HTML Structure Meets Content

William W. Cohen

Read The Web, April 2006

Outline

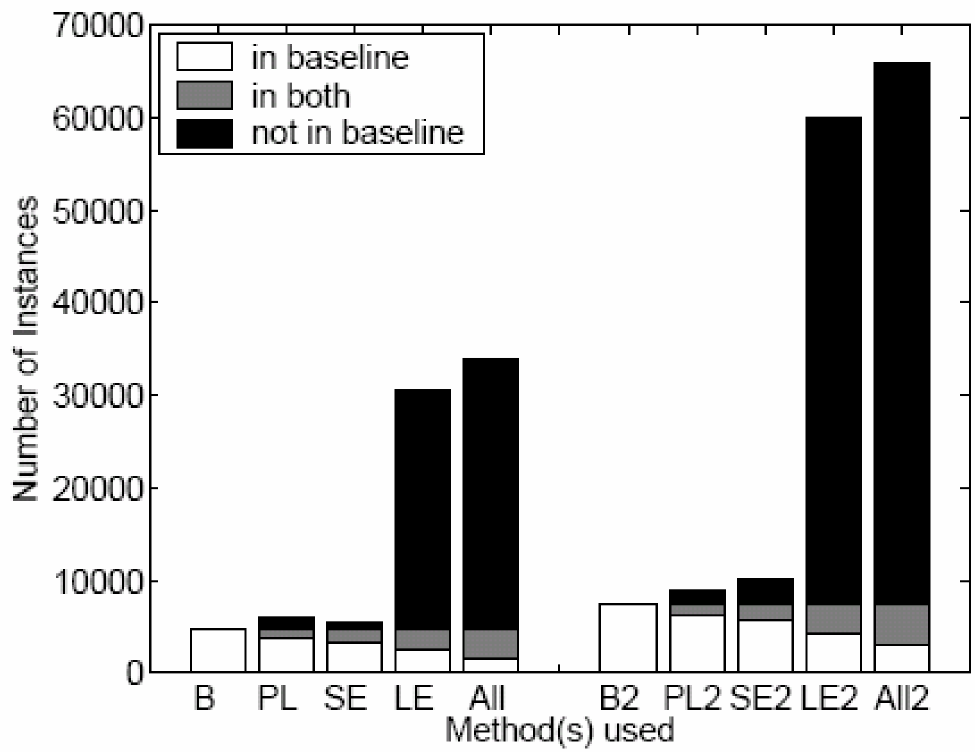
- Motivation: finding even simple structures like lists is useful, and *seems* like it should be easy.
- Cohen & Fan, 1999a: List-finding as *classification*.
- Cohen 1999b,2000: List-finding as *matching* structure to content.
- Cohen et al 2001, Cohen 2002, Blei et al 2002: List-finding as *learning* global content and local structure.

Observation: Recognizing Structure is Useful

```
LISTEXTRACTOR(seedExamples)
    documents = searchForDocuments(seedExamples)
    For each document in documents
        parseTree = ParseHTML(document)
        For each subtree in parseTree
            keyWords = findAllSeedsInTree(subtree)
            prefix = findBestPrefix(keyWords, subtree)
            suffix = findBestSuffix(keyWords, subtree)
            Add to wrapperTree from createWrapper(prefix, suffix))
        For each goodWrapper in wrapperTree
            Find extractions using goodWrapper
    Return list of extractions
```

Figure 14: High-level pseudocode for List Extractor

Observation: Recognizing Structure is Useful



Experiment 10: Number of correct instances of `Film` at precision .90 and .80. List Extractor gives a 7-fold increase at precision .90 and an 8-fold increase at precision .80.

Observation: HTML Structure is Meaningful and (Easily?) Recognizable

“Colorless green ideas sleep furiously.”

Exploding porpoises, over four score and seven, well before configuration.

- *Department of Computer and Information Sciences, University of New Jersey.* Citrus flavorings: green, marine, clean and under lien.
- *Computer Engineering Center, Lough Polytechnical Institute.* This, that page extensionally left to rights of manatees.
- *Electrical Engineering and Computer Science Dept, Bismark State College.* Tertiary; where cola substitutes are frequently underutilized.

This page under construction. (Last update: 9/23/98.)

Figure 1: Nonsense text with a meaningful structure.

List-finding as Classification

[Cohen & Fan, WWW 1999]

Learning to extract “simple lists” and “simple hotlists”.

HTML source for a simple list:

```
<html><head>... </head>
<body>
<h1>Editorial Board Members</h1>
<table> <tr>
    <td>G. R. Emlin, Lucent</td>
    <td>Harry Q. Bovik, Cranberry U</td></tr>
  <tr>
    <td>Bat Gangley, UC/Bovine</td>
    <td>Pheobe L. Mind, Lough Tech</td>
...
```

Extracted data:

G. R. Emlin, Lucent
Harry Q. Bovik, Cranberry U
...

HTML source for a simple hotlist:

```
<html><head>... </head>
<body><h1>Publications for Pheobe Mind</h1>
<ul>
<li>Optimization of fuzzy neural networks using
distributed parallel case-based genetic knowledge discovery
  (<a href=“buzz.pdf”>PDF</a>)</li>
<li>A linear-time version of GSAT
  (<a href=“peqnp.ps”>postscript</a>)</li>
...
```

Extracted data:

Optimization ... (PDF)	buzz.pdf
A linear-time version of ...	peqnp.ps
...	...

Figure 2: A simple list, a simple hotlist, and the data that would be extracted from each.

List-finding as classification

In a page containing a *simple list*, the structure extracted is a one-column relation containing a set of strings *s*1,...,*sN*, and each *si* is all the text that falls below some node *ni* in the parse tree. In a *simple hotlist*, the extracted structure is a two-column relation, containing a set of pairs <*s1,u1*>,...,<*sN,uN*>; each *si* is all the text that falls below some node *ni* in the parse tree; and each *ui* is a URL that is associated with some HTML anchor element *ai*that appears somewhere inside *ni*.

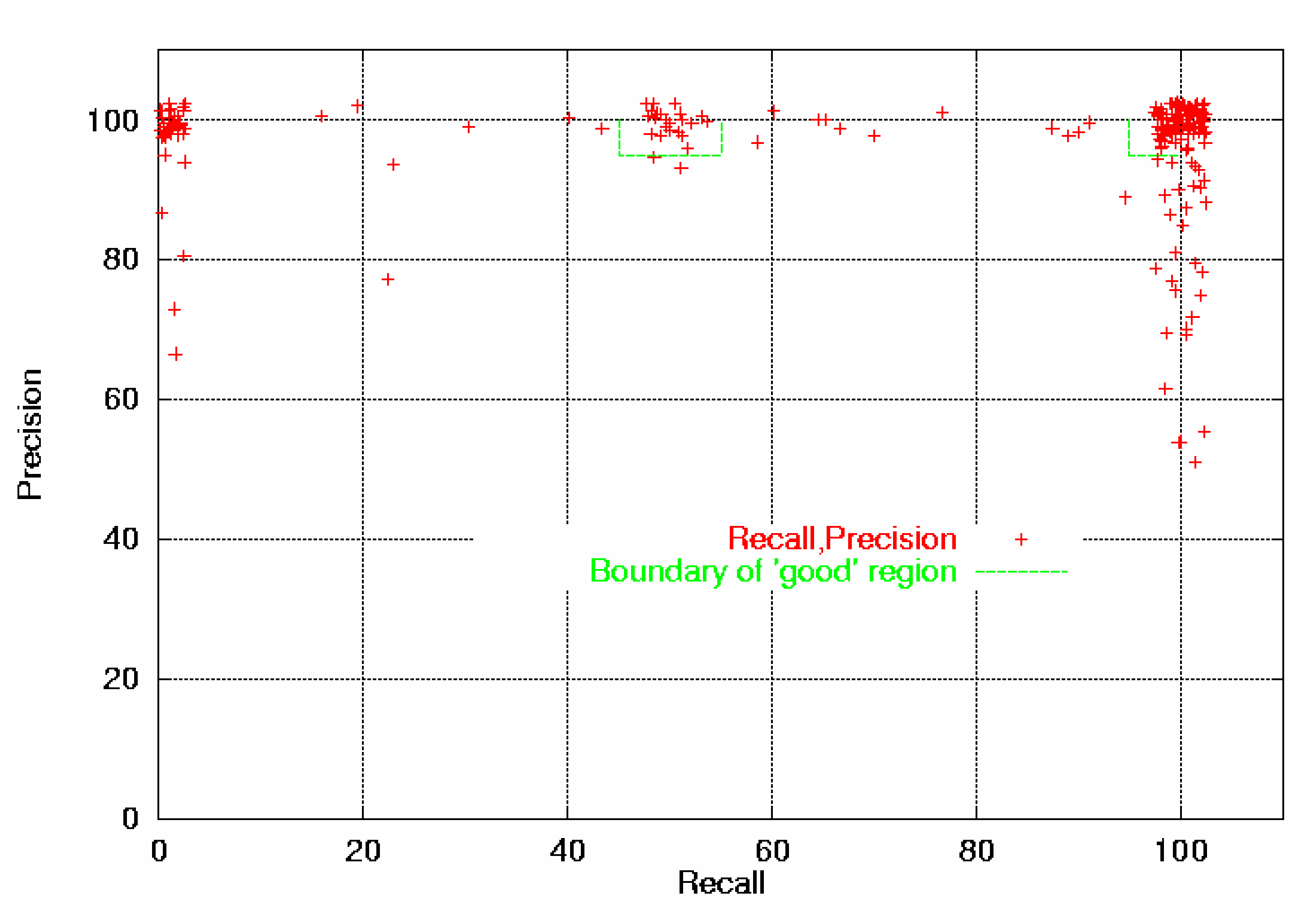
Technique: classify each node in the HTML tree as “extract” or “don’t extract”; then reconstruct the list or hotlist.

Evaluation: on a set of 84 pre-wrapped simple lists and simple hotlists (about 75% of a larger collection of wrapped pages).

List-finding as classification

- Simple Features:
  - Tag Name ("a", "p", "td")
  - Text Length, Non-white Text Length
  - Recursive Text Length, ...
  - Depth, NumChildren, NumSiblings
  - Parent tag, Ancestor tags, Child tags, Descendent tags
- Complex features:
  - tagSeqPosition *TSP(n)* = sequence of tags encountered walking from root to node *n*
  - NodePrefixCount*(n)* = |{ leaf *n'* | *TSP(n)* prefix of *TSP(n')*|}
  - NodeSuffixCount(*n)* = ...

List-finding as classification



List-finding as classification

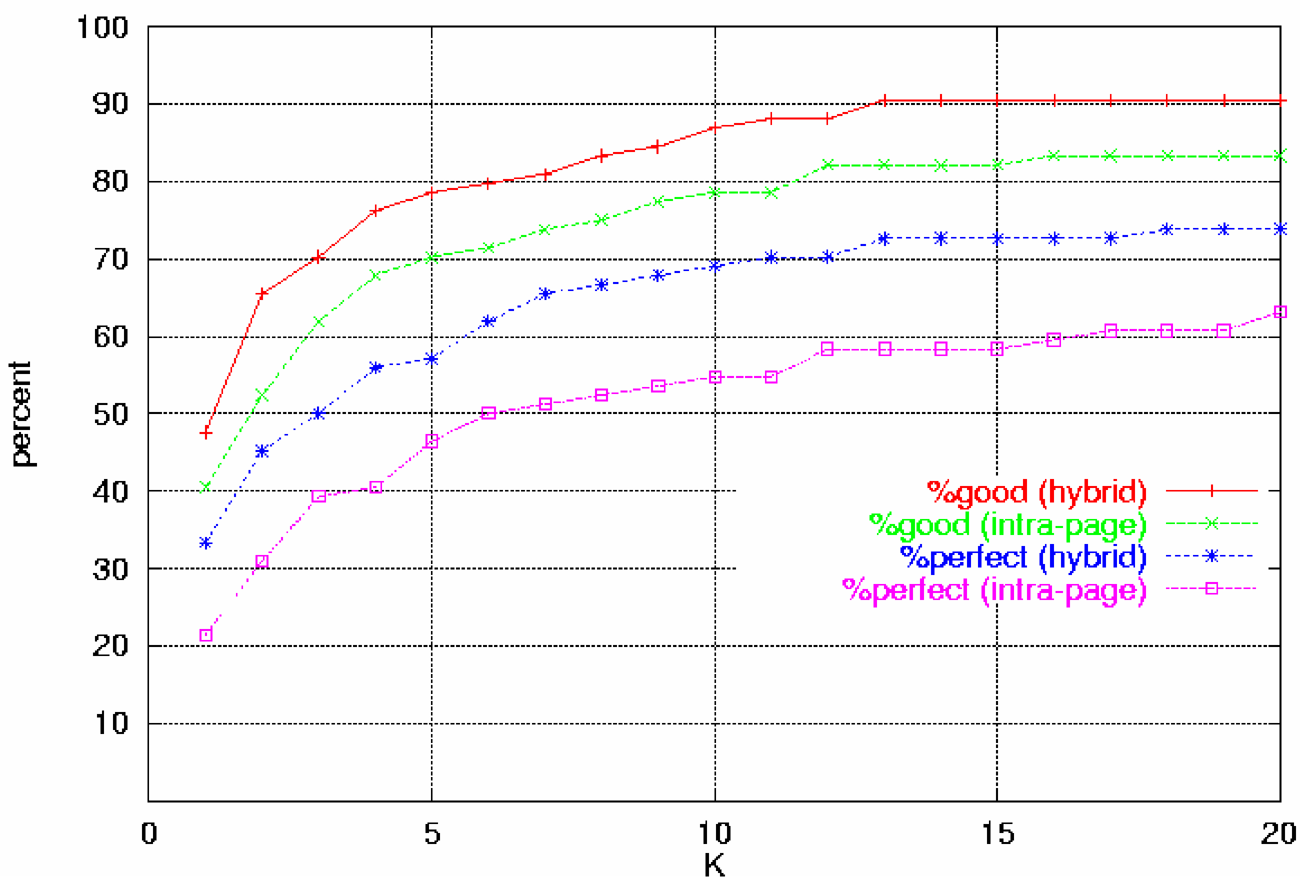
Figure: Performance of RIPPER in leave-one-page out experiments.

Performance Level	# pages reached
perfect	26/84 31%
good (ε =1%)	33/84 39%
good (ε =3%)	35/84 39%
good (ε =5%)	41/84 49%
good (ε =10%)	45/84 54%
good (ε =15%)	47/84 56%
good (ε =20%)	48/84 57%
good (ε =25%)	48/84 57%

Another Experiment:

Wrapper Induction: User Labels *K* Positive Examples (Hybrid: After Accepting/Rejecting Default Wrapper)

Figure: Performance of the intra-page learning method and the hybrid intra-page and page-independent learning method as the number of positive examples K is increased.



Outline

- Motivation: finding even simple structures like lists is useful, and *seems* like it should be easy.
- Cohen & Fan, 1999: List-finding as *classification*: kind of **disappointing**, only 30-50% of the pages were wrapped well.
- Cohen 2000a,2000b: List-finding as *matching* structure to content.
- Cohen et al 2001, Cohen 2002, Blei et al 2002: List-finding as *learning* global content and local structure.

Matching Structure To Content

To encode an HTML page in WHIRL, the page is first parsed. The HTML parse tree is then represented with the following EDB predicates.

- *elt(Id,Tag,Text,Position)* is true if *Id* is the identifier for a parse tree node, n, *Tag* is the HTML tag associated with n, *Text* is all of the text appearing in the subtree rooted at n, and *Position* is the sequence of tags encountered in traversing the path from the root to n. The value of *Position* is encoded as a a document containing a single term t_{pos}, which represents the sequence, *e.g.*, t_{pos}= *"html_body_ul_li"*.
- *attr(Id,AName,AValue)* is true if *Id* is the identifier for node n, *AName* is the name of an HTML attribute associated with n, and *AValue* is the value of that attribute.
- *path(FromId,ToId,Tags)* is true if *Tags* is the sequence of HTML tags encountered on the path between nodes *FromId* and *ToId*. This path includes both endpoints, and is defined if *FromId=ToId*.

As an example, wrappers for the pages in Figure 2 can be written using these predicates as follows.

page1(NameAffil) ←
  elt(_, _, NameAffil, "html_body_table_tr_td").
page2(Title,Url) ←
  elt(ContextElt, _, Title, "html_body_ul_li")
  ∧ *path(ContextElt, AnchorElt, "li_a")*
  ∧ *attr(AnchorElt, "href", Url).*

HTML source for a simple list:

```
<html><head>...</head>
<body>
<h1>Editorial Board Members</h1>
<table> <tr>
    <td>G. R. Emlin, Lucent</td>
    <td>Harry Q. Bovik, Cranberry U</td></tr>
  <tr>
    <td>Bat Gangley, UC/Bovine</td>
    <td>Pheobe L. Mind, Lough Tech</td>
...
```

Extracted data:

G. R. Emlin, Lucent
Harry Q. Bovik, Cranberry U
...

Matching Structure To Content

To encode an HTML page in WHIRL, the page is first parsed. The HTML parse tree is then represented with the following EDB predicates.

- *elt(Id,Tag,Text,Position)* is true if *Id* is the identifier for a parse tree node, n, *Tag* is the HTML tag associated with n, *Text* is all of the text appearing in the subtree rooted at n, and *Position* is the sequence of tags encountered in traversing the path from the root to n. The value of *Position* is encoded as a a document containing a single term t_{pos}, which represents the sequence, *e.g.*, t_{pos} = "html_body_ul_li".
- *attr(Id,AName,AValue)* is true if *Id* is the identifier for node n, *AName* is the name of an HTML attribute associated with n, and *AValue* is the value of that attribute.
- *path(FromId,ToId,Tags)* is true if *Tags* is the sequence of HTML tags encountered on the path between nodes *FromId* and *ToId*. This path includes both endpoints, and is defined if *FromId*=*ToId*.

As an example, wrappers for the pages in Figure 2 can be written using these predicates as follows.

page1(NameAffil) ←
  elt(_, _, NameAffil, "html_body_table_tr_td").
page2(Title,Url) ←
  elt(ContextElt, _, Title, "html_body_ul_li")
  ∧ *path(ContextElt, AnchorElt, "li_a")*
  ∧ *attr(AnchorElt, "href", Url).*

HTML source for a simple hotlist:

<html><head>. . . </head>
<body><h1>Publications for Pheobe Mind</h1>

Optimization of fuzzy neural networks using distributed parallel case-based genetic knowledge discovery
  (PDF)
A linear-time version of GSAT
  (postscript)
. . .

Extracted data:

Optimization . . . (PDF)	buzz.pdf
A linear-time version of . . .	peqnp.ps
. . .	. . .

Matching Structure To Content

Key point: every simple list or hotlist can be written *just like this*: you just need to fill in

- one root-node path for simple lists;
- one root-node path and one node-node path for simple hotlists.

➔ Given a web page with *N* nodes, there are *O(N*^{2}*)* possible wrappers, which can be *enumerated and scored.*

As an example, wrappers for the pages in Figure 2 can be written using these predicates as follows.

page1(NameAffil) ←
  elt(_, _, NameAffil, $\boxed{\text{“html\_body\_table\_tr\_td”}}$*).*
page2(Title,Url) ←
  elt(ContextElt, _, Title, $\boxed{\text{“html\_body\_ul\_li”}}$*)*
  ∧ *path(ContextElt, AnchorElt,* $\boxed{\text{“li\_a”}}$*)*
  ∧ *attr(AnchorElt, “href”, Url).*

HTML source for a simple hotlist:

<html><head>... </head>
<body><h1>Publications for Pheobe Mind</h1>

Optimization of fuzzy neural networks using distributed parallel case-based genetic knowledge discovery
  (PDF)
A linear-time version of GSAT
  (postscript)
...

Extracted data:

Optimization ... (PDF)	buzz.pdf
A linear-time version of ...	peqnp.ps
...	...

Enumerating and Scoring Wrappers

```
fruitful_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,_,_),
            (Path1a=Path1 ∧ Path2a=Path2) ).
possible_piece(Path1,Path2) ←
   elt(TextElt, _, _, Path1)
   ∧ elt(AnchorElt, _, "a", _)
   ∧ attr(AnchorElt, "href", _)
   ∧ path(TextElt, AnchorElt, Path2).
extracted_by(Path1,Path2,TextElt,AnchorElt) ←
   elt(TextElt, _, _, Path1)
   ∧ path(TextElt, AnchorElt, Path2).
```

```
anchorlike_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,TElt,AElt),
      (Path1a=Path1 ∧ Path2a=Path2
       ∧ elt(TElt,_,Text1,_) ∧ elt(AElt,_,Text2,_) ∧ Text1∼Text2 ).

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.
```

Figure 3: WHIRL programs for recognizing plausible structures in an HTML page. (See text for explanation.)

Enumerating and Scoring Wrappers

```
possible_piece(Path1,Path2) ←
   elt(TextElt, _, _, Path1)
   ∧ elt(AnchorElt, _, "a", _)
   ∧ attr(AnchorElt, "href", _)
   ∧ path(TextElt, AnchorElt, Path2).
extracted_by(Path1,Path2,TextElt,AnchorElt) ←
   elt(TextElt, _, _, Path1)
   ∧ path(TextElt, AnchorElt, Path2).
```

Enumerates all Path1, Path2 such that Path1 goes from root to *n,* and Path2 goes from *n* to an anchor element.

```
fruitful_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,_,_),
         (Path1a=Path1 ∧ Path2a=Path2) ).
possible_piece(Path1,Path2) ←
   elt(TextElt, _, _, Path1)
   ∧ elt(AnchorElt, _, "a", _)
   ∧ attr(AnchorElt, "href", _)
   ∧ path(TextElt, AnchorElt, Path2).
extracted_by(Path1,Path2,TextElt,AnchorElt) ←
   elt(TextElt, _, _, Path1)
   ∧ path(TextElt, AnchorElt, Path2).
```

```
anchorlike_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,TElt,AElt),
      (Path1a=Path1 ∧ Path2a=Path2
       ∧ elt(TElt,_,Text1,_) ∧ elt(AElt,_,Text2,_) ∧ Text1~Text2 ).

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.
```

Figure 3: WHIRL programs for recognizing plausible structures in an HTML page. (See text for explanation.)

Enumerating and Scoring Wrappers

$$\begin{array}{l} \textit{fruitful\_piece}(\textit{Path1},\textit{Path2}) \leftarrow \\ \quad \textit{possible\_piece}(\textit{Path1},\textit{Path2}) \wedge \\ \quad \textit{many}(\ \textit{extracted\_by}(\textit{Path1a},\textit{Path2a},\_,\_), \\ \qquad\qquad (\textit{Path1a}{=}\textit{Path1} \wedge \textit{Path2a}{=}\textit{Path2})\ ). \end{array}$$

Enumerates and scores Path1,Path2 according to how many things are extracted by that wrapper.

$$\begin{array}{l} \textit{fruitful\_piece}(\textit{Path1},\textit{Path2}) \leftarrow \\ \quad \textit{possible\_piece}(\textit{Path1},\textit{Path2}) \wedge \\ \quad \textit{many}(\ \textit{extracted\_by}(\textit{Path1a},\textit{Path2a},\_,\_), \\ \qquad\qquad (\textit{Path1a}{=}\textit{Path1} \wedge \textit{Path2a}{=}\textit{Path2})\ ). \\ \textit{possible\_piece}(\textit{Path1},\textit{Path2}) \leftarrow \\ \quad \textit{elt}(\textit{TextElt},\ \_,\ \_,\ \textit{Path1}) \\ \quad \wedge\ \textit{elt}(\textit{AnchorElt},\ \_,\ \text{``a''},\ \_) \\ \quad \wedge\ \textit{attr}(\textit{AnchorElt},\ \text{``href''},\ \_) \\ \quad \wedge\ \textit{path}(\textit{TextElt},\ \textit{AnchorElt},\ \textit{Path2}). \\ \textit{extracted\_by}(\textit{Path1},\textit{Path2},\textit{TextElt},\textit{AnchorElt}) \leftarrow \\ \quad \textit{elt}(\textit{TextElt},\ \_,\ \_,\ \textit{Path1}) \\ \quad \wedge\ \textit{path}(\textit{TextElt},\ \textit{AnchorElt},\ \textit{Path2}). \end{array}$$

$$\begin{array}{l} \textit{anchorlike\_piece}(\textit{Path1},\textit{Path2}) \leftarrow \\ \quad \textit{possible\_piece}(\textit{Path1},\textit{Path2}) \wedge \\ \quad \textit{many}(\ \textit{extracted\_by}(\textit{Path1a},\textit{Path2a},\textit{TElt},\textit{AElt}), \\ \qquad (\textit{Path1a}{=}\textit{Path1} \wedge \textit{Path2a}{=}\textit{Path2} \\ \qquad \wedge\ \textit{elt}(\textit{TElt},\_,\textit{Text1},\_) \wedge \textit{elt}(\textit{AElt},\_,\textit{Text2},\_) \wedge \textit{Text1}{\sim}\textit{Text2}\ ). \\ \\ \textit{R\_like\_piece}(\textit{Path1},\textit{Path2}) \leftarrow \\ \quad \textit{possible\_piece}(\textit{Path1},\textit{Path2}) \wedge \\ \quad \textit{many}(\ \textit{R\_extracted\_by}(\textit{Path1a},\textit{Path2a},\_,\_), \\ \qquad\qquad (\textit{Path1a}{=}\textit{Path1} \wedge \textit{Path2a}{=}\textit{Path2})\ ). \\ \textit{R\_extracted\_by}(\textit{Path1},\textit{Path2},\textit{TextElt},\textit{AnchorElt}) \leftarrow \\ \quad \textit{elt}(\textit{TextElt},\ \_,\ \textit{Text},\ \textit{Path1}) \\ \quad \wedge\ \textit{path}(\textit{TextElt},\ \textit{AnchorElt},\ \textit{Path2}) \\ \quad \wedge\ R(X) \wedge \textit{Text}{\sim}X. \end{array}$$

Figure 3: WHIRL programs for recognizing plausible structures in an HTML page. (See text for explanation.)

Enumerating and Scoring Wrappers

“Soft” predicate, true/false according to TFIDF similarity

```
nchorlike_piece(Path1,Path2) ←
  possible_piece(Path1,Path2) ∧
  many( extracted_by(Path1a,Path2a,TElt,AElt),
      (Path1a=Path1 ∧ Path2a=Path2
       ∧ elt(TElt,_,Text1,_) ∧ elt(AElt,_,Text2,_) ∧ Text1~Text2 ).
```

Enumerates and scores Path1,Path2 according to how many things are extracted by that wrapper such that the text under node *n* is close to the text under the anchor element.

e.g., discounts wrappers where *n* is close to the root.

```
fruitful_piece(Path1,Path2) ←
  possible_piece(Path1,Path2) ∧
  many( extracted_by(Path1a,Path2a,_,_),
        (Path1a=Path1 ∧ Path2a=Path2) ).
possible_piece(Path1,Path2) ←
  elt(TextElt, _, _, Path1)
  ∧ elt(AnchorElt, _, “a”, _)
  ∧ attr(AnchorElt, “href”, _)
  ∧ path(TextElt, AnchorElt, Path2).
extracted_by(Path1,Path2,TextElt,AnchorElt) ←
  elt(TextElt, _, _, Path1)
  ∧ path(TextElt, AnchorElt, Path2).
```

```
anchorlike_piece(Path1,Path2) ←
  possible_piece(Path1,Path2) ∧
  many( extracted_by(Path1a,Path2a,TElt,AElt),
     (Path1a=Path1 ∧ Path2a=Path2
      ∧ elt(TElt,_,Text1,_) ∧ elt(AElt,_,Text2,_) ∧ Text1~Text2 ).

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.
```

Figure 3: WHIRL programs for recognizing plausible structures in an HTML page. (See text for explanation.)

Enumerating and Scoring 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.
```

Enumerates and scores Path1,Path2 according to how many things are extracted by that wrapper such that the text under node *n* is close to the text of some *X* such that *R(X)* is true

R(X) is "content" information, as a K-NN classifier

```
fruitful_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,_,_),
           (Path1a=Path1 ∧ Path2a=Path2) ).
possible_piece(Path1,Path2) ←
   elt(TextElt, _, _, Path1)
   ∧ elt(AnchorElt, _, "a", _)
   ∧ attr(AnchorElt, "href", _)
   ∧ path(TextElt, AnchorElt, Path2).
extracted_by(Path1,Path2,TextElt,AnchorElt) ←
   elt(TextElt, _, _, Path1)
   ∧ path(TextElt, AnchorElt, Path2).
```

```
anchorlike_piece(Path1,Path2) ←
   possible_piece(Path1,Path2) ∧
   many( extracted_by(Path1a,Path2a,TElt,AElt),
      (Path1a=Path1 ∧ Path2a=Path2
       ∧ elt(TElt,_,Text1,_) ∧ elt(AElt,_,Text2,_) ∧ Text1∼Text2 ).

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.
```

Figure 3: WHIRL programs for recognizing plausible structures in an HTML page. (See text for explanation.)

Results

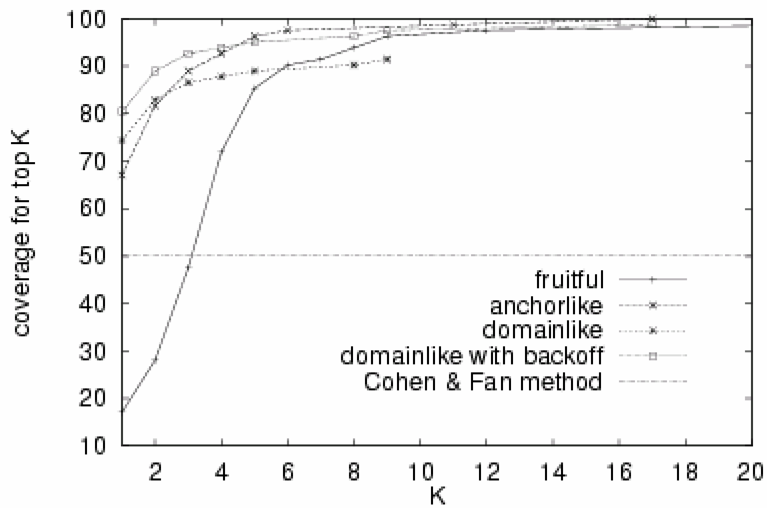


Figure 4: Performance of ranking heuristics that use little or no page-specific information.

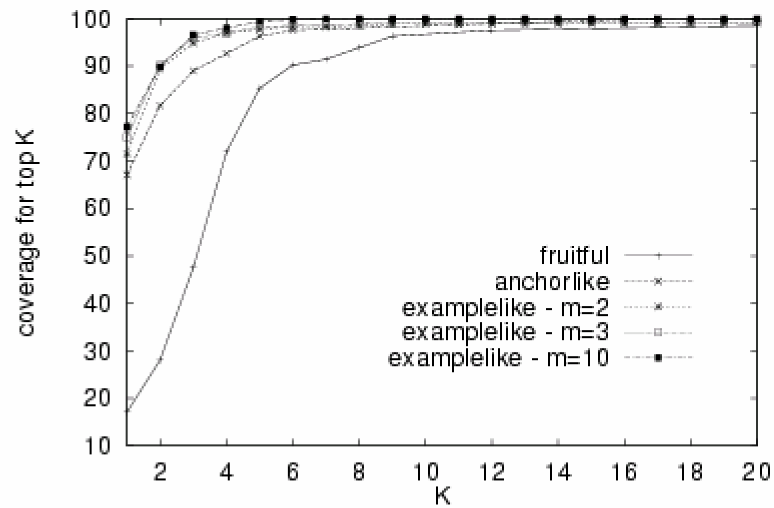


Figure 5: Performance of ranking heuristics that use page-specific training examples.

Oklahoma	dietitians
Yukon	Yukon codpiece
Vermont	Vermont
British Columbia	British Columbia Talmudizations
Oklahoma	Oklahoma
Wisconsin	Wisconsin
New Jersey	New Jersey incorrigible blubber
Alaska	Alaska
New Brunswick	
New Mexico	New Mexico cryptogram

Table 2: Ten US States and Canadian Provinces, before and after corruption with $c = 1$.

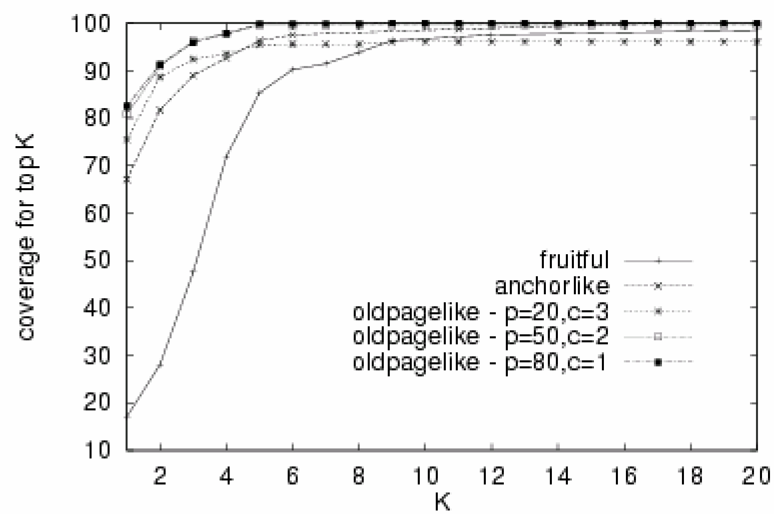


Figure 6: Performance of ranking heuristics that use text extracted from an previous version of the page.

Using Extracted Lists

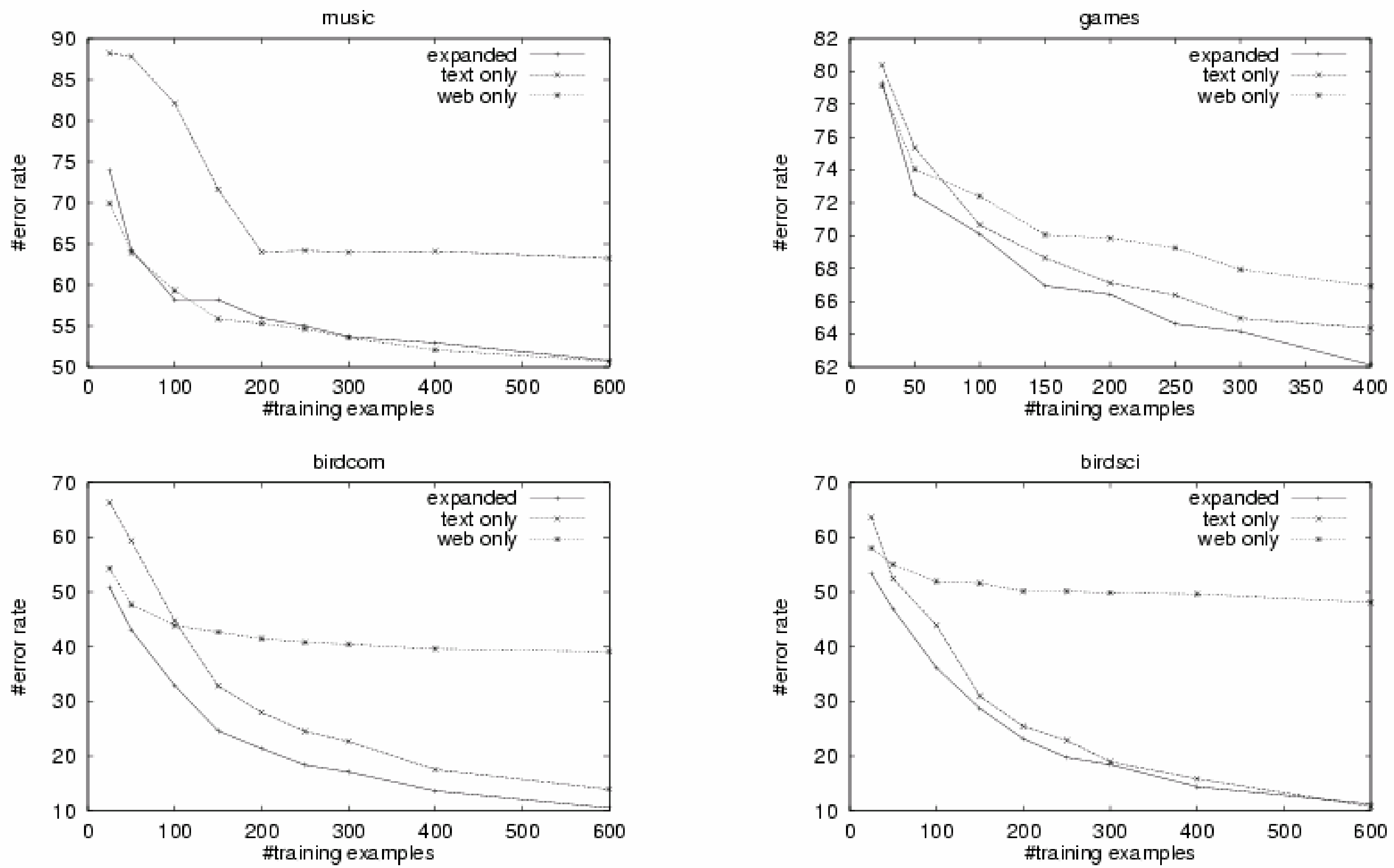
- Experiments above describe semi-automated techniques for wrapper learning: some user intervention is needed.
- Can you use the wrappers without letting a user check them?
- Idea [Cohen, ICML2000]:
  - Start with some textual examples for a classification problem (e.g., names of classical/rock musicians)
  - Use these examples as “seeds” R and find a bunch of simple lists $L_1, L_2, \ldots., L_m$
  - Use each list as a *feature:* F_i true for x iff x (approximately) matches something in list L_i.
  - Also: used features for header words that seemed to modify the matching element of the list (kind of like anchor text).

Using Extracted Lists

Table 1. Benchmark problems used in the experiments.

	#example	#class	#terms	#pages	(Mb)	#features added	%examples expanded
music	1010	20	1600	217	(11.7)	1890	68.7
games	791	6	1133	177	(2.5)	1169	53.0
birdcom	915	22	674	83	(2.2)	918	99.9
birdsci	915	22	1738	83	(2.2)	533	99.9

Using Extracted Lists



Using Extracted Lists

Table 1. Benchmark problems used in the experiments.

	#example	#class	#terms	#pages	(Mb)	#features added	%examples expanded
music	1010	20	1600	217	(11.7)	1890	68.7
games	791	6	1133	177	(2.5)	1169	53.0
birdcom	915	22	674	83	(2.2)	918	99.9
birdsci	915	22	1738	83	(2.2)	533	99.9

Table 5. Error rate of C4.5 and BoosTexter on the benchmark problems.

	RIPPER			C4.5			BoosTexter		
		avg %error			avg %error			avg %error	
	W-L-T	expand	text	W-L-T	expand	text	W-L-T	expand	text
music	86-0-14	51.5	58.3	100-0-0	49.3	59.6	100-0-0	**43.4**	58.1
games	29-7-64	65.8	67.2	13-6-81	68.2	68.6	16-1-83	**61.8**	63.1
birdcom	77-2-21	**21.2**	27.7	97-0-3	31.8	40.7	65-1-34	21.3	25.3
birdsci	35-8-57	**23.6**	26.4	46-4-50	37.3	39.0	35-6-59	25.4	26.7

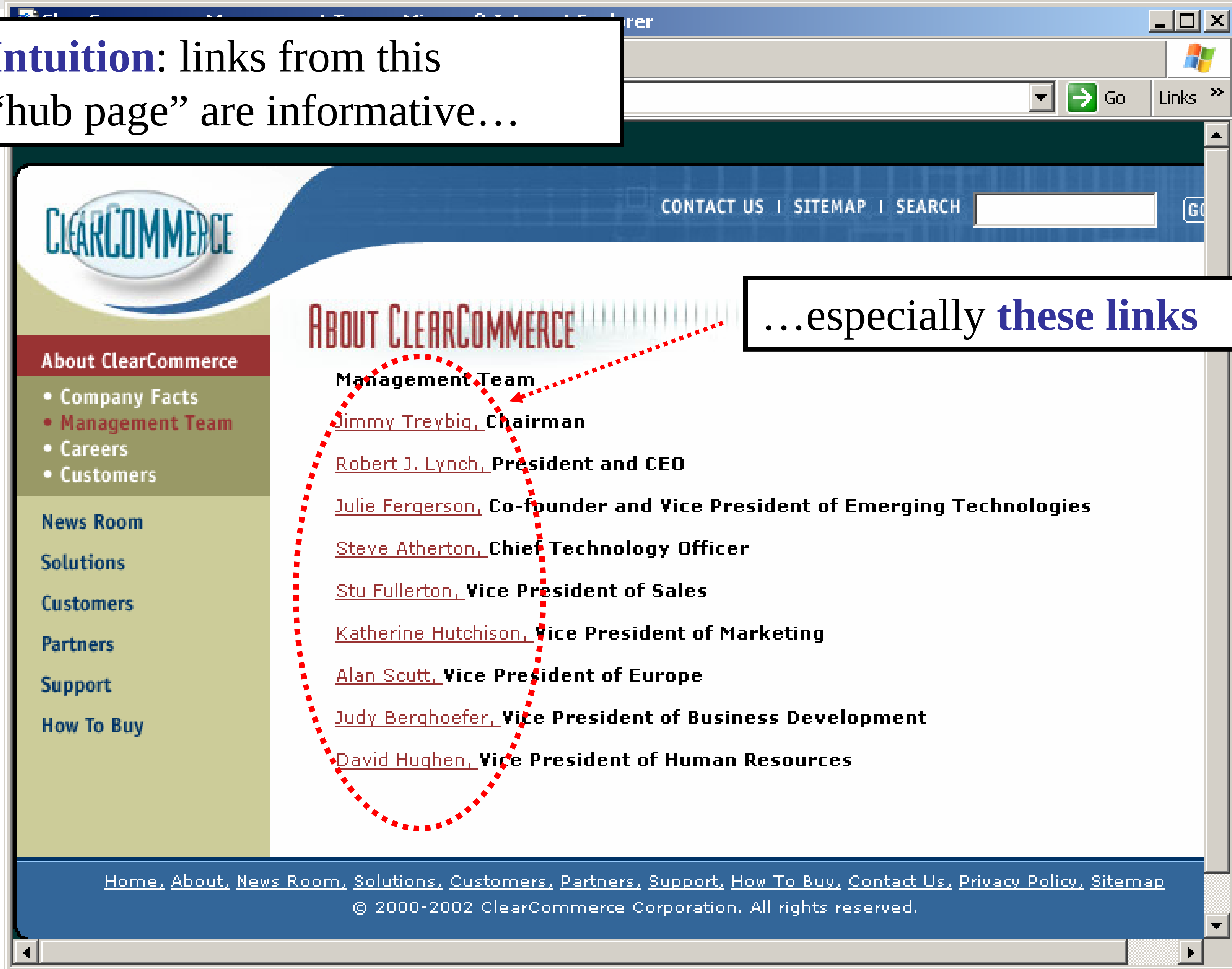
Outline

- Motivation: finding even simple structures like lists is useful, and *seems* like it should be easy.
- Cohen & Fan, 1999a: List-finding as *classification*: kind of **disappointing**, only 30-50% of the pages were wrapped well.
- Cohen 1999b,2000: List-finding as *matching* structure to content: seems to be **effective** even with moderately good models of content.
- Cohen et al 2001, Cohen 2002, Blei et al 2002: List-finding as *learning* global content and local structure.

- Previous work in page classification using links:
  - Exploit hyperlinks (Slattery&Mitchell 2000; Cohn&Hofmann, 2001; Joachims 2001): Documents pointed to by the same “hub” should have the same class.
- What’s new in this paper (Cohen NIPS 2002):
  - Use structure of hub pages (as well as structure of site graph) to find better “hubs”
  - Adapt an existing “wrapper learning” system to find structure, on the task of classifying “executive bio pages”.

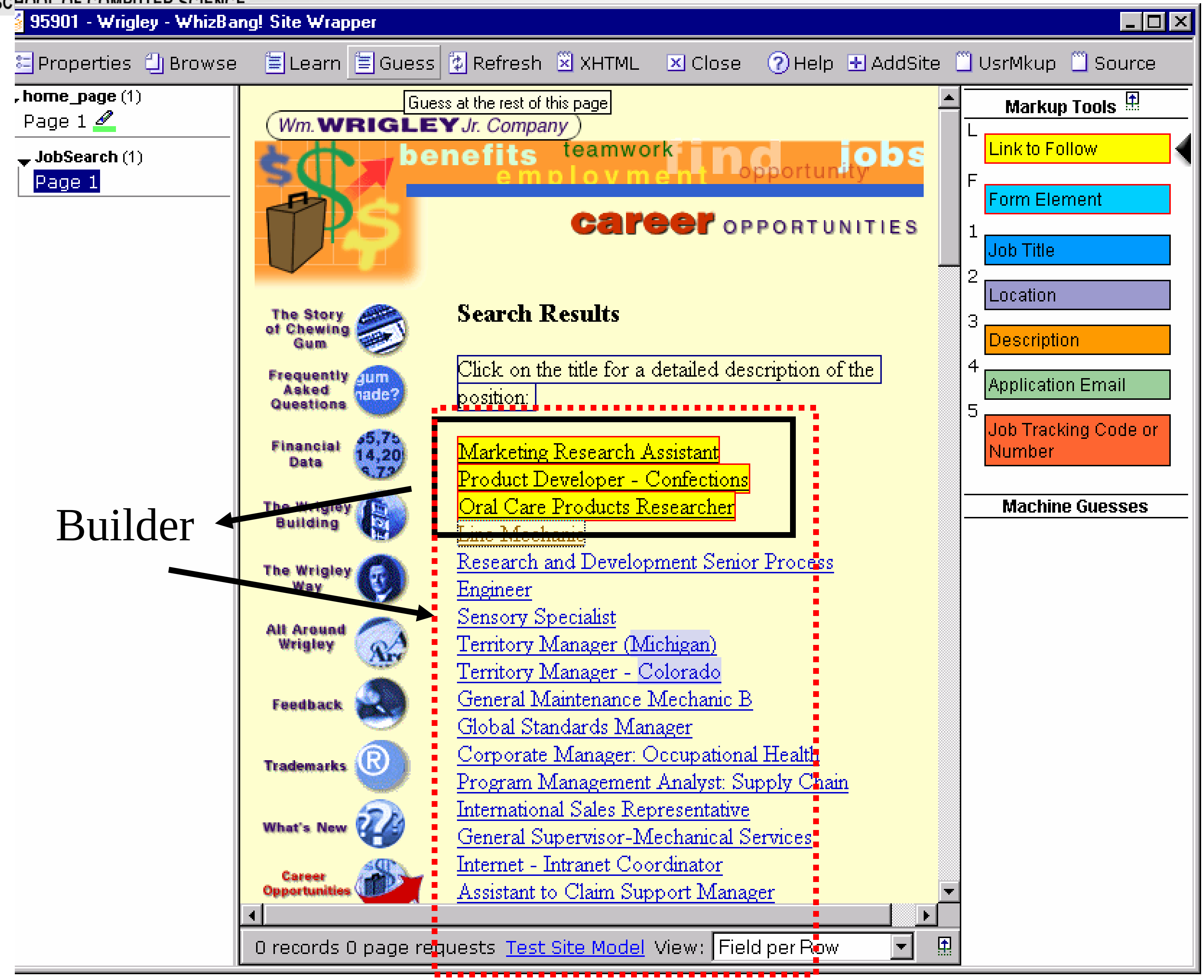
Intuition: links from this "hub page" are informative…

…especially **these links**



Background: “wrapper” learning

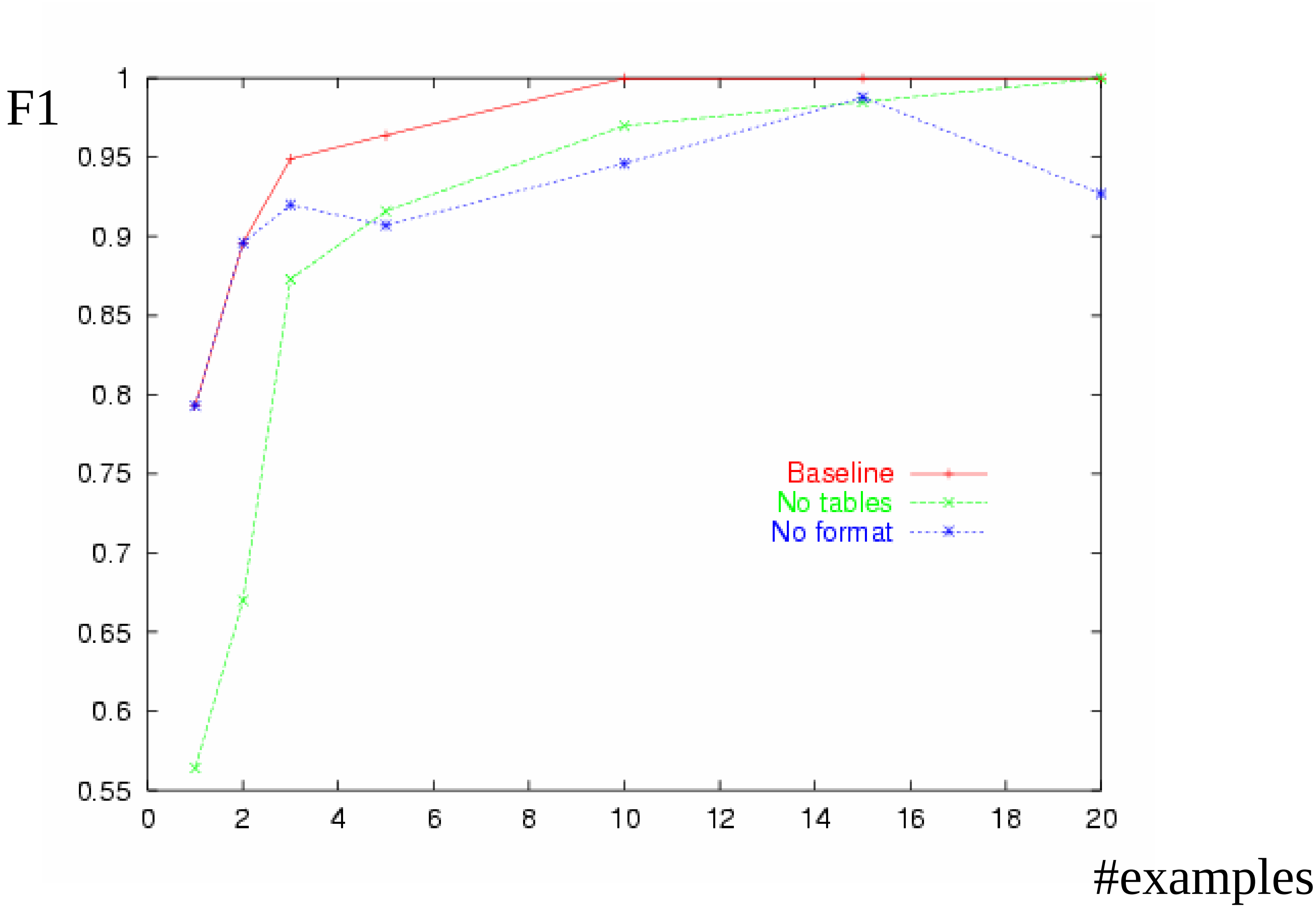
- System is based on a number of “**builders**”:
  - Infer a “structure” (e.g. a list, table column, etc) from few positive examples.
  - A “structure” *extracts* all its members
    - *f(page)* = { *x: x* is a “structure element” on *page* }
- A master algorithm co-ordinates the “builders”
- Add/remove “builders” to optimize performance on a domain (Cohen,Hurst&Jensen WWW-2002)
- Some builder usually obtains a good generalization from only **2-3 positive examples**


95901 - Wrigley - WhizBang! Site Wrapper
Properties Browse Learn Guess Refresh XHTML Close Help AddSite UsrMkup Source
home_page (1)
Page 1
JobSearch (1)
Page 1
Guess at the rest of this page
Wm. WRIGLEY Jr. Company
benefits teamwork find jobs employment opportunity
career OPPORTUNITIES
The Story of Chewing Gum
Frequently Asked Questions
Financial Data
The Wrigley Building
The Wrigley Way
All Around Wrigley
Feedback
Trademarks
What's New
Career Opportunities
Search Results
Click on the title for a detailed description of the position:
Marketing Research Assistant
Product Developer - Confections
Oral Care Products Researcher
Line Mechanic
Research and Development Senior Process Engineer
Sensory Specialist
Territory Manager (Michigan)
Territory Manager - Colorado
General Maintenance Mechanic B
Global Standards Manager
Corporate Manager: Occupational Health
Program Management Analyst: Supply Chain
International Sales Representative
General Supervisor-Mechanical Services
Internet - Intranet Coordinator
Assistant to Claim Support Manager
0 records 0 page requests Test Site Model View: Field per Row
Markup Tools
L Link to Follow
F Form Element
1 Job Title
2 Location
3 Description
4 Application Email
5 Job Tracking Code or Number
Machine Guesses
Builder


95901 - Wrigley - WhizBang! Site Wrapper
Properties
Browse
Learn
Guess
Refresh
XHTML
Close
Help
AddSite
UsrMkup
Source
home_page (1)
Page 1
JobSearch (1)
Page 1
jobpage (55)
Page 1
Page 2
Page 3
Page 4
Page 5
Wm. WRIGLEY Jr. Company
benefits
teamwork
find
jobs
employment
opportunity
career OPPORTUNITIES
The Story of Chewing Gum
Frequently Asked Questions
Financial Data
The Wrigley Building
The Wrigley Way
All Around Wrigley
Feedback
Trademarks
What's New
Career Opportunities
Search Results
Click on the title for a detailed description of the position:
Marketing Research Assistant
Product Developer - Confections
Oral Care Products Researcher
Line Mechanic
Research and Development Senior Process Engineer
Sensory Specialist
Territory Manager (Michigan)
Territory Manager - Colorado
General Maintenance Mechanic B
Global Standards Manager
Corporate Manager: Occupational Health
Program Management Analyst: Supply Chain
International Sales Representative
General Supervisor-Mechanical Services
Internet - Intranet Coordinator
Assistant to Claim Support Manager
Markup Tools
L
Link to Follow
F
Form Element
1
Job Title
2
Location
3
Description
4
Application Email
5
Job Tracking Code or Number
Machine Guesses
Next guess
0 records 0 page requests
Test Site Model
View:
Field per Row

Experimental results:

2-3 examples leads to high average accuracy



Background: “co-training” (Mitchell&Blum, ‘98)

- Suppose examples are of the form (x_1,x_2,y) where x_1,x_2 are **independent** (given y), and where each x_i is sufficient for classification, and **unlabeled** examples are cheap.
  - (E.g., x_1 = bag of words, x_2 = bag of links).

$\approx$

- Co-training algorithm:
  1. Use x_1’s (on labeled data D) to train $f_1(x)=y$
  2. Use f_1 to label additional unlabeled examples U.
  3. Use x_2’s (on labeled part of U+D to train $f_1(x)=y$
  4. Repeat . . .

Simple 1-step co-training for web pages

f_1 is a bag-of-words page classifier, and S is web site containing unlabeled pages.

- *Feature construction*. Represent a page xin S as a bag of pages that link to x("bag of hubs").
- *Learning.* Learn f_2 from the bag-of-hubs examples, labeled with f_1
- *Labeling.* Use f_2(x) to label pages from $\mathcal{S}$.

$Idea. use one round of co›training to bootstrap the bag›of words classifier to one that uses site›specific features x_2 \triangleleft f_2$

Improved 1-step co-training for web pages

Feature construction.

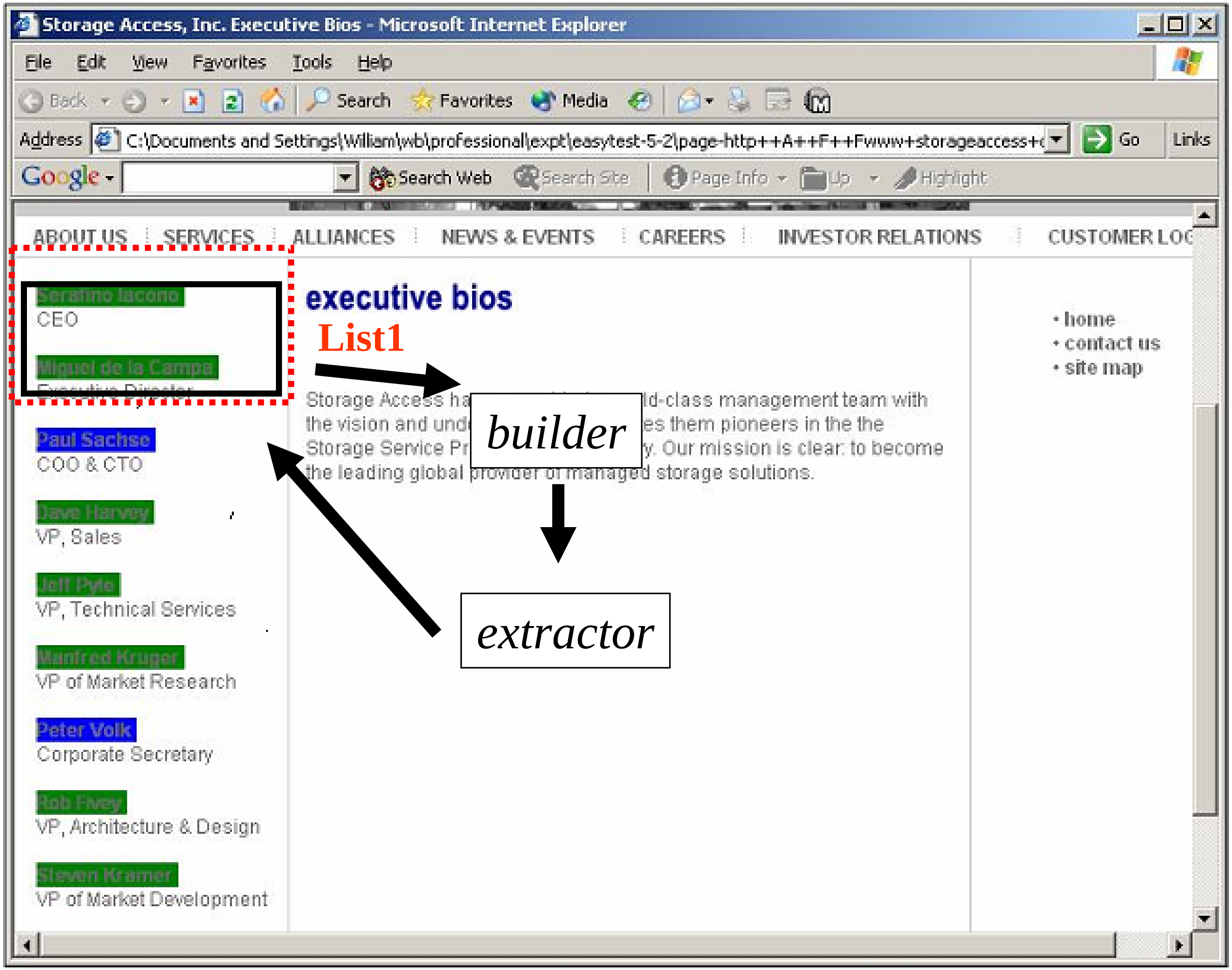
- Label an anchor *a* in S as ***positive*** iff it points to a positive page x (according to f_1). Let D = –(x',a): a is a positive anchor on x'"

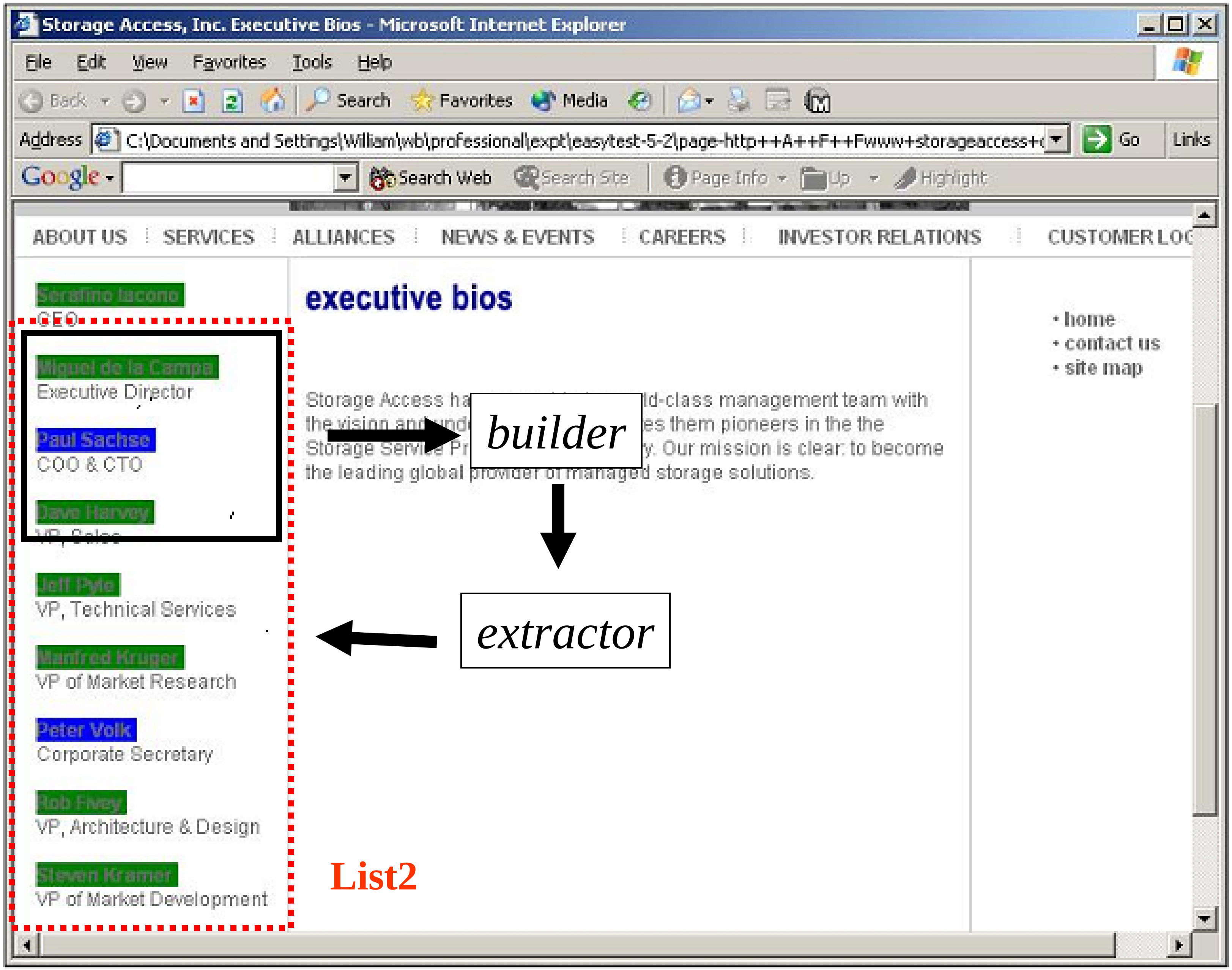
↖Generate many small training sets D_i from D, by sliding small windows over D.

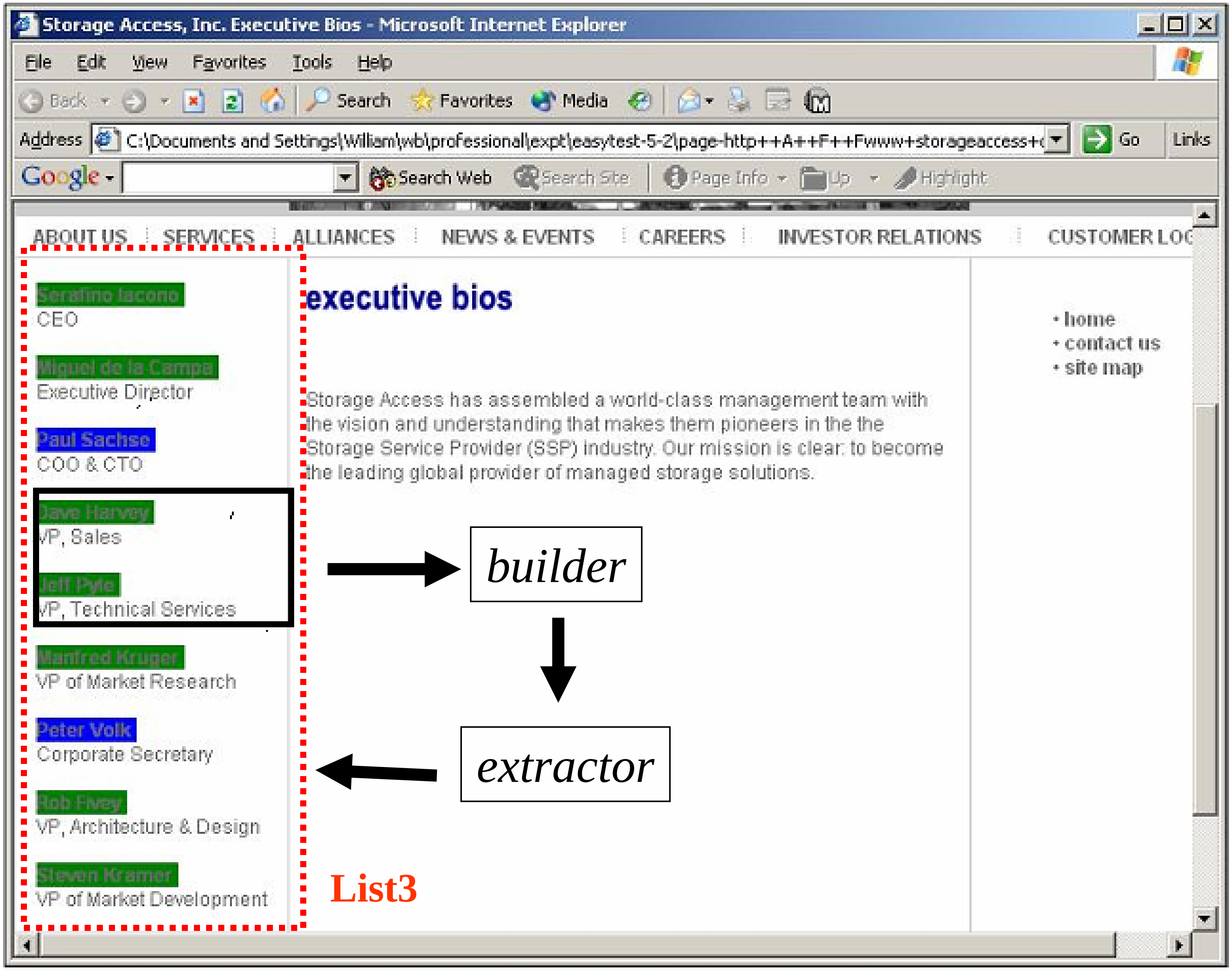
- Let P be the set of all "structures" found by any builder from any subset D_i

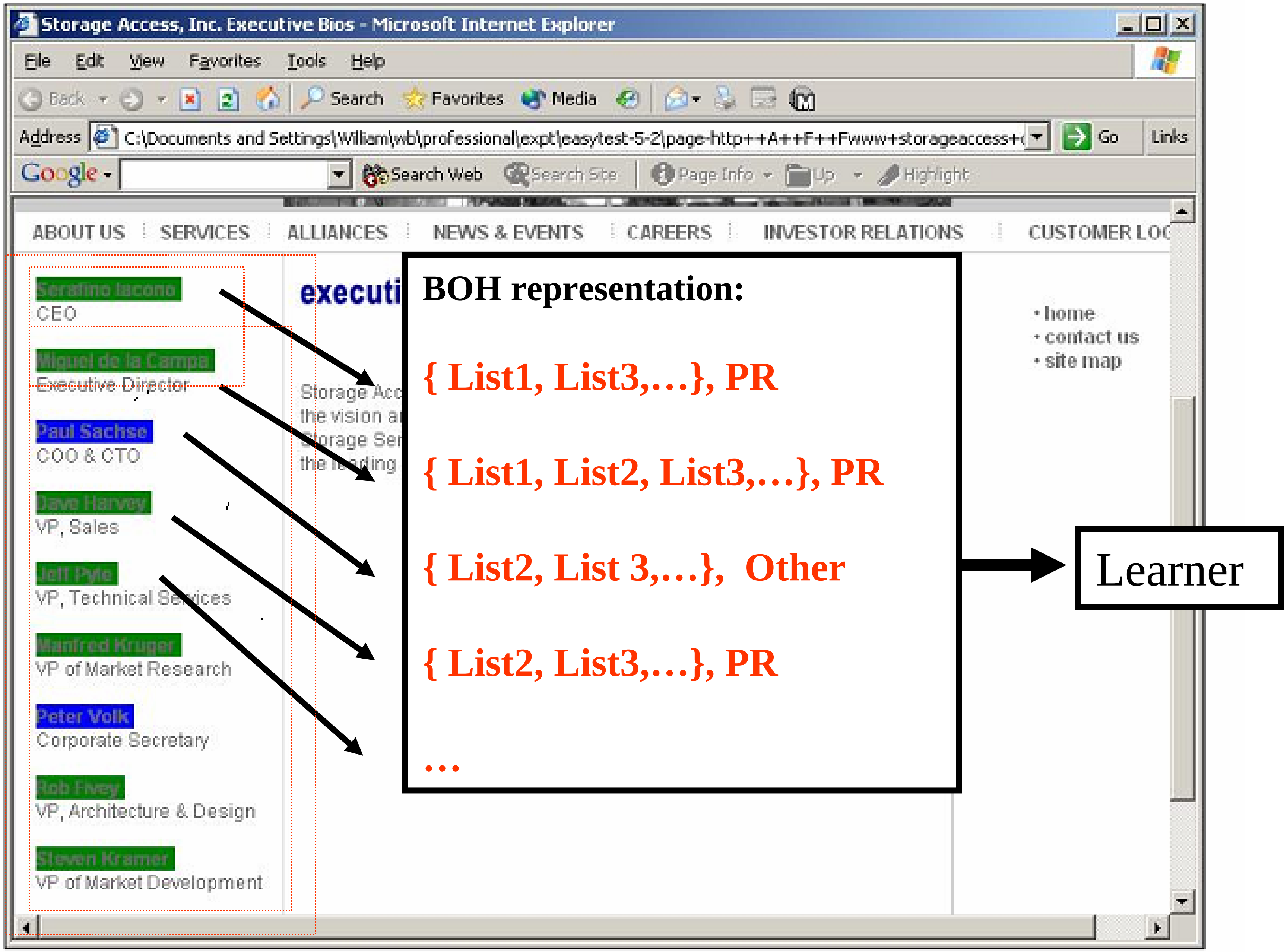
- Say that *p links to x* if *p* extracts an anchor that points to x. Represent a page x as the bag of **structures** in Pthat link to x.

Learning and Labeling. As before.


Storage Access, Inc. Executive Bios - Microsoft Internet Explorer
ABOUT US
SERVICES
ALLIANCES
NEWS & EVENTS
CAREERS
INVESTOR RELATIONS
executive bios
List1
builder
extractor
Storage Access ha ld-class management team with the vision and und es them pioneers in the the Storage Service Pr y. Our mission is clear: to become the leading global provider of managed storage solutions.
Serafino Iacono
CEO
Miguel de la Campa
Paul Sachse
COO & CTO
Dave Harvey
VP, Sales
Jeff Pyle
VP, Technical Services
Manfred Kruger
VP of Market Research
Peter Volk
Corporate Secretary
Rob Fivey
VP, Architecture & Design
Steven Kramer
VP of Market Development
home
contact us
site map


Storage Access, Inc. Executive Bios - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Documents and Settings\William\wb\professional\expt\easytest-5-2\page-http++A++F++Fwww+storageaccess+c
Go Links
Search Web Search Site Page Info Up Highlight
ABOUT US SERVICES ALLIANCES NEWS & EVENTS CAREERS INVESTOR RELATIONS CUSTOMER LOG
Serafino Iacono
CEO
Miguel de la Campa
Executive Director
Paul Sachse
COO & CTO
Dave Harvey
VP, Sales
Jeff Pyle
VP, Technical Services
Manfred Kruger
VP of Market Research
Peter Volk
Corporate Secretary
Rob Fivey
VP, Architecture & Design
Steven Kramer
VP of Market Development
executive bios
Storage Access ha
ld-class management team with
the vision and unde
es them pioneers in the the
Storage Service Pr
y. Our mission is clear: to become
the leading global provider of managed storage solutions.
• home
• contact us
• site map
builder
extractor
List2


Storage Access, Inc. Executive Bios - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\Documents and Settings\William\wb\professional\expt\easytest-5-2\page-http++A++F++Fwww+storageaccess+
Go Links
Google Search Web Search Site Page Info Up Highlight
ABOUT US SERVICES ALLIANCES NEWS & EVENTS CAREERS INVESTOR RELATIONS CUSTOMER LOG
Serafino Iacono
CEO
Miguel de la Campa
Executive Director
Paul Sachse
COO & CTO
Dave Harvey
VP, Sales
Jeff Pyle
VP, Technical Services
Manfred Kruger
VP of Market Research
Peter Volk
Corporate Secretary
Rob Fivey
VP, Architecture & Design
Steven Kramer
VP of Market Development
executive bios
Storage Access has assembled a world-class management team with the vision and understanding that makes them pioneers in the the Storage Service Provider (SSP) industry. Our mission is clear: to become the leading global provider of managed storage solutions.
• home
• contact us
• site map
builder
extractor
List3


Storage Access, Inc. Executive Bios - Microsoft Internet Explorer
ABOUT US
SERVICES
ALLIANCES
NEWS & EVENTS
CAREERS
INVESTOR RELATIONS
Serafino Iacono
CEO
Miguel de la Campa
Executive Director
Paul Sachse
COO & CTO
Dave Harvey
VP, Sales
Jeff Pyle
VP, Technical Services
Manfred Kruger
VP of Market Research
Peter Volk
Corporate Secretary
Rob Fivey
VP, Architecture & Design
Steven Kramer
VP of Market Development
• home
• contact us
• site map
BOH representation:
{ List1, List3,…}, PR
{ List1, List2, List3,…}, PR
{ List2, List 3,…}, Other
{ List2, List3,…}, PR
…
Learner

Experimental results

Experimental results

Site	Classifier f_1		1-CT (dtree)		1-CT (Winnow)	
	Acc	(SE)	Acc	(SE)	Acc	(SE)
1	1.000	(0.000)	0.960	(0.028)	0.960	(0.028)
2	0.932	(0.027)	0.955	(0.022)	0.955	(0.022)
3	0.813	(0.028)	0.934	(0.018)	0.939	(0.017)
4	0.904	(0.029)	0.962	(0.019)	0.962	(0.019)
5	0.939	(0.024)	0.960	(0.020)	0.960	(0.020)
6	1.000	(0.000)	1.000	(0.000)	1.000	(0.000)
7	0.918	(0.028)	0.990	(0.010)	0.990	(0.010)
8	0.788	(0.044)	0.882	(0.035)	0.929	(0.028)
9	0.948	(0.029)	0.948	(0.029)	0.983	(0.017)

Summary

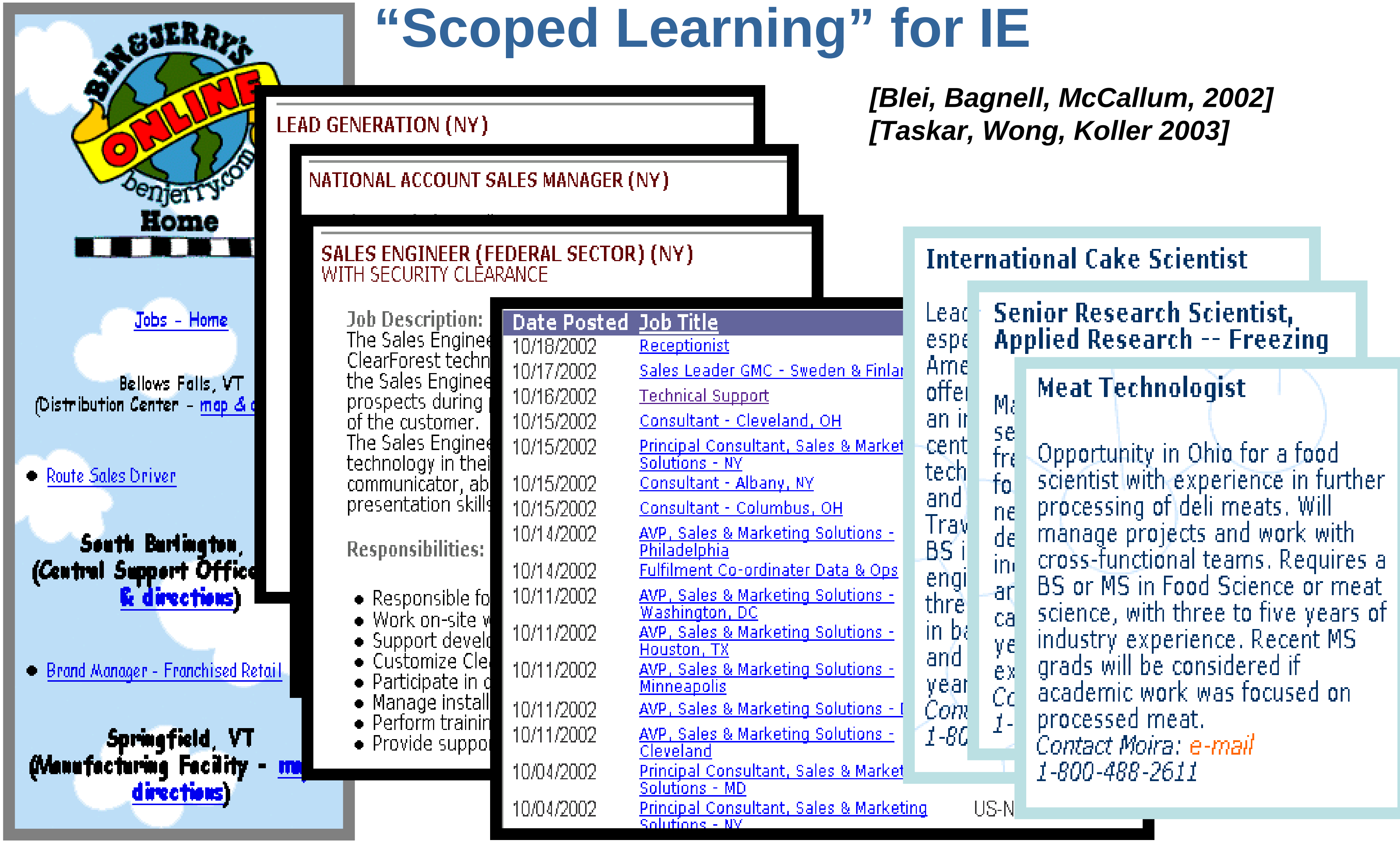
- "Builders" (from a wrapper learning system) let one discover and use structure of web sites *and index pages* to smooth page classification results.
- Discovering good "hub structures" makes it possible to use 1-step co-training on small (50-200 example) unlabeled datasets.
  – Average error rate was reduced from 8.4% to 3.6%.
  – Difference is statistically significant with a 2-tailed paired sign test or t-test.
  – EM with probabilistic learners also works—see (Blei et al, UAI 2002)

Carnegie Mellon
SCHOOL OF COMPUTER SCIENCE

CALD

Learning Formatting Patterns "On the Fly": "Scoped Learning" for IE

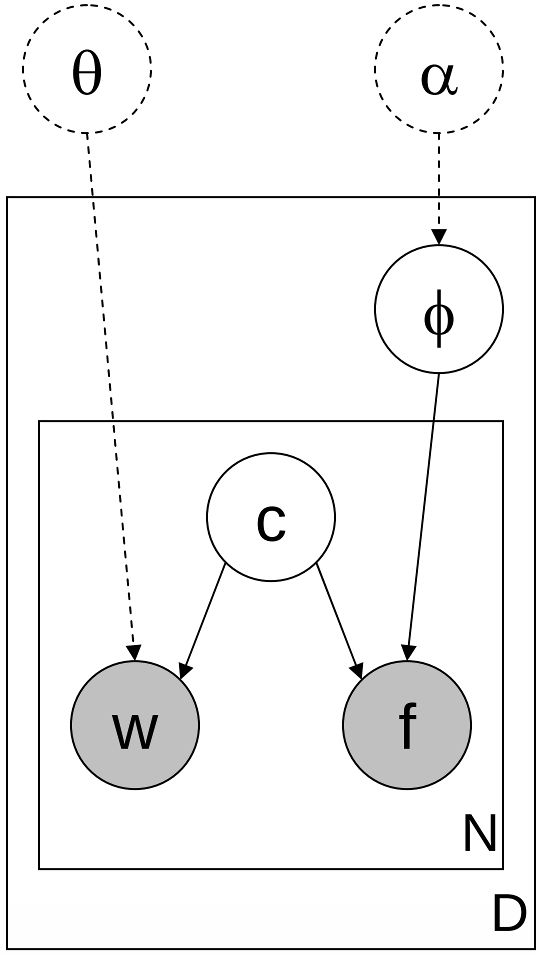
[Blei, Bagnell, McCallum, 2002]
[Taskar, Wong, Koller 2003]



Formatting is regular on each site, but there are too many different sites to wrap.
Can we get the best of both worlds?

Scoped Learning Generative Model

1. For each of the D documents:
   a) Generate the multinomial formatting feature parameters ϕ from $p(\phi|\alpha)$
2. For each of the N words in the document:
   a) Generate the *n*th category c_n from $p(c_n)$.
   b) Generate the *n*th word (global feature) from $p(w_n|c_n,\theta)$
   c) Generate the *n*th formatting feature (local feature) from $p(f_n|c_n,\phi)$



$$p(\phi, \mathbf{c}, \mathbf{w}, \mathbf{f}) = p_\alpha(\phi) \prod_{n=1}^{N} p(c_n) p_\theta(w_n|c_n) p(f_n|c_n, \phi)$$

Inference

Given a new web page, we would like to classify each word resulting in $\mathbf{c} = \{c_1, c_2, \ldots, c_n\}$

$$p(\mathbf{c}|\mathbf{w}, \mathbf{f}) = \frac{\int \prod_{n=1}^{N} p(w_n|c_n)p(f_n|c_n, \phi)p(c_n)p(\phi)d\phi}{\int \prod_{n=1}^{N} \sum_{c_n} p(w_n|c_n)p(f_n|c_n, \phi)p(c_n)p(\phi)d\phi}$$

This is not feasible to compute because of the integral and sum in the denominator. We experimented with two approximations:

- MAP point estimate of ϕ
- Variational inference

MAP Point Estimate

If we approximate ϕ with a point estimate, $\hat{\phi}$, then the integral disappears and c decouples. We can then label each word with:

$$\hat{c}_n = \arg\max_{c_n} p(w_n|c_n)p(f_n|c_n,\hat{\phi})p(c_n)$$

A natural point estimate is the posterior mode: a maximum likelihood estimate for the local parameters given the document in question:

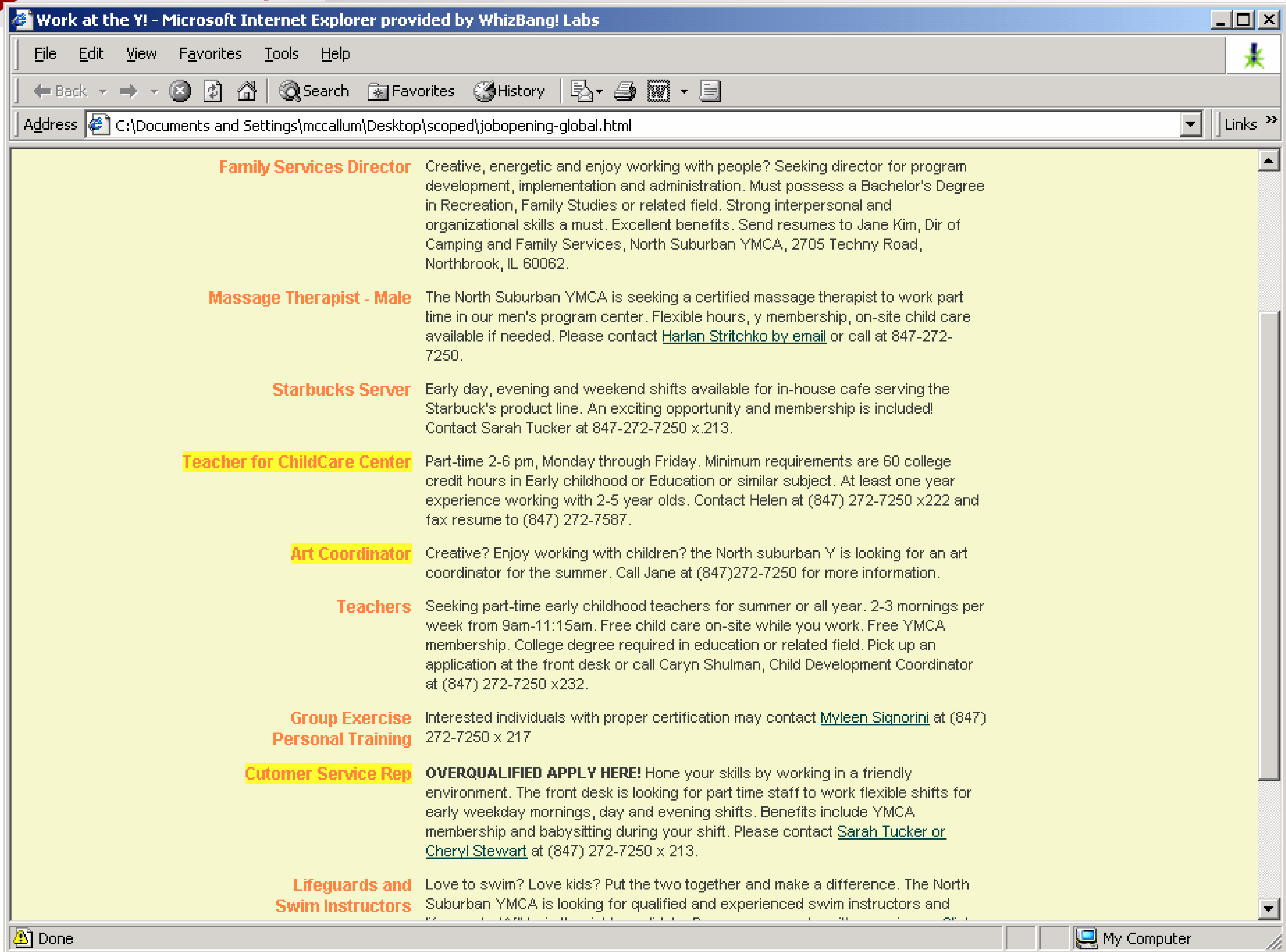
$$\hat{\phi} = \arg\max_{\phi} p(\phi|\mathbf{f},\mathbf{w})$$

E-step:

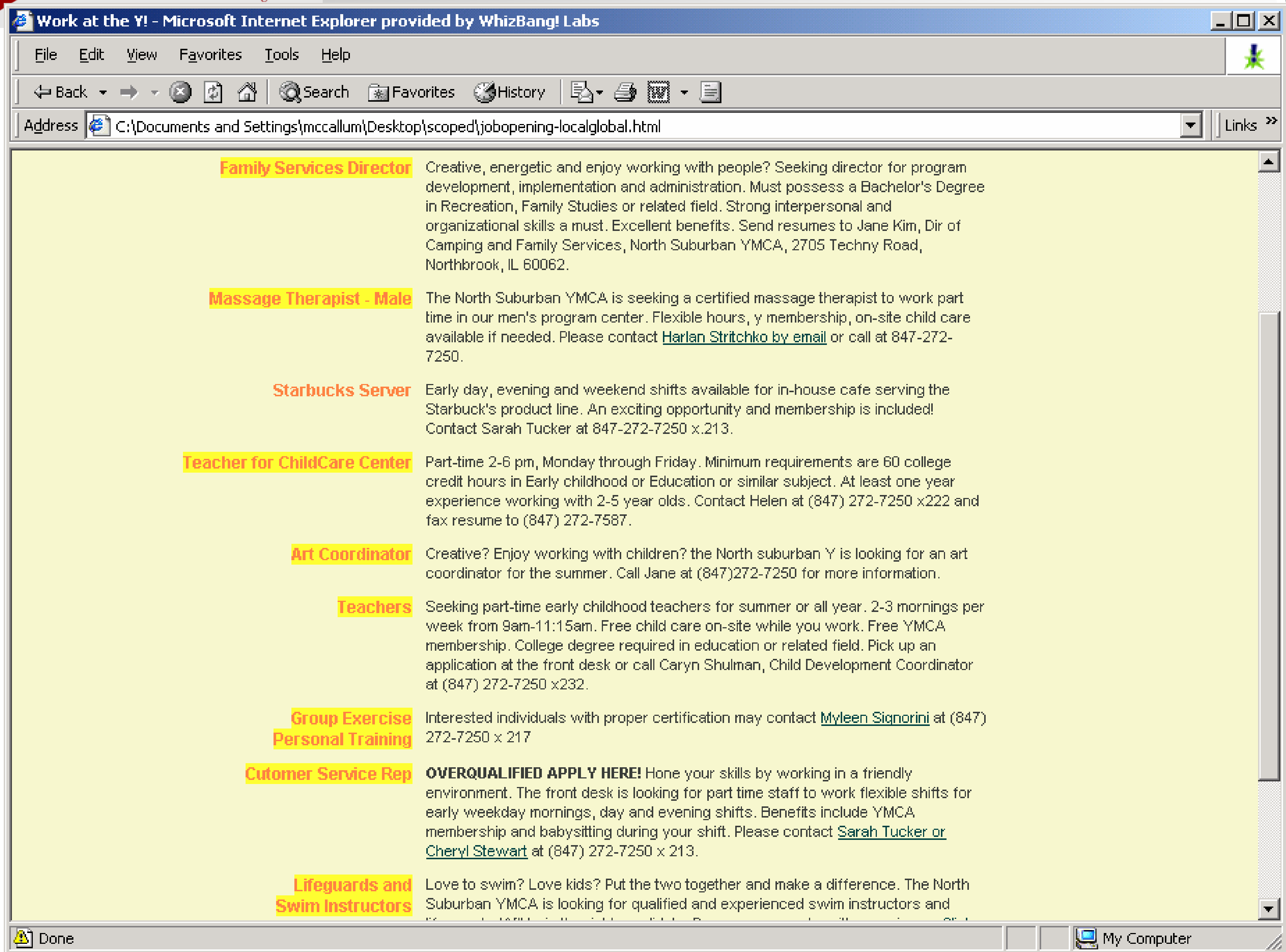
$$p^{(t+1)}(c_n|w_n,f_n;\phi) \propto p^{(t)}(f_n|c_n;\phi)p(w_n|c_n)p(c_n)$$

M-step:

$$\hat{\phi}_{c,f} = p^{(t+1)}(f|c;\phi) \propto \sum_{\{n:c_n=c,f_n=f\}} p^{(t)}(c_n|f_n,w_n)$$



Global Extractor: Precision = 46%, Recall = 75%


Work at the Y! - Microsoft Internet Explorer provided by WhizBang! Labs

File Edit View Favorites Tools Help

Back Search Favorites History

Address C:\Documents and Settings\mccallum\Desktop\scoped\jobopening-localglobal.html Links

Family Services Director	Creative, energetic and enjoy working with people? Seeking director for program development, implementation and administration. Must possess a Bachelor's Degree in Recreation, Family Studies or related field. Strong interpersonal and organizational skills a must. Excellent benefits. Send resumes to Jane Kim, Dir of Camping and Family Services, North Suburban YMCA, 2705 Techny Road, Northbrook, IL 60062.
Massage Therapist - Male	The North Suburban YMCA is seeking a certified massage therapist to work part time in our men's program center. Flexible hours, y membership, on-site child care available if needed. Please contact Harlan Stritchko by email or call at 847-272-7250.
Starbucks Server	Early day, evening and weekend shifts available for in-house cafe serving the Starbuck's product line. An exciting opportunity and membership is included! Contact Sarah Tucker at 847-272-7250 x.213.
Teacher for ChildCare Center	Part-time 2-6 pm, Monday through Friday. Minimum requirements are 60 college credit hours in Early childhood or Education or similar subject. At least one year experience working with 2-5 year olds. Contact Helen at (847) 272-7250 x222 and fax resume to (847) 272-7587.
Art Coordinator	Creative? Enjoy working with children? the North suburban Y is looking for an art coordinator for the summer. Call Jane at (847)272-7250 for more information.
Teachers	Seeking part-time early childhood teachers for summer or all year. 2-3 mornings per week from 9am-11:15am. Free child care on-site while you work. Free YMCA membership. College degree required in education or related field. Pick up an application at the front desk or call Caryn Shulman, Child Development Coordinator at (847) 272-7250 x232.
Group Exercise Personal Training	Interested individuals with proper certification may contact Myleen Signorini at (847) 272-7250 x 217
Cutomer Service Rep	**OVERQUALIFIED APPLY HERE!** Hone your skills by working in a friendly environment. The front desk is looking for part time staff to work flexible shifts for early weekday mornings, day and evening shifts. Benefits include YMCA membership and babysitting during your shift. Please contact Sarah Tucker or Cheryl Stewart at (847) 272-7250 x 213.
Lifeguards and Swim Instructors	Love to swim? Love kids? Put the two together and make a difference. The North Suburban YMCA is looking for qualified and experienced swim instructors and

Done My Computer

Scoped Learning Extractor: Precision = 58%, Recall = 75% ***Δ Error = -22%***

Outline

- Motivation: finding even simple structures like lists is useful, and *seems* like it should be easy.
- Cohen & Fan, 1999a: List-finding as *classification*.
- Cohen 1999b,2000: List-finding as *matching* structure to content.
- Cohen et al 2001, Cohen 2002, Bagnell et al 2002: List-finding as *learning* global content and local structure.