COGNITIVE COMPUTATION GROUP
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

# Learning and Inference in Structured Prediction Models

Kai-Wei Chang, Gourab Kundu, Dan Roth, Vivek Srikumar

MSR, IBM, Illinois, Utah

**February 2016**

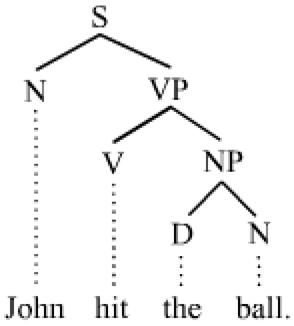
**AAAI-16, Phoenix, AZ**

# Structures

- All interesting decisions are structured

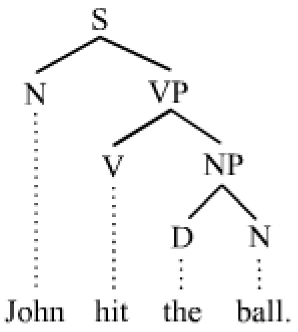
# Structures

- All interesting decisions are structured

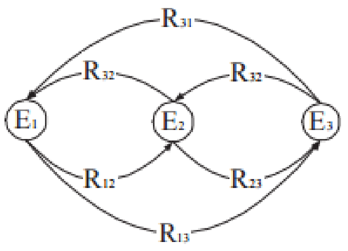



Constituency-based parse tree

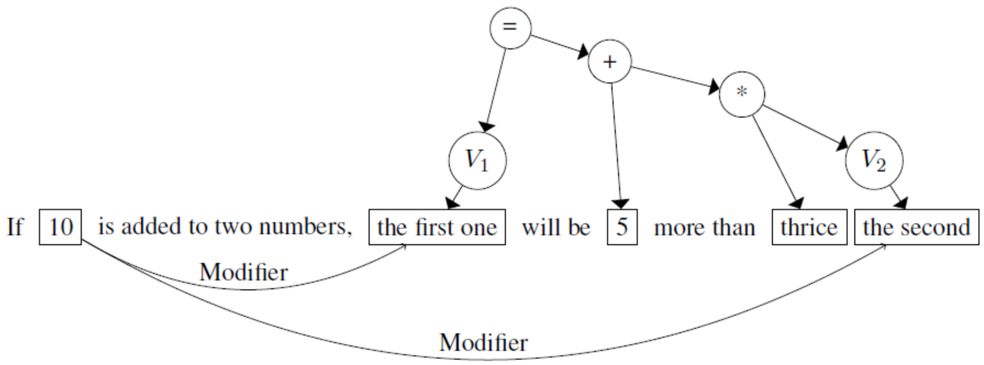
- All interesting decisions are structured

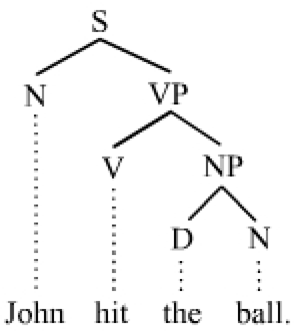



Constituency-based parse tree

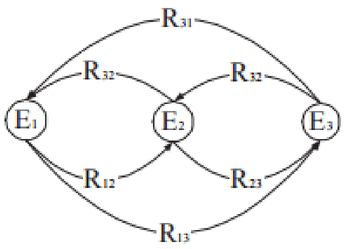


- All interesting decisions are structured



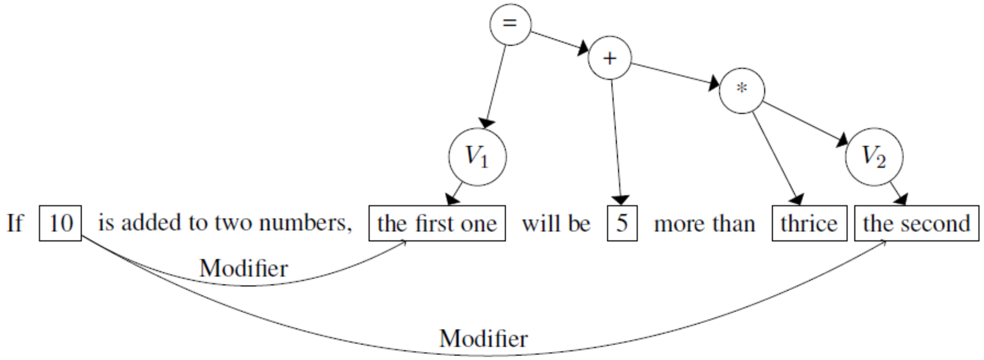



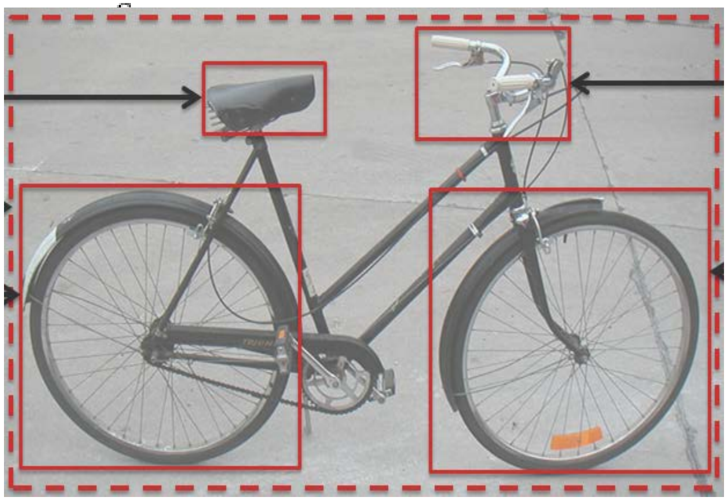

Constituency-based parse tree



# Structures

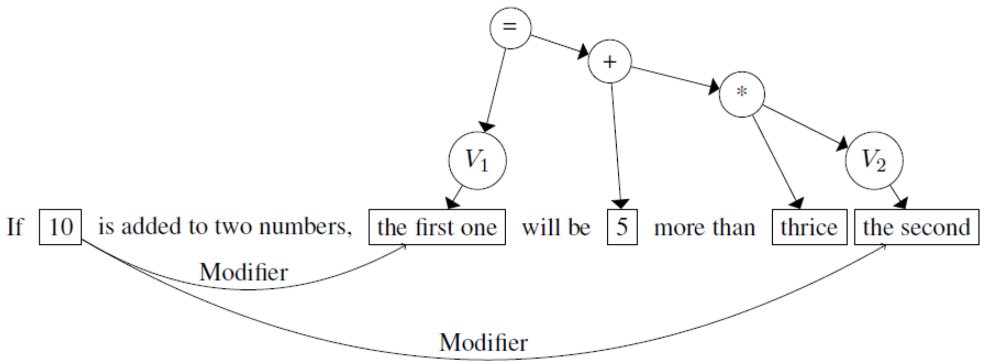
- All interesting decisions are structured

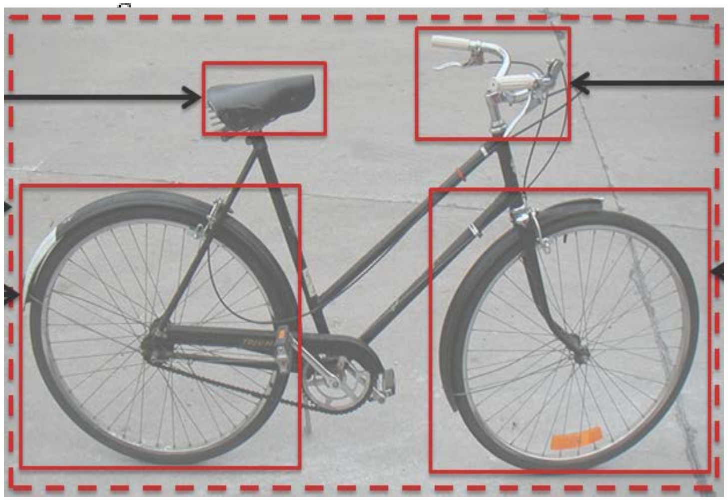


# Structures

- All interesting decisions are structured

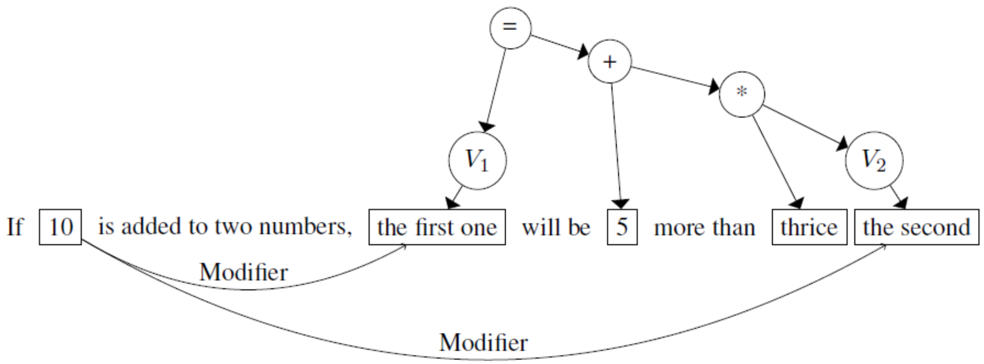


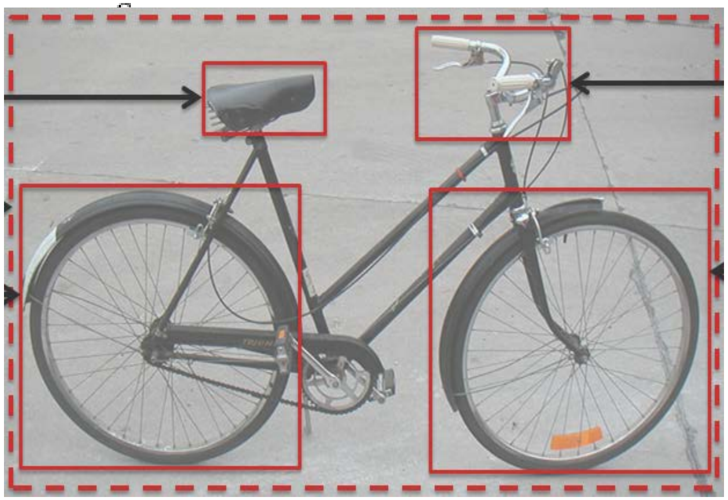



- "Understanding" is a global decision in which several local decisions play a role but there are mutual dependencies on their outcome.
- It is essential to make coherent decisions in a way that takes the interdependencies into account. Joint, Global Inference.

# Structures

- All interesting decisions are structured



- "Understanding" is a global decision in which several local decisions play a role but there are mutual dependencies on their outcome.
- It is essential to make coherent decisions in a way that takes the interdependencies into account. Joint, Global Inference.
  - Inference: How to support making these global, coherent decisions
  - Learning: How to learn models to support these decisions.

# Learning and Inference in Structured Prediction

- Part 1: Introduction to Structured Prediction (60min)
  - Motivation
  - Examples:
    - **NE + Relations**
    - **Vision**
    - **Additional NLP Examples**
  - Problem Formulation
    - **Constrained Conditional Models: Integer Linear Programming Formulations**
  - Initial thoughts about learning
    - **Learning independent models**
    - **Constraints Driven Learning**
  - Initial thoughts about Inference
    - **Amortized Inference**

- Part 2: Learning a Structured Prediction Model (45min)
  - Definition
  - Local Learning v.s. Global Learning
  - Global Learning Algorithms
    - **Online learning: Structured Perceptron**
    - **Batch learning: Structured SVM**
  - Optimization methods for Structured SVM
    - **Stochastic Gradient Decent**
    - **Dual Coordinate Descent**
    - **Learning on a multi-core machine**
- BREAK

- Part 3: Amortized Inference (45min)
  - Overview
  - Amortization at Inference Time
    - **Theorems**
    - **Decomposition**
    - **Results**
  - Amortization during Learning
    - **Approximate Inference**
    - **Results**

# Learning and Inference in Structured Prediction

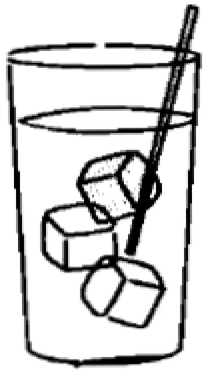
- Part 4: Distributed Representations for Structured Prediction (30 min)
  - Distributional representations for inputs is a success story
    - **Eg. word vectors**
  - Outputs are discrete objects
    - **One of a set of labels (document classification)**
    - **Label sequences (POS tagging, Chunking, NER)**
    - **Trees with labeled edges/nodes (Parsing)**
    - **Arbitrary graphs (Semantic Role Labeling, event extraction)**
  - Can we think of distributional representations for structures?
    - **Starting with individual labels to compose full structures**
    - **A natural generalization of standard structured prediction formalism**

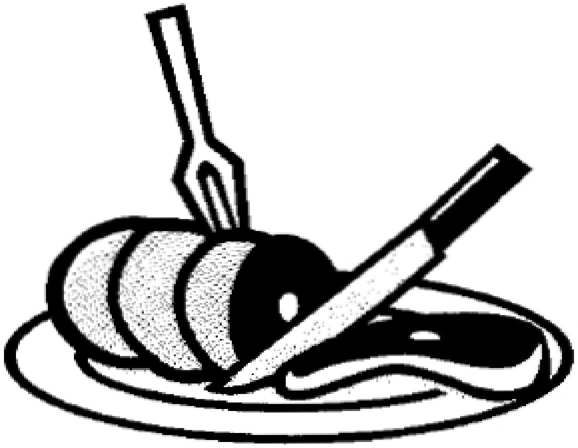
- Part 5: Structured Prediction Software (15min)
  - Illinois Structured Learning Library
    - **A general purpose learning library in JAVA**
    - **Support Structured Perceptron and Structured SVM**
  - Implement your own applications
- Part 6: Conclusion and Discussion (15min)

# PART 1: INTRODUCTION

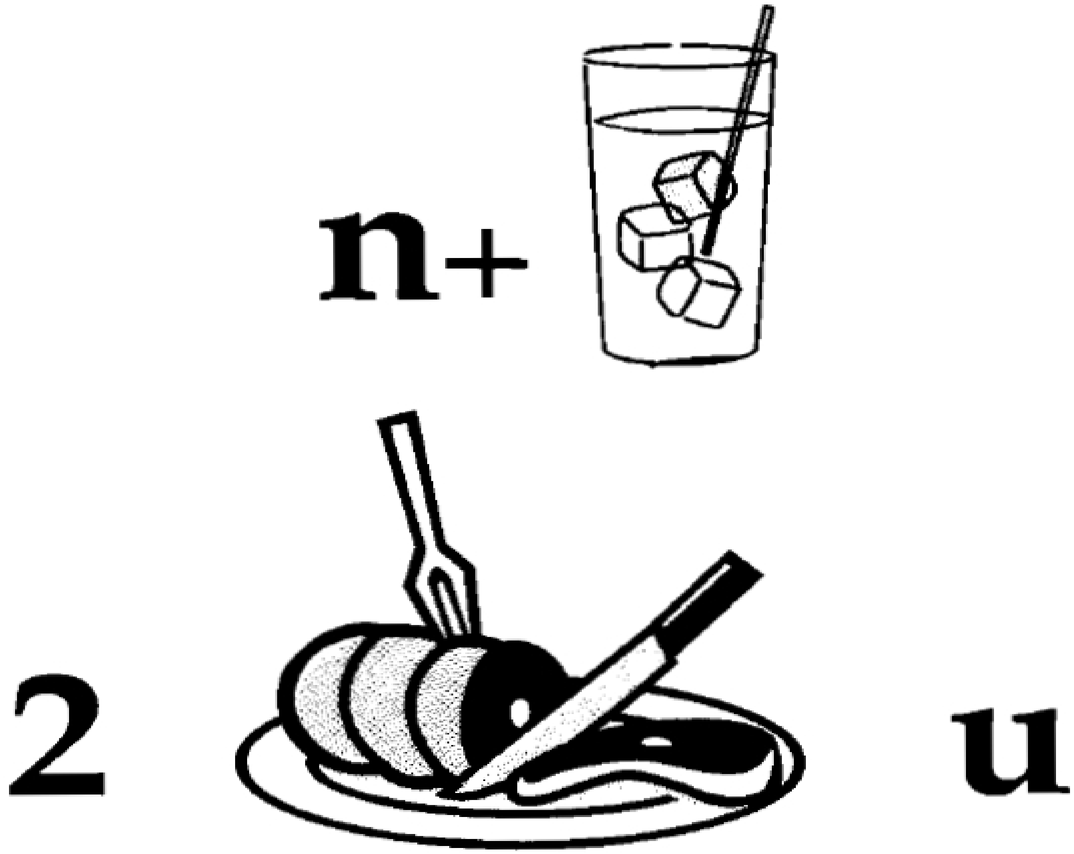
**Part 1: Introduction to Structured Prediction** (55min)

- **Motivation**
- **Examples:**
  - **NE + Relations**
  - **Vision**
  - **Additional NLP Examples**
- **Problem Formulation**
  - **Constrained Conditional Models: Integer Linear Programming Formulations**
- **Initial thoughts about learning**
  - **Learning independent models**
  - **Constraints Driven Learning**
- **Initial thoughts about Inference**
  - **Amortized Inference**

n+ 

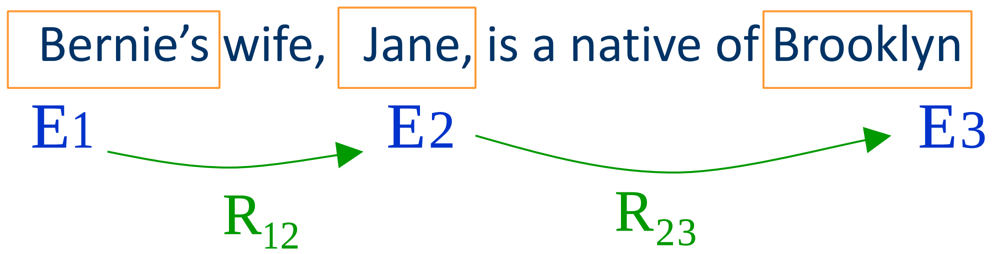
2  u

# Nice to Meet You



# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations



# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| per | 0.85 |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| per | 0.60 |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| loc | 0.45 |

Bernie's wife, Jane, is a native of Brooklyn

E1 E2 E3

$R_{12}$ $R_{23}$

| | |
|---|---|
| irrelevant | 0.05 |
| spouse_of | 0.45 |
| born_in | 0.50 |

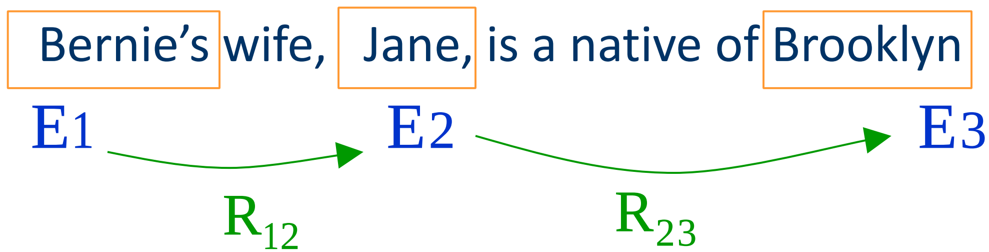
| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| born_in | 0.85 |

## Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| **per** | **0.50** |
| loc | 0.45 |




| | |
|---|---|
| irrelevant | 0.05 |
| spouse_of | 0.45 |
| **born_in** | **0.50** |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| **per** | **0.50** |
| loc | 0.45 |

Bernie's wife, Jane, is a native of Brooklyn

$E_1$ → $E_2$ → $E_3$

$R_{12}$ $R_{23}$

| | |
|---|---|
| irrelevant | 0.05 |
| spouse_of | 0.45 |
| **born_in** | **0.50** |

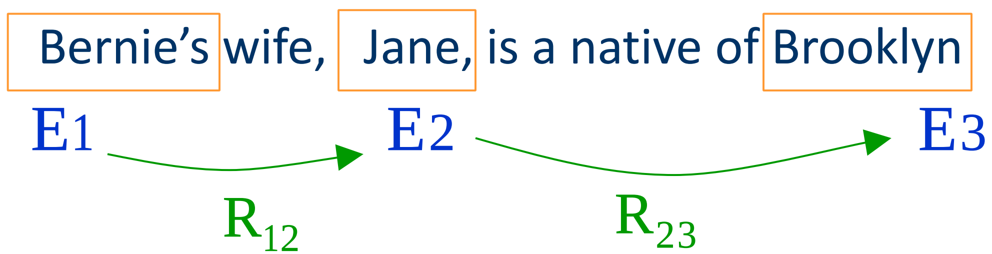
| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

## Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |




| | |
|---|---|
| irrelevant | 0.05 |
| spouse_of | 0.45 |
| **born_in** | **0.50** |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

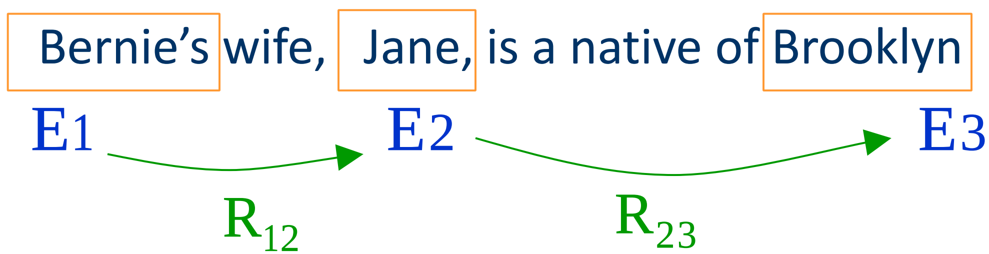
# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |




| | |
|---|---|
| irrelevant | 0.05 |
| spouse_of | 0.45 |
| **born_in** | **0.50** |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

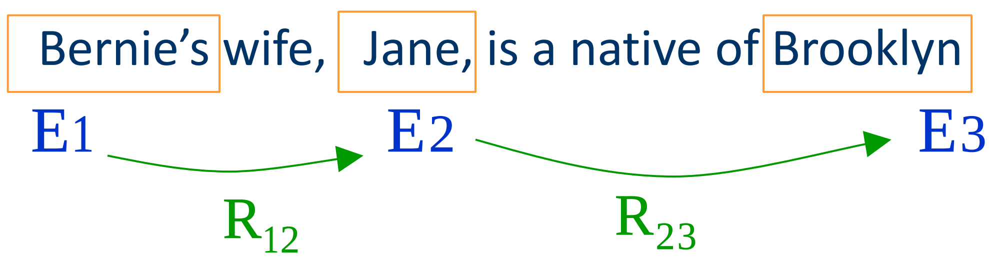
# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |




| | |
|---|---|
| irrelevant | 0.05 |
| **spouse_of** | **0.45** |
| born_in | 0.50 |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

**Joint inference gives good improvement**

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |

Bernie's wife, Jane, is a native of Brooklyn

E1 → E2 → E3

$R_{12}$ $R_{23}$

| | |
|---|---|
| irrelevant | 0.05 |
| **spouse_of** | **0.45** |
| born_in | 0.50 |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

**Joint inference gives good improvement**

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |

Bernie's wife, Jane, is a native

E1 E2

$R_{12}$ $R_{23}$

Key Questions:

**How to learn the model(s)?**

**What is the source of the knowledge?**

**How to guide the global inference?**

| | |
|---|---|
| irrelevant | 0.05 |
| **spouse_of** | **0.45** |
| born_in | 0.50 |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

**Joint inference gives good improvement**

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |

Bernie's wife, Jane, is a native

E1 → E2

$R_{12}$ $R_{23}$

Key Questions:

**How to learn the model(s)?**

**What is the source of the knowledge?**

**How to guide the global inference?**

| | |
|---|---|
| irrelevant | 0.05 |
| **spouse_of** | **0.45** |
| born_in | 0.50 |

| | |
|---|---|
| irrelevant | 0.10 |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

Models could be learned separately/jointly; constraints may come up only at decision time.

# Joint Inference with General Constraint Structure [Roth&Yih'04,07,….]

Recognizing Entities and Relations

**Joint inference gives good improvement**

| | |
|---|---|
| other | 0.05 |
| **per** | **0.85** |
| loc | 0.10 |

| | |
|---|---|
| other | 0.10 |
| **per** | **0.60** |
| loc | 0.30 |

| | |
|---|---|
| other | 0.05 |
| per | 0.50 |
| **loc** | **0.45** |

Key Questions:
… learn the model(s)?
… the knowledge?
… ference?

An Objective function that incorporates learned models with knowledge (**output constraints**)

A Constrained Conditional Model

| | |
|---|---|
| irrelevant | 0.05 |
| **spouse_of** | **0.45** |
| born_in | 0.50 |

| | |
|---|---|
| [illegible] | [illegible] |
| spouse_of | 0.05 |
| **born_in** | **0.85** |

Models could be learned separately/jointly; constraints may come up only at decision time.

# Pipeline

Motivation I

- **Most problems are not single classification problems**

# Pipeline



Raw Data

- **Most problems are not single classification problems**

# Pipeline

Motivation I

- **Most problems are not single classification problems**

Raw Data

POS Tagging → Phrases → Semantic Entities → Relations

Parsing → WSD → Semantic Role Labeling

# Pipeline

Motivation I

- **Most problems are not single classification problems**

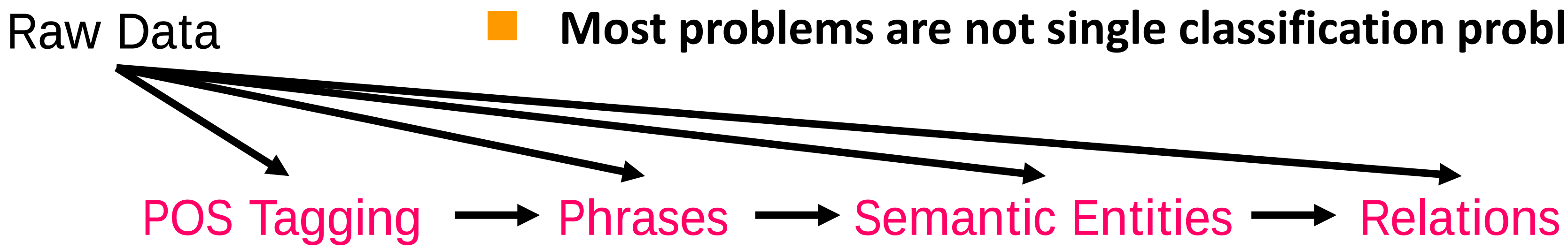
Raw Data

POS Tagging → Phrases → Semantic Entities → Relations

# Pipeline

Motivation I

- **Most problems are not single classification problems**




- Conceptually, Pipelining is a crude approximation

# Pipeline

Motivation I

Raw Data

- **Most problems are not single classification problems**

POS Tagging → Phrases → Semantic Entities → Relations

- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and down stream decisions often interact with previous decisions.
  - Leads to propagation of errors
  - Occasionally, later stages could be used to correct earlier errors.

# Pipeline

Motivation I

- **Most problems are not single classification problems**

Raw Data

POS Tagging → Phrases → Semantic Entities → Relations

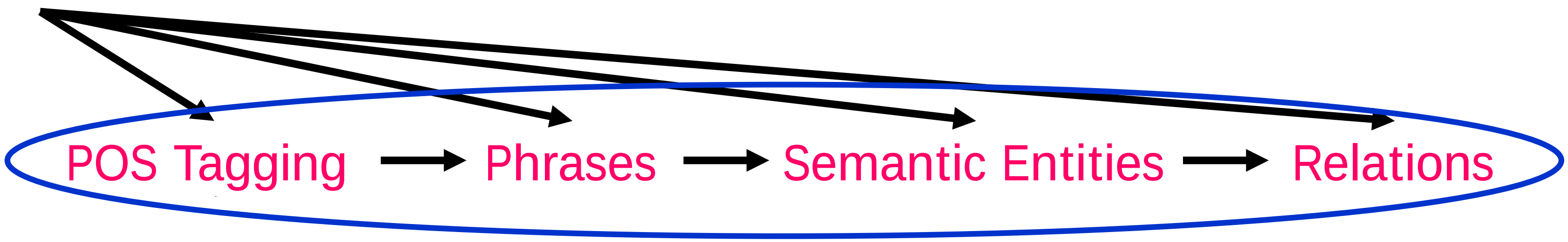
- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and down stream decisions often interact with previous decisions.
  - Leads to propagation of errors
  - Occasionally, later stages could be used to correct earlier errors.
- But, there are good reasons to use pipelines
  - Putting everything in one basket may not be right
  - How about choosing some stages and think about them jointly?

# Pipeline



- **Most problems are not single classification problems**

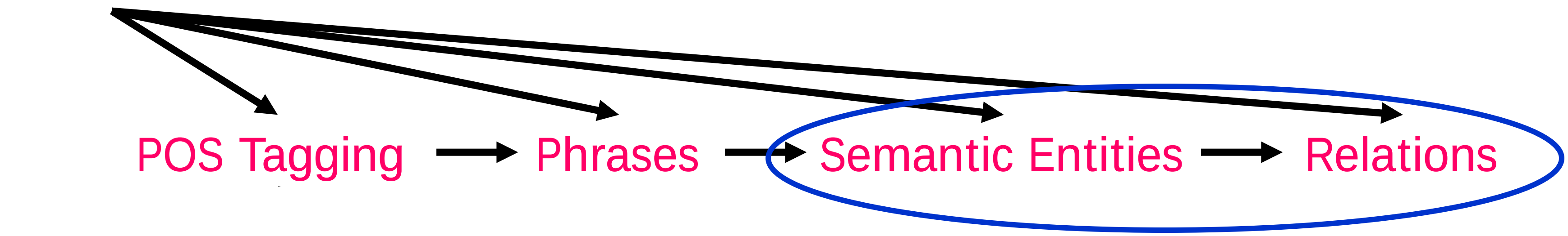
Raw Data




- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and down stream decisions often interact with previous decisions.
  - Leads to propagation of errors
  - Occasionally, later stages could be used to correct earlier errors.
- But, there are good reasons to use pipelines
  - Putting everything in one basket may not be right
  - How about choosing some stages and think about them jointly?

# Pipeline

- **Most problems are not single classification problems**

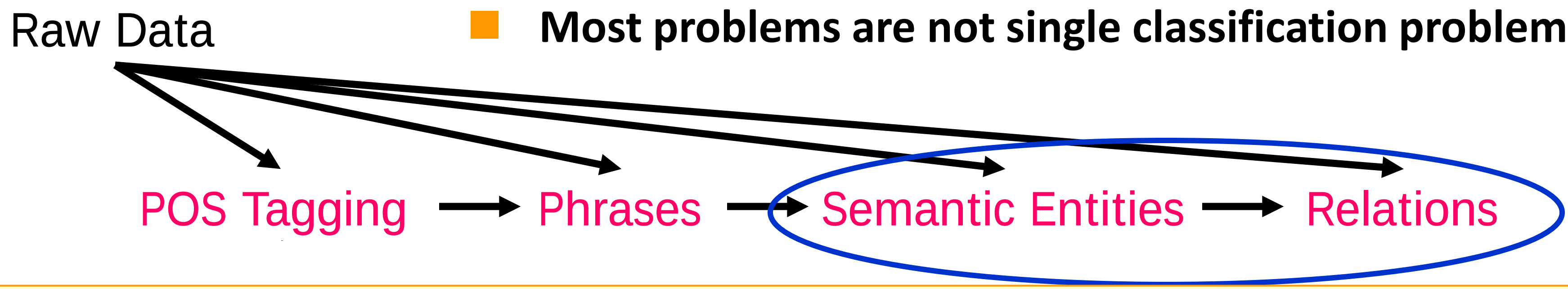



- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and down stream decisions often interact with previous decisions.
  - Leads to propagation of errors
  - Occasionally, later stages could be used to correct earlier errors.
- But, there are good reasons to use pipelines
  - Putting everything in one basket may not be right
  - How about choosing some stages and think about them jointly?

# Pipeline

Motivation I

- **Most problems are not single classification problems**




Either way, we need a way to **learn models** and **make predictions (inference; decoding) that assign** values to multiple interdependent variables

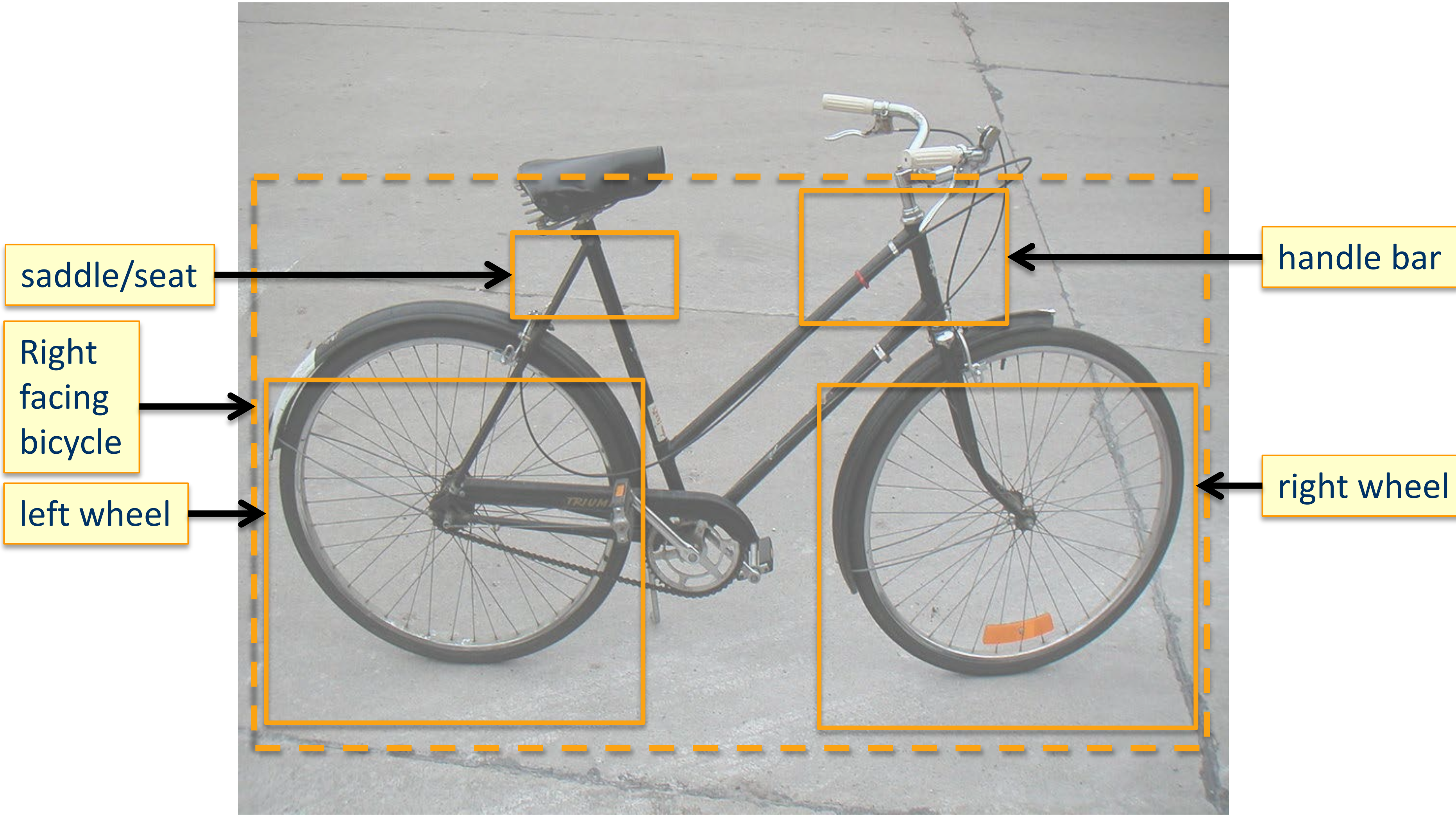
- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and down stream decisions often interact with previous decisions.
  - Leads to propagation of errors
  - Occasionally, later stages could be used to correct earlier errors.
- But, there are good reasons to use pipelines
  - Putting everything in one basket may not be right
  - How about choosing some stages and think about them jointly?

# Example 2: Object detection

Right facing bicycle

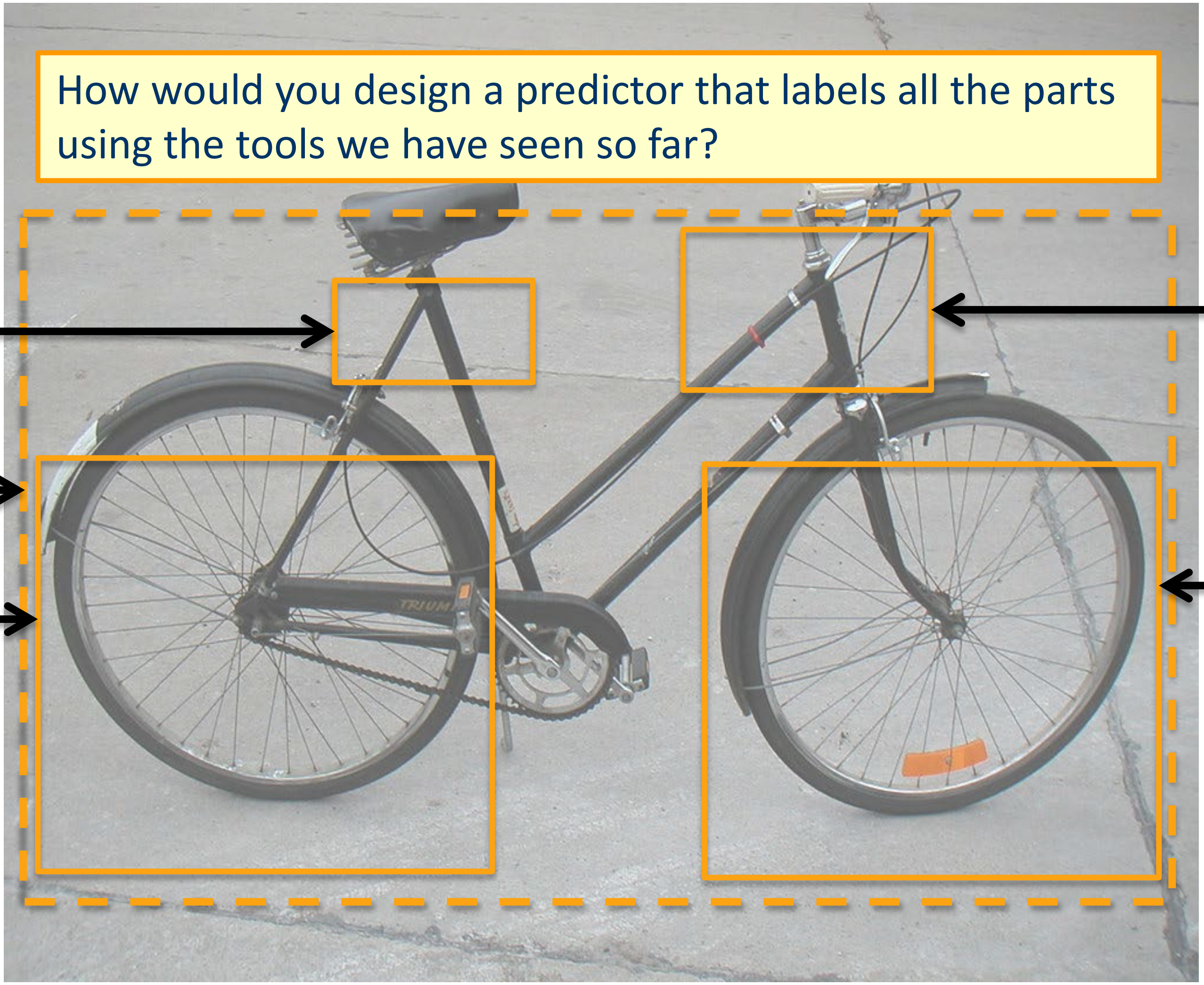
Photo by Andrew Dressel - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0

# Example 2: Object detection

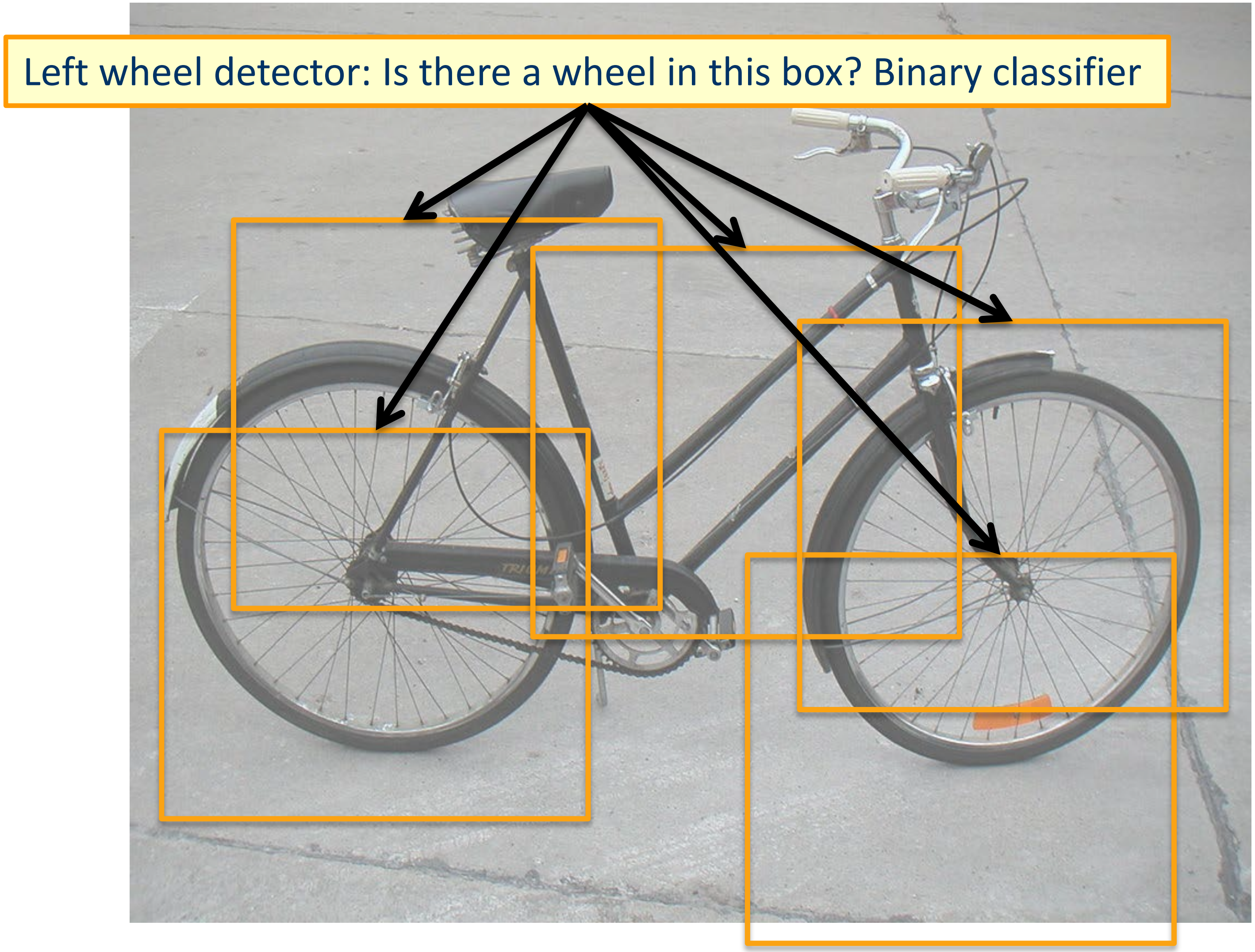


# Example 2: Object detection

How would you design a predictor that labels all the parts using the tools we have seen so far?



# One approach to build this structure



# One approach to build this structure

1. Left wheel detector

2. Right wheel detector

3. Handle bar detector
4. Seat detector

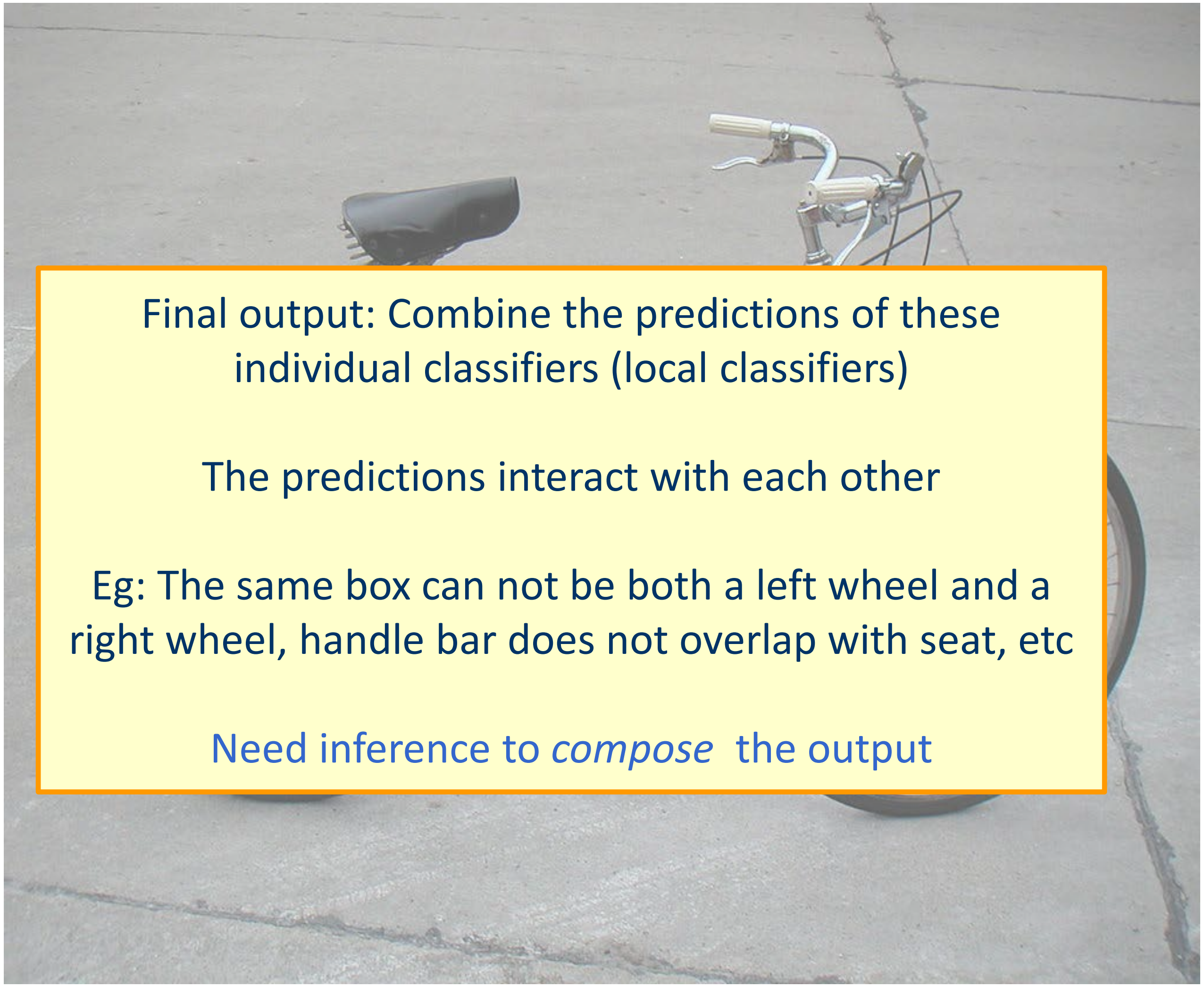
# One approach to build this structure

1. Left wheel detector

2. Right wheel detector

3. Handle bar detector

4. Seat detector



Final output: Combine the predictions of these individual classifiers (local classifiers)

The predictions interact with each other

Eg: The same box can not be both a left wheel and a right wheel, handle bar does not overlap with seat, etc

Need inference to *compose* the output

# Task of Interests: Structured Output

- For each instance, assign values to a set of variables
- Output variables depend on each other

# Task of Interests: Structured Output

- For each instance, assign values to a set of variables
- Output variables depend on each other
- Common NLP tasks
  - Parsing; Semantic Parsing; Summarization; Co-reference…
- Common Information Extraction Tasks:
  - Entities, Relations,…
- Common Vision Task:
  - Parsing objects; scene segmentation and interpretation,….

# Task of Interests: Structured Output

- For each instance, assign values to a set of variables
- Output variables depend on each other
- Common NLP tasks
  - Parsing; Semantic Parsing; Summarization; Co-reference…
- Common Information Extraction Tasks:
  - Entities, Relations,…
- Common Vision Task:
  - Parsing objects; scene segmentation and interpretation,….
- Many "pure" machine learning approaches exist
  - Hidden Markov Models (HMMs); CRFs […there are special cases…]
  - Structured Perceptrons and SVMs… [… to be discussed later]
- However, …

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

[AUTHOR]

[TITLE]

[EDITOR]

[BOOKTITLE]

[TECH-REPORT]

[INSTITUTION]

[DATE]

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language.  PhD thesis. DIKU , University of Copenhagen, May 1994 .

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | .  PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language.  PhD thesis. DIKU , University of Copenhagen, May 1994 .

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | .  PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

**Prediction result of a trained HMM**

[AUTHOR] Lars Ole Andersen . Program analysis and

[TITLE] specialization for the

[EDITOR] C

[BOOKTITLE] Programming language

[TECH-REPORT] . PhD thesis .

[INSTITUTION] DIKU , University of Copenhagen , May

[DATE] 1994 .

Violates lots of natural constraints!

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples

Increasing the model complexity

Increase difficulty of Learning

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples
  - What if we only have a few labeled examples?

Increasing the model complexity

Increase difficulty of Learning

Can we keep the learned model simple and still make expressive decisions?

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples
  - What if we only have a few labeled examples?

Increasing the model complexity

Increase difficulty of Learning

Can we keep the learned model simple and still make expressive decisions?

- Instead:
  - Constrain the output to make sense – satisfy our output expectations
  - Push the (simple) model in a direction that makes sense – minimally violates our expectations.

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

Easy to express pieces of "knowledge"

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

Easy to express pieces of “knowledge”

Non Propositional; May use Quantifiers

# Information Extraction with Expectation Constraints

- Adding constraints, we get correct results!
  - Without changing the model

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

- Adding constraints, we get correct results!
  - Without changing the model

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

# Information Extraction with Expectation Constraints

- Adding constraints, we get correct results!
  - Without changing the model

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

We introduce the **Constrained Conditional Models formulation** which allows:

- Learning a simple model
- Making decisions with a more complex model
  - Some of the structure imposes externally/declaratively
- Accomplished by directly incorporating constraints to bias/re-rank decisions made by the simpler model

# Constrained Conditional Models

$$\mathbf{y} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^{\mathsf{T}}\mathcal{C}(\mathbf{x}, \mathbf{y})$$

Weight Vector for "local" models

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T C(\mathbf{x}, \mathbf{y})$$

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

(Soft) constraints component

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

# Constrained Conditional Models

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

Penalty for violating the constraint.

(Soft) constraints component

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal" assignment

# Constrained Conditional Models

Penalty for violating the constraint.

$$\mathbf{y} = \mathrm{argmax}_{y \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

(Soft) constraints component

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal" assignment

How to solve?

This is an Integer Linear Program

Solving using ILP packages gives an exact solution.

Cutting Planes, Dual Decomposition & other search techniques are possible

**Amortized ILP inference Scheme**

How to train?

**Training** is learning the objective function

Decompose objective? Decouple? Train Jointly?

How to exploit the structure to minimize supervision?

**New (joint and distributed algorithms**

# Structured Prediction: Inference

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** y = $\{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s

- Inference is expressed as a maximization of a ***scoring function***

$$y' = \text{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x,y)$$

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s
- Inference is expressed as a maximization of a ***scoring function***

$$y' = \text{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x,y)$$

Joint features on inputs and outputs

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s
- Inference is expressed as a maximization of a ***scoring function***

$$y' = \text{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x,y)$$

Joint features on inputs and outputs

Feature Weights (estimated during learning)

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s

- Inference is expressed as a maximization of a ***scoring function***

$$y' = \text{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x,y)$$

Set of allowed structures

Feature Weights (estimated during learning)

Joint features on inputs and outputs

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s

- Inference is expressed as a maximization of a ***scoring function***

$$y' = \mathrm{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x, y)$$

Set of allowed structures

Feature Weights (estimated during learning)

Joint features on inputs and outputs

- Inference requires, in principle, touching all $y \in Y$ at decision time, when we are given $x \in X$ and attempt to determine the best $y \in Y$ for it, given $w$

# Structured Prediction: Inference

Placing in context: a very high level view of what you will see next

- Inference: given input **x** (a document, a sentence),

  predict **the best structure** $y = \{y_1, y_2, \ldots, y_n\} \in Y$ (entities & relations)

  - Assign values to the $y_1, y_2, \ldots, y_n$, accounting for **dependencies among** $y_i$s
- Inference is expressed as a maximization of a ***scoring function***

$$y' = \text{argmax}_{y \in \mathcal{Y}} w^T \phi(x,y)$$

Set of allowed structures

Feature Weights (estimated during learning)

Joint features on inputs and outputs

- Inference requires, in principle, touching all $y \in Y$ at decision time, when we are given $x \in X$ and attempt to determine the best $y \in Y$ for it, given $w$
  - For some structures, inference is computationally easy.
  - Eg: Using the Viterbi algorithm
  - In general, NP-hard (can be formulated as an ILP)

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}

  find a scoring function w that minimizes empirical loss.

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.

- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.

- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

**Score of annotated structure** $\geq$ **Score of any other structure** $+$ **Penalty for predicting other structure**

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.

- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

$$\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$$

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.
- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

$$\forall y \quad \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$$

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.
- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

$$\forall y \quad \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$$

- We call these conditions the learning constraints.

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.

- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

$$\forall y \quad \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$$

- We call these conditions the learning constraints.

- In most learning algorithms used today, the update of the weight vector w is done in an on-line fashion,
  - Think about it as Perceptron; this procedure applies to Structured Perceptron, CRFs, Linear Structured SVM

# Structured Prediction: Learning

- Learning: given a set of structured examples {(x,y)}
  find a scoring function w that minimizes empirical loss.
- Learning is thus driven by the attempt to find a weight vector w such that for each given annotated example $(x_i, y_i)$:

$$\forall y \quad \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}, \mathbf{y}_i)$$

- We call these conditions the learning constraints.
- In most learning algorithms used today, the update of the weight vector w is done in an on-line fashion,
  - Think about it as Perceptron; this procedure applies to Structured Perceptron, CRFs, Linear Structured SVM
- W.l.o.g. (almost) we can thus write the generic structured learning algorithm as follows:

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do: (with the current weight vector $w$)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = \text{argmax}_{y \in \mathcal{Y}}\, w^T \phi(x_i, y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update $w$ on this example
- EndFor

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = argmax_{y \in \mathcal{Y}} w^T \phi(x_i, y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = argmax_{y \in \mathcal{Y}} \, w^T \phi ( x_i , y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do: (with the current weight vector $w$)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = \text{argmax}_{y \in \mathcal{Y}} w^T \phi(x_i, y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update $w$ on this example
- EndFor

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = \text{argmax}_{y \in \mathcal{Y}}\, w^T \phi(x_i, y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

In the structured case, prediction (inference) is often intractable but needs to be done many times

# Structured Prediction: Learning Algorithm

Solution I: decompose the scoring function to EASY and HARD parts

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}} \; w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo

# Structured Prediction: Learning Algorithm

Solution I: decompose the scoring function to EASY and HARD parts

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = argmax_{y \in \mathcal{Y}}\ w_{EASY}^T\, \phi_{EASY}(x_i, y) + w_{HARD}^T\, \phi_{HARD}(x_i, y)}$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo

EASY: could be feature functions that correspond to an HMM, a linear CRF, or even $\phi_{EASY}(x,y) = \phi(x)$, omiting dependence on y, corresponding to classifiers. May not be enough if the HARD part is still part of each inference step.

# Structured Prediction: Learning Algorithm

Solution II: Disregard some of the dependencies: **assume a simple model.**

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = \text{argmax}_{y \in \mathcal{Y}}\ w_{EASY}^T\ \phi_{EASY}(x_i, y) + w_{HARD}^T\ \phi_{HARD}(x_i, y)}$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo

# Structured Prediction: Learning Algorithm

Solution II: Disregard some of the dependencies: **assume a simple model.**

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}} \; w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo

# Structured Prediction: Learning Algorithm

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}}\ w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo

# Structured Prediction: Learning Algorithm

Solution III: Disregard some of the dependencies during learning; take into account at decision time

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}}\ w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo
- $y_i' = \text{argmax}_{y \in \mathcal{Y}}\ w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$

# Structured Prediction: Learning Algorithm

Solution III: Disregard some of the dependencies during learning; take into account at decision time

- For each example $(x_i, y_i)$
- Do:
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}} \; w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$
  - **Check** the learning constraint
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndDo
- $y_i' = \text{argmax}_{y \in \mathcal{Y}} \; w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)$

**This is the most commonly used solution in NLP today**

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}, \mathbf{y})$$

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^{T}\phi(\mathbf{x}, \mathbf{y})$$

Features, classifiers; log-linear models (HMM, CRF) or a combination

# Constrained Conditional Models

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

# Constrained Conditional Models

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

# Constrained Conditional Models

Penalty for violating the constraint.

$$\mathbf{y} = \mathbf{argmax}_{y \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

# Constrained Conditional Models

Penalty for violating the constraint.

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^{\mathrm{T}}\mathcal{C}(\mathbf{x}, \mathbf{y})$$

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

Knowledge component: (Soft) constraints

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?

# Constrained Conditional Models

Penalty for violating the constraint.

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?
- A way to push the learned model to **satisfy our output expectations** (or expectations from a latent representation)
  - [CoDL, Chang, Ratinov, Roth (07, 12); Posterior Regularization, Ganchev et. al (10); Unified EM (Samdani & Roth(12)]

# Constrained Conditional Models

Penalty for violating the constraint.

$$\arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{p\in\Gamma_{\mathbf{x}}} \mathbf{1}_{[Y_p=\mathbf{y}_p]}\mathbf{w}^T\Phi_p\left(\mathbf{x},\mathbf{y}_p\right)$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?
- A way to push the learned model to **satisfy our output expectations** (or expectations from a latent representation)
  - [CoDL, Chang, Ratinov, Roth (07, 12); Posterior Regularization, Ganchev et. al (10); Unified EM (Samdani & Roth(12)]

# Constrained Conditional Models

Any MAP problem w.r.t. any probabilistic model, can be formulated as an **Integer Linear Program (ILP)** Roth+ 04, Taskar 04]

$$\arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{p\in\Gamma_{\mathbf{x}}} \mathbf{1}_{[Y_p=\mathbf{y}_p]} \mathbf{w}^T \Phi_p(\mathbf{x}, \mathbf{y}_p)$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?
- A way to push the learned model to **satisfy our output expectations** (or expectations from a latent representation)
  - [CoDL, Chang, Ratinov, Roth (07, 12); Posterior Regularization, Ganchev et. al (10); Unified EM (Samdani & Roth(12)]

# Constrained Conditional Models

Any MAP problem w.r.t. any probabilistic model, can be formulated as an **Integer Linear Program (ILP)** Roth+ 04, Taskar 04]

**Variables are "parts"**

$$\arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{p\in\Gamma_{\mathbf{x}}} \mathbf{1}_{[Y_p=\mathbf{y}_p]} \mathbf{w}^T \Phi_p(\mathbf{x}, \mathbf{y}_p)$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?
- A way to push the learned model to **satisfy our output expectations** (or expectations from a latent representation)
  - [CoDL, Chang, Ratinov, Roth (07, 12); Posterior Regularization, Ganchev et. al (10); Unified EM (Samdani & Roth(12)]

# Constrained Conditional Models

Any MAP problem w.r.t. any probabilistic model, can be formulated as an **Integer Linear Program (ILP)** Roth+ 04, Taskar 04]

Variables are "parts"

$$\arg\max_{\mathbf{y}\in\mathcal{Y}} \sum_{p\in\Gamma_{\mathbf{x}}} \mathbf{1}_{[Y_p=\mathbf{y}_p]}\mathbf{w}^T\Phi_p\left(\mathbf{x},\mathbf{y}_p\right)$$

Knowledge component: (Soft) constraints

Weight Vector for "local" models

Features, classifiers; log-linear models (HMM, CRF) or a combination

How far y is from a "legal/expected" assignment

- Training: learning the objective function (**w, u**)
  - Decouple? Decompose? Force **u** to model hard constraints?
- A way to push the learned model to **satisfy our output expectations** (or expectations from a latent representation)
  - [CoDL, Chang, Ratinov, Roth (07, 12); Posterior Regularization, Ganchev et. al (10); Unified EM (Samdani & Roth(12)]
- The benefits of thinking about it as an ILP are conceptual and computational.

# Examples: CCM Formulations

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^{\mathrm{T}}\mathcal{C}(\mathbf{x}, \mathbf{y})$$

# Examples: CCM Formulations

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

# Examples: CCM Formulations

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

# Examples: CCM Formulations

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T\mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

# Examples: CCM Formulations

$$\mathbf{y} = \mathbf{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

$$\mathbf{y} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

The second part of the tutorial is on how to learn

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

Formulate NLP Problems as ILP problems (inference may be done otherwise)

1. Sequence tagging (HMM/CRF + Global constraints)
2. Sentence Compression (Language Model + Global Constraints)
3. SRL (Independent classifiers + Global Constraints)

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

$$\mathbf{y} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T\mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

Formulate NLP Problems as ILP problems (inference may be done otherwise)

1. Sequence tagging (HMM/CRF + Global constraints)
2. Sentence Compression (Language Model + Global Constraints)
3. SRL (Independent classifiers + Global Constraints)

**Sequential Prediction**

HMM/CRF based:

Argmax $\sum \lambda_{ij}$ $x_{ij}$

**Knowledge/Linguistics Constraints**

Cannot have both A states and B states in an output sequence.

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

The second part of the tutorial is on how to learn

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

Formulate NLP Problems as ILP problems (inference may be done otherwise)

1. Sequence tagging (HMM/CRF + Global constraints)
2. Sentence Compression (Language Model + Global Constraints)
3. SRL (Independent classifiers + Global Constraints)

Sentence Compression/Summarization:

Language Model based:

Argmax $\sum \lambda_{ijk} x_{ijk}$

Knowledge/Linguistics Constraints

If a modifier chosen, include its head

If verb is chosen, include its arguments

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

$$\mathbf{y} = \mathrm{argmax}_{y \in \mathcal{Y}}\ \mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T\mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

While $\phi$**(x, y) and** $\mathcal{C}$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

Formulate NLP Problems as ILP problems (inference may be done otherwise)

1. Sequence tagging (HMM/CRF + Global constraints)
2. Sentence Compression (Language Model + Global Constraints)
3. SRL (Independent classifiers + Global Constraints)

Sentence Compression/Summarization:

Language Model based:

Argmax $\sum \lambda_{ijk} x_{ijk}$

Knowledge/Linguistics Constraints

If a modifier chosen, include its head

If verb is chosen, include its arguments

# Examples: CCM Formulations

The third part of the tutorial is on how to do inference

The second part of the tutorial is on how to learn

$$y = \text{argmax}_{y \in \mathcal{Y}}\ w^T\phi(x, y) + u^T C(x, y)$$

While $\phi$**(x, y) and** $C$**(x, y)** could be the same; we want C(x, y) to express high level declarative knowledge over the statistical models.

Formulate NLP Problems as ILP problems (inference may be done otherwise)

1. Sequence tagging (HMM/CRF + Global constraints)
2. Sentence Compression (Language Model + Global Constraints)
3. SRL (Independent classifiers + Global Constraints)

Constrained Conditional Models Allow:

- Decouple complexity of the learned model from that of the desired output
- Learn a simple model (multiple; pipelines); reason with a complex one.
- Accomplished by incorporating constraints to bias/re-rank global decisions to satisfy (minimally violate) expectations.

# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**

**$[\text{I}]_{A0}$ *left* $[\text{my pearls}]_{A1}$ $[\text{to my daughter}]_{A2}$ $[\text{in my will}]_{AM\text{-}LOC}$ .**

- ***A0*** Leaver
- ***A1*** Things left
- ***A2*** Benefactor
- ***AM-LOC*** Location

**I *left* my pearls to my daughter in my will .**

# Semantic Role Labeling (SRL)

**Archetypical Information Extraction Problem**: E.g., Concept Identification and Typing, Event Identification, etc.

**I *left* my pearls to my daughter in my will .**

**[I]$_{A0}$ *left* [my pearls]$_{A1}$ [to my daughter]$_{A2}$ [in my will]$_{AM\text{-}LOC}$ .**

- ***A0*** Leaver
- ***A1*** Things left
- ***A2*** Benefactor
- ***AM-LOC*** Location

**I *left* my pearls to my daughter in my will .**

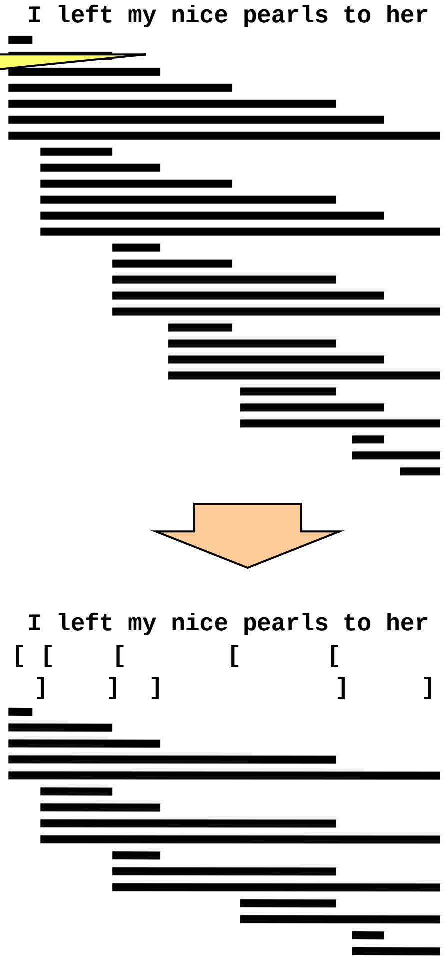
# Algorithmic Approach

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output

# Algorithmic Approach

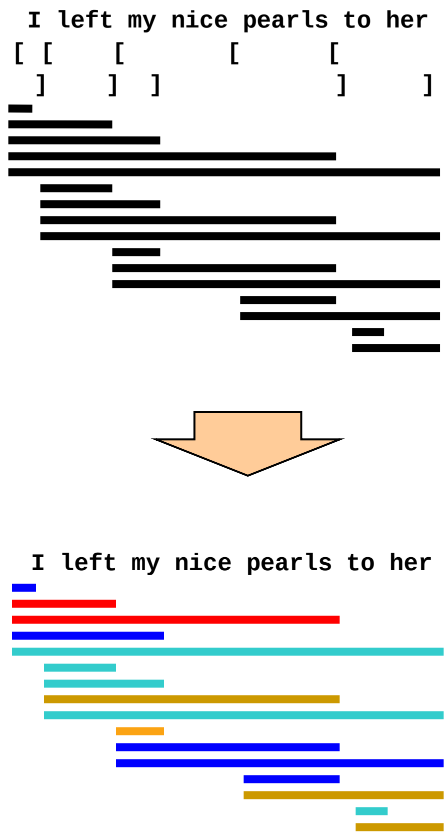
candidate arguments

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output



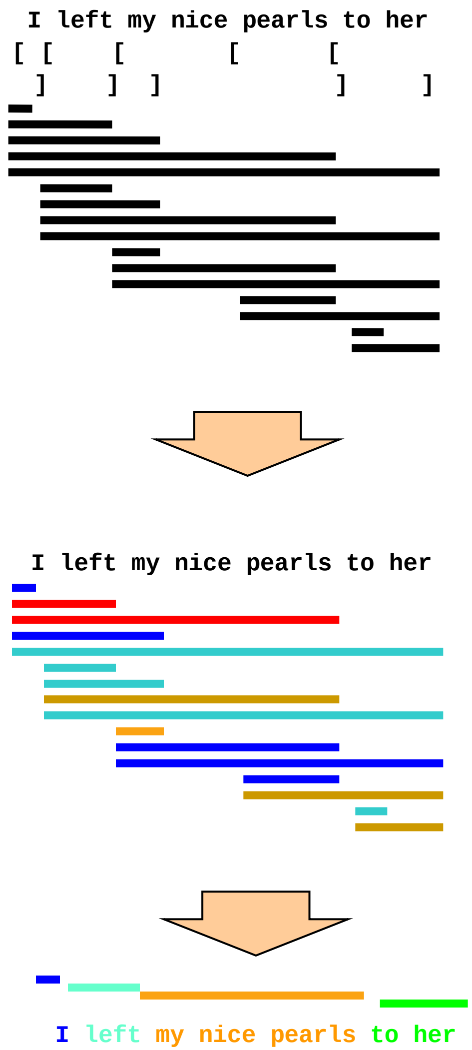
# Algorithmic Approach

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output



# Algorithmic Approach

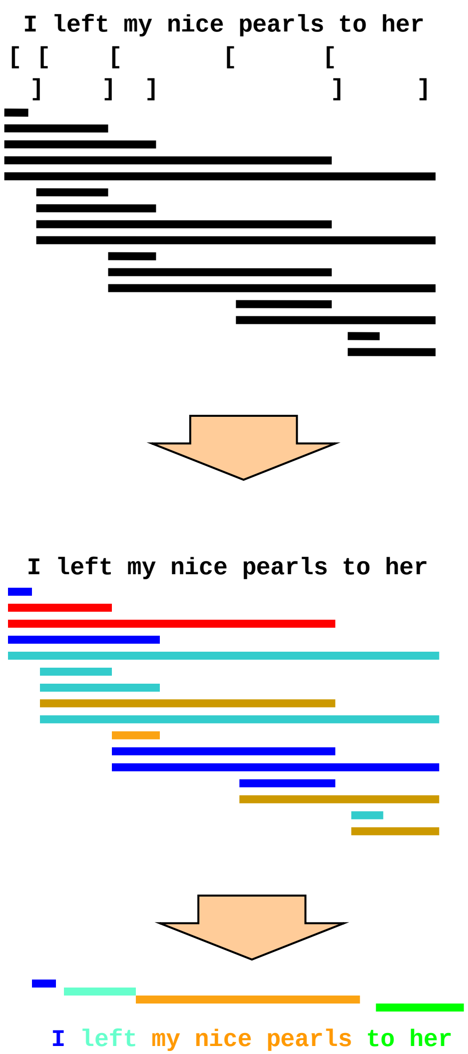
- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output



# Algorithmic Approach

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer t utput

One inference problem for each verb predicate.



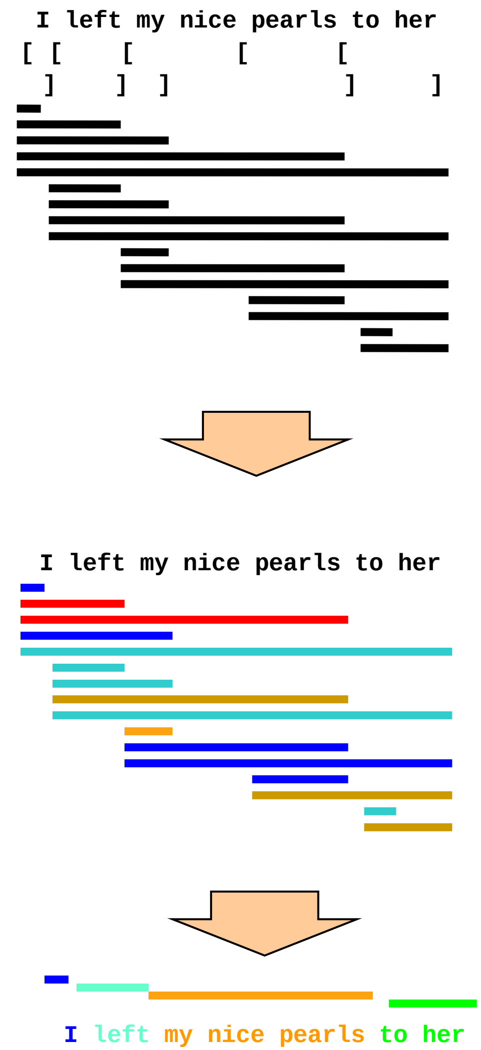
# Algorithmic Approach

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given

argmax $\sum_{a,t} y^{a,t} c^{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}$

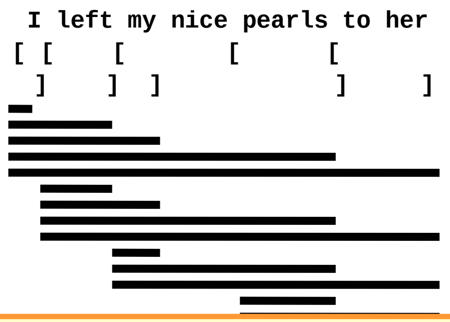
Subject to:

- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,....



# Algorithmic Approach

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given

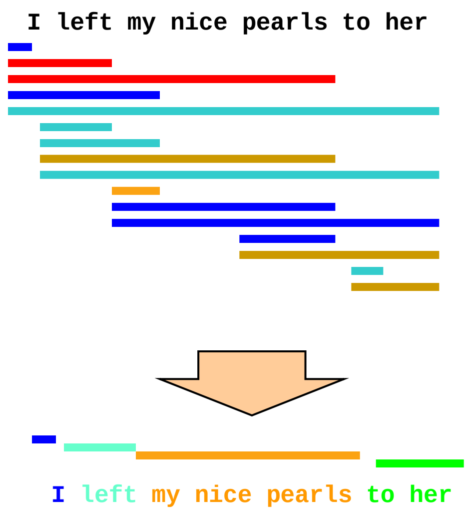



Variable $y^{a,t}$ indicates whether candidate argument a is assigned a label t.
$c^{a,t}$ is the corresponding model score

$$\text{argmax} \sum_{a,t} y^{a,t} c^{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}$$

Subject to:

- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,….



# Algorithmic Approach

- Identify argument candid
  - Pruning [Xue&Palmer, EM
  - Argument Identifier
    - **Binary classification**
- Classify argument candida
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given

No duplicate argument classes

$$\forall i, \sum_{y \in \mathcal{Y}} 1_{\{y_i = y\}} = 1$$

Unique labels

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \leq 1$$

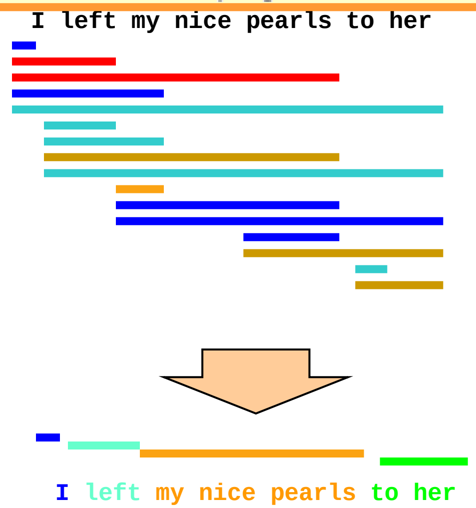
$$\forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1_{\{y_i = y = \text{“R-Ax”}\}} \leq \sum_{i=0}^{n-1} 1_{\{y_i = \text{“Ax”}\}}$$

$$\forall j, y \in \mathcal{Y}_C, 1_{\{y_j = y = \text{“C-Ax”}\}} \leq \sum_{i=0}^{j} 1_{\{y_i = \text{“Ax”}\}}$$

argmax $\sum_{a,t} y^{a,t} c^{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}$

Subject to:

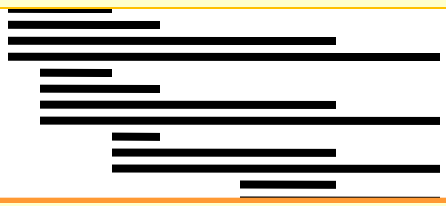
- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,….



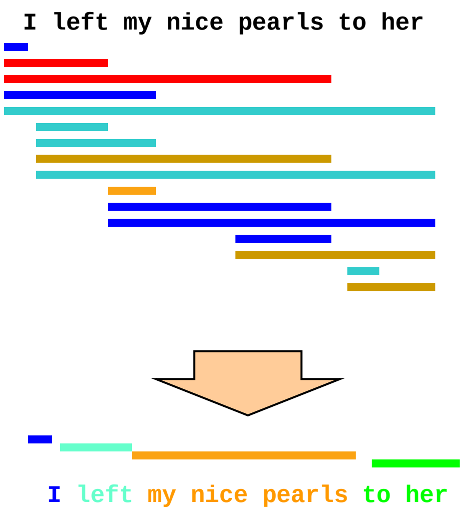
# Algorithmic Approach

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given



Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score




$\text{argmax} \sum_{a,t} y^{a,t} c^{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}$

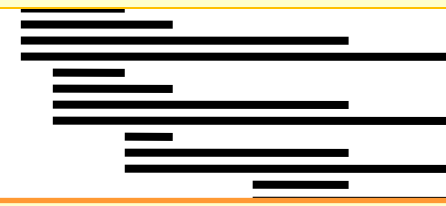
Subject to:

- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,….

# Algorithmic Approach

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

- Identify argument candidates
  - Pruning [Xue&Palmer, EMNLP'04]
  - Argument Identifier
    - **Binary classification**
- Classify argument candidates
  - Argument Classifier
    - **Multi-class classification**
- Inference
  - Use the estimated probability distribution given

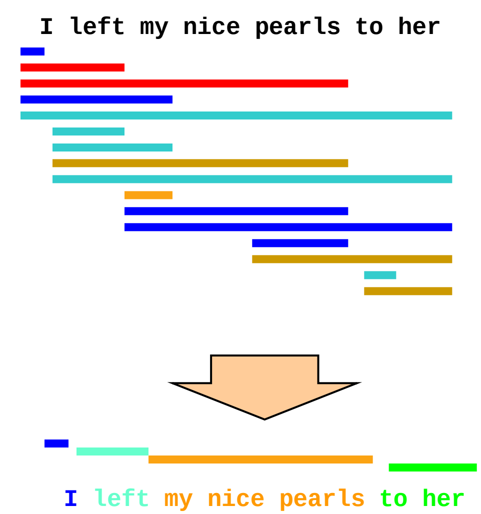


Variable $y^{a,t}$ indicates whether candidate argument a is assigned a label t.
$c^{a,t}$ is the corresponding model score

$$\text{argmax} \sum_{a,t} y^{a,t} c^{a,t} = \sum_{a,t} 1_{a=t} \, c_{a=t}$$

Subject to:

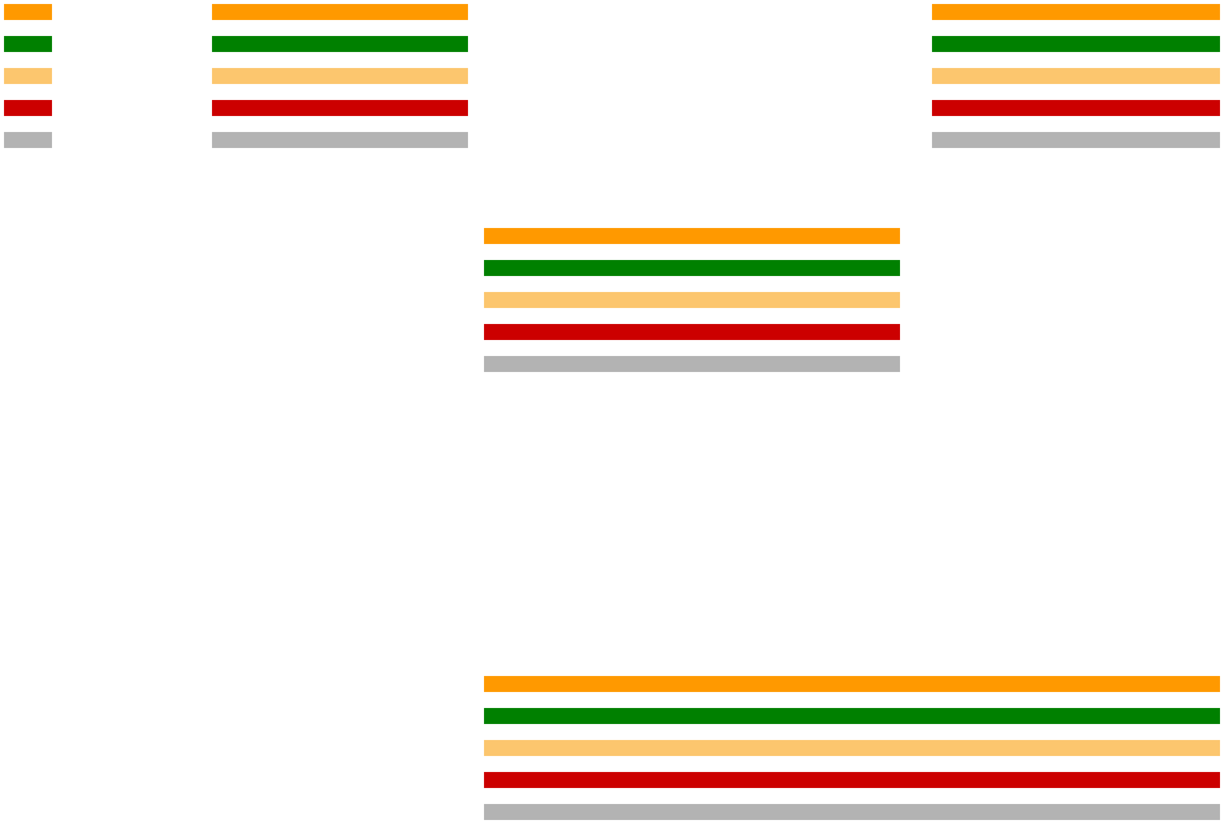
- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,….




Use the **pipeline architecture's simplicity** while **maintaining uncertainty**: keep probability distributions over decisions & use global inference at decision time.

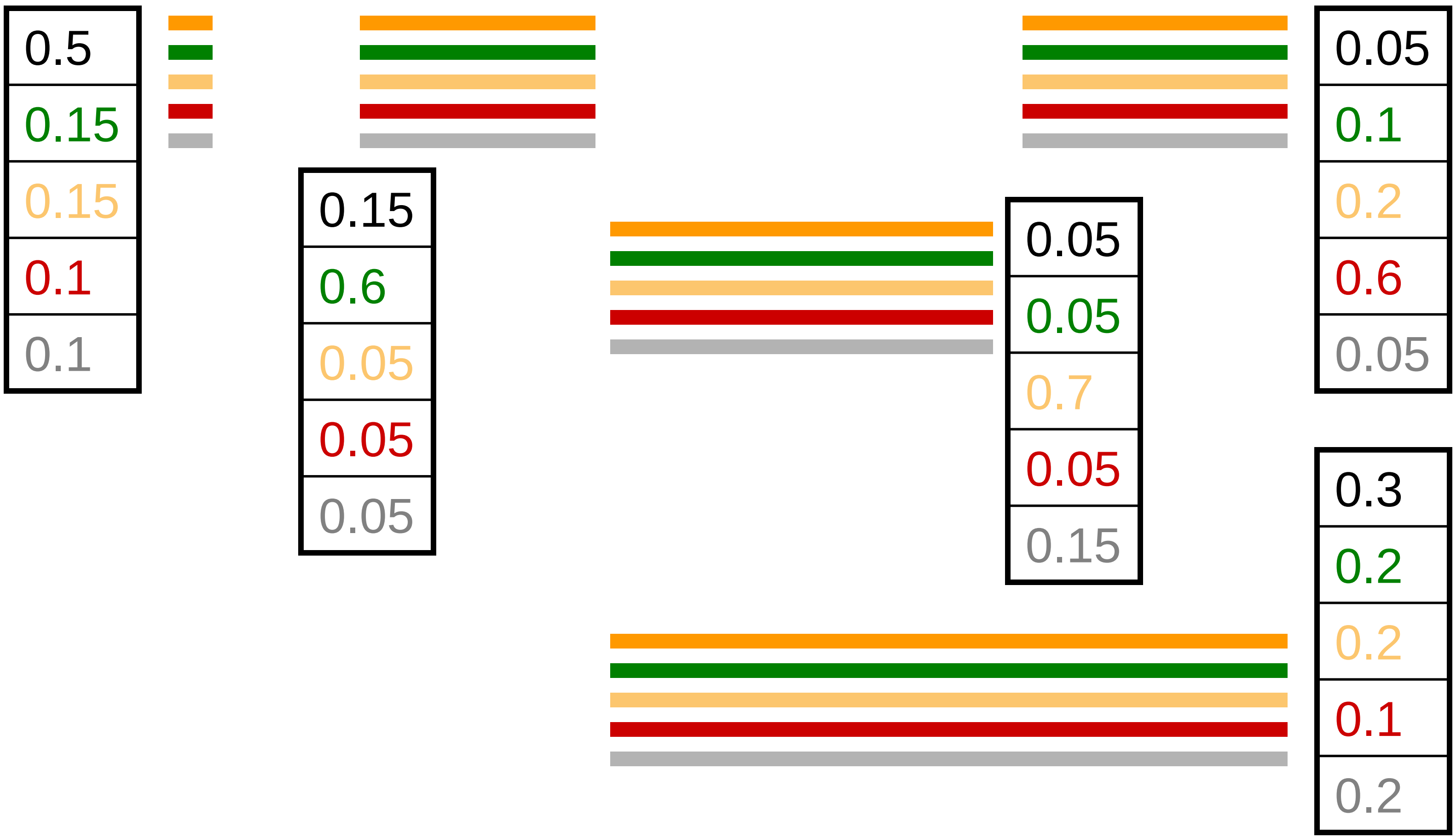
# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**



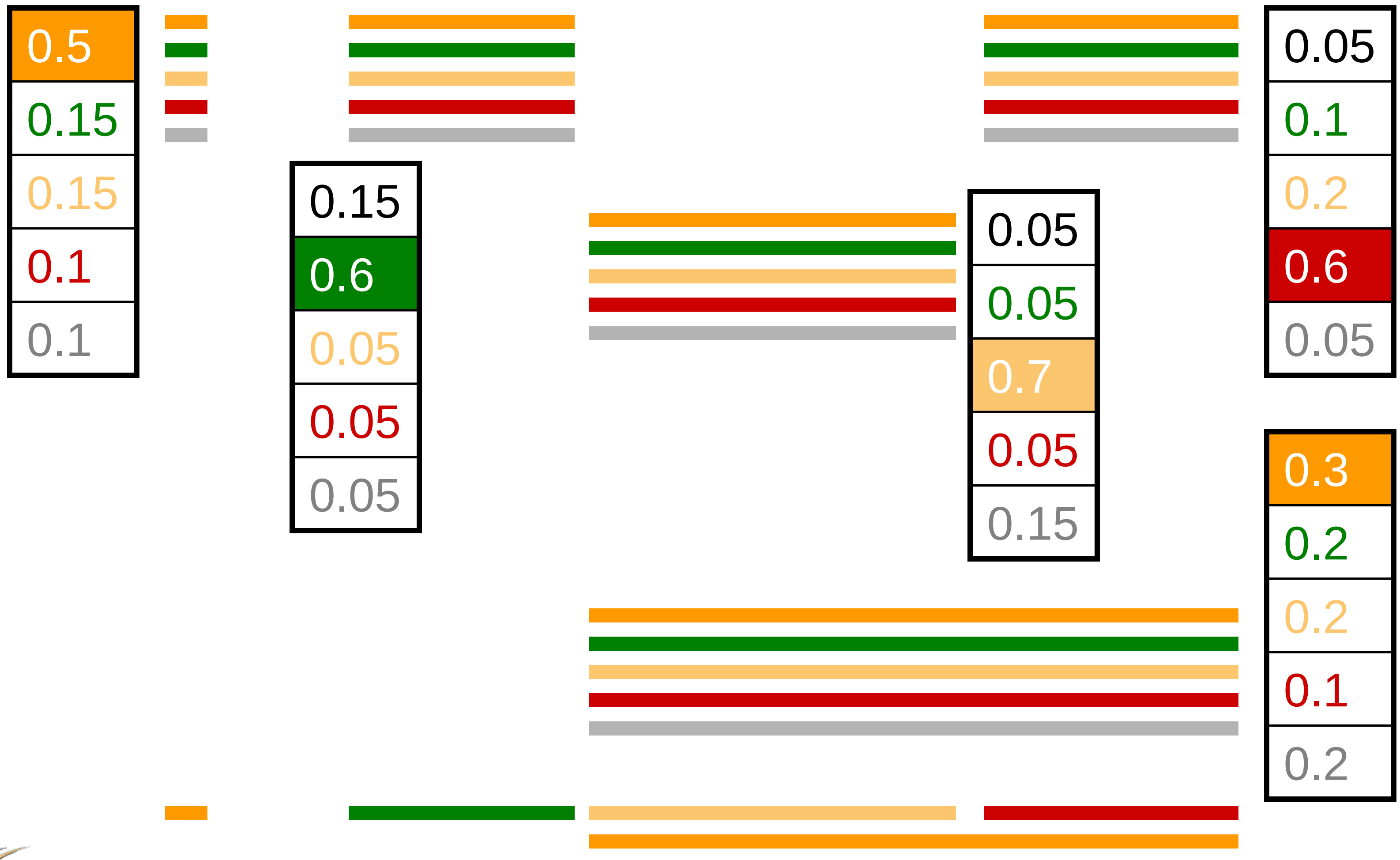
# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**



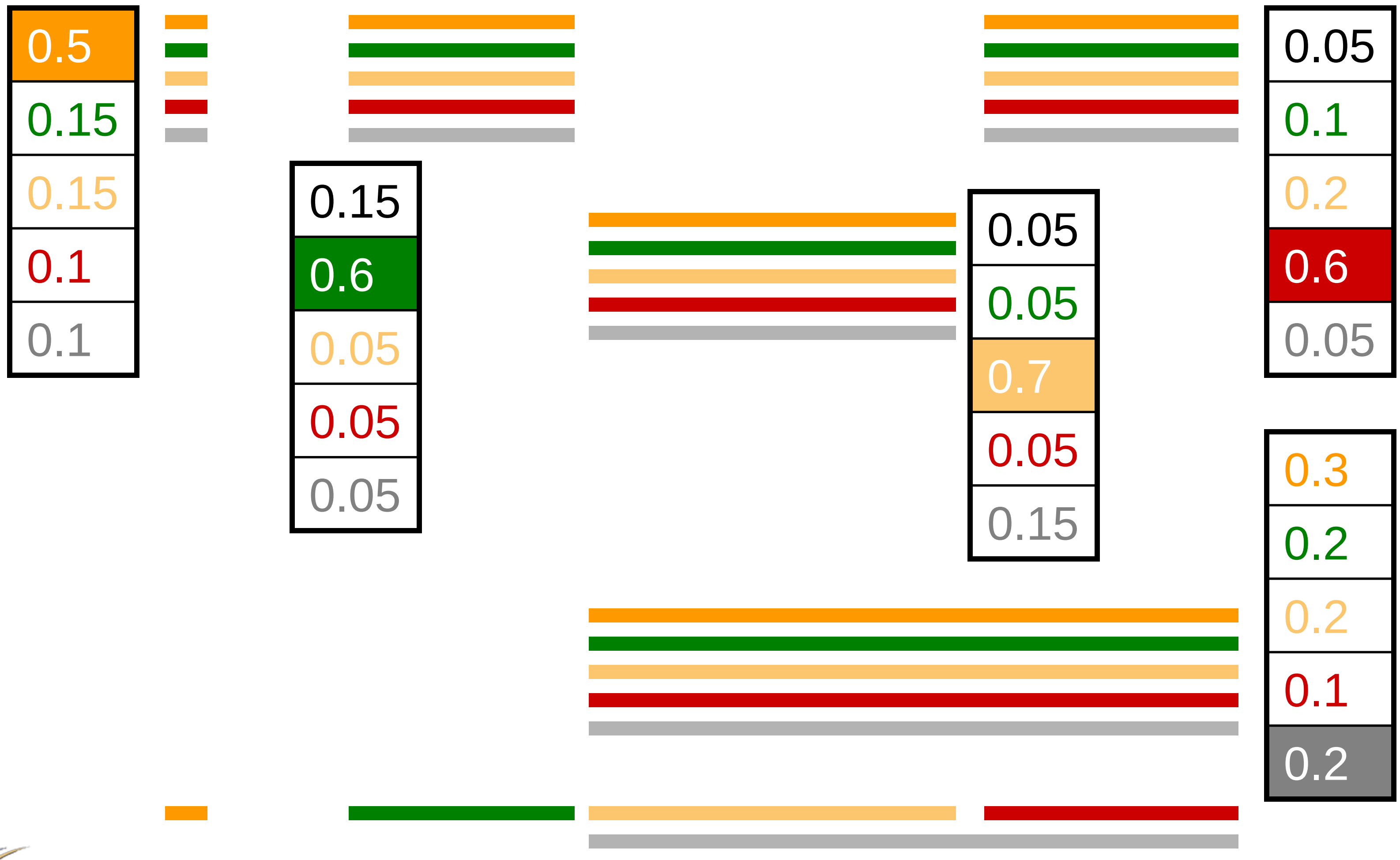
# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**



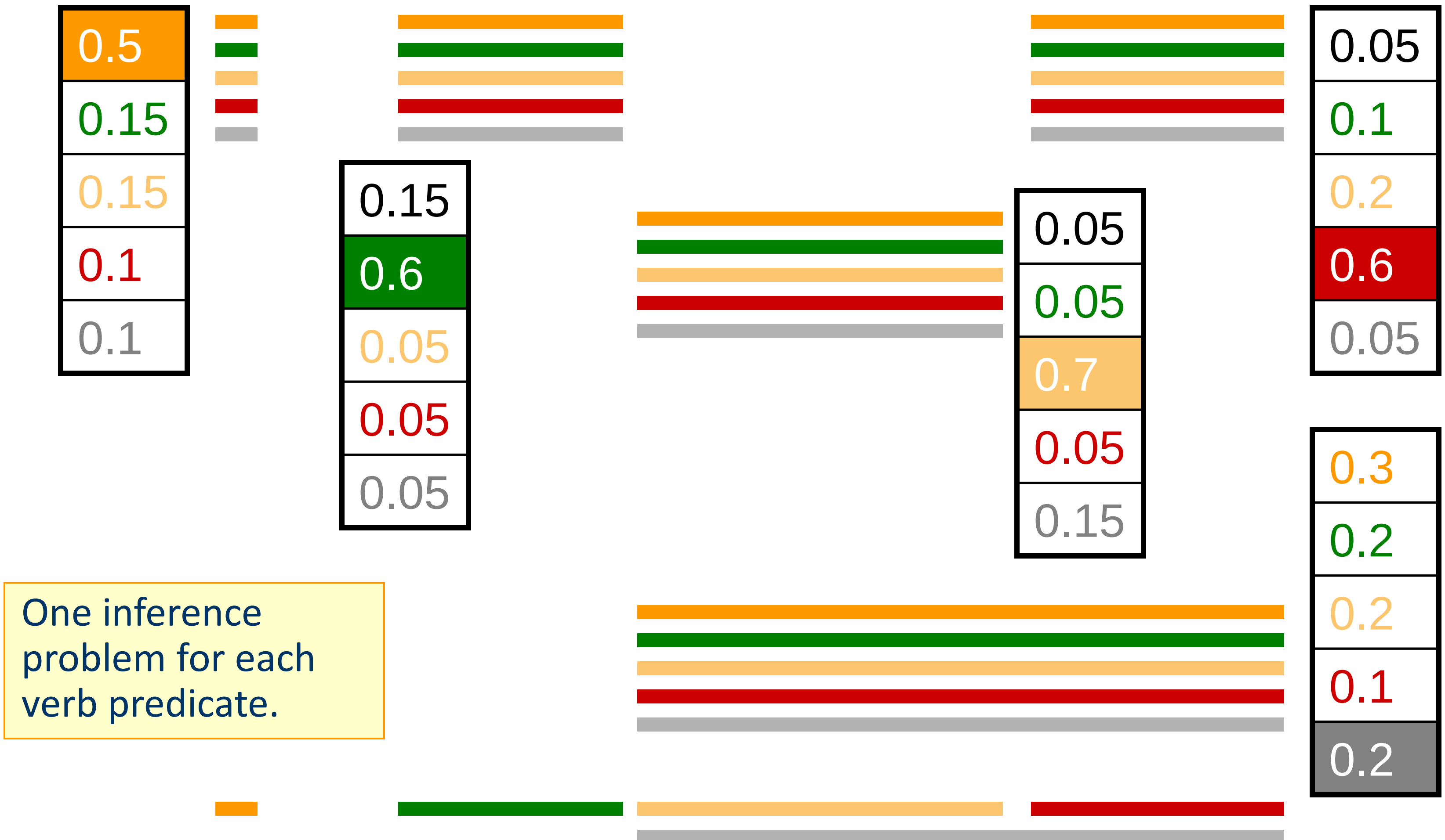
# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**



# Semantic Role Labeling (SRL)

**I *left* my pearls to my daughter in my will .**



# Constraints

- No duplicate argument classes
- Reference-Ax

  If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

  If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type  B**

# Constraints

- No duplicate argument classes

Any Boolean rule can be encoded as a set of linear inequalities.

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

# Constraints

- No duplicate argument classes

Any Boolean rule can be encoded as a set of linear inequalities.

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

Universally quantified rules

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

# Constraints

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.

- No duplicate argument classes

Any Boolean rule can be encoded as a set of linear inequalities.

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type  B**

# Constraints

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.

- No duplicate argument classes

Any Boolean rule can be encoded as a set of linear inequalities.

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

# Constraints

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.

Any Boolean rule can be encoded as a set of linear inequalities.

- No duplicate argument classes

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \leq 1$$

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

# Constraints

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.

Any Boolean rule can be encoded as a set of linear inequalities.

- No duplicate argument classes

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \leq 1$$

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

$$\forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1_{\{y_i = y = \text{“R-Ax”}\}} \leq \sum_{i=0}^{n-1} 1_{\{y_i = \text{“Ax”}\}}$$

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

# Constraints

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.

Any Boolean rule can be encoded as a set of linear inequalities.

- No duplicate argument classes

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \leq 1$$

- Reference-Ax

If there is an Reference-Ax phrase, there is an Ax

$$\forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1_{\{y_i = y = \text{``R-Ax''}\}} \leq \sum_{i=0}^{n-1} 1_{\{y_i = \text{``Ax''}\}}$$

- Continuation-Ax

If there is an Continuation-x phrase, there is an Ax before it

$$\forall j, y \in \mathcal{Y}_C, \ 1_{\{y_j = y = \text{``C-Ax''}\}} \leq \sum_{i=0}^{j} 1_{\{y_i = \text{``Ax''}\}}$$

- Many other possible constraints:
  - **Unique labels**
  - **No overlapping or embedding**
  - **Relations between number of arguments; order constraints**
  - **If verb is of type A, no argument of type B**

Learning Based Java: allows a developer to encode constraints in First Order Logic; these are compiled into linear inequalities automatically.

# SRL: Posing the Problem

$$\text{maximize} \quad \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{\mathbf{x}_i, y} 1_{\{y_i = y\}}$$

$$\text{where} \quad \lambda_{\mathbf{x},y} = \lambda \cdot F(\mathbf{x}, y) = \lambda_y \cdot F(\mathbf{x})$$

subject to

| | | |
|---|---|---|
| A | bomb [A1] | killer [A0] |
| car | | |
| bomb | | |
| that | bomb (Reference) [R-A1] | |
| exploded | V: explode | |
| outside | location [AM-LOC] | |
| the | | |
| U.S. | | |
| military | temporal [AM-TMP] | |
| base | | |
| in | location [AM-LOC] | |
| Beniji | | |
| killed | | V: kill |
| 11 | | corpse [A1] |
| Iraqi | | |
| citizens | | |
| . | | |

# SRL: Posing the Problem

$$\text{maximize} \quad \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{\mathbf{x}_i, y} 1_{\{y_i = y\}}$$

$$\text{where} \quad \lambda_{\mathbf{x},y} = \lambda \cdot F(\mathbf{x}, y) = \lambda_y \cdot F(\mathbf{x})$$

subject to

$$\forall i, \sum_{y \in \mathcal{Y}} 1_{\{y_i = y\}} = 1$$

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \le 1$$

$$\forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1_{\{y_i = y = \text{“R-Ax”}\}} \le \sum_{i=0}^{n-1} 1_{\{y_i = \text{“Ax”}\}}$$

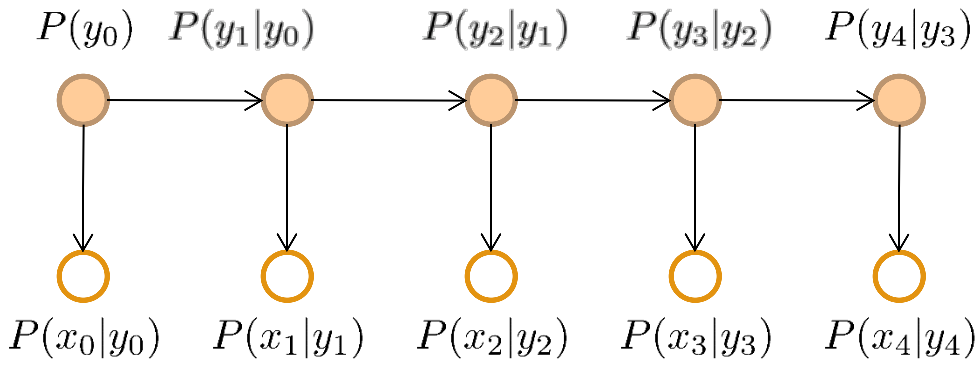
$$\forall j, y \in \mathcal{Y}_C, \ 1_{\{y_j = y = \text{“C-Ax”}\}} \le \sum_{i=0}^{j} 1_{\{y_i = \text{“Ax”}\}}$$

| | | |
|---|---|---|
| A | bomb [A1] | killer [A0] |
| car | | |
| bomb | | |
| that | bomb (Reference) [R-A1] | |
| exploded | V: explode | |
| outside | location [AM-LOC] | |
| the | | |
| U.S. | | |
| military | temporal [AM-TMP] | |
| base | | |
| in | location [AM-LOC] | |
| Beniji | | |
| killed | | V: kill |
| 11 | | corpse [A1] |
| Iraqi | | |
| citizens | | |
| . | | |

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

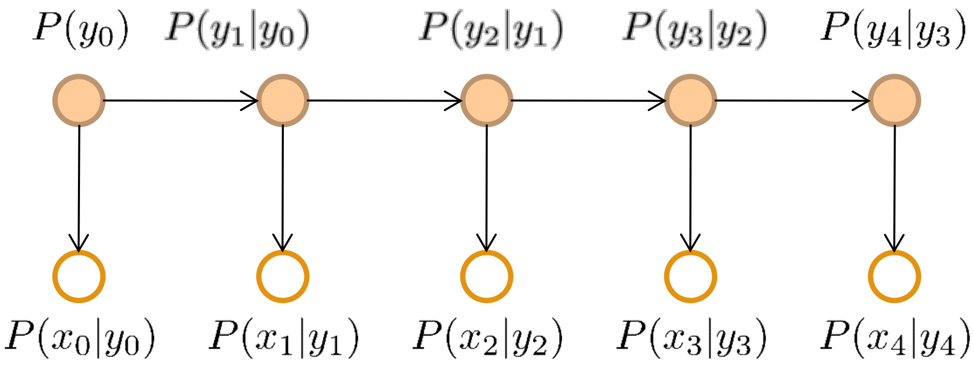


# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Here, *y*'s are labels; *x*'s are observations.



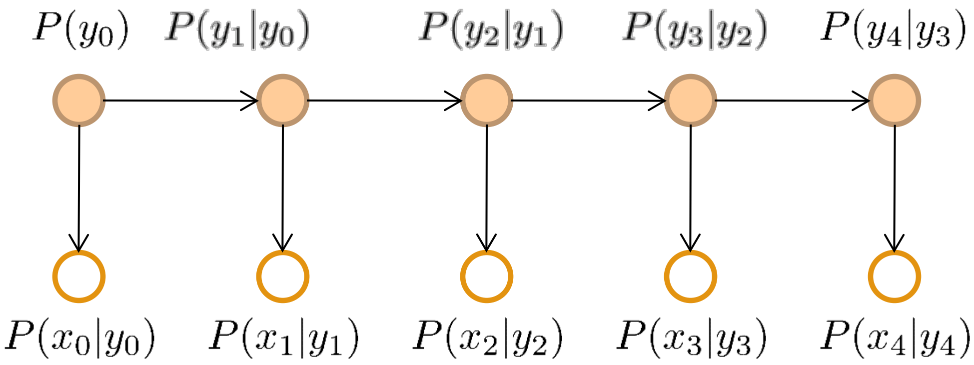
# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Here, *y*'s are labels; *x*'s are observations.

The ILP's objective function must include all entries of the Conditional Probability Table.



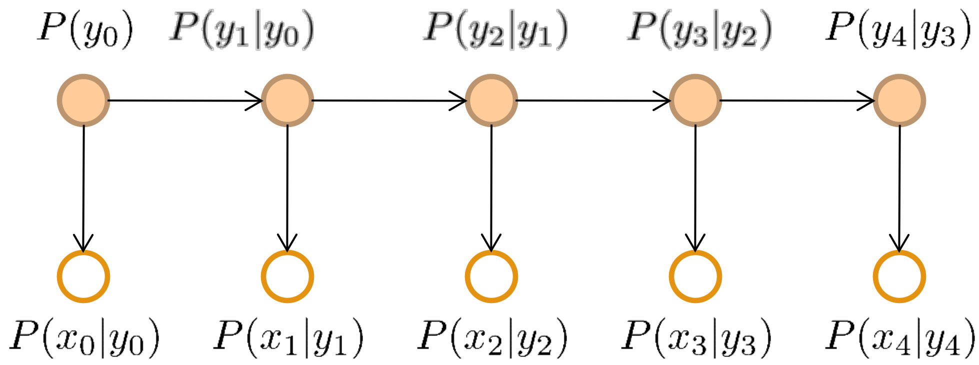
# Example 2: Sequence Tagging

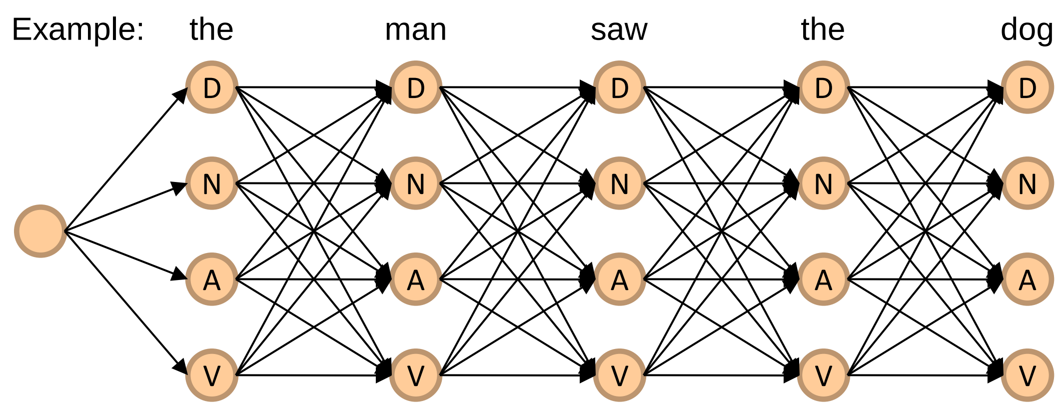
HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Here, *y*'s are labels; *x*'s are observations.

The ILP's objective function must include all entries of the Conditional Probability Table.



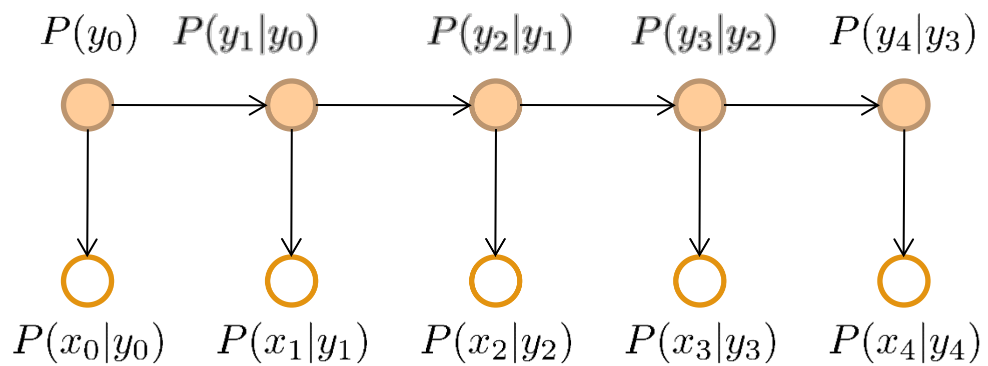
# Example 2: Sequence Tagging

HMM :

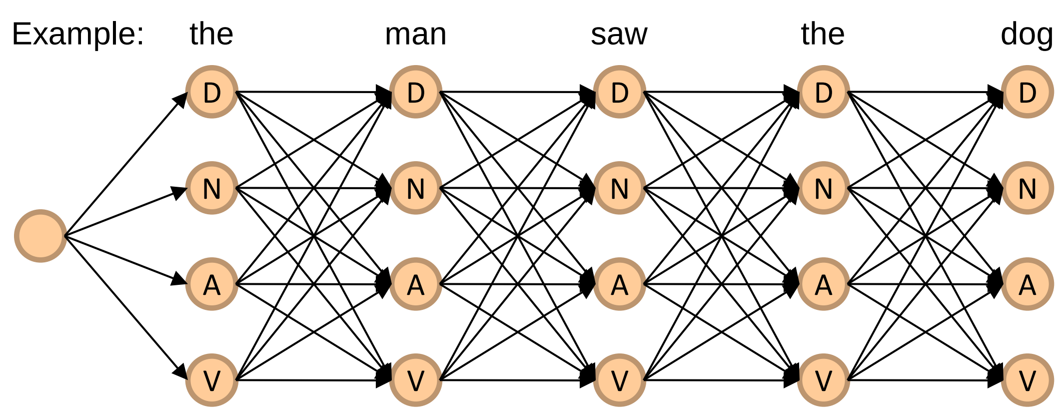
$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Here, *y*’s are labels; *x*’s are observations.

The ILP’s objective function must include all entries of the Conditional Probability Table.




Every edge is a Boolean variable that selects a transition CPT entry.

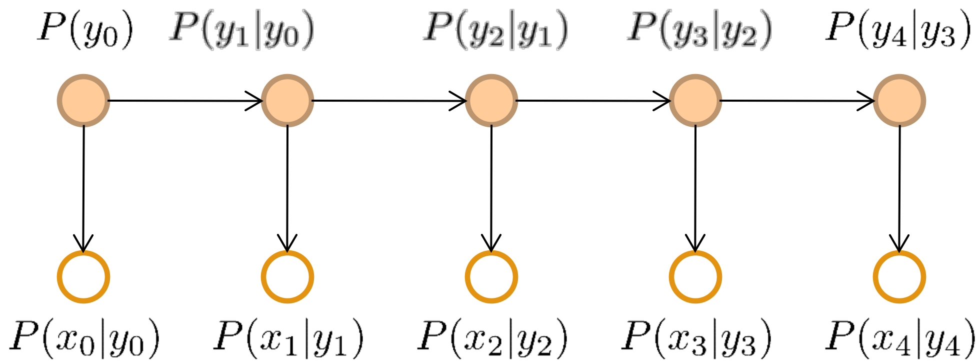


# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

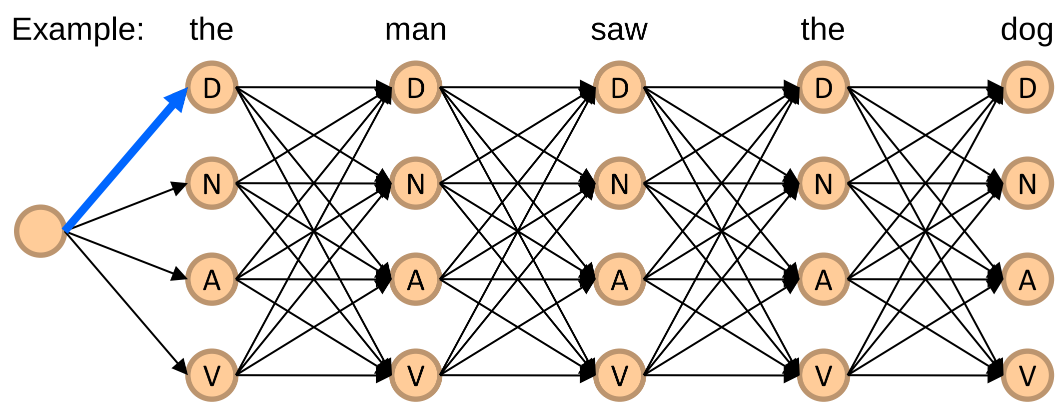
Here, *y*'s are labels; *x*'s are observations.




The ILP's objective function must include all entries of the Conditional Probability Table.

Every edge is a Boolean variable that selects a transition CPT entry.

They are related: if we choose $y_0 = D$



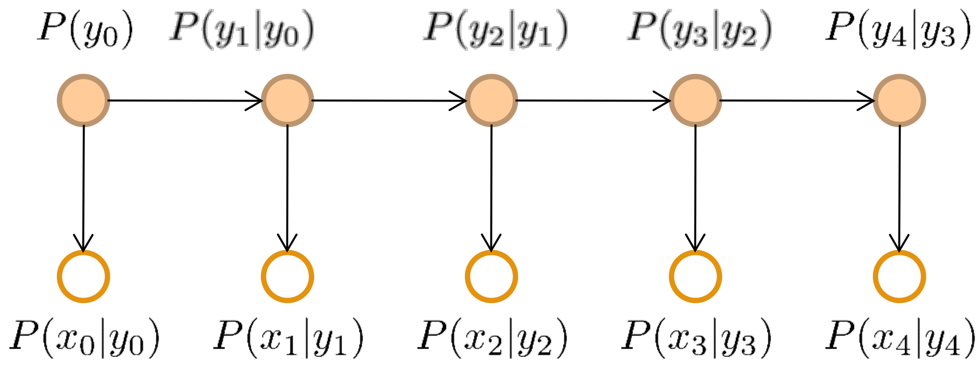
# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

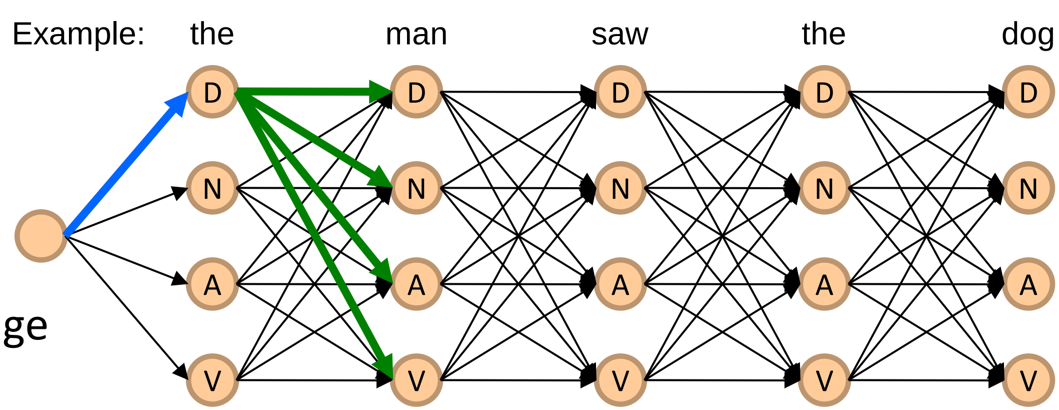
Here, *y*'s are labels; *x*'s are observations.

The ILP's objective function must include all entries of the Conditional Probability Table.




Every edge is a Boolean variable that selects a transition CPT entry.

They are related: if we choose $y_0$ = D then we must choose an edge $y_0$ = D $\wedge$ $y_1$ = ? .

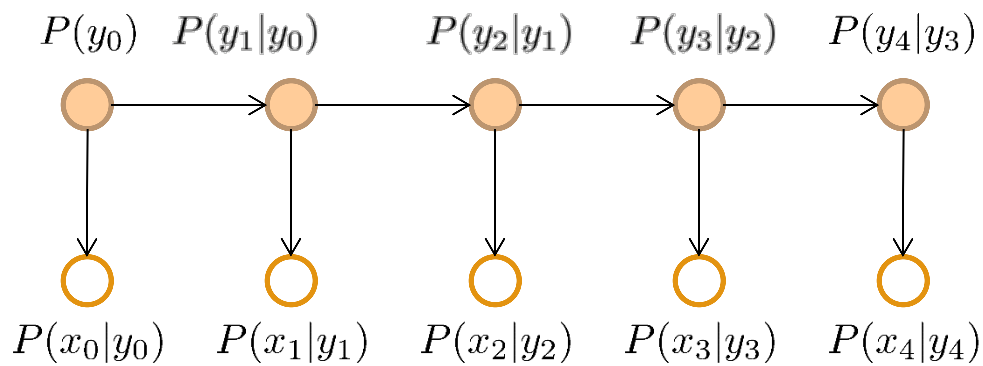


# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

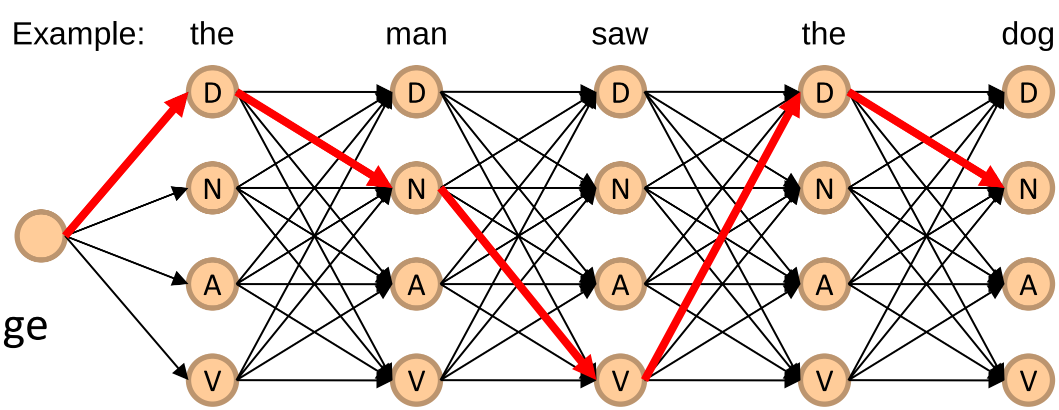
Here, *y*'s are labels; *x*'s are observations.




The ILP's objective function must include all entries of the Conditional Probability Table.

Every edge is a Boolean variable that selects a transition CPT entry.

They are related: if we choose $y_0 = D$ then we must choose an edge $y_0 = D \wedge y_1 = ?$ .



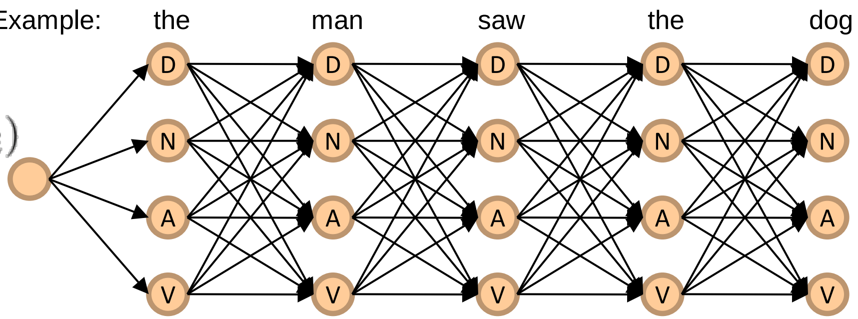

Every assignment to the *y*'s is a path.

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

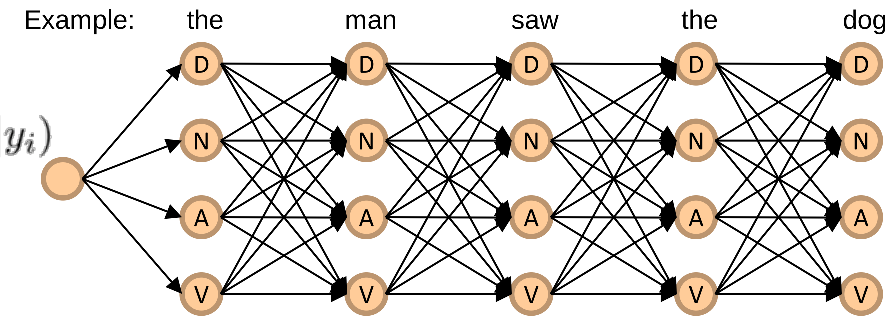
Example:



# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

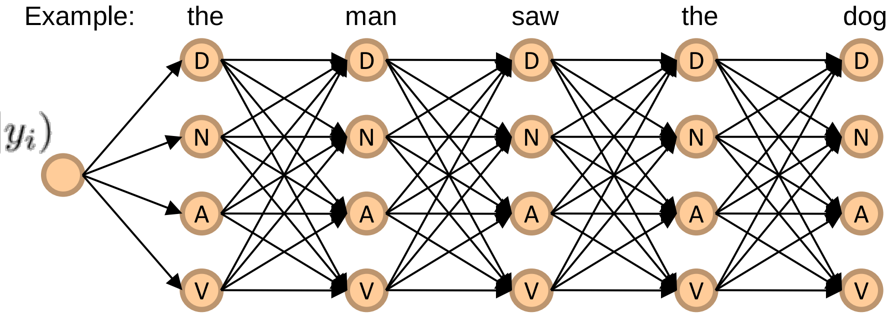
$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

subject to

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

subject to

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$

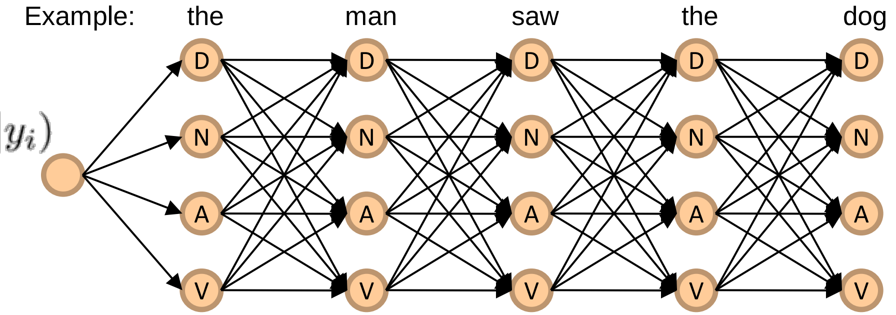
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

Learned Parameters

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

Inference Variables

$$\text{maximize} \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y\ \wedge\ y_{i-1}=y'\}}$$

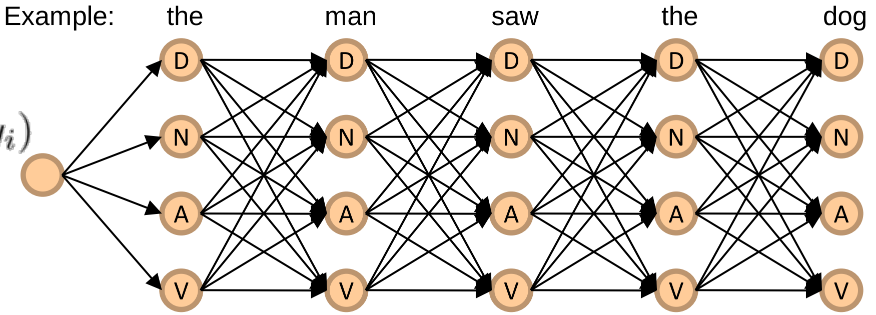
$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

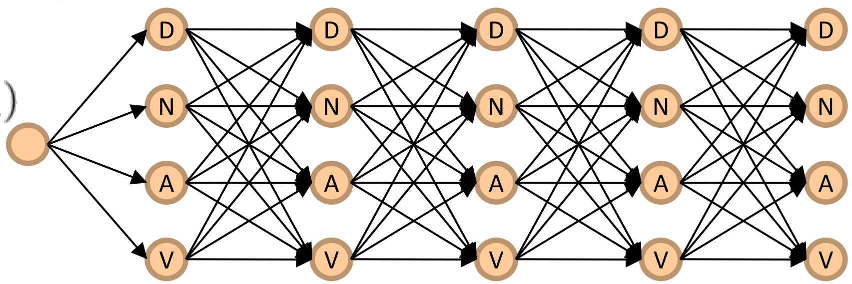
subject to

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Example:




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

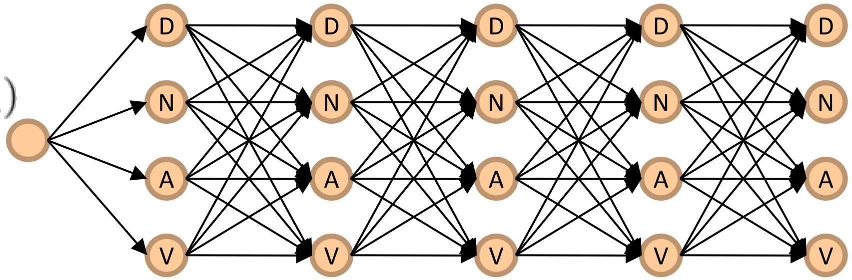
subject to

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Example:




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

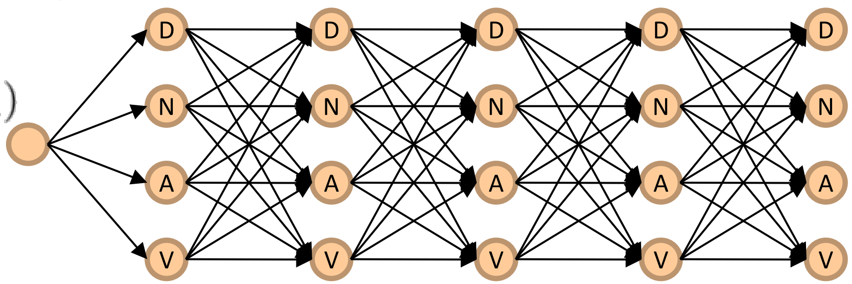
subject to

$$1_{\{y_0=\text{“NN”}\}} = 1$$
$$1_{\{y_0=\text{“VB”}\}} = 1$$
$$1_{\{y_0=\text{“JJ”}\}} = 1$$

# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\text{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

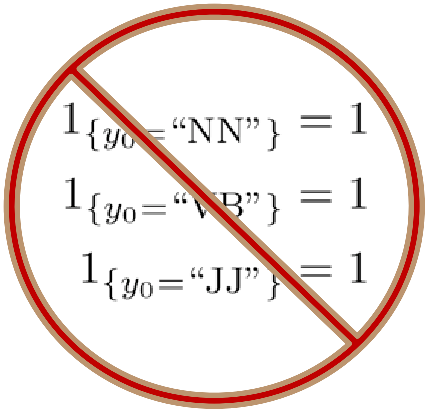
$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

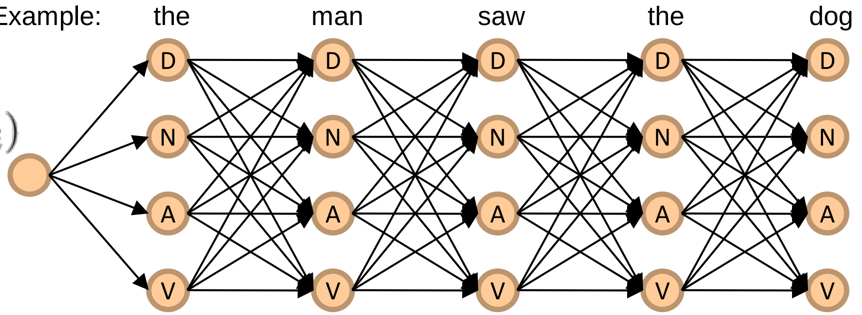


# Example 2: Sequence Tagging

HMM:

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$

Example:




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y}\underline{1_{\{y_0=y\}}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'}\underline{1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

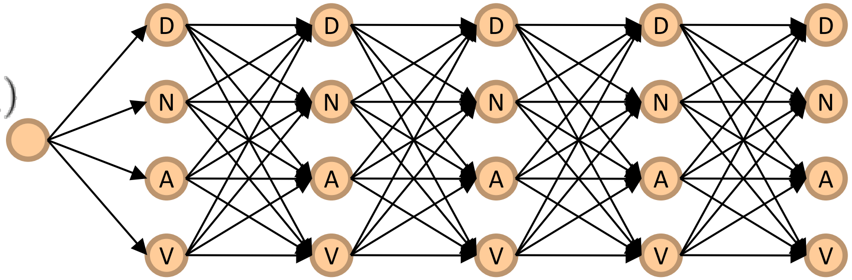
$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

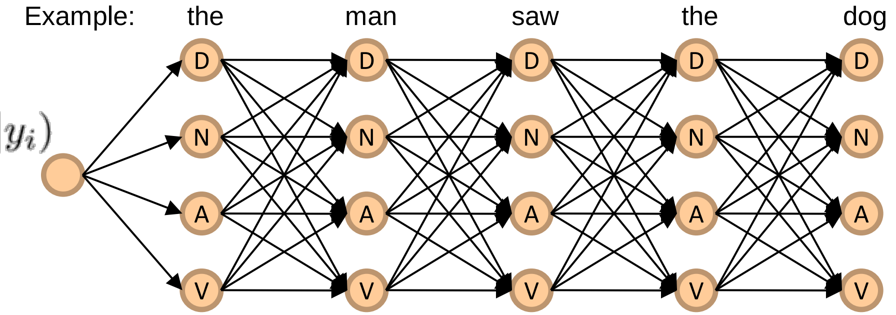
$$1_{\{y_0=\text{“NN”}\}} = 1 \qquad 1_{\{y_0=\text{“DT”} \,\wedge\, y_1=\text{“JJ”}\}} = 1$$

$$1_{\{y_0=\text{“DT”} \,\wedge\, y_1=\text{“JJ”}\}} = 1 \qquad 1_{\{y_1=\text{“NN”} \,\wedge\, y_2=\text{“VB”}\}} = 1$$

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y, \quad 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y \,\wedge\, y_1=y'\}}$$

$$\forall y, i>1 \quad \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y' \,\wedge\, y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y \,\wedge\, y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

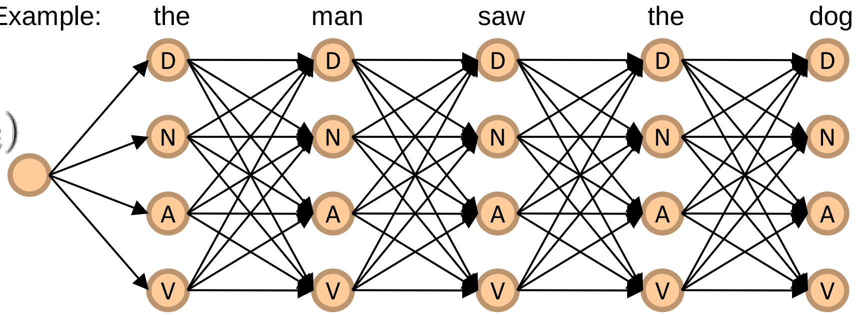
$$1_{\{y_0=\text{"NN"}\}} = 1$$
$$1_{\{y_0=\text{"DT"} \,\wedge\, y_1=\text{"JJ"}\}} = 1$$

$$1_{\{y_0=\text{"DT"} \,\wedge\, y_1=\text{"JJ"}\}} = 1$$
$$1_{\{y_1=\text{"NN"} \,\wedge\, y_2=\text{"VB"}\}} = 1$$

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y,\;\; 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y \,\wedge\, y_1=y'\}}$$

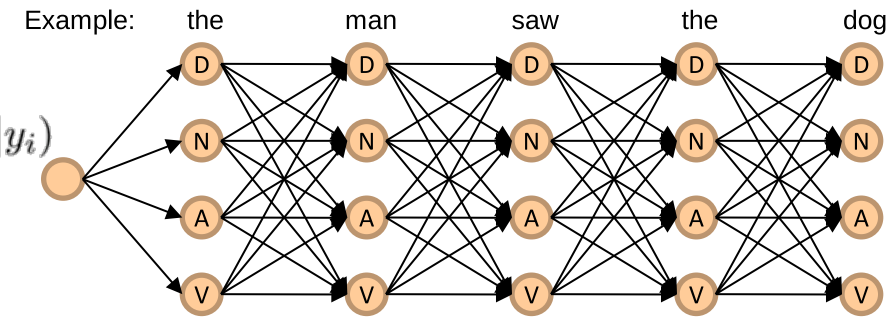
$$\forall y, i>1 \;\; \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y' \,\wedge\, y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y \,\wedge\, y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y,\quad 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y \,\wedge\, y_1=y'\}}$$

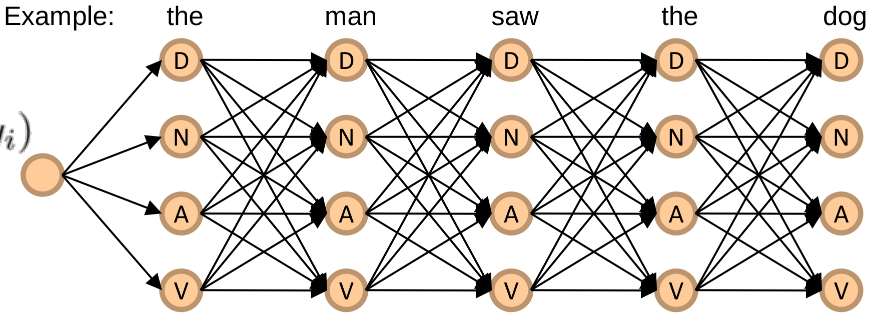
$$\forall y, i>1 \quad \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y' \,\wedge\, y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y \,\wedge\, y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \quad \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y\ \wedge\ y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y,\quad 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y\ \wedge\ y_1=y'\}}$$

$$\forall y, i>1 \quad \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y'\ \wedge\ y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y\ \wedge\ y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

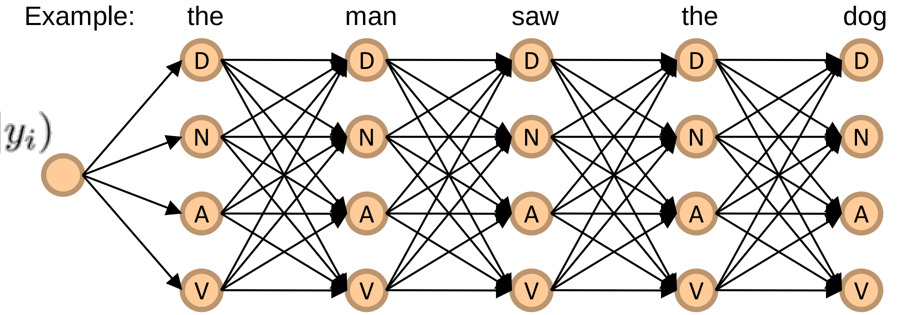
$$1_{\{y_0=\text{“V”}\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}} 1_{\{y_{i-1}=y\ \wedge\ y_i=\text{“V”}\}} \geq 1$$

***There must be a verb!***

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \sum_{y\in\mathcal{Y}} \lambda_{0,y} 1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'} 1_{\{y_i=y \,\wedge\, y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y,\ 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y \,\wedge\, y_1=y'\}}$$

$$\forall y, i > 1 \ \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y' \,\wedge\, y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y \,\wedge\, y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

Without additional constraints the ILP formulation of an HMM is totally unimodular

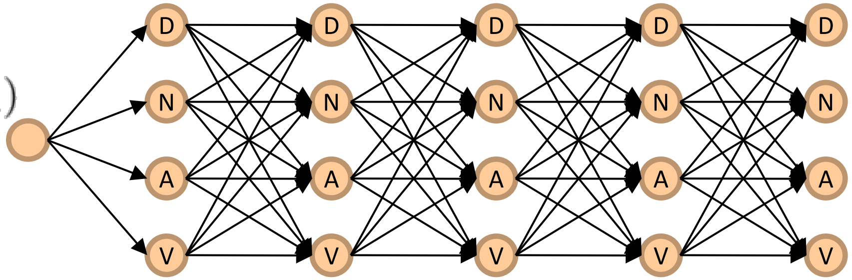
$$1_{\{y_0=\text{“V”}\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}} 1_{\{y_{i-1}=y \,\wedge\, y_i=\text{“V”}\}} \geq 1$$

***There must be a verb!***

# Example 2: Sequence Tagging

HMM :

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} P(y_0)P(x_0|y_0)\prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)$$




As an ILP:

$$\text{maximize} \sum_{y\in\mathcal{Y}} \lambda_{0,y}1_{\{y_0=y\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} \lambda_{i,y,y'}1_{\{y_i=y\ \wedge\ y_{i-1}=y'\}}$$

$$\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y))$$
$$\lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))$$

subject to

$$\sum_{y\in\mathcal{Y}} 1_{\{y_0=y\}} = 1$$

*Unique label for each word*

$$\forall y,\ \ 1_{\{y_0=y\}} = \sum_{y'\in\mathcal{Y}} 1_{\{y_0=y\ \wedge\ y_1=y'\}}$$

$$\forall y, i>1\ \ \sum_{y'\in\mathcal{Y}} 1_{\{y_{i-1}=y'\ \wedge\ y_i=y\}} = \sum_{y''\in\mathcal{Y}} 1_{\{y_i=y\ \wedge\ y_{i+1}=y''\}}$$

*Edges that are chosen must form a path*

Without additional constraints the ILP formulation of an HMM is totally unimodular

$$1_{\{y_0=\text{"V"}\}} + \sum_{i=1}^{n-1}\sum_{y\in\mathcal{Y}} 1_{\{y_{i-1}=y\ \wedge\ y_i=\text{"V"}\}} \geq 1$$

***There must be a verb!***

[Roth & Yih, ICML'05] discuss training paradigms for HMMs and CRFs, when augmented with additional knowledge

# Constraints

- We have seen three different constraints in this example
  - Unique label for each word
  - Chosen edges must form a path
  - There must be a verb
- All three can be expressed as linear inequalities
- In terms of modeling, there is a difference
  - The first two define the output structure (in this case, a sequence)
  - The third one adds knowledge to the problem

# Constraints

- We have seen three different constraints in this example
  - Unique label for each word
  - Chosen edges must form a path
  - There must be a verb
- All three can be expressed as linear inequalities

A conventional model

- In terms of modeling, there is a difference
  - The first two define the output structure (in this case, a sequence)
  - The third one adds knowledge to the problem

# Constraints

- We have seen three different constraints in this example
  - Unique label for each word
  - Chosen edges must form a path
  - There must be a verb
- All three can be expressed as linear inequalities
- In terms of modeling, there is a difference
  - The first two define the output structure (in this case, a sequence)
  - The third one adds knowledge to the problem

A conventional model

In CCMs, knowledge is an integral part of the modeling

# Learning and Inference in Structured Prediction

- Part 1: Introduction to Structured Prediction (60min)
  - Motivation
  - Examples:
    - **NE + Relations**
    - **Vision**
    - **Additional NLP Examples**
  - Problem Formulation
    - **Constrained Conditional Models: Integer Linear Programming Formulations**
  - Initial thoughts about learning
    - **Learning independent models**
    - **Constraints Driven Learning**
  - Initial thoughts about Inference
    - **Amortized Inference**

# Constrained Conditional Models—ILP Formulations

- Have been shown useful in the context of many NLP problems
- [Roth&Yih, 04,07: Entities and Relations; Punyakanok et. al: SRL ...]
  - Summarization; Co-reference; Information & Relation Extraction; Event Identifications and causality ; Transliteration; Textual Entailment; Knowledge Acquisition; Sentiments; Temporal Reasoning, Parsing,...

- Some theoretical work on training paradigms [Punyakanok et. al., 05 more; Constraints Driven Learning, PR, Constrained EM...]
- Some work on Inference, mostly approximations, bringing back ideas on Lagrangian relaxation, etc.

# Constrained Conditional Models—ILP Formulations

- Have been shown useful in the context of many NLP problems
- [Roth&Yih, 04,07: Entities and Relations; Punyakanok et. al: SRL …]
  - Summarization; Co-reference; Information & Relation Extraction; Event Identifications and causality ; Transliteration; Textual Entailment; Knowledge Acquisition; Sentiments; Temporal Reasoning, Parsing,…

- Some theoretical work on training paradigms [Punyakanok et. al., 05 more; Constraints Driven Learning, PR, Constrained EM…]
- Some work on Inference, mostly approximations, bringing back ideas on Lagrangian relaxation, etc.

- Good summary and description of training paradigms:

# Constrained Conditional Models—ILP Formulations

- Have been shown useful in the context of many NLP problems
- [Roth&Yih, 04,07: Entities and Relations; Punyakanok et. al: SRL …]
  - Summarization; Co-reference; Information & Relation Extraction; Event Identifications and causality ; Transliteration; Textual Entailment; Knowledge Acquisition; Sentiments; Temporal Reasoning, Parsing,…
- Some theoretical work on training paradigms [Punyakanok et. al., 05 more; Constraints Driven Learning, PR, Constrained EM…]
- Some work on Inference, mostly approximations, bringing back ideas on Lagrangian relaxation, etc.
- Good summary and description of training paradigms:
  - **[Chang, Ratinov & Roth, Machine Learning Journal 2012]**
- Summary of work & a bibliography: http://L2R.cs.uiuc.edu/tutorials.html

# LEARNING

$$y = \text{argmax}_{y \in \mathcal{Y}} \; w^T \phi(x, y) + u^T C(x, y)$$

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - Learning with latent representations

$$y = \text{argmax}_{y \in \mathcal{Y}} \; \mathbf{w}^T\phi(x, y) + \mathbf{u}^T\mathcal{C}(x, y)$$

The second part of the tutorial is on how to learn

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - Learning with latent representations

# LEARNING

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - Learning with latent representations

# LEARNING

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^T\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T\mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - Learning with latent representations

# LEARNING

The third part of the tutorial is on how to do inference

The second part of the tutorial is on how to learn

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}}\ \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^{\mathsf{T}}\mathcal{C}(\mathbf{x}, \mathbf{y})$$

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - Learning with latent representations

# Verb SRL

$$\text{maximize} \quad \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{\mathbf{x}_i, y} 1_{\{y_i = y\}}$$

$$\text{where} \quad \lambda_{\mathbf{x},y} = \lambda \cdot F(\mathbf{x}, y) = \lambda_y \cdot F(\mathbf{x})$$

subject to

| | | |
|---|---|---|
| A | bomb [A1] | killer [A0] |
| car | | |
| bomb | | |
| that | bomb (Reference) [R-A1] | |
| exploded | V: explode | |
| outside | location [AM-LOC] | |
| the | | |
| U.S. | | |
| military | temporal [AM-TMP] | |
| base | | |
| in | location [AM-LOC] | |
| Beniji | | |
| killed | | V: kill |
| 11 | | corpse [A1] |
| Iraqi | | |
| citizens | | |
| . | | |

# Verb SRL

$$\text{maximize} \quad \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{\mathbf{x}_i, y} 1_{\{y_i = y\}}$$

$$\text{where} \quad \lambda_{\mathbf{x},y} = \lambda \cdot F(\mathbf{x}, y) = \lambda_y \cdot F(\mathbf{x})$$

subject to

$$\forall i, \sum_{y \in \mathcal{Y}} 1_{\{y_i = y\}} = 1$$

$$\forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1_{\{y_i = y\}} \leq 1$$

$$\forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1_{\{y_i = y = \text{“R-Ax”}\}} \leq \sum_{i=0}^{n-1} 1_{\{y_i = \text{“Ax”}\}}$$

$$\forall j, y \in \mathcal{Y}_C, \; 1_{\{y_j = y = \text{“C-Ax”}\}} \leq \sum_{i=0}^{j} 1_{\{y_i = \text{“Ax”}\}}$$

| | ⊟ | ⊟ |
|---|---|---|
| A | bomb [A1] | killer [A0] |
| car | | |
| bomb | | |
| that | bomb (Reference) [R-A1] | |
| exploded | V: explode | |
| outside | location [AM-LOC] | |
| the | | |
| U.S. | | |
| military | temporal [AM-TMP] | |
| base | | |
| in | location [AM-LOC] | |
| Beniji | | |
| killed | | V: kill |
| 11 | | corpse [A1] |
| Iraqi | | |
| citizens | | |
| . | | |

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**

# Verb SRL is not Sufficient

skip

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**
  - Relation: Destination (preposition)
  - train to Chicago

# Extended Semantic Role Labeling

- Many predicates; many roles; how to deal with more phenomena?

BEIRUT, Lebanon — Lebanon's main opposition group called for widespread protests on Sunday in the wake of a powerful bomb attack for which it blamed Syria, posing a challenge to a shaky coalition government that is led by pro-Syrian factions and intensifying fears that Syria's civil war is spilling over into this country.

BEIRUT, Lebanon — Lebanon's main opposition group called for widespread protests on Sunday in the wake of a powerful bomb attack for which it blamed Syria, posing a challenge to a shaky coalition government that is led by pro-Syrian factions and intensifying fears that Syria's civil war is spilling over into this country.

[Beirut] is in [Lebanon].

[Lebanon] has a main opposition group.

[Lebanon's main opposition group] called for [widespread protests] [on Sunday].

There was [a powerful bomb attack].

[Lebanon's main opposition group] blamed [Syria].

[Pro-Syrian factions] lead [a shaky coalition government]

[Syria] has a [civil war].

[Someone] fears that [Syria's civil war is spilling over into this country].

…

Sentence level analysis may be influenced by other sentences

# Computational Questions

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**
  - Relation: Destination (preposition)
  - train to Chicago

# Computational Questions

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**
  - Relation: Destination (preposition)
  - train to Chicago

Identify the relation expressed by the predicate, and its arguments

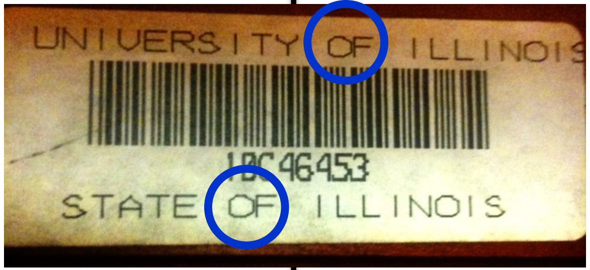
# Computational Questions

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**
  - Relation: Destination (preposition)
  - train to Chicago

Identify the relation expressed by the predicate, and its arguments

# Computational Questions

- *John, a fast-rising politician, slept on the train to Chicago.*
- **Verb Predicate: sleep**
  - Sleeper: John, a fast-rising politician
  - Location: on the train to Chicago
- **Who was John?**
  - Relation: Apposition (comma)
  - John, a fast-rising politician
- **What was John's destination?**
  - Relation: Destination (preposition)
  - train to Chicago




Identify the relation expressed by the predicate, and its arguments

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, ’11]
- Identify the relation’s arguments
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, '11]
- Identify the relation's arguments
  - [PP: Trans. Of ACL, '13, Comma: AAAI'16]
- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level

# Computational Challenges

- Predict the preposition **relations**
  - [EMNLP, ’11]
- Identify the relation’s **arguments**
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]


- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level
- Learning models in these settings exploits two principles:
  - Coherency among multiple phenomena

# Coherency in Semantic Role Labeling

**Predicate-arguments generated should be consistent across phenomena**

The touchdown scored by Bradford cemented the victory of the Eagles.

| Verb | Nominalization | Preposition |
|---|---|---|
| **Predicate:** score | **Predicate:** win | **Sense:** 11(6) |
| **A0:** Bradford (scorer)<br>**A1:** The touchdown (points scored) | **A0:** the Eagles (winner) | "the object of the preposition is the object of the underlying verb of the nominalization" |

Linguistic Constraints:

- A0: the Eagles ⇔ Sense(of): 11(6)
- A0: Bradford ⇔ Sense(by): 1(1)

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, ’11]
- Identify the relation’s arguments
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]


- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level
- Learning models in these settings exploits two principles:
  - Coherency among multiple phenomena

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, ’11]
- Identify the relation’s arguments
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]


- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level
- Learning models in these settings exploits two principles:
  - Coherency among multiple phenomena
  - Constraining latent structures (relating observed and latent variables)

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, ’11]
- Identify the relation’s arguments
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]


- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level
- Learning models in these settings exploits two principles:
  - Coherency among multiple phenomena
  - Constraining latent structures (relating observed and latent variables)

Input & relation

Argument & their types

# Computational Challenges

- Predict the preposition relations
  - [EMNLP, ’11]
- Identify the relation’s arguments
  - [PP: Trans. Of ACL, ’13, Comma: AAAI’16]


- Very little supervised data
  - per phenomena
- Minimal annotation
  - only at the predicate level
- Learning models in these settings exploits two principles:
  - Coherency among multiple phenomena
  - Constraining latent structures (relating observed and latent variables)
  - Done via global inference via CCM

Input & relation

Argument & their types

# Joint inference (CCMs)

Verb arguments

$$\max_{\mathbf{y}} \sum_{t} \sum_{a} y^{a,t} c^{a,t}$$

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

Verb arguments

$$\max_{\mathbf{y}} \sum_{t} \sum_{a} y^{a,t} c^{a,t}$$

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

Verb arguments

$$\max_{\mathbf{y}} \sum_{t} \sum_{a} y^{a,t} c^{a,t}$$

$a$ → Argument candidates

$t$ → Each argument label

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

Verb arguments

$$\max_{\mathbf{y}} \sum_{t} \sum_{a} y^{a,t} c^{a,t}$$

**Constraints:**

Verb SRL constraints

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

**Verb arguments**

$$\max_{\mathbf{y}} \sum_t \sum_a y^{a,t} c^{a,t}$$

**Preposition relations**

$$\max_{\mathbf{y}} \sum_r \sum_p y^{r,p} c^{r,p}$$

**Constraints:**

Verb SRL constraints

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

**Verb arguments**

$$\max_{\mathbf{y}} \sum_t \sum_a y^{a,t} c^{a,t}$$

**Preposition relations**

$$\max_{\mathbf{y}} \sum_r \sum_p y^{r,p} c^{r,p}$$

$r$: Preposition relation label

$p$: Preposition

**Constraints:**

Verb SRL constraints

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

**Verb arguments**

$$\max_{\mathbf{y}} \sum_{t} \sum_{a} y^{a,t} c^{a,t}$$

**Preposition relations**

$$\max_{\mathbf{y}} \sum_{r} \sum_{p} y^{r,p} c^{r,p}$$

**Constraints:**

Verb SRL constraints

Preposition SRL Constraints

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument a is assigned a label t.
$c^{a,t}$ is the corresponding model score

Verb arguments

Preposition relations

$$\max_{\mathbf{y}} \sum_t \lambda^t \sum_a y^{a,t} c^{a,t} + \sum_r \lambda^r \sum_p y^{r,p} c^{r,p}$$

**Constraints:**

Verb SRL constraints

Preposition SRL Constraints

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument a is assigned a label t.
$c^{a,t}$ is the corresponding model score

Verb arguments

Preposition relations

$$\max_{\mathbf{y}} \sum_t \lambda^t \sum_a y^{a,t} c^{a,t} + \sum_r \lambda^r \sum_p y^{r,p} c^{r,p}$$

**Constraints:**

Verb SRL constraints

Preposition SRL Constraints

**+** Joint constraints between tasks; easy with ILP formulations

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument $a$ is assigned a label $t$.
$c^{a,t}$ is the corresponding model score

Verb arguments

Preposition relations

$$\max_{\mathbf{y}} \sum_t \lambda^t \sum_a y^{a,t} c^{a,t} + \sum_r \lambda^r \sum_p y^{r,p} c^{r,p}$$

**Constraints:**

Verb SRL constraints

Preposition SRL Constraints

**+** Joint constraints between tasks; easy with ILP formulations

Joint Inference – no (or minimal) joint learning

# Joint inference (CCMs)

Variable $y^{a,t}$ indicates whether candidate argument a is assigned a label t.
$c^{a,t}$ is the corresponding model score

Verb arguments

Preposition relations

$$\max_{\mathbf{y}} \sum_t \lambda^t \sum_a y^{a,t} c^{a,t} + \sum_r \lambda^r \sum_p y^{r,p} c^{r,p}$$

+ ....

**Constraints:**

Verb SRL constraints

Preposition SRL Constraints

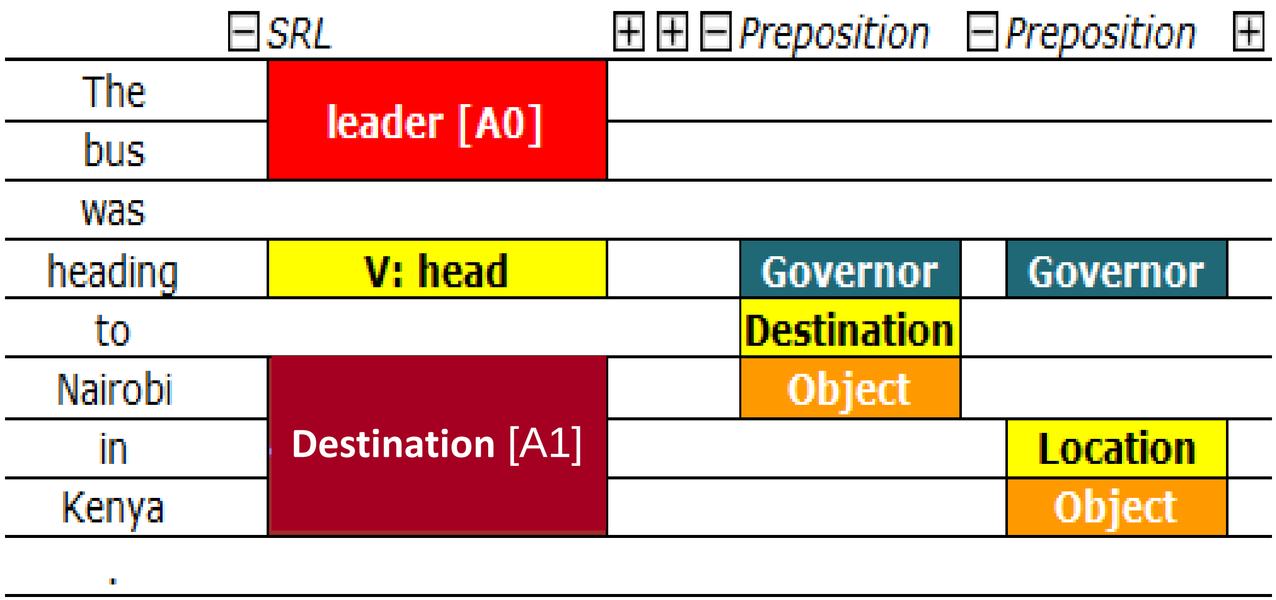
**+** Joint constraints between tasks; easy with ILP formulations

Joint Inference – no (or minimal) joint learning

# Extended SRL [Demo]

| | SRL | Preposition | Preposition |
|---|---|---|---|
| The | leader [A0] | | |
| bus | | | |
| was | | | |
| heading | V: head | Governor | Governor |
| to | | Destination | |
| Nairobi | Destination [A1] | Object | |
| in | | | Location |
| Kenya | | | Object |
| . | | | |

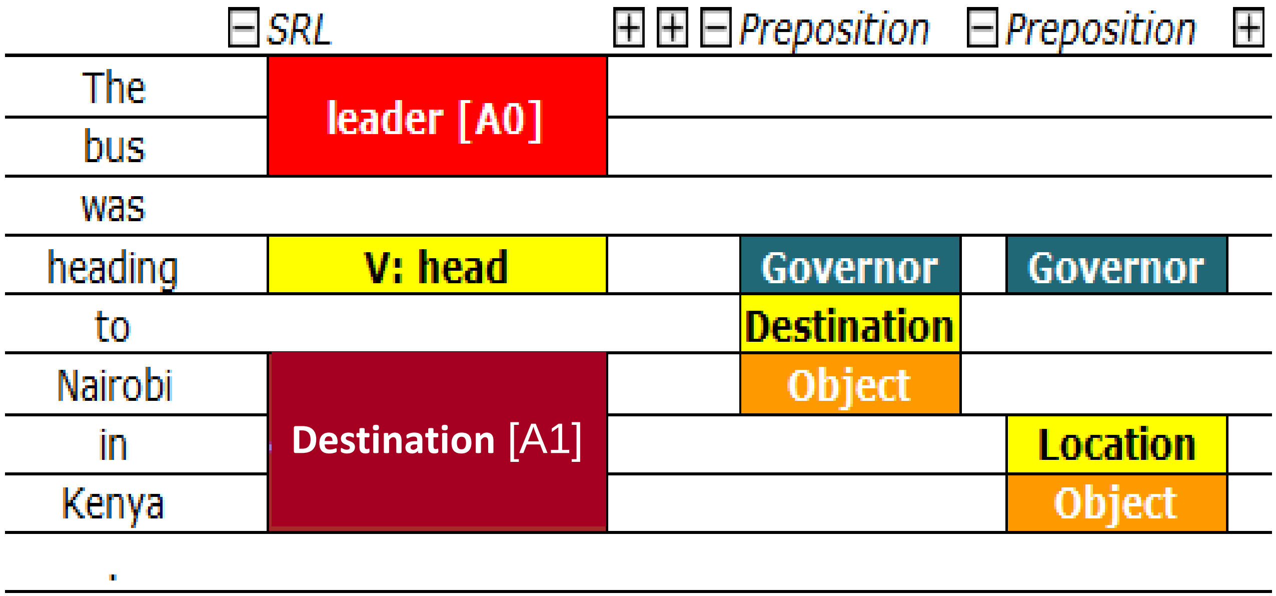
# Extended SRL [Demo]




Joint inference over phenomena–specific models to enforce consistency

Models trained with latent structure: senses, types, arguments

# Extended SRL [Demo]




Joint inference over phenomena–specific models to enforce consistency

Models trained with latent structure: senses, types, arguments

- More to do with other relations, discourse phenomena,…

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

[AUTHOR]

[TITLE]

[EDITOR]

[BOOKTITLE]

[TECH-REPORT]

[INSTITUTION]

[DATE]

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | . PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | . PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language.  PhD thesis. DIKU , University of Copenhagen, May 1994 .

$$\underset{y}{\operatorname{argmax}}\, \boldsymbol{\lambda} \cdot F(x, y)$$

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | .  PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

# Information Extraction without Output Expectations

Lars Ole Andersen . Program analysis and specialization for the C Programming language. PhD thesis. DIKU , University of Copenhagen, May 1994 .

$$\operatorname*{argmax}_{y} \boldsymbol{\lambda} \cdot F(x, y)$$

**Prediction result of a trained HMM**

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . Program analysis and |
| [TITLE] | specialization for the |
| [EDITOR] | C |
| [BOOKTITLE] | Programming language |
| [TECH-REPORT] | . PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , May |
| [DATE] | 1994 . |

Violates lots of natural constraints!

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples

Increasing the model complexity

Increase difficulty of Learning

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples
  - What if we only have a few labeled examples?

Increasing the model complexity

Increase difficulty of Learning

Can we keep the learned model simple and still make expressive decisions?

# Strategies for Improving the Results

- (Standard) Machine Learning Approaches
  - Higher Order HMM/CRF?
  - Increasing the window size?
  - Adding a lot of new features
    - Requires a lot of labeled examples
  - What if we only have a few labeled examples?

Increasing the model complexity

Increase difficulty of Learning

Can we keep the learned model simple and still make expressive decisions?

- Instead:
  - Constrain the output to make sense – satisfy our expectations
  - Push the (simple) model in a direction that makes sense – minimally violates our expectations.

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

Easy to express pieces of "knowledge"

# Expectations from the output (Constraints)

- Each field must be a consecutive list of words and can appear at most once in a citation.
- State transitions must occur on punctuation marks.
- The citation can only start with AUTHOR or EDITOR.
- The words pp., pages correspond to PAGE.
- Four digits starting with 20xx and 19xx are DATE.
- Quotations can appear only in TITLE
- …….

Easy to express pieces of "knowledge"

Non Propositional; May use Quantifiers

- Adding constraints, we get correct results!
  - Without changing the model

$$\operatorname*{argmax}_{y} \boldsymbol{\lambda} \cdot F(x, y)$$

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

# Information Extraction with Expectation Constraints

- Adding constraints, we get correct results!
  - Without changing the model

$$\underset{y}{\operatorname{argmax}}\, \boldsymbol{\lambda} \cdot F(x, y)$$

| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

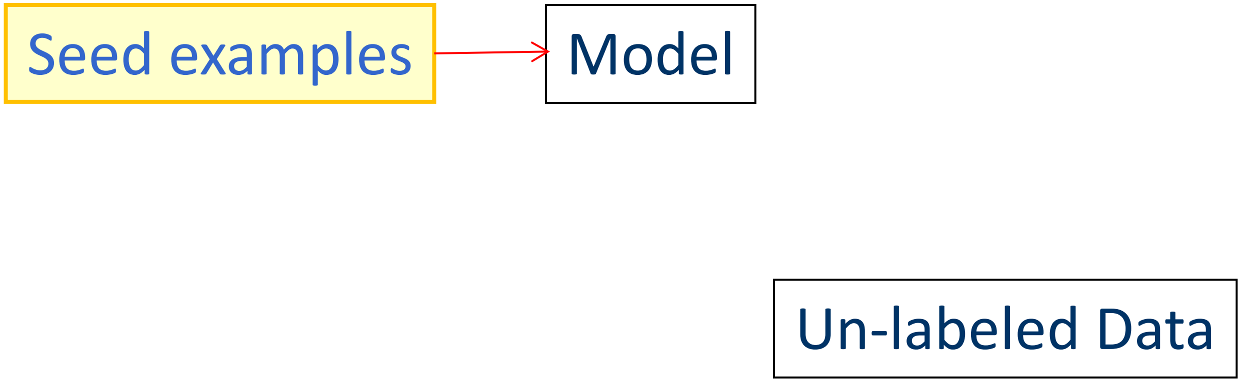
- Adding constraints, we get correct results!
  - Without changing the model

$$\underset{y}{\operatorname{argmax}}\, \boldsymbol{\lambda} \cdot F(x, y) - \sum_{i=1}^{K} \rho_i d(y, 1_{C_i(x)})$$

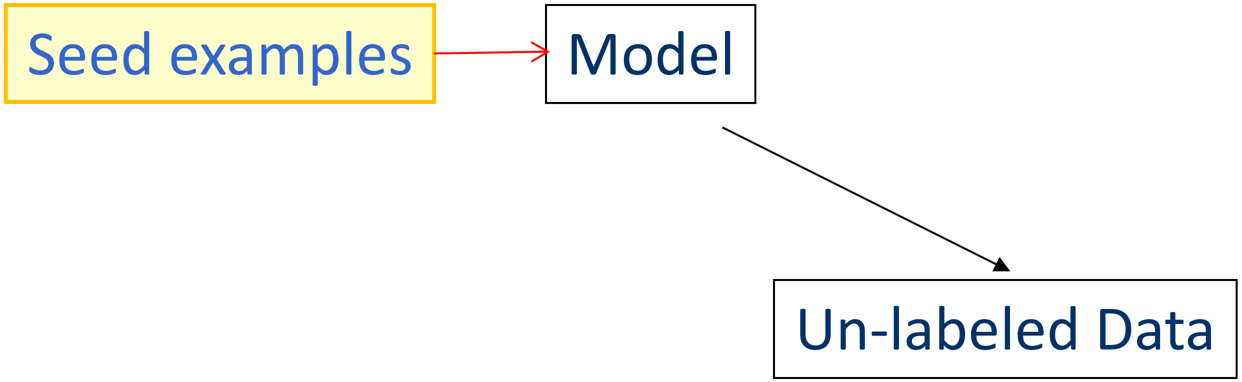
| | |
|---|---|
| [AUTHOR] | Lars Ole Andersen . |
| [TITLE] | Program analysis and specialization for the C Programming language . |
| [TECH-REPORT] | PhD thesis . |
| [INSTITUTION] | DIKU , University of Copenhagen , |
| [DATE] | May, 1994 . |

# Guiding (Semi-Supervised) Learning with Constraints

# Guiding (Semi-Supervised) Learning with Constraints

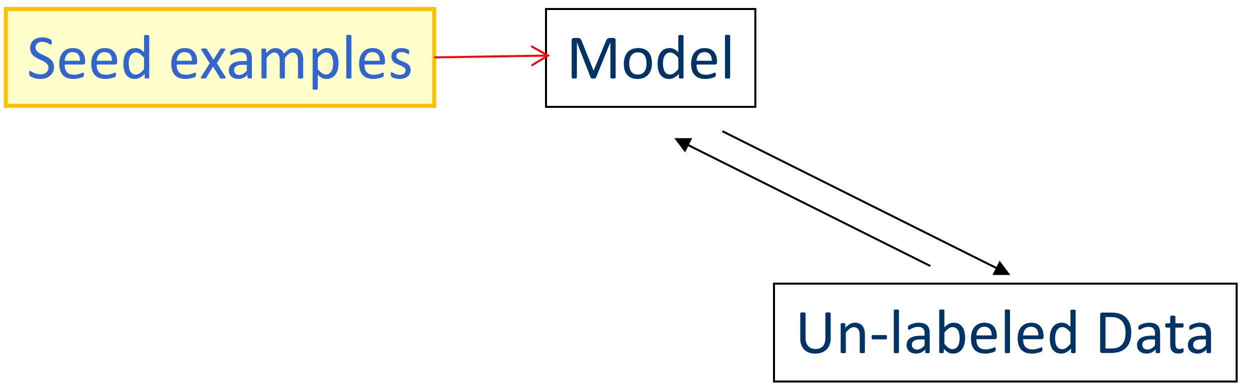


# Guiding (Semi-Supervised) Learning with Constraints



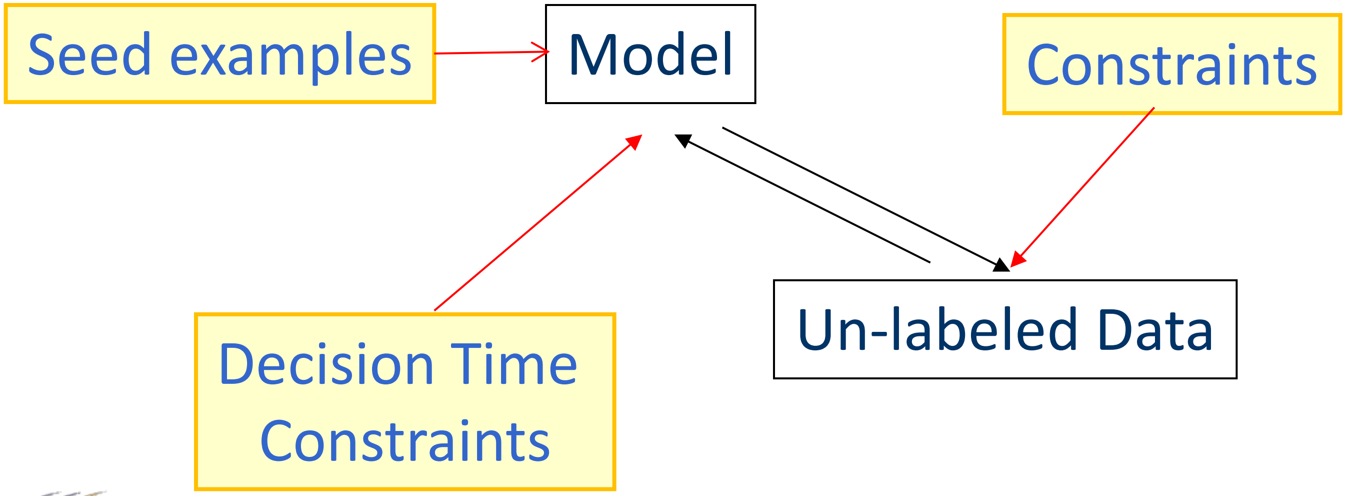
# Guiding (Semi-Supervised) Learning with Constraints

- In traditional Semi-Supervised learning the model can drift away from the correct one.



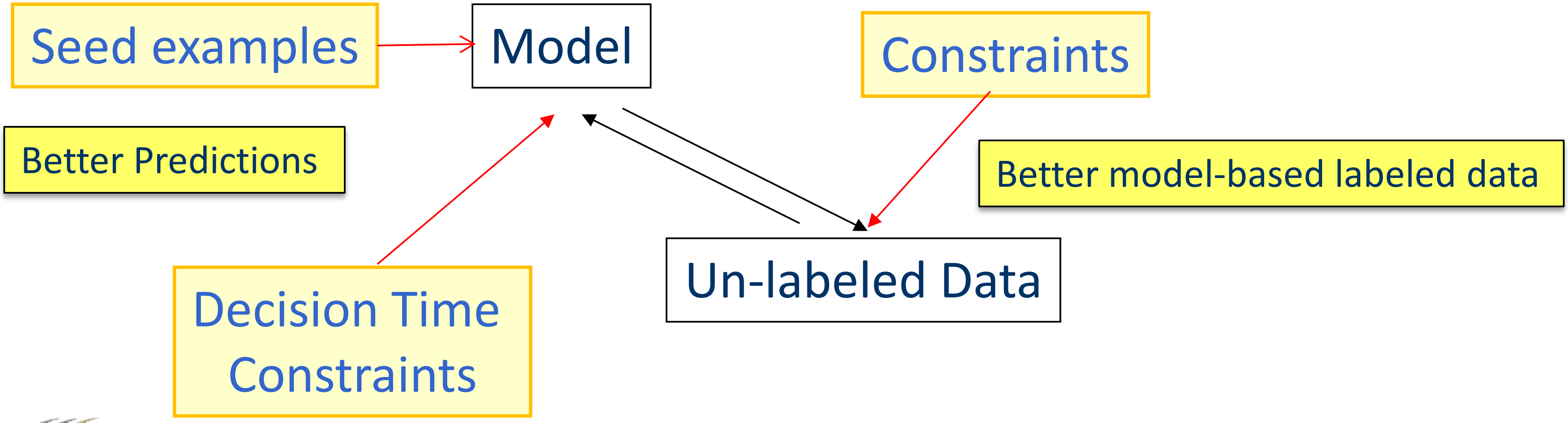
# Guiding (Semi-Supervised) Learning with Constraints

- In traditional Semi-Supervised learning the model can drift away from the correct one.



# Guiding (Semi-Supervised) Learning with Constraints

- In traditional Semi-Supervised learning the model can drift away from the correct one.
- Constraints can be used to generate better training data
  - At training to improve labeling of un-labeled data (and thus improve the model)
  - At decision time, to bias the objective function towards favoring constraint satisfaction.



# Constraints Driven Learning (CoDL)

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

$(w,\rho)$=learn(L)

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y \; w^T \phi(x,y) - \sum \rho \, d_C(x,y)$

$T=T \cup \{(x, h)\}$

$(w,\rho) = \gamma \, (w,\rho) + (1- \gamma) \text{ learn}(T)$

# Constraints Driven Learning (CoDL)

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

Supervised learning algorithm parameterized by $(w,\rho)$. **[LATER]**

$(w,\rho)=\text{learn}(L)$

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y\ w^T \phi(x,y) - \sum \rho\ d_C(x,y)$

$T=T \cup \{(x, h)\}$

$(w,\rho) = \gamma\ (w,\rho) + (1- \gamma)\ \text{learn}(T)$

# Constraints Driven Learning (CoDL)

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

Supervised learning algorithm parameterized by $(w,\rho)$. **[LATER]**

$(w,\rho)$=learn(L)

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y\ w^T \phi(x,y) - \sum \rho\, d_C(x,y)$

$T=T \cup \{(x, h)\}$

Inference with constraints: augment the training set

$(w,\rho) = \gamma\,(w,\rho) + (1- \gamma)\,\text{learn}(T)$

# Constraints Driven Learning (CoDL)

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

$(w,\rho)$=learn(L)

Supervised learning algorithm parameterized by $(w,\rho)$. **[LATER]**

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y\ w^T \phi(x,y) - \sum \rho\ d_C(x,y)$

$T=T \cup \{(x, h)\}$

**Inference with constraints:** augment the training set

$(w,\rho) = \gamma\ (w,\rho) + (1- \gamma)\ \text{learn}(T)$

**Learn from new training data**
Weigh supervised & unsupervised models.

# Constraints Driven Learning (CoDL)

Archetypical Semi/un-supervised learning: **A constrained EM**

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

$(w,\rho)$=learn(L)

Supervised learning algorithm parameterized by $(w,\rho)$. **[LATER]**

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y \; w^T \phi(x,y) - \sum \rho \; d_C(x,y)$

$T=T \cup \{(x, h)\}$

**Inference with constraints:** augment the training set

$(w,\rho) = \gamma \, (w,\rho) + (1-\gamma) \, \text{learn}(T)$

**Learn from new training data**
Weigh supervised & unsupervised models.

# Constraints Driven Learning (CoDL)

Archetypical Semi/un-supervised learning: **A constrained EM**

[Chang, Ratinov, Roth, ACL'07;ICML'08,MLJ'12]
See also: Ganchev et. al. 10 (PR)

$(w,\rho)$=learn(L)

Supervised learning algorithm parameterized by $(w,\rho)$. **[LATER]**

For N iterations do

$T=\phi$

For each x in unlabeled dataset

$h \leftarrow \text{argmax}_y \, w^T \phi(x,y) - \sum \rho \, d_C(x,y)$

$T=T \cup \{(x, h)\}$

**Inference with constraints:** augment the training set

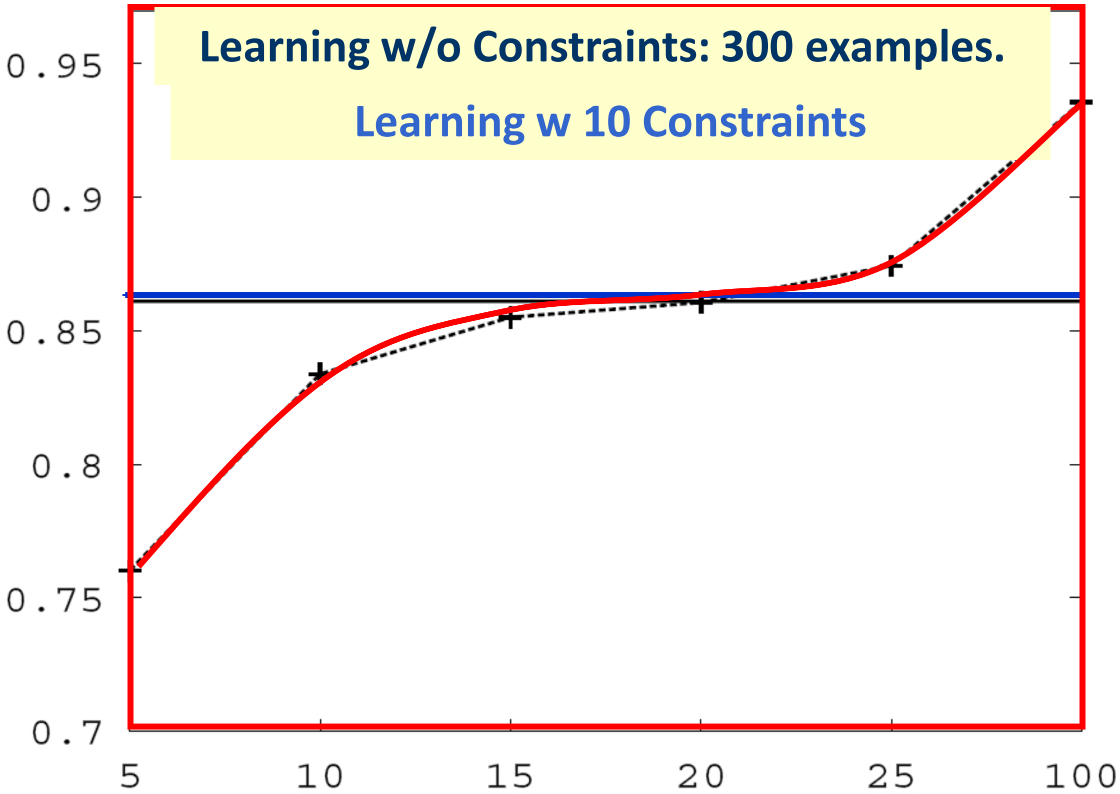
$(w,\rho) = \gamma\,(w,\rho) + (1-\gamma)\,\text{learn}(T)$

**Learn from new training data**
Weigh supervised & unsupervised models.

**Excellent Experimental Results** showing the advantages of using constraints, especially with small amounts of labeled data [Chang et. al, Others]

# Value of Constraints in Semi-Supervised Learning

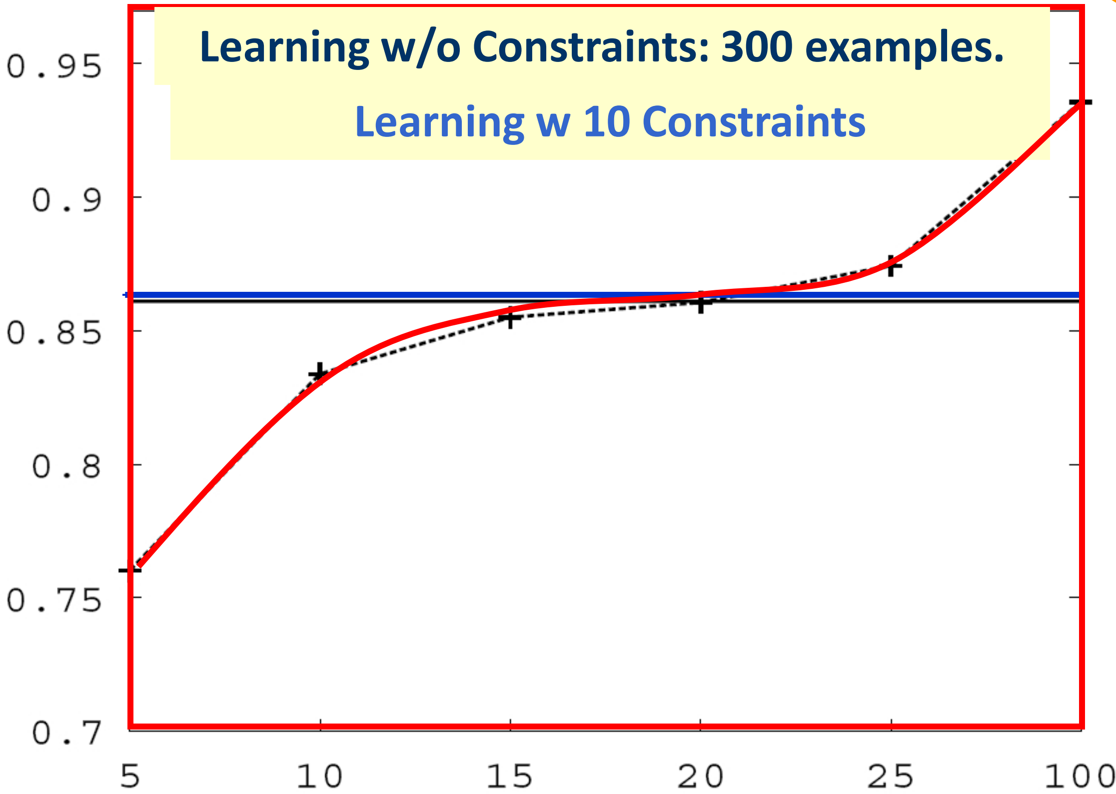
**Objective function:** $f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \sum w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum \rho_i d_{C_i}(\mathbf{x},\mathbf{y}).$




**# of available labeled examples**

# Value of Constraints in Semi-Supervised Learning

**Objective function:** $$f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \sum w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum \rho_i d_{C_i}(\mathbf{x},\mathbf{y}).$$



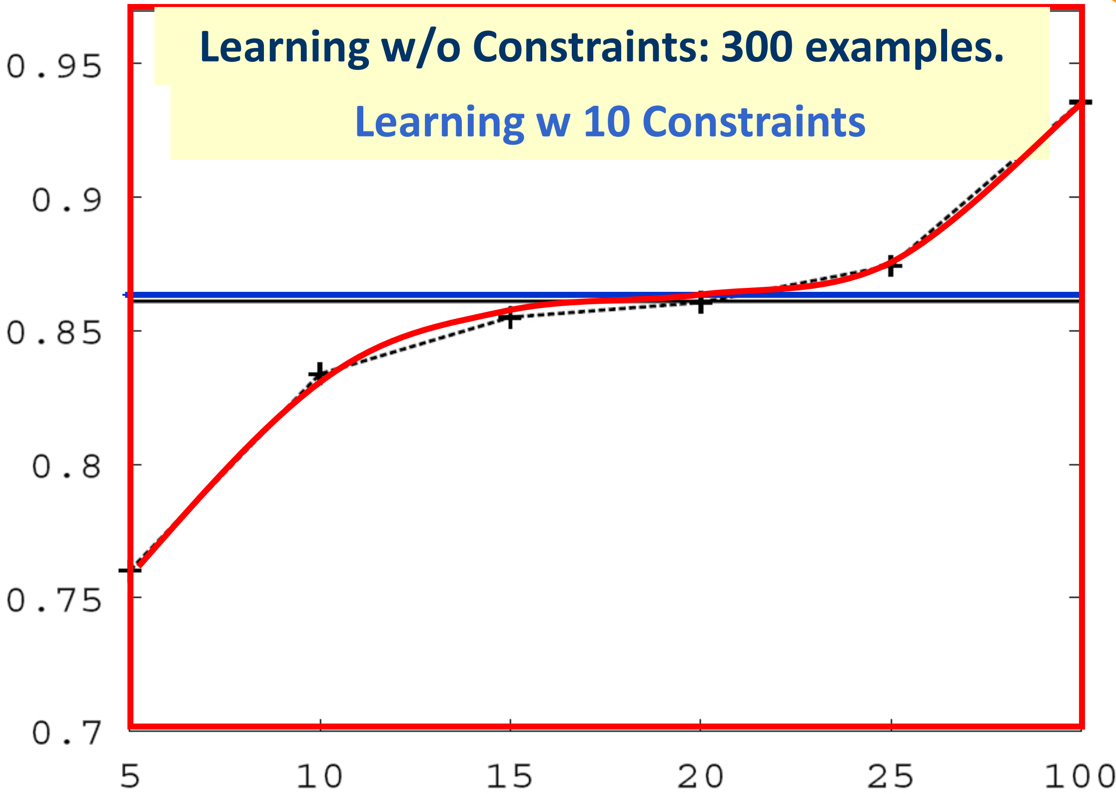

**# of available labeled examples**

**Constraints are used to Bootstrap a semi-supervised learner**

simple model + constraints used to annotate unlabeled data, which in turn is used to keep training the model.

# Value of Constraints in Semi-Supervised Learning

**Objective function:** $f_{\Phi,C}(\mathbf{x},\mathbf{y}) = \sum w_i \phi_i(\mathbf{x},\mathbf{y}) - \sum \rho_i d_{C_i}(\mathbf{x},\mathbf{y}).$




**# of available labeled examples**

**Constraints are used to Bootstrap a semi-supervised learner**
simple model + constraints used to annotate unlabeled data, which in turn is used to keep training the model.

See Chang et. al. MLJ'12 on the use of **soft constraints** in CCMs.
The tutorial's web page will include a write-up on ILP formulations **incorporating soft constraints.**

# CoDL as Constrained Hard EM

- Hard EM is a popular variant of EM
- While EM estimates a distribution over hidden variables in the E-step,
- … Hard EM predicts the **best** output in the E-step

$$h = y^* = \mathbf{argmax}_{\mathrm{y}}\ P_w(\mathbf{y} \,|\, \mathbf{x})$$

- Alternatively, hard EM predicts a peaked distribution

$$q(y) = \delta_{y=y^*}$$

# CoDL as Constrained Hard EM

- Hard EM is a popular variant of EM
- While EM estimates a distribution over hidden variables in the E-step,
- … Hard EM predicts the **best** output in the E-step

$$h = y^* = \mathbf{argmax}_{\mathrm{y}}\ P_w(\mathbf{y} \mid \mathbf{x})$$

- Alternatively, hard EM predicts a peaked distribution

$$q(y) = \delta_{y=y^*}$$

- Constrained-Driven Learning (CODL) – can be viewed as a constrained version of hard EM:

$$y^* = \mathbf{argmax}_{\mathrm{y:Uy \leq b}}\ P_w(y \mid x)$$

Constraining the feasible set

# Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:
- $$y^* = \mathbf{argmax}_{y:Uy \leq b} P_w(y|x)$$

Constraining the feasible set

- … It is possible to derive a constrained version of EM:

# Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:
- $$y^* = \mathbf{argmax}_{\mathrm{y:Uy} \leq \mathrm{b}} P_w(y|x)$$

Constraining the feasible set

- ... It is possible to derive a constrained version of EM:
- To do that, constraints are relaxed into expectation constraints on the posterior probability q:

# Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:
- $$y^{*} = \mathbf{argmax}_{y:Uy \leq b}\ P_w(y|x)$$

Constraining the feasible set

- … It is possible to derive a constrained version of EM:
- To do that, constraints are relaxed into expectation constraints on the posterior probability q:

$$\mathrm{E}_q[Uy] \leq b$$

# Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:
- $$y^* = \mathbf{argmax}_{y:Uy \leq b} P_w(y|x)$$

Constraining the feasible set

- … It is possible to derive a constrained version of EM:
- To do that, constraints are relaxed into expectation constraints on the posterior probability q:

$$E_q[Uy] \leq b$$

- The E-step now becomes: [Neal & Hinton '99 view of EM]

q' = $$\arg\min_{q:q(\mathbf{y})\geq 0, E_q[\mathbf{Uy}]\leq \mathbf{b}, \sum_{\mathbf{y}} q(\mathbf{y})=1} KL(q(\mathbf{y})||P(\mathbf{y}|\mathbf{x},\mathbf{w}))$$

# Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:
- $$y^* = \mathbf{argmax}_{y:Uy \le b}\ P_w(y|x)$$

Constraining the feasible set

- … It is possible to derive a constrained version of EM:
- To do that, constraints are relaxed into expectation constraints on the posterior probability q:

$$E_q[Uy] \le b$$

- The E-step now becomes: [Neal & Hinton '99 view of EM]

q' = $$\arg\min_{q: q(\mathbf{y}) \ge 0, E_q[\mathbf{Uy}] \le \mathbf{b}, \sum_{\mathbf{y}} q(\mathbf{y}) = 1} KL(q(\mathbf{y}) || P(\mathbf{y}|\mathbf{x}, \mathbf{w}))$$

- This is Taskar's Posterior Regularization [PR] [Ganchev et al, 10]

# Which (Constrained) EM to use?

- There is a lot of literature on EM vs hard EM
  - Experimentally, the bottom line is that with a good enough initialization point, hard EM is probably better (and more efficient).
    - **E.g., EM vs hard EM (Spitkovsky et al, 10)**
- Similar issues exist in the constrained case: CoDL vs. PR
  - The constraints view helped developing additional algorithmic insight
  - $\gamma$ that
    - Provides a continuum of algorithms – from EM to hard EM, and infinitely many new EM algorithms in between.
    - Implementation wise, not more complicated than EM

# Which (Constrained) EM to use?

$\gamma$ $\gamma\gamma$ that

There is a lot of literature on EM vs hard EM

- Experimentally, the bottom line is that with a good enough initialization point, hard EM is probably better (and more efficient).
  - **E.g., EM vs hard EM (Spitkovsky et al, 10)**
- Similar issues exist in the constrained case: CoDL vs. PR
  - The constraints view helped developing additional algorithmic insight
- Unified EM (UEM) [Samdani & Roth, NAACL-12]
  - Provides a continuum of algorithms – from EM to hard EM, and infinitely many new EM algorithms in between.
  - Implementation wise, not more complicated than EM
  - Implementation wise, not more complicated than EM

The third part of the tutorial is on how to do inference

$$\mathbf{y} = \mathrm{argmax}_{\mathbf{y} \in \mathcal{Y}} \; \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) + \mathbf{u}^T \mathcal{C}(\mathbf{x}, \mathbf{y})$$

The second part of the tutorial is on how to learn

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios

# LEARNING

The third part of the tutorial is on how to do inference

$$y = \text{argmax}_{y \in \mathcal{Y}} \; w^T\phi(x, y) + u^T C(x, y)$$

The second part of the tutorial is on how to learn

- The following (high level) examples will briefly present several **learning paradigms** where
  - The building blocks are the **learning algorithms** introduced later
  - **Inference** is necessary, as part of learning and the final decision.
- The focus is on scenarios where
  - There is a need to learn more than one model (combine via inference)
  - Semi-supervised scenarios
  - **Learning with Latent Structured Representations**
    - A meta-algorithm that makes use of structured learning algorithm s
    - Including approaches that make use of declarative constraints to minimize the  level of supervision using constraints
    - **[Chang et.al. ICML'10, NAACL'10,…]**

# INFERENCE

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}}\, w^T \phi(x_i, y)$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \text{argmax}_{y \in \mathcal{Y}} w^T \phi(x_i, y)$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

# INFERENCE

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = argmax_{y \in \mathcal{Y}} w^T \phi ( x_i , y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

# INFERENCE

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $\mathbf{y_i' = argmax_{y \in \mathcal{Y}}\, w^T \phi ( x_i ,y)}$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

# INFERENCE

- For each example $(x_i, y_i)$
- Do: (with the current weight vector w)
  - **Predict:** perform Inference with the current weight vector
    - $y_i' = \mathrm{argmax}_{y \in \mathcal{Y}} \, w^T \phi(x_i, y)$
  - **Check** the learning constraints
    - **Is the score of the current prediction better than of $(x_i, y_i)$?**
  - If **Yes** – a mistaken prediction
    - **Update w**
  - Otherwise: no need to update w on this example
- EndFor

- **Inference is done many times** – both at decision time (one inference per predicates…) and during training.

# Amortized ILP based Inference

- Imagine that you already solved many structured output inference problems
  - Co-reference resolution; Semantic Role Labeling; Parsing citations; Summarization; dependency parsing; image segmentation,…
  - Your solution method doesn't matter either

# Amortized ILP based Inference

- Imagine that you already solved many structured output inference problems
  - Co-reference resolution; Semantic Role Labeling; Parsing citations; Summarization; dependency parsing; image segmentation,…
  - Your solution method doesn’t matter either
- How can we exploit this fact to save inference cost?

> After solving **n** inference problems, can we make the (**n+1**)$^{th}$ one faster?

# Amortized ILP based Inference

- Imagine that you already solved many structured output inference problems
  - Co-reference resolution; Semantic Role Labeling; Parsing citations; Summarization; dependency parsing; image segmentation,…
  - Your solution method doesn't matter either
- How can we exploit this fact to save inference cost?

> After solving **n** inference problems, can we make the (**n+1**)$^{th}$ one faster?

- We will show how to do it when your problem is formulated as a 0-1 Linear Program:

# Amortized ILP based Inference

- Imagine that you already solved many structured output inference problems
  - Co-reference resolution; Semantic Role Labeling; Parsing citations; Summarization; dependency parsing; image segmentation,…
  - Your solution method doesn't matter either
- How can we exploit this fact to save inference cost?

> After solving **n** inference problems, can we make the (**n+1**)$^{th}$ one faster?

- We will show how to do it when your problem is formulated as a 0-1 Linear Program:

$$\text{Max } c \cdot x$$
$$Ax \leq b$$
$$x \in \{0,1\}$$

# Amortized ILP based Inference

- Imagine that you already solved many structured output inference problems
  - Co-reference resolution; Semantic Role Labeling; Parsing citations; Summarization; dependency parsing; image segmentation,…
  - Your solution method doesn't matter either
- How can we exploit this fact to save inference cost?

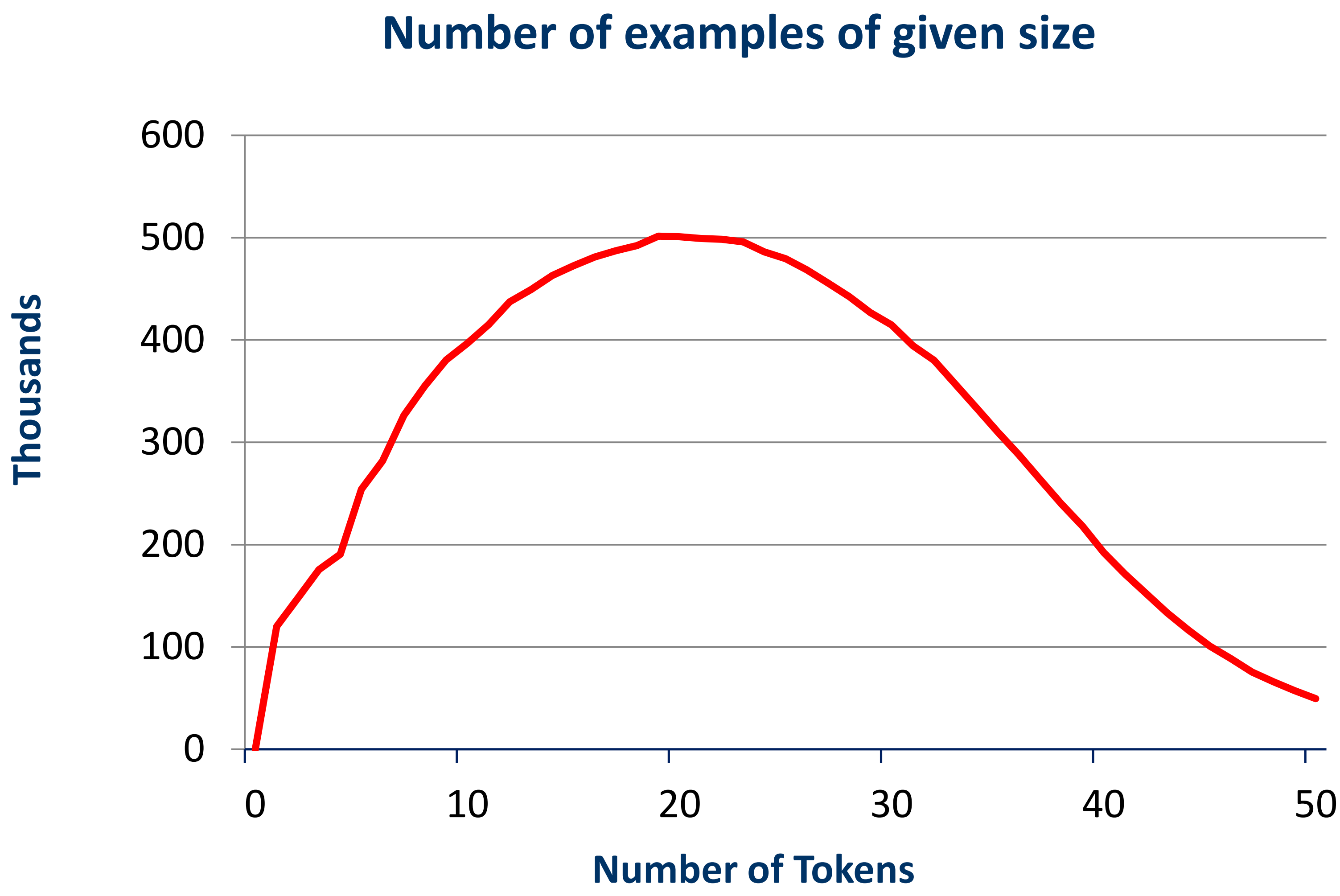
After solving **n** inference problems, can we make the (**n+1**)$^{th}$ one faster?

- We will show how to do it when your problem is formulated as a 0-1 Linear Program:

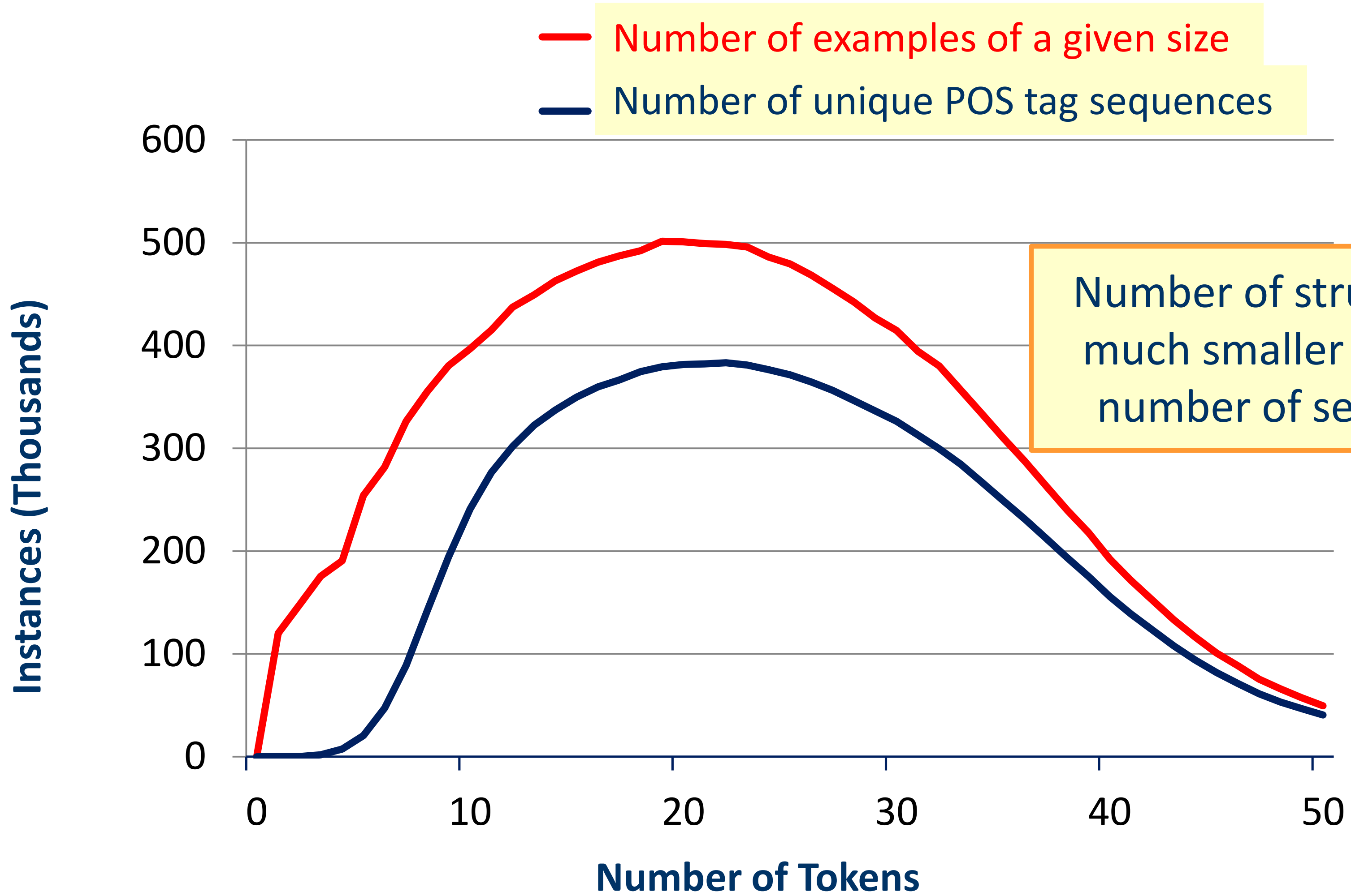
$$\text{Max } c \cdot x$$
$$Ax \leq b$$
$$x \in \{0,1\}$$

- Very general: All discrete MAP problems can be formulated as 0-1 LPs [Roth & Yih'04; Taskar '04]
- We only care about inference formulation, not algorithmic solution

# The Hope: POS Tagging on Gigaword

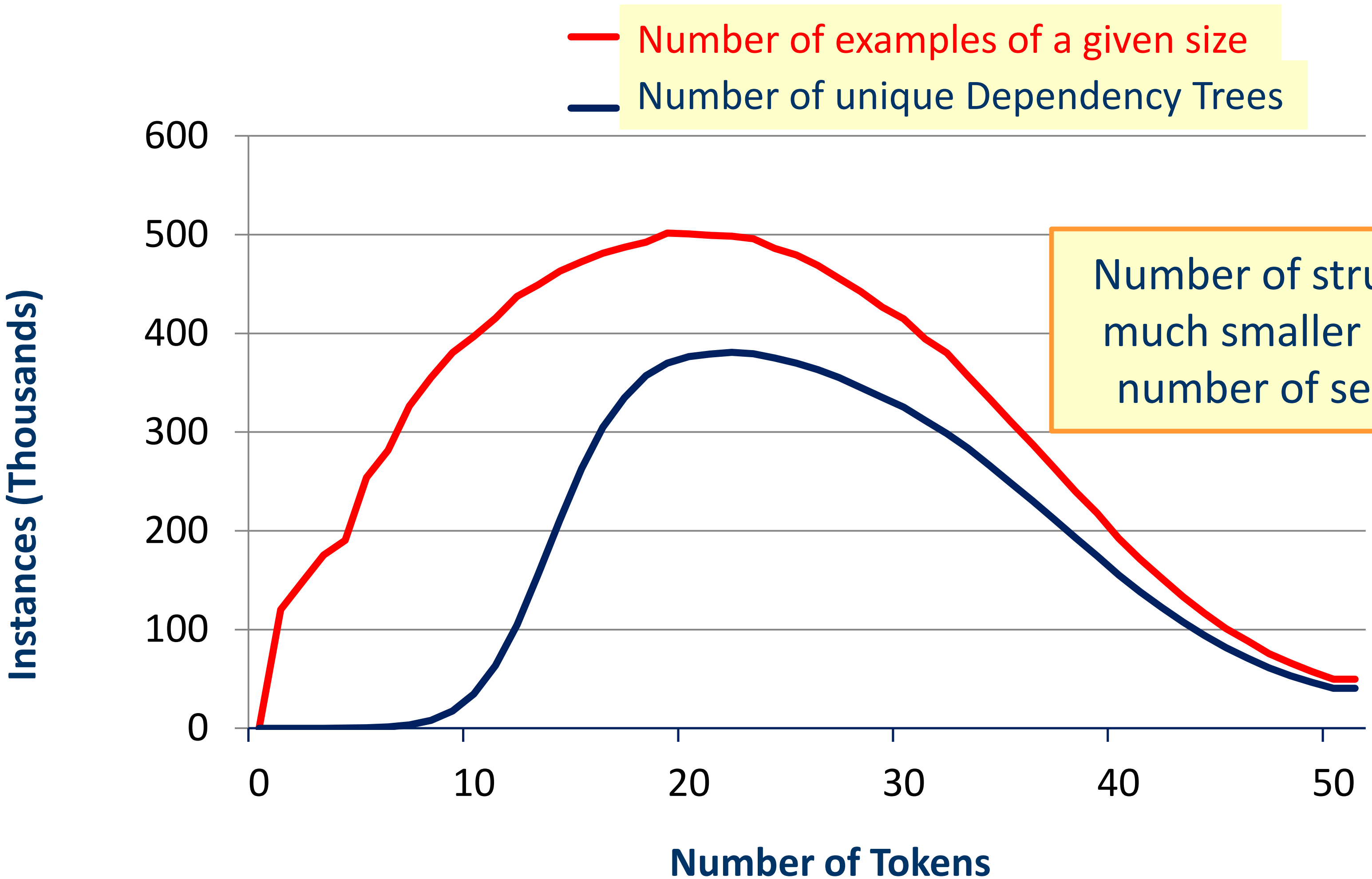


# The Hope: POS Tagging on Gigaword



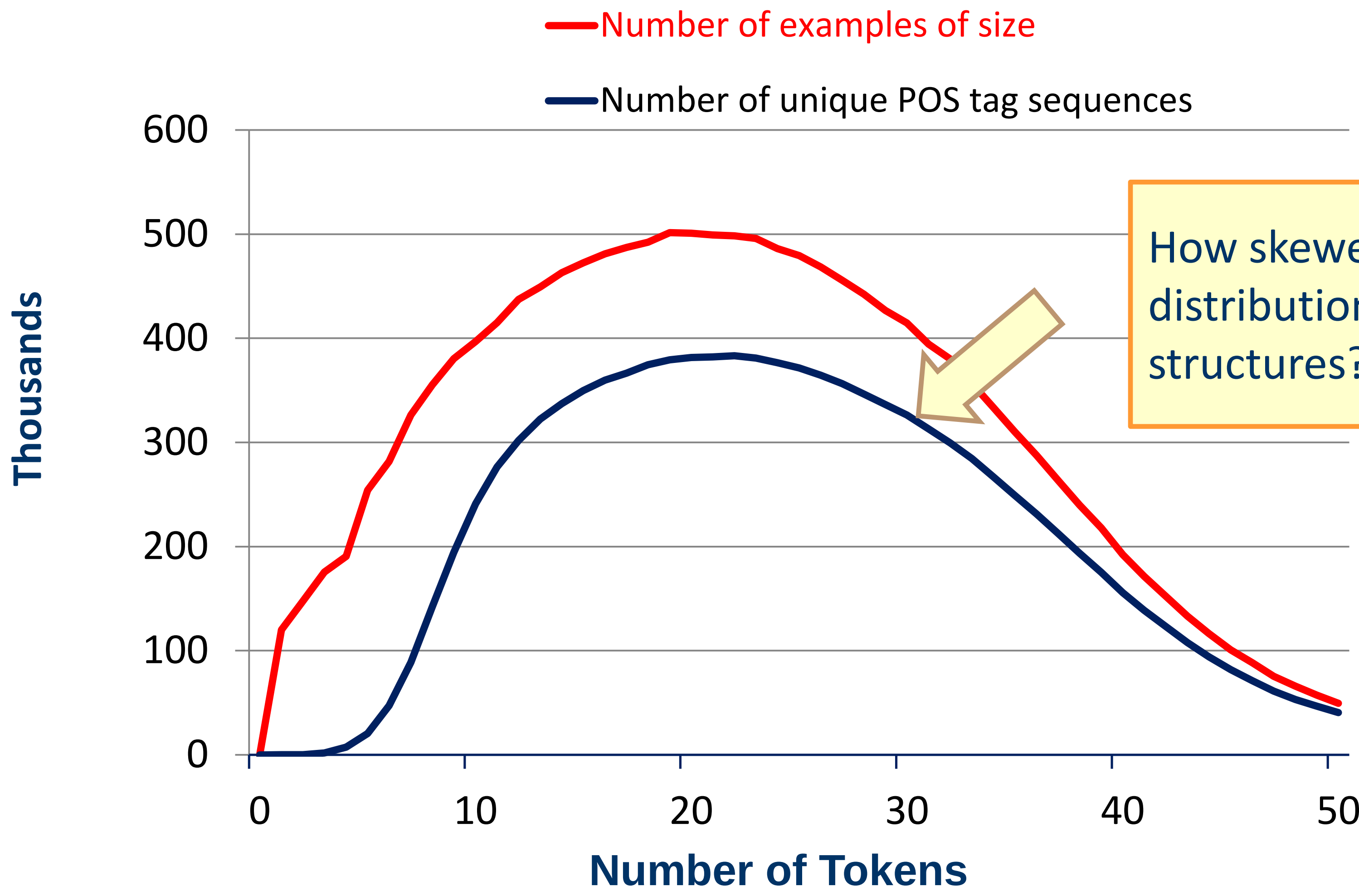

Number of structures is much smaller than the number of sentences

# The Hope: Dependency Parsing on Gigaword



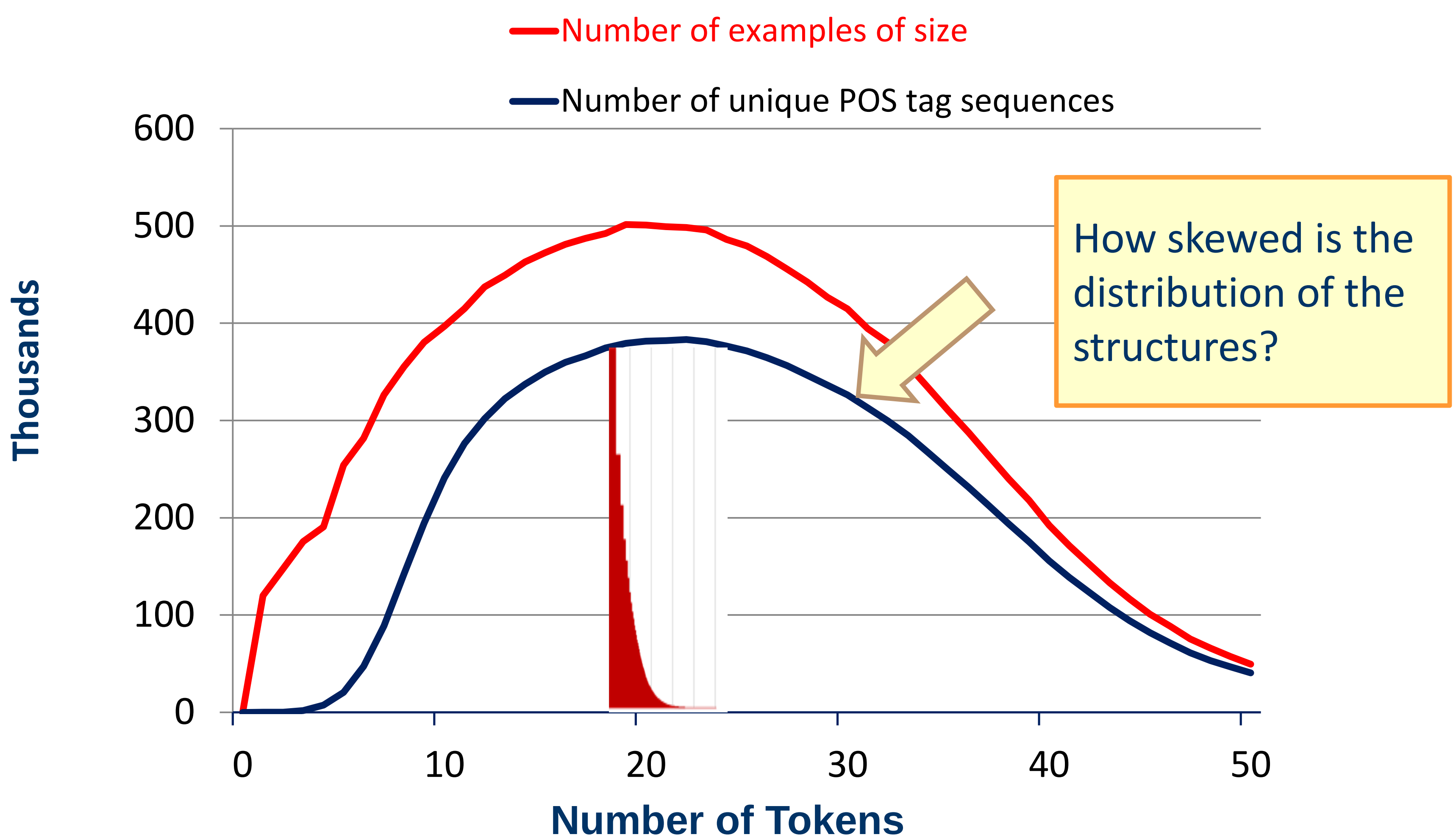

Number of structures is much smaller than the number of sentences

# POS Tagging on Gigaword

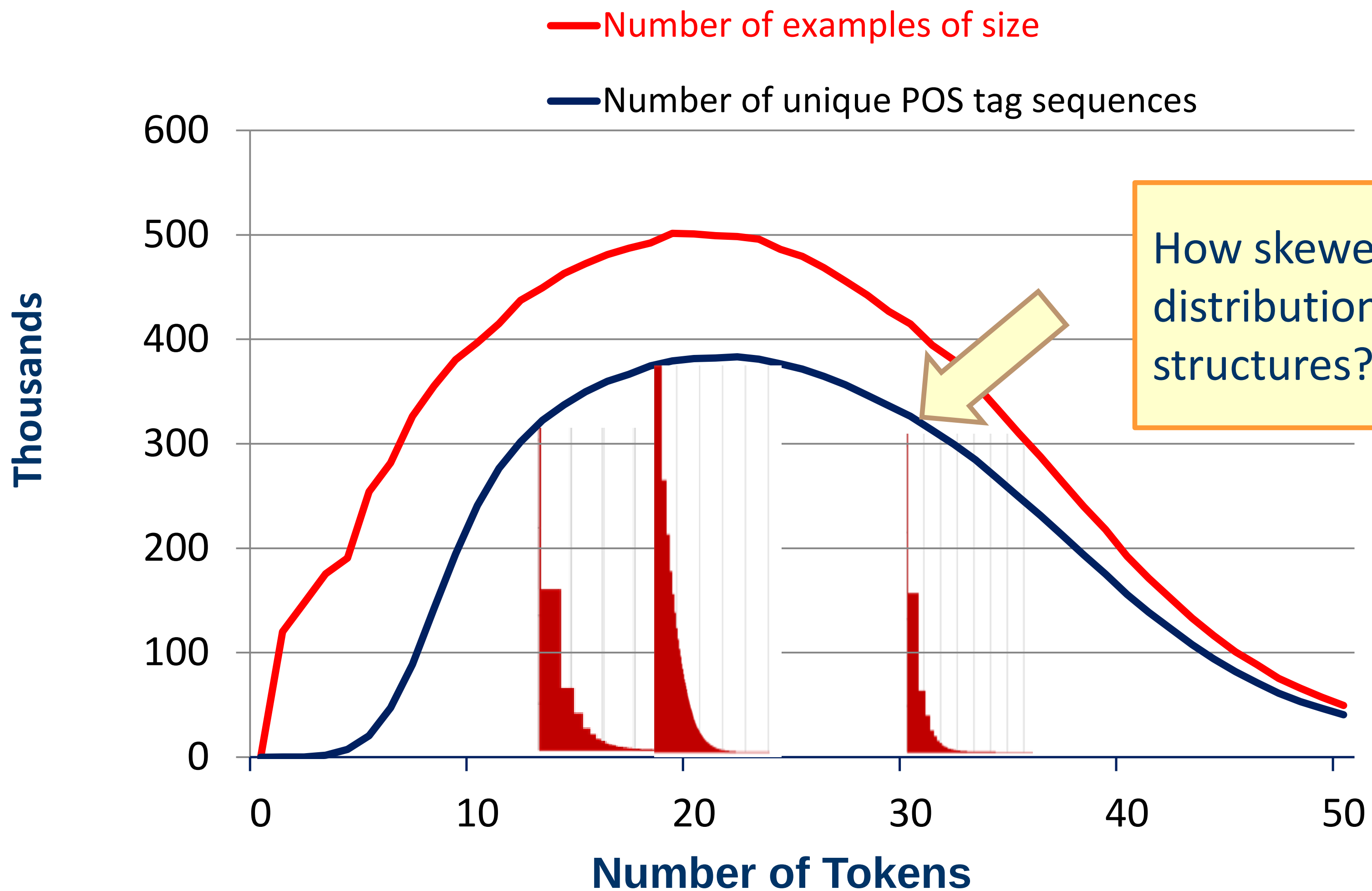



How skewed is the distribution of the structures?

# POS Tagging on Gigaword

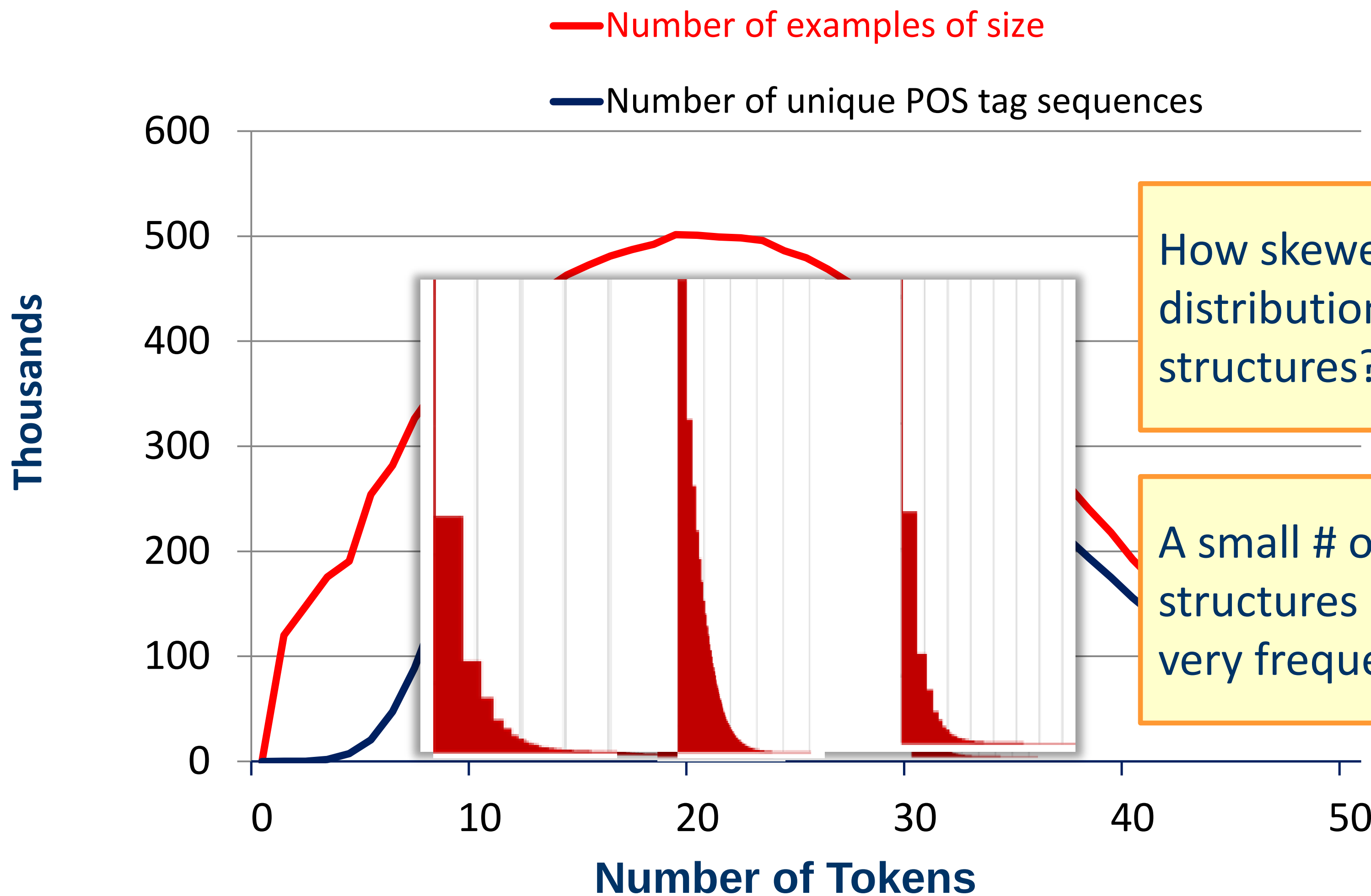


# POS Tagging on Gigaword




How skewed is the distribution of the structures?

# POS Tagging on Gigaword

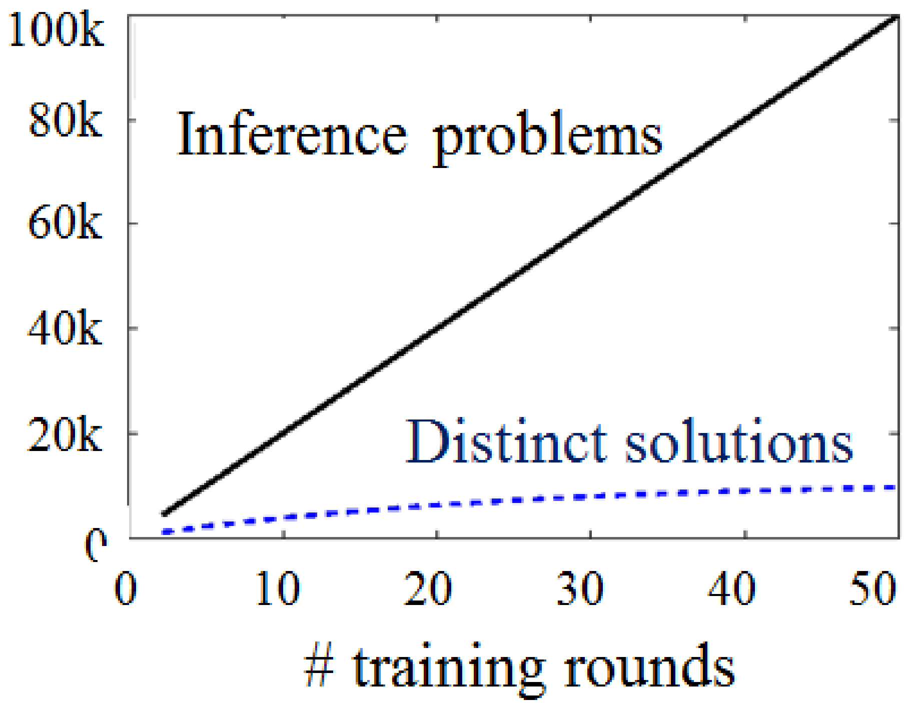



How skewed is the distribution of the structures?

A small # of structures occur very frequently

# Redundancy in Inference and Learning

- This redundancy is important since in all NLP tasks there is a need to solve many inferences, at least one per sentence.
- However, it is as important in structured learning, where algorithms cycle between
  - performing inference, and
  - updating the model.



# Amortized ILP Inference

- These statistics show that many different instances are mapped into identical inference outcomes.
  - Pigeon Hole Principle

# Amortized ILP Inference

- These statistics show that many different instances are mapped into identical inference outcomes.
  - Pigeon Hole Principle
- How can we exploit this fact to save inference cost over the life time of the learning & Inference program?

# Amortized ILP Inference

- These statistics show that many different instances are mapped into identical inference outcomes.
  - Pigeon Hole Principle
- How can we exploit this fact to save inference cost over the life time of the learning & Inference program?

We give conditions on the objective functions
(for all objectives with the same # or variables and same feasible set),
under which the solution of a new problem Q is the same as the one of P (which we already cached)

# Amortized ILP Inference

We argue here that the inference formulation provides a new level of abstraction.

- These statistics show that many different instances are mapped into identical inference outcomes.
  - Pigeon Hole Principle
- How can we exploit this fact to save inference cost over the life time of the learning & Inference program?

We give conditions on the objective functions
(for all objectives with the same # or variables and same feasible set),
under which the solution of a new problem Q is the same as the one of P (which we already cached)

# Amortized ILP Inference

We argue here that the inference formulation provides a new level of abstraction.

- These statistics show that many different instances are mapped into identical inference outcomes.
  - Pigeon Hole Principle
- How can we exploit this fact to save inference cost over the life time of the learning & Inference program?

We give conditions on the objective functions
(for all objectives with the same # or variables and same feasible set),
under which the solution of a new problem Q is the same as the one of P (which we already cached)

If **CONDITION** (**problem** ***cache***, *new problem*)
then (no need to call the solver)
**SOLUTION**(*new problem*) = old solution
Else
Call **base solver** and update ***cache***
End

0.04 ms

2 ms

# Speedup & Accuracy

$$\text{Speedup} = \frac{\text{number of inference calls without amortization}}{\text{number of inference calls with amortization}}$$

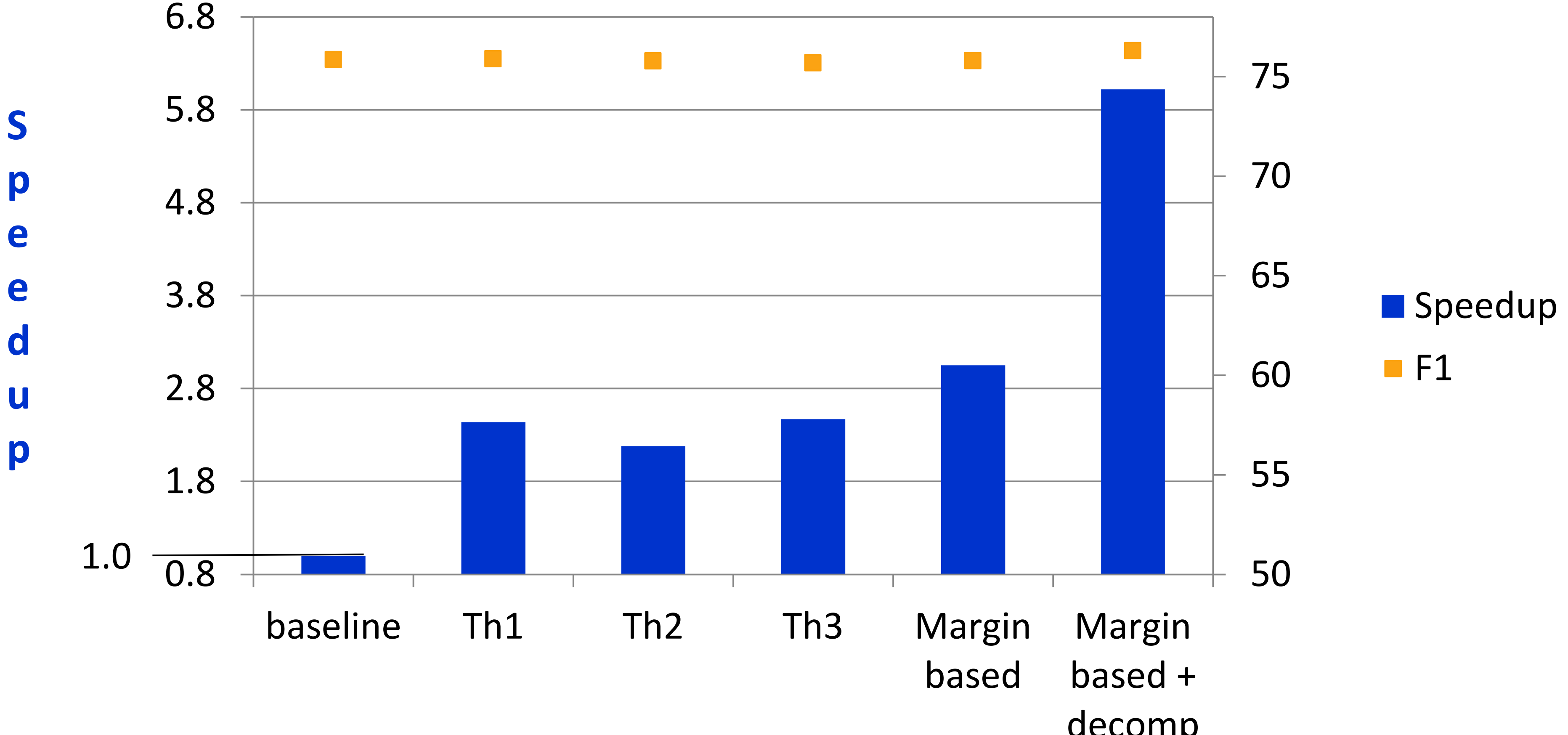
S
p
e
e
d
u
p

Amortization schemes [EMNLP'12, ACL'13, AAAI'15]

# Speedup & Accuracy

By decomposing the objective function, building on the fact that "smaller structures" are more redundant, it is possible to get even better results.

$$\text{Speedup} = \frac{\text{number of inference calls without amortization}}{\text{number of inference calls with amortization}}$$

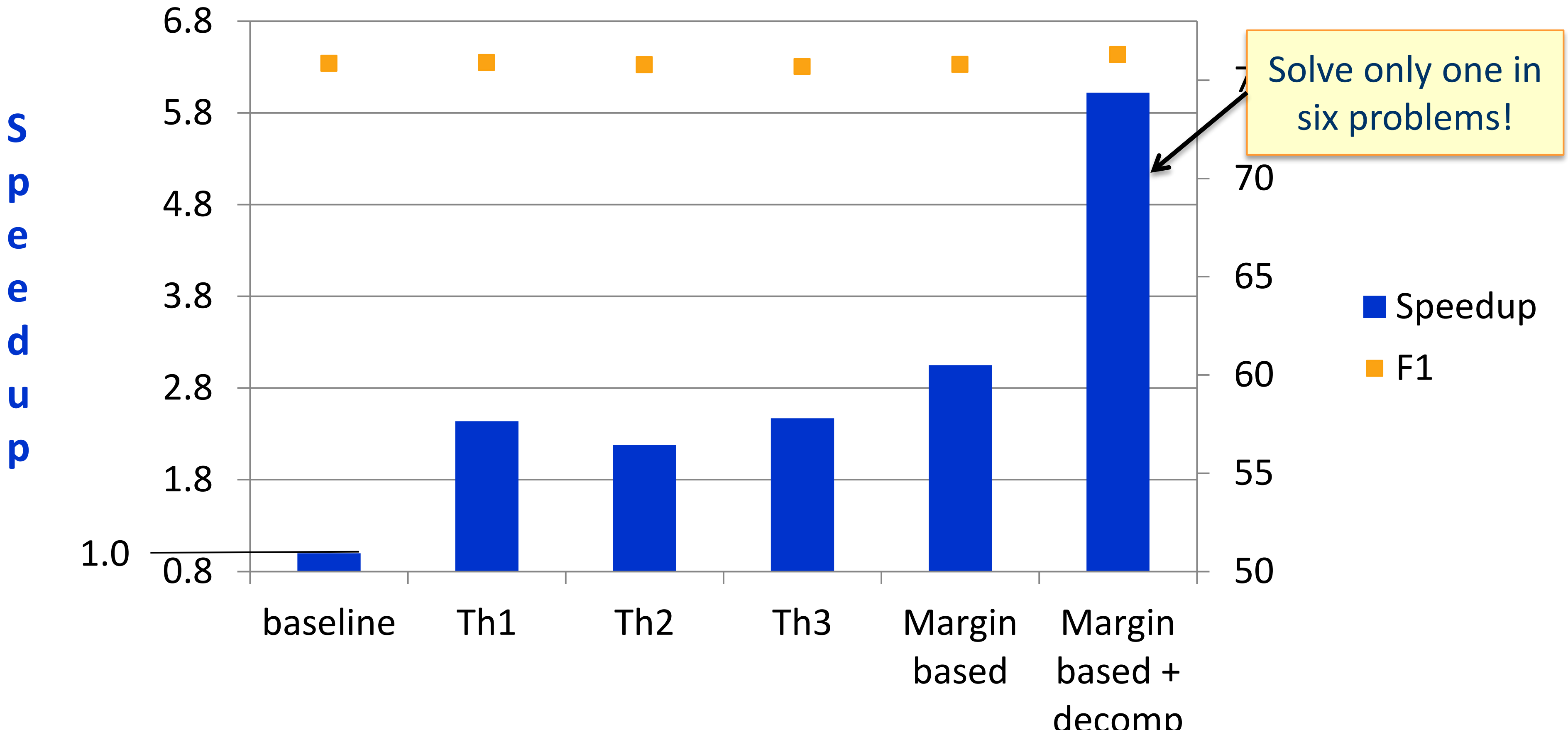



Amortization schemes [EMNLP'12, ACL'13, AAAI'15]

# Speedup & Accuracy

By decomposing the objective function, building on the fact that "smaller structures" are more redundant, it is possible to get even better results.

$$\text{Speedup} = \frac{\text{number of inference calls without amortization}}{\text{number of inference calls with amortization}}$$

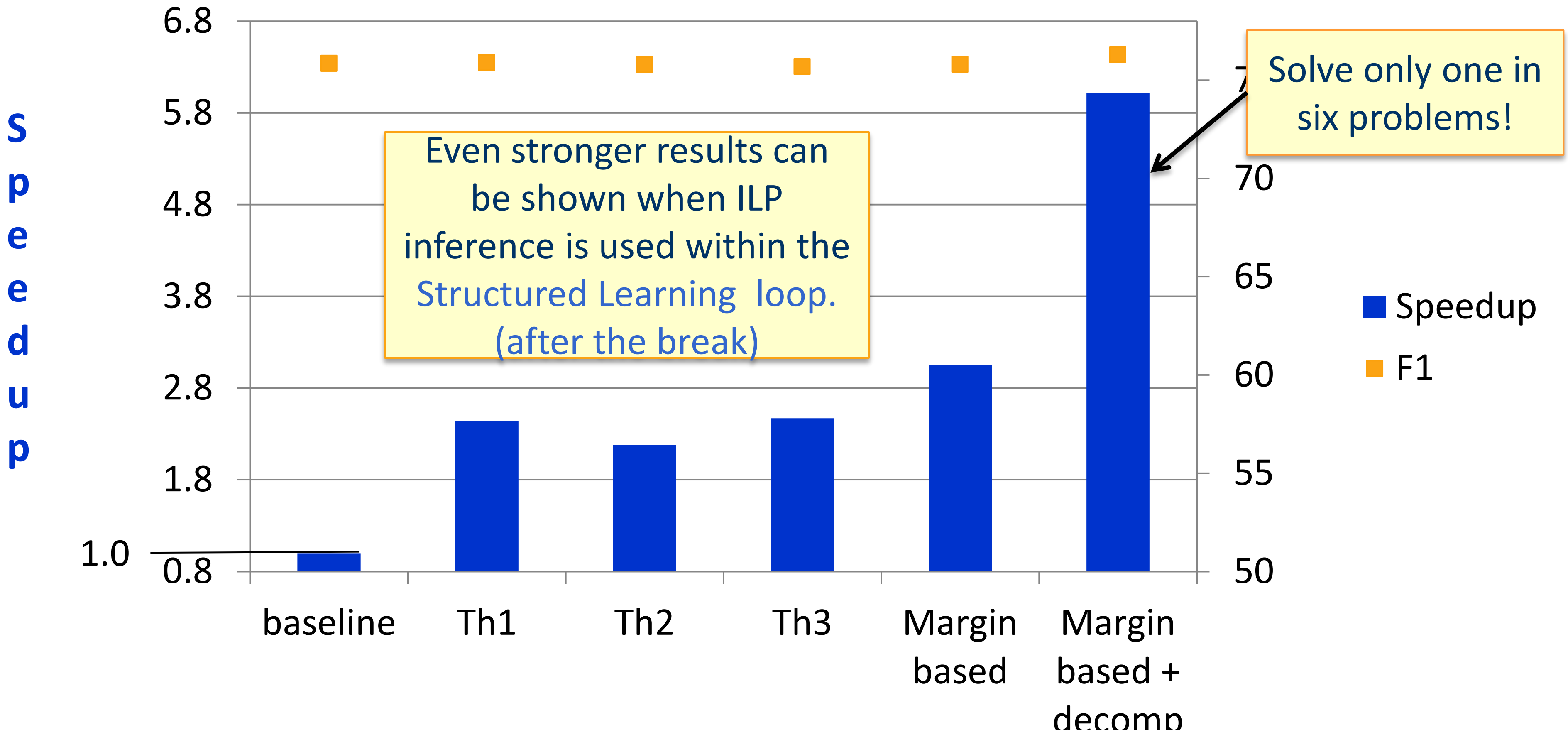



Amortization schemes [EMNLP'12, ACL'13, AAAI'15]

# Speedup & Accuracy

The results show that, indeed, the inference formulation provides a new level of abstraction that can be exploited to re-use solutions

$$\text{Speedup} = \frac{\text{number of inference calls without amortization}}{\text{number of inference calls with amortization}}$$




Amortization schemes [EMNLP'12, ACL'13, AAAI'15]

# First Summary

- Introduced Structured Prediction
- **Many examples**
- Introduced the key building blocks of **structured learning** and **inference**
- Focused on Constraints Conditional Models
- CCMS: The motivating scenario is the case in which
  - Joint INFERENCE is essential
  - Joint LEARNING should be done thoughtfully
    - **Not everything can be learned together**
    - **We don't always want to learn everything together**
- Moving on to
  - Details on Joint Learning
  - Details on Inference