WHIRL: A Formalism for Representing Web Data

William Cohen
AT&T Labs—Research
Shannon Laboratory

What's wrong with the Web?

- The Web is a huge advance in distributing information ...
- ...but it's often hard to find what you want.

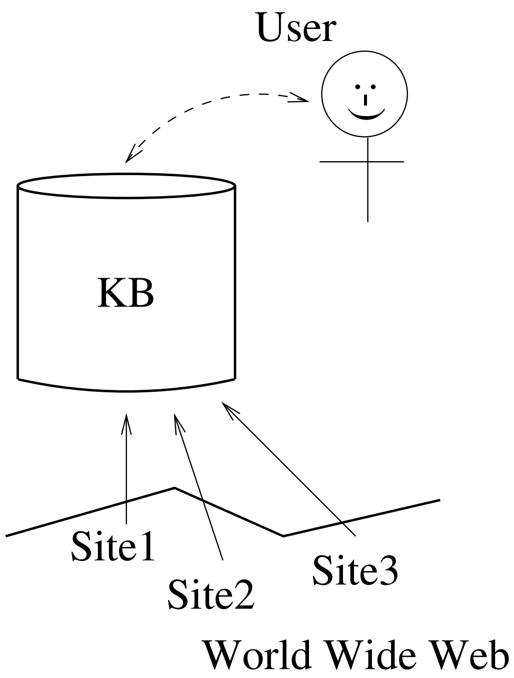
  "Although your workstation can currently *retrieve* 200 million Web pages, it currently *understands* none of these." (Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam & Slattery, AAAI98)

- To fundamentally improve information access, the Web must evolve from a distribution system to a reasoning system.

What's the research problem?

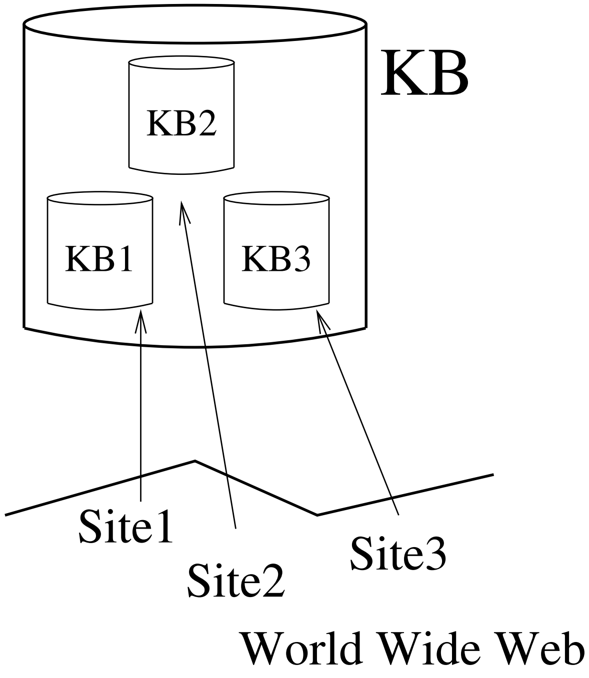
One approach:

1. Make (part of) the Web look like a knowledge base.
2. Query the knowledge base.
3. What's the research problem?



What's the research problem?

We don't know how to represent and reason with information that comes from many different, autonomous sources.



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.

all mallards are waterfowl	+	`duck.jpg` is a picture of a mallard	=	`duck.jpg` is a picture of a waterfowl

Taxonomy

Order	Species
waterfowl	mallard
waterfowl	bufflehead
raptor	osprey
raptor	bald eagle
...	...

+

Images

Species	File
robin	robin.jpg
mallard	duck.jpg
osprey	hawk.jpg
penguin	tweety.jpg
...	...

=

Order	Species	File
waterfowl	mallard	duck.jpg
raptor	osprey	hawk.jpg
...	...	...

mallards are found in New Jersey	+	`duck.jpg` is a picture of a mallard	=	`duck.jpg` is a picture of something found in New Jersey

NJ Birds

Species
robin
mallard
osprey
...

\+

Images

Species	File
robin	robin.jpg
mallard	duck.jpg
penguin	tweety.jpg
...	...

=

Species	File
robin	robin.jpg
mallard	duck.jpg
...	...

Deduction enables modularity.

Why deduction requires co-operation

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

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

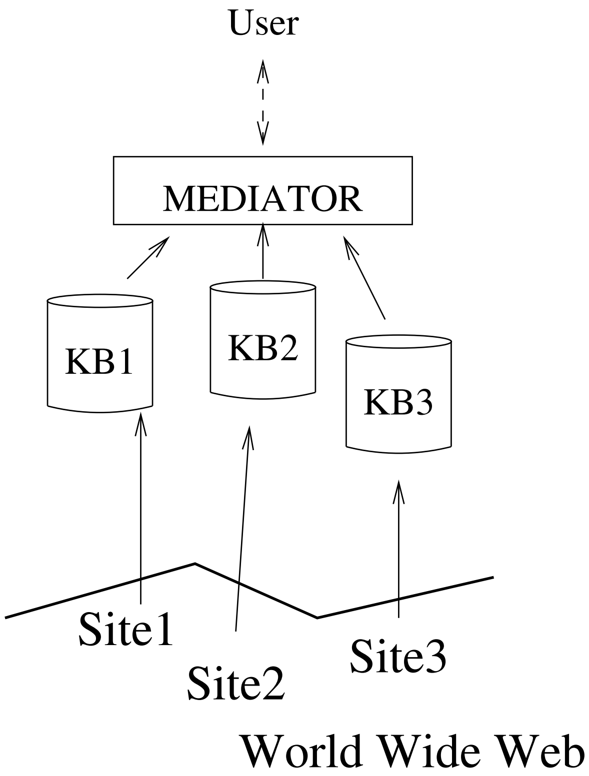
- predicate names and argument positions (schema);
- taxonomic information;
- formal names (OIDs) for every entity they describe;
- ...

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?



Humongous	Humongous Entertainment
Headbone	Headbone Interactive
The Lion King: Storybook	Lion King Animated StoryBook
Disney's Activity Center, The Lion King	The Lion King Activity Center

Microsoft	Microsoft Kids Microsoft/Scholastic
Kestrel	American Kestrel Eurasian Kestrel
Canada Goose	Goose, Aleutian Canada
Mallard	Mallard, Mariana

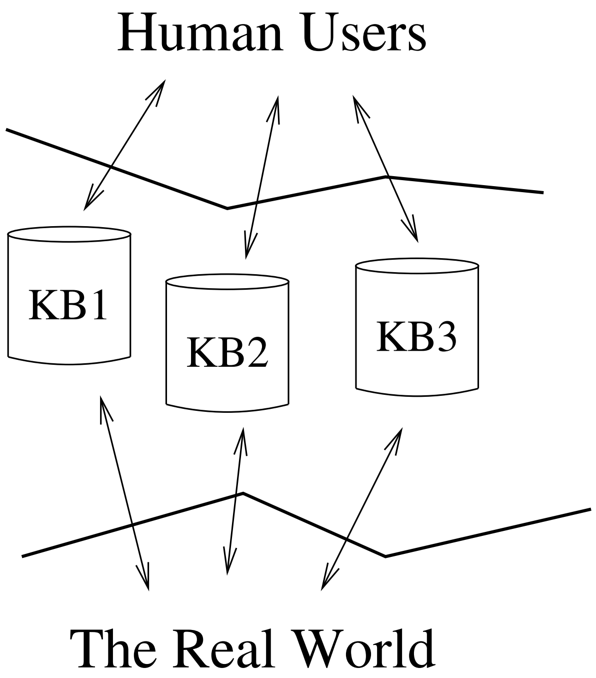
Bell Labs	AT&T Bell Labs
AT&T Research	AT&T Labs
Bell Telephone Labs	AT&T Labs—Research
AT&T Labs–Research, Shannon Laboratory	Lucent Innovations Bell Labs Technology

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 ($\approx$ 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 & implementation
  - Applications of WHIRL
    - Data integration
    - ...without extraction
    - Classification learning and collaborative filtering using Web data

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:

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

- Heuristic: make $weight(t, d)$ 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:
  - Low weight: "Inc", "Corp", ...
  - High weight: "Microsoft", "Lucent", ...
  - Medium weight: "Acme", "American", ...

Background: Information retrieval

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

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

$$\begin{aligned} SIM(d_i, d_j) &= \sum_{t \in T} weight(t, d_i) \cdot weight(t, d_j) \\ &= \vec{v}_i \cdot \vec{v}_j \end{aligned}$$

Also, $0 \leq SIM(d_i, d_j) \leq 1$.

Cinema	Movie	Show Times
Roberts Theaters Chatham	Brassed Off	7:15 - 9:10
Berkeley Cinema	Hercules	4:15 - 7:30
Sony Mountainside Theater	Men In Black	7:40 - 8:40 - 9:30 - 10:10

listing($\vec{v}_{RTC}$, $\vec{v}_{BO}$, $\vec{v}_{T79}$), 1.
listing($\vec{v}_{BC}$, $\vec{v}_{H}$, $\vec{v}_{T47}$), 1.
listing($\vec{v}_{SMT}$, $\vec{v}_{MIB}$, $\vec{v}_{T789}$), 1.

review($\vec{w}_{MIB97}$, $\vec{w}_{R1}$), 1.
review($\vec{w}_{FO}$, $\vec{w}_{R2}$), 1.
review($\vec{w}_{SB}$, $\vec{w}_{R3}$), 1.

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

Movie	Review
Men in Black, 1997	$(* * *)$ One of the biggest hits of ...
Face/Off, 1997	$(* * \frac{1}{2})$ After a slow start, ...
Space Balls, 1987	$(* \frac{1}{2})$ Not one of Mel Brooks' best efforts, this spoof ...

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

Queries in WHIRL

- Syntax: WHIRL = (similarity) Prolog − function symbols − recursion − negation + X∼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.
  - $\text{Score}(a \sim b) = \mathit{SIM}(a,b)$ for similarity literals.
  - $\text{Score}(\phi \wedge \psi) = \text{Score}(\phi) \cdot \text{Score}(\psi)$.
  - $\text{Score}(\phi \vee \psi) = 1 - (1 - \text{Score}(\phi))(1 - \text{Score}(\psi))$
  - 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).

Sample WHIRL queries

Standard ranked retrieval:

"find reviews of sci-fi comedies".

?- review(Title,Rev) $\wedge$ Rev$\sim$"sci-fi comedy"

(score 0.22): $\theta_1 = \{\mathrm{Title}/\vec{w}_{MIB97}, \mathrm{Rev}/\vec{w}_{R1}\}$
(score 0.19): $\theta_2 = \{\mathrm{Title}/\vec{w}_{SB}, \mathrm{Rev}/\vec{w}_{R4}\}$
(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,Rev) $\wedge$ listing(C,Id2,T2,Time)
$\wedge$ Id1$\sim$Id2 $\wedge$ C$\sim$"Sony Mountainside Theater"

(score 1.00): $\theta_1 = \{\mathrm{Id1}/\vec{v}_{93}, \mathrm{Id2}/\vec{w}_{93}, \mathrm{Rev}/\vec{w}_{R1}, \ldots\}$

(score 1.00): $\theta_2 = \ldots$

Cinema	Id	Movie	Time
...	21	Brassed Off	...
Sony ...	93	Men In Black	...

Id	Movie	Review
93	Men in Black, 1997	...
44	Face/Off, 1997	...

Sample WHIRL queries

Mixed queries: *"where is [Men in Black] playing?"*

?- review(Id1,T1,Rev) $\wedge$ listing(C,Id2,T2,Time)
$\wedge$ Id1$\sim$Id2 $\wedge$ Rev$\sim$"sci-fi comedy with Will Smith"
(score 0.22): $\theta_1 = \{\mathrm{Id1}/\vec{v}_{93}, \mathrm{Id2}/\vec{w}_{93}, \mathrm{Rev}/\vec{w}_{R1}, \ldots\}$
(score 0.13): $\theta_2 = \ldots$

Cinema	Id	Movie	Time
...	21	Brassed Off	...
Sony ...	93	Men In Black	...

Id	Movie	Review
93	Men in Black, 1997	...
44	Face/Off, 1997	...

A realistic situation

Cinema	Movie	Show Times
Roberts Theaters Chatham	Brassed Off	7:15 - 9:10
Berkeley Cinema	Hercules	4:15 - 7:30
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.

Movie	Review
Men in Black, 1997	$(* * *)$ One of the biggest hits of ...
Face/Off, 1997	$(* * \frac{1}{2})$ After a slow start, ...
Space Balls, 1987	$(* \frac{1}{2})$ Not one of Mel Brooks' best efforts, this spoof ...

Sample WHIRL queries

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

?- review(Title1,Rev) $\wedge$ listing(_,Title2,Time) $\wedge$ Title1$\sim$Title2

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

(Men in Black) (Men in Black, 1997)

...

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

(Brassed Off) (Face/Off)

...

How well do similarity joins work?

?- top500(X), hiTech(Y), X~Y

top500:	*hiTech:*
Abbott Laboratories	ACC CORP
Able Telcom Holding Corp.	ADC TELECOMMUNICATION INC
Access Health, Inc.	ADELPHIA COMMUNICATIONS CORP
Acclaim Entertainment, Inc.	ADT LTD
Ace Hardware Corporation	ADTRAN INC
ACS Communications, Inc.	AIRTOUCH COMMUNICATIONS
ACT Manufacturing, Inc.	AMATI COMMUNICATIONS CORP
Active Voice Corporation	AMERITECH CORP
Adams Media Corporation	APERTUS TECHNOLOGIES INC
Adolph Coors Company	APPLIED DIGITAL ACCESS INC
...	APPLIED INNOVATION INC
	...

Sample company-name pairings

Evaluating similarity joins

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

1	$\surd$	a_1	b_1
2	$\surd$	a_2	b_2
3	$\times$	a_3	b_3
4	$\surd$	a_4	b_4
5	$\surd$	a_5	b_5
6	$\surd$	a_6	b_6
7	$\times$	a_7	b_7
8	$\surd$	a_8	b_8
9	$\surd$	a_9	b_9
10	$\times$	a_{10}	b_{10}
11	$\times$	a_{11}	b_{11}
12	$\surd$	a_{12}	b_{12}

Precision at K: G_K/K

Recall at K: G_K/G

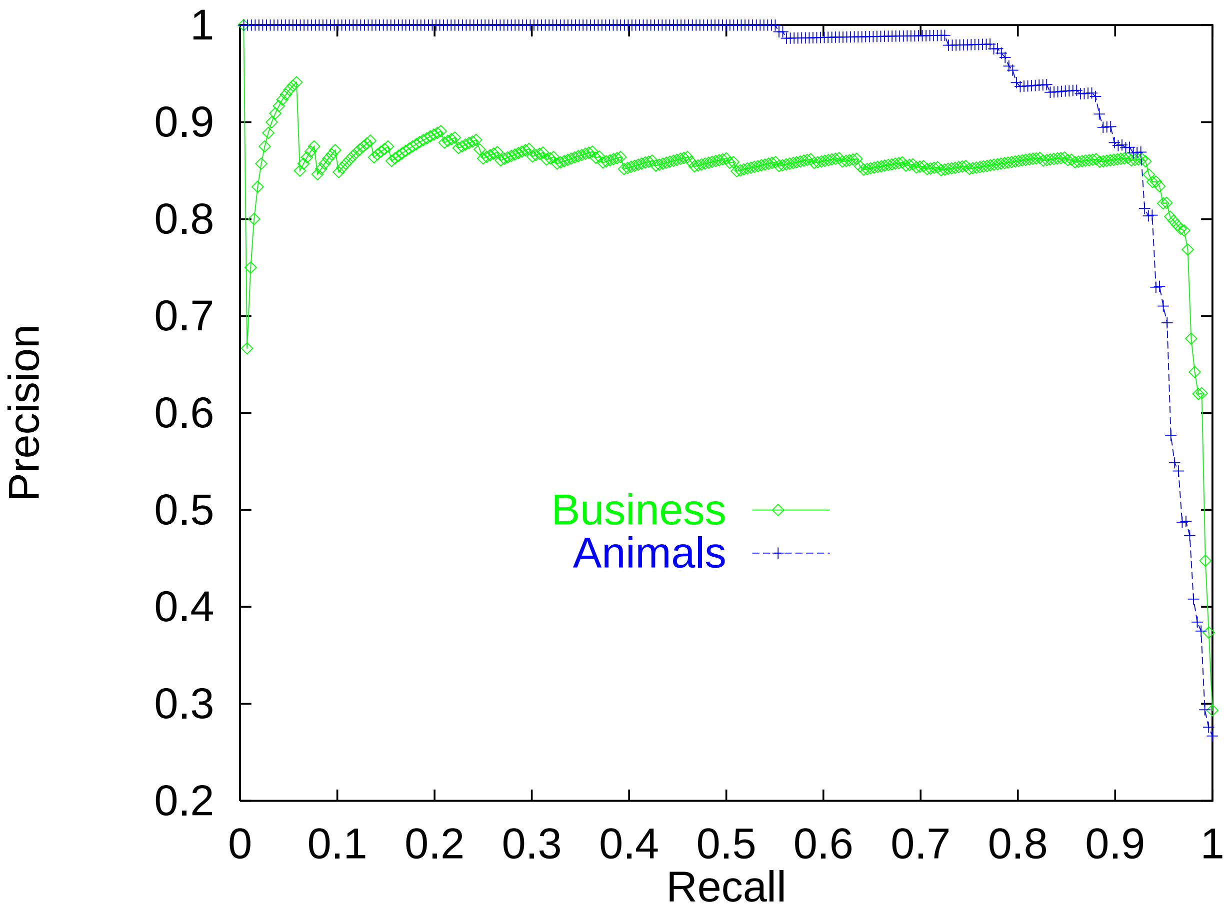
G: # good pairings

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



Evaluating WHIRL queries

Additional experiments:

- Repeat with more datasets from more domains.
  - Average precision (≈ 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?

Efficiently implementing WHIRL

Consider these queries:

$$\texttt{?- r(X),s(Y),X=Y} \tag{1}$$

$$\texttt{?- r(X),s(Y),X}\sim\texttt{Y} \tag{2}$$

- Goal with 1: given x, quickly find $y : s(y) \wedge y = x$ (hash joins)
- Goal with 2: given x, quickly find $y : s(y) \wedge y \sim x$
- Use inverted index for terms $t \in x$ to find candidate y's, and $A*$ search to schedule access to indices.

An efficient implementation

- For instance: I used WHIRL as the DBMS for two real-life integration systems. Benchmarking performance on real-world queries:
  - Birds of North America: 34 sites, 140,000+ facts, about 5 query/sec.
  - Computer Games for Kids: 15 sites, 20,000+ facts, about 3 query/sec.
  - Simple (2-3 join) queries average 12-50 query/sec; complex (5-6 join) queries average > 0.5 query/sec.

Screen shot from query system

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"

$v(\text{Url}) \leftarrow$
$\quad \text{review}(\text{Game1},\text{Url}) \wedge \text{category}(\text{Game2},\text{Cat})$
$\quad \wedge\ \text{Game1} \sim \text{Game2} \wedge \text{Cat} \sim \text{"adventure"}$

To answer this query, WHIRL guesses the class "adventure" based on similarities between names.

More uses of WHIRL: Classification

A learner inputs pairs $(x_1, \ell_{i_1}), \ldots, (x_m, \ell_{i_m})$ and outputs a rule

$$h : X \longrightarrow \{\ell_1, \ldots, \ell_k\}$$

Consider the WHIRL query:

category(Cat) $\leftarrow$ test(X) $\wedge$ train(Y,Cat) $\wedge$ X$\sim$Y

- Here test contains a single unclassified example, and train 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:
     Score($\phi \wedge \psi$) = 1 $-$ (1-Score(ϕ)) $\cdot$ (1-Score(ψ))

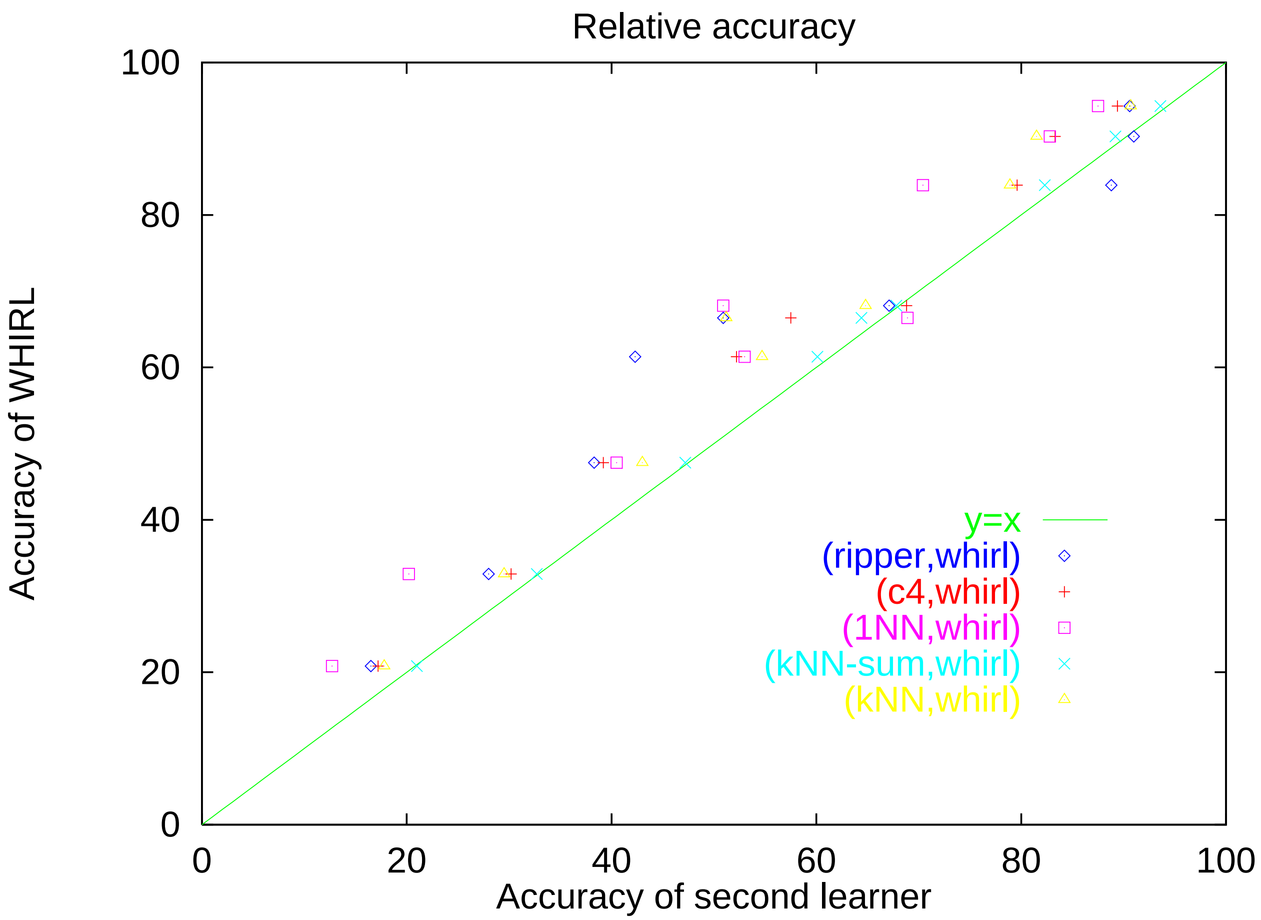
Using WHIRL for Classification

- Created nine representative datasets using data from Web.
- All instances were short "names"
  - *book title:* inst="The Humbugs of the World by P. T. Barnum (page images at MOA)", class="General Works"
  - *company name:* inst="National City Corporation", class="Banks–Midwest"
  - Also bird names, Web page titles, ...
- # classes: 6–228, #instances: 334–3596.
- Summary: WHIRL is comparable to or better than C4.5, RIPPER, and other mature classification-learning systems.

Benchmark classification problems

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

Using WHIRL for Classification



WHIRL for classification

Follow-up work by Zelikovitz & Hirsh replaces

$\text{category(Cat)} \leftarrow \text{test(X)} \wedge \text{train(Y,Cat)} \wedge \text{X}\sim\text{Y}$

with a more complex query using unlabeled data:

$\text{category(Cat)} \leftarrow$
$\quad \text{test(X)} \wedge \text{unlabeled(Z)} \wedge \text{train(Y,Cat)} \wedge \text{X}\sim\text{Z} \wedge \text{Z}\sim\text{Y}$

and shows improvement in classifying short documents with small amounts of training data.

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)

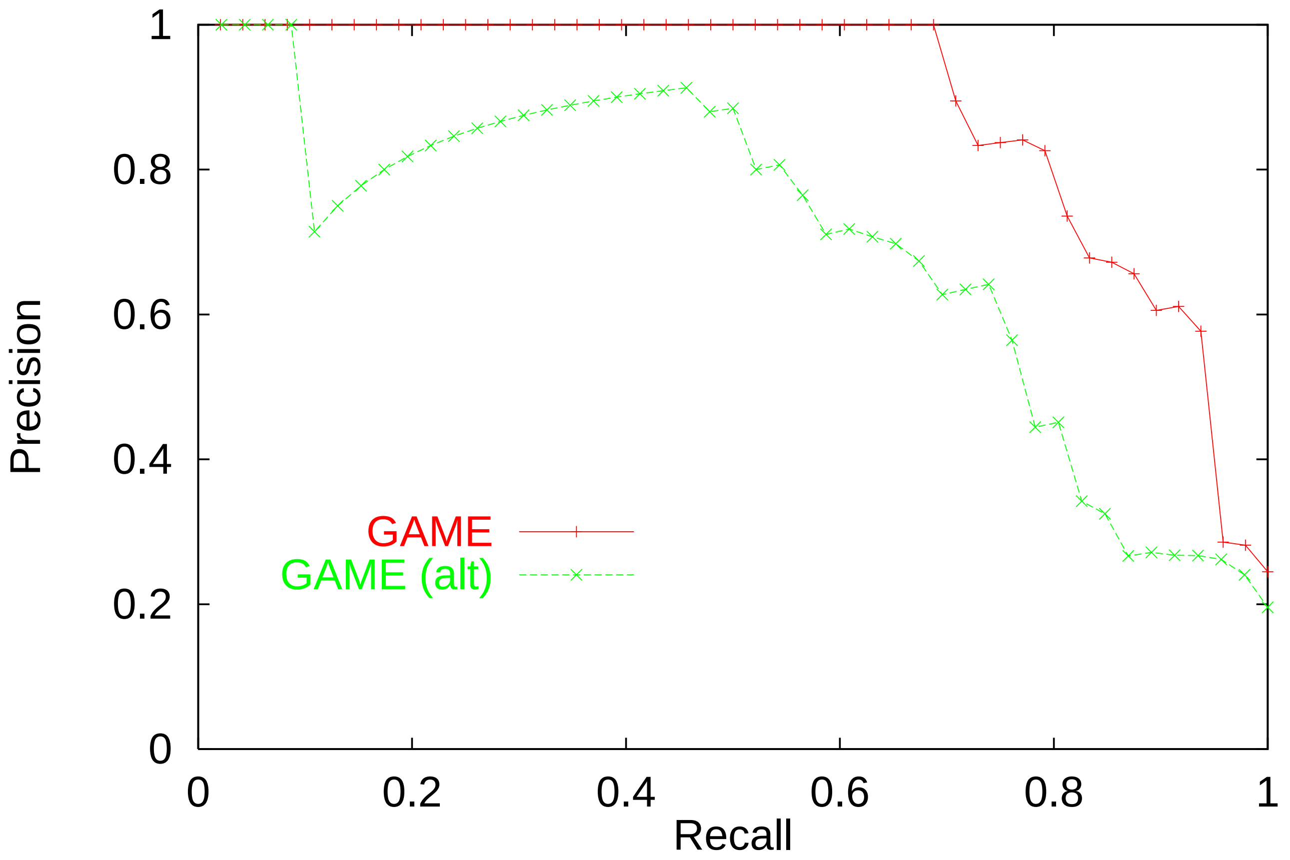
The extraction problem

Idea: use text without trying to extract names.

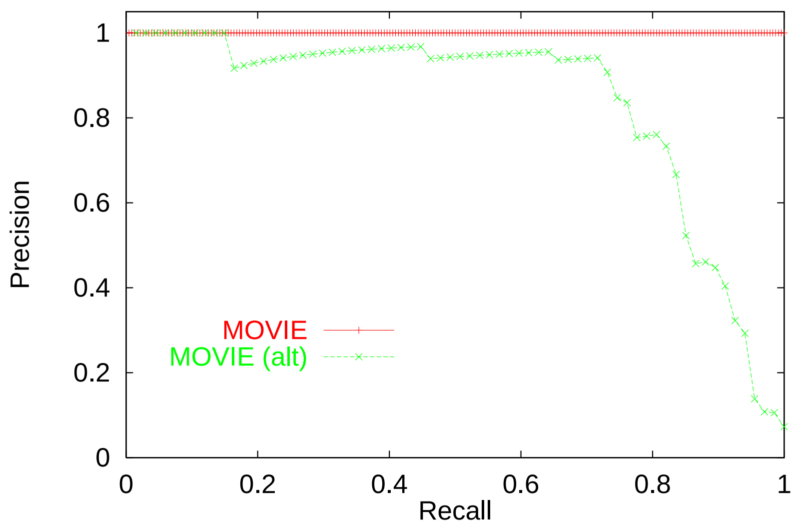
?- paragraph(X),name(Y),X~Y

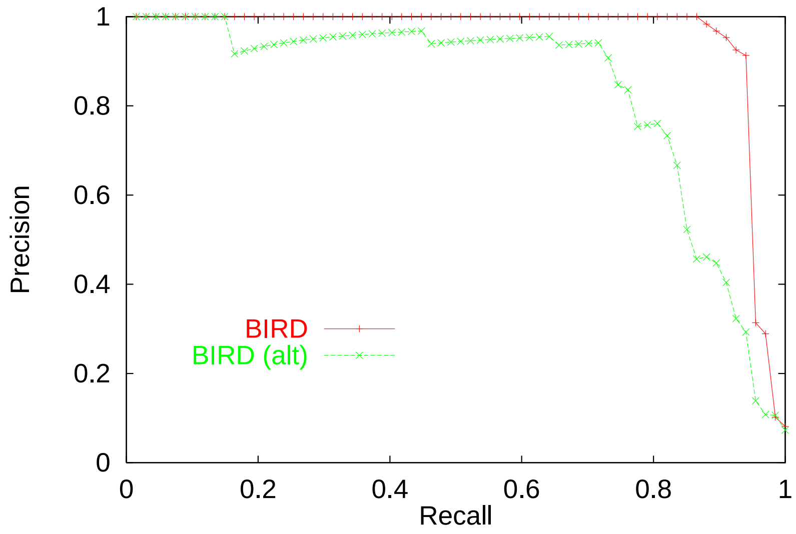
80.26	Ubi Software has a demo of Amazing Learning Games with Rayman.	Amazing Learning Games with Rayman	√
78.25	Interplay has a demo of Mario Teaches Typing. (PC)	Mario Teaches Typing	√
75.91	Warner Active has a small interactive demo for Where's Waldo at the Circus and Where's Waldo? Exploring Geography (Mac and Win)	Where's Waldo? Exploring Geography	√
74.94	MacPlay has demos of Marios Game Gallery and Mario Teaches Typing. (Mac)	Mario Teaches Typing	√
71.56	Interplay has a demo of Mario Teaches Typing. (PC)	Mario Teaches Typing 2	×

Deduction without extraction



Deduction without extraction





Movie 1: full review (no extraction).

Movie 2: movie name, cinema name & address, showtimes.

The current extraction process

Extracting data from HTML pages is a bottleneck in creating an information integration system.

- Start with an HTML page.
- Parse the markup
- Uses a manually-written “wrapper” program to convert the HTML parse tree to WHIRL facts.

demo page; tree

An alternative process

1. Store WHIRL facts about the HTML parse tree: e.g., "position" of each short marked-up text segment:

at_pos(html_head_title,node1, "PC Downloads").

...

at_pos(html_body_ul_li,node13,"Baby Games by Danny Stoll (D)").

at_pos(html_body_ul_li_a,node15,"Baby Games").

...

at_pos(html_body_ul_li_a,node71,"Follicle Follies).

...

2. Query this database directly.

An alternative process

Sample queries:

- What positions contain a game name?

  q(Pos,Y) ← at_pos(Pos,N,X),game(Y),Y∼X

- What positions correspond to lists of games?

  aggregate q(Pos,Y) above (part of WHIRL)

- What is “Baby Games”?

  q(Y) ← at_pos(P1,N1,X), X∼“Baby Games”,
  at_pos(P2,N2,Y), header_pos(P2), in_scope(P1,P2).

When are these types of queries useful?

Semi-automatic extraction

Human specifies:

- page to wrap
- expected type of wrapper (e.g., list of birds, . . .)

System provides (based on a WHIRL query):

- ranked list of proposed wrappers (e.g. candidate positions, . . .)

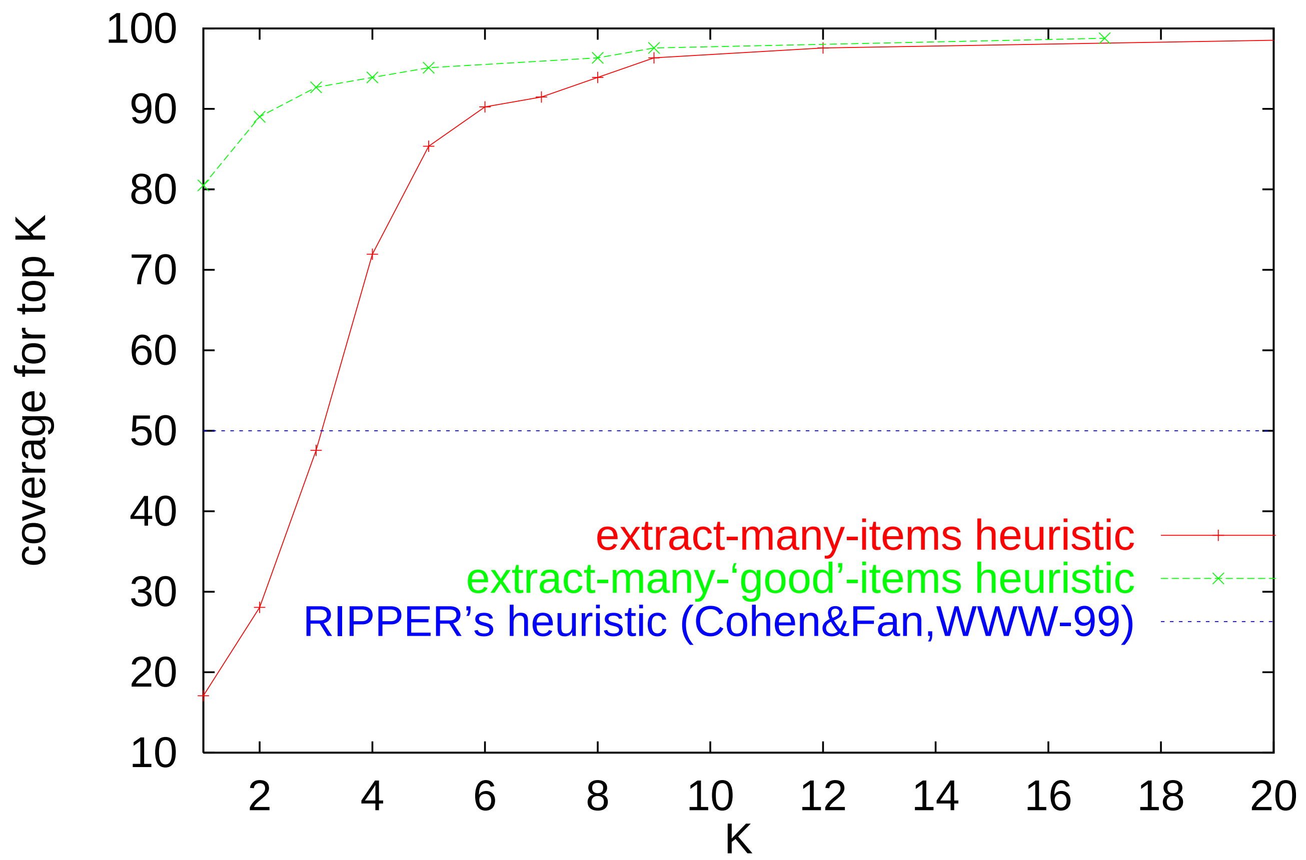
Human:

- picks wrapper from ranked list

A query that ranks complete "wrappers"

```
R_like_piece(Path1,Path2) ←
    possible_piece(Path1,Path2) ∧
    many( R_extracted_by(Path1a,Path2a,_,_),
          (Path1a=Path1 ∧ Path2a=Path2) ).
R_extracted_by(Path1,Path2,TextElt,AnchorElt) ←
    elt(TextElt, _, Text, Path1)
    ∧ path(TextElt, AnchorElt, Path2)
    ∧ R(X) ∧ Text∼X.
R_like_piece(Pos) ←
    possible_piece(Pos) ∧
    many( R_extracted_by(Pos1,_), (Pos=Pos1)).
R_extracted_by(Pos,TextElt) ←
    elt(TextElt, _, Text, Pos) ∧ R(X) ∧ Text∼X.
```

Using WHIRL for Extraction



- Graph shows percent of 80+ benchmarks such that all correct wrappers for a page are ranked in the top K.

From semi-automatic to automatic extraction

Summary: simple WHIRL heuristics can be used to propose wrappers to a human, who will

- check the wrappers
- determine semantics (where in the DB to store the data).

Can we do better?

- What will the DB be used for?

The collaborative filtering task

Given: New user u_*, previous users $U = \{u_1, \ldots, u_n\}$, and a set $favorite\_bands(u_i)$ for each user.

Goal: Recommend a new band to u_*.

Method (k-NN):

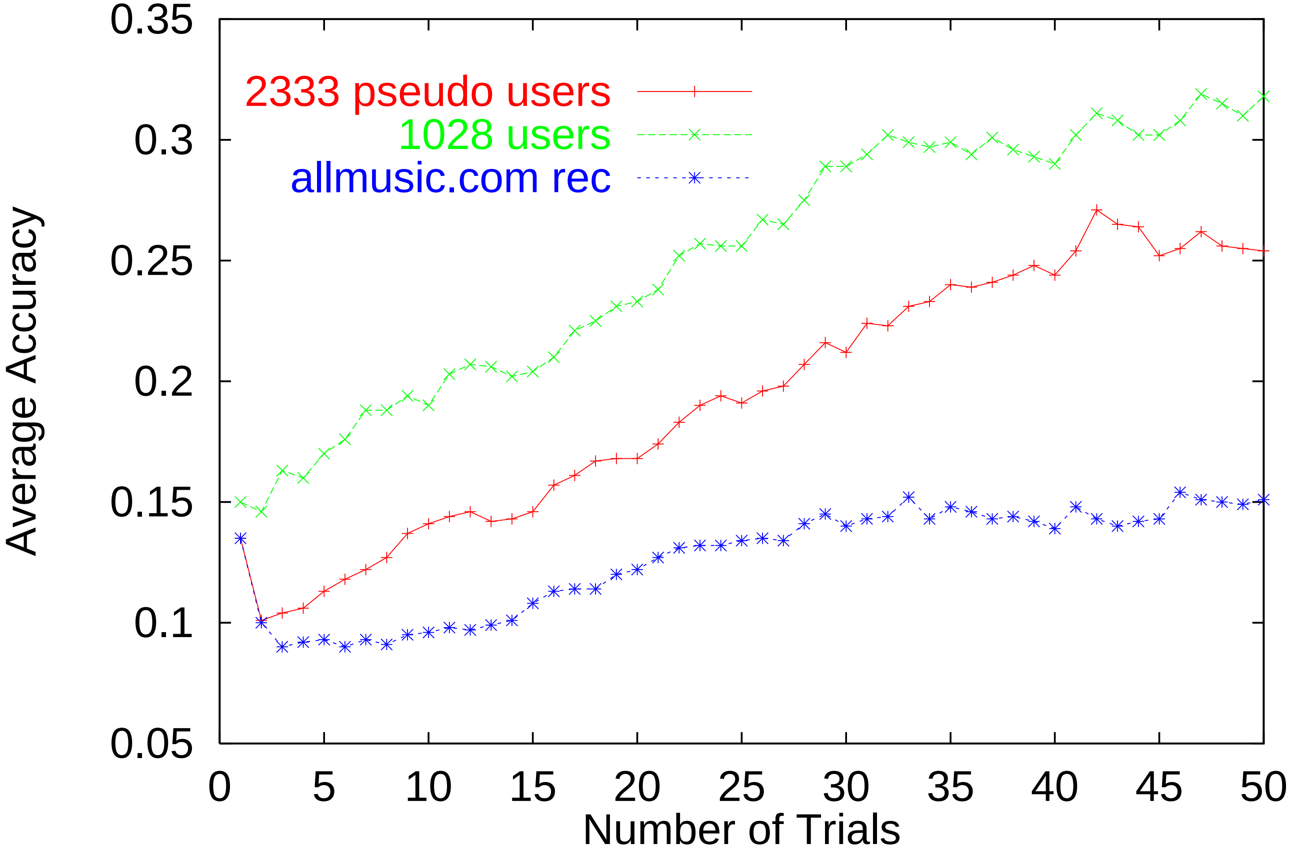
1. Find k users $u_{j_1}, \ldots, u_{j_k} \in U$ with tastes most similar to u_*'s.
2. Recommend the artist most popular with the $u'_{j_i}s$

Idea: look for Web positions p containing lists of bands, and replace/augment (u_i,$favorite\_bands(u_i)$) with "pseudo-users"

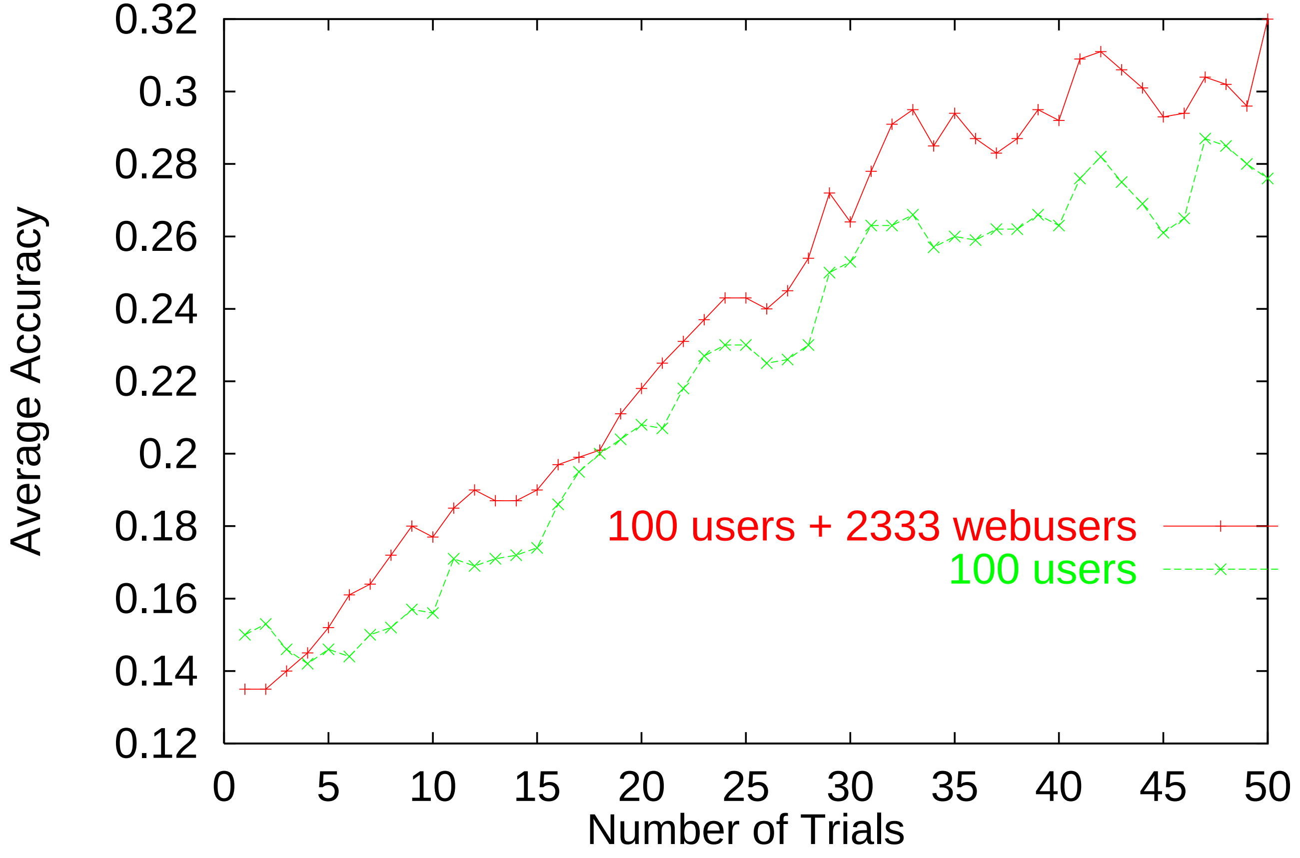
(position p_j, bands in position p_j)

completely ignoring the semantics of the "list" p

CF using Web pseudo-users



CF using Web pseudo-users



Extraction for Classification Learning

Recall a learner inputs pairs $(x_1, \ell_{i_1}), \ldots, (x_m, \ell_{i_m})$ and outputs a general rule

$$h : X \longrightarrow \{\ell_1, \ldots, \ell_k\}$$

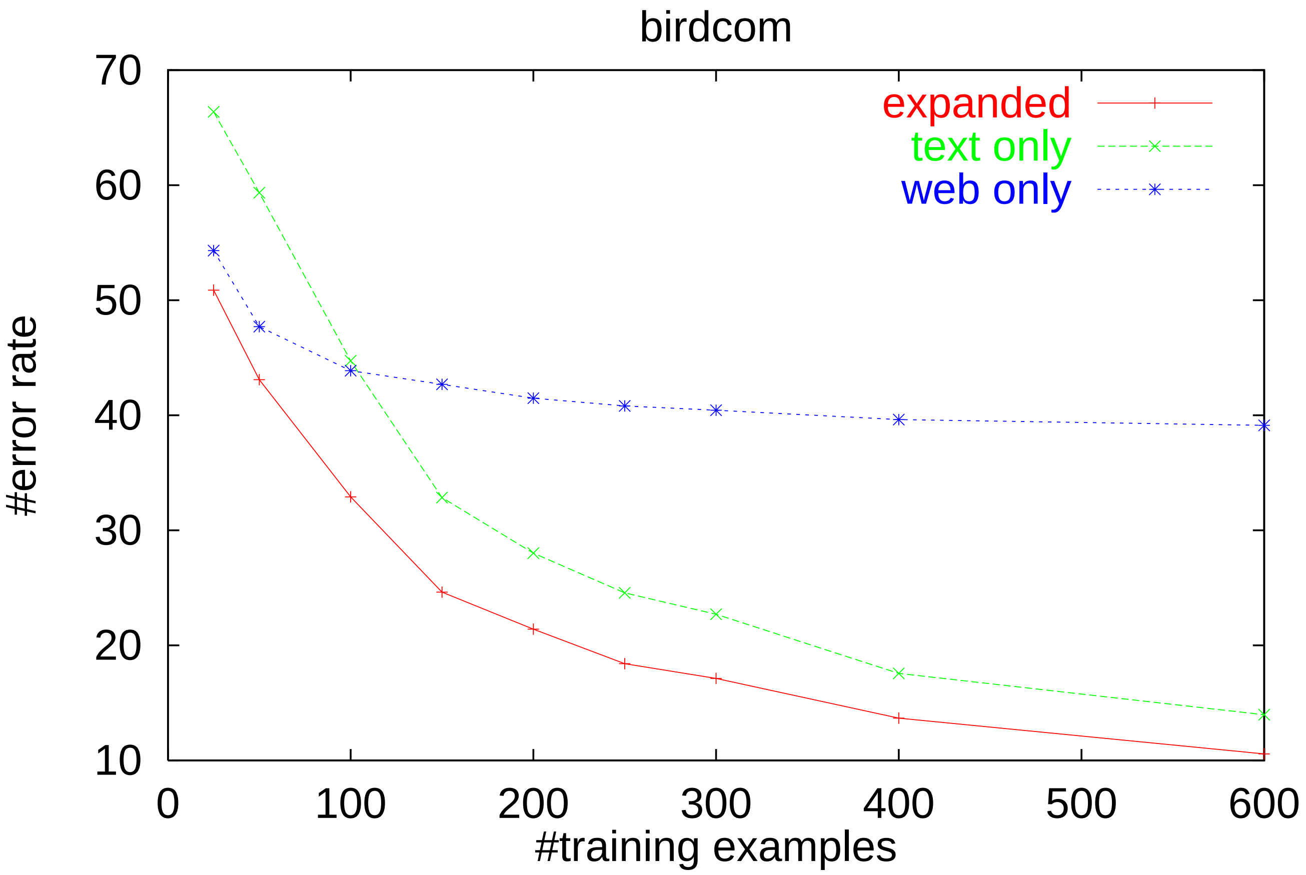
The classification learning experiments used a binary representation of x:

$$x = \langle f_{w_1}(x), \ldots, f_{w_n}(x) \rangle$$

where $f_{w_i}(x) = 1 \Leftrightarrow$ word $w_i \in x$. This can be extended by adding new features:

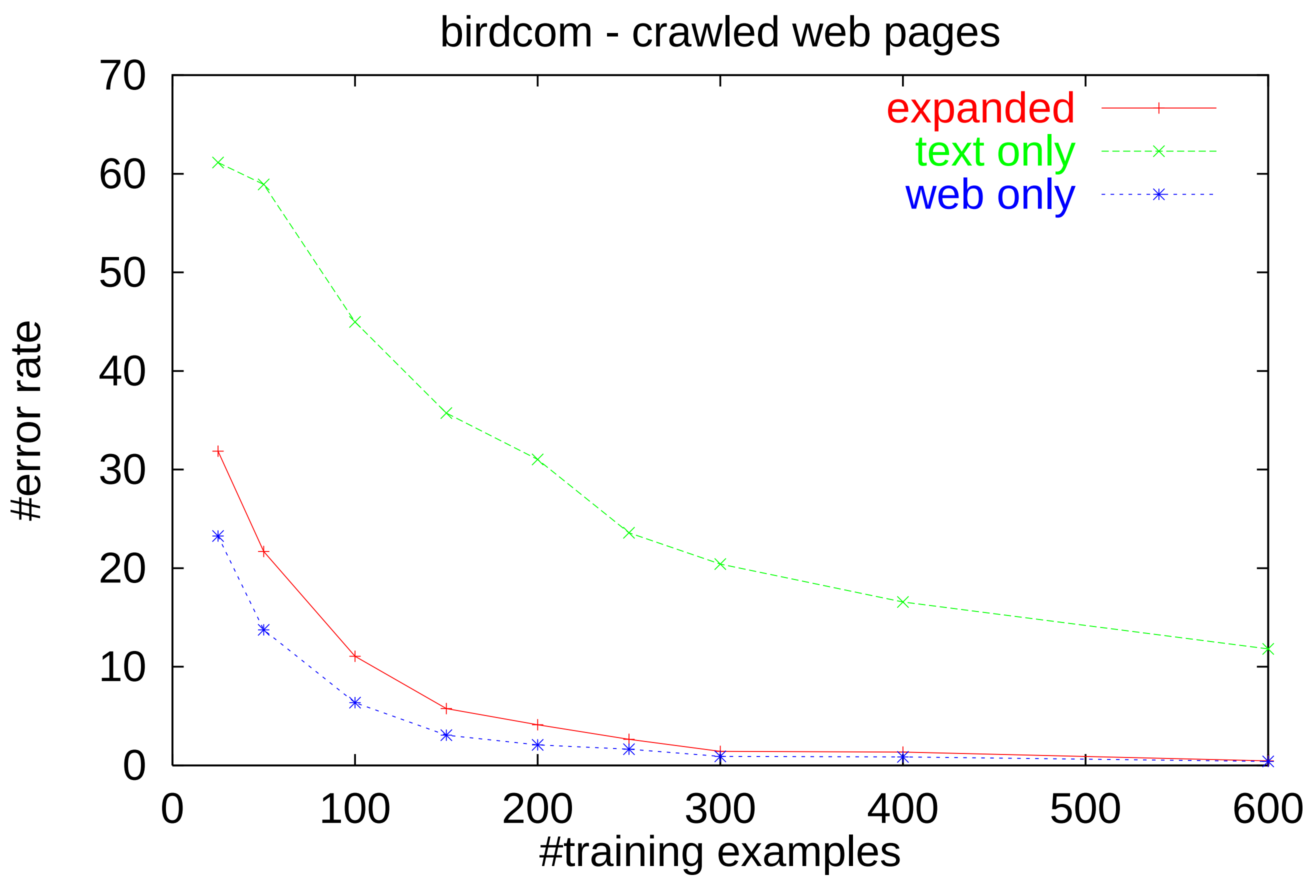
$$g_{p_i}(x) = 1 \Leftrightarrow x \text{ (seems to) be in position } p_i$$
$$g_{w_j}(x) = 1 \Leftrightarrow w_j \in Y \wedge \text{header Y (seems to) modify } x$$

Learning using Web-constructed features



- 84 sample pages from data integration system

Learning using Web-constructed features



- 174 automatically spidered pages

Learning without expansion:

phoenicopteriform $\leftarrow f_{flamingo} = 1$

trogoniform $\leftarrow f_{trogon} = 1$

piciform $\leftarrow f_{woodpeck} = 1$

piciform $\leftarrow f_{sapsuck} = 1$

piciform $\leftarrow f_{flicker} = 1$

piciform $\leftarrow f_{wryneck} = 1$

psittaciform $\leftarrow f_{parrot} = 1$

...

Learning with expansion:

phoenicopteriform $\leftarrow f_{flamingo} = 1$

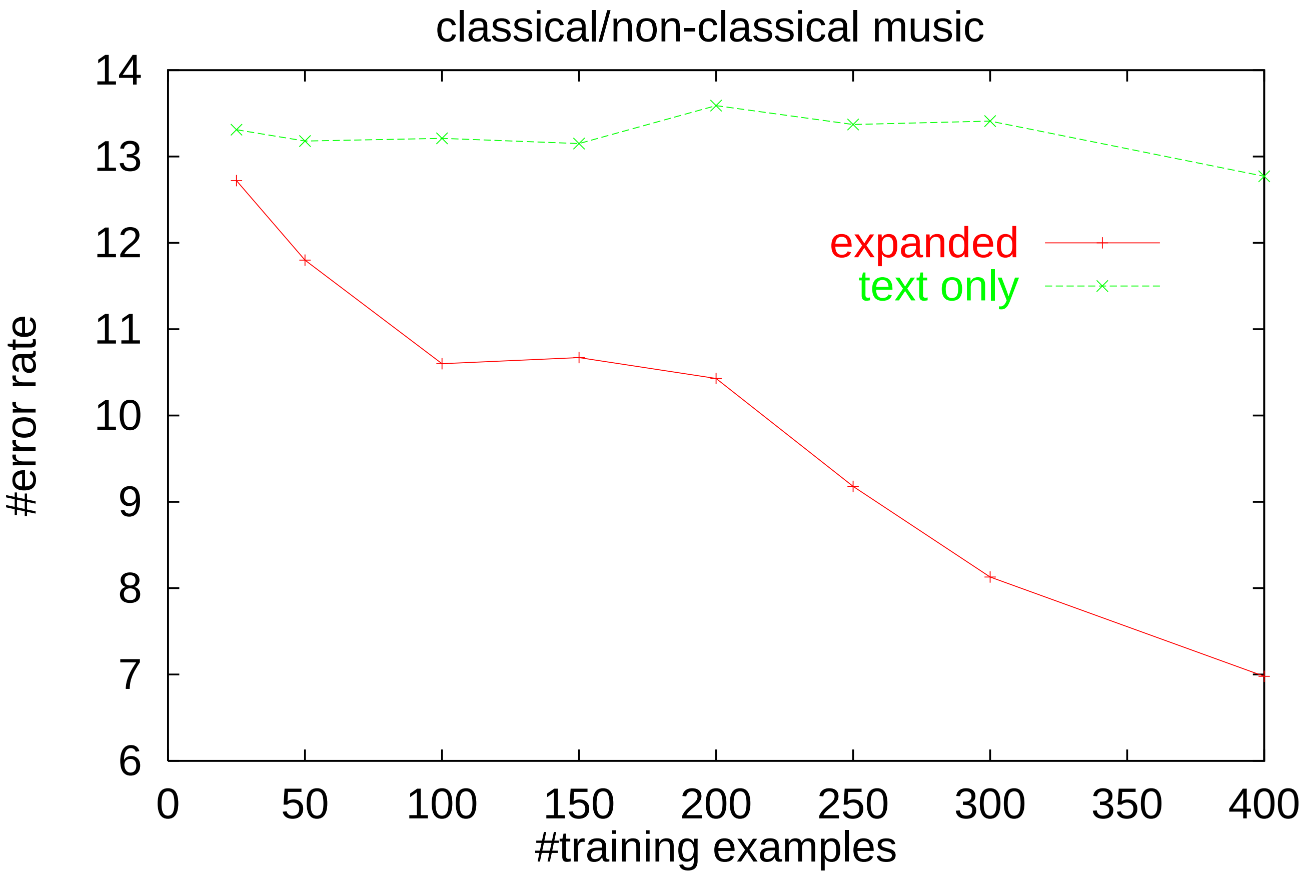
trogoniform $\leftarrow f_{trogon} = 1$

piciform $\leftarrow g_{Piciformes} = 1$

psittaciform $\leftarrow g_{Psittaciformes} = 1$

...

Learning using Web-constructed features



- Learning impossible without expansion

Summary/Conclusions

- Motivation: representation is key.
- Data model: how information is stored.

  Intermediate between IR model and DB model.
- WHIRL query language & implementation
- Results & applications
  - Efficient deduction without formal identifiers
  - Classification queries that generalize
  - Deduction with partial extraction
  - Learning from web data without extraction

    "Learning using the Web as background knowledge"