Approaches to Search over Structured Data

Benny Kimelfeld

LogicBlox

* Mentioned work was done while at IBM Research – Almaden
  and The Hebrew University of Jerusalem
Outline

- Overview of Approaches
- Search for Data Connections
- Search for Schema Connections
- Search Interpretation via Auxiliary Data
- Concluding Remarks
<table>
<thead>
<tr>
<th>Query</th>
<th>Data</th>
<th>Answer</th>
<th>Comments</th>
</tr>
</thead>
</table>
| (Structured) DB | Connected set of tuples/items | • An answer connects the keywords  
|                |                               | • Usually data=graph & answer=subtree |
| DB + schema   | DB query (over the schema)    | • Can be an intermediate step towards tuple sets  
|                |                               | • Usually, answer = CQ that connects the keywords |
| Docs + aux. DB | Document                      | • Auxiliary DB has a meaningful relationships between keywords; extracted from docs |
| Tree document (XML / JSON) | node/subtree                 | • Each subtree is a separate document (overlaps…)  
|                |                               | • XML structure used to infer relationship among kw.s |
## Plan

<table>
<thead>
<tr>
<th>Query</th>
<th>Data</th>
<th>Answer</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>(Structured) DB</td>
<td>Connected set of tuples/items</td>
<td><strong>Data Connection</strong> [PODS'06, SIGMOD'08, SIGMOD'10, PVLDB’11]</td>
</tr>
<tr>
<td></td>
<td>DB + schema</td>
<td>DB query (over the schema)</td>
<td><strong>Schema Connection</strong> [SIGMOD’09, PODS’11, ICDT’13]</td>
</tr>
<tr>
<td></td>
<td>Docs + aux. DB</td>
<td>Document</td>
<td><strong>Search Interpretation</strong> [PODS’10, PODS’11, CIKM’12, SIGIR’12]</td>
</tr>
<tr>
<td></td>
<td>Tree document (XML / JSON)</td>
<td>node/subtree</td>
<td><strong>Not Covered</strong></td>
</tr>
</tbody>
</table>
Outline

• Overview of Approaches
• Search for Data Connections
• Search for Schema Connections
• Search Interpretation via Auxiliary Data
• Concluding Remarks
Systems

[X]Keyword [Balmin et al., 2003] (XML)

BANKS [Bhalotia & al., 2002] (rel.)

[DBXploer [Agrawal et al.]
[DISCOVER [Hristidis et al.]
[SPARK [Lou et al.]
[Blinks [He et al.]
[RDF search [Tran et al.]

[eu brussels] search

[Carrey Williams] search

[European Union] search

Result
author["Widom"] ← paper["TSIMMIS System"] ←
author["Ullman"]

author["Widom"] ← paper["Production Rules"] ←
paper["Constraint Checking"] → author["Ullman"]

author["Widom"] ← paper["LORE"] → paper["TSIMMIS System"] → author["Ullman"]
## Core Problem – Formulation

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
</tr>
</thead>
</table>
| • Graph $G$  
  – Data  
  – Nodes =  
    • Tuples  
    • XML elements  
    • Values/words  
  – Directed / undirected  
  – Weights represent connection strength  
| • Top-k answers  
| • Answer = subtree $T$ of $G$  
  – $T$ contains all nodes of $Q$  
  – $T$ is non-redundant  
    ▪ Cannot remove any node/edge  
| • Score($T$) is inversely proportional to weight($T$)  
  – Smaller is better  
  – $1^{st}$ is “minimal Steiner tree”  
| • **Top-k Steiner trees**  
| • Usually, heuristic algorithms w/o provable guarantees |
Why Heuristics? Justifications vs. Theory

“**The Steiner-tree problem is NP-hard**” (that is, hard to find a minimal tree connecting a set of nodes)

- The corresponding Steiner-tree problem can be solved in time \(3^{\#kws} \cdot \text{PTIME}(G)\)
  - Adaptations of Dreyfus-Wagner’s [1972]
- Plethora of Steiner-tree approximations

“**Even if Steiner tree is solved, only gives top-1**”

[K & Sagiv PODS’06]:

Every Steiner-tree alg. can be extended to top-k
Delay between consecutive results (& approx. ratio) is ≈ that of top-1
System Research

• Search-engine implementation
  – [Achiezra, Golenberg, K, Sagiv SIGMOD’08, SIGMOD’10]
  – Optimization, diversity of search results, …

• Key technique: Lawler-Murty top-k algorithm
  – [Murty 1968, Lawler 1972]

• Not so simple…
  – Initial implementation was too slow…
  – Purchased a stronger server
  – Didn’t help! **Cores highly under utilized**
  – Needed deeper understanding of ranked enumeration to benefit from parallelization
Goal: **Find top-k answers**

- **$O = \{ A \}$** is a collection of objects.
- **$A$** is huge, described by a condition on $A$'s subsets.
- **Score** of answers $a \subseteq O$.
- **Goal**: Find top-$k$ answers $a_1, a_2, \ldots, a_k$ where $score(a)$ is high, implying $a$ is of high-quality.
Lawler-Murty’s Procedure

Lawler-Murty’s gives a general reduction:

Finding top-k answers

then PTIME

if PTIME

Finding top-1 answer under simple constraints

We understand optimization much better!

Often, amounts to classical optimization, e.g., shortest path (but sometimes it may get involved, e.g., [KS 2006])
Among the Uses of Lawler-Murty’s

**Graph/Combinatorial Algorithms:**
- Shortest simple paths [Yen 1972]
- Minimum spanning trees [Gabow 1977, Katoh et al., 1981]
- Best solutions in resource allocation [Katoh et al. 1981]
- Best perfect matchings, best cuts [Hamacher & Queyranne 1985]
- Minimum Steiner trees [K & Sagiv 2006]

**Bioinformatics:**
- Yen’s algorithm to find sets of metabolites connected by chemical reactions [Takigawa & Mamitsuka 2008]

**Data Management:**
- ORDER-BY queries [K & Sagiv 2006, 2007]
- Graph/XML search [Golenberg, K, Sagiv 2008]
- Generation of forms over integrated data [Talukdar et al. 2008]
- Course recommendation [Parameswaran & Garcia-Molina 2009]
- Querying Markov sequences [K & Ré 2010]
Experiments: Graph Search
2 Intel Xeon processors (2.67GHz), 4 cores each (8 total); 48GB memory

Mondial

DBLP

Improvements Achieved [PVLDB’11]
Outline

• Overview of Approaches
• Search for Data Connections
• Search for Schema Connections
• Search Interpretation via Auxiliary Data
• Concluding Remarks
Data vs. Schema Connection

DB → data graph

graph algorithm
  e.g., top-k Steiner-trees

search results

schema → query gen.

query\_1 → DB

query\_2 → search results

query\_3 → DB

query engine
  SQL, XQuery, …
The Problem (Informal)

```
SELECT * FROM employee e, dept d
WHERE e.name='Mary' and e.dept=d.id

(Mary’s department)

SELECT * FROM employee e1, employee e2, dept d
WHERE e1.name='Mary' AND e1.dept=d.id AND
d.head=e2.eid AND e2.type='manager'

(Mary’s manager)

SELECT * FROM employee e
WHERE e.name='Mary' AND e.type='manager'

(managers named Mary)

{ Jacob, Mary, manager } { dept, dept } { Mary, dept, dept } ...
```
And in XML

DTD

<!ELEMENT department (facility|(branch*,headq))>
<!ELEMENT branch (facility)>
...

{ facility, headq } ➔ department[branch/facility][headq]

department
  └── branch
        └── facility

{ facility, facility }

department
  └── branch
        └── facility

department[branch/facility][branch/facility]
Aided Query Formulation

QUICK [Zenz et al., 2009]

ExQueX [K, Sagiv, Weber, SIGMOD ’09]
Building Integration Forms

Q System
[Talukdar et al., 2008]
Techniques for Query Extraction

Incremental query construction (*candidate networks*)
- Discover/XKeyword [Balmin et al.]
- SPARK [Lou et al.]
- QUICK [Zenz et al.]

Subtree enumeration on the *schema graph*
- DBXploer [Agrawal et al.]
- ExQueX [K et al.]
- Q System [Talukdar et al.]

- Acyclic queries — *tree patterns*
- Ranking dominated by size: **smaller = better**
  - Post-processing resorting to support more complex scoring functions
Techniques for Query Extraction

Incremental query construction (*candidate networks*)

- Discover/XKeyword [Balmin et al.]
- SPARK [Lou et al.]
- QUICK [Zenz et al.]

**No guarantee on the running time**

Can construct exponentially many intermediate, partial patterns, before finding even 1 tree pattern

Subtree enumeration on the *schema graph*

- DBXploer [Agrawal et al.]
- ExQueX [K et al.]
- Q System [Talukdar et al.]

**Repeated labels are not allowed**

No neighborhood constraints!
Why Neighborhood Constraints?

<table>
<thead>
<tr>
<th>eid</th>
<th>name</th>
<th>type</th>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>e68</td>
<td>Jacob</td>
<td>regular</td>
<td>a</td>
</tr>
<tr>
<td>e22</td>
<td>Marie</td>
<td>manager</td>
<td>b</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>sales</td>
<td>e78</td>
</tr>
<tr>
<td>b</td>
<td>db</td>
<td>e22</td>
</tr>
</tbody>
</table>
Efficiency Matters!

Existing solutions usually experiment on tiny schemas; an exception:

**Q System** [Talukdar et al, VLDB’ 08]

<table>
<thead>
<tr>
<th>K</th>
<th>KBEST-STEINER (s)</th>
<th>SPCH (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>43.8</td>
<td>3.0</td>
</tr>
<tr>
<td>3</td>
<td>111.8</td>
<td>5.5</td>
</tr>
<tr>
<td>5</td>
<td>1006.9</td>
<td>13.9</td>
</tr>
</tbody>
</table>

- **408** relations
- **1366** table references

**Speed is important**

queries are typically extracted in response to a user query…

Schemas can be much larger, e.g.,
- SAP instances w/ > 13k relations [Kemper et al. 99]
- Sybase/SAP instances w/ >20k views [Mulero 07]
- KB concepts: >350K in YAGO, >2.5M in ProBase …
### Problem Definition

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Schema $S$</td>
<td>$k$ smallest $(S, \Lambda)$-patterns</td>
</tr>
<tr>
<td>• Bag $\Lambda$ of labels</td>
<td></td>
</tr>
<tr>
<td>• Number $k \in \mathbb{N}$</td>
<td></td>
</tr>
</tbody>
</table>

Weights allowed  
Defined next:
Example

\( S \)

\( \Lambda = \{ \text{author, author, chair} \} \quad k = 10 \)
$(S, \Lambda)$-Pattern

$S$:
- session
- chair
- paper
- demo
- author

$\Lambda = \{\text{author, author, chair}\}$  \hspace{1cm} k = 10

A tree $t$, such that:
- $t$ contains $\Lambda$
- $t$ is consistent with $S$
- $t$ is nonredundant
\((S, \Lambda)\)-Pattern (consistency)

\(S\) = 
- session
- chair
- paper
- demo
- author

\(\Lambda = \{\text{author, author, chair}\}\) \hspace{1cm} k = 10

A tree \(t\), such that:

- \(t\) contains \(\Lambda\)
- \(t\) is consistent with \(S\)
- \(t\) is nonredundant

\(\forall\) nodes \(v \in t: \)

\(\text{labels}(\text{nbrs}(v)) \subseteq \text{some allowed bag for label}(v)\)
\((S, \Lambda)\)-Pattern (consistency)

\(S\) contains:
- session
- chair
- paper
- demo
- author

\(\Lambda = \{author, author, chair\}\)

A tree \(t\), such that:
- \(t\) contains \(\Lambda\)
- \(t\) is consistent with \(S\)
- \(t\) is nonredundant

- \(t\) contains \(\Lambda\) and is consistent with \(S\)
- \(t\) is nonredundant
\((S, \Lambda)\)-Pattern (consistency)

\(S\):
- session
- chair
- paper
- demo
- author

\(\Lambda = \{\text{author, author, chair}\}\) \(k = 10\)

A tree \(t\), such that:
- \(t\) contains \(\Lambda\)
- \(t\) is consistent with \(S\)
- \(t\) is nonredundant
$(S, \Lambda)$-Pattern (nonredundancy)

$\Lambda = \{\text{author}, \text{author}, \text{chair}\}$  \hspace{1cm} k = 10

A tree $t$, such that:

- $t$ contains $\Lambda$
- $t$ is consistent with $S$
- $t$ is nonredundant

leaf-labels($t$) $\subseteq \Lambda \subseteq$ labels($t$)
## Our Contribution

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Schema $S$</td>
<td>$k$ smallest $(S, \Lambda)$-patterns</td>
</tr>
<tr>
<td>• Bag $\Lambda$ of labels</td>
<td></td>
</tr>
<tr>
<td>• Number $k \in \mathbb{N}$</td>
<td></td>
</tr>
</tbody>
</table>

Contribution [PODS’11, ICDT’13]:

**A Polynomial-Time Algorithm**

- $\Lambda$ is fixed! (Necessary already for $k=1$)
- In fact, Fixed Parameter Tractable (FPT): $O(c^{|\Lambda|} \cdot \text{poly}(S,k))$
- Constraint language should satisfy a tractability condition
  - Regular expressions are fine
  - So are the tractable *mutual-exclusion graphs* [K & Sagiv PODS’11]
Not Just Search…

- Query generation is useful beyond keyword search!
- Aided query formulation [Q System, ExQueX, QUICK]
- “Schema-free” queries [Li, Cong, Jagadish 2008, Li. Pan, Jagadish 2014]: abstract query languages with built-in “naturally connected” predicates
  - For example, Connected[SSN, Address]
- Feature generation for ML models
Outline

• Overview of Approaches
• Search for Data Connections
• Search for Schema Connections
• Search Interpretation via Auxiliary Data
• Concluding Remarks
Email Search (Running Example)

**Search**

**from sara john number**

Interpretation:
Find emails that contain the words “from” “sara” “john” and “number”

Emma,
Please call John at 713-853-4145.
Sara Shackleton
Enron North America Corp.
1400 Smith Street, EB 3801a
Houston, Texas 77002
713-853-5620
sara@enron.com
Sources of Auxiliary Data

• Information Extraction (IE)
  – For example, find all the *persons*, *phone numbers*, and the *person-phone* pairs in the corpus of documents

• Domain Knowledge
  – Email search: email headers (metadata), user’s address book, etc.
  – Enterprise search: business data, HR data, etc.
  – Online store search: product database, etc.
  – Generic dictionaries (e.g., country names)
  – …
Motivating Systems at IBM

- **IBM Gumshoe** (enterprise search)
  - **Backend**: crawling, text analytics (e.g., IE), page classification, grouping “similar” pages, etc.
  - **Search runtime**: rule-based interpretation of the user query against the rich index
  - Powers IBM search

- **IBM OmniFind Personal Email Search**
A schema is a partially ordered set of concepts

+ a partial order over the atomic concepts

employee ⊆ person

“employee is more specific than person”
A database is a set of records
(atomic & compound records)

- **Record Identifier**: r3087
- **Atomic Concept**: company
- **Textual Content**: IBM Corp.
- **Document Identifier**: doc15

- **Record Identifier**: r1034
- **Atomic Concept**: person
- **Textual Content**: John
- **Document Identifier**: doc23

- **Record Identifier**: r309
- **Atomic Concept**: phone
- **Textual Content**: (713) 853-4145
- **Document Identifier**: doc23

- **Record Identifier**: r2437
- **Atomic Concept**: prph
- **Textual Content**: r1034, r309
- **Document Identifier**: doc23

**Notes**:
- Atomic records
- Compound record
- Compound concept
- Pointers to records of subconcepts
From Search Queries to DB Queries

We augment the search query with a set of interpretations, which are database queries:

- Thus, we go from a string of words to a structured database query.

The interpretations are produced by applying rewrite rules.

Each interpretation can be evaluated over the database, resulting in a set of documents:
- In practice, only a few interpretations are evaluated.

**Interpretation:**

Find emails from sara, s.t. the phone no. of the person “john” is included.
Interpretations in the Running Example

**from sara john number**

Each of the four keywords should occur in the document.

The sender is “sara, “john” is included, some phone no. is included.

The sender is “sara,” person “john” is included along w/ his phone no.
Examples of Rewrite Rules
Research: Foundations

- **Framework** [Fagin, K, Li, Raghavan, Vaithyanathan, PODS’10]
  - “Search database systems”
  - Specificity (or containment) of interpretations
  - How to produce (top-specific, nonempty) interpretations?

- **Convergence** [Fagin, K, Li, Raghavan, Vaithyanathan PODS’11]
  - *How to apply rewrite rules to the search query?*
  - **Simple way**: each rule applied once, predefined order
  - **Thorough way**: least fixpoint (apply repeatedly)
    - Problem: “bad” rule sets lead to non-termination
      - Real problem: detecting non-termination is undecidable
    - Robust & tractable “safety” guarantees termination
Research: Administrator & User Tools

• Automating rule inference [Bao, K, Li, SIGIR 2012]
  – Given a “user complaint,” suggest an appropriate rewrite
    • Appropriate = solves the problem + semantically “natural”
    • We took a machine learning approach (classification)
  – Given a large collection of (probably conflicting) rules, select a subset of the rules to optimize performance on a benchmark

• Utilize domain knowledge for search spelling correction [Bao, K, Li, ACL 2011]

• Toolkit [Bao, K, Li, Raghavan, Yang, CIKM 2012]
  – Search provenance to explain ranking
  – Embedded rule manager (search/edit)
  – Rule suggestion (in progress)
  – Used daily by IBM search administrators
Outline

• Overview of Approaches
• Search for Data Connections
• Search for Schema Connections
• Search Interpretation via Auxiliary Data
• Concluding Remarks
Concluding Remarks

• Discussed core ideas underlying various paradigms for search on data with structure
  – Data connection: data as graph, answers as subtrees
  – Schema connection: extract coherent queries from the schema to connect the keywords
  – Interpretation: construct queries by applying grammar rules

• Direction: effective & elegant way of combining schema-connection and rules for interpretation