Keyword Searching on Databases

Ramdas Rao

Indian Institute of Technology Bombay

CS632 Course Seminar Presentation
April 6, 2007

Outline

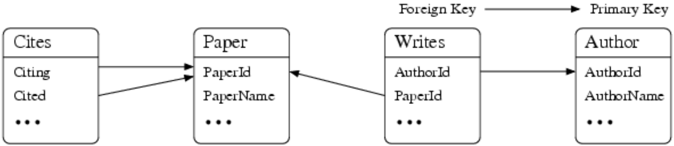
Outline

Motivation

- For searching db
  - Knowledge of detailed schema, SQL needed
  - Need to create separate UI forms for searching relations
- IR seems to be appropriate:
  - But cannot be directly applied to databases
  - Answer to a query typically split across multiple tuples
  - Alternative: combine db data into a “document"
  - Disadvantage: Duplication of data; Sync with db

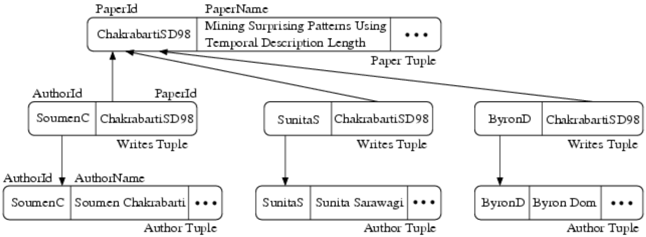
DBLP Example

- Normalization $\Rightarrow$ multiple tuples (through fk)



DBLP Example

- Note: 1 paper spread across 7 tuples



Outline

Motivation

BANKS - Introduction

BANKS - Bidirectional Expanding Search

SphereSearch

Summary

BANKS

- BANKS = Browsing and Keyword Searching
- Convergence of IR and searching structured databases
- User specifies keyword(s)
  - no SQL, no detailed schema knowledge required
- Answers are ranked
  - Further user interaction may be needed to narrow down info
- Useful for publishing data on web: no coding required!

System Architecture

- Java Servlets (for web interface)
- JDBC to communicate with the RDBMS
- Configuration by administrator

BANKS : Browsing

- Browsable view of database relations
  - no content programming / user intervention required
- Drop-down menu with operations on column headers
- Projections, selections
- Joins for fk columns (or for pk used by a referencing fk)
- Grouping of results; drill-down
- Sorting
- Pagination and Schema browsing

BANKS : Templates

- Templates can be used for formatting display of tuples
  - Can contain HTML code snippets
  - Hyperlinks to attributes
  - Relationships to be folded in
- Cross-tabs
- Group-by template
- Folder-tree views
- Pie, bar, line charts (with drill down)
- Templates can be composed together in visual manner

BANKS : Templates...contd

[STUDENTS, THESIS]

SNAME	FEMAIL	TITLE
Nand Kumar Singh	sudhakar@aero.iit	nd : of and :is
N. Shama Rao	mujumdar@aero.iitb.ernet.in	L ON OF THROUGH THICKNESS ELASTIC CONSTANTS AND STRENGTHS OF ADVANCED FIBRE COMPOSITES
Mini N Balu	svs@math.iitb.ernet.in	Some Preservation Results in Mathematical Theory of Reliability

Get column info
Drop column
Sort in Ascending order
Sort in Descending order
Group by
Group by prefix
Join (FACULTY)
Select

BANKS : Keyword Searching

- Specify keywords to be searched for
- Answers to query in relevance order
- Each answer displayed in hierarchical form
- Example answer tree
- Indentation and color used to depict the tree structure

BANKS Query Model

- DB as a directed graph
  - Graph is in-memory
- Each tuple in the db corresponds to a node in the graph
- Each fk-pk link is a directed edge between the corr. tuples
  - Can be easily extended for other types of connections

BANKS Query Model...contd

- Keyword query has $n \geqslant 1$ terms (t_1, t_2, t_3, ... , t_n)
- Locate the nodes matching the search terms
  - Matching on attribute value or metadata (col name, tbl name, view name)
  - Use disk-resident indices to map keywords to RIDs
  - Another (in-memory) index to map RIDs to graph nodes
- S_i: set of nodes matching keyword t_i
- S_i's may overlap

BANKS Query Model...contd

- An answer is a subgraph connecting a set of nodes that cover the keywords
- Important to identify a “central" node that connects all the keyword nodes
- An answer is then a rooted directed tree
  - at least one node from each S_i
  - edges are directed away from the root
- Tree may also contain nodes that are not in S_i (a Steiner tree)

Answer Relevance

- Two types of weights:
  - Edge weights
  - Node weights (Prestige ranking, such as PageRank)

Edge weights

- Importance of a link depends on the type of link (relations, semantics)
  - link between Paper and Writes v/s link between Paper and Cites
- Semantically stronger links given lower weights
- Wt. of a tree $\propto$ sum of its edge weights
- Relevance of a tree inversely $\propto$ to its weight
  - Sort Answer trees in increasing order of weight

Need for directionality

- Consider earlier example: some links point toward root of tree, others away (e.g., Writes to Author and to Paper)
  - we require paths from Paper to Author; that is, traverse fk edge in opposite direction
- Can we ignore directionality?
  - If we do, problem of "hubs"
  - E.g., a dept. with large # of faculty and students
  - Many nodes would be within a short distance of many other nodes
  - Reduces the effectiveness of tree-wt based scoring mechanism

Backward Edges

- For each edge (u, v), create a backward edge (v, u)
- This ensures that a directed tree exists that is rooted at the "paper" with a path to each leaf
- To solve the hub problem
  - wt. of (v, u) = wt. of (u, v) * f(# of links to v from the nodes of the same type as u)
  - if an edge already exists from v to u, set the edge weight to the lower of the 2 weights
- Experiments indicate that the function $log(1 + x)$, where x is the # of inlinks, provides good results.

Node Weights

- Inspired by prestige rankings (Google PageRank)
- Nodes with more inlinks get higher node weight (higher prestige)
  - DBLP: More citations for paper, more inlinks
- Node weight function could be $log(1 + x)$, where x is the in-degree

Overall relevance score

- Combine node weights and tree weights (total of edge weights)
- Additive or multiplicative combination
  - λ controls relative weightage
  - Additive:
$$(1-\lambda)\mathit{Escore} + \lambda \mathit{Nscore}$$
  - Multiplicative:
$$\mathit{Escore} \cdot \mathit{Nscore}^{\lambda}$$
  - Both $+$ and $*$ work well when relative weights are appropriately chosen

Algo

- First, for each keyword
  - find the set of nodes S_i that satisfy the keyword term t_i
- Let $S = \cup S_i$
- Backward Expanding Search algo (Heuristic incremental solution)
  - Concurrently run $\mid S \mid$ copies of Djikstra's single src shortest path algo
  - One copy for each node n in S, with n as the source

Algo...contd

- Each copy of the single src SP algo traverses the graph edges in the reverse direction
- Try to find a common vertex from which forward path exists to at least one node in each set S_i
- Rooted directed tree (connection tree):
  - info node as root
  - keyword nodes as leaves

Algo...contd

- Connection trees approximately sorted in increasing order of weights
- All connection trees could be generated and then sorted in decreasing relevance order
- Better alternative:
  - When output heap is full, output highest relevance tree and replace it
  - No guarantees trees sorted in decreasing relevance order, but works well

Isomorphic trees

- Trees with similar structure modulo direction (“duplicate trees")
- These represent the same result, with diff. info. nodes.
- Retain only one with the highest relevance
  - Note: Results are output when the heap is full

Implementation

- Efficiency of graph traversals important
- Entire db graph is stored in memory. Acts as an index on the db
- Graph stores only id for each node and edge plus pointers
  - Each graph node: 30 bytes
  - No strings in memory
- Tens of millions records using modest amount of memory

Results

- Most intuitive answers ahead of less intuitive ones in almost all cases
- Space and Time:
  - For a bib. db with 100K nodes and 300K edges, mem. util around 120 MB
  - 2 minutes initial loading time
  - Once loaded, queries take a second / few seconds
  - Feasible to use BANKS for moderately large db

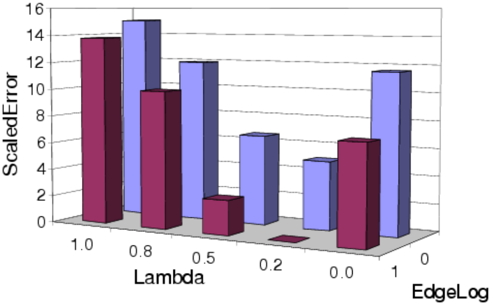
Effect of Parameters : Settings

- 7 different queries; for each, $\sim$4 ideal answers were listed
- Each query run with diff. param combinations (10 answers)
- Rank diffs computed for each run
- Raw error score = $\sum$ *rankdiffs*
- For missing answers
  - rank diff = 11

Effect of Parameters

- Important to keep the effect of node ranking relatively small, but non-zero
  - $\lambda = 0.2$, $\mathit{EdgeLog} = 1$ did best with error score of 0.0
  - $\lambda = 0.5$, $\mathit{EdgeLog} = 1$ - almost well with error scores of around 3
  - $\lambda = 1$, (ignore edge weights) - error score   15
- Conclusion: $\lambda = 0.2$, $\mathit{EdgeLog} = 1$ - does best

Effect of Parameters...contd



Effect of Parameters...contd

- Reducing edge wt. range by log scaling important
  - else
    - back edges from some popular nodes get high weights
    - some intuitive answers got a very poor relevance ranking
- Mode of score combination has almost no impact on the ranking
- For node weights, log scaling gave same results as no log scaling

Extensions

- Extended to handle XML???
- Selection conditions (*year* $= 2007$), (*year* ~ 2007)
- Ranking function: “near" `movies (near hitchcock, reagan)`
- User Feedback:
  - Disambiguation of nodes
  - Selecting answer tree patterns
  - Re-scoring

Related Work

- DataSpot system
  - similar model, relevance scores, trees of max relevance returned
  - Back edges based on in-degree and node weights not present in DataSpot
- Proximity search in db - Goldman
  - find object near object
- EasyAsk:
  - keyword search on data stored in RDBMS
  - but details are not available publicly

BANKS and related work

- BANKS differs from prior work:
  - Techniques for edge wt. computation and prestige ranking
  - Use of an in-memory graph structure for very efficient searching

Future Work

- Improved user feedback
- Querying across multiple data sources using different data models
- XML data
- attribute:keyword queries (e.g., author:Levy)

Outline

Motivation

BANKS - Introduction

BANKS - Bidirectional Expanding Search

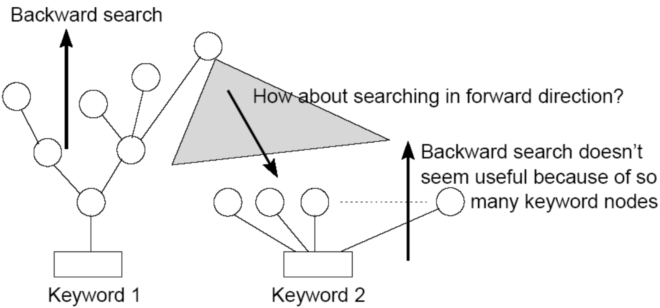
SphereSearch

Summary

Issues with Backward Expanding Search Algo

- Travel backwards from keyword nodes till you hit a common node
- Performs poorly if:
  - Some keywords match many nodes
  - Some node has a very large indegree
- In these cases, a large number of nodes must be examined
- Wasteful exploration of graph
- Longer time to generate answers

Bidirectional Expanding Search Algo



Bidirectional Expanding Search Algo

- Basic idea:
  - Do not explore backward if:
    - Next node is a hub
    - Keyword matches a large number of nodes
- But, at what number do we switch over?

Bidirectional Expanding Search Algo

- Prioritize on the basis of *spreading activation*
  - Like propagating “scent" spread from keyword nodes
  - Edge weights as well as spread of the next node(s)
- Nodes with the highest activation explored first
- Higher the spread, lower the activation

Bidirectional Expanding Search Algo

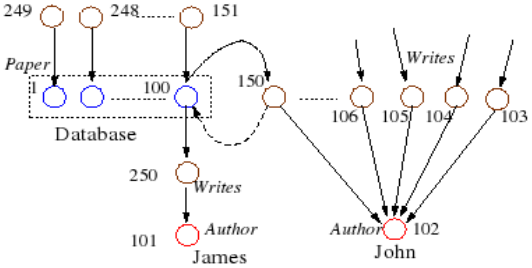
- Initial activation:

$$a_{u,i} = \frac{nodePrestige(u)}{\mid S_i \mid}, \forall u \in S_i$$

- For spreading, use an “attenuation factor" μ
- Each node keeps μ fraction of the activation it receives
- Rest $(1 - \mu)$ is divided amongst its neighbors
- Overall activation of a node u is:

$$a_u = \sum_{i=0}^{n} a_{u,i}$$

Bidirectional Expanding Search Algo



Bidirectional Expanding Search Algo

- Use a single combined iterator for all nodes in each direction
- Lesser state maintenance overhead than the Backward Expanding Search Algo
- Also, the iterator is not a single source shortest path iterator
- So,
  - need to update path lengths as they become known
  - need to worry about output of answers in relevance order

Bidirectional Expanding Search Algo

- Activate matching nodes; insert into backward iterator
- while (iterators are not empty)
  - Choose iterator for expansion in best-first manner
  - Explore node with highest activation
  - Spread activation to neighbors
  - Update path weights (and other datastructures)
  - Propagate values to ancestors if necessary
  - Insert nodes explored in the backward dir into fwd iterator // *for future forward exploration*
  - Stop when top-k results are produced

Top-k results

- Naïve approach:
  - Store results in an intermediate heap
  - Output top k results after mk total results have been generated ($m \sim 10$)
- Can do better:
  - Compute upper bound on score of next result
  - Output answers with a higher score

Experimental Results

- Bidirectional Expanding Search outperforms Backward Expanding Search
- Current BANKS demo on site has flexibility
  - User can choose which algo to use for searching

Outline

SphereSearch : Motivation

- Web search engines use mostly keyword query paradigm
  $\Rightarrow$ less expressive querying capabilities
- Example
  - Web search: *researcher Max Planck*
  - We want to say: *researcher person="Max Planck"*
- We want:
  - Concept-aware search (Tag-aware querying)
  - Context-aware search
    - Answers to queries might be a *set* of pages, than a *single* page
  - Abstraction-aware search (Ontology-enabled search)

SphereSearch Features

- Uniform treatment for XML as well as present Web data
- Structured queries on semistructured data without a global schema
  - Heterogeneous XML
- Relevance-ordered results using ranked retrieval paradigm

SphereSearch Query Language

- Query Groups and Joins

  `A(gift, vendor)`

  `B(courier, vendor)`

  `A.location = B.location`
- Similarity operator, $\sim$, used for:
  - Ontology
  - For numeric, “approximately" (year $\sim$ 2007)

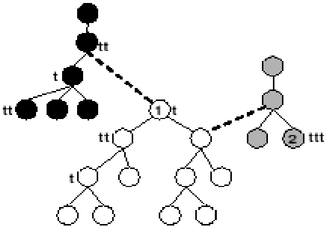
SphereSearch Transformation and Annotation

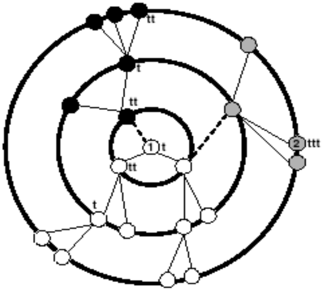
- Convert HTML, PDF, plain text to XML

```
<H1>Experiments</H1>   <Experiments>
     ...Text1...         ...Text1...
<H2>Settings</H2>   =>   <Settings>
       ...Text2..            ...Text2..
<H1>...                  </Settings>
                       </Experiments>
```

- Then, annotate data (e.g., identify places and tag them `<places>`)

Spheres





Spheres

- Sphere of node n at distance d is $S_d(n)$: set of all nodes at distance d from node n
- Sphere Score at distance d of node n wrt condition t is:

$$s_d(n,t) = \sum_{v \in S_d(n)} ns(v,t)$$

- Sphere Score of node n wrt t is:

$$s(n,t) = \sum_{i=1}^{D} s_i(n,t) * \alpha^i$$

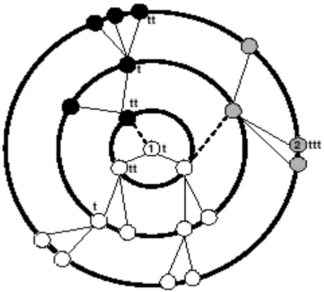
  - D: Sphere size limit
  - α: Damping coefficient

Spheres

- For $\alpha = 0.5$ and $D = 3$, we get:

$$\begin{aligned} s(1,t) &= 1 + 4 \cdot 0.5 + 2 \cdot 0.5^2 + 5 \cdot 0.5^3 = 4.175 \\ s(2,t) &= 3 + 0 \cdot 0.5 + 0 \cdot 0.5^2 + 1 \cdot 0.5^3 = 3.125 \end{aligned}$$

- Node 1 is a better result for “t" than node 2



Outline

Summary

- BANKS: useful for web publishing of data
- Bidirectional Expanding Search outperforms Backward Expanding Search
- SphereSearch: More expressive query language

References I

Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan.
Keyword Searching and Browsing in Databases using BANKS.
ICDE, 2002.

Varun Kacholia, Shashank Pandit, Soumen Chakrabarti, S. Sudarshan, Rushi Desai, Hrishikesh Karambelkar.
Bidirectional Expansion For Keyword Search on Graph Databases.
VLDB, 2005.

Jens Graupmann, Ralf Schenkel, Gerhard Weikum.
The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents.
VLDB, 2005.