Query and Answer Models for Keyword Search

Rose Catherine K.
Roll no: 07305010

Seminar under the guidance of

Prof. S. Sudarshan
Computer Science and Engineering
Indian Institute of Technology Bombay

Introduction

- **Keyword Searching :** unstructured method of querying
- greatest advantage: requires no knowledge of the underlying schema
- keyword search in databases:
    - database normalization
    - table joins done on the fly
    - unique characteristics of databases: different types of edges, attributes of nodes, semantics associated with tables
    - physical database design affects performance: availability of indexes on certain columns
- notion of relevance

Representing Data as a Graph

1. Schema Graph:
   - describes the schema of the data
   - meta-level representation of the data
   - constraints the edges that are permissible in the data graph
   - general construction: the tables in the database form the nodes; edges capture some relationship or constraint between the corresponding relations
2. Data Graph:
   - instantiation of its schema graph
   - contains actual data which is split across different nodes and edges
   - general construction: the tuples of the database form the nodes; cross-references like foreign key references, inclusion dependencies, etc., form the edges of the graph
   - nodes can be set according to the granularity required - table, tuple or cell
3. Concept of Node weight and Edge weight

Keyword Query System Model I

1. Data Model:
   - describes the high-level representation of the data in the system
   - reflects the constraints, associations, and organization of the data
   - graph model
2. Query Model:
   - specifies the structure of the input that can be given to the system
   - keyword queries - set of words
   - graph, tree patterns - the user can specify constraints which the answer must satisfy
3. Answer Model:
   - specifies, what an answer to a query is
   - specifies the structure, requirements that it must satisfy according to the semantics of the system
   - common form of representation: graph, tree, tuple, term

Keyword Query System Model II

4. Scoring Model:
   - assigns a score to the answers, based on their relevance
   - notion of relevance - ambigous; returns top scoring answers
   - a simple scheme: higher score to an answer with smaller number of joins
   - most systems use complex rules to assign scores, to improve the quality of the top ranked answers

Object Rank System I

- adapts the notion of PageRank to suit the database setting
- concept of authority: nodes having query terms have authority
- nodes transfer authority to neighbours in a fixed manner
- final score given by the accumulated authority

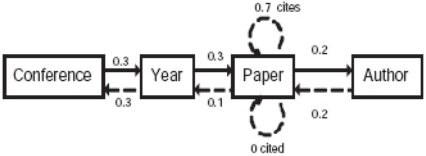
Graph Representation

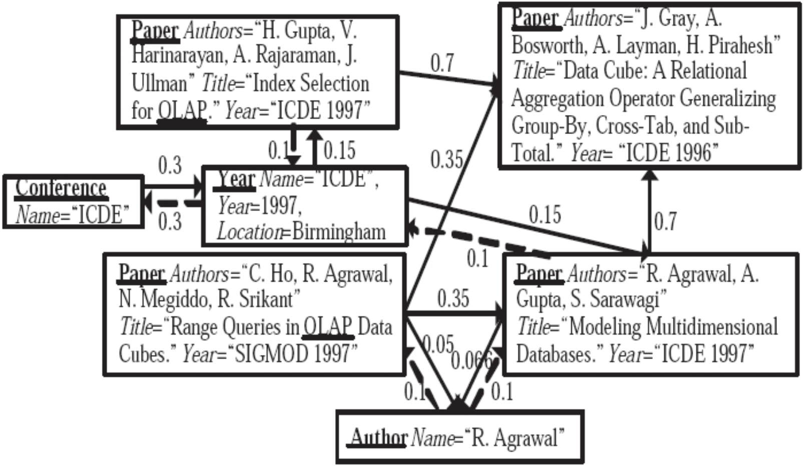
1. Data graph - labelled graph $D(V_D, E_D)$
2. Schema graph - directed graph $G(V_G, E_G)$
3. Authority Transfer Schema graph $G^A(V_G, E^A)$
   - for each edge $e_G = (u, v)$ in the schema graph, insert two authority transfer edges:
     1. forward edge $e_G^f = (u, v)$ with authority transfer rate: $\alpha(e_G^f)$
     2. backward edge $e_G^b = (v, u)$ with authority transfer rate: $\alpha(e_G^b)$
   - intuition: authority could flow in both directions at different rates

Object Rank System II

4. Authority Transfer Data Graph $D^A(V_D, E_D^A)$
   - for every edge $e = (u, v) \in E_D$, add two edges $e^f = (u, v)$ with authority transfer rate $\alpha(e^f)$ and $e^b = (v, u)$ with authority transfer rate $\alpha(e^b)$
   - e^f be of type e_G^f
   - $OutDeg(u, e_G^f)$ - number of outgoing edges from u of type e_G^f
   - authority transfer rate $\alpha(e^f)$ is defined as:

$$\alpha(e^f) = \begin{cases} \frac{\alpha(e_G^f)}{OutDegree(u, e_G^f)} & ifOutDegree(u, e_G^f) > 0 \\ 0 & ifOutDegree(u, e_G^f) = 0 \end{cases}$$


0.7 cites
Conference
0.3
0.3
Year
0.3
0.1
Paper
0.2
0.2
Author
0 cited


Paper Authors="H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman" Title="Index Selection for OLAP." Year="ICDE 1997"
Paper Authors="J. Gray, A. Bosworth, A. Layman, H. Pirahesh" Title="Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total." Year= "ICDE 1996"
0.7
0.1
0.15
0.35
0.3
Conference Name="ICDE"
0.3
Year Name="ICDE", Year=1997, Location=Birmingham
0.15
0.7
0.1
Paper Authors="C. Ho, R. Agrawal, N. Megiddo, R. Srikant" Title="Range Queries in OLAP Data Cubes." Year="SIGMOD 1997"
0.35
Paper Authors="R. Agrawal, A. Gupta, S. Sarawagi" Title="Modeling Multidimensional Databases." Year="ICDE 1997"
0.05
0.066
0.1
0.1
Author Name="R. Agrawal"

Object Rank System - Random Surfer Model for Ranking

- initially, large number of random surfers start from objects containing the specified keyword; they traverse the database graph along the edges
- at any point of time, a random surfer at a node does one of the following:
  - move to an adjacent node by moving along an edge
  - jump to a randomly chosen node containing the keyword
- **ObjectRank of a node:** expected percentage of surfers at that node, as time goes to infinity

Keyword-Specific and Global ObjectRanks I

Keyword-Specific ObjectRank

- gives the relevance with respect to a keyword
- w - keyword; $S(w)$ - keyword base set - set of objects that contain w
- $r^w(v_i)$ of node v_i obtained as the solution to:

$$\mathbf{r}^w = d\mathbf{A}\mathbf{r}^w + \frac{(1-d)}{|S(w)|}\mathbf{s}$$

  - $A_{ij} = \alpha(e)$ if there is an edge $e = (v_j, v_i)$ in E_D^A; 0 otherwise
  - $\mathbf{s} = [s_1, ..., s_n]^T$ - base set vector; $s_i = 1$ if $v_i \in S(w)$; 0 otherwise
  - d - damping factor

Global ObjectRank

- gives the general importance regardless of the query
- calculated from the above equation, but with **all** nodes included in the base set

Keyword-Specific and Global ObjectRanks II

Combined ObjectRank

- $r^G(v)$ - Global ObjectRank of v
- $r^w(v)$ - Keyword-specific ObjectRank of v w.r.t w

Combined Rank

$$r^{w,G}(v) = r^w(v).(r^G(v))^g$$

g - Global ObjectRank weight

Multiple-Keyword Queries

extending the random surfer model

- multiple-keyword query : $w_1, \ldots, w_m$
- m independent random surfers, where the i^{th} surfer starts from the keyword base set $S(w_i)$
- **AND** semantics: probability that the m random surfers are simultaneously at node v

$$r_{AND}^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} r^{w_i}(v)$$

- **OR** semantics: probability that atleast one of them is at node v

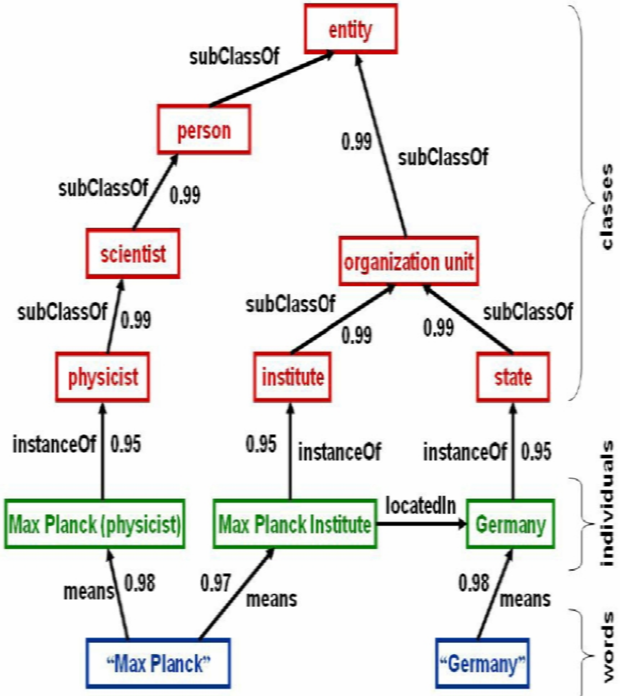
$$r_{OR}^{w_1,\ldots,w_m}(v) = \sum_{i=1,\ldots,m} r^{w_i}(v)$$

The NAGA System

- semantic search engine

Data Model :

- Knowledge graph: directed, weighted, labeled multi-graph $G = (V, E, L_V, L_E)$
- facts: binary relationships derived from the web
- represented as an edge together with its end nodes e.g. $e(u, v)$, $l(u) = MaxPlanck(physicist)$, $l(e) = bornInYear$, $l(v) = 1858$
- witnesses of a fact: the pages from which it has been extracted

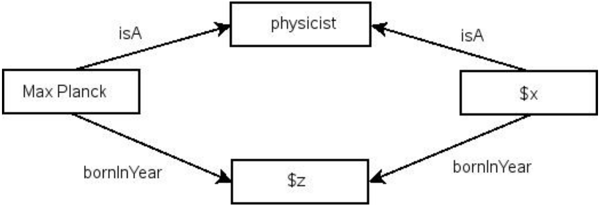

entity
subClassOf
person
0.99
subClassOf
subClassOf
0.99
scientist
organization unit
subClassOf
0.99
subClassOf
subClassOf
0.99
0.99
subClassOf
physicist
institute
state
instanceOf
0.95
0.95
instanceOf
instanceOf
0.95
Max Planck (physicist)
Max Planck Institute
locatedIn
Germany
means
0.98
0.97
means
0.98
means
"Max Planck"
"Germany"
classes
individuals
words

NAGA - Graph Pattern Query Model I

- connected, directed graph
- nodes, edges can be labeled with variables or constants
- fact template: edge label and the two node labels. e.g. `AlbertEinstein friendOf $x`
- answer - subgraph of the data graph, that has valid objects which can take the place of the variables and also satisfy the edge constraints

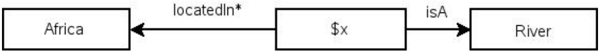
Queries supported:

1. Discovery query: to discover pieces of information
   e.g. to find physicists who were born in the same year as Max Planck:

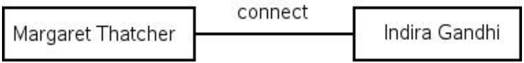


NAGA - Graph Pattern Query Model II

2. Regular expression query: to find out some particular path connecting pieces of information
   e.g. to find out the rivers located in Africa:



3. Relatedness query: to find out a broad relationship between pieces of information
   e.g. How are Margaret Thatcher and Indira Gandhi related?



NAGA - Answer Model I

- matching path: e.g. `Nile locatedIn Egypt, Egypt locatedIn Africa` is a valid match for `$x locatedIn* Africa`
- Answer Graph- subgraph of the knowledge graph such that:
  - for each fact template in the query, there is a matching path
  - each fact in the answer is part of only one matching path
  - each vertex of the query is bound to exactly one vertex of answer
- for query $q = q_1 q_2 \ldots q_n$, find subgraph g for which $P(g|q)$ is the highest

NAGA - Answer Model II

confidence value of a fact

$$P_{conf}(f) = \frac{1}{n}\sum_{i=1}^{n} acc(f, p_i).tr(p_i)$$

- p_i : witnesses of f
- $acc(f, p)$: estimated accuracy with which f was extracted from p
- $tr(p)$: trust in p - computed by an algorithm similar to PageRank

informativeness of a fact

- $P_{finfo}(f)$ - depends on number of witnesses, query
  e.g. query:`AlbertEinstein isA $x` - `AlbertEinstein isA physicist` ranked higher than `AlbertEinstein isA politician`
  $\frac{|W(\texttt{AlbertEinstein isA physicist})|}{\sum_{\$x} |W(\texttt{AlbertEinstein isA \$x})|}$
  query: `$x isA physicist`
  $\frac{|W(\texttt{AlbertEinstein isA physicist})|}{\sum_{\$x} |W(\texttt{\$x isA physicist})|}$

NAGA - Answer Model III

confidence and informativeness of query q_i

$$P_{conf}(q_i|g) = \prod_{f \in match(q_i,g)} P_{conf}(f)$$
$$P_{info}(q_i|g) = \prod_{f \in match(q_i,g)} P_{finfo}(f|q_i)$$

probability of the query being generated by g

$$\tilde{P}(q_i|g) = \beta P_{conf}(q_i|g) + (1-\beta)P_{info}(q_i|g)$$
$$P(q_i|g) = \alpha\tilde{P}(q_i|g) + (1-\alpha)\tilde{P}(q_i)$$

where, $\tilde{P}(q_i)$ gives different weights to fact templates

estimate probability of an answer graph, given the query

$$P(g|q) \sim P(q|g)P(g)$$

where, $P(q|g) = \prod_{i=1}^{n} P(q_i|g)$

NAGA - Scoring Model

Scoring model captures the following:

1. Confidence:
   - certainity about a specific fact
   - independent of the query and the popularity of the fact
   - facts extracted from authoritative pages, with high accuracy, will be given a higher score
2. Informativeness:
   - relevance of a fact for a given query
   - dependent on the formulation of the query
   - fact deemed to be relevant if it is highly visible in the web
   - intuition: the more the number of pages that state the fact, the higher is the likelihood that the fact is true and is important
3. Compactness of the resulting graph:
   - implicitly captured by the likelihood of the graph given the query
   - likelihood is the product over the probabilities of its component facts

Conclusion

- Other systems studied: System by Goldman et. al. for search incorporating the notion of proximity, DBXplorer, DISCOVER, BANKS, System by Hristidis et. al. for IR style Keyword search, Proximity Search in Type-Annotated Corpora and FleXPath
- Keyword Searching is an important paradigm for searching in databases
- methods of querying: set of words, graph/tree patterns
- answer models: from rows in the database, to trees and graphs
- different semantics: OR, AND, proximity
- scoring models: number of joins, complex combinations of node and edge scores, concept of authority, probabilities etc.
- future work:
  - oriented towards incorporating more semantics into the search systems
  - alternate structure for answers which will make it more intuitive
  - fine tuning of the scoring model, based on feedback from the user - instead of having a static function

References I

[1] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. *ICDE, 2002*.
[2] Sihem Amer-Yahia, Laks V.S. Lakshmanan, and Shashank Pandit. FleXPath: Flexible Structure and FullText Querying for XML. *SIGMOD, 2004*.
[3] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. *ICDE, 2002*.
[4] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. ObjectRank: Authority-Based Keyword Search in Databases. *VLDB Conference, 2004*.
[5] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *WWW Conference, 1998*.
[6] Soumen Chakrabarti, Kriti Puniyani, and Sujatha Das. Optimizing Scoring Functions and Indexes for Proximity Search in Type-annotated Corpora. *DBLP Conference, pages 717726, 2006*.

References II

[7] Roy Goldman, Narayanan Shivakumar, Suresh Venkatasubramanian, and Hector Garcia-Molina. Proximity Search in Databases. *VLDB Conference, 1998*.

[8] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. *VLDB Conference, 2003*.

[9] Vagelis Hristidis and Yannis Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. *VLDB Conference, 2002*.

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

[11] Georgia Koutrika, Alkis Simitsis, and Yannis Ioannidis. Précis: The Essence of a Query Answer. *ICDE, 2006*.

[12] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. NAGA: Searching and Ranking Knowledge. *ICDE, 2008*.

DBXplorer

- Answer: row that contains all keywords
- rows may be either from single tables, or by joining tables connected by foreign-key relationships
- ranking of rows - by the number of joins involved

DISCOVER

- Answer: Minimal Total Joining Networks of Tuples (MTJNT)
- MTJNT - Joining Network of Tuples that satisfy Totality and Minimality requirements
- Joining Network of Tuples j is a tree of tuples where for each pair of adjacent tuples $t_i, t_j \in j$, where $t_i \in R_i, t_j \in R_j$, there is an edge (R_i, R_j) in the schema graph and $(t_i \bowtie t_j) \in (R_i \bowtie R_j)$
- Total: answer graph should contain ALL the words in the query
- Minimal: if any node is removed from the answer graph, then either, it becomes disconnected or it is no longer total
- ranking of rows - by the number of joins involved

IR style Keyword search by Hristidis et. al.

- idea: use the underlying RDBMS, to efficiently process a keyword query. incorporates IR techniques of proximity, in answering keyword queries on a database. Contemporary RDBMS possess efficient querying capabilities for text attributes, but
- data, query model - same as that in DISCOVER
- Scoring model:
  - for each textual attribute a_i in T, the joining tree of tuples, find single-attribute score using the IR engine employed in the underlying database
  - final score: combination of single-attribute scores using *Combine* $Combine(Score(A, Q), size(T)) = \frac{\sum_{a_i \in A} Score(a_i, Q)}{size(T)}$
- AND semantics: 0 score for tuple trees that don't have **all** keywords; else, score given by *Combine* function
- OR semantics: score given by the *Combine* function

The BANKS System I

Data Graph - tuples: nodes and edges: foreign key - primary key relationships

Answer Model

- connection tree - a directed rooted tree containing **all** the keywords
- keywords nodes form the leaves of the tree
- root node - the information node; is a common vertex from where there exists path to all the keyword nodes

Scoring Model

- overall relevance score of an answer tree:
  - additive combination: $(\mathbf{1} - \lambda)\mathbf{Escore} + \lambda\mathbf{Nscore}$
  - multiplicative combination: $\mathbf{Escore} \times \mathbf{Nscore}^{\lambda}$

  λ - controls relative weightage
- *Nscore* of a tree : average of node scores of (i) leaf nodes (ii) root node

The BANKS System II

- *Escore* of a tree : $1/(1+\sum_{e} Escore(e))$, where $Escore(e)$ - normalized score of individual edges
- gives lower relevance to larger trees

Bidirectional Search : Scoring Model

- $s(T, t_i)$ - score of answer tree T with respect to keyword t_i: defined as the sum of the edge weights on the path from the root of T to the leaf containing t_i
- aggregate edge-score E of T: $\sum_i s(T, t_i)$.
- tree node prestige N: sum of the node prestiges of the leaf nodes and the answer root
- Prestige: computed by a biased random walk, where, the probability of moving along a particular edge is inversely proportional to its edge weight
- overall tree score: $\mathbf{EN}^{\lambda}$
  λ controls relative weightage

Search incorporating the notion of proximity by Goldman et. al.

- proximity measured as the shortest distance between nodes
- query model: pair of queries
  Find Query:
  - specifies the type of the answer e.g. objects of type `movie`
  - defines *FindSet*: set of objects that can potentially be the answer

  Near Query: specifies the keywords that define a *NearSet*.
- idea: rank *FindSet* objects based on proximity to *NearSet* objects
- *bond* between *FindSet* object f and *NearSet* object n:
  $b(f, n) = \frac{r_F(f) r_N(n)}{d(f,n)^t}$
  - $r_F(f)$ - ranking of f in *FindSet*, F; $r_N(n)$ - ranking of n in *NearSet*, N
  - $d(f, n)$ - distance between f and n
  - t - tuning component
- Scoring model:
  - Additive : $score(f) = \sum_{n \in N} b(f, n)$
  - Maximum : $score(f) = max_{n \in N} b(f, n)$
  - Beliefs : $score(f) = 1 - \prod_{n \in N} (1 - b(f, n))$

Proximity Search in Type-Annotated Corpora

- query model: `type=atype` `NEAR` $S_1 S_2 \ldots S_k$
- candidate answer token: any token connected to a descendant of `atype`
- nearness is a function of:
  - matching selectors
  - frequency of selectors in the corpus
  - distance of selectors from the candidate answer
- scoring model:
  - $\mathit{energy}(s)$: similar to inverse document frequency (IDF)
  - $\mathit{gap}(w, s)$: number of tokens present between a candidate token and a matched selector
  - energy received: $\mathit{energy}(s)\mathit{decay}(\mathit{gap}(w, s))$, where $\mathit{decay}(g)$ is a function of the gap
  - decay function is automatically learned - found that its not monotonically decreasing with gap, as was expected
  - score of a candidate a:
    $\mathit{score}(a) = \oplus_s \oslash_i \mathit{energy}(s_i)\mathit{decay}(\mathit{gap}(s_i, a))$
    s_i: multiple occurrences of s near a

FleXPath I

- query model - *tree pattern query* (TPQ) (T, F):
  - T: rooted tree with nodes denoting variables; edges denoting structural predicates - parent-child (pc), ancestor-descendant (ad) relationships
  - F: predicate expression - specifies constraints on the contents of the nodes
  - distinguished node: usually, the root node; designated as the answer
- query relaxation:
  - replacing `parent-child` by `ancestor-descendant` predicate
  - dropping an `ancestor-descendant` constraint
  - promoting a `contains` predicate to the parent
- Predicate Penalty: measures the extend of the loss of context, when a predicate is dropped to get the relaxed query
  $penaltyOfDropping(pc(\$i, \$j)) = \frac{\#_{pc}(t_i,t_j)}{\#_{ad}(t_i,t_j)} w_Q(pc(\$i, \$j))$
  where, $w_Q(p)$ - weight of the predicate - measure of its importance

FleXPath II

- score of an answer- *ss*: structural score; *ks*:keyword score
- $ss = \sum_{p \in P} w_Q(p) - \sum_{p \in S} \pi(p)$
  - P: set of all predicates in the original query, Q
  - S: set of predicates that have been dropped from P to obtain relaxed version
  - $\pi(p)$: penalty incurred for dropping predicate p
- final score:
  - structure first: (ss, ks)
  - keyword first: (ks, ss)
  - arithmetic function that combines *ks* and *ss*