Automatic Categorization of Query Results

SIGMOD '04

F. Kaushik Chakrabarti [1] S. Surajit Chaudhuri [1] T. Seung-won Hwang[2]

[1]Microsoft Research
[2]Univ. of Illinois, Urbana Champaign

February 22, 2008

MOTIVATION

- *Exploratory queries* are increasingly becoming a common phenomenon in database systems.
  - e.g. search for a book on a given *subject* on Amazon.com
- These queries return *too-many results*, but only a small fraction is relevant
  - the user ends up examining all or most of the result tuples to find the interesting ones.
- Can happen when the user is unsure about what is *relevant*
  - e.g.user shopping for a home is often unsure of the exact neighborhood, price range . . .

This phenomenon is commonly referred to as **information-overload**

Motivation

- *Exploratory queries* are increasingly becoming a common phenomenon in database systems.
  - e.g. search for a book on a given *subject* on Amazon.com
- These queries return *too-many results*, but only a small fraction is relevant
  - the user ends up examining all or most of the result tuples to find the interesting ones.
- Can happen when the user is unsure about what is *relevant*
  - e.g.user shopping for a home is often unsure of the exact neighborhood, price range . . .

This phenomenon is commonly referred to as **information-overload**

MOTIVATION

- *Exploratory queries* are increasingly becoming a common phenomenon in database systems.
  - e.g. search for a book on a given *subject* on Amazon.com
- These queries return *too-many results*, but only a small fraction is relevant
  - the user ends up examining all or most of the result tuples to find the interesting ones.
- Can happen when the user is unsure about what is *relevant*
  - e.g.user shopping for a home is often unsure of the exact neighborhood, price range ...

This phenomenon is commonly referred to as **information-overload**

MOTIVATION

- *Exploratory queries* are increasingly becoming a common phenomenon in database systems.
  - e.g. search for a book on a given *subject* on Amazon.com
- These queries return *too-many results*, but only a small fraction is relevant
  - the user ends up examining all or most of the result tuples to find the interesting ones.
- Can happen when the user is unsure about what is *relevant*
  - e.g.user shopping for a home is often unsure of the exact neighborhood, price range . . .

This phenomenon is commonly referred to as **information-overload**

Common approaches to avoid information-overload

from the IR scenario

- Ranking
- Categorization

Common approaches to avoid information-overload

from the IR scenario

- Ranking
- Categorization

CATEGORIZATION IN DATABASE SYSTEMS

- Category structures are decided in advance.
- Categories of a result tuple is decided in advance.
  - Examples: Amazon, Walmart, e-Bay . . .
- Problem: Susceptibility to *skew* - defeats the purpose of categorization
  User still experiences **information-overload**.

CATEGORIZATION IN DATABASE SYSTEMS

- Category structures are decided in advance.
- Categories of a result tuple is decided in advance.
  - Examples: Amazon, Walmart, e-Bay . . .
- Problem: Susceptibility to *skew* - defeats the purpose of categorization

  User still experiences **information-overload**.

Categorization in database systems

- Category structures are decided in advance.
- Categories of a result tuple is decided in advance.
  - Examples: Amazon, Walmart, e-Bay . . .
- Problem: Susceptibility to *skew* - defeats the purpose of categorization
  User still experiences **information-overload**.

Automatic Categorization of Query Results

based on query results

- Previous categorization techniques were query *independent* - the category structure were decided *apriori*.
- Solution: Generate the category structure based on the *contents of tuples* in the *answerset*
- Ensure "even" distribution of query results across the category

Automatic Categorization of Query Results

based on query results

- Previous categorization techniques were query *independent* - the category structure were decided *apriori*.
- Solution: Generate the category structure based on the *contents of tuples* in the *answerset*
- Ensure "even" distribution of query results across the category

AUTOMATIC CATEGORIZATION OF QUERY RESULTS

based on query results

- Previous categorization techniques were query *independent* - the category structure were decided *apriori*.
- Solution: Generate the category structure based on the *contents of tuples* in the *answerset*
- Ensure "even" distribution of query results across the category

Automatic Categorization of Query Results

Example:

Example of hierarchical categorization

TABLE OF CONTENTS

Categorization Model

Space of Categorization

- A hierarchical categorization of $\boldsymbol{R}$ is a recursive partitioning of the tuples in $\boldsymbol{R}$ defined inductively as follows:
  - **Base Case:** Given a `ALL` node containing all tuples in $\boldsymbol{R}$, partition $\boldsymbol{R}$ using a single attribute.
  - **Inductive Step:** Given a node $\boldsymbol{C}$ at level `l - 1`, partition (level `l`) set of tuples `tset(`$\boldsymbol{C}$`)` using a single attribute for all nodes in for all nodes at level `l - 1` *iff* C contains more than a “certain” number of tuples.
- Associated with each category $\boldsymbol{C}$ is:
  - `tset(`$\boldsymbol{C}$`)` : Set of tuples contained in a category C.
  - `label(`$\boldsymbol{C}$`)` :
    - For categorical attribute `A` is of the form $A \in B$ where $B \subset dom_R(A)$
    - For numeric attribute `A` is of the form $a_1 \leq A \leq B_2$ where $a_1, a_2 \in dom_R(A)$.

Categorization Model

Exploration Model

To generate a particular instance of hierarchical categorization:
At each level `l`:

- Determine the categorizing attribute `A` for level `l`
- Determine the partition of domain of values of `A` for `tset(C)`

Objective: Choose the attribute-partition combination at each level such that the resulting instance T_{opt} has least possible information overload on the user.

Categorization Model

Exploration Model

To generate a particular instance of hierarchical categorization:
At each level `l`:

- Determine the categorizing attribute `A` for level `l`
- Determine the partition of domain of values of `A` for `tset(C)`

Objective: Choose the attribute-partition combination at each level such that the resulting instance T_{opt} has least possible information overload on the user.

Categorization Model

Exploration Model : scenarios

Common exploration scenarios:

- ALL User explores the result set R until she finds *every* tuple $t \in R$ relevant to her.
- ONE User explores the result set R until she finds *one (or few)* tuple(s)

Categorization Model

Exploration Model : scenarios

Common exploration scenarios:

- ALL User explores the result set R until she finds *every* tuple $t \in R$ relevant to her.
- ONE User explores the result set R until she finds *one (or few)* tuple(s)

Categorization Model

Exploration Model : ALL

Model of exploration of node C in `ALL` scenario:

Algorithm 1 Explore C

```
if C is a non-leaf node then
    Choose one of the following:
    (1) Examine all tuples in tset(C) {Option SHOWTUPLES}
    (2) {Option SHOWCAT}
    for i = 1; i ≤ n; i++ do
        Examine the label of ith subcategory
        Choose one of the following
        (2.1) Explore C_i
        (2.2) Ignore C_i
    end for
else
    Examine all tuples in tset(C)
end if
```

Categorization Model

Exploration Model : ALL

Model of exploration of node C in `ONE` scenario:

Algorithm 2 Explore C

if C is a non-leaf node **then**
  Choose one of the following:
  (1) Examine tuples in `tset(C)` till the first relevant tuple found {Option SHOWTUPLES}
  (2){Option SHOWCAT}
  for $(i = 1; i \leq n; i++)$ **do**
    Examine the label of ith subcategory
    Choose one of the following
    (2.1) Explore C_i
    (2.2) Ignore C_i
    if choice = Explore **then**
      break
    end if
  end for
else
  Examine tuples in `tset(C)` till the first relevant tuple found
end if

Cost Model

- Define *cost* as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the *cost* also minimizes the information-overload a user encounters.
- The choices for a *given* user for a given query is not known *apriori*
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the *cost* for the *average* case.

COST MODEL

- Define *cost* as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the *cost* also minimizes the information-overload a user encounters.
- The choices for a *given* user for a given query is not known *apriori*
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the *cost* for the *average* case.

COST MODEL

- Define *cost* as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the *cost* also minimizes the information-overload a user encounters.
- The choices for a *given* user for a given query is not known *apriori*
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the *cost* for the *average* case.

Cost Model

- Define *cost* as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the *cost* also minimizes the information-overload a user encounters.
- The choices for a *given* user for a given query is not known *apriori*
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the *cost* for the *average* case.

Cost Model

- Define *cost* as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the *cost* also minimizes the information-overload a user encounters.
- The choices for a *given* user for a given query is not known *apriori*
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the *cost* for the *average* case.

Cost Model
Probabilities

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

Cost Model

Probabilities

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

COST MODEL
PROBABILITIES

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

Cost Model

Probabilities

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

Cost Model

Probabilities

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

Cost Model

Probabilities

- Re-define *cost* as the total number of items, *on average*,both tuples and category labels, examined by the user
- The user choices in either exploration model are non-deterministic and not equally likely.
- This *uncertainty* and *preference* is captured by the following two probabilities
  - **Exploration Probability** $P(C)$: Probability that the user explores category C, using either SHOWCAT or SHOWTUPLES.
  - **SHOWTUPLES Probability** $P_w(C)$: Probability that the user goes for the option SHOWTUPLES, given that she *explores* C.
    - $P_w(C) = 1$ for a leaf category.
    - $(1 - P_w(C))$ is the probability that the user goes for the SHOWCAT option given that she explores C.

Cost Model

Cost :All

- For the ALL scenario,
  - For a given node a user chooses to explore, she user can either:
    - execute SHOWTUPLES with cost : $P_w(C) \times |tset(C)|$
    - execute a SHOWCAT with cost: $(1 - P_w(C)) \times [|C_t| + \sum_{i=1}^{|C_t|} P(C_i) \times Cost_{All}(C_i)]$
  - $$Cost_{All}(C) = P_w(C) \times |tset(C)| + (1 - P_w(C)) \times [|C_t| + \sum_{i=1}^{|C_t|} P(C_i) \times Co$$
- where C_t is the set of sub-categories of C

Cost Model

Cost :ONE

- For the ONE scenario,
  - For a given node a user chooses to explore, she user can either:
    1. execute SHOWTUPLES with cost : $P_w(C) \times frac(C) \times |tset(C)|$
    2. examine some(i) category labels until the relevant label is found and then explore that category further.
    3. The probability that C_i is the first category explored $(\prod_j^{i-1}(1 - P(C_j)) \times P(C_i)$
    4. The cost of exploring $C_i = |C_t| + Cost_{All}(C_i)])$
  - $Cost_{One}(C) =$ $P_w(C) \times frac(C) \times |tset(C)| + (1 - P_w(C)) \times \sum i = 1|C_t|P(C_i)$ $(\prod_j^{i-1}(1 - P(C_j)) \times P(C_i) \times [|C_t| + Cost_{All}(C_i)])$
  - where C_t is the set of sub-categories of C and, $frac(C)$ is the fraction of tuples the user needs to examine before finding the first relevant tuple

Using Workload to Estimate Probabilities

- $P(C)$ and $P_w(C)$ are needed for the $Cost_{One}(T)$ and $Cost_{All}(T)$
- Use aggregate knowledge of previous user behavior
- Specifically, infer user behavior from the queries executed previously by users of a given application - DBMS query Log

Using Workload to Estimate Probabilities

Computing SHOWTUPLES Probability

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_W(C)$, the probability that the user executes SHOWTUPLES.

Using Workload to Estimate Probabilities

Computing SHOWTUPLES Probability

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_w(C)$, the probability that the user executes SHOWTUPLES.

USING WORKLOAD TO ESTIMATE PROBABILITIES

COMPUTING SHOWTUPLES PROBABILITY

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_w(C)$, the probability that the user executes SHOWTUPLES.

Using Workload to Estimate Probabilities

Computing SHOWTUPLES Probability

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_w(C)$, the probability that the user executes SHOWTUPLES.

Using Workload to Estimate Probabilities

Computing SHOWTUPLES Probability

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_w(C)$, the probability that the user executes SHOWTUPLES.

USING WORKLOAD TO ESTIMATE PROBABILITIES

COMPUTING SHOWTUPLES PROBABILITY

Intuition :
A user does a SHOWTUPLES on a category C, if the user is interested in *all or most* values of C, or If a user is interested in only a few results (or sub-categories) of C, then she chooses the SHOWCAT option.

- W_i : Workload Query
- C_A : The categorizing attribute of C.
- N : total number queries in query log
- If W_i has a selection condition on C_A, then user is interested in a few categories of A.
- $\frac{N_{Attr}(C_A)}{N}$: the probability that the user executes SHOWCAT
- $\frac{1-N_{Attr}(C_A)}{N}$: $P_w(C)$, the probability that the user executes SHOWTUPLES.

USING WORKLOAD TO ESTIMATE PROBABILITIES
COMPUTING EXPLORATION PROBABILITY

P(C), probability that the user explores a category C, either by SHOWCAT or SHOWTUPLES

- = P(User explores C | User examines the label of C)
- = P(User explores C) $\div$ P(User examines the label of C)
- = P(User explores C) $\div$ P(User explores parent(C) and User examines the label of parent(C))
- = P(User explores C) $\div$ (P(User explores parent(C)) $\times$ P(User chooses SHOWCAT for parent(C) | User explores parent(C)))

Now,

- P(User chooses SHOWCAT for parent(C) | User explores parent(C)) = $N_{Attr}(parent(C_A)) \div N$
- P(User explores C) $\div$ P(User explores parent(C)) = P(User interested in label of C)
- P(User interested in label of C) = $\frac{N_{overlap}(C)}{N}$
- P(C) = P(User interested in label of C) $\left(\frac{N_{Attr}(parent(C))}{N}\right)$

$$P(C) = \frac{N_{overlap}(C)}{N_{Attr}(parent(C)_A)}$$

Using Workload to Estimate Probabilities

Computing Exploration Probability

P(C), probability that the user explores a category C, either by SHOWCAT or SHOWTUPLES

- = P(User explores C | User examines the label of C)
- = P(User explores C) $\div$ P(User examines the label of C)
- = P(User explores C) $\div$ P(User explores parent(C) and User examines the label of parent(C))
- = P(User explores C) $\div$ (P(User explores parent(C)) $\times$ P(User chooses SHOWCAT for parent(C) | User explores parent(C)))

Now,

- P(User chooses SHOWCAT for parent(C) | User explores parent(C)) = $N_{Attr}(parent(C_A)) \div N$
- P(User explores C) $\div$ P(User explores parent(C)) = P(User interested in label of C)
- P(User interested in label of C) = $\frac{N_{overlap}(C)}{N}$
- P(C) = P(User interested in label of C) $\left(\frac{N_{Attr}(parent(C))}{N}\right)$

$$P(C) = \frac{N_{overlap}(C)}{N_{Attr}(parent(C)_A)}$$

USING WORKLOAD TO ESTIMATE PROBABILITIES

COMPUTING EXPLORATION PROBABILITY

P(C), probability that the user explores a category C, either by SHOWCAT or SHOWTUPLES

- = P(User explores C | User examines the label of C)
- = P(User explores C) $\div$ P(User examines the label of C)
- = P(User explores C) $\div$ P(User explores parent(C) and User examines the label of parent(C))
- = P(User explores C) $\div$ (P(User explores parent(C)) $\times$ P(User chooses SHOWCAT for parent(C) | User explores parent(C)))

Now,

- P(User chooses SHOWCAT for parent(C) | User explores parent(C)) = $N_{Attr}(parent(C_A)) \div N$
- P(User explores C) $\div$ P(User explores parent(C)) = P(User interested in label of C)
- P(User interested in label of C) = $\frac{N_{overlap}(C)}{N}$
- P(C) = P(User interested in label of C) $\left(\frac{N_{Attr}(parent(C))}{N}\right)$

$$P(C) = \frac{N_{overlap}(C)}{N_{Attr}(parent(C)_A)}$$

Building the Category Tree

Naive Algorithm:

- Enumerate all possible category trees and the $Cost_{All}(T)$ for each Tree T.
- Choose the tree T_{opt} with the minimum cost

Exponential, in $|A| \times |C_A|!$
Apply heuristics to

- Eliminate "uninteresting" attributes.
- For every remaining attribute, obtain a "good" partitioning instead of enumerate all possible partitioning
- Level-wise partitioning - at each step choose the attribute and its partitioning that has the least cost.

Building the Category Tree

Naive Algorithm:

- Enumerate all possible category trees and the $Cost_{All}(T)$ for each Tree T.
- Choose the tree T_{opt} with the minimum cost

Exponential, in $|A| \times |C_A|$!

Apply heuristics to

- Eliminate "uninteresting" attributes.
- For every remaining attribute, obtain a "good" partitioning instead of enumerate all possible partitioning
- Level-wise partitioning - at each step choose the attribute and its partitioning that has the least cost.

Building the Category Tree

Naive Algorithm:

- Enumerate all possible category trees and the $Cost_{All}(T)$ for each Tree T.
- Choose the tree T_{opt} with the minimum cost

Exponential, in $|A| \times |C_A|$!
Apply heuristics to

- Eliminate “uninteresting” attributes.
- For every remaining attribute, obtain a “good” partitioning instead of enumerate all possible partitioning
- Level-wise partitioning - at each step choose the attribute and its partitioning that has the least cost.

Building the Category Tree

Reducing choices of categorizing attributes

- Presence of a selection condition on an attribute reflects user's interest in that attribute.
- Eliminate an attribute if it occurs infrequently in the workload queries i.e. $\frac{N_{Attr}(C_A)}{N} \leq X_{threshold}$,

Building the Category Tree

Reducing choices of categorizing attributes

- Presence of a selection condition on an attribute reflects user's interest in that attribute.
- Eliminate an attribute if it occurs infrequently in the workload queries i.e. $\frac{N_{Attr}(C_A)}{N} \leq X_{threshold}$,

Building the Category Tree

Partitioning for categorical attributes

For a query Q that contains a selection condition of the form: "A in $v_1, v_2, ..., v_k$" :

- $v_1, v_2, ..., v_k$ are potential categories
- Consider only *single-value* partitioning
- For *single-value* partitioning, only the presentation order (for categories) matters.
- $Cost_{All}(T)$ is not affected by the order.
- So, minimize for only $Cost_{One}(T)$

Theorem

$Cost_{One}(T)$ is minimum when the categories are presented to the user in increasing order of $\frac{1}{P(C_i)} + Cost_{One}(C_i)$

Heuristic: $Cost_{One}(C_i)$ as a constant (drop it)

The categories are presented in decreasing order of $N_{overlap}(C_i)$, or $occ(v_i)$

BUILDING THE CATEGORY TREE
PARTITIONING FOR CATEGORICAL ATTRIBUTES

For a query Q that contains a selection condition of the form: "A in $v_1, v_2, \ldots., v_k$" :

- $v_1, v_2, \ldots., v_k$ are potential categories
- Consider only *single-value* partitioning
- For *single-value* partitioning, only the presentation order (for categories) matters.
- $Cost_{All}(T)$ is not affected by the order.
- So, minimize for only $Cost_{One}(T)$

THEOREM

$Cost_{One}(T)$ is minimum when the categories are presented to the user in increasing order of $\frac{1}{P(C_i)} + Cost_{One}(C_i)$

Heuristic: $Cost_{One}(C_i)$ as a constant (drop it)
The categories are presented in decreasing order of $N_{overlap}(C_i)$, or $occ(v_i)$

Building the Category Tree

Partitioning for Numeric attributes

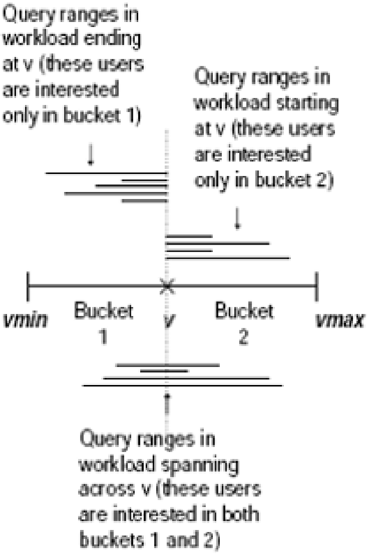
- Let V_{min} and V_{max} be the minimum and maximum values that the tuples in R can take in attribute A.
- Consider a point v ($V_{min} < v < V_{max}$):
  - If a significant number of query ranges in the workload begin or end at v, it is a good point to split as the workload suggests that most users would be interested in just one bucket,
  - If none of them begin or end at v, hence v is not a good point to split, if we partition the range into m-buckets then (m-1) points should be selected where queries begin or end splitpoints.
- the other factor is the number of tuples in each bucket.
- Define a *goodness* score, as $SUM(start_v, end_v)$, where
  - $start_v$ is the number of query ranges in the workload starting at v
  - end_v is the number of query ranges in the workload ending at v
- Precomute the *goodness* score for all potential split-points.

Building the Category Tree
Partitioning for Numeric attributes

- Let V_{min} and V_{max} be the minimum and maximum values that the tuples in R can take in attribute A.
- Consider a point v ($V_{min} < v < V_{max}$):
  - If a significant number of query ranges in the workload begin or end at v, it is a good point to split as the workload suggests that most users would be interested in just one bucket,
  - If none of them begin or end at v, hence v is not a good point to split, if we partition the range into m-buckets then (m-1) points should be selected where queries begin or end splitpoints.
- the other factor is the number of tuples in each bucket.
- Define a *goodness* score, as $SUM(start_v, end_v)$, where
  - $start_v$ is the number of query ranges in the workload starting at v
  - end_v is the number of query ranges in the workload ending at v
- Precomute the *goodness* score for all potential split-points.

Building the Category Tree

Partitioning for Numeric attributes



(a)

Splitpoint v	$start_v$	end_v	SUM ($start_v$, end_v)
1000	0	0	0
2000	10	40	50
3000	0	0	0
4000	0	0	0
5000	40	90	130
6000	0	0	0
7000	0	0	0
8000	80	20	100
9000	0	0	0
10000	30	-	-

(b)

Building the Category Tree

Multilevel Categorization

Greedy Algorithm:

1. For multilevel categorization, for each level l, determine the categorizing attribute A and for each category C in level (l-1), partition the domain of values of A in tset(C) such that the information overload is minimized.
2. The algorithm creates the categories level by level all categories at level (l-1) are created and added to tree T before any category at level l. S denote the set of categories at level (l-1) with more than M tuples.
3. For each such candidate attribute A, we partition each category C in S using the partitioning for Categorical Attributes and Numerical attributes.
4. Compute the cost of the attribute-partitioning combination for each candidate attribute A and select the attribute A with the minimum cost. For each category C in S, we add the partitions of C based on A to T.
5. This Completes the node creation at level l.

Building the Category Tree

Multilevel Categorization

Algorithm *CategorizeResults*(R)
begin
Create a root ("ALL") node (level = 0) and add to T
$l = 1$; // set current level to 1
while there exists at least one category at level l-1 with $|tset(C)|$>M
  $S \leftarrow$ {C | C is a category at level (l-1) and $|tset(C)|$>M}
  for each attribute A retained and not used so far
    if A is a categorical attribute
      $SCL \leftarrow$ list of single value categories in desc order of $occ(v_i)$
      for each category C in S
        Tree(C,A)$\leftarrow$Tree with C as root and each non-empty cat
          C'$\in$SCL in same order as children of C
    else // A is a numeric attribute
      SPL$\leftarrow$list of potential splitpoints sorted by goodness score
      for each category C in S
        Select (m-1) top necessary splitpoints from SPL
        Tree(C,A)$\leftarrow$Tree with C as root with corr. buckets in
          ascending order of values as children of C
    $COST_A \leftarrow \sum_{C \in S} P(C) * Cost_{All}(Tree(C,A))$
  Select α = argmin$_A$ $COST_A$ as categorizing attribute for level l
  for each category C in S
    Add partitioning Tree(C,α)obtained using attribute α to T
  $l = l + 1$; //finished creating nodes at this level, go to next level
end

EXPERIMENTAL EVALUATION

Empirical studies to:

- Evaluate the accuracy of the the cost-model
- Comparision of the cost-based categorization model and compare it “other” models

Experimental Evaluation
Methodology

- Dataset
  - A single *ListProperty* table, with about 1.7m tuples
  - Attributes include *Location*, *price*, *year-built*,*square-footage* . . .
- Workload : Over 176,000 query strings representing searches on the “MSN House and Home” web-site.
- Comparision Models
  - *No Cost* Categorization attribute and partitioning selected arbitrarily.
  - *Attr-Cost* Attribute selection is cost-based but partitioning is arbitrary.

Experimental Evaluation

Results

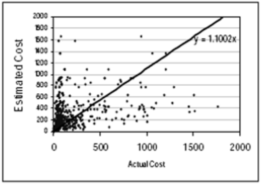


Figure 7: Correlation between actual cost and estimated cost

Subset	Correlation
1	0.39
2	0.98
3	0.32
4	0.48
5	0.16
6	0.16
7	0.19
8	0.76
All	0.90

Table 1: Pearson's Correlation between estimated cost and actual cost

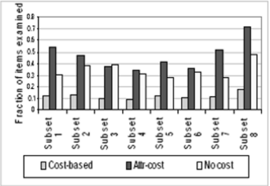


Figure 8: Cost of various techniques

Experimental Evaluation
Conclusion

- Accurate Categorization model
- Better Categorization Algorithm