$
    **end for**
**end while**
**return** $S, H$
```

used in [72] and [54] as explained in section 2.5.2. The usage of β decreases the influence of neighbors with distance and dampens the recursive effect. We provide more details about how the value of β is chosen in our evaluations described in section 5.7. Also, $l_1(u, x)$ and $l_2(v, y)$ represent directed edge labels s.t. $l_1, l_2 \in L$, and $l_1 = l_2$, $x \in N(u)$ and $y \in N(v)$.

For example, let's consider the similarity of the sample graph with the nodes of two universities in Figure 10. At the initial state, the similarity of the two nodes $Kent\_State$ and $Case\_Western$ is 1. As we run through the iterations, the similarity values are being updated and eventually converge to final values. Note that u is $Kent\_State$ and v is $Case\_Western$.

$$
\begin{aligned}
S(u,v)^k =& \frac{1}{|u \cup v|} \times (\sum_{j=1}^{|u \cap v|} max(\frac{\sum\limits_{(x,y)\in Mm(u,v)} S(x,y)^{k-1}}{N_u^j + N_v^j - |Mm^j(u,v)|}))\\
S(u,v)^1 =& \frac{1}{10} \times (\frac{1}{1+1-1} + \frac{1}{1+1-1} + \frac{1}{1+1-1}\\
& + \frac{1}{1+1-1} + \frac{1}{2+1-1} + \frac{1}{2+1-1}\\
& + \frac{1}{1+2-1} + \frac{1}{1+1-1} + \frac{1}{1+1-1})\\
S(u,v)^1 =& \frac{1}{10} \times (6 + 0.5 + 0.5 + 0.5) = 0.75
\end{aligned}
$$

4.6 Related Work

There have been many studies investigating similarity measures in various contexts, including but not limited to Social Networks, Recommendation Systems, Semantic Search,

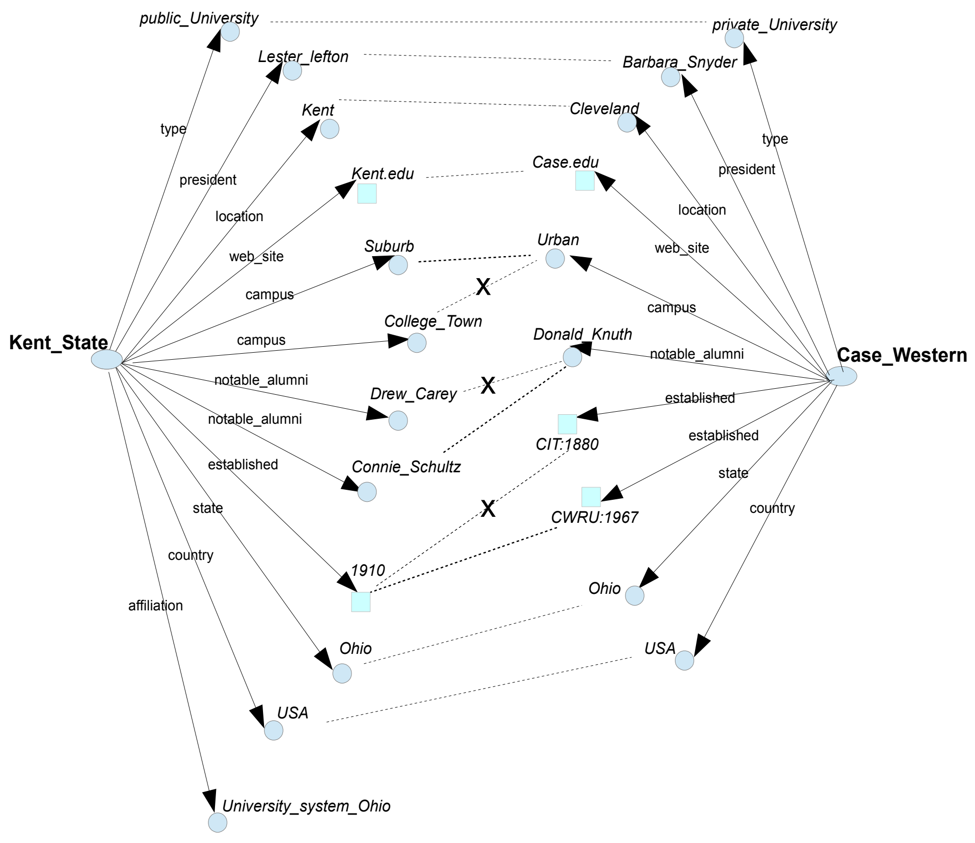


Figure 10: Sample graph demonstrating two nodes

Ontology Mapping, and Record Linkage. Among many others, the Jaccard measure [50] and Cosine Similarity [77] are notable measures for comparing the similarity of pair objects.

Some studies explored neighborhood-based similarity measures such as Co-Citation [79], SimRank [53], PageSim [59], MatchSim [60], SimRank++ [7], and PathSim [84]. Particularly, SimRank is well-known for calculating similarity in a structural context. However, SimRank does not work well on objects with multiple connections. By taking the mean of the edges, it sometime reduces the power of strong relations when it should increase the similarity score, intuitively.

While most of the similarity calculations do not take the property weights into account as in [53], there are some studies that try to calculate the property weights and apply them in similarity calculations.

H-Match [26] tried to detect the property weights using the distinct value-based weight generation, assigning higher weight to a property that references more distinct values. However, a training set of instances may not always be available.

In [29] the authors suggest that properties with a maximum or an exact cardinality of one have a higher impact in instance matching, thus having a higher property weight. This assumption does not work well in instance type discovery. For example, in a university-related dataset a more specific property such as hasPresident should have more impact in type discovery than a more general property hasName, even though both of the properties have the cardinality of one. In this case, the assumption would misleadingly assign the same weight to both of the properties.

On the other hand, [78] considers the ratio of the number of distinct values of a property to the number of instances in a dataset in addition to the number of distinct values referenced by the property. However, they primarily focus on instance matching, where property weights naturally yield precedence to properties that make the instances more unique. Unlike the instance matching approach, we emphasize the properties that would help describe the entity types more distinctively.

4.7 Conclusion

In this chapter, we introduced a proficient pairwise graph node similarity algorithm. The algorithm utilizes a pairwise graph node similarity metric and runs in iterations. We explained the descriptors of an RDF entity to be utilized in the similarity calculation. We also introduced an importance weighting metric for the descriptors of the RDF nodes. Our pairwise graph node similarity metric incorporates the common descriptors of an RDF node pair augmented with their auto-generated importance weights and their neighbor node similarities.

The subsequent chapters make use of the pairwise graph node similarity concepts explained in this chapter. Chapter 5 introduces a summary graph generation method based on graph node similarities. Also, the semi-automatic instance matching technique presented in chapter 6 utilizes the pairwise graph node similarities.

CHAPTER 5

Building Summary Graphs of RDF Data in the Semantic Web

In this chapter, we introduce RSummarizer (RDF Summarizer), a summary graph generation approach which utilizes the pairwise graph nodes similarity methods presented in chapter 4. The main ideas and sections explained in this chapter are part of the published research studies [10, 13].

As we represent the healthcare data in an RDF model, processing an entire graph for various algorithmic tasks can be costly in terms of time and resources due to the size of RDF graphs. The summary graph structure demonstrating the inferred class types and class relations can be beneficial for intelligent explorations of graph shaped data as it helps understand underlying structure, and provides an intermediate index structure for faster computational processing as the entire input data does not need to be completely processed. Explicitly, the entity types in RDF triples provide invaluable information for graph processing as they enable the graph processing algorithms to traverse the hierarchical structure of the RDF graph. Unfortunately, many structured data sources available in the Web today commonly suffer from (1) not containing the type triples due to the flexibility of the RDF data model not imposing constraints on the schema, (2) publishing data with non-standard vocabulary, which makes it problematic to locate the type triples, and (3) defining the type information too generally to ensure that entity types will have related sub-types. In the current chapter, we propose an algorithm for discovering the types of entities in RDF data and for building a summary graph structure for faster computational processing.

The summary data structure consists of the type classes, members of the type classes, and the relations between the type classes. In our work the summary data structure is also represented in an RDF model that forms a summary graph as defined in section 2.6.1. A collection of RDF resources having the same type are a class which is represented as a node type entity in the summary graph, and the relations between the class type entities are represented as a predicate. However, auto-generation of a summary graph demands a metric to evaluate the degree of accuracy of a type class relation in the summary graph. Therefore we introduce a metric to measure the degree of confidence in the class relations, which we call the stability measure.

In chapter 3, we introduced a data dictionary structure which includes the data model details for classes, concepts, concept values, and their relations. We indicated that such a data dictionary model may not always be available, especially for the linked data represented in RDF model. In the current chapter, we describe the methods of extracting the data dictionary structure from the summary graph. We also explain how the summary graph can be exploited for various tasks, including filtering the entities by type and type relations.

5.1 Contributions and Outline

In this study, we investigate the problem of discovering very similar entities that could be classified as the same type so that multiple entities can be represented by one class entity type. This has been traditionally studied as a clustering problem. In RDF data, we consider this problem a graph summarization problem as RDF can be expressed as a graph. Also, we propose building a supplementary summary graph structure for more efficient computation. In this work, we don't rely on the existence of rdf:type or owl:sameas. Our approach is based on the pairwise graph node similarity calculation introduced in chapter 4 and obtaining the class structure and hierarchy from the data itself.

Our main contributions:

- We provide a novel machine learning algorithm that automatically infers the type semantics from the general RDF graph.
- We construct a summary graph structure based on pairwise similarity matrices of entities that can enhance the performance of graph explorations.
- We generate the summary graph along with the classes and class relations with a stability measure for each class relation.

The chapter is organized as follows. We discuss our approach and the algorithm in detail. The subsequent section presents the class predicate stability metric. Then, we present the results of the evaluations. Finally, we review the related work and follow this with our conclusion.

5.2 Methods

Our graph summarization approach is based on method 3 of the summary graph generation methods described in section 2.6.2, in which the summary graph is built automatically by inferring the class types based on the similarity of the RDF nodes.

We consider graph summarization as the discovery of the set of classes such that each class in this set is the collection of the same or very similar entities. As the similarity measurements between pairs of entities are calculated, the determining of the elements, i.e., classes, in V' can be treated as a clustering problem with the aim being the combining of the same or similar entities into the same class type.

The intuition that is new to our summarization graph work is that the nodes that have similar edges to similar neighbors tend themselves to be similar nodes; thus, they should be in the same class. To achieve our clustering goal, we define the similarity of two nodes based on the entity similarity metric described in chapter 4.

5.3 The Algorithm

The pairwise graph similarity algorithm introduced in chapter 4 runs iteratively until the similarity measure for each pair converges. At the end of the iteration, the nodes u and v are put into the same class if their dissimilarity is less than a defined threshold ϵ. The class creation algorithm generates the distinct classes based on the given epsilon threshold.

5.4 Generation of the Classes and Properties

The summary graph for an RDF dataset is built automatically, and the constructed summary graph is also represented in RDF in our approach. The IRI nodes that have similarity higher than a defined threshold are considered to be of the same type, and they are categorized in the same class in the summary graph. In our work, we do not classify the literal nodes based on their similarity: instead, we add an additional type class to the summary graph that incorporates all the literal nodes. A class relation between a

Algorithm 2 $CreateClasses(S, H)$

```
Input: SimMatrix S, Pairs H
Output: Classes C
  $C \leftarrow \emptyset$
  for each($(u, v, L^j, N^j(u), N^j(v)) \in H$) do
      if $C(u)$ exists then
          $c_i \leftarrow C(u)$
      else
          $c_i \leftarrow \{u\}$
      end if
      if $1 - S(u, v) < \epsilon$ then
          if $C(v)$ exists then
              $c_i \leftarrow c_i \cup C(v)$
          else
              $c_i \leftarrow c_i \cup \{v\}$
          end if
      end if
      $C \leftarrow C \cup c_i$
  end for
  return $C$
```

class $c1$ and a class $c2$ is generated as a predicate and represented as $l(c1, c2)$ when there is at least one relation $l(u, v)$ such that u is an IRI and v is either an IRI or a literal node in the dataset, $G = (V, E, L)$, and $u \in c1$ and $v \in c2$ with both $c1$ and $c2$ being type classes in the summary graph, $G' = (V', E', L')$. Then we have $l \in L'$ and $l(c1, c2) \in E'$.

5.5 Class Relation Stability Metric

Automatically generated summary graphs can be error prone. Therefore, a metric to measure the degree of confidence of a relation between classes in the summary graph would be beneficial. We call this metric Class Predicate Stability (CPS). The CPS is similar to the stability concept introduced by Paige and Tarjan [73].

For a triple $(c1, p, c2)$ in the summary graph G' with $c1$ and $c2$ being type class IRI nodes and p being a predicate between them, the CPS metric is calculated as the number of the IRI nodes u in class $c1$ having a triple of the form (u, p, v) with $u \in c1$ and $v \in c2$

divided by the total number of the IRI nodes in $c1$ in the summary graph. $CPS(c1, p, c2)$ is formulated as

$$CPS(c1, p, c2) = \frac{|\{u \in c1 : \exists v \in c2, p(u, v) \in E\}|}{|c1|} \tag{9}$$

where $|c1|$ is the number of IRI nodes in the class $c1$. Note that $|c1| > 0$. We define full CPS as follows: for two classes $c1$ and $c2$ and the predicate p in the summary graph either all the IRI nodes from $c1$ are connected with the predicate p to at least one IRI node in $c2$ or none of the IRI nodes in $c1$ is connected with the predicate p to an IRI node in $c2$. Thus, $CPS(c1, p, c2) = 1$ for a full CPS.

The CPS value for a triple $(c1, p, c2)$ in the summary graph indicates how strongly connected and how coarsely partitioned the type classes $c1$ and $c2$ are with the predicate p. Thus, the average of all the CPS values in the summary graph is a measure of accuracy for the generated summary graph. $CPS(G')$ is formulated as

$$CPS(G') = \frac{\sum\limits_{i=1}^{|E'|} CPS(c1^i, p^i, c2^i)}{|E'|} \tag{10}$$

where $G' = (V', E', L')$ is the summary graph and $p^i(c1^i, c2^i) \in E'$, and thus $|E'| > 0$.

Another advantage of calculating the CPS metric is that it can further be utilized in semantic search algorithms. In traditional semantic search algorithms, the relations between two different type classes are assumed to be tightly coupled [86]. In real situations this assumption may not always be true, especially if the summary graph is auto-generated as in our study. We propose that the CPS metric can be used as an impact factor between two type classes and utilized in the semantic search graph traversal for more accurate results.

5.6 Data Dictionary Extraction

In chapter 3, we introduced a translation-based healthcare information interoperability framework that makes use of data dictionaries. Such a data dictionary can be obtained from the data sources that have a well defined schema in place, e.g., a relational database schema, XML schema, RDFS or OWL ontology. Nonetheless, extracting the data dictionary from an RDF dataset is a challenging task if the RDF dataset is not tied to an ontology. In this section, we explain how a summary graph can be utilized to obtain the data dictionary elements from RDF representation of the instance data.

A data dictionary includes data model elements for classes, concepts, concept values, and their relations. On the other hand, the summary graph incorporates the type classes, members of the type classes, and the predicates between the type classes. Accordingly, each type class in the summary graph becomes a class in the data dictionary and each predicate of the type class becomes a concept of its associated class in the data dictionary.

The concepts extracted from the predicates that reference IRI nodes type classes in the summary graph are called IRI concepts. A string data type is considered for the IRI concepts. The concepts extracted from the predicates that reference to the literal node type class in the summary graph are called literal concepts, and the collection of the literal nodes that the original predicate reference to is called the concept's literal nodes. The actual data type of one of the concept's literal nodes is considered for the literal concept if all the concept's literal nodes have the same data type, otherwise, a string data type is considered for the literal concept.

Value sets extraction is done for the literal concepts that have a string data type. The values extracted from the concept's literal nodes, those that are used more than a defined threshold, constitute the concept's value set.

5.7 Evaluation

In the evaluations, we assessed the effectiveness of the proposed approach on three datasets: a subset of DBpedia [9]; a subset of SemanticDB [32], a Semantic Web content repository for Clinical Research and Quality Reporting; and a subset of Lehigh University Benchmark (LUBM) [43], a benchmark for OWL knowledge base systems.

The reason for selecting these three datasets was that they represent different aspects of real-world semantic data. Thus, we tested the applicability of our approach in different types of datasets. SemanticDB is a domain-specific semantic data repository in Healthcare. It provides structured type information for the entities, which we utilized as the ground truth for automatic verification of the accuracy in the evaluations. Lehigh University Benchmark (LUBM) is a structured and well-known benchmark dataset, which has type information available. However, the entities can have multiple types. Unlike SemanticDB, LUBM data has hierarchical types. For instance, an entity can have both Student type and Graduate Student type. Therefore, we performed a manual verification process for the ground truth to ensure the accuracy of evaluations. Lastly, DBpedia is a commonly used general purpose dataset, which is a central source in the Linked Open Data Cloud [18]. Type information is not always present for entities in DBpedia. Moreover, some entities have several types, including hierarchical types, which makes it problematic for automatic verification of accuracy results. Therefore, we manually verified the accuracy of the ground truth in the evaluations. Table 1 demonstrates a sample of RDF triples from each dataset in the evaluations.

We ran the graph summarization algorithm in two stages. On the first stage, we utilized the core pairwise IRI graph node similarity algorithm without taking the descriptor importance and literal neighbor nodes similarity into account. On the second stage, the IRI nodes similarity is enhanced by the improvements of taking the literal neighbor similarity and the dynamic property weight assignment into account. The goal for running the evaluations in two stages is to investigate the impact of different factors on graph node similarity in real world datasets. We tested several parameters of the algorithm, including the maximum iteration, beta factor, class dissimilarity threshold, iteration convergence threshold, and the size of the dataset.

We also evaluated the performance of dynamic assignment of descriptor weights. Table 5.7 shows a sample of dynamically assigned descriptor weights from each dataset. As expected, the algorithm assigned higher weights to the properties with a higher degree of distinctiveness describing the resource type. For instance, in the LUBM dataset, the takesCourse property is more descriptive of the Student type than the name property, which is a common property for all class types in the dataset. Thus, takesCourse was assigned a weight of 44.1% as compared to the weight of 7.5% for name.

While the number of iterations depends on the dataset characteristics, we found that the similarity measures typically converge quickly after a few iterations. The algorithm stops the iteration once the rate of change in the similarity measures drops below the

Table 1: A Sample of RDF Triples from Each Dataset

Dataset	Subject	Predicate	Object
SemanticDB	SurgeryProcedure:236	hasCardiacValveAnatomyPathologyData	CardiacValveAnatomyPathologyData:70
SemanticDB	SurgeryProcedure:236	hasCardiacValveRepairProcedureData	CardiacValveRepairProcedureData:16
SemanticDB	SurgeryProcedure:236	SurgeryProcedureClass	"cardiac valve"
SemanticDB	SurgeryProcedure:236	CardiacValveEtiology	"other"
SemanticDB	SurgeryProcedure:236	CardiacValveEtiology	Event:184
SemanticDB	SurgeryProcedure:236	belongsToEvent	Event:184
SemanticDB	SurgeryProcedure:236	SurgeryProcedureDescription	"pulmonary valve repair"
SemanticDB	SurgeryProcedure:236	CardiacValveStatus	"native"
SemanticDB	SurgeryProcedure:104	hasCardiacValveAnatomyPathologyData	CardiacValveAnatomyPathologyData:35
SemanticDB	SurgeryProcedure:104	SurgeryProcedureClass	"cardiac valve"
SemanticDB	SurgeryProcedure:104	CardiacValveEtiology	"rheumatic"
SemanticDB	SurgeryProcedure:104	belongsToEvent	Event:81
SemanticDB	SurgeryProcedure:104	SurgeryProcedureDescription	"mitral valve repair"
SemanticDB	SurgeryProcedure:104	CardiacValveStatus	"native"
LUBM	Student49	telephone	"xxx-xxx-xxxx"
LUBM	Student49	memberOf	http://www.Department3.University0.edu
LUBM	Student49	takesCourse	Course32
LUBM	Student49	name	"UndergraduateStudent49"
LUBM	Student49	emailAddress	"Student49@Department3.University0.edu"
LUBM	Student49	type	UndergraduateStudent
LUBM	Student10	telephone	"xxx-xxx-xxxx"
LUBM	Student10	memberOf	http://www.Department3.University0.edu
LUBM	Student10	takesCourse	Course20
LUBM	Student10	name	"UndergraduateStudent10"
LUBM	Student10	emailAddress	"Student10@Department3.University0.edu"
LUBM	Student10	type	UndergraduateStudent
DBpedia	Allen_Ginsberg	wikiPageUsesTemplate	Template:Infobox_writer
DBpedia	Allen_Ginsberg	influenced	John_Lennon
DBpedia	Allen_Ginsberg	occupation	"Writer, poet"@en
DBpedia	Allen_Ginsberg	influences	Fyodor_Dostoyevsky
DBpedia	Allen_Ginsberg	deathPlace	"New York City, United States"@en
DBpedia	Allen_Ginsberg	deathDate	"1997-04-05"
DBpedia	Allen_Ginsberg	birthPlace	"Newark, New Jersey, United States"@en
DBpedia	Allen_Ginsberg	birthDate	"1926-06-03"
DBpedia	Allen_Ginsberg	deathPlace	"New York City, United States"@en
DBpedia	Albert_Camus	wikiPageUsesTemplate	Template:Infobox_philosopher
DBpedia	Albert_Camus	influenced	Orhan_Pamuk
DBpedia	Albert_Camus	influences	Friedrich_Nietzsche
DBpedia	Albert_Camus	schoolTradition	Absurdism
DBpedia	Albert_Camus	deathPlace	"Villeblevin, Yonne, Burgundy, France"@en
DBpedia	Albert_Camus	deathDate	"1960-01-04"
DBpedia	Albert_Camus	birthPlace	"Drean, El Taref, Algeria"@en
DBpedia	Albert_Camus	birthDate	"1913-11-07"

Table 2: An Excerpt from Dynamically Assigned Weights of Descriptors

Dataset	Node_Pair	Descriptor_Type	Descriptor	Weight
LUBM	(Student49,Student10)	Property	memberOf	14.7%
LUBM	(Student49,Student10)	Property	takesCourse	44.1%
LUBM	(Student49,Student10)	Property	emailAddress	14.0%
LUBM	(Student49,Student10)	Property	type	5.7%
LUBM	(Student49,Student10)	Property	name	7.5%
LUBM	(Student49,Student10)	Property	telephone	14.0%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“cardiac”	13.6%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“native”	15.2%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“other”	14.3%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“pulmonary”	22.8%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“repair”	17.2%
SemanticDB	(Procedure:236,Procedure:104)	Literal	“valve”	16.9%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	wikiPageUsesTemplate	2.2%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	influences	58.3%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	deathDate	2.2%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	birthDate	2.4%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	birthPlace	2.1%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	deathPlace	2.1%
DBpedia	(Allen_Ginsberg,Albert_Camus)	Property	influenced	30.7%

threshold, defined as the minimum value for the similarity iteration convergence, without having to iterate until maximum iteration. Hence, the maximum iteration and the convergence threshold values are kept relatively small as 10 and 0.001, respectively, in our experiments.

We observed that a higher class dissimilarity threshold results in more coarse classes, whereas the classes become more granular when the threshold is chosen smaller. The beta factor and the class dissimilarity threshold can be tuned differently in various datasets. Their optimum values depend on the characteristics of the datasets. In our evaluations, we found that the original value of 0.15 for the beta factor as introduced in [72] works well for our test datasets. For each dataset, we did not change the beta factor nor the class dissimilarity threshold in the two evaluations: the core similarity algorithm and similarity algorithm with the improvements. We found that a class dissimilarity threshold ranging between 0.3 to 0.6 in combination with a beta factor of 0.15 appeared to work well in our evaluations.

It is clear that the evaluations with the improvements of taking literal neighbor nodes similarity and auto-generated descriptor weights into account in IRI nodes similarity calculation generate a summary graph with better accuracy and stability, as demonstrated in Table 3. We noticed that the literal similarity improves the class generation accuracy in datasets that have frequently used terminology as in the case of SemanticDB and LUBM. On the other hand, it may have an adverse effect in datasets with lengthy and diverse vocabulary of literals, as in the example of DBpedia.

Figure 11 illustrates a small sample set of entities in the RDF graph from SemanticDB and their corresponding class types in the summary graph. As demonstrated in Figure

Table 3: Evaluation Results

Dataset	Algorithm	#Triples	Class_Threshold	#Iterations	Stability	Accuracy
SemanticDB	Core	6,450	0.5	4	61.0%	87.3%
SemanticDB	Improvements	6,450	0.5	4	68.2%	94.1%
LUBM	Core	6,484	0.3	3	67.8%	90.7%
LUBM	Improvements	6,484	0.3	3	78.4%	98.6%
DBpedia	Core	10,000	0.6	3	82.4%	92.8%
DBpedia	Improvements	10,000	0.6	3	89.1%	92.2%

11, the classes C-E1 and C-E2 represent the entities that are patient event types. They are classified in two different classes because when we compared them with the original dataset, we observed that the entities in C-E1 are more specifically patient surgery-related event types, while the entities in C-E2 are patient-encounter related event types. Also, the classes E-SP1 and E-SP2 are surgical procedure types. More specifically, the entities in E-SP1 are coronary artery and vascular related procedures, while the entities in E-SP2 are cardiac valve related-procedures. The classes C-VP and C-CAG represent the entities that are related to vascular procedures and coronary artery grafts, respectively. We implemented a basic algorithm to name the classes based on the class member IRIs. The classes C-E1, C-E2, C-SP1, C-SP2, C-VP and C-CAG are named as C-Event-1, C-Event-2, C-SurgicalProcedure-1, C-SurgicalProcedure-2, C-VascularProcedure and C-CoronaryArteryGraft, respectively.

The summary graph is generated along with the classes and the class relations with a stability measure for each relation. Figure 12 shows an excerpt from the summary graph representing the class relations from SemanticDB dataset. The percentage values besides the predicates are the stability (CPS) measure.

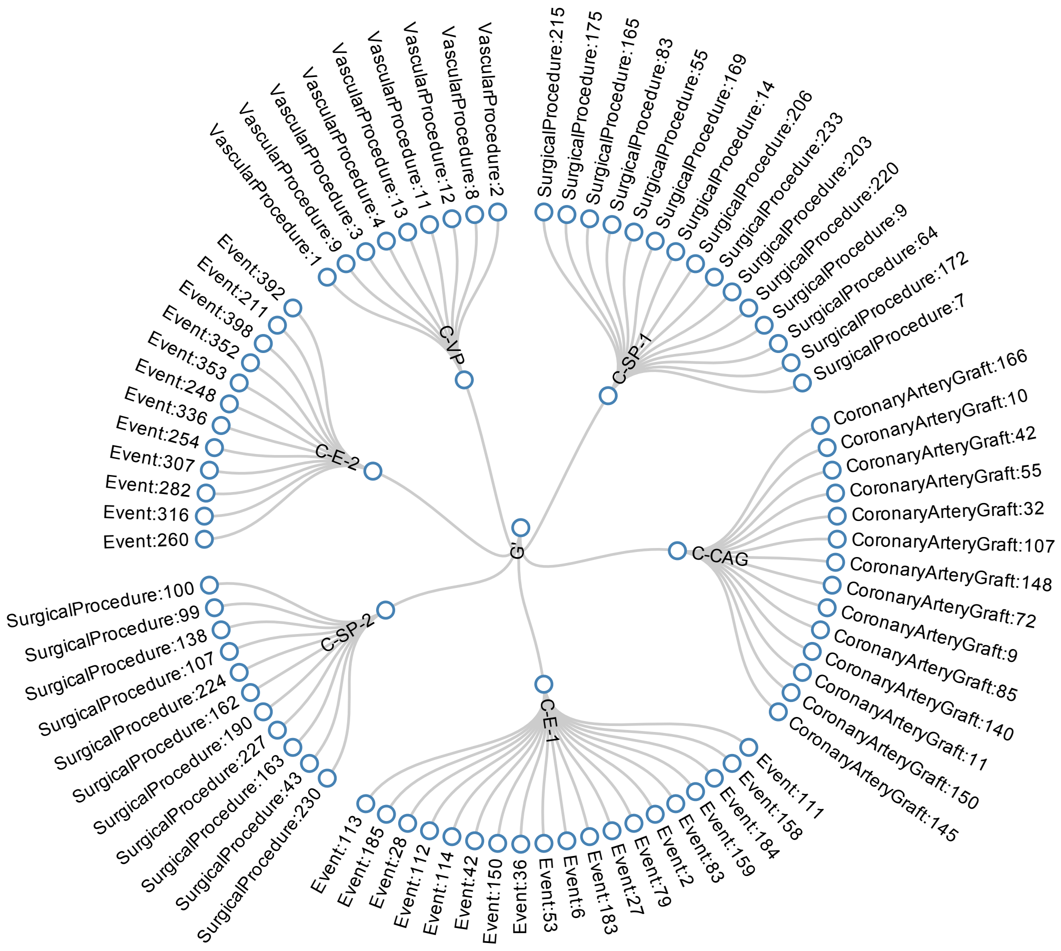


Figure 11: A figure consisting of different types of entities and elements belonging to the class types.

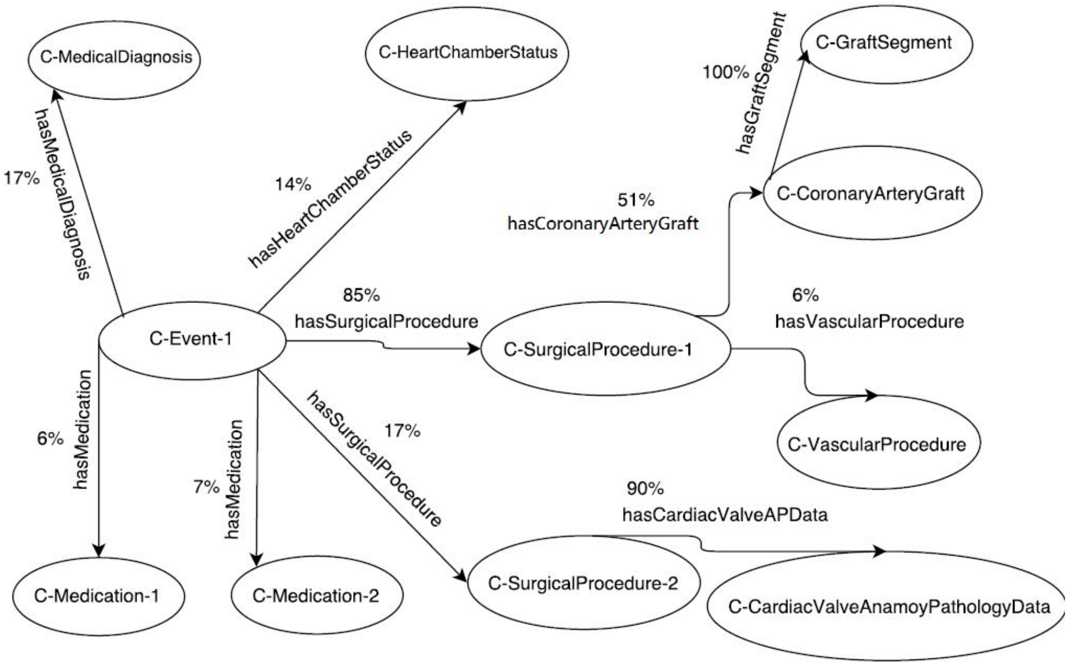


Figure 12: An excerpt from the generated summary graph.

5.8 Related Work

Clustering methods have been extensively studied, including KMeans [62], DBSCAN [38], and OPTICS [6]. There is a variety of techniques for finding clusters within data, and they are often successful in low-dimensional data. However, many of the existing clustering methods fail in high-dimensional data.

In [73] the authors defined the stability concept to be used in a coarsest partitioning problem. They utilized the stability concept on directed graphs. In our work, we leverage the stability concept to be used in a summary graph which is an RDF model.

5.9 Conclusion

In this chapter, we have focused on laying the foundations for graph summarization problems in RDF graphs. We conducted preliminary experiments that demonstrated significant results identifying the classes of entities. We introduced the Class Predicate Stability metric, which allows evaluation of the degree of confidence of each class predicate in the summary graph.

We note that the summary graph can further be utilized for automatic instance matching algorithms in the healthcare domain. In chapter 6, we present an instance matching technique in RDF graphs, and we clarify how the summary graph introduced in the current chapter can be exploited for instance matching in RDF graphs.

CHAPTER 6

RinsMatch: a suggestion-based instance matching system in RDF Graphs

In this chapter, we present RinsMatch (RDF Instance Match), a suggestion-based instance matching tool for RDF graphs. The main ideas and sections explained in this chapter are part of a published research study [11].

In chapter 3, we presented a method for the translation of data models from one format to another. The approach defined a number of steps required to perform the translation in section 3.1 to support heterogeneous data interoperability. In step 3 of the approach, the mapping and the translation rules between various schemata are manually defined a priori from a domain expert. However, the manual way of characterizing the mappings may cause time-consuming and costly work in the process due to the size of the healthcare data. The RDF instance matching system introduced in the current chapter contains a semi-automatic instance matching framework to help experts find linkages between different data elements.

RinsMatch utilizes a graph node similarity algorithm and returns to the user the subject node pairs that have similarities higher than a defined threshold. If the user approves the matching of a node pair, the nodes are merged. Then more instance matching candidate pairs are generated and presented to the user based on the common predicates and neighbors of the already matched nodes. RinsMatch then reruns the similarity algorithm with the merged RDF node pairs. This process continues until there is no more feedback from the user and the similarity algorithm suggests no new matching candidate pairs.

Each time the similarity algorithm runs, it produces more accurate results assuming the user provided accurate feedback. Merging the RDF nodes reduces the size of the input data graph that the algorithm operates on, yielding less complexity each time.

6.1 Contribution and Outline

In this study, we investigate the problem of discovering the linkages between semantically related entities that could be classified as the same entity within and among different data sources, so that multiple entities can be represented by one entity. In the literature, this problem has been studied by the research community as the task of instance matching or concept matching. In this context, we consider this problem an instance matching of RDF entities, as we represent the healthcare data instances and data model details in an RDF model. Our approach is semi-automatic: it utilizes the pairwise graph node similarity computation introduced in chapter 4 for finding similar entities and the mapping interface introduced in section 3.4 for presenting the similar entities to the domain experts.

Our main contributions:

- We present a suggestion-based semi-automatic instance matching system utilized for the RDF representation of data that contributes in the information translation process in the healthcare domain.
- We use neighborhood ideas to infer possible connections between the entities, such that if two entities are matched, then other nodes with similar predicates and in similar neighborhoods are considered.

- We merge the matched entities and run the process in iterations, with each iteration producing more accurate results and yielding less complexity.

The chapter is organized as follows. We discuss the application of instance matching in an RDF representation of healthcare data elements. Subsequently, we review the computation of entity similarity that is used for the matching. Then, we describe the user interaction for semi-automatic instance matching process. The subsequent section presents the results of the evaluations. Finally, we review the related work and follow this with our conclusion.

6.2 Instance Matching in Healthcare Data

Healthcare data exist in heterogeneous data sources with different formats and data models. In this work, we represent the data elements in RDF: i.e., the instance data, the data dictionary elements that belong to the instance data, and the mapping between them are represented in RDF. For the data sources that do not have a pre-defined data dictionary in place, we exploit the summary graph to extract the data dictionary elements as explained in section 5.6. We link the main data dictionary classes to the instance data elements with the rdf:type [24] predicate. The RDF relations allow linking the data elements from different systems, constituting an extensive and connected RDF graph.

In a healthcare data ecosystem, it is common to have redundant instances and concepts between different data sources. Thus, the instance matching technique explained in this chapter covers both de-duplication and data concepts matching. Merging the redundant nodes helps to reduce the size of the dataset. Also, linking the concepts between diverse data models assists in the information translation process and semantic data interoperability of different systems.

6.3 RDF Entity Similarity for Instance Matching

In chapter 4, we proposed an algorithm for computation of pairwise graph node similarity using graph locality, neighborhood similarity, and the Jaccard measure. The predicates of the subject nodes are treated as the dimensions of the entities when measuring the similarity of two subject entities. The direct similarities of the entities are taken into account along with the similarities of the neighbors with which they interact. The accuracy of the similarity metric is enhanced by incorporating the auto-generated importance weights of the predicates and the string literals referenced by the subject RDF entities. The algorithm runs in multiple iterations until the pair similarity results converge. In the current study we use the proposed RDF entities similarity algorithm for pairing entities which may be merged if approved by the user.

6.4 User Interaction

The size of data in healthcare seems to require the automation of instance matching techniques. Nonetheless, fully automated techniques can be error prone. Therefore, a semi-automated instance matching technique with user interactions yields more accurate results. Since our goals include matching the instances belonging to the heterogeneous data sources, RinsMatch, introduced in the current chapter, allows the user to provide initial matches between the source and target graph elements. RinsMatch then runs the entity similarity algorithm introduced in chapter 4. After the entity similarities converge, it follows the steps below:

- RinsMatch presents the subject node pairs that have similarities higher than a defined threshold to the user for possible instance matching. The threshold is a configurable parameter and may be determined by the user.
- If $s1$ and $s2$ are two subject nodes which have similarity higher than the threshold, then we denote this pair by ($s1$,$s2$). If the user approves the matching of the subject node pair ($s1$,$s2$), then RinsMatch merges the two subjects into a single subject node which we denote by [$s1$,$s2$], and then all the predicates from both merged subjects are retained by the newly created subject [$s1$,$s2$].
- It then checks the common neighbors and predicates of $s1$ and $s2$ and generates more instance matching candidate pairs by pairing the predicates $p1$ and $p2$ to get ($p1$,$p2$) if $p1$ and $p2$ are connected to a common object from both $s1$ and $s2$.
- It also pairs the object nodes ($o1$, $o2$) which are connected with a common predicate from $s1$ and $s2$.
- RinsMatch then presents the generated matching candidate pairs to the user and merges the pairs to get [$p1$,$p2$] and [$o1$,$o2$] if the user approves that they match.
- RinsMatch then reruns the RDF node similarity algorithm on the new RDF graph formed by merging matching entities.

The steps above are repeated until there is no more feedback from the user and no new matching pairs suggested by the matching algorithm.

Figure 13 shows an example of the instance matching and merging process. As shown in the figure, based on the similarities found from the similarity iterations, at phase 1

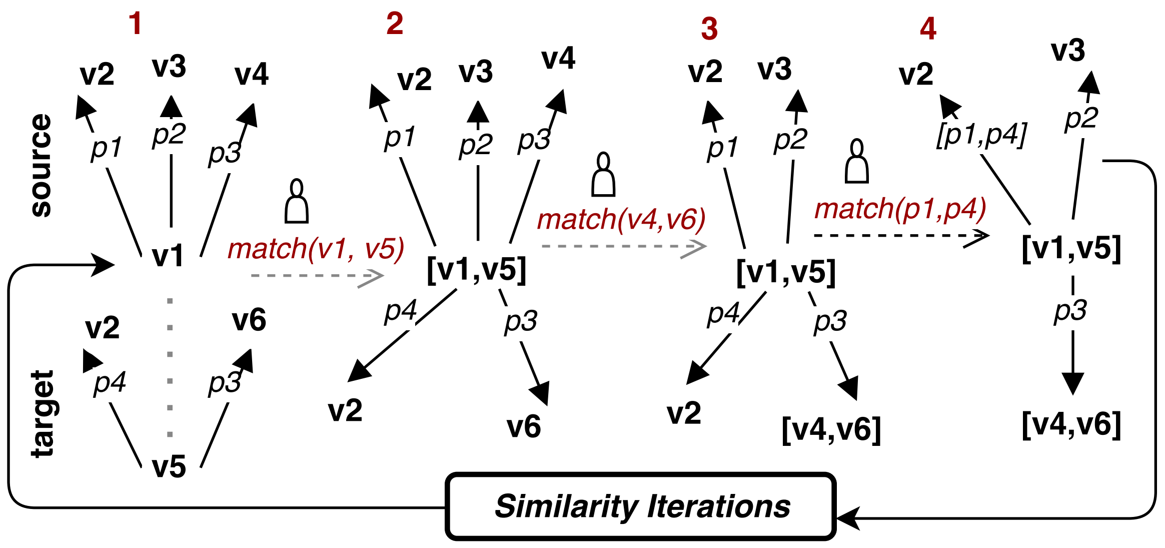


Figure 13: Instance matching process

RinsMatch suggests matching the subject nodes ($v1$, $v5$), and they are merged to get [$v1$,$v5$] with the approval of the user. On phase 2, the algorithm checks the common predicates of the new node [$v1$,$v5$]. Seeing that it connects to the neighbor nodes $v4$ and $v6$ with the common predicate $p3$, RinsMatch merges the nodes $v4$ and $v6$ to get [$v4$,$v6$] once the user approves. On phase 3, the common neighbors of the new node [$v1$,$v5$] are checked. Seeing that [$v1$,$v5$] is connected to a common neighbor $v2$ with the predicates $p1$ and $p4$, RinsMatch presents the pair ($p1$,$p4$) to the user, and they are merged upon approval by the user to get [$p1$,$p4$]. At phase 4, the size of the graph is reduced from five triples to three triples, a 40% reduction in size. The output graph of phase 3 is input to phase 1, and the similarity iterations are repeated until the optimum similarities and instance matching pairs are found.

6.5 Evaluation

We conducted preliminary experiments based on a subset of DBpedia [9] and a subset of SemanticDB [32], a Semantic Web content repository for Clinical Research and Quality Reporting. For verification, we duplicated the original dataset and changed the names of the nodes in the duplicated dataset by following a specific naming pattern. We used the original dataset as the source, and the duplicated dataset as the target for the instance matching process, and we leveraged the node naming pattern for verification. To summarize our experiments: for the DBpedia, the source dataset had 90 triples with 60 distinct subject and predicate nodes. 100% of the nodes were matched to a target graph node semi-automatically. The algorithm generated 20 instance matching candidates with 85% accuracy. For SemanticDB, the source dataset had 2500 triples with 520 distinct subject and predicate nodes. 86% of the nodes were matched to a target graph node semi-automatically. The algorithm generated 310 instance matching candidates with 95% accuracy.

6.6 Related Work

In the literature, there has been much research in the subject of ontology mapping. The terminology used for defining the problem has varied such as matching, alignment, merging, articulation, fusion, integration, morphism, etc. Inherently, many tools and methods have emerged in the field.

Many of these approaches are solely based on the use of string similarity mechanisms for finding matching entities between two ontologies [81,82]. Approaches that have studied the ontology mapping problem from the schema matching perspective in the context of data integration have also been explored since schemata can be considered as ontologies with restricted relation types. These include [63, 67].

Some others have suggested asking the user for feedback, as in our work, to perform the mapping generation interactively by providing proper visualization to support the decision. We think this is a useful feature for generating high-quality links. However, in this work we minimize the need for user feedback to reduce the load on the users.

Doan et al. [20] provided a system, GLUE, which employs learning techniques to semi-automatically find mappings between two given ontologies. For each concept in one ontology, their framework uses a multi-learning strategy to predict a similar concept in the other using probabilistic definitions of several similarity measures. GLUE calculates a joint probability distribution to measure the overlap between two input sets. The authors offer two learners: a content learner and a name learner for learning information such as the word frequencies, instance names and value formats. The content learner uses Naive Bayesian learning, a text classification method, for the instance content whereas the name learner uses the full name instead of its content. Then they combine the predictions of the two learners and assign weights to them using a meta-learner. Additionally, they use a technique, relaxation labelling, which gives labels to nodes of a graph, based on a set of constraints. Similar to their system, we also propose a machine learning framework. In contrast to them, we don't employ active an learning technique, which relies on the quality of the training dataset.

Jain et al. offered a framework called BLOOMS [51] and later an improved version under BLOOMS+ [52]. They propose a metric to determine which classes to align between two ontologies and a technique for using contextual information to support the alignment process. However, they rely on existence of a human-generated upper ontology and the concept categorization in the form of a tree structure. Our approach does not rely on an upper ontology, as we think these assumptions are problematic since the quality of the mapping strictly depends on the categorization of the concept by humans, and any potential categorization issue in the upper ontology will have an impact on the whole context.

As broadly considered, there has been some related work in the area of instance matching and graph partitioning in the Semantic Web community. With a data partitioning perspective, [85] proposed developing an index structure based on a bisimulation technique. In this respect, some research studies have also investigated comparing instances based on the properties and roles in the area of ontology instance matching but they primarily focus on the ontology mapping [49, 90] or ontology population [25] tasks.

On the other hand, [78] studies an automatic instance matching problem in RDF graphs with the focus on property weights, where property weights yield precedence to properties that make the instances more unique. The similarity metric utilized in this work also employs auto-generated property weights, which is a similar notion to the term frequency-inverse document frequency(tf-idf) [61, 80]. Our work is also different in the sense that it is semi-automatic, allowing for user feedback.

A matching algorithm called similarity flooding (SF) is proposed in [68]. SF matches two directed and labeled graphs to produce a multi-mapping of analogous nodes. For the similarity computation, SF relies on the intuition that elements of two graphs are similar when their adjacent elements are similar, exploiting the neighboring structure of a concept map, the semantic meaning of the content of the graph node and the labels of the relations between the nodes. In SF, the similarity of the node pairs starts either with a string similarity between the content of the nodes or with a similarity of 1. It then propagates the initial similarity of any two nodes through the graphs. The algorithm runs in multiple iterations until the similarity values are converged, or until a pre-defined maximum number of iterations. SF also does not distinguish between a schema node and the instance data node. Filters are utilized to select the best mappings. The mappings are then manually reviewed. The accuracy of the algorithm is measured by the estimated human labor savings obtained utilizing the algorithm for the matching tasks. In our work, we make an assumption for similarity computation like that of SF, that the nodes connected to similar neighbors with similar predicates are similar. We also run our similarity algorithm in iterations, and we do not distinguish between the data model elements and the instance data elements like in SF. However by the end of the iterations, we suggest the similar node pairs to the user for matching, and we merge the triples of the approved matched nodes. Consequently, we suggest matching of the nodes and predicates based on the common predicates and neighbors of the already matched nodes, and we rerun the similarity iterations. In this sense, our technique requires more user interactions but the consequent similarity iterations produces more accurate results assuming the user provides accurate feedback.

6.7 Conclusion

In this chapter, we have focused on instance matching in RDF graphs. We conducted preliminary experiments that demonstrated significant results in matching similar graph elements. Our instance matching technique is semi-automatic and does not distinguish between the instance data elements and the data model elements. The system makes use of the similarity algorithm introduced in chapter 4, and it makes smart suggestions to the user by utilizing the neighborhood concept to infer possible connections between the graph entities for finding the linkages between different data elements. The linkages found can further be utilized in the translation framework introduced in chapter 3 to support healthcare data interoperability.

CHAPTER 7

Conclusion

This dissertation has presented a semantic framework for healthcare information interoperability. We discussed two main routes for healthcare information interoperability: adapting standards and/or translation between different standards. Given the fact that there are significant barriers in implementing standards in the healthcare domain and it is unrealistic to have one universal standard that fits all the use cases, we suggested that a route for translation between different data models is more practical, and it is conceivable to implement standards in information translation. In this regard, we suggested utilizing RDF to depict in a flexible form the healthcare data entities having diverse data models and to perform the translation between the RDF representation of the data elements.

We then gave background information on the Semantic Web and its related concepts regarding healthcare information interoperability. Then, we presented a framework for translation of RDF representation of healthcare instance data, based on structured mapping definitions between the data concepts belonging to different data models. The framework includes a metadata repository, a user-friendly interface and a translation engine. The metadata repository stores the data dictionaries, which include data model details for classes, concepts, concept values, and their associations to each other. The interface allows the data managers to view and manage the data dictionary elements and the mappings between them. The mapping details are also stored in the metadata repository in a mapping schema. The translation engine utilizes the data model details and the structured mapping definitions to translate the source RDF data with regards to the target data model.

The defined translation framework includes a number of steps required to perform the translation to support heterogeneous data interoperability. However, a manual approach may be time-consuming and costly due to the size of the healthcare data. Therefore, we introduced a pairwise graph node entity similarity metric and its algorithm in order to be utilized for the automation of numerous interoperability tasks, e.g., auto classification of a dataset and automatically discovering the mappings between different data models. The graph node similarity metric utilizes the Jaccard index with the common relations of the data entities and common string literal words referenced by the data entities and augmented with data entity neighbors similarity. We also presented an importance weighting metric for the properties and the string literals referenced from the RDF nodes, and the auto-generated importance weights are used by the similarity metric. The similarity algorithm utilizes the similarity metric and runs in iteratively until the node pair similarity values converge.

The translation framework presented in this work makes use of data dictionaries. A data dictionary may not always be available for some datasets, e.g., extracting the data dictionary from an RDF dataset is a challenging task if the RDF dataset is not tied to an ontology. Hence, we presented an automatic classification method, which we call summary graph generation, based on the pairwise graph node similarities, and we explained extracting the data dictionary elements from the auto-generated summary graph. Similar entities are classified in the same type class, and the relations between the type classes are created with regards to their members' relations in the summary graph.

The summary graph is beneficial for understanding the underlying structure of the graph representation of the healthcare data, and provides an intermediate index structure for faster computational processing as the entire input data does not need to be completely processed. We also introduced the Class Predicate Stability metric, which measures the degree of confidence of each class relation in the auto-generated summary graph.

The interface introduced in our translation framework lets the data manager define the mappings between various schemata. Nonetheless, manually defining the mappings may require significant amounts of time. Therefore, we presented a suggestion-based semi-automatic instance matching system in RDF Graphs to help the data managers find linkages between different data elements. The presented instance matching technique does not distinguish between the instance data elements and the data model elements. It makes use of the pairwise graph node similarities, and it makes smart suggestions to the user by utilizing the common relations and neighbors to infer possible connections between the graph entities for finding the linkages between different data elements. The linkages are then stored in the metadata repository to further be utilized in the translation framework.

The current work lays a promising foundation for future research. To further improve the scalability, both the pairwise graph node similarity algorithm and the instance matching algorithm introduced in this work are good candidates for parallel computation. One could investigate how the algorithms can be applied to a distributed processing programming model such as Hadoop [92].

Looking further ahead, a translation-based healthcare information interoperability framework suggests several interesting questions: What are the practical methods of converting datasets from their native formats to RDF model and vice versa, for various kind of data formats? Can we query multiple databases to answer a question utilizing the structured mapping definitions? In the current work, the mappings are stored in the metadata repository within a mapping schema. Can we create a universal mapping ontology that defines the mappings and translation rules between different data models in the healthcare domain? How can we validate the results of a translation process? These research questions build upon what this dissertation has established: achieving information interoperability in healthcare domain is a formidable yet a significantly important task, and a translation based route for healthcare information interoperability is conceivable.

BIBLIOGRAPHY

[1] Etl (extract-transform-load) — data integration info. [Online; accessed 26-July-2015].

[2] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE, 1990.

[3] GeoNames, June 2015. http://www.geonames.org/.

[4] 3M. 3m health information systems. [Online; accessed 26-July-2015].

[5] Ben Adida, Mark Birbeck, Shane McCarron, and Steven Pemberton. RDFa in XHTML: Syntax and processing. *Recommendation, W3C*, 2008.

[6] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999.

[7] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. Simrank++: Query rewriting through link analysis of the click graph. *Proceedings of the VLDB Endowment*, 1(1):408–421, 2008.

[8] Grigoris Antoniou and Frank Van Harmelen. Web ontology language: Owl. In *Handbook on ontologies*, pages 67–92. Springer, 2004.

[9] Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[10] Mehmet Aydar, Serkan Ayvaz, and Austin Melton. Automatic weight generation and class predicate stability in rdf summary graphs. In *Workshop on Intelligent Exploration of Semantic Data (IESD2015), co-located with ISWC2015*, 2015.

[11] Mehmet Aydar and Austin C Melton. Rinsmatch: a suggestion-based instance matching system in rdf graphs. In *Workshop on Ontology Matching (OM-2015), co-located with ISWC2015*, 2015.

[12] Mehmet Aydar and Austin C Melton. Translation of instance data using rdf and structured mapping definitions. In *14th International Semantic Web Conference ISWC 2015*, 2015.

[13] Serkan Ayvaz, Mehmet Aydar, and Austin Melton. Building summary graphs of rdf data in semantic web. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 2, pages 686–691. IEEE, 2015.

[14] BBC. Bbc - languages of the world - interesting facts about languages. [Online; accessed 26-July-2015].

[15] Tim Berners-Lee. Linked data, 2006, 2006.

[16] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifier (uri): Generic syntax. Technical report, 2004.

[17] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[18] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.

[19] David Blumenthal and Marilyn Tavenner. The meaningful use regulation for electronic health records. *New England Journal of Medicine*, 363(6):501–504, 2010. PMID: 20647183.

[20] H BOHRING and S AUER. Doan a.-h., madhavan j., domingos p. & halevy a., ontology matching: a machine learning approach. *Handbook of ontologies, STAAB S. & STUDER R.(Eds.), International handbooks on information systems, Springer Verlag, Berlin*, pages 385–404, 2004.

[21] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

[22] David Booth, Conor Dowling, Dumontier Michel, Josh Mandel, Claude Nanjo, and Rafael Richards. The yosemite project. a roadmap for healthcare information interoperability. Semantic Technology and Business Conference. San Jose, California USA, 2014.

[23] David Booth et al. Yosemite manifesto on rdf as a universal healthcare exchange language, 2013.

[24] Dan Brickley and R. V. Guha. RDF Schema 1.1. W3c Recommendation, February 2014.

[25] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Davide Lorusso. Instance matching for ontology population. In *SEBD*, pages 121–132, 2008.

[26] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and C Quix. H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *SWDB*, pages 231–250. Citeseer, 2003.

[27] Artem Chebotko. Literature survey on the semantic web and search.

[28] James Clark et al. Xsl transformations (xslt). *World Wide Web Consortium (W3C). URL http://www. w3. org/TR/xslt*, 1999.

[29] Keith Cortis, Simon Scerri, Ismael Rivera, and Siegfried Handschuh. Discovering semantic equivalence of people behind online profiles. In *In Proceedings of the Resource Discovery (RED) Workshop, ser. ESWC*, 2012.

[30] John Currier. Schemaspy: Graphical database schema metadata browser. *Source Forge, Aug*, 2005.

[31] Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3c Recommendation, February 2014.

[32] Christopher D Pierce, David Booth, Chimezie Ogbuji, Chris Deaton, Eugene Blackstone, and Doug Lenat. Semanticdb: A semantic web infrastructure for clinical research and quality reporting. *Current Bioinformatics*, 7(3):267–277, 2012.

[33] Mariana Damova, Atanas Kiryakov, Kiril Simov, and Svetoslav Petrov. Mapping the central lod ontologies to proton upper-level ontology. *Ontology Matching*, page 61, 2010.

[34] Chris J Date. *Relational Database: Selected Writtings*, volume 1. Addison-Wesley, 1986.

[35] Jérôme David, Fabrice Guillet, and Henri Briand. Matching directories and owl ontologies with aroma. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 830–831. ACM, 2006.

[36] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 145–156. ACM, 2011.

[37] Marc Ehrig and Steffen Staab. Qom–quick ontology mapping. In *The Semantic Web–ISWC 2004*, pages 683–697. Springer, 2004.

[38] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[39] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 333. Springer, 2007.

[40] Centers for Disease Control and Prevention. Icd - icd-9 - international classification of diseases, ninth revision. [Online; accessed 26-July-2015].

[41] Centers for Medicare Medicaid Services. Meaningful use stage 1 requirements overview. [Online; accessed 26-July-2015].

[42] Tom Gruber. Ontology. *Encyclopedia of database systems*, pages 1963–1965, 2009.

[43] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.

[44] West Health. The value of medical device interoperability: — west health. [Online; accessed 26-July-2015].

[45] HealthIT.gov. What is an electronic health record (ehr)? [Online; accessed 26-July-2015].

[46] Nancy Ide and James Pustejovsky. What does interoperability mean, anyway? toward an operational definition of interoperability for language technology. In *Proceedings of the Second International Conference on Global Interoperability for Language Resources. Hong Kong, China*, 2010.

[47] Healthcare Information and Management Systems Society. What is interoperability? [Online; accessed 26-July-2015].

[48] N Ingebrigtsen. The differences between data, information and knowledge. *Diperoleh pada*, 26, 2007.

[49] Antoine Isaac, Lourens Van Der Meij, Stefan Schlobach, and Shenghui Wang. *An empirical study of instance-based ontology matching.* Springer, 2007.

[50] Anil K Jain and Richard C Dubes. *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[51] Prateek Jain, Pascal Hitzler, Amit P Sheth, Kunal Verma, and Peter Z Yeh. Ontology alignment for linked open data. In *The Semantic Web–ISWC 2010*, pages 402–417. Springer, 2010.

[52] Prateek Jain, Peter Z Yeh, Kunal Verma, Reymonrod G Vasquez, Mariana Damova, Pascal Hitzler, and Amit P Sheth. Contextual ontology alignment of lod with an upper ontology: A case study with proton. In *The Semantic Web: Research and Applications*, pages 80–92. Springer, 2011.

[53] Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[54] Ruoming Jin, Victor E Lee, and Hui Hong. Axiomatic ranking of network role similarity. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 922–930. ACM, 2011.

[55] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The knowledge engineering review*, 18(01):1–31, 2003.

[56] Rohit Khare and Tantek elik. Microformats: a pragmatic path to the semantic web. In *Proceedings of the 15th international conference on World Wide Web*, pages 865–866. ACM, 2006.

[57] Graham Klyne and Jeremy J Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[58] Lee Lau, Joni Endo, Steve Karren, Michelle Willis, Susan Harada, Shane Beeney, Brian Larsen, Edward Cassin, Martha Gerard, et al. Mapping clinical data with a health data dictionary, January 5 2001. US Patent App. 09/755,966.

[59] Zhenjiang Lin, Michael R Lyu, and Irwin King. Pagesim: a novel link-based measure of web page aimilarity. In *Proceedings of the 15th international conference on World Wide Web*, pages 1019–1020. ACM, 2006.

[60] Zhenjiang Lin, Michael R Lyu, and Irwin King. Matchsim: a novel neighbor-based similarity measure with maximum neighborhood matching. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1613–1616. ACM, 2009.

[61] Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.

[62] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.

[63] Jayant Madhavan, Philip A Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *VLDB*, volume 1, pages 49–58, 2001.

[64] Janet Marchibroda. Health policy brief: Interoperability. *Health Affairs*, 2014.

[65] Susan A Matney, Richard L Bradshaw, Oren E Livne, Bruce E Bray, L Frey, JA Mitchell, and SP Narus. Developing a semantic framework for clinical and translational research. *AMIA Summit on Translational Bioinformatics*, pages 7–9, 2011.

[66] Deborah L McGuinness, Richard Fikes, James Rice, and Steve Wilder. The chimaera ontology environment. *AAAI/IAAI*, 2000:1123–1124, 2000.

[67] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[68] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[69] Natalya F Noy and Mark A Musen. Anchor-prompt: Using non-local context for semantic matching. In *Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial intelligence (IJCAI)*, pages 63–70, 2001.

[70] Natalya Fridman Noy and Mark A Musen. Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831*, 2000.

[71] U.S. National Library of Medicine. Snomed clinical terms (snomed ct). [Online; accessed 26-July-2015].

[72] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. 1999.

[73] Robert Paige and Robert E Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[74] Rahul Parundekar, Craig A Knoblock, and José Luis Ambite. Linking and building ontologies of linked data. In *The Semantic Web–ISWC 2010*, pages 598–614. Springer, 2010.

[75] Rahul Parundekar, Craig A Knoblock, and José Luis Ambite. Discovering concept coverings in ontologies of linked data sources. In *The Semantic Web–ISWC 2012*, pages 427–443. Springer, 2012.

[76] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[77] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1983.

[78] Md Hanif Seddiqui, Rudra Pratap Deb Nath, and Masaki Aono. An efficient metric of automatic weight generation for properties in instance matching technique. *International Journal of Web & Semantic Technology*, 6(1):1, 2015.

[79] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for information Science*, 24(4):265–269, 1973.

[80] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[81] Dennis Spohr, Laura Hollink, and Philipp Cimiano. A machine learning approach to multilingual and cross-lingual ontology matching. In *The Semantic Web–ISWC 2011*, pages 665–680. Springer, 2011.

[82] Giorgos Stoilos, Giorgos Stamou, and Stefanos Kollias. A string metric for ontology alignment. In *The Semantic Web–ISWC 2005*, pages 624–637. Springer, 2005.

[83] Xiaomeng Su. A text categorization perspective for ontology mapping. *Norway: Department of Computer and Information Science, Norwegian University of Science and Technology*, 2002.

[84] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB11*, 2011.

[85] Thanh Tran and Günter Ladwig. Structure index for rdf data. 2010.

[86] Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE, 2009.

[87] The World Wide Web Consortium (W3C). Rdf 1.1 primer. [Online; accessed 26-July-2015].

[88] The World Wide Web Consortium (W3C). Standards - w3c. [Online; accessed 26-July-2015].

[89] The World Wide Web Consortium (W3C). Url. [Online; accessed 26-July-2015].

[90] Chao Wang, Jie Lu, and Guangquan Zhang. Integration of ontology data through learning instance matching. In *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pages 536–539, Dec 2006.

[91] Chao Wang, Jie Lu, and Guangquan Zhang. Integration of ontology data through learning instance matching. In *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pages 536–539. IEEE, 2006.

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

[93] Wikipedia. Data transformation, 2014. [Online; accessed 28-July-2015].

[94] Wikipedia. Cohort (statistics), 2015. [Online; accessed 26-July-2015].

[95] Wikipedia. Data dictionary, 2015. [Online; accessed 3-July-2015].

[96] Wikipedia. Data mapping, 2015. [Online; accessed 28-July-2015].

[97] Wikipedia. Morphism, 2015. [Online; accessed 21-September-2015].

[98] Wikipedia. Pagerank - damping factor, 2015. [Online; accessed 13-September-2015].

[99] Wikipedia. Semantic web, 2015. [Online; accessed 28-July-2015].

[100] Wikipedia. Upper ontology, 2015. [Online; accessed 10-September-2015].

[101] Wikipedia. World wide web, 2015. [Online; accessed 26-July-2015].

[102] Dell Zhang and Wee Sun Lee. Web taxonomy integration using support vector machines. In *Proceedings of the 13th international conference on World Wide Web*, pages 472–481. ACM, 2004.