# The MIDAS Data-Mining Project at Stanford

Jeffrey D. Ullman
*Dept. of Computer Science*
*Stanford University*

*ABSTRACT*

The following notes summarize recent research into data-mining techniques that are in progress at Stanford:

1. The *Google* search engine: beating Yahoo et al. at their own game.

2. *Query Flocks*: Generalizing association rules/market baskets in a query precompiler that uses a relational DBMS effectively.

3. *Synthesizing Knowledge from the Web*: exploiting the Web's redundancy to extract data automatically.

4. *Detecting Low-Frequency Events*: Unlike marketing, where you only care about items that lots of people buy, extracting intelligence from text usually requires looking for a *small* number of unexpected juxtapositions of terms.

**The Participants**

- *Faculty*: Rajeev Motwani, Jeff Ullman.

- *Visitors*: The following have participated, at least to the extent of coauthoring one or more papers with us: Serge Abiteboul (INRIA), Chris Clifton (MITRE), Edith Cohen (Compaq), Shinji Fujiwara (Hitachi), Arnon Rosenthal (MITRE), Frank Tompa (U. Waterloo), and Dick Tsur (Hitachi/Surromed).

- *Students*: The following are current PhD students in the group: Mayur Datar, Aristides Gionis, Taher Haveliwala, Piotr Indyk, Svetlozar Nestorov, Cheng Yang.

- *Former students now at the Google startup*: Sergey Brin, Larry Page, and Craig Silverstein.

## I. Google

A Web search engine developed by Sergey Brin and Larry Page, it uses two principles:

1. *Important pages*: Recursively defined by "a page is important if important pages link to it."

   ❖ Fixups needed for pages that have no links and "traps" (cycles of links you can't get out of).

   ❖ *Page rank* = relative importance of pages.

2. *Anchor text tells the truth*: The words in anchor text often tell more about the page linked to than the page does itself.

   ❖ Filters out "spam."

- Recently gone commercial; try `http://www.google.com` .

### Operation of Google

Like other search engines, you give Google keywords. It ranks pages by a combination of:

a) Page rank, and

b) Presence of the keywords on the page itself and in anchors linking to it.

**Example 1:** Given the keyword "baseball":

Google: The major-league baseball home page.

Excite: A page from the sports department of some newspaper in Tampa Bay, Florida.

## II. Market-Basket Analysis

The most significant successes in mining of very large databases, initiated by Rakesh Agrawal and several colleagues at IBM/Almaden.

- Given *market baskets*, i.e., collections of items bought together at a store, find sets of items that are bought unusually often together.
- Famous example: diapers and beer. Customers buying diapers in the afternoon are more likely than random to buy beer.
  - ❖ Possible explanation: they have a baby at home and are unlikely to go out to a bar.

**Measures of Significance**

1. *Support*: the items must appear in many baskets.
2. *Confidence*: the probability of one item given the others are in the basket must be high.
3. *Interest*: (statistical correlation) that probability must be significantly higher or lower than the expected probability if items were purchased at random.
4. *Causality*: purchase of one item "causes" another to be purchased.
   - ❖ Probable example: because "diapers causes beer," lowering the price of diapers will increase the sales of beer; lowering the price of beer will not increase the sales of diapers.
   - ❖ Requires expanding "Bayes nets" technology from small nets to very large-scale nets.

**Variants of Market-Basket**

1. "Basket" = document; "item" = word.
   - ❖ Find words that are unusually often found together.
   - ❖ Correlation and causality are of interest.
2. "Basket" = sentence; "item" = document.
   - ❖ Documents with many of the same sentences may indicate plagiarism.

## III. Query Flocks

- A parametrized query, normally involving aggregation and/or grouping.
- Generalizes market-basket problems:
- Two components:

1. *Generate* the queries by replacing the parameters by each of their possible values in all combinations. In market-baskets, the parameters represent possible items.
2. *Filter* each of the generated queries for some conditions. These conditions may include such properties as support and confidence from the typical market-basket analysis.

**Example 2:** Here is the market-basket problem as a query flock. Assume a relation `Baskets(basketID, item)`. In Datalog:

```
Answer(b) :-
        Baskets(b,$1) AND
        Baskets(b,$2)
```

The filter:

```
COUNT(Answer) >= c
```

**Optimizing Queries from Flocks**

- It would be nice if such a query could be converted to SQL in the straightforward way and optimized by a conventional optimizer.
- However, it appears that the trick for market-basket analysis known as the *a priori* algorithm will not be used by existing relational query optimizers.
- The "a priori" trick is to notice that if a pair of items appears in at least *c* market baskets, then each item by itself must appear in at least *c* market baskets.
  - ❖ If *c* is high enough that few items meet this threshold (e.g., 1% is often chosen), then we can eliminate many pairs from consideration before running the query.

**Solution: Use a Precompiler to Turn A Query Flock into SQL That The DBMS Can Handle**

Figure 1 suggests the architecture.

**Generalizing the "A Priori Trick"**

If we express our query flock as a conjunctive query, then we can look for suitable subsets of the subgoals that can be used to restrict the value of one or more parameters.

- The subset of subgoals must form a *safe* query.
  - ❖ Variables of the head appear somewhere in the body.
  - ❖ Variables in negated subgoals also appear in a nonnegated subgoal.

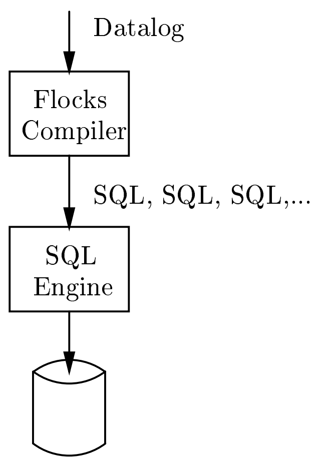

**Fig. 1.** The query-flocks architecture.

- Variables in arithmetic comparisons also appear in a nonnegated, relational subgoal.

**Example 3:** The following is an example of finding side effects of drugs. The relations are:

1. `Diagnoses(patient, disease)`: The patient has been diagnosed as having the disease.
2. `Exhibits(patient, symptom)`: The patient exhibits the symptom.
3. `Treatments(patient, medicine)`: The medicine has been prescribed for the patient.
4. `Causes(disease, symptom)`: The disease is known to cause the symptom.

Query flock: Find medicines $\$m$ and symptoms $\$s$ such that the patient is taking the medicine and exhibits the symptom, but the patient's disease is not known to cause the symptom.

```
Answer(p) :- Diagnoses(p,d) AND
        Exhibits(p,$s) AND
        Treatments(p,$m) AND
        NOT Causes(d,$s)
```

Filter: At least 20 patients for a given $\$m$ and $\$s$ to be significant.

**Choosing Preliminary Steps**

Here are some possible subqueries that could restrict our search for $\$s$ and/or $\$m$, yet are easier to compute than the full query flock.

- In each case, we can check that the subquery produces at least 20 values of $p$ and eliminate values of the parameter for which that condition is false.

(1) `Answer(p) :- Exhibits(p,$s)`

20 patients exhibit the symptom.

(2) `Answer(p) :- Treatments(p,$m)`

20 patients have been given the medicine.

```
(3) Answer(p) :- Diagnoses(p,d) AND
            Exhibits(p,$s) AND
            NOT Causes(d,$s)
```

20 patients exhibit the symptom, but their disease does not cause the symptom.

```
(4) Answer(p) :- Exhibits(p,$s) AND
            Treatments(p,$m)
```

20 patients are taking the medicine and exhibiting the symptom.

**Conclusions: Query Flocks**

- Query flocks are an approach to formulating and executing complex mining queries about a wide variety of information resources.
- Converting a query flock and its filter condition to a conventional query may be feasible under fairly broad circumstances, but the resulting queries may "break" conventional query optimizers.
- New query-optimization techniques need to be developed, and one approach is the generalization of the "a-priori" technique to non-market-basket queries.

## IV. Extraction of Knowledge

- Big idea #1: The Web is redundant. Even if you can find "the world is flat," you can find "the world is round" many more times. majority voting helps.
- Big idea #2: Data of a certain type will tend to appear in similar contexts.
- Architecture of knowledge-extraction system for Web text:
    1. Seed facts are examples of what you want.
    2. Find patterns in which several seeds appear; *pattern* may involve host, directories, HTML tags, text strings.
    3. Find data that appears in several of the selected patterns.
    4. Repeat (2) and (3) several times with each new data set.

- Figure 2 suggests how a data seed results in patterns that cover at least two seeds, and how these patterns then add to the data set by including data that is connected to at least two patterns.

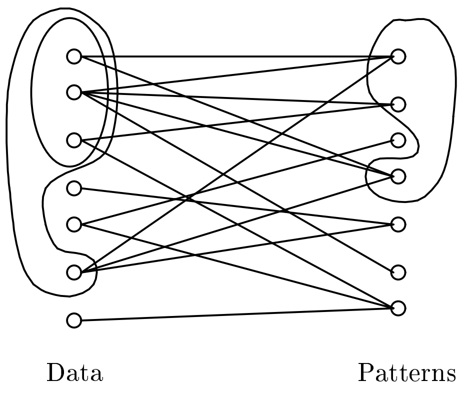

**Fig. 2.** The data/pattern construction.

**Example 4:** (Books and Authors) Sergey Brin started with 5 examples of author-title pairs (for books) and searched WebBase, the database for Google.

- Data/pattern cycle repeated four times.
- Result: 15,000 pairs, most of which were true author-titles.

**Example 5:** (Sports Teams) Frank Tompa (Univ. of Waterloo) started with the names of five US baseball teams and searched 100Mb of on-line newspaper articles from a Quebec newspaper.

- Found about 700 team names from all sports, essentially all the teams that were mentioned in the database.

## V. Detecting Low-Frequency Connections

- Market-basket data is typically mined for "high support" rules.
- But there are other applications where the interesting connections do not occur frequently.
    - ❖ Example: Bill Gates and Steve Jobs reported in a few news articles to have had lunch together. Is Apple buying Microsoft?
    - ❖ Example: Two articles have a number of common sentences. Is one derivative of the other?
    - ❖ "Yachts and caviar": rare but valuable market-basket items.

### A Model

- There is an $n \times m$ binary matrix with a low density $d$, e.g. 0.001.
- The number of rows $n$ is so large that the matrix does not fit in main memory, even if compressed; e.g., $n = 10^8$.
- The number of columns $m$ is sufficiently large (e.g., $10^4$) that $\binom{m}{2}$ items will not fit in main memory; i.e., we cannot scan the matrix and keep data about each *pair* of columns. However, $O(m)$ items will fit in memory.

### Desiderata

- Correlated columns.
- Causality between columns.

### Approach

- It is too hard to compute correlations among all pairs of columns.
- Causality is even harder; $O(nm^3)$ at best.
- Thus, find simple surrogates for these quantities that can be computed in one or a few passes through the data.

### Problems

1. Find *similar columns* = two columns that have 1's in almost the same rows.
2. *Contained columns*: column $B$ has 1 in almost all the rows where column $A$ has 1.

- Finding *most*, rather than all, instances is OK.

$$\boxed{\begin{array}{cc} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{array}} \qquad \begin{array}{cc} \boxed{\begin{array}{cc} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{array}} \\ A \quad B \end{array}$$

**Fig. 3.** Similar and contained columns.

### Min-Hashing for Similar Columns

(Due to Rajeev Motwani and Edith Cohen.)

1. Order rows at random. "Hash" each column to the number of first row in which it has 1.
   - ❖ But clever implementation avoids really permuting rows.
2. Repeat several times to get a *signature* (list of integers) for each column.

- Similar columns will, with high probability, have similar signatures.
  - ❖ If columns $A$ and $B$ agree in 90% of their 1's, then probability = 90% that the first row in which column $A$ has a 1 will also be the first row in which $B$ has 1.
- Signatures are small enough that the comparisons can be done in main memory.

**Example 6:** Suppose we first order the rows as given in Fig. 3, and then in the reverse order. The signatures for the four columns in the earlier figure are: $(2, 7)$, $(2, 9)$, $(3, 9)$, and $(1, 9)$.

### $k$-Min Hashing

Instead of $k$ independent min-hash operations, use as the signature the list of the first $k$ rows in which each column has a 1.

- Similar columns will, with high probability, have similar signatures.
- Again, careful implementation avoids actually randomizing the order of the rows.

## VI. Summary

- Web search can be made much more accurate by techniques from Google and probably by techniques to be invented.
- General-purpose data-mining tools are a possibility. The architecture of using a specialized preprocessor to translate to SQL is viable.
- The possibility that knowledge may be created by relatively simple means through large-scale Web queries is open.
- Pushing the limits of association-rule (market-basket) mining to low-support events is also promising.

## VII. Acknowledgements

The work described here was partially supported by NSF grants CCR–93–57849 and IIS–98–11904, and a grant of Hitachi Corp.

## VIII. References

- The MIDAS home page: `http://www-db.stanford.edu/midas/midas.html`.
- Google: `http://www.google.com` .
- Query flocks: D. Tsur, J. D. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal, "Query flocks: a generalization of association-rule mining," *VLDB, 1998*, and `http://www-db.stanford.edu/papers/flocks.ps`.
- Discovering books and authors: S. Brin, "Extracting patterns and relations from the World-Wide Web," `http://www-db.stanford.edu/~sergey/extract.ps` .
- Low-support/high confidence: E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang, "Finding Interesting Associations without Support Pruning," unpublished memorandum, Dept. of CS, Stanford Univ., Feb., 1999.