

# Linear Models for Structure Prediction

Fernando Pereira

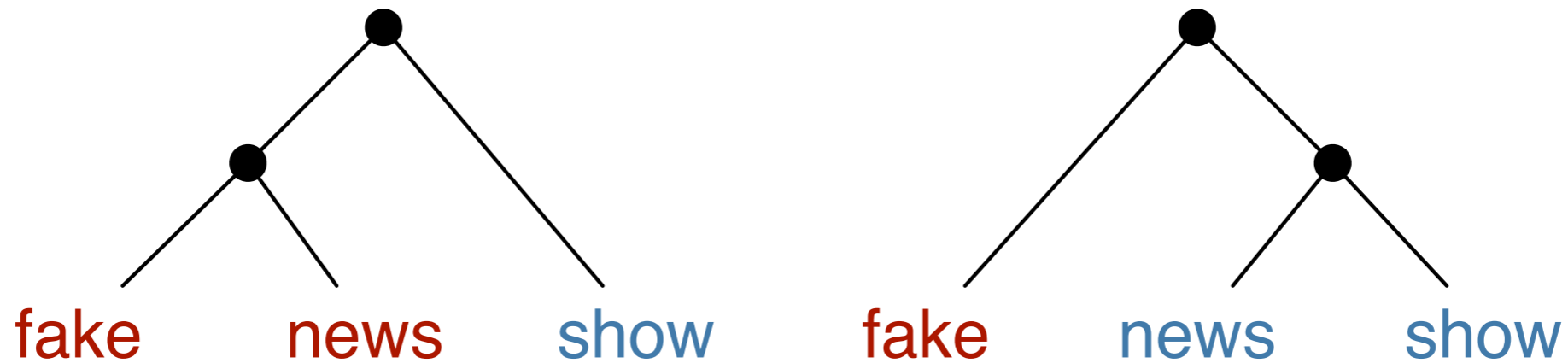
CIS 620 Intro

# Analyzing sequences

- Language
  - Syntactic analysis
  - Information extraction
- Biological sequences
  - Genes, regulatory regions
  - Secondary structure (folding)

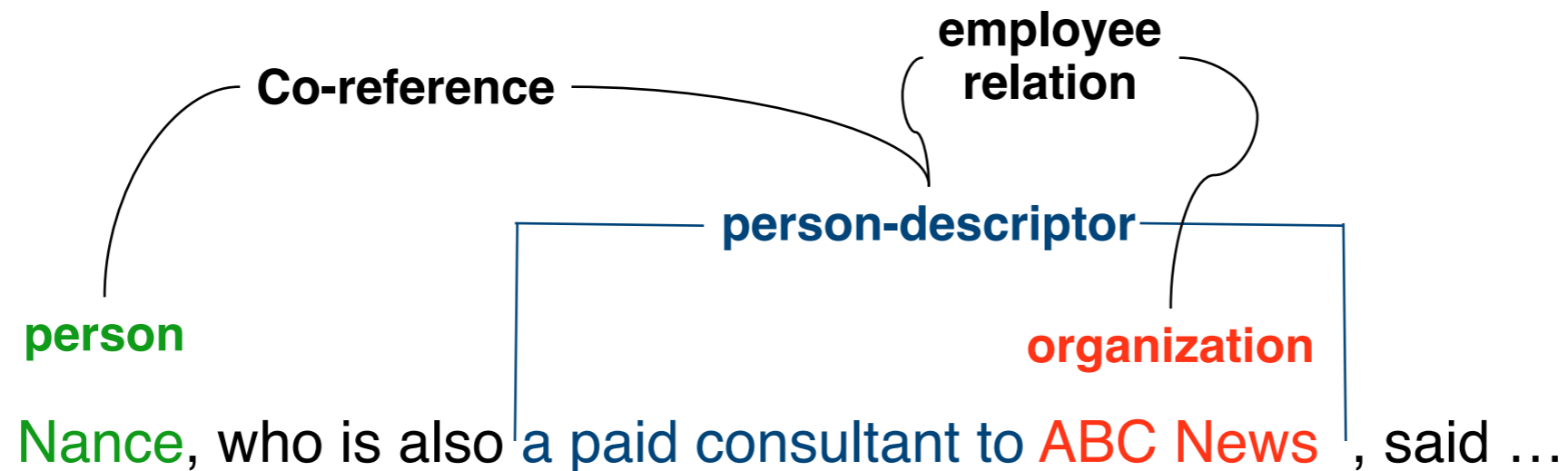
# Challenges

- Interacting decisions



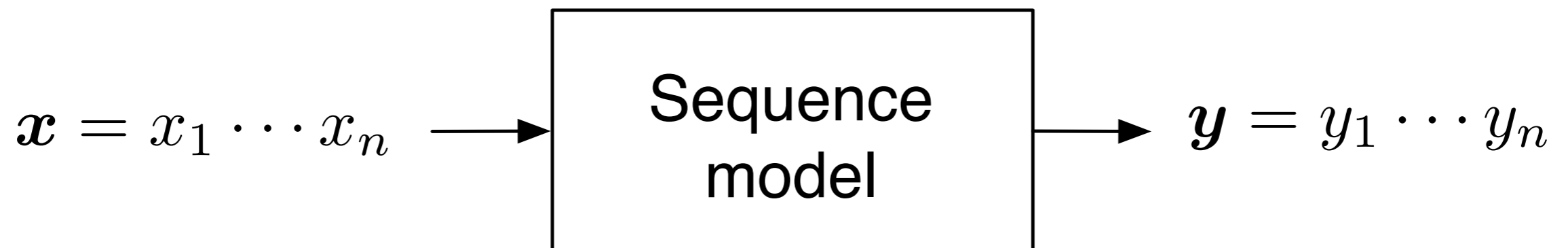
- A wide range of sequence features
- Computing an answer is relatively costly

# Information Extraction



- Information extraction as labeling:
  - Labels represent (parts of) entities
  - Relations
    - Cascaded labelers
    - Structured labels

# Analysis as labeling



- Labels give the role of corresponding inputs
  - Part-of-speech tagging
  - Shallow parsing
  - (Some) information extraction
  - Gene finding

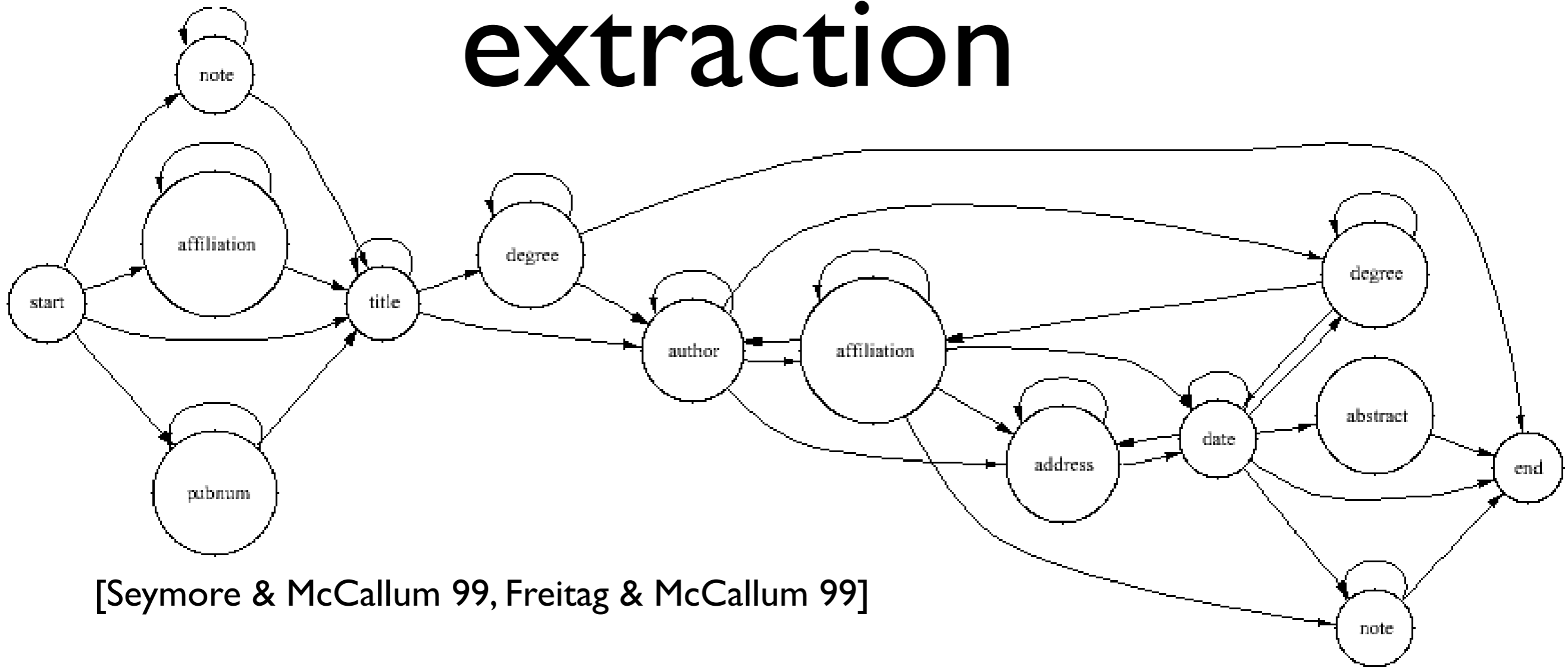
# Beyond Sequences

| <i>Vertex</i> | <i>Edge</i>  | <i>Label</i>          |
|---------------|--------------|-----------------------|
| Web page      | hyperlink    | page class            |
| pixel         | neighbor     | segment<br>label      |
| phrase        | constituency | syntactic<br>category |

# Previous approaches

- *Generative modeling*: probabilistic generators of sequence-structure pairs
  - HMMs, probabilistic context-free grammars
  - Hard to model overlapping features
- *Sequential classification*: decompose structure assignment into a sequence of structural decisions
  - Cannot trade-off decisions at different locations: *label-bias* problem

# HMMs in information extraction



- **Inputs**  $x$ : words

$$p(\mathbf{x}, \mathbf{y}) = \prod_i p(y_i | y_{i-1}) p(x_i | y_i)$$

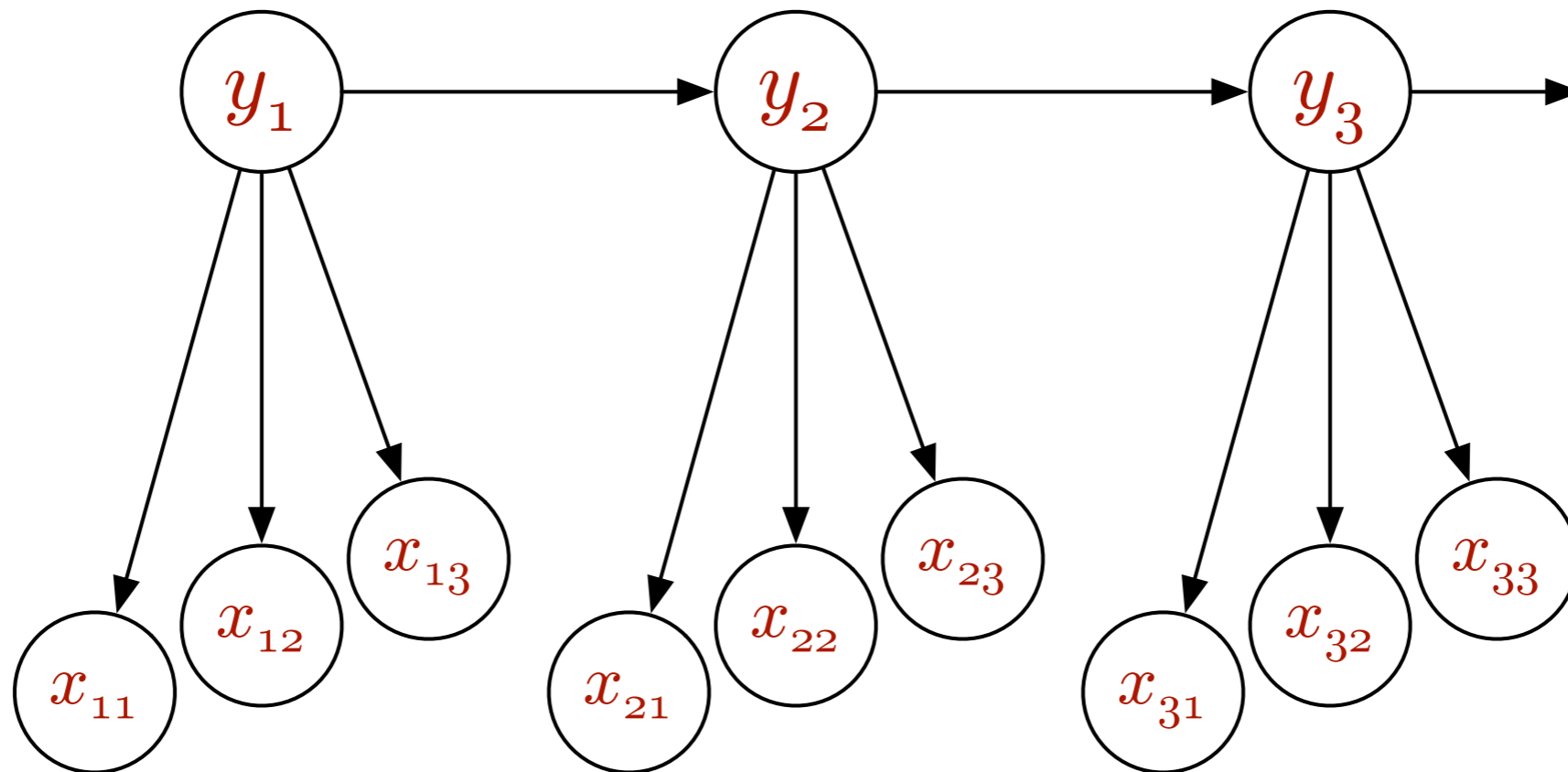
- **States**  $y$ : fields to extract

# Problems with HMMs

- Applications need richer input representation

| <i>Word features</i> | <i>Formatting features</i> |
|----------------------|----------------------------|
| word identity        | centered                   |
| capitalization       | indentation                |
| ends in “-tion”      | white space ratio          |
| word in word list    | begins with number         |
| word font            | ends with “?”              |

# Generating multiple features



- Relax conditional independence of features on labels  $\Rightarrow$  *intractability*

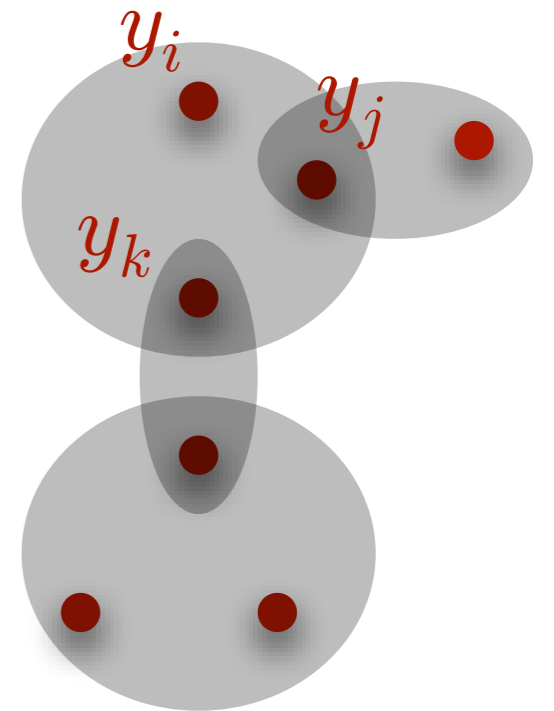
# Linear structure models

- Generalize linear classification (eg. perceptron)

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \lambda \cdot F(\mathbf{y}, \mathbf{x})$$

- Features based on local domains

$$F(\mathbf{y}, \mathbf{x}) = \sum_{C \in \mathcal{C}(\mathbf{x})} f_C(\mathbf{y}, \mathbf{x})$$
$$f_C(\mathbf{y}, \mathbf{x}) = f_C(\mathbf{y}_C, \mathbf{x})$$



# Why linear structure models

- Combine the best of generative and classification models:
  - Trade off labeling decisions at different vertices
  - Allow overlapping features
- Modular
  - factored scoring
  - loss function
- From features to kernels

# The two main questions

- What is a good linear scoring function
  - *Loss*: how to penalize bad labelings
  - *Regularization*: how to avoid being misled by non-representative training instances
- How to find a good labeling  $y$  given a linear scoring function: *inference*

# Aren't LSMs too specific?

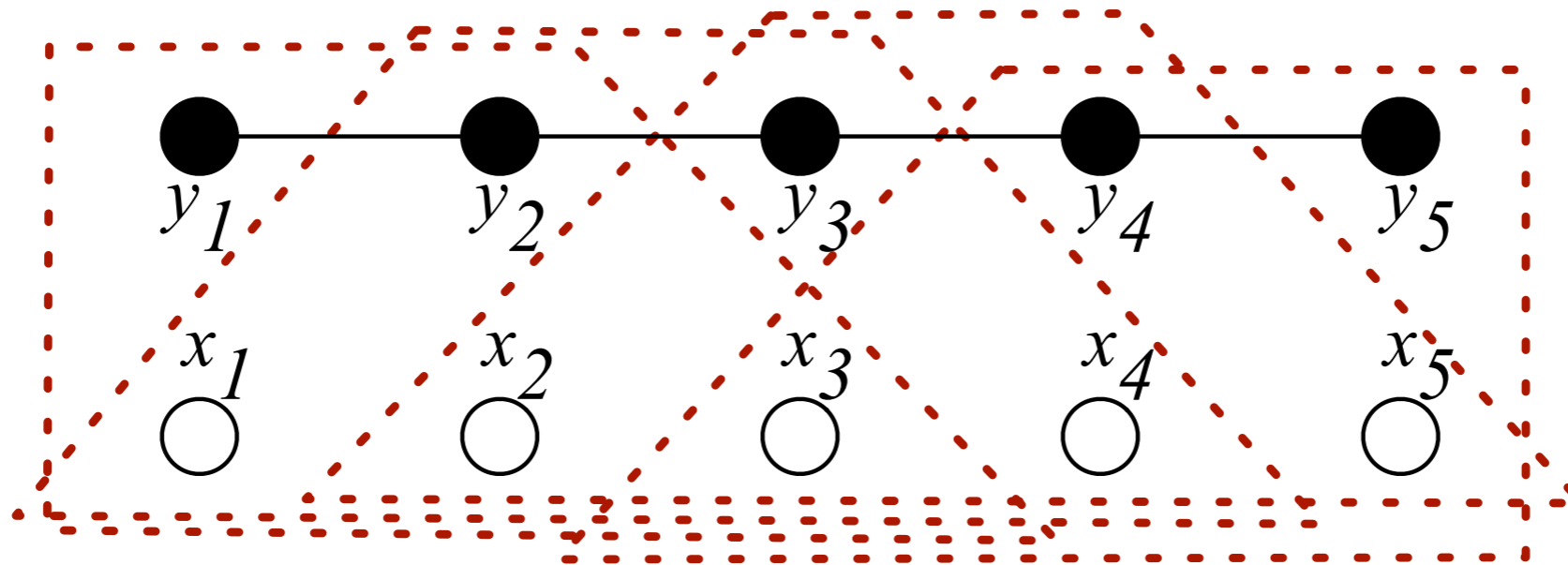
- Not really: the power is in
  - *structure*: graph of label interactions
  - *feature function  $F$* : how to measure a labeling (generalizes to kernels)
- They subsume
  - HMMs
  - probabilistic and weighted grammars (with some tweaks)
  - some kinds of random fields

# Overall goals

- Bring you to the edge of research in structure prediction problems
- Teach two basic ideas in machine learning and (probabilistic) inference
  - Large margin scoring for classification, ranking, and structure prediction
  - Undirected graphical models for inference over structured domains

# How it really started

- Sequence *conditional random fields*



$$\begin{aligned} p_{\lambda}(\mathbf{y}|\mathbf{x}) &= \frac{\exp \lambda \cdot F(\mathbf{y}, \mathbf{x})}{Z_{\lambda}(\mathbf{x})} \\ Z_{\lambda}(\mathbf{x}) &= \sum_{\mathbf{y}} \exp \lambda \cdot F(\mathbf{y}, \mathbf{x}) \\ F(\mathbf{y}, \mathbf{x}) &= \sum_i f_i(\mathbf{y}, \mathbf{x}) \\ f_i(\mathbf{y}, \mathbf{x}) &= f_i(y_{i-1}, y_i, \mathbf{x}) \end{aligned}$$

# Training CRFs

- Minimize data log-likelihood

$$\begin{aligned}\mathcal{L}_\lambda &= \sum_k \log p_\lambda(\mathbf{y}_k | \mathbf{x}_k) \\ &= \sum_k [\boldsymbol{\lambda} \cdot \mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - \log Z_\lambda(\mathbf{x}_k)]\end{aligned}$$

- ... by finding unique zero of gradient

$$\nabla \mathcal{L}_\lambda = \sum_k [\mathbf{F}(\mathbf{y}_k, \mathbf{x}_k) - E_{p_\lambda(\mathbf{Y} | \mathbf{x}_k)} \mathbf{F}(\mathbf{Y}, \mathbf{x}_k)]$$

- Dynamic programming for expectations  
(forward-backward algorithm)

# Noun-phrase chunking

(Sha)

Rockwell International Corp. 's Tulsa unit said

B I I I B I I O

it signed a tentative agreement extending its contract

B O B I I I O B I

with Boeing Co. to provide structural parts

O B I O O B I

for Boeing 's 747 jetliners

O B B I I

# Model details

- Second-order Markov  $y_i = c_{i-1}c_i$
- Predicates  $p(\mathbf{x}, i)$ 
  - word unigram, bigram
  - POS tag unigram, bigram, trigram
- Features  $y_i = y$   
 $y_i = y \ \& \ y_{i-1} = y'$   
 $c_i = c$   
 $y_i = y \ \& \ p(\mathbf{x}, i)$   
 $c_i = c \ \& \ p(\mathbf{x}, i)$

# Evaluation

- *Precision  $P$* : what proportion of predicted entities are correct
- *Recall  $R$* : what proportion of correct entities are predicted
- *$F_1$  measure*: 
$$\frac{2PR}{P + R}$$

# Test results

- F scores

| model           | F score |
|-----------------|---------|
| SVM combination | 0.9439  |
| CRF             | 0.9438  |
| MEMM            | 0.937   |

- 

- 

- Significance (McNemar's)

| hypothesis   | p-value |
|--------------|---------|
| CRF vs. SVM  | 0.469   |
| CRF vs. MEMM | 0.001   |

# Biomedical text

(McDonald)

- Gene/protein mentions:

In the absence of **MHC class II, purified soluble D10 TCR** bound to **Staphylococcus aureus enterotoxin C2** with an association rate of 1.

- Variation events: **type**, **location**, and **state change**

One ER showed a **G** to **T point mutation** in the **second position of codon 12**

# Gene/Protein Results

|        |                                      | Precision | Recall | F     |
|--------|--------------------------------------|-----------|--------|-------|
| AbGene |                                      | 0.63      | 0.65   | 0.64  |
| CRF    | words +<br>spelling                  | 0.83      | 0.773  | 0.801 |
|        | (non-)gene tokens<br>+ rare trigrams | 0.864     | 0.787  | 0.824 |

- Exact match
- AbGene: Brill-style POS and gene tagger, post-processor

# Variation Results

|          | Precision | Recall | F    |
|----------|-----------|--------|------|
| Type     | 0.80      | 0.72   | 0.76 |
| Location | 0.85      | 0.73   | 0.79 |
| State    | 0.90      | 0.80   | 0.85 |

# Technical challenges

- Very large number of features:
  - 820,000 at least once on training set
  - 3,800,000 predicates true at least once
- *Overfitting*: Gaussian prior on weights loses effect near convergence
  - Proposed solutions
    - Feature induction (McCallum 03)
    - Maximum margin (Tasker et al. 03)
  - Expensive search/optimization

# Online algorithms

- Process one training instance at a time
- Very simple
- Predictable runtime, small memory
- Adaptable to different loss functions
- Basic idea:  $w = 0$ 
  - for  $t = 1, \dots, T$  :
    - for  $i = 1, \dots, N$  :
      - classify  $x_i$  incurring loss  $l$
      - adjust  $w$  to reduce  $l$

# Loss functions

- Measures of error

Bill Clinton visited the Redmond-based software giant Microsoft Corp.

|     |     |   |   |     |   |   |   |     |     |
|-----|-----|---|---|-----|---|---|---|-----|-----|
| per | per | - | - | loc | - | - | - | org | org |
| per | per | - | - | org | - | - | - | org | -   |

- 0-1 loss = 1 (not completely correct)
- labeling loss = 2 (differs by two labels)

# Generalized perceptron

- Based by Collins on voted perceptron:

$$\mathbf{w}_0 = \mathbf{0}; k = 0$$

for  $t = 1, \dots, T$  :

for  $i = 1, \dots, N$  :

$$\mathbf{y}^* = \arg \max_{\mathbf{y}} \mathbf{w}_k \cdot \mathbf{F}(\mathbf{y}, \mathbf{x}_i)$$

if  $\mathbf{y}^* \neq \mathbf{y}_i$  :

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{F}(\mathbf{y}_i, \mathbf{x}_i) - \mathbf{F}(\mathbf{y}^*, \mathbf{x}_i)$$

else :

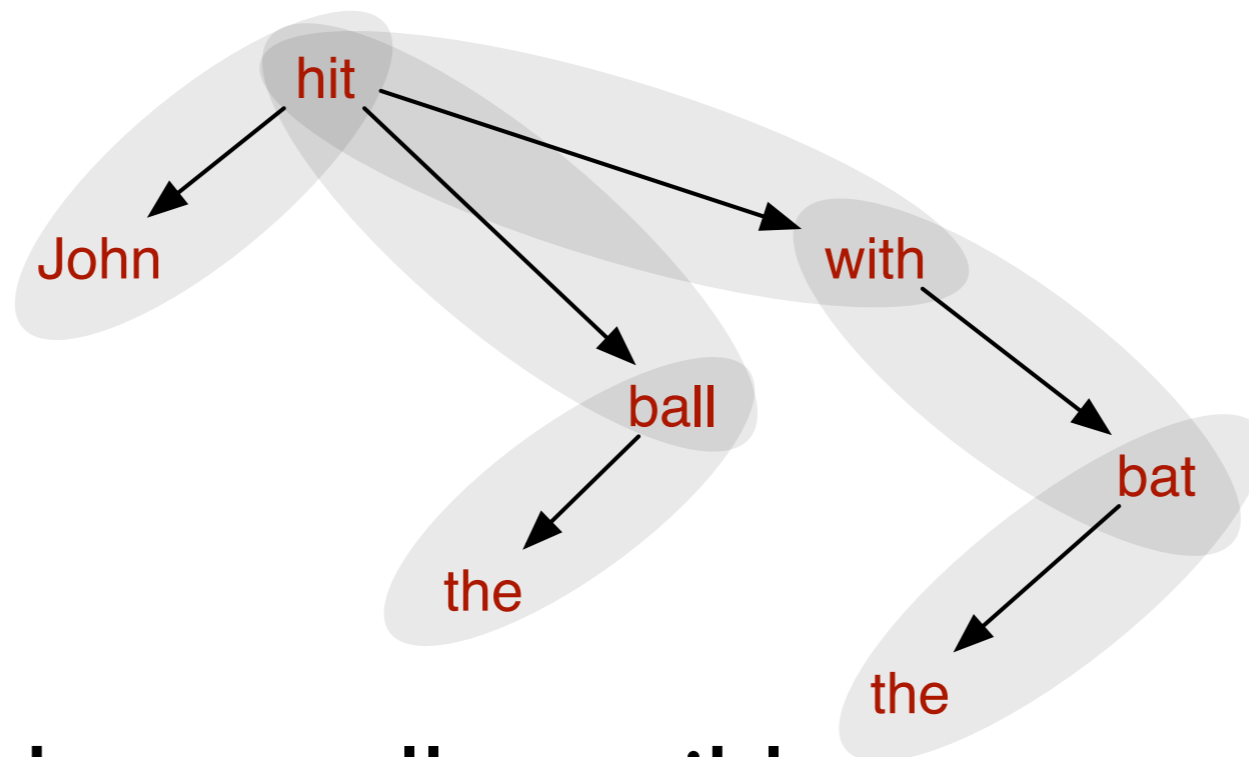
$$\mathbf{w}_{k+1} = \mathbf{w}_k$$

$$k \leftarrow k + 1$$

$$\lambda = \sum_k \mathbf{w}_k / NT$$

# Dependency parsing

- Meaningful relationships between words
- Local domains over head-dependent pairs



- Search over all possible structures in cubic time (Eisner, Satta)

# Method comparisons

## *NP chunking*

| algorithm        | F score |
|------------------|---------|
| Perceptron       | 0.935   |
| Voted Perceptron | 0.94    |
| MIRA             | 0.942   |
| CRF              | 0.943   |

## *dependency parsing*

## *named entities*

| algorithm        | F score |
|------------------|---------|
| Perceptron       | 0.794   |
| Voted Perceptron | 0.819   |
| MIRA             | 0.827   |
| CRF              | 0.831   |

| algorithm         | accuracy |
|-------------------|----------|
| Perceptron        | 0.875    |
| Voted Perceptron  | 0.901    |
| MIRA              | 0.904    |
| SVM (seq. class.) | 0.903    |