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 ...

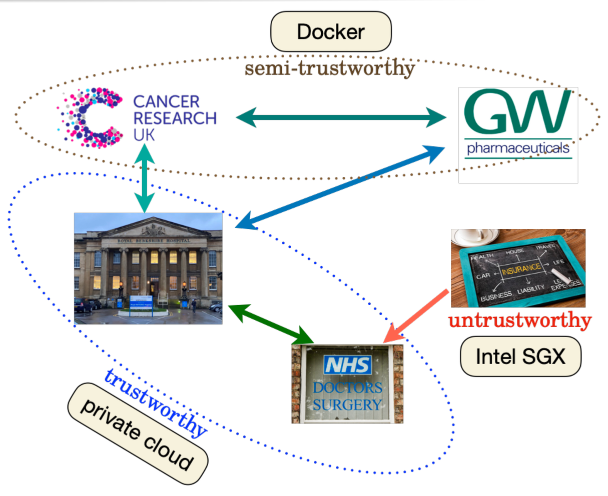
- *Tasks*: querying and linking
- *Data*: distributed, heterogeneous

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



Rethinking query processing for heterogeneous data sharing

A Model for Querying Shared Data[1]

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

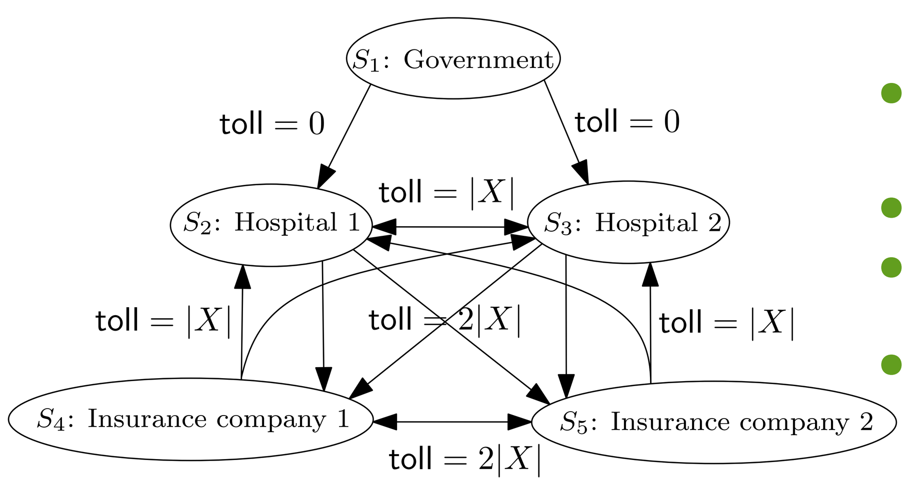
$$\textit{atomic operations}: \delta = (\mathbf{op}, \mathbf{t_c}, X_1, \ldots, X_n, j)$$

- **op**: an **RA** operator
- $\mathbf{t_c}$: type of the security facility for operation δ
- X_i: data from site i for δ
- j: the host site of δ.

Cost model: **toll**(δ):

- upfront cost, secure communication cost, computation cost
- submodular set function: X_i to capsule at j

An example:



- Government: Household(household_id, person_id, name),
- Hospital: Medical(person_id), disease),
- Insurance: Insurance(person_id, policy),
- moving data between the government, hospitals and insurance incurs various tolls.

Q (government): find all households with at least one member contracted disease Z but has no health insurance.

Two possible logic plans:

- ξ_1: $\pi_{\text{household\_id}}((\text{Household} \bowtie \sigma_{\text{disease}=Z}\text{Medical}) - \text{Insurance})$
- ξ_2: $\pi_{\text{household\_id}}(\text{Household} \bowtie (\sigma_{\text{disease}=Z}\text{Medical} - \text{Insurance}))$

ξ_1 yields distributed plans with smaller toll than ξ_2.

Query Evaluation with data sharing heterogeneity

Main problem (TBA): given a database schema $\mathcal{R}$, n sites $\mathcal{S}$, distributed database $\mathcal{D}$ of $\mathcal{R}$ over $\mathcal{S}$, data sharing protocol ρ, RA query Q over $\mathcal{R}$, natural number B, compute a distributed plan ξ for Q over $\mathcal{D}$ with toll at most B.

[Complexity]: TBA (decision version) is

1. decidable in NEXPTIME;
2. PSPACE-hard even when ρ is linear;
3. Σ_3^p-hard even when Q is in **SPC** and ρ is linear.

Moreover, 2 and 3 hold even when $\mathcal{S}$ has two sites only.

Intractable to make the best (optimal) use of heterogeneity in data sharing.

[Algorithms]: a two phase approach:

Step (1): generating toll-minimized distributed plan ξ_Q

- toll-minimized $\xi_{\mathbf{op}}$ for each **op** of Q
- an $O(\log n)$-approximation algorithm for $\bowtie$

Step (2): optimizing ξ_Q within the toll budget

- via an automic operaton κ for "rebalancing" ξ_Q
- a near-optimal design of κ (2-approximation of the optimal for $\bowtie$)

[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[2]

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 G_D following RDB2RDF and then invoke parametric simulation over G_D and G
- decide whether $t \in D$ and v of G make a *match*;
- compute all vertices in G that match a given tuple $t \in D$; or
- find all matches across D and G.

Joins Across Relations and Graphs[3]

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

Algebra: `Graph Relational Algebra` across relations D and graphs G

$$Q ::= R \mid \pi_X Q \mid \sigma_C Q \mid Q_1 \times Q_2 \mid Q_1 \cup Q_2 \mid Q_1 \setminus Q_2 \mid R \bowtie G \mid Q \bowtie G.$$

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

$$\{(t, t') \mid (t, v) \in f(D, G), (v.\text{id}, t') \in h(D, G)\}$$

- $f(D, G)$: $(t, v) \in f(D, G)$ iff t and v make a HER match
- $h(S, G)$ (*join attribute extraction*): given a set of tuples, returns a schema R_G and an instance $h(S, G)$ of schema R_G 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:

$$\{(t, t') \mid S = Q(\mathcal{D}), (t, v) \in f(S, G), (v.\text{id}, t') \in h(S, G)\}$$

[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.
- Entitic joins: dynamic joins that can be reduced to static joins.
- Heuristic joins: approximation of non-entitic dynamic joins

References

[1] Yang Cao, Wenfei Fan, Yanghao Wang, and Ke Yi. Querying shared data with security heterogeneity. In *SIGMOD*, 2020.

[2] Wenfei Fan, Ruochun Jin, Ping Lu, Resul Tugay, and Wenyuan Yu. Linking entities across relations and graphs. In *ICDE*, 2022.

[3] Yang Cao, Wenfei Fan, Wenzhi Fu, Ruochun Jin, Weijie Ou, and Wenliang Yi. Joins across relations and graphs. In *in submission*.

Data Sharing: Querying and Linking Distributed and Autonomous Data

WP6: Intelligent Query Answering

Kwabena Nuamah, Alan Bundy
k.nuamah@ed.ac.uk, a.bundy@ed.ac.uk

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.

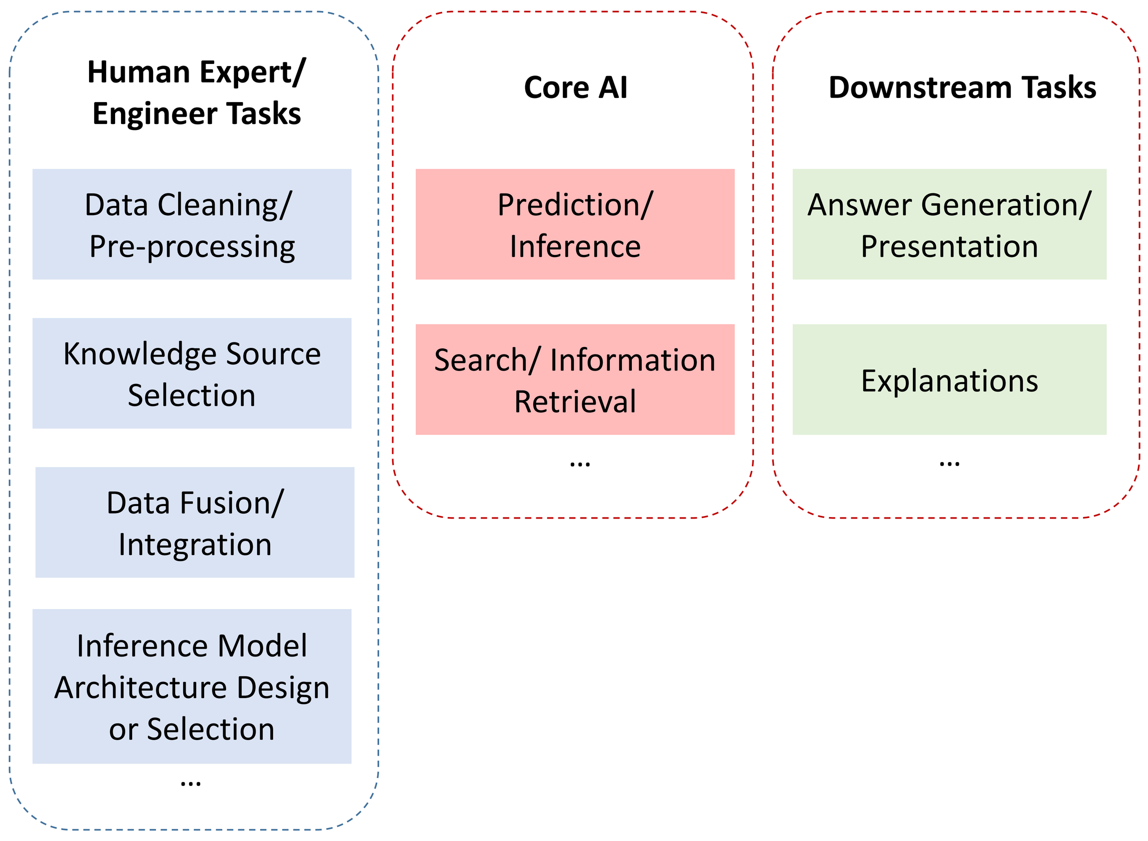
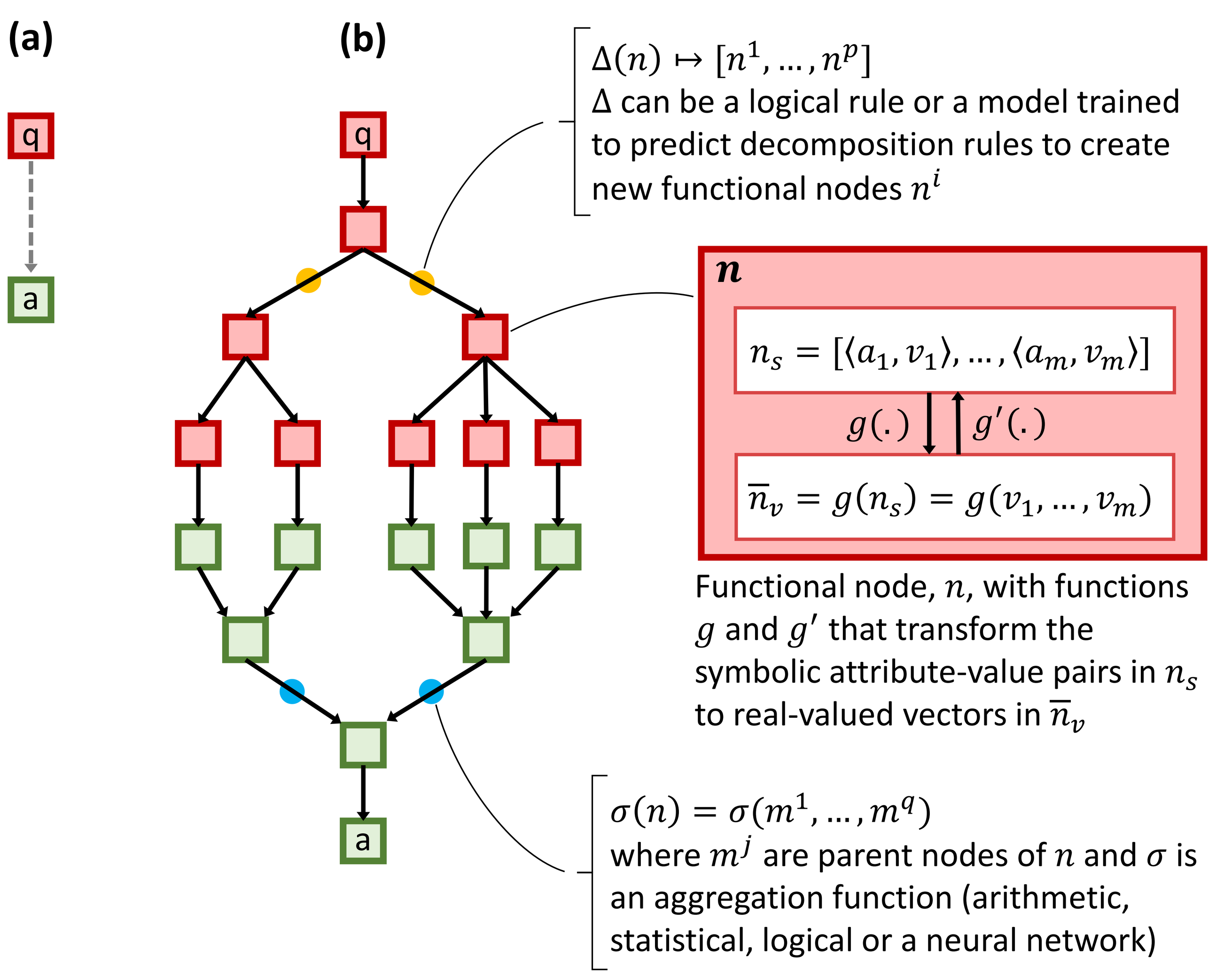


***Fig. 1.** Diverse tasks that are part of the open-domain question answering process. However, most of the attention in work related to QA focus on the core AI tasks related to information retrieval, inference or prediction*



Proposed Model

Hybrid inference graphs with functional nodes.

Inference Graph

Constructed 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.

Fig. 2.
***(a)** Shows the base inference graph with a question node and an answer node that is to be inferred. They are linked by an edge that can be split by applying decomposition operations on the question node.*
***(b)** An inference graph made up of functional nodes and edges labelled by operations for predicting decomposition and aggregation functions. Decomposition sub-graph (in red) is guided by a function Δ that decomposes a functional node to create new continuations of the inference graph, and aggregation sub-graph (in green) which uses a model σ to select appropriate functions to combine nodes. Functional nodes provide both a symbolic and vector representation of the node's attribute-value internal representation, as well as function g and g' for converting between the two representations.*

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.

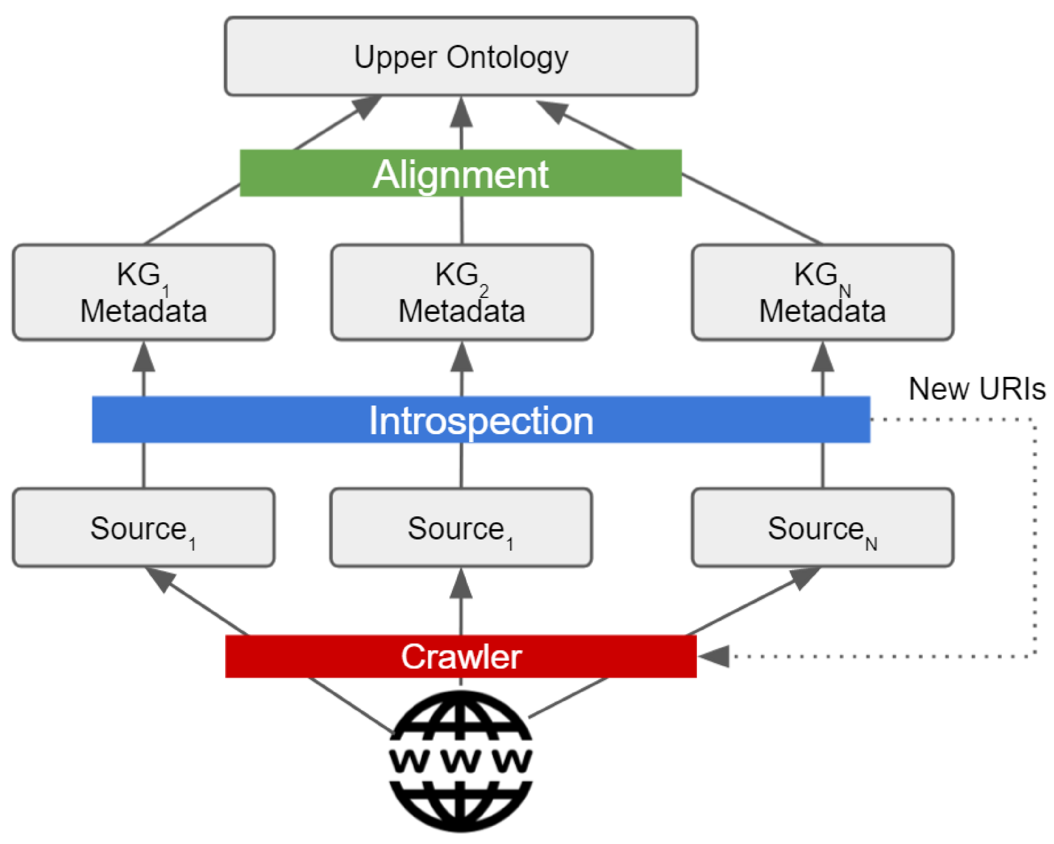


***Fig. 3.** Automatic knowledge source discovery process*

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

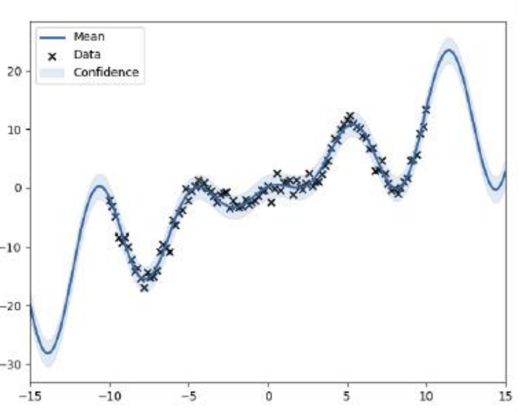
SMART: Statistical Methodology Advisor at Reasoning Time

- Selects and performs statistical methods given a query and data features;
- Uses an ontology of various query tags, statistical methods and output types.

Fit Example

- Synthetic data: $X = [-10,10]$ every 0.2, $Y = 2X\cos\left(\frac{X-5}{2}\right)^2 + \epsilon,\ \epsilon \in \{-1,1\}$
- Top result:

```
LIN * (PER + C)
  mul.                                |                value  |  constraints
  linear_with_offset.variance         |  0.00868862409368869  |      +ve
  linear_with_offset.offset           | -0.08541236236743312  |
  sum.pure_std_periodic.variance      |    62.48387020446294  |      +ve
  sum.pure_std_periodic.period        |   6.2443744834482535  |      +ve
  sum.pure_std_periodic.lengthscale   |    1774.034537156076  |      +ve
  sum.bias.variance                   |   119.28065536635225  |      +ve
```



- Description:
  "The fit kernel consists of 2 components:
  - A linear function with offset -0.09; this component has overall variance 1.04
  - A periodic function with period 6.24 and lengthscale 1774.03, with linearly varying amplitude with offset -0.09; this component has overall variance 0.54"
- The runners-up are direct generalisations: `LIN * (PER + LIN)` and `LIN * LIN * (PER + C)`

Acknowledgement: This work was funded by Huawei grant HO2017050001B8s.

References:

[1] Nuamah, K. (2021). *Deep Algorithmic Question Answering: Towards a Compositionally Hybrid AI for Algorithmic Reasoning*. KR 2021 Workshop on Knowledge Representation for Hybrid and Compositional AI (KRHCAI).
[2] Bundy, A.; Nuamah, K. (2021). *Combining Deductive and Statistical Explanations in the FRANK Query Answering System*. Keynote talk at the 12th IEEE International Conference on Big Knowledge (ICBK 2021).
[3] Brimble, G. (2021). *Automatic knowledge discovery*. UG Project Dissertation, School of Informatics, University of Edinburgh.
[4] Fletcher, T.; Bundy, A.; Nuamah, K. (2021). *GPy-ABCD: A Configurable Automatic Bayesian Covariance Discovery Implementation*. 8th ICML Workshop on Automated Machine Learning (AutoML).

School of Informatics