CMSC 828E: Proximity Search in Graphs

Amol Deshpande

University of Maryland, College Park

October 13, 2010

DBXplorer

- DBXplorer: A System for Keyword-Based Search over Relational Databases; ICDE 2002
- Takes a relational approach to the problem
- Step 1: Build a symbol table that associates keywords with either
  - columns in the database tables
  - cells (row-column pairs) in the database tables
- Step 2: Given the set of keywords, identify the tables that contain at least one keyword

DBXplorer

- Step 3: Create a list of join "trees" such that all results will be enumerated

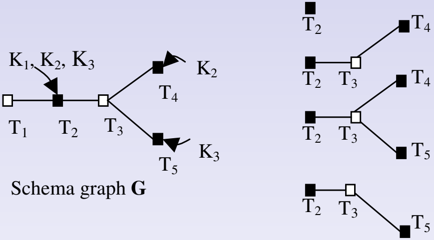

Figure 4. Join trees

- Step 4: Add appropriate predicates based on the keywords
  - e.g., column = "keyword"
- Step 5: Issue join queries to the database
  - Only step that actually touches the data

DBXplorer

- One of the first papers on keyword search in databases
  - Goldman et al.'s work (VLDB 1998) precedes, but focuses on graph data
- Efficiency likely to be a big concern
  - Too many queries issued
  - Too much repeated work since queries issued to the database are quite similar
  - No easy way to force Top-K

BANKS

- Keyword Searching and Browsing in Databases using BANKS; ICDE 2002
- Why keyword search ?
  - More intuitive
  - Information spread throughout a relational database
- BANKS allowed searching and browsing without any knowledge of schema
- Why "directionality is important" ?
  - "Ignoring directionality would cause problems because of "hubs" which are connected to a large numbers of nodes. For example, in a university database a department with a large number of faculty and students would act as a hub. As a result, many nodes would be within a short distance of many other nodes, reducing the effectiveness of proximity-based scoring."

BANKS

- Use "directionality" and "weights" so answers are ranked properly
  - Somehow penalize use of non-informative links
  - e.g., imagine two people that are members of: "Facebook" and "UMD CS Dept"
  - Should give higher weight to edges involving latter
- BANKS had a formal mechanism/rules for assigning weights based on in-degree/out-degree
- Paper also discusses how to combine different weights into a single relevance score
  - Somewhat ad hoc, but not really much choice
  - Can try to "learn" from user preferences
    - An interesting problem that may have been looked at

BANKS

- Backward Expanding Search Algorithm
  - Heuristic search – assume graph is in-memory
  - The goal to find a rooted directed tree so that there is a directed path from root to each keyword
  - Find all nodes that match one of the keywords
  - Run "backward" (using incoming directed edges) shortest-path searches on all nodes simultaneously
  - Find nodes that are common in all of them
- Answers approximately sorted according to the relevance score, but still need to do more processing
- BANKS also had a strong visualization component
- They may miss some answers (see discussion in Goldenberg et al.)

BANKS

- Bidirectional Expansion For Keyword Search on Graph Databases; VLDB 2005
- A followon paper from the same group
- Better "Bidirectional" search algorithm
- Imagine a query: "Gray", "transaction"
  - Many nodes in the tree match the second term
  - Starting backward from all nodes may be inefficient
  - Better: Go backwards from all nodes that match "Gray", and then for each of the visited nodes, go "forward" to look for "transaction"
- Many details to get this right
- But able to handle graphs with 20 million nodes on a desktop PC

DISCOVER

- DISCOVER: Keyword Search in Relational Databases; VLDB 2002
- Similar to DBExplorer, but can handle some additional things
  - Shares computation among different queries issues to the database
  - Can have multiple tuples from the same relation in the answer
- Uses an Oracle keyword index for the first step of identifying all rows that contain keywords
- NO RANKING
  - As with DBExplorer, all results are output

XML Graphs

- Several works
- Keyword Proximity Search on XML Graphs; ICDE 2003
  - Follow-up work extending DISCOVER
- XRank; SIGMOD 2003
  - Ranking function more involved
  - Incorporates some aspect of "PageRank" (i.e., inherent importance of some nodes)

Kimelfeld et al. [PODS 2006]

- Finding and Approximating Top-k Answers in Keyword Proximity Search; PODS 2006
- Followon paper in SIGMOD 2008 (assigned reading)
- Theoretical work
  - Can we get some bounds/provably optimal algorithms ?
- Keyword search in graphs == Identifying Steiner trees
  - Steiner trees NP-Hard to compute, but not if # of terminals is fixed
  - If the # of terminals is k, then trivial n^k algorithm
- We also want to do "ranking" at the same time

Kimelfeld et al. [PODS 2006]

- Question: How do you define "approximation" here ?
  - We don't want to stray too far from the true ranking
  - θ-approximation of a top-k answer: A selected answer is within a factor θ of a non-chosen answer
- Polynomial delay: The time taken to return the "next" answer is bounded
- Show how to use exact and approximate Steiner tree algorithms to get approximations for the keyword search
- NOTE: No indexing here – everything is polynomial in the size of the graph
  - With indexing, the goal would be to go sub-linear

Kimelfeld et al. [PODS 2006]

- Enumeration algorithm
  - Based on early work by Lawler
  - Find the top answer (in this case, the top Steiner tree)
  - Partition the remaining answers using constraints
  - Find the top answer in each partition and insert in a priority queue
  - Repeat by choosing the top answer in the priority queue
- As far as I can tell, a systematic way to do enumerate in ranked order, but not much deeper than that
- Adapting this to proximity search not straightforward, but can be done

Goldenberg et al. [SIGMOD 2008]

- A practical approach based on the prior work [Kimelfeld et al.; PODS 2006]
- Incorporates a notion of "diversity" (without calling it that)
  - Based on repeated information
  - Somewhat tricky definition and not easy to optimize
- θ-approximate answer: if one answer precedes another (in enumeration), it is worse by a factor of at most θ
- Proposed algorithm achieve 2-approximation w.r.t. "height"

Goldenberg et al. [SIGMOD 2008]

- Algorithmically: combines the shortest-path iterator approach of BANKS with Lawler's method used in Kimelfeld et al.
- Solving the first instance is easy
  - i.e., finding the lowest-height tree connecting the keywords (BANKS does that)
- Solving after that is tricky
  - In particular, incorporating the "inclusion" constraints

More recent work

- STAR: Steiner-Tree Approximation in Relationship Graphs; ICDE 2009
    - Improved Steiner tree finding algorithms with approximation guarantees
    - Uses "taxonomy" information to more quickly find initial trees
- EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data; SIGMOD 2008
    - Uses both inverted indexes and subgraph indexes
- Several other works focusing on efficiency etc.
    - See "Related Work" in the EASE paper

More recent work

- BLINKS: ranked keyword searches on graphs; ICDE 2007
  - Provable performance bounds, based on dynamic programming
- Toward Industrial-Strength Keyword Search Systems over Relational Data; ICDE 2010
  - Considers placing absolute limits on the amount of time taken to return answers
- Language-model-based Ranking for Queries on RDF-Graphs; CIKM 09
  - RDF Graphs; Language-model based ranking