On Data Dependencies in Dataspaces

Shaoxu Song
Tsinghua University

This is a joint work with Lei Chen (HKUST) and Philip S. Yu (UIC)
sxsong@tsinghua.edu.cn
2011
Dataspaces

- provide a co-existing system of heterogeneous data
- consider three levels of elements,
  \[ object : \{(attribute : value)\} \]

Example

We consider a dataspace with following objects,

\[
t_1 : \{(\text{name} : \text{iPod}), (\text{color} : \text{red}), (\text{manu} : \text{Apple Inc.}), (\text{tel} : 567), (\text{addr} : \text{InfiniteLoop, CA}), (\text{website} : \text{itunes.com})\};
\]

\[
t_2 : \{(\text{name} : \text{iPod}), (\text{color} : \text{cardinal}), (\text{prod} : \text{Apple}), (\text{tel} : 123), (\text{post} : \text{InfiniteLoop, Cupert}), (\text{website} : \text{apple.com})\};
\]

\[
t_3 : \{(\text{name} : \text{iPad}), (\text{color} : \text{white}), (\text{manu} : \text{Apple Inc.}), (\text{post} : \text{InfiniteLoop}), (\text{website} : \text{apple.com}), (\text{phn} : 567)\}.
\]
Comparable Correspondence

Relationship between elements in heterogeneous data

- **metric operator** ‘manu $\approx_{\leq 5}$ prod’
  any two respective values of manu and prod are said comparable, e.g., Apple Inc and Apple, if their edit distance is $\leq 5$.

- **matching operator** ‘color ⇔ color’
  e.g., red and cardinal are said matched as comparable color, via users’ feedback

  often incrementally recognized in a pay-as-you-go style

A query of (manu : Apple)

- search value similar to Apple in both manu and prod
- e.g., (manu : Apple Inc.) in $t_1$ and (prod : Apple) in $t_2$
Data Dependencies

For wider applications

- integrity constraints, schema design
- optimizing query evaluation, capturing data inconsistency, removing data duplicates

Conventional data dependencies not directly applicable to dataspaces

- often defined on the equality function
- functional dependencies (FDs), $X \rightarrow A$
- specify the constraint of equality between the values of two objects on the same attribute
- e.g., $\text{manu} \rightarrow \text{addr}$
- cannot address the comparable correspondence, in $(\text{manu}, \text{prod})$ or $(\text{addr}, \text{post})$
Comparable Function

Specify constraints on comparable attributes

\[ \theta(\text{manu}, \text{prod}) : [\text{manu} \approx \leq 5 \text{ manu}, \text{manu} \approx \leq 5 \text{ prod}, \text{prod} \approx \leq 5 \text{ prod}] \]

Two objects are said comparable on \((\text{manu}, \text{prod})\) if at least one of these three comparison operators in \(\theta(\text{manu}, \text{prod})\) is applicable.

- \(t_1, t_2\) are comparable on \((\text{manu}, \text{prod})\), since edit distance of \((t_1[\text{manu}], t_2[\text{prod}])\) is \(\leq 5\)
- \(t_1, t_3\) are also comparable on \((\text{manu}, \text{prod})\), where \((t_1[\text{manu}], t_3[\text{manu}])\) satisfy ‘\(\text{manu} \approx \leq 5 \text{ manu}\)’

\[
\begin{align*}
t_1 & : \{ (\text{name} : \text{iPod}), (\text{color} : \text{red}), (\text{manu} : \text{Apple Inc.}), (\text{tel} : 567), \\
& \quad \quad (\text{addr} : \text{InfiniteLoop, CA}), (\text{website} : \text{itunes.com}) \}; \\
t_2 & : \{ (\text{name} : \text{iPod}), (\text{color} : \text{cardinal}), (\text{prod} : \text{Apple}), (\text{tel} : 123), \\
& \quad \quad (\text{post} : \text{InfiniteLoop, Cupert}), (\text{website} : \text{apple.com}) \}; \\
t_3 & : \{ (\text{name} : \text{iPad}), (\text{color} : \text{white}), (\text{manu} : \text{Apple Inc.}), \\
& \quad \quad (\text{post} : \text{InfiniteLoop}), (\text{website} : \text{apple.com}), (\text{phn} : 567) \}.
\end{align*}
\]
Comparable Dependencies (CDs)

A general form of dependencies on comparable functions

\[ \varphi_1 : \theta(\text{manu, prod}) \rightarrow \theta(\text{addr, post}) \]

- if the \text{manu} or \text{prod} values of two products are comparable
- then their corresponding \text{addr} or \text{post} values should also be comparable

where

\[ \theta(\text{addr, post}) : [\text{addr} \approx \leq_9 \text{addr}, \text{addr} \approx \leq_9 \text{post}, \text{post} \approx \leq_9 \text{post}] \]

is another comparable function
Application Example

Query optimization

- consider an object $t_1$ as the query
- to query objects having values similar to $(\text{manu}: \text{Apple Inc.})$ and $(\text{addr}: \text{InfiniteLoop}, \text{CA})$ of $t_1$
- search in the $\text{manu, addr}$ attributes specified in the query,
- also search in the comparable attributes $\text{prod, post}$ according to the comparable functions $\theta(\text{manu, prod})$ and $\theta(\text{addr, post})$
- according to $\varphi_1$, rewrite the query by using $(\text{manu, prod})$ only

\[
\begin{align*}
  t_1 & : \{(\text{name} : \text{iPod}), (\text{color} : \text{red}), (\text{manu} : \text{Apple Inc.}), (\text{tel} : 567), \\
                      & (\text{addr} : \text{InfiniteLoop}, \text{CA}), (\text{website} : \text{itunes.com})\}; \\
  t_2 & : \{(\text{name} : \text{iPod}), (\text{color} : \text{cardinal}), (\text{prod} : \text{Apple}), (\text{tel} : 123), \\
                      & (\text{post} : \text{InfiniteLoop}, \text{Cupert}), (\text{website} : \text{apple.com})\}; \\
  t_3 & : \{(\text{name} : \text{iPad}), (\text{color} : \text{white}), (\text{manu} : \text{Apple Inc.}), \\
                      & (\text{post} : \text{InfiniteLoop}), (\text{website} : \text{apple.com}), (\text{phn} : 567)\}.
\end{align*}
\]
Related Work

Metric functional dependencies (MFDs)
- $X \overset{\delta}{\rightarrow} A$
- equality operator in the left-hand-side
- similarity operator in the right-hand-side
- for violation detection
- e.g., $\text{manu} \overset{2}{\rightarrow} \text{addr}$

Matching dependencies (MDs)
- $[X \approx X] \rightarrow [A \leftrightarrow A]$
- similarity operator in the left-hand-side
- matching operator in the right-hand-side
- for record matching
- e.g., $[\text{addr} \approx \text{addr}] \rightarrow [\text{tel} \leftrightarrow \text{tel}]$
Outline

Introduction

Definition

Validation

Discovery

Experiment

Conclusion
Comparison Operator

We consider a general form of comparison operators, which include the previous operators.

Let $A_i \leftrightarrow_{ij} A_j$ denote a comparison operator between two attributes $A_i, A_j$ in a dataspace $S$:

- equality operator $A_i = A_j$ in functional dependencies (FDs)
- metric operator $A_i \approx_{\lambda} A_j$ in metric functional dependencies (MFDs)
- matching operator $A_i \models A_j$ in matching dependencies (MDs)

The comparison operator indicates true, if two values satisfy the corresponding constraint.
A general \textit{comparable function}

\[
\theta(A_i, A_j) : [A_i \leftrightarrow_{ii} A_i, A_i \leftrightarrow_{ij} A_j, A_j \leftrightarrow_{jj} A_j]
\]

specifies a comparable constraint of two values from attribute \(A_i\) or \(A_j\), according to their corresponding comparison operators.

A \textit{comparable dependency (CD)} \(\varphi\) with general comparable functions over a dataspace \(S\) is in the form of

\[
\varphi : \bigwedge \theta(A_i, A_j) \rightarrow \theta(B_1, B_2)
\]

If two objects have comparable values on \(A_i\) or \(A_j\), then they must have comparable values on \(B_1\) or \(B_2\).
Example

Consider

$$\varphi_4 : \theta(\text{manu, prod}) \rightarrow \theta(\text{tel, phn})$$

where $$\theta(\text{tel, phn})$$ is $$[\text{tel} = \text{tel}, \text{tel} = \text{phn}, \text{phn} = \text{phn}]$$

- we have $$(t_1, t_3) \preceq \text{LHS}(\varphi_4)$$ also agree $$(t_1, t_3) \preceq \text{RHS}(\varphi_4)$$
- denoted by $$(t_1, t_3) \models \varphi_4$$.

$$t_1 : \{ (\text{name} : \text{iPod}), (\text{color} : \text{red}), (\text{manu} : \text{Apple Inc.}), (\text{tel} : 567), (\text{addr} : \text{InfiniteLoop, CA}), (\text{website} : \text{itunes.com}) \};$$
$$t_2 : \{ (\text{name} : \text{iPod}), (\text{color} : \text{cardinal}), (\text{prod} : \text{Apple}), (\text{tel} : 123), (\text{post} : \text{InfiniteLoop, Cupert}), (\text{website} : \text{apple.com}) \};$$
$$t_3 : \{ (\text{name} : \text{iPad}), (\text{color} : \text{white}), (\text{manu} : \text{Apple Inc.}), (\text{phn} : 567) \}. $$
Approximate Dependencies

Due to the extremely high heterogeneity, data dependencies might not exactly hold in a given dataspace.

$$
\varphi_4 : \theta(\text{manu, prod}) \rightarrow \theta(\text{tel, phn}),
$$

- e.g., $$(t_1, t_2) \preceq \text{LHS}(\varphi_4)$$ but $$(t_1, t_2) \not\preceq \text{RHS}(\varphi_4)$$
- i.e., $$(t_1, t_2) \not\models \varphi_4$$

$t_1 : \{ \text{name : iPod}, (\text{color : red}), (\text{manu : Apple Inc.}), (\text{tel : 567}),
(\text{addr : InfiniteLoop, CA}), (\text{website : itunes.com}) \}$;
$t_2 : \{ \text{name : iPod}, (\text{color : cardinal}), (\text{prod : Apple}), (\text{tel : 123}),
(\text{post : InfiniteLoop, Cupert}), (\text{website : apple.com}) \}$;
$t_3 : \{ \text{name : iPad}, (\text{color : white}), (\text{manu : Apple Inc.}),
(\text{post : InfiniteLoop}), (\text{website : apple.com}), (\text{phn : 567}) \}.$
Measure

To evaluate how a dependency “almost” holds in a data instance

- **Error measure**

  \[ g_3(\varphi, S) = \frac{|S| - \max\{|T| \mid T \subseteq S, T \models \varphi\}}{|S|}, \]

  the minimum number of objects that have to be removed from the dataspace \( S \) for a dependency \( \varphi \) to hold.

- **Confidence measure**

  \[ \text{conf}(\varphi, S) = \frac{\max\{|T| \mid T \subseteq S, T \models \varphi\}}{|S|}. \]

  the maximum number of objects reserved after removing minimum objects of violations with respect to \( \varphi \).
Example

\[ \varphi_4 : \theta(\text{manu, prod}) \rightarrow \theta(\text{tel, phn}), \]

Error measure

- \( \{t_2\} \) is a minimum violation set w.r.t. \( \varphi_4 \)
- such that all the remaining objects \( \{t_1, t_3\} \) satisfy \( \varphi_4 \)
- \( g_3 = \frac{1}{3} \)

Confidence measure

- \( \{t_1, t_3\} \) a maximum keeping set w.r.t. \( \varphi_4 \)
- \( \text{conf} = \frac{2}{3} \)

\[ t_1 : \{(\text{name : iPod}), (\text{color : red}), (\text{manu : Apple Inc.}), (\text{tel : 567}), (\text{addr : InfiniteLoop, CA}), (\text{website : itunes.com})\}; \]
\[ t_2 : \{(\text{name : iPod}), (\text{color : cardinal}), (\text{prod : Apple}), (\text{tel : 123}), (\text{post : InfiniteLoop, Cupert}), (\text{website : apple.com})\}; \]
\[ t_3 : \{(\text{name : iPad}), (\text{color : white}), (\text{manu : Apple Inc.}), (\text{post : InfiniteLoop}), (\text{website : apple.com}), (\text{phn : 567})\}. \]
Validation Problem

Unfortunately, computation of error or confidence is generally hard

Given
- a dataspace $S$
- a dependency $\varphi$
- a measure requirement $\eta$

the validation problem is to decide whether or not the measure of $\varphi$ over $S$ satisfies $\eta$.

E.g., to determine whether $g_3(\varphi, S) \leq 0.2$ or $\text{conf}(\varphi, S) \geq 0.8$.

**Theorem**

*The error and confidence validation problems are NP-complete.*
The Hardness

The transitivity cannot be assumed, i.e., from \((t_1, t_2) \simeq \theta(A_i, A_i)\) and \((t_2, t_3) \simeq \theta(A_i, A_i)\) it does not necessarily follow that \((t_1, t_3) \simeq \theta(A_i, A_i)\).

\[
\begin{align*}
t_1 & : \{(A_1 : abc), \ldots \}; \\
t_2 & : \{(A_1 : abcd), \ldots \}; \\
t_3 & : \{(A_1 : abcde), \ldots \}.
\end{align*}
\]

E.g., \(\theta(A_1, A_1) : [A_1 \approx_{\leq 1} A_1]\) with edit distance as metric \(d\)

- \(d(t_1[A_1], t_2[A_1]) = 1 \leq 1\)
- \(d(t_2[A_1], t_3[A_1]) = 1 \leq 1,\)
- but \(d(t_1[A_1], t_3[A_1]) = 2 > 1,\) that is, \((t_1, t_3) \not\simeq \theta(A_1, A_1)\).

The efficient validation computation based on disjoint grouping cannot be applied in this case of comparable functions.
Approximation Computation

Compute an approximate error/confidence measure of $\varphi$ over $S$

- the approximate measure has a relative performance guarantee compared with exact measure,
- e.g., $\hat{g}/g \leq \rho$
- where $\hat{g}$ is an approximation of exact error measure $g$ and $\rho$ is approximation ratio
Greedy Algorithm

- greedily count both objects when a violation occurs
- the complexity is $O(n^2)$

The error approximation

- outputs an estimate $\hat{g}$ with a bound $g \leq \hat{g} \leq 2g$ compared with the exact error measure $g$

Theorem

*The confidence has no constant-factor approximation unless P=NP*

- confidence is NP-hard to approximate within a constant factor
- $g_3$ error and confidence are not equivalent in an approximation-preserving way
Randomized Algorithm

Greedy algorithm still has to consider all the objects in a dataspace.

Randomized algorithm evaluates just a small subset of objects

- randomly draw $m$ samples
- estimate the error/confidence measure by using the violations to these $m$ samples
- the estimate measure is still guaranteed by certain approximation bound with high probability

- e.g., $\Pr[\hat{g} \leq \rho g + \epsilon] \geq \delta$
- $\epsilon$ is an additive error
- $\delta$ is a probability guarantee
- $m$ is determined by $\epsilon$ and $\delta$
Outline

Introduction

Definition

Validation

Discovery

Experiment

Conclusion
Discovery Problem

The strict dependency discovery problem

- find a canonical cover of all dependencies that hold in data
- a canonical cover can be exponential in # of attributes
- high dimensionality in dataspaces (attributes and comparable functions)
- highly non-trivial (if not impossible) to discover a canonical cover of all dependencies

The $k$-length dependencies

- contain $k$ or less comparable functions
- motivated by the concept of mining $k$-length itemsets in association rules
Pay-as-you-go Discovery

In previous work of dataspaces, comparable attributes are identified in a pay-as-you-go style.

The discovery of dependencies should be conducted in an incremental way as well.

- given a set $\Sigma$ of currently discovered dependencies
- and a newly identified comparable functions $\theta(A_i, A_j)$,
- we can generate new dependencies w.r.t. $\theta(A_i, A_j)$ according to the augmentation property
Validation Evaluation

Experiments on real data sets

- compute the exact/approximate measures of dependencies
- observe the corresponding performance
- exact computation does not scale well
- approximation computation keeps significantly lower time cost and scale well
Discovery Evaluation

Illustrate the incremental discovery of dependencies with the increase of functions.

- y-axis is scaled logarithmically
- time cost increases heavily with the number of functions
- the intrinsic hardness in discovering dependencies with respect to attributes (and the corresponding comparable functions)
- size \( k \) of functions also affects the discovery performance largely
Discovery Evaluation

The discovery algorithm scales well in large size of objects

- greedy approximation is adopted for validation
- verify the efficiency of approximation computation proposed for validating dependencies

![Graph showing time cost vs. number of objects for different k values](image-url)
Conclusion

This is the first work to adapt dependencies to dataspaces with the consideration of comparable attribute values.

- It is already hard to tell whether a dependency almost holds in the data.
- The confidence validation is also proved hard to approximate to within any constant factor.
- Propose several greedy and randomized approaches for approximately solving the validation problem.
- Study the pay-as-you-go discovery of dependencies from dataspaces.