



# Text Processing on the Web

## Week 12 Information Extraction

(source of IE slides mostly taken from Stanford, in turn borrowed from Oren Etzioni, Andrew McCallum, Nick Kushmerick, BBN, Ray Mooney and Tat-Seng Chua)



# Recap

- **Named Entities** are a key milestone in transitioning to doc retrieval to information retrieval
  - Documents are about human interests.
- Features are source dependent: spoken, written
- Contextual features and enough data can supplant manual resources



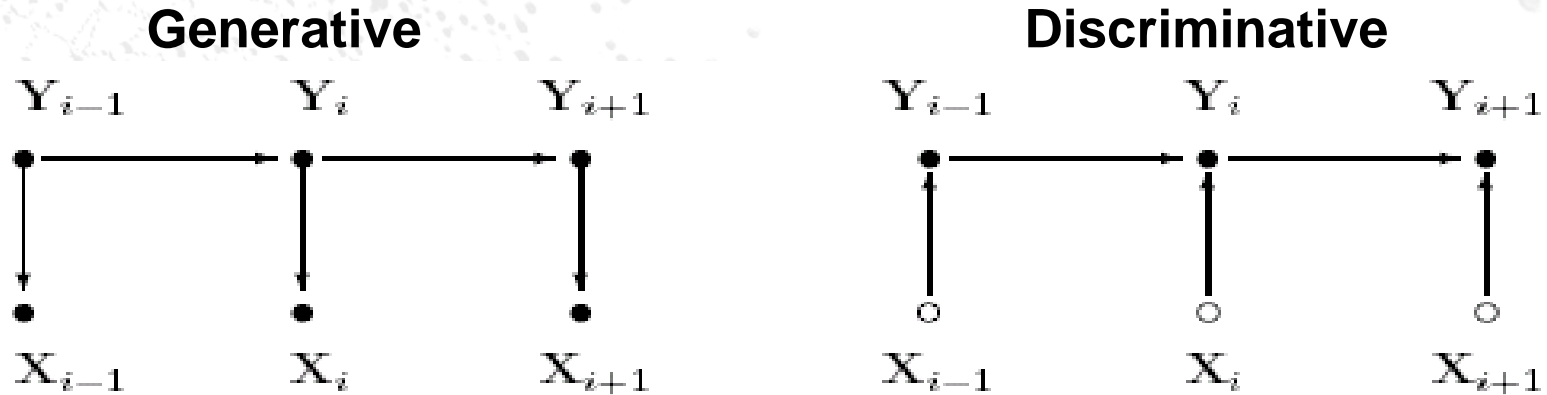
# Outline

## Information Extraction

- Discriminative Models for Sequence Labeling
- Precursors: MUC
- Learning extraction patterns
  - Wrapper induction:  $\langle H, L, R, T \rangle$  &  $\langle L, M, R \rangle$
  - WHISK
  - GRID



# Discriminative Probabilistic Models



“Solve the problem you need to solve”:

The traditional approach inappropriately uses a generative *joint* model in order to solve a *conditional* problem in which the observations are given.

To classify we actually need  $p(y|x)$  – there’s no need to implicitly approximate  $p(x,y)$ .



# Discriminative / Conditional Models

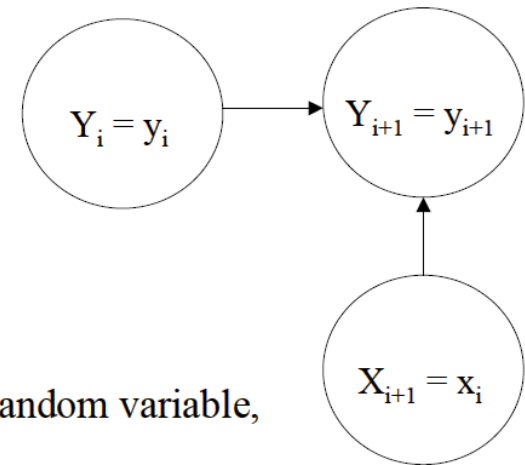
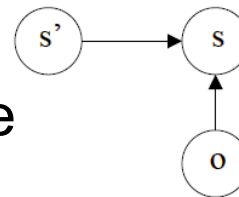
- Conditional probability  $P(\text{label seq } y \mid \text{observed seq } x)$  rather than joint probability  $P(y, x)$ 
  - Specify the probability of possible label sequences given an observation sequence
- Allow arbitrary, non-independent features on the observation sequence  $X$
- The probability of a transition between labels may depend on **past** and **future** observations
  - Relax strong independence assumptions in HMM



# Maximum Entropy Markov Models (MEMMs)

a.k.a. Conditional Markov Models CMMs

- Models probability of a state **given** an observation and just the previous state
  - Conditional probs are represented as exponential models based on arbitrary observation features



$A = a$ , where  $A$  is a random variable, and  $a$  is an outcome

- Given training set  $X$  with label sequence  $Y$ :
  - Train a model  $\theta$  that maximizes  $P(Y|X, \theta)$
  - For a new data sequence  $x$ , predict label  $y$  that maximizes  $P(y|x, \theta)$

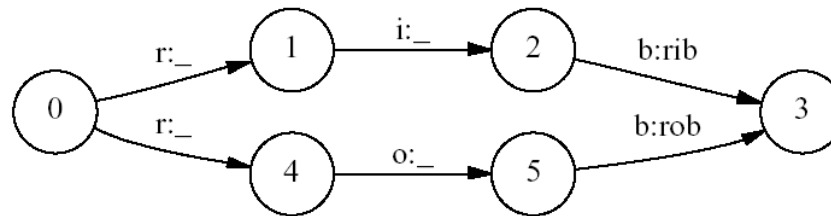
$$P(y' | y, x) = \frac{1}{Z(y, x)} \exp \left( \sum_k \underbrace{\lambda_k}_{\text{weight}} \underbrace{f_k(x, y, y')}_{\text{feature}} \right)$$

Per state normalization: all prob mass that arrives is distributed among its successor states



# The Label Bias Problem

- In MaxEnt's formulation, the prob mass that arrives at the state must be distributed among the possible successor states



- If one of transitions leaving state 0 occurs more frequently in training, its transition prob is greater, irrespective of the observation sequence
  - Especially in cases where there are few outgoing transitions (as in states 1, 2, 4 and 5).
- In the example, say that 'rib' is slightly more common than 'rob' in the training data. Then in the test data, if 'rob' occurs it will be classified as 'rib' as the the transition to 1 is more likely than to 4; the observation of the 'o' is effectively ignored as that it is only observed later at state 1.



# Conditional Random Fields (CRFs)

- CRFs have all the advantages of MEMMs without label bias problem
  - MEMM uses **per-state exponential** model for the conditional probabilities of next states given the current state
  - CRF has a **single exponential** model for the joint probability of the entire sequence of labels given the observation sequence
  - This difference means that some transitions have more influence than others depending on the corresponding observations in our previous example
- Undirected acyclic graph

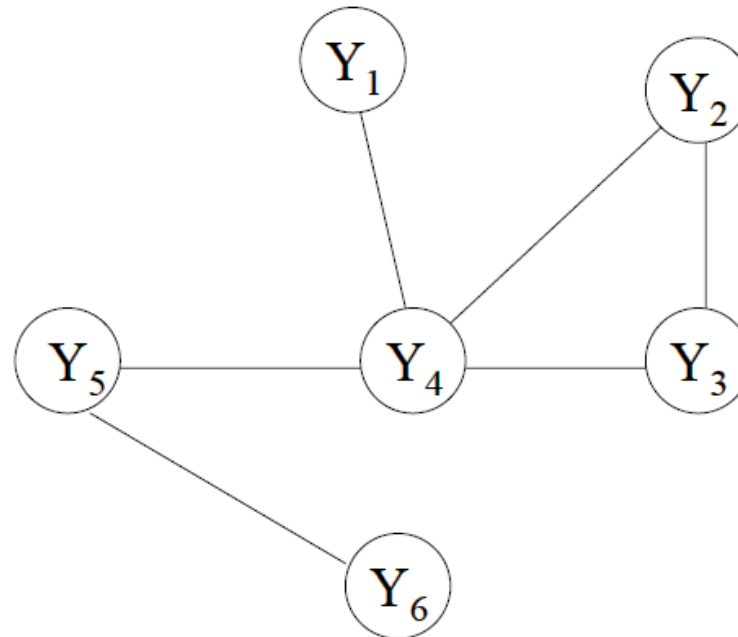




# Random Fields – Undirected Graphical Models

Let  $G = (Y, E)$  be a graph where each vertex  $Y_v$  is a random variable  
Suppose  $P(Y_v | \text{all other } Y) = P(Y_v | \text{neighbors}(Y_v))$  then  $Y$  is a  
random field

Example:



- $P(Y_5 | \text{all other } Y) = P(Y_5 | Y_4, Y_6)$



# CRF Definition

- $X$  – random variable over data sequences
- $Y$  – random variable over label sequences
- $Y_i$  is assumed to range over a finite label alphabet  $A$
- Discriminative approach:
  - We construct a conditional model  $p(y|x)$  and do not explicitly model marginal  $p(x)$



# CRF Distribution Function

$$p_{\theta}(y | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp \left( \sum_{e \in E, k} \lambda_k f_k(e, y|_e, \mathbf{x}) + \sum_{v \in V, k} \mu_k g_k(v, y|_v, \mathbf{x}) \right)$$

Where :

$\mathbf{V}$  = Set of random variables (observed and hidden)

$f_k$  and  $g_k$  = features

$g_k$  = State feature

$f_k$  = Edge feature

$\theta = (\lambda_1, \lambda_2, \dots, \lambda_n; \mu_1, \mu_2, \dots, \mu_n); \lambda_k$  and  $\mu_k$

are parameters to be estimated

$y|_e$  = Set of components of  $y$  defined by edge  $e$

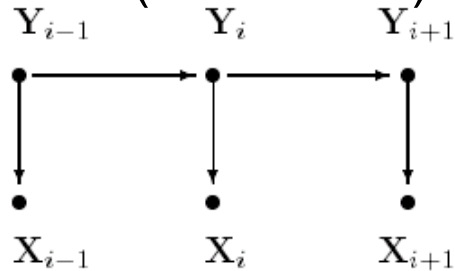
$y|_v$  = Set of components of  $y$  defined by vertex  $v$



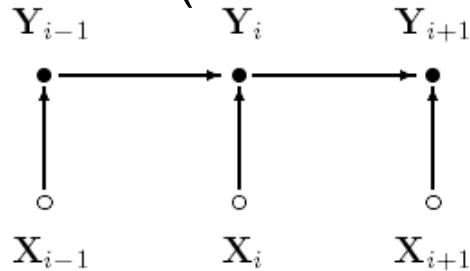
# Linear chain CRF

- We will handle the case when  $G$  is a simple chain:  $G = (V = \{1, \dots, m\}, E = \{(l, l+1)\})$

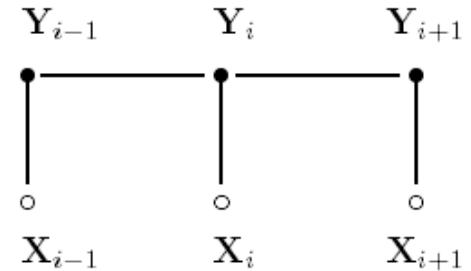
HMM (Generative)



MEMM (Discriminative)



CRF





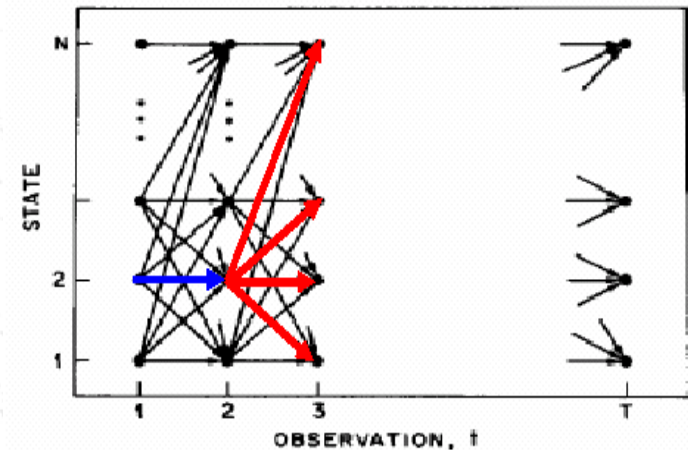
# CRF – the Learning Problem

- Assumption: the *features*  $f_k$  and  $g_k$  are given and fixed.
  - For example, a boolean feature  $g_k$  is TRUE if the word  $X_i$  is upper case and the label  $Y_i$  is a “noun”.
- The learning problem
  - We need to determine the parameters  $\Theta = (\lambda_1, \lambda_2, \dots ; \mu_1, \mu_2, \dots)$  from training data  $D = \{(x(i), y(i))\}$  with empirical distribution  $p_{\sim}(x, y)$ .



# Parameter Estimation for CRFs

- The parameter vector  $\Theta$  that maximizes the log-likelihood is found using an iterative scaling algorithm.
- We define standard HMM-like **forward** and **backward vectors**  $\alpha$  and  $\beta$ , which allow polynomial time calculations.
- However as the normalization is conditioned over the entire CRF (not over single vertices), it is expensive to compute  $\rightarrow$  **slow training time**





# Summary

- HMM:
  - Basic sequence labeling framework where labels are considered hidden and generate the observed words, and are just dependent on the previous state (Markovian assumption)
  - Fast algorithms for three classic problems
- CRF:
  - Discriminatively trained models  $P(y|x)$  as compared to modeling joint  $P(x,y)$  probability
  - Allows combination of arbitrary and overlapping observation features from both the past and future
  - main current limitation is the slow convergence of the training algorithm relative to MEMMs or HMMs, for which training is efficient.



# Web Information Extraction






# Product information

CNET.com - Shopping - Search Results for "ibm x21" - Microsoft Internet Explorer

Address: .0-1257,00.html?tag=&qat=ibm+x21&cn=&ca=1257

Found: 12 Displaying: 1-12

Resort by  
Most Popular | [Product Name](#) | [Manufacturer Name](#) | [CNET Review](#) | [Lowest Price](#)

<a href="#">256MB PC100 IBM THINKPAD A20M A21 A22 X21 X20 T21 T22 390X 600X</a> <a href="#">Add to my list</a>	Crucial Technology		<a href="#">Check Latest Prices</a> ▶ Price range: \$83.20-\$141.95
<a href="#">THINKPAD X21 P3-700 20GB 128MB W2K 12-SVGA ENET INTEL 56K</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">product info</a>	<a href="#">Check Latest Prices</a> ▶ Price range: \$1899.00-\$2923.36
<a href="#">IBM Thinkpad X21 (Pentium III, 700 MHz, 128 MB, 20 GB)</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">product info</a>	<a href="#">Check Latest Prices</a> ▶ Price range: \$2489.00-\$2996.47
<a href="#">THINKPAD X21 P3-700 20GB 128MB 98 12-XGA ENET INTEL 56K</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">product info</a>	<a href="#">Check Latest Prices</a> ▶ Price range: \$2048.95-\$2818.91
<a href="#">IBM ThinkPad X21 (Pentium III 700MHz, 128MB RAM, 20GB)</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">review</a> 	<a href="#">Check Latest Prices</a> ▶ Price range: \$2339.00-\$2818.91
<a href="#">TP X21 NB P3/700 128MB 20GB 12.1 56K ETH W2K</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">product info</a>	<a href="#">Check Latest Prices</a> ▶ Price range: \$2420.39-\$2925.21
<a href="#">IBM ThinkPad X21 (Pentium III, 700 MHz, 128 MB, 20 GB)</a> <a href="#">Add to my list</a>	IBM Corp.	<a href="#">product info</a>	<a href="#">Check Latest Prices</a> ▶ Price range: \$2322.00-\$2892.02
<a href="#">THINKPAD X21 P3-700 20GB 128MB 98 12 SVGA ENET</a>	IBM Corp.		<a href="#">Check Latest Prices</a> ▶ Price range: \$2378.96-\$2818.91

Done Internet



# Textual inconsistency

- Image Capture Device: 1.68 million pixel 1/2-inch CCD sensor
- Image Capture Device Total Pixels Approx. 3.34 million  
Effective Pixels Approx. 3.24 million
- Image sensor Total Pixels: Approx. 2.11 million-pixel
- Imaging sensor Total Pixels: Approx. 2.11 million 1,688 (H) x 1,248 (V)
- CCD Total Pixels: Approx. 3,340,000 (2,140[H] x 1,560 [V] )
  - Effective Pixels: Approx. 3,240,000 (2,088 [H] x 1,550 [V] )
  - Recording Pixels: Approx. 3,145,000 (2,048 [H] x 1,536 [V] )

These all came off the same manufacturer's website.  
And this is a very technical (precise) domain. Try sofa beds.



# Classified Advertisements (Real Estate)

## Background:

- Advertisements are plain text
- Lowest common denominator: only thing that 70+ newspapers with 20+ publishing systems can all handle

Won't IR work on this since it's plain text?

```
<ADNUM>2067206v1</ADNUM>  
<DATE>March 02, 1998</DATE>  
<ADTITLE>MADDINGTON  
$89,000</ADTITLE>  
<ADTEXT>  
OPEN 1.00 - 1.45<BR>  
U 11 / 10 BERTRAM ST<BR>  
NEW TO MARKET Beautiful<BR>  
3 brm freestanding<BR>  
villa, close to shops & bus<BR>  
Owner moved to Melbourne<BR>  
ideally suit 1st home buyer,<BR>  
investor & 55 and over.<BR>  
Brian Hazelden 0418 958 996<BR>  
R WHITE LEEMING 9332 3477  
</ADTEXT>
```



# Why doesn't IR work?

What you search for in real estate advertisements:

- Suburbs. You might think easy, but:
  - Multiple property ads have different suburbs
  - Phrases: Only 45 minutes from Parramatta
- Money: want a range, not a textual match
  - Multiple amounts: was \$155K, now \$145K
  - Variations: offers in the high 700s [*but not* rents for \$270]
- Bedrooms: similar issues (br, bdr, beds, B/R)



# Information Extraction: Filling **templates** with **slot values**

- Goal: being able to answer semantic queries (a.k.a. “database queries”) using “unstructured” natural language sources
- Identify specific pieces of information in a un-structured or semi-structured textual document.
- Transform this unstructured information into structured relations in a database/ontology.

## Suppositions:

- A lot of information that *could* be represented in a structured semantically clear format isn't
- It may be costly, not desired, or not in one's control (screen scraping) to change this.



# Other applications of IE Systems

- Job resumes [Whizbang labs]
- Seminar announcements [CMU corpus]
- Continuing education courses info from the web
- Molecular biology information from MEDLINE, e.g,  
Extracting gene drug interactions from biomed texts
- Summarizing medical patient records by extracting  
diagnoses, symptoms, physical findings, test results

Many more ...



# Template Types

- Slots in template typically filled by a substring from the document.
- Some slots may have a fixed set of pre-specified possible fillers that may not occur in the text itself.
  - Terrorist act: threatened, attempted, accomplished.
  - Job type: clerical, service, custodial, etc.
  - Company type: SEC code
- Some slots may allow multiple fillers.
  - Programming language
- Some domains may allow multiple extracted templates per document.
  - Multiple apartment listings in one ad



# MUC: the genesis of IE

- US defense organization DARPA funded significant efforts in IE in the early to mid 1990's.
- Message Understanding Conference (MUC) was an annual event/competition where results were presented.
- Focused on extracting information from news (unstructured) articles:
  - Terrorist events
  - Industrial joint ventures
  - Company management changes
- Information extraction of particular interest to the intelligence community (CIA, NSA).



## *Example of IE from FASTUS (1993)*

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be supplied to Japan.

The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and “metal wood” clubs a month.

### TIE-UP-1

Relationship: TIE-UP

Entities: “Bridgestone Sport Co.”

“a local concern”

“a Japanese trading house”

Joint Venture Company:

“Bridgestone Sports Taiwan Co.”

Activity: ACTIVITY-1

Amount: NT\$200000000

### ACTIVITY-1

Activity: PRODUCTION

Company:

“Bridgestone Sports Taiwan Co.”

Product:

“iron and ‘metal wood’ clubs”

Start Date:

DURING: January 1990

## *Example of IE: FASTUS(1993): Resolving anaphora*

Bridgestone Sports Co. said Friday it had set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be supplied to Japan.

The joint venture, **Bridgestone Sports Taiwan Co.**, capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and “metal wood” clubs a month.

### TIE-UP-1

**Relationship:** TIE-UP

**Entities:** “Bridgestone Sport Co.”

“a local concern”

“a Japanese trading house”

**Joint Venture Company:**

“Bridgestone Sports Taiwan Co.”

**Activity:** ACTIVITY-1

**Amount:** NT\$2000000000

### ACTIVITY-1

**Activity:** PRODUCTION

**Company:**

“Bridgestone Sports Taiwan Co.”

**Product:**

“iron and ‘metal wood’ clubs”

**Start Date:**

DURING: January 1990

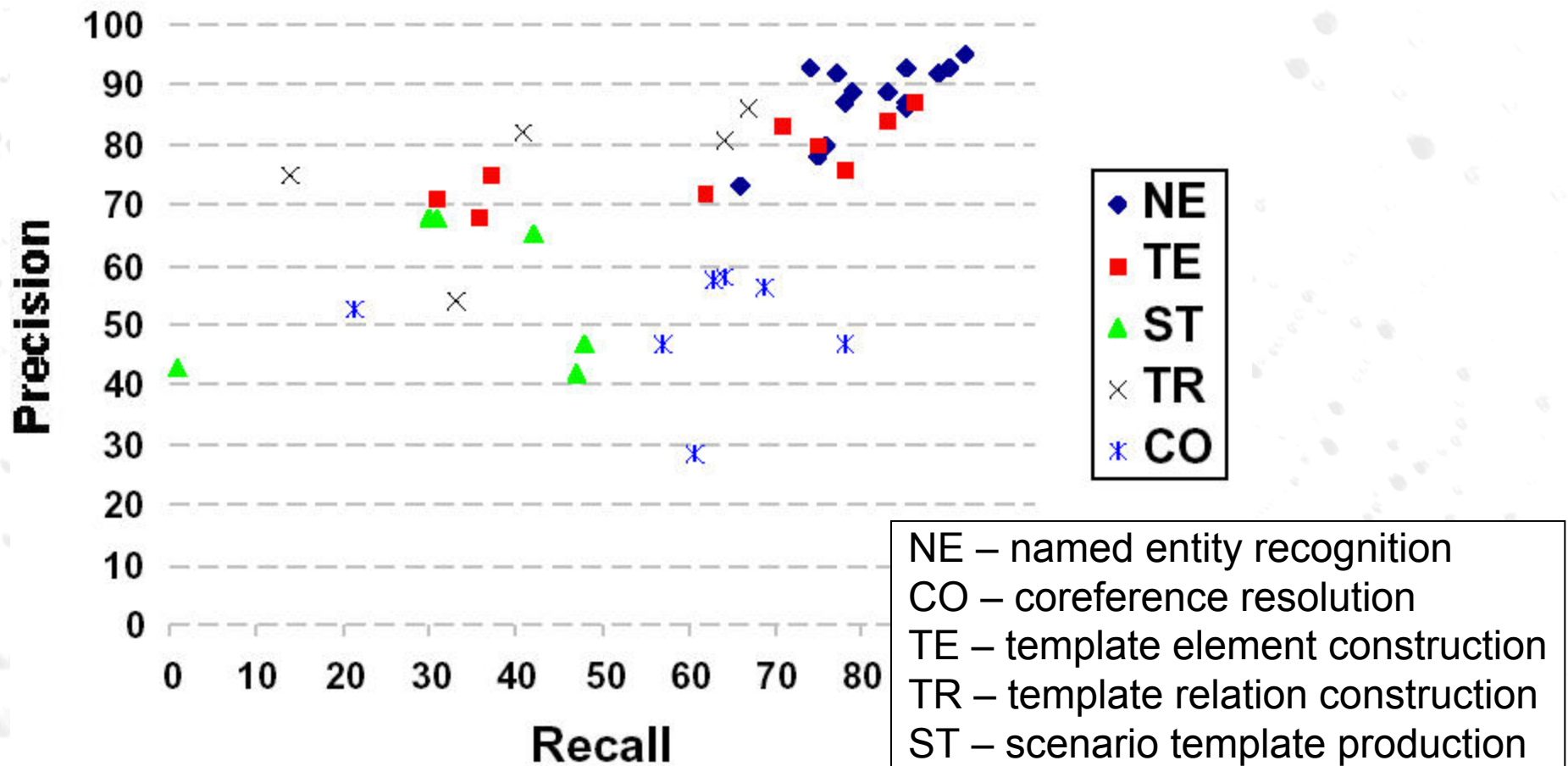


# Evaluating IE Accuracy

- Evaluate performance on independent, manually-annotated test data unused during system development.
- Measure for each test document:
  - Total number of correct extractions in the solution template:  $N$
  - Total number of slot/value pairs extracted by the system:  $E$
  - Number of extracted slot/value pairs that are correct (i.e. in the solution template):  $C$
- Compute average value of metrics adapted from IR:
  - Recall =  $C/N$
  - Precision =  $C/E$
  - F-Measure = Harmonic mean of recall and precision



# MUC Information Extraction: State of the Art c. 1997





# Wrappers: Simple Extraction Patterns

- Specify an item to extract for a slot using a regular expression pattern.
  - Price pattern: “\b\\$\d+(\.\d{2})?\b”
- May require preceding (pre-filler) pattern to identify proper context.
- May require succeeding (post-filler) pattern to identify the end of the filler.
- E.g., Amazon list price:
  - Pre-filler pattern: “<b>List Price:</b> <span class=listprice>”
  - Filler pattern: “.+”
  - Post-filler pattern: “</span>”



# Simple Template Extraction

- Extract slots in order, starting the search for the filler of the  $n+1$  slot where the filler for the  $n$ th slot ended. Assumes slots always in a fixed order.
  - Title
  - Author
  - List price
  - ...
- Make patterns specific enough to identify each filler always starting from the beginning of the document.



# Pre-Specified Filler Extraction

- If a slot has a fixed set of pre-specified possible fillers, text categorization can be used to fill the slot.
  - Job category
  - Company type
- Treat each of the possible values of the slot as a category, and classify the entire document to determine the correct filler.



# Wrapper tool-kits

- Wrapper toolkits: Specialized programming environments for writing & debugging wrappers by hand
- Examples
  - World Wide Web Wrapper Factory (W4F)  
[[db.cis.upenn.edu/W4F](http://db.cis.upenn.edu/W4F)]
  - Java Extraction & Dissemination of Information (JEDI)  
[[www.darmstadt.gmd.de/oasys/projects/jedi](http://www.darmstadt.gmd.de/oasys/projects/jedi)]
  - Jungle Corporation





# Task: Wrapper Induction

- Handcoding a wrapper isn't very viable
  - sites are numerous, and their surface structure mutates rapidly (around 10% failures each month)
- How about **Wrapper Induction** ?
  - Sometimes, the relations are structural.
    - Web pages generated by a database.
    - Tables, lists, etc.
  - Wrapper induction is usually regular relations which can be expressed by the *structure* of the document:
    - E.g., the item in bold in the 3<sup>rd</sup> column of the table is the price



# Wrapper induction

Highly regular  
source documents



Relatively simple  
extraction patterns



Efficient  
learning algorithm

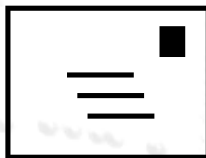
- Writing accurate patterns for each slot for each domain (e.g. each web site) requires laborious software engineering.
- Alternative is to use machine learning:
  - Build a training set of documents paired with human-produced filled extraction templates.
  - Learn extraction patterns for each slot using an appropriate machine learning algorithm.



# Wrapper induction: Delimiter-based extraction



```
<HTML><TITLE>Some Country Codes</TITLE>  
<B>Congo</B> <I>242</I><BR>  
<B>Egypt</B> <I>20</I><BR>  
<B>Belize</B> <I>501</I><BR>  
<B>Spain</B> <I>34</I><BR>  
</BODY></HTML>
```



Use **<B>**, **</B>**, **<I>**, **</I>** for extraction



# Learning LR wrappers

labeled pages

wrapper

```
<HTML><HEAD>Some Country Codes</HEAD>
<HTML><HEAD>Some Country Codes</HEAD>
<HTML><HEAD>Some Country Codes</HEAD>
<HTML><HEAD>Some Country Codes</HEAD>
<B>Congo</B> <I>242</I><BR>
<B>Egypt</B> <I>20</I><BR>
<B>Belize</B> <I>501</I><BR>
<B>Spain</B> <I>34</I><BR>
</BODY></HTML>
```

→  $\langle l_1, r_1, \dots, l_K, r_K \rangle$

**Example:** Find 4 strings

$\langle \langle B \rangle, \langle /B \rangle, \langle I \rangle, \langle /I \rangle \rangle$

$\langle l_1, r_1, l_2, r_2 \rangle$



# LR: Finding $r_1$

```
<HTML><TITLE>Some Country Codes</TITLE>  
<B>Congo</B> <I>242</I><BR>  
<B>Egypt</B> <I>20</I><BR>  
<B>Belize</B> <I>501</I><BR>  
<B>Spain</B> <I>34</I><BR>  
</BODY></HTML>
```

*$r_1$  can be any prefix  
eg </B>*



# LR: Finding $l_1$ , $l_2$ and $r_2$

```
<HTML><TITLE>Some Country Codes</TITLE>  
<B>Congo</B> <I>242</I><BR>  
<B>Egypt</B> <I>20</I><BR>  
<B>Belize</B> <I>501</I><BR>  
<B>Spain</B> <I>34</I><BR>  
</BODY></HTML>
```

$r_2$  can be any *prefix*  
eg <I>

$l_2$  can be any *suffix*  
eg <I>

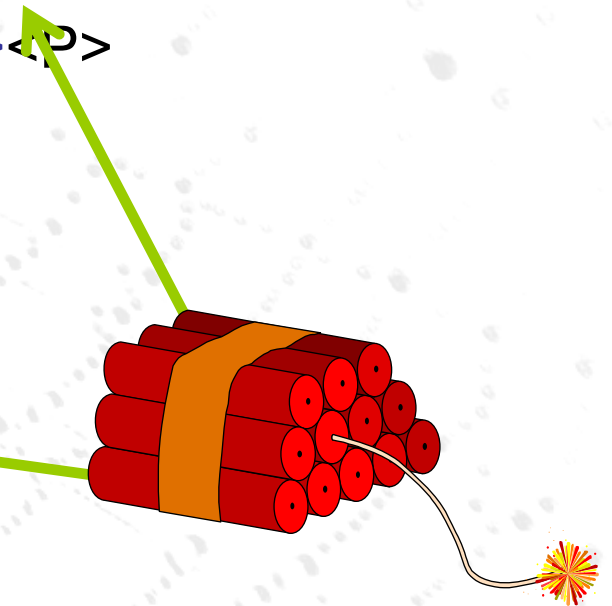
$l_1$  can be any *suffix*  
eg <B>



# A problem with LR wrappers

## *Distracting text in head and tail*

```
<HTML><TITLE>Some Country Codes</TITLE>  
<BODY><B>Some Country Codes</B><P>  
<B>Congo</B> <I>242</I><BR>  
<B>Egypt</B> <I>20</I><BR>  
<B>Belize</B> <I>501</I><BR>  
<B>Spain</B> <I>34</I><BR>  
<HR><B>End</B></BODY></HTML>
```





# One (of many) solutions: HLRT

Ignore page's **head** and **tail**

```
<HTML><TITLE>Some Country Codes</TITLE>  
<BODY><B>Some Country Codes</B><P>  
<B>Congo</B> <I>242</I><BR>  
<B>Egypt</B> <I>20</I><BR>  
<B>Belize</B> <I>501</I><BR>  
<B>Spain</B> <I>34</I><BR>  
<HR><B>End</B></BODY></HTML>
```

} head  
} body  
} tail

start of tail



Head-Left-Right-Tail wrappers





# More sophisticated wrappers

- LR and HLRT wrappers are extremely simple (though useful for  $\sim 2/3$  of real Web sites!)
- Recent wrapper induction research has explored more expressive wrapper classes  
[See references at end of lecture]
  - Disjunctive delimiters
  - Multiple attribute orderings
  - Missing attributes
  - Multiple-valued attributes
  - Hierarchically nested data
  - Wrapper verification and maintenance (!!)



# Natural Language Processing

- If extracting from (semi-)structured text, simple regex patterns usually work.
- If extracting from more unstructured, human-written text, NLP is often necessary.
  - Part-of-speech (POS) tagging: noun, verb, preposition, etc.
  - Syntactic parsing: NP, VP, PP
  - Named Entity Recognition:
  - Semantic categories (via WordNet):
    - KILL: kill, murder, assassinate, strangle, suffocate
- Extraction patterns can then use any of these sources of evidence.
  - Crime victim:
    - Prefiller: [POS: V, Hypernym: KILL]
    - Filler: [Phrase: NP]



# Three generations of IE systems

- Hand-Built Systems – Knowledge Engineering [1980s– ]
  - Rules written by hand
  - Require experts who understand both the systems and the domain
  - Iterative guess-test-tweak-repeat cycle
- Automatic, Trainable Rule-Extraction Systems [1990s– ]
  - Rules discovered automatically using predefined templates, using methods like ILP
  - Require huge, labeled corpora (effort is just moved!)
- Statistical Generative Models [1997 – ]
  - One decodes the statistical model to find which bits of the text were relevant, using HMMs or statistical parsers
  - Learning usually supervised; may be semi-supervised



# Trainable IE systems

## Pros

- Annotating text is simpler & faster than writing rules.
- Domain independent
- Domain experts don't need to be linguists or programmers.
- Learning algorithms ensure full coverage of examples.

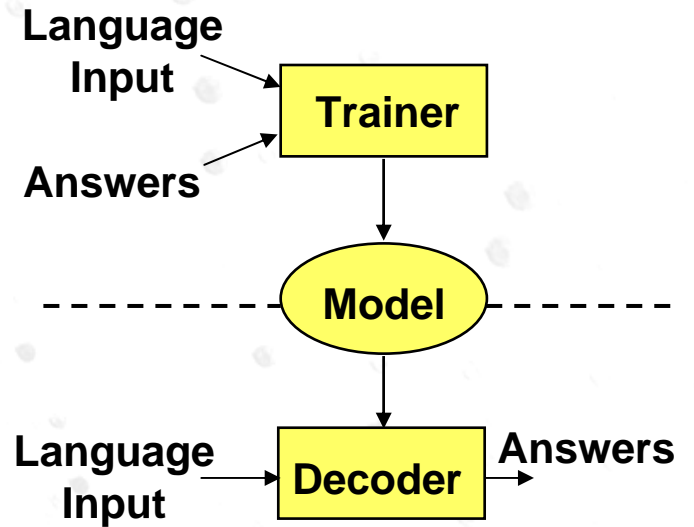
## Cons

- Hand-crafted systems perform better, especially at hard tasks.
- Training data might be expensive to acquire
- May need huge amount of training data
- Hand-writing rules isn't **that** hard!!



# Automatic Pattern-Learning Systems

- **Pros:**
  - Portable across domains
  - Tend to have broad coverage
  - Robust in the face of degraded input.
  - Automatically finds appropriate statistical patterns
  - System knowledge not needed by those who supply the domain knowledge.
- **Cons:**
  - Annotated training data, and lots of it, is needed.
  - Isn't necessarily better or cheaper than hand-built sol'n
- **Examples:**
  - Riloff et al., AutoSlog (UMass); Soderland WHISK (UMass); Mooney et al. Rapier (UTexas):
  - learn lexico-syntactic patterns from templates; all quite similar





# WHISK

- Learns a set of “regular expression” patterns to cover training instances

Capitol Hill – 1br twnhome. Fplc D/W W/D. Undergrnd pkg incl \$675. 3 BR, upper flr of turn of ctry HOME. Incl gar, grt N. Hill loc \$995 (206) 999-9999 <br> <i> <font size=-2> (This ad last run on 08/03/97.) </font> </i> <hr>

- Given the above as source and training instances:

**Rental:** Neighborhood: Capitol Hall, Bedrooms: 1, Price: 675

**Rental:** Neighborhood: Capitol Hall, Bedrooms: 3, Price: 995

Structured templates:  
Requires multi-slot  
extraction



# WHISK walkthrough

Is a covering algorithm

Picks new instances to be annotated to help make better rules (a form of *active learning*) by looking at its error rate

Given a training instance, WHISK tries to produce the most general rule that will cover the instance (Grow\_Rule)

```
Whisk(Reservoir) {  
  RulesSet = NULL  
  Training = NULL
```

```
  Repeat at user's request  
    Select batch of NewInst from Reservoir  
    (User tags the NewInst)  
    Add NewInst to Training  
    Discard rules with error on NewInst
```

```
    For each Inst in Training  
      For each Tag of Inst  
        If Tag is not cover by RuleSet  
          Rule = Grow_Rule(Inst, Tag,  
                          Training)
```

```
  Prune RuleSet  
}
```



# WHISK rule induction

- Starts off with empty rule (e.g., for three slots:

$( * ) * ( * ) * ( * ) *$

- For each slot, anchor the extraction pattern given annotated data:

- Try using slot terms themselves:

\* ( Neighborhood )

- Try using boundary terms just outside slot (context):

@s ( \* ) '-'

- Can use any syntactic semantic class or tag in creating tags, WHISK uses expected error to judge which tag to use in rules

@s[ Capitol Hill – 1 br  
twnhme fplc ...

- Test which does better and keep it, then proceed to next slot

'\*' in WHISK is not greedy, will take up least # of characters to match





# Problems with WHISK

- Selecting seeds by random
  - Potential for better approach
- Prefers semantic over syntactic, syntactic over lexical in cases of Laplacian error ties
  - Should use other information to break ties
- Uses token info instead of chunk info
  - Chunk info can be more useful in certain contexts



# Global Rule Induction for Text Documents (GRID)

A generalization of WHISK to a grid framework

- Use all data to select seeds, rather than by random choice
  - Select seeds by coverage and Lapacian error rate
  - Coverage accounts for positive and negative instances
- Uses phrases and chunks rather than tokens
  - More expressive but more expensive and error-prone
  - Features for context vector includes all levels of representation (like WHISK)



- Arrange all slot instances in a table

Inst 0:  $(f_1, w_{-k}), \dots (f_n, w_{-k}), (f_1, w_0), \dots (f_n, w_0), (f_1, w_k), \dots (f_n, w_k)$

Inst 1:  $(f_1, w_{-k}), \dots (f_n, w_{-k}), (f_1, w_0), \dots (f_n, w_0), (f_1, w_k), \dots (f_n, w_k)$

...

Inst m:  $(f_1, w_{-k}), \dots (f_n, w_{-k}), (f_1, w_0), \dots (f_n, w_0), (f_1, w_k), \dots (f_n, w_k)$

Look for most  $l$  frequent features (coverage)

Generate rules from these feature

Constrain rules until it meets Laplacian error threshold:

$$(\text{error}+1)/(\text{total}+1) > \text{epsilon}$$



# GRID Example

	-2	-1	0
Time			3:30 pm
Time		2: Add to constrain rule 1	2p.m.
Time			4 p.m. 1: All tagged as NP_Time by NER
Start		at 3. Add to constrain rule 1	10am
Begin		from 4. Add to constrain rule 1	11:30 AM

Text of 5 positive instances of seminar starting times



# Good Basic IE References

- Douglas E. Appelt and David Israel. 1999. Introduction to Information Extraction Technology. IJCAI 1999 Tutorial. <http://www.ai.sri.com/~appelt/ie-tutorial/>.
- Kushmerick, Weld, Doorenbos: [Wrapper Induction for Information Extraction](http://www.cs.ucd.ie/staff/nick/), IJCAI 1997. <http://www.cs.ucd.ie/staff/nick/>.
- \* Stephen Soderland: Learning Information Extraction Rules for Semi-Structured and Free Text. [Machine Learning 34](#)(1-3): 233-272 (1999)



# References

- Mary Elaine Califf and Raymond J. Mooney: *Relational Learning of Pattern-Match Rules for Information Extraction*. In AAAI 1999: 328-334.
- Fabio Ciravegna: *Adaptive Information Extraction from Text by Rule Induction and Generalisation*. In IJCAI 2001.
- Leek, T. R. 1997, *Information Extraction using Hidden Markov Models*, Master's thesis, UCSD
- D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997, *Nymble: a high-performance learning name-finder*. In *Proceedings of ANLP-97*, 194-201. [Also in MLJ 1999]
- Kristie Seymore, Andrew McCallum, Ronald Rosenfeld, 1999, *Learning Hidden Markov Model Structure for Information Extraction*, In *Proceedings of the AAAI-99 Workshop on ML for IE*.
- Dayne Freitag and Andrew McCallum, 2000, *Information Extraction with HMM Structures Learned by Stochastic Optimization*. AAAI-2000.
- John Lafferty, Andrew McCallum, Fernando Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. ICML 2001.