CRSI: A Compact Randomized Similarity Index for Set-Valued Features

Petros Venetis[1] Yannis Sismanis[2] Berthold Reinwald[2]

[1]Stanford University

[2]IBM Research – Almaden

March 29, 2012

Problem Statement

Input

- Relational database(s) with columns $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$
- Similarity function $\text{Sim} : \mathcal{C} \times \mathcal{C} \to [0, 1]$
- Similarity threshold t

Output

- Pairs of similar relational columns
  - $\{(a_i, a_j) \mid a_i, a_j \in \mathcal{C}, \text{Sim}(a_i, a_j) \geq t\}$

Problem Statement

Input

- Relational database(s) with columns $\mathcal{C} = \{a_1, a_2, \ldots, a_n\}$
- Similarity function $\text{Sim} : \mathcal{C} \times \mathcal{C} \to [0, 1]$
- Similarity threshold t

Output

- Pairs of similar relational columns
  - $\{(a_i, a_j) \mid a_i, a_j \in \mathcal{C}, \text{Sim}(a_i, a_j) \geq t\}$

Applications

Data Profiling

Process of examining the data available in an existing data source and collecting statistics and information about that data

Data Integration

Combining data residing in different sources and providing users with a unified view of these data

Duplicate Document Detection

Detect similar documents while crawling to minimize index redundancy

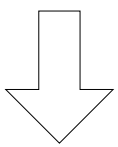
Example: Data Profiling

Example: Data Profiling

Employee	Manager	Salary

Example: Data Profiling

Employee	Manager	Salary

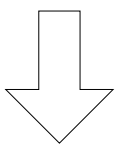


Copy

Employee	Manager	Salary

Example: Data Profiling

Employee	Manager	Salary

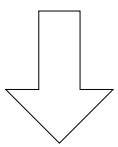


Copy

Employee	Manager	Salary

Example: Data Profiling

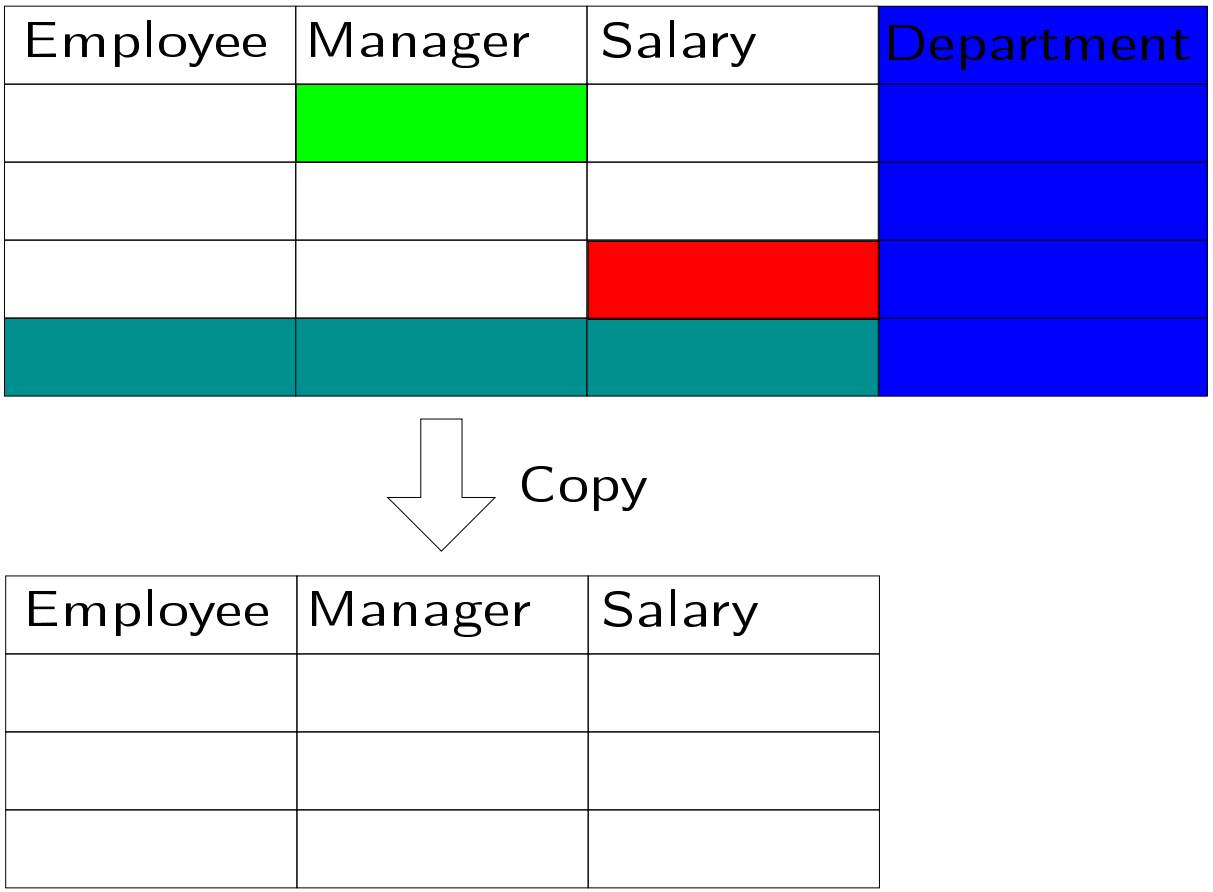
Employee	Manager	Salary



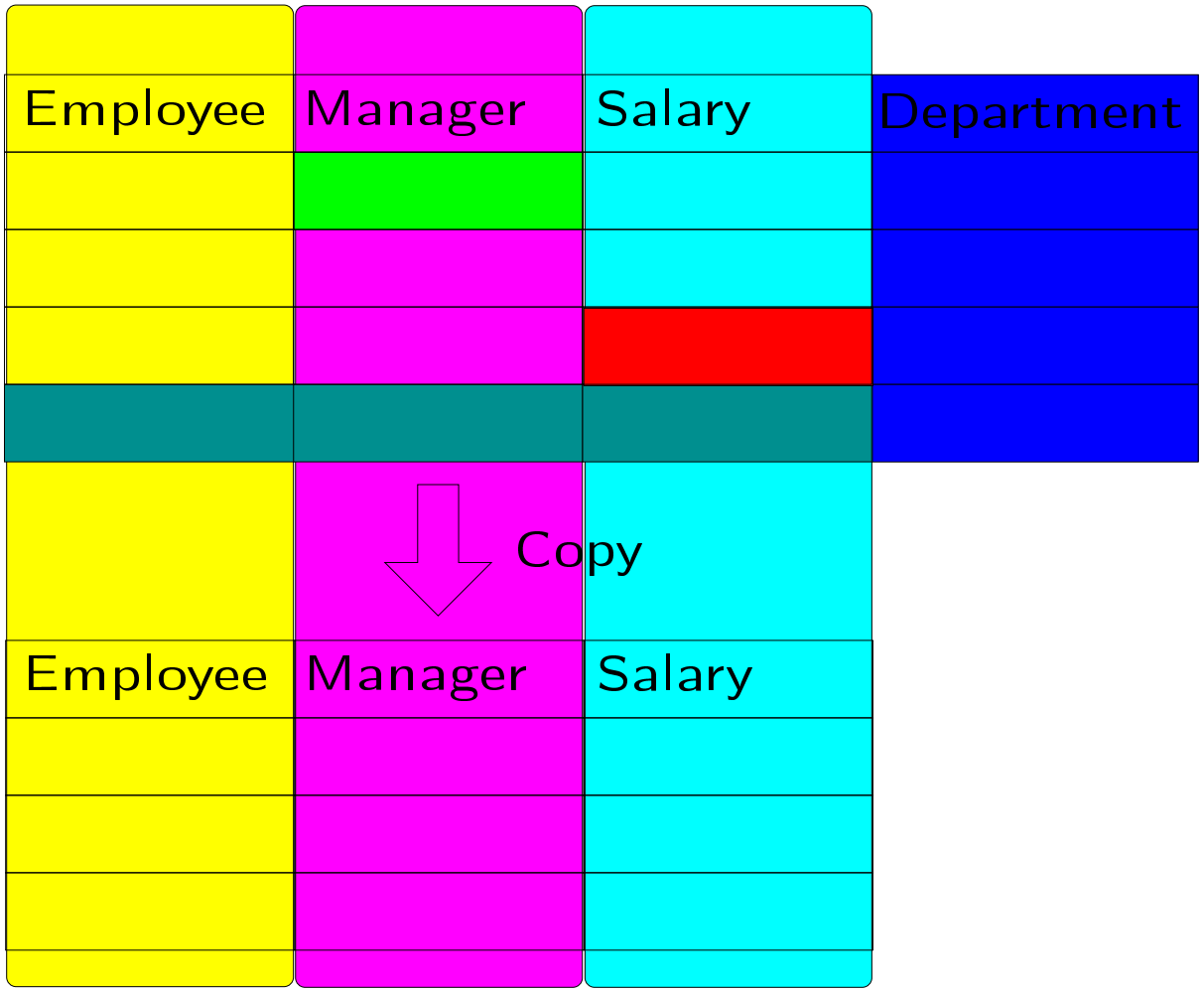
Copy

Employee	Manager	Salary

Example: Data Profiling



Example: Data Profiling



Typical Solutions

Set-Similarity Queries

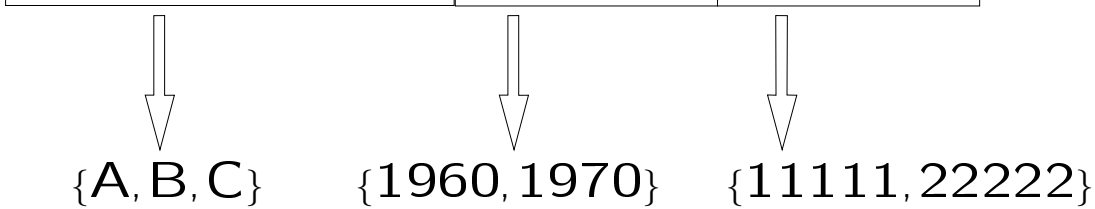
Similarities over sets

- Treat relational columns as sets
- Use some similarity function to “compare” sets
- Determine pairs of sets with high similarity and mark as interesting

Similarities over sets

- Treat relational columns as sets
- Use some similarity function to "compare" sets
- Determine pairs of sets with high similarity and mark as interesting

CustomerName	YOB	ZIP
A	1960	11111
B	1960	22222
C	1970	11111



Typical Solutions

Set-Similarity Queries

Similarities over sets

- Treat relational columns as sets
- Use some similarity function to “compare” sets
- Determine pairs of sets with high similarity and mark as interesting

Typical Solutions

Set-Similarity Queries

Similarities over sets

- Treat relational columns as sets
- Use some similarity function to “compare” sets
- Determine pairs of sets with high similarity and mark as interesting

Possible Similarity Functions

- Jaccard similarity function
- Dice similarity function
- Overlap function
- Foreign key function

Possible Similarity Functions

- Jaccard similarity function

$$\text{Jaccard}(a_1, a_2) = \frac{|a_1 \cap a_2|}{|a_1 \cup a_2|}$$

- Dice similarity function
- Overlap function
- Foreign key function

Solution Summary

Process

1. Find KMV synopsis of each relational column
2. Create inverted index from KMV synopsis hash values to the relational columns
3. Query the inverted index to determine similar columns

Contributions

- Similarity function is an art $\rightarrow$ Support many Sim() functions
- Fast querying
- Little additional space
- Updateable

Approximate Results

- Acceptable if we can get controllable errors

Solution Summary

Process

1. Find KMV synopsis of each relational column
2. Create inverted index from KMV synopsis hash values to the relational columns
3. Query the inverted index to determine similar columns

Contributions

- Similarity function is an art $\rightarrow$ Support many Sim() functions
- Fast querying
- Little additional space
- Updateable

Approximate Results

- Acceptable if we can get controllable errors

Solution Summary

Process

1. Find KMV synopsis of each relational column
2. Create inverted index from KMV synopsis hash values to the relational columns
3. Query the inverted index to determine similar columns

Contributions

- Similarity function is an art $\rightarrow$ Support many Sim() functions
- Fast querying
- Little additional space
- Updateable

Approximate Results

- Acceptable if we can get controllable errors

Solution Summary

Process

1. Find KMV synopsis of each relational column
2. Create inverted index from KMV synopsis hash values to the relational columns
3. Query the inverted index to determine similar columns

Contributions

- Similarity function is an art $\rightarrow$ Support many Sim() functions
- Fast querying
- Little additional space
- Updateable

Approximate Results

- Acceptable if we can get controllable errors

Solution Summary

Process

1. Find KMV synopsis of each relational column
2. Create inverted index from KMV synopsis hash values to the relational columns
3. Query the inverted index to determine similar columns

Contributions

- Similarity function is an art $\rightarrow$ Support many Sim() functions
- Fast querying
- Little additional space
- Updateable

Approximate Results

- Acceptable if we can get controllable errors

Tools

- KMV Synopsis (or bottom-k)
- Inverted Indexes

KMV Synopsis: Definition

Definition: k-Minimum hash Values

The KMV synopsis of a set $a = \{w_1, w_2, \ldots, w_m\} \subseteq \mathcal{U}$ under a hash function $h : \mathcal{U} \to \{0, 1, \ldots, M\}$ is the set of the k minimum hash values to which the items map to.

KMV Synopsis: Definition

Definition: **k-Minimum hash Values**

The KMV synopsis of a set $a = \{w_1, w_2, \ldots, w_m\} \subseteq \mathcal{U}$ under a hash function $h : \mathcal{U} \to \{0, 1, \ldots, M\}$ is the set of the k minimum hash values to which the items map to.

Example

{dog, cat, snake,
elephant, tiger}

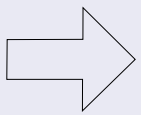
KMV Synopsis: Definition

Definition: **k-Minimum hash Values**

The KMV synopsis of a set $a = \{w_1, w_2, \ldots, w_m\} \subseteq \mathcal{U}$ under a hash function $h : \mathcal{U} \rightarrow \{0, 1, \ldots, M\}$ is the set of the k minimum hash values to which the items map to.

Example

{dog, cat, snake, elephant, tiger}



Hashing

dog	1
cat	3
snake	7
elephant	5
tiger	1

KMV Synopsis: Definition

Definition: **k-Minimum hash Values**

The KMV synopsis of a set $a = \{w_1, w_2, \ldots, w_m\} \subseteq \mathcal{U}$ under a hash function $h : \mathcal{U} \rightarrow \{0, 1, \ldots, M\}$ is the set of the k minimum hash values to which the items map to.

Example

{dog, cat, snake, elephant, tiger} ⇨ $\{1, 3, 5, 7\}$

Hashing

dog	1
cat	3
snake	7
elephant	5
tiger	1

KMV Synopsis: Definition

Definition: **k-Minimum hash Values**

The KMV synopsis of a set $a = \{w_1, w_2, \ldots, w_m\} \subseteq \mathcal{U}$ under a hash function $h : \mathcal{U} \to \{0, 1, \ldots, M\}$ is the set of the k minimum hash values to which the items map to.

Example

{dog, cat, snake, elephant, tiger} $\Rightarrow$ $\{1, 3, 5, 7\}$ $\xRightarrow[k=2]{}$ KMV $= \{1, 3\}$

Hashing

dog	1
cat	3
snake	7
elephant	5
tiger	1

KMV Synopsis: Existing Estimators

- Distinct values
- Overlap size
- Union size
- Difference size

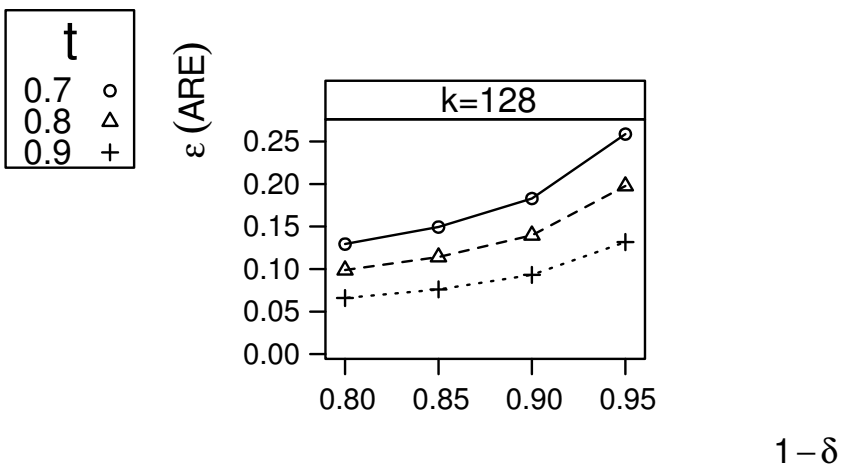
KMV Synopsis: Jaccard Similarity Estimators

- $\hat{J}(a_i, a_j) = \frac{|\text{KMV}(a_i) \oplus \text{KMV}(a_j)|}{k}$ is an unbiased estimator of $J(a_i, a_j)$
- Probability δ that ARE$=\frac{|\hat{J}-J|}{J}$ is greater than ε is bounded by:

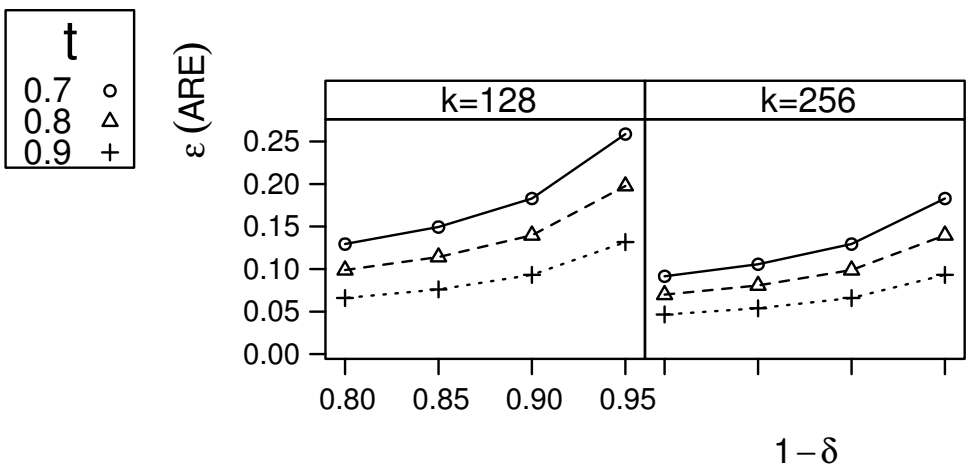
$$\delta \leq \frac{1-J}{kJ\varepsilon^2} \leq \frac{1-t}{k\varepsilon^2 t}$$

$\rightarrow$ Guidelines for choosing k

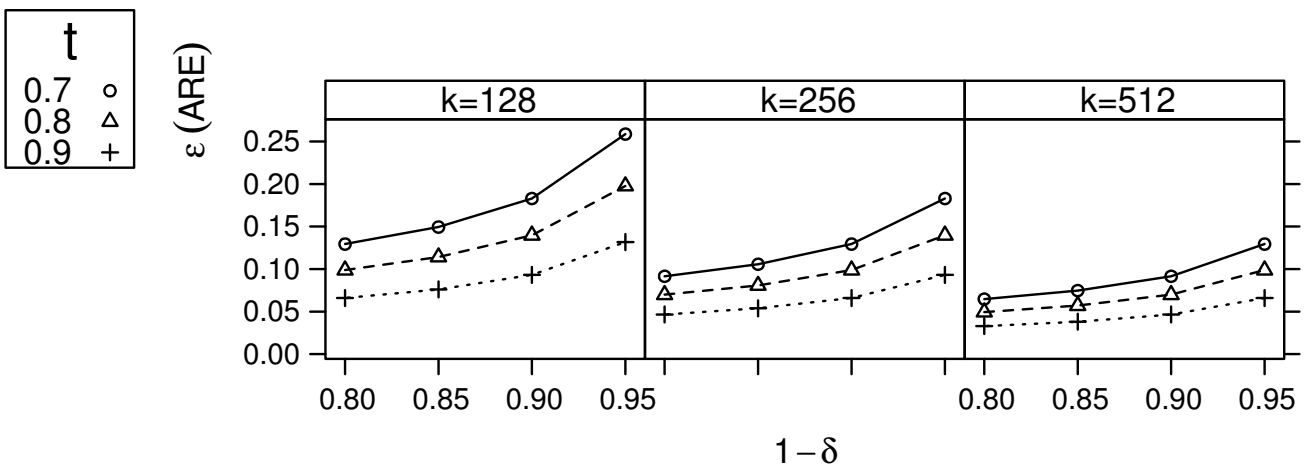
KMV Synopsis: Errors decrease on k and t



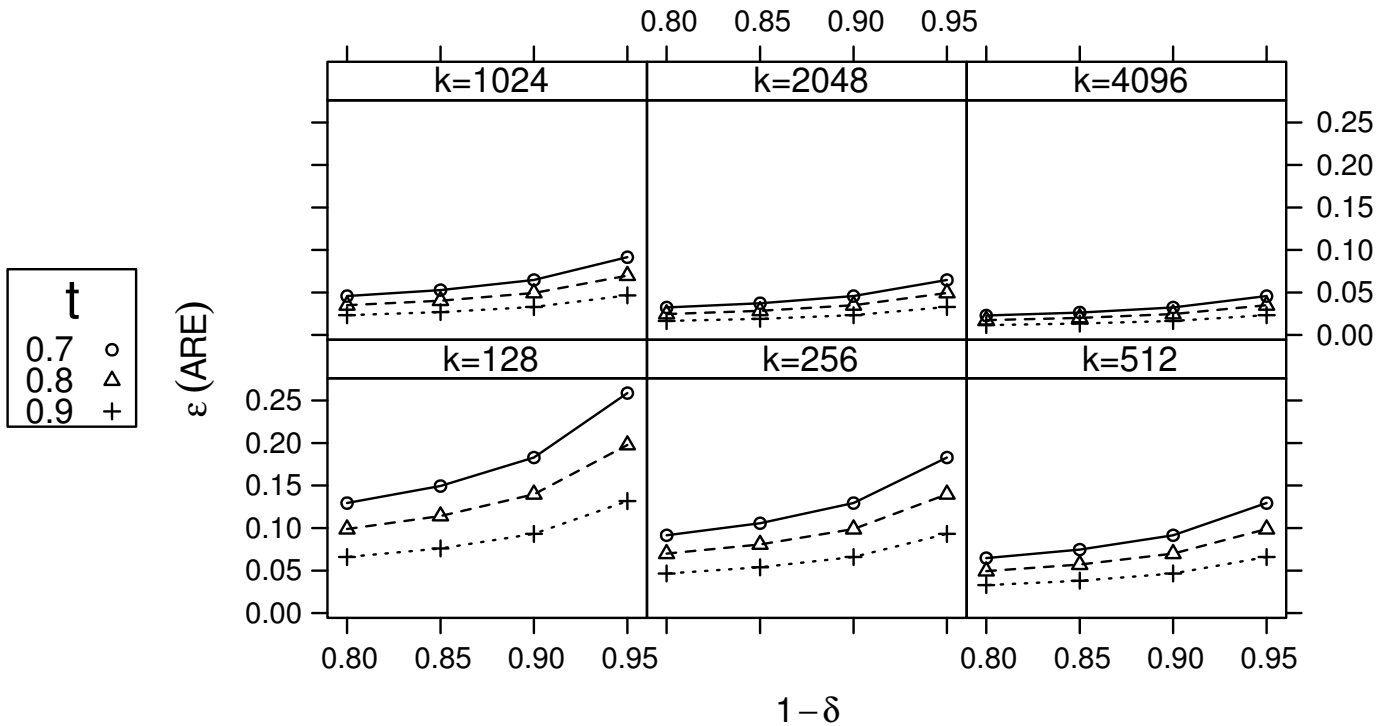
KMV Synopsis: Errors decrease on k and t



KMV Synopsis: Errors decrease on k and t



KMV Synopsis: Errors decrease on k and t

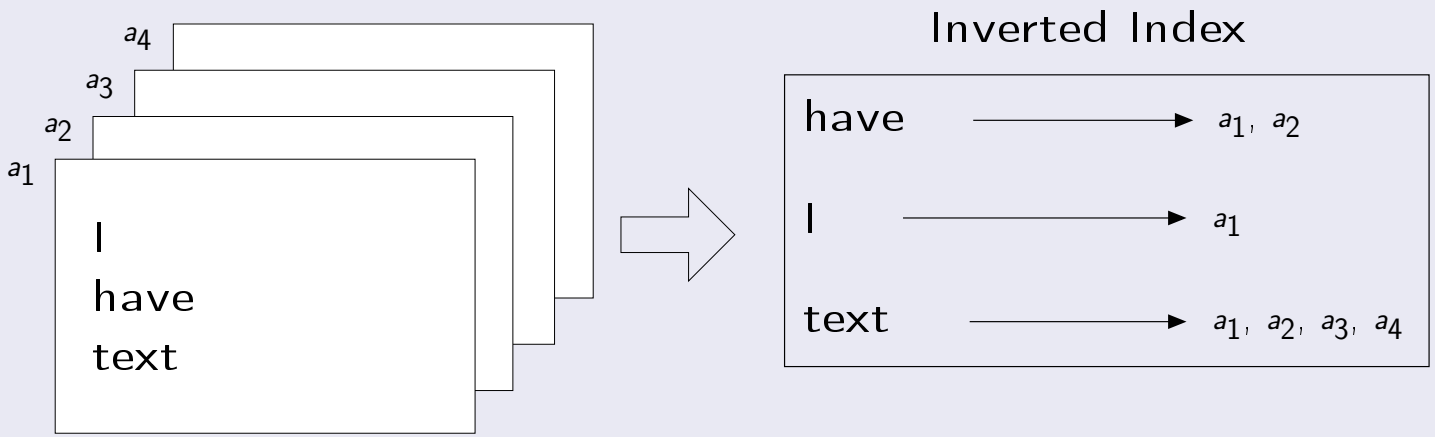


Tools

- KMV Synopsis (or bottom-k)
- Inverted Indexes

Inverted Indexes

Mapping from content to container



Skip Lists

Expected $O(\log n)$ lookup (doc id $\geq a_i$) containing term

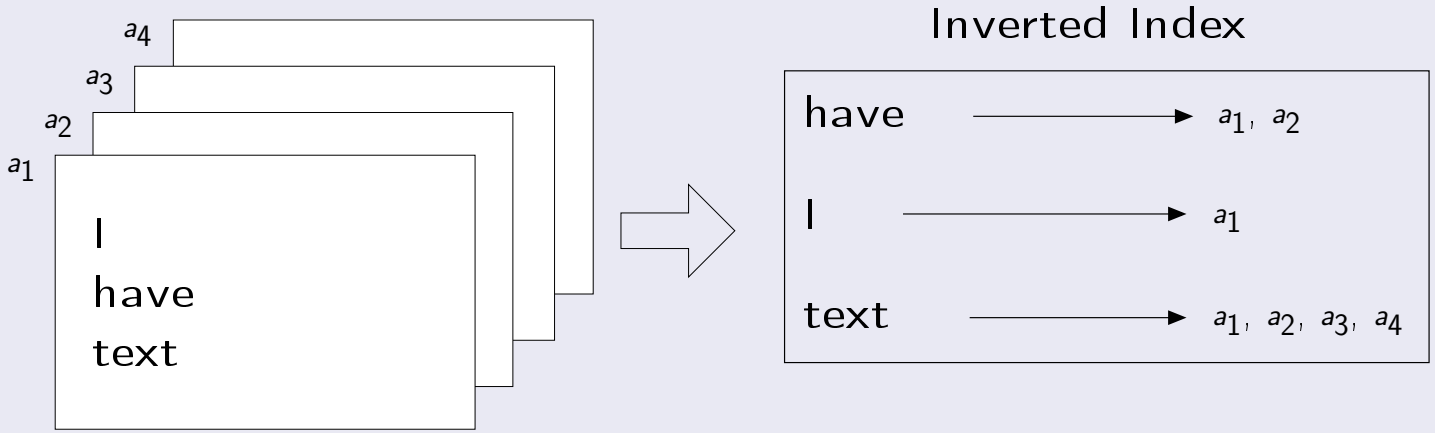
Stored Fields

Metadata associated with a document id

a_1: 3

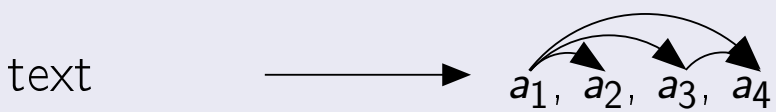
Inverted Indexes

Mapping from content to container



Skip Lists

Expected $O(\log n)$ lookup (doc id $\geq a_i$) containing term



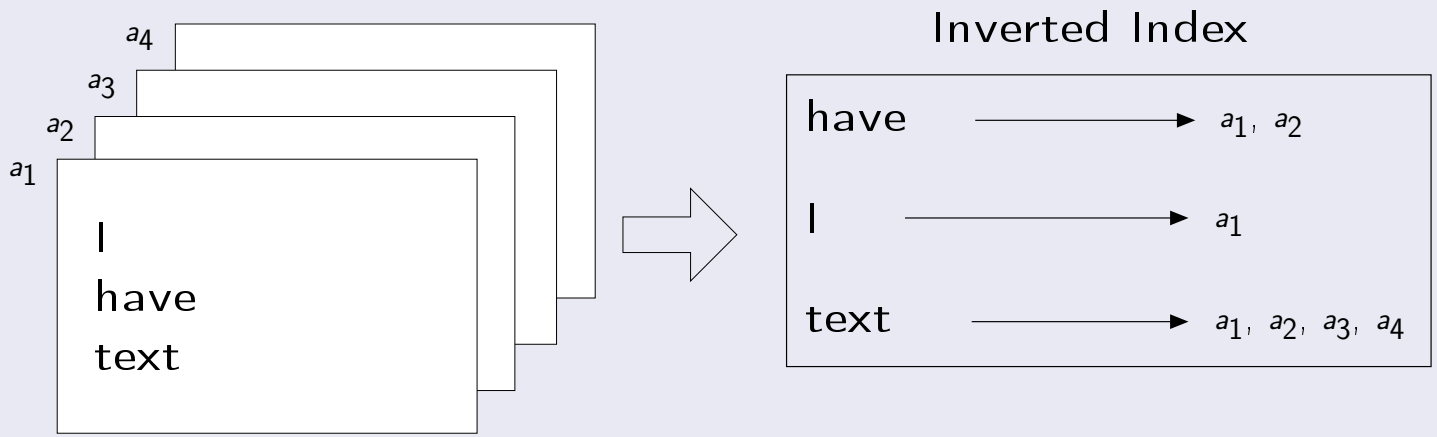
Stored Fields

Metadata associated with a document id

a_1: 3

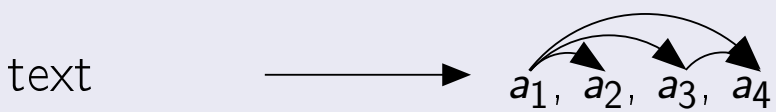
Inverted Indexes

Mapping from content to container



Skip Lists

Expected $O(\log n)$ lookup (doc id $\geq a_i$) containing term



Stored Fields

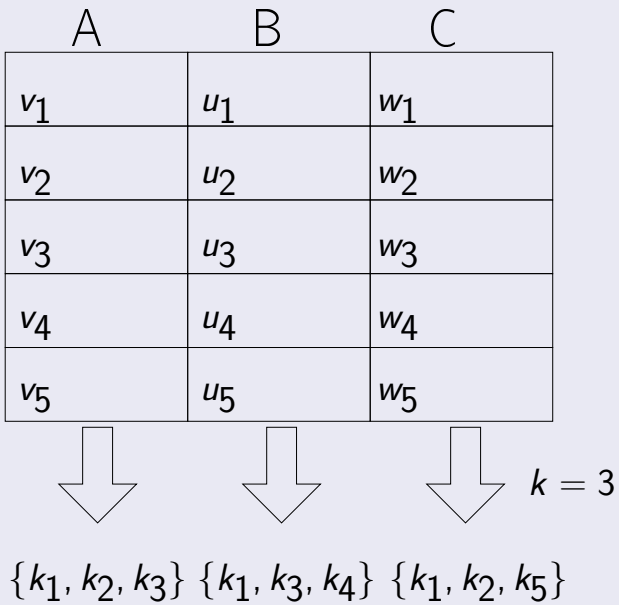
Metadata associated with a document id

a_1: 3

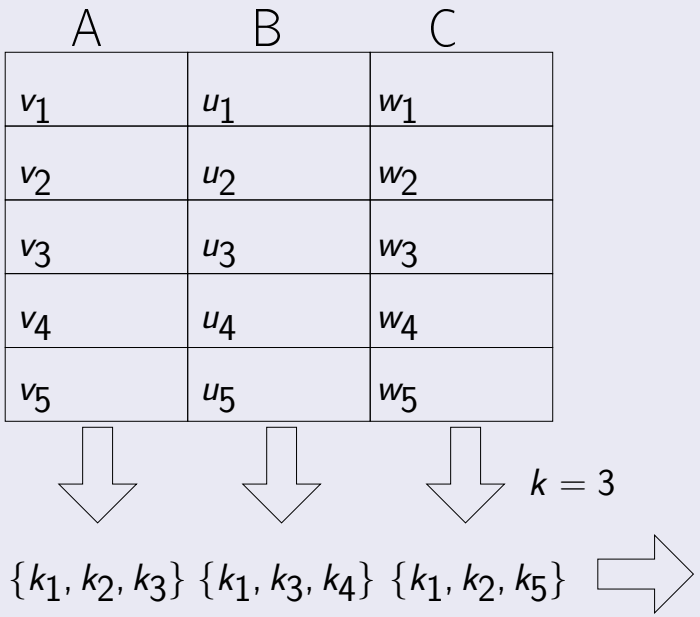
Index Construction

A	B	C
v_1	u_1	w_1
v_2	u_2	w_2
v_3	u_3	w_3
v_4	u_4	w_4
v_5	u_5	w_5

Index Construction



Index Construction



Inverted Index

$k_1 \rightarrow$ A, B, C	
$k_2 \rightarrow$ A, C	A: $\{k_1, k_2, k_3\}$
$k_3 \rightarrow$ A, B	B: $\{k_1, k_3, k_4\}$
$k_4 \rightarrow$ B	C: $\{k_1, k_2, k_5\}$
$k_5 \rightarrow$ C	

Index Querying

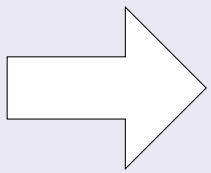
Inverted Index

$k_1 \rightarrow$ A, B, C	
$k_2 \rightarrow$ A, C	A: $\{k_1, k_2, k_3\}$
$k_3 \rightarrow$ A, B	B: $\{k_1, k_3, k_4\}$
$k_4 \rightarrow$ B	C: $\{k_1, k_2, k_5\}$
$k_5 \rightarrow$ C	

Index Querying

Inverted Index

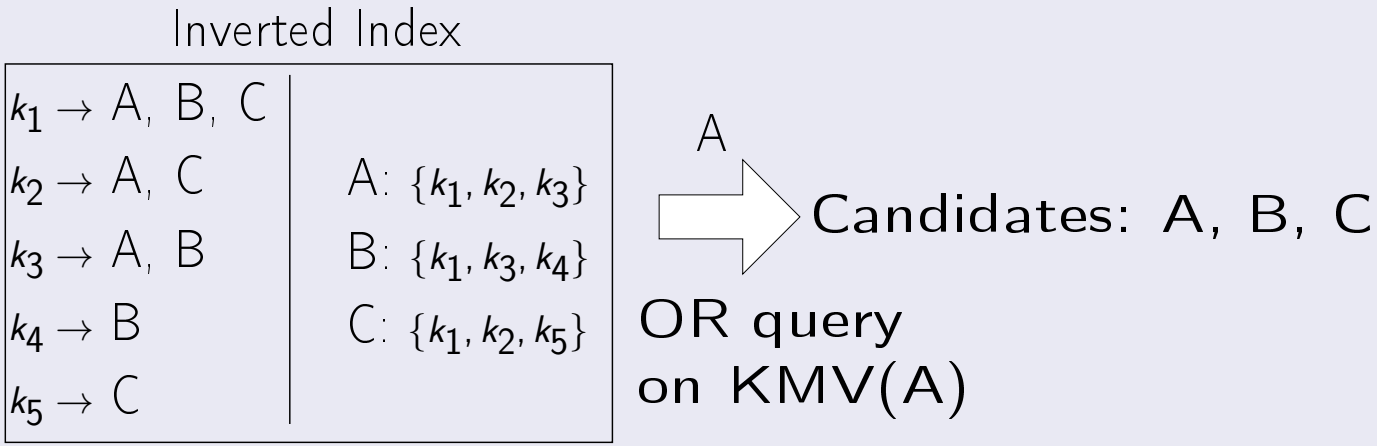
$k_1 \rightarrow$ A, B, C	
$k_2 \rightarrow$ A, C	A: $\{k_1, k_2, k_3\}$
$k_3 \rightarrow$ A, B	B: $\{k_1, k_3, k_4\}$
$k_4 \rightarrow$ B	C: $\{k_1, k_2, k_5\}$
$k_5 \rightarrow$ C	



OR query on KMV(A)

Algorithms

Index Querying



Easy Solution: OR Queries

For a_i: Find all a_j with one overlapping KMV hash value

Other Solutions

- We discuss multiple improvements in the paper
- Similarity queries using CRSI can be mapped to variations of WAND queries
- $10\times$ improvement

WAND Queries

For a query of m terms, find all doc ids with at least n overlapping terms

Algorithms – Pseudocode

CRSI Construction

1. Iterate over relational columns
2. Find KMV for each column
3. Create inverted index from hash values to relational columns
4. Store KMV as a stored field for each column ($\sim$ doubling the index size, but offers significant query answering improvements)

Similarity Index Querying – Similarity Self-Join

1. For each relational column a_i in the index
2. Perform a query to find other columns a_j for which $\widehat{\mathrm{Sim}}(a_i, a_j) \geq t$
3. Emit (a_i, a_j)

Experimental Setup

Implementation Details

- DBMS: IBM DB2
- Inverted Index: Lucene
- Algorithms: UDFs

Datasets

- Data warehouses of a large financial company and product information data

Dataset	Columns	Tuples
RDW	504	2,661,506
OPIC	1,802	27,757,807

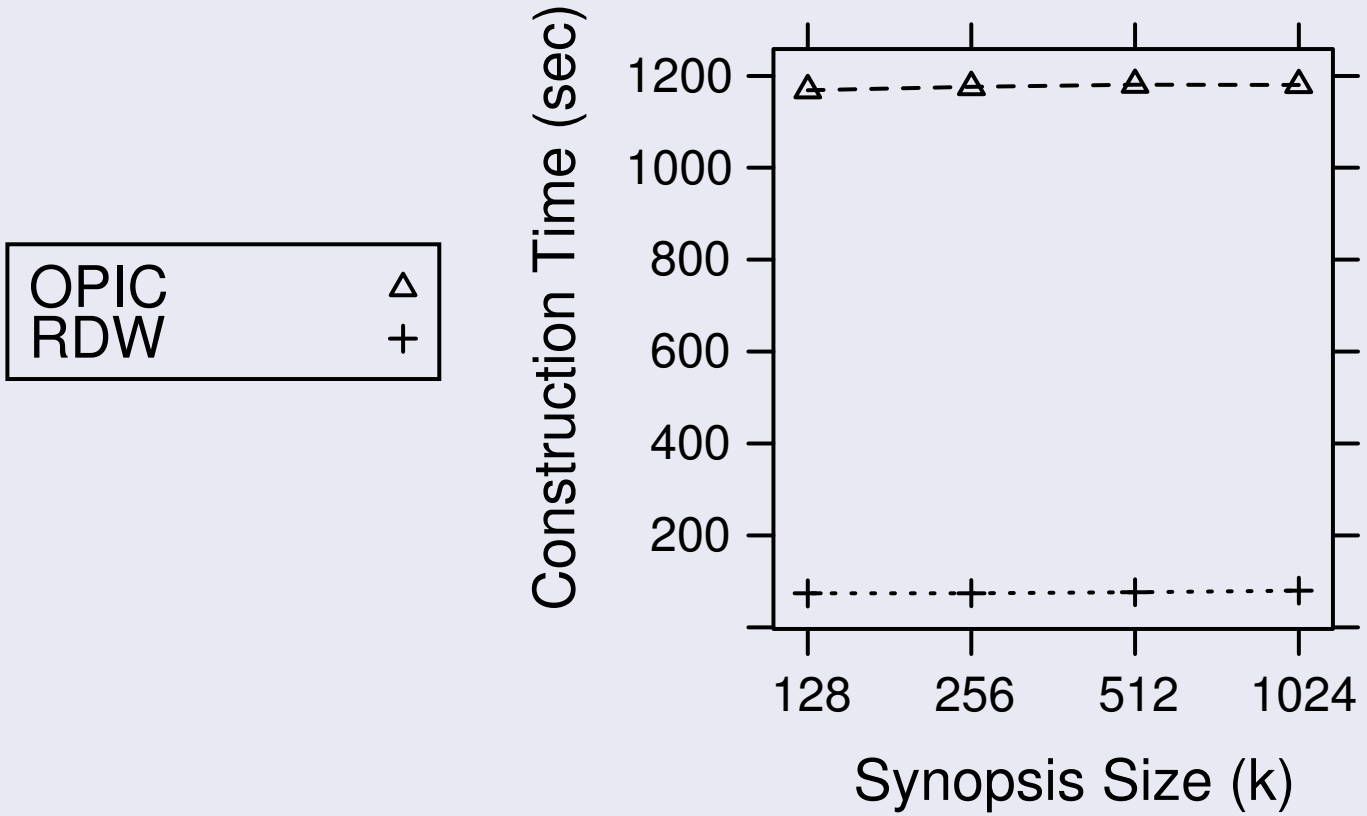
Experiments: Small Index Size

Index Size Study

Dataset	Dataset Size (MB)	k	Index Size (MB)
OPIC	5,409	128	1.27
		256	2.05
		512	3.3
		1,024	5.2
RDW	665	128	0.774
		256	1.36
		512	2.53
		1,024	4.53

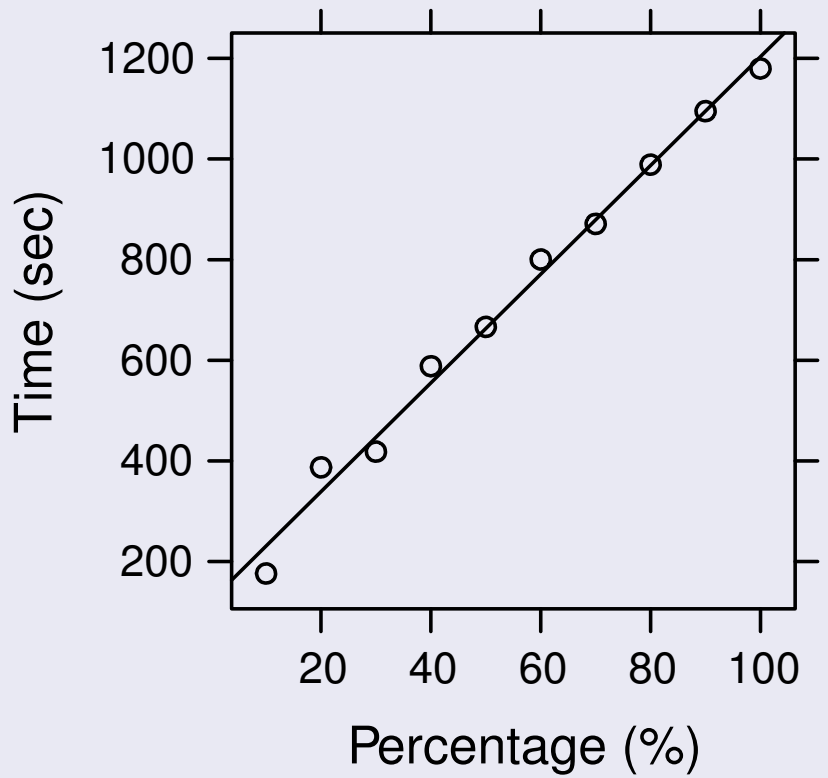
Experiments: Constant Construction Time for k

Time

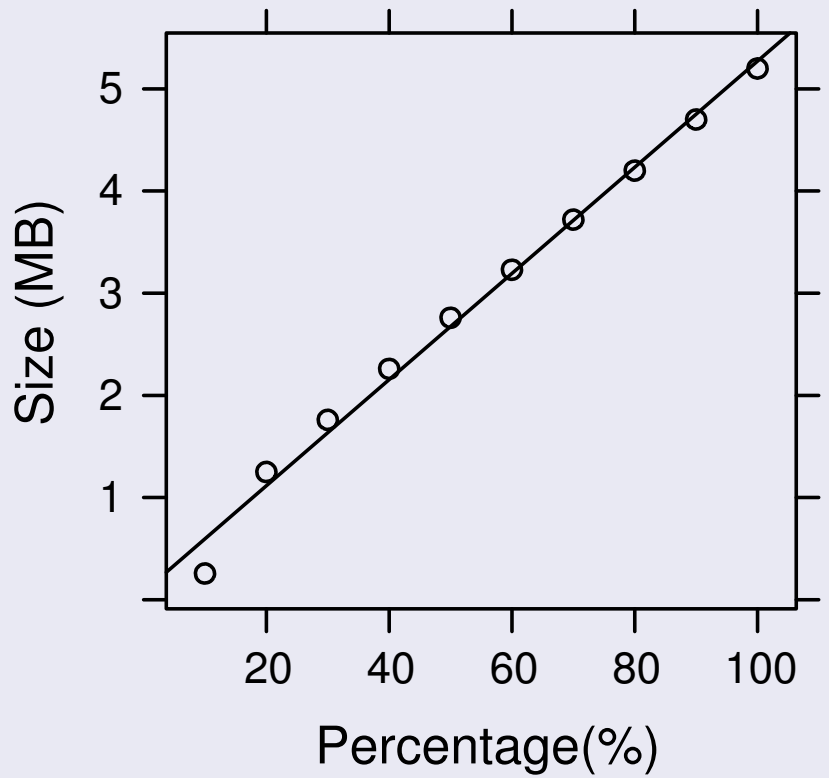


Experiments: Linear Construction Scaling

Construction Time

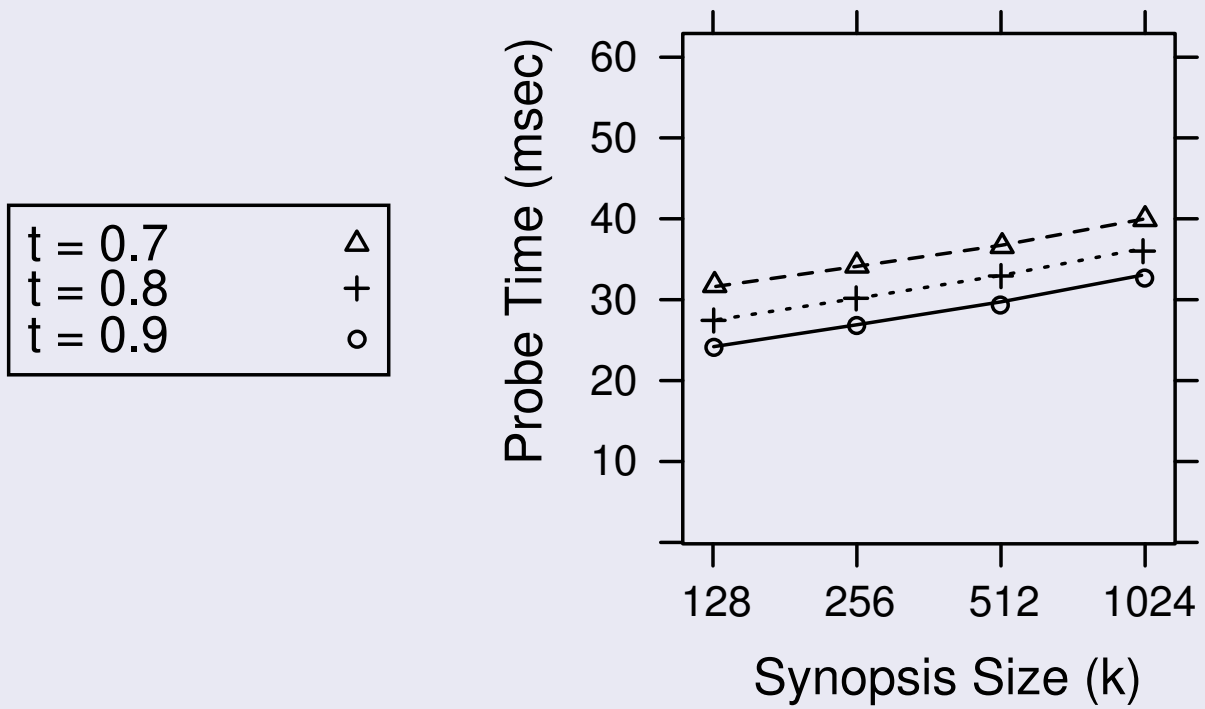


Index Size



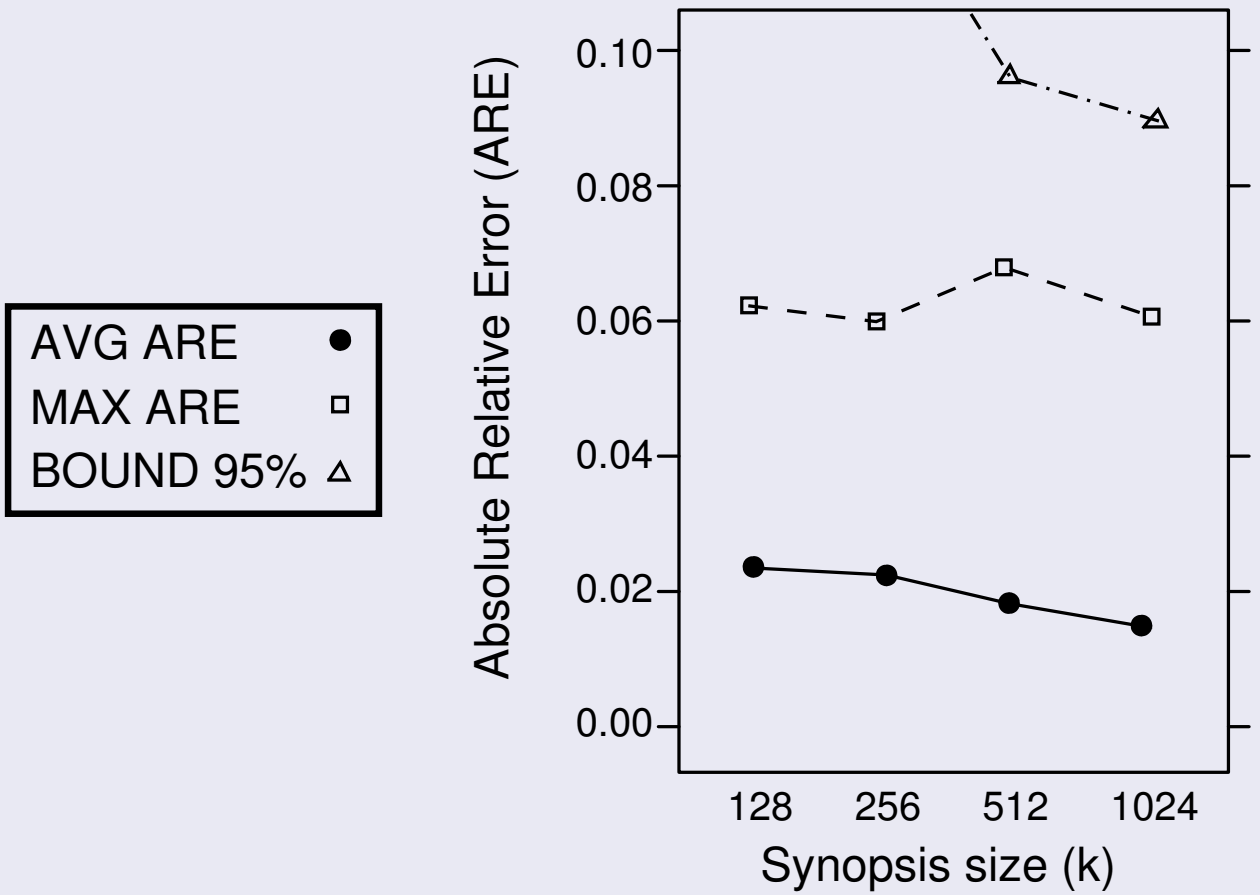
Experiments: Probe Time Increases on k and Decreases on t

Query Time



Experiments: Errors are lower than expected

ARE Study



More in Paper

- Similarity queries (OR queries is the simplest method)
- Error bounds proofs
- Comparison with other methods
- Handling XML data

Conclusions

- Use KMV synopsis and standard inverted index technology to construct a similarity index
- Small index size
- Fast queries
- Although randomized, accurate results
- Supports multiple similarity functions (focused on Jaccard)