Supporting OLAP Operations over Imperfectly Integrated Taxonomies

Yan Qi
Arizona State Univ.
Tempe, AZ 85287
yan.qi@asu.edu

K. Selçuk Candan
Arizona State Univ.
Tempe, AZ 85287
candan@asu.edu

Junichi Tatemura
NEC Laboratories America
Cupertino, CA 95014
tatemura@sv.nec-labs.com

Songting Chen
NEC Laboratories America
Cupertino, CA 95014
songting@sv.nec-labs.com

Fenglin Liao
University of California Santa Barbara
Santa Barbara, CA 93106
fenglin@cs.ucsb.edu

ABSTRACT

OLAP is an important tool in decision support. With the help of domain knowledge, such as hierarchies of attribute values, OLAP helps the user observe the effects of various decisions. One assumption of most OLAP operations is that the available domain knowledge is precise. In particular, they assume that the hierarchy of values over which the user can navigate forms a taxonomy. In this paper, we first note that when multiple heterogeneous data sources are involved in the gathering of the data and the associated domain knowledge, the integrated knowledge-base, constructed by combining locally available taxonomies based on the concept matchings, may not be a taxonomy itself. Specifically, existence of intersections among concepts from different sources compromises the tree-structure of the integrated taxonomy and prevents effective use of hierarchical navigation techniques, such as drill-down and roll-up. To cope with this, we introduce *concept un-classification*, where a select few of the concepts are eliminated to ensure that the remaining structure is a navigable taxonomy, without concept intersections. Since un-classifying an originally classified data is not desirable, we consider ways to minimize un-classification in the process. We introduce a cost model which captures the imprecision caused by the un-classification process and we formulate the problem of finding an un-classification strategy which eliminates intersections and which adds minimal imprecision to the resulting structure. We show that, when performed naively, this task can be very costly and thus we propose a bottom-up preprocessing strategy which supports basic navigational analytics operations, such as drill-down and roll-up. Experiments over synthetic and real-life data verified the effectiveness and efficiency of our approach.

Categories and Subject Descriptors

H.2 [**Database Management**]

General Terms

Algorithms, Measurement

Keywords

OLAP, imperfect integration, taxonomy correction, imprecise data

1. INTRODUCTION

OLAP is an important tool to support business decisions through analytical queries over multidimensional (multi-attribute) data such as sales information. The value of each dimension (e.g., business divisions, product names) is typically associated with a structure of hierarchical categories (i.e., a taxonomy), letting the user aggregate data along with this hierarchy.

Besides the traditional use cases within an enterprise, navigational operations in OLAP are also useful to understand data in more general cases such as shopping decisions in e-commerce. A typical e-commerce site provides a search result of items (e.g., auction) divided into hierarchical categories. Navigation over such hierarchical structure helps the user not only to locate a specific item but also to understand distribution of available items within the categorical space. In this paper we focus on providing support for the navigational OLAP operators, such as *drill-down*, which navigates from less specific data to more detailed data, by stepping down on a given hierarchy of concepts (i.e., taxonomy), and *roll-up*, which performs aggregation on a set of data objects by climbing up a hierarchy of concepts (i.e., the given taxonomy).

EXAMPLE 1.1 (DRILL-DOWN). *Consider the two data sets shown in Figure 1. In this figure, we see that each data set comprises of a set of data objects and a taxonomy which organizes and clusters these data objects.*

The drill-down operation starts at the root of the taxonomy (i.e., at the concept of 'America' for data set, D_1) and provides user with statistics (such as count) regarding the *objects that are covered under this concept (count = 7 for 'America' in this example). The user can then* drill-down *from the root to get more detailed information about the sub-concepts('west' and 'east' for D_1). Compared with their parent, these sub-concepts represent smaller disjoint concept spaces.*

A challenge in wider application of OLAP operations, especially in the Web, is how to support efficient navigation over multiple data sets maintained by autonomous organizations. These data sets cannot always agree on a standard taxonomy since they have own objectives: for instance, one online music shop might be specialized for jazz music and want a customized taxonomy categorizing music pieces appropriately. The Semantic Web has been advocated to address integration between such heterogeneous concept structures: OWL [2] is a standard language that lets the user describe not only a concept structure but also a mapping between different concept structures. Various tools and techniques have been developed for (semi-)automated matching between concepts [14, 25, 10, 26]. However, there remains a question: Can we enjoy OLAP operations over the data set after integration?

In this paper, we first note that the integrated concept graph, constructed by combining locally available taxonomies based on the concept matchings, may not be a taxonomy itself (Section 3.3). Specifically, existence of intersections among concepts from different sources compromises the tree-structure of the integrated taxonomy and prevents effective use of hierarchical navigation techniques, such as drill-down and roll-up. To cope with this, we propose elimination of concepts from the integrated concept structure in such a way that the remaining concepts is a navigable taxonomy without concept intersections. Note that this elimination introduces *imprecision* [22, 6] – when a concept (e.g., San Jose, CA) is eliminated, data objects classified in it need to be *un-classified*.

We formulate this process in terms of the task of finding an un-classification strategy which eliminates intersections and adds only minimal imprecision to the resulting structure. We show that when performed naively, this can be very costly in terms of imprecisions introduced and thus we propose an effective bottom-up preprocessing strategy which optimally eliminates intersections. Our contributions in this paper are the following:

- To the best of our knowledge, ours is the first work that addresses the problem of finding navigable taxonomies for OLAP operations over a data set derived from integration between heterogeneous taxonomies.
- We introduce the notion of *un-classification of concepts* for removal of intersections and present a cost model to capture the resulting imprecision in the data. We then formulate the problem of finding an un-classification strategy which eliminates intersections and adds only minimal imprecision to the resulting structure.
- We propose an optimal, yet efficient, taxonomy cleaning algorithm, based on a bottom-up correction strategy, to support navigational OLAP operations.

Experiments presented in Section 5 verify the effectiveness and efficiency of our approach.

2. RELATED WORK

The concept of imprecise data in terms of OLAP dimensions is discussed in [22]. In that framework, a fact (i.e. data object) with imprecise data is associated with dimension values of coarser granularities, resulting in the dimensional imprecision hierarchy. Given an aggregate query, the system determines if the result should be precise enough. Burdick *et al.* take further steps with an allocation-based approach [5, 6, 7] to estimate aggregation values even when the data is not precise for a given query. Their work is complementary to ours: after we derive a navigable taxonomy with minimal imprecision, we can further apply their approach to derive aggregate values.

In schema design for traditional relational-OLAP systems, issues around heterogeneous dimensions have been discussed [18, 17]. A dimension is called heterogeneous if two members in a given category are allowed to have ancestors in different categories. In such a dimension, an aggregate view for a category may not be correctly derived from views for its sub-categories (a summarizability problem).

Fagin *et al.* [15] propose a data framework, called the *multi-structural database,* to support analysis of complex data sets. In this framework, each dimension is formalized as a lattice in order to incorporate both numerical and categorical dimensions. They introduce several operations over this data model based on the notion of pairwise disjoint collection (PDC). A DIVIDE operation is similar to the drill-down operation and breaks data set into k conceptually disjoint collections. Although the problem setting is different, we share the same intuition with them: for navigation and analysis of data, a concept should be divided into conceptually disjoint sub-concepts.

Several approaches have been proposed to automatically divide a query result into sub-categories based on multidimensional values [8, 27, 9]. Chakrabarti et al. [8] propose techniques to automatically categorize results of SQL queries on a relational database. Wu et al. [27] propose a framework, KDAP (Keyword-Driven Analytical Processing), to support keyword-based search over OLAP data. Both approaches support automated partitioning over data with numerical and categorical attributes based on the cost model of user overload [8] or pluggable ranking functions on application-specific interestingness [27]. To our knowledge, none of the existing work targets the problem of integrated taxonomies.

Conflict resolution and uncertainty in data integration has been studied intensively [4, 19, 23, 16, 3]. Recently on-line (query time) resolution approaches are proposed since the best resolution is often context-dependent. For example, Yan *et al.* [24] propose a feedback-based approach to query processing in the presence of conflicts and alternative mapping interpretations in integrated data. Our work can be seen as another example of dynamic conflict resolution in data integration. Whereas their optimization is based on a given query structure (as well as user feedback), ours is based on imprecision to represent a data set (e.g., query results).

3. PRELIMINARIES

In this section, we discuss why data integration may result in concept graphs which are not suitable for common OLAP operations, such as drill-down and roll-up, and then formulate the problem.

3.1 Data Set and Taxonomies

We first define the concept of *data set*, which captures a single consistent data source:

DEFINITION 3.1 (DATA SET). *A data set D, is a triple (O, X, R), where*

- *O is a set $\{o_1, o_2, ..., o_n\}$ of data objects;*
- *$X = \{T_1, ..., T_n\}$ is a set of dimensions to describe objects in O.*

  Each dimension $T_i \in X$ is represented using a taxonomy, $T_i = (V_i, E_i)$, where

  - *V_i is a set of terms[1], and*
  - *$E \subseteq V \times V \times \{\overset{\geq}{\rightarrow}\}$ is a set of subsumption relations.*

  Here '$\overset{\geq}{\rightarrow}$' is the edge label indicating a (transitive) subsumption relation over V_i. Given $x, y \in V$, $(x, y, \overset{\geq}{\rightarrow}) \in E$ indicates that x subsumes y in concept.

- *$R \subseteq O \times \bigcup_i V_i$, is a membership relation set specifying the elements of the taxonomies to which data objects in the set belong.*

In this paper, we assume that each data dimension is described by a taxonomy (or a hierarchical structure), in which each node represents one concept (or term) and each edge indicates a subsumption relation, $\overset{\geq}{\rightarrow}$. Naturally, a given data set can have multiple dimensions, each describing the data from different views. During OLAP operations, each dimension can be used independently to explore the data from different angles. In the rest of the paper, without loss of generality, we focus on a single dimension of the data and as a matter of convenience use (O, T, R) as a shorthand for (O, X, R), where X is a singleton set $X = \{T\}$ of the relevant dimension of D.

Ideally, the taxonomy, T, of D should categorize all data objects in O precisely:

DEFINITION 3.2 (GENERAL TAXONOMY). *Given a data set $D = (O, T, R)$, we say it has a* (general) taxonomy *$T = (V, E)$, iff*

- *$\forall \langle o, t \rangle \in R$ where $o \in O$ and $t \in V$, t is a node in T, and*
- *$\forall \langle o_i, t_j \rangle, \langle o_k, t_l \rangle \in R$, $(t_j \neq t_l) \rightarrow (o_i \neq o_j)$.*

In many OLAP systems, however, each data object is assigned to a leaf in the hierarchical taxonomy. We call these leaf-centric taxonomies.

DEFINITION 3.3 (LEAF-CENTRIC TAXONOMY). *Given a data set $D = (O, T, R)$, we say it has a* leaf-centric taxonomy *$T = (V, E)$, iff*

- *$\forall \langle o, t \rangle \in R$ where $o \in O$ and $t \in V$, t is a leaf node in T, and*
- *$\forall \langle o_i, t_j \rangle, \langle o_k, t_l \rangle \in R$, $(t_j \neq t_l) \rightarrow (o_i \neq o_j)$.*

In the rest of the paper, we assume that the taxonomies obtained from the data sources are leaf-centric. We then relax this assumption when we talk about integrated data sets as the integration process may prevent enforcement of this constraint even when the individual data sources are leaf-centric.

[1]In this paper, we use 'term' and 'concept' interchangeably.

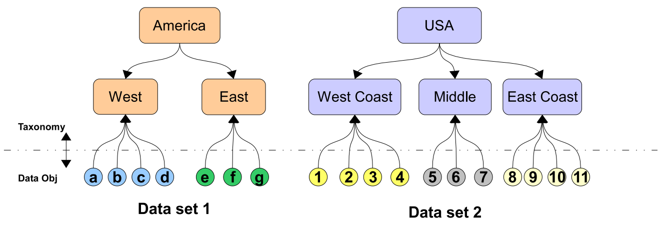


Figure 1: Data sets D_1 and D_2

3.2 Data Set Alignment and Matching

When integrating data from different sources, even when their schemas are aligned, the underlying taxonomies used for identical attributes may differ from each other:

EXAMPLE 3.1. *Figure 1 presents two data sets D_1 and D_2, each containing a different taxonomy for the location dimension of the data. First data set, $D_1 = (O_1, T_1, R_1)$, is as follows:*

$$\begin{aligned}
O_1 &= \{a, b, c, d, e, f, g\}, \\
T_1 &= (\{America, West, East\}, \\
&\quad \{(America, West, \overset{\geq}{\rightarrow}), (America, East, \overset{\geq}{\rightarrow})\}), \\
R_1 &= \{(a, West), (b, West), ..., (f, East), (g, East)\}.
\end{aligned}$$

and the second data set, $D_2 = (O_2, T_2, R_2)$, is

$$\begin{aligned}
O_2 &= \{1, 2, 3, ..., 11\}, \\
T_2 &= (\{USA, WestCoast, Middle, EastCoast\}, \\
&\quad \{(USA, WestCoast, \overset{\geq}{\rightarrow}), (USA, Middle, \overset{\geq}{\rightarrow}), \\
&\quad (USA, EastCoast, \overset{\geq}{\rightarrow})\}), \\
R_2 &= \{(1, WestCoast), (2, WestCoast), ..., (11, EastCoast)\}.
\end{aligned}$$

In the above example, both taxonomies (T_1 and T_2) are leaf-centric; i.e., they do not have any non-leaf concepts associated with any data objects. Although both T_1 and T_2 are about location dimension of the data, they are clearly different: (1) they use different terms for the same concept, like 'America' and 'USA'; (2) different schemes are exploited to divide the concept space; for example, T_1 divides 'America' into two independent partitions, whereas T_2 has three partitions under 'USA'.

To handle such mismatches, integration of data from different sources includes an *alignment* phase, which takes two data sets and their metadata as inputs and produces a matching between data and metadata elements. Work on alignment of schemas/taxonomies and on languages to describe matchings between data sources include [13, 20, 21]. In this paper, we do not focus on the alignment process itself, but assume that a (semi-)automated alignment technology has been used to identify a matching between given taxonomies:

DEFINITION 3.4 (MATCHING). *Given two taxonomies $T_i = (V_i, E_i)$ and $T_j = (V_j, E_j)$ from different data sets, a matching between T_i and T_j is defined as a set $M(T_i, T_j) = \{\mu_h = (x_h, y_h, \gamma_h) \mid (x_h \in V_i) \wedge (y_h \in V_j) \wedge (\gamma_h \in \{\overset{=}{\leftrightarrow}, \overset{\geq}{\rightarrow}, \overset{\triangle}{\leftrightarrow}\})\}$, where each μ_h specifies a matching relationship γ_h between $x_h \in V_i$ and $y_h \in V_j$.*

Common matching relationships reported in the literature include *equivalence* ($\overset{=}{\leftrightarrow}$), where x_h and y_h mean the same thing, *subsumption* ($\overset{\geq}{\rightarrow}$), where the concept x_h includes y_h,

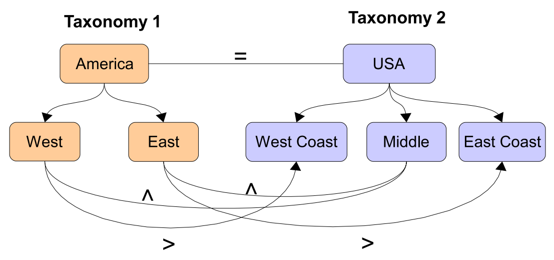


Figure 2: Matchings between T_1 and T_2

and *intersection* ($\overset{\triangle}{\leftrightarrow}$), where x_h and y_h are overlapping concepts. A term $x_h \in V_i$ is said to be disjoint from $y_h \in V_j$, if there is no matching relationship between them or no relationship can be inferred.

EXAMPLE 3.2. *Figure 2 shows a possible matching between the two taxonomies in Example 3.1. The two top-level concepts are identified to be identical to each other, while "West" in T_1 is identified to be subsuming "West Cost" and overlapping with "Middle" in T_2. Similarly, "East" in T_1 is identified to be subsuming "East Cost" and overlapping with "Middle".*

Note that matchings that are not explicitly discovered during the alignment process may also be inferred later. For instance, in the above example, $(America, WestCoast, \overset{\geq}{\rightarrow})$ can be inferred according to the transitiveness property of subsumption. Unlike subsumption and equivalence, however, the intersection relationship is not transitive.

3.3 Data Set Integration

Based on the above definition of matchings between taxonomies, we can create an integrated data set, which brings together the taxonomies and data objects from different sources. In particular, we define a *data union*, describing the combination of a given set,$\{D_1, D_2, ..., D_n\}$, of data sets, as follows:

DEFINITION 3.5 (DATA UNION). *Given a set of (object-disjoint) data sets $\{D_1, D_2, ..., D_n\}$ and a matching M, the data union $\mathcal{D} = \biguplus(D_1, D_2, ..., D_n)$ is a triple $\{\mathcal{O}, \mathcal{T}, \mathcal{R}\}$, where*

- *$\mathcal{O} = O_1 \cup O_2 \cup ... \cup O_n$;*
- *$\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \subseteq V_1 \cup ... \cup V_n$, and $\mathcal{E}$ is such that all relationships in $M \cup E_1 \cup ... \cup E_n$ are captured by the edges in $\mathcal{E}$, and*
- *$\mathcal{R}$ contains all object-to-concept mappings in $R_1 \cup R_2 \cup ... \cup R_n$.*

Naturally, the process of integrating data to obtain a *data union* is not a trivial one, especially due to possible conflicts in the data sets. Work on elimination of inconsistencies in the alignment process include [4, 19, 23, 16, 3, 24]. In this paper, we simply assume that the result of the data union process is *consistent*; i.e., it is subsumption-cycle free, each concepts has a single parent in the subsumption hierarchy, and all equivalent concepts are collapsed into one. Figure 3(a) depicts an example of what we take as an *inconsistent* integration of taxonomies: node '*Arizona*' has two parents, '*Mid-West*' and '*MountainStates*'; furthermore, node '*MountainStates*' is identified to be equivalent to node '*Mid-West*', but these nodes cannot be collapsed into a single one. The integration depicted in Figure 3(b) on the other hand is consistent, but *imperfect*: it contains intersection edges in addition to the subsumption hierarchy and, in this paper, we regard these edges not as inconsistencies, but as imperfections that need to be corrected. Note that given an *inconsistent* integration, as in Figure 3(a), it can be interpreted as an *imperfect* hierarchy, as in Figure 3(b), by treating offending subsumption and equivalence edges as *intersections*. The process of choosing which conflicting subsumption edges to be treated as *intersections* is outside of the scope of this paper, but is part of our ongoing work.

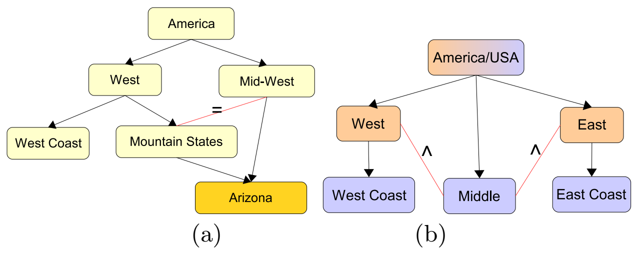


Figure 3: (a) Inconsistent integration and (b) consistent, but imperfect integration

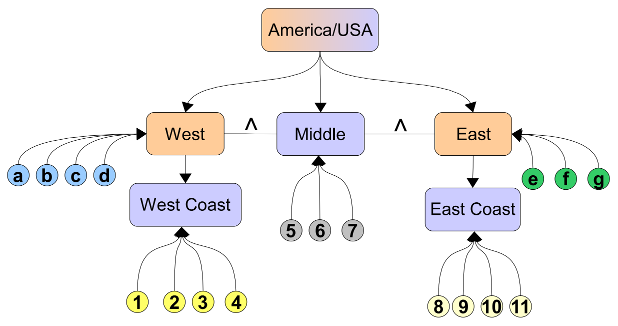


Figure 4: Data union of D_1 and D_2

EXAMPLE 3.3. *Figure 4 depicts a data union of the data sets, D_1 and D_2, in Example 3.1, based on the matchings shown in Figure 2.*

In this example, the two top-level concepts that were identified to be identical to each other have been combined into a single concept. Furthermore, the subsumption relationships in the matchings (such as "West" in T_1 subsuming "West Cost" in T_2) have been captured implicitly in the form of parent/child relationships between the corresponding concepts. Consequently, the integrated concept graph, $\mathcal{T}$, is almost a *general* taxonomy, except for intersection ($\overset{\triangle}{\leftrightarrow}$) relationships that have to be explicitly captured and included in the resulting concept graph. The integrated concept graph, $\mathcal{T}$, is not a perfect taxonomy because of the matchings with $\overset{\triangle}{\leftrightarrow}$. Such imperfections pose challenges for OLAP operations, such as drill-down and roll-up, as these usually assume that the underlying knowledge structures for the corresponding data dimensions are taxonomies. In the next subsection, we discuss this problem in further detail.

3.4 Problem: Navigational OLAP over Imperfectly Integrated Data Sources

In this section, we first revisit Examples 1.1 and 3.3 to shed light to problem of implementing OLAP operations over integrated data sources.

Example 3.4. *As discussed in Example 1.1, available taxonomies can help users analyze the data through navigational operations, such as drill-down and roll-up. For instance, the user can use drill-down operation to narrow down her scope from general concepts to the specific ones to get information more and more focused on certain aspects of the data set. Knowledge taxonomies support this, as their tree structures capture subsumption (i.e., ancestor/descendant relationships) and disjointness (i.e., sibling relationships) well.*

Unfortunately, as we have discussed in Example 3.3 (and as can be seen in Figure 4), the concept graphs resulting from data integration operations are not always valid taxonomies because of the intersection matchings. Therefore, OLAP operations, such as the drill-down operation, which rely on taxonomies, cannot be applied directly. For example, drilling-down from America/USA through the concept graph in Figure 4 will not result in two disjoint sub-concepts, since intersections exist among 'Middle', 'East Coast' and 'West Coast'.

As illustrated in the above example, when the knowledge structure resulting from the data union operation is not a valid taxonomy, the traditional OLAP operators cannot be applied. This problem can be addressed in two different ways:

- *Redefining OLAP operations:* One solution is to redefine (and re-implement) OLAP operations, such as *drill-down* and *roll-up*, in such a way that they operate on non-taxonomical knowledge structures (for example by locally eliminating intersections as they navigate over the concept graph).
- *Taxonomy cleaning:* The second solution is to preprocess the integrated concept graph resulting from the union to eliminate intersections and obtain a taxonomy (which approximately represents the original concept graph) over which the regular OLAP operations can be performed.

Naturally, any modifications in the concept graph may introduce a certain degree of imprecision; thus the goal of any solution should be to minimize the imprecision during the correction process. As we will see in Section 4 and evaluate in Section 5, the taxonomy cleaning solution causes less information loss and, thus, results in significantly lower degrees of imprecision than solutions which involve local corrections in the taxonomy. A second immediate advantage of the preprocessing-based approach is that it does not require re-implementation of existing tools and OLAP systems.

4. CORRECTING NON-TAXONOMICAL CONCEPT GRAPHS

As discussed in the previous section, a major challenge in navigational OLAP over imperfectly integrated data sets

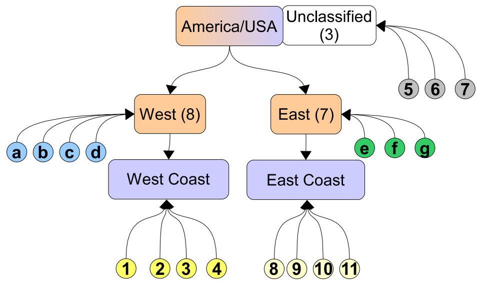


Figure 5: A possible correction for the concept graph in Figure 4

is the resulting non-taxonomical concept-graphs[2]. Both of the available approaches (i.e., redefining OLAP operations and preprocessing of concept graphs) to the problem involve transformation of the concept graphs resulting from the integration process into taxonomies. In this paper, we focus on corrections that eliminate intersections by un-classifying some of the concepts in the concept graph.

4.1 Correction of a Concept Graph

Converting concept graphs with intersection edges into navigable taxonomies involves finding a subset, $\bar{\mathcal{D}}$, of the integrated data, $\mathcal{D}$, such that the corresponding concept graph is a general taxonomy.

Definition 4.1 (Correction of a Concept Graph). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not necessarily a taxonomy, a correction of $\mathcal{D}$ is a data set $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ ($\sqsubseteq \mathcal{D}$), such that*

- $\bar{\mathcal{O}} = \mathcal{O}$ *(i.e., no objects are lost),*
- $\bar{\mathcal{T}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$, *where* $\bar{\mathcal{V}} \subseteq \mathcal{V}$ *and* $\bar{\mathcal{E}} \subseteq \mathcal{E}$, *is a general taxonomy, and*
- $\bar{\mathcal{R}} = \bar{\mathcal{R}}_T \cup \bar{\mathcal{R}}_U$, *such that*
  - $\bar{\mathcal{R}}_T \subseteq \mathcal{R}$; *i.e., some objects may not be classified under any node in the new taxonomy, and*
  - $\forall \langle o_i, t_j \rangle \in \mathcal{R} \backslash \bar{\mathcal{R}}_T$, $\exists \langle o_i, t_k \rangle \in \bar{\mathcal{R}}_U$, *such that* $t_k \in ancestor(t_j, \mathcal{T})$; *i.e., in the new taxonomy, those* unclassified *objects are attached to the closest possible ancestor node.*

Example 4.1. *Given the non-taxonomical data union in Figure 4, one possible correction is illustrated in Figure 5. Here the concept of 'Middle' is removed (i.e., taken to be un-classified). Therefore, 'East' and 'West' are the only two (disjoint) siblings under 'USA'. Thus, the remaining concept graph is a taxonomy.*

All the data corresponding to the removed (i.e., unclassified concept) are then assigned to the closest ancestor ('USA') of 'Middle' as un-classified data objects. Consequently, there are three data objects (i.e., 5, 6 and 7) associated to 'USA' as un-classified *objects. Intuitively, the amount of imprecision introduced by this correction is 3.*

[2]In the rest of the paper, we use the term *concept graph* to refer to the taxonomies with intersection edges introduced in the previous section to distinguish them from taxonomies without intersection edges.

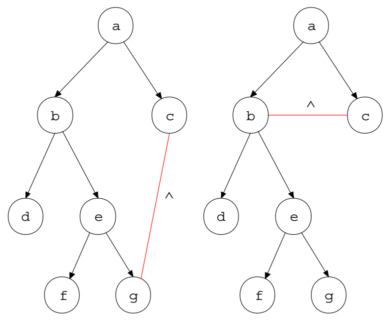


Figure 6: (a) A concept graph which is <u>not</u> sibling-normal and (b) a concept graph which is sibling-normal

Based on the above example, we can formally define the degree of imprecision resulting from a correction of a data union as follows:

Definition 4.2 (Imprecision due to a Correction). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not necessarily a taxonomy, and a correction $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ $(\sqsubseteq \mathcal{D})$, such that $\bar{\mathcal{R}} = \bar{\mathcal{R}}_T \cup \bar{\mathcal{R}}_U$, we define the amount of* imprecision due to the correction *as*

$$imprecision(\mathcal{D}, \bar{\mathcal{D}}) = |\bar{\mathcal{R}}_U|.$$

Thus, the goal of the *concept graph to taxonomy* conversion process is to find an *optimal correction* of the given data set:

Definition 4.3 (Optimal Correction). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not necessarily a taxonomy, a correction $\bar{\mathcal{D}}$ is said to be* optimal *iff there exists no other correction, $\bar{\mathcal{D}}'$, of $\mathcal{D}$, such that*

$$imprecision(\mathcal{D}, \bar{\mathcal{D}}') < imprecision(\mathcal{D}, \bar{\mathcal{D}}).$$

In the rest of the section, we discuss methods for searching for optimal corrections of a given data set with intersection relationships in its concept graph. In Section 4.2, we first focus on the correction of a sub-class of concept graphs that we refer to as *sibling-normal* concept graphs. In Section 4.3, we build on the framework developed in Section 4.2 to address more general, non-sibling-normal concept-graphs.

4.2 Correcting Sibling-Normal Concept Graphs

We define sibling-normal concept graphs as follows:

Definition 4.4 (Sibling-normal). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not necessarily a taxonomy, we refer to it as* sibling-normal *if, in $\mathcal{T}$, all the intersection edges are only between siblings.*

Figure 6 shows examples of sibling-normal and non-sibling-normal concept graphs. In the rest of this subsection, unless explicitly specified otherwise, when we refer to a concept graph, we mean a sibling-normal concept graph (e.g., Figure 6(b)). We first discuss correction of single-level sibling-normal concept graphs. In Subsection 4.2.3, we extend this to sibling-normal concept graphs of arbitrary depth.

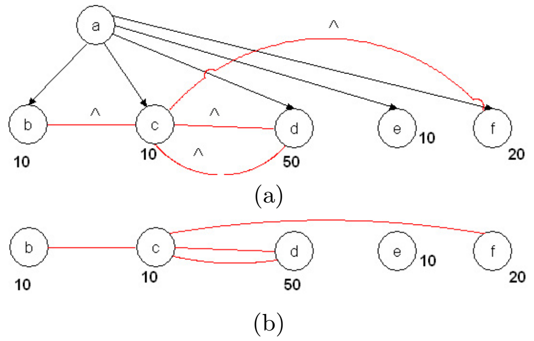


Figure 7: (a) A single-level concept graph and (b) its intersection graph

4.2.1 Correcting a Single-Level Sibling-Normal Concept Graph

Let us first consider a data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T} = (V, E)$ is a single-level concept graph:

Definition 4.5 (Single-level Concept Graph). *$T = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ are the vertices and E are the subsumption and intersection relationships between these vertices, is called a* single-level concept graph *iff E is such that*

- *$\forall_{v_i \in \{v_1, \ldots, v_n\}}\ (v_0, v_i, \overset{\geq}{\rightarrow}) \in E$ and*
- *the rest of the edges in E are of the form, $(v_i, v_j, \overset{\triangle}{\leftrightarrow})$, where $1 \leq i, j, \leq n$.*

In other words, T is a single level subsumption tree, where some of the leaves might be connected to each other with edges denoting intersection relationships (Figure 7(a))[3].

Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a single-level concept graph, we can compactly represent $\mathcal{D}$ using an intersection graph:

Definition 4.6 (Intersection Graph). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a single-level concept graph, the corresponding* intersection graph, *$I(\mathcal{D})$, is a node-weighted undirected graph (V_I, E_I, w), such that*

- *$V_I = \{v_1, \ldots, v_n\} \subset V$ (i.e., only the leaves of $\mathcal{T}$),*
- *$E_I = \{e \in E \mid e = (v_i, v_j, \overset{\triangle}{\leftrightarrow})\}$ (i.e., only the edges representing the intersection relationships between the concepts in the concept graph), and*
- *$\forall_{v_i \in V_I}\ \ w(v_i) = |\{r \in R \mid r = (o, v_i)\}|$ (i.e., w measures the number of data objects in $\mathcal{O}$ that are indexed under the concept that corresponds to the vertex v_i).*

Given this definition of intersection graph, it is easy to see that if $\mathcal{T}$, is a taxonomy, then the corresponding intersection graph, $I(\mathcal{D})$ will not contain any edges. Thus, we can state the following lemma:

Lemma 4.1 (Optimal Correction). *Given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a single-level concept graph, and an optimal correction, $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$, of $\mathcal{D}$, the intersection graph $I(\bar{\mathcal{D}}) = (\bar{V}_I, \bar{E}_I, \bar{w})$ is such that*

[3]Note that all single-level concept graphs are sibling-normal.

- $\bar{V}_I \subseteq V_I$,
- $\bar{E}_I = \emptyset$, *and*
- $\forall v_i \in \bar{V}_I,\ \ \bar{w}(v_i) = w(v_i)$.

Furthermore, there exists no other correction, $\bar{\mathcal{D}}' = (\bar{\mathcal{O}}', \bar{\mathcal{T}}', \bar{\mathcal{R}}')$, *with intersection graph* $I'(\bar{\mathcal{D}}) = (\bar{V}_I', \bar{E}_I', \bar{w}')$ *(also satisfying the above three conditions), such that*

$$\sum_{v \in \bar{V}_I} \bar{w}(v) < \sum_{v \in \bar{V}_I'} \bar{w}'(v).$$

The proof of this lemma follows from the above definition of intersection graphs, Definition 4.2 of imprecisions, and Definition 4.3 of optimal corrections.

Elimination of intersections from a concept graph is done through picking some of the concepts and removing them from the concept graph (or, more specifically, treating them as un-classified concepts). An important corollary of the above lemma is that the set of un-classified concepts is a vertex cover of the intersection graph:

COROLLARY 4.1 (VERTEX COVER). *Given a data union,* $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, *where* $\mathcal{T}$ *is a single-level concept graph, and an optimal correction,* $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$, *of* $\mathcal{D}$, *the set of vertices that are in* $I(\mathcal{D})$ *but not in* $I(\bar{\mathcal{D}})$ *(i.e., the un-classified vertices) is a vertex cover of* $I(\mathcal{D})$ *with the minimal total node weight.*

A vertex cover of an undirected graph is a subset of its vertices, such that this subset contains at least one of the two endpoints of each edge. Consequently, when the vertices in the vertex cover are eliminated, no edges are left in the graph. For example, in the intersection graph in Figure 7(b), removal of the node labeled "c" eliminates all edges in the graph. While there are other subsets of edges to consider as well, {"c"} is the minimal weight vertex cover when the node weights are also considered.

The problem of minimal size (or minimal weight) vertex cover of a graph is known to be an NP-complete problem [11], with an approximation within $2 - \Theta\left(\frac{1}{\sqrt{log|V|}}\right)$ of the optimal (where V is the number of vertices in the graph) [12]. In the experiments reported in Section 5, we formulate the vertex cover problem with a integer linear programming (ILP), which we solve optimally using the *Lingo* optimizer. In the next subsection, we describe how we formulate the ILP (Integer Linear Programming) version of the problem. One advantage of the ILP formulation (as opposed to algorithmic solutions to the vertex cover problem) is that it enables us to extend the formulation (as in Section 4.3, when we deal with non-sibling normal concept graphs) with constraints that are beyond the scope of the traditional vertex-cover problem.

4.2.2 ILP Formulation of the Optimal Correction Problem for a Single-Level Concept Graph

Let us be given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a single-level concept graph and let $I(\mathcal{D})$ be its intersection graph. As before, let $\mathcal{T} = (V, E)$ be such that $V = \{v_0, v_1, \ldots, v_n\}$ are the vertices and E are the subsumption and intersection relationships between these vertices. Furthermore, let $\bar{\mathcal{D}}$ and $I(\bar{\mathcal{D}})$ denote the corrected version of the data union and the corresponding intersection graph, respectively. As we mentioned above, we refer to those vertices that are in $I(\mathcal{D})$ but not in $I(\bar{\mathcal{D}})$ as *un-classified* vertices and the corresponding set of concepts is referred to as the *un-classified concept set (UCS)*.

Based on Definition 4.2, the number of data objects that were *covered by* the concepts in the set, UCS, gives us the degree of imprecision caused by this correction. Naturally, we are looking for corrections that have minimal imprecision, i.e., with minimal number of data objects covered by concepts in UCS. Thus, among the different schemes that can eliminate intersections in $I(\mathcal{D})$, the optimal solution must minimize the imprecision corresponding to this taxonomy rooted at node, v_0.

$$\begin{aligned} MIN\ \ imprecision(v_0, \bar{\mathcal{T}}) &= \sum_{v_i \in UCS} w(v_i) \\ &= \sum_{v_i \in \{v_1, \ldots, v_n\}} w(v_i) \cdot U_i \quad (1) \end{aligned}$$

Here $U_i \in \{0, 1\}$ is a 0-1 integer variable, which determines whether vertex v_i is un-classified (for $U_i = 1$) or not (for $U_i = 0$).

An intersection edge in $\mathcal{T}$ is removed if any of the endpoints is placed in UCS. Since we would like all intersection edges in $\mathcal{T}$ being eliminated, we also have the constraint that

$$\forall_{e_k \in R\ s.t. e_k = (v_i, v_j, \overset{\triangle}{\leftrightarrow})}\ (v_i \in UCS) \vee (v_j \in UCS).$$

We can represent this constraint in ILP form as follows:

$$\forall_{e_k \in R\ s.t. e_k = (v_i, v_j, \overset{\triangle}{\leftrightarrow})}\ U_i + U_j \geq 1. \quad (2)$$

Thus, the problem of identifying the correction with minimum imprecision for a single level data union is equal to finding a solution that minimizes the *imprecision* objective function in Equation 1 given the integer linear program consisting of the constraints specified in Equation 2.

4.2.3 Correcting Multi-Level Sibling-Normal Concept Graphs

As described in the previous two subsections, correction of a single-level concept graph can be achieved by formulating the problem as a weighted minimum vertex cover problem, where weights are the number of objects that are classified under the leaves of the taxonomy. The underlying assumption in the above formulation is that all concept nodes, except for the root, are children of the root. Naturally, in general, concept graphs are deeper and the subsumption relationships between the concept nodes have a tree structure. Therefore, this single-level approach cannot be directly applied to general concept graphs and taxonomies. In this section, we extend the above framework to multi-level sibling-normal concept graphs.

Let us remember that, by Definition 4.2, given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not necessarily a taxonomy, and a correction $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ ($\sqsubseteq \mathcal{D}$), such that $\bar{\mathcal{R}} = \bar{\mathcal{R}}_T \cup \bar{\mathcal{R}}_U$, the amount of *imprecision due to this correction* (i.e., *imprecision*$(\mathcal{D}, \bar{\mathcal{D}})$) is defined as $|\bar{\mathcal{R}}_U|$. Furthermore, by Definition 4.1, $\bar{\mathcal{R}}_U$ maps those objects that were initially covered by the newly un-classified concepts to the closest possible ancestors of each un-classified concept. Let v_0 be the root of $\bar{\mathcal{T}}$, then we can rewrite $\bar{\mathcal{R}}_U$, as

$$\bar{\mathcal{R}}_U = \bar{\mathcal{R}}_U(v_0) \cup \bar{\mathcal{R}}_U(desc(v_0, \bar{\mathcal{T}})),$$

where $\bar{\mathcal{R}}_U(v_0)$ are those objects that are mapped to v_0 due to the un-classification process and $\bar{\mathcal{R}}_U(desc(v_0, \bar{\mathcal{T}}))$ are objects mapped to the descendants of v_0 due to un-classification. Since each object is mapped to one and only one vertex in $\bar{\mathcal{T}}$, we can state the degree of imprecision of the taxonomy $\bar{\mathcal{T}}$, rooted at v_0 as

$$\begin{aligned} imprecision(v_0, \bar{\mathcal{T}}) &= |\bar{\mathcal{R}}_U(v_0)| + |\bar{\mathcal{R}}_U(desc(v_0, \bar{\mathcal{T}}))| \qquad (3) \\ &= |\bar{\mathcal{R}}_U(v_0)| + \sum_{v_i \in desc(v_0, \bar{\mathcal{T}})} |\bar{\mathcal{R}}_U(v_i)| \\ &= |\bar{\mathcal{R}}_U(v_0)| + \\ & \sum_{v_i \in child(v_0, \bar{\mathcal{T}})} imprecision(v_i, \bar{\mathcal{T}}(v_i)). \end{aligned}$$

In other words, the amount of imprecision of the taxonomy $\bar{\mathcal{T}}$ rooted at v_0 can be written in terms of the un-classified objects mapped to v_0 and the imprecisions of the sub-taxonomies rooted at the children of v_0 [4]. Given this observation, we can state the following theorem:

Theorem 4.1 (Child Optimality). *Given a data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is rooted at v_0, if $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ (where $\bar{\mathcal{T}}$ is also rooted at v_0) is an optimal correction, then for each $v_i \in child(v_0, \bar{\mathcal{T}})$, the following holds:*

- *Let $\mathcal{D}(v_i) \sqsubset D$ be the subset of $\mathcal{D}$ rooted at v_i. Let also $\bar{\mathcal{D}}(v_i) \sqsubset \bar{\mathcal{D}}$ be the subset of $\bar{\mathcal{D}}$ rooted at v_i. Then, $\bar{\mathcal{D}}(v_i)$ is an optimal correction for $\mathcal{D}(v_i)$.*

The proof of this theorem follows from Equation 3 by contradiction. Intuitively, if Theorem 4.1 does not hold, then one can take the optimal correction, $\bar{\mathcal{D}}'(v_i)$ of $\mathcal{D}(v_i)$, and replace it in $\bar{\mathcal{D}}$ to obtain a smaller degree of imprecision, which would give rise to a contradiction.

The set $\bar{\mathcal{R}}_U(v_0)$ contain those objects that are covered in $\mathcal{T}$ by sub-taxonomies rooted at the children of v_0 that become later un-classified:

$$\begin{aligned} |\bar{\mathcal{R}}_U(v_0)| &= \sum_{v_i \in child(v_0, \mathcal{T}) \setminus child(v_0, \bar{\mathcal{T}})} \sum_{v_j \in \mathcal{T}(v_i)} w(v_j) \\ &= \sum_{v_i \in UCS(v_0)} \sum_{v_j \in \mathcal{T}(v_i)} w(v_j). \qquad (4) \end{aligned}$$

Here, once again, $\forall_v \; w(v) = |\{r \in R \mid r = (o, v)\}|$ (i.e., w measures the number of data objects in $\mathcal{O}$ that are indexed under the concept that corresponds to the vertex v). As a corollary of Equation 4 and Theorem 4.1, we can state the following:

Corollary 4.2 (Single-Level Treatment). *Given a data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is rooted at v_0, if $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ (where $\bar{\mathcal{T}}$ is also rooted at v_0) is an optimal correction, then for each $v_i \in child(v_0, \mathcal{T})$, the following holds:*

- *Let $\mathcal{D}(v_i) \sqsubset D$ be the subset of $\mathcal{D}$ rooted at v_i. Let $\bar{\mathcal{D}}(v_i)$ be an optimal correction for $\mathcal{D}(v_i)$.*

  Then, $v_i \in child(v_0, \bar{\mathcal{T}})$ iff v_i is also in the optimal correction of the single level concept graph, $\mathcal{D}'(v_0)$, composed of v_0 and its children in $\mathcal{T}$, with the intersection graph, $I(\mathcal{D}'(v_0)) = (V'_I, E'_I, w')$, such that

  - $V'_I = child(v_0, \bar{\mathcal{T}})$,
  - $E'_I = \{e \in \mathcal{E} \mid e = (v_i, v_j, \overset{\triangle}{\leftrightarrow}) \; \wedge \; v_i, v_j \in child(v_0, \bar{\mathcal{T}})\}$ *(i.e., only the edges representing the intersection relationships between the child concepts of v_0), and*
  - $\forall_{v_i \in child(v_0)}$
  $$w'(v_i) = \left(\sum_{v_j \in \mathcal{T}(v_i)} w(v_j) \right) - imprecision(v_i, \bar{\mathcal{T}}(v_i)).$$

Intuitively, this corollary states that, given optimal corrections of all children of the root v_0, the optimal correction for the overall data union can be found by treating the root level of the data union $\mathcal{D}$ as a single-level data union, which can independently be corrected. Since the optimal degree of imprecision for the sub-taxonomy rooted at v_i is fixed at $imprecision(v_i, \bar{\mathcal{T}}(v_i))$ whether this sub-taxonomy is declared un-classified or not, the node weights for the corresponding root level intersection graph discounts those objects that have already been declared un-classified for the optimally corrected sub-taxonomies.

4.2.4 ILP Formulation of the Optimal Correction Problem for Multi-Level Sibling-Normal Concept Graphs

As in Section 4.2.2, let us be given a data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is a single-level concept graph and let $I(\mathcal{D})$ be its intersection graph. As before, let $\mathcal{T} = (V, E)$ be such that V are the concept vertices and E are the subsumption and intersection relationships between these vertices. Let again v_0 be the root of $\mathcal{T}$. Unlike Section 4.2.2, on the other hand, for each $v_i \in child(v_0)$, let us also be given $\bar{\mathcal{D}}(v_i)$, i.e., the optimal correction for $\mathcal{D}(v_i) \sqsubset D$.

We revise the ILP objective function and constraints we have introduced to deal with simple single-level data unions as follows: the objective function in Equation 1 is modified with the weights introduced in Corollary 4.2:

$$MIN \;\; SL\_Imprecision(v_0, \bar{\mathcal{T}}) = \sum_{v_i \in child(v_0)} w'(v_i) \cdot U_i,$$

where $U_i \in \{0, 1\}$ and $w'(v_i)$ is defined in Corollary 4.2 in terms of (a) the number objects in the sub-taxonomy rooted at v_i (i.e., $w(v_i)$) and (b) the number of objects that are un-classified by an optimal correction of this sub-taxonomy. $SL\_Imprecision(v_0)$ denotes the single level of imprecision due to unclassification of the children of v_0. Therefore, the total of imprecision due to unclassifications under v_0 can be measured by

$$\begin{aligned} imprecision(v_0, \bar{\mathcal{T}}) &= SL\_Imprecision(v_0, \bar{\mathcal{T}}) \\ &+ \sum_{v_i \in child(v_0)} imprecision(v_i, \bar{\mathcal{T}}(v_i)). \end{aligned}$$

We then rewrite the constraint specified in Equation 2 in a way that focuses only on the children of v_0:

$$\forall_{e_k \in R \; s.t. e_k = (v_i, v_j, \overset{\triangle}{\leftrightarrow}) \; \wedge \; v_i, v_j \in child(v_0)} \;\; U_i + U_j \geq 1.$$

4.2.5 Summary: Correcting Sibling-Normal Graphs

In the previous subsection, we have seen that given an imperfectly integrated sibling-normal data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T}$ is not a taxonomy, we can correct $\mathcal{D}$

[4]We use $\bar{\mathcal{T}}(v_i)$ to denote the sub-taxonomy rooted at v_i.

Algorithm 1: Optimal SNTaxonomyCorrection($\mathcal{D}$)

```
Input: an imperfectly integrated data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$
Output: a corrected data set, $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$;
        the degree of imprecision, $imprec$
1  Let $v_0$ be the root of $\mathcal{T}$;
2  if $children(v_0) \neq \emptyset$ then
3      for $v_i \in children(v_0)$ do
4          let $\mathcal{D}(v_i) = (\mathcal{O}(v_i), \mathcal{T}(v_i), \mathcal{R}(v_i))$ be the subset of $\mathcal{D}$ rooted
           at $v_i$;
5          $(\bar{\mathcal{D}}(v_i), imprec(v_i)) \leftarrow$ SNTaxonomyCorrection($\mathcal{D}(v_i)$);
6          $w'(v_i) \leftarrow$countCoveredObjects($v_i, \mathcal{D}$) $- imprec(v_i)$;
7      end
8      $(\bar{\mathcal{D}}, imprec) \leftarrow$ construct&SolveILP($children(v_0), w'$);
9  end
10 else
11     return $(\mathcal{D}, 0)$.
12 end
```

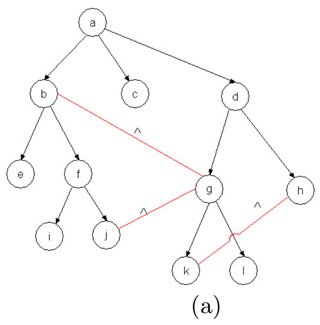
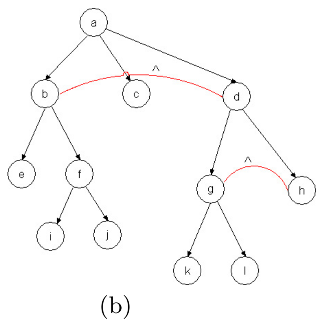


(a) (b)

Figure 8: (a) A concept graph which is not sibling-normal and (b) sibling-normalized version

using an ILP formulation as long as we are given optimal corrections for the children of the root of $\mathcal{T}$. In other words, an optimal solution for the taxonomy $\mathcal{T}$ first requires solutions for its sub-taxonomies. This, however, can be achieved by an algorithm which first eliminates the intersections observed at the leaf nodes and then gradually moves up higher in the hierarchy by eliminating intersections among siblings at each level (see Algorithm 1). Note that, while the optimal taxonomy correction algorithm (Algorithm 1) is written in a recursive manner, starting from the root of the given concept graph, it actually resolves the intersections in a bottom-up manner, starting from the leaves, as discussed in Section 4.2.3.

4.3 Correcting Non-Sibling-Normal Graphs

The data union correction approach presented in the previous section assumes that the input taxonomy is sibling-normalized; i.e, given the data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, in $\mathcal{T}$, all the intersection edges are only between siblings. This however is a very strict constraint, which does not necessarily hold in real data integration scenarios.

One naive way to leverage the discussions in the previous subsection is to take a non-sibling normal data union (e.g., Figure 8(a)) and convert it to a sibling-normal one (e.g., Figure 8(b)). This can be achieved by recalling from Section 3.2 that an explicitly provided intersection edge between a given pair of concept nodes implies intersection edges between the ancestors of this pair of nodes in the taxonomy. Intuitively, sibling-normalization process includes identification of all the implied intersection edges (such as the edge between b and d in Figure 8(b)) and, then, elimination of all non-sibling edges that are dominated by a sibling-edge (such as elimination of the edges between b and g as well as g and j in the figure). While it is possible to apply the SNTaxonomyCorrection Algorithm (Algorithm 1) introduced in the previous section on such sibling-normalized taxonomies, the result will not necessarily be optimal. For example, while an optimal correction in the taxonomy shown in Figure 8(a) may involve un-classification of node g (which would remove all the intersection edges), the same removal in g cannot remove the edge between nodes b and d. In this example one of the sub-taxonomies b and d must be declared un-classified, thus introducing a large degree of overall imprecision.

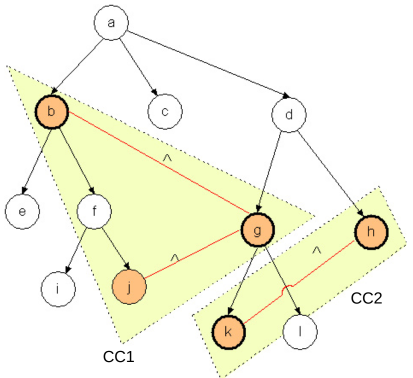


Figure 9: Two intersection-connected components of the concept graph in Figure 8(a)

Therefore, an optimal correction algorithm for non-sibling-normal data unions cannot rely on sibling-normalization and SNTaxonomyCorrection algorithm, directly. However, in the rest of this section, we show, through a restructuring of the problem, that we can in fact leverage a similar bottom-up scheme for optimal correction of non-sibling-normal taxonomies as well. First, let us define *intersection-connected components* of a concept graph that act as building blocks for the bottom up processing:

DEFINITION 4.7 (INTERSECTION-CONNECTED COMP.). *Let the data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, be a concept graph. An* intersection-connected component *of $\mathcal{D}$, is an undirected graph $CC = (V, E)$, where*

- *$V \subseteq \mathcal{V}$,*
- *$E \subset \mathcal{E}$ is such that,*
  - *all $e_h \in E$ are of the form, $(v_i, v_j, \overset{\wedge}{\leftrightarrow})$, where $v_i, v_j \in V$,*
  - *all $v_i, v_j \in V$ are reachable from each other in CC, and*
  - *there exists no $v_k \in \mathcal{V} \backslash V$, such that there is an edge $(v_i, v_k, \overset{\wedge}{\leftrightarrow}) \in \mathcal{E}$ for some $v_i \in V$.*

Figure 9 shows the two intersection-connected components of the concept graph in Figure 8(a). Unlike concept nodes which are simply subsuming and intersecting with other nodes, intersection-connected components may have more complex relationships with each other. We next introduce a particular relationship, critical in correcting non-sibling-normal concept graphs, between intersection-connected components:

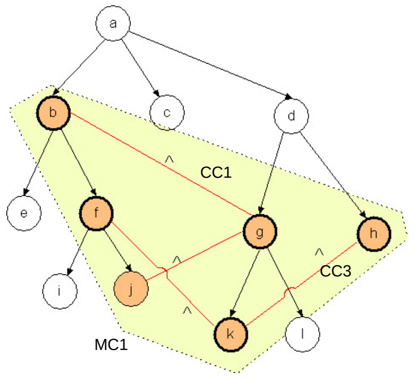


Figure 10: CC_1 and CC_3 are doubly-crossing

DEFINITION 4.8 (DOUBLE-CROSSINGS). *Let the data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, be a concept graph and let $CC_1 = (V_1, E_1)$ and $CC_2 = (V_2, E_2)$ be two intersection-connected components of $\mathcal{D}$. Then, CC_1* is double-crossing *with CC_2, denoted as $\overleftrightarrow{CC_1, CC_2}$ (or equivalently CC_2* is double-crossing *with CC_1, denoted as $\overleftrightarrow{CC_2, CC_1}$) iff*

- *$\exists v_i, v_j \in V_1$ and $\exists v_u, v_t \in V_2$,*

such that

- *$v_u \in desc(v_i) \in \mathcal{T}$ and $v_t \in desc(v_j) \cup anc(v_j) \in \mathcal{T}$.*

Intuitively, if CC_1 *double crosses with* CC_2, then there is a gap between two branches in $\mathcal{T}$ that is doubly crossed, once by an edge in CC_1 and once by an edge in CC_2. For example, in Figure 9, there is no double crossings between the intersection-connected components CC_1 and CC_2. On the other hand, in the slightly modified concept graph in Figure 10, the new intersection edge between nodes f and k causes double crossing: in this case, the intersection edge between b and g in CC_1 and the edge between f and k in CC_3 are crossing the same gap (between the subtrees rooted at b and d) in the concept graph.

DEFINITION 4.9 (MIXED COMPONENT PARTITIONING). *Let the data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, be a concept graph and let $\mathcal{CC}$ be the set of all intersection connected components of $\mathcal{D}$. $\mathcal{MC}$ is referred to as a* mixed component *partitioning of $\mathcal{D}$ iff*

- *if $CC_i \in \mathcal{CC}$ and $\forall CC_j \in \mathcal{CC}$ $\overleftrightarrow{CC_i, CC_j} = false$, then $CC_i \in \mathcal{MC}$.*
- *if $\mathcal{CC}_k$ is a maximal subset of $\mathcal{CC}$ such that $\forall CC_i, CC_j \in \mathcal{CC}_k$ $\overleftrightarrow{CC_i, CC_j} = true$, then there exists $MC_k \in \mathcal{MC}$, where*

$$MC_k = \left(\bigcup_{CC_i \in \mathcal{CC}} V_i, \bigcup_{CC_i \in \mathcal{CC}} E_i \right),$$

- *nothing else is in $\mathcal{MC}$.*

For example, in Figure 9, since there are no doubly-crossing intersection-connected components, $\mathcal{MC} = \{CC_1, CC_2\}$. On the other hand, since in Figure 9, $\overleftrightarrow{CC_1, CC_3}$ is true, $\mathcal{MC} = \{MC_1\}$, where MC_1 is the combination of CC_1 and CC_3.

Note that mixed intersection components can be either *independent* from each other (i.e., none of their vertices have subsumption relationship with each other) or one mixed intersection component may dominate the other (i.e., one vertex in one mixed intersection component subsumes some of the vertices of the other mixed component, but not vice versa - e.g. Figure 9). Any other relationships between two components would in fact cause double-crossing and necessitate the two mixed components be merged. Thus, given a mixed component partitioning, $\mathcal{MC}$ of $\mathcal{D}$, we can define a mixed component tree, $\mathcal{MT}$ of $\mathcal{D}$ as follows:

DEFINITION 4.10 (MIXED COMPONENT TREE). *Let the data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$, where $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, be a concept graph and let $\mathcal{MC}$ be a* mixed component *partitioning of $\mathcal{D}$. The* mixed component tree*, $\mathcal{MT} = (VM, EM)$, of $\mathcal{D}$ is a tree where*

- *$mv_\top = (\{v_0\}, \emptyset) \in VM$ is the root of the tree,*
- *$MC_i \in \mathcal{MC}$ are the other vertices in VM,*
- *$\forall MC_i, MC_j \in \mathcal{MC}$ such that $\exists v_k \in V_i$ $v_l \in V_j$ where $v_k \in anc(v_l)$, there is a corresponding edge $(MC_i, MC_j) \in EM$, on condition that*
  - *$\nexists MC_k \in \mathcal{MC}$ st. $(MC_i, MC_k) \in EM$ and $(MC_k, MC_j) \in EM$,*
- *there exists no other vertex or edge in $\mathcal{MT}$.*

Figures 11(a) depicts the mixed-component tree for the concept graph in Figure 9. Given a data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$ and its mixed component tree, $\mathcal{MT} = (VM, EM)$, we can show that the following *child optimality* theorem (analogous to Theorem 4.1) holds:

THEOREM 4.2 (CHILD OPTIMALITY). *Given a data union $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$ (where $\mathcal{T}$ is also rooted at v_0) and its mixed component tree, $\mathcal{MT} = (VM, EM)$, if $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$ (where $\bar{\mathcal{T}}$ is also rooted at v_0) is an optimal correction, then for each $mv_i \in child(mv_\top, \mathcal{MT})$, the following holds:*

- *Let $\mathcal{MT}(mv_i) \sqsubset \mathcal{MT}$ be the subset of $\mathcal{T}$ rooted at mv_i. Then $\bar{\mathcal{D}}$ also contains an optimal correction for the subset of $\mathcal{D}$ consisting of the nodes in $\mathcal{MT}(mv_i)$ and all their descendants in $\mathcal{D}$.*

The proof of this theorem follows from Equation 3 and the definition of the mixed component tree. Relying on this theorem, next we develop a bottom-up correction scheme for non-sibling-normal concept graphs.

4.3.1 Correcting a Single Mixed Component

Let $MC = (V, E)$ be a mixed component in $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$. In order to correct MC, we revise the ILP objective function and constraints we have introduced to deal with sibling-normal data sets as follows. First, let us specify the constraints for enforcing intersection edge removal:

$$\forall_{e_k \in R \ s.t. e_k = (v_i, v_j, \overset{\wedge}{\leftrightarrow}) \wedge v_i, v_j \in V} \ U_i + U_j \geq 1.$$

Furthermore, we know that if a node is removed all other nodes subsumed by it are also removed; i.e.,

$$\forall_{v_i, v_j \in V \ s.t. v_i \ subsumes \ v_j \ in \ \mathcal{T}} \ U_i \leq U_j.$$

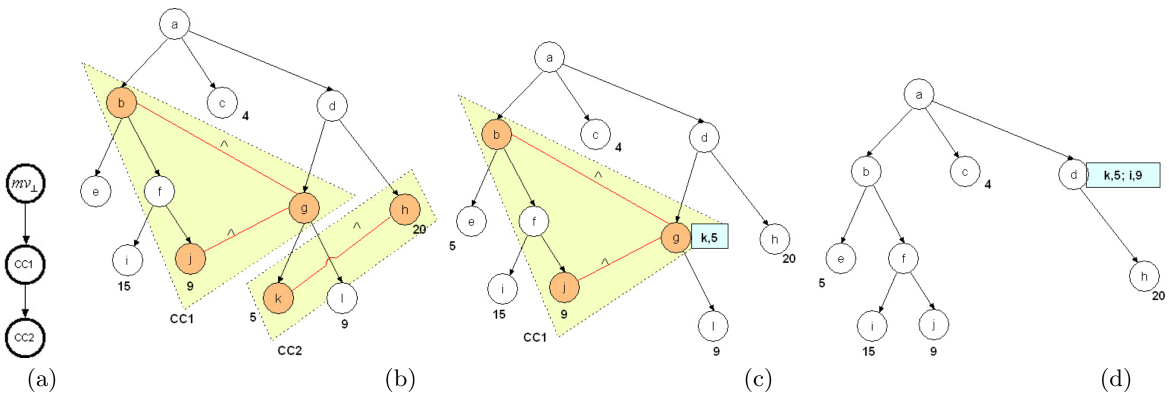


Figure 11: (a)The mixed component tree for the concept graphs in Figures 9, (b) initial state before correction, (c) correction after the first, lowest-level mixed component, and (d) correction after the last mixed component

```
Algorithm 2: Optimal TaxonomyCorrection($\mathcal{D}, \mathcal{MT}$)
Input: an imperfectly integrated data union, $\mathcal{D} = (\mathcal{O}, \mathcal{T}, \mathcal{R})$;
       a mixed component tree, $\mathcal{MT}$, of $\mathcal{T}$,
Output: a corrected data set, $\bar{\mathcal{D}} = (\bar{\mathcal{O}}, \bar{\mathcal{T}}, \bar{\mathcal{R}})$;
       the degree of imprecision, imprec
1  Let $mv_\top$ be the root of $\mathcal{MT}$;
2  if $children(mv_t, \mathcal{MT}) \neq \emptyset$ then
3      for $mv_i \in children(mv_t, \mathcal{T})$ do
4          let $\mathcal{D}(mv_i) = (\mathcal{O}(mv_i), \mathcal{T}(mv_i), \mathcal{R}(mv_i))$ be the subset of
           $\mathcal{D}$ covered by $mv_i$;
5          let $\mathcal{MT}(mv_i)$ be subtree of $\mathcal{MT}$ rooted at $mv_i$;
6          $(\bar{\mathcal{D}}(mv_i), imprec(mv_i)) \leftarrow$
               TaxonomyCorrection($\mathcal{D}(mv_i), \mathcal{MT}(mv_i)$);
7          moveUnclassifieds_basedon_Correction($\mathcal{D}, \bar{\mathcal{D}}(mv_i)$);
8      end
9      $(\bar{\mathcal{D}}, imprec) \leftarrow$ construct&SolveILP($mv_\top$);
10 end
11 else
12     return $(\mathcal{D}, 0)$.
13 end
```

Finally, the objective function for $MC(V, E)$ is

$$MIN \;\; SL\_Imprecision(MC) = \sum_{v_i \in V} w''(v_i) \cdot U_i,$$

where $U_i \in \{0, 1\}$ and $w''(v_i)$ is the number of *classified* objects in the sub-taxonomy of $\mathcal{T}$ rooted at v_i *but not contained under any other* $v_j \in V$ [5]. $SL\_Imprecision(MC)$ measures the imprecision due to unclassification of vertices in V. Therefore, the total of imprecision caused by unclassifications under vertices in V can be written as

$$\begin{aligned} imprecision(MC) &= SL\_Imprecision(MC) \\ &+ \sum_{v_i \in V} imprecision(v_i, \bar{\mathcal{T}}(v_i)), \end{aligned}$$

All the objects covered by a vertex, v_i, marked un-classified by the above process will be mapped to the closest ancestor of v_i which is not in MC.

4.3.2 *Correcting a Concept Graph with Multiple Mixed Components*

Finally, the pseudo-code for correcting non-sibling-normal data sets is depicted in Algorithm 2. Figure 11 explains this bottom-up correction process using a data union with multiple components. The concept graph of this data union is the same as the one in Figure 9 and has the mixed component tree depicted in Figure 11(a). Figure 11 also includes the numbers of data objects mapped onto the leaves of the graph.

The correction process starts with the most dominated mixed component, CC_2 (Figures 11(b,c)). Here, since node k has a smaller number of data objects, it is selected for unclassification and its data objects are mapped to the closest ancestor, g, outside of CC_2 as un-classified objects.

In the second, and last step, the mixed component, CC_1 is corrected (Figures 11(c,d)). Here, since g has 9 classified objects covered by it and since its un-classification would eliminate all intersection edges in CC_1 with the lowest unclassification cost (i.e., imprecision), this node (and all its descendants) are un-classified. These un-classified data objects are mapped to the closest ancestor of g, which is d.

5. EXPERIMENTS

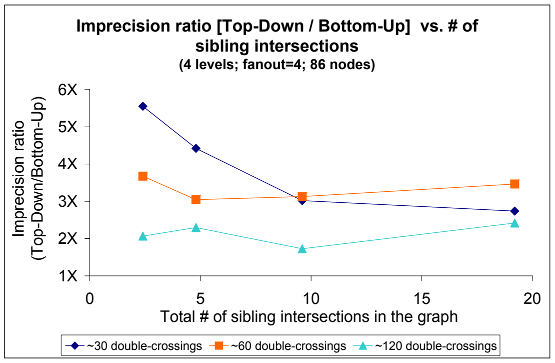
In this section, we experimentally evaluate the proposed bottom-up taxonomy cleaning algorithm using synthetic and real-data sets. In particular, we compare the imprecision resulting from the proposed bottom-up correction scheme with a top-down mechanism which implements navigational OLAP operations, such as *drill-down* with local corrections as it visits various nodes in the hierarchy. Since in the top-down version, correction decisions need to be provided with local information, the trees are sibling-normalized by propagating implied intersection relationships (see Figure 8).

The code is implemented in Java and LINGO 9.0 optimizer is used to solve the integer program formulation of the correction problem. The run-time environment is a 2.33GHz Intel(R) Core 2 Duo CPU with 2GB physical memory.

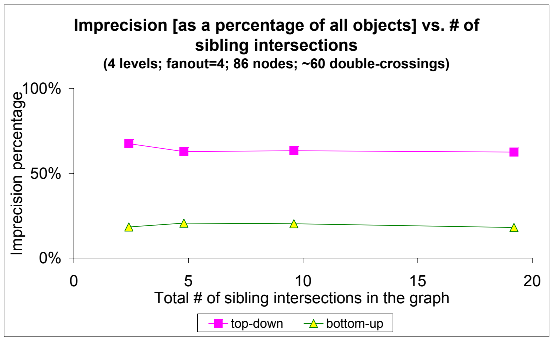
5.1 Evaluation on Synthetic Data

In the experiments with synthetic data reported in this section, we constructed taxonomies (i.e., balanced tree structures) of varying size and properties: (a) we varied the number of concept nodes between $\sim$ 100 to $\sim$ 1000, (b) the fanout of the nodes in the taxonomies was set to 4, and (c) the height of the taxonomies varied between 4 and 6. Data objects were randomly mapped to the leaves of the taxonomies. We also varied the sibling intersection and

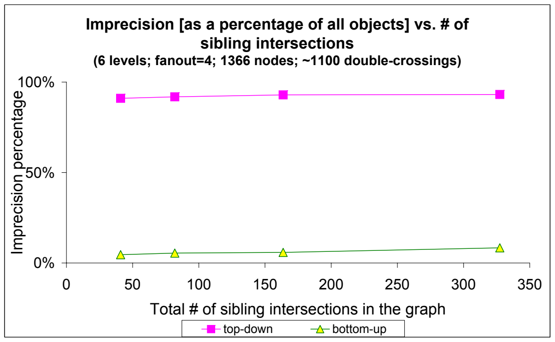
[5]Note that nodes with ancestor-descendant relationship can exist in the same MC, thus data objects under descendants should not be double counted.



(a)



(b)



(c)

Figure 12: Degrees of imprecision due to top-down and bottom-up schemes over synthetic data

double-crossing rates as described below.

Figure 12 compares the degrees of imprecisions caused by the proposed bottom-up taxonomy correction scheme with degrees of imprecisions caused by the top-down approach with local corrections. In these experiments, intersections rates are set relatively high: the number of sibling intersections is such that there is upto ~ 1 intersection per sibling group and the double crossings is such that there is $\sim 1 - 1.5$ double crossings per node. Note that this puts the bottom-up approach to a relative disadvantage as with unrealistically high intersection rates, the portion of objects that would need to be un-classified anyhow also increases.

Figures 12(a) and (b) show the effects of varying rates of sibling intersections and double-crossings in relatively small concept graphs (86 nodes). As can be seen in in Figure 12(a), under this configuration, imprecision by bottom-up approach is $\sim \frac{1}{1.5}$ to $\sim \frac{1}{6}$ of the imprecision resulting when the top-down scheme is used. In fact, as shown in

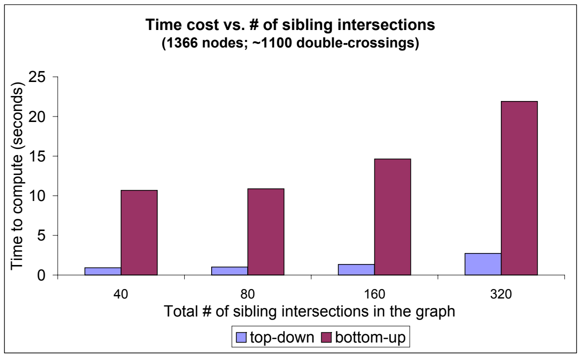


Figure 13: Processing times for taxonomy correction on synthetic data

Figure 12(b), while the top-down approach marks upto 60% of objects unclassified, the rate is less than 20% for the proposed scheme. For a larger data set (Figure 12(c), with $\sim$ 1300 concept nodes, $\sim$ 1 double-crossing per node, and $\sim$ 1 intersection per sibling group) the difference in performance is even more pronounced, with top-down scheme declaring almost 100% of the data un-classified, while the bottom up approach un-classifying less than 10% of the data.

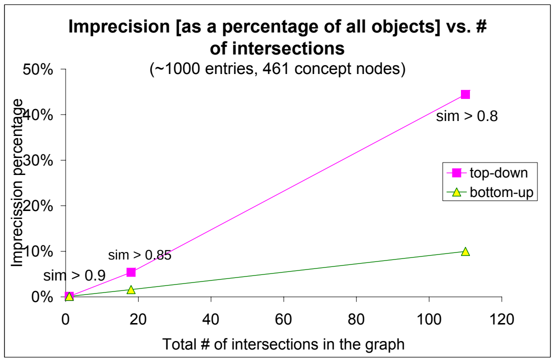
As shown in Figure 13, the execution time of the bottom-up scheme for this data set is higher than that of the top-down scheme; however, considering that the process is done off-line and considering the very significant gains in precision, $\sim$ 20 seconds for a concept graph of $\sim$ 1300 nodes and large amounts of imperfections is acceptable.

5.2 Evaluation on Real Data

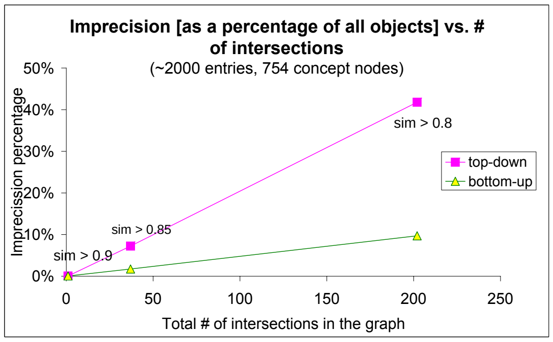
In order to evaluate the effectiveness of the scheme presented in this paper, we used the *ODP* (Open Directory Project) [1] dataset, which is a semantic hierarchy on various topics. For the experiments presented in this subsection, we choose entries under the topic of "Arts" as input. We extracted three different data sets, with $\sim$ 1000, $\sim$ 2000, and $\sim$ 6000 nodes, respectively. For each data set, we use the first three levels of the hierarchy (461, 754 and 1592 nodes respectively) as the concept taxonomy and the rest of the nodes as the data objects classified under the taxonomy. Intersections between the entries in the taxonomy are computed by treating the descriptive text associated with the entries as keyword vectors and computing cosine similarities between the entries. Those entry node pairs with similarities larger than a given threshold (set to 0.8, 0.85 and 0.9 in the experiments) are treated as conceptually intersecting.

Figure 14 depicts the imprecision results. As can be seen here, the degree of imprecision introduced by the bottom-up scheme is significantly less than that introduced by the top-down scheme. For instance, for all three data sets, when 0.8 is taken as the similarity threshold, the degree of imprecision introduced by the bottom-up scheme is less than 10%, while the imprecision introduced by the top-down scheme is close to $\sim$ 50%.

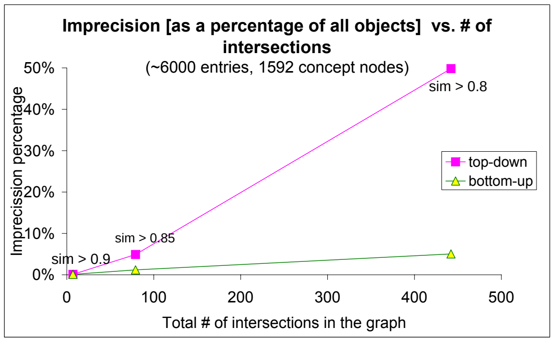
Figure 15 shows the time cost of the corrections on this data set. As can be seen here the execution cost of the top-down and bottom-up schemes are close to each other on these data sets. In fact, especially when the number of intersections is large, the proposed bottom-up scheme works faster than the top-down approach.



(a)



(b)



(c)

Figure 14: Degrees of imprecision due to taxonomy correction on ODP data

As can be seen in Figure 16, this behavior is explained quite well by the number of integer linear program (ILP) calls the two schemes make during taxonomy correction. In other words, the time cost to process taxonomy correction is largely determined by the number of ILP calls invoked[6].

Note that, since siblings in ODP hierarchy are similar in content, sibling intersections dominate when the similarity threshold is large (i.e., strict ≥ 0.85). When similarity threshold is further reduced (i.e., relaxed to ~ 0.8), however, the number of crossing intersections increases. As a result, larger, but fewer mixed components (each of which corresponds to a single ILP call) are produced in the bottom-up scheme. This reduces the number of ILP calls as observed in Figure 16. In the top-down scheme, increased number of crossing intersections leads to a higher number of intersections resolved at the higher levels. While, this also reduces the number of ILP calls, it significantly increases the amount of resulting imprecision (Figure 14).

While, a higher number of ILP calls generally results in longer execution times, the sizes of the ILPs also affect the execution time. For example, consider the cases with similarity threshold 0.8 in Figures 16(b) and 16(c). In these two cases, the bottom-up scheme has the same number of ILP calls. However, when we look at the corresponding execution times in Figures 15(b) and 15(c), we see that the case with $\sim$ 2000 nodes run faster than the case with $\sim$ 6000 nodes. This is because, the average number of vertices involved in the vertex cover problem are $\sim$ 24 and $\sim$ 50, respectively. Naturally, the bottom-up scheme runs slower when the ILPs are larger; nevertheless, on this data set, the bottom-up scheme works faster than the top-down approach in both cases and also provides better results (Figure 14).

[6]We observed a similar behavior for the processing times, reported in Figure 13, for the synthetic data. These results are not included due to space constraints.

6. CONCLUSION

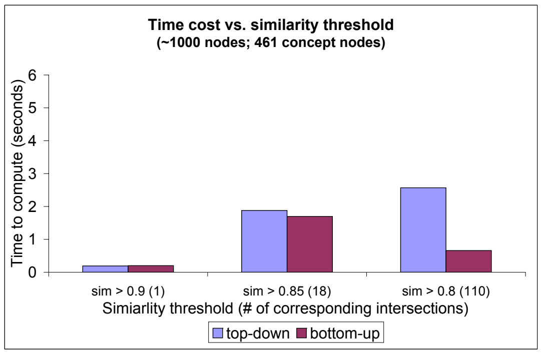
In this paper, we first noted that an integrated concept graph, constructed by combining locally available taxonomies, may not be a taxonomy itself. Such imperfections in integrated data sets pose challenges for OLAP operations, such as drill-down and roll-up, as these usually rely on taxonomical knowledge structures. Specifically, existence of intersections among concepts from different sources compromises the tree-structure of the integrated taxonomy and prevents effective use of hierarchical navigation techniques. In this paper, we formulated the problem of cleaning the concept graph into taxonomies in terms of the task of finding an un-classification strategy which adds minimal imprecision to the resulting structure while eliminating intersections. We also presented an optimal yet efficient algorithm for correcting concept graphs based on a bottom-up preprocessing strategy. Experiments over synthetic and real-life data verified the effectiveness and efficiency of our approach.

Repeatability assessment results

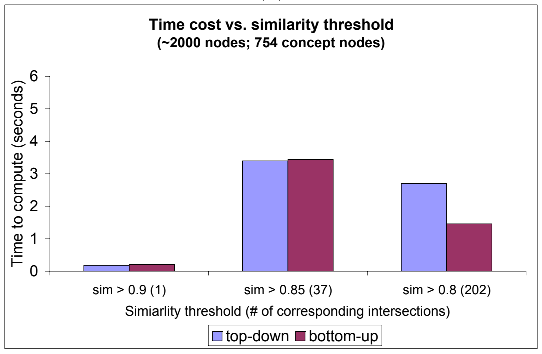
The experimental repeatability committee was not able to repeat the experiments presented in this paper due to the absence of a necessary software library.

7. REFERENCES

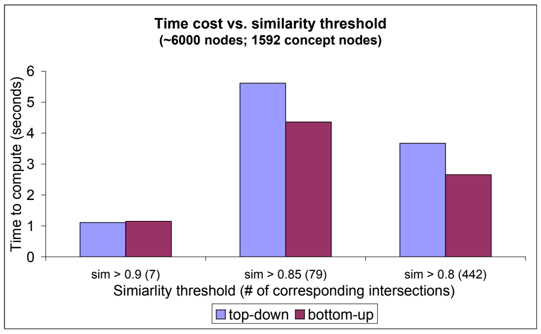
[1] Open directory project (ODP). http://www.dmoz.org/, 2007.
[2] Owl web ontology language. W3C Recommendation, 2004.
[3] O. Banjelloun, A. D. Sarma, A. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. *VLDB*, 2006.
[4] L. Bertossi. SIGMOD Record, pages 68-76, 35(2), 2006.
[5] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Efficient allocation algorithms for olap over imprecise data. *VLDB* 2006.
[6] D. Burdick, P. M. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Olap over uncertain and imprecise data. *The VLDB Journal*, 16(1):123–144, 2007.
[7] D. Burdick, A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Olap over imprecise data with domain constraints. *VLDB* 2007.
[8] K. Chakrabarti, S. Chaudhuri, and S. Won Hwang. Automatic categorization of query results. *SIGMOD* 2004.
[9] Z. Chen and T. Li. Addressing diverse user preferences in sql-query-result navigation. *SIGMOD* 2007.
[10] N. Choi, I.-Y. Song, and H. Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, 2006.



(a)

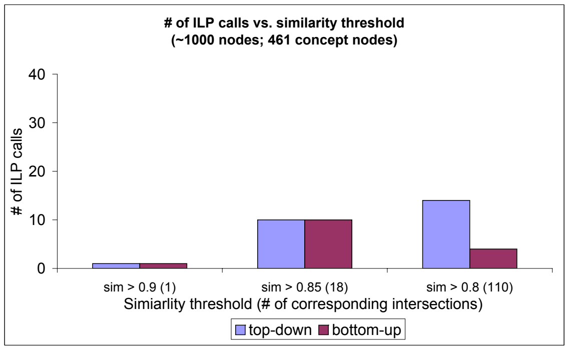


(b)

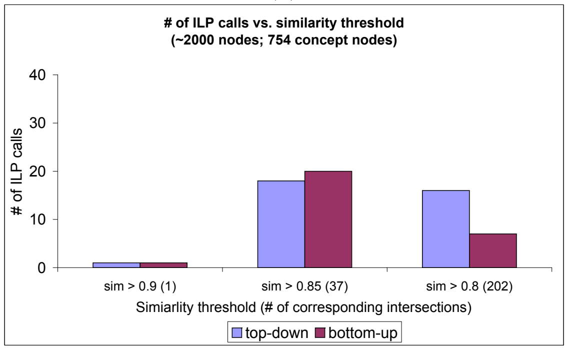


(c)

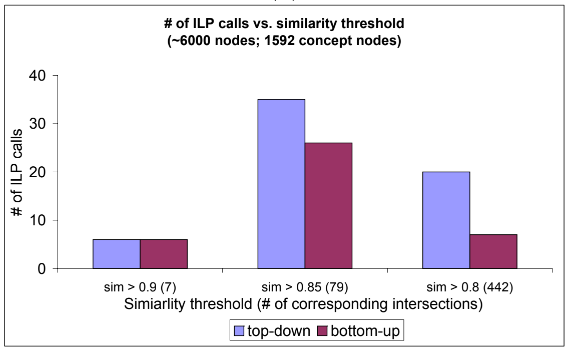
Figure 15: Processing times for taxonomy correction on ODP data



(a)



(b)



(c)

Figure 16: Number of ILP calls for taxonomy correction on ODP data

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[12] I. Dinur and S. Safra. On the hardness of approximating minimum vertex-cover. *Annals of Mathematics*, 162(1), 2005.

[13] A. Doan, P. Domingos, and A. Y. Levy. Learning source description for data integration. *WebDB* ppr. 81-86, 2000.

[14] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal*, 12(4):303–319, 2003.

[15] R. Fagin, R. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. *PODS*, 184–195, 2005.

[16] A. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management. *ICDE*, 2003.

[17] C. A. Hurtado, C. Gutierrez, and A. O. Mendelzon. Capturing summarizability with integrity constraints in olap. *ACM Trans. Database Syst.*, 30(3):854–886, 2005.

[18] C. A. Hurtado and A. O. Mendelzon. Reasoning about summarizability in heterogeneous multidimensional schemas. *ICDT*, pages 375–389, 2001.

[19] M. Lenzerini. Data integration: a theoretical perspective. *PODS*, pages 233–246, June 2002.

[20] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. *VLDB*, 2001.

[21] L. Palopoli, D. Sacca, and D. Ursino. An automatic technique for detecting type conflicts in database schemes. *CIKM*, 1998.

[22] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Supporting imprecision in multidimensional databases using granularities. *SSDBM* , page 90, 1999.

[23] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. *VLDB*, pages 826–873, 2003.

[24] Y. Qi, K. S. Candan, and M. L. Sapino. Ficsr: feedback-based inconsistency resolution and query processing on misaligned data sources. *SIGMOD* 2007.

[25] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. Mediators over taxonomy-based information sources. *The VLDB J.*, 14(1):112–136, 2005.

[26] O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. *SIGMOD*, pp.449-460, 2007.

[27] P. Wu, Y. Sismanis, and B. Reinwald. Towards keyword-driven analytical processing. In *SIGMOD*, pages 617–628, 2007.