ATTRIBUTE LATTICE: A GRAPH-BASED CONCEPTUAL MODELING GRAMMAR FOR HETEROGENEOUS DATA

by © Mojtaba Asgari

A Dissertation submitted

to the School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Faculty of Business Administration

Memorial University of Newfoundland

May 2020

St. John's, Newfoundland and Labrador

ABSTRACT

One key characteristic of big data is variety. With massive and growing amounts of data existing in independent and heterogeneous (structured and unstructured) sources, assigning consistent and interoperable data semantics, which is essential for meaningful use of data, is an increasingly important challenge. I argue, conceptual models, in contrast to their traditional roles in the Information System development, can be used to represent data semantics as perceived by the user of data. In this thesis, I use principles from philosophical ontology, human cognition (i.e., classification theory), and graph theory to offer a theory-based conceptual modeling grammar for this purpose. This grammar reflects data from users of data perspective and independent from data source schema. I formally define the concept of ***attribute lattice*** as *a graph-based, schema-free conceptual modeling grammar that represents attributes of instances in the domain of interest and precedence relations among them*. Each node in an attribute lattice represents an attribute - a true statement (predicate) about some instances in the domain. Each directed arc represents a precedence relation indicating that possessing one attribute implies possessing another attribute.

In this thesis, based on the premise that inherent classification is a barrier that hinders semantic interoperation of heterogeneous data sources, a human cognition based conceptual modeling grammar is introduced as an effective way to resolve semantic heterogeneity. This grammar represents the precedence relationship among attributes as perceived by human user and provides a mechanism to infer classes based on the pattern of precedences. Hence, a key contribution of attribute lattice is ***semantic relativism*** – that is, the classification in this grammar relies on the

pattern of precedence relationship among attributes rather than fixed classes. This modeling grammar uses the ***immediate and semantic neighbourhoods*** of an attribute to designate an attribute as either a category, a class or a property and to specify the expansion of an attribute – attributes which are semantically equal to the given attribute. The introduced conceptual modeling grammar is implemented as an artifact to store and manage attribute lattices, to graphically represent them, and integrate lattices from various heterogeneous sources.

With the ever-increasing amount of unstructured data (mostly text data) from various data sources such as social media, integrating text data with other data sources has gained considerable attention. This massive amount of data, however, makes finding the data relevant to a topic of interest a new challenge. I argue that the attribute lattice provides a robust semantic foundation to address this information retrieval challenge from unstructured data sources. Hence, a topic modeling approach based on the attribute lattice is proposed for Twitter. This topic model conceptualizes topic structure of tweets related to the domain of interest and enhances information retrieval by improving the semantic interpretability of hashtags.

Keywords: Attribute lattice, Conceptual modeling grammar, Semantic data integration, Attribute-lattice-based topic modeling, Twitter content analysis

ACKNOWLEDGMENTS

First and foremost, I want to thank my supervisor, Dr. Jeff Parsons. Jeff, you have supported me in so many important ways throughout my Ph.D. journey. I would not have been able to complete this journey without your persistent and kind support. I would also like to thank my committee members, Dr. Yair Wand and Dr. Joerge Evermann, for their valuable, and constructive feedback. I want to thank Dr. Jan Recker, Dr. Matti Rossi, and Dr. Sherrie Komiak for their helpful comments in the examination process.

I would also like to thank my amazing parents for their love and support. Thank you for being there for me through all the ups and downs.

Finally, I would like to dedicate this dissertation to my extraordinary wife, Hani, whose love, patience and support is immeasurable, and to our amazing son Ryan, whom I love dearly.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

1 Introduction

Conceptual models formally describe "*some aspects of the physical or social world around us for the purposes of understanding and communication*" (e.g., Mylopoulos, 1992). Conceptual models are supposed to represent the information in the domain of interest in a direct and natural manner (Mylopoulos, 1998). Traditionally, conceptual models are an early and essential part of requirements engineering for information system development (Wand et al., 1995; Mylopoulos, 1998; Olivé, 2007). However, with the explosion of available data, often created without any schema, the traditional paradigm of "model first, data after" is breaking down (Roussopoulos & Karagiannis, 2009; Lukyanenko & Parsons, 2013).

Traditional conceptual modeling grammars commonly have two assumptions, which I call *schema dependency* assumptions. First, they assume the subject domain that they represent consists of classes (entities) and instances belong to these classes (entities). This assumption is reflected in various conceptual modeling languages. For instance, Chen (1976) argues that *"[t]he entity-relationship model adopts the more natural view that the real world consists of entities and relationships*" , while Olivé (2007, p. 383) emphasizes that *"[o]ne principle of conceptual modeling is that domain objects are instances of entity types."* However, this assumption has been criticized for not offering a natural representation of the real world around us (Parsons & Wand, 2000).

Second, data is collected, accessed and used only for predefined purposes by known users who have a shared understanding of classes in the schema (Parsons & Wand, 2014).

However, with the rapid advance in Internet technology, more and more collected data is used for emerging purposes. For instance, social media data, which is collected to capture online interactions between people, can also be used to support decision making in organizations (LaValle et al., 2011). Data from external sources – outside of organization boundaries – either has no schema (or unknown schema), or has a schema that is not designed for the current particular emergent purpose.

With the unprecedented growth of data, challenges include not only efficient collection and storage of data, but also its meaningful use (Bizer et al., 2012, p. 51). Central to meaningful use is representing data in a semantically clear and interpretable manner, and, when combining data from multiple sources, providing a unified semantic view over data from independent and heterogeneous sources (independent from their logical data model). Nearly all existing conceptual modeling grammars assume the goal of creating a domain-knowledge-based, predefined schema (Lukyanenko et al., 2019) and provide modeling constructs consistent with this assumption. This dependency on predetermined classes hinders the meaningful use of data (Parsons & Wand, 2000, 2003; Lukyanenko et al., 2019). To address the limitations of schema-based conceptual modeling grammars, Lukyanenko et al. (2019) call for research on instance-based conceptual modeling grammars.

This thesis introduces a conceptual modeling grammar (Wand & Weber, 2002) that captures the semantics of data independent of a fixed, class-based schema and provides a foundation for representing and combining data from independent and heterogeneous

sources. This schema-free conceptual modeling grammar, which I call ***attribute lattice*** grammar[1], represents data semantics of the subject domain in a graph like structure.

Traditionally, data semantics has been defined as the "meaning and the use of data" (Woods, 1975). The information system community adopts this definition and describes semantics as a mapping between objects modeled, represented, and stored in the information system, and the real-world objects they represent (Sheth, 1997). This grammar aims to help data users understand data. Using cognitive principles (Parsons & Wand, 2008), the key to developing this grammar is mapping constructs (attributes) to the classes that are meaningful for data consumers. In other words, this grammar aims to provide data users with a data-consumer-oriented schema. In this context, semantics refers to mapping attributes in the subject domain to meaningful classes that data users need to understand and analyze data.

The notion of attribute lattice grammar is proposed in line with the instance-based data model (IBDM) (Parsons & Wand, 2000). The IBDM argues that instances (things) exist independent of classes, and classes are human-created constructs that provide useful abstractions (Parsons & Wand, 2000). The IBDM proposes a two-layered structure in which one layer is responsible for the (storage of) data about individual entities (instances) and their attributes, and the other keeps track of the definition of classes in terms of attributes of instances. In the IBDM approach, instances are stored only with their attributes, rather than classes (Parsons & Wand, 2000). By freeing data from predefined classes and

[1] In this thesis, hereafter, "*attribute lattice grammar*" refers to the conceptual modeling grammar, and an "*attribute lattice*" or a "*lattice*" refers to a model (script) generated from this grammar.

schemas and eliminating the need to map class-level constructs between independent schemas, the IBDM simplifies semantic interoperation.

1.1 Research Objectives

The attribute lattice grammar utilizes principles from human cognition and philosophical ontology to offer a theory-based, lightweight conceptual model - that is, a conceptual model with a minimal set of components (attributes and the relationships among them) to capture the semantics of the domain. This conceptual modeling grammar aims to offer a form of representation that reflects users of the data point of view, independent of the schema of the data source (schema-free).

The first research objective is to define components of this conceptual modeling grammar, which is independent of fixed classes but supports classification. This research objective also aims to discuss how this grammar provides a mechanism to infer unobserved attributes based on observed ones, to encapsulate attributes that are common to all members of a class, and to assign new attributes to all members of a class (Parsons & Wand, 2008).

The grammar provides a basis to create a unified view over heterogeneous data sources. Hence, another goal of this research objective is to define the notion of "similarity" (Evermann, 2008a) in this grammar, and elaborate on how this grammar enables semantic data integration. Finally, this research objective aims to compare this grammar with other knowledge representation languages (i.e., DL). The first research objective of this thesis, therefore, is:

***Research Objective 1*:** To formalize the notion of attribute lattice grammar by using principles from human cognition and philosophical ontology.

The next research objective is to develop a software artifact that supports the attribute lattice conceptual modeling grammar. This artifact: (1) provides basic features such as the ability to store, query, and edit lattices; (2) provides a declarative graphical representation of lattices; (3) supports basic analyses such as class structure and attribute lattice validation; and (4) supports the integration of lattices from distinct data sources. Hence, the second research objective of this thesis is:

***Research Objective 2*:** To implement an IT artifact to support attribute lattice creation, manipulation, graphical representation, validation, and integration.

The research strategy that guides this study is Design Science Research (DSR). The DSR typically involves the creation and evaluation of IS artifacts (March & Smith, 1995; Hevner et al., 2004). In this context, the artifact is a conceptual modeling grammar (components, and rules), and a tool that supports the creation of this artifact. The last research objective of the thesis concerns the evidence-based evaluation of this artifact, as a crucial part in DSR (Hevner et al., 2004).

The evaluation of the artifact can be absolute or relative (to comparable artifacts, or to the absence of artifact) (Prat et al., 2014). The former evaluation techniques examine if the artifact achieves its goal, where the later techniques compare the artifact to the absence of artifact or to other comparable artifacts. The introduced grammar is an innovative approach to address representing data semantics coming from heterogeneous data sources. I propose an absolute evaluation approach to demonstrate how this grammar reaches its goal

of representing data (text data) semantics. In specific, I propose the use of attribute-lattice-based topic modeling for Twitter - the most popular micro-blogging site.

With the explosive amount of data, finding relevant information from data sources is not a trivial task. Specifically, the lack of schema in semi-structured and unstructured data sources (e.g., text data) presents new challenges for information retrieval. An attribute lattice grammar can be used to represent data semantics of structured, semi-structured, or unstructured data. This notion is being used, here, to summarize and to conceptualize the topic structure of text data (tweets) in the domain of interest. Hence, the third research objective of this thesis is:

***Research Objective 3*:** To demonstrate the practical usefulness of attribute lattice grammar.

Table 1 summarizes the research objectives of this thesis and specific objectives related to each research objective.

In the following, I begin by formally defining the notion of attribute lattice and discussing a procedure to create a unified attribute lattice (Chapter 2). Then, the implemented artifact is discussed (Chapter 3). This is followed by introducing topic modeling for Twitter (Chapter 4). The thesis concludes by summarizing the primary contributions of the research to theory and practice and suggesting several areas for future research (Chapter 5).

Table 1. Research Objectives and Specific Objectives

Chapter 2	***RO1***	Formalize the notion of attribute lattice by using principles from human cognition and philosophical ontology.
	SO 1.1	Define attribute lattice grammar components and characteristics
	SO 1.2	Elaborate attribute lattice validation rules
	SO 1.3	Compare the attribute lattice grammar with other knowledge representation languages
	SO 1.4	Develop a foundation for the semantic integration of data conceptualized with the attribute lattice grammar
Chapter 3	***RO2***	Implement an IT artifact to support attribute lattice creation, manipulation, graphical representation, validation, and integration.
Chapter 4	***RO3***	Demonstrate the practical usefulness of attribute lattice grammar.
	SO 3.1	Develop the procedure of lattice extraction for topic modeling in Twitter (attribute-lattice-based topic model.)
	SO 3.2	Extend the artifact to retrieve tweets and to suggest precedences based on them for attribute-lattice-based topic modeling.
	SO 3.3	Demonstrate the practical applications of attribute-lattice-based topic modeling.

2 Attribute Lattice: A Graph-Based Conceptual Modeling Grammar

2.1 Introduction

Traditionally, conceptual modeling has been considered a critical step of requirements engineering in Information System (IS) analysis and design (Wand & Weber, 2002). Previous studies have emphasized that conceptual models help IS stakeholders to understand and communicate relevant knowledge in a domain. Furthermore, conceptual models are a way to document the original IS development requirements, and they serve as input for the design process (Kung & Solvberg, 1986; Wand & Weber, 2002; Recker, 2015). Over the years, extensive work has been conducted on this topic. There exists a considerable body of literature on how these models are used to capture and represent both static phenomena (e.g., instances and their attributes) and dynamic phenomena (e.g., events and processes) in a domain. For instance, Moody (2005), Recker et al. (2009), and Van der Aalst (2013) provide a more comprehensive review on conceptual modeling approaches.

In the era of big data, with the rapidly growing amount of available data, the environment of IS development has changed. Parsons and Wand (2014) coined the open information environment (OIE) to explain the characteristics of this new environment. In this environment, *"users have access to sources over which they may have no control; new sources of data may emerge; applications of data might change radically over time; and new uses of data might emerge"* (Parsons & Wand, 2014. p. 2). In OIEs, support is needed to enable users to apply their own conceptual models to the information coming from various data sources (Parsons & Wand, 2014).

Based on the premise that dependency on the schema of data sources is a barrier to conceptual modeling for OIEs (Lukyanenko et al., 2019), I propose a schema-free (i.e., not requiring or producing requiring a fixed schema) conceptual modeling grammar, which I call ***attribute lattice grammar***. This grammar, independent from the original schema of data source, captures data semantics and represents the structure of data.

Conceptual models represent knowledge in a domain as understood by humans (Hirschheim et al., 1995; Wand et al., 1995). As has been previously shown in the literature, ontology and cognition are appropriate theoretical foundations to create, and evaluate conceptual modeling grammars and scripts (e.g., Shanks et al., 2003; Parsons & Wand, 2008; Burton-Jones et al., 2009; Recker et al., 2011). The notion of attribute lattice, as a theory-based conceptual modeling grammar, is developed using principles from philosophical ontology and cognitive psychology. The components of this grammar are defined by elaborating and differentiating various types of subsumption relationships. These types have been defined such that the introduced grammar supports the meaningful classification of data.

2.2 Related Research

In this section, I discuss principles from philosophical ontology and cognitive psychology that offer guidance in defining the attribute lattice conceptual modeling grammar. Then, I briefly present a review of approaches for resolving semantic heterogeneity through data integration to highlight a common assumption underlying many approaches – the reliance on class-based schemas – and to point out that this dependency, in turn, leads to several known challenges in these approaches. This is followed by elaborating the mathematical foundation (i.e., graph theory) used to define the attribute lattice as a graph-based grammar.

2.2.1 Principles from cognitive psychology and philosophical ontology

Ontology - the branch of philosophy which deals with the order and the structure of reality (Angeles, 1981; Wand et al., 1995) – has been used as a foundation for prescribing components of conceptual modeling grammars, as well as for analyzing and improving conceptual models (Guizzardi & Wagner, 2010). In particular, Bunge's ontology (Bunge, 1977) , as elaborated for conceptual modeling by Wand and Weber (1990, 1993), has been popular in conceptual modeling research. Three ontological principles, which are widely adopted in IS research, are central to our approach: (1) the world consists of, either tangible or intangible, things that are assumed to exist; (2) things possess attributes; and (3) subsumption relations between attributes can be expressed by attribute precedence (Wand & Weber, 1990, 1993; Parsons & Wand, 2002, 2003; Chen & Parsons, 2008; Parsons, 2011)

The concept of ***attribute precedence***[2] -i.e., subsumption relationships between attributes - provides an essential foundation for the definition of attribute lattice. As has been previously reported in the literature, this concept can be utilized to improve semantics captured and conveyed by conceptual models (Parsons & Wand, 2003; Parsons, 2011). To elaborate on this concept, suppose **r** and **s** are two attributes, **s** precedes **r** means that any instance possessing **r** also possesses **s** (Bunge, 1977; Parsons & Wand, 2000, 2003). For example, suppose **r** is *'is blue'* and **s** is *'has a color'*, every instance that *'is blue',* also *'has a color'*. Likewise, *'is visible'* precedes *'has a color'*, which in turn, precedes *'is blue'*.

Cognitive psychology provides a guideline for the definition of attribute lattice. The classical view of categories assumes classes independently exist and views them as abstract containers with things either inside or outside of them (Lakoff, 1987). Along the same lines, the inherent classification of data is a common assumption in conceptual modeling and database design (Parsons & Wand, 2000).

Inherent classification entails that (1) each class is defined by its properties, (2) all instances must belong to classes and (3) each instance of a class possesses the same set of properties. Traditionally, identifying classes is an initial step in conceptual modeling, and database design assumes (either explicitly or implicitly) that (1) instances are inherently classified and must belong to at least one class to exist in a database (Parsons & Wand, 2000) and (2) there is a clear and fundamental distinction between classes and properties of instances (i.e., instances belong to the classes and possess properties). As a result, the

[2] The term "attribute precedence" refers to "property precedence" using the terminology in Parsons and Wand (2000)

main body of research on resolving semantic heterogeneity focuses on schema mapping techniques, that is, the identification of similar schema elements in various data sources (Rahm & Bernstein, 2001; Noy, 2004; Dong & Srivastava, 2013).

Parsons and Wand (1997, 2000, 2008) criticized the inherent classification assumption in conceptual modeling and database design. They argued classification should be guided by cognitive principles. Humans use concepts to classify phenomena they encounter based on observable properties. In fact, without categorizing the world into concepts, humans cannot function at all (Lakoff, 1987). These concepts, which manifest as classes in conceptual modeling and database design, enable us to understand and communicate the phenomena of interest (Parsons & Wand, 2008)

It is commonly accepted that classification has two major functions. First, it promotes *cognitive economy*. A class abstracts all the relevant attributes (properties) of its instances. Hence, by classifying instances and assigning them to classes, humans decrease the amount of information that is needed to perceive, learn and communicate about each individual instance. Second, it supports *inference* – that is, it enables us to go beyond the information given. When humans come across an instance, based on their direct (observed) knowledge, they can infer the unobserved properties of instances (Smith & Medin, 1981; Smith, 1988; Parsons, 1996; Parsons & Wand, 2000, 2008).

Parsons and Wand (2008), utilizing these functions, offers a model of "*good*" classification structure in conceptual modeling, and provides a set of rules for constructing such structures. These rules provide a guideline to develop high-quality conceptual models in a

domain of interest, with meaningful classes, that better support and reflect users' perspectives. For instance, the good classification model emphasizes that a proper subset of a class must exist that the class membership can be inferred from this subset. As will be discussed in further detail in the attribute lattice component definition section (2.3.1), the model of good classification guides the process of attribute lattice component development.

2.2.2 Resolving Semantic Data Heterogeneity via Data Integration

Semantic data modeling and integration is an active research area in several research communities such as databases, domain ontologies and big data (Rahm & Bernstein, 2001; Noy, 2004; Dong & Srivastava, 2013). Despite its pervasiveness and the substantial work in this area, resolving semantic heterogeneity remains a key challenge in using data from independent sources. The lack of deep data understanding, and a focus on syntax and structure, rather than on data semantics, hinders semantic data integration (Uschold & Gruninger, 2004; Haas, 2007).

Semantic data integration is an approach for providing unified access to disparate and semantically heterogeneous data (Bergamaschi et al., 1999). The field has been an active area of research since the 1980s (Batini et al., 1986; Doan & Halevy, 2005). However, in spite of abundant literature, concerns have persisted about the lack of (1) consistent theory and methodology, (2) in-depth understanding of semantics, and (3) a unified approach for integration (Sheth, 1999; Uschold & Gruninger, 2004; Haas, 2007; Hendler, 2014).

The aim of this section is not to review all approaches for semantic integration in all disciplines. For example, Rahm (2011), and, Shvaiko and Euzenat (2005) provide a comprehensive review of integration approaches. Instead, I try to highlight the common assumptions underlying many approaches (the reliance on class-based schemas) and provide a general overview of the limitations arising from these assumptions.

- ***From Traditional to Domain Ontology-based Data Integration***

Traditional semantic data integration can be divided into two main steps (Rahm & Bernstein, 2001; Doan et al., 2004). The first step, a match operation, takes two schemas as input and provides a semantic mapping between schema elements. The second step defines mapping expressions formally. Depending on the context, the mapping can be expressed using different approaches such as LAV (local as view), or GAV (global as view). In these methods, the data reside in data sources, while the global schema provides a unified, integrated, and virtual view (Lenzerini, 2002).

Generally speaking, matcher types can be categorized into schema level and data (instance) level matchers (Rahm & Bernstein, 2001). As argued by Parsons and Wand (2000), in traditional data models, classification is inherently part of data management and storage. In this regard, schema reconciliation is a prerequisite to accessing data. Not surprisingly, then, the main body of semantic data integration literature focuses on schema integration and data integration based on a so-called global schema (or a mediated schema). Match methods in the data level are often used as a complementary method, or for semi-structured data when a schema cannot be constructed from data. These methods are either based on linguistic characteristics (for text elements) such as keyword relative frequency

and string match (e.g., Clifton et al., 1998), or constraint characteristics (for more structured data), such as value ranges and averages (Rahm & Bernstein, 2001). Probabilistic and statistical models are the key common approaches used in match methods in the data level (e.g., Doan et al., 2003; Kang & Naughton, 2003).

The initial approach for data integration was hard-coding the integration points. In this approach, developers were supposed to implement separate and specific code to get access to components of other schemas. Therefore, it had no flexibility, and it was hard to maintain. Although subsequent methods were loosely coupled and easier to manage, data semantics was a missing component in the integration process (Uschold & Gruninger, 2004). Domain ontology-based approaches were introduced to address this lack of semantics. Domain ontology has two primary roles to play in these methods (Wache et al., 2001): first, map concepts in the content to fixed classes (an ontology); and second, integrate these concepts from different ontologies.

Schemas and ontologies have different purposes: ontologies have been used for interoperability, search, and automated reasoning purposes, while, schemas have been used for structuring and querying data in a single database (for a detailed comparison see Uschold & Gruninger, 2004). However, a common practice in both is to utilize fixed classes for structuring the data. As a result, similar techniques were used for schema mapping and ontology mapping (Shvaiko & Euzenat, 2005). The ontology mapping techniques, like their ancestors (schema mapping), still suffer from a lack of deep (cognitive) semantics - that is, although ontology-based semantic data integration approaches reach the agreement about the semantics of data within the individual ontology, their ties to the schema and

fixed classes makes defining a shared interlingua ontology challenging (Uschold & Gruninger, 2004).

- ***Semantic Web and Linked Data***

The notion of Semantic Web, first coined by Tim Berners-Lee (Berners-Lee et al., 2001), has been used for semantically integrating semi-structured data on the web. To achieve this goal, Linked Data provides a set of best practices, and offers principles (Berners-Lee, 2006; Heath & Bizer, 2011) to publish and interlink machine-readable data on the web (Heath & Bizer, 2011). In brief, Linked Data uses URIs (Berners-Lee et al., 2005) to define uniquely identifiable web resources and RDF (Consortium, 2014) triples (subject, predicate, and object) to encode how these resources are related (Bizer et al., 2011).

As a semantic extension of the RDF data model, RDF schema (Brickley & Guha, 2014) provides a data model vocabulary (schema) for RDF-based data sets. It provides mechanisms to describe groups of resources in terms of classes and properties by using the RDF-based syntax (Brickley & Guha, 2014). RDF schema improves the capability of RDF data sources in important ways such as adding subsumption hierarchy to the classes and properties (Horrocks et al., 2003)

During the past two decades, multiple web ontology languages such as OIL (Fensel et al., 2001), DAML + OIL (Connolly et al., 2001), OWL (McGuinness & Van Harmelen, 2004), and OWL2 (Hitzler et al., 2009) have been introduced. The latest version of OWL, OWL2, is based on Description Logics (DL) - a family of class- and property-based (concept-based) knowledge representation languages and a subset of first-order logic (Baader, 2003). It became a W3C recommendation in 2009 – W3C recommendations promote the

interoperability of Web technologies. OWL2 provides a richer vocabulary for describing instances of RDF-based resources in comparison to RDFS, such as relations between classes, and characteristics of properties (e.g., symmetry). However, like other schema-based approaches, there is a clear fundamental distinction between class and property in this language. In the standard OWL2 semantics and reasoners, different usages of the same term (e.g., both as a class and a property) will be considered different.

- ***Known Issues in Schema-Based Approaches***

There are several well-known problems in schema-based approaches, which arise because of their inherent classification assumption. I argue these longstanding problems can be addressed by using attribute-lattice-based integration approach. First, in schema-based approaches, there is a clear distinction between the concept of class and the concept of property. With this assumption, it is not easy (if possible at all) to integrate a class from one data source to a property in another (Omelayenko, 2002; Ghidini & Serafini, 2006; Šváb-Zamazal & Svátek, 2009). For instance, *faculty member* or *graduate student* could be a property (or property value) of the *customer* class in one data source, and each can be separate classes in another source.

Second, concepts (either properties or classes) in distinct data sources may have various levels. This means a concept in one data source can be more general, or more specific than a related concept in another data source. For instance, *'person'* class in one data source is more general than *'student'* class in another. I refer to this as the "general/specific concepts integration" problem (e.g., Barrasa Rodríguez et al., 2004; Dragut & Lawrence, 2004; Lammari et al., 2007).

Finally, schema-based approaches are susceptible to the "complex matching problem", in which possessing several concepts (class or property) at the same time in one schema is semantically equal to possessing one concept (or several concepts) in the second schema (e.g., Barrasa Rodríguez et al., 2004; Dragut & Lawrence, 2004). For instance, *'is a Ph.D. candidate'* could be a property of '*Ph.D. students'* class in one data source, however, it might be semantically equivalent to and calculable from several properties of *'student'* class (*'is a Ph.D. student'*, *'has completed comprehensive exam'* and *'has a thesis topic'*) in another data source.

2.2.3 Mathematical Foundation

Mathematically, an attribute lattice is a set of attributes and a set of precedences that show the subsumption relationships among attributes. An attribute lattice can be represented in a graph-like structure. The mathematical structure of attributes and pairwise (precedence) relationships among them, however, can be modeled using graph theory. The mathematical structure helps, first, to represent attribute lattices where the graphical model is large, second, to make inferences based on known precedences, and finally, to develop an artifact for representing and manipulating an attribute lattice. The aim of the following brief review is to summarise digraph (directed graph) definitions needed for the mathematical definition of an attribute lattice. These definitions are adopted from Agnarsson and Greenlaw (2007).

A directed graph (Agnarsson & Greenlaw, 2007, p. 21) is an ordered triple $\vec{G} = (V, E, \eta)$, where

$V \neq \phi$
$V \cap E = \phi$
$\eta : E \mapsto V \times V$ *is a map.*

Here, ϕ denotes an empty set, V a set of vertices, E a set of directed edges (or just edges) and $\mapsto$ a '*maps to'* symbol. If $\eta(e) = (u, v)$, then u is called the tail of e and v the head of e. Also, v is called a successor of u, and u is called predecessor of v.

If $\eta(e) = (u, u)$, then e is called a directed loop. If $\vec{G}$ is a digraph, then $V(\vec{G})$ and $E(\vec{G})$ will always denote the set of vertices and directed edges of $\vec{G}$, respectively. Two directed edges e and $\acute{e}$ are said to be parallel edges if $\eta(e) = \eta(\acute{e})$. That is, the edges are mapped onto the same ordered pair of vertices.

Let $\vec{G} = (V, E, \eta)$ be a digraph, and $u \in V$ a vertex in $\vec{G}$, the indegree of u (Agnarsson & Greenlaw, 2007, p. 24) denoted by $d^-(u)$, is the number of directed edges having u as head, that is,

$d^-(u) = |\{e \in E : \eta(e) = (x, u) \text{ for some } x \in V\}|$

The outdegree of u, denoted by $d^+(u)$, is the number of directed edges having u , as tail, that is,

$d^+(u) = |\{e \in E : \eta(e) = (u, y) \textit{ for some } y \in V\}|$

For a given vertex $u \in V(\vec{G})$, the inneighbours, denoted by$N^-(u)$, and the outneighbours, denoted by$N^+(u)$, of u are given by the following:

$N^-(u) = \{x \in V(\vec{G}) : \eta(e) = (x, u) \textit{ for some } e \in E(\vec{G})\}$,
$N^+(u) = \{y \in V(\vec{G}) : \eta(e) = (u, y) \textit{ for some } e \in E(\vec{G})\}$,

respectively.

This section reviewed mathematical and theoretical foundations – graph theory, cognitive psychology and philosophical ontology – that will be used to develop a graph-based attribute lattice grammar. This section also pointed out a common assumption in semantic integration approaches, that is, schema dependency. Schema dependency, from traditional data integration approaches to domain ontology-based data integration approaches to the semantic web integration approaches, contributes to the known integration challenges. In the following, I introduce the notion of attribute lattice, and I discuss how this grammar offers a new approach for semantic data integration based on classes that are constructed from the point of view of users of data.

2.3 Attribute Lattice Grammar Components and Characteristics

In traditional schema-based notation, classes are concepts to which instances belong and properties are attributes of classes that are part of the class definition, and are possessed by instances in the class. In contract with this approach, I argue attributes are statements about instances, and whether a particular statement is considered a class or a property depends on the relationship between attributes[3].

Grounded in ***classification*** – d a human cognition process to understand the semantics of instances (Parsons, 1996; Parsons & Wand, 1997, 2008) – d and the concept of ***attribute precedence*** from philosophical ontology (Bunge, 1977; Parsons & Wand, 2003; Chen & Parsons, 2008; Parsons, 2011), this sections formally defines the concept of attribute lattice as a graph-based conceptual modeling grammar that represents the class structure of the subject domain from perspective of the users of data independent from a data source schema. This modeling grammar focuses on attributes (not classes nor properties) and the relationship among attributes. However, the class structure of the domain can be inferred from the pattern of precedences and attributes in the grammar.

A key contribution of an attribute lattice grammar is ***semantic relativism*** **–** that is, the pattern of precedences around an attribute designates the attribute as a class, or a property. A precedence relationship in an attribute lattice represents how a human (user of data) perceives the relationship between two attributes (true statements) in the domain. For any given attribute in a lattice, a set of precedences around it represent its relationship with

[3] Hereafter in this thesis, attribute refers to the node itself, and property denotes one type of node.

other attributes. In other words, changing the pattern of precedences around an attribute over time (i.e., adding/removing precedence relationships) may change the type of attribute. For instance, the type of attribute may change from a property to a class by adding new precedences.

2.3.1 Definition of Attribute Lattice and its Components

An attribute lattice is a representation of attributes linked to each other in a graph like structure. It can be shown in a graphical representation in which each node represents an attribute, and each directed arc represents a precedence relation between attributes indicating subsumption relations between attributes. Here, the grammar is defined formally, a mathematical notation for lattice structure presentation is introduced by expanding digraph (directed graph) notation, and examples are used to elaborate each definition.

Definition 1 (Attribute): An ***attribute*** refers to any true statement (predicate) describing instances[4].

Expr 1. $A = \{a_1, a_2, a_3, \dots, a_m\}$;

Definition 2 (Domain of interest): A ***domain of interest*** is a set of phenomena (instances), X, and a set of attributes, A, possessed by the instances in X. D = (X, A) such that each attribute in A is possessed by at least one instance in X.

Expr 2. $X = \{x_1, x_2, x_3, \dots, x_n\}$

$\forall a_i \in A, \exists x_j \in X \; such \; that \; a_i \in \mathcal{A}(x_j)$;

[4] The "attribute" in this notation is equal to the definition of ***property*** in Parsons and Wand (2008)

$\mathcal{A}$ is a function (map) from an instance to its attributes, that is, for any given instance (x_i), the output (A_i) of this function is a set of attributes possessed by this instance.

Expr 3. $\mathcal{A} : X \longrightarrow A$ *is a map such that for each* $x_i \in X, \mathcal{A}(x_i) = A_i \subseteq A$;

Principles from philosophical ontology suggest that ***attribute precedence*** can represent the semantic relationship between attributes. Attribute precedence provides a formalism to represent subsumption relations between attributes. Assume that *r* and *s* are two attributes in a domain of interest; *s* precedes *r* (denoted as $r \rightarrow s$) means any instance that possesses *r* it also possesses *s*. For instance, we know that instances can be seen, if they have colour. This relationship can be represented by attribute precedence formalism. In this example, *r* is *has a colour* and *s* is *visible*; *visible* precedes *has a colour* which is denoted as *has a colour* → *visible* (see Parsons and Wand (2008) for a detailed discussion).

The attribute precedence formalism can be used to represent the relationship between specialized and general attributes (Parsons & Wand, 2003). Attributes in a lattice can be manifestations of higher-level attributes (Parsons & Wand, 2003), and such higher-level attribute supports the semantic integration of lattices (discussed in section 2.5). For instance, both *nurse* and *doctor*, in a hospital context, are manifestation of a more general attribute which is *hospital staff*, meaning that any individual who is either a *nurse* or a *doctor* is a *hospital staff* (*doctor* → *hospital staff* and *nurse* → *hospital staff)*.

Definition 3 (Attribute Precedence): Assume *r* and *s* are attributes such that *r*, s ∈ A. A ***precedence*** exists between *r* and *s* (denoted by $r \rightarrow s$) if and only if every instance that possesses *r* also possesses *s*.

The set of all precedences in an attribute lattice is denoted by Pr, and is defined as following:

Expr 4. $Pr = \{ pr_1, pr_2, pr_3, \dots, pr_k\} =$
$\{(r,s): r,s \in A \text{ , such that } \forall x_i \in X, r \in \mathcal{A}(x_i) \rightarrow s \in \mathcal{A}(x_i)\}$;

For brevity, hereafter, the above expression will be presented as follows:

Expr 5. $Pr = \{ pr_1, pr_2, pr_3, \dots, pr_k\} = \{(r,s): r \rightarrow s\}$;

Definition 4 (Attribute lattice): In mathematical terms, an attribute lattice is a directed acyclic graph which denoted by an ordered triple $L = (A, Pr, \mathcal{F})$ where

Expr 6. $A \neq \phi$
$Pr \neq \phi$
$\mathcal{F} : Pr \mapsto A \times A$ *is a map;*

The set A is the set of attributes, and the set Pr the precedences. If $\mathcal{F}(Pr) = (r,s)$, then r is called the preceded and s the inferred (preceding) attribute. If L is a acyclic digraph, then $A(L)$ and $Pr(L)$ will always denote the set of attributes and precedences of L, respectively.

Following constraint (Expr 7) states that the lattice is acyclic, and loops and parallel edges are not allowed in the lattice.

Expr 7. $For\ any\ r \in A, \nexists pr_j\ such\ that\ \mathcal{F}(pr_j) = (r,r).$
$For\ any\ given\ pr_i, pr_j \in Pr;\ \mathcal{F}\ (pr_i) \neq \mathcal{F}\ (pr_j).$
$For\ any\ \{r_1, r_2, \dots, r_n\} \subseteq A, and\ \{(r_1,r_2),(r_2,r_3), \dots, (r_{n-1}, r_n)\} \subseteq L, \nexists (r_n, r_1) \in L.$

Parsons and Wand (2008) define the ***full expansion*** of a class as the set of all attributes common to all class members. In an attribute lattice, this definition is adopted to formally define the expansion of an attribute.

Definition 5 (Expansion of an attribute): Expansion of an attribute a_i, denoted as a_i^E, is a set of attributes such that possessing the given attribute a_i is equivalent to possessing all attributes in the set (a_i^E). In other words, any given attribute is semantically equal to the union of attributes in its expansion.

Expr 8. $$a_i^E = \bigcup_{j=1}^{m} a_{ij} \quad for\ some\ a_{ij} \in A(L)$$
$$such\ that\ a_i^E \subset (A(L) - \{a_i\})$$
$$and\ a_i \in \mathcal{A}(x_i) \approx \{a_{i1}, a_{i2}, \dots, a_{im}\} \in \mathcal{A}(x_i)$$

where

$\approx$ *represents semantical equivalency*

Parsons and Wand (1997, 2008) offer criteria for meaningful classification in conceptual modeling. A set of attributes possessed by some instances in the domain (potential class) is a ***class***[5] whenever it has a ***base*** - a strict subset of attributes that is sufficient to identify an instance as a member of the class, and from which the remaining attributes of the class can be inferred (Parsons & Wand, 1997, 2008). In other words, a class must provide information (in terms of new attributes) about its members beyond the attributes required to identify members as belonging to the class. Where a base provides sufficient information to identify class membership, classes inferred from a single set of attributes

[5] The concept of "class" in this thesis is equivalent to the definition of ***useful class*** in Parsons and Wand (2008)

(base) will be identical classes. Moreover, Parsons and Wand (2008) define a *qualifying set* as a set of properties possessed by some instances in a class (not all) which provides sufficient information to infer class membership.

Following the same principles and using the notion of expansion of attributes, three types of attributes can be distinguished in the attribute lattice. Attributes can be designated as either a class, a category or a property attribute. The pattern of precedences around an attribute, which reflects its semantic relationships, is utilized for this designation. In the following, each attribute type is defined, following by an explanation of how the pattern of precedences specifies these types.

Definition 6 (Class): Let a_i denote an attribute in an attribute lattice. a_i is a ***class*** if and only if: (1) the cardinality of its expansion is greater than one ($|a_i^E| > 1$); (2) at least one proper subset of a_i^E exists such that other attributes in a_i^E can be inferred from it ($\exists\ B \subset a_i^E\ such\ that\ B \rightarrow \{a_i^E - B\}$); and (3) at least one instance exists that possesses all attributes in a_i^E.

Definition 7 (Category): Let a_i denote an attribute in an attribute lattice. a_i is a ***category*** if and only if: (1) the cardinality of its expansion is greater than one ($|a_i^E| > 1$); (2) no proper subset of a_i^E exist such that other attributes in a_i^E can be inferred from it; and (3) at least one instance exists that possesses all attributes in a_i^E.

Note that, a category is also referred to as a potential class using the terminology in Parsons and Wand (2008).

Definition 8 (Property): Assume a_i is an attribute in an attribute lattice. a_i is a ***property*** if and only if its expansion is an empty set ($|a_i^E| = 0$).

To summarize, attributes in an attribute lattice can be designated as either a class, a category or a property. The attribute is a property if the semantics of the attribute cannot be represented by the union of other attributes in the domain ($|a_i^E| = 0$). Otherwise, it is either a class or a category. If all attributes in the expansion (a_i^E) of a non-property attribute can be inferred from a strict subset of it, the attribute is a class; otherwise, it is a category.

Mathematically, c is a function (map) from an attribute to its designated type. In other words, for any given attribute (a_j), the output of this function will be either class, category or property.

Expr 9. $c : A(L) \longrightarrow \{Class, Category, Property\}$

is a map, such that for each $a_i \in A(L)$, $c(a_i)$ *is a designated type of the attribute*

For example, in a university context, assume that saying an individual is a *student* is semantically equal to saying an individual has a *student number*, a *program* of study, a *degree*, and a *start date* for her program. In this setting, *student* is an attribute in the attribute lattice, and its expansion is *student number*, *program*, *degree*, and *start date.* This attribute, *student*, is a candidate to be a class or category. Also assume that possessing *student number* is sufficient to infer that an individual is a *student*, and has a *student number*, a *program* of study, a *degree*, and a *start date* for her program. This means a proper subset of *student* expansion (*student number*) exist such that possessing other attributes in the expansion can be inferred from possessing it. As a result, *student* is a class, and *student number* is a base for this class. Expression 10 shows the mathematical representation of this class.

Expr 10. $A = \{ student, student\ number, program, degree, start\ date\};$

$student \in \mathcal{A}(x_i) \approx \{ student\ number, program, degree, start\ date\} \in \mathcal{A}(x_i)$

$student^E = \{student\ number, program, degree, start\ date\}$

$c(student) = Class;$ *Note that other attributes are properties.*

In addition, in the same context suppose that some *graduate students* are eligible for teaching courses as a part-time *instructor*. These *graduate instructors* (not all graduate students) are considered *employees* of the university. Thus, it is reasonable to define a new attribute – that is, *graduate instructor*. Possessing *graduate instructor*, as an attribute in the attribute lattice, is semantically equal to being a *graduate student,* and an *instructor* at the same time. As a result, the expansion of *graduate instructor* is *graduate student* and *instructor.* Since no proper subset of this expansion provides sufficient information to infer other attributes in the expansion, *graduate instructor* is a category. Mathematical description of these relationships are given in Expression 11.

Expr 11. $A = \{ graduate\ student, instructor,\ graduate\ instructor\};$

$graduate\ instructor \in \mathcal{A}(x_i) \approx \{ graduate\ student, instructor\} \in \mathcal{A}(x_i)$

$graduate\ instructor \rightarrow employee$

$graduate\ instructor^E = \{graduate\ student, instructor\}$

$c(graduate\ instructor) = Category;$ *Note that other attributes are properties.*

Semantic relativism states that a given attribute should be designated as a class, category or property, based on the pattern of arc and attributes around it. To be able to, first, capture the semantics of relationships between attributes, and second, infer the type of attribute based on precedences, three types of precedences are introduced – ***simple precedence***, ***base precedence***, and ***subcategory precedence***.

In the following, the function which mathematically expresses the type of precedence, $t(pr_i)$, is formally defined. This is followed by the formal definition of the semantics being represented by each precedence type.

t is a function (map) from a precedence to its type. For any given precedence relationship (pr_i), the output of this function will be either P (*simple precedence*), S (*subcategory precedence*), or B (*base precedence.*)

Expr 12. $t : Pr\ (L) \longrightarrow \{P, S, B\}$

is a map, such that for each $pr_i \in Pr\ (L)$, $t\ (pr_i)$ *is a type of precedence*

Precedence relations with the type P, provide semantics not further than simple precedence which is defined earlier (Expr 4). It is worthwhile to note this relationship (simple precedence) is a transitive relationship, that is, if $r \rightarrow s$ and $s \rightarrow$ t, by definition $r \rightarrow t$.

Expr 13. *For any given* $pr_i = (r_i, s_i)$; *if* $t\ (pr_i) = P$, *then* $r_i \rightarrow s_i$

Following simple precedence definition, the ***direct inference*** of any given attribute (r_i), denoted by $dp(r_i)$, refers to the union of attributes which are connected as inferred attributes via simple precedence to the given attribute.

Expr 14. $dp(r_i) = \bigcup_{j=1}^{k} s_{ij}$ *for all* $s_{ij} \in A(L), (r_i, s_{ij}) \in Pr(L)$, *and* $t\left((r_i, s_{ij})\right) = P$.

Definition 9 (Subcategory precedence): Assume *r* is a category and *s* is an attribute in its expansion ($s \in r^E$). A ***subcategory precedence*** exists between *r* and *s.* This type of precedence is denoted by an arc labeled with S ($r \xrightarrow{S} s$).

Subcategory precedence (s), declares that a preceded attribute is a category, and the expansion of this attribute (category) is equal to the union of all attributes connected to it via outgoing subcategory precedence.

Expr 15. $For\ any\ given\ pr_j = (r_i, s_i);\ \ if\ t(pr_i) = S,\ then$

$a)\ r_i \rightarrow s_i$

$b) c(r_i) = Category$

$$c)\ r_i^E = \bigcup_{j=1}^{k} s_{ij}\ \ for\ all\ s_{ij} \in A(L), (r_i, s_{ij}) \in Pr(L)\ and\ \ t\left((r_i, s_{ij})\right) = S$$

Definition 10 (Base precedence): Assume *r* is a class and *s* is an attribute in its expansion ($s \in r^E$) such that other attributes in r^E can be inferred from it. A ***base precedence*** exists between *r* and *s.* This type of precedence is denoted by an arc labeled with B ($r \xrightarrow{B} s$).

Base precedence (B), declares that the inferred attribute (s_i) is a class, and the expansion of this attribute (class) is equal to the union of expansion of its bases and attributes which are directly inferred from each base.

Expr 16. $For\ any\ given\ pr_j = (r_i, s_i);\ \ if\ t(pr_i) = B,\ then$

$a)\ r_i \rightarrow s_i$

$b) c(s_i) = Cls$

$$c)\ s_i^E = \bigcup_{j=1}^{k} (r_{ij}^E \cup dp(r_{ij}))$$

$$for\ all\ r_{ij} \in A(L), (r_{ij}, s_i) \in Pr(L)\ and\ \ t\left((r_{ij}, s_i)\right) = B.$$

To illustrate, consider the example mentioned above. The following base and subcategory precedence relations exist:

Expr 17. $graduate\ instructor \xrightarrow{S} graduate\ student$

$graduate\ instructor \xrightarrow{S} instructor$

$student\ number \xrightarrow{B} student$

An attribute lattice can be represented as a graph in which nodes represent attributes and directed arcs represent precedences. Directed arcs labeled with S and B are used in the graphical representation to depict the subcategory and base precedence relationship, respectively. The graphical representation of sample attributes with these three precedence type are shown in Figure 1. Note that to be able to quickly identify various type of precedences, three different arc styles are used to represent these types. Also, class, category, and property attributes are represented by different colours.

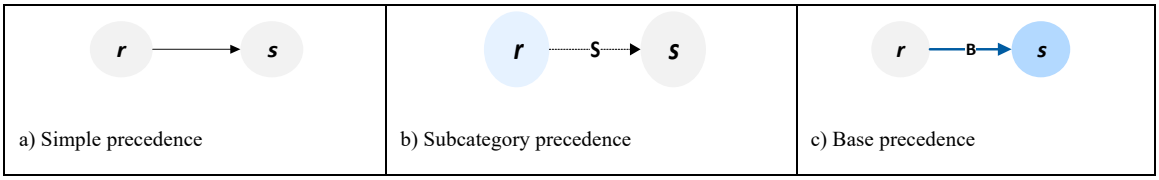


Figure 1. Three type of precedence relationships

Figure 2.a and 2.b represent the attribute lattice structure of the above discussed examples (Expressions 10 and 11, respectively.)

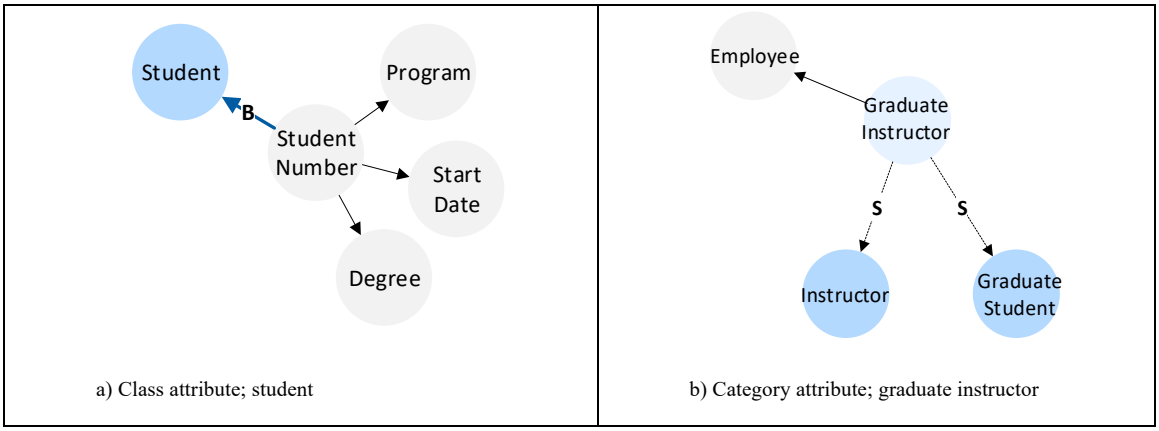


Figure 2. Class and category attribute

2.3.2 Attribute Lattice Grammar Characteristics

The previous section introduced the components of the attribute lattice grammar. This section discusses attribute lattice characteristics. These characteristics explain the semantic relativism in an attribute lattice – that is, how the pattern of precedences provide sufficient information to: (1) identify attribute type; (2) validate an attribute lattice (section 2.3.3); and (3) integrate distinct lattices (section 2.5). To discuss these characteristics, the immediate and semantic neighbourhood of attributes are defined by adopting neighbour definitions in regular digraph (section 2.2.3). These neighbourhoods are used to designate the attribute type, and to deduce the class structure of the domain represented by the attribute lattice.

In graph theory (Agnarsson & Greenlaw, 2007), specifically in digraphs, inneighbours and outneighbours of a node refer to a set of nodes that have the given node as a head or tail, respectively. An attribute lattice has two key differences from a regular digraph. First, in contrast with a regular digraph, which has only one type of arc, the attribute lattice has three types of precedences (P, S, B) with different semantics. Second, an

attribute (a node) in the attribute lattice might have an expansion. Precedences that are connected to attributes in its expansion might convey information that is of interest. As a result, inneighbours and outneighbours of attributes in the attribute lattice are defined such that they cover *type of precedence*, and whether the precedence is a *direct precedence,* or a *precedence through expansion* of an attribute.

The following expressions provide formal definitions of inneighbours and outneighbours in an attribute lattice. These expressions elaborate whether attributes are connected to the given attribute directly or through its expansion ($d\ or\ e$), the type of precedence (P, S, B), and whether the given attribute is a preceded or an inferred attribute ($-\ or\ +$) in the relationship. For instance, for a given attribute a_i, $N^{-dB}\ (a_i)$ is a set of attributes that are *directly* connected to a_i with a *base* precedence relationship, and a_i appears as an *inferred* attribute in the precedence relationship.

Expr 18. $N^{-dP}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, a_i), \mathcal{t}(pr) = P,$ *for some* $pr\ \in Pr(L)\}$.

$N^{-dS}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, a_i), \mathcal{t}(pr) = S,$ *for some* $pr\ \in Pr(L)\}$.

$N^{-dB}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, a_i), \mathcal{t}(pr) = B,$ *for some* $pr\ \in Pr(L)\}$.

$N^{+dP}\ (a_i) = \{s \in A(L): \mathcal{F}(pr) = (a_i, s), \mathcal{t}(pr) = P,$ *for some* $pr\ \in Pr(L)\}$.

$N^{+dS}\ (a_i) = \{s \in A(L): \mathcal{F}(pr) = (a_i, s), \mathcal{t}(pr) = S,$ *for some* $pr\ \in Pr(L)\}$.

$N^{+dB}\ (a_i) = \{s \in A(L): \mathcal{F}(pr) = (a_i, s), \mathcal{t}(pr) = B,$ *for some* $pr\ \in Pr(L)\}$.

$N^{-eP}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, s), r \notin a_i^E, s \in a_i^E, \mathcal{t}(pr) = P,$
for some $pr\ \in Pr(L)\}$.

$N^{-eS}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, s), r \notin a_i^E, s \in a_i^E, \mathcal{t}(pr) = S,$
for some $pr\ \in Pr(L)\}$.

$N^{-eB}\ (a_i) = \{r \in A(L): \mathcal{F}(pr) = (r, s), r \notin a_i^E, s \in a_i^E, \mathcal{t}(pr) = B,$
for some $pr\ \in Pr(L)\}$.

$N^{+eP}(a_i) = \{s \in A(L): \mathcal{F}(pr) = (r,s), r \in a_i^E, s \notin a_i^E, t(pr) = P,$
$for\ some\ pr \in Pr(L)\}.$
$N^{+eS}(a_i) = \{s \in A(L): \mathcal{F}(pr) = (r,s), r \in a_i^E, s \notin a_i^E, t(pr) = S,$
$for\ some\ pr \in Pr(L)\}.$
$N^{+eB}(a_i) = \{s \in A(L): \mathcal{F}(pr) = (r,s), r \in a_i^E, s \notin a_i^E, t(pr) = B,$
$for\ some\ pr \in Pr(L)\}.$

Using the same notation, relationships of a given attribute (a_i) to another attribute in its neighbours (a_j), denoted by $Rel(a_i, a_j)$, can be abbreviated as follow:

Expr 19. $Rel(a_i, a_j) \in$
$\{-dP, -dS, -dB, +dP, +dS, +dB, -eP, -eS, -eB, +eP, +eS, +eB\}$
Where
$-\ and\ +$ *represents whether the given attribute is the inferred or preceded attribute,*
$d\ and\ e$ *represents whether the precedence is a direct precedence or precedence through its expansion,*
$P, S\ and\ B$ *represents the type of precedence*

Characteristic 1 (Immediate neighbourhood): The pattern of precedences around each attribute can be represented by a set of paired elements in which the first component is an attribute a_j which is <u>*directly*</u> connected to the given attribute and the second component is the type of this relationship, this pattern is called the immediate neighbourhood (IN) of the given attribute (a_i).

Expr 20. $IN(a_i) = \{(a_j, Rel(a_i, a_j): for\ all\ a_j\ where\ (a_i, a_j)\ or\ (a_j, a_i) \in$
$Pr(L)\ , and\ Rel(a_i, a_j) \in \{-dP, -dS, -dB, +dP, +dS, +dB\}\}$

Characteristic 2 (Attribute type): Each attribute has one type at a time and this type can be deduced based on its immediate neighbourhood.

The type of an attribute is defined based on the expansion of the attribute – that is, a set of attributes that are semantically equal to the given attribute. If the cardinality of an expansion of the attribute is zero, the attribute is a property. Also, if at least one proper subset of the expansion exists such that other attributes in the expansion can be inferred from it, the attribute is a class, otherwise it is a category. Hence, an attribute has only one type at any time. By definition (Expr 15), the preceded attribute in a subcategory precedence is a category, and the inferred attribute in the base precedence is a class. As a result, the immediate neighbourhood of an attribute provides sufficient information to deduce the type of attribute.

Corollary (Class attribute): An attribute a_i in the lattice represents a class if and only if it has at least one incoming base precedence. In other words, a_i is a class if and only if $|N^{-dB}(a_i)| \geq 1$.

Corollary (Category attribute): An attribute a_i in the lattice represents a category if and only if it has at least one outgoing category precedence. In other words, a_i is a category if and only if $|N^{+dS}(a_i)| \geq 1$.

Corollary (Property attribute): An attribute a_i in the lattice represents a property if and only if it has neither incoming base precedence nor outgoing category precedence. In other words, a_i is a property if and only if $|N^{+dS}(a_i)| = 0$ and $|N^{-dB}(a_i)| = 0$.

Characteristic 3 (Expansion of each attribute): The expansion of each attribute can be deduced based on its immediate neighbourhood.

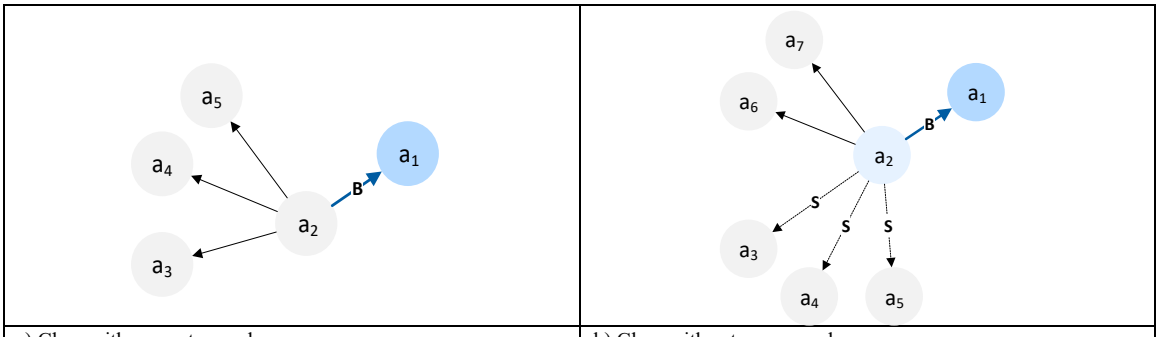
The expansion of class and category attributes can be deduced by the pattern of precedences in the immediate neighbourhood of attributes. Following subcategory precedence definition (Expr 15), for a given category attribute, the union of all inferred attributes from subcategory precedences defines the category expansion.

Expr 21. $If\ |N^{+dS}\ (a_i)| \geq 1\ , then\ c(a_i) = Category, and\ a_i^E = N^{+dS}\ (a_i)$

A class attribute in an attribute lattice has at least one base (Expr 16), and other attributes of its expansion can be inferred from base attributes. Hence, the union of all bases of a class, and attributes that can be directly inferred from these bases constitute the expansion of a class. It is worthwhile to note that the base attribute can be a category or a property attribute. The following expression shows the formal definition.

Expr 22. $If\ |N^{-dB}\ (a_i)| \geq 1\ , then\ c(a_i) = Class, and$
$a_i^E = \cup\{N^{-dB}\ (a_i)\ , N^{+dP}\ (a_{ij})\}, for\ all\ a_{ij}\ \in N^{-dB}\ (a_i)$

Figure 3a and 3b show the class structure in an attribute lattice with a property and category attribute as a base, respectively.



a) Class with property as a base

b) Class with category as a base

Figure 3. Class structure in an attribute lattice

Corollary (Qualifying Attribute for a Class): An attribute a_i in the lattice represents a qualifying attribute for a class attribute a_j if and only if a_j is a class, and it can be inferred with the direct precedence from a_i. In other words, a_i is a qualifying attribute for a_j if and only if $c(a_j) = Class$ and $Rel(a_i, a_j) = +dP$.

Characteristic 4 (Semantic neighbourhood): The pattern of precedences related to each attribute and its expansion can be represented by the set of paired elements in which the first component is an attribute (a_k) that is connected to the given attribute either directly or through its expansion and it is not part of its expansion. The second component is the type of this relationship, this pattern is called the semantic neighbourhood (SN) of the given attribute (a_i).

Expr 23. $SN^+(a_i) = \{(a_k, Rel(a_i, a_k): for\ all\ a_k \notin a_i^E, a_l \in \{a_i \cup a_i^E\}, where\ (a_l, a_k) \in Pr(L)\}$.

$SN^-(a_i) = \{(a_k, Rel(a_i, a_k): for\ all\ a_k \notin a_i^E, a_l \in \{a_i \cup a_i^E\}, where\ (a_k, a_l) \in Pr(L)\}$.

$SN(a_i) = SN^+(a_i) \cup SN^-(a_i)$

Note, immediate neighbourhood and semantic neighbourhood are defined to formalize the relation of a given attribute to other attributes in the lattice. IN formalizes how an attribute, individually, is related to other attributes. In contrast, SN encapsulates the expansion of attributes and formalizes how the given attribute, directly and through its expansion, is related to other attributes in its neighbourhood. Hence, to define the SN of an attribute, the interrelationships among attributes that are part of the attribute expansion are excluded.

For attributes that are designated as a class, or category, not all pairs in the IN are in the SN. However, the IN and SN of properties will be identical.

Expr 24. $if\ c(a_i) \in \{Class, Category\}, Then\ IN(a_i) \not\subset SN(a_i).$

$if\ c(a_i) = Property, Then\ IN(a_i) = SN(a_i).$

As discussed later in section 2.5, the semantic neighbourhood of attributes is utilized to integrate lattices and to suggest potential merge nodes based on the known merge nodes.

2.3.3 Attribute Lattice Validation

This section develops attribute lattice validation rules – that is, rules that can be used to validate if the structure of a given attribute lattice is consistent with the basic attribute lattice grammar component definitions. In other words, given an attribute lattice, these rules can be utilized to verify that each attribute has only one type, to verify that the class and category attributes are meaningful, to eliminate redundant precedences, and to eliminate precedences that can be immediately inferred from other precedences in an attribute lattice.

Rule 1 (Multiple precedence relationship): Following the lattice definition (Definition 4), multiple precedences (of any type) between two attributes are not permitted. That is, $for\ any\ given\ pr_i, pr_j \in Pr;\ \mathcal{F}\ (pr_i) \neq \mathcal{F}\ (pr_j)$

Rule 2 (Attribute type): An attribute should be designated as either a class, a category, or a property, that is, it cannot a class and a category at the same time. As a result, it cannot have both direct incoming base precedence and outgoing subcategory precedence.

In other words, for any a_i in the lattice, $|N^{-dB}(a_i)| \geq 1$ and $|N^{+dS}(a_i)| \geq 1$ cannot happen at the same time.

Rule 3 (Class validation): Assume a_i is designated as a class based on its immediate neighbourhood – that is, $|N^{-dB}(a_i)| \geq 1$. There should be at least one attribute in its expansion such that it is inferred form one of its bases – that is, $N^{-dB}(a_i) \subset a_i^E$.

This rule ensures that the class provides information (in term of attributes) for class members *beyond what is needed to identify the members* (base attributes).

Note that if a class attribute precedes another class attribute with a simple precedence relationship, then a superclass/subclass relationship exists between them. Similarly, a superclass/subclass can be represented by a base precedence - that is, a class could be a base for another class. However, the structure is valid if and only if it satisfies above-mentioned rule (class validation rule).

Rule 4 (Category validation): Assume a_i is designated as a category based on its immediate neighbourhood – that is, $|N^{+dS}(a_i)| \geq 1$. There should be at least two attributes in its expansion – that is, $a_i^E = |N^{+dS}(a_i)| > 1$.

Figure 4 shows invalid attribute lattice structures. In the first lattice (a), a_2 has no valid attribute type. It has both incoming base precedence and outgoing subcategory precedence, hence, this structure violates attribute lattice definitions. Second and third partial lattices (b and c) are invalid because no attribute can be inferred from a_1 bases. And finally, in the last lattice (d), $a_1$has only one subcategory precedence which is in contrast with category definition. As a result, it is not a valid lattice structure.

One desirable quality (for simplicity) for an attribute lattice is to represent only ***non-redundant*** relationships. The precedence relation (an arc) in the lattice is considered redundant if it can be inferred from (i.e., is implied by) other precedence relations. The following guidelines provide mechanisms to eliminate unnecessary precedences in the lattice.

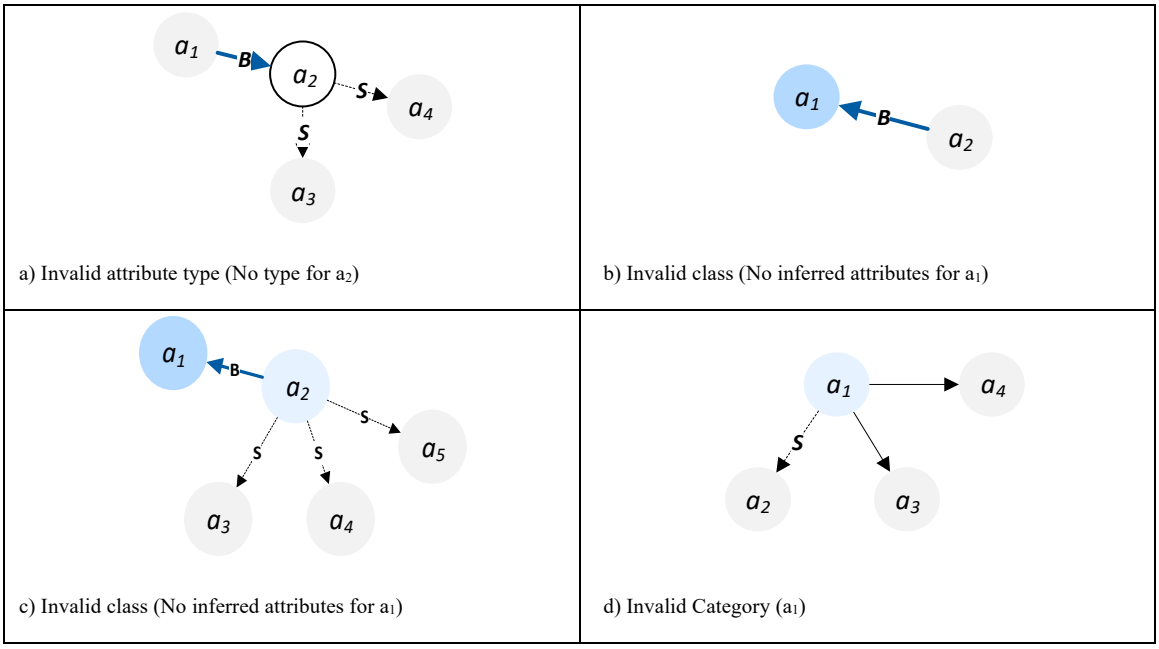


Figure 4. Invalid Attribute Lattice Structures

Rule 5 (Transitive Redundancy): The precedence between two attributes is redundant if it can be inferred from a transitive chain of two simple precedences.

The simple precedence relationship is a transitive relationship, that is, if $a_1 \rightarrow a_2$ and $a_2 \rightarrow a_3$, by definition $a_1 \rightarrow a_3$. Representing all precedences that can be inferred from the chain of two other simple precedences unnecessarily increases the number of

precedences and decreases the attribute lattice clarity. Figure 5.a shows this redundant precedence.

Rule 6 (Class Attribute Redundancy): Precedence from a class to an attribute that can be immediately inferred from one of its bases (attributes in class expansion) is redundant.

Possessing a class attribute is semantically equal to possessing all attributes in its expansion. Following attribute characteristics (Characteristic 3), the expansion of a class can be identified from the immediate neighbourhood of the class, and hence, precedence from a class to its expansion provides redundancy. Figure 5.b shows a class attribute (a_1) with a redundant precedence.

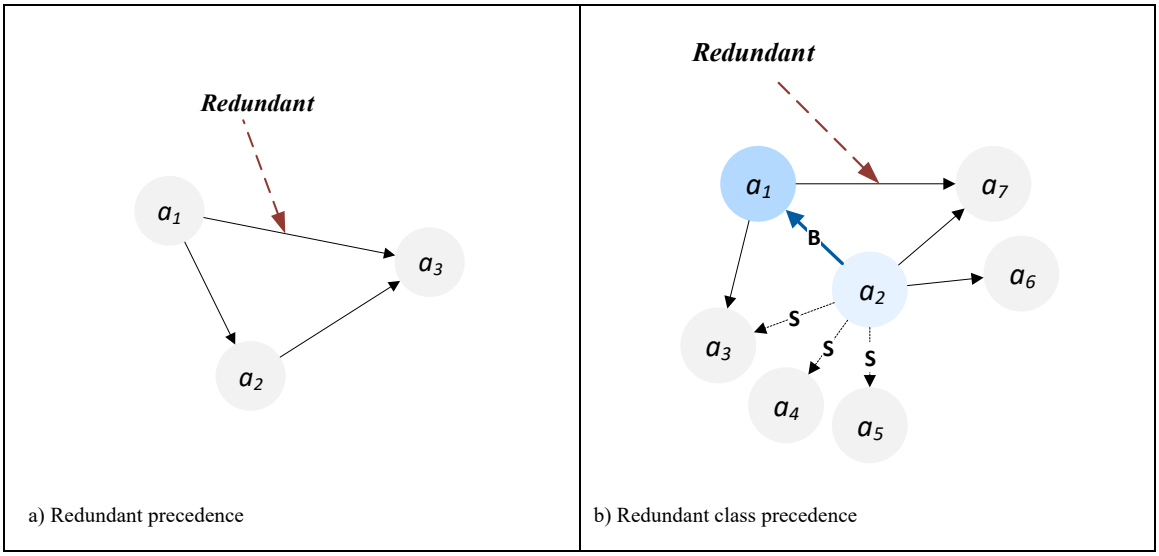


Figure 5. Redundant Precedences

2.3.4 Attribute Lattice Grammar Comparison with Description Logics (DL)

The attribute lattice conceptual modeling grammar represents relations among attributes – true statements about instances – as perceived by users of data in the domain of interest. This grammar is developed to provide an inferential representation scheme for data sources. Moreover, this inferred scheme provides a semantic foundation to address semantic data heterogeneity among various data sources. This grammar follows the assumptions that (1) classification is a critical ability in humans to understand and communicate about the world (Lakoff, 1987; Parsons & Wand, 2008), and (2) finding classes (cognitive abstractions) in different data sources that refer to the same set of instances in the real-world is key in data integration (Clifton et al., 1998). Following these assumptions, this grammar offers a minimal set of components (i.e., attributes and three types of precedences) to represent the class structure of the domain. Classification (in the sense of the role of classification in human cognition) is central to the definition of this notion and development of attribute lattice integration.

This section compares this conceptual modeling grammar with other knowledge representation languages – that is, Description Logics (DL) based languages. The attribute lattice grammar is translatable to DL; however this translation will lead to losing a part of the semantics captured and represented in the attribute lattice.

Similar to Description Logics (DL), subsumption relations among attributes are used to develop the attribute lattice grammar. Hence, this approach has some shared assumptions with DL. However, grounded on different assumptions, the semantics represented by attribute lattices and DL languages are different. DL languages offer a variety of constructors

to represent a knowledge base (Baader, 2003). Generally speaking, research in DL languages examines, first, the extent to which the constructors of languages can capture the semantics of the knowledge base, and second, the trade off between language expressiveness and the complexity of reasoning (Brachman & Levesque, 1984). In the attribute lattice grammar, however, we focus on the minimal set of components, constructed based on human cognition, to capture subsumption relationships among attributes as perceived by human users. This conceptual modeling grammar aims to represent the class structure of heterogeneous data sources from the perspective of users of data, independent from initial (predefined) schema of data source, and to provide a basis for class-based data integration.

- ***Attributes (in an attribute lattice) are neither concepts nor roles (in DL)***

In DL, concepts - unary predicates - refer to expressions that denote the set of individuals, and roles – binary predicates – express relationships between concepts. Attributes in the attribute lattice grammar are true statements about instances, hence, they are comparable with concepts. The notion of "concept" in description logic languages treats all the concepts as classes. As a result, classification methods in this approach provide a class lattice based on all unary predicates.

In the attribute lattice grammar, I follow classification guidelines (Parsons & Wand, 2008) for meaningful classification of instances in the domain. In other words, following the principle that we form classes to allow us to infer attributes of an instance that are not required to classify it (a key element of usefulness), *not all attributes in a lattice are meaningful domain classes*. I f attributes are represented by concepts in a DL, the resulting lattice

will contain “unnecessary” concepts (that is, concepts that do not enable inferences), which makes the conceptual model unclear for humans.

- ***Precedence relationship in the attribute lattice grammar is not necessarily IS-A relation.***

In the DL paradigm, statements can be divided into two groups; the TBox, general properties of concepts, and the ABox, declaration of knowledge involving individuals. Intuitively, the notion of attribute lattice can be compared to TBox. Two primary types of axiom exist in the terminology (TBox); inclusion axioms denoted by $C \sqsubseteq D$ and equality axioms denoted by $C \equiv D$ in which $C\ and\ D$ are concepts. The subsumption relationship (inclusion) declares that concept D (the subsumer) is considered more general than the concept C (the subsumee.) This subsumption relationship further will be used to make inferences related to IS-A relationships that are not directly declared in the knowledge base, and to build the hierarchy of concepts.

The interpretation of subsumption relationship in the DL – any instance belong to the subsumee, also belongs to subsumer - is similar to the interpretation of precedence relationship in the attribute lattice grammar – any instances possessing preceded attributes will possess inferred attributes. However, these two are not semantically equal.

Grounded on semantic networks, in the DL paradigm subsumption relationship reflects IS-A relation, where the attribute lattice grammar utilizes three types of precedence relationships to capture the semantics of subsumption relationships among attributes.

Base precedence in the lattice, ($r \xrightarrow{B} s$) is crucial for meaningful classification (based on classification guidelines (Parsons & Wand, 2008)) and provides information beyond the simple subsumption relation (inclusion). Base precedence declares that *s* is a class attribute and other attributes in the lattice exist such that possessing *r* is semantically equal to possessing them. This precedence also states a *sufficient* attribute, such that instances possessing the attribute are a member of class *s* and implies other attributes can be inferred from this membership. Likewise, subcategory precedence provides semantics beyond the simple subsumption relationship. In addition to simple subsumption, this precedence also reflects that possessing the preceded attribute is semantically equal to possess all inferred attributes with subcategory precedence relationship.

For instance, using the above-discussed example (2.3.1), the *student* (class) precedes the *student number* (attribute) with the base precedence relationship. This base relationship can be translated into DL with defining both *student* and *student number* as concepts and defining an equivalence relationship between them. However, this translation miss capturing part of the semantics which represented by base precedence in this attribute lattice grammar.

To summarize, concept and inclusion in DL languages can be used to represent components of attribute lattice (i.e., attributes and precedences). However, in the resulting structure, all attributes will be considered as classes, all precedences will be considered as IS-A relationships, and the class structure (meaningful class structure from the human point of view) cannot be inferred.

2.4 Attribute Lattice Example

2.4.1 Attribute lattice creation

Extending the above example from a university context (section 2.3.1), this section demonstrates how an attribute lattice represents the attributes and precedence relationships among them, and how attribute lattice characteristics are used to infer the class structure based on the pattern of precedences. In addition, I elaborate by an example how a given attribute lattice can be validated using above discussed validation rules (section 2.3.3).

As discussed, let *student* be a class attribute in the attribute lattice with *student number*, *program*, *degree*, and *start date* in its expansion.

$$Student^E = \{Student\ number, Program, Degree, Start\ date\}$$

Assume *instructors*, in the university context, have a separate *contract* for their teaching that includes a *course to teach,* and a *course-based salary*. As a result, possessing *instructor* attribute is semantically equal to possessing *instructor contract, course to teach,* and *course-based salary.* In this setting, possessing *instructor contract* provides sufficient information to infer that an instance is an *instructor,* and it possesses *course to teach* and *course-based salary*. Hence, *instructor contract* is a base for *instructor*. Likewise, *faculty, person* and *graduate student* are other class attributes in this domain.

$$Instructor^E = \{Instructor\ contract, Course\ to\ teach, Course\ based\ salary\}$$
$$Faculty^E = \{Hiring\ contract, Faculty\ salary\}$$
$$Graduate\ student^E = \{Student\ with\ fund, RA/TA\ position\}$$
$$Person^E = \{SSN, DoB, Name\}$$

As discussed (section 2.3.1), in this domain *graduate instructors* are considered to be employees of the university. As a result, an attribute is defined to represent the *graduate instructor*. Any instance that possesses this attribute possess both *graduate student* and *instructor* attributes. However, no proper subset of the expansion of *graduate instructor* exists such that other attributes in its expansion can be inferred from it. Hence, *graduate instructor* is a category. Table 2 shows the list of class and category attributes with their expansions. Note that underlined attributes in this table are base attributes.

Table 2. The list of class and category attributes and their expansions

Attribute	***Type***	***Attribute Expansion***
Person	Class	***SSN***, DoB, Name
Faculty	Class	***Hiring contract***, Faculty salary
Student	Class	***Student Number***, Program, Start Date, Degree
Graduate student	Class	***Student with fund***, RA/TA position
Instructor	Class	***Instructor Contract***, Course based salary, Course to Teach
Graduate instructor	Category	Instructor, Graduate student

Let assume, in addition to above discussed information, following precedence relationships are given in this domain.

$$Graduate\ instructor \rightarrow Employee$$
$$Faculty \rightarrow Employee$$
$$Instructor \rightarrow Employee$$
$$Graduate\ student \rightarrow Student$$
$$Student \rightarrow Person$$
$$Employee \rightarrow Person$$

Finally, assume the university offers insurance for graduate students. However, not all graduate students use this insurance.

$$Graduate\ insurance \rightarrow Graduate\ student$$

Using the above information, an attribute lattice can be created as in Figure 6.

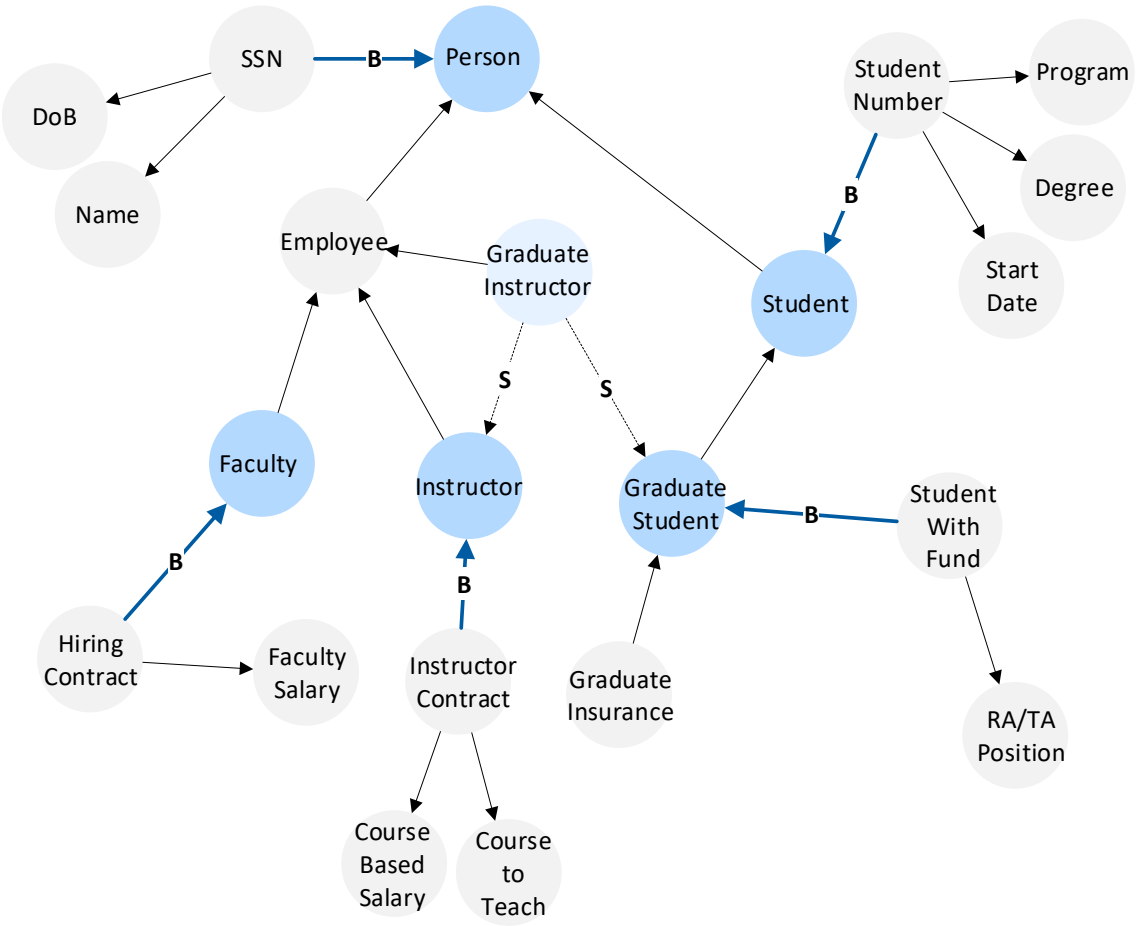


Figure 6. Attribute lattice

The first characteristic of the attribute lattice describes how the immediate neighbourhood of each node can be represented. Using this characteristic, the pattern of precedences directly connected to the given attribute is represented as a set of pairs in which the first element shows an attribute that is connected to the given attribute, and the second element shows their relationship. Expression 25 shows the immediate neighbourhood of some of the attributes in Figure 6.

Expr 25. $IN(Person) = \{(SSN, -dB), (Emploee, -dP), (Studemt, -dP)\}$

$IN(SSN) = \{(Person, +dB), (Name, +dP), (DoB, +dP)\}$

$IN(DoB) = \{(SSN, -dP)\}$

$IN(Name) = \{(SSN, -dP)\}$

$IN(Employee) = \{(Person, +dP), (Faculty, -dP), (Instructor, -dP),$
$(GraduateIntructor, -dP)\}$

$IN(GraduateIntructor) = \{(GraduateStudent, -dS), (Intructor, -dS)\}$

Using the second and third characteristics of the attribute lattice grammar, attribute types and their expansion can be identified by the immediate neighbourhood of attributes. For instance, in Expression 25, attribute *Person* has only one incoming direct base precedence ($-dB$). This relationship reflect that this attribute is a class, with one base. The base of this class (SSN) has two outgoing direct simple precedences ($+dP$). The base attribute (SSN), and two attributes ($Name, DoB$) which are connected with ($+dP$) precedence to the base constitute the expansion of the class attribute ($Person$).

2.4.2 Attribute lattice validation

Attribute lattice validation (Section 2.3.3) offers a set of rules to verify whether the pattern of precedences in a given attribute lattice is consistent with the attribute lattice principles.

The first four validation rules (Rules 1 to 4) discuss the correctness of the model and assert that a model is correct if each attribute has only one type, and class/category attributes are meaningful. The last two rules (Rules 5, and 6) focus on the precedence redundancy. This section elaborates these rules with an example.

Assume that the attribute lattice in Figure 7 is created based on domain knowledge and we are interested in examining if the lattice structure (the pattern of precedences) is valid. Table 3 shows the validation results for this attribute lattice.

Given an attribute lattice, finding patterns that lead to violating rules could be a cumbersome task. Hence, as will be discussed in Chapter 3, an artifact is developed to examine validation rules for a given attribute lattice and to facilitate attribute lattice validation.

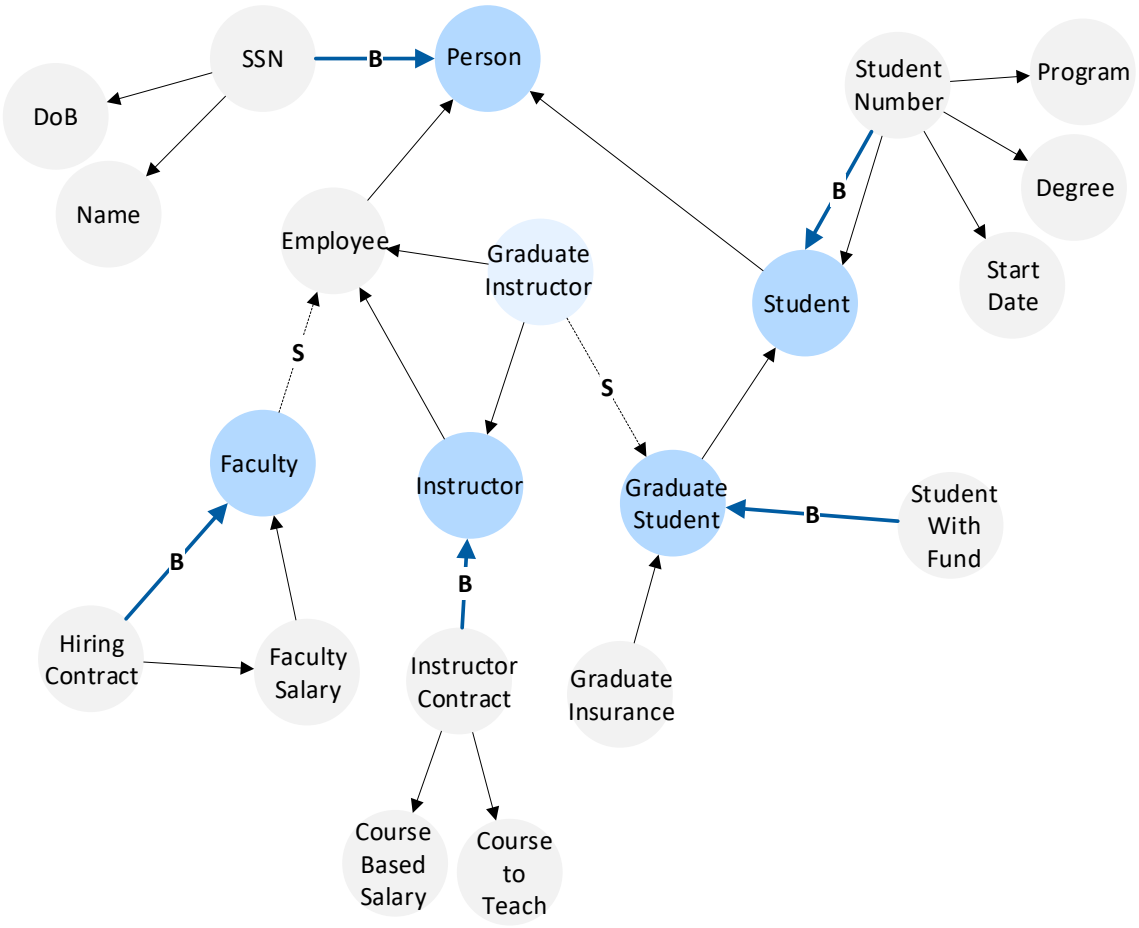


Figure 7. Preliminary Attribute Lattice

Table 3. Validation results for Lattice in Figure 7

Rule No.	***Rule Name***	***Result***
Rule 1	Multiple precedence relationship	Two precedences exist between *Student Number* and *Student*
Rule 2	Attribute type	Using the immediate neighbourhood of *Faculty*, this attribute has both incoming base precedence and outgoing subcategory precedence. This means this attribute is a category attribute, and class attribute at the same time.
Rule 3	Class validation	*Graduate Student* has an incoming base precedence. Hence, it is designated as a class. However, no attribute can be inferred from the attributes in its base.
Rule 4	Category validation	*Graduate instructor* is a category. However, it has only one attribute in its expansion. If *Faculty* is a category (it has two types for now), it cannot have only one attribute in its expansion.
Rule 5	Transitive Redundancy	*Instructor* precedes *Graduate Instructor. Employee*, in turn, precedes *Instructor*. By definition, *Employee* precedes *Graduate instructor.* As a result, *Graduate instructor* → *Employee* is redundant.
Rule 6	Class Attribute Redundancy	*Faculty Salary* is included in the expansion of *Faculty* attribute. Any direct precedence between these two attributes is redundant.

2.5 Attribute Lattice Integration

Given a structured, semi-structured, or even unstructured data source, an attribute lattice can represent its semantic structure. Attribute lattice integration aims to: (1) define the concept of similarity and merge nodes between attribute lattices; (2) show how similar attributes in distinct attribute lattices can be merged to provide a federated (unified) attribute lattice; and (3) elaborate how attribute lattice grammar principles can be utilized to find potential similar attribute nodes based on known merge nodes. In this context, a federated (unified) attribute lattice is an attribute lattice that includes all attributes and precedences from distinct lattices that represent a common underlying subject domain, and shows the relationship between similar attributes in these distinct lattices. Let L_1 and L_2be two distinct attribute lattices as follows:

Expr 26. $L_1 = (A_1, Pr_1, \mathcal{F}_1)$*, and* $L_2 = (A_2, Pr_2, \mathcal{F}_2)$ *are two distinct attribute lattices where*

$A_1 = \{a_{11}, a_{12}, a_{13}, \dots, a_{1i}\}, and\ Pr_1 = \{\, pr_{11}, pr_{12}, pr_{13}, \dots, pr_{1k}\}$

$A_2 = \{a_{21}, a_{22}, a_{23}, \dots, a_{2j}\}, and\ Pr_2 = \{\, pr_{21}, pr_{22}, pr_{23}, \dots, pr_{2m}\}$

In the following I discuss how these lattices can be integrated to offer a federated view over them.

2.5.1 Federated Attribute Lattice

The approach introduced in this section aims to find ***similar attributes*** (based on the semantic neighbourhood of attributes) in distinct lattices and create a unified view over these lattices by joining them on the ***merge nodes***. This federated view, in turn, gives the user of data the ability to query data from distinct sources.

Arguably, the most basic problem in semantic data integration is finding "similar" attributes which state the same kind of real-world information about instances in distinct data sources (Clifton et al., 1998). The definition of similarity varies greatly in different schema-based approaches for data integration. These approaches employ various methods (based on different similarity theories) to identify similar concepts (Evermann, 2008a, 2008b). Independent from the method employed to find similarity, the successfulness of the approach is measured by the extent to which similar concepts identified by the approach are perceived to be similar by a human user (Evermann, 2008a, 2008b).

To improve the similarity judgment quality, Evermann (2009) emphasizes the importance of attributes of instances, suggesting that integration approaches should use the most relevant attributes to find similar concepts. In the same line, the lattice integration procedure emphasizes semantic relativism, i.e. the ***immediate neighbourhood*** and the ***semantic neighbourhood*** of attributes, to find similar attributes.

In attribute lattice integration, following Parsons and Wand (2002, 2003) similar attributes have been defined as attributes in distinct lattices which either represent the same kind of real-world information about instances (i.e. semantically equal attributes) or represent the same kind of real-world information about instances in various levels. Based on

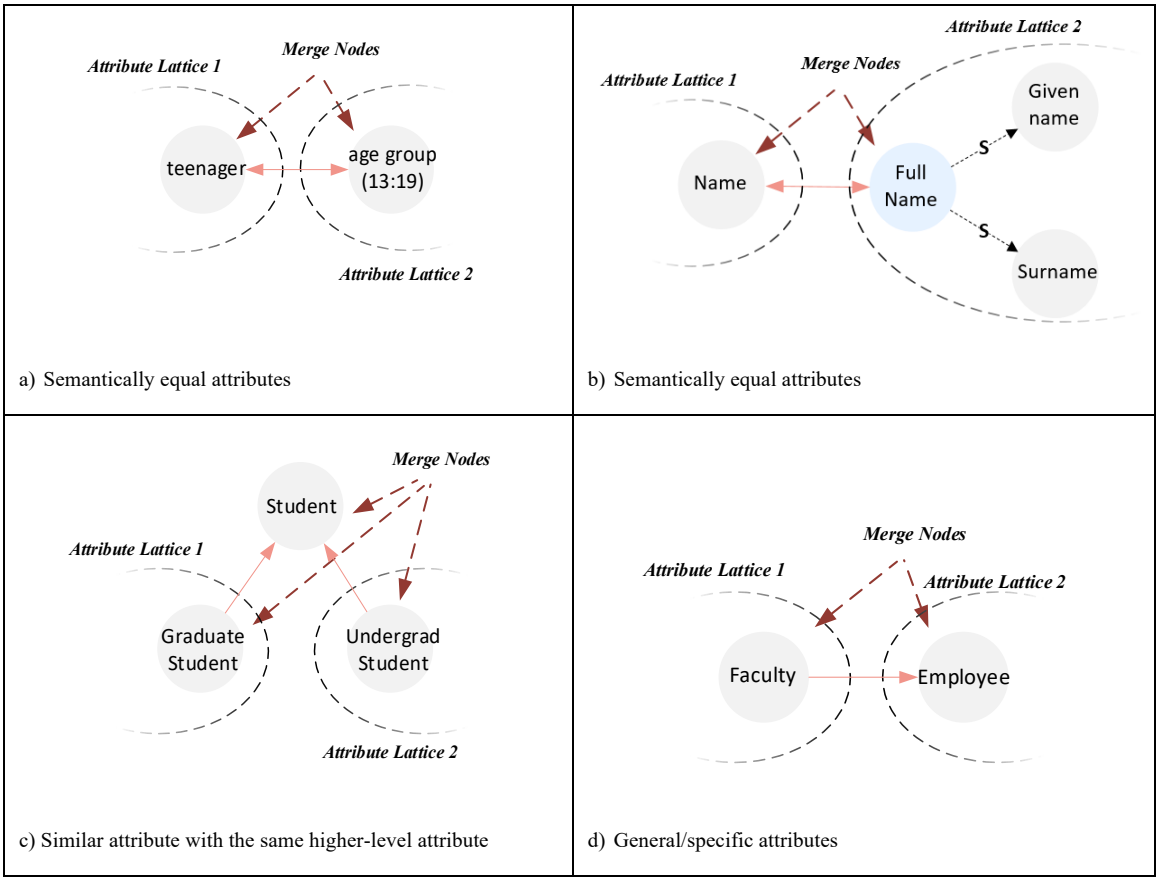
this definition, three types of similarity are envisioned – that is, semantically equal attributes, attributes that are a manifestation of the same higher-level attribute, and general/specific attributes.

First, attributes in distinct lattices are similar when they are semantically equivalent. That is, possessing an attribute in the first lattice is semantically equal to possessing an attribute in the second lattice. For instance, possessing *teenager* attribute in one lattice is equal to say an instance possess the *age group (13:19)* in another one. It is worthwhile to note that categories can be added to attribute lattices for cases in which possessing multiple attributes (a set of attributes) are semantically equivalent. For example, attribute *name* in lattice A is similar to attributes *given name* and *surname* in lattice B. In this case, a category (such as *full name*) can be defined with *given name* and *surname* as its expansion. Next, *name* in lattice A can be merged to the *full name* in lattice B (Figure 8a and 8b).

Second, similarity also refers to attributes that are a manifestation of the same higher-level attribute. For instance, although being *graduate student* and being *undergraduate student* are not semantically equivalent, these attributes are similar. These attributes can be merged by introducing a higher-level attribute, *student* (Figure 8.c).

Finally, if an attribute in the first lattice is more general (or more specific) than an attribute in the second lattice, these attributes are similar. For example, the attribute *faculty* in the first lattice is similar to attribute *employee* in the second lattice. Although these two attributes are not semantically equal, the *faculty* is a manifestation (a more specific attribute) of the *employee* (Figure 8.d).

Similar attributes in distinct lattices can be merged to provide a unified view over data sources. ***Merge nodes*** in a federated attribute lattice refers to attributes that either 1) represent higher-level attributes introduced for merging similar attributes or 2) attributes connected to an attribute in another lattice either with semantic equivalency relationship or precedence relationship. The merge nodes are represented in Figure 8.



a) Semantically equal attributes

b) Semantically equal attributes

c) Similar attribute with the same higher-level attribute

d) General/specific attributes

Figure 8. Similar Attributes and Merge Nodes

2.5.2 Potential Merge Nodes in Attribute Lattice Integration

The majority of approaches in this area are manual (Spanos et al., 2012). However, semi-automated approaches exist for schema-based semantic data integration (such as Volz et al., 2004; Hu & Qu, 2007). Here, I argue the intrinsic characteristics of attribute lattice can be utilized to semi-automatically integrate conceptual models that represent the semantics of various data sources to provide a unified view for the semantic structure of the subject domain.

As discussed earlier (section 2.3.2), the immediate and semantic neighbourhood of attributes is defined to infer the class structure of the domain. In the following, lemmas that suggest potential similar attributes based on the known merge nodes are proposed. That is, given initial merge nodes in distinct lattices, these lemmas suggest new merge nodes based on the immediate and semantic neighborhood of known merge nodes.

Lemma 1: Let a_{1i} and a_{2j} be attributes in L_1 and L_2, respectively. Assume both attributes are either category or class. If a_{1i} is semantically equal to a_{2j}, attributes in their expansion are candidates to be similar and new merge nodes.

Explanation: Attributes a_{1i} and a_{2j} are either class, or category, that is, there are other attributes in the lattices which are semantically equal to them (their expansion). Since a_{1i} and a_{2j} are semantically equal, the attributes that define these attributes (their expansions) are potential candidates to be similar and new merge nodes. An example for this lemma is presented in Figure 9.

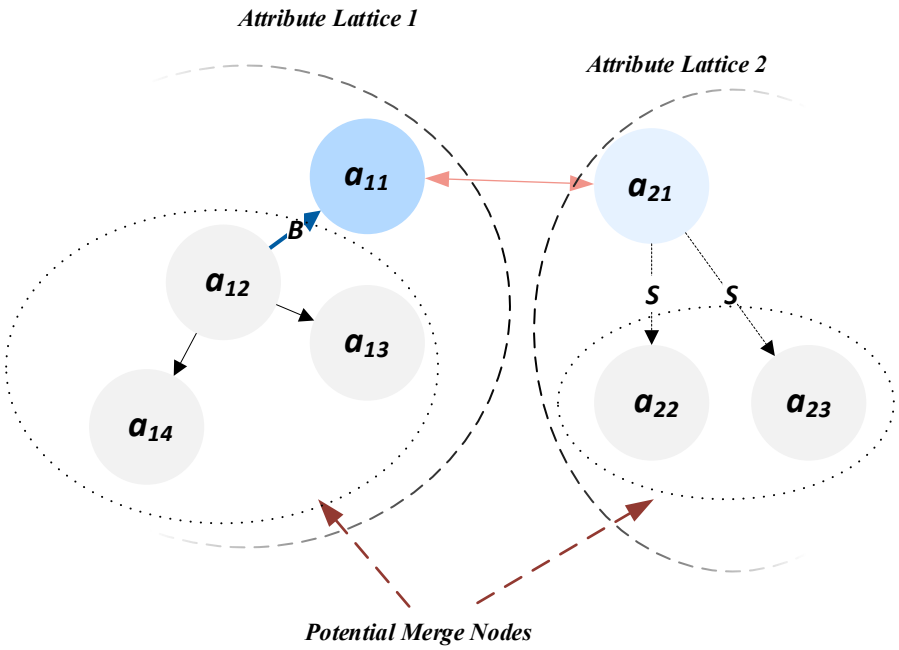


Figure 9. Potential merge nodes based on lemma 1

Lemma 2: Let a_{1i} and a_{2j} be attributes in L_1 and L_2, respectively. If a_{1i} is semantically equal to a_{2j}, attributes in their *semantic neighbourhood* with outgoing precedence relation ($SN^+(a_{1i})$ and $SN^+(a_{2j})$) are candidates attributes to be similar and new merge nodes.

Explanation: Attributes a_{1i} and a_{2j} are either class, category, or property. If attribute is a class or category, any attribute that can be inferred from its expansion can be inferred from the class or category itself. In all cases, if attributes in L_1 and L_2 can be inferred from semantically equal attributes (a_{1i} and a_{2j}), they are potentially merge nodes.

Figure 10 presents an example of this lemma. In this figure, a_{11} and $a_{21}$are two semantically equal attributes in distinct lattices. a_{11} is a class with three attributes in its expansion ($a_{11}^{E} = \{a_{12}, a_{13}, a_{14}\}$.) In the first lattice, a_{15} is inferred from a_{13}, and a_{13} is a part of a_{11} expansion, since possessing a_{11} is semantically equal to possessing $a_{12}, a_{13}, and\ a_{14}$, a_{15} can be inferred from a_{11}itself. In this example $a_{11}$and a_{21} are semantically equal attributes, and, a_{15} and a_{22} are inferred from these two equal attributes, respectively. Following lemma two, these two attributes (a_{15} and a_{22}) are potential new merge nodes.

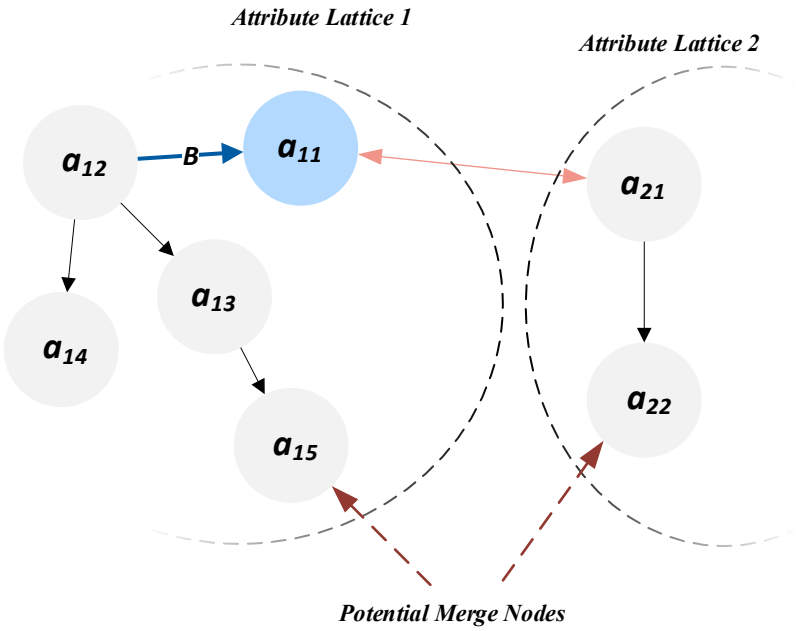


Figure 10. Potential merge nodes based on lemma 2

Lemma 3: Let a_{1i} and a_{2j} be attributes in L_1 and L_2, respectively. Assume both attributes are either category or class. If a_{1i} and a_{2j} are manifestations of the same higher-level attribute(m_k), attributes in their expansion are potential merge nodes.

Explanation: a_{1i} and a_{2j} are either a class or a category. As a result, these attributes can be represented as a union of other attributes in the lattice. On the other hand, these two attributes (a_{1i} and a_{2j}) are manifestations of the same higher-level attributes, which means a proper subset of their expansions are semantically equal (Figure 11).

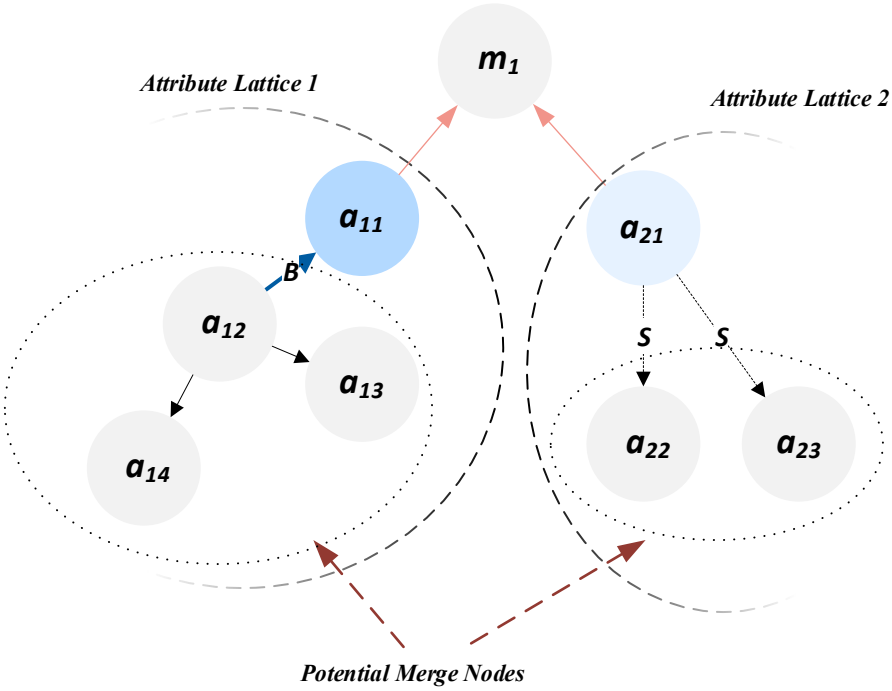


Figure 11. Potential merge nodes based on lemma 3

The attribute lattice integration procedure starts with an initial list of merge nodes and suggests new merge node based on known merge nodes. In this approach, attributes can be merged independent from their types (class, category, or property). However, the

immediate and semantic neighbourhood of attributes are utilized, through lemmas, to find new merge nodes.

To elaborate these lemmas, consider the following example adopted from Bergamaschi et al. (1999). Assume the following federated attribute lattice (Figure 12) represents the partial lattices related to patients in two departments in a given hospital - that is, intensive care department and cardiology department (showed by ID and CD pre-fixes, respectively.) Let assume the only known merge nodes are *patients* in both lattices.

Given these merge nodes, Lemma 1 suggests that attributes in their expansion are similar and potential merge nodes. Using this suggestion, *patient name* in cardiology department data set can be merged to *patient first name* and *patient last name* in intensive care department (excluded from Figure 12). Moreover, Lemma 2 suggests that *nurse* and *doctor* are potentially similar and potentially new merge nodes.

Let assume that user identified that *nurse* and *doctor* are actually similar and both are manifestations of a higher-level attribute, *hospital staff*. Lemma 3, using this new merge node, suggests that attributes in the expansion of *nurse* and *doctor* are potentially similar.

However, finding all potentially similar attributes and merge nodes could be a cumbersome and time-consuming task. Chapter 3 presents an artifact that utilizes these lemmas to facilitate the integration process.

To summarize, a key contribution of attribute lattice is semantic relativism. Semantic relativism makes possible to merge property attributes with category and class attributes. In this context, the similarity is defined in a broader sense that covers semantically equal

attributes, attributes that are manifestations of a same higher-level attribute, and specific/general attributes. Using attribute lattice characteristics, I proposed lemmas that can be utilized to find potential similar attributes based on the known merge nodes. In Section 3.4, details of the artifact developed to assist attribute lattice integration procedure are presented.

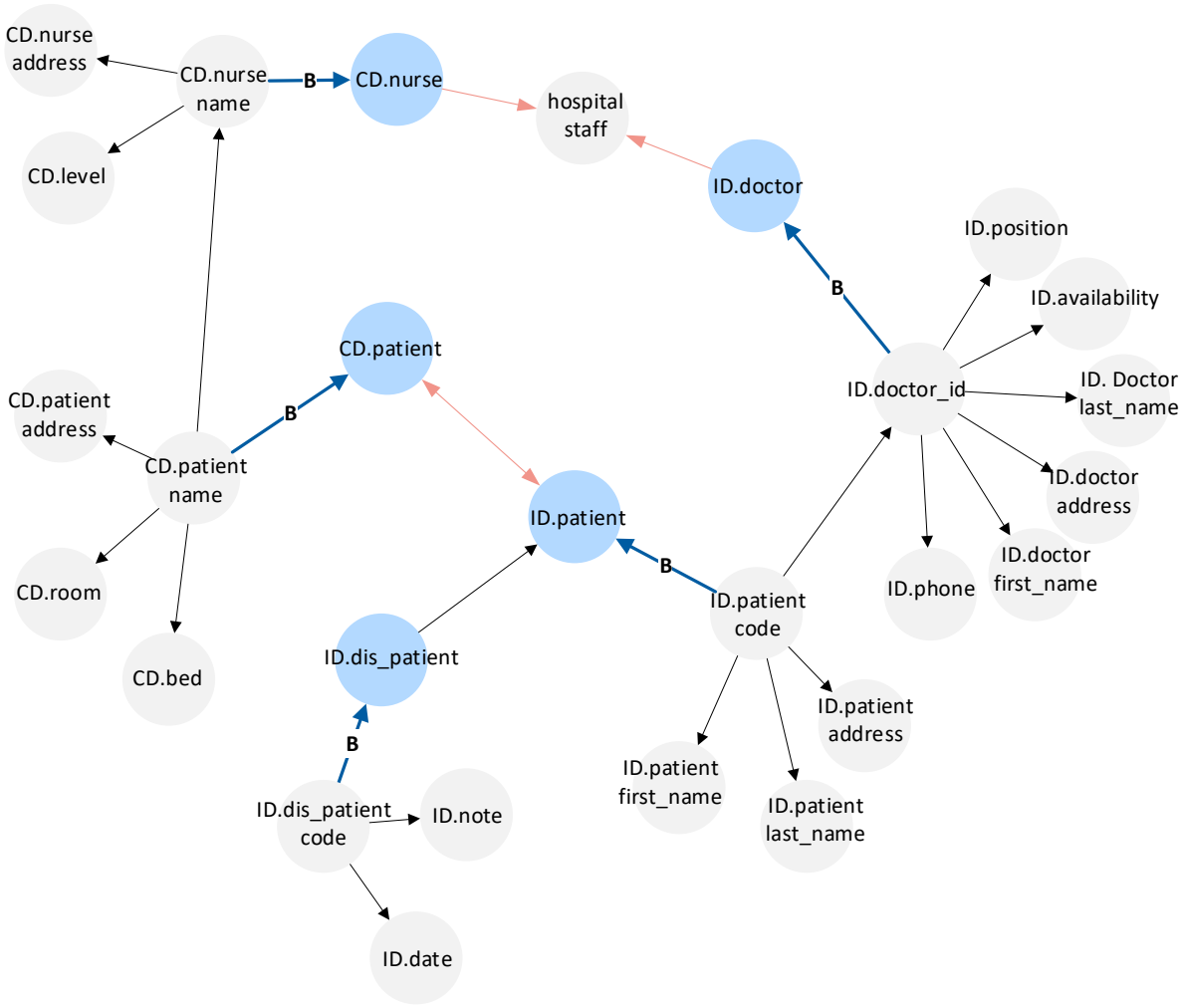


Figure 12. Lattices Integration. Adopted Form Bergamaschi et al. (1999)

2.6 Discussion

The traditional paradigm of information system development considers conceptual modeling as an important step in requirements engineering. This paradigm assumes that data is created, stored, and queried for pre-defined purposes. However, in the era of Big Data, this assumption often does not hold. In the current environment, users of data query analyze it for purposes beyond what data contributor might anticipate. Here, I argue that conceptual models can be utilized to understand the data semantics as perceived by users of data.

In this thesis, I use principles from cognitive psychology, philosophical ontology, and graph theory to define a lightweight graph-based conceptual modeling grammar, which I call attribute lattice grammar. The attribute lattice grammar has several important characteristics. First, it offers a minimal set of components designed to capture the class structure of the domain of interest. The attribute lattice grammar has two components: nodes and directed arcs. Nodes represent attributes – true statements about some instances in a domain – and directed arcs represent precedence relationships among attributes as perceived by users of data.

Second, it represents precedences as perceived by the user of data. Defined by using classification guidelines, three types of precedences are introduced in this conceptual modeling grammar. These precedence types reflect how the users of data perceive the relationship that exists between attributes. Hence, it is independent of the schema of the data source that it presents.

Third, an attribute lattice grammar supports classifications – that is, the pattern of precedences around attributes can be used to infer the class structure of the domain it represents. Following the assumption that the classes are constructed to provide useful abstractions of similarity, we use the pattern of precedences (and their types) to designate an attribute as a class, a category or a property.

Finally, an attribute lattice grammar provides a robust semantic foundation to integrate various data sources. A key challenge in semantic data integration is to find similar classes which refer to the same real-world concept. The attribute lattice grammar enables users of data to construct their classes over heterogeneous data sources. This uniformity of representation affords greater flexibility in viewing information (multiple perspectives can co-exist), which in turn supports integration.

The first theoretical contribution of the attribute lattice modeling grammar is utilizing conceptual models to understand the semantics of data sources that come from a developed information system, and independent from the schema of the data source. This grammar helps users of data to represent the class structure of based on the current purpose and sue.

The second contribution of this grammar is extending the concept of attribute precedence to capture the type of subsumption relationship among attributes more clearly. The base and subcategory precedences in this grammar convey semantics beyond the simple precedence relationship. These types, in turn, are utilized to define semantic relativism in an attribute lattice.

The third contribution is utilizing human view of classification to define ***semantic relativism*** – whether a node represents a class or property depends entirely on its semantic

neighbourhood, the pattern of incoming and outgoing arcs and nodes connected to these arcs. Semantic relativism frees instances from predefined (or undefined) schemas, and enables users to query and analyze data based on their understanding of it.

Finally, this chapter suggests an attribute-lattice-based approach to provide a unified view over distinct heterogeneous data sources. In this approach, attributes are similar when referring to the same characteristics of instances, or convey the same kind of characteristics in a different level. This approach iteratively finds similar attributes in separate data sources and joins them on the merge nodes.

This grammar provides several core uses. First, attribute lattices are well-suited for analysis. Given a lattice, it is possible to automatically analyze it to identify attributes that represents meaningful abstraction – that is, classes. This, in turn, enables data users to identify set of instances belonging to each class, and make inferences about instances using these classes. These analyses are incorporated in a tool for lattice visualization (Chapter 3) that enables users of data to view the data structure from various perspectives, thereby contributing to gaining insights from data.

Second, lattices provide a strong semantic foundation for data integration. In traditional approaches, an impediment to data integration is structural heterogeneity between independent data sources. However, this approach is built based on a class schema-free artifact (attribute lattice) and can be used for integration of data from structured (e.g., traditional relational databases), semi-structured (e.g., web data which has no rigid structure), and even unstructured data sources (e.g., text data).

3 Artifact Implementation

The previous chapter defined the attribute lattice grammar, its components, its characteristics, and a set of rules to validate a given attribute lattice. Also, it suggested an approach to find potentially similar attributes and to create a unified view over distinct lattices by joining the merge nodes. This chapter elaborates on the design and implementation of the artifact that supports attribute lattice. Here, the programming language and functions which are developed to implement the ***attribute lattice artifact*** are discussed. This implemented artifact, which is available to the research community and accessible on the web[6], enables a wide range of empirical studies on the usefulness and adoption of the attribute lattice.

This section presents a set of features implemented in the artifact that allows users to create, update, represent, validate, and integrate lattices. These features aim to: (1) enable users to create and manipulate an attribute lattice; (2) provide a graphical representation of attribute lattices; (3) represent the expansion of attributes; (4) validate a given attribute lattice; and (5) suggest similar attributes based on known merge nodes.

Later, section 4.4 explains how this artifact can be expanded for various attribute lattice-based use cases. In particular, it will discuss additional features that have been implemented to support attribute lattice-based topic modeling.

[6] The application is available at this address (https://attribute-lattice.shinyapps.io/thesis/)

Table 4 summarizes the list of features covered in this chapter. Also, Appendix A shows the user interfaces (screenshots) of the artifact.

Table 4. Artifact Features

Feature Category	***Main Features***
1. Lattice Operation	1.1 Provide a mechanism to store, and manipulate attributes lattices
2. Lattice Representation and Analysis	2.1 Illustrate an attribute lattice graphically
	2.2 Represent expansion of attributes
	2.3 Validate a given attribute lattice
3. Lattice Integration	3.1 Provide a mechanism to create a federated attribute lattice, and to merge nodes
	3.2 Suggest similar attributes based on known merge nodes

3.1 Programming language

Initially developed for statistical computation and graphics, ***R*** (R Core Team, 2000) and its packages offer powerful sets of tools to handle graphical representation of the lattice, lattice operation, and lattice integration. Therefore, R is used for the implementation attribute lattice in ***three tiers*** – that is, storage, logic, and representation tiers. A web-based structure has been used for implementation to make the attribute lattice available for future studies.

The attribute lattice is structurally simple, that is, an attribute lattice is a set of attributes and precedences relationships among them. Hence, there is no need to use database management systems to store attribute lattices data, and it can be stored in files directly. These files are saved into cloud storage by using Dropbox services in R (rdrop2[7]). A combination of R packages (including igraph[8], DT[9], and dplyr[10]) is utilized to develop required functions in the logical tier. Finally, the representation layer of the application is developed using the ***shiny*** (Chang et al., 2015) package. This package provides an interactive web-based environment for development.

The developed application has two panels – control panel and main panel (Figure 13). Located on the left-hand side of the screen, the control panel is used to capture users' inputs and commands - that is *lattice operation*, *lattice integration*, *similar attributes* and

[7] https://cran.r-project.org/web/packages/rdrop2/index.html
[8] https://cran.r-project.org/web/packages/igraph/index.html
[9] https://cran.r-project.org/web/packages/DT/index.html
[10] https://cran.r-project.org/web/packages/dplyr/index.html

plot adjustment. The main panel, with five tab panels (*lattice definition*, *plot*, *attribute structure*, *lattice validation*, and *lattice integration*), is used for attribute lattice representation purposes (graphical attribute lattice and report tables.)

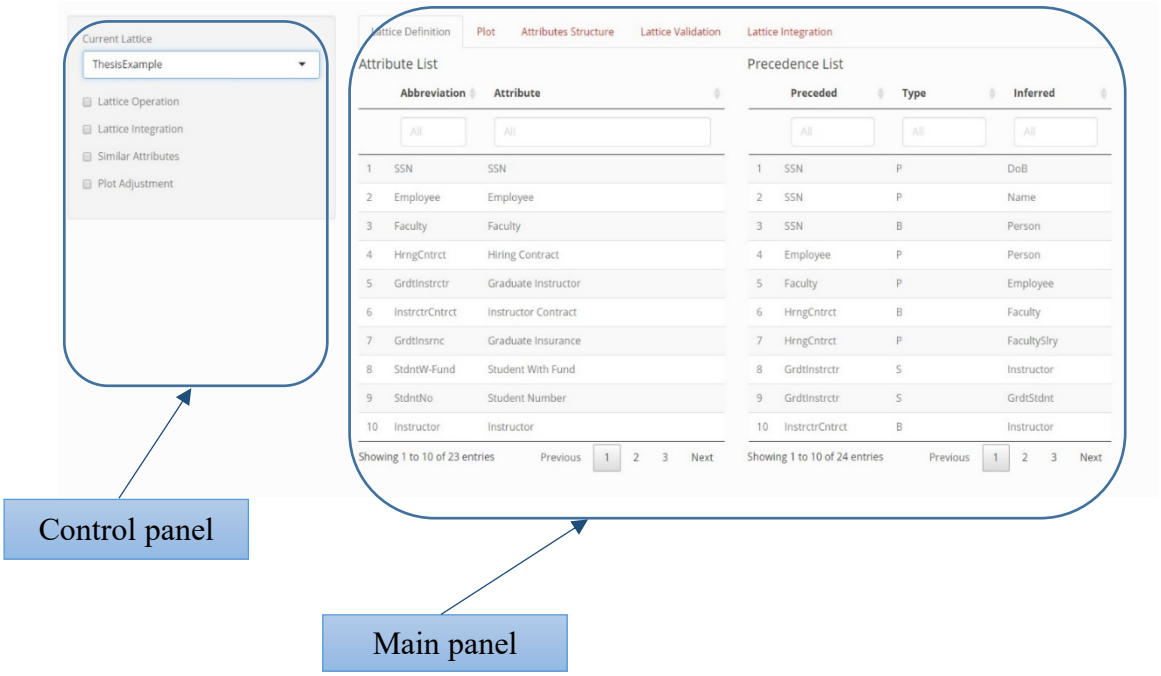


Figure 13. Control panel and main panel

3.2 Lattice Operation

3.2.1 Store, and manipulate attributes lattices

As discussed earlier (Expr 6), an attribute lattice is a set of attributes and precedences. An attribute is a true statement about at least one instance in the domain of interest. Long attribute names, in the attribute lattice, might make the graphical representation of the lattice unreadable. Hence, attribute abbreviations are also stored in the application. The *data-frame* object (R Core Team, 2000) with two columns (attribute itself, and its abbreviation) is used to store attributes in each lattice. Similarly, the data-frame object with three columns is used to store the precedence relationships, in which the first column stores the abbreviation of the preceded attribute, the second one stores the precedence type, and the last column stores the abbreviation of the inferred attribute.

The *list* object (R Core Team, 2000) is used to store each lattice. This *R* object can contain many different types of elements, such as vectors, and data frames inside it. A list object that stores a lattice in the application includes two data frame objects (R Core Team, 2000). The first one stores attributes with their abbreviations, and the second one stores precedences of each lattice. The logical model of an attribute lattice is demonstrated in Figure 14. A set of functions is developed to support lattice creation and manipulation. In the following these functions are discussed.

Attribute Lattice Name

Abbreviation	Attribute Label
Abb 1	Att Label 1
Abb 2	Att Label 2
Abb 3	Att Label 3
…	…

Preceded Attribute	Precedence Type	Inferred Attribute
Abb 1	S	Abb 2
Abb 1	P	Abb 3
…	…	…

Figure 14. The logical model of list object to store attribute lattice

- ***Save and load attribute lattices***

By default, the storage of a lattice is volatile and local. This means the attribute lattices are accessible just for the current user, and only while the session is active. In other words, as soon as the user closes the web browser, the existing lattices will be deleted. However, the application provides the capability to store a lattice to the server. At the application start-up, all server-saved lattices are accessible for users for manipulation, reasoning, and integration. Note that the application can store and manage multiple lattices; however, at any point, there is only one active lattice, and all the tab panels in the main tab represent different information about the active lattice.

- ***Create initial (empty) attribute lattice***

This function takes an attribute lattice name as input. Using the user provided name, it creates a list object (attribute lattice) with two data-frames in it – one for attributes in the lattice, and one for the precedence relationships in the lattice.

- ***Add attributes to the lattice***

This function takes an attribute name and abbreviation as input and adds a row to attributes data-frame of the active lattice. It is worthwhile to note that attribute abbreviation is utilized as an identifier for the attribute. That is, precedence relationships are stored, queried, and analyzed using attribute abbreviations.

- ***Edit attributes in the lattice***

This function takes an existing attribute and a new value for either attribute name or attribute abbreviation and updates the attribute information with the given value. As mentioned earlier, precedence relationships are stored by using attribute abbreviations. As a result, if the update happens on an attribute abbreviation, all the precedences will be updated by the new value.

- ***Delete a given attribute from the lattice***

This function is developed to delete an existing attribute. If an attribute has been used as a preceded or an inferred attribute in any precedence relationship, it cannot be deleted. In this case, this function returns an error to the user – "The attribute has been used in a precedence relationship and cannot be deleted".

- ***Add precedence to the lattice***

Given a pair of existing attributes and a precedence type, this function adds a precedence relationship to the current lattice by adding a row to the current attribute lattice precedence data-frame.

- ***Delete a given precedence from the lattice***

This function is developed to delete an existing precedence from the active attribute lattice.

- ***Edit a given precedence***

The precedence type of an existing precedence relationship can be edited by this function.

- ***Import attributes and precedence relationships from Excel file***

Using the above-discussed functions to add attributes and their precedence relationship one by one could be a cumbersome task. This function is developed to facilitate attribute lattice definition by adding both attributes and precedences from an Excel file. Note that, using this function, attributes and precedences can be added either to an initial (empty) attribute lattice, or lattices that already have attributes and precedences.

A predefined structure is defined for the excel file. There must be two pages in the file in which the first page has two named columns (*abbreviation* and *attribute*) and the second one has three named columns (*precededAbr*, *prcType*, and *inferredAbr*).

Given a file, this function first validates the file structure – that is, it verifies if the file is an excel file with two pages and above mentioned named columns. Then, it adds all attributes and precedences to the active attribute lattice. In case the function cannot find the attribute name of attribute abbreviations that have been used in precedence relationship definitions (neither in the excel file nor within existing attributes), it automatically adds an

attribute to the attributes data frame and sets attribute abbreviation as the attribute name as well.

The functions discussed above are available for users through the *lattice operation* section of the control panel. These functions enable users to create, store and modify an attribute lattice. At any point, the detail of a defined attribute lattice can be viewed in the *lattice definition* tab panel, which is located in the main panel. The lattice definition tab panel represents the list of attributes and their precedence relationships. Figure 15 shows the lattice definition tab panel for the attribute lattice discussed in section 2.3.4 (see Appendix A for user interfaces details).

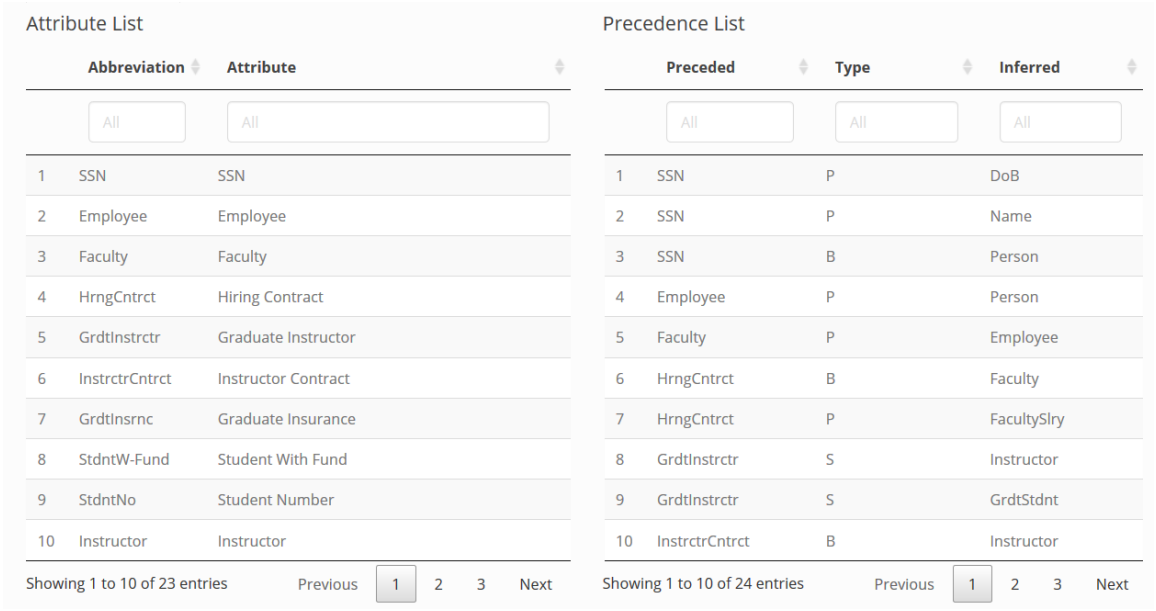


Figure 15. Lattice definition tab panel, in the artifact main panel

3.3 Lattice Representation and Analysis

An attribute lattice is a set of attributes and precedence relationships among them which can be represented in a graph-like structure. A set of functions developed to represent the lattice graphically, represent the lattice structure, and validate a given attribute lattice, is discussed in the following.

- ***Attribute type***

Given an attribute lattice, this function return type of all attributes in the lattice. If an attribute a_i has either incoming base precedence ($|N^{-dB}(a_i)| \geq 1$), or outgoing subcategory precedence ($|N^{+dS}(a_i)| \geq 1$) in its immediate neighbourhood, it will be designated as a class or a category, respectively. Otherwise, it will be designated as a property. This function returns a *list object* with three *vectors*, that is, classes, categories, and properties vectors.

- ***Illustrate an attribute lattice graphically***

This function is developed to represent an attribute lattice graphically. This function has two steps. First, it provides a list of arcs (precedence) with their types and nodes (attributes) with their types. Second, it passes the arcs and nodes to the *igraph* (Csardi & Nepusz, 2006) package for graph representation.

As discussed in Expression 12, three type of precedence relationships exist in the attribute lattice. These three types of precedence are represented in the lattice with directed arcs with various styles (Figure 1). The same styles have been used in the application to

represent precedences. This function passes precedences, their type and their style to the igraph package. Moreover, using the above-discussed function (attribute type function), this function creates a list of all attributes, with their types and their styles and passes them to the igraph package as well.

To draw the attribute lattice, the Fruchterman-Reingold layout algorithm (Fruchterman & Reingold, 1991) has been used. This heuristic algorithm, which keeps the distance of attributes as equal as possible and minimizes crossing edges, provides an intuitive representation for the attribute lattice. Figure 16 shows the plotted lattice in the application for the lattice introduced in section 2.3.4.

This function provides the capability to adjust the plotted attribute lattice as well. Attributes (nodes) are represented by circles in the lattice, with an attribute name written inside the circle. To improve graph clarity, node and font size of attributes can be adjusted using *plot adjustment* section in the control panel.

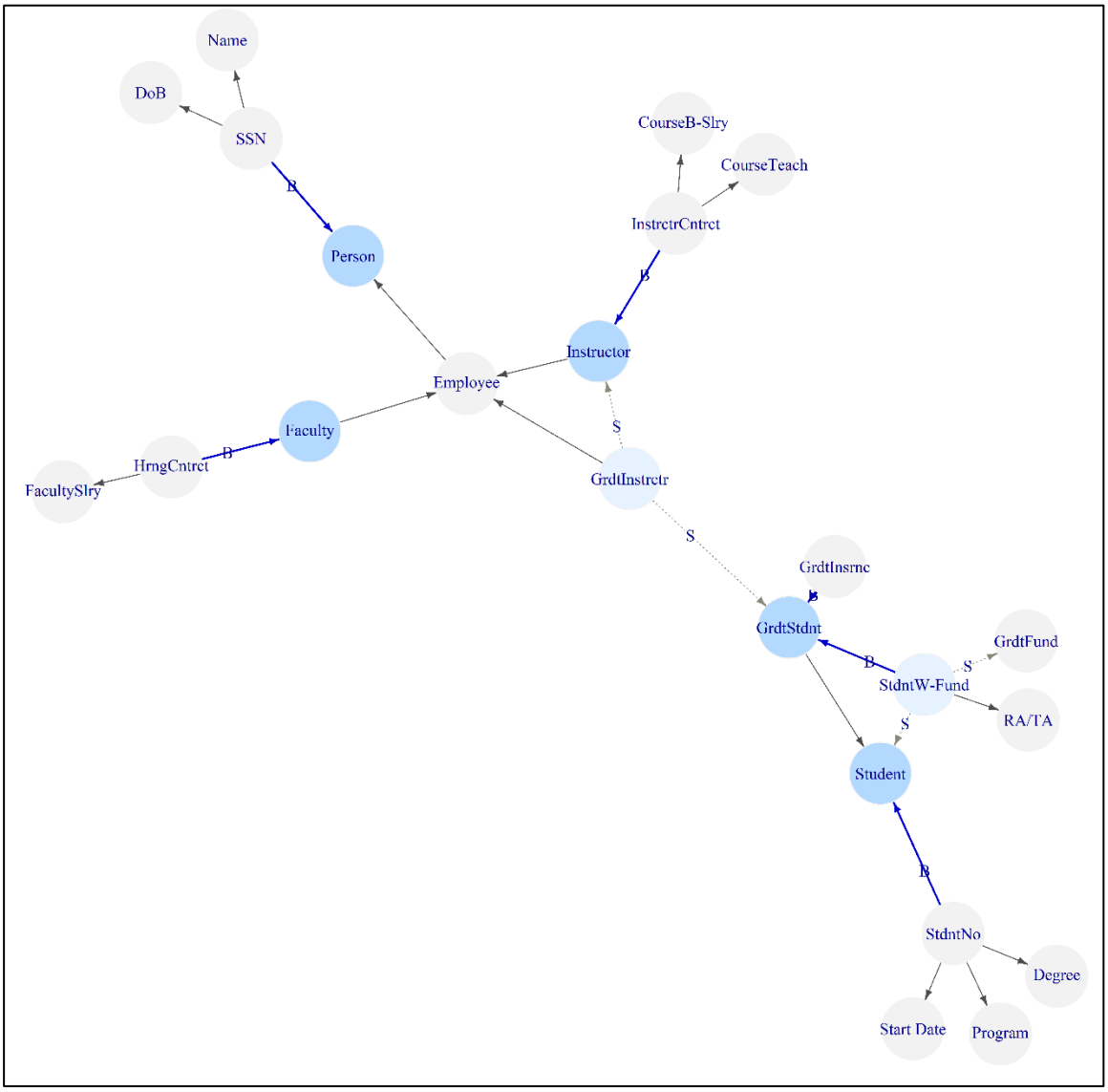


Figure 16. Graphical representation of Attribute lattice

- ***Semantic relations of attributes***

This function summarizes the semantic relationships for each attribute a_i - that is, it extracts all attributes which are related to the given attribute a_i, either directly or through its expansion (as elaborated in Expression 19).

- ***Attribute expansion***

Given an attribute lattice, using the third characteristic of attribute lattice (section 2.3.2), this function returns the expansion of class and category attributes. First, this function finds the expansion of category attributes by following subcategory precedences in their immediate neighbourhood (Expr 21). Second, it identifies all bases for each class attribute and uses these bases to determine inferred attributes for each base (Expr 22). The result of this function, for the active attribute lattice, is represented in the *attributes structure* tab panel in the main panel (Figure 17).

	Attribute	Type	Base	Expansion
	All	All	All	All
1	Faculty	Class	HrngCntrct	HrngCntrct, FacultySlry
2	GrdtStdnt	Class	GrdtInsrnc, StdntW-Fund	GrdtInsrnc, StdntW-Fund, RA/TA
3	Instructor	Class	InstrctrCntrct	InstrctrCntrct, CourseB-Slry, CourseTeach
4	Person	Class	SSN	SSN, DoB, Name
5	Student	Class	StdntNo	StdntNo, Program, Start Date, Degree
6	GrdtInstrctr	Category		Instructor, GrdtStdnt
7	StdntW-Fund	Category		GrdtFund, Student

Figure 17. Attribute structure

- ***Validate a given attribute lattice***

This function is developed to verify a given attribute lattice validity. Specifically, this function uses the functions described above to determine if the active attribute lattice violates any of the attribute lattice rules (discussed in section 2.3.3). This function has six steps (for six rules). After each step, it creates a data-frame object with two columns –the rule which is violated, and the message that represents how the pattern of precedences leads to the violation. At the end, this function adds up all data-frame objects and represent the result in the *lattice validation* tab panel in the main panel. Figure 18 shows the result for the example discussed in section 2.3.4.

	Validation Rule	Note
	All	All
1	Rule 1	Multiple precedence relationship exist between StdntNo and Student.
2	Rule 2	Faculty cannot be a class and category at the same time.
3	Rule 3	No attribute can be inferred from GrdtStdnt bases.
4	Rule 4	Faculty should be preceded (subcategory precedence) by at least two attributes.
5	Rule 4	GrdtInstrctr should be preceded (subcategory precedence) by at least two attributes.
6	Rule 5	GrdtInstrctr --P-> Employee is a redundant precedence.
7	Rule 6	FacultySlry --P-> Faculty is a redundant precedence.

Figure 18. Validating a given attribute lattice

3.4 Lattice Integration

The application supports attribute lattice integration through the following three functions. The first function creates an initial federated lattice, the second function captures the known merge nodes, and the third suggests similar attributes based on the known merge nodes.

- ***Create initial federated attribute lattice***

This function takes two attribute lattices and a name for the federated lattice and creates a federated attribute lattice by first, adding a prefix to attributes abbreviations (*l1* and *l2*) and second, adding all attributes and precedences from two attribute lattices.

- ***Capture merge nodes***

This function takes two attributes (from distinct lattices in the federated attribute lattice) and a merge type (Figure 8) as an input and creates a corresponding merge node in the federated attribute lattice. If two attributes are semantically equal, this function adds a row to the precedence data-frame of federated attribute lattice which represents semantic equivalency (represented by a line with arrows at both ends). If one attribute is more general (specific) than the other one, this function adds a precedence relationship (a row) between these two similar attributes with a different style. And finally, if both attributes are manifestations of a higher-level attribute this function asks for a name for this higher-level attribute and creates a new merge node. This new merge node precedes both given attributes.

- ***Suggest similar attributes based on known merge nodes***

This function suggests potentially similar attributes based on known merge nodes. Following Lemma 1, if merged nodes are semantically equal and both are either class or category attributes, attributes in their expansion will be suggested as new similar attributes. Also, this function uses the 'semantic relations of attributes' function (discussed in section 3.3) to find attributes with outgoing precedence relationship in the semantic neighbourhood of merged nodes and suggests them as similar attributes (Lemma 2).

If given merged nodes are manifestations of the same higher-level attribute and both are either a class or a category attribute, attributes in their expansion will be suggested as potentially similar attributes based on the Lemma 3.

	Known Merge Nodes	Lemma	Suggested Attributes	Suggested Attributes
	All	All	All	All
1	l1.patient <-> l2.patient	Lemma1	l1.patntCode, l1.patntAddress, l1.patntLastName, l1.patntFirstName, l1.docId	l2.patientName, l2.patientAddress, l2.room, l2.bed, l2.nurseName
2	l1.patient <-> l2.patient	Lemma2	l1.doctor, l1.docLastName, l1.docAddress, l1.docFirstName, l1.position, l1.availability, l1.phone	l2.nurse, l2.nurseAddress, l2.level
3	l1.doctor --> Hospital Staff; l2.nurse --> Hospital Staff	Lemma3	l1.docId, l1.docLastName, l1.docAddress, l1.docFirstName, l1.position, l1.availability, l1.phone	l2.nurseName, l2.nurseAddress, l2.level

Figure 19. Using lemmas to suggest attributes that are candidates to be similar

Finally, this function summarizes known merge nodes and potentially similar attributes in a data-frame object. The result of this function for the example discussed earlier (section 2.5.2) is represented in Figure 19.

3.5 Discussion

This chapter describes the design and implementation of a web-based, publicly available artifact that supports the attribute lattice conceptual modeling grammar. This application provides significant benefits for users who wish to utilize the proposed conceptual modeling grammar.

First, it represents graphically models created by the attribute lattices grammar. This graphical representation will help users to gain insight into the data semantics of the domain that the model represents.

Second, as discussed in the previous chapter, class bases and attribute expansions explain inferences that can be made for instances in the domain. The implemented artifact demonstrates the type of attributes, and their expansions by analyzing patterns of precedences around attributes. Hence, it enables users to understand instance-related inferences. Additionally, this artifact enables users to manipulate the model by adding precedences to or removing precedences from the model. Through these manipulations, users can conduct what-if analysis. That is, they can learn how changes in precedences will affect instance-related inferences.

Further, the implemented artifact enables users to validate the captured class structure of the domain. Users capture the precedences in the domain of interest, and these precedences constitute an attribute lattice. The artifact enables users to examine if the captured precedences create a meaningful class structure - that is, an inferred class structure from the pattern of precedence represents a meaningful class structure of the subject domain.

Finally, this application enables attribute lattice-based semantic data integration. Given a set of known merge nodes, this artifact uses integration lemmas (section 2.5) to suggest potentially new merge nodes. Hence, it enables users to create a federated view over distinct lattices.

4 Attribute-lattice-based Topic Modeling

4.1 Introduction

The Big Data era brings new challenges for meaningful use of data, such as the increasing number and heterogeneity of data sources (Dong & Srivastava, 2013). In contrast to traditional approaches, which mainly focus on structured data, with the explosive number of unstructured data sources there is a growing interest in integrating unstructured data (mostly text) for data-driven decision making (Russom, 2011). LaValle et al. (2011) argue that more and more strategic information comes from unstructured data sources such as social media. With the ever-increasing amount of data, the challenge is to understand the content of unstructured data and to find relevant information.

The notion of attribute lattice is defined in Chapter 2 as a schema-free conceptual modeling grammar. The attribute lattice has a simple structure and provides a foundation for representing the semantics of the data source. However, building a lattice for unstructured data from the beginning can be a challenging and cumbersome process. This chapter aims to elaborate how: 1) an initial lattice can be created automatically by analyzing the content of unstructured data sources, and 2) this model can be used to find and retrieve relevant information.

This chapter utilizes the concept of attribute lattice grammar for the task of topic modeling of unstructured data – representing the latent semantics of unstructured data. Twitter, the most popular micro-blogging site, which generates a massive amount of text

data every day, has been selected as the unstructured data source for this study. The attribute-lattice-based topic model, built based on content analysis of tweets, provides valuable insight into the topic structure of tweets, and helps to identify related tweets and retrieve the most relevant tweets on the topic of interest.

In the following, I begin by reviewing topic visualization approaches, and research in which hashtags have been utilized for Twitter analysis. Then, I examine the use of attribute lattice grammar principles for the task of topic modeling. This is followed by illustrating an artifact implemented for the task of Twitter topic modeling. Next, the result of an empirical study in the domain of technology is presented. Finally, contributions, limitations, and opportunities for future research are discussed.

4.2 Related Literature

Tweets have several properties such as user, geo-location, number of replies and number of retweets. Also, tweets might include mentions, hashtags, and URLs to external pages. Among the properties, hashtags provide valuable insight into the topic of a given tweet. Given a set of tweets, attribute-lattice-based topic modeling aims to ***conceptualize*** the ***topic structure*** of tweets by utilizing tweets' ***hashtags*** and user-defined topics. In this regard, after discussing general approaches for analyzing Twitter the review in this section focuses on ***topic modeling***, ***topic visualization*** approaches, and approaches that use ***hashtags*** for Twitter analysis.

4.2.1 Twitter Analysis

Analyzing text data - all data in the form of natural language text - is an active research area in both Information Retrieval (IR) and Text Mining (TM) research communities. Although these two areas of research address text analysis from two different points of view, research conducted in these areas is highly related. In general, IR aims to find text documents from large collections that satisfies an information need (Manning et al., 2008), whereas the goal of TM is to extract and discover useful, actionable knowledge (Zhai & Massung, 2016). IR and TM can be considered as two steps of finding relevant text data from an extensive collection of text data (Zhai & Massung, 2016). The explosive growth of tweets, a short text data, makes it impossible for people to retrieve all data related to their interest. As a result, a wide range of IR and TM approaches have been applied to

Twitter for various analyses (Zimmer & Proferes, 2014), such as topic identification, event detection, sentiment analysis, and user analysis.

4.2.2 Topic Modeling and Topic Classification

Both supervised and unsupervised techniques have been employed for analyzing the topic of text data. *Topic modeling* approaches utilize unsupervised or weakly supervised techniques. On the other hand, *topic classification* approaches employ supervised techniques. Here, I review key research in both categories and point out how this research uses predefined topics for analyzing topics of text data.

Topic modeling in the IR and TM literature refers to unsupervised and weakly supervised mining methods that find latent topics in text documents. Probabilistic Latent Semantic Analysis (PLSA) (Hofmann, 1999), Latent Dirichlet Allocation (LDA) (Blei et al., 2003), Correlated Topic Modeling (CTM) (Blei & Lafferty, 2005), and labeled LDA (Ramage et al., 2009) are the most well-known methods for topic analysis. Generally speaking, these techniques reveal the latent topics by implicitly capturing word co-occurrence patterns at the document level (Yan et al., 2013).

The result of topic modeling tasks is the identification of topic words in given documents and the coverage of these topics in each document (Zhai & Massung, 2016). Tweets, which contain less than 280 characters (140 characters before Nov 2017), and shortened words that are not like standard written English, presented new challenges to topic modeling. As a result, several PLSA, and LDA extensions have been proposed (e.g., Hong & Davison, 2010). To tackle these new challenges, PLSA and LDA extensions usually assume that each tweet has only one topic (e.g., Zhao et al., 2011), and incorporate additional

tweets' properties such as user information (e.g., Hong & Davison, 2010; Diao et al., 2012), and time period of tweets (e.g., Diao et al., 2012).

Unsupervised topic modeling approaches represent each topic as a set of topic words (a bag of words). However, these sets of words do not have explicit semantics (Lau et al., 2010; Sun et al., 2015); as a result, these approaches suffer from the lack of semantic interpretability for human users (Chang et al., 2009). Hence, in weakly supervised approaches, a common practice for labeling the topics is to map latent topics to a set of predefined topics. Ramage et al. (2010), for instance, map the contents of tweets into four classes (i.e. substance, social, status, and style topics). Zhao et al. (2011) assigns the result of topic analysis to a set of predefined topic categories by extending LDA approach for Twitter and comparing tweet topics with traditional media topics.

Topic classification approaches are supervised topic modeling techniques, similar to weakly supervised approaches that map tweets to a set of predefined topics. Yang et al. (2014) define approximately 300 topics in a hierarchical structure with six levels. Yang et al. (2014) use a spectrum of topic modeling techniques and crowdsourced labelers to classify tweets. Sriram et al. (2010) proposed to use domain-specific features extracted from author profile of tweets (eight features) to classify tweets to five general, high-level categories (i.e., news, events, opinions, deals, and private messages). And finally, Lee et al. (2011) employ both tweets and Twitter users (tweeters[11]) network models to classify trending topics into 18 general categories.

[11] In this thesis, "tweeter" refers to the Twitter user who wrote the tweet, and "user" refers to an individual who uses the topic model for tweet retrieval.

In topic modeling and topic classification, independent of the employed technique, a common practice is to map tweets to a set of predefined topics. This approach restricts users in classifying and retrieving tweets – that is, users need to select the most relevant topic among a set of pre-defined topics to retrieve tweets.

4.2.3 Topic Visualization

Text visualization, as an important subfield of information visualization, has gained considerable attention in recent years (Liu et al., 2014a; Kucher & Kerren, 2015). The focus of text visualization techniques is to assist users in exploring, understanding and analyzing text data through visual representations (Liu et al., 2014a; Kucher & Kerren, 2015).

Several survey papers focus on characteristics of text and information visualization techniques (for example, Šilić and Bašić (2010); Liu et al. (2014a); Kucher and Kerren (2015)). A recent taxonomy (Kucher & Kerren, 2015) categorizes text visualization techniques based on six dimensions - that is, analytic tasks, visualization tasks, domain, data source and properties, and visualization type (dimensionality, representation, and alignment). The web-based version of this survey, as of June 2019, covers 440 text visualization techniques.

Topic visualization, a common analytics task in text visualization, improves user experience regarding topic exploration, and provides a better understanding of evolutionary patterns of the topics. In the context of Twitter, topic visualization practices either visualize the topic information statically (e.g., Liu et al., 2014b), illustrate topic evolution over time (e.g., Havre et al., 2000; Cui et al., 2014) or integrate both static and dynamic aspects of

topics (e.g., Dou et al., 2013). However, the semantics of the topics and their relationships are not the focus of the research.

4.2.4 Hashtags

A hashtag is a word that starts with "#" symbol on Twitter. Hashtags are tweeter-composed keywords for a tweet which can be included in tweets for various purposes. Approximately one-tenth of tweets have hashtags; however, tweets with hashtags tend to have more valuable information (Suh et al., 2010). Hashtags provide valuable insight into the topic of a given tweet. However, their semantic interpretability is hindered by characteristics such as shortened formation (Liu et al., 2011; Pöschko, 2011), conversational nature (Huang et al., 2010) and sparseness (Yang et al., 2012). In some cases, it can be hard or even impossible to understand the intended meaning of a given hashtag (Liu et al., 2011; Pöschko, 2011). This happens not only because of words with multiple meanings (homonyms) but also because words may have been written in a shortened format to fit into character count of Twitter. For instance, #ttot (a highly adopted hashtag in the Travel domain) does not convey meaning clearly by itself.

The use of hashtags has attracted considerable interest and research attention. The research in this area widely can be categorized into three topics. First, there is research in which hashtags have been used for IR/TM tasks. For instance, Lin et al. (2011) use hashtags for topic filtering, Livne et al. (2011) for community analysis, and, Liu et al. (2011) for topic summarization. Second, some research addresses the low rate of hashtag adoption by tweeters and offers hashtag recommendation (e.g., Godin et al., 2013). Finally, research explores the characteristics of hashtags and how/why tweeters adopt hashtags. For instance,

Huang et al. (2010) look at the time series chart of traditional tagging platforms (such as Delicious) and hashtags, and asserts that tagging nature of Twitter is different from traditional tags. Traditional tags have been widely utilized for organizational purposes whereas hashtags have been used for conversational purposes, and filtering and directing the tweets. Yang et al. (2012) argue that tweeters adopt hashtags to indicate their membership in a community.

To summarize, hashtags are a potentially valuable source of information for topic modeling and information retrieval. However, the semantic interpretation of hashtags needs to be addressed to exploit this potential.

The proposed topic model aims to utilize attribute lattice grammar principles to improve semantic interpretation of hashtags (tweeter-defined topics) and to visualize semantics of topics and their relationships. Built based on tweeter-defined topics, this topic model (a graph-like conceptual model) represents the topic structure of a given set of tweets and enables users to incorporate their topics in the model. Hence, it frees users from pre-defined topics and enables them to use topics that reflect their interest for tweet retrieval and analysis purposes.

4.3 Attribute-lattice-based Topic Model

The attribute lattice is a schema-free conceptual modeling grammar. Developed based on IBDM principles, this model frees instances from predefined classes and offers classification that relies on the relationship among attributes. Classes, useful cognitive abstractions, can be defined and represented by the precedence relationship among the attributes possessed by instances. These classes provide support for information retrieval and semantic data integration. In this section, I first investigate how attribute lattice grammar principles can be used to represent the latent semantics of unstructured data (i.e., tweets). Second, the automated topic model extraction procedure is discussed. Finally, I illustrate how the topic model is utilized to represent the topic structure of the domain, to find related information from a vast amount of data, to retrieve more related information, and to gain insight into the meaning of unknown data.

4.3.1 Attribute-lattice-based Topic Model

In computational linguistic models, "topic" refers to a set of words that capture the latent semantics of a document (Newman et al., 2010). Users of these models can interpret the "topic label" by using the set of words in a topic (Mei et al., 2007). The topic label, in turn, refers to the concept that provides meaning to the set of words and makes the latent semantics explicit (Sun et al., 2015). Note that a topic is a set of words and should be distinguished from a topic label, which is a single word that makes the latent semantics explicit for users.

However, previous research suggests not all the words in a topic contribute equally to the interpretation of the topic label. In other words, a set of words (which I call representative words) in the topic can be found that contribute to this interpretation more than others do. These words best summarize the latent semantics of the document (Lau et al., 2010; Sun et al., 2015). For instance, Sun et al. (2015), given a set of topical words that need to be covered, used data-driven semantic network to find a minimum set of words that summarize the words in the topic and eventually extract the topic labels.

Twitter allows users to send tweets for all sorts of reasons in real time (Zhao & Rosson, 2009). The explosive growth of tweets makes them a valuable source of information. According to a recent statistic, by the end of 2017 Twitter, had 330 million monthly active users (Statista, 2018) who created an average 500 million tweets per day ("Twitter Usage Statistics," 2018). In addition to text, tweets may include mentions of specific users ("@" sign immediately followed by Twitter account), URLs to external pages, or user-specified hashtags (single words with the symbol "#").

Tweets cover a wide range of topics, from users' opinions to users' life events to emerging political news. As a result, topic identification of tweets has attracted intensive interest in recent years. Hashtags are tweeter-specified labels that convey the topic of the tweet. Tweeters adopt existing hashtags or create new hashtags to associate their tweets with other tweets with a similar topic. However, the shortened formation (Liu et al., 2011; Pöschko, 2011) and conversational nature (Huang et al., 2010) of hashtags hamper their semantic interpretation.

I argue the attribute lattice conceptual modeling grammar can be used to improve semantic interpretability of hashtags (tweeter-specified topics) and to create a topic modeling on Twitter. Although tweets (instances) possess several attributes such as author, mentions, hashtags, URLs, and geo-location, in developing the topic model, I only consider hashtags as attributes of tweets.

An attribute lattice can serve as a topic model to represent the topic structure of tweets in a subject domain. This topic structure shows topic labels (a hashtag or a word), topics (a set of frequent hashtags in the topic), and the representative hashtags for each topic label. In this topic modeling approach, a topic label (like a class in the attribute lattice) is a useful cognitive abstraction that meaningfully conveys the semantics of words in a topic. The representative words for a topic label are semantically equal to bases of a class in attribute lattice (i.e., they can be used to infer the topic). And finally, a topic (a set of frequent hashtags) is semantically equal to class expansion in an attribute lattice.

For instance, assume the technology domain is the domain of interest. Also, assume the set of hashtags, including *Bigdata*, *Analytics*, *BI*, *Hadoop*, *MachineLearning*, *DataScience*, and *IoT,* is the set of frequently used hashtags in Twitter to talk about this topic. For this topic, *Bigdata* can be a topic label, and *DataScience* and *Hadoop* can be representative hashtags – that is, if a tweeter adopts either *DataScience* or *Hadoop* in her tweet, it can be inferred that the topic label of the tweet is *Bigdata.*

The following section (4.3.2) proposes a semi-automatic approach to extract the initial attribute lattice from Twitter. This section aims to represent how an initial topic model can be created from data. This initial model can further be adjusted and modified by users

to include user-defined topics and relationships. This is followed by a discussion on the practical usefulness of the topic model on section 4.3.3. The section elaborates how this model enables users to predict the topics of emerging hashtags, facilitates finding domain-related terms/hashtags among trending terms/hashtags, and improves information retrieval.

4.3.2 Constructing the Topic Model

As discussed earlier (section 4.2.4), hashtags provide valuable insight into the topic of a tweet. However, they are extremely sparse and consequently not interpretable. Attribute-lattice-based topic modeling aims to utilize attribute lattice grammar principles to represent the topic structure of a given set of tweets on a subject domain. This topic model shows the topic structure by identifying the most influential hashtags in the set[12], conceptualizing the relationship among hashtags (attributes of tweets) and elaborating how these hashtags contribute to the user-specified topics in the domain of interest. In this section, I discuss how tweets related to a subject domain can be retrieved using the Twitter API, and how the attribute-lattice-based model is constructed using this set of tweets.

The proposed topic modeling approach starts with identifying and refining tweeters who are regularly tweeting about the topic of interest. Next, tweets of these tweeters are retrieved to calculate the ***popularity*** and ***frequency*** of hashtags related to the topic of interest. The popularity of a hashtag is defined as ***the number of unique tweeters*** who have adopted the given hashtag in the retrieved tweets. The frequency of a hashtag is defined as

[12] Note that if tweets are based on a highly specific subject domain (e.g., a specific fun activity or specific news category), a single hashtag might become dominant and the topic structure will become ineffective.

the number of original tweets (not retweets or quotes) that have used a given hashtag in the retrieved tweets. The popularity and frequency of co-occurrences of pairs of hashtags are defined in the same manner.

Finally, the topic model is constructed by comparing popularity and frequency of pairs with user-defined thresholds – that is, *popularity rate, precedence threshold, topic threshold, and frequency threshold* (Table 5). The first two thresholds (popularity rate and precedence threshold) are used to identify potentially meaningful subsumption relationship among hashtags. The popularity threshold refers to the minimum number of tweeters who need to adopt a pair to consider the pair as a meaningful co-occurrence. This threshold depends on the number of tweeters who contribute to the tweet set. Hence, it is calculated by multiplying the user-defined popularity rate by the number of tweeters who contributed to the topic model. The precedence threshold refers to the minimum value of a conditional probability that defines a meaningful subsumption relationship among hashtags in a pair.

The next two thresholds (topic threshold and frequency threshold) are used to suggest potential topic labels, and to find the representative hashtags for each topic. The topic threshold refers to the minimum number of hashtags that should co-occur with a given hashtag to consider the given hashtag as a topic label. For each topic label, a *minimum frequency* is defined by multiplying 'the frequency of the most frequent hashtag for the topic label' by the 'frequency threshold'. The most frequent hashtag and hashtags with a frequency higher than the minimum frequency are potential representative hashtags for the topic (Table 5).

Table 5. A summary of user-defined thresholds

Thresholds	***Definition***
Popularity threshold	The minimum number of tweeters need to adopt a pair to consider it meaningful
Popularity rate	A user-defined ratio to calculate the popularity threshold (popularity threshold is equal to the number of tweeters in the tweets set multiplied by the popularity rate)
Precedence threshold	The minimum value of a conditional probability that defines a meaningful subsumption relationship
Topic threshold	The minimum number of hashtags that should co-occur with a topic label
Frequency threshold	A user-defined ratio to find representative hashtags of each topic

The following six steps elaborate the procedure (Figure 20).

4.3.2.1 Searching for known tweeters

Based on the assumption that Twitter users (tweeters) tend to tweet about a narrow range of topics (Sriram et al., 2010; Yang et al., 2014), the first step in the procedure is to find tweeters who are known for frequently tweeting about the topic of interest.

Given a topic of interest, two approaches can be used to identify known tweeters; the published peer-reviewed list of tweeters (for instance Borison (2014)) and/or the "Twitter List." The former approach (using a peer-reviewed list of tweeters) might have two limitations. First, they usually cover a limited number of tweeters for each topic of interest (ranging from 10 to 100 tweeters.) Second, considering the growth rate of Twitter, the list might become obsolete quickly. The Twitter list provides a social annotation (Bao et al., 2007) mechanism to find tweeters who are known for tweeting about specific topics (Yang et al.,

2014). The latter approach provides a more updated list of tweeters; however, the list is not peer-reviewed. For this procedure, I suggest utilizing both types of lists together.

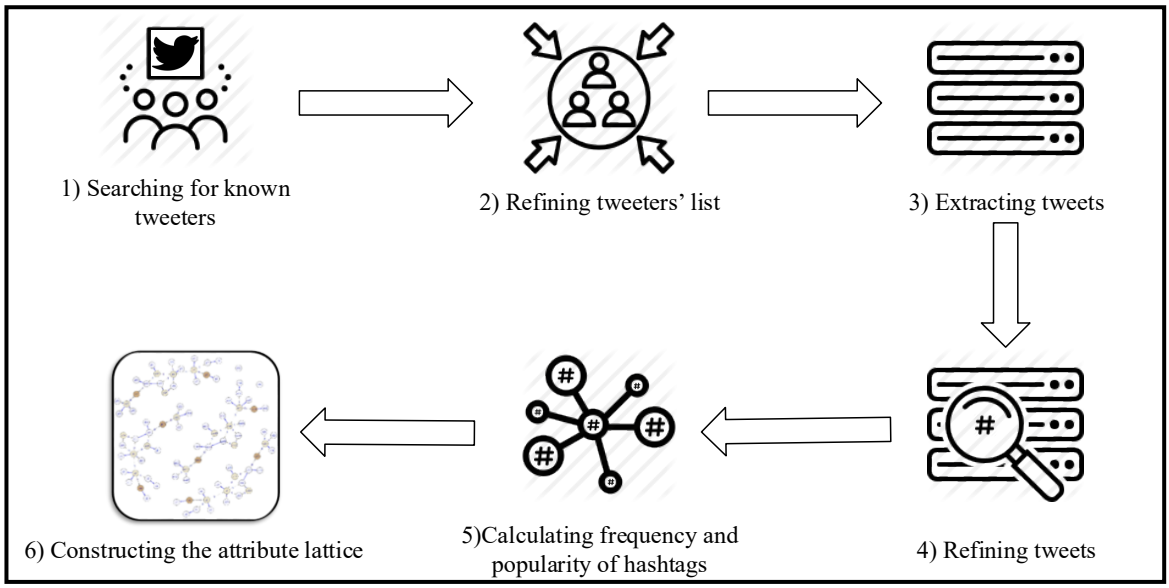


Figure 20. Twitter attribute-lattice-based topic modeling

4.3.2.2 Refining tweeters' list

The Twitter API provides access only to the tweets of public accounts. As a result, the initial list of tweeters should be pruned to include only tweeters whose accounts are publicly available.

4.3.2.3 Extracting tweets

Tweets of known tweeters can be retrieved at least by two approaches: borrowing tweets from previously published and available studies, or extracting directly from Twitter by the Twitter API. The former approach has several disadvantages. First, tweets in these data sets will be limited in the time covered. Second, there is no guarantee we can find tweets of all tweeters in the list in a specific data set. Third, considering the high rate of tweets, an average 500 million tweets per day ("Twitter Usage Statistics," 2018), the data sets might be obsolete, and they cannot reflect a recent topic structure of the domain. The latter approach, on the other side, imposes some limitations to the tweet retrieval process. The Twitter API provides the last 3200 tweets for each tweeter. Also, the service offers only 32,000 tweets every 15 minutes. As a result, tweet retrieval will be a time-consuming process (less than three million tweets can be retrieved each day.) Although using the Twitter API imposes some limitations to tweet retrieval process, it provides the most recent tweets to build the topic model. Hence, I use the Twitter API to directly retrieve the latest tweets for the tweeters in the refined tweeters' list for the empirical study.

4.3.2.4 Refining tweets

We are interested in tweets written in English, and tweets which have at least one hashtag. Therefore, I will discard non-English tweets and tweets without hashtags.

4.3.2.5 Calculating frequency and popularity of hashtags

This step aims to summarize data (tweets) based on co-occurrences of hashtags and to calculate the frequency and popularity of hashtag pairs. The output of this step is a summary table that contains pairs of hashtags and the popularity and frequency of each pair. This table is used in the next step to estimate the conditional probability of hashtags and to create the attribute-lattice-based topic model. Note that, for tweets with more than two hashtags, all the pair combinations will be used to calculate the popularity and frequency. For instance, if a given tweet has three hashtags (#h1, #h2, and #h3), the summary table will include three pairs for this tweet {#h1,#h2} , {#h1,#h3}, and {#h2,#h3}.

4.3.2.6 Constructing the attribute lattice

The last step in the procedure is to extract suggested precedence relationships among hashtags. The precedence relationship in the attribute lattice grammar reflects the subsumption relation among attributes. This relationship can be represented mathematically as follow:

Expr 27. $X\ subsumes\ Y\ If$:

$P(X|Y) = 1, P(Y|X) < 1;$

$Where\ a|b\ represents\ the\ conditional\ probability\ of\ a\ given\ b$

Previous studies adopted the simple conditional statistical model to construct hierarchical organization of concepts from text data (Sanderson & Croft, 1999) and to find subsumption relations among tags (Schmitz, 2006; Wu, 2015). Sanderson and Croft (1999)

argue a strict value ($P(X|Y) = 1$) will miss capture all meaningful pairs, hence, they propose to use relaxed conditional probability ($P(X|Y) \geq 0.8$) to construct hierarchical organization of concepts from text data (Expr 28).

Expr 28. $X\ potentially\ subsumes\ Y\ If$: *(Sanderson & Croft, 1999)*

$P(X|Y) \geq 0.8, P(Y|X) < 1;$

Schmitz (2006) adopted the same statistical model to extract subsumption relationship among Flickr tags. However, in this study, to filter meaningful tags, they use a lower limit both for the number of documents in which a given tag occurs and for the number of users that use a given tag, that is, D_{min}, and U_{min} respectively.

Expr 29. $X\ potentially\ subsumes\ Y\ If$: *(Schmitz, 2006)*

$P(X|Y) \geq t, P(Y|X) < t;$

$D_X \geq D_{min}\ ; D_Y \geq D_{min};$

$U_X \geq U_{min}\ ; U_Y \geq U_{min}$

$Where$:

$t\ is\ the\ co-occurrence\ threshold$

$D_X\ is\ the\ number\ of\ documents\ in\ which\ term\ X\ occurs, and\ must\ be\ greater\ than\ a\ minimum\ value\ D_{min},$

$U_X is\ the\ number\ of\ users\ that\ use\ X\ in\ at\ least\ one\ image\ tag, and\ must\ be\ greater\ than\ a\ minimum\ value\ U_{min}$

In this model, the author utilizes a fixed range for U_{min}, and D_{min} to find the most meaningful subsumption relationships among tags. The evidence from this experiment indicates, in comparison to the original model (Sanderson & Croft, 1999), the lower thresholds - that is, between 0.7 and 0.8 - provides more meaningful subsumption relationships.

Following the relaxed statistical model offered by Sanderson and Croft (1999), I use the same statistical model to extract potential meaningful precedences. In the context of Twitter, tweeters adopt existing hashtags, or, create new hashtags freely and without any constraints. As a result, hashtags are extremely sparse and we expect to see lower threshold ($t_{precedence}$) for the meaningfulness of precedences.

To identify meaningful tags, Schmitz (2006) offer to use a lower bound for each tag. I use the same principles to extract meaningful hashtags, however, the adopted model needs to be adjusted with the structural and semantical characteristics of hashtags (discussed in section 4.2.4).

First, to create a topic model, I assume tweeters tend to tweet about a narrow range of topics. Still, as reflected in the result of the experimental study, they might tweet about trending social topics which are not necessarily related to their topic of interest. For instance, #metoo is a popular hashtag among the tweeters who are known for tweeting about the domain of technology even though this hashtag is not related to the domain of technology. Hence, to eliminate hashtags that reflect topics that are not necessarily associated with the domain of interest, the statistical model should focus on the co-occurrences of hashtags. Note that, if tweeters adopt any hashtag, it is not an uncommon practice to adopt more than one hashtag. My empirical study (section 4.5.1) suggests that in the domain of technology, close to 65 percent of tweets have one hashtag and the rest have at least two hashtags.

Second, in contrast with Schmitz (2006), I argue the extent to which co-occurrence of hashtags contributes to the topic model in the domain of interest should be measured based on the ratio of known tweeters who adopt/use the hashtag rather than fixed lower

bounds. As a result, the lower bound for the meaningfulness of co-occurrence is defined based on ratio of popularity, which is *popularity rate (pop_{rate})*. This lower bound, minimum popularity (pop_{min}), is calculated by multiplying the popularity rate by the number of tweeters who contribute to the topic model. For instance, if the tweet set include tweets from one thousand tweeters ($no.of\ Tweeters = 1000$), with the pop_{rate} of one percent ($pop_{rate} = 0.01$), a co-occurrence is meaningful if it is adopted/used by at least ten tweeters ($pop_{min} = 1000 * 0.01 = 10$). Expression 30 summarizes the adopted statistical model.

Expr 30. $Hashtag\ B\ potentially\ precedes\ Hashtag\ A\ (A \rightarrow B)\ if$:

$$P(B \mid A) \geq t_{\ precedence}$$

$$P(B \mid A) > P(A \mid B)$$

$$pop_{A,B} \geq pop_{min}$$

$Where$:

$$pop_{min} = no.of\ Tweeters * pop_{rate}$$

$$t_{precedence}\ is\ a\ threshold\ for\ meaningful\ precedence$$

A key contribution of the attribute lattice grammar is semantic relativism. Hashtags (attributes) can be labeled as topic labels (classes) based on their semantic neighbourhood. As discussed earlier (section 4.3.1), a topic is a set of words that are likely to appear in the same context and the topic label provides a semantically clear and meaningful summary of the words in the set.

Given the above definition, any hashtag that co-occurs with a set of other hashtags in several tweets is a potential topic label for this set. In other words, the hashtag should be identified as a potential topic label if it precedes at least a minimum number of other

hashtags. The topic threshold (t_{topic}) is defined in the model to capture this minimum number that is required to identify a hashtag as a topic label. The number of identified potential topic labels depends on this threshold. A low threshold (i.e., $t_{topic} = 1$ or 2) identifies too many hashtags as potential topic labels. On the other hand, a more restricted threshold (i.e., $t_{topic} > 5$) misses capturing several potentially meaningful topic labels. Expression 31 represents the statistical model mathematically.

Expr 31. $Hashtag\ B\ is\ a\ potential\ topic\ label\ if$:

$\exists\ \{A_1, A_2, \dots, A_i\}\ a\ set\ of\ hashtags\ such\ that$

$\forall A_i\ in\ the\ set, A_i \rightarrow B, \quad and\ i \geq\ t_{topic}$

$Where$: $t_{topic}\ is\ a\ threshold\ for\ the\ minimum\ number\ of\ hashtags$ $that\ need\ to\ be\ preceded\ by\ hashtag\ B\ to\ label\ hashtag\ B\ as\ a\ topic.$

Given hashtag B as a topic label, following the above expression, the next step is to find base(s) of this topic. A base for a topic is a set of words that best summarizes the latent semantics. If a preceded hashtag frequently appears with the given topic, we can assume that the preceded hashtag provides a good summary of the topic; as a result, it is a base for the topic label. The frequency of pairs ($frq_{A,B}$) has been utilized in the model to identify potential base precedences. In the proposed approach, for any given topic label, the most frequent hashtag and hashtags with a close range of frequency are identified as bases for topic labels.

Suppose hashtag B is identified as a topic label, and $\{A_1, A_2, \dots, A_i\}$ is a set of hashtags preceded by hashtag B. Also, assume $\{frq_{A_1,B}, frq_{A_2,B}, \dots, frq_{A_i,B}\}$ is a set that shows the frequency of pairs. A_m, the hashtag with the highest pair frequency ($frq_{max} =$

$frq_{A_m,B} = \max\{frq_{A_1,B}, frq_{A_2,B}, \dots, frq_{A_i,B}\}$), is deemed a base for the topic. Also, if there exists another hashtag, A_n, such that $A_n \in \{A_1, A_2, \dots, A_i\}$ with a pair frequency close to the maximum frequency, it will be deemed a base for the topic label as well. Frequency threshold ($t_{\text{frequency}}$) is defined to measure the closeness. The frequency of a pair of hashtags considered to be close to the maximum frequency if it is higher than the frequency threshold multiplied by the maximum frequency. For instance, let the frequency of the most frequent pair be one thousand ($frq_{max} = 1000$), and the frequency threshold be 0.8, a hashtag with the pair frequency of 850 has frequency close to maximum frequency, and it is a base for the topic label ($\max\{frq\} * t_{\text{frequency}} = 1000 * 0.8 = 800$). The following expression represents the mathematical model to extract potential bases for topic labels.

Expr 32. $Hashtag\ A_k\ is\ a\ potential\ base\ for\ topic\ label\ B\ \left(A_k \xrightarrow{B} B\right) if$:

$B\ is\ a\ potential\ topic\ label$

$A_k \in A$

$frq_{A_k,B} \geq frq_{min}$

$Where$:

$A = \{A_1, A_2, \dots, A_i\}\ is\ a\ set\ of\ hashtags\ preceded\ by\ hashtag\ B$

$frq_{max} = max\ \{frq_{A_1,B}, frq_{A_2,B}, \dots, frq_{A_i,B}\}$

$frq_{min} = frq_{max} * t_{frequency}$

Finally, the last step is to find inferred hashtags for each base. Assume, hashtag B is identified as a potential topic label in the model, $A = \{A_1, A_2, \dots, A_i\}$ is a set of hashtags that preceded by this topic label, and $\{A_{k1}, A_{k2}, \dots, A_{kj}\} \subset A$ is a set of attributes identified as bases for the topic label using Expression 32. Preceded hashtags that are not identified

as a base ($A - \{A_{k1}, A_{k2}, \dots, A_{kj}\}$) are qualifying hashtags for the topic label – the topic label can be inferred from them, but they are not part of the topic. However, some of these hashtags might be, indeed, part of the topic. In this step, we are interested to find attributes that are potentially part of the topic among these hashtags.

For instance, assume *bigdata*, *opendata*, *hadoop*, *datascience*, and *analytics* are five hashtags in the tweet set. Suppose, *bigdata* precedes other four hashtags (i.e. *datascience, hadoop*, *opendata*, and *analytics*), *bigdata* is identified as a potential topic label, and *analytics* is identified as a potential base for it. In this step, we are interested to find if *opendata*, *hadoop*, and *datascience* can be inferred from *analytics*, and as a result, be part of the *bigdata* topic.

It is reasonable to assume that if a hashtag $A_m \in A - \{A_{k1}, A_{k2}, \dots, A_{kj}\}$ is a frequent hashtag in a given topic (part of the topic expansion), there is a higher chance that tweeters adopt hashtag A_m and hashtag B (topic) for positioning their tweets rather than hashtag A_m and A_{kj} (any of the bases). For instance, assume *datascience* is part of the *bigdata* topic. I assume there is a higher chance for tweeter to adopt *datascience* with *bigdata* (the topic) rather than with *analytics* (the base).

Following this assumption, if hashtag A_m can be inferred from a base with a weak precedence relationship, the model considers A_m as a part of the topic. The weak precedence relationship is a precedence with the probability not higher than precedence threshold ($t_{\text{precedence}}$). The following expressions represent the mathematical model to extract inferred hashtags for each base.

Expr 33. $Hashtag\ A_m\ is\ preceded\ by\ A_{kj}\ (A_{kj} \rightarrow A_m)\ if$:

$B\ is\ a\ potential\ topic\ label$

$A_m\ and\ A_{kj} \in A$

$A_{kj}\ is\ a\ potential\ base\ for\ B$

$P(A_m|A_{kj}) > 0$

$Where$:

$\{A_1, A_2, \ldots, A_i\}\ is\ a\ set\ of\ hashtags\ preceded\ by\ hashtag\ B$

While categories provide a mechanism to have shorthand access to a set of hashtags, users might arbitrarily group hashtags and create categories. Hence, at this point, this procedure has no step to identify and extract potential categories.

To summarize, the procedure starts with identifying meaningful precedences to construct a topic model from hashtag co-occurrences. Based on the semantic neighbourhood of each hashtag (attribute) in the model, the procedure identifies potential topic labels. Finally, the procedure identifies bases, and hashtags that can be inferred from each base. The function developed for this procedure is elaborated in section 4.4.1.

4.3.3 Topic Model Applications

There are several applications for the attribute-lattice-based topic model. First, it represents the topic structure of the domain. Second, it can be used to identify and keep track of emerging and/or trending hashtags related to the domain of interest. Third, using the immediate and sematic neighbourhoods of topics, the model can be used to retrieve more tweets related to the hashtag and/or topic of interest. And finally, it can be used to accurately position tweets by adopting an appropriate set of hashtags. These applications are discussed in the following.

4.3.3.1 Topic structure of the domain

This model not only provides a mechanism to integrate tweeter (data contributor) defined topic labels with user-defined (data user) topic labels, but also, using attribute lattice grammar principles, it reflects how hashtags contribute to the topic structure of the domain. The model represents: 1) topic labels, which presumably convey meaningful concepts to users; 2) sets of hashtags that reflect the underlying semantics of topics; and 3) sets of hashtags that best summarize the latent semantics. Also, based on attribute lattice principles, the supertopic and subtopic relationship among topic labels can be inferred from the model.

Note that a node in the model will be labeled as a topic label based on its immediate neighbourhood. Labeling a node, either hashtag or user-defined node, as a topic label changes the semantics of the node in two important ways. First, it declares that the node is a meaningful concept for users rather than an atomic word. In other words, when a node becomes a topic label, users potentially can define the topic hierarchy for it, can determine its types, and assign tweets to it and so on. For example, in the domain of technology, ***mobile*** (cellphone) can be an atomic word. However, if this word becomes a topic label, it will convey a meaningful concept and can be considered as sub-topic of ***handheld devices***. Also, ***smartphone*** will be a type of ***mobile***.

Second, labeling a node as a topic label states that a set of hashtags exists in the model that capture the latent semantics of tweets belonging to this topic, and this set of hashtags can be best summarized by the hashtags in the base.

4.3.3.2 Topic prediction and trend identification

A hashtag, either emerging or trending, might not be a meaningful word, and even for meaningful words, the topic that hashtag represents might not be semantically clear for users. Hashtags are conversational in nature – they can emerge at some point and may vanish soon thereafter (Huang et al., 2010), and tweeters can freely adopt/use any set of hashtags for their tweets. Considering the massive volume of tweets and the variety of hashtags adopted in these tweets, it is not possible for the human user to identify and keep track of all hashtags related to the domain of interest.

The attribute-lattice-based topic model offers mechanisms, first, to predict the topics of a given set of tweets, and second, to identify trending hashtags/terms related to the domain of interest. These two mechanisms are enabled by modeling hashtags as term vectors. The Vector Space Model (VSM) is a known model for representing text documents as a vector (Turney & Pantel, 2010). Using this model, the tweets vector (V_{tweets}) represents a term vector of unique hashtags in a given set of tweets, and, the attribute lattice vector (V_{model}) represents a term vector of the all the nodes in a given attribute lattice.

- ***Find topics of a given term/hashtag***

The topic model of a domain enables mapping a set of tweets onto the known, user-defined topics. Given a search term/hashtag, Twitter offers a set of tweets that contains the search term/hashtag. This tweet retrieval comes in three modes – that is, popular, recent, and mixed tweets. Here, the goal of topic prediction is to associate the search term/hashtag with known topics in the topic model. Hence, the proposed method for topic prediction

collects recent tweets (rather than popular, or mixed tweets which are filtered by Twitter) and uses them as a tweet set.

This set of tweets has been used to identify a vector of popular hashtags related to the search term/hashtag (V_{tweets}). The related topics can be identified by comparing hashtags in the set vector with the nodes in an existing topic model (V_{model}), and following the precedence relationships in the model. In section 4.5.2.2, I discussed how this mechanism has been utilized to identify the related topics of selected hashtags.

- ***Find domain-related terms/hashtags among trending terms/hashtags***

Twitter offers a set of trending hashtags/terms, every few minutes, in each geographical location and worldwide. For each *trending* hashtag/term, Twitter retrieves the *top* tweets by real-time analysis of related conversations and using sophisticated Machine Learning algorithms. Here, the proposed procedure uses top tweets for each trending hashtag/term to create a term vector for each trending hashtag/term (V_{tweets}). Using top tweets improves the procedure in two ways. First, it allows retrieving tweets that are semantically related to the current conversation, not all the tweets with the same term/hashtag. Second, it enables to accelerate tweet retrieval, that is, even with the smaller number of tweets, popular hashtags in the conversations can be retrieved.

Then, the cosine similarity (Han et al., 2011) has been used to compare the extent to which a term vector of each trending hashtag/term is related to the topic model of interest. The cosine similarity of two vectors is measured as indicated in Expression 34. A trending hashtag/term with a higher similarity will be more related to the topic model.

In this expression, the numerator represents the number of shared terms between V_{tweets} and V_{model}, and the denominator represents the multiplication of the number of terms in these vectors.

Expr 34. $Similarity = \frac{V_{tweets} \cdot V_{model}}{\|V_{tweets}\| . \|V_{model}\|}$

$where$

$V_{tweets}\ represents\ a\ vector\ of\ related\ hashtags\ for\ each\ trending\ topic$

$V_{model}\ represents\ a\ vector\ of\ nodes/hashtags\ in\ a\ given\ topic\ model$

$||\ represents\ the\ norm\ (size)\ of\ vector$

4.3.3.3 Tweet retrieval

The model can be used to retrieve more related tweets. The topic structure in the lattice (i.e., different types of hashtags and precedences) provides a mechanism to identify the most relevant hashtags to the topic of interest, and therefore, to retrieve more related tweets. This relationship leads to retrieving tweets that cannot be retrieved otherwise.

For instance, assume ***infosec*** and ***security*** are two hashtags in the domain of technology and ***security*** precedes ***infosec***. Users interested in tweets about security, can use this precedence relationship and include tweets with ***infosec*** hashtag. Tweets which have ***infosec*** hashtag but not ***security*** cannot be retrieved by overlooking the precedence relationship.

4.3.3.4 Tweet positioning

Hashtags can be utilized to direct tweets (Huang et al., 2010). That is, a tweeter adopts hashtags for her tweet in a way that the tweet will be seen in a specific stream. The topic model provides necessary information for tweeters to position their tweets. The model represents influential hashtags in a subject domain, frequent hashtags in a specific topic, and subsumption relationships among hashtags. Using this information, selecting the proper set of hashtags helps a tweeter to convey her message to the right audience.

For instance, suppose a tweeter wants to raise her concern about *data security* in a tweet. Topic model extracted from tweets in the domain of technology (Figure 21) suggests that *security* and, specifically, *infosec* are influential hashtags related to *data security* that can be adopted.

4.4 Implementation of Attribute-lattice-based Topic Model

The artifact implementation chapter (chapter 3) discussed the design and implementation of the attribute lattice artifact. The artifact helps users of the conceptual modeling grammar to create, update, visualize, and validate attribute lattices. This section aims to present how the artifact can be expanded to support other attribute lattice-based use cases. The proposed additional features help users to create an initial lattice from data, and to gain new insight into data using the attribute lattice.

In particular, this section introduces a set of additional features that supports the task of topic modeling. These features have been implemented in the artifact, and they are accessible to the research community[13]. These features enable users to use tweets to automatically create an initial topic model, to find related topics among trending topics, and to retrieve relevant tweets. Like before, using the domain knowledge, the automatically extracted model can be modified to reflect users' perspectives more accurately. Similarly, the model can be visualized and validated using basic features of attribute lattice artifact.

It is worthwhile to note that the model extraction and model modification procedures are independent – that is, once an initial topic model has been extracted from tweets, any update on the tweet set will not be reflected on the topic model, and vice versa. Table 6 summarizes implemented features that support attribute-lattice-based topic modeling.

[13] The application is available at this address (https://attribute-lattice.shinyapps.io/thesis/)

Table 6. Artifact Features; Extended for Attribute-lattice-based topic Modeling

Feature Area	*Feature*
4. Twitter Topic Modeling	4.1 Provide a mechanism to build the initial topic model from tweets automatically
	4.2 Provide a mechanism to manipulate and adjust the initial topic model
	4.3 Find topics related to the subject of interest among trending topics
	4.4 Search Twitter for a given hashtag/term (either trending or emerging) and find related topics in an existing topic model

4.4.1 Implemented Functions to Extract Initial Topic Model

Following the topic model procedure (Figure 20), the Twitter API has been used to retrieve the known tweeters based on user-provided lists. The developed function for refining tweeters filters out tweeters whose accounts are not publicly available. This is followed by the function that extracts recent tweets for tweeters in the list from the previous step. The Twitter API has certain limits for tweet retrieval. First, it only provides the last 3200 tweets of each tweeter, second, it limits retrieval to 32,000 tweets every 15 minutes. As a result, tweet retrieval is a time-consuming process.

The developed function splits the refined tweeters list and retrieves the maximum number of tweets available for each tweeter – that is, max (number of tweeters' tweet, and 3200). The function repeats the process every 15 minutes to retrieve latest tweets for all tweeters. It is worthwhile to note that the Twitter API is configured, in this function, to extract only the original tweets (not quotes nor retweets) and tweets written in English.

The Twitter API provides a variety of information about each tweet. In addition to the tweeter and hashtags, it includes other information such as the location of the tweeter, number of retweets and quotes, and whether the tweet is written to respond to another tweet. The developed model focuses on the tweet itself, tweeter, and all the hashtags in the tweet. The next function is developed to refine the tweets, that is, 1) to make sure all the tweets are original tweets written in English, 2) to filter out tweets without any hashtags 3) to remove special characters from hashtags, and 4) to change all the hashtags' characters to lower case. The result of this function will be an ***id for the tweet***, ***the tweeter name,*** and ***a list of hashtags for each tweet.***

The next function summarizes tweets based on co-occurrences of hashtags and calculates the frequency and popularity of them. The following pseudocode elaborates the function.

Input: *Retrieved tweets (tweet* $_{id}$*, tweeter* $_{id}$*, hashtags' list)*

Create *a separate row for each hashtag (the result will be tweet* $_{id}$*, tweeter* $_{id}$*, hashtag)*
Group by *tweet* $_{id}$*, tweeter* $_{id}$
Filter out *tweets with less than two hashtags*
Find *all pairs of hashtags for tweets with more than two hashtags (hashtag A, hashtag B)*
Filter out *pairs in which hashtag A = hashtag B (this will happen if the tweeter uses a single hashtag more than once in a single tweet.)*
Group by *hashtag A, hashtag B*
Count *number of tweets for each pair*
Add Column *to save count as the frequency of the pair*

Group by *hashtag A, hashtag B, tweeter* $_{id}$*, frequency*
Remove *duplicates*
Group by *hashtag A, hashtag B, frequency*
Count *number of pairs (this will provide the number of unique tweeters for each pair)*
Add Column *to save count as the popularity of the pair*
Group by *hashtag A, hashtag B, frequency, popularity*
Remove *duplicates*

Output: *hashtag co-occurrences (hashtag A, hashtag B, frequency, popularity)*

Finally, the last function creates a topic model based on the output of the above-mentioned function (i.e., hashtag A, hashtag B, frequency, and popularity), the total number of tweeters in the refined tweeters' lists, and a set of user-defined thresholds (i.e., popularity rate, precedence threshold, topic threshold, and frequency threshold). The following pseudocode elaborates the topic model creation function.

Input: *hashtag co-occurrences (hashtag A, hashtag B, frequency, popularity), no. of Tweeters,* pop_{rate}, $t_{precedence}$, t_{frq}, t_{topic}

Calculate $pop_{min} = no.\ of\ Tweeters * pop_{rate}$
Filter out *pairs with popularity less than* pop_{min}
Calculate *popularity for each hashtag (hashtag A and hashtag B separately) in the remaining pairs (i.e., the summation of the popularity of the pairs that have the given tag as hashtag A or hashtag B)*
Add Columns *to store* $pop_{hashtag\ A}$, *and* $pop_{hashtag\ B}$
Find *rows in which* $pop_{hashtag\ A} > pop_{hashtag\ B}$

Permute *(hashtag A, hashtag B) and (pop* $_{hashtag\ A}$*, pop* $_{hashtag\ B}$*). To make sure hashtag A and hashtag B are always arranged in a way that pop* $_{hashtag\ A}$ *<= pop* $_{hashtag\ B}$

Calculate *conditional probabilities*

- ***Calculate*** *p(hashtag A | hashtag B) = pop* $_{hashtag\ A,\ hashtag\ B}$*/ pop* $_{hashtag\ B}$
- ***Add Column*** *to store p (hashtag A | hashtag B)*
- ***Calculate*** *p(hashtag B | hashtag A) = pop* $_{hashtag\ A,\ hashtag\ B}$*/ pop* $_{hashtag\ A}$
- ***Add Column*** *to store p (hashtag B | hashtag A)*[14]

Find *precedence relationships from pairs*

- ***Find*** *pairs in which p (hashtag B | hashtag A)>= t* $_{pop}$ *and p (hashtag A | hashtag B) < 1*

Prune *the topic model*

- ***Filter out*** *transitive precedences (for instance, If hashtag A → hashtag B, hashtag B → hashtag C, and hashtag A → hashtag C , Then filter out hashtag A → hashtag C)*

Find *topic labels and base precedences from pairs*

- ***Group by*** *hashtag B*
- ***Calculate*** *Number of preceded hashtags for each hashtag B (B* $_{input}$*)*
- ***Calculate*** *the maximum frequency for each hashtag B (frq* $_{max}$*)*
- ***Calculate*** *the minimum frequency for each hashtag B (frq* $_{min}$*)*
- ***Find*** *precedences in which B* $_{input}$ *>= t* $_{topic}$
- ***Find*** *precedences in which frq* $_{hashtag\ A,\ hashtag\ B}$ *>= frq* $_{min}$
- ***Label*** *pairs as base precedence*

Find *inferred hashtags for each base*

[14] Note that, the previous step ensures pop $_{tag\ A}$ <= pop $_{tag\ B}$, as a result, p (tag B | tag A) >= p (tag A | tag B)

- ***Find*** *candidate pair for weak precedences (pairs in which the first hashtag, A, is a base for topic label B, and the second hashtag, C, preceded by topic label B)*
- ***Find*** *weak precedences p (hashtag A | hashtag C) > 0*
- ***Label*** *pairs as precedence*

Find *all attributes (hashtag A, hashtag B) from the precedence*

Output: *Topic model (attributes, precedences)*

4.4.2 Topic Model Manipulation

The initial topic model has been built based on hashtag co-occurrences. Hashtags are conversational in nature, and they are extremely sparse. As a result, the initial topic model is not necessarily accurate. Moreover, users should be able to add/modify the topic labels in the model. The basic functions discussed in Chapter 3 can be used to add hashtag/topic label, add hashtag categories, and modify suggested precedences.

4.4.3 Trends - Topic Model Similarity

One application of the topic model is to track trending hashtags/terms and examine if they are related to a user domain of interest. Using the Twitter API, the implemented function retrieves the trending hashtags/terms in the user-specified location and measures similarity of each hashtag/term to an existing topic model.

This function starts with getting a geographical location from the user. For each location, the Twitter API provides a list of top 50 trending hashtags/terms. Hence, the result of this step is a list of trending topics/terms based on a given location. Next, this function retrieves a small sample of popular tweets for each term (e.g., one hundred popular tweets

for each hashtag/term). Twitter measures the popularity of tweets by the extent to which they capture the attention of users. For this step, I retrieve a small sample of popular tweets to rapidly retrieve tweets for all trending hashtags/terms without reaching to the API limits (a hundred tweets for all trending hashtags/terms can be retrieved in a few seconds.) Next, for each hashtag/term, the function defines a vector that represents a list of unique hashtags in tweets retrieved for it. Finally, it measures the cosine similarity of vectors of hashtag/term with the vector of an existing topic model.

4.4.4 Topics/Hashtags of a Given Search Term/Hashtag

The trends - topic model similarity, introduced in the previous section, aims to find hashtags/terms that are potentially related to the domain of interest. The function measures and compares similarities for trending hashtags/terms. However, since the similarity is measured based on the small sample size, it might not be able to identify hashtags/topics in an existing topic model accurately. As a result, another function is developed for in-depth analyzing of tweets retrieved based on searching for a hashtag/term. This function aims to find meaningful hashtags in the tweets and show how these hashtags are related to the known hashtags/topics in a given topic model.

The function starts with a user-specified search term/hashtag and a popularity rate. It retrieves the maximum number of tweets available for the term/hashtag. The latest 18,000 tweets for each search term/hashtag can be retrieved by using the Twitter API. Tweets are refined to extract English tweets. Next, the function creates a list of hashtags and their popularities. The function calculates the minimum popularity based on the number of tweeters who contribute to the retrieved tweets multiple by user-specified popularity

rate. Finally, it returns a list of hashtags for which popularities are higher than the minimum popularity as meaningful hashtags in retrieved tweets. For instance, assume the given term is *bigdata,* and the given popularity rate is one percent. This function, first, retrieves and refines the latest 18,000 tweets related to this search term and calculates the number tweeters in the tweet set. Assume 2000 tweeters contribute to this tweet set. Then, with the given popularity rate, the function creates a list of hashtags that have been adopted by at least 20 unique tweeters and returns them as a set of meaningful hashtags for this search term. The topic structure of the tweets will be presented to users by comparing this result with the known hashtags in a given topic model.

4.5 Evaluation

This section presents the results of an evaluation of attribute-lattice-based topic modeling in the domain of technology. The topic model described in this section is extracted directly from tweets without user modification.

4.5.1 Topic Model for the Domain of Technology

4.5.1.1 Searching for known tweeters

For the first step, Google search is used to find known tweeters in the domain of technology. Borison (2014) offers a peer-reviewed list of the most influential tweeters in this domain. Although the report was written a couple of years ago, it still provides a valuable list of tweeters. Five other Twitter lists which represent influential tweeters in this domain has been added to the initial list of tweeters. Table 7 represents six selected lists and the number of tweeters in each list.

4.5.1.2 Refining tweeters' list

The initial list was refined to filter out duplicate tweeters (the same tweeter appearing in various lists) and to find tweeters who their accounts are publicly available. 1,533 unique tweeters with the publicly available account were found after refining the initial tweeters' list (Table 7).

4.5.1.3 Extracting tweets

The most recent 3,200 tweets for each tweeter can be retrieved via the Twitter API. However, not all tweeters have 3,200 tweets. For instance, Bill Gates is among the top 10 tweeters in this domain (based on Business Insider peer-reviewed list (Borison, 2014)) and, as of May 2018, he has posted only 2,681 tweets (Table 8). For this step, the maximum number of available tweets for each tweeter is retrieved.

Table 7. Lists of tweeters

List	***Type of list***	***Owner***	***No. of Tweeter***
Most Influential in Tech	Twitter List	Scobleizer	404
Digital and Social Media	Twitter List	courtenaybird	483
Toptechbloggers	Twitter List	louisgray	116
CIO	Twitter List	abbielundberg	236
Legal tech thinkers	Twitter List	nikiblack	87
Cloud	Twitter List	GeorgeReese	288
Business Insider	Peer reviewed List	----	100
Total Number of Tweeters			1714
Unique Number of Tweeters (Public)			1533

Table 8. Top ten tweeters - based on peer reviewed list

	Name	*Twitter ID*	*No. of Followers*	*No. of Lists*	*No. of Tweets*
1	jack	jack	4,195,463	27,395	23,463
2	Jeremiah Owyang	jowyang	176,216	13,193	69,252
3	Aaron Levie	levie	2,659,674	5,353	3,722
4	OM	om	1,525,552	14,321	49,520
5	Robert Scoble	Scobleizer	426,649	24,830	69,735
6	Elon Musk	elonmusk	21,642,048	43,970	4,181
7	Mathew Ingram	mathewi	86,865	5,978	221,083
8	Benedict Evans	BenedictEvans	246,539	6,415	128,786
9	Bill Gates	BillGates	45,996,671	121,953	2,681
10	Anil Dash	anildash	630,855	8,914	168,723

4.5.1.4 Refining tweets

After filtering out retweets, non-English, and quoted tweet, 2,591,322 tweets were retrieved for tweeters in the list. As demonstrated in Table 9, approximately 20 percent of the remaining tweets have at least one hashtag. The distribution of hashtags is shown in Table 10. In the attribute-lattice-based topic model, the co-occurrences of hashtags will be used to create a topic model and to suggest the precedences relationship among hashtags. As shown in the table, more than one-third of tweets containing hashtags have more than one hashtag.

Table 9. Refining tweets

	No of Tweets	***No of Tweeters***
Retrieved tweets	3,761,288	1533
Refined Tweets (eliminating quotes, retweets, non-English tweets)	2,591,322	1532
tweets with hashtag	532,150	1491

Table 10. Distribution of Hashtags

No. of Hashtags	***No. of Tweets***	***% of Tweets***	***No. of Tweeters***
1	344,771	64.79	1488
2	116,581	21.91	1351
3	41,031	7.71	1115
4	15,505	2.91	785
5	6,606	1.24	506
6	2,933	0.55	335
More than 6	4,723	0.89	
TOTAL	532,150		

4.5.1.5 Calculating frequency and popularity of hashtags

A meaningful co-occurrence refers to a pair of hashtags such that their popularities are higher than minimum popularity. As shown in Table 11, low popularity rate (low minimum popularity) identifies a vast number of hashtags as meaningful. However, more restricted popularity rates (high minimum popularity) recognizes only a few (or even no) meaningful hashtags.

Table 10 reflects the sparseness of hashtags in the given set of tweets. It shows how tweeters (even within the same subject domain) adopt/use a varied set of hashtags in their tweets. Among 245,292 initial pairs of hashtags, only 326 pairs are adopted by more than

1 percent of tweeters. Where a pair of hashtags is adopted by more tweeters, there is higher chance that users perceive the pair to be meaningful. By increasing the popularity rate, the remaining pairs have more chance to be perceived as meaningful by the user. However, the cost of this increase is a higher probability of missing some meaningful pairs.

Table 11. Meaningful co-occurrence

Popularity rate	***Minimum popularity***	***Meaningful co-occurrence***	***No. of hashtags***
0	0	245,292	90,956
0.005	8	1140	579
0.01	16	326	212
0.02	31	83	84
0.05	77	2	3

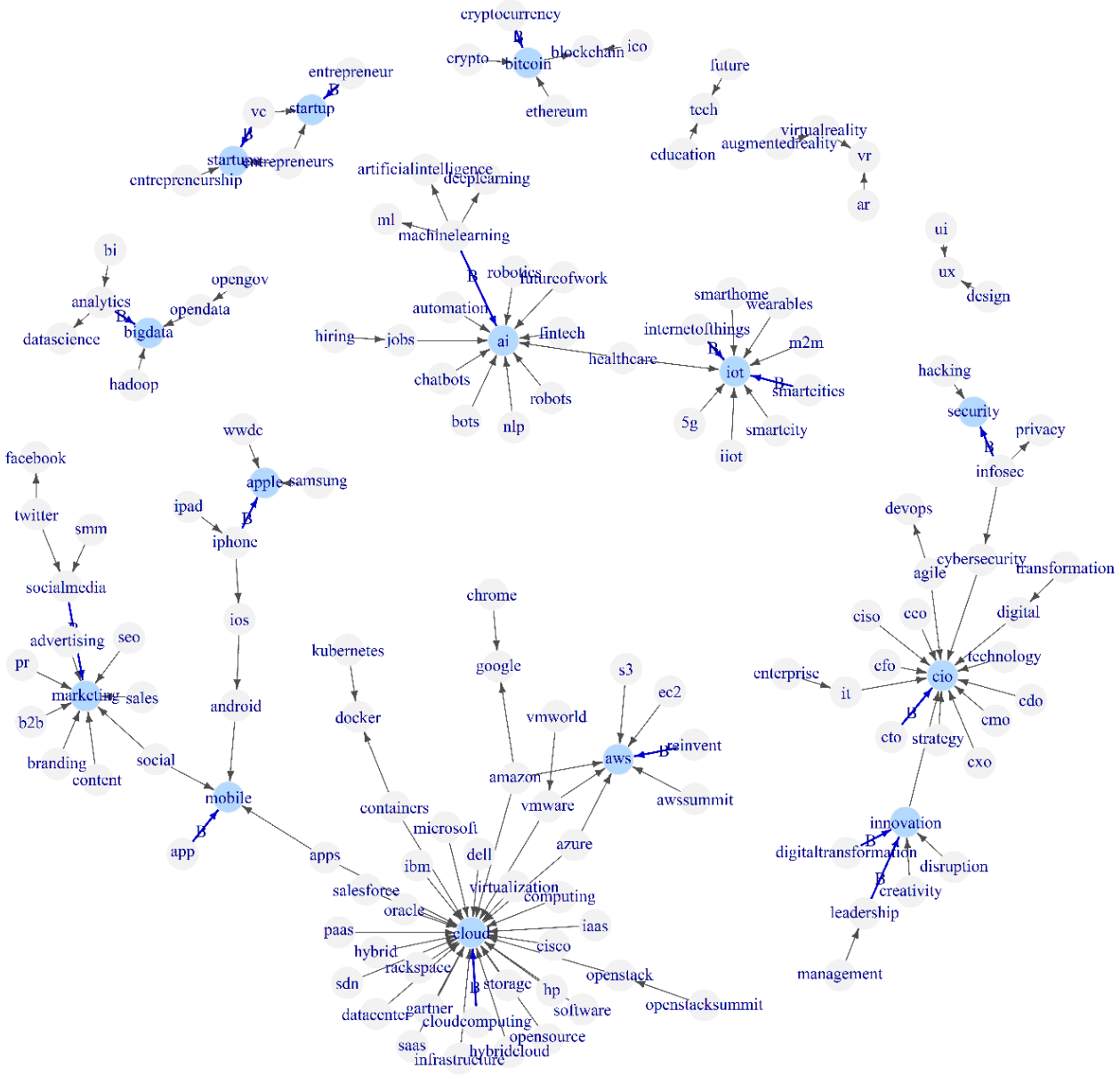


Figure 21. Topic Model; pop $_{rate}$ = 1%, t $_{precedence}$ = 0.3, t $_{frq}$ = 0.8, t $_{topic}$ = 3

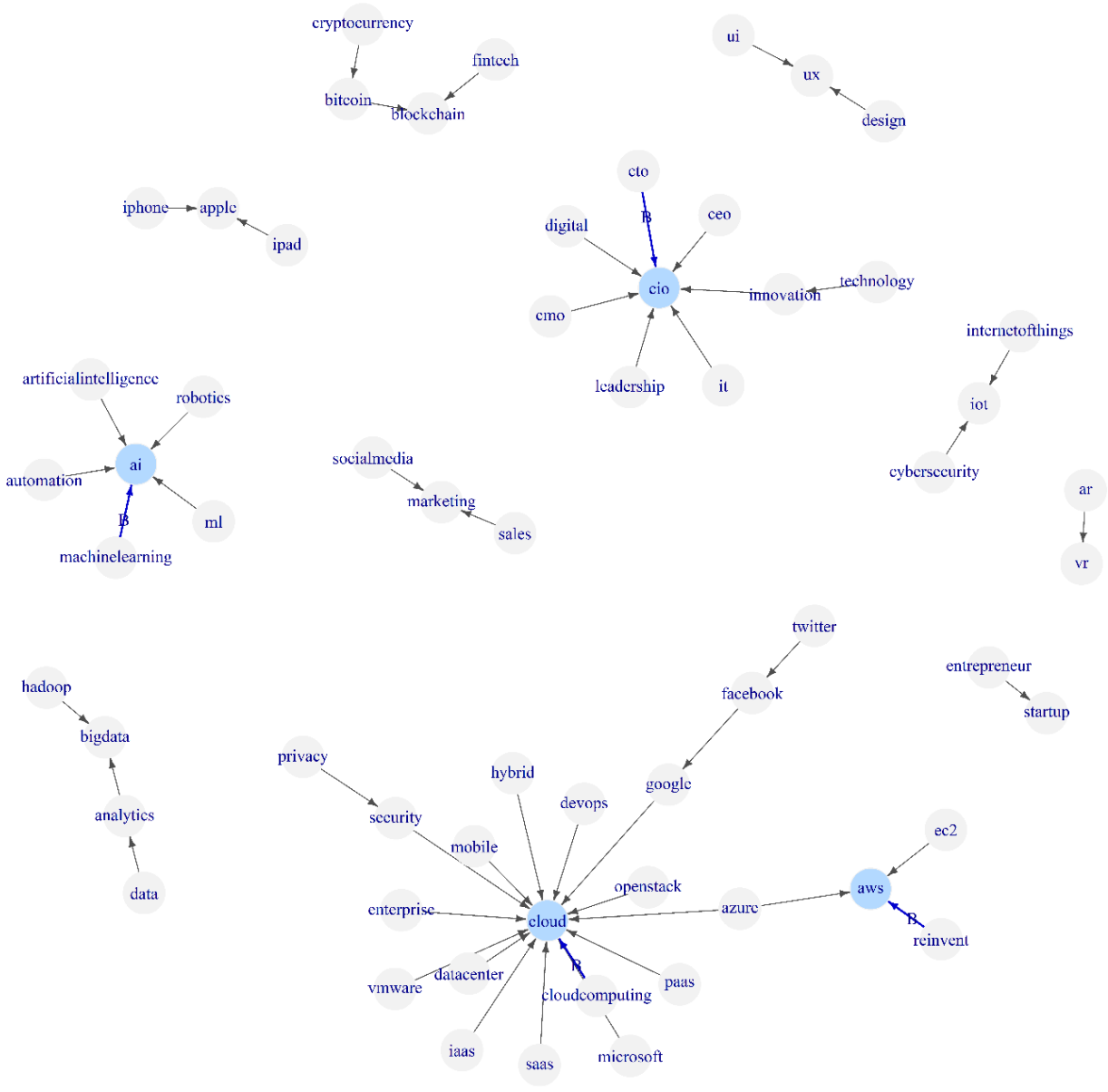


Figure 22. Topic Model; pop $_{rate}$ = 2%, t $_{precedence}$ = 0.4, t $_{frq}$ = 0.8, t $_{topic}$ = 3

4.5.1.6 Constructing the attribute lattice

The model uses a set of user-defined thresholds - popularity rate, precedence threshold, topic threshold, and frequency threshold - to suggest initial relations and a topic model. The number of identified attributes, topic labels, precedences and base precedences is sensitive to these user-defined thresholds. Popularity rate and precedence threshold affect the number of precedences that the model considers as meaningful.

As represented in Figures 21 and 22, independent from the topic threshold and frequency threshold, increasing the popularity rate and precedence thresholds leads to identify fewer precedence relationships. A low popularity rate (low popularity threshold) might result in a very high number of hashtags being considered as meaningful. However, more restrictive popularity rates (higher popularity threshold) might recognize only a few (or even no) meaningful hashtags.

The number of potential subsumption relationships in the model depends on both popularity rate and precedence threshold. Increasing the popularity rate and precedence thresholds leads to identify fewer subsumption relationships (Table 12). Higher popularity rates indicate that more tweeters adopted/used a pair. Hence, it has a higher chance to be perceived as meaningful by users. Higher precedence thresholds indicate higher confidence that a subsumption relationship exists between the hashtags in the pair. As a result, increasing popularity rate and precedence threshold increases precision– that is, subsumption relationship with higher confidence. However, this increase comes at a cost of decrease in recall – that is, a cost of missing potentially meaningful precedences.

Table 12. Popularity rate and precedence threshold sensitivity analysis

			Popularity Rate				
			0	**0.005**	**0.01**	**0.02**	**0.05**
		Minimum Popularity	0	8	16	31	77
		Unique co-occurrence	245,292	1140	326	83	2
Precedence Threshold	0.1	No. of Attributes	61,322	409	146	63	3
		No. of Precedences	91,175	514	165	63	2
	0.2	No. of Attributes	58,593	408	145	63	3
		No. of Precedences	73,850	453	153	63	2
	0.3	No. of Attributes	51,998	399	144	63	3
		No. of Precedences	54,861	389	138	55	2
	0.4	No. of Attributes	44,258	376	137	61	3
		No. of Precedences	40,351	332	114	50	2
	0.5	No. of Attributes	43,471	351	126	59	3
		No. of Precedences	39,341	288	100	46	2
	0.6	No. of Attributes	27,583	312	115	57	3
		No. of Precedences	19,730	243	89	43	2
	0.7	No. of Attributes	26,451	286	109	53	3
		No. of Precedences	18,699	217	83	39	2
	0.8	No. of Attributes	26,156	265	103	51	3
		No. of Precedences	18,421	197	77	36	2
	0.9	No. of Attributes	25,940	254	96	50	3
		No. of Precedences	18,220	187	71	35	2
	1	No. of Attributes	25,887	251	96	50	3
		No. of Precedences	18,171	184	71	35	2

The topic threshold determines the number of attributes designated as topic labels. Increasing the topic threshold results in a model that suggests fewer potential topic labels. Conversely, relaxing topic thresholds produces a model that identifies more topic labels but increases the risk of suggesting topics that might not be semantically clear for users.

Finally, the number of representative hashtags for each topic - that is, precedences that are considered as base precedence- is influenced by frequency threshold. With a restricted frequency threshold (i.e., frequency threshold of 1), the model identifies only one representative hashtag for each topic label. However, with a more relaxed frequency threshold, there is a chance to identify more representative hashtags (base precedences). For instance, assume the frequency of three hashtags in a given topic are 1000, 950, and 700. With a restricted frequency threshold, only one hashtag with the frequency of 1000 will be identified as a representative hashtag. However, with the frequency threshold of 0.9, all hashtags that their frequencies are higher than 900 (1000 and 950 in this example) will be identified as representative hashtags. Table 13 shows the number of identified topic labels and representative hashtags with the fixed popularity rate and precedence threshold ($pop_{rate} = 1\%$, $t_{precedence} = 0.3$).

The unsupervised nature of the proposed topic modeling approach makes evaluating the performance of user-defined thresholds a challenging task. Popularity rate and precedence threshold contribute to identifying a hashtag as a topic and topic label and including it in the model. It is reasonable to assume that user-defined thresholds are performing better if they offer a more consistent topic model for the domain of interest. That is, topic models created based on different subsets of tweets are similar.

Table 13. Topic and frequency thresholds sensitivity analysis

Topic Threshold	No. of Topic Labels and Representative Hashtags	Frequency Thresholds 0.5	0.6	0.7	0.8	0.9
1	Topic labels	35	35	35	35	35
	Representative Hashtags	51	50	43	38	37
2	Topic labels	21	21	21	21	21
	Representative Hashtags	37	36	29	24	23
3	Topic labels	14	14	14	14	14
	Representative Hashtags	27	26	20	16	16
4	Topic labels	10	10	10	10	10
	Representative Hashtags	22	21	15	12	12
5	Topic labels	6	6	6	6	6
	Representative Hashtags	15	15	9	7	7

In this study, I randomly select and set aside 30% of the retrieved tweets as a test set and use the rest of the tweets as a training set. Using training and testing sets, a series of topic models were created by manipulating popularity rate and precedence threshold to identify thresholds that provide a more consistent topic model. For each manipulation, the cosine similarity (Expr 35) of the train and test models has been measured (Table 14).

$$Expr\ 35.\ Similarity = \frac{V_{training} \cdot V_{test}}{\|V_{training}\| . \|V_{test}\|}$$

$where$

$V_{training}\ represents\ a\ vector\ of\ nodes/hashtags\ in\ the\ training\ topic\ model$

$V_{test}\ represents\ a\ vector\ of\ nodes/hashtags\ in\ the\ test\ topic\ model$

$||\ represents\ the\ norm\ (size)\ of\ vector$

The results show that the popularity rate of 0.01 (i.e., co-occurrences that have been adopted by at least one percent of tweeters) provides more similar models in comparison to other popularity rates (Table 14, and Figure 23). Also, a precedence threshold between 0.1 and 0.4 offers a more consistent topic model for this domain. Although these thresholds provide models with higher consistency for this specific domain, the results might not be generalized to other domains.

Table 14. Performance analysis of popularity rate and Precedence threshold

Training/Test Model Similarity		Popularity Rate			
		0	**0.005**	**0.01**	**0.02**
Precedence Threshold	0.1	0.43	0.69	0.73	0.61
	0.2	0.41	0.71	0.78	0.57
	0.3	0.37	0.71	0.73	0.58
	0.4	0.34	0.65	0.72	0.52
	0.5	0.32	0.57	0.68	0.6
	0.6	0.27	0.49	0.5	0.61
	0.7	0.25	0.54	0.54	0.61
	0.8	0.25	0.43	0.53	0.5
	0.9	0.24	0.46	0.46	0.5
	1	0.24	0.44	0.45	0.44

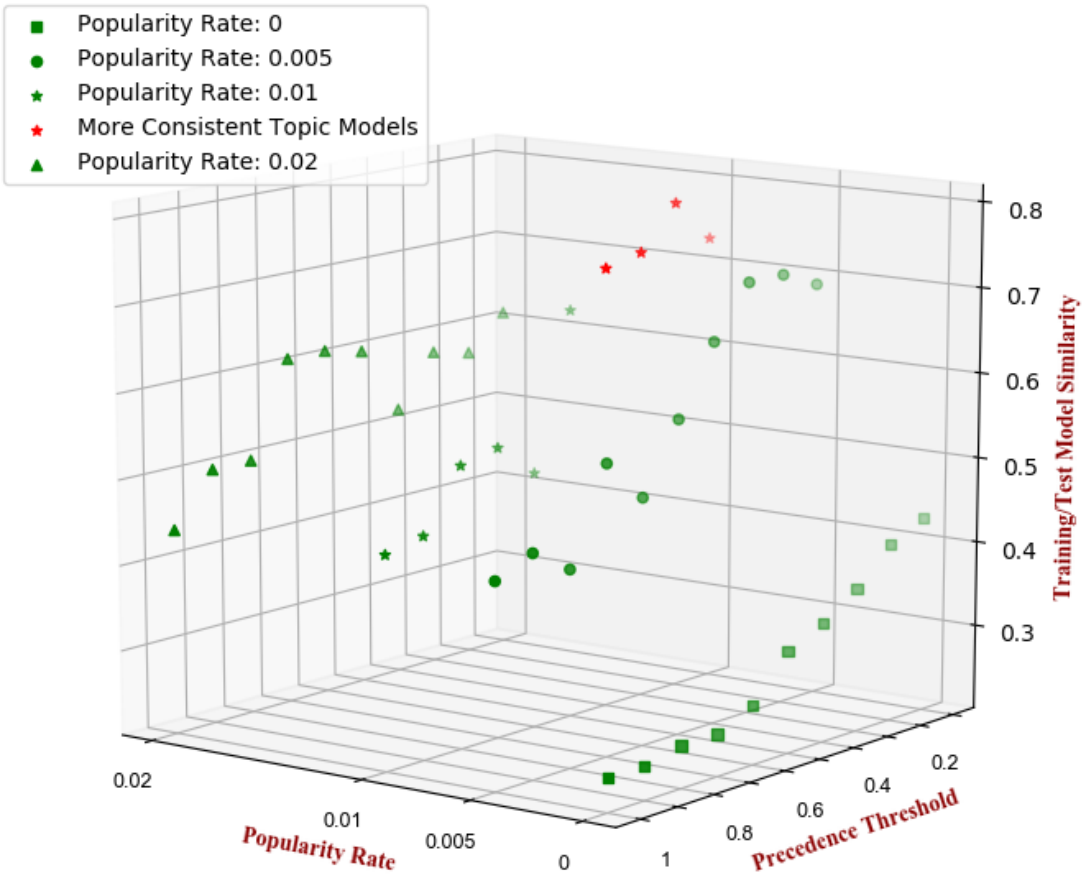
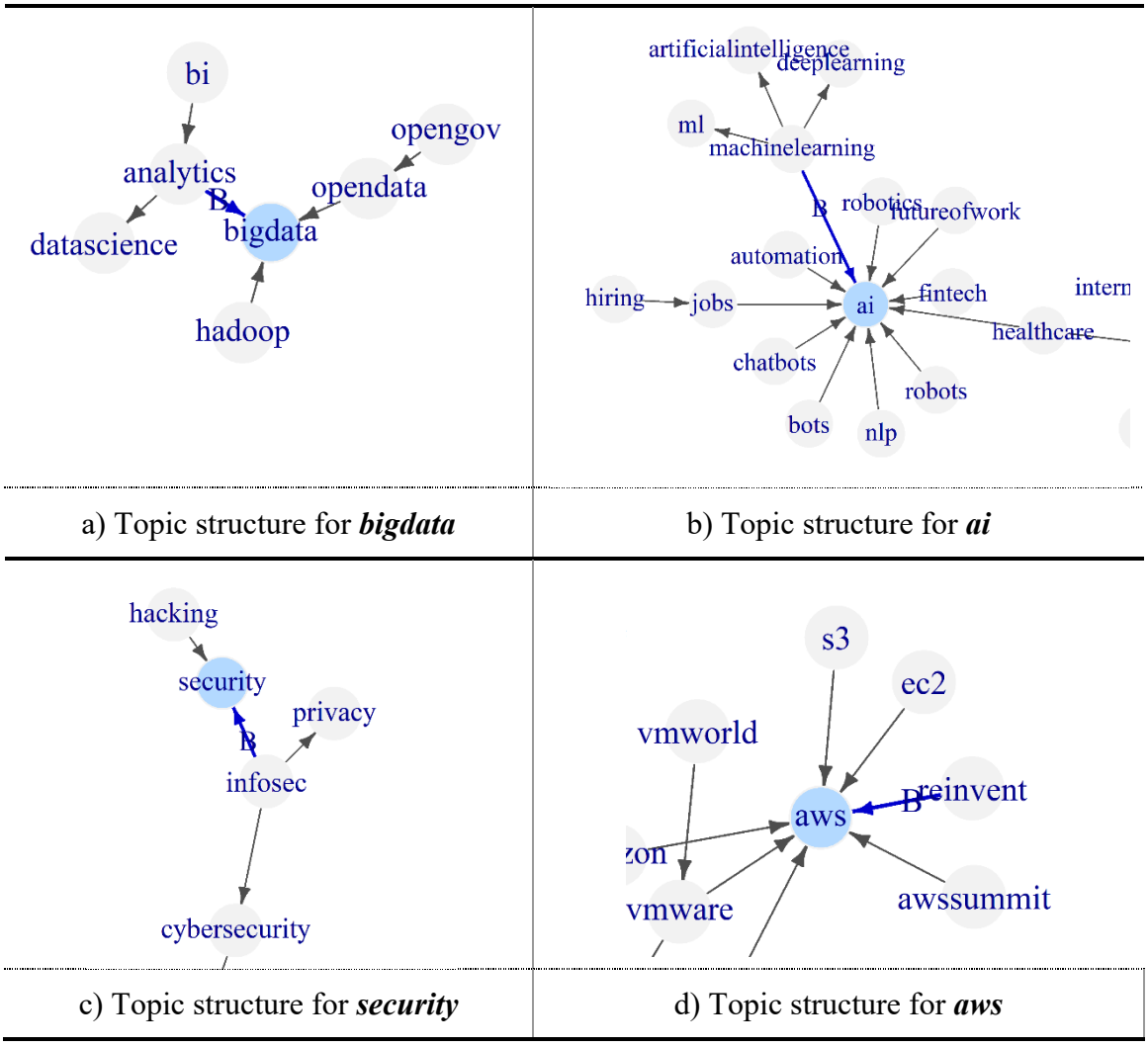


Figure 23. Performance analysis of popularity rate and Precedence threshold

4.5.2 Practical Applications of Topic Model

4.5.2.1 Topic structure of the domain

The topic model represents topics, topic labels, and representative hashtags for each topic. Figure 24 a-d illustrates different parts of the topic model represented in Figure 21.



a) Topic structure for ***bigdata*** | b) Topic structure for ***ai***

c) Topic structure for ***security*** | d) Topic structure for ***aws***

Figure 24. Topics, bases, and their expansions

In the technology domain, *bigdata* (Figure 24.a) is a topic label, and *analytics* is a representative hashtag for this topic; that is, *analytics* can best summarize hashtags in the *bigdata* topic. Also, *bigdata*, *analytics*, and *bi* are hashtags that frequently appear in this topic. For this topic, both *opendata*, and *hadoop* are qualifying hashtags, that is, *bigdata* can be inferred from these hashtags, but they are not part of the topic. As shown in Figure 24.b, *ai* is another topic label, *machinelearning* is a base for it, and *ai*, *machinelearning*, *deeplearning*, and *artificialintelligence* are frequent hashtags in this topic.

Once again, this model has been built based on tweets without any adjustment. It can be further adjusted to provide clearer semantic structure. For instance, based on tweets, we do not have enough evidence to include *hadoop* as a frequent hashtag in the *bigdata* topic however, users might be interested to include *hadoop*. The initial model can be adjusted to include this hashtag as a part of topic as well.

The topic structure enables users to retrieve tweets on the topic of interest that cannot be retrieved otherwise. Following precedences, specifically base precedences, users can include other hashtags which contribute to the topic of interest to retrieve tweets. For instance, as shown in Figure 24.c, *infosec* is preceded by *security.* Users interested retrieving tweets related to security, can retrieve tweets with both *infosec* and *security* hashtags. In the current dataset, 3420 tweets exist with *security* hashtag, 927 with *infosec* hashtag, and 194 with both hashtags. Including the *infosec* in the tweet retrieval will result to find 733 (21 percent more) tweets related to the topic that cannot be retrieved using only the *security* hashtag. Figure 25 shows tweets related to the topic of *security*, which include *infosec*

hashtag but not *security* hashtag. Table 15 shows the number of tweets that can be retrieved based on the representative words of the topics (base precedences) in the model.

In the topic model created by the above-discussed thresholds, assuming all the initial base precedences are perceived as meaningful relationships by users, by following base precedences, an average 24 percent more tweets can be retrieved. That is, using representative hashtags for topics leads to retrieve an average 24 percent more related tweets for each topic.

Table 15. Increased number of tweet retrieval, using representative words

Topic	Representative Hashtag	Tweets with both Hashtags	Tweets with topic hashtag	Tweets without topic hashtag	Increased percentage
cio	cto	2261	15054	122	0.81
iot	smartcities	280	9284	161	1.73
iot	internetofthings	274	9284	185	1.99
ai	machinelearning	672	8163	774	9.48
mobile	app	205	3422	351	10.26
cloud	cloudcomputing	1407	15934	2063	12.95
bigdata	analytics	473	4158	778	18.71
startups	vc	129	2006	396	19.74
security	infosec	194	3420	733	21.43
startup	entrepreneur	367	2886	658	22.80
bitcoin	cryptocurrency	133	1323	323	24.41
apple	iphone	159	1858	549	29.55
aws	reinvent	706	4892	1574	32.17
innovation	digitaltransformation	864	4387	2084	47.50
innovation	leadership	886	4387	2662	60.68
marketing	socialmedia	205	3603	2294	63.67

Finally, the topic structure enables users to adopt a more accurate set of hashtags to increase the chance of being seen in the right community. For instance, the *aws* topic structure suggests (Figure 24.d) that *aws* is a hashtag that tweeters adopt to talk about amazon web services. Suppose, a tweeter has a comment on one of the amazon web services. The

tweeter can adopt this influential hashtag in her tweet to increase the chance of being seen in the right stream.

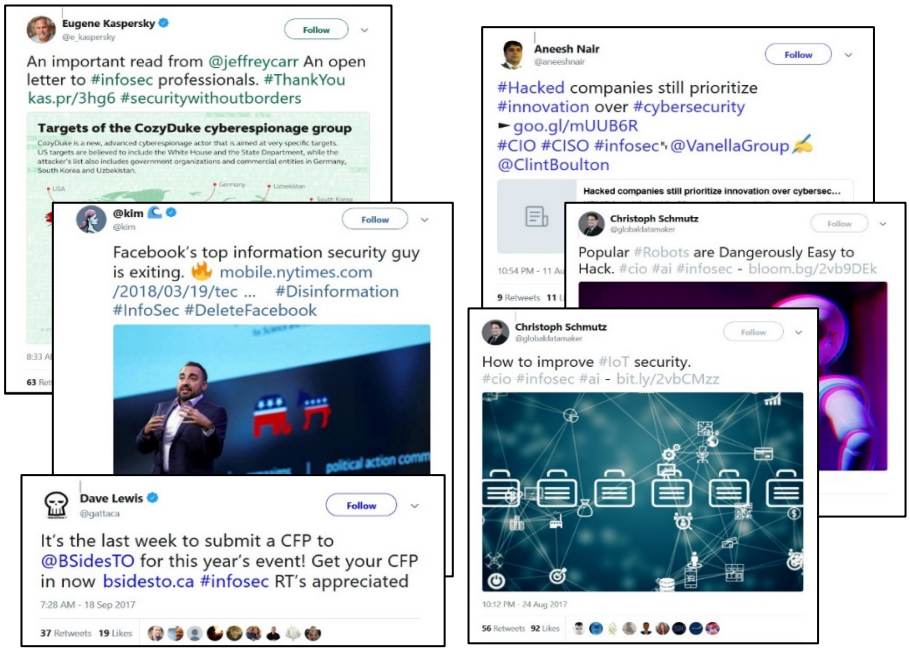


Figure 25. Examples of tweets related to *security* retrieved by *infosec* hashtag

To summarize, the topic model represents the topic structure of the domain - the precedence relationship among hashtags, how hashtags can be grouped into topics, and how each hashtag contributes to the topic. This topic structure improves semantic interpretability of hashtags and information (tweets) retrieval based on hashtags.

4.5.2.2 Hashtag detection

Twitter provides a list of 50 trending hashtags/terms based on locations around the world. These trending hashtags/terms are a valuable source of information for users to follow their topics of interest. However, detecting hashtags/terms related to the domain of interest is not a trivial task for three reasons. First, a massive number of tweets are created every minute. As a result, Twitter updates the list of trending hashtags/terms frequently and tracking all the trending hashtags/terms is a cumbersome (if not impossible) task for the human user. Second, hashtags are mostly written in a shortened format, as a result, they are not necessarily semantically clear for users. And third, even if a hashtag represents a meaningful word, understanding topics that hashtag represents is not a trivial task, particularly when same hashtags (e.g., #apple) may appear in unrelated domains. This section aims to describe how the initial topic model created from tweets can be used to identify topics of emerging hashtags/terms.

Using the developed function (elaborated in section 4.4.4), I retrieved Canada trending hashtags/terms for two consecutive days (June 11 and 12, 2018), and measured the similarity of hashtags/terms with the topic model represented in Figure 21. Figure 26 and Figure 27 show the result of similarity analysis. Two hashtags on day 1 (#CLUS, #TOIC2018), and one hashtag on day 2 (#ConfMTL) had similarity with the given topic model. However, these hashtags are not among known hashtags in the given topic model. Also, the meanings of these hashtags are not clear.

As represented in in Figures 28 and 29, the function identifies nine related nodes for the first hashtag (#CLUS). Following the precedences, six topic labels and four hashtags

can be identified for this hashtag. Hashtags and topic labels suggest that #CLUS should be related to cloud technology of cisco. In fact, this hashtag had been used by tweeters to talk about Cisco live event in Orlando focusing on network and collaboration[15].

The second hashtag (#TOIC2018) was related to TribalScale's TakeOver Innovation Conference in Toronto, focusing on innovation[16]. Identified hashtags and topics provide a good insight into the topics of this hashtag (Figures 30 and 31.) Finally, the last hashtag (Figures 32 and 33) was related to an annual economic event (Conference of Montreal) organized by the International Economic Forum of the Americas[17] (IEFA) which focuses on major current economic issues. The identified hashtags and topic labels suggest that tweeters had tweets related to technological issues of current economy (*tech*, *fintech*, *innovation*, *ai*). Table 16 summarizes these three hashtags and identified hashtags and topic labels for each one of them.

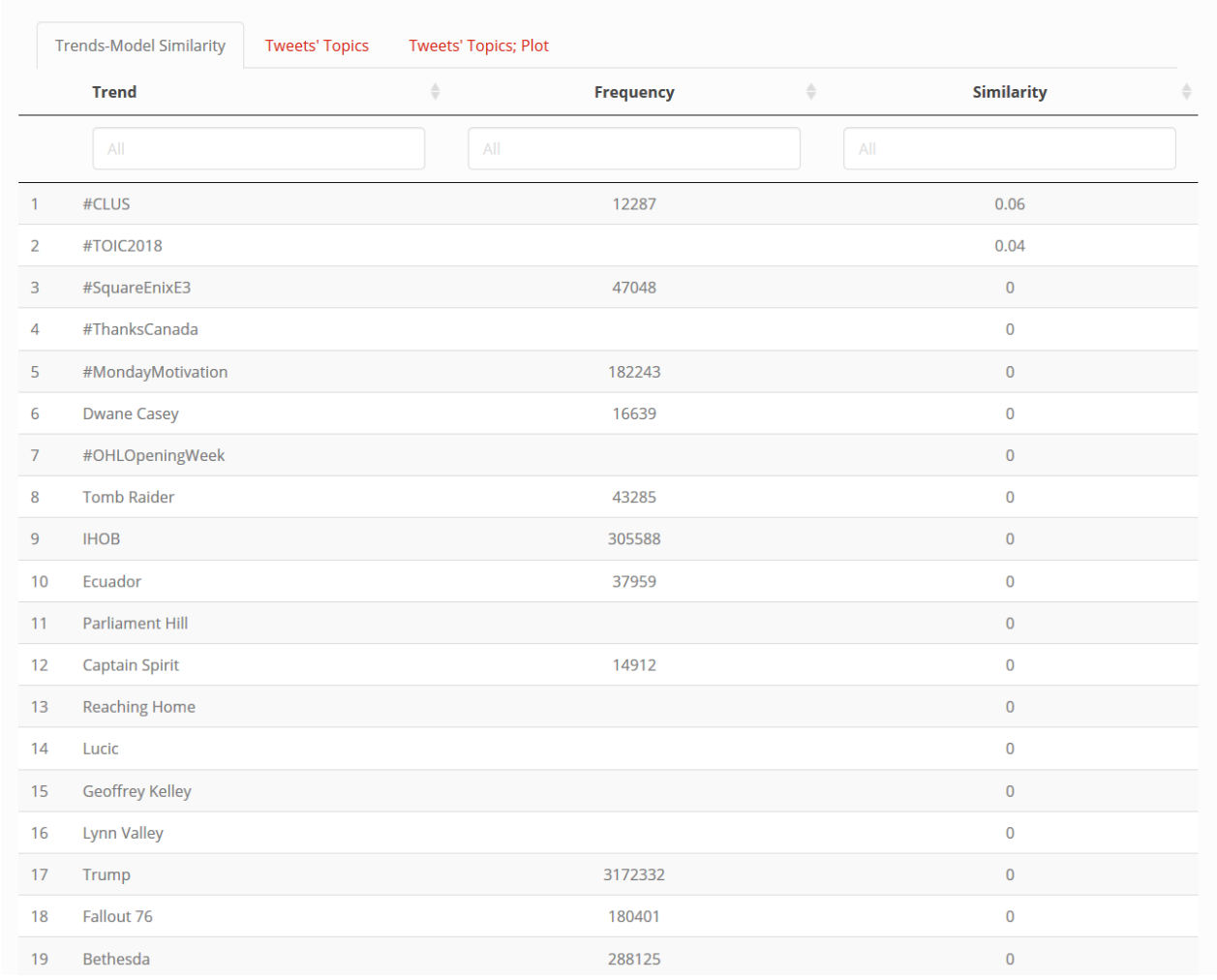
To summarize, the practical contribution of hashtag detection is twofold. First, it provides a procedure to detect hashtags that are related to a given topic model. Tweeters utilize hashtags and keywords to contribute to the discussion about an issue, an event or a topic. Every day hundreds of hashtags/terms become trending on Twitter. The proposed method measures the extent to which trending hashtag or keyword is related to users' topic model. Second, an in-depth analysis of a given hashtag finds co-occurring hashtags/topics that are known to users. This enables users to infer the semantics of ambiguous hashtags.

---

[15] https://www.ciscolive.com/us/attend/about
[16] https://takeoverinnovationconference.com
[17] http://forum-americas.org/montreal/home/

Table 16. Hashtag detection

Hashtag	*Identified Hashtags*	*Identified Topic Labels*
#CLUS	***cisco***, automation, machinelearning, analytics	bigdata, innovation, security, iot, cloud, ai
#TOIC2018	ar, vr, healthcare, finetech, futureofwork, trans-formation, blockchain, entrepreneurs, tech	***innovation***, ai, startup
#ConfMTL	***tech***, ***fintech***, artificialintelligence	ai, innovation



Trends-Model Similarity | Tweets' Topics | Tweets' Topics; Plot

	Trend	Frequency	Similarity
	All	All	All
1	#CLUS	12287	0.06
2	#TOIC2018		0.04
3	#SquareEnixE3	47048	0
4	#ThanksCanada		0
5	#MondayMotivation	182243	0
6	Dwane Casey	16639	0
7	#OHLOpeningWeek		0
8	Tomb Raider	43285	0
9	IHOB	305588	0
10	Ecuador	37959	0
11	Parliament Hill		0
12	Captain Spirit	14912	0
13	Reaching Home		0
14	Lucic		0
15	Geoffrey Kelley		0
16	Lynn Valley		0
17	Trump	3172332	0
18	Fallout 76	180401	0
19	Bethesda	288125	0

Figure 26. Trends – topic model similarity, day 1

Trends-Model Similarity | Tweets' Topics | Tweets' Topics; Plot

	Trend	Frequency	Similarity
	All	All	All
1	#ConfMTL		0.06
2	#TuesdayThoughts	58675	0
3	#OttawaDiner		0
4	Former MP Paul Dewar		0
5	#TravelTuesday		0
6	Korean Peninsula	104636	0
7	#FoxKatZ1035		0
8	Death Stranding	151280	0
9	Dennis Rodman	262258	0
10	Ubisoft	185396	0
11	Ghost of Tsushima	42572	0
12	Nintendo	192255	0
13	Kingdom Hearts	205562	0
14	Assassin's Creed	81954	0
15	Last of Us 2	63861	0
16	Dwane Casey	23700	0
17	For Honor	21196	0
18	Bethesda	62180	0
19	How Canadians		0

Figure 27. Trends – topic model similarity, day 2

Trends-Model Similarity | Tweets' Topics | Tweets' Topics; Plot

	Hashtag	Frequency	Popularity	Shared Topic
	All	All	All	All
1	cisco	157	96	Yes
2	cloud	155	92	Yes
3	automation	108	84	Yes
4	security	84	65	Yes
5	machinelearning	69	57	Yes
6	innovation	63	52	Yes
7	iot	94	36	Yes
8	ai	47	36	Yes
9	analytics	53	35	Yes
10	clus	9794	2004	
11	ciscolive	234	133	
12	devnet	398	107	
13	clus18	175	95	
14	multicloud	115	80	
15	accelerateit	112	70	
16	ciscochampion	290	62	
17	ciscoinvests	177	61	
18	cluswin	46	46	
19	network	59	45	
20	orlando	65	43	

Figure 28. Tweets' topics and topic labels, #CLUS

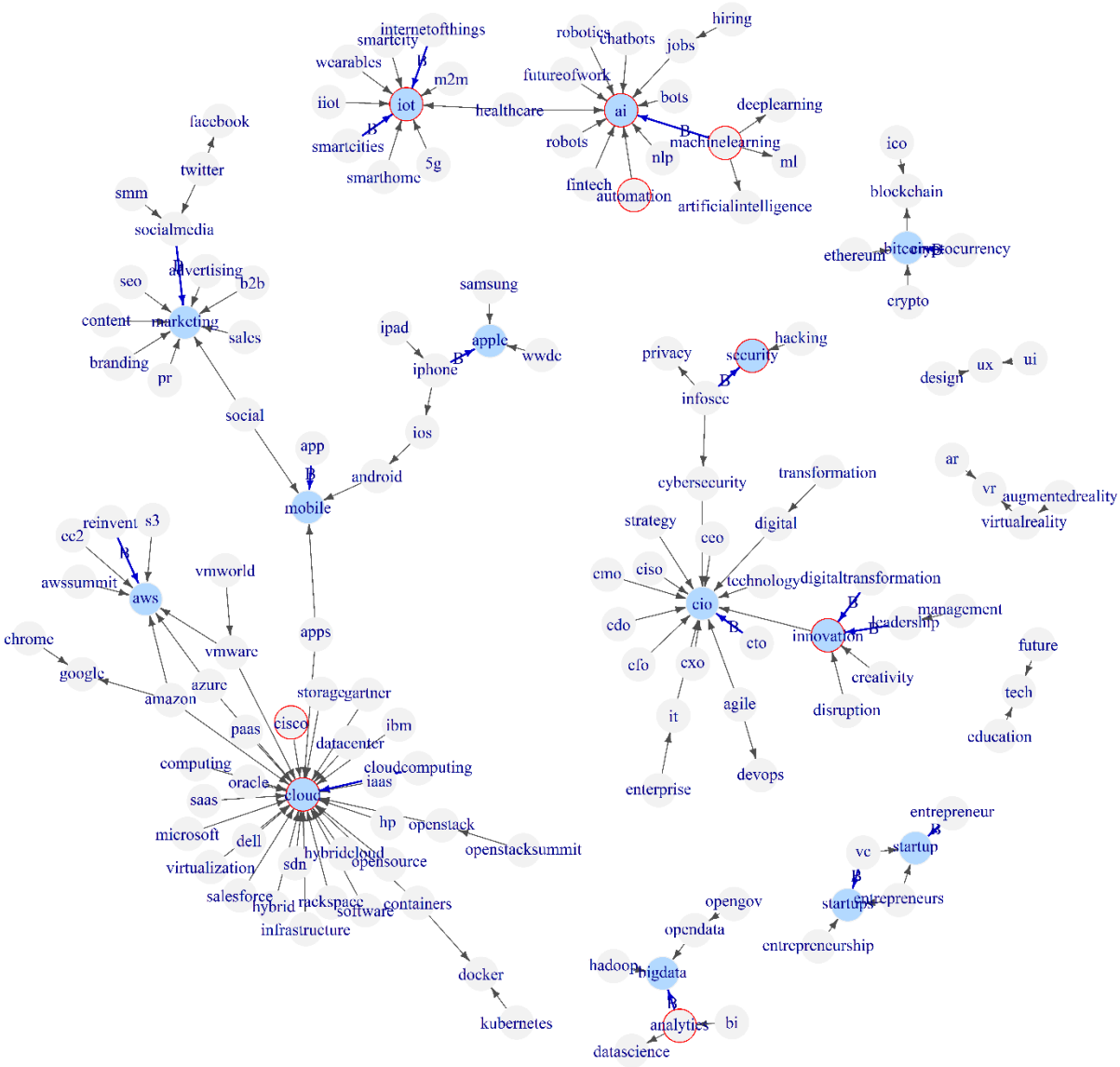


Figure 29. Tweets' topics and topic labels; plot, #CLUS

Trends-Model Similarity | Tweets' Topics | Tweets' Topics; Plot

	Hashtag	Frequency	Popularity	Shared Topic
	All	All	All	All
1	innovation	78	44	Yes
2	blockchain	19	15	Yes
3	ai	27	15	Yes
4	tech	17	12	Yes
5	fintech	12	10	Yes
6	ar	12	9	Yes
7	vr	11	8	Yes
8	startup	8	5	Yes
9	healthcare	4	4	Yes
10	marketing	4	3	Yes
11	entrepreneurs	3	3	Yes
12	futureofwork	3	3	Yes
13	transformation	3	3	Yes
14	toic2018	890	257	
15	toronto	33	21	
16	yourwineisourpassion	31	17	
17	takeover	16	10	
18	movethedial	12	10	
19	mondaymotivation	22	8	
20	gdpr	8	7	
21	yourwineismypassion	9	5	

Figure 30. Tweets' topics and topic labels, #TOIC2018

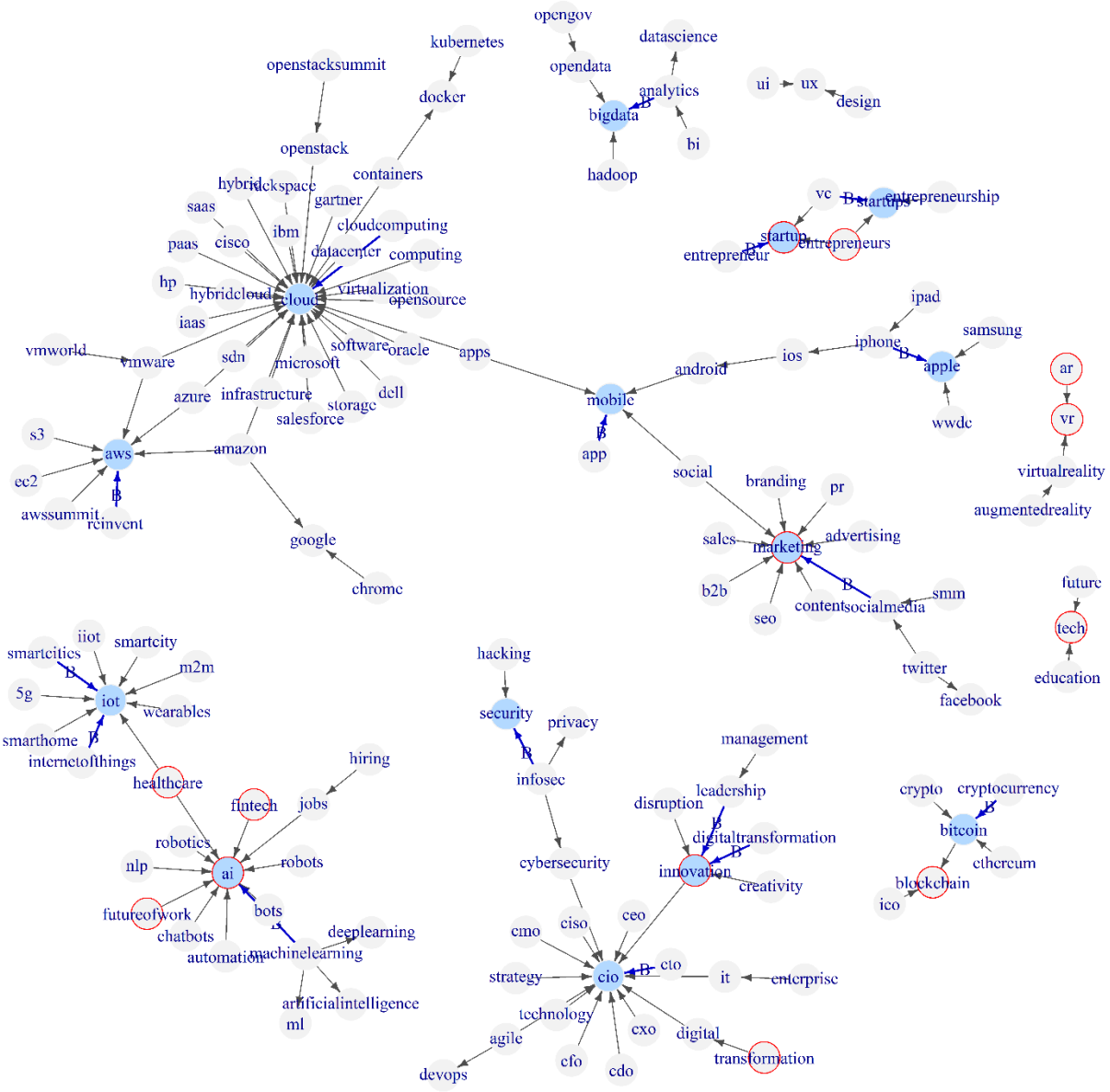


Figure 31. Tweets' topics and topic labels; plot, #TOIC2018

Trends-Model Similarity | Tweets' Topics | Tweets' Topics; Plot

	Hashtag	Frequency	Popularity	Shared Topic
	All	All	All	All
1	ai	3	3	Yes
2	artificialintelligence	2	2	Yes
3	innovation	1	1	Yes
4	tech	1	1	Yes
5	fintech	1	1	Yes
6	confmtl	179	69	
7	montreal	4	4	
8	mondaymotivation	18	4	
9	toic2018	18	4	
10	stkitts	2	2	
11	nevis	2	2	
12	canada	2	2	
13	montral	2	2	
14	trade	2	2	
15	maharashtra	2	2	
16	ihob	5	2	
17	ihop	5	2	
18	foxkatz1035	13	2	
19	iiex	2	2	
20	otttraffic	13	2	

Figure 32. Tweets' topics and topic labels, #ConfMTL

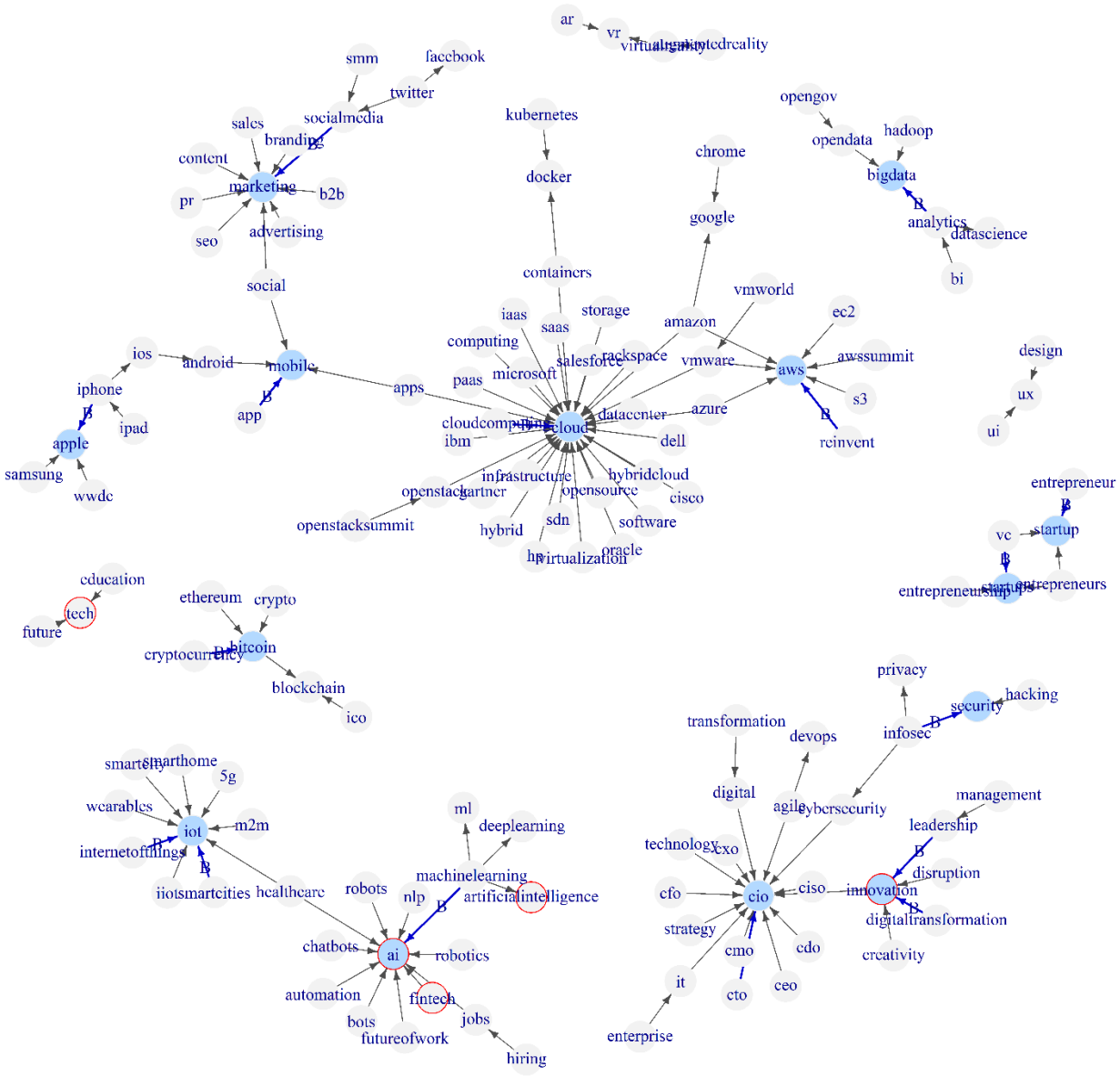


Figure 33. Tweets' topics and topic labels; plot, #ConfMTL

4.6 Discussion

In the era of big data, decision-makers increasingly need to rely on data from external data sources such as social media for decision making. Data heterogeneity is one of the main challenges for the meaningful use of data. The concept of attribute lattice is introduced chapter 2 as a schema-free conceptual modeling grammar to represent the semantic structure of data from various data sources. This chapter elaborates how attribute lattice principles can be adopted for the task of topic modeling, that is, representing the topic structure of data.

One challenge of attribute lattices, and consequently for the attribute-lattice-based topic model, is defining attributes and precedences from the beginning. Creating an attribute lattice (specifically for unstructured data) without an initial schema would be a cumbersome process. This chapter addressed this concern by offering an automated procedure to extract an initial attribute lattice from unstructured data. Twitter data has been selected to elaborate on how the initial topic model can be constructed from tweets.

In the context of Twitter, hashtags – tweeter defined labels for tweets – provide valuable insight into the topics of a given tweet. However, the structure of hashtags makes them semantically unclear, and they cannot be used directly to retrieve tweets of interest. This chapter argues, first, the attribute lattice approach can be used to mitigate the semantic ambiguity of hashtags, and second, the topic model provides a semantic grounding for users to create the topic model of the domain of interest.

This chapter offers a mechanism to extract an initial attribute lattice from tweets. The initial attribute lattice can be adjusted to include user-defined topic labels. The extracted topic model from tweets not only provides a foundation for users to create the topic model of the domain, but also improves the usefulness of hashtags for tweet retrieval in several ways.

First, it can be used to identify hashtags/terms of interest among the vast amount of trending hashtags/terms. Second, given an ambiguous hashtag, the topic model suggests related topics/hashtags from known topics/hashtags. And third, the model can be used to retrieve more related tweets.

4.7 Limitations

This chapter demonstrates how a topic model can be constructed from tweets automatically. However, a limitation of this study is the user-defined thresholds (i.e., popularity rate, precedence threshold, topic threshold, and frequency threshold). The procedure relies on these thresholds to create the model, and the semantic clarity of the extracted model depends on them. The sensitivity analysis of the extracted models suggests an initial value for thresholds in the domain of technology. However, to be able to generalize the suggested values future studies are needed. A potential future study in the attribute-lattice-based topic model is to examine to what extent the extracted model is perceived as semantically clear for various thresholds in different domains.

Another concern about the automated topic model extraction procedure is domain selection. If retrieved tweets come from a specific and focused domain, a key hashtag might become dominant, and the initial topic model becomes ineffective. For instance, the result of an application to the travel domain shows that *#travel* is a dominant hashtag in this domain (i.e., *#travel* appear in the majority of pairs). This leads to a topic model in which *#travel* precedes most other hashtags. Hence, the initial model is ineffective. This problem can be mitigated by two approaches. First, the topic model can be pruned to decrease the effects of the dominant hashtag. Second, other sources of information (such as DBpedia) can be incorporated into the model to identify more semantically clear topics. For instance, in the travel domain, a part of the extracted hashtags are meaningful, known words such as #paris, #cruise, #food, and so on). The DBpedia metadata, as an additional data source,

may help the initial lattice extraction procedure by defining new topics such as a place to go, a place to stays, a way to travel and so on, and including them in the initial model.

5 Contributions, Future Work and Conclusion

In the era of Big Data, data comes from a wide variety of sources in different formats and structures. In most cases, the data schema is unknown to the data users or data does not have a schema. Where data users query and analyze these heterogeneous data sources for purposes beyond what data contributor might anticipate, the ability to assigning consistent and interoperable data semantics to data sources has become more important than ever before. This thesis develops a conceptual modeling grammar to represent data semantics of independent and heterogeneous (structured and unstructured) data sources. This thesis makes several contributions to theory and practice.

5.1 Contributions to Research and Practice

5.1.1 Developing a conceptual modeling grammar

This thesis conceptualizes an alternative role for conceptual modeling. In contrast with the current information system development paradigm that considers conceptual modeling as a part of requirements engineering, the proposed conceptual modeling grammar has been developed to understand semantics of existing data.

Parsons and Wand (2014) suggest that in the environment in which data comes from sources with unknown schema, conceptual model grammars need to enable users to apply their own conceptual models to data. Hence, this thesis proposed a theory-based conceptual

modeling grammar for this purpose. This lightweight, graph-based grammar is: (1) developed based on principles from cognitive psychology, philosophical ontology, and graph theory; and (2) independent from the schema of the data source that it presents.

An important contribution of the attribute lattice grammar is its semantic relativism. Semantic relativism is enabled by extending the concept of attribute precedence to capture subsumption relationships among attributes more distinctly. This grammar defines three types of precedence relationships - that is, simple precedence, base precedence, and subcategory precedence. The patterns of arcs and nodes (i.e., precedences and attributes) around each attribute reflects how users of data perceive the subsumption relationships related to the attribute. These patterns enable data users to infer the type of attributes, expansion of attributes, class bases, class properties and the class structure of the domain. This inferential representation contributes to the meaningful use of data by enabling users to define a schema based on the current data inquiry task and making data consumers independent from the structure of data source.

This thesis extends the concept of attribute similarity for attribute lattice-based semantic data integration (Evermann, 2008a, 2008b). It defines similar attributes as attributes that are: (1) semantically equal; (2) a manifestation of the same higher-level attribute; or (3) in a generalization/specialization relation to each other. Using similar attributes as merge nodes, the proposed semantic data integration approach provides a unified view over varied and heterogeneous data sources, and hence, enables data consumers to identify related instances in different data sources.

5.1.2 Gaining insights from data

In open information environments (OIEs), users need to apply their own conceptual model to data to gain insight into it (Parsons & Wand, 2014). The proposed grammar is well suited for creating and analyzing the class structure of the data source and attaining new knowledge about the domain that the model represents.

Enabled by the implemented artifact, given a set of precedences it is possible to automatically: (1) analyze the lattice to determine which nodes are classes and make inferences about instances that belong to each of these classes; (2) validate the models (scripts) against the model of good classification (Parsons & Wand, 2008); (3) visualize the model and view the data structure from various perspectives; and finally (4) conduct what-if analysis.

The type of attributes may change by adding precedences to or removing precedences from an attribute lattice. This means changes may affect the inferences that can be made about instances. The implemented artifact helps users to understand how changing precedences may affect class structure, and inferences more clearly.

5.1.3 Topic Modeling

This thesis further contributes by adopting the attribute lattice principles to conceptualize the topic structure of tweets related to a domain of interest and enhances information retrieval by improving semantic interpretability of hashtags.

The proposed topic modeling approach contributes to the practical usefulness of this grammar in two important ways. First, it represents a theory-based process through which

practitioners can create an attribute lattice model from unstructured (or semi-structured) data itself. To create an attribute lattice model, users need to identify all precedence relationships. It might not be practically feasible to identify all these relationships only based on domain knowledge and without incorporating the data. Here, based on the assumptions that came from literature - assumptions such as using tweets with hashtags rather than all tweets - a procedure has been defined through which an attribute lattice can be extracted from data (Figure 20).

Second, attribute lattice-based topic modeling demonstrates how practitioners, using the model, get a deeper understanding of the domain. Attribute lattice grammar can be utilized to represent data semantics as perceived by the user of the data. The proposed topic modeling approach elaborates on how practitioners can use this conceptual model to understand the topical structure of a domain and to use the model to improve information retrieval (section 4.3.3).

5.2 Future Research

5.2.1 Expanding the grammar

This thesis proposes a lightweight conceptual modeling grammar with a minimum set of components to understand data semantics. This grammar enables users to infer the classes that an instance belong to based on the instance attributes. Based on philosophical ontology guidelines the precedence relationship in this grammar is pairwise in the sense that an attribute or a set of the attributes (class/category) precedes another attribute or a set of attributes (class/category). However, utilizing a logic-based approach to develop this grammar might lead to adding other components to it. For instance, the more logic-based approach suggests adding 'NOT' and 'XOR' operators where the underlying ontology deals only with the presence of properties (not their absence). Future research on attribute lattice might expand this grammar to include logic-based components and examine if the additional components improve the performance of the grammar.

5.2.2 Attribute lattice-based semantic data integration

Although the primary focus of the second chapter is defining attribute lattice as a conceptual modeling language, it represents a procedure through which this grammar can be utilized for the semantic data integration of heterogeneous data sources. This integration process starts with an initial set of merge nodes and iteratively suggests new merge nodes based on the immediate and semantic neighbourhoods of similar attributes. This procedure, however, is subject to two limitations. It assumes the initial set of merge nodes is available

based on the domain knowledge, and, the process leaves users with the potential merge nodes. A possible avenue for the future study in attribute lattice-based semantic data integration is to use multi-method integration approach (e.g., Li et al. (2008)) to create the initial list of merge nodes, and to suggest more accurate and definite merge nodes at each step.

5.2.3 An initial attribute lattice and model quality

The grammar is intrinsically simple, and it has only two components –nodes that represent attributes, and arcs that represent subsumption relationships. However, because data sources might have a huge number of attributes, creating an attribute lattice could be a cumbersome task.

This study demonstrates how data can be used to create a conceptual model (the topic model in this case), and how the model can improve the understanding of data users (compared to not having such a model) by predicting the topic of trending hashtags, and improving information retrieval. Here, based on the assumptions that come from literature - assumptions such as using tweets with hashtags rather than all tweets - we define a procedure to show the possibility of extracting an attribute lattice extraction from data. And, we demonstrate that even this initial attribute lattice will provide new insight into data. However, an important open question is how to use a pre-existing schema and/or data itself to create attribute lattice from other data sources. Future research can examine challenges (i.e., the complexity of the lattice) that data consumers may face when creating an attribute lattice from different data sources.

The similarity of the model created based on the train, and the test tweet sets have been used to evaluate the consistency of the models and to offer a set of values for the user-defined thresholds. However, it will be important that future research investigates the overall *quality* of the lattice. Two approaches can be envisioned to evaluate the lattice quality in further research.

First, the study could use labeled data to examine the quality of the initially extracted lattices. For example, the quality of lattices extracted from Twitter can be evaluated by using a standardized set of tweets with "known" topics. Given a set of labeled tweets, and using various thresholds, the quality of extracted lattices could be compared to each other, or even this method can be compared to other methods. Second, the research can evaluate how the quality of the created conceptual model can be measured based on the purpose of the task at hand. For instance, the quality of a lattice created to summarize the topics in a domain should be measured differently from a lattice created for improving information retrieval. In the former case, the focus should be on human-readability of the model, however, in the latter on the adequacy of precedence threshold.

5.2.4 Improving the quality of decision-making

In the era of big data, organizations increasingly use external data sources, such as social media data, to make strategic decisions (LaValle et al., 2011). Using social media, all stakeholders can share information in a short time. This offers organizations a new opportunity to actively listen to their customers and other stakeholders and use their feedback in the decision-making process (Power & Phillips-Wren, 2011; Malthouse et al., 2013). To

be able to capture this information and use it for decision making, decision-makers need a tool to actively monitor social media (Del Giudice et al., 2016).

The attribute lattice-topic modeling approach has provided evidence for using this grammar to find the relevant information among the trending topics and to find more related data in social media. Future work can investigate the extent to which the proposed conceptual modeling approach can benefit decision-makers to retrieve more related information from external sources (especially social media) to make more informed decisions.

Moreover, this thesis provides a theoretical argument for integrating attribute lattices representing distinct data sources. It elaborates rules to find potential attributes that are candidates to be similar based on the known merge nodes. Another avenue for future research is examining the extent to which the proposed semantic data integration approach will improve meaningful use of data. For example, in the context of healthcare, previous research suggests that both publicly available and patients' social media data can be utilized to predict the pattern of Emergency Department (ED) visits (Ram et al., 2015). As patients are willing to share their social media data to compare it with their electronic medical records (Padrez et al., 2016), future work can investigate whether and how the proposed topic modeling approach for social media data can contribute to the understanding of the health-related data and predicting patients patterns more accurately.

5.2.5 Incorporate topic identification approaches for topic modeling

Based on the assumption that tweets with hashtags tend to be more meaningful for topic analysis, the procedure elaborated in Chapter four focuses on hashtags and their co-occurrences to suggest potential precedence relationship and, consequently, to create an initial topic model. A possible avenue for future study is to utilize existing topic identification approaches, find keywords for tweets without hashtags and include all tweets in the topic model.

5.3 Thesis Conclusions

This research focuses attention on the data variety aspect of big data. The meaningful use of data that comes in different structures from varied data sources entails assigning consistent and interoperable data semantics to it. This thesis argues conceptual models, in contrast with their traditional roles, can be utilized to visualize data semantics of pre-existing data sources. In this regard, by using principles from philosophy and human cognition, a conceptual modeling grammar has been introduced. The proposed grammar is independent of the data structure in the data source. This grammar provides data users with a data consumer aimed schema to analyze data and integrate it with other sources.

This research adopts the grammar to visualize the topic structure of social media content. The practical evaluation of the topic modeling approach confirms this modeling grammar provides insight into data and improves information retrieval.

REFERENCES

Agnarsson, G., & Greenlaw, R. (2007). *Graph theory: Modeling, applications, and algorithms*: Pearson/Prentice Hall.

Angeles, P. A. (1981). A dictionary of philosophy.

Baader, F. (2003). *The description logic handbook: Theory, implementation and applications*: Cambridge university press.

Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B., & Su, Z. (2007). *Optimizing web search using social annotations.* Proceedings of the 16th international conference on World Wide Web.

Barrasa Rodríguez, J., Corcho, Ó., & Gómez-Pérez, A. (2004). R2O, an extensible and semantically based database-to-ontology mapping language.

Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Comput. Surv., 18*, 323–364.

Bergamaschi, S., Castano, S., & Vincini, M. (1999). Semantic integration of semistructured and structured data sources. *ACM SIGMOD Record, 28*, 54-59.

Berners-Lee, T. (2006). Linked data-design issues. *URL http://www.w3.org/DesignIssues/LinkedData.html, 10*, 11.

Berners-Lee, T., Fielding, R. T., & Masinter, L. (2005). Uniform resource identifier (URI): Generic syntax.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web.

Bizer, C., Boncz, P., Brodie, M. L., & Erling, O. (2012). The meaningful use of big data: four perspectives--four challenges. *ACM SIGMOD Record, 40*(4), 56-60.

Bizer, C., Heath, T., & Berners-Lee, T. (2011). Linked Data: The Story So Far. 205-227.

Blei, D. M., & Lafferty, J. D. (2005). *Correlated topic models.* Proceedings of the 18th International Conference on Neural Information Processing Systems.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research, 3*(Jan), 993-1022.

Borison, R. (2014). The 100 Most Influential Tech People On Twitter. Retrieved from http://www.businessinsider.com/100-influential-tech-people-on-twitter-2014-4#looking-for-more-people-to-follow-on-twitter-101

Brachman, R. J., & Levesque, H. J. (1984). *The tractability of subsumption in frame-based description languages.* AAAI.

Brickley, D., & Guha, R. (2014). RDF schema 1.1. W3c recommendation, W3C.

Bunge, M. (1977). *Treatise on Basic Philosophy: Ontology I: The Furniture of the World*. Dordrecht: Springer Netherlands.

Burton-Jones, A., Wand, Y., & Weber, R. (2009). Guidelines for empirical evaluations of conceptual modeling grammars. *Journal of the Association for Information Systems, 10*(6), 1.

Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J., & Blei, D. M. (2009). *Reading tea leaves: How humans interpret topic models.* Advances in neural information processing systems.

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2015). Shiny: web application framework for R. *R package version 0.11, 1*.

Chen, P. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9-36.

Chen, T., & Parsons, J. (2008). *Using Property Precedence to Enhance The Effectiveness of Queries of Unstructured Data.* Workshop on Information Technologies and Systems.

Clifton, C., Housman, E., & Rosenthal, A. (1998). Experience with a combined approach to attribute-matching across heterogeneous databases *Data Mining and Reverse Engineering* (pp. 428-451): Springer.

Connolly, D., Van Harmelen, F., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2001). DAML+OIL (march 2001) reference description.

Consortium, W. W. W. (2014). RDF 1.1 concepts and abstract syntax.

Csardi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems, 1695*(5), 1-9.

Cui, W., Liu, S., Wu, Z., & Wei, H. (2014). How hierarchical topics evolve in large text corpora. *IEEE Transactions on Visualization and Computer Graphics, 20*(12), 2281-2290.

Del Giudice, M., Caputo, F., & Evangelista, F. (2016). How are decision systems changing? The contribution of social media to the management of decisional liquefaction. *Journal of Decision systems, 25*(3), 214-226.

Diao, Q., Jiang, J., Zhu, F., & Lim, E.-P. (2012). *Finding bursty topics from microblogs.* Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1.

Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: A brief survey. *AI magazine, 26*, 83.

Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., & Halevy, A. (2003). Learning to match ontologies on the semantic web. *The VLDB Journal—The International Journal on Very Large Data Bases, 12*, 303-319.

Doan, A., Noy, N. F., & Halevy, A. Y. (2004). Introduction to the special issue on semantic integration. *ACM SIGMOD Record, 33*, 11-13.

Dong, X. L., & Srivastava, D. (2013). *Big data integration.* 2013 IEEE 29th International Conference on Data Engineering (ICDE).

Dou, W., Yu, L., Wang, X., Ma, Z., & Ribarsky, W. (2013). Hierarchicaltopics: Visually exploring large text collections using topic hierarchies. *IEEE Transactions on Visualization and Computer Graphics, 19*(12), 2002-2011.

Dragut, E., & Lawrence, R. (2004). Composing mappings between schemas using a reference ontology. *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, 783-800.

Evermann, J. (2008a). An exploratory study of database integration processes. *IEEE Transactions on Knowledge and Data Engineering, 20*(1), 99-115.

Evermann, J. (2008b). Theories of meaning in schema matching: A review. *Journal of Database Management, 19*(3), 55.

Evermann, J. (2009). Theories of meaning in schema matching: An exploratory study. *Information Systems, 34*(1), 28-44.

Fensel, D., Van Harmelen, F., Horrocks, I., McGuinness, D. L., & Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE intelligent systems, 16*(2), 38-45.

Fruchterman, T. M., & Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and experience, 21*(11), 1129-1164.

Ghidini, C., & Serafini, L. (2006). *Reconciling concepts and relations in heterogeneous ontologies.* ESWC.

Godin, F., Slavkovikj, V., De Neve, W., Schrauwen, B., & Van de Walle, R. (2013). *Using topic models for twitter hashtag recommendation.* Proceedings of the 22nd International Conference on World Wide Web.

Guizzardi, G., & Wagner, G. (2010). Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages *Theory and applications of ontology: computer applications* (pp. 175-196): Springer.

Haas, L. (2007). Beauty and the beast: The theory and practice of information integration *Database Theory–ICDT 2007* (pp. 28–43): Springer.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*: Elsevier.

Havre, S., Hetzler, B., & Nowell, L. (2000). *ThemeRiver: Visualizing theme changes over time.* Information visualization, 2000. InfoVis 2000. IEEE symposium on.

Heath, T., & Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology, 1*(1), 1-136.

Hendler, J. (2014). Data integration for heterogenous datasets. *Big Data, 2*(4), 205-215.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*, 75-105.

Hirschheim, R., Klein, H. K., & Lyytinen, K. (1995). *Information systems development and data modeling: conceptual and philosophical foundations*: Cambridge University Press.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., & Rudolph, S. (2009). OWL 2 web ontology language primer. *W3C recommendation, 27*(1), 123.

Hofmann, T. (1999). *Probabilistic latent semantic analysis.* Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence.

Hong, L., & Davison, B. D. (2010). *Empirical study of topic modeling in twitter.* Proceedings of the first workshop on social media analytics.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(1), 7-26.

Hu, W., & Qu, Y. (2007). Discovering simple mappings between relational database schemas and ontologies. *The Semantic Web*, 225-238.

Huang, J., Thornton, K. M., & Efthimiadis, E. N. (2010). *Conversational tagging in twitter.* Proceedings of the 21st ACM conference on Hypertext and hypermedia.

Kang, J., & Naughton, J. F. (2003). *On schema matching with opaque column names and data values*.

Kucher, K., & Kerren, A. (2015). *Text visualization techniques: Taxonomy, visual survey, and community insights.* Visualization Symposium (PacificVis), 2015 IEEE Pacific.

Kung, C., & Soelvberg, A. (1986). *Activity modeling and behavior modeling.* Proc. of the IFIP WG 8.1 working conference on Information systems design methodologies: improving the practice.

Lakoff, G. (1987). *Women, fire, and dangerous things*: Chicago: University of Chicago Press.

Lammari, N., Comyn-Wattiau, I., & Akoka, J. (2007). Extracting generalization hierarchies from relational databases: A reverse engineering approach. *Data & Knowledge Engineering, 63*(2), 568-589.

Lau, J. H., Newman, D., Karimi, S., & Baldwin, T. (2010). *Best topic word selection for topic labelling.* Proceedings of the 23rd International Conference on Computational Linguistics: Posters.

LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., & Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. *MIT Sloan Management Review, 52*(2), 21.

Lee, K., Palsetia, D., Narayanan, R., Patwary, M. M. A., Agrawal, A., & Choudhary, A. (2011). *Twitter trending topic classification.* Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on.

Lenzerini, M. (2002). *Data integration: A theoretical perspective*.

Li, J., Tang, J., Li, Y., & Luo, Q. (2008). Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering, 21*(8), 1218-1232.

Lin, J., Snow, R., & Morgan, W. (2011). *Smoothing techniques for adaptive online language models: topic tracking in tweet streams.* Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.

Liu, F., Liu, Y., & Weng, F. (2011). *Why is sxsw trending?: exploring multiple text sources for twitter topic summarization.* Proceedings of the Workshop on Languages in Social Media.

Liu, S., Cui, W., Wu, Y., & Liu, M. (2014a). A survey on information visualization: recent advances and challenges. *The Visual Computer, 30*(12), 1373-1393.

Liu, S., Wang, X., Chen, J., Zhu, J., & Guo, B. (2014b). *Topicpanorama: A full picture of relevant topics.* Visual Analytics Science and Technology (VAST), 2014 IEEE Conference on.

Livne, A., Simmons, M. P., Adar, E., & Adamic, L. A. (2011). The Party Is Over Here: Structure and Content in the 2010 Election. *ICWSM, 11*, 17-21.

Lukyanenko, R., & Parsons, J. (2013). *Is Traditional Conceptual Modeling Going to Become Obsolete?*

Lukyanenko, R., Parsons, J., & Samuel, B. M. (2019). Representing instances: the case for reengineering conceptual modelling grammars. *European Journal of Information Systems, 28*(1), 68-90.

Malthouse, E. C., Haenlein, M., Skiera, B., Wege, E., & Zhang, M. (2013). Managing customer relationships in the social media era: Introducing the social CRM house. *Journal of interactive marketing, 27*(4), 270-280.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1): Cambridge university press Cambridge.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems, 15*(4), 251-266.

McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation, 10*(10), 2004.

Mei, Q., Shen, X., & Zhai, C. (2007). *Automatic labeling of multinomial topic models.* Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.

Moody, D. L. (2005). Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering, 55*(3), 243-276.

Mylopoulos, J. (1992). Conceptual modelling and Telos. *Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development, New York: John Wiley & Sons*, 49-68.

Mylopoulos, J. (1998). Information Modeling in the Time of the Revolution. *Information Systems, 23*(3-4), 127-155.

Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010). *Automatic evaluation of topic coherence.* Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Noy, N. F. (2004). Semantic integration: a survey of ontology-based approaches. *ACM SIGMOD Record, 33*, 65–70.

Olivé, A. (2007). *Conceptual modeling of information systems*: Springer Science & Business Media.

Omelayenko, B. (2002). *RDFT: A mapping meta-ontology for business integration.* Proc. of the Workshop on Knowledge Transformation for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002).

Padrez, K. A., Ungar, L., Schwartz, H. A., Smith, R. J., Hill, S., Antanavicius, T., . . . Merchant, R. M. (2016). Linking social media and medical record data: a study of adults presenting to an academic, urban emergency department. *BMJ Qual Saf, 25*(6), 414-423.

Parsons, J. (1996). An information model based on classification theory. *Management Science, 42*, 1437–1453.

Parsons, J. (2011). An Experimental Study of the Effects of Representing Property Precedence on the Comprehension of Conceptual Schemas*. *Journal of the Association for Information Systems, 12*, 401.

Parsons, J., & Wand, Y. (1997). Choosing classes in conceptual modeling. *Communications of the ACM, 40*, 63–69.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems (TODS), 25*, 228–268.

Parsons, J., & Wand, Y. (2002). Property-Based Semantic Reconciliation of Heterogeneous Information Sources. In S. Spaccapietra, S. T. March, & Y. Kambayashi (Eds.), *Conceptual Modeling — ER 2002* (pp. 351-364): Springer Berlin Heidelberg.

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources *Journal on Data Semantics I* (pp. 21–47): Springer.

Parsons, J., & Wand, Y. (2008). Using cognitive principles to guide classification in information systems modeling. *MIS Quarterly*, 839–868.

Parsons, J., & Wand, Y. (2014). A Foundation for Open Information Environments.

Pöschko, J. (2011). Exploring twitter hashtags. *arXiv preprint arXiv:1111.6553*.

Power, D. J., & Phillips-Wren, G. (2011). Impact of social media and Web 2.0 on decision-making. *Journal of Decision systems, 20*(3), 249-261.

Prat, N., Comyn-Wattiau, I., & Akoka, J. (2014). *Artifact Evaluation in Information Systems Design-Science Research-a Holistic View.* PACIS.

R Core Team. (2000). R language definition. *Vienna, Austria: R foundation for statistical computing*.

Rahm, E. (2011). Towards large-scale schema and ontology matching *Schema matching and mapping* (pp. 3-27): Springer.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal, 10*, 334-350.

Ram, S., Zhang, W., Williams, M., & Pengetnze, Y. (2015). Predicting asthma-related emergency department visits using big data. *IEEE journal of biomedical and health informatics, 19*(4), 1216-1223.

Ramage, D., Dumais, S., & Liebling, D. (2010). Characterizing microblogs with topic models. *ICWSM, 10*(1), 16.

Ramage, D., Hall, D., Nallapati, R., & Manning, C. D. (2009). *Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora.* Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1.

Recker, J. (2015). *Research on conceptual modelling: less known knowns and more unknown unknowns, please.* Proceedings of the 11th Asia-Pacific Conference on Conceptual Modelling.

Recker, J., Rosemann, M., Green, P., & Indulska, M. (2011). Do ontological deficiencies in modeling grammars matter? *MIS Quarterly, 35*(1), 57-79.

Recker, J., Rosemann, M., Indulska, M., & Green, P. (2009). Business process modeling-a comparative analysis. *Journal of the Association for Information Systems, 10*(4), 1.

Roussopoulos, N., & Karagiannis, D. (2009). Conceptual modeling: past, present and the continuum of the future *Conceptual modeling: Foundations and applications* (pp. 139-152): Springer.

Russom, P. (2011). Big data analytics. *TDWI Best Practices Report, Fourth Quarter, 19*(4), 1-34.

Sanderson, M., & Croft, B. (1999). *Deriving concept hierarchies from text.* Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.

Schmitz, P. (2006). *Inducing ontology from flickr tags.* Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland.

Shanks, G., Tansley, E., & Weber, R. (2003). Using ontology to validate conceptual models. *Communications of the ACM, 46*(10), 85-89.

Sheth, A. (1997). Panel: Data semantics: what, where and how? *Database Applications Semantics* (pp. 601-610): Springer.

Sheth, A. P. (1999). Changing Focus on Interoperability in Information Systems:From System, Syntax, Structure to Semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, & C. Kottman (Eds.), *Interoperating Geographic Information Systems* (pp. 5-29): Springer US.

Shvaiko, P., & Euzenat, J. (2005). A survey of schema-based matching approaches *Journal on data semantics IV* (pp. 146-171): Springer.

Šilić, A., & Bašić, B. D. (2010). *Visualization of text streams: A survey.* International Conference on Knowledge-Based and Intelligent Information and Engineering Systems.

Smith, E. E. (1988). Concepts and thought. *The psychology of human thought, 147*.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts* (Vol. 9): Harvard University Press Cambridge, MA.

Spanos, D.-E., Stavrou, P., & Mitrou, N. (2012). Bringing relational databases into the semantic web: A survey. *Semantic Web, 3*(2), 169-209.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., & Demirbas, M. (2010). *Short text classification in twitter to improve information filtering.* Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval.

Statista. (2018). Dossier Details: Twitter. Retrieved from https://www.statista.com/study/9920/twitter-statista-dossier/

Suh, B., Hong, L., Pirolli, P., & Chi, E. H. (2010). *Want to be retweeted? large scale analytics on factors impacting retweet in twitter network.* Social computing (socialcom), 2010 ieee second international conference on.

Sun, X., Xiao, Y., Wang, H., & Wang, W. (2015). *On Conceptual Labeling of a Bag of Words.* IJCAI.

Šváb-Zamazal, O., & Svátek, V. (2009). *Towards ontology matching via pattern-based detection of semantic structures in owl ontologies.* Proceedings of the Znalosti Czecho-Slovak Knowledge Technology conference.

Turney, P. D., & Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research, 37*, 141-188.

Twitter Usage Statistics. (2018). Retrieved from http://www.internetlivestats.com/twitter-statistics/.

Uschold, M., & Gruninger, M. (2004). Ontologies and semantics for seamless connectivity. *ACM SIGMOD Record, 33*, 58-64.

Van der Aalst, W. M. (2013). Business process management: a comprehensive survey. *ISRN Software Engineering, 2013*.

Volz, R., Handschuh, S., Staab, S., Stojanovic, L., & Stojanovic, N. (2004). Unveiling the hidden bride: deep annotation for mapping and migrating legacy data to the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(2), 187-206.

Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001). *Ontology-based integration of information-a survey of existing approaches*.

Wand, Y., Monarchi, D. E., Parsons, J., & Woo, C. C. (1995). Theoretical foundations for conceptual modelling in information systems development. *Decision Support Systems, 15*, 285–304.

Wand, Y., & Weber, R. (1990). An Ontological Model of an Information System. *IEEE Transactions on Software Engineering, 16*, 1282-1292.

Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal, 3*, 217–237.

Wand, Y., & Weber, R. (2002). Research commentary: information systems and conceptual modeling—a research agenda. *Information Systems Research, 13*(4), 363-376.

Woods, W. A. (1975). What's in a link: Foundations for semantic networks *Representation and understanding* (pp. 35-82): Elsevier.

Wu, F. (2015). *Emergent ontology discovered from folksonomies.* Memorial University of Newfoundland. Primo database.

Yan, X., Guo, J., Lan, Y., & Cheng, X. (2013). *A biterm topic model for short texts.* Proceedings of the 22nd international conference on World Wide Web.

Yang, L., Sun, T., Zhang, M., & Mei, Q. (2012). *We know what@ you# tag: does the dual role affect hashtag adoption?* Proceedings of the 21st international conference on World Wide Web.

Yang, S., Kolcz, A., Schlaikjer, A., & Gupta, P. (2014). *Large-scale high-precision topic modeling on twitter.* Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.

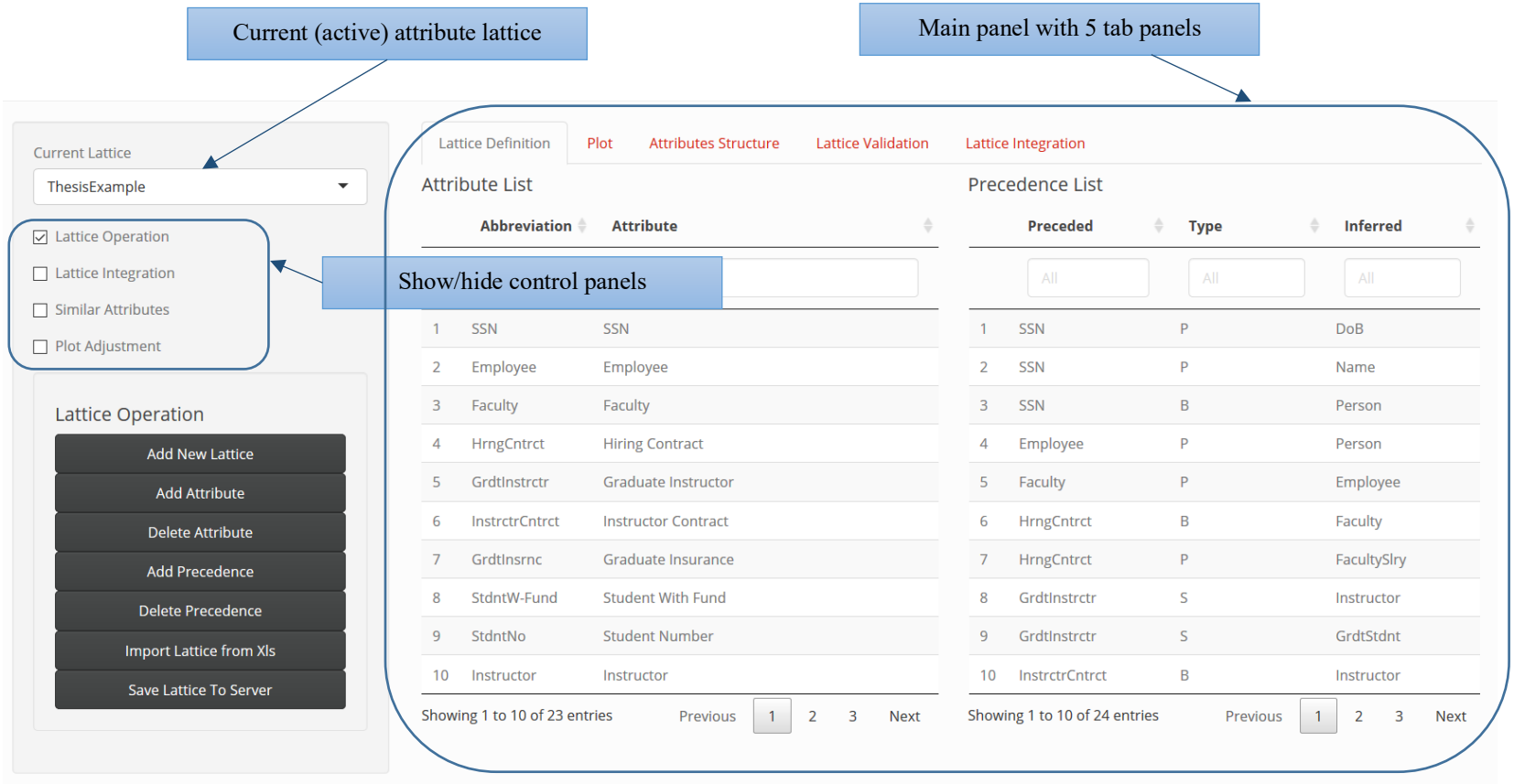
Zhai, C., & Massung, S. (2016). *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*: Morgan & Claypool.

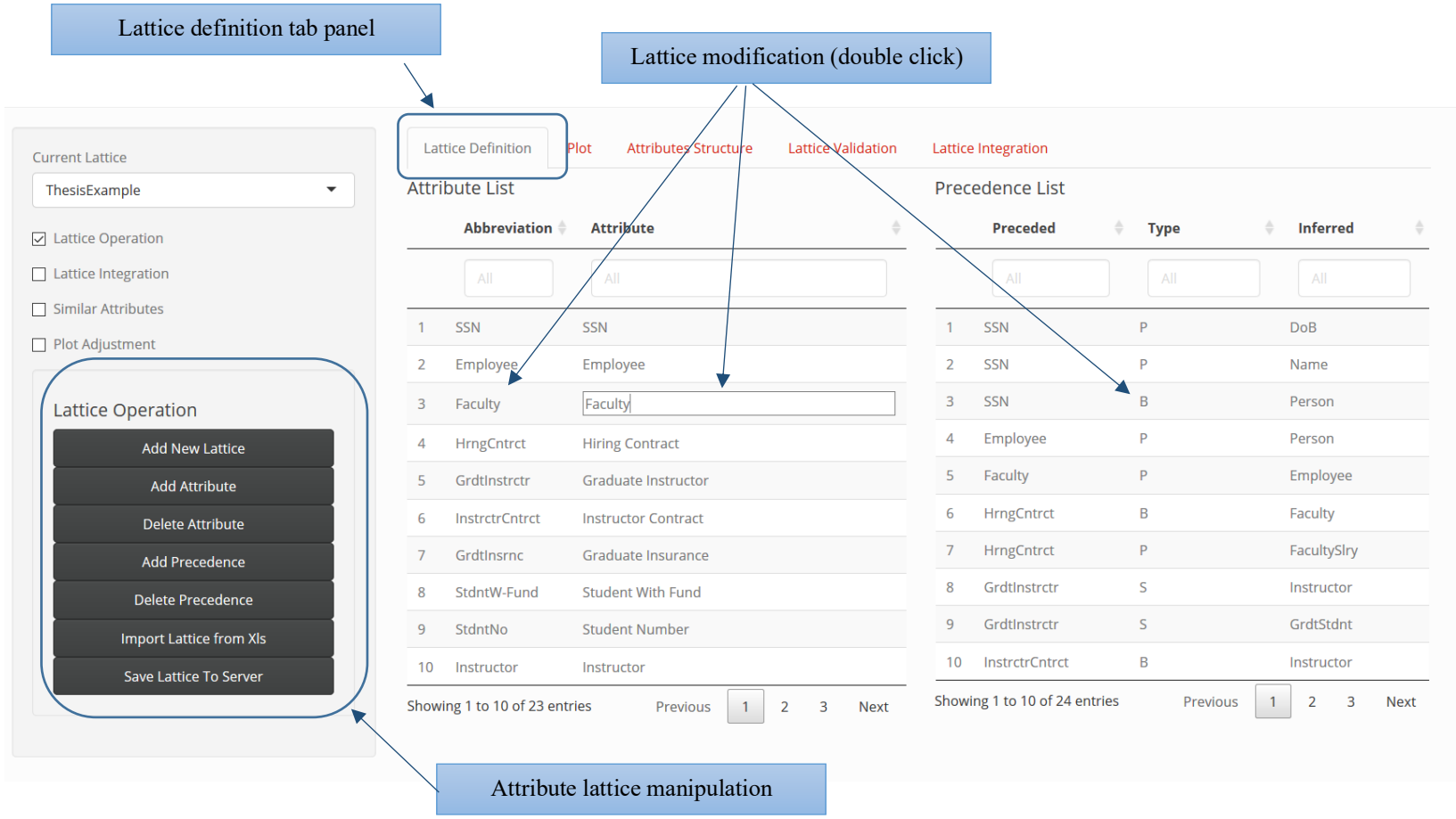
Zhao, D., & Rosson, M. B. (2009). *How and why people Twitter: the role that micro-blogging plays in informal communication at work.* Proceedings of the ACM 2009 international conference on Supporting group work.

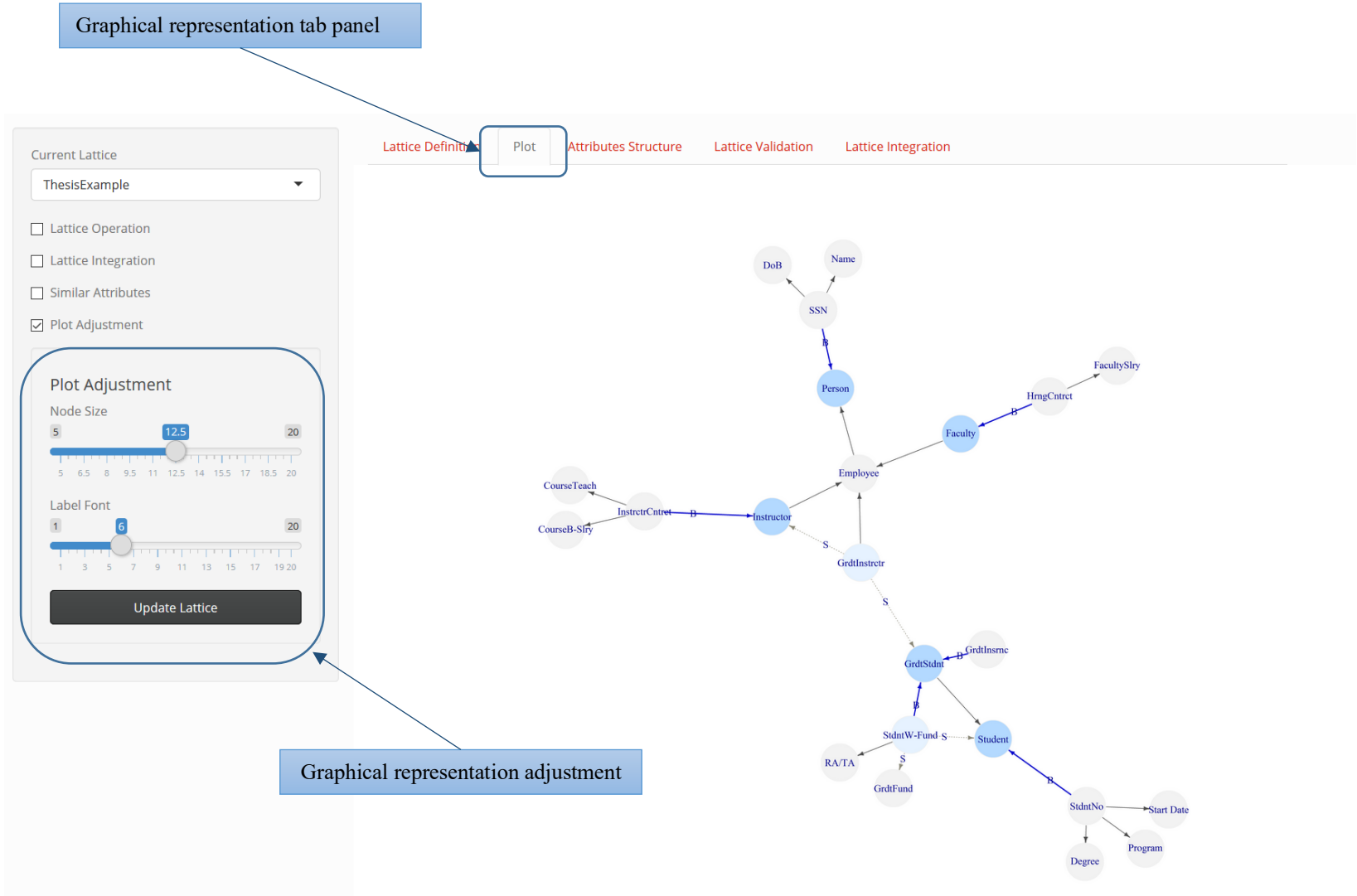
Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., & Li, X. (2011). *Comparing twitter and traditional media using topic models.* European Conference on Information Retrieval.

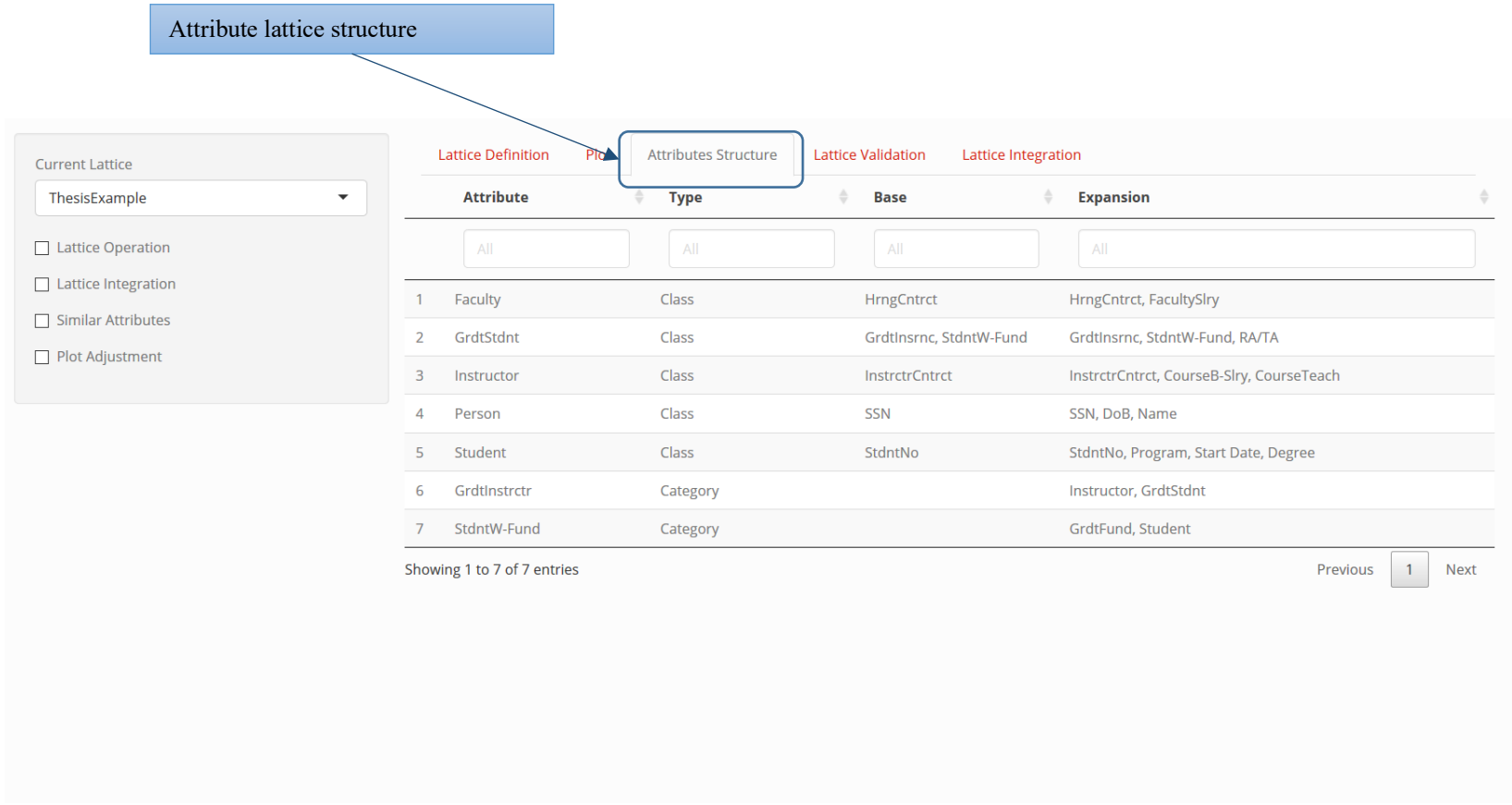
Zimmer, M., & Proferes, N. J. (2014). A topology of Twitter research: disciplines, methods, and ethics. *Aslib Journal of Information Management, 66*(3), 250-261.

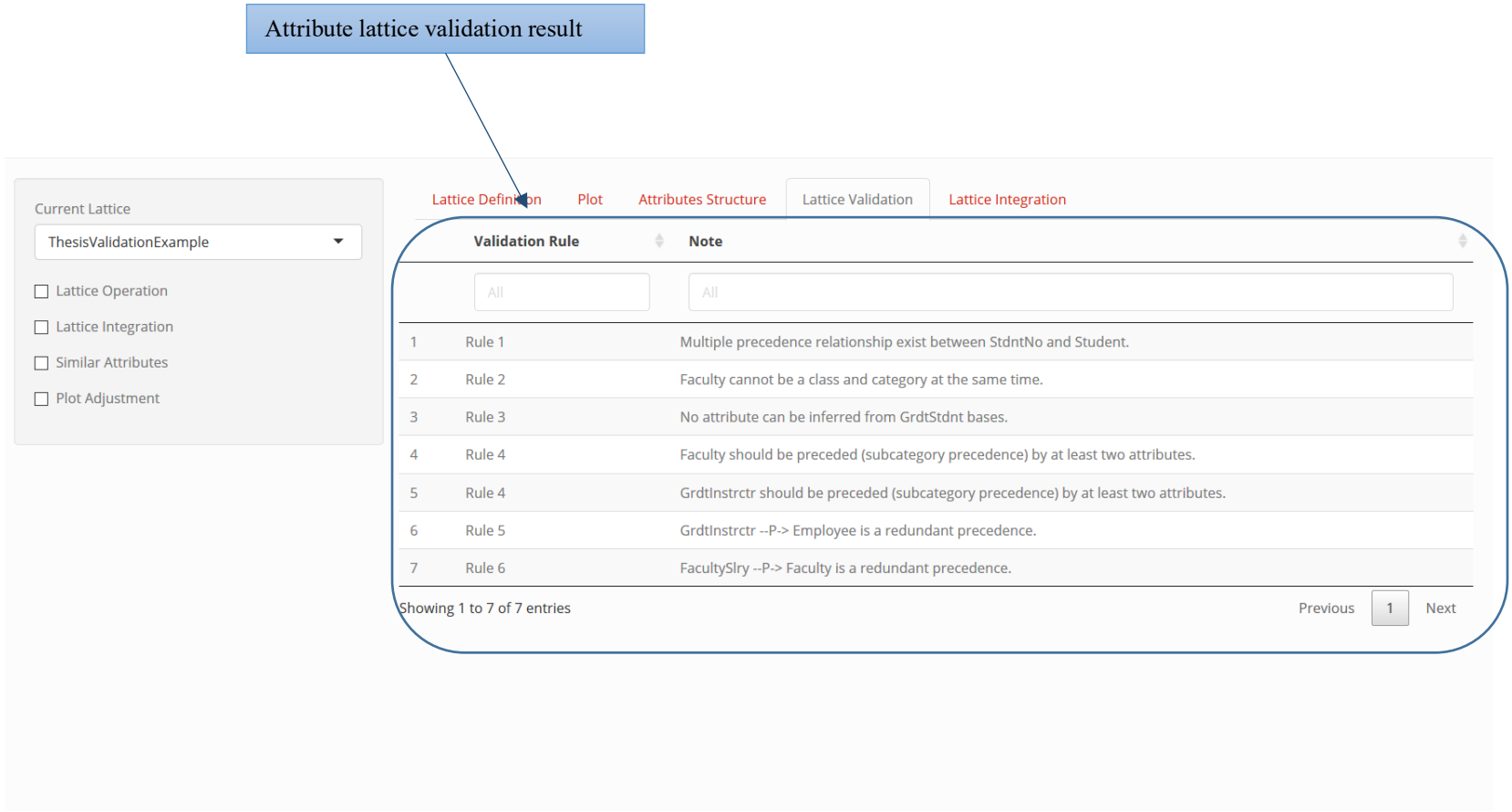
APPENDIX A (Implemented Artifact; Chapter 3)

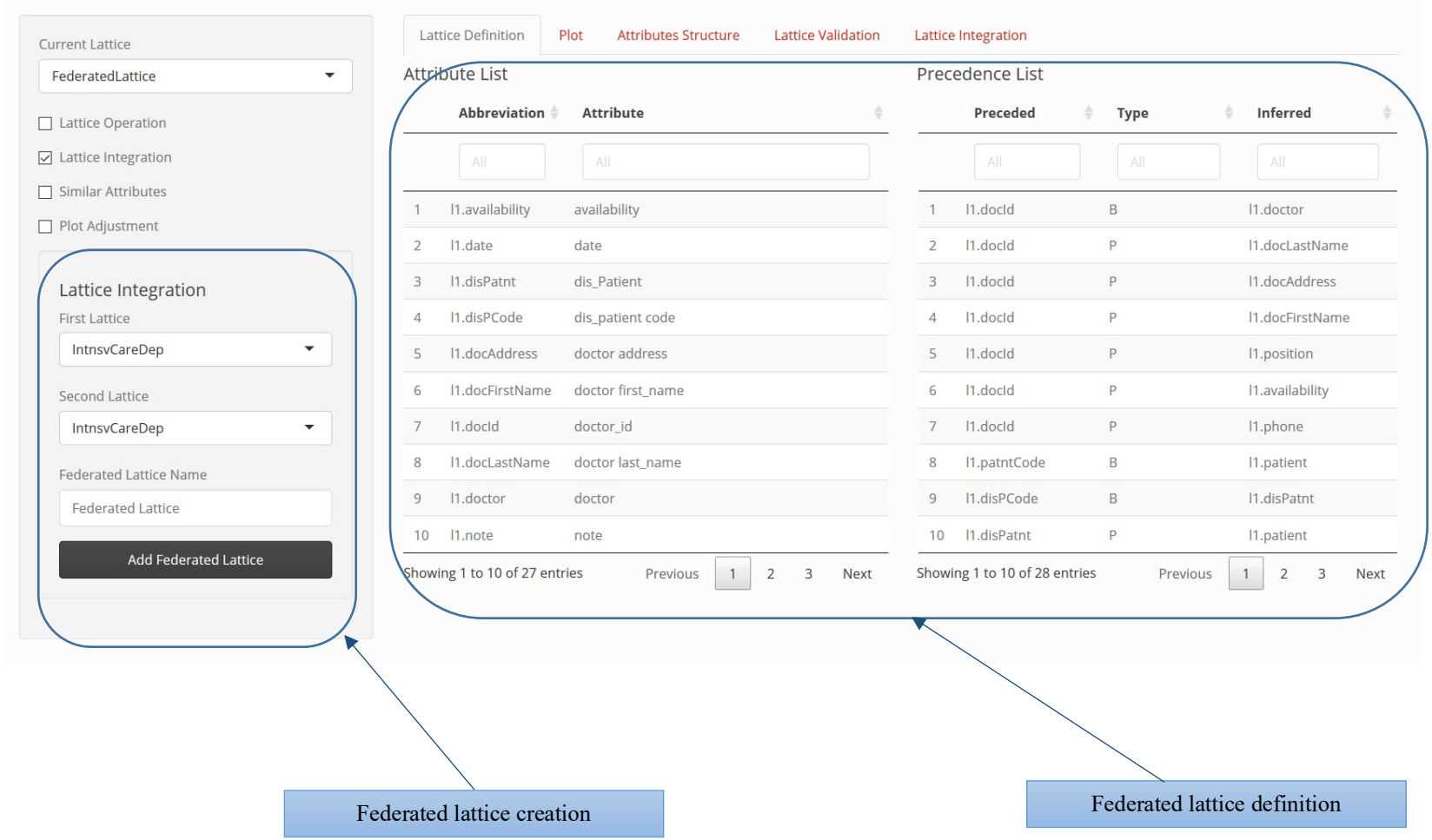


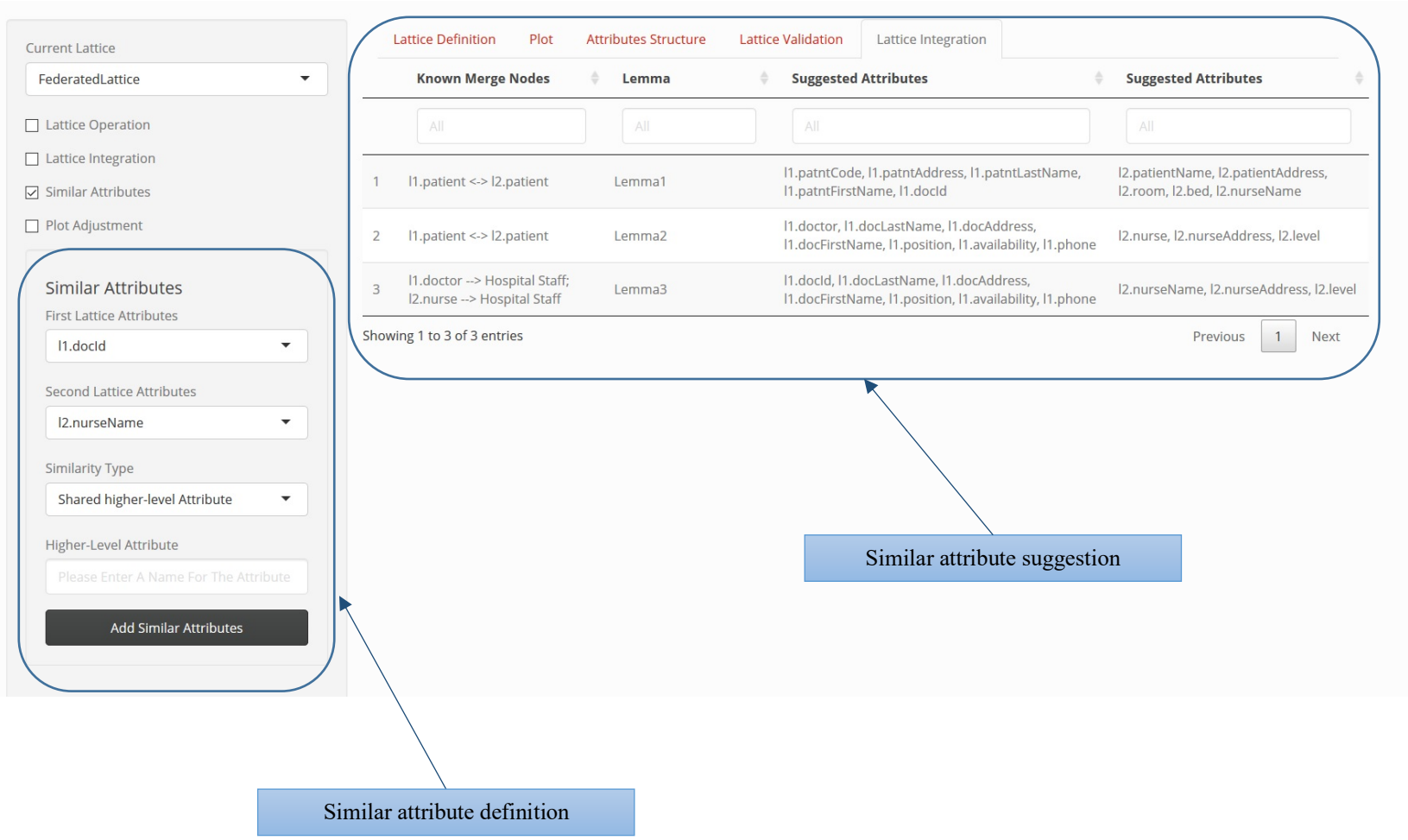

Lattice definition tab panel
Lattice modification (double click)
Current Lattice
ThesisExample
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Lattice Operation
Add New Lattice
Add Attribute
Delete Attribute
Add Precedence
Delete Precedence
Import Lattice from Xls
Save Lattice To Server
Lattice Definition
Plot
Attributes Structure
Lattice Validation
Lattice Integration
Attribute List
Abbreviation
Attribute
1 SSN SSN
2 Employee Employee
3 Faculty Faculty
4 HrngCntrct Hiring Contract
5 GrdtInstrctr Graduate Instructor
6 InstrctrCntrct Instructor Contract
7 GrdtInsrnc Graduate Insurance
8 StdntW-Fund Student With Fund
9 StdntNo Student Number
10 Instructor Instructor
Showing 1 to 10 of 23 entries
Previous 1 2 3 Next
Precedence List
Preceded
Type
Inferred
1 SSN P DoB
2 SSN P Name
3 SSN B Person
4 Employee P Person
5 Faculty P Employee
6 HrngCntrct B Faculty
7 HrngCntrct P FacultySlry
8 GrdtInstrctr S Instructor
9 GrdtInstrctr S GrdtStdnt
10 InstrctrCntrct B Instructor
Showing 1 to 10 of 24 entries
Previous 1 2 3 Next
Attribute lattice manipulation


Graphical representation tab panel
Lattice Definition
Plot
Attributes Structure
Lattice Validation
Lattice Integration
Current Lattice
ThesisExample
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Plot Adjustment
Node Size
5
12.5
20
5 6.5 8 9.5 11 12.5 14 15.5 17 18.5 20
Label Font
1
6
20
1 3 5 7 9 11 13 15 17 19 20
Update Lattice
Graphical representation adjustment
DoB
Name
SSN
B
Person
FacultySlry
HrngCntrct
B
Faculty
Employee
CourseTeach
InstrctrCntrct
B
Instructor
CourseB-Slry
S
GrdtInstrctr
S
GrdtStdnt
B
GrdtInsrnc
B
StdntW-Fund
S
Student
RA/TA
S
GrdtFund
B
StdntNo
Start Date
Program
Degree

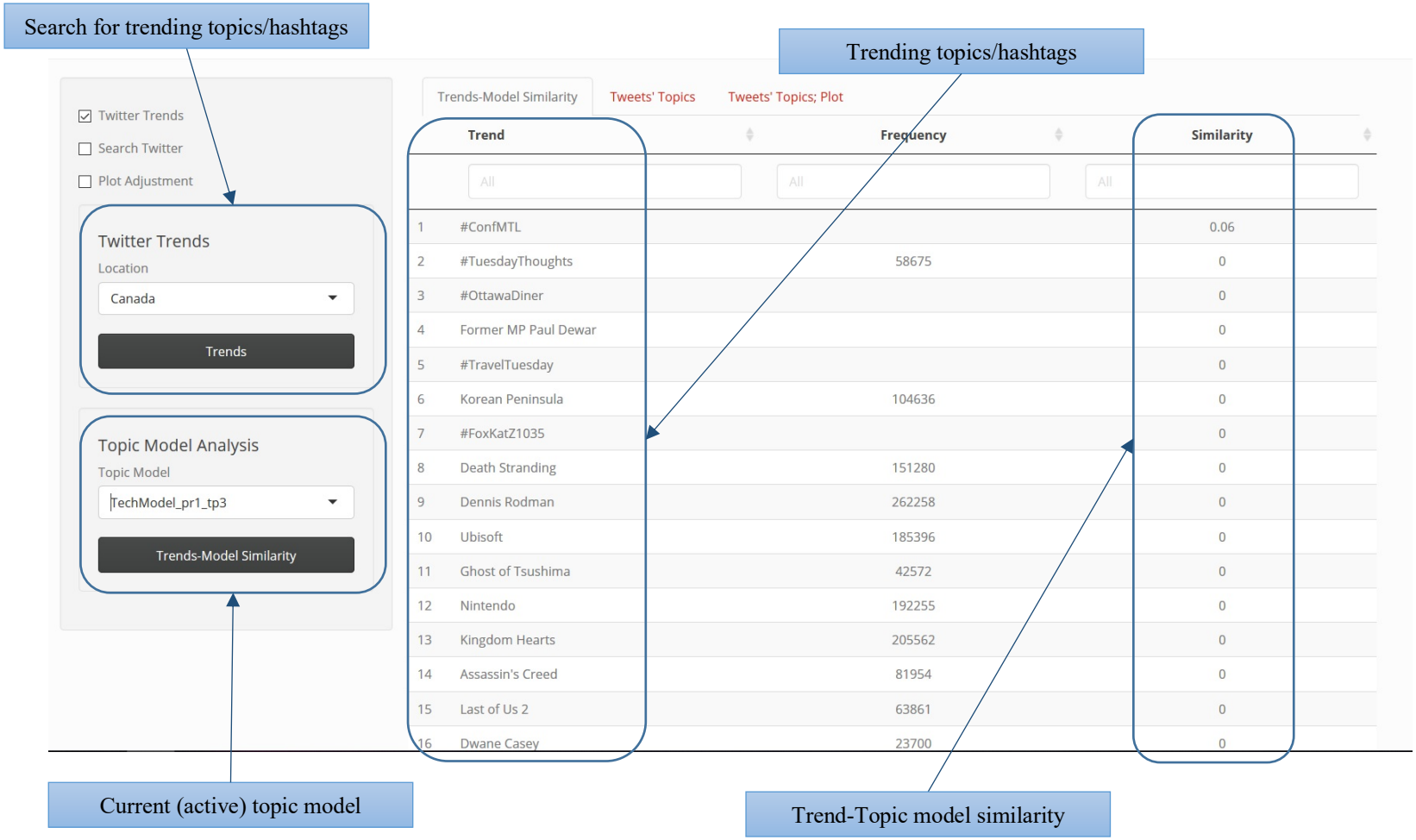

Attribute lattice structure
Current Lattice
ThesisExample
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Lattice Definition
Attributes Structure
Lattice Validation
Lattice Integration
Attribute
Type
Base
Expansion
All
1 Faculty Class HrngCntrct HrngCntrct, FacultySlry
2 GrdtStdnt Class GrdtInsrnc, StdntW-Fund GrdtInsrnc, StdntW-Fund, RA/TA
3 Instructor Class InstrctrCntrct InstrctrCntrct, CourseB-Slry, CourseTeach
4 Person Class SSN SSN, DoB, Name
5 Student Class StdntNo StdntNo, Program, Start Date, Degree
6 GrdtInstrctr Category Instructor, GrdtStdnt
7 StdntW-Fund Category GrdtFund, Student
Showing 1 to 7 of 7 entries
Previous
1
Next

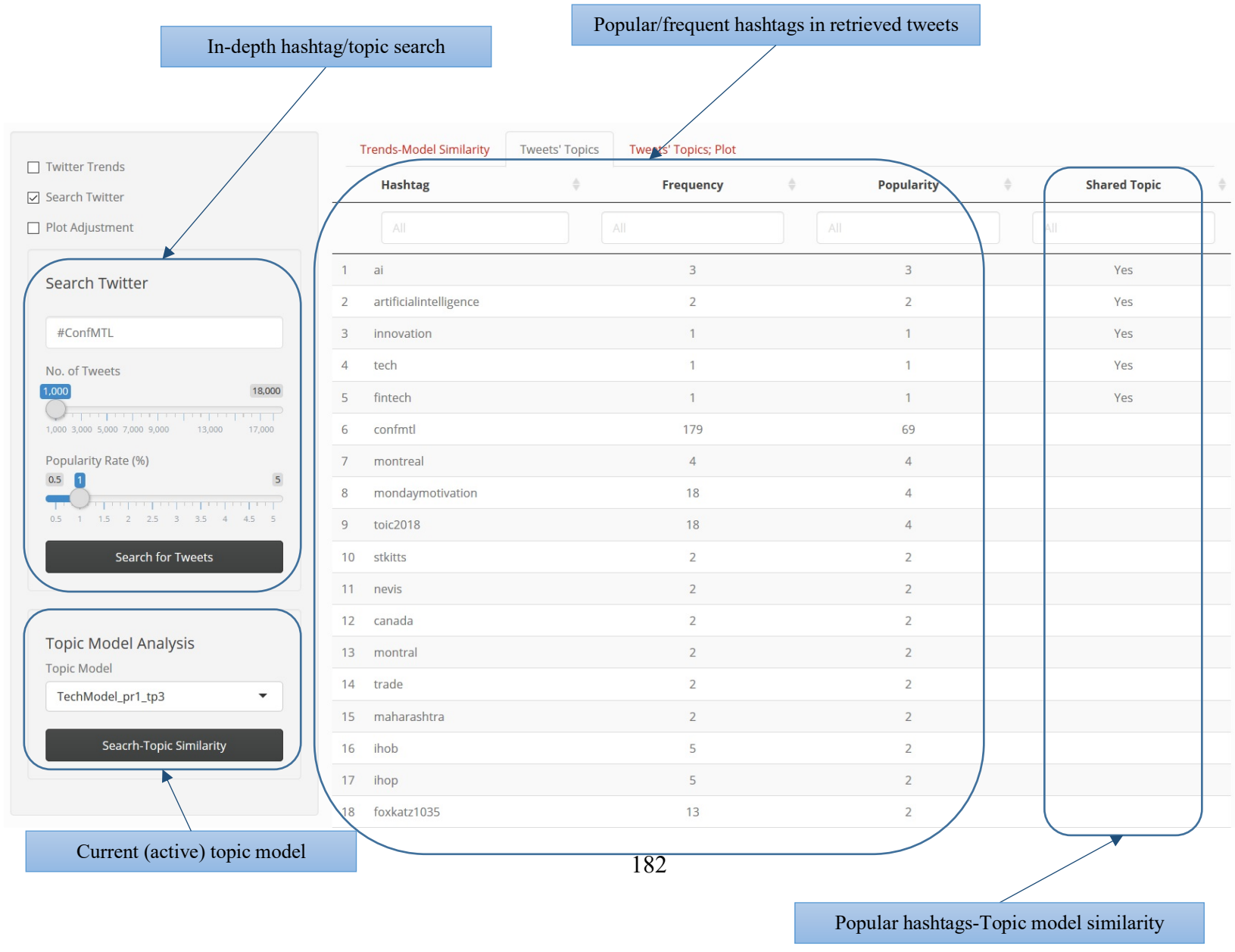

Attribute lattice validation result
Current Lattice
ThesisValidationExample
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Lattice Definition
Plot
Attributes Structure
Lattice Validation
Lattice Integration
Validation Rule
Note
All
All
1 Rule 1 Multiple precedence relationship exist between StdntNo and Student.
2 Rule 2 Faculty cannot be a class and category at the same time.
3 Rule 3 No attribute can be inferred from GrdtStdnt bases.
4 Rule 4 Faculty should be preceded (subcategory precedence) by at least two attributes.
5 Rule 4 GrdtInstrctr should be preceded (subcategory precedence) by at least two attributes.
6 Rule 5 GrdtInstrctr --P-> Employee is a redundant precedence.
7 Rule 6 FacultySlry --P-> Faculty is a redundant precedence.
Showing 1 to 7 of 7 entries
Previous
1
Next

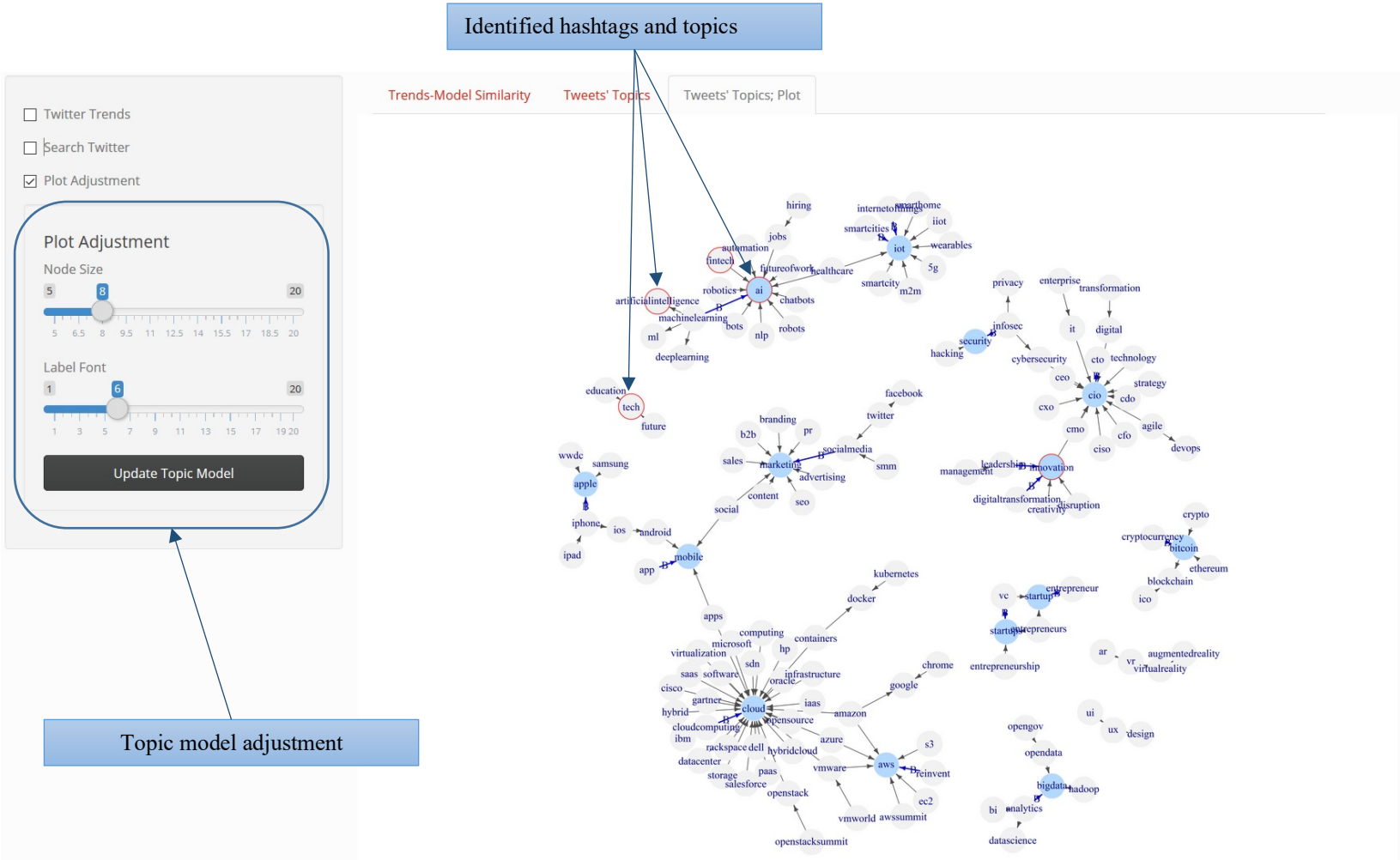

Current Lattice
FederatedLattice
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Lattice Integration
First Lattice
IntnsvCareDep
Second Lattice
IntnsvCareDep
Federated Lattice Name
Federated Lattice
Add Federated Lattice
Lattice Definition
Plot
Attributes Structure
Lattice Validation
Lattice Integration
Attribute List
Abbreviation
Attribute
1 l1.availability availability
2 l1.date date
3 l1.disPatnt dis_Patient
4 l1.disPCode dis_patient code
5 l1.docAddress doctor address
6 l1.docFirstName doctor first_name
7 l1.docId doctor_id
8 l1.docLastName doctor last_name
9 l1.doctor doctor
10 l1.note note
Showing 1 to 10 of 27 entries
Previous 1 2 3 Next
Precedence List
Preceded
Type
Inferred
1 l1.docId B l1.doctor
2 l1.docId P l1.docLastName
3 l1.docId P l1.docAddress
4 l1.docId P l1.docFirstName
5 l1.docId P l1.position
6 l1.docId P l1.availability
7 l1.docId P l1.phone
8 l1.patntCode B l1.patient
9 l1.disPCode B l1.disPatnt
10 l1.disPatnt P l1.patient
Showing 1 to 10 of 28 entries
Previous 1 2 3 Next
Federated lattice creation
Federated lattice definition


Current Lattice
FederatedLattice
Lattice Operation
Lattice Integration
Similar Attributes
Plot Adjustment
Similar Attributes
First Lattice Attributes
l1.docId
Second Lattice Attributes
l2.nurseName
Similarity Type
Shared higher-level Attribute
Higher-Level Attribute
Please Enter A Name For The Attribute
Add Similar Attributes
Lattice Definition
Plot
Attributes Structure
Lattice Validation
Lattice Integration
Known Merge Nodes
Lemma
Suggested Attributes
Suggested Attributes
All
1
l1.patient <-> l2.patient
Lemma1
l1.patntCode, l1.patntAddress, l1.patntLastName, l1.patntFirstName, l1.docId
l2.patientName, l2.patientAddress, l2.room, l2.bed, l2.nurseName
2
l1.patient <-> l2.patient
Lemma2
l1.doctor, l1.docLastName, l1.docAddress, l1.docFirstName, l1.position, l1.availability, l1.phone
l2.nurse, l2.nurseAddress, l2.level
3
l1.doctor --> Hospital Staff; l2.nurse --> Hospital Staff
Lemma3
l1.docId, l1.docLastName, l1.docAddress, l1.docFirstName, l1.position, l1.availability, l1.phone
l2.nurseName, l2.nurseAddress, l2.level
Showing 1 to 3 of 3 entries
Previous
1
Next
Similar attribute suggestion
Similar attribute definition

APPENDIX B (Implemented Artifact; Topic modeling on Twitter)




In-depth hashtag/topic search
Popular/frequent hashtags in retrieved tweets
Twitter Trends
Search Twitter
Plot Adjustment
Search Twitter
#ConfMTL
No. of Tweets
1,000
18,000
1,000 3,000 5,000 7,000 9,000 13,000 17,000
Popularity Rate (%)
0.5
1
5
0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
Search for Tweets
Topic Model Analysis
Topic Model
TechModel_pr1_tp3
Seacrh-Topic Similarity
Trends-Model Similarity
Tweets' Topics
Tweets' Topics; Plot
Hashtag
Frequency
Popularity
Shared Topic
All
1 ai 3 3 Yes
2 artificialintelligence 2 2 Yes
3 innovation 1 1 Yes
4 tech 1 1 Yes
5 fintech 1 1 Yes
6 confmtl 179 69
7 montreal 4 4
8 mondaymotivation 18 4
9 toic2018 18 4
10 stkitts 2 2
11 nevis 2 2
12 canada 2 2
13 montral 2 2
14 trade 2 2
15 maharashtra 2 2
16 ihob 5 2
17 ihop 5 2
18 foxkatz1035 13 2
Current (active) topic model
Popular hashtags-Topic model similarity


Identified hashtags and topics
Twitter Trends
Search Twitter
Plot Adjustment
Plot Adjustment
Node Size
Label Font
Update Topic Model
Trends-Model Similarity
Tweets' Topics
Tweets' Topics; Plot
Topic model adjustment