Automatic Categorization of Query Results

SIGMOD ’04

F. Kaushik Chakrabarti ¹ S. Surajit Chaudhuri ¹ T. Seung-won Hwang²

¹Microsoft Research
²Univ. 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.
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.
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 *answer set*.

Ensure “even” distribution of query results across the category.
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 *answer set*.

Ensure “even” distribution of query results across the category.
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 *answer set*.

Ensure “even” distribution of query results across the category.
Automatic Categorization of Query Results

Example:

```
All
  \--- Neighborhood Redmond
        \- Price 200-225K
            \- Price 225-250K
                \- Price 250-275K
  \--- Neighborhood Issaquah
        \- Price 200-275K
            \- Price 275-300K
    \--- Neighborhood Seattle
        ..
 0 1 2 ..
```

Example of hierarchical categorization
# Table of Contents

- Categorization basics
- Exploration Model - simulating a “typical” user
- Cost estimation - probabilistic
- Estimating probabilities using workload
- Heuristics
- Categorization algorithm
- Experimental evaluation
A hierarchical categorization of $R$ is a recursive partitioning of the tuples in $R$ defined inductively as follows:

- **Base Case:** Given a ALL node containing all tuples in $R$, partition $R$ using a single attribute.
- **Inductive Step:** Given a node $C$ at level $l - 1$, partition (level 1) set of tuples $tset(C)$ 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 $C$ is:

- $tset(C)$ : Set of tuples contained in a category $C$.
- $label(C)$ :
  - 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 $a_1, a_2 \in dom_R(A)$.
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

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).
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)
Model of exploration of node C in \texttt{ALL} scenario:

\textbf{Algorithm 1 Explore C}

1: \textbf{if} \ C \ \text{is a non-leaf node} \ \textbf{then}
2: \quad \text{Choose one of the following:}
3: \quad (1) \ \text{Examine all tuples in } \texttt{tset}(C) \ \{\text{Option SHOWTUPLES}\}
4: \quad (2) \ \{\text{Option SHOWCAT}\}
5: \quad \textbf{for} \ i = 1; \ i \leq n; \ i++ \ \textbf{do}
6: \quad \quad \text{Examine the label of ith subcategory}
7: \quad \quad \text{Choose one of the following}
8: \quad \quad (2.1) \ \text{Explore } C_i
9: \quad \quad (2.2) \ \text{Ignore } C_i
10: \quad \textbf{end for}
11: \ \textbf{else}
12: \quad \text{Examine all tuples in } \texttt{tset}(C)
13: \ \textbf{end if}
Model of exploration of node C in ONE scenario:

\textbf{Algorithm 2} Explore C

1: \textbf{if} C is a non-leaf node \textbf{then}
2: \hspace{1em} Choose one of the following:
3: \hspace{2em} (1) Examine tuples in $tset(C)$ till the first relevant tuple found
4: \hspace{3em} \{Option SHOWTUPLES\}
5: \hspace{2em} (2)\{Option SHOWCAT\}
6: \hspace{1em} \textbf{for} ($i = 1; i \leq n; i++$) \textbf{do}
7: \hspace{2em} Examine the label of ith subcategory
8: \hspace{2em} Choose one of the following
9: \hspace{3em} (2.1) Explore $C_i$
10: \hspace{3em} (2.2) Ignore $C_i$
11: \hspace{2em} \textbf{if} choice = Explore \textbf{then}
12: \hspace{3em} break
13: \hspace{2em} \textbf{end if}
14: \hspace{1em} \textbf{end for}
15: \hspace{1em} \textbf{else}
16: \hspace{2em} Examine tuples in $tset(C)$ till the first relevant tuple found
17: \hspace{1em} \textbf{end if}
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 \textit{cost} as the total number of items, both tuples and category labels, examined by the user.
- Minimizing the \textit{cost} also minimizes the information-overload a user encounters.
- The choices for a \textit{given} user for a given query is not known \textit{apriori}.
  - but the aggregate-knowledge of previous user behavior is known!
- Use the previous knowledge to estimate the \textit{cost} for the \textit{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.
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 $C$. 

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.
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 $C$. 
For the ALL scenario,

- For a given node a user chooses to explore, she can either:
  - execute SHOWTUPLES with cost: $P_w(C) \times |tset(C)|$
  - execute a SHOWCAT with cost:
    $\left(1 - P_w(C)\right) \times \left[|C_t| + \sum_{i=1}^{\left|C_t\right|} P(C_i) \times Cost_{All}(C_i)\right]$

$Cost_{All}(C) = P_w(C) \times |tset(C)| + \left(1 - P_w(C)\right) \times \left[|C_t| + \sum_{i=1}^{\left|C_t\right|} P(C_i) \times Cost_{All}(C_i)\right]$

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: \( P_w(C) \times \text{frac}(C) \times |tset(C)| \)
    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:
       \[
       (\prod_{j=1}^{i-1} (1 - P(C_j))) \times P(C_i)
       \]
    4. The cost of exploring \( C_i = |C_t| + \text{Cost}_{All}(C_i) \))

- \( \text{Cost}_{One}(C) =
  P_w(C) \times \text{frac}(C) \times |tset(C)| + (1 - P_w(C)) \times \sum_{i = 1} |C_t| P(C_i)
  
  (\prod_{j=1}^{i-1} (1 - P(C_j))) \times P(C_i) \times [|C_t| + \text{Cost}_{All}(C_i)]\)

- where \( C_t \) is the set of sub-categories of \( C \) and, \( \text{frac}(C) \) is the fraction of tuples the user needs to examine before finding the first relevant tuple.


$P(C)$ 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.
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.
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.

$$\frac{N_{\text{Attr}}(C_A)}{N}$$ : the probability that the user executes SHOWCAT

$$\frac{1 - N_{\text{Attr}}(C_A)}{N}$$ : $P_w(C)$, 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.
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**

\[
P(C) = P(\text{User explores } C \mid \text{User examines the label of } C) \\
= P(\text{User explores } C) \div P(\text{User examines the label of } C) \\
= P(\text{User explores } C) \div P(\text{User explores parent(C) and User examines the label of parent(C)}) \\
= P(\text{User explores } C) \div (P(\text{User explores parent(C)}) \times P(\text{User chooses SHOWCAT for parent(C) } | \text{ User explores parent(C)}))
\]

Now,

\[
P(\text{User chooses SHOWCAT for parent(C) } | \text{ User explores parent(C)}) = \frac{N_{Attr}(\text{parent(C)})}{N} \\
P(\text{User explores C} \div P(\text{User explores parent(C)}) = P(\text{User interested in label of C}) \\
P(\text{User interested in label of C}) = \frac{N_{overlap}(C)}{N} \\
P(C) = P(\text{User interested in label of C}) \times \left( \frac{N_{Attr}(\text{parent(C)})}{N} \right)
\]

\[
P(C) = \frac{N_{overlap}(C)}{N_{Attr}(\text{parent(C)})}
\]
P(C), probability that the user explores a category C, either by SHOWCAT or SHOWTUPLES

\[ P(C) = P(\text{User explores C} \mid \text{User examines the label of C}) \]
\[ = P(\text{User explores C}) \div P(\text{User examines the label of C}) \]
\[ = P(\text{User explores C}) \div P(\text{User explores parent(C) and User examines the label of parent(C)}) \]
\[ = P(\text{User explores C}) \div (P(\text{User explores parent(C)}) \times P(\text{User chooses SHOWCAT for parent(C) \mid User explores parent(C)})) \]

Now,

\[ P(\text{User chooses SHOWCAT for parent(C) \mid User explores parent(C)}) = \frac{N_{\text{Attr}}(\text{parent}(C_A))}{N} \]
\[ P(\text{User explores C}) \div P(\text{User explores parent(C)}) = P(\text{User interested in label of C}) \]
\[ P(\text{User interested in label of C}) = \frac{N_{\text{overlap}}(C)}{N} \]
\[ P(C) = P(\text{User interested in label of C}) \times (\frac{N_{\text{Attr}}(\text{parent}(C))}{N}) \]

\[ P(C) = \frac{N_{\text{overlap}}(C)}{N_{\text{Attr}}(\text{parent}(C)_A)} \]
P(C), probability that the user explores a category C, either by SHOWCAT or SHOWTUPLES

\[ P(C) = \frac{P(\text{User explores } C \mid \text{User examines the label of } C)}{P(\text{User explores the label of } C)} \]

\[ = \frac{P(\text{User explores } C)}{P(\text{User examines the label of } C)} \]

\[ = \frac{P(\text{User explores } C)}{P(\text{User explores parent(C) and User examines the label of parent(C)})} \]

\[ = \frac{P(\text{User explores } C)}{P(\text{User explores parent(C)}) \times P(\text{User chooses SHOWCAT for parent(C) } | \text{ User explores parent(C))})} \]

Now,

\[ P(\text{User chooses SHOWCAT for parent(C) } | \text{ User explores parent(C)}) = \frac{N_{Attr}(\text{parent}(C))}{N} \]

\[ P(\text{User explores C}) = P(\text{User interested in label of C}) \cdot \frac{N_{Attr}(\text{parent}(C))}{N} \]

\[ P(C) = \frac{N_{\text{overlap}}(C)}{N_{Attr}(\text{parent}(C))} \]
BUILDING THE CATEGORY TREE

Naive Algorithm:

- Enumerate all possible category trees and the $\text{Cost}_{\text{All}}(T)$ 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.
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.
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. \[
\frac{N_{Attr}(C_A)}{N} \leq X_{\text{threshold}},
\]
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. \( \frac{N_{\text{Attr}}(C_A)}{N} \leq X_{\text{threshold}} \),
**Building the Category Tree**

**Partitioning for categorical attributes**

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

- $v_1, v_2, \ldots, v_k$ 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 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 $occ(v_i)$
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 $V_{\text{min}}$ and $V_{\text{max}}$ be the minimum and maximum values that the tuples in $R$ can take in attribute $A$.

Consider a point $v$ ($V_{\text{min}} < v < V_{\text{max}}$):

- 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 $\text{SUM}(\text{start}_v, \text{end}_v)$, where
  - $\text{start}_v$ is the number of query ranges in the workload starting at $v$
  - $\text{end}_v$ is the number of query ranges in the workload ending at $v$

- Precompute the *goodness* score for all potential split-points.
BUILDING THE CATEGORY TREE
Partitioning for Numeric Attributes

(a)

(b)

<table>
<thead>
<tr>
<th>Splitpoint</th>
<th>start&lt;sub&gt;v&lt;/sub&gt;</th>
<th>end&lt;sub&gt;v&lt;/sub&gt;</th>
<th>SUM (start&lt;sub&gt;v&lt;/sub&gt;, end&lt;sub&gt;v&lt;/sub&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2000</td>
<td>10</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>3000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5000</td>
<td>40</td>
<td>90</td>
<td>130</td>
</tr>
<tr>
<td>6000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8000</td>
<td>80</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>9000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10000</td>
<td>30</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
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 \( \text{tset}(C) \) 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 \( l \).
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 $|tset(C)| \geq M$
    $S \leftarrow \{ C | C \text{ is a category at level (l-1) and } |tset(C)| \geq M \}$
    for each attribute $A$ retained and not used so far
        if $A$ is a categorical attribute
            $SCL \leftarrow \text{list of single value categories in desc order of } \text{occ}(v_i)$
            for each category $C$ in $S$
                $\text{Tree}(C,A) \leftarrow \text{Tree with } C \text{ as root and each non-empty cat}$
                $C' \in SCL \text{ in same order as children of } C$
        else // $A$ is a numeric attribute
            $SPL \leftarrow \text{list of potential splitpoints sorted by goodness score}$
            for each category $C$ in $S$
                Select (m-1) top necessary splitpoints from SPL
                $\text{Tree}(C,A) \leftarrow \text{Tree with } C \text{ as root with corr. buckets in}$
                $\text{ascending order of values as children of } C$
                $\text{COST}_A \leftarrow \sum_{C \in S} P(C) \cdot \text{Cost}_{\text{All}}(\text{Tree}(C,A))$
                Select $\alpha = \arg\min_A \text{COST}_A$ as categorizing attribute for level $l$
            for each category $C$ in $S$
                Add partitioning $\text{Tree}(C,\alpha)$ obtained using attribute $\alpha$ to $T$
        $l = l + 1; // finished creating nodes at this level, go to next level$
end
Empirical studies to:

- Evaluate the accuracy of the cost-model
- Comparison 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.

- **Comparison 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

**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