Data Sharing:
Querying and Linking Distributed and Autonomous Data
Yang Cao, Wenfei Fan, Peter Buneman
University of Edinburgh

Background

Data sharing: healthcare, e-government, business intelligence, open bank . . .
● Tasks: querying and linking
● Data: distributed, heterogeneous

Challenges:
● data owners ("private") vs. users ("open")
● limited resources (distributed databases) vs. big data analytics (cloud)
● heterogeneous vs. homogeneous

Heterogeneity in Data Sharing

● multiple types of peers
● data owners and users
● varying trust levels and security guarantees
● consequently, varying privacy overheads
● centralised database does not fit

A Model for Querying Shared Data

Data sharing protocols ρ specifying:
● capsules: logic units for computations over shared data
● hosts: data owners that host capsules
● pair-wise privacy requirements:
  ○ permitted capsule specifications
  ○ secure communication overheads

Heterogeneous distributed query plan: A DAG of atomic operations

Step (1): generating toll-minimized distributed plan ξQ
● toll-minimized ξop for each op of Q
● an O(log n)-approximation algorithm for Π

Step (2): optimizing ξQ within the toll budget
● via an atomic operator κ for "rebalancing" ξQ
● a near-optimal design of κ (2-approximation of the optimal for Π)

[Experimental study]:
● heterogeneity has a big impact on querying shared data
● existing systems can be integrated with our method as capsules and alleviate efficiency bottleneck in the heterogeneous setting
● it speeds up SMCQL by 18+ times over 1GB of TPCH data.

Linking Entities across Relations and Graphs

Heterogeneous Entity Linking: given relational database D and graph G, identity tuples in D and vertices in G that refer to the same real-world entity.

[Parametric simulation]:
● take functions and thresholds for measuring vertex closeness, path associations and properties as parameters
● combine topological and semantic matching by extending graph simulation
● decide whether (t, v) is a match in quadratic-time

[Learning parameters]:
● label similarity functions: BERT-based embedding + metric learning
● picking top-k properties of vertices via LSTM network and path resource allocation.

[System HER]:
● convert D to a canonical graph GD following RDB2RDF and then invoke parametric simulation over GD and G
● decide whether t ∈ D and v of G make a match;
● compute all vertices in G that match a given tuple t ∈ D;
● find all matches across D and G.

Joins Across Relations and Graphs

SQL across a relational database D and a graph G via semantic joins.

Algebra: Graph Relational Algebra across relations D and graphs G

Q := R \ P x Q \ σ \{Q | Q1 \times Q2 \ | Q1 \cup Q2 \ | Q1 \ \ Q2 \} \ \ R \bowtie G \ Q \bowtie G.

[Static Join]: R \bowtie G over a relational database D and graph G:

\{(t, v) \ | (t, v) \in f(D, G), (v, id, t') \in h(D, G)\}

\{f(D, G): (t, v) \in f(D, G) \iff t \in D \ \ \mbox{and} \ v \in G \ MAKES A HER MATCH\}

h(S, G) (join attribute extraction): a set of tuples, returns a schema R_C and an instance h(S, G) of schema R_C by extracting corresponding properties of the vertices in f(S, G) that match tuples in S.

[Dynamic Join]: Q \bowtie G where Q is a sub-query that returns a set S of tuples (Q may also contain static/dynamic joins), it returns:

\{(t, v) \ | S = Q(D), (t, v) \in f(S, G), (v, id, t') \in h(S, G)\}

[System RGAP]: supporting semantic joins over existing SQL systems
● h(S, G): (a) sentence and sequence embedding for vertex and edge label encoding; (b) K-means clustering for attribute extraction.
● Emissive joins: dynamic joins that can be reduced to static joins.
● Heuristic joins: approximation of non-entitic dynamic joins

References

Deep Algorithmic Question Answering

Motivation
Ability to reason in a step-by-step "algorithmic" manner that can be inspected and verified for its correctness in the domain of question answering (QA).

We propose **Deep Algorithmic Question Answering** ([1]), an approach to algorithm reasoning for QA based on three desirable properties: interpretability, generalizability, and robustness. We conclude that they are best achieved with a combination of hybrid and compositional AI.

Problem
• Tasks such as the automatic selection of KBs and relevant knowledge, choice of inference algorithms, and how to combine them, are all important to fully automate the QA process.
• We argue that these tasks should be part of the AI models which are built for QA tasks, as they are key ingredients in the full automation of the QA process.

**Proposed Model**
Hybrid inference graphs with functional nodes.

**Inference Graph**
Construct and expanded dynamically through the decompositions of its functional nodes using rules that are learned (see Fig. 2).

**Functional nodes**
• Represent data
• Specify operations to be applied
• Encode a model to convert between the symbolic and vectorized representations of the node.

A Systems Approach
Improving the inference capabilities and explainability of QA systems via "whole system reasoning" [1,2].

**Automatic Knowledge Source Selection** [3]

Aim:
• Discover new knowledge sources.
• Identify and align equivalent entities and relationships (properties) across different knowledge graphs (KGs)

Process:
• Discovery
  • Crawl websites following Linked Data URIs
  • (in Schema.org or JSON-LD formats)
• Introspection: "Upper Ontology" to capture metadata about KGs.
• Alignment: Update existing upper ontology

Usage:
• LOOKUP operation uses upper ontology to find KGs that have relevant data.

**Automatic Statistical Model Selection**

GPy-ABCD ([4])
• A more configurable implementation of the ABCD (Automatic Bayesian Covariance Discovery) system
• An iterative modular Gaussian Process regression framework
• A flexible class of nonparametric models to fit data
• Produces short text descriptions of fit models

SMART: Statistical Methodology Advisor at Reasoning Time
• Selects and performs statistical methods given a query and data features;
• Uses an ontology of various query tags, statistical methods and output types.

**References:**