Indexing Uncertain Categorical Data

Sarvejith Singh etc.
Presented by: Yi Zhang

The problem

- Uncertainty in discrete-domain data
  - E.g. Nurse 10 in Room {3,5} at 10:05
  - Results from text classifiers
- Existing relational DB doesn’t handle well
  - Allowing multiple values? app use own complex model to manage uncertainty
  - Just keep most likely value? lossy
  - Need proper index structure for categorical uncertain data

Data Model

- UDA: uncertain discrete attribute
  - A distribution over its discrete domain
  - Example
  - UDA u can be represented as prob. vector \( \langle p_1, \ldots, p_N \rangle \) s.t. \( \Pr(u=d_i) = p_i \)
  - For UDAs u and v

Distribution Similarity Metrics

- Given two UDAs, how similar are they?
- Useful for clustering in index structures
- Manhattan distance
  - \( L_1(u, v) = \sum_{i=1}^N |u.p_i - v.p_i| \)
- Euclidean distance
  - \( L_2(u, v) = \sqrt{\sum_{i=1}^N (u.p_i - v.p_i)^2} \)
- Kullback-Leibler divergence
  - \( KL(u, v) = \sum_{i=1}^N u.p_i \log(u.p_i/v.p_i) \)

Queries

- Equality queries
  - PEQ: given UDA q, return all tuples t from R s.t. \( \Pr(q=t.a) \geq \tau \)
  - PETQ: same as PEQ, except \( \Pr(q=t.a) \geq \tau \) (constant)
  - PEQ-top-k: return k tuples w/ highest equality prob.
- Distributional queries
  - DSTQ: given UDA q, divergence function f, return all tuples t from R s.t. \( f(q,t.a) \leq \tau \)
  - DSCQ-top-k
- Equality join queries
  - PEJ: \( f(q,t.a) \leq \tau \) s consists all pairs of tuples t, s from R, S s.t. \( \Pr(r.a=s.b) \geq \tau \)
  - PEJ-top-k
DB Inverted Index

- Outer list: for each domain value di
  - Inner list: list of <t,p> where t.di=p
- Rank aggregation...
- Simple insertion/deletion
- What about searching?

DB Search on Inverted Index

- PETQ queries: given UDA q and const τ
- Inv-index-search
  - I/O significant
- Highest-prob-first
- Row pruning
- Column pruning
  - Needs random I/O

DB PDR-tree

- Consider UDA as point in high-dim space
- Similar UDAs stored in a page
- Page described by MBR
- Parent’s MBR covers all its children

DB Operations on PDR-trees

- Insert(u)
  - To insert into current page, update MBR; then choose best child page and insert.
  - Min area increase
  - Most similar MBR
  - Start from root and recurse
- Split()
  - Top-down: pick two children MBRs as cluster centers and all other UDAs go to a closer cluster
  - Bottom-up: each UDA forms an independent cluster and proceed by merging closest pair of clusters until two remains

DB Queries on PDR-trees

- PETQ(q, τ)
  - Depth-first search
  - Enters an node c if qualifies c.mbr.q ≥ τ; prune otherwise
  - At leaf level, check each UDA and output
  - For top-k queries, update threshold dynamically during search
  - Doesn’t work with distribution similarity queries...

DB Experimental Setup

- Datasets
  - Real
    - CRM databases, 100k test doc, 50 categories
    - CRM2: prob values generated by auto categorization
    - CRM3: unsupervised fuzzy clustering
  - Synthetic
    - Uniform: 5 categories, prob. random chosen, 10k tuples
    - Pairwise: 5 categories, each tuple has 2 non-zero categories w/ roughly equal prob., 10k tuples
    - Gen4: for each tuple, size of non-zero categories follows geometric dist.
- Page size 8KB, 100 blocks for buffering
- PETQ and PEQ-top-k queries
- Measure # of I/O operations

DB Inverted Index

- Outer list: for each domain value di
- Inner list: list of <t,p> where t.di=p
- Rank aggregation...
- Simple insertion/deletion
- What about searching?

DB Search on Inverted Index

- PETQ queries: given UDA q and const τ
- Inv-index-search
  - I/O significant
- Highest-prob-first
- Row pruning
- Column pruning
  - Needs random I/O

DB PDR-tree

- Consider UDA as point in high-dim space
- Similar UDAs stored in a page
- Page described by MBR
- Parent’s MBR covers all its children

DB Operations on PDR-trees

- Insert(u)
  - To insert into current page, update MBR; then choose best child page and insert.
  - Min area increase
  - Most similar MBR
  - Start from root and recurse
- Split()
  - Top-down: pick two children MBRs as cluster centers and all other UDAs go to a closer cluster
  - Bottom-up: each UDA forms an independent cluster and proceed by merging closest pair of clusters until two remains

DB Queries on PDR-trees

- PETQ(q, τ)
  - Depth-first search
  - Enters an node c if qualifies c.mbr.q ≥ τ; prune otherwise
  - At leaf level, check each UDA and output
  - For top-k queries, update threshold dynamically during search
  - Doesn’t work with distribution similarity queries...

DB Experimental Setup

- Datasets
  - Real
    - CRM databases, 100k test doc, 50 categories
    - CRM2: prob values generated by auto categorization
    - CRM3: unsupervised fuzzy clustering
  - Synthetic
    - Uniform: 5 categories, prob. random chosen, 10k tuples
    - Pairwise: 5 categories, each tuple has 2 non-zero categories w/ roughly equal prob., 10k tuples
    - Gen4: for each tuple, size of non-zero categories follows geometric dist.
- Page size 8KB, 100 blocks for buffering
- PETQ and PEQ-top-k queries
- Measure # of I/O operations
KL: if hit an MBR, most UDAs will qualify cause they are similar in dist.

Top-k: explore more tuples

Non-zero probability in many categories
Large number of lists accessed in inverted index

Inverted Index
- Reduction in avg length of each list as # of lists increase

PDR-tree
- Data generation related
- Non-zero entries at both ends of experimental space smaller than in the middle

How to use the mentioned criteria to insert a UDA into a PDR-tree

No motivation for MBR-based approach used in PDR-Tree

Not obvious how to extend PDR-Tree to work for Distributional Similarity Queries