Machine Learning Department
School of Computer Science
Carnegie Mellon University

Course Overview

+

What is Structured Prediction?

Matt Gormley
Lecture 1
Aug. 29, 2022

WHAT IS STRUCTURED PREDICTION?

Structured Prediction

- The focus of most Intro ML courses is **classification**
  – Given observations: $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$
  – Predict a (binary) **label:** y
- Many real-world problems require **structured prediction**
  – Given observations: $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$
  – Predict a **structure:** $\boldsymbol{y} = (y_1, y_2, \ldots, y_J)$
- Some *classification* problems benefit from **latent structure**

Structured Prediction

Classification / Regression

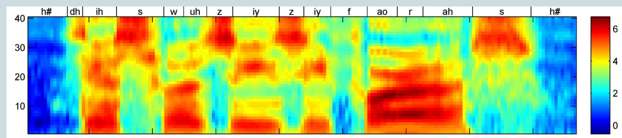
1. Input can be semi-structured data
2. Output is a single number (integer / real)
3. In linear models, features can be arbitrary combinations of [input, output] pair
4. Output space is small
5. Inference is trivial

Structured Prediction

1. Input can be semi-structured data
2. Output is a sequence of numbers representing a structure
3. In linear models, features can be arbitrary combinations of [input, output] pair
4. Output space may be exponentially large in the input space
5. Inference problems are NP-hard or #P-hard in general and often require approximations

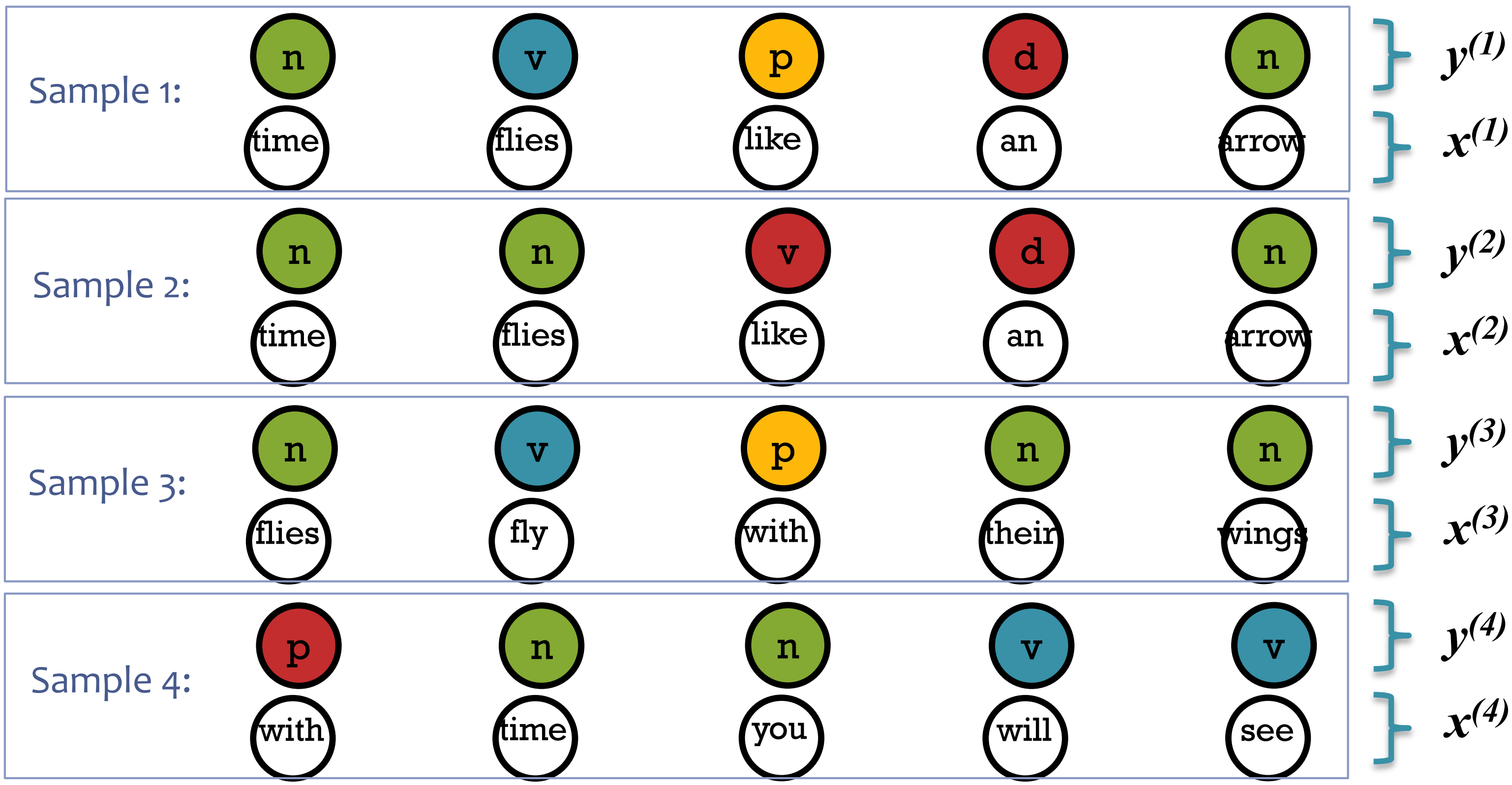
Structured Prediction Examples

[Human Language Technologies]

Task	Input	Output
Speech Recognition		*This was easy for us.*
Syntactic Parsing	time flies like an arrow	time flies like an arrow
Semantic Parsing	`Send a text to Alice that I'll be late`	`txt(recipient = Alice, msg = "I'll be late")`
Machine Translation	WHERE IS THE TRAIN STATION?	¿DONDE ESTA LA ESTACION DE TRENES?

Structured Prediction Training Dataset: Part-of-Speech (POS) Tagging

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



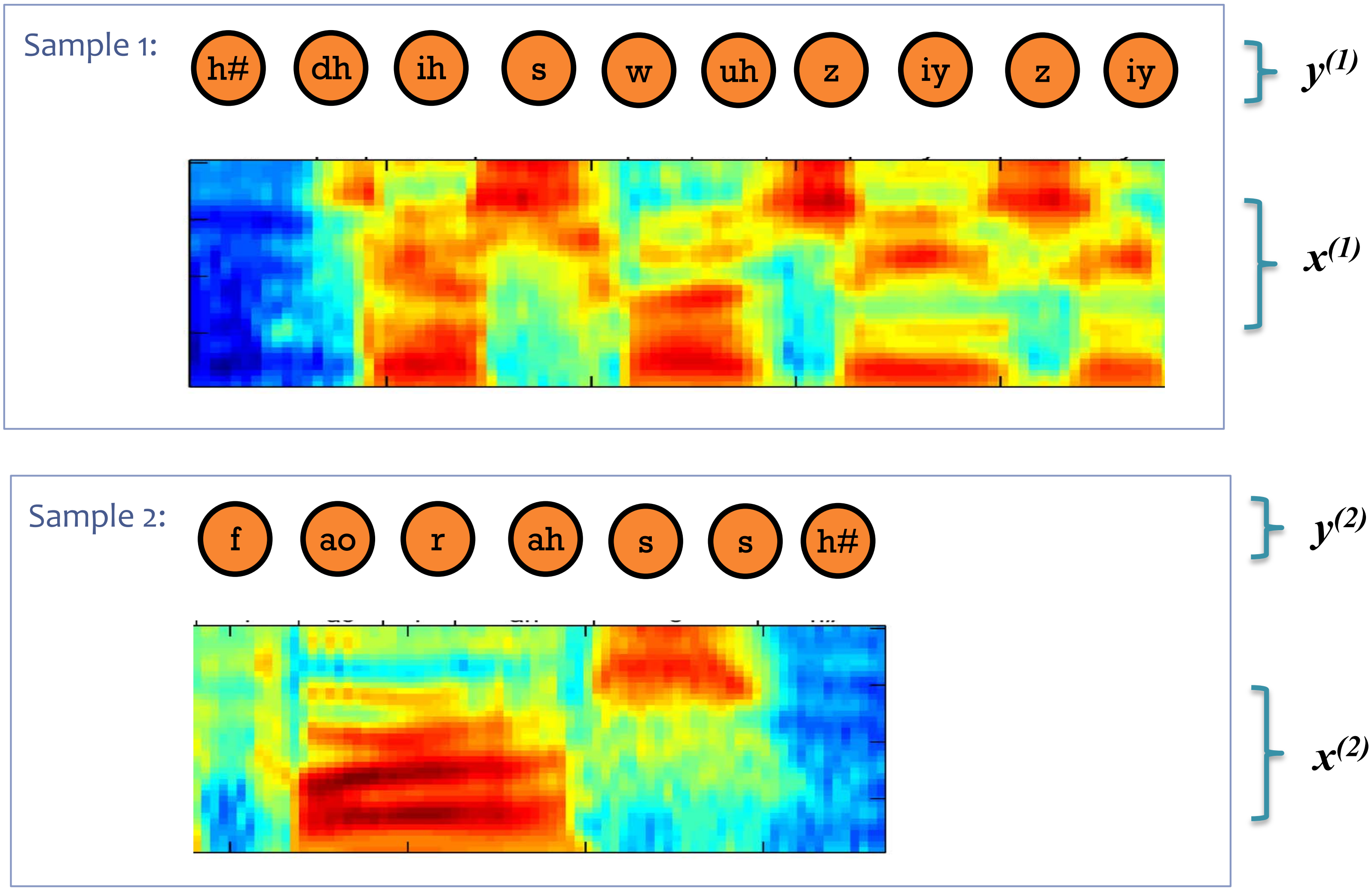
Structured Prediction Training Dataset: Handwriting Recognition

Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$

Figures from (Chatzis & Demiris, 2013)

Structured Prediction Training Dataset: Phoneme (Speech) Recognition

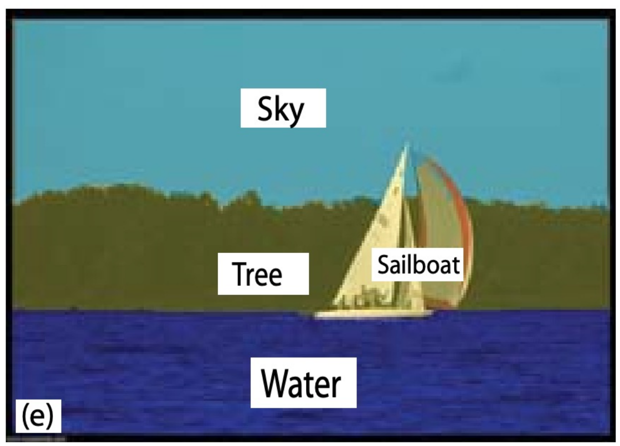
Data: $\mathcal{D} = \{\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}\}_{n=1}^{N}$



Figures from (Jansen & Niyogi, 2013)

Structured Prediction Training Dataset: Scene Understanding

$x^{(1)}$

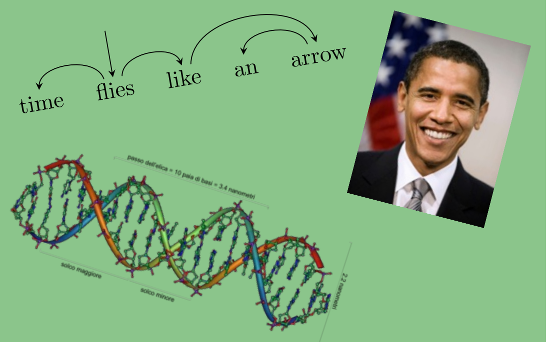


$y^{(1)}$

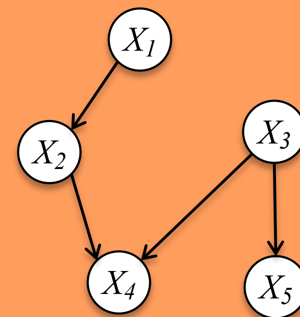
(Li et al., 2009)

Structured Prediction

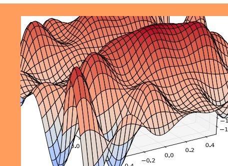
Data



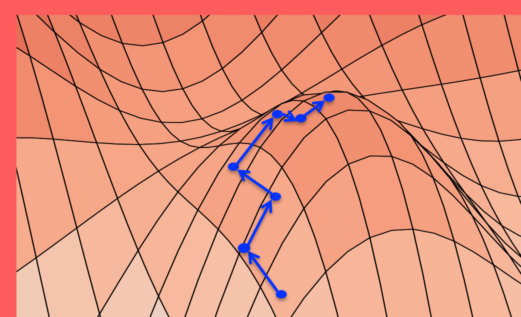
Model



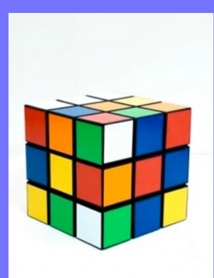
Objective



Learning



Inference



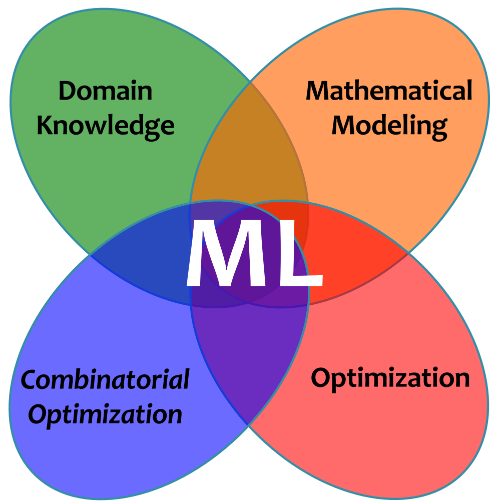
(**Inference** is usually called as a subroutine in learning)

Structured Prediction

The **data** inspires the structures we want to predict

Our **model** defines a score for each structure

It also tells us what to optimize



Learning tunes the parameters of the model

Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

DECOMPOSING A STRUCTURE INTO PARTS

Decomposing a Structure into Parts

- Many real-world problems require **structured prediction**
  - Given observations: $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$
  - Predict a **structure:** $\boldsymbol{y} = (y_1, y_2, \ldots, y_J)$

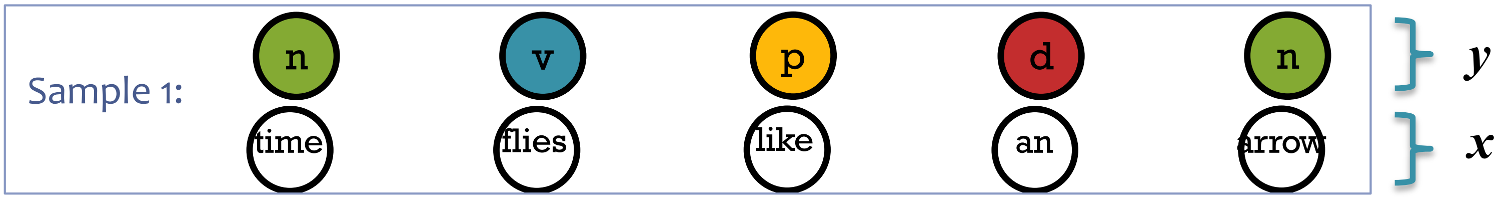
- The most important idea in structured prediction:
  - Do NOT treat the output structure $\boldsymbol{y}$ as a single monolithic piece of data
  - Instead, divide that structure into its pieces

Decomposing a Structure into Parts

- Why divide a **structure** into its **pieces**?
  - amenable to **efficient inference**
  - enable natural **parameter sharing** during learning
  - easier definition of fine-grained **loss functions**
  - clearer depiction of **model's uncertainty**
  - easier specification of **interactions** between the parts
  - (may) lead to natural definition of a **search problem**
- A key step in **formulating a task as a structured prediction**

Decomposing a Structure into Parts

Example 1: Part-of-speech Tagging



Question:
How would you decompose the structure y into parts?

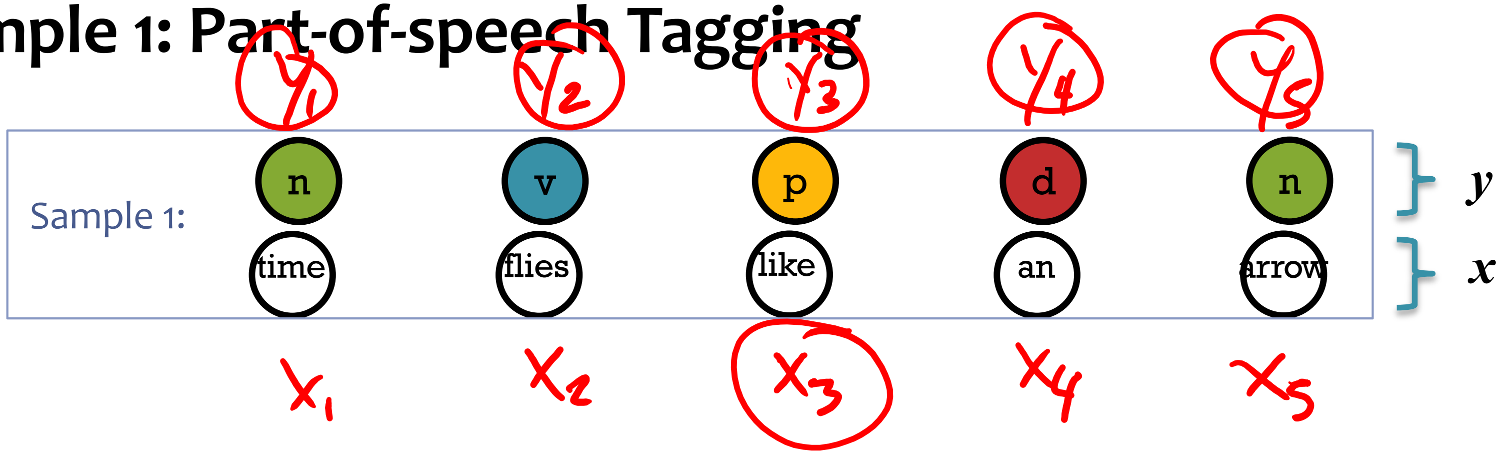
A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

Decomposing a Structure into Parts

Example 1: Part-of-speech Tagging



Question:

How would you decompose the structure y into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

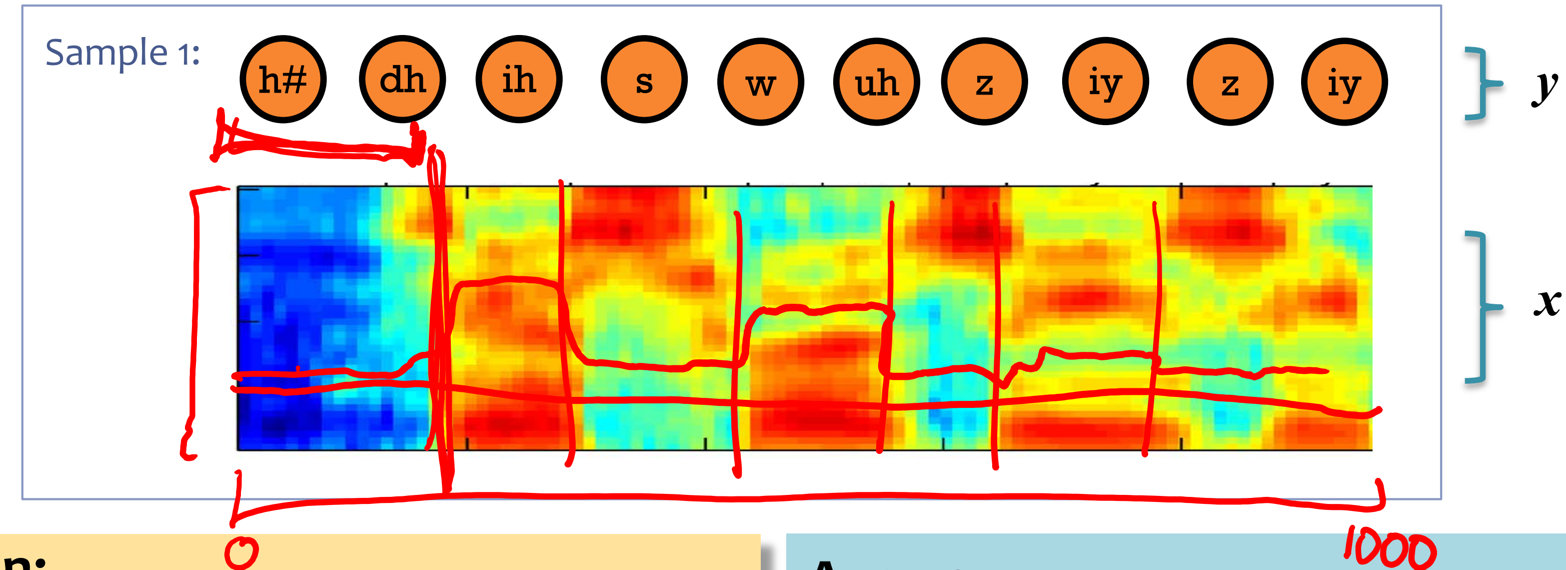
Answer:

A. For each word in the sentence create one tag variable, e.g. the t'th word x_t has a tag variable y_t.

B. Each tag variable y_t ranges over the set of possible part-of-speech tags {a, d, n, p, v, … }

Decomposing a Structure into Parts

Example 2: Phoneme Recognition



Question:

How would you decompose the structure y into parts?

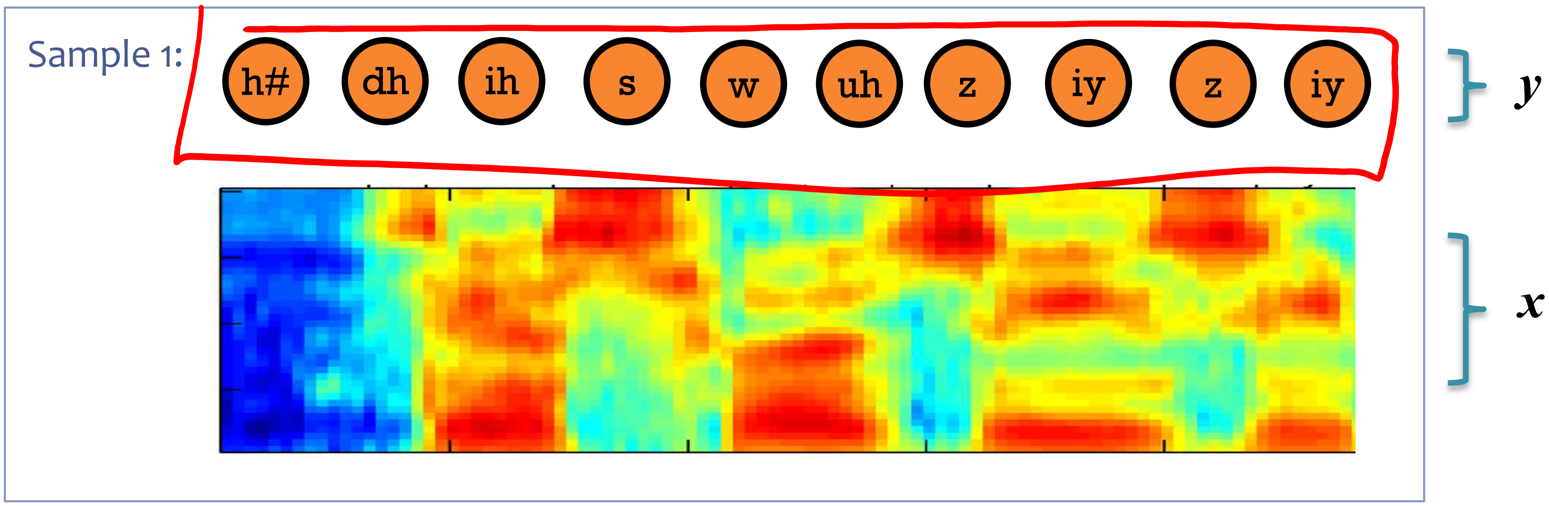
A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

Decomposing a Structure into Parts

Example 2: Phoneme Recognition



Question:

How would you decompose the structure $\boldsymbol{y}$ into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

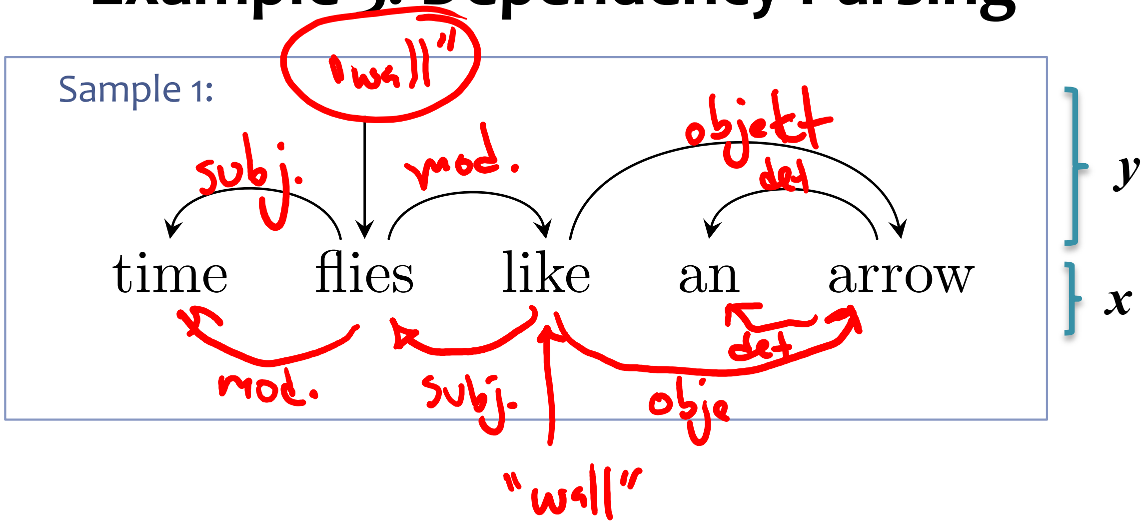
Answer:

A. Assume the speech signal consists of T segments of 10 milliseconds each, then create T phoneme variables $y_1, y_2, \ldots, y_T$

B. Each phoneme variable y_t can be a phoneme {dh, h#, ih, iy, … } or the special symbol "—" meaning "no phoneme"

Decomposing a Structure into Parts

Example 3: Dependency Parsing



Definition of a Dependency Parse:

1. Each word must have exactly one parent
2. The parent must be another word in the sentence OR the "wall"
3. Exactly one word must have the "wall" as its parent
4. The resulting directed graph must be acyclic

Question:

How would you decompose the structure y into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

g-Parent:	0	-1	0	3	2
parent:	2	0	2	5	3
word:	time	flies	like	an	arrow
idx:	1	2	3	4	5

Decomposing a Structure into Parts

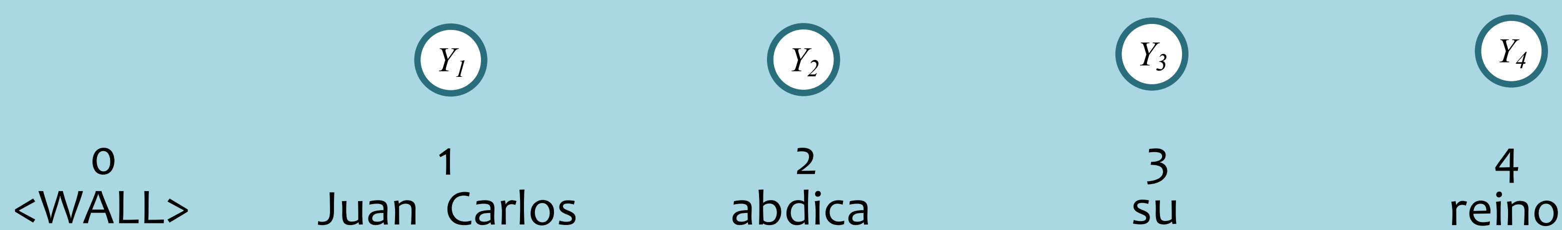
Example 3: Dependency Parsing

Answer:

Solution #1: (most obvious solution)

A. Have one variable for each word in the sentence

B. Each variable can take on an integer indicating which word is its parent



Decomposing a Structure into Parts

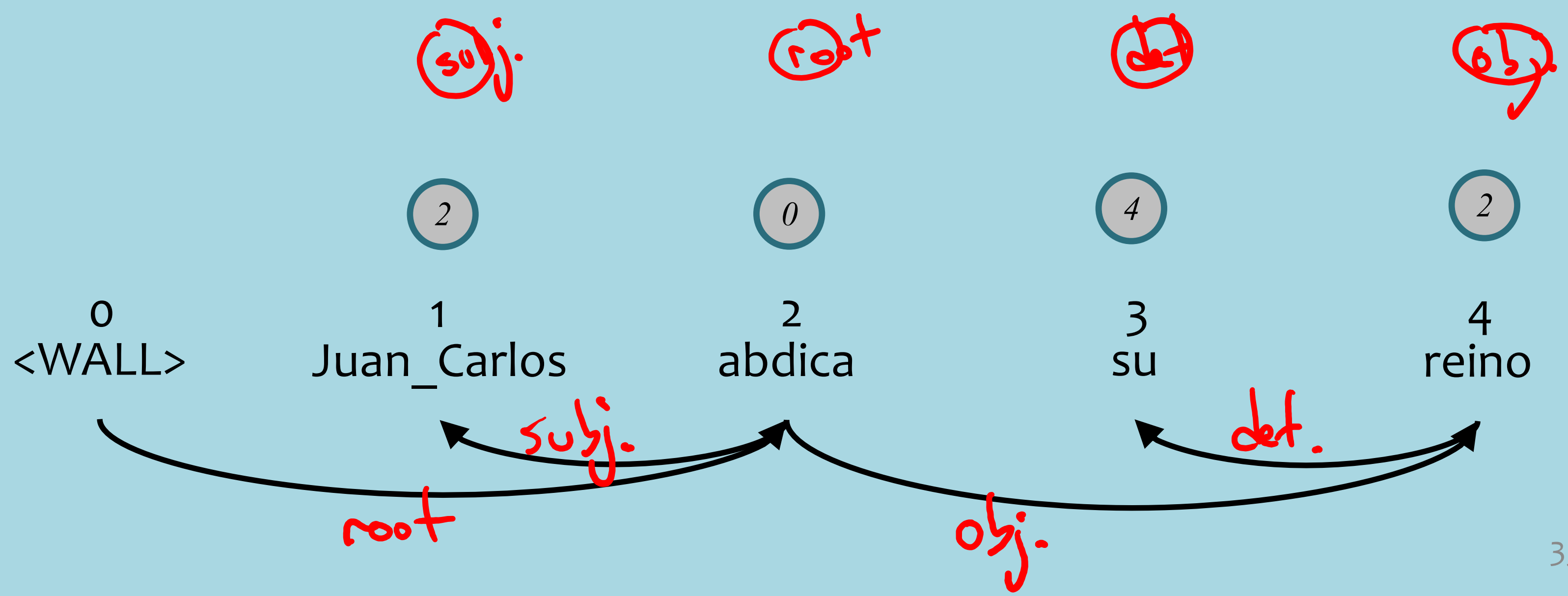
Example 3: Dependency Parsing

Answer:

Solution #1: (most obvious solution)

A. Have one variable for each word in the sentence

B. Each variable can take on an integer indicating which word is its parent



Decomposing a Structure into Parts

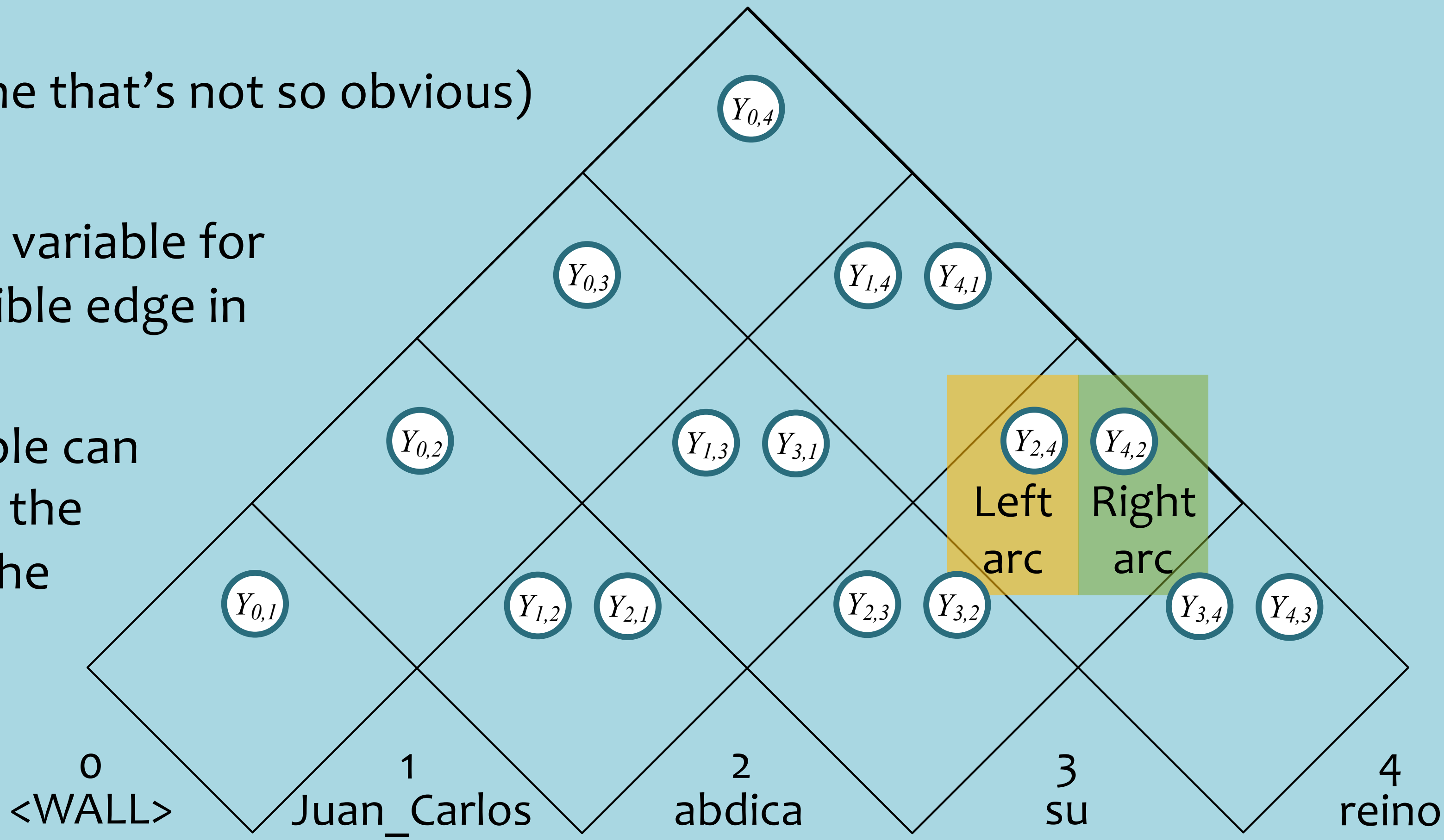
Example 3: Dependency Parsing

Answer:

Solution #2: (one that's not so obvious)

A. Create one variable for every possible edge in the graph

B. Each variable can take either the value 1 (if the edge is present) or 0 (if the edge is not present)



Decomposing a Structure into Parts

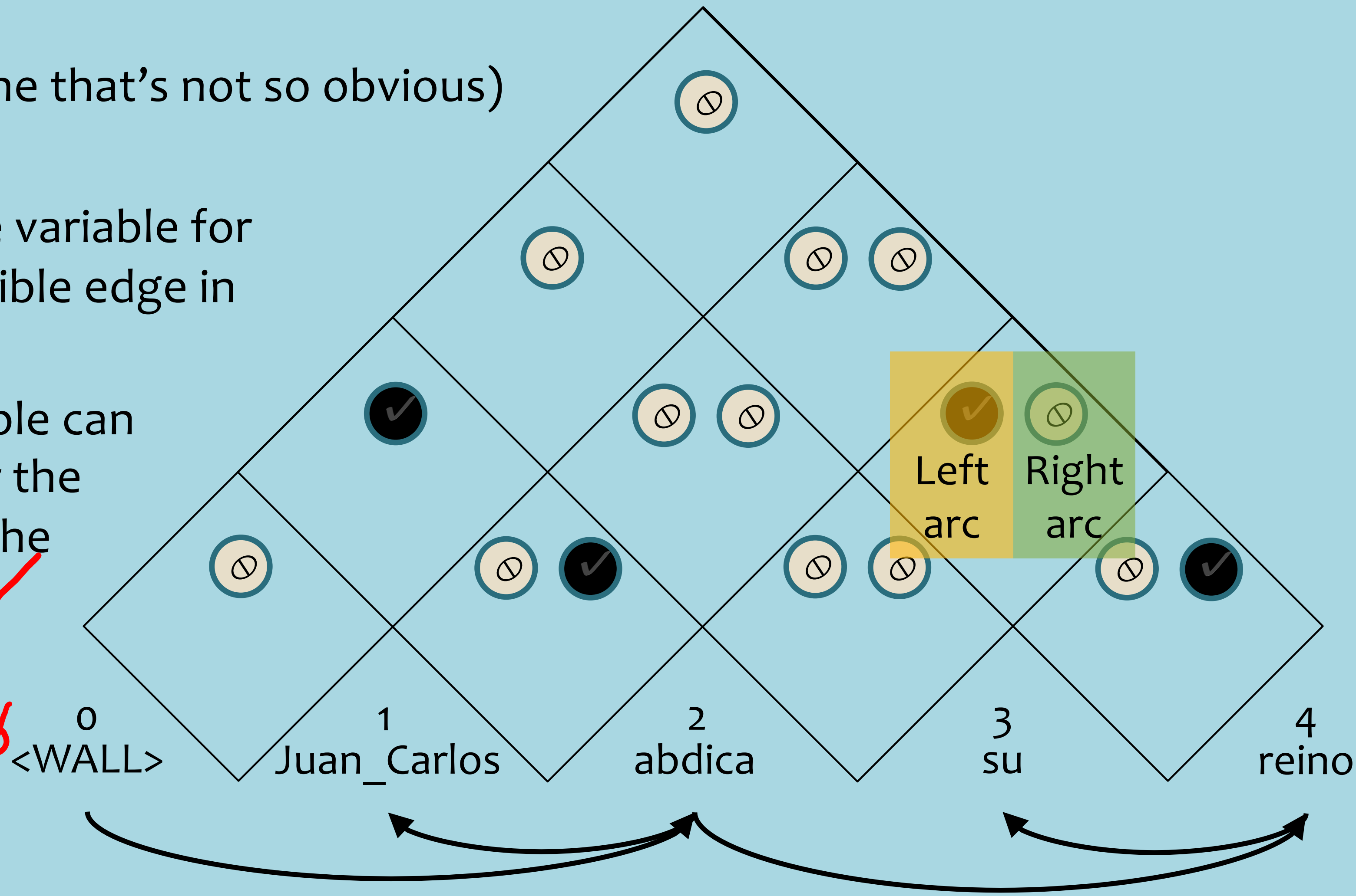
Example 3: Dependency Parsing

Answer:

Solution #2: (one that's not so obvious)

A. Create one variable for every possible edge in the graph

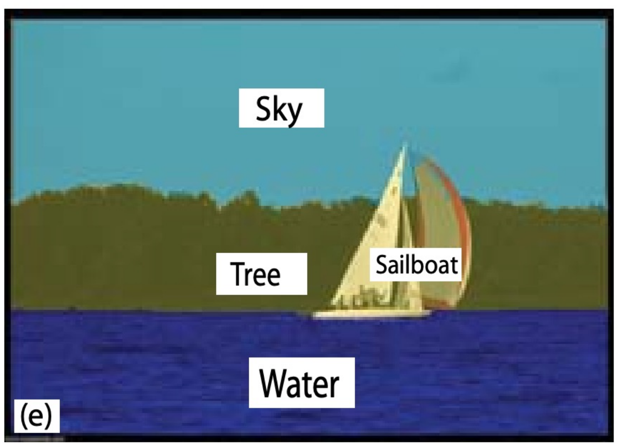
B. Each variable can take either the value 1 (if the edge is present) = ✓ or 0 (if the edge is not present) = ∅



Decomposing a Structure into Parts

Example 4: Scene Understanding

x



y

Question:

How would you decompose the structure y into parts?

A. How many variables would you need to represent said decomposition?

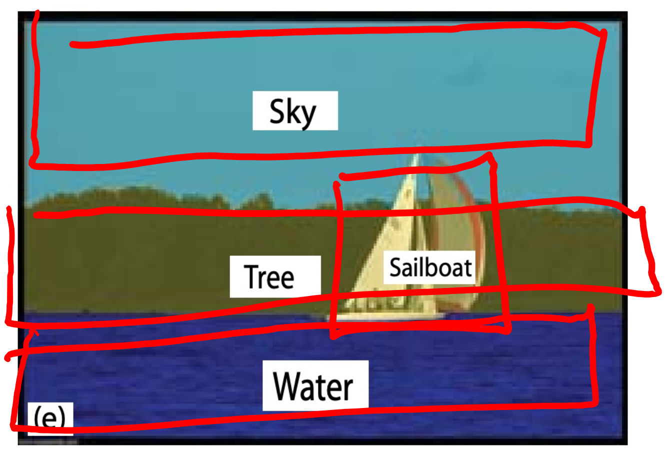
B. What values could each variable take?

Answer:

Decomposing a Structure into Parts

Example 4: Scene Understanding

x



y

y_{sky} y_{tree} y_{sail} y_{water}

Question:

How would you decompose the structure $\boldsymbol{y}$ into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

A. One output variable $y_{i,j}$ for each of pixel $x_{i,j}$

B. The value of each $y_{i,j}$ would be one of the possible labels, e.g. {sailboat, sky, tree, water, mountain, … }

Decomposing a Structure into Parts

Example 5: Medical Diagnosis

patient's diagnosis } y

patient's chart } x

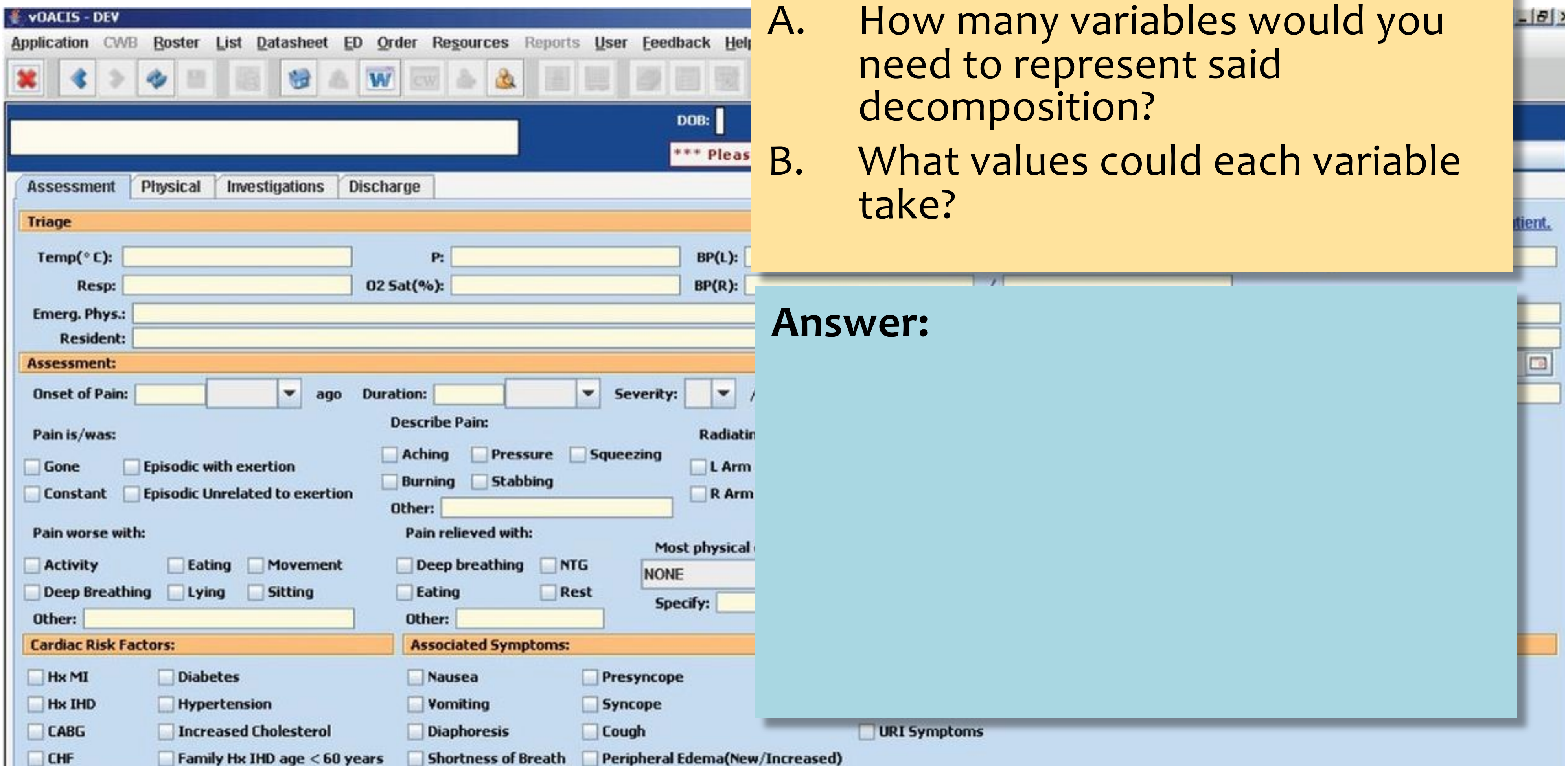
Question:

How would you decompose the structure y into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

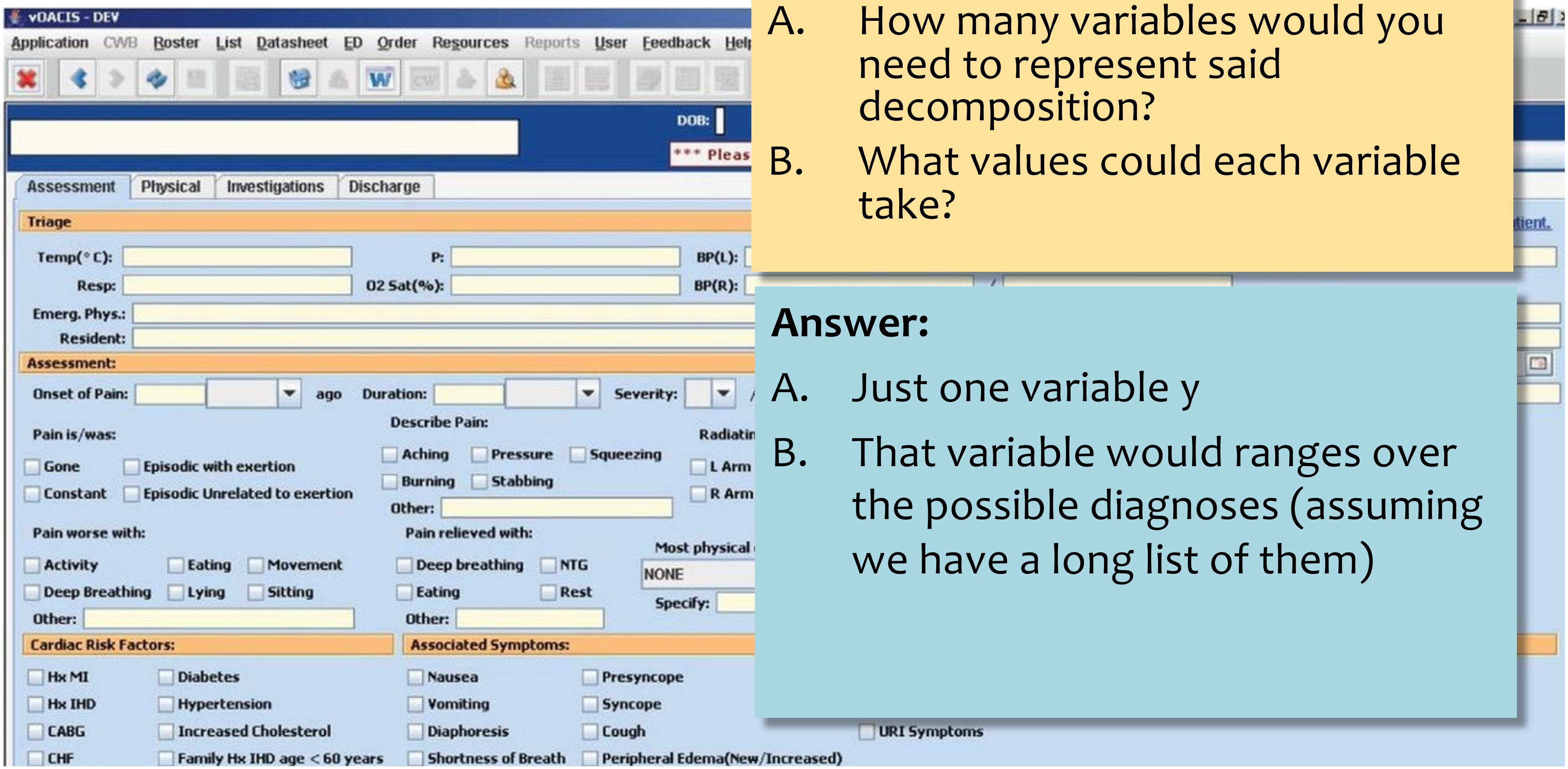


Decomposing a Structure into Parts

Example 5: Medical Diagnosis

patient's diagnosis } y

patient's chart } x



Question:

How would you decompose the structure y into parts?

A. How many variables would you need to represent said decomposition?

B. What values could each variable take?

Answer:

A. Just one variable y

B. That variable would ranges over the possible diagnoses (assuming we have a long list of them)

Decomposing a Structure into Parts

Takeaways from these examples

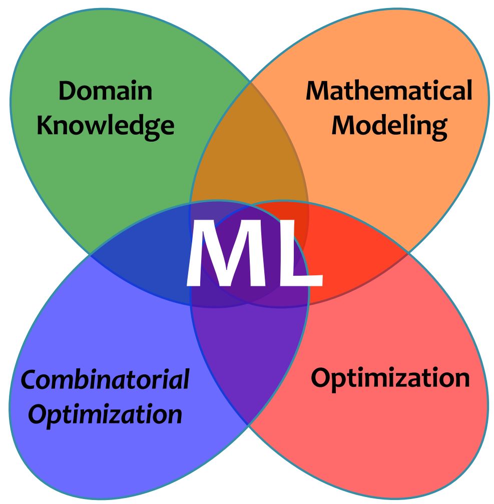
1. The structure often provides an obvious decomposition (e.g. POS tagging)
2. Dealing with variable size structures can be tricky (e.g. phoneme recognition)
3. There are often many ways to decomposition the structure (e.g. dependency parsing)
4. Sometimes the less obvious decomposition may be the ”simpler” one (e.g. scene understanding)
5. Don’t confuse structure in the input for structure in the output (e.g. medical diagnosis)

Structured Prediction

The **data** inspires the structures we want to predict

Our **model** defines a score for each structure

It also tells us what to optimize



Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

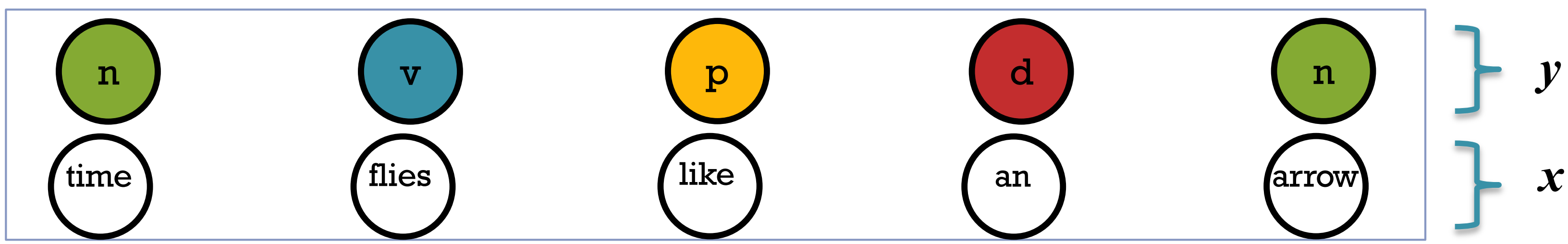
Learning tunes the parameters of the model

(without any math!)

WHAT IS A MODEL?

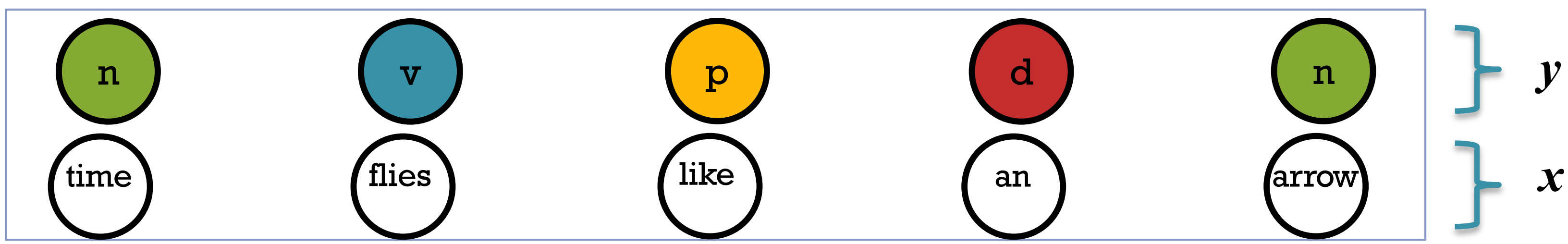
A Not-very-interesting Model

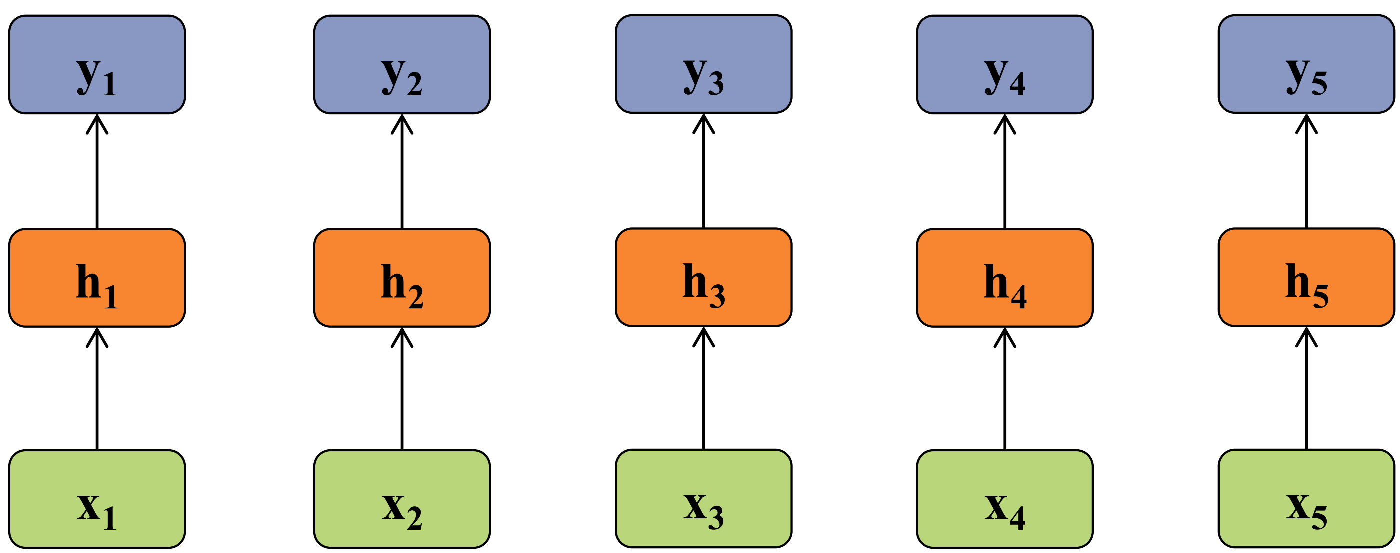
Question: How could we apply a standard feed-forward neural network (MLP) that expects a **fixed size input/output** to a prediction task with **variable length input/output**?



A Not-very-interesting Model

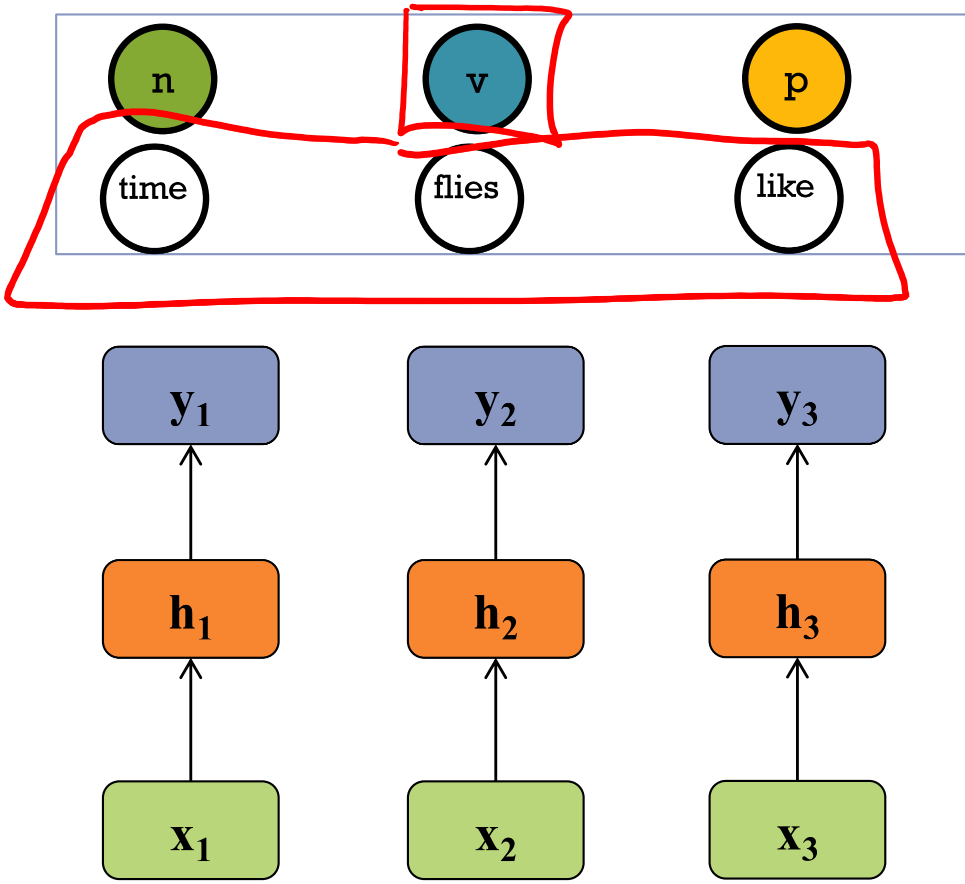
Question: How could we apply a standard feed-forward neural network (MLP) that expects a **fixed size input/output** to a prediction task with **variable length input/output**?





A Not-very-interesting Model

Question: How could we apply a standard feed-forward neural network (MLP) that expects a **fixed size input/output** to a prediction task with **variable length input/output**?



Q: *Why is this model not-very-interesting?*

A: Because it only considers the interaction between the current word x_t and the current tag y_t

In other words, it makes an independent classification decision for each tag.

For part-of-speech tagging, we know that verbs are much more likely to follow nouns. But this model CANNOT learn that.

Joint Modeling

After we come up with a way to decompose our structure into variables, what comes next?

- We can define a **joint model** over those variables
- The joint model defines a **score for each possible structure** allowed by our decomposition
- The model should give high scores to "good" structures and low scores to "bad" structures
  - in probability terms: **high scores for likely structures** and **low scores for unlikely structures**
  - "likely structures" could be defined as those appearing in your **training dataset**
- (Hopefully, the joint model is also able to capture interesting interactions between pairs, triples, quadruples, … of variables)

How do we write down a joint model?

(Factor Graphs)

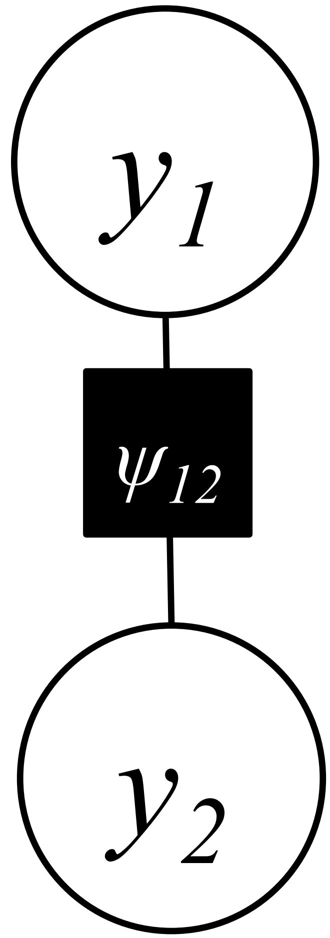
Mathematical Modeling

An Abstraction for Modeling

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)



Mathematical Modeling

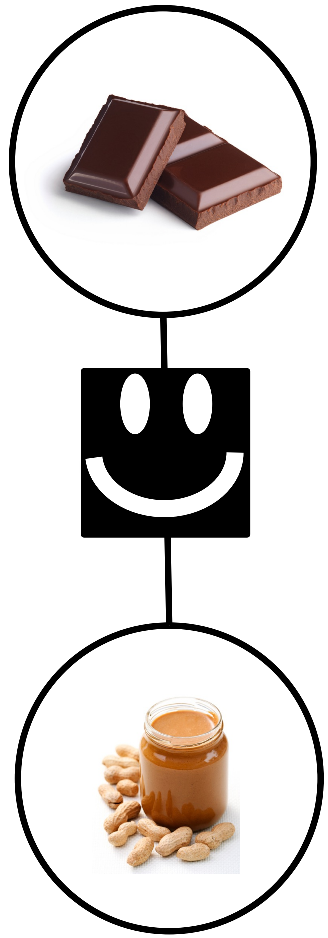
An Abstraction for Modeling

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions



Mathematical Modeling

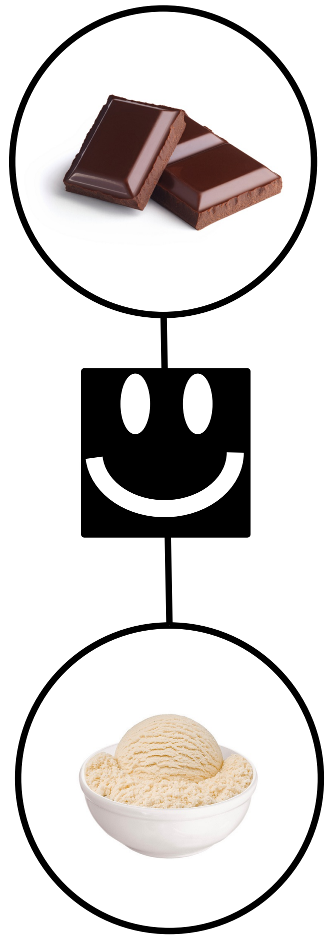
An Abstraction for Modeling

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions



Mathematical Modeling

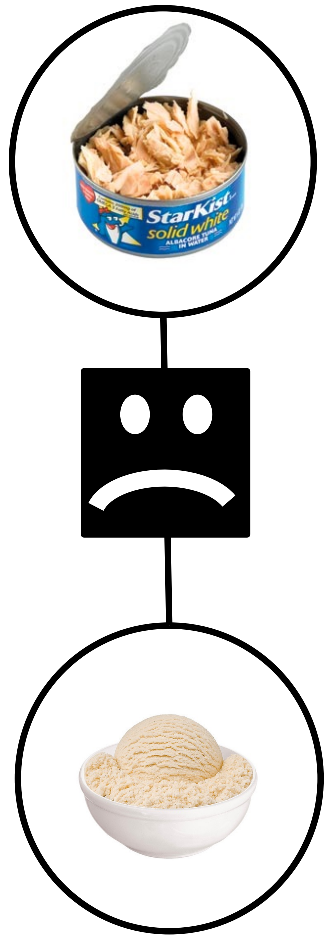
An Abstraction for Modeling

Factor Graph

(bipartite graph)

- variables (circles)
- factors (squares)

Factors have local opinions

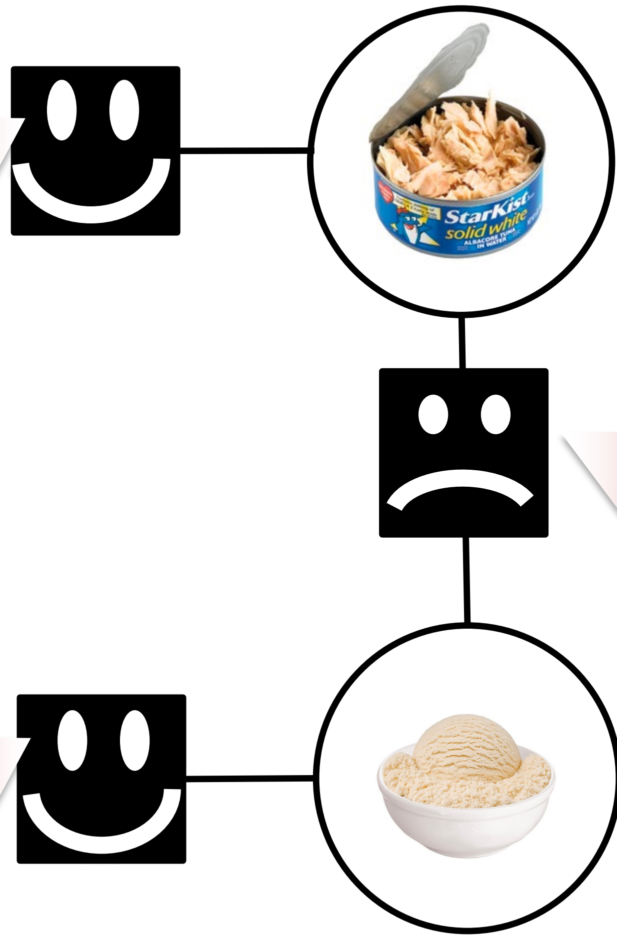


Mathematical Modeling

An Abstraction for Modeling

$$P(\text{tuna, ice cream}) \;=\; ?$$

chocolate	0.1
peanut butter	5
ice cream	1
tuna	6
…	



	chocolate	peanut butter	ice cream	tuna	⋮
chocolate	2	9	7	0.1	
peanut butter	4	2	3	0.2	
ice cream	7	3	2	0.1	
tuna	0.1	0.2	0.1	2	
…					

chocolate	4
peanut butter	8
ice cream	7
tuna	3
…	

Mathematical Modeling

An Abstraction for Modeling

$$P(\text{tuna, ice cream}) = \frac{1}{Z}(6 * 7 * 0.1)$$

chocolate	0.1
peanut butter	5
ice cream	1
tuna	6
...	

Uh-oh! The probabilities of the various assignments sum up to $Z > 1$.

So divide them all by Z.

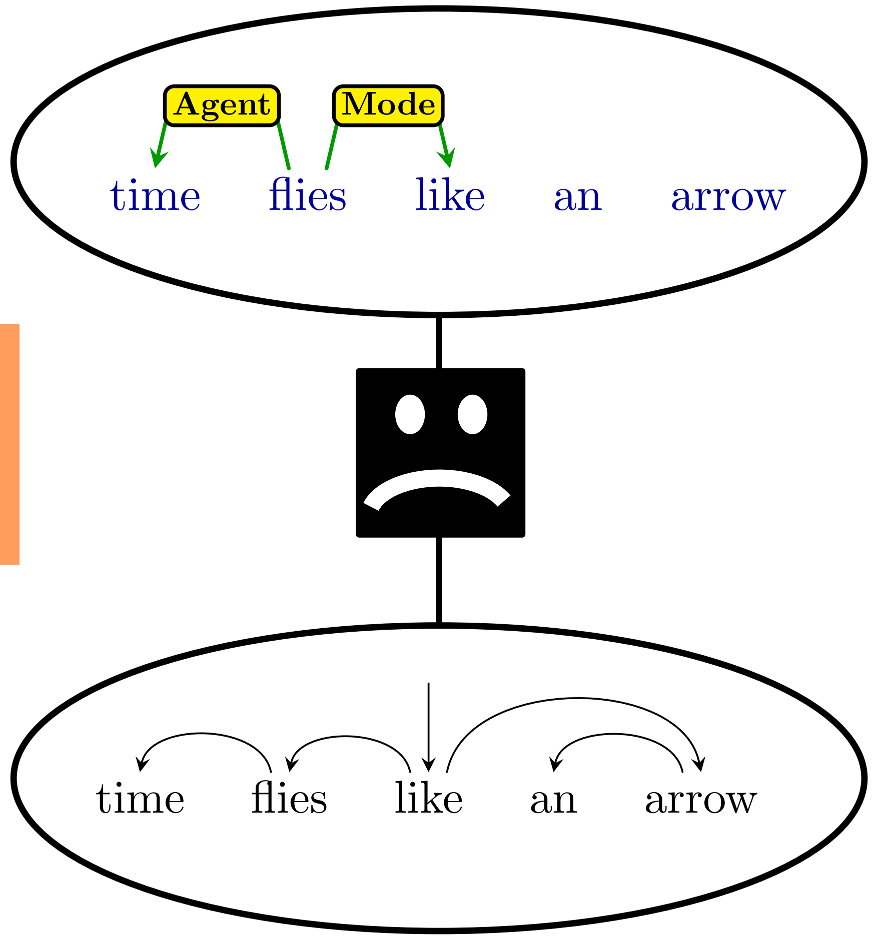
	cho	peanu	ice c	tuna	⋮
chocolate	2	9	7	0.1	
peanut butter	4	2	3	0.2	
ice cream	7	3	2	0.1	
tuna	0.1	0.2	0.1	2	
...					

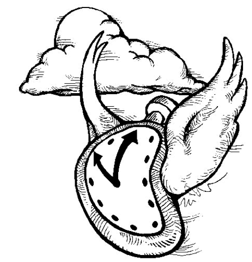
chocolate	4
peanut butter	8
ice cream	7
tuna	3
...	

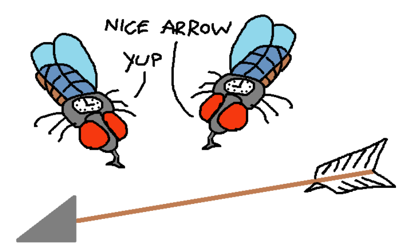
Mathematical Modeling

An Abstraction for Modeling

Factors have local opinions



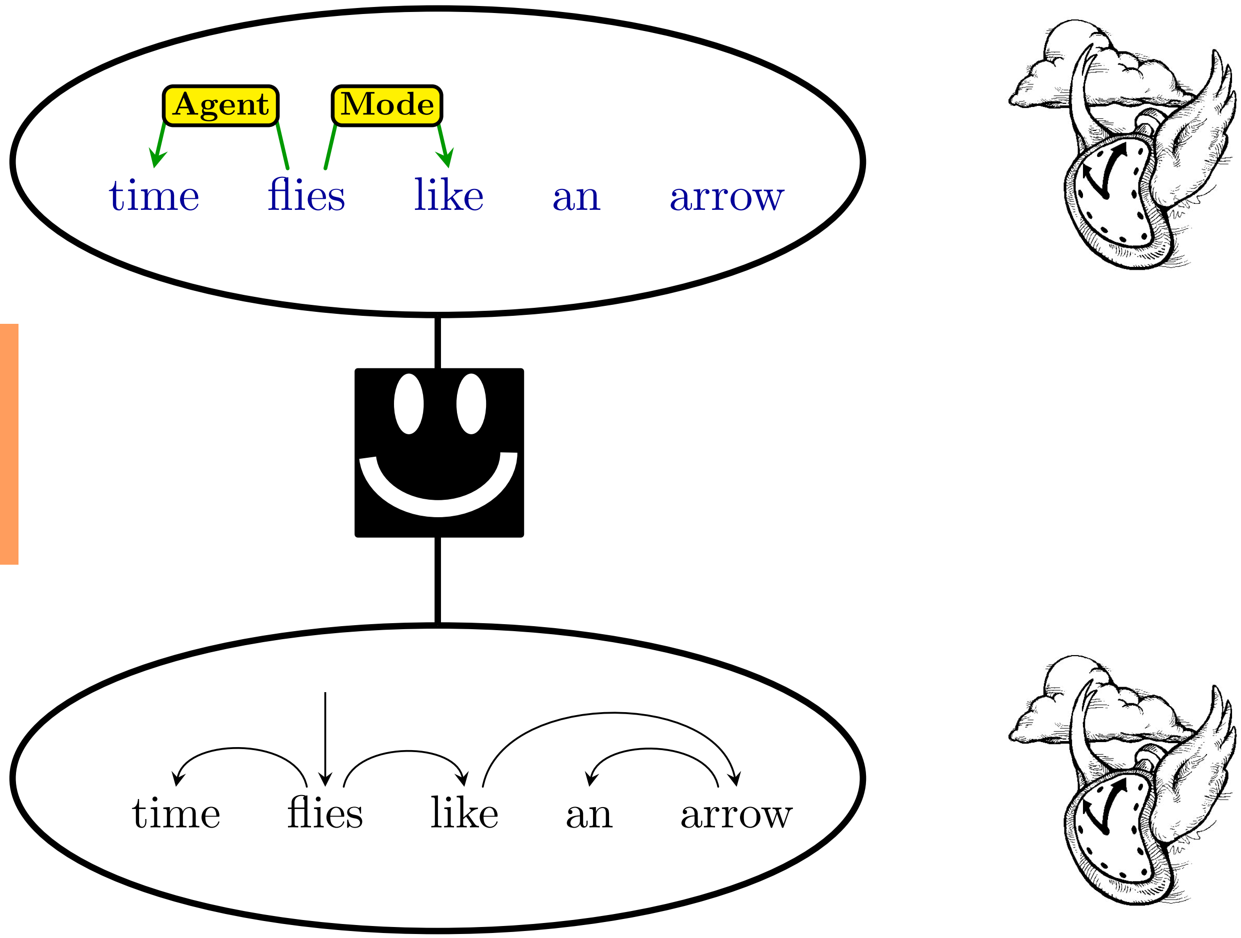




Mathematical Modeling

An Abstraction for Modeling

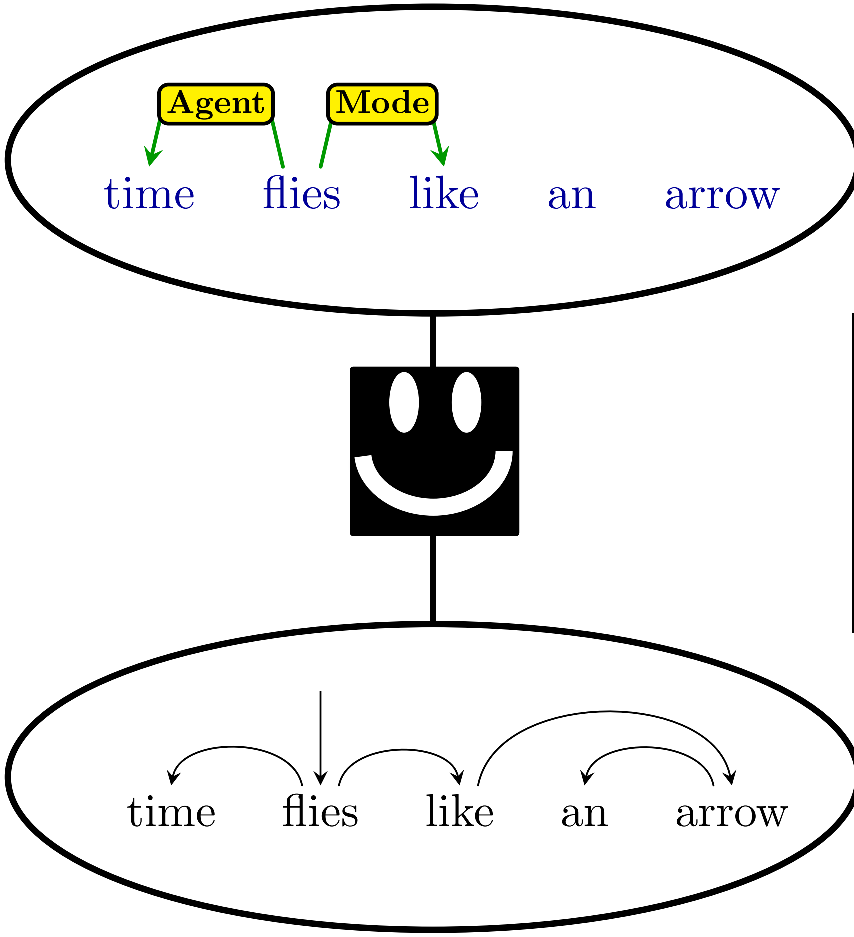
Factors have local opinions



Mathematical Modeling

An Abstraction for Modeling

The domains of these variables is exponential in the length of the sentence!

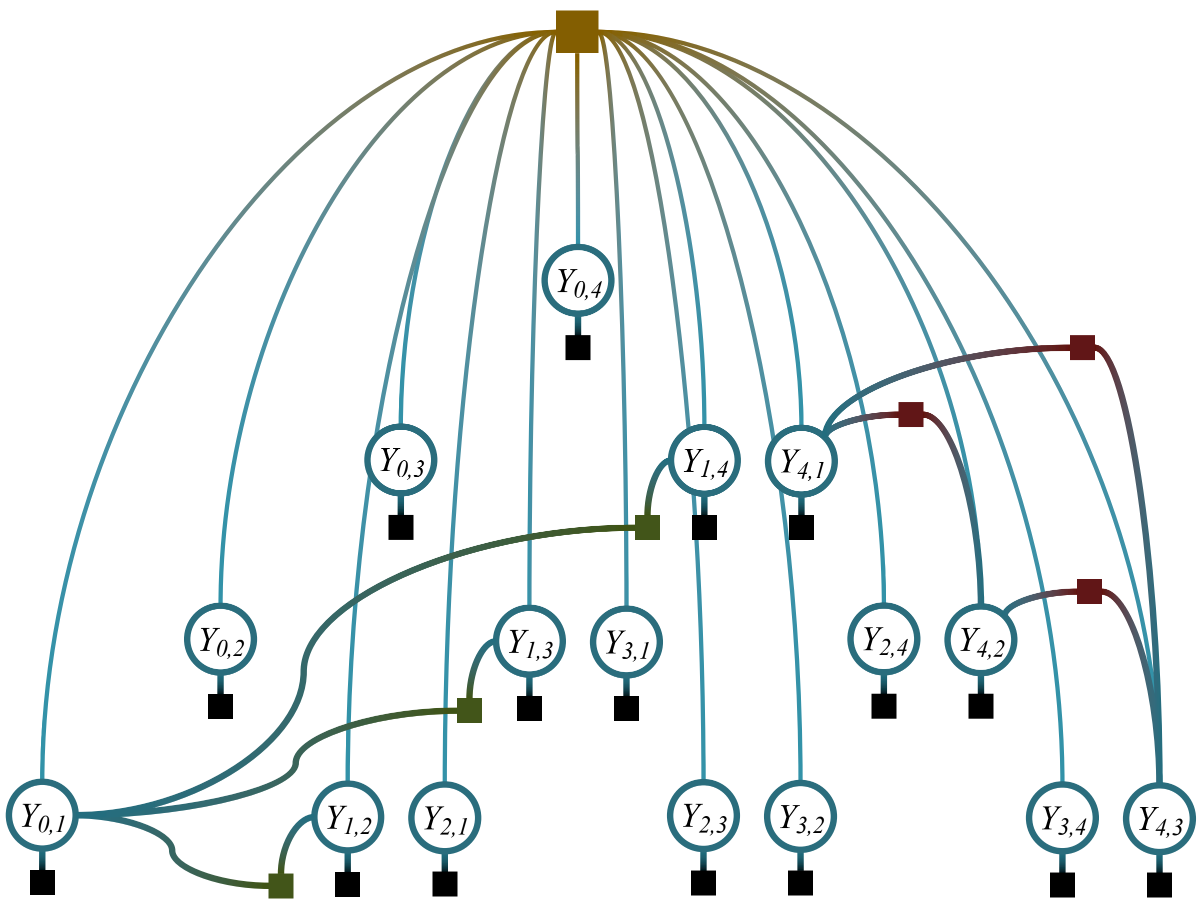


This factor would be massive

That's why decomposing into many small variables is so important

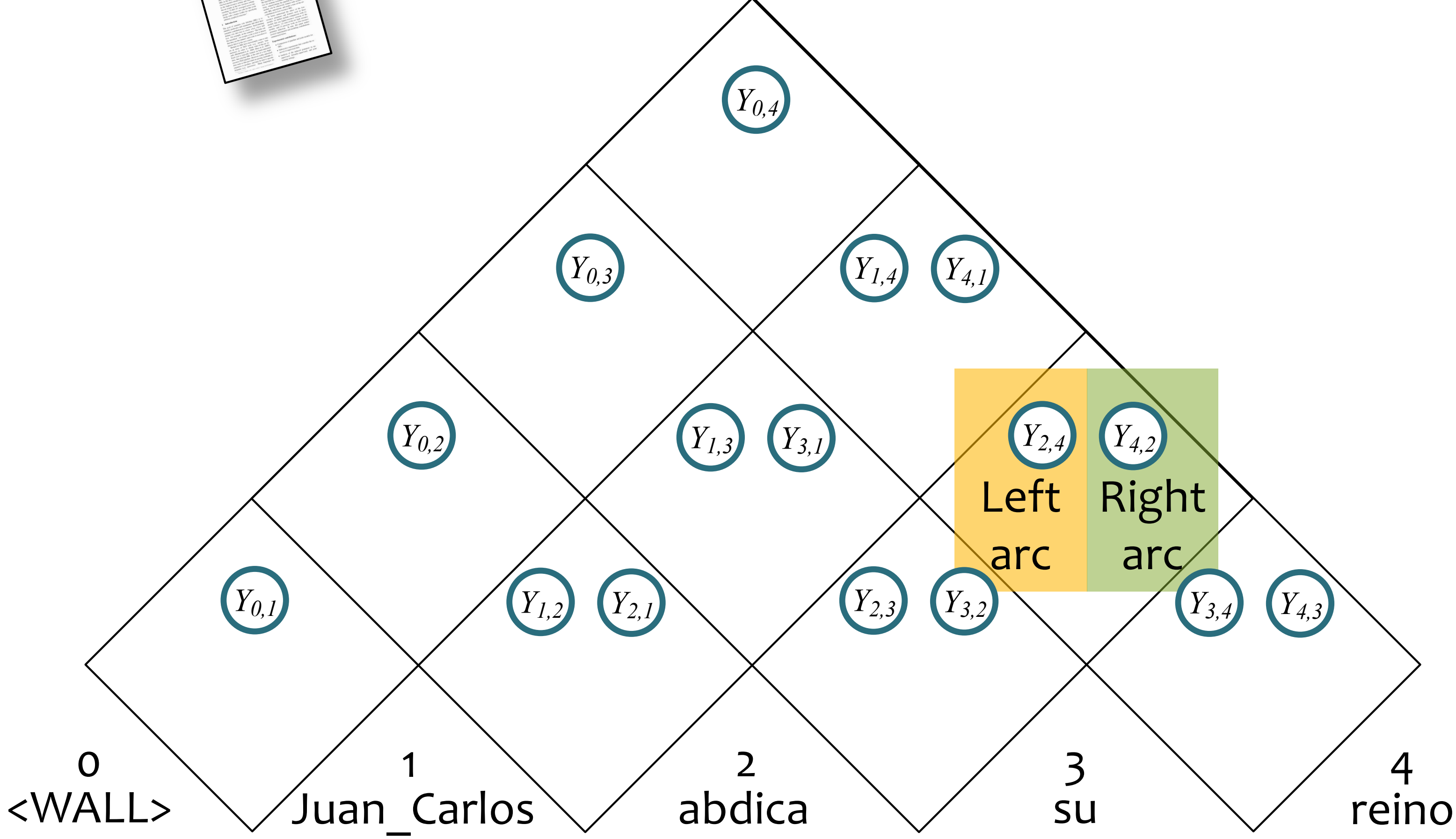
EXAMPLE: FACTOR GRAPH FOR DEPENDENCY PARSING

Factor Graph for Dependency Parsing



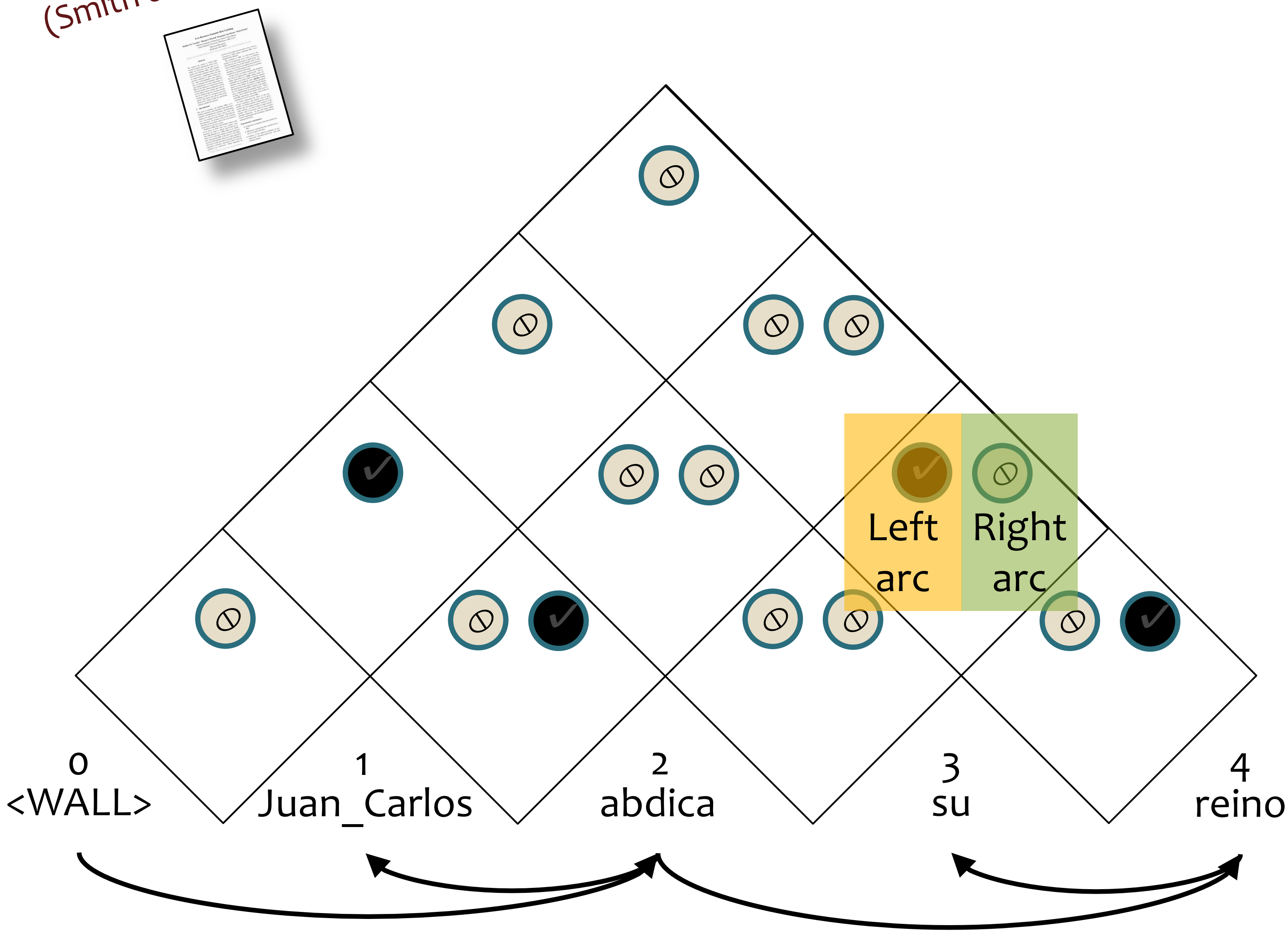
Factor Graph for Dependency Parsing

(Smith & Eisner, 2008)



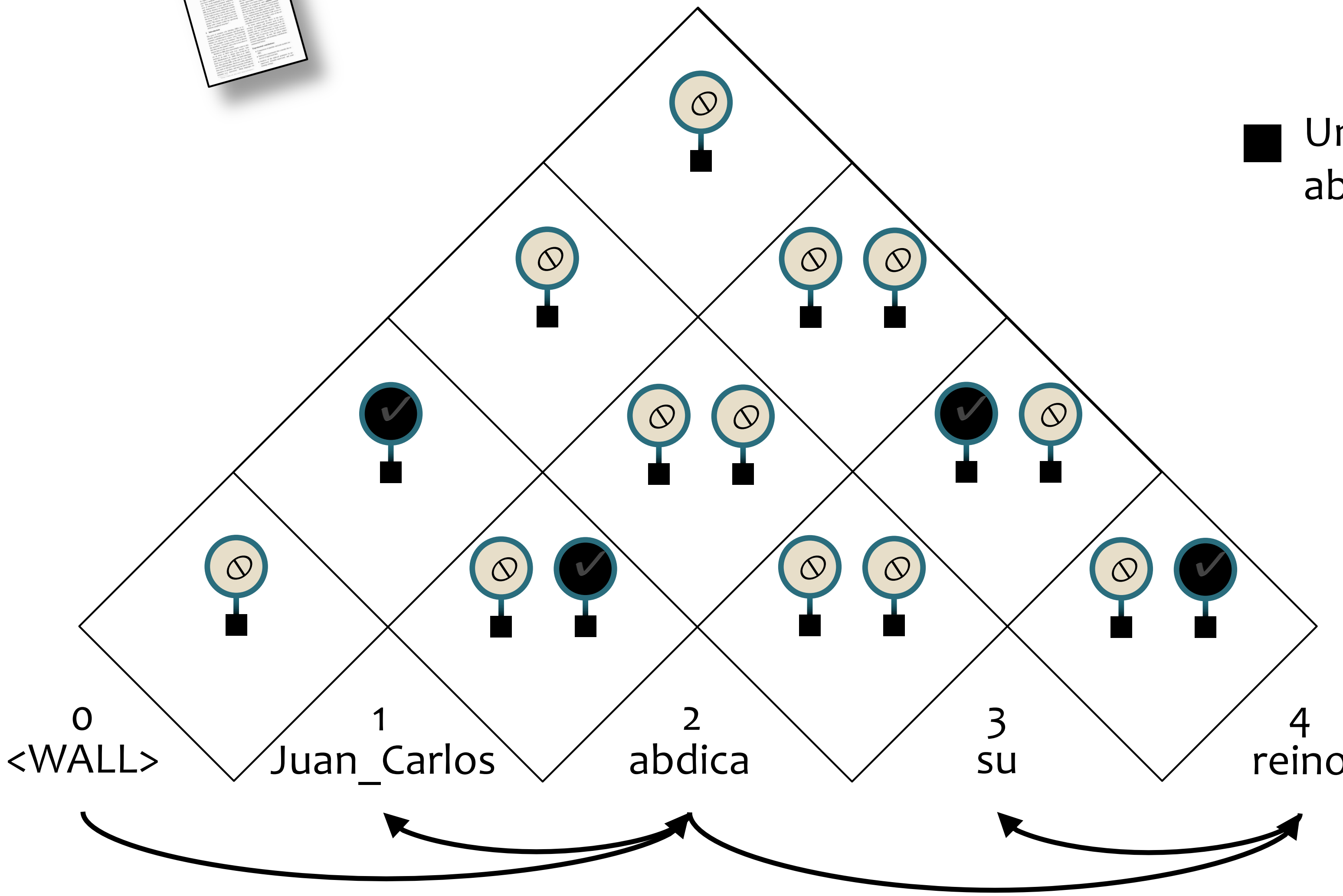
Factor Graph for Dependency Parsing

(Smith & Eisner, 2008)



Factor Graph for Dependency Parsing

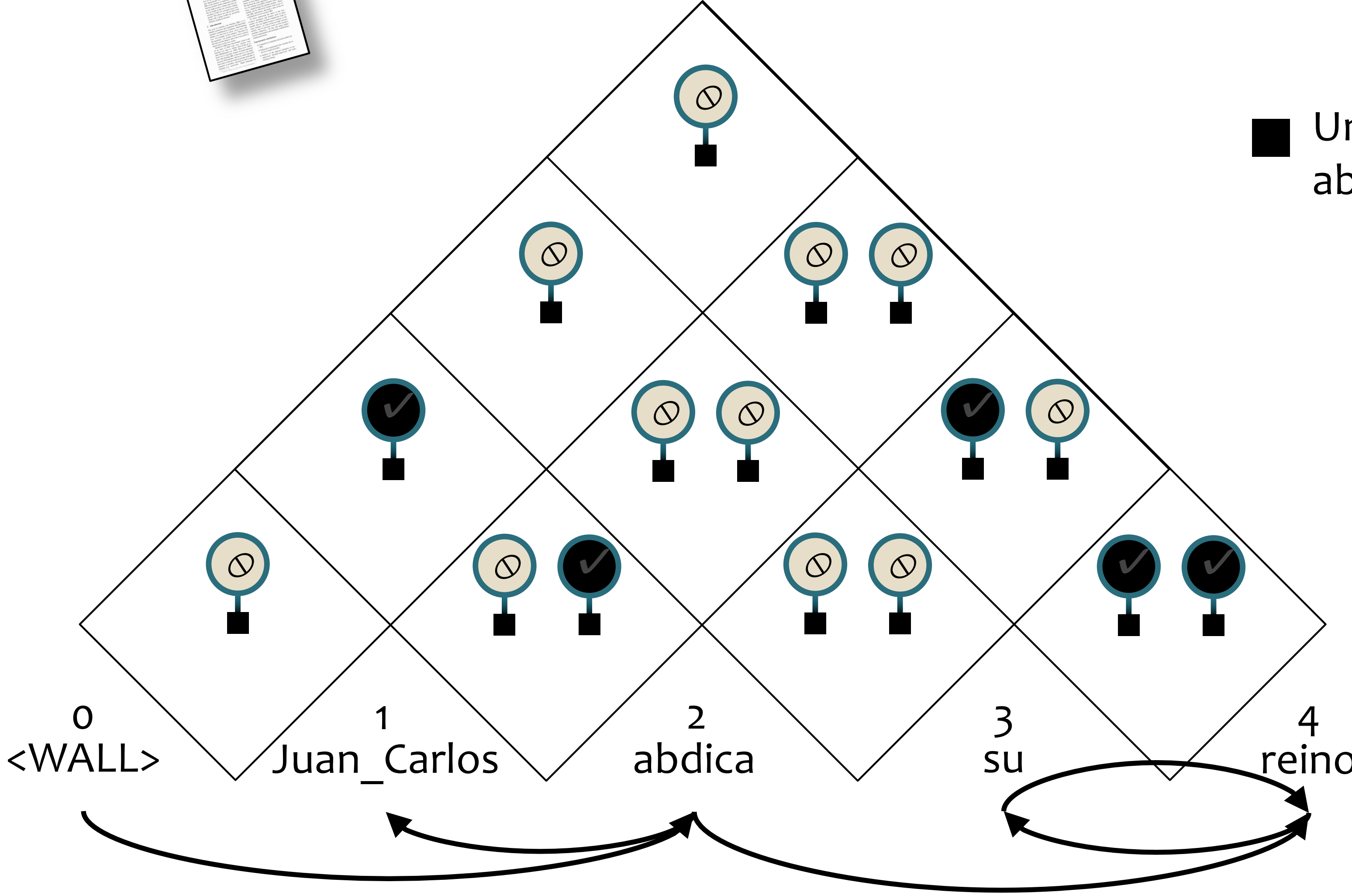
(Smith & Eisner, 2008)



■ Unary: local opinion about one edge

Factor Graph for Dependency Parsing

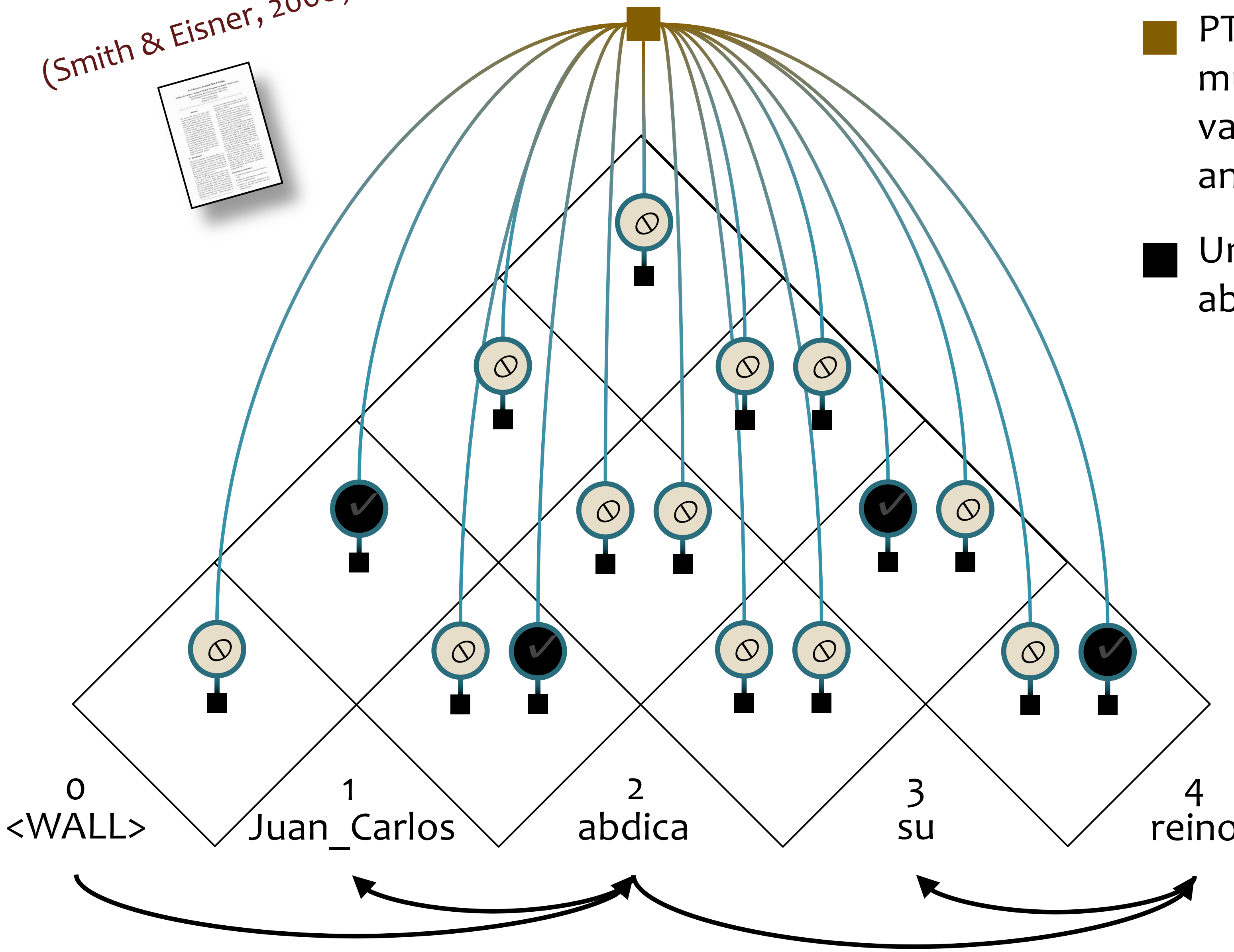
(Smith & Eisner, 2008)



■ Unary: local opinion about one edge

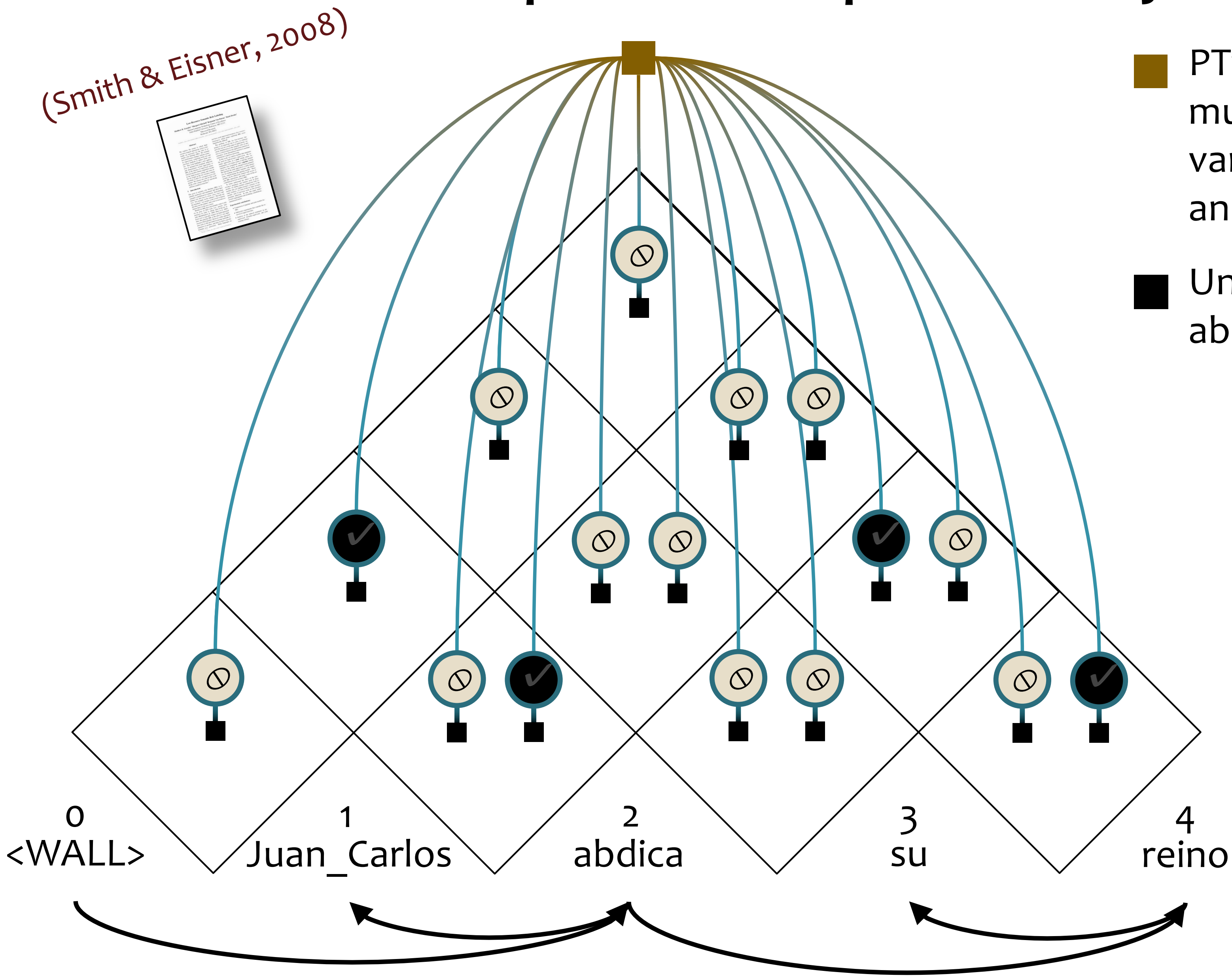
Factor Graph for Dependency Parsing

(Smith & Eisner, 2008)



- PTree: Hard constraint, multiplying in 1 if the variables form a tree and 0 otherwise.
- Unary: local opinion about one edge

Factor Graph for Dependency Parsing

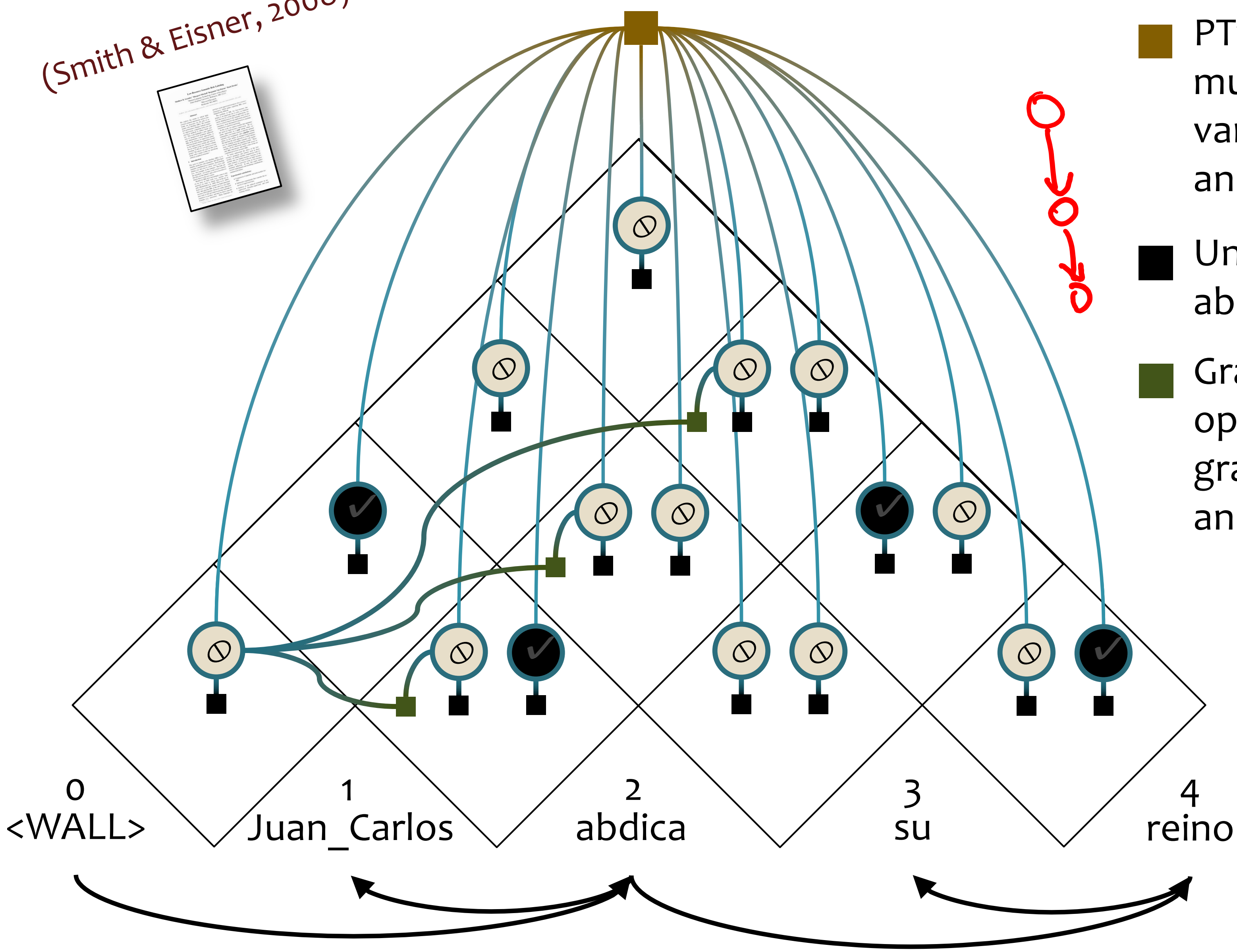


PTree: Hard constraint, multiplying in 1 if the variables form a tree and 0 otherwise.

Unary: local opinion about one edge

Factor Graph for Dependency Parsing

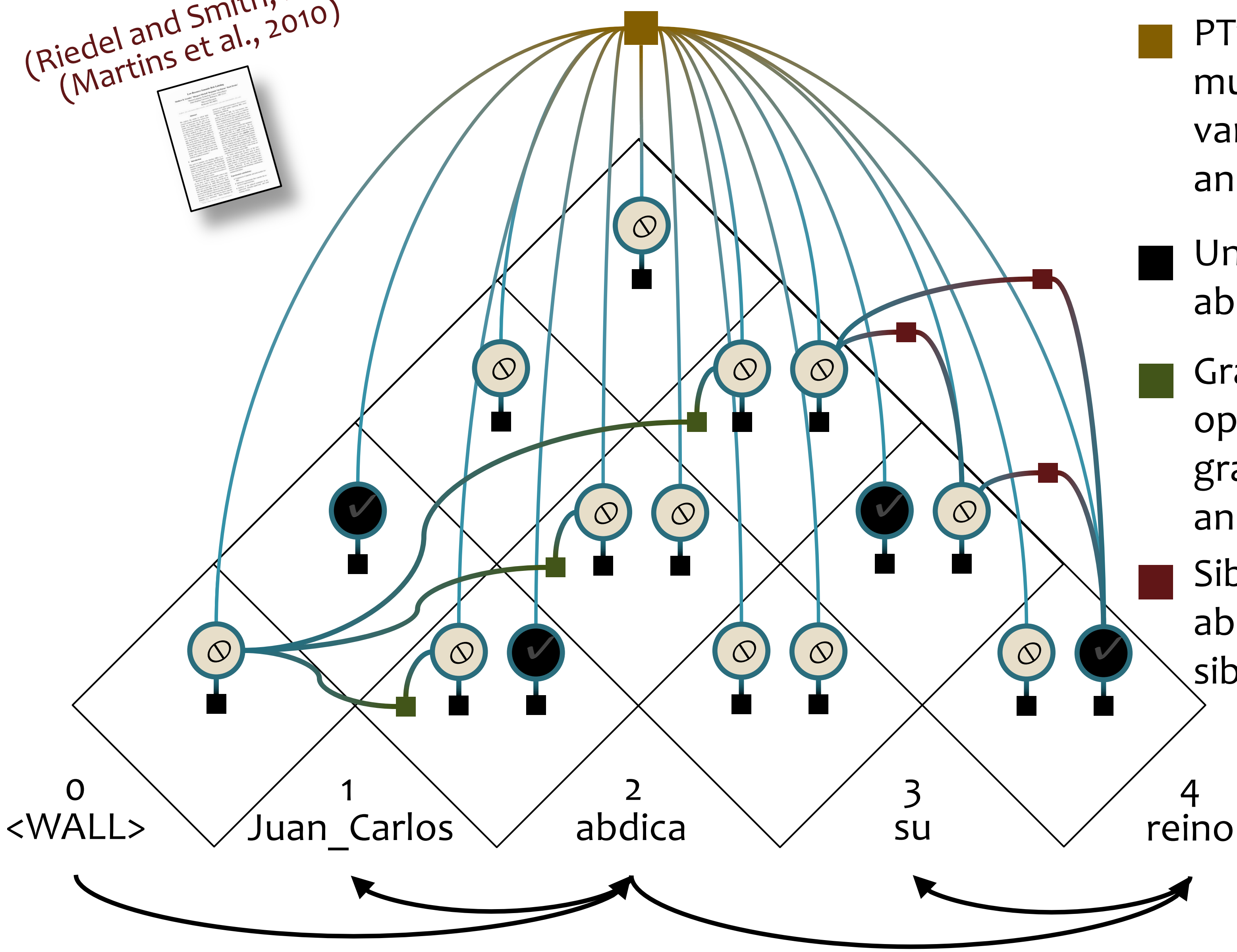
(Smith & Eisner, 2008)



- PTree: Hard constraint, multiplying in 1 if the variables form a tree and 0 otherwise.
- Unary: local opinion about one edge
- Grandparent: local opinion about grandparent, ~~head~~ parent, and ~~modifier~~ child

Factor Graph for Dependency Parsing

(Riedel and Smith, 2010)
(Martins et al., 2010)

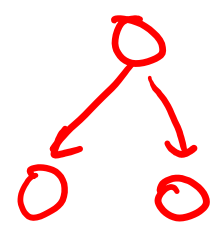


PTree: Hard constraint, multiplying in 1 if the variables form a tree and 0 otherwise.

Unary: local opinion about one edge

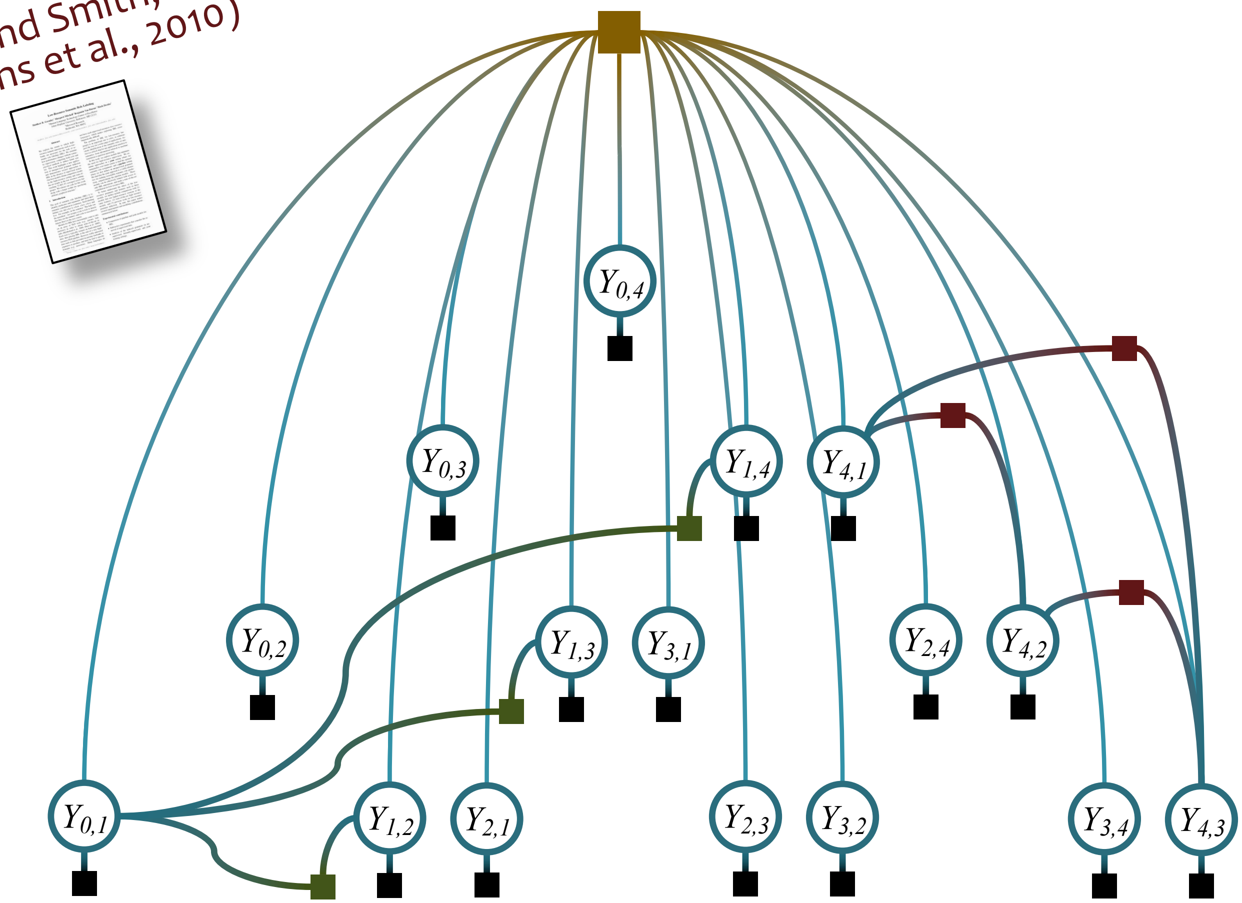
Grandparent: local opinion about grandparent, head, and modifier

Sibling: local opinion about pair of arbitrary siblings



Factor Graph for Dependency Parsing

(Riedel and Smith, 2010)
(Martins et al., 2010)



Now we can work at this level of abstraction.

$$p_\theta(\boldsymbol{y}) = \frac{1}{Z} \prod_{\alpha} \psi_\alpha(\boldsymbol{y}_\alpha)$$

VARIABLES AND INTERACTIONS

Joint Modeling

When do we add factors?

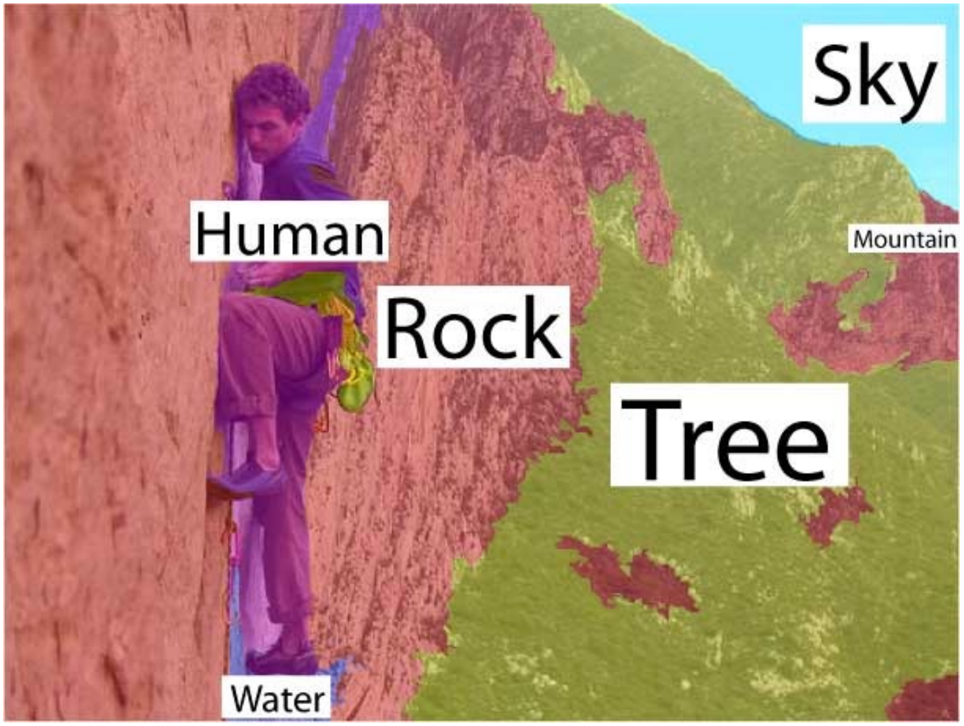
In order to determine which subsets of variables should have factors between them, we need to think about which variable **interactions** we want to model.

If we expect there to be an **interesting interaction between some collection of variables**, then we should **add a factor** to express an opinion about it

Scene Understanding

- **Variables:**
  - boundaries of image regions
  - tags of regions
- **Interactions:**
  - semantic plausibility of nearby tags
  - continuity of tags across visually similar regions (i.e. patches)

Labels **with** top-down information

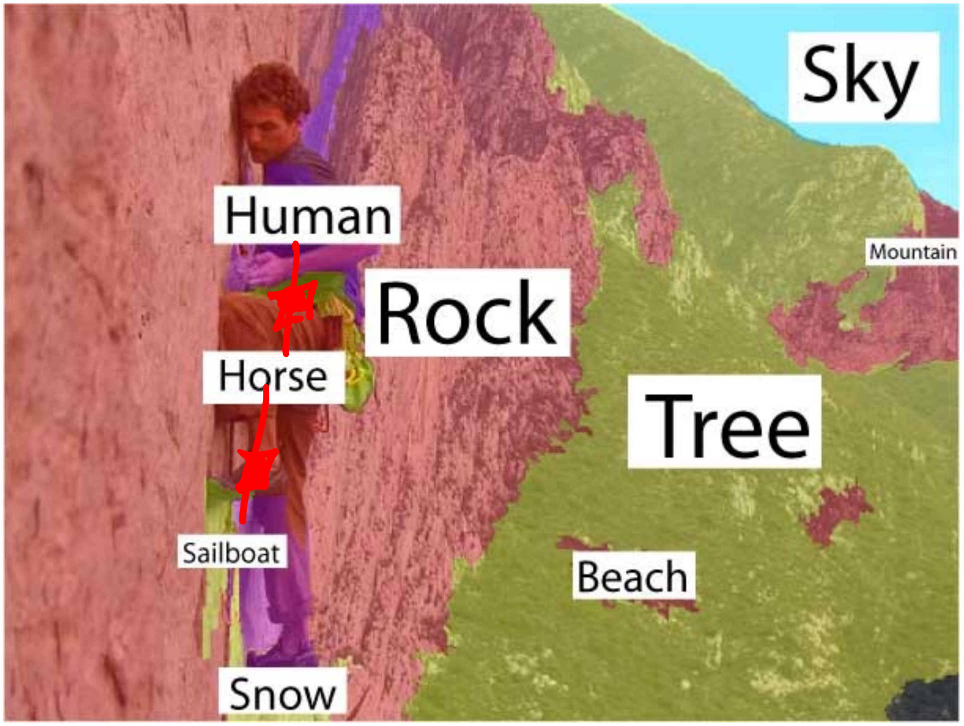


(Li et al., 2009)

Scene Understanding

- **Variables:**
  - boundaries of image regions
  - tags of regions
- **Interactions:**
  - semantic plausibility of nearby tags
  - continuity of tags across visually similar regions (i.e. patches)

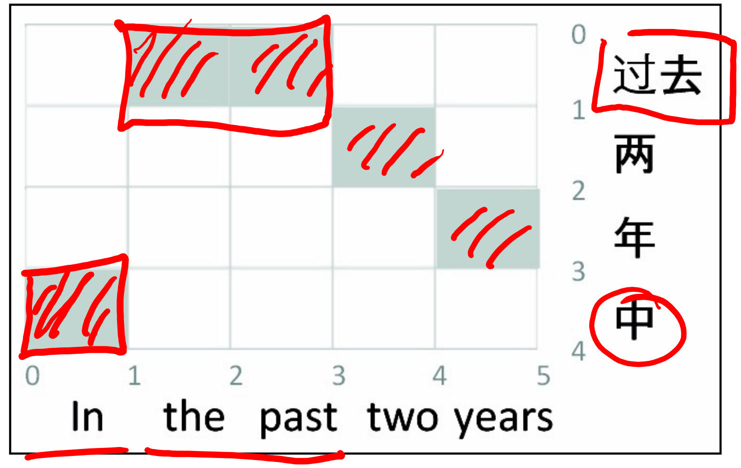
Labels **without** top-down information

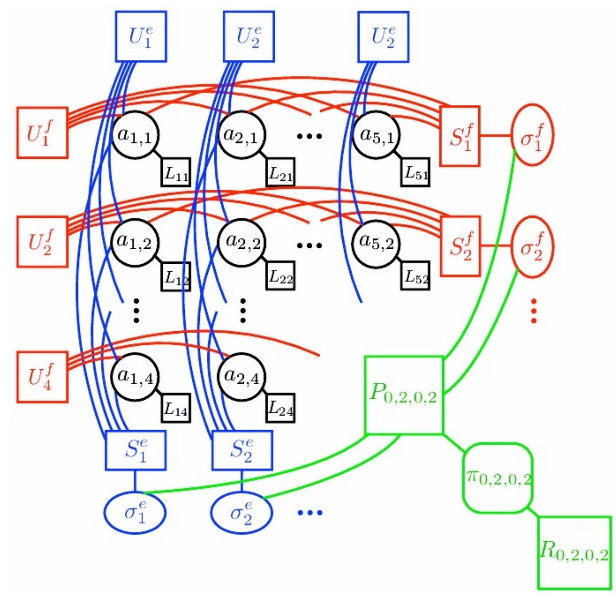


(Li et al., 2009)

Word Alignment / Phrase Extraction

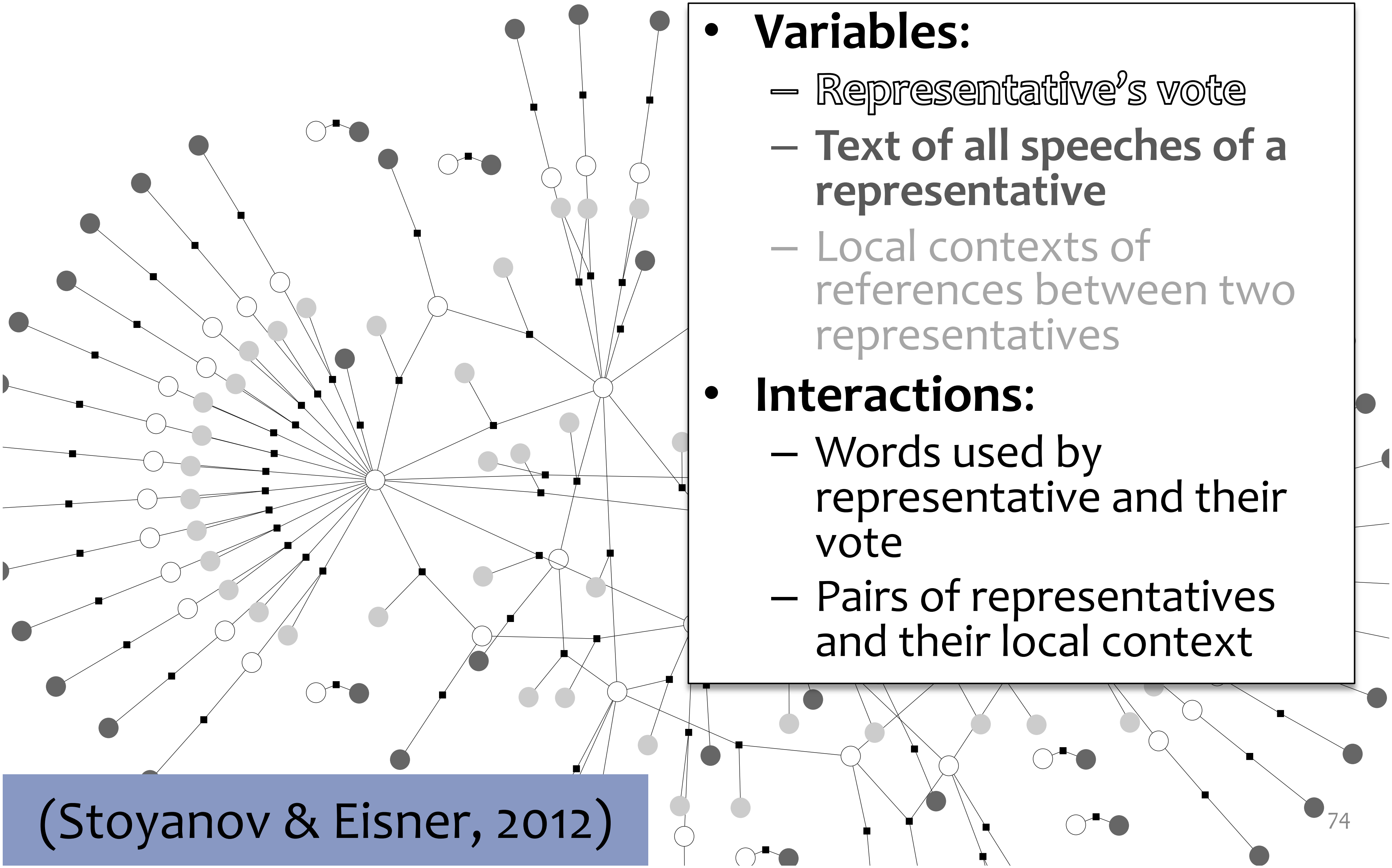
- **Variables (boolean):**
  - For each (Chinese phrase, English phrase) pair, are they linked?
- **Interactions:**
  - Word fertilities
  - Few "jumps" (discontinuities)
  - Syntactic reorderings
  - "ITG contraint" on alignment
  - Phrases are disjoint (?)





(Burkett & Klein, 2012)

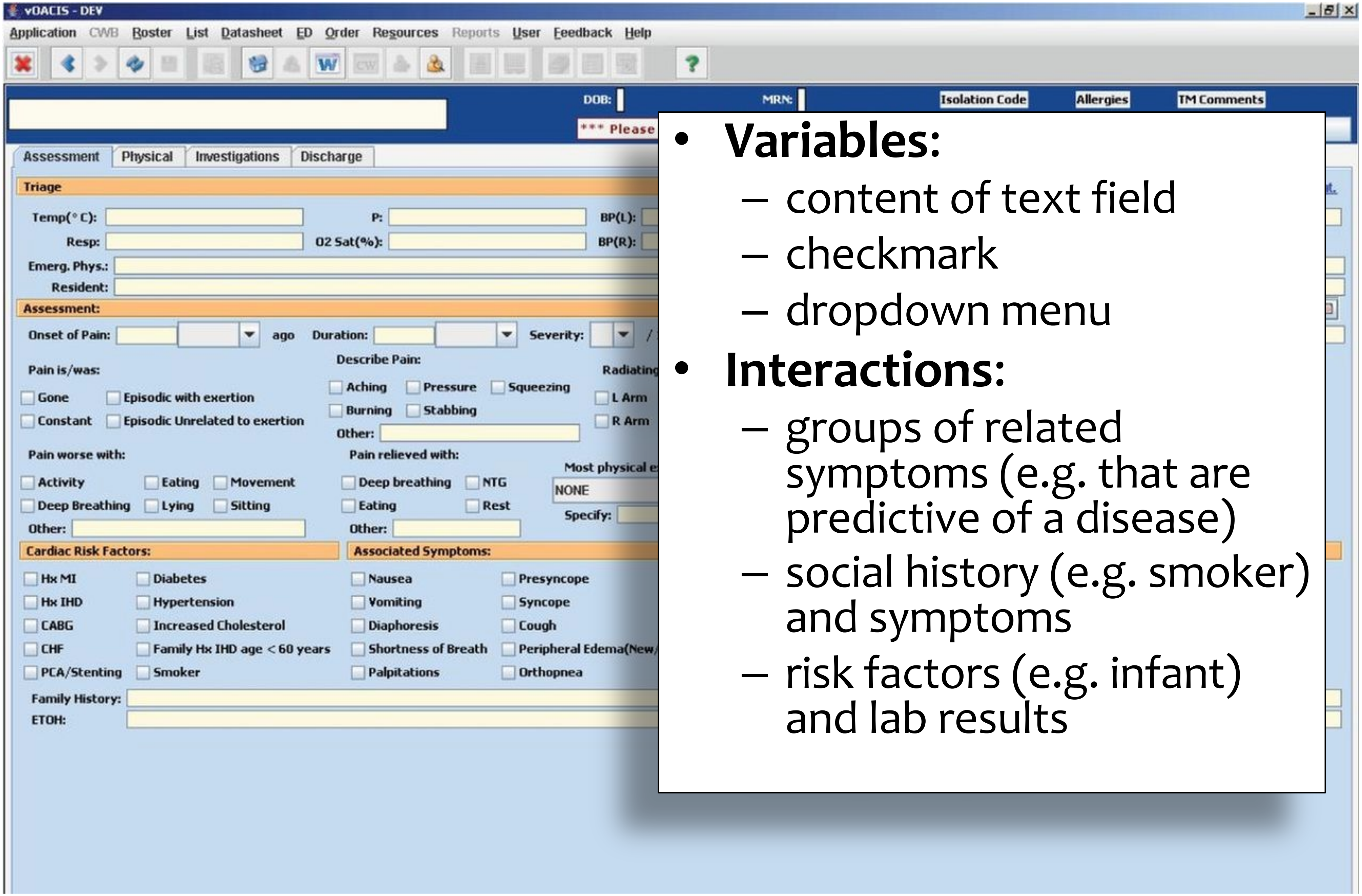
Congressional Voting



- **Variables:**
  - Representative's vote
  - **Text of all speeches of a representative**
  - Local contexts of references between two representatives
- **Interactions:**
  - Words used by representative and their vote
  - Pairs of representatives and their local context

(Stoyanov & Eisner, 2012)

Medical Diagnosis



- **Variables:**
  - content of text field
  - checkmark
  - dropdown menu
- **Interactions:**
  - groups of related symptoms (e.g. that are predictive of a disease)
  - social history (e.g. smoker) and symptoms
  - risk factors (e.g. infant) and lab results

(we'll talk about this in a later lecture…)

EXAMPLE: RECURRENT NEURAL NETWORK LANGUAGE MODEL

What if I want to model EVERY possible interaction?

…or at least the interactions of the current variable with all those that came before it…

(RNN-LMs)

RNN Language Model

RNN Language Model: $p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid f_{\boldsymbol{\theta}}(w_{t-1}, \ldots, w_1))$

$$p(w_1, w_2, w_3, \ldots, w_6) =$$

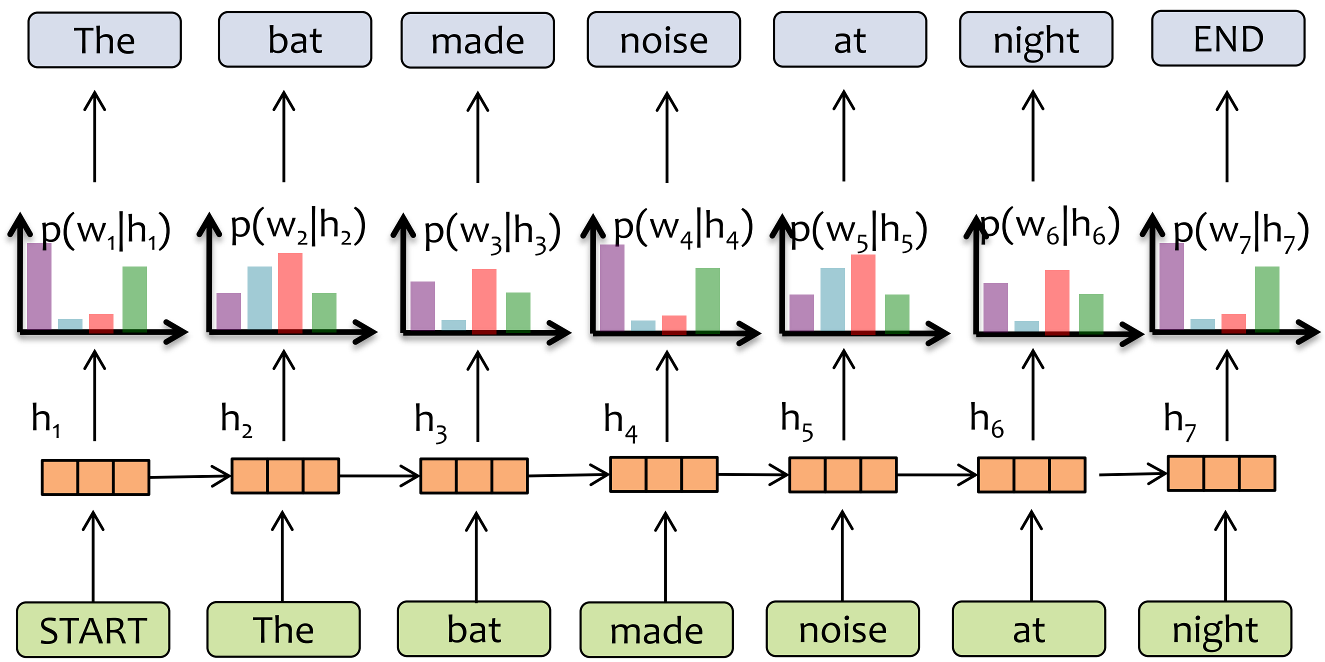
The						$p(w_1)$
The	bat					$p(w_2 \mid f_\theta(w_1))$
The	bat	made				$p(w_3 \mid f_\theta(w_2, w_1))$
The	bat	made	noise			$p(w_4 \mid f_\theta(w_3, w_2, w_1))$
The	bat	made	noise	at		$p(\boxed{w_5} \mid f_\theta(\underline{w_4, w_3, w_2, w_1}))$
The	bat	made	noise	at	night	$p(w_6 \mid f_\theta(w_5, w_4, w_3, w_2, w_1))$

<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector

RNN Language Model

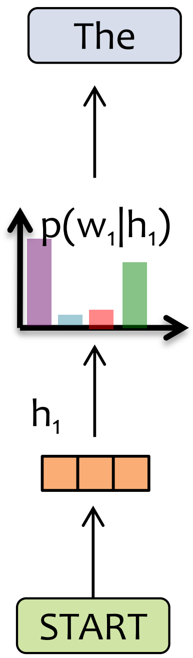


<u>*Key Idea*</u>:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

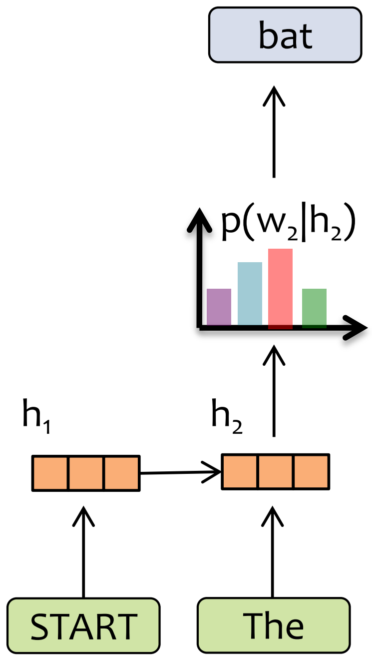


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

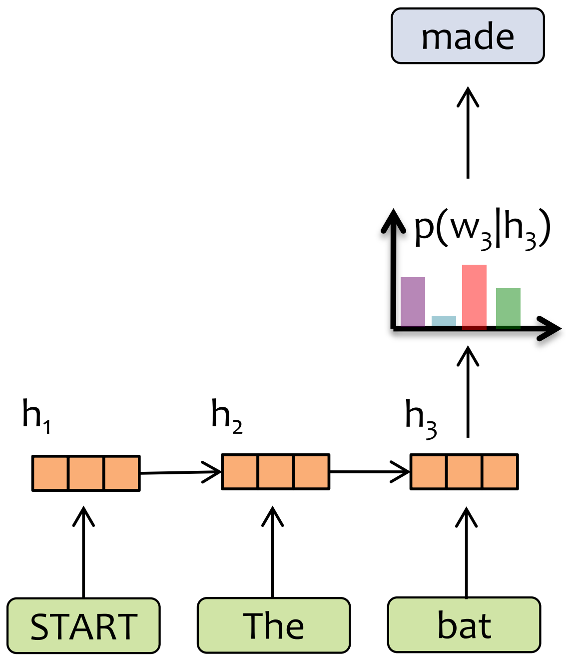
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

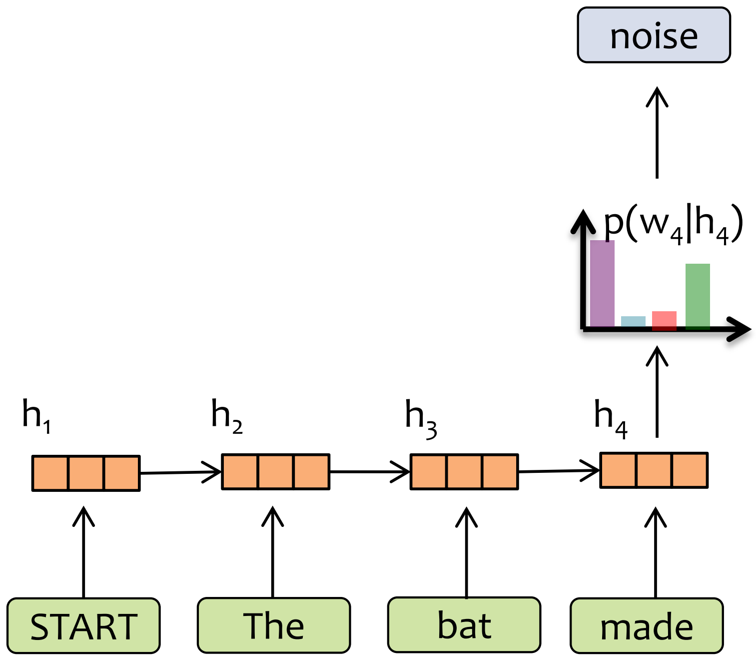


Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

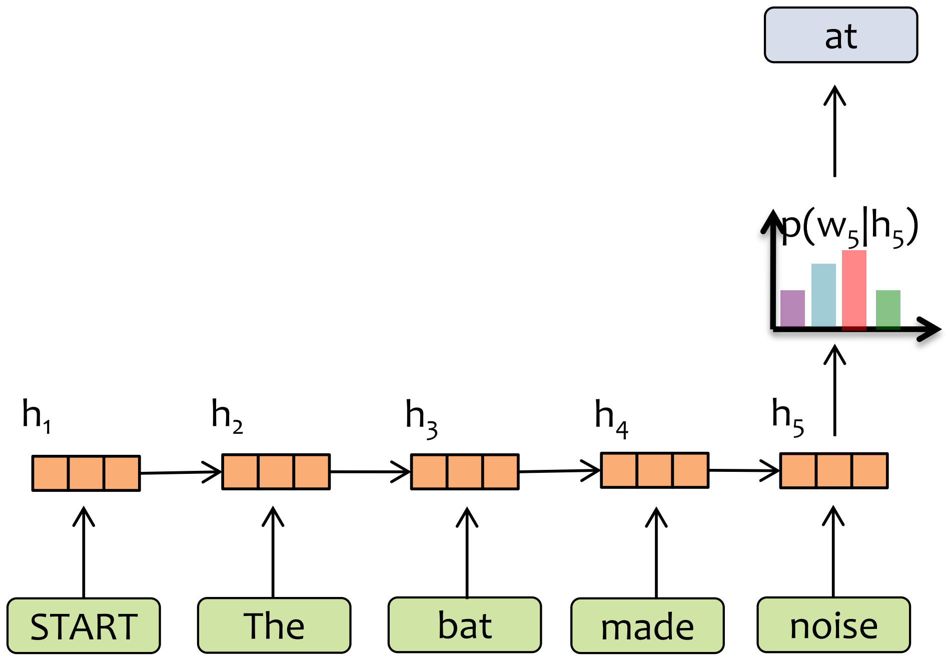


Key Idea:

(1) convert all previous words to a **fixed length vector**

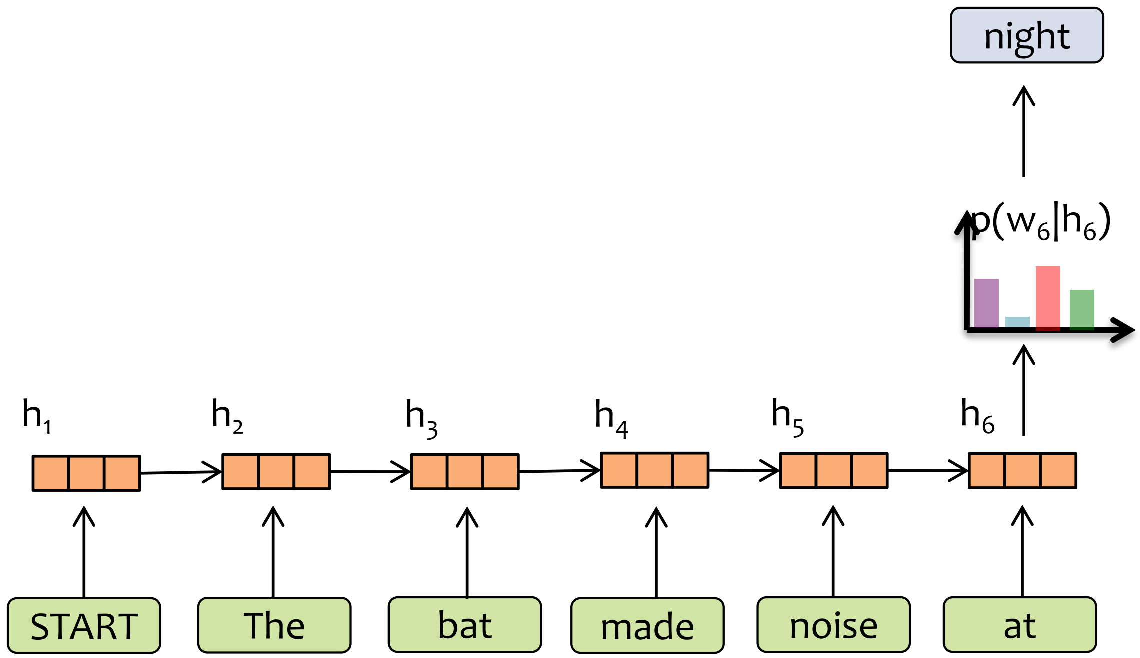
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



Key Idea:
(1) convert all previous words to a **fixed length vector**
(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

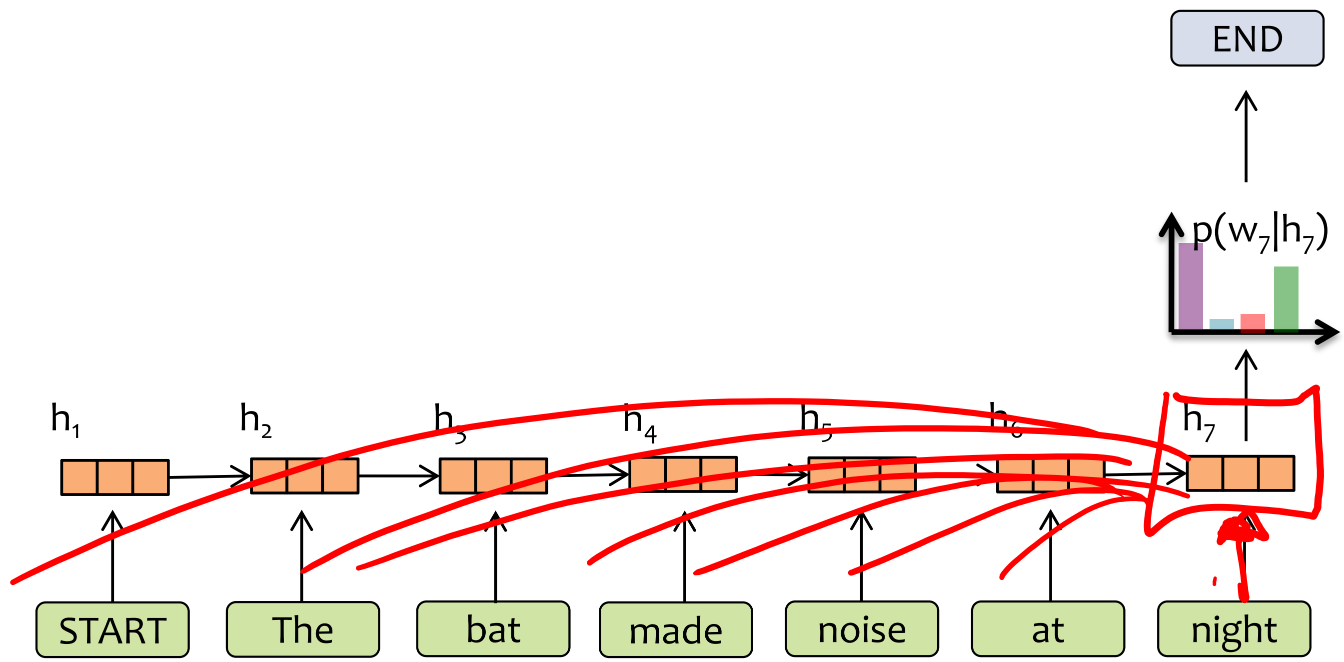


Key Idea:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model

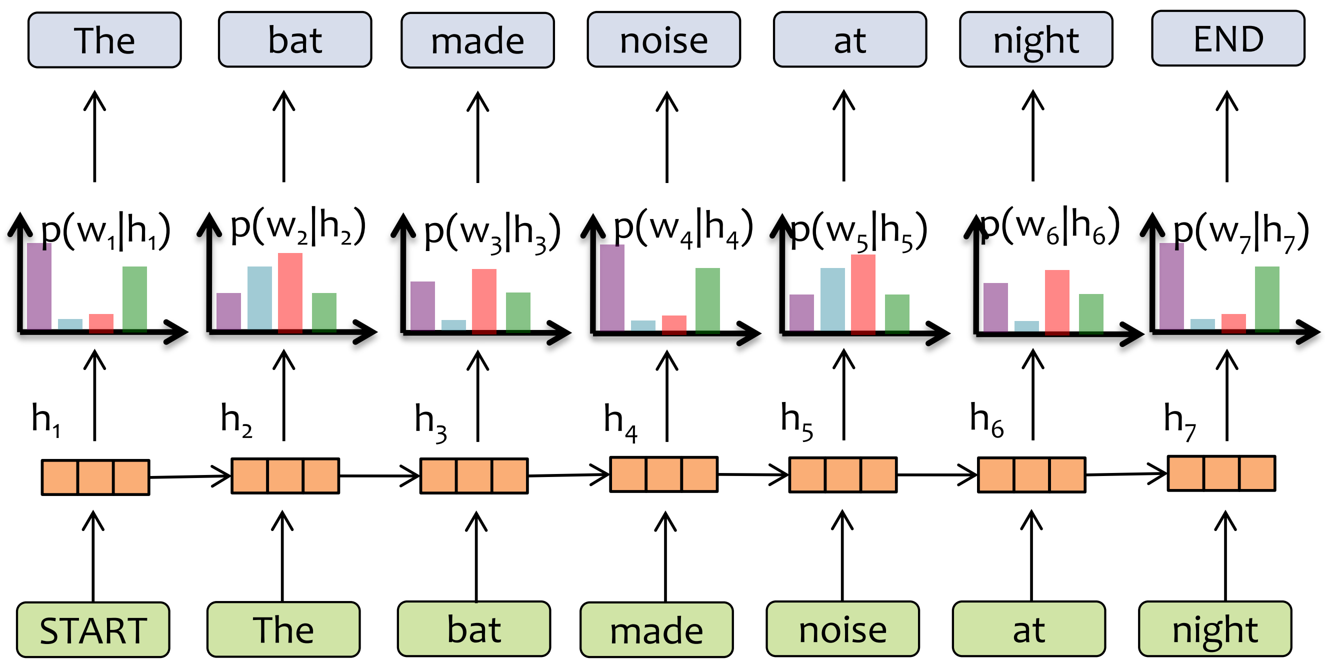


<u>Key Idea</u>:

(1) convert all previous words to a **fixed length vector**

(2) define distribution $p(w_t \mid f_\theta(w_{t-1}, \ldots, w_1))$ that conditions on the vector $\mathbf{h}_t = f_\theta(w_{t-1}, \ldots, w_1)$

RNN Language Model



$$p(w_1, w_2, w_3, \ldots , w_T) = p(w_1 \mid h_1)\, p(w_2 \mid h_2) \ldots\; p(w_2 \mid h_T)$$

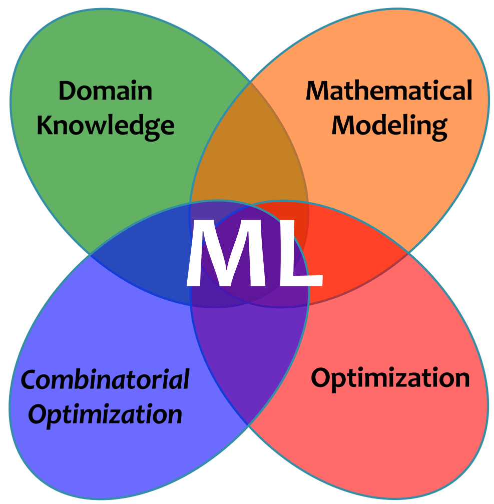
A PREVIEW OF INFERENCE

Structured Prediction

The **data** inspires the structures we want to predict

Our **model** defines a score for each structure

It also tells us what to optimize



Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

Learning tunes the parameters of the model

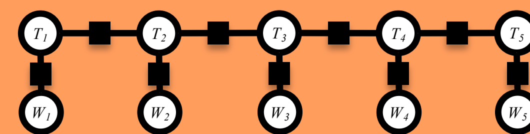
Structured Prediction

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$



3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

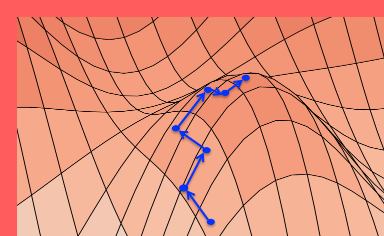
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\ p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

1. Marginal Inference (#P-Hard)

Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad \Big| \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function (#P-Hard)

Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference (NP-Hard)

Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

4. Sampling (cf. convergence, variance)

Draw a sample variable assignment

$$\mathbf{x} \sim p(\cdot | \boldsymbol{\theta})$$

Q&A

Q: But in **deep learning** we don't need to solve these inference problems, right?

A: Wrong…it's not that we don't *need* to solve them, it's that we often can't!

Questions you *could* ask your RNN-LM or seq2seq model:

X 1. What is the probability of the 7^{th} token being 'zebra' (marginal inference)

X 2. For an unnormalized model, what is the normalization constant? (partition function)

X 3. What is the most probable output sequence? (MAP inference)

✓ 4. Give me 10 samples from the distribution.

Topics (Part I)

- Search-Based Structured Prediction
  - Reductions to Binary Classification
  - Learning to Search
  - RNN-LMs
  - seq2seq models
- Graphical Model Representation
  - Directed GMs vs. Undirected GMs vs. Factor Graphs
  - Bayesian Networks vs. Markov Random Fields vs. Conditional Random Fields
- Graphical Model Learning
  - Fully observed Bayesian Network learning
  - Fully observed MRF learning
  - Fully observed CRF learning
  - Parameterization of a GM
  - Neural potential functions
- Exact Inference
  - Three inference problems: (1) marginals (2) partition function (3) most probably assignment
  - Variable Elimination
  - Belief Propagation (sum-product and max-product)

Topics (Part II)

- Learning for Structure Prediction
  - Structured Perceptron
  - Structured SVM
  - Neural network potentials
- (Approximate) MAP Inference
  - MAP Inference via MILP
  - MAP Inference via LP relaxation
- Approximate Inference by Sampling
  - Monte Carlo Methods
  - Gibbs Sampling
  - Metropolis-Hastings
  - Markov Chains and MCMC
- Parameter Estimation
  - Bayesian inference
  - Topic Modeling
- Approximate Inference by Optimization
  - Variational Inference
  - Mean Field Variational Inference
  - Coordinate Ascent V.I. (CAVI)
  - Variational EM
  - Variational Bayes
- Bayesian Nonparametrics
  - Dirichlet Process
  - DP Mixture Model
- Deep Generative Models
  - Variational Autoencoders

SYLLABUS HIGHLIGHTS

Syllabus Highlights

The syllabus is located on the course webpage:

http://418.mlcourse.org
http://618.mlcourse.org → … cs.cmu.edu…

The **course policies** are **required** reading.

Syllabus Highlights

- **Grading 418**: 60% homework, 15% midterm, 20% final, 5% participation
- **Grading 618**: 55% homework, 15% midterm, 15% final, 5% participation, 10% project
- **Midterm Exam**: in-class exam, Fri, Oct. 14
- **Final Exam**: final exam week, date/time TBD by registrar
- **Homework**: ~6 assignments
  - 8 grace days for homework assignments
  - Late submissions: 75% day 1, 50% day 2, 25% day 3
  - No submissions accepted after 3 days w/o extension
  - Extension requests: for emergency situations, see syllabus
- **Recitations**: Fridays, same time/place as lecture (optional, interactive sessions)
- **Readings**: required, online PDFs, recommended for after lecture
- **Technologies**:
  - Piazza (discussion),
  - Gradescope (homework),
  - Google Forms (polls),
  - Zoom (livestream),
  - Panopto (video recordings)
- **Academic Integrity**:
  - Collaboration encouraged, but must be documented
  - Solutions must always be written independently
  - No re-use of found code / past assignments
  - Severe penalties (i.e.. failure)
- **Office Hours**: posted on Google Calendar on “Office Hours” page

Lectures

- You should ask lots of questions
  - Interrupting (by raising a hand, turning on your video, and waiting to be called on) to ask your question is strongly encouraged
  - Use the chat to ask questions in real time (TAs will be monitoring the chat and will either answer or interrupt the instructor)
  - Asking questions later on Piazza is also great
- When I ask a question…
  - I want you to answer
  - Even if you don't answer, think it through as though I'm about to call on you
- Interaction improves learning (both in-class and at my office hours)

Homework

There will be 6 homework assignments during the semester. The assignments will consist of both conceptual and programming problems.

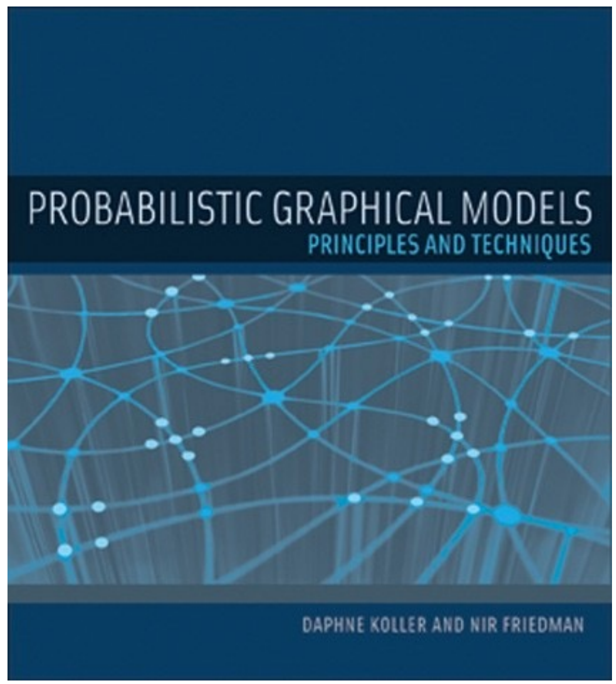
	Main Topic	Implementation	Application Area	Type
HW1	PyTorch Primer	MLP for Sequence Tagging	NLP	written + programming
HW2	Learning to Search	seq2seq + Dagger	speech recognition	written + programming
HW3	Marginal inference and MLE	RNN + Tree CRF	NLP	written + programming
HW4	MCMC	word embeddings + Gibbs sampler	topic modeling	written + programming
HW5	Variational Inference	mean field for cyclic CRF	computer vision	written + programming
HW6	Advanced Topics	NA		written

Mini-Project (10-618 only)

- Goals:
  - Explore a learning / inference technique of your choosing
  - Application and dataset will be provided (in the style of a Kaggle competition)
  - Deeper understanding of methods in real-world application
  - Work in teams of 2 students

Textbooks

You are not *required* to read a textbook, but Koller & Friedman is a thorough reference text that includes a lot of the topics we cover.



Prerequisites

What they are:

1. Introductory machine learning.
   (i.e. 10-301, 10-315, 10-601, 10-701)
2. Significant experience programming in a general programming language.
   - The homework will require you to use Python, so you will need to be **proficient in Python**.
3. College-level probability, calculus, linear algebra, and discrete mathematics.

Q&A