Searching with Numbers

Rakesh Agrawal

Ramakrishnan Srikant

IBM Almaden Research Center
ragrawal@almaden.ibm.com

Motivation

- A large fraction of useful web consists of specification documents
- Current search technology is inadequate for retrieving specification documents

Specfication Documents

- Consist of <attribute name, value> pairs embedded in text
- Examples:
  - Data sheets for electronic parts
  - classified Ads
  - product catalogs

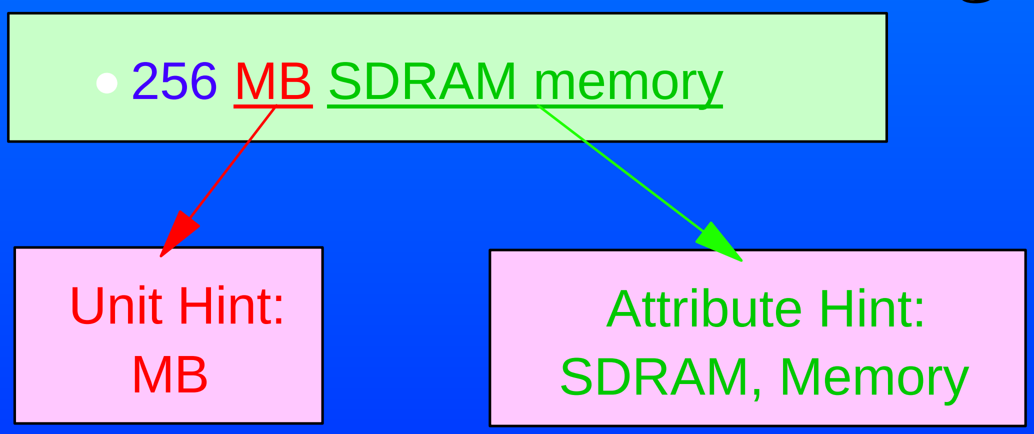
Sources of Problems

- Synonyms for attribute names and units.
  - "lb" and "pounds", but no "lbs" or "pound".
- Attribute names are often missing.
  - No "Speed", just "MHz Pentium III"
  - No "Memory", just "MB SDRAM"
- Accurate data extraction is hard, e.g. partial datasheet for Cypress CY7C225A PROM:

> - High Speed
>   - 18 ns address set-up
>   - 12 ns clock to output
> - Low Power
>   - 495 mW (commercial)
>   - 660 mW (military)

An end run!

- Use a simple regular expression extractor to get numbers
- Do simple data extraction to get hints, e.g.
  - Hint for unit: the word following the number.
  - Hint for attribute name: *k* following numbers.



- Use only numbers in the queries
  - Treat any attribute name in the query also as hint

Documents and Queries

Document $D = \{\langle n_i, H_i \rangle \mid n_i \in N, H_i \in A, 1 \leq i \leq m\}$

Query $Q = \{\langle q_i, A_i \rangle \mid n_i \in N, A_i \in A, 1 \leq i \leq k\}$

H_i and A_i are hints

D = {<256, {MB, SDRAM, Memory}>, <700, {<MHz, CPU>}}

Q = <200 MB, 750 MHz>

Document $D = \{n_i \mid n_i \in N, 1 \leq i \leq m\}$

Query $Q = \{q_i \mid n_i \in N, 1 \leq i \leq k\}$

No hints with Documents and Queries

D = {256, 700}   Q = {200, 750}

Can it work?

- Yes!!!!!
  - Provided data is non-reflective
  - Reflectivity can be computed a priori for a given data set and provides estimate of exprected accuracy.

Demo

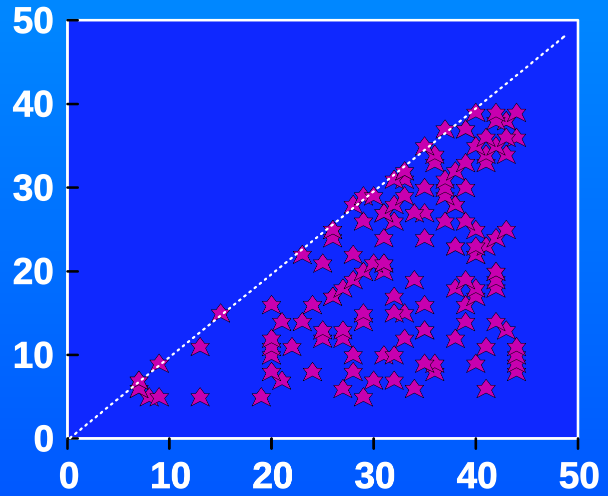
Search Engines Treat Numbers as Strings

- Search for 6798.32
  - Lunar Nutation Cycle
- Returns 2 pages on Goggle
- However, search for 6798.320 yielded no page on Google (and all other search engines)

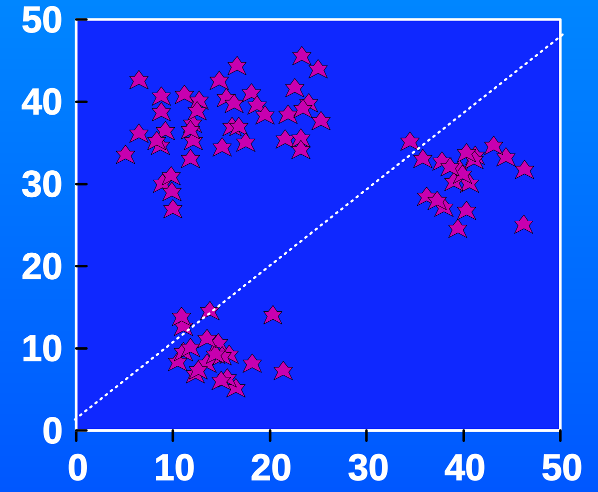
Reflectivity

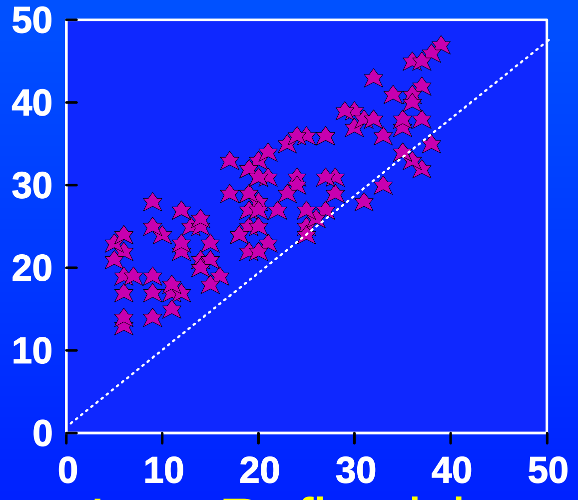
Non-reflective

<x=i, y=j> => ∄<x=j, y=i>

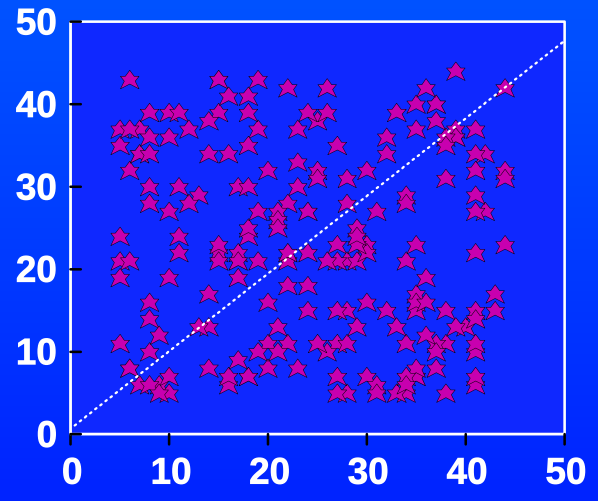


Low Reflectivity





Low Reflectivity



High Reflectivity

Non-reflectivity in real life

- Non-overlapping attributes:
  - Memory: 64 - 512 Mb, Disk: 10 - 40 Gb
- Correlations:
  - Memory: 64 - 512 Mb, Disk: 10 - 100 Gb still fine.
- Clusters

Reflectivity

- D: set of m-dimensional points
  $\underline{n}^i$: coordinates of point x^i
  $\theta(\underline{n}^i)$: number of points within distance r of $\underline{n}^i$
- Reflections(x^i): permutations of $\underline{n}^i$
  $\rho(\underline{n}^i)$: number of points in D that have at least one reflection within distance r of $\underline{n}^i$
- Reflectivity(m,r) = 1 - 1/|D| $\Sigma_{xi \in D}\ \theta(\underline{n}^i)/\rho(\underline{n}^i)$
- See the paper for reflectivity of D over k k-dimensional subspaces
- Non-reflectivity = 1- Reflectivity

Basic Idea

- Consider co-ordinates of a point
- If there is no data point at the permutations of its co-ordinates, this point is non-reflective
  - Only a few data points at the permutations of its co-ordinates => point is largely non-reflective
- Compute reflectivity as this ratio summed over all the points
  - Consider neighborhood of a point in the above calculation

Algorithms

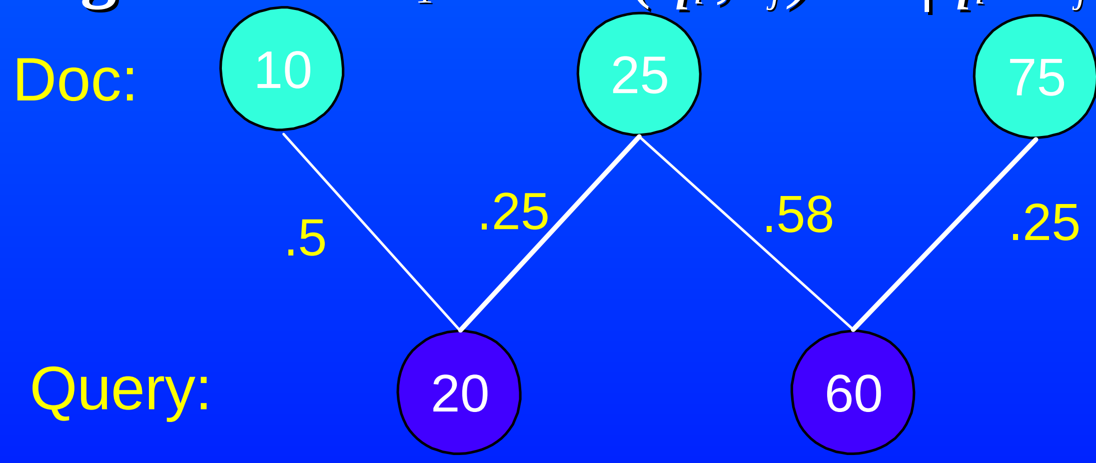
- How to compute match score (rank) of a document for a given query?
- How to limit the number of documents for which the match score is computed?

Match Score of a Document

- **Select k numbers from D yielding minimum distance between Q and D:**
  - $F(Q,D) = (\Sigma_{i=1}^{k} w(q_i, n_{ji})^p)^{1/p}$
- **Map problem to Bipartite Matching in graph G:**
  - k **source nodes: corresponding to query numbers**
  - m **target nodes: corresponding to document numbers**
  - **An edge from each source to** k **nearest targets. Assign weight** $w(q_i, n_j)^p$ **to the edge** (q_i, n_j).

Bipartite Matching

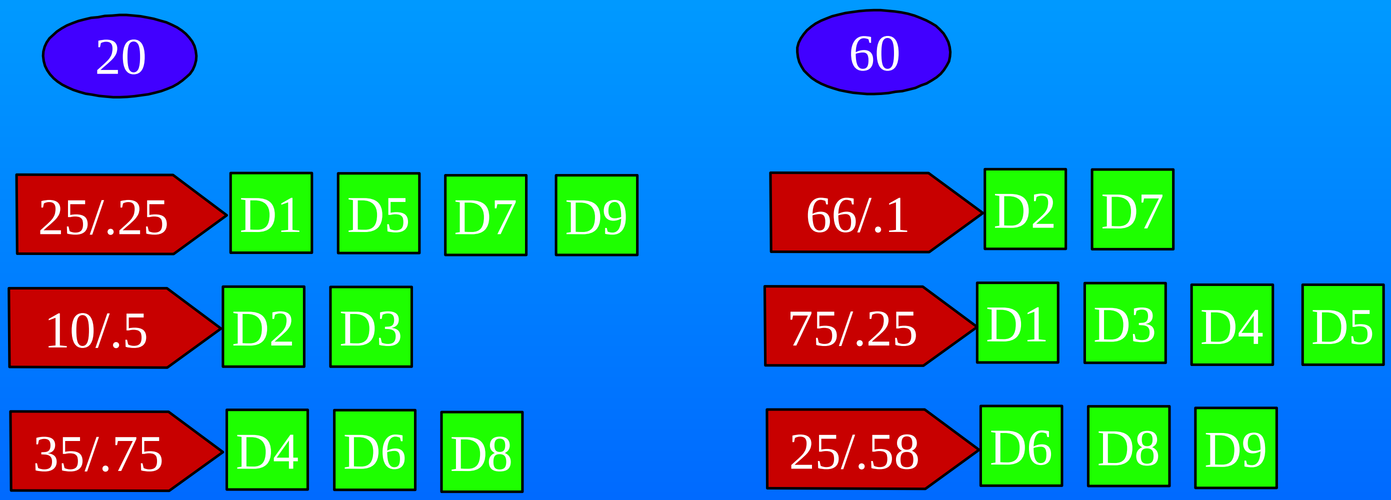
- The optimum solution to the minimum weight bipartite graph matching problem matches each number in Q with a distinct number in D such that the distance score F(Q,D) is minimized.
- The minimum score gives the rank of the document D for the Query Q.
- Assuming F to be L_1 and $w(q_i, n_j) := |q_i - n_j| / |q_i + \varepsilon|$:



Limiting the Set of Documents

- Similar to the score aggregation problem [Fagin, PODS 96]
- Proposed algorithm is an adaptation of the TA algorithm in [Fagin-Lotem-Naor, PODS 01]

Limiting the Set of Documents



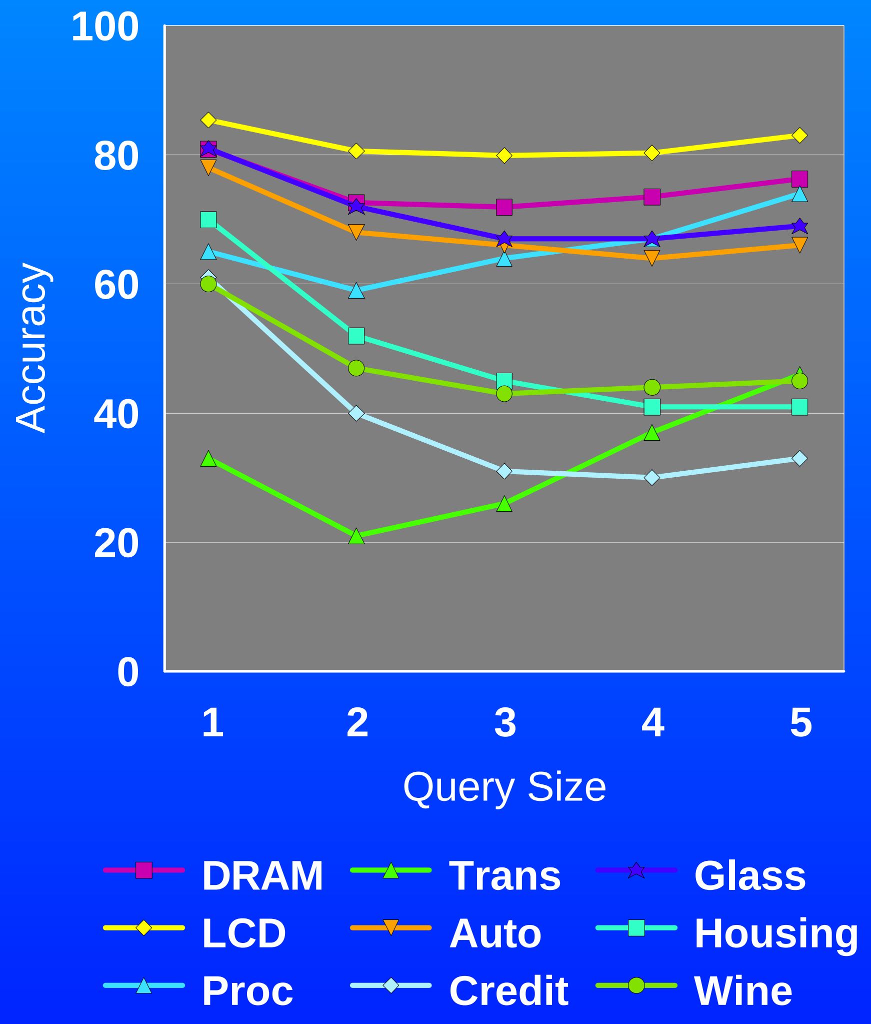
- **Make k conceptual sorted lists, one for each query term [use: documents = index(number)]**
- **Do a round robin access to the lists. For each document found, compute its distance F(D,Q)**
- **Let n_i := number last looked at for query term q_i**
  Let $\tau := (\Sigma_{i=1}^{k} w(q_i, n_i)^p)^{1/p}$
- **Halt when t documents found whose distance $\leq \tau$**
  - ► **τ is lowerbound on distance of unseen documents**

Evaluation Metric

- Benchmark: Set of answers when attribute names are precisely known in the document and query
- What fraction of the top 10 "true" answers are present in the top 10 answers when attribute names are unknown in both document and query?

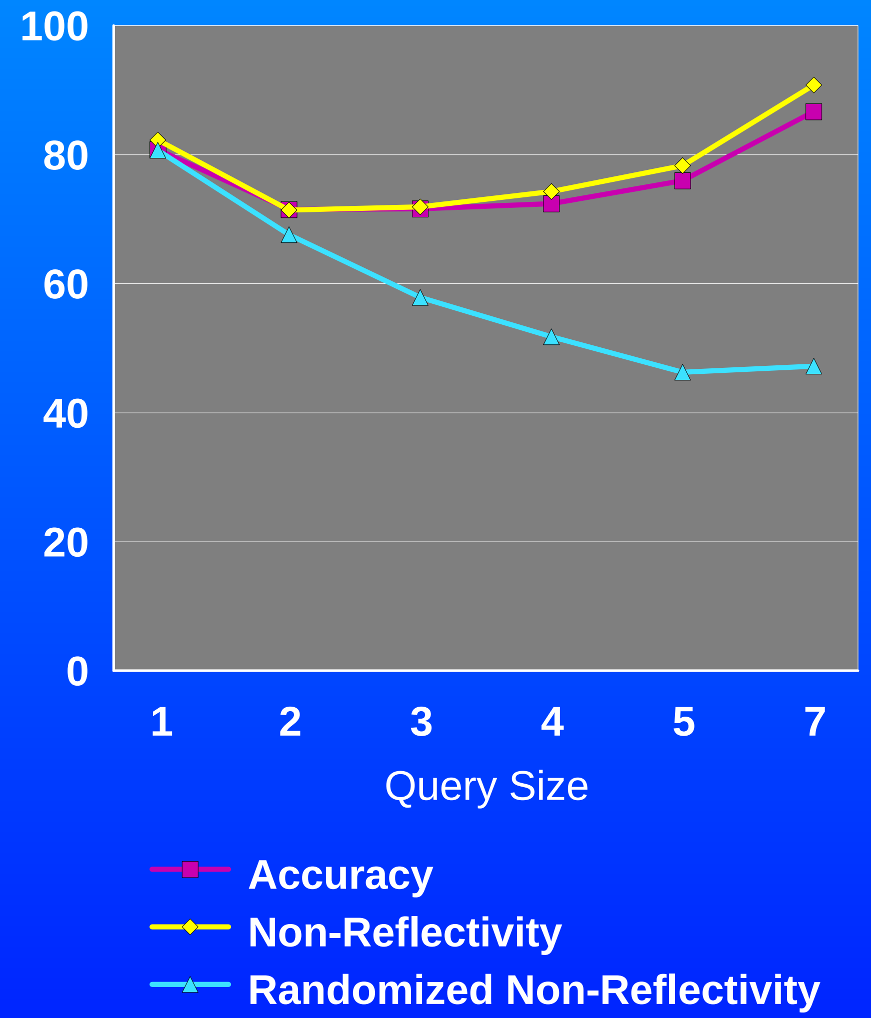
Accuracy Results

	Records	Fields
DRAM	3,800	10
LCD	1,700	12
Proc	1,100	12
Trans	22,273	24
Auto	205	16
Credit	666	6
Glass	214	10
Housing	506	14
Wine	179	14



Reflectivity estimates accuracy

- Non-reflectivity closely tracked accuracy for all nine data sets
- Non-reflectivity arises due to clustering and correlations in real data *(Randomized non-reflectivity*: value obtained after permuting values in the columns)



Incorporating Hints

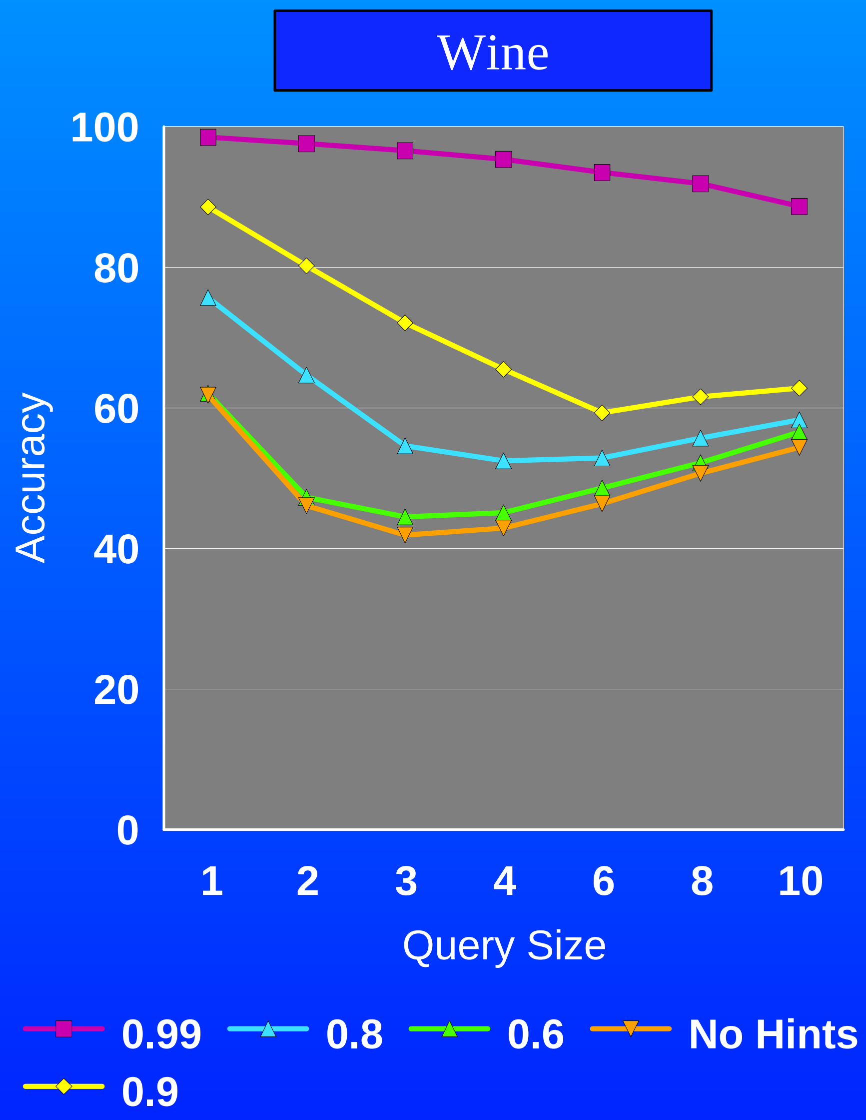
- $L_1 = \Sigma\, w(q_i, n_i) + B\, v(A_i, H_i)$
  - $v(A_i, H_i)$: distance between attribute name (or unit) for q_i and set of hints for n_i
  - B: relative importance of number match vs. name/unit match

Balance between Number Match & Hint Match

- Weightage to hints should depend on the accuracy of hints
- Use tune set to determine B on per dataset basis

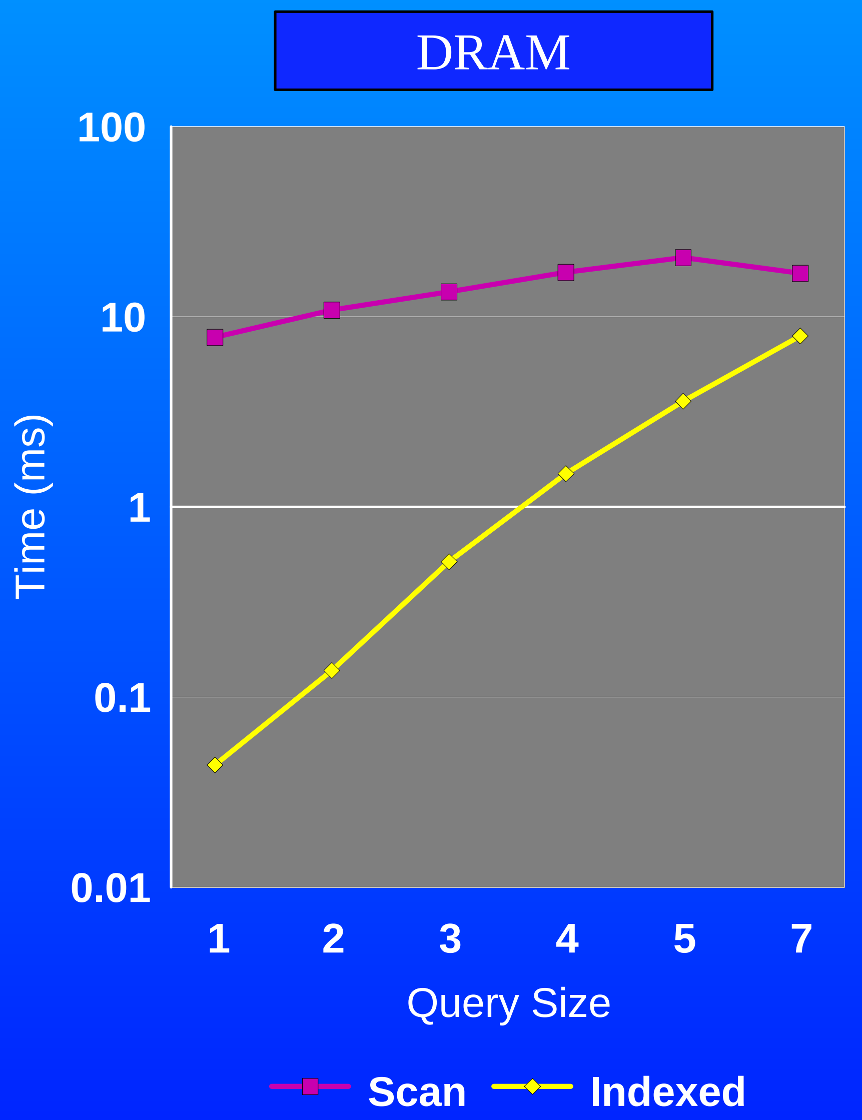
Effectiveness of Hints

- Improvement in accuracy depends on how good are hints



Effectiveness of Indexing

- 1 million docs:
  - 1 sec for qsize = 5
  - .03 sec for qsize=1



Summary

- Allows querying using only numbers or numbers + hints.
- Data can come from raw text (e.g. product descriptions) or databases.
- End run around data extraction.
  - Use simple extractor to generate hints.
- Can ascertain apriori when the technique will be effective

Future Work

- Integration with classic IR (key word search)
  - PROM speed 20 ns power 500 mW
- Extension to non-numeric values