IE Systems

Fang Li

What is IE system?

IE systems extract domain-specific information from natural language text. The domain and types of information to be extracted must be defined in advance. IE systems often focus on object identification, such as references to people, places, companies, and physical objects. [...] Domain-specific extraction patterns (or something similar) are used to identify relevant information.

(Riloff and Lorenzen, 1999, p. 169)

Some limitations:

- Information defined in advance
- Domain –specific

Types of IE systems introduced

Aim: Extract information from the Internet.

- Wrapper Systems
- NLP based extraction system
- Open-domain extraction system

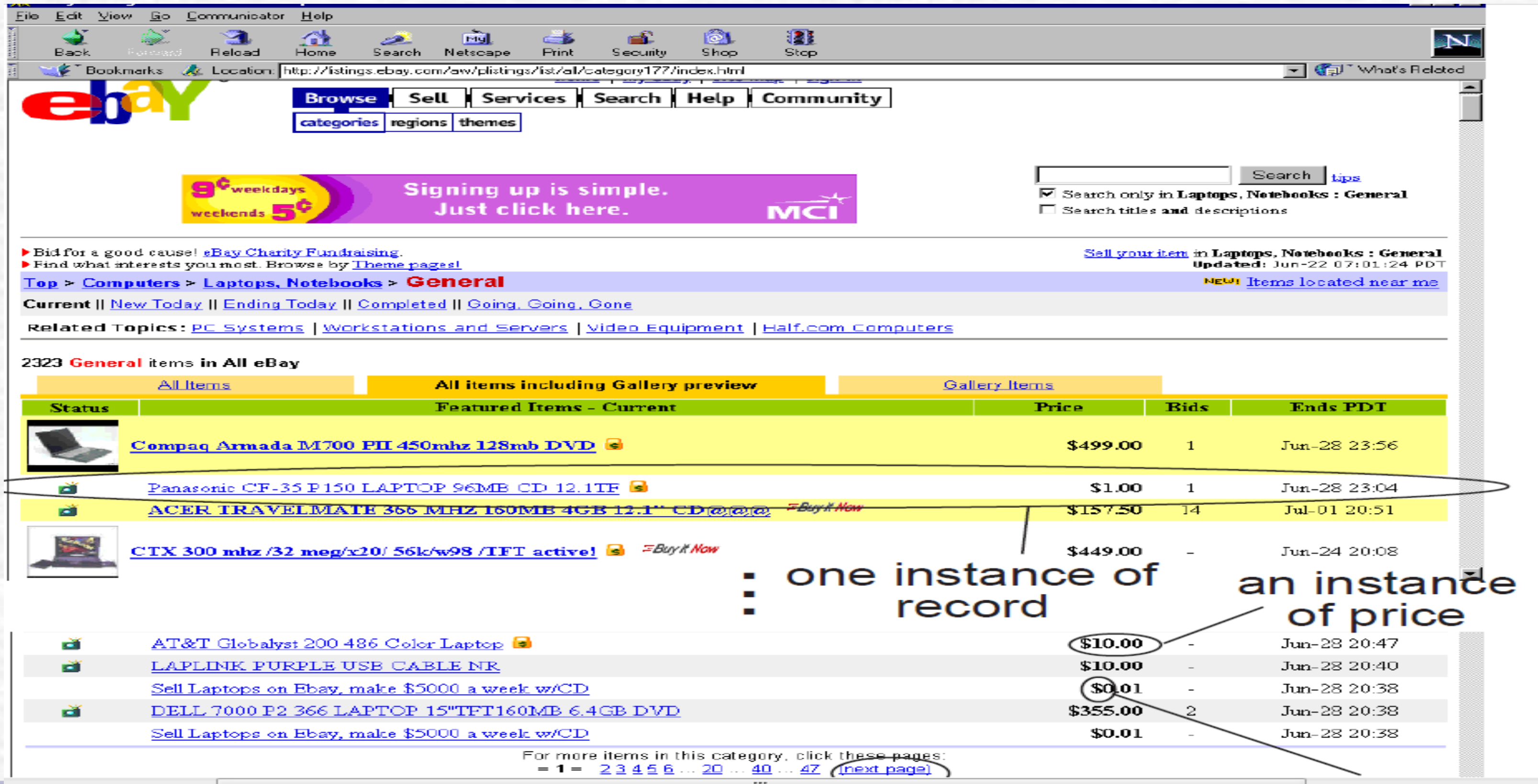
Contents

- Wrapper-based IE systems:

  - lixTo system (semi-automatically)
  - Roadrunner system (automatically)
  - Never-Ending Learning (NELL)

Discussion

- Given a page with many data in it
- How to extract it automatically?



Lixto System

- A system and method for the visual and interactive generation of wrappers for web pages under the supervision of a human developer, for automatically extracting information from Web pages using such wrappers, and for translating the extracted content into XML
- www.lixto.com (the company founded in 2001 as a spin-0ff of the Vienna Technical University.)

Tools & Middleware

Lixto provides enterprise-class development tools and middleware to rapidly develop maintainable and robust data extraction programmes and to effectively use these applications to gather and process data from the web on a large scale. Technology from Lixto is in particular well-suited to access, augment and deliver content and data from highly dynamic web applications which take advantage of client-side processing technologies such as JavaScript, AJAX and dynamic HTML in general.

Tools

Visual development environment

Lixto provides an integrated development environment (IDE) for web data extraction programmes. This framework forms the basis for the Lixto Visual Developer and the Lixto Web Application Testing Suite.

Middleware

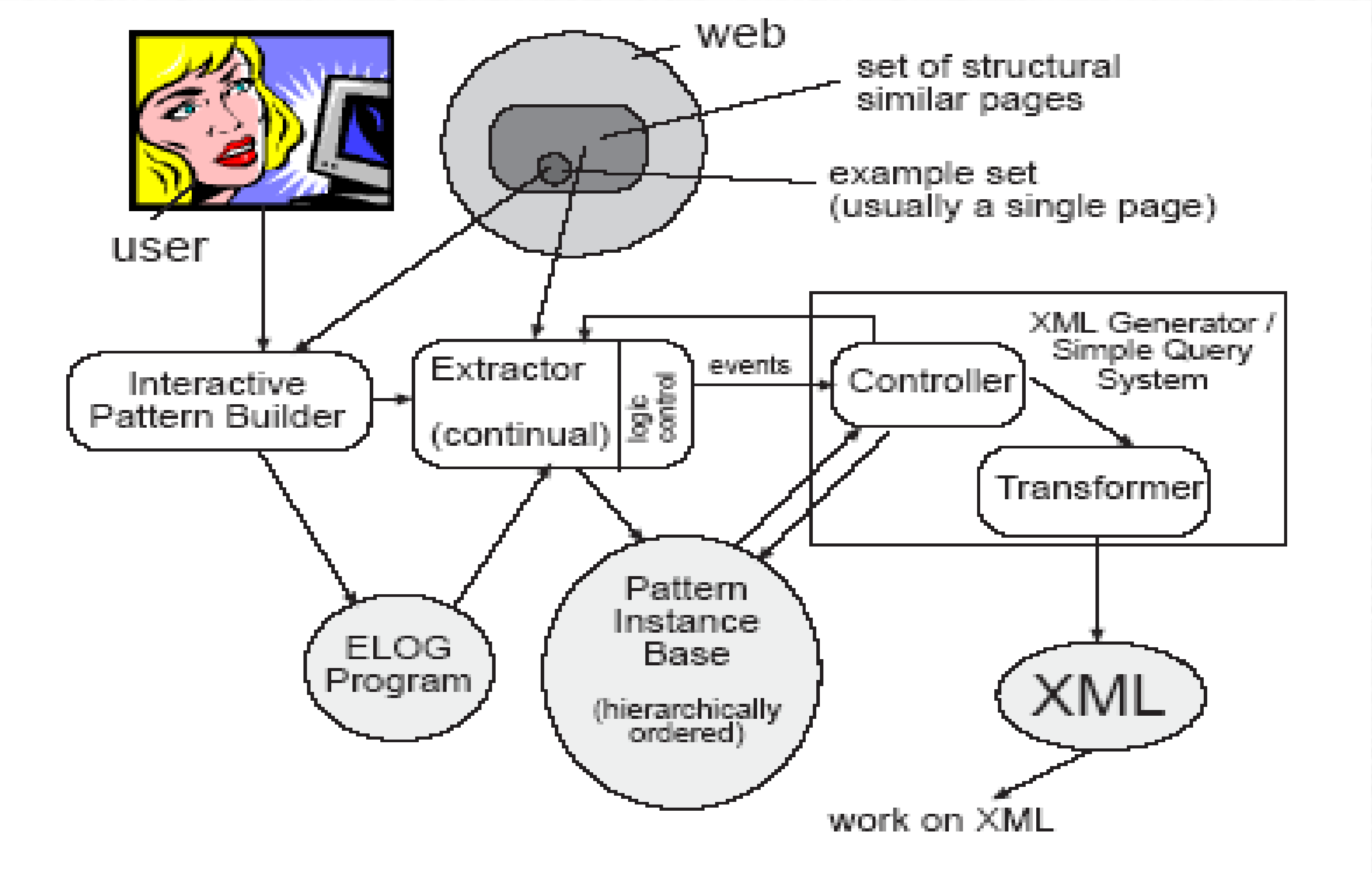
Scalable web data extraction processes

Lixto Middleware enables enterprises to run extremely reliable web data extraction processes. Lixto Middleware is a highly scalable web data extraction infrastructure and supports cloud computing for instance deployment.

Features of Lixto

- Very high expressive power:
- of defining sophisticated extraction patterns
- Excellent visual support
- for marking extraction patterns
- Good learnability
- No extraction language needs to be learned
- Sample parsimony
- Very few sample pages are needed in order to define robust wrappers
- Simple and smooth XML translation mechanism

Architecture of Lixto System



Architecture of Lixto System (cont.)

- Interactive pattern builder: provides the visual UI that allows a user to specify the desired extraction patterns and the basic algorithm for creating a corresponding *Elog* wrapper as output.
- Extractor: *Elog* program interpreter that performs the actual extraction based on a given *Elog* program.
- The controller of XML Generator: the user chooses how to map extracted information to XML.

About extraction language: *Elog*

- *Elog*: system-internal datalog-like rule based language specially designed for hierarchical and modular data extraction.

Datalog Rule:

Happy(d) <- Frequents(d,bar) AND Likes(d,beer) AND Sells(bar,beer,p)

Rule body

Head predicate: Happy(d)

If and only if all atoms of the body are true, the head is true

Extraction language: *Elog*

The head of a rule r is of the form p(S,X):

- p is a pattern name,
- S is a variable which **is bound in the body of the rule** to the parent-pattern instances of the filter corresponding to r,
- X is **the target variable** which, at extraction time, is bound to some target pattern instance (a tree region or string) to be extracted.

Extraction language: *Elog (cont.)*

A standard extraction rule:

New(S,X) ← Par(_,S), Ex(S,X),Co(S,X,…)[a,b]

Par(_,S): parent pattern predicate

Ex(S,X): extraction definition predicate

Co(S,X,..): further imposed conditions

[a,b] are optional, range parameter.

Rule example

record(S, X) ← tableseq(_, S), subelem(S, :*table*, X)

If S is an instance in tableseq, and X is a tree region contained in S and the root of X matches *table* then X is a table contained in S.

The first atom: the parent pattern is an instance of <tableseq>.
The second atom: looks for subelements that qualify as tables inside the unique tableseq instance and instantiates X with them.

Extraction language: *Elog* *(body of the rule)*

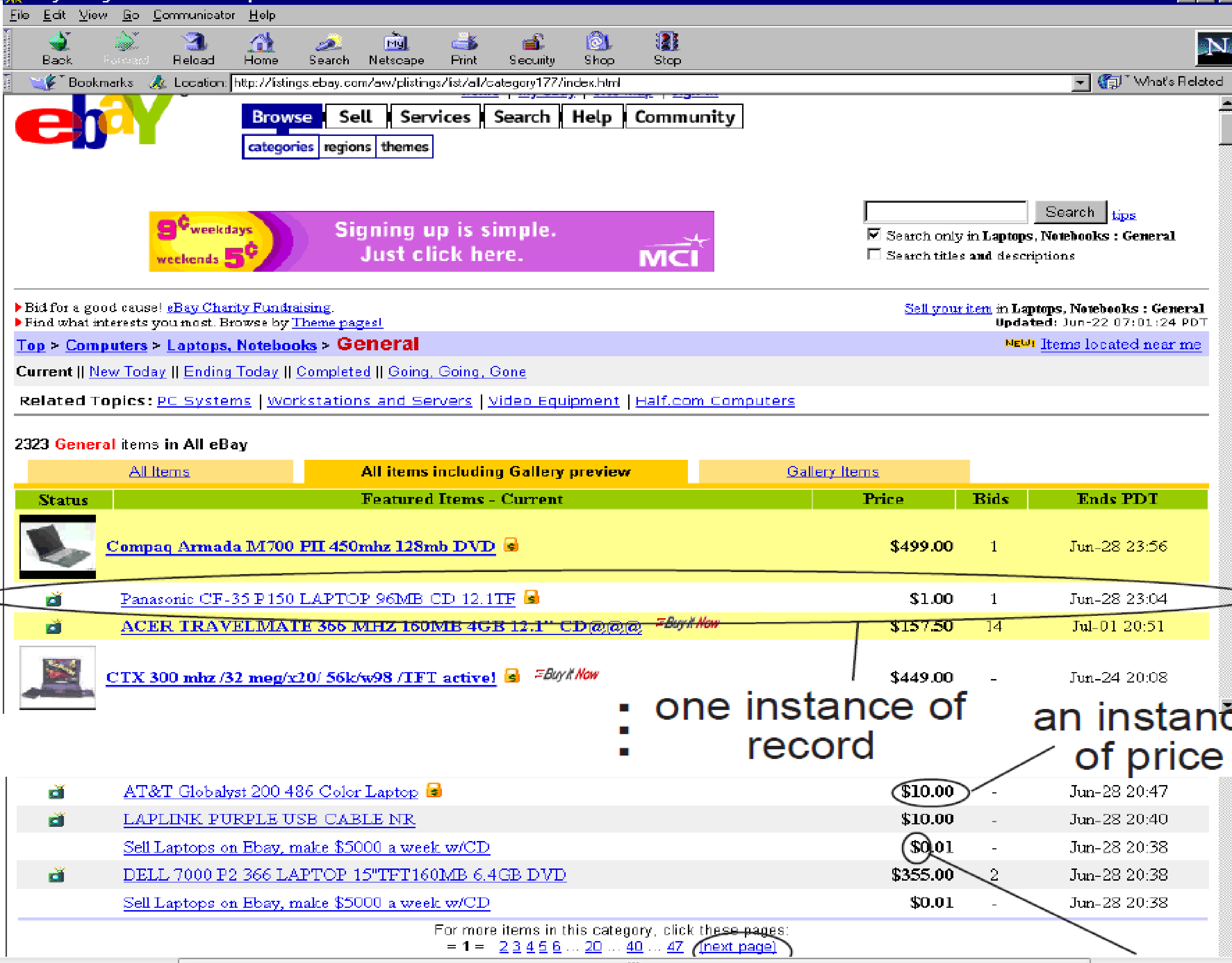
- Attribute conditions: impose restrictions on matched elements. E.g the value is *italics*
- Element characterizations: the value is a concept like "isCity".
- Tree Extraction Definition Predicates: a variable should be instantiated with a node in the HTML tree which matches an element path definition.

Extraction language: *Elog* *(body of the rule)*

- String extraction definition predicates: every node n of the parse tree by concatenating all strings corresponding to leaves of the subtree rooted in n.
- Contextual conditions: some other elements must or must not appear either before or after some instances.
- Internal conditions: some characteristic feature must or must not appear with an instance.

Extraction language: *Elog* *(body of the rule)*

- <u>Concept conditions</u>: predicates like isEmail(X), isCurrency(X)
- <u>Comparison conditions</u>: compare two dates,
- <u>Pattern References</u>: parent pattern defines the context of a rule.
- <u>Range conditions</u>: any rule a range condition such as "[3,7]" can be added.



⋮

one instance of record

an instance of price

Elog Extraction Program for a single eBay page

$$
\begin{array}{rcl}
\mathtt{tablesq}(S,X) & \leftarrow & \mathtt{document}(\text{“www.ebay.com/”}, S), \mathtt{subsq}(S, (.\mathtt{body}, []), (.\mathtt{table}, []), (.\mathtt{table}, []), X), \\
 & & \mathtt{before}(S, X, (.\mathtt{table}, [(\mathtt{elementtext}, \mathtt{item}, \mathtt{substr})]), 0, 0, \_, \_), \mathtt{after}(S, X, .\mathtt{hr}, 0, 0, \_, \_) \\
\mathtt{record}(S,X) & \leftarrow & \mathtt{tableseq}(\_, S), \mathtt{subelem}(S, .\mathtt{table}, X) \\
\mathtt{itemnum}(S,X) & \leftarrow & \mathtt{record}(\_, S), \mathtt{subelem}(S, \star.\mathtt{td}, X), \mathtt{notbefore}(S, X, .\mathtt{td}, 100) \\
\mathtt{itemdes}(S,X) & \leftarrow & \mathtt{record}(\_, S), \mathtt{subelem}(S, (\star.\mathtt{td}.\star.\mathtt{content}, [(\mathtt{a}, , \mathtt{substr})], X) \\
\mathtt{price}(S,X) & \leftarrow & \mathtt{record}(\_, S), \mathtt{subelem}(S, (\star.\mathtt{td}, [(\mathtt{elementtext}, \backslash\mathtt{var}[Y].*, \mathtt{regvar})]), X), \mathtt{isCurrency}(Y) \\
\mathtt{bids}(S,X) & \leftarrow & \mathtt{record}(\_, S), \mathtt{subelem}(S, \star.\mathtt{td}, X), \mathtt{before}(S, X, .\mathtt{td}, 0, 30, Y, \_), \mathtt{price}(\_, Y) \\
\mathtt{currency}(S,X) & \leftarrow & \mathtt{price}(\_, S), \mathtt{subtext}(S, \backslash\mathtt{var}[Y], X), \mathtt{isCurrency}(Y) \\
\mathtt{pricewc}(S,X) & \leftarrow & \mathtt{price}(\_, S), \mathtt{subtext}(S, [0-9]^{+}\backslash.[0-9]^{+}, X)
\end{array}
$$

Element characteriza

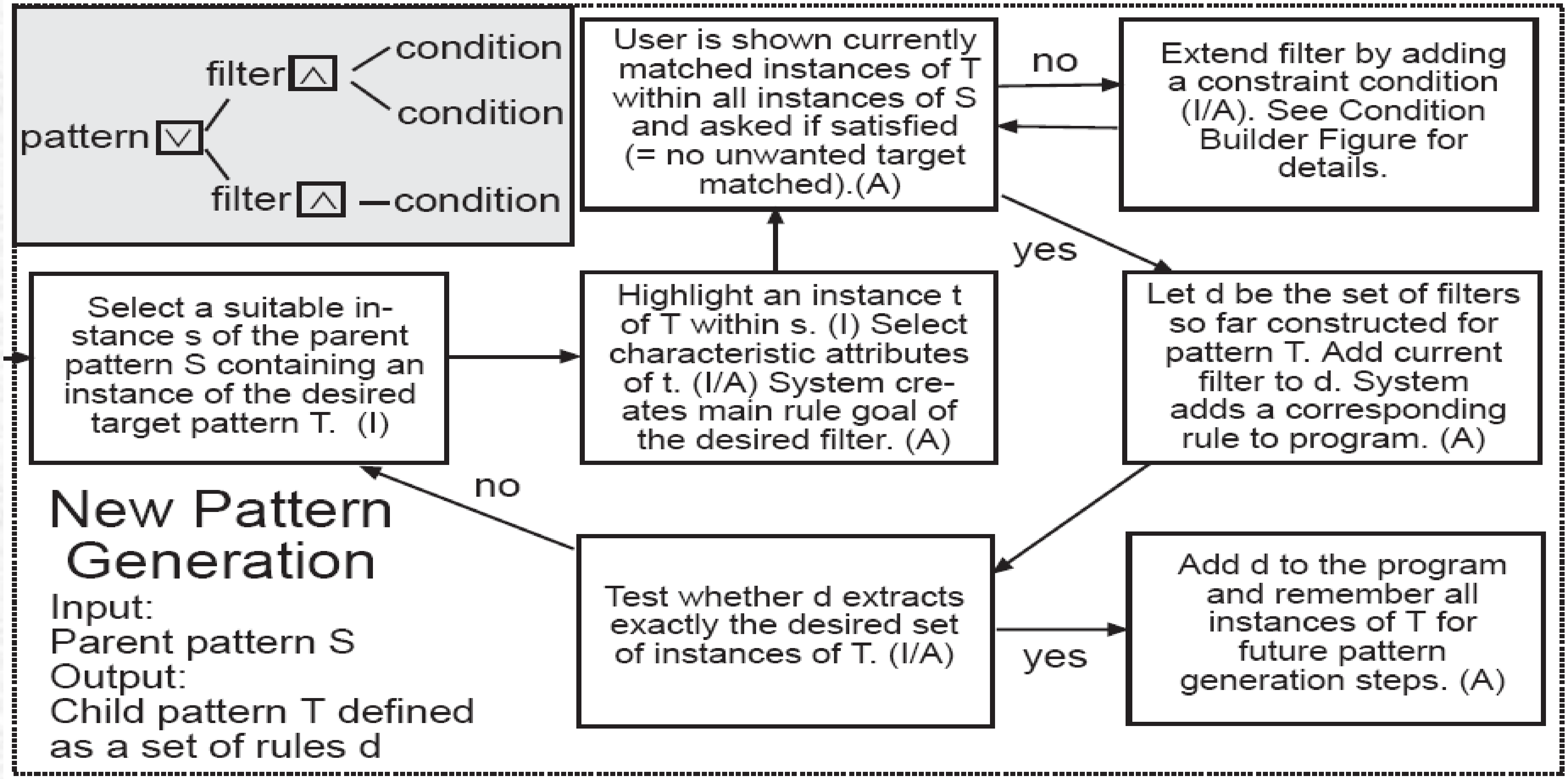
Context condition

String extraction
Definition predicates

How to build the extraction rules?

- Pattern： A set of rules defining the same head.
- Rule： A rule defines many extraction conditions, such as attribute condition, element characterization,…
- Filter： like a rule.

How to Build Wrapper



- I: interactive A: automatic
- **Interactively generate a new pattern**

Recursive Wrapping

"$1" is interpreted as a constant whose value is the URL of the start document.

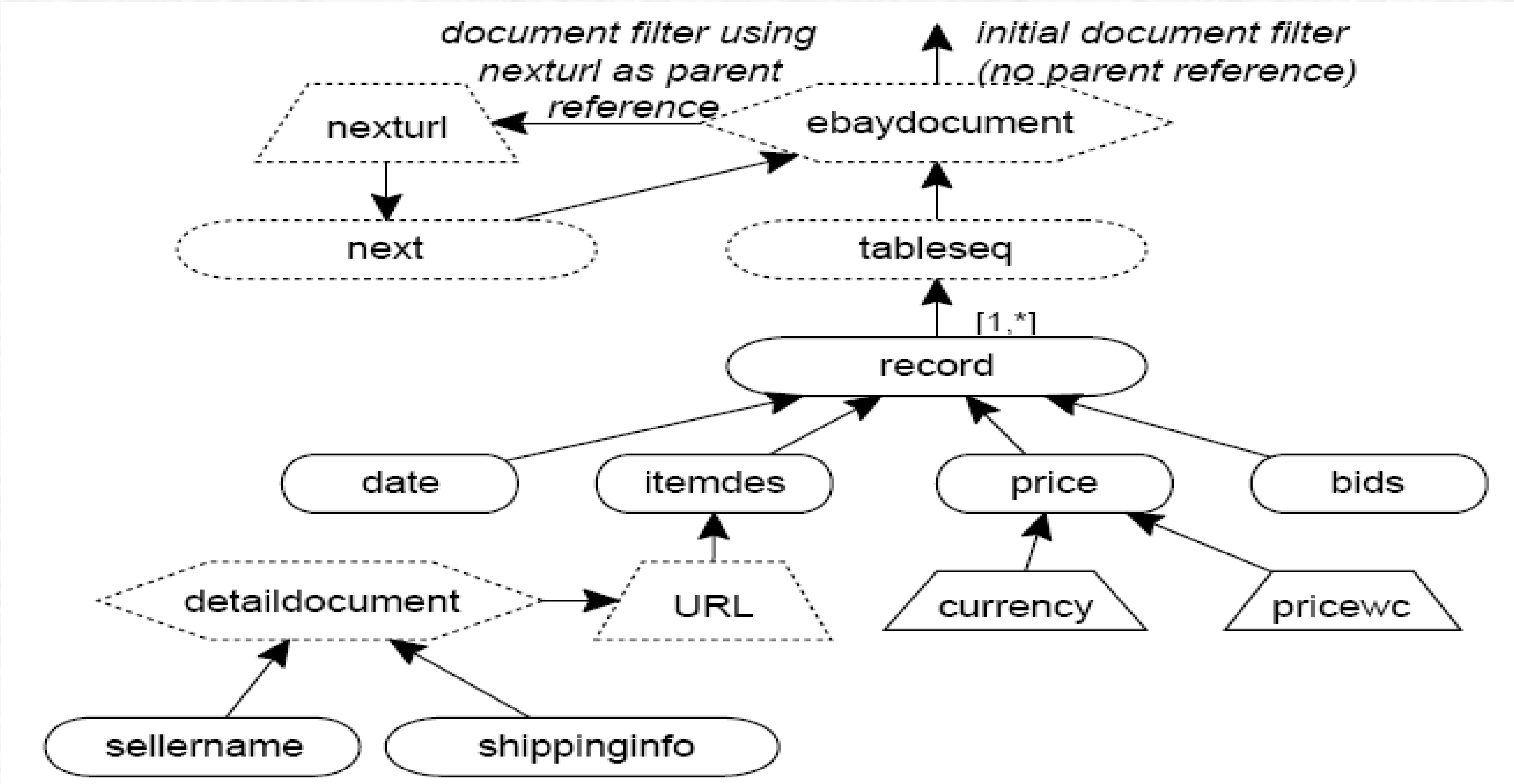
$$\texttt{document}(S,X) \leftarrow \texttt{getDocument}(\$1,X)$$

$$\texttt{table}(S,X) \leftarrow \texttt{document}(\_,S),\texttt{subelem}(S,.\star.\texttt{table},X)$$

$$\boxed{\texttt{table}(S,X) \leftarrow \texttt{table}(\_,S),\texttt{subelem}(S,.\star.\texttt{table},X)}$$

It extracts all nested tables within one page, starting with the outermost, and stores them in this hierarchical order in the pattern instance base. The second rule of <table> is iteratively called, until no further table can be extracted.

Recursive extract pages which are connected to each other via a "next page" link.



$$\mathtt{ebaydocument}(S,X) \leftarrow \boxed{\mathtt{ebaydocument}(\_,S),}\ \mathtt{subatt}(Y,\mathtt{href},Z), \mathtt{getDocument}(Z,X)$$
$$\mathtt{subelem}(S,(\star.\mathtt{content},[(\mathtt{href},,\mathtt{substr}),$$
$$(\mathtt{elementtext},(\mathtt{next\ page}),\mathtt{exact})]),Y),$$

Results reported from Lixto

Name	*Website*	*Used Example Page*	*Testpages*
Amazon	http://www.amazon.com/	Lord of the Rings	10
CIA Factbook	www.odci.gov/cia/publications/factbook/	United Kingdom	12
Cinemachine	www.cinemachine.com/	The World is not enough	15
DBLP	www.informatik.uni-trier.de/~ley/db/	Michael Ley	10
Election Res. / State	www.cnn.com/ELECTION/2000/results/		0
eBay	www.ebay.com/	que	0
Excite Weather	www.excite.com/weather/forecast		2
Jobs-Jobs-Jobs	www.jobsjobsjobs.com/		0
Perl Module List	www.cpan.org/modules/00modlist.long.html	si	pg.
Travelnotes	www.travelnotes.org/	qu	0
Yahoo People Email	people.yahoo.com/	q	5
Yahoo Weather	weather.yahoo.com/		5

Table 1: Some of the test-sites used for *Lixto*

how many example pages are necessary to get 100 percent of correctly matched pattern instances

Name	*wrapable?*	*Complexity*	*Correct*	*for 100%*	*Time/Pattern (mins)*	*Depth*
Amazon	yes	16/9 = 1.78	95%	3	22/9 = 2.44	4
CIA Factbook	yes	17/5 = 3.4	80%	3	18/5 = 3.6	3
Cinemachine	yes	6/4 = 1.5	100%	1	16/4 = 4	2
DBLP	yes	27/9 = 3	90%	2	54/9 = 6	8
Election Results / State	yes	4/2 = 2	100%	1	6/2 =	
eBay	yes	19/8 = 2.38	99.9%	2	21/8 =	
Excite Weather	yes	22/7 = 3.14	100%	1	30/7 =	
Jobs Jobs Jobs	yes	21/12 = 1.75	90%	3	40/12 =	
Perl Module List	yes	22/5 = 4.4	(100 %)	(1)	60/5 =	
Travelnotes	yes	11/4 = 2.75	95%	2	20/4 =	
Yahoo People Email	yes	10/3 = 3.3	100%	1	24/3 =	
Yahoo Weather	yes	22/10 = 2.2	100%	1	12/10 =	

Table 2: Evaluation of wrapper generation

the time needed for constructing the initial wrapper based on one example page

Question

- How does the company profit from the data extraction program?
- http://www.lixto.com

RoadRunner System

- Aim:
  - Extract data-intensive web sites.
  - Data is stored in a back-end DBMS, HTML pages are dynamically generated using scripts
- Methods:
  - Unsupervised wrapper generation
  - Do not assume that sample pages are manually selected→ the system is able to automatically **cluster pages** in a site into homogeneous classes
  - Does not rely on user-specified labeled examples → wrappers are generated and data are extracted in a completely automatic way.
  - Do not assume any a priori knowledge about the target schema→ deal with flat records and also nested structures.

Overview of RoadRunner System

- Given a set of HTML pages, find a schema for the content of these pages.
- A set of extraction rules parse the HTML code and retrieve the data according to the discovered schema.
- Pattern discovery can be based on the study of similarities and dissimilarities between the pages

A running example

a. Source Dataset

Name	Books				
	Title	Descr.	Editions		
			Details	Year	Price
John Smith	Database Primer	This book...	First Edition, Paperback	1998	20$
			Second Edition, Hard Cover	2000	30$
	Computer Systems	An undergraduate...	First Edition, Paperback	1995	40$
Paul Jones	XML at Work	A comprehensive...	First Edition, Paperback	1999	30$
	HTML and Scripts	A useful HTML...	null	1998	30$
			Second Edition, Hard Cover	1999	45$
	JavaScript	A must in...	null	2000	50$
...	...	...	...	...	...

b. HTML Pages

www.csbooks.com/author?John+Smith　　www.csbooks.com/author?Paul+Jones

- Fig. a: a nested dataset by querying a database.
- Fig. b: each author's book information with the same style.

Method: compares the HTML codes of the two pages, infers a common structure and a wrapper, and use that to extract the source dataset.

Result of the extraction in the example

c. Data Extraction Output

Total number of SCHEMAs found: 1

Schema Number 1: A (B ((C)? D E)+ F)+ **Total Time: 0" 180 ms**

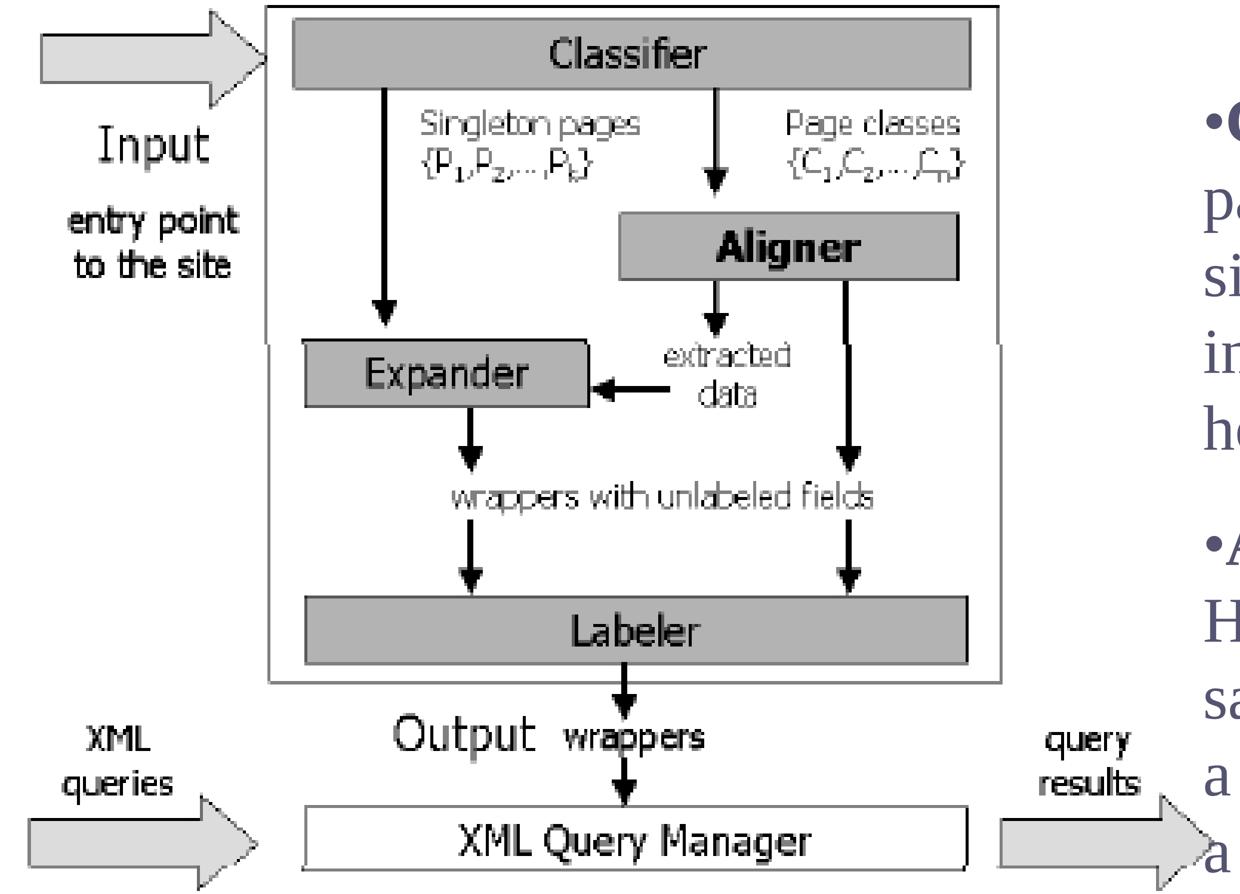
sample1.html

A	B	C	D	E	F
John Smith	Database Primer	First Edition, Paperback	1998	$20	This book introduces the reader to the theory and technology... *[TRUNCATED]*
		Second Edition, Hard Cover	2000	$30	
	Computer Systems	First Edition, Paperback	1995	$40	An undergraduate level introduction to computer... *[TRUNCATED]*

sample2.html

A	B	C	D	E	F
Paul Jones	XML at Work	First Edition, Paperback	1999	$30	A comprehensive description of XML and all related standards ...*[TRUNCATED]*
	HTML and Scripts	null	1993	$30	A useful HTML handbook, with a good tutorial on the use of sc... *[TRUNCATED]*
		Second Edition, Hard Cover	1999	$45	
	JavaScripts	null	2000	$50	A must in every Webmaster's bookshelf ...

The Architecture of the System



- **Classifier:** analyzes pages from the target site and collect them into clusters with a homogeneous structure.
- **Aligner**: compares the HTML sources of some samples pages to infer a a grammar to be used as a wrapper.

How to identify different pages classes in the target sites?

(**Classifier: mapping a sample to the feature space)**

- Tag Probability: it is reasonable to assume that pages complying the same grammar have a similar "distribution" of tags, i.e., tags appear in the pages with similar probability
- Tag Periodicity: there are cases in which tag probabilities may be misleading, since they do not give information about the relative positions of tags. Tag frequency is used to complement tag probability.
- Distance from the Home Page :if navigation paths in the site are well organized, it is reasonable to assume that pages containing homogeneous information are approximately at the same distance from the home page in the site graph.
- URL Similarity

Architecture of the System (cont.)

- **Expander**: infer a wrapper for those singleton pages. Most singleton pages are indices or links to other pages.
- **Labeler**: associates a semantic meaning to the data fields that can be extracted by running the wrappers generated by the above modules.

Discussion:

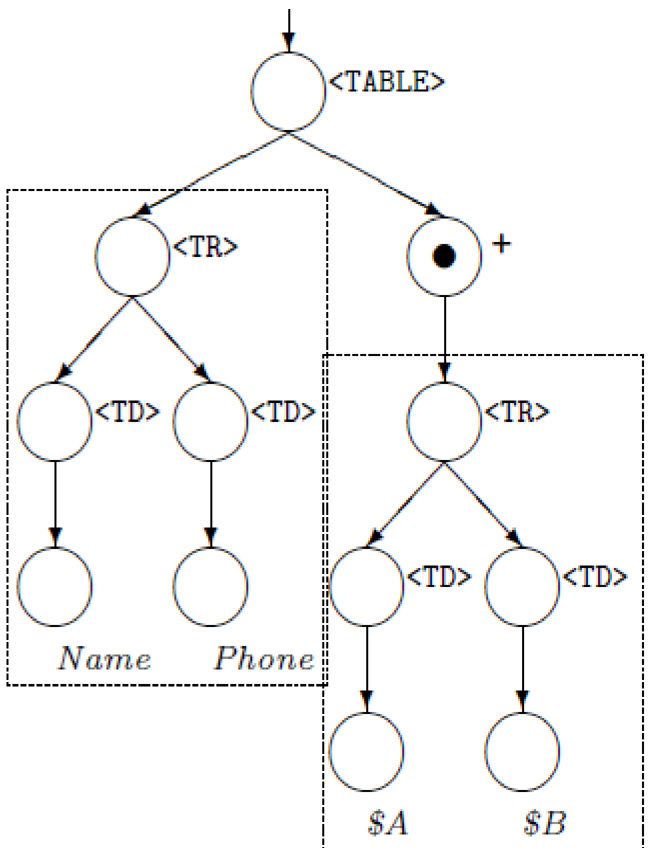
How to label the data item extracted from the page?

The Labeler (methods)

- To be done manually.
- Adoption of knowledge representation techniques, by some domain ontology.
- Based on a generalized notion of closeness between wrapper's tokens and non-terminal symbols.

The Labeler

... <TABLE> <TR> <TD> *Name* </TD>
<TD> *Phone* </TD> </TR>
(<TR> <TD> $A </TD> <TD> $B </TD> </TR>)$^+$
</TABLE> ...



- Check whether the pattern sub-tree is adjacent with some isomorphic sub-tree. The leaves of the discovered tree can be selected as names for the non-terminals of the patterns tree.
- namely, the strings "name" and "phone" – are candidate to be used as names for the non-terminals $A and $B respectively.

The Labeler (cont.)

Richness of the Web itself :

- it is possible that in some page *a given data item is associated with some information describing its meaning*.
- It is reasonable that in some of the pages retrieved by the search engine, *the input value is explicitly associated with some descriptive text*.

Further Research:

Simultaneous Record Detection and Attribute Labeling in Web Data Extraction

Jun Zhu[*†] Zaiqing Nie[‡] Ji-Rong Wen[‡] Bo Zhang[†] Wei-Ying Ma[‡]

[†]Department of Computer Science & Technology
Tsinghua University
Beijing, China

[‡]Web Search & Mining Group
Microsoft Research Asia
Beijing, China

[†]{jun-zhu, dcszb}@mails.tsinghua.edu.cn [‡]{znie, jrwen, wyma}@microsoft.com

ABSTRACT

Recent work has shown the feasibility and promise of template-independent Web data extraction. However, existing approaches use decoupled strategies – attempting to do data record detection and attribute labeling in two separate phases. In this paper, we show that separately extracting data records and attributes is highly ineffective and propose a probabilistic model to perform these two tasks simultaneously. In our approach, record detection can benefit from the availability of semantics required in attribute labeling and, at the same time, the accuracy of attribute labeling can be improved when data records are labeled in a collective manner. The proposed model is called Hierarchical Conditional Random Fields. It can efficiently integrate all useful features by learning their importance, and it can also incorporate hierarchical interactions which are very important for Web data extraction. We empirically compare the proposed model with existing decoupled approaches for product information extraction, and the results show significant improvements in both record detection and attribute labeling.

labeling in two separate phases. This paper studies how to extend existing Web data extraction methods to achieve the mutual enhancement of record detection and attribute labeling.

1.1 Motivating Example

We begin by illustrating the problem with an example, drawn from an actual application of product information extraction. The goal of the application is to extract meta-data about real-world products from every product page on the Web. Specifically, for each crawled Web page, we first use a classifier to decide whether it is a product page and then extract the *name*, *image*, *price* and *description* of each product from detected product pages.

Our statistical study on 51K randomly crawled Web pages shows that about 12.6 percent are product pages. That is, there are about 1 billion product pages within a search index containing 9 billion crawled Web pages. If all of these pages or just half of them are correctly extracted, we will have a huge collection of meta-data about real-world products that could be used for further

Summarization

- lixTo system (interactive wrapper generation, semi-supervised)

- Roadrunner system (data-intensive page extraction, unsupervised)

References

- Robert Baumgartner, et al.”Visual web information extraction with Lixto” Proceedings of the 27th VLDB Conference,2001.
- Valter Crescenzi, et al. ” Automatic Web Information Extraction in the ROADRUNNER system” LNCS 2465, pp. 264–277, 2002.
- Jun Zhu,et al, Simultaneous record detection and attribute labeling in Web data Extraction KDD 2006

Never-Ending Learning (NELL)

- Read the web 24 hours/day since Jan.2010.
- Acquired a knowledge base with 80 million confidence-weighted beliefs.
- http://rtw.ml.cmu.edu

Problem Statement

- A set $L=\{L_i\}$ of **learning tasks**.

where $L_i=(T_i,P_i,E_i)$ performance metric P_i,on a given performance task T_i, through a given type of experience E_i;

- A set of **coupling constraints** $C=\{\varphi_K,V_{ki}\}$

φ_K is a real-valued function over two or more learning tasks, specifying the degree of satisfaction of the constraint.

V_{ki} a vector of indices over learning tasks.

Problem Statement (cont.)

$$
\begin{aligned}
\mathscr{L} &= (L, C) \\
L &= \{\langle T_i, P_i, E_i \rangle\} \\
C &= \{\langle \phi_k, V_k \rangle\}
\end{aligned}
$$

Above, each performance task T_i is a pair $T_i \equiv \langle X_i, Y_i \rangle$ defining the domain and range of a function to be learned $f_i^* : X_i \rightarrow Y_i$. The performance metric $P_i : f \rightarrow \mathbb{R}$ defines the optimal learned function f_i^* for the ith learning task:

$$f_i^* \equiv \arg\max_{f \in F_i} P_i(f)$$

where F_i is the set of all possible functions from X_i to Y_i.

Input of the System

- Input
  - Ontology and binary relations (~800 categories and relations)
  - 10-20 Labeled training examples for each category and relation
  - The web and access to 100,000 Google API search queries.
  - Occasional interaction with humans
- System Doing
  - read (extract) more beliefs from the web
  - remove old incorrect beliefs
  - populate a growing knowledge base containing a confidence and provenance for each belief
  - learn to read better than the previous day.
- Result: KB with +90.000,000 extracted beliefs (different levels of confidence)

Output of the system

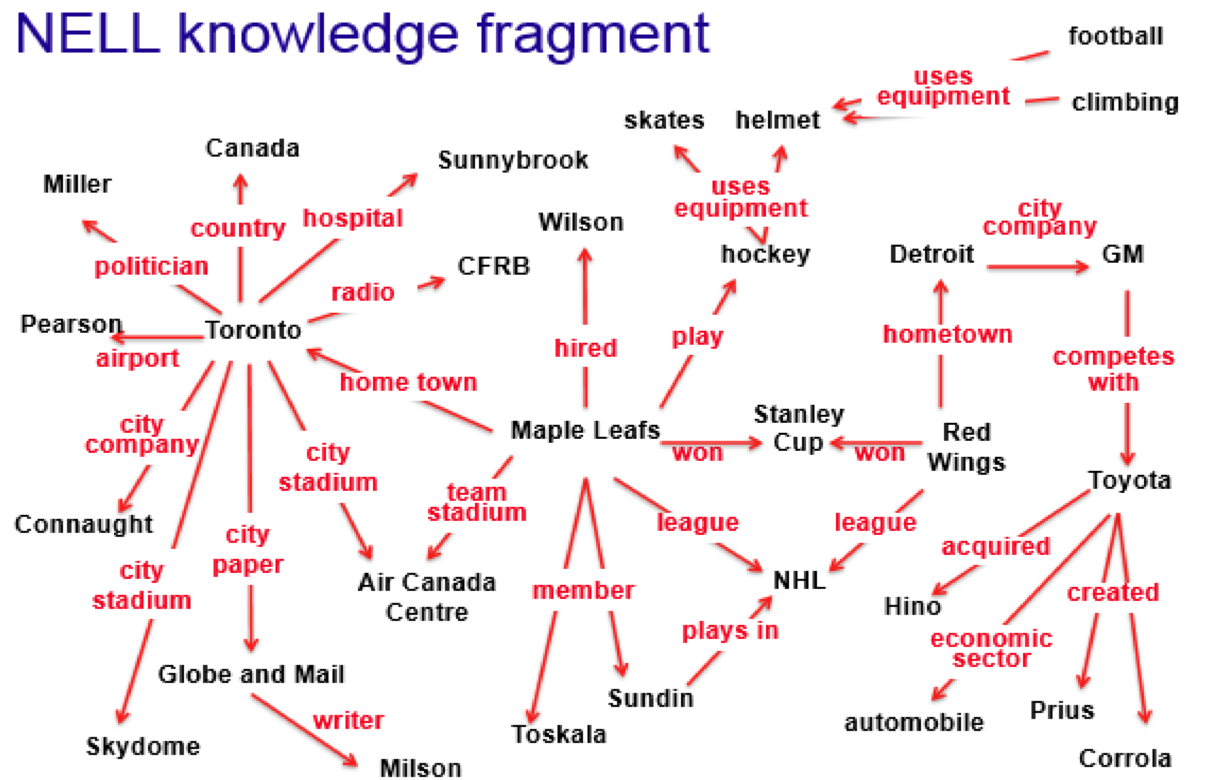
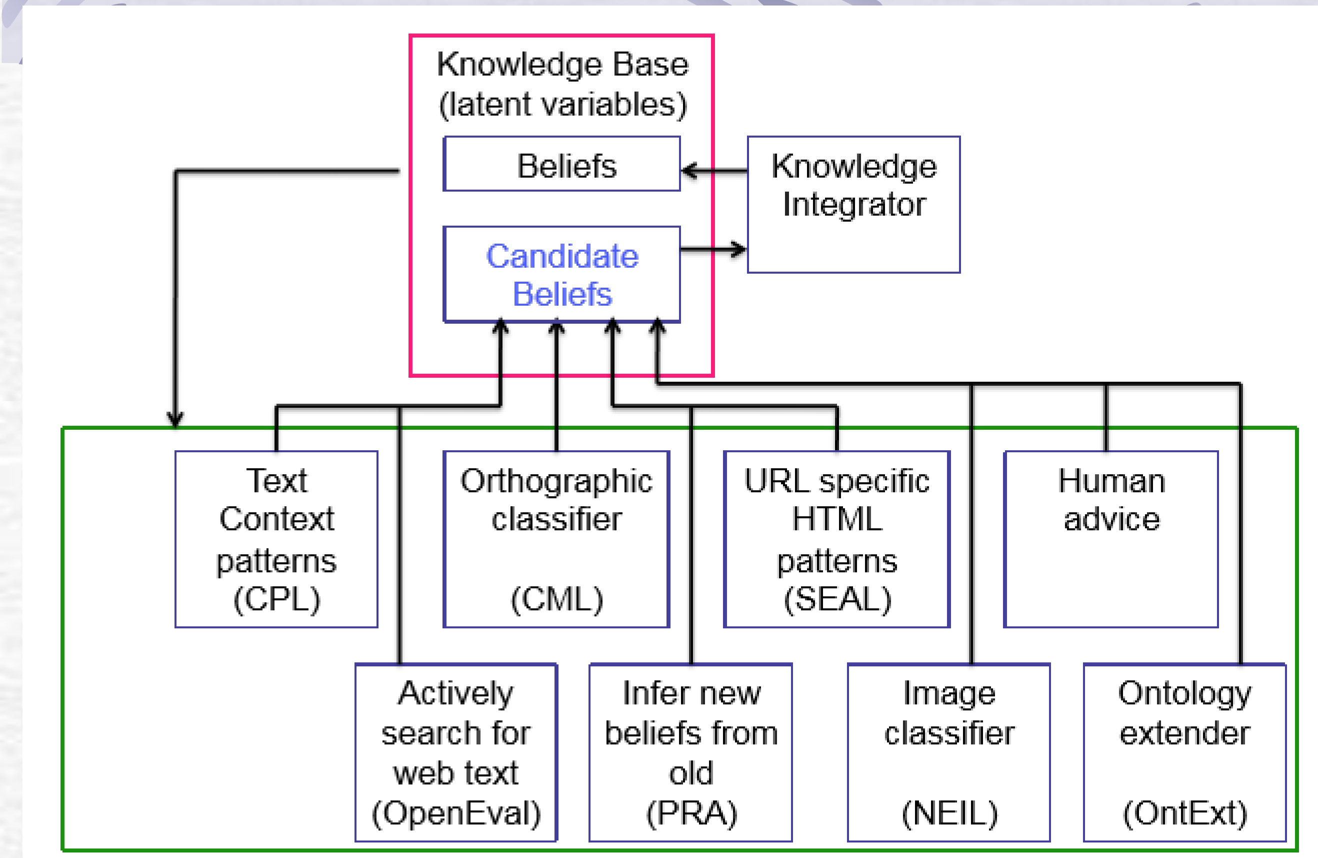
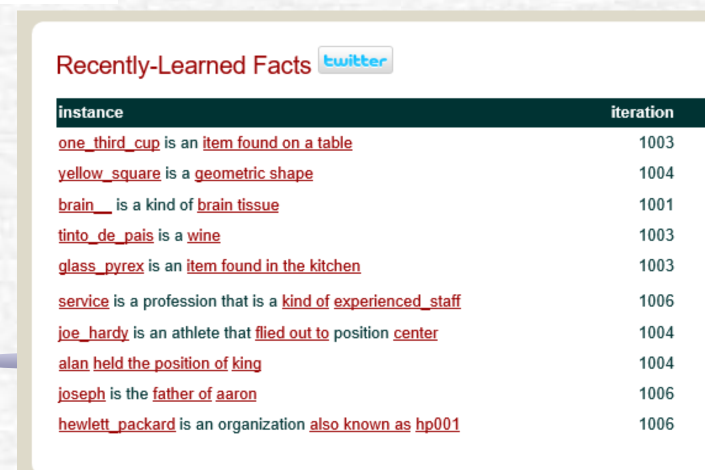


Figure 1: **Fragment of the 80 million beliefs NELL has read from the web.** Each edge represents a belief triple (e.g., play(MapleLeafs, hockey), with an associated confidence and provenance not shown here. This figure contains only correct beliefs from NELL's KB – it has many incorrect beliefs as well since NELL is still learning.

NELL architecture





Recently-Learned Facts twitter

instance	iteration
one_third_cup is an item found on a table	1003
yellow_square is a geometric shape	1004
brain__ is a kind of brain tissue	1001
tinto_de_pais is a wine	1003
glass_pyrex is an item found in the kitchen	1003
service is a profession that is a kind of experienced_staff	1006
joe_hardy is an athlete that flied out to position center	1004
alan held the position of king	1004
joseph is the father of aaron	1006
hewlett_packard is an organization also known as hp001	1006

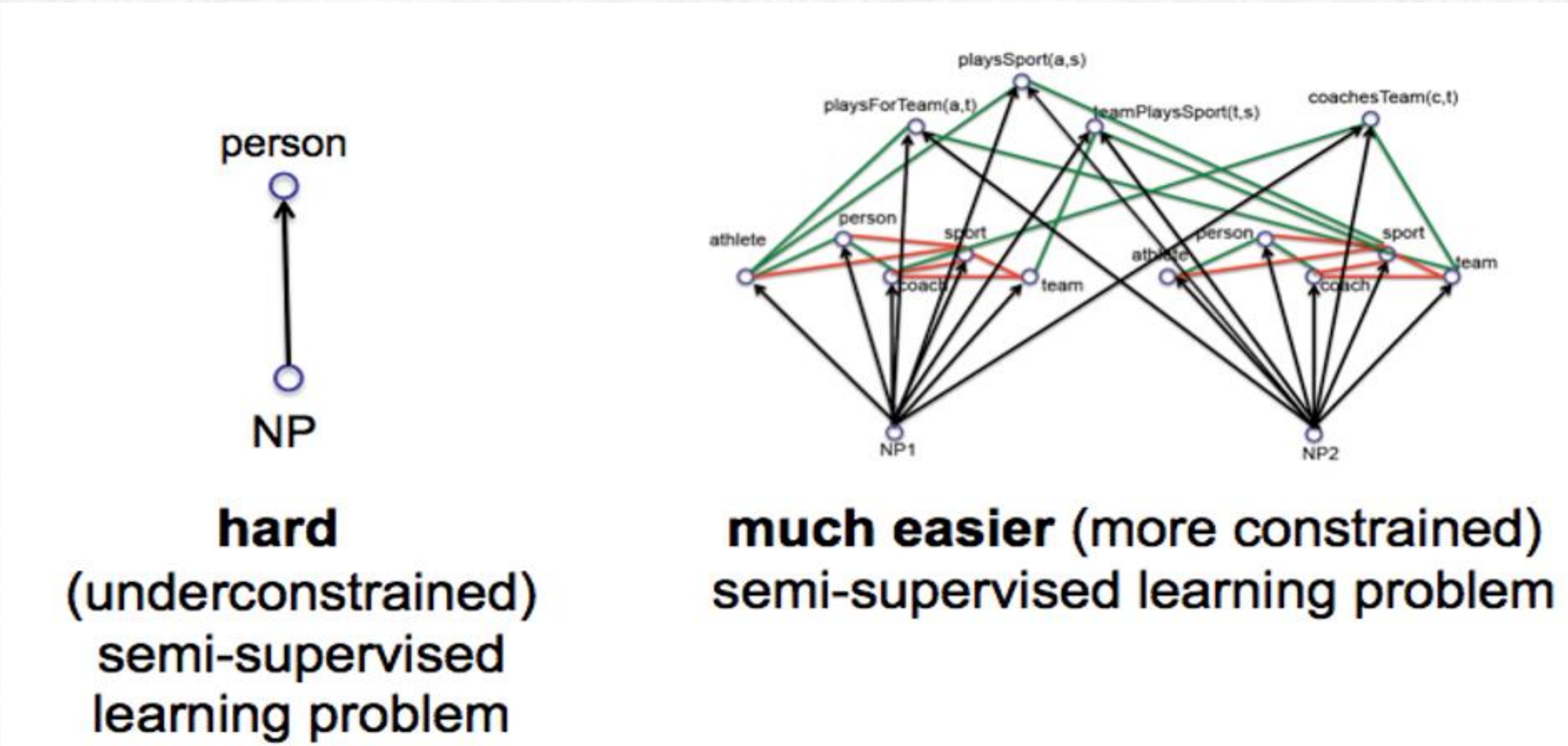
Techniques used for learning tasks

- **Category classification**: NELL learns different boolean functions for each of the 280 categories in its ontology, allowing noun phrases to refer to entities in multiple semantic categories.
- **Relation classification**: NELL learns distinct boolean-valued classification functions for each of the 327 relations in its ontology.
- **Entity Resolution**: Functions that classify noun phrase pairs by whether they are synonyms.
- **Inference rules among belief triples**: Functions that map from NELL's current KB, to new beliefs it should add to its KB.

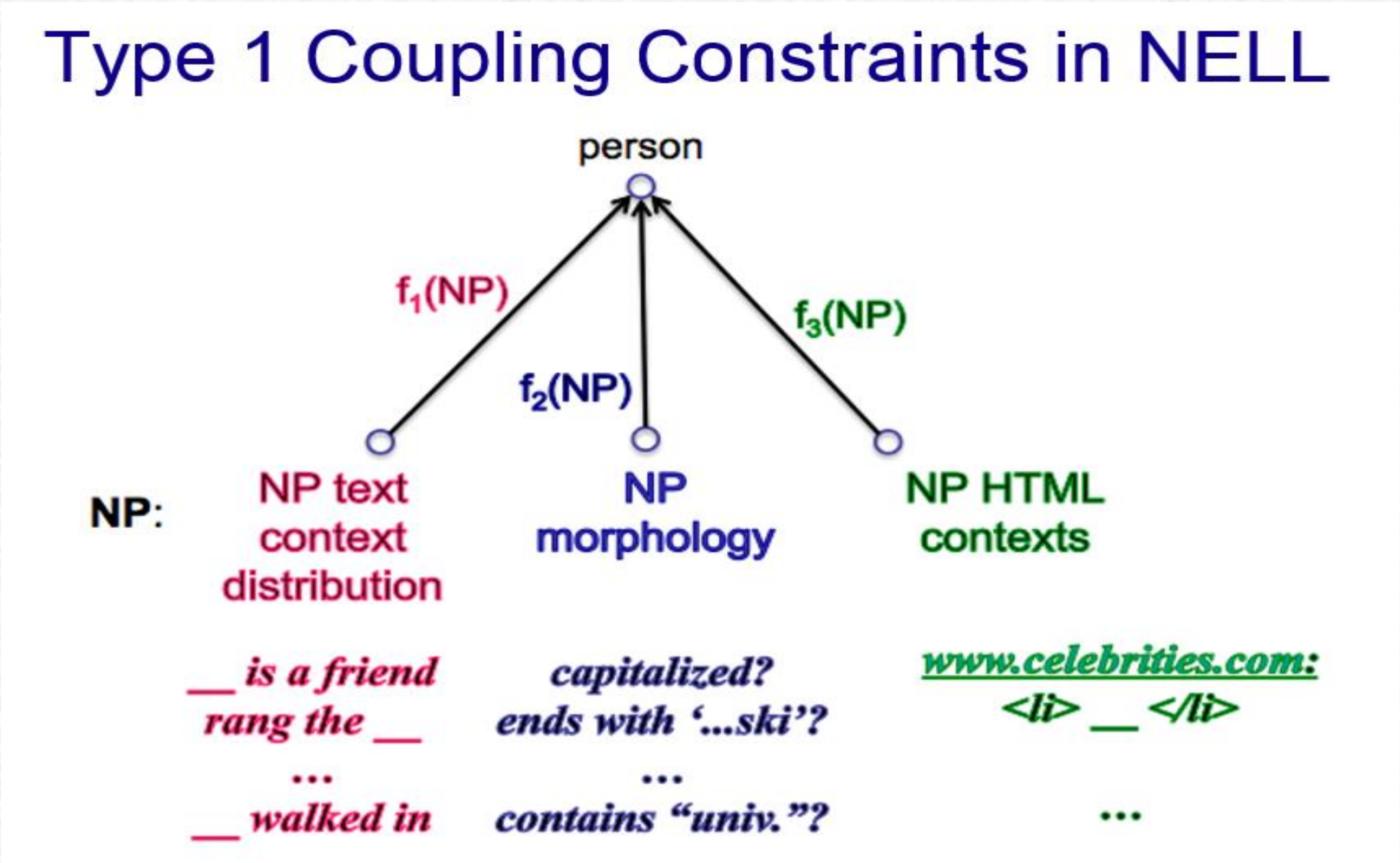
Techniques used for coupling constraints

- Multi-view co-training coupling.
- Subset/superset coupling.
- Multi-label mutual exclusion coupling.
- Coupling relations to their argument types.
- Horn clause coupling.

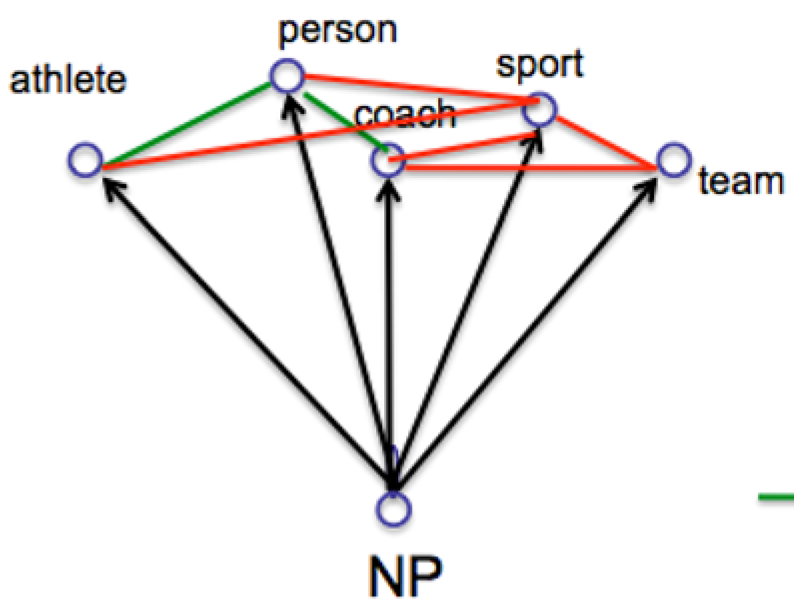
Coupled semi-supervised training of many functions



Co-training, Multiview



Type 2 Coupling Constraints in NELL

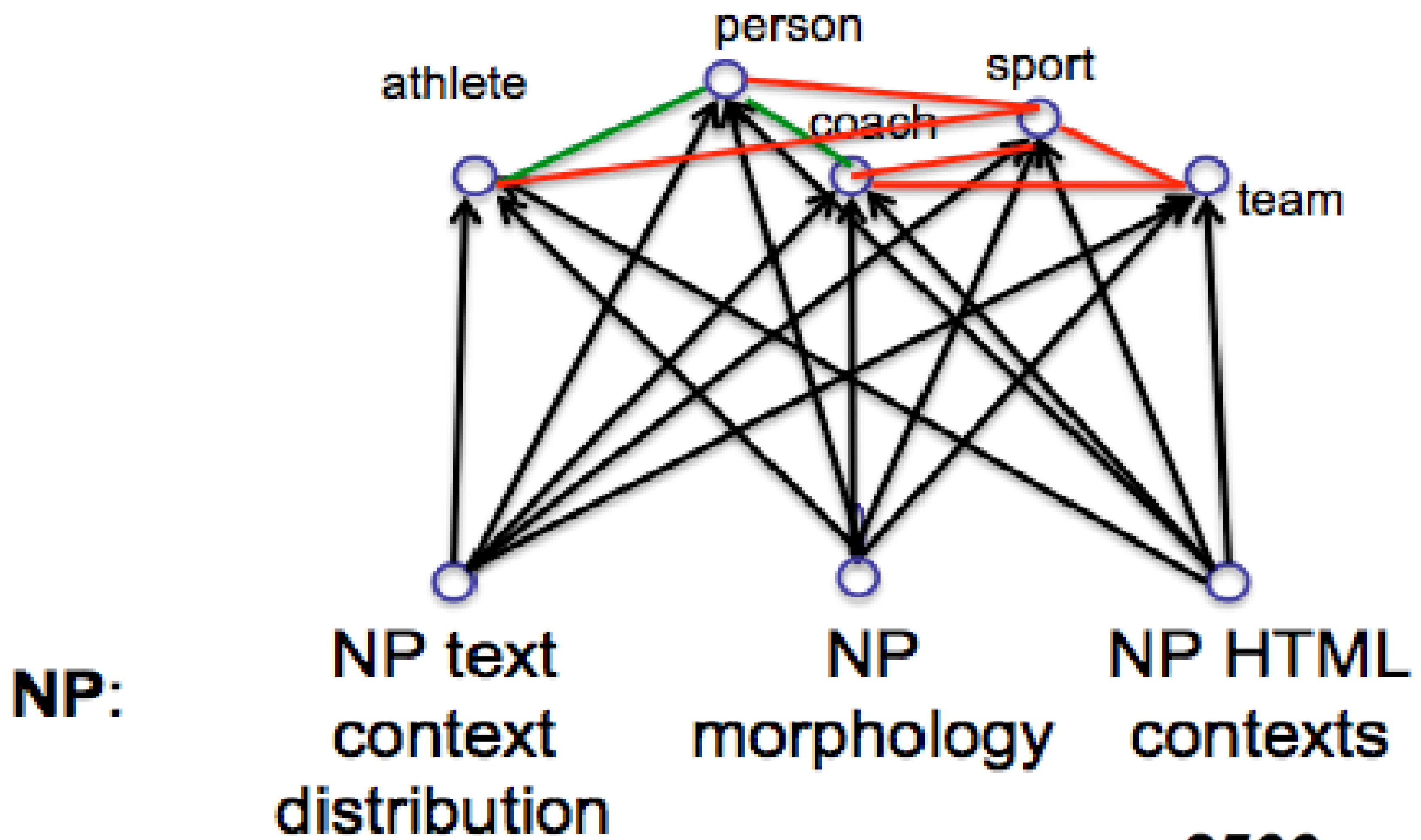


<u>Learn functions with the same input, different outputs, where some constraint are known.</u>

athlete(NP) → person(NP)

athlete(NP) → NOT sport(NP)

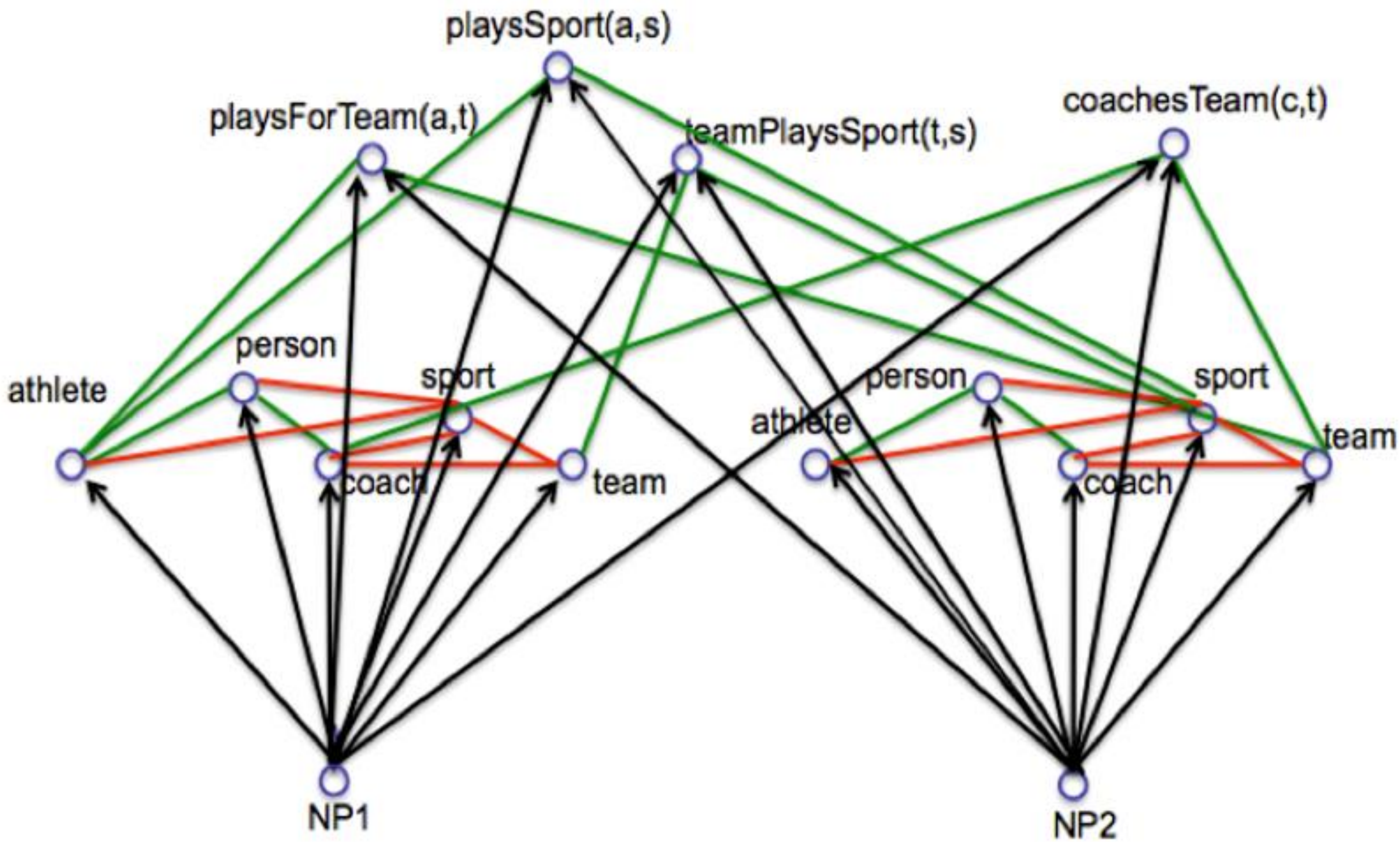
NOT athlete(NP) ← sport(NP)



over 2500 coupled functions in NELL

Type 3 Coupling: Argument Types

Constraint: f3(x1,x2) → (f1(x1) AND f2(x2))



playsSport(NP1,NP2) → athlete(NP1), sport(NP2)

Advantages of NELL

- To achieve successful semi-supervised learning, couple the training of many different learning tasks.
- Allow the agent to learn additional coupling constraints.
- Learn new representations that cover relevant phenomena beyond the initial representation.
- Organize the set of learning tasks into an easy-to increasingly-difficult curriculum