Query-Driven Integration: The Q System

Zachary G. Ives
University of Pennsylvania

with Partha Pratim Talukdar, Marie Jacob, Fernando Pereira, Koby Crammer, Sudipto Guha

Funded in part by NSF IIS-0477972, 0513778, 0629846 and the DARPA CSSP program

AKBC Workshop
May 18, 2010
A Goal for Many Domains, esp. in the Sciences: Making the Internet into a Topic-Specific Oracle

“Show me the evidence for links between exposure to cell phone radiation and levels of brain cancer”
Standard Data Integration Architecture: A Pipeline with Many Stages

Query-time (End user)

Design-time (DB admin)

Standardize schema

Discover sources

Entity resolution

Schema mapping

Information extraction

Corresp. table

Query via UI

Design-time (DB admin)
Standard Data Integration Architecture: A Pipeline with Many Stages

Query-time (End user)

Design-time (DB admin)
Standard Data Integration Architecture: A Pipeline with Many Stages

**Design-time (DB admin)**

1. Target schema may depend on user & context
2. Not all query results relevant: some sources may conflict, or have data user finds less useful
3. Schema mapping may produce wrong answers
4. Entity resolution may produce wrong answers
5. New source may require changes to target schema and query formulation

**Query-time (End user)**

Corresp. table
The Problems, in a Nutshell

Design-time stages need to be all things to all people

- Need to predict usage models \textit{a priori}
- Too much complexity, too diffuse attention – leads to bugs
  - even output of semi-automated tools needs vetting
- Difficult to move back and forth between the stages

Data integrators don’t scale! (But we might have lots of data experts...)
The Q System: An Integration Loop, Not a One-Directional Pipeline

Eliminate the design-time / query time separation (and data integrator) altogether:

- Everything happens at query time – when we can see the output!

  "Query-driven data integration"

- Schema matching, entity resolution focused on what’s needed by the query

- Customize behavior (ranking, mappings, ...) based on feedback on the results!

Key assumptions (true in life sciences, sometimes true elsewhere):

- Data sources are “fairly” amenable to automated schema alignment & record linking: good recall, not necessarily precision

- Users know their data and will invest time to get the right results:
  Provide feedback on output results to improve data quality
Q System Architecture: Facilitating Data Integration as a Loop

Ranked results based on evidence supporting each answer

- Evidence factors = *features*, with strengths

Keywords:
- query concepts + predicates
- via schema alignments, join paths
- sets of features recorded with each query
- weighted combination of features’ strengths

1. Find source data items
2. Connect source data items
3. Create unions of queries as query form
4. Compute results & scores
5. Collect feedback on answers

sources & indices

Learn from feedback

XML sources & indices via schema alignments, join paths recorded with each query weighted combination of features’ strengths

metadata

reweighting of evidence factors

provenance of answers + feedback
Outline

- Basic usage model of Q
- Query-driven data integration
- Experimental validation
- Related Work
- Conclusions and next steps
Basic Usage Model of Q: Initial Setup

- Crawl or collect sets of databases, extracted Web sources, etc.
  - Represent these (and their attributes, and data) as nodes in a search graph

- Add association edges representing known links, key-foreign key dependences, etc.

- Run initial schema alignment tools to create association edges – with no human vetting
  - Weights on edges based on confidence
Search-Driven Integration in Q

Given a search graph
And a set of keywords

- Compute set of query trees matching keywords
  - Generate a parameterizable query form for the user
  - Includes $q_1 \cup q_2$

Query Keywords $a, e, f$
The Parameterizable Query Form

The Q System: Answering Your Information Needs

Query: MembraneInfo — "protein", "plasma membrane", "gene", and "disease"

Show more attributes  Change keywords  Create related query

Please specify any filter values for query query, and check fields that should be returned in the results:

- PROTEIN_NAME
- PROTEIN_FUNCTION
- GENE_DESCRIPTION
- DISEASE_FEATURES

Run query!

from ProSite.TblProtein.DESCRIPTION, ...

from GO.Term

from Entrez.GeneInfo

from OMIM.MAIN.SA
## Getting User Feedback

### The Q System: Answering Your Information Needs

<table>
<thead>
<tr>
<th>Query</th>
<th>MembraneInfo</th>
<th>- &quot;gene&quot;, &quot;protein&quot;, &quot;plasma_membrane&quot;, &quot;disease&quot;,</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GENE_DESCRIPTION</strong></td>
<td><strong>PROTEIN_NAME</strong></td>
<td></td>
</tr>
<tr>
<td>ETHYLENE INSENSITIVE ROOT I</td>
<td>Phosphopantetheine attachment site.</td>
<td></td>
</tr>
<tr>
<td>alkaline phosphatase 4</td>
<td>Prokaryotic membrane lipoprotein lipid attachment site profile.</td>
<td></td>
</tr>
<tr>
<td>This gene encodes a sialoglycoprotein that is expressed on mature granulocytes</td>
<td>ADF-H domain profile.</td>
<td></td>
</tr>
<tr>
<td>ETHYLENE INSENSITIVE ROOT I</td>
<td>Casein kinase II phosphorylation site.</td>
<td></td>
</tr>
<tr>
<td>iisopropylmalate synthase</td>
<td>AdoMet activation domain profile.</td>
<td></td>
</tr>
<tr>
<td>alkaline phosphatase 4</td>
<td>Asparagine-rich region profile.</td>
<td></td>
</tr>
</tbody>
</table>

System determines “producer” queries using provenance.

- $q_1$
- $q_1, q_2$
- $q_2$
- $q_2$
Learning New Weights

Change weights so $q_2$ is “cheaper” than $q_1$
Outline

- Basic operation of Q
- Query-driven data integration

- Experimental validation
- Related Work
- Conclusions and next steps
Finding Source Data: Search Graph (Bioinformatics Example)

Populate a graph with nodes for relations, attributes, values:

- Interrelate items through *part-of* and *is-a* edges
- Add edges from foreign key to key attributes
Finding Source Data: Matching Keywords Against the Graph

Query: “GO term name ‘plasma membrane’ publication titles”

Given a set of keywords $K_1 \ldots K_n$:
- Add a keyword node for each $K_i$
Finding Source Data: Matching Keywords Against the Graph

Given a set of keywords \( K_1 \ldots K_n \):

- Add a **keyword node** for each \( K_i \)
Finding Source Data: Matching Keywords Against the Graph

Given a set of keywords $K_1...K_n$:

- Add a **keyword node** for each $K_i$
- For each node $n$ matching a $K_i$ (using TF/IDF, n-grams, etc.)
  - Add an edge $(K_i, n)$, with cost $\approx -\log_2(\text{similarity}(K_i, \text{label}(n)))$
Combining Source Data: Schema Element Alignment

Consider “frontier” nodes transitively connected to keywords

- For each relation node $n_f$
  - Run schema matcher against every other unconnected relation
Combining Source Data: Schema Element Alignment

Consider “frontier” nodes transitively connected to keywords

- For each relation node $n_f$
  - Run schema matcher against every other unconnected relation
  - Add edges $(n_{f.a}, n_{o.b})$ to the graph; cost $\approx -\log_2(\text{confidence}(n_{f.a}, n_{o.b}))$
Combining Source Data: Schema Element Alignment

Consider “frontier” nodes transitively connected to keywords

- For each relation node $n_f$
  - Run schema matcher against every other unconnected relation
  - Add edges $(n_{f,a}, n_{o,b})$ to the graph; cost $\approx -\log_2(\text{confidence}(n_{f,a}, n_{o,b}))$
Combining Source Data: Schema Matching

Consider “frontier” nodes transitively connected to keywords

- For each relation node $n_f$ @ frontier and other rel. node $n_o$ in the graph
  - Align attributes of $n_f$, $n_o$ using schema matcher(s)
  - Add edges $(n_{f.a}, n_{o.b})$ to the graph; cost $\approx -\log_2(\text{confidence}(n_{f.a}, n_{o.b}))$
Creating a Query Template & Form

- Find trees connecting all keyword nodes
Creating a Query Template & Form

- Find trees connecting all keyword nodes
  \[
  \text{term}(a, \text{"plasma membrane"}), \text{Interpro2GO}(a,e,p), \text{entry2pub}(a,p), \text{pub}(p,t)
  \]
  \[
  \text{cost} = f(s_1, s_2, s_4, c_{a1}, c_{a2}, c_{f2}, s_5, c_{abbrev})
  \]
  \[
  \text{UNION}
  \]

...
Query Answering: Basic Query Cost Model

Each query’s cost is the dot product of a vector of features with strengths, plus vector of learned weight assignments:

- e.g., schema alignment cost $c_{a2}$ has features for Interpro2GO, entry2pub, confidence score of each matcher

- we assign weights to the significance / relevance of each
  
  cost = $w_{I2G} f_{I2G} + w_{e2p} f_{e2p} + w_{coma} f_{coma} + …$

- The query will produce tuples *annotated* with feature vector and weight vector – not merely the final cost

Note: summing negative log likelihoods is equivalent to multiplying probabilities, assuming independence
Each tuple is annotated with provenance recording query that produced it, and evidence for the query

Query provenance is the feature vector + weight vector for the query

Data provenance for a tuple is the source tuples joined, for each derivation of the tuple

We return top-k results, with cost defined in terms of a variant of the provenance semiring model of ORCHEstra [GKT07]:

- cost of each feature (including data-dependent features) is its learned weight times its strength
- combining multiple features: sum the costs
- alternate derivations: return the min of the costs
Generalizing Feedback: MIRA [Crammer+06]

User sees results, provides feedback of the form:
- “Query tree T* should have less cost than query tree T”

\[
\mathbf{w}^{(i)} = \operatorname{arg\,min}_{\mathbf{w}} \| \mathbf{w} - \mathbf{w}^{(i-1)} \|
\]

\[s.t. \quad C(T, \mathbf{w}) - C(T^*, \mathbf{w}) \geq L(T, T^*)\]

\[
L(T, T') = |E(T) \setminus E(T')| + |E(T') \setminus E(T)|
\]
Suppose a New Source Comes Along…
Incrementally Add Mappings to Queries

Relatively easy to incrementally update our search graph, query results, etc.

Can do this more efficiently using pruning:

- Use cost of the $k$th best answer to the query as a cost bound $B$
- Only consider potential schema alignments (edges) that are within $B$ of any of the keyword nodes
Outline

- Basic operation of Q
- Query-driven data integration

- Experimental validation
- Related work
- Conclusions and next steps
Highlights from Experimental Evaluation [PVLDB08], [SIGMOD10]

- Two different sets of questions, evaluated over real bioinformatics data with gold standard:
  - Can we learn a user’s perception of the ranks / authoritativeness / relevance of different data sources?
    - ... in a scalable way?

- Can we automate the whole data integration process – directly use the results of “black-box” schema matching tools?
  - Combine confidence scores from different schema matchers
  - “Repair” alignments based on feedback: raise cost of bad alignments, drop cost of good alignments
  - COMA++ [Do] and a graph-random-walk algorithm called MAD
Given a search graph with correct join associations, can we learn the right rankings, as determined by experts?

Series of 25 queries, 28 relations from BioGuide [Cohen-Boulakia+07]

After feedback on 40-60% queries, Q finds the top query for all remaining queries on its first try!

For each individual query, feedback on one item is enough

Can it scale?

Generated top queries at interactive rates for ~500 tables
Can It Combine & Repair Automated Schema Matchers’ Output? [SIGMOD10]

**COMPOSITION OF MATCHER OUTPUTS:**
Precision vs recall at different confidence thresholds; top-5 results.

Q is given feedback on 10 2-term queries

**REPAIRING ALIGNMENTS:**
Comparison of average cost for correct ("gold") and incorrect ("non-gold") edges as feedback is applied
Outline

- Basic operation of Q
- Query-driven data integration

- Experimental validation
- Related work
- Conclusions and next steps
Related Work

**Dataspaces** [Franklin+ 07], **pay-as-you-go integration** [Das Sarma+08][Vas Salles+07], **mass collaboration** [Doan+06], YAGO [Suchanek+07], [Sarawagi], **many others seek scalable integration**

**Incomplete and uncertain information** [Imielinski & Lipski 84], [Sadri 98], [Dalvi & Suciu 04], [Widom 05], [Antova+ 07], [Deshpande+ 08], [Domingos], [Getoor], [McCallum]...

**Integrated data provenance and debugging** [Cui&Widom01], [Buneman+01], [Bagwat+04], [Widom+05], [Chiticariu & Tan 06], [Green+07], Clio, ...

**Search and ranking in DBs:** [Bhalotia+ 02], [Kacholia+ 05], [Hristidis & Papakonstantinou 02], [Marian+04], ...

**Authority and rank:** [Balmin+ 04][Varadarajan+ 08][Kasneci+ 08]
Conclusions & Next Steps

- We Avoid the design-time / query-time divide via query-driven integration and a user feedback loop
  - Learning and provenance are key!
  - More details in [Talukdar+ VLDB08][Talukdar+ SIGMOD10]
  - Results are very promising, but this is only a first step...

Open questions:

- Incorporation of IE into the feedback loop
- Curation of output – what about updates to results? Where and how to propagate?
- How do we cluster and share among users?
- Can we incorporate full probabilistic models, scalably?
- Can we demonstrate effectiveness through user studies?