Qunits: Queried Units in Database Search

Arnab Nandi[1] and H. V. Jagadish[2]
[1]Computer Science and Engineering, [2]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI

Motivation

- Keyword Search in databases
- Simplicity is important
  - Keyword Search works
  - SQL / Xquery does not
- Consider search "PWP1"
  - Ambiguous information need
    - User may not how to search
    - May not know what DB has
- There is a need to define exactly what we expect as a result

Qunits

- The user has a "mental model" of how a database is organized
  - It does ***not*** need to correspond with the internal schema
- Our goal is to guide the user to their information need
- ***Qunit: Queried Unit***
  - basic, independent semantic unit of information in a database.
  - Atomic piece of information to be returned for a query
- A small number of *Qunit Definitions* exist
  - e.g. "*interactions*"
- When applied to the database, they generate *Qunit Instances*
  - *e.g. "list of interactions with CKM… FHL2, GAMT, MYOC…"*
- Qunits are ranked based on their "Qunit Utility"
  - The utility of a qunit is the importance of a qunit to a user query, in the context of the overall intuitive organization of the database.

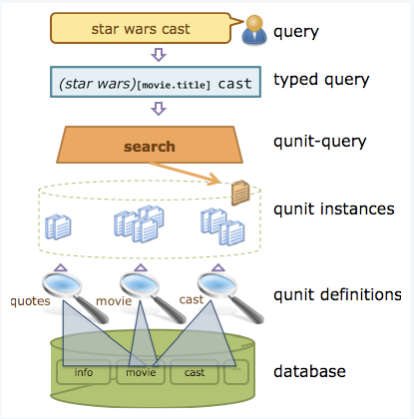
The Qunit Paradigm

- Derive Qunits
- Index qunits
  - (just like documents)
- Process Query
- Search over collection
  - return best qunit(s)

Example, for MiMI:

- Analyze MIMI database
- Derive Qunit definitions
  - *Interactions*
  - *Genomic Info*
  - *Pathways*
  - *Protein info*
- For search "CK"
- Look for closest Qunit
  - *[gene.name]* usually matches *Genomic Info* qunits
  - return *"CK" Genomic Info* qunit that closest matches the query string "CK"
- Example Qunit-based result for query *"CK"*

CK: Creatine Kinase	
locations	brain, muscle
locus	CKM: 19q13.2-q13.3 CKB: 14q32 …
interactions	SERP2



Why are Qunits better?

- Search Quality
  - Predefined qunits = meaningful results
  - Comparing 2 results makes sense
- Data Integration
  - Search across multiple databases = solving multiple search problems at the same time
  - Each database outputs qunits, which are put into a unified search pool

Qunit Derivation

- Human generated
  - Ideal, but not tractable
- Schema & Data
  - Look at Schema
- Query rollup
  - Look at existing query logs
- External Evidence
  - Look at published instances of results

Qunit Derivation: Query Rollup

- Look *outside* of the database
- Great when data+schema are not sufficient
- **"The result of a query is the union of all *specialized(i.e. stricter) versions* of that query"**
  - ***[gene] :=***
    - SELECT gene FROM gene X interaction X gene as G WHERE G.name = "$"
    - SELECT location FROM gene WHERE gene.name = "$"
    - [gene.name] isoenzymes
    - SELECT organism.location FROM gene X organism WHERE gene.name = "$"

Query Rollup Example

- Sample Query Log

query	freq.	classifier
creatine kinase	4233	[gene]
pwp1 homolog	3000	[gene]
creatine kinase, brain	233	[gene] [org.location]
NCAM1 genomic location	100	[gene] genomic [location]
creatine kinase brain location	21	[gene] [org.location] [location]
pwp1 SMURF1 in vivo	19	
creatine kinase muscle	11	[gene] [org.location]
creatine kinase isoenzymes	10	[gene] isoenzymes
creatine kinase, brain SERP2	10	[gene] [org.location] [gene]

- Given a keyword query log
- For each query
  - *creatine kinase, brain* = [gene] [org.location]
  - Look for all *specializations* of it
    - *creatine kinase brain location*
      - *[gene] [org.location] [location]*
    - *creatine kinase, brain SERP2*
      - *[gene] [org.location] [gene]*
  - Construct queries from specialization
  - Unify typed specializations into qunit definition
- Return qunit definitions for popular query types

Evaluation

- IMDB Dataset
  - 14 tables, 34M rows.
- Query workload
  - 25 keyword queries, 13 types
  - Most popular types in AOL query log
- Perform search using each query algorithm
  - Results "normalized" to plaintext
- Classify each result: *does this satisfy the query*
  - 5 possible classes

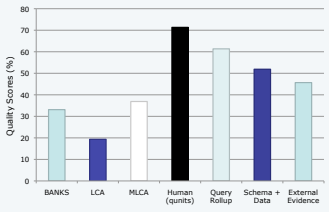
User Consensus

Exact	1	3	0	2	1	0	0	2	0	3
Correct	16	1	16	0	0	18	18	1	17	0
Partial	2	13	3	16	14	1	1	14	2	14
Query	1	2	1	2	1	1	1	1	1	1
Incorrect	0	1	0	0	4	0	0	2	0	2

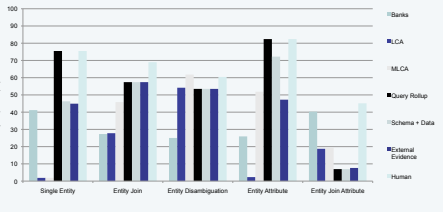
Relevance Classes

score	rating
0	provides incorrect information
0	provides no information above the query
.5	provides correct, but incomplete information
.5	provides correct, but excessive information
1.0	provides correct information

Evaluating Search Quality



Types of queries



Further details

Qunits is part of the Database Usability project. Further details are available at **http://www.eecs.umich.edu/db/usable**

Acknowledgements

This work was supported by National Institutes of Health: Grant #U54 DA021519 and a Yahoo! Research Fellowship.