An Overview of WHIRL, or
Matching Almost Anything Quickly: How and Why

William Cohen
Carnegie Mellon University
My background

- Rutgers, 1986-1990: Explanation based learning (learning from examples and prior knowledge)
- AT&T Bell Labs/AT&T Research, 1990-2000:
  - Learning logic programs/description logics
  - What representations work well for learners?
  - Scalable rule learning (RIPPER system)
  - Text categorization/information extraction
  - WHIRL (this talk)
My background

• WhizBang Labs, April 2000-May 2002
  – More text categorization, IE, matching
  – Improving text classifiers by recognizing structure of “index pages” (Cohen, NIPS-2002)

• Carnegie Mellon’s CALD center: last year
  – Information extraction from on-line biomedical publications: subcellular location information from text and images
  – Evaluating aspect retrieval systems
  – Privacy issues related to data integration
Grand visions for information access

- The semantic web: a world-wide database as widely distributed, fluid, and easy to extend as the web.
- Peer-to-peer databases: exchange and query structured information across thousands of client/server machines.
- Large-scale information extraction: extract a database of structured information from thousands or millions of documents.
- Large-scale information integration, e.g. across deep-web sources: make thousands of databases look like one.
- The “world wide knowledge base”: make the existing web look like a single huge knowledge base.
A common thread: merging structured data

Notice: planning people see a planning problem, learning people see a learning problem, programming language people see a language problem, ... 

The real problem is representation.
What’s the research problem?

Clarification: There are two kinds of information systems:

1. Search engines, clipping services, hypertext, … store and deliver potentially relevant documents to a user.
   - Easy to handle information from diverse sources.

2. Databases, KR systems, … store facts and perform deduction on behalf of a user.
   - Very hard to handle information from diverse sources.
What’s the research problem?

We don’t know how to reason with information that comes from many different, autonomous sources.
all mallards are waterfowl + a picture of a mallard = a waterfowl

duck.jpg is duck.jpg is

<table>
<thead>
<tr>
<th>Order</th>
<th>Species</th>
<th>Species</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>waterfowl</td>
<td>mallard</td>
<td>robin</td>
<td>robin.jpg</td>
</tr>
<tr>
<td>waterfowl</td>
<td>bufflehead</td>
<td>mallard</td>
<td>duck.jpg</td>
</tr>
<tr>
<td>raptor</td>
<td>osprey</td>
<td>osprey</td>
<td>hawk.jpg</td>
</tr>
<tr>
<td>raptor</td>
<td>bald eagle</td>
<td>penguin</td>
<td>tweety.jpg</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

= 

<table>
<thead>
<tr>
<th>Order</th>
<th>Species</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td>waterfowl</td>
<td>mallard</td>
<td>duck.jpg</td>
</tr>
<tr>
<td>raptor</td>
<td>osprey</td>
<td>hawk.jpg</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
mallards are duck.jpg
found in New Jersey a mallard

Deduction enables modularity.
Why deduction requires co-operation

?- nj_bird(X),image(X,File).
image(mallard,’duck.jpg’). image(american_robin,’robin.jpg’). . . .

The providers of the nj_bird and image facts have to agree on:

• predicate names and argument positions (schema);
• taxonomic information;
• formal names (OIDs) for every entity they describe;
• . . .
Deduction without co-operation

If information providers don’t co-operate, then a “mediator” program must translate:

’robin’ → ’american_robin’

How hard is it to determine if two names refer to the same thing?
<table>
<thead>
<tr>
<th>Humongous Entertainment</th>
<th>Microsoft</th>
<th>Microsoft Kids Microsoft/Scholastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headbone Interactive</td>
<td></td>
<td>American Kestrel Eurasian Kestrel</td>
</tr>
<tr>
<td>The Lion King: Storybook</td>
<td>Lion King Animated StoryBook</td>
<td>Canada Goose Goose, Aleutian Canada</td>
</tr>
<tr>
<td>Disney’s Activity Center, The Lion King Activity Center</td>
<td>Mallard</td>
<td>Mallard, Mariana</td>
</tr>
</tbody>
</table>
### Conclusion

name-coreference is an AI-complete problem.
What’s the research problem?

We need a general means for integrating formally unconnected knowledge bases.

We must exploit these facts: the individual KB’s model the same real world, and communicate with the same users.
The WHIRL approach

Key points:

• Use informal names and descriptions as object identifiers.

• Use techniques from information retrieval (IR) to guess if two descriptions refer to the same object.

• Use soft (∼ probabilistic) reasoning for deduction.

Formal reasoning methods over informally identified objects.
Overview of WHIRL

• WHIRL (Word-based Heterogeneous Information Representation Language) is somewhere between IR systems (document delivery) and KR systems (deduction).

• Outline:
  – Data model: how information is stored.
  – WHIRL query language
  – Accuracy results
  – Key ideas for implementation
  – Efficiency results
  – More results and conclusions
Background: Information retrieval

Ranked retrieval: (e.g., Altavista, Infoseek, . . . ) given a query $Q$, find the documents $d_1, \ldots, d_r$ that are most similar to $Q$.

Similarity of $d_i$ and $d_j$ is measured using set of terms $T_{ij}$ common to $d_i$ and $d_j$:

$$SIM(d_i, d_j) = \sum_{t \in T_{ij}} weight(t, d_i) \cdot weight(t, d_j)$$

- A term is a single word (modulo stemming, . . . )
- Heuristic: make $weight(t, d)$ large if $t$ is frequent in $d$, or if $t$ is rare in the corpus of which $d$ is an element.
**Background: Information retrieval**

**Similarity** of $d_i$ and $d_j$ is measured using set of terms $T_{ij}$ common to $d_i$ and $d_j$:

$$SIM(d_i, d_j) = \sum_{t \in T_{ij}} \text{weight}(t, d_i) \cdot \text{weight}(t, d_j)$$

- **Heuristic**: make $\text{weight}(t, d)$ large if $t$ is frequent in $d$ (TF), or if $t$ is rare in the corpus of which $d$ is an element (IDF).

- **Example**: if the corpus is a list of company names:
Background: Information retrieval

It’s notationally convenient to think of a document $d_i$ as a long, sparse vector, $v_i$.

If $\vec{v}_i = \langle v_{i,1}, \ldots, v_{i,|T|} \rangle$, $v_{i,t} = \text{weight}(t, d_i)$, and $||v_i|| = 1$:

\[
\text{SIM}(d_i, d_j) = \sum_{t \in T} \text{weight}(t, d_i) \cdot \text{weight}(t, d_j) = \vec{v}_i \cdot \vec{v}_j
\]

Also, $0 \leq \text{SIM}(d_i, d_j) \leq 1$. 
Effectiveness of the TF-IDF “vector space” representation

![Graph showing the effectiveness of the TF-IDF representation compared to other algorithms. The graph plots the accuracy of Ripper against the accuracy of the Nearest-Neighbour Algorithm. The linear relationship is depicted by the line y=x.]
<table>
<thead>
<tr>
<th>Cinema</th>
<th>Movie</th>
<th>Show Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roberts Theaters</td>
<td>Brassed Off</td>
<td>7:15 - 9:10</td>
</tr>
<tr>
<td>Chatham</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Berkeley Cinema</td>
<td>Hercules</td>
<td>4:15 - 7:30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sony Mountainside Theater</td>
<td>Men In Black</td>
<td>7:40 - 8:40 - 9:30 - 10:10</td>
</tr>
</tbody>
</table>

Each \( \vec{v}_i, \vec{w}_i \) is a document vector. Each fact has a score \( s \in [0, 1] \).

### Movie Review

<table>
<thead>
<tr>
<th>Movie</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Men in Black, 1997</td>
<td>((***)) One of the biggest hits of ...</td>
</tr>
<tr>
<td>Face/Off, 1997</td>
<td>((** \frac{1}{2} )) After a slow start, ...</td>
</tr>
<tr>
<td>Space Balls, 1987</td>
<td>((\frac{1}{2})) Not one of Mel Brooks’ best efforts, this spoof ...</td>
</tr>
</tbody>
</table>

\[
\vec{v}_{MIB} = \langle \ldots, \vec{v}_{black}, \ldots, \vec{v}_{in}, \ldots, \vec{v}_{men}, \ldots \rangle \\
\vec{w}_{MIB97} = \langle \ldots, \vec{w}_{black}, \ldots, \vec{w}_{in}, \ldots, \vec{w}_{men}, \ldots, \vec{w}_{1997}, \ldots \rangle \\
w_{1997} \approx 0 \implies \text{sim}(\vec{v}_{MIB}, \vec{w}_{MIB97}) \approx 1
Queries in WHIRL

- **Syntax:** \( \text{WHIRL} = \) (similarity)
  Prolog – function symbols – recursion – negation + \( X \sim Y \)

- **Semantics** (details in Cohen, SIGMOD98):
  - A ground formula gets a score \( s \in [0, 1] \)
  - \( \text{Score}(p(a_1, \ldots, a_k)) = s \) for DB literals.
  - \( \text{Score}(a \sim b) = \text{SIM}(a, b) \) for similarity literals.
  - \( \text{Score}(\phi \land \psi) = \text{Score}(\phi) \cdot \text{Score}(\psi) \).
  - \( \text{Score}(\phi \lor \psi) = 1 - (1 - \text{Score}(\phi))(1 - \text{Score}(\psi)) \)
  - Answer to a query \( Q \) is an ordered list of the \( r \) substitutions \( \theta_1, \ldots, \theta_r \) that give \( Q\theta_i \) the highest scores.
    (User provides \( r \)).
Queries in WHIRL

- **Syntax**: WHIRL = unions of conjunctive SQL queries + X~Y

- **Semantics** (details in Cohen, SIGMOD98):
  
  SELECT $r_{i_1}.f_{i_1}, r_{i_2}.f_{i_2}, \ldots$
  FROM $R_1$ as $r_1, R_2$ as $r_2, \ldots, R_k$ as $r_K$
  WHERE $\phi(R_1, \ldots, R_K)$
  - Answer is an ordered list of tuples.
  - A tuple is defined by binding each $r_i$ to a tuple $t_j = \langle a_{j,1}, \ldots, a_{j,\ell} \rangle \in R_i$, and then SELECT-ing the appropriate fields.
  - Answer: the $n$ tuples with max score for $\phi$ (and $t_j$’s).
- \( \text{Score}(a \sim b) = SIM(a, b) \) for similarity literals.
- \( \text{Score}(\phi \land \psi) = \text{Score}(\phi) \cdot \text{Score}(\psi) \).
- \( \text{Score}(\phi \land \psi) = \text{Score}(\phi) \cdot \text{Score}(\psi) \).
- \( \text{Score}(\phi \lor \psi) = 1 - (1 - \text{Score}(\phi))(1 - \text{Score}(\psi)) \)
- Score for \( r_i \rightarrow t_j \) is taken from DB score for \( t_j \).
- Final score: \( \text{Score}(\phi) \cdot \Pi_i \text{Score}(r_i \rightarrow t_j) \)
Sample WHIRL queries

Standard ranked retrieval:

“find reviews of sci-fi comedies”.

?- review(Title, Text) ∧ Text~“sci-fi comedy”
FROM review as r SELECT * WHERE r.text~“sci-fi comedy”

(score 0.22): $\theta_1 = \{\text{Title}/\vec{w}_{MIB97}, \text{Text}/\vec{w}_{R1}\}$
(score 0.19): $\theta_2 = \{\text{Title}/\vec{w}_{SB}, \text{Text}/\vec{w}_{R4}\}$
(score 0.13): $\theta_2 = \ldots$
Sample WHIRL queries

Standard DB queries: “find reviews for movies playing in Mountainside” (assume single-term “movie IDs” in DB)

?- review(Id1,T1,Text) ∧ listing(C,Id2,T2,Time)
∧ Id1~Id2 ∧ C~“Sony Mountainside Theater”
FROM review as r, listing as l
SELECT *
WHERE r.id=l.id AND l.cinema~“Sony Mountainside Theater”"

(score 1.00): \( \theta_1 = \{ \text{Id1}/\vec{v}_{93}, \text{Id2}/\vec{w}_{93}, \text{Text}/\vec{w}_{R1}, \ldots \} \)
(score 1.00): \( \theta_2 = \ldots \)

| Cinema   | Id | Movie       | Time | | Id       | Movie                     | Review   |
|----------|----|-------------|------| |----------|---------------------------|----------|
| ...      | 21 | Brassed Off | ...  | | 93       | Men in Black, 1997        | ...      |
| Sony ... | 93 | Men In Black| ...  | | 44       | Face/Off, 1997             | ...      |
Sample WHIRL queries

Mixed queries: “where is [Men in Black] playing?”

?- review(Id1,T1,Text) ∧ listing(C,Id2,T2,Time)
   ∧ Id1∼Id2 ∧ Text∼“sci-fi comedy with Will Smith”
FROM review as r,listing as l SELECT *
WHERE r.id=l.id AND r.text∼“sci-fi comedy with Will Smith”

(score 0.22): \( \theta_1 = \{ Id1/\vec{v}_{93}, Id2/\vec{w}_{93}, Text/\vec{w}_{R1}, \ldots \} \)
(score 0.13): \( \theta_2 = \ldots \)

<table>
<thead>
<tr>
<th>Cinema</th>
<th>Id</th>
<th>Movie</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>21</td>
<td>Brassed Off</td>
<td>...</td>
</tr>
<tr>
<td>Sony ...</td>
<td>93</td>
<td>Men In Black</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Movie</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>Men in Black, 1997</td>
<td>...</td>
</tr>
<tr>
<td>44</td>
<td>Face/Off, 1997</td>
<td>...</td>
</tr>
</tbody>
</table>
A realistic situation

<table>
<thead>
<tr>
<th>Cinema</th>
<th>Movie</th>
<th>Show Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roberts Theaters Chatham</td>
<td>Brassed Off</td>
<td>7:15 - 9:10</td>
</tr>
<tr>
<td>Berkeley Cinema</td>
<td>Hercules</td>
<td>4:15 - 7:30</td>
</tr>
</tbody>
</table>
| Sony Mountainside Theater  | Men In Black | 7:40 - 8:40 -  
                              |               | 9:30 - 10:10   |

With real Web data, there will be no common ID fields, only informal names.

<table>
<thead>
<tr>
<th>Movie</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Men in Black, 1997</td>
<td>(** *) One of the biggest hits of ...</td>
</tr>
<tr>
<td>Face/Off, 1997</td>
<td>(** (\frac{1}{2}) ) After a slow start, ...</td>
</tr>
<tr>
<td>Space Balls, 1987</td>
<td>((\frac{1}{2}) *) Not one of Mel Brooks’ best efforts, this spoof ...</td>
</tr>
</tbody>
</table>
Sample WHIRL queries

“Similarity” joins: “find reviews of movies currently playing”

?- review(Title1,Text) ∧ listing(_,Title2,Time) ∧ Title1∼Title2
FROM review as r,listing as l SELECT *
WHERE r.title∼l.title

(score 0.97): $\theta_1 = \{ \text{Title1}/\vec{v}_{MIB}, \text{Title2}/\vec{w}_{MIB97}, \ldots \}$
(Men in Black) (Men in Black, 1997)

... 

(score 0.41): $\theta_2 = \{ \text{Title1}/\vec{v}_{BO}, \text{Title2}/\vec{w}_{FO}, \ldots \}$
(Brassed Off) (Face/Off)

...
How well do similarity joins work?

?- top500(X), hiTech(Y), X∼Y

FROM top500,hiTech SELECT * WHERE top500.name∼hiTech.name

top500:
Abbott Laboratories
Able Telcom Holding Corp.
Access Health, Inc.
Acclaim Entertainment, Inc.
Ace Hardware Corporation
ACS Communications, Inc.
ACT Manufacturing, Inc.
Active Voice Corporation
Adams Media Corporation
Adolph Coors Company
...

hiTech:
ACC CORP
ADC TELECOMMUNICATION INC
ADELPHIA COMMUNICATIONS CORP
ADT LTD
ADTRAN INC
AIRTOUCH COMMUNICATIONS
AMATI COMMUNICATIONS CORP
AMERITECH CORP
APERTUS TECHNOLOGIES INC
APPLIED DIGITAL ACCESS INC
APPLIED INNOVATION INC
...

30
Sample company-name pairings

WHIRL output on business.html
Evaluating similarity joins

- **Input**: query
- **Output**: ordered list of documents

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>Precision at $K$: $G_K/K$</th>
<th>Recall at $K$: $G_K/G$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>$a_1$</td>
<td>$b_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>$a_2$</td>
<td>$b_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>$a_3$</td>
<td>$b_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✓</td>
<td>$a_4$</td>
<td>$b_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>✓</td>
<td>$a_5$</td>
<td>$b_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>✓</td>
<td>$a_6$</td>
<td>$b_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>×</td>
<td>$a_7$</td>
<td>$b_7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>✓</td>
<td>$a_8$</td>
<td>$b_8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>✓</td>
<td>$a_9$</td>
<td>$b_9$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>×</td>
<td>$a_{10}$</td>
<td>$b_{10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>×</td>
<td>$a_{11}$</td>
<td>$b_{11}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>✓</td>
<td>$a_{12}$</td>
<td>$b_{12}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$G$: # good pairings

$G_K$: # good pairings in first $K$
Evaluating similarity joins

• Pick relations \( p, q \) with \( > 2 \) plausible keys
• Perform "similarity join" using first key field
• Mark a pairing correct ("relevant") if secondary key matches
• Compute precision and recall over first 1000 rankings
• Examples
  – Business: company name, web site
  – Animals: common name, scientific name
  – etc
Evaluating similarity joins

![Graph showing precision and recall for different categories including Business and Animals.]
Evaluating WHIRL queries

Additional experiments:

- Repeat with more datasets from more domains.
  - Average precision ($\approx$ area under precision-recall curve) ranges from 85% to 100% over 13 joins in 6 domains.

- Repeat for more complex join queries.
  - Average precision drops from 94% for 2-way joins to 90% for 5-way joins (averaged over many queries in one domain).

- Evaluate other things to do with WHIRL.

- How can you implement WHIRL efficiently?
An efficient implementation

Key ideas for current implementation:

• Central problem: given $Q$, find best substitution.
  – Currently, using $A^*$ search.

• Search space: partial substitutions.
  e.g., for “?- r(X),s(Y),X$\sim$Y”, possible state is \{\(X = \vec{x}\}\}.

• Key operator: when $Q$ contains “\(\vec{x}$\$\sim$Y”’, find good candidate bindings for $Y$ quickly.
  – Use inverted indices.
An efficient implementation

- Key step: state is a substitution $\theta$, $Q\theta$ contains “$s(Y), \vec{x} \sim Y$”. Need to find good candidate bindings for $Y$ quickly.
  1. Pick some term $t$ with large weight in $\vec{x}$.
  2. Use inverted index to get

$$I_{t,s,1} = \{ \vec{y} : s(\vec{y}) \in DB \text{ and } y_t > 0 \}$$

- To compute heuristic value of state, use fact that

$$score(\vec{x} \sim Y) \leq \max_{\vec{z} \in I_{t,s,1}} (\sum_t x_t \cdot z_t) \leq \sum_t x_t \cdot (\max_{\vec{z} \in I_{t,s,1}} z_t)$$

- Indexing and bounds well-known in IR
  (Buckley-Lewitt, Turtle-Flood’s $maxscore$ alg)
An efficient implementation

- Controlled experiments: for 2-relation soft joins WHIRL is:
  - about 20x faster than naive use of inverted indices
  - from 4-10x faster than Turtle-Flood’s maxscore

- In practice, for typical queries to two real web-based integration systems:
  - Game domain: 15 sites, 23k+ tuples, avg 0.3sec/query
  - Birding domain: 35 sites, 140k+ tuples, avg 0.2sec/query
The extraction problem

Sometimes it’s difficult to extract even an informal name from its context:

- Fox Interactive has a fully working demo version of the Simpsons Cartoon Studio. (Win and Mac)
- Vividus Software has a free 30 day demo of Web Workshop (web authoring package for kids!) Win 95 and Mac
- Scarlet Tanager (58kB) *Piranga olivacea*. New Paltz, June 1997. “...Robin-like but hoarse (suggesting a Robin with a sore throat).” (Peterson) “...a double-tone which can only be imitated by strongly humming and whistling at the same time.” (Mathews)
### The extraction problem

Idea: use text *without* trying to extract names.

?- paragraph(X), name(Y), X\sim Y

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
<th>Name</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>80.26</td>
<td>Ubi Software has a demo of Amazing Learning Games with Rayman.</td>
<td>Amazing Learning Games with Rayman</td>
<td>✓</td>
</tr>
<tr>
<td>78.25</td>
<td>Interplay has a demo of Mario Teaches Typing. (PC)</td>
<td>Mario Teaches Typing</td>
<td>✓</td>
</tr>
<tr>
<td>75.91</td>
<td>Warner Active has a small interactive demo for Where’s Waldo at the Circus and Where’s Waldo? Exploring Geography (Mac and Win)</td>
<td>Where’s Waldo? Exploring Geography</td>
<td>✓</td>
</tr>
<tr>
<td>74.94</td>
<td>MacPlay has demos of Marios Game Gallery and Mario Teaches Typing. (Mac)</td>
<td>Mario Teaches Typing</td>
<td>✓</td>
</tr>
<tr>
<td>71.56</td>
<td>Interplay has a demo of Mario Teaches Typing. (PC)</td>
<td>Mario Teaches Typing 2</td>
<td>×</td>
</tr>
</tbody>
</table>
Deduction without extraction
Movie 1: full review (no extraction).
Movie 2: movie name, cinema name & address, showtimes.
More uses of WHIRL: Classification?

review(“Putt-Putt Travels Through Time”, url1).
category(“Putt-Putt’s Fun Pack”, “adventure”).
category(“Time Traveler CD”, “history”).
...
“find me reviews of adventure games”
\[ v(Url) \leftarrow \]
\[ \text{review}(\text{Game1},Url) \land \text{category}(\text{Game2},\text{Cat}) \]
\[ \land \text{Game1} \sim \text{Game2} \land \text{Cat} \sim \text{“adventure”} \]

To answer this query, WHIRL guesses the class “adventure” based on similarities between names.
More uses of WHIRL: Classification

category(Cat) ← test(X) ∧ train(Y,Cat) ∧ X∼Y

• Here train contains a single unclassified example, and test contains a set of training examples with known categories. (from Cohen&Hirsh, KDD-98)

• WHIRL here performs a sort of K-NN classification.
  1. Find $r$ best bindings for $X,Y,Cat$
  2. Combine evidence using noisy-or:
     \[ \text{Score}(\phi \land \psi) = \text{Score}(\phi) \cdot \text{Score}(\psi) \]
Using WHIRL for Classification

- Created nine representative datasets using data from Web.
- All instances were short “names”
  - *company name*: inst="National City Corporation", class="Banks–Midwest"
  - Also bird names, Web page titles, ...
- # classes ranged from 6 to 228, #instances ranged from \(\approx 300\) to \(\approx 3000\).
Benchmark classification problems

<table>
<thead>
<tr>
<th>problem</th>
<th>#train/#test</th>
<th>#classes/#terms</th>
<th>text-valued field/label</th>
</tr>
</thead>
<tbody>
<tr>
<td>memos</td>
<td>334/10cv</td>
<td>11/1014</td>
<td>document title/category</td>
</tr>
<tr>
<td>cdroms</td>
<td>798/10cv</td>
<td>6/1133</td>
<td>CDRom game name/category</td>
</tr>
<tr>
<td>birdcom</td>
<td>914/10cv</td>
<td>22/674</td>
<td>common name of bird/phylogenic order</td>
</tr>
<tr>
<td>birdsci</td>
<td>914/10cv</td>
<td>22/1738</td>
<td>common+sci name/phylogenic order</td>
</tr>
<tr>
<td>hcoarse</td>
<td>1875/600</td>
<td>126/2098</td>
<td>company name/industry (coarse grain)</td>
</tr>
<tr>
<td>hfine</td>
<td>1875/600</td>
<td>228/2098</td>
<td>company name/industry (fine grain)</td>
</tr>
<tr>
<td>books</td>
<td>3501/1800</td>
<td>63/7019</td>
<td>book title/subject heading</td>
</tr>
<tr>
<td>species</td>
<td>3119/1600</td>
<td>6/7231</td>
<td>animal name/phylum</td>
</tr>
<tr>
<td>netvet</td>
<td>3596/2000</td>
<td>14/5460</td>
<td>URL title/category</td>
</tr>
</tbody>
</table>
Using WHIRL for Classification

Joint work with Haym Hirsh
Classification with “side information”

Consider classification...

- **Observation**: Performance can often be improved by obtaining additional features about the entities involved.

- **Question**: Can performance be improved using weaker “side information”—like additional features that might or might not be about the entities involved in the classification task?
<table>
<thead>
<tr>
<th>Instance</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itzak Perlman</td>
<td>BMG</td>
</tr>
<tr>
<td>Billy Joel</td>
<td>RCA</td>
</tr>
<tr>
<td>Metallica</td>
<td>... pop</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Goal:** from the data above, learn to classify musical artists as classical vs. popular.

**Basic ideas:** introduce new features for artist names that

- appear in certain lists or tables; (e.g., italicized names in the ‘Guest Artist’ page)
- are modified by certain words (e.g., “KØØL”)

**Guest Artists: Spring 2000**

- Apr 9, *Itzak Perlman*
- May 3, *Yo Yo Ma*
- May 17, *The Guanari Quartet*
- ...

**Biff’s KØØL Band Links**

- Nine Inch Nails (new!)
- **Metallica!!** Rockin’! Anyone know where can I find some MP3s?
- ...

...
The extraction algorithm

1. From HTML pages, create a table of \((\text{possible-name}, \text{position})\) pairs.

2. Soft-join with \textit{instance names} to get \((\text{instance-name}, \text{position})\) pairs.
   
   \textit{Position} is a new feature for the instance.

3. Can also create features from \((\text{possible-name}, \text{header-word})\) pairs.
html(head(...),
    body(
        h2(K∅∅L Band Links),
        table(
            tr(td(Metallica),
                td(Nine Inch Nails (new!))),
            tr(td(Barry Manilow),
                ... )
    ...

(“K∅∅L Band Links”, www.biff.com/html_body_h1)
(“Metallica”, www.biff.com/html_body_table_tr_td)
(“Nine Inch Nails (new!)”, www.biff.com/html_body_table_tr_td)
(“Barry Manilow”, www.biff.com/html_body_table_tr_td)
    soft-join with instances and threshold

Instances:
    ...
    Metallica
    Nine Inch Nails
    Itzak Perlman
    ...

h2(K∅∅L Band Links),
  table(
    tr(td(Metallica),
       td(Nine Inch Nails (new!))),
    tr(td(Barry Manilow),
      ...

(instance-name, position)
  (“Metallica”, www.biff.com/html_body_table_tr_td)
  (“Nine Inch Nails”, www.biff.com/html_body_table_tr_td)
  (“Barry Manilow”, www.biff.com/html_body_table_tr_td)
Features from "header words"

h2(K00L Band Links),
  | table(
  |   tr(td((Metallica)),
  |     td(Nine Inch Nails (new!))),
...

(instance-name, position)
("Metallica", www.biff.com/html_body_table_tr_td) ...

(instance-name, header-word)
("Metallica", "K00L")
("Metallica", "Band")
("Metallica", "Links")
...

53
## Benchmark problems

<table>
<thead>
<tr>
<th></th>
<th>#example</th>
<th>#class</th>
<th>#terms</th>
<th>#pages</th>
<th>#features added</th>
</tr>
</thead>
<tbody>
<tr>
<td>music</td>
<td>1010</td>
<td>20</td>
<td>1600</td>
<td>217</td>
<td>1890</td>
</tr>
<tr>
<td>games</td>
<td>791</td>
<td>6</td>
<td>1133</td>
<td>177</td>
<td>1169</td>
</tr>
<tr>
<td>birdcom</td>
<td>915</td>
<td>22</td>
<td>674</td>
<td>83</td>
<td>918</td>
</tr>
<tr>
<td>birdsci</td>
<td>915</td>
<td>22</td>
<td>1738</td>
<td>83</td>
<td>533</td>
</tr>
</tbody>
</table>

- original data: names as bag-of-words
- music: (Cohen&Fan,WWW00) others: (Cohen&Hirsh,KDD98)
- note: test data must be processed as well (transduction).
RIPPER: 200 training examples, 100 trials

<table>
<thead>
<tr>
<th></th>
<th>W-L-T</th>
<th>Average error</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>text</td>
<td>expanded</td>
</tr>
<tr>
<td>music</td>
<td>86-0-14</td>
<td>58.3</td>
<td>51.5</td>
</tr>
<tr>
<td>cdroms</td>
<td>29-7-64</td>
<td>67.2</td>
<td>65.8</td>
</tr>
<tr>
<td>birdcom</td>
<td>77-2-21</td>
<td>27.7</td>
<td>21.2</td>
</tr>
<tr>
<td>birdsci</td>
<td>35-8-57</td>
<td>26.4</td>
<td>23.6</td>
</tr>
</tbody>
</table>
Results (with RIPPER)

![Graph showing error rate vs. training examples for different categories: music, expanded, text only, and web only. The graph displays the trend of error rate decreasing with an increase in training examples.]
Results

![Graph showing error rate vs. training examples for different categories: expanded, text only, and web only. The graph illustrates a decreasing trend in error rate as the number of training examples increases.]
Web pages automatically crawled—not sampled from WHIRL DB on birds.
The show so far:

- **Motivation**: why this is the big problem.
- **WHIRL**: Data model, query language, efficient implementation

Results & applications:

- Queries without formal identifiers
- Queries that don’t require extraction
- Queries that generalize (Cohen & Hirsh, KDD98)
- Queries that automatically collect background knowledge for learning (Cohen ML2000, Cohen&Fan WWW2000)
- Comparison of TFIDF metric with other distance metrics for strings (Cohen, Ravikumar, Fienberg, *in progress*)
Other common distance metrics for strings

- **Bioinformatics**: edit distance metrics like Levenstein, Needleman-Wunch, Smith-Waterman, ... Can cope with misspelled tokens; not sensitive to frequency statistics (matching “Incorp” ≈ matching “Lucent”).

- **Information retrieval**: token-based distance metrics like TFIDF (used in WHIRL), Jaccard, Dice, ..., statistical distances based on language modeling, ...
  Generally applied to long documents (prior to WHIRL).

- **Probabilistic record linkage**: statistical agreement measures like Fellegi-Sunter; *ad hoc* string distance metrics like Jaro, Jaro-Winkler.
  Generally used in a hand-constructed statistical model of matching/non-matching records, not as “hands-off” metrics.
## Evaluation datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Src</th>
<th>#Strings</th>
<th>#Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>animal</td>
<td>Whirl</td>
<td>5709</td>
<td>30,006</td>
</tr>
<tr>
<td>bird1</td>
<td>Whirl</td>
<td>377</td>
<td>1,977</td>
</tr>
<tr>
<td>bird2</td>
<td>Whirl</td>
<td>982</td>
<td>4,905</td>
</tr>
<tr>
<td>bird3</td>
<td>Whirl</td>
<td>38</td>
<td>188</td>
</tr>
<tr>
<td>bird4</td>
<td>Whirl</td>
<td>719</td>
<td>4,618</td>
</tr>
<tr>
<td>business</td>
<td>Whirl</td>
<td>2139</td>
<td>10,526</td>
</tr>
<tr>
<td>game</td>
<td>Whirl</td>
<td>911</td>
<td>5,060</td>
</tr>
<tr>
<td>park</td>
<td>Whirl</td>
<td>654</td>
<td>3,425</td>
</tr>
<tr>
<td>fodorZagrat</td>
<td>Ariadne</td>
<td>863</td>
<td>10,846</td>
</tr>
<tr>
<td>ucdFolks</td>
<td>Monge-Elkan</td>
<td>90</td>
<td>454</td>
</tr>
<tr>
<td>census</td>
<td>Winkler</td>
<td>841</td>
<td>5,765</td>
</tr>
</tbody>
</table>
Evaluation metrics

From IR community:

- 11-pt interpolated average precision, averaged across all datasets.
- Non-interpolated average precision, on each dataset.
- Maximum F1-measure on each dataset (see paper).
Edit distance metrics:

- Measure distance between strings $s$ and $t$ as cost of the least expensive sequence of edit operations that transform $s$ to $t$.

- Example: to transform “Will Cohon” to “William Cohen” might use: copy, copy, copy, copy, insert(i), insert(a), insert(m), copy, copy, copy, copy, replace(o,e), copy.

- Different operations/costs lead to different metrics:
  - Levenstein: cost(cpy)=0, cost(ins($x$))=1, cost(replace($x$, $y$))=1.

- Minimal cost edit sequence usually can be found with dynamic programming in time $O(|s| \cdot |t|)$.
### Matrix $i, j$

The matrix $i, j$ represents the cheapest path from the northwest corner to any cell $(i, j)$.

Here is the matrix:\n
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>i</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>a</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>o</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>h</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>e</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>n</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Insert in $s$:** move **east**, pay $1
- **Insert in $t$:** move **south**, pay $1
- **Copy:** move **southeast**, pay $0
- **Replace:** move **southeast**, pay $1
- **Matrix $i, j$:** cheapest path from northwest corner to $i, j$.
- **Edit cost:** cheapest path to southeast corner (4).
### Jaro distance metric

\[
Jaro(s, t) = \frac{1}{3} \left( \frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s', t'}}{|s'|} \right)
\]

- Find matching letters near the main diagonal, then find “common parts” of \( s \) and \( t \): here \( s' = t' = \text{“ed chn”} \)
- Count transpositions in \( s' \) relative to \( t' \): \( T_{s', t'} \)
- Average fraction of \( s, t \) that are “common” with fraction of \( s' \) in the same order as \( t' \).
- **Jaro-Winkler**: increase weight for weak matches if first few characters match well.
Jaro distance metric

\[
\text{Jaro}(s, t) = \frac{1}{3} \cdot \left( \frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s', t'}}{|s'|} \right)
\]

- Find matching letters near the main diagonal, then find “common parts” of \(s\) and \(t\): here \(s' = \text{“ed hcn”}\), \(t' = \text{“ed chn”}\)
- Count transpositions in \(s'\) relative to \(t'\): \(T_{s', t'}\)
- Average fraction of \(s, t\) that are “common” with fraction of \(s'\) in the same order as \(t'\).
- Jaro-Winkler: increase weight for weak matches if first few characters match well.
Edit-distance and Jaro-based distances

Monge-Elkan: edit distance with well-tuned costs, affine gaps.
Token-based distance metrics

- View strings as sets (or bags) of tokens, $S$ and $T$.
- **Jaccard distance**: $\frac{|S \cap T|}{|S \cup T|}$.
- View set $S$ of tokens as a sample from an unknown distribution $P_S$, and consider differences between $P_S$ and $P_T$:

$$\text{Jensen-Shannon}(S, T) = \frac{1}{2} (KL(P_S||Q) + KL(P_T||Q))$$

where $KL(P||Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$, $Q = \text{avg}(P_S, P_T)$. 
Token-based distance metrics

- **Simplified Fellegi-Sunter**: estimate log-odds of $P(S, T|s$ and $t$ match) as

  $$\sum_{w \in S \cap T} \log \frac{1}{P(w)} - \sum_{w \in (S-T) \cup (T-S)} -k \log \frac{1}{P(w)}$$

- **TFIDF** (WHIRL method): weight $w$ by

  $$\log (1+\text{freq of } w \text{ in string}) \times \log (\frac{\#\text{strings}}{\#\text{strings containing } w})$$

  Scale vectors to unit length, then similarity is:

  $$\sum_{w \in S \cap T} \text{weight}(w, S) \cdot \text{weight}(w, T)$$
Token-based distance metrics

![Graph showing token-based distance metrics](image)
Hybrid distance measures

Assume sets of tokens $S$, $T$ and a similarity measure for tokens $\text{sim}(w, v)$.

- Monge-Elkan propose a \textbf{Level two} similarity function between $S = \{w_1, \ldots, w_K\}$ and $T = \{v_1, \ldots, v_L\}$:

  $$\text{Level2}(S, T) = \frac{1}{K} \sum_{i=1}^{K} \max_{j=1}^{L} \text{sim}(w_i, v_j)$$
Hybrid distance measures

- We propose a “softer” TFIDF measure. Recall:

\[
\text{TFIDF}(S, T) = \sum_{w \in S \cap T} \text{weight}(w, S) \cdot \text{weight}(w, T)
\]

\[
\text{SoftTFIDF}(S, T) = \sum_{w \in \text{CLOSE}(\theta, S, T)} \text{weight}(w, S) \cdot \text{weight}(w, T) \cdot c(w, T)
\]
Hybrid distance measures

- We propose a “softer” TFIDF measure:

\[
\text{SoftTFIDF}(S, T) = \sum_{w \in \text{CLOSE}(\theta, S, T)} \text{weight}(w, S) \cdot \text{weight}(w, T) \cdot c(w, T)
\]

where
- \( \text{CLOSE}(\theta, S, T) = \{ w \in S : \exists v \in T \text{ and } \text{sim}(w, v) > \theta \} \)
  (Similar tokens in \( S \) and \( T \))
- \( c(w, T) = \max_{v \in T} \text{sim}(w, v) \)
  (Similarity to closest token in \( T \))

- Will use \( \theta = 0.9 \), \( \text{sim} \)=Jaro-Winkler.
Hybrid distance metrics
Grand summary of best metrics
### Prospective test: two more datasets

<table>
<thead>
<tr>
<th>Method</th>
<th>UVA (Monge-Elkan)</th>
<th>CoraATDV (JPRC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MaxF1</td>
<td>AvgPrec</td>
</tr>
<tr>
<td>SoftTFIDF</td>
<td>0.89</td>
<td>0.91</td>
</tr>
<tr>
<td>TFIDF</td>
<td>0.79</td>
<td>0.84</td>
</tr>
<tr>
<td>SFS</td>
<td>0.71</td>
<td>0.75</td>
</tr>
<tr>
<td>Level2 J-W</td>
<td>0.73</td>
<td>0.69</td>
</tr>
</tbody>
</table>
Conclusions

- The next step (?) after distributing text world-wide: learn how to reason with a world-wide knowledge base.
- Integrating structured data from multiple sources is a crucial problem.
  - Object identity issues dominate in many domains.
- WHIRL efficiently propagates uncertainty about object identity.
- TFIDF distance is fast and surprisingly robust.
- WHIRL data model and query language allow an intermediate between “document delivery” and “deductive” information systems.
Beyond data integration, WHIRL is useful for many other tasks:

- Querying imperfectly extracted data
- Queries that generalize (Cohen & Hirsh, KDD98)
- Automatically collecting features for learning (Cohen, ML2000)
- Queries that suggest extraction rules (Cohen, AAAI99)
- Content-based recommendation (Basu et al, JAIR2001)
- Bootstrapping-based extraction of relations from text (Agichtein & Gravano, DL2000)
- Extensions for semistructured data (Chinenyanga & Kushmerick, SIGIR2001)
- Stochastic matrix multiplication for better performance on conjunctive chain queries (Gravano et al, WWW2003)