Learning and Inference in Structured Prediction Models

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

MSR, IBM, Illinois, Utah
All interesting decisions are structured
All interesting decisions are structured
All interesting decisions are structured
All interesting decisions are structured

If 10 is added to two numbers, the first one will be 5 more than thrice the second.

Constituency-based parse tree

S
  / \
N   VP
  /   /
V   NP
 |   D
|   N
|   /
|  John hit the ball.
- All interesting decisions are structured.
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.
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.
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
Learning and Inference in Structured Prediction

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
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
Nice to Meet You

2

n+

u
Recognizing Entities and Relations

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

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$
Recognizing Entities and Relations

\[
\begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.85 \\
\text{loc} & 0.10 \\
\end{array}
\quad
\begin{array}{c|c}
\text{other} & 0.10 \\
\text{per} & 0.60 \\
\text{loc} & 0.30 \\
\end{array}
\quad
\begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.50 \\
\text{loc} & 0.45 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{irrelevant} & 0.05 \\
\text{spouse\_of} & 0.45 \\
\text{born\_in} & 0.50 \\
\end{array}
\quad
\begin{array}{c|c}
\text{irrelevant} & 0.10 \\
\text{spouse\_of} & 0.05 \\
\text{born\_in} & 0.85 \\
\end{array}
\]

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

\[E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3\]
Recognizing Entities and Relations

<table>
<thead>
<tr>
<th>Entity</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>0.05</td>
</tr>
<tr>
<td>per</td>
<td>0.85</td>
</tr>
<tr>
<td>loc</td>
<td>0.10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>0.10</td>
</tr>
<tr>
<td>per</td>
<td>0.60</td>
</tr>
<tr>
<td>loc</td>
<td>0.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>0.05</td>
</tr>
<tr>
<td>per</td>
<td>0.50</td>
</tr>
<tr>
<td>loc</td>
<td>0.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>irrelevant</td>
<td>0.05</td>
</tr>
<tr>
<td>spouse_of</td>
<td>0.45</td>
</tr>
<tr>
<td>born_in</td>
<td>0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>irrelevant</td>
<td>0.10</td>
</tr>
<tr>
<td>spouse_of</td>
<td>0.05</td>
</tr>
<tr>
<td>born_in</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Recognizing Entities and Relations

Bernie’s wife, Jane, is a native of Brooklyn.

<table>
<thead>
<tr>
<th>Role</th>
<th>E1</th>
<th>R12</th>
<th>E2</th>
<th>R23</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>per</td>
<td>0.85</td>
<td>0.60</td>
<td>0.50</td>
<td>0.50</td>
<td>0.85</td>
</tr>
<tr>
<td>loc</td>
<td>0.10</td>
<td>0.30</td>
<td>0.45</td>
<td>0.45</td>
<td>0.30</td>
</tr>
<tr>
<td>irrelevant</td>
<td>0.05</td>
<td>0.05</td>
<td>0.10</td>
<td>0.05</td>
<td>0.10</td>
</tr>
<tr>
<td>spouse_of</td>
<td>0.45</td>
<td>0.45</td>
<td>0.10</td>
<td>0.05</td>
<td>0.45</td>
</tr>
<tr>
<td>born_in</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Recognizing Entities and Relations

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

E₁ —→ R₁₂ —→ E₂ —→ R₂₃ —→ E₃

\[
\begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.85 \\
\text{loc} & 0.10 \\
\end{array}
\quad \begin{array}{c|c}
\text{other} & 0.10 \\
\text{per} & 0.60 \\
\text{loc} & 0.30 \\
\end{array}
\quad \begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.50 \\
\text{loc} & 0.45 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{irrelevant} & 0.05 \\
\text{spouse_of} & 0.45 \\
\text{born_in} & 0.50 \\
\end{array}
\quad \begin{array}{c|c}
\text{irrelevant} & 0.10 \\
\text{spouse_of} & 0.05 \\
\text{born_in} & 0.85 \\
\end{array}
\]

Joint Inference with General Constraint Structure [Roth & Yih’04,07,....]
Recognizing Entities and Relations

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

\[ E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3 \]

<table>
<thead>
<tr>
<th>Relation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>irrelevant</td>
<td>0.05</td>
</tr>
<tr>
<td>spouse_of</td>
<td>0.45</td>
</tr>
<tr>
<td>born_in</td>
<td>0.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>irrelevant</td>
<td>0.10</td>
</tr>
<tr>
<td>spouse_of</td>
<td>0.05</td>
</tr>
<tr>
<td>born_in</td>
<td>0.85</td>
</tr>
</tbody>
</table>
Recognizing Entities and Relations

### Entities:
- **E1**: Bernie’s wife, Jane
- **E2**: a native of
- **E3**: Brooklyn

### Relations:
- **R_{12}**: spouse_of
- **R_{23}**: born_in

### Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>0.05</td>
<td>0.10</td>
<td>0.05</td>
</tr>
<tr>
<td>per</td>
<td>0.85</td>
<td>0.60</td>
<td>0.50</td>
</tr>
<tr>
<td>loc</td>
<td>0.10</td>
<td>0.30</td>
<td>0.45</td>
</tr>
</tbody>
</table>

- **E1**
  - irrelevant: 0.05
  - spouse_of: 0.45
  - born_in: 0.50

- **E2**
  - irrelevant: 0.10
  - spouse_of: 0.05

- **E3**
  - irrelevant: 0.10
  - born_in: 0.85
Recognizing Entities and Relations

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

Joint inference gives good improvement
Joint Inference with General Constraint Structure [Roth&Yih’04,07,....]

Recognizing Entities and Relations

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

\[ \begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.85 \\
\text{loc} & 0.10 \\
\end{array} \quad \begin{array}{c|c}
\text{other} & 0.10 \\
\text{per} & 0.60 \\
\text{loc} & 0.30 \\
\end{array} \quad \begin{array}{c|c}
\text{other} & 0.05 \\
\text{per} & 0.50 \\
\text{loc} & 0.45 \\
\end{array} \]

Key Questions:
How to learn the model(s)?
What is the source of the knowledge?
How to guide the global inference?

Joint inference gives good improvement
Recognizing Entities and Relations

Joint inference gives good improvement

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

Key Questions:
- How to learn the model(s)?
- What is the source of the knowledge?
- How to guide the global inference?

Bernie’s wife, Jane, is a native of Brooklyn.

E1 \[\rightarrow\] R_{12} \[\rightarrow\] E2

R_{23}

Models could be learned separately/jointly; constraints may come up only at decision time.
Joint Inference with General Constraint Structure

Recognizing Entities and Relations

Models could be learned separately/jointly; constraints may come up only at decision time.
Most problems are not single classification problems
Pipeline

Raw Data

- Most problems are not single classification problems
Pipeline

Raw Data → Most problems are not single classification problems → POS Tagging → Phrases → Semantic Entities → Relations → Parsing → WSD → Semantic Role Labeling

Motivation I
Most problems are not single classification problems

Raw Data -> POS Tagging -> Phrases -> Semantic Entities -> Relations
Pipeline

Most problems are not single classification problems

- Conceptually, Pipelining is a crude approximation
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.
Pipeline

- 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?

- Most problems are not single classification problems

Raw Data

- POS Tagging  ➔ Phrases  ➔ Semantic Entities  ➔ Relations
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

Most problems are not single classification problems

- Conceptually, Pipelining is a crude approximation
  - Interactions occur across levels and downstream 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?
Conceptually, Pipelining is a crude approximation

- Interactions occur across levels and downstream 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?

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

**Most problems are not single classification problems**
Example 2: Object detection

Right facing bicycle
Example 2: Object detection

Right facing bicycle
saddle/seat
left wheel
handle bar
right wheel
Example 2: Object detection

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

saddle/seat

Right facing bicycle

left wheel

handle bar

right wheel
One approach to build this structure

Left wheel detector: Is there a wheel in this box? Binary classifier
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, ...

### Prediction result of a trained HMM

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>[AUTHOR]</td>
</tr>
<tr>
<td>EDITOR</td>
<td></td>
</tr>
<tr>
<td>BOOKTITLE</td>
<td></td>
</tr>
<tr>
<td>TECH-REPORT</td>
<td></td>
</tr>
<tr>
<td>INSTITUTION</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
</tbody>
</table>

Prediction result of a trained HMM


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?

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
Adding constraints, we get **correct** results!

- Without changing the model

[AUTHOR] Lars Ole Andersen.
[TITLE] Program analysis and specialization for the C Programming language.
[INSTITUTION] DIKU, University of Copenhagen,
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

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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]
Constrained Conditional Models

\[ y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) + u^T C(x, y) \]
Constrained Conditional Models

\[ y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) + u^T C(x, y) \]

Weight Vector for “local” models
Constrained Conditional Models

\[ y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) + u^T C(x, y) \]

Weight Vector for “local” models

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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

Weight Vector for “local” models

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

(Soft) constraints component
Constrained Conditional Models

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

- Weight Vector for "local" models
- Features, classifiers; log-linear models (HMM, CRF) or a combination
- Penalty for violating the constraint.
- (Soft) constraints component
- How far \( y \) is from a "legal" assignment
Constrained Conditional Models

\[ y = \arg\max_{y \in \mathcal{Y}} \ w^T \phi(x, y) + u^T C(x, y) \]

**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

**Penalty for violating the constraint.**

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

**Weight Vector for “local” models.**

**How far \( y \) is from a “legal” assignment.**

**(Soft) constraints component.**
Inference: given input $\mathbf{X}$ (a document, a sentence),

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

- Assign values to the $y_1, y_2, ..., y_n$, accounting for dependencies among $y_i$s
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: 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

$$y' = \arg\max_{y \in Y} \mathbf{w}^T \phi (x, y)$$
Structured Prediction: Inference

- Inference: given input $\mathbf{X}$ (a document, a sentence),

  predict the best structure $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\} \in Y$ (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

  $$y' = \arg\max_{y \in \mathcal{Y}} w^T \phi (\mathbf{x}, y)$$

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

Joint features on inputs and outputs
Structured Prediction: Inference

- Inference: given input $\mathbf{x}$ (a document, a sentence),

  predict the best structure $\mathbf{y} = \{y_1, y_2, \ldots, y_n\} \in Y$ (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

  $$y' = \arg\max_{y \in Y} w^T \phi(x, y)$$

Joint features on inputs and outputs

Feature Weights (estimated during learning)
Inference: given input $\mathbf{x}$ (a document, a sentence),

predict the best structure $\mathbf{y} = \{y_1, y_2, \ldots, y_n\} \in \mathbf{Y}$ (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

$$\mathbf{y}' = \arg\max_{\mathbf{y} \in \mathbf{Y}} \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y})$$

Set of allowed structures

Feature Weights (estimated during learning)

Joint features on inputs and outputs
Inference: given input $X$ (a document, a sentence),
predict the best structure $Y = \{y_1, y_2, \ldots, y_n\} \subseteq Y$ (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

$$y' = \arg\max_{y \in Y} w^T \phi(x, y)$$

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 $w$
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

$$y' = \arg\max_{y \in Y} w^T \phi(x, y)$$

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

find a scoring function \(w\) that minimizes empirical loss.
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 \( (x_i, y_i) \):

  \[
  \text{Score of annotated structure} \geq \text{Score of any other structure} + \text{Penalty for predicting other structure}
  \]
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 \((x_i, y_i)\):

\[
 w^T \phi(x_i, y_i) \geq w^T \phi(x_i, y) + \Delta(y, y_i)
\]
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 \((x_i, y_i)\):

\[
\forall y \quad w^T \phi(x_i, y_i) \geq w^T \phi(x_i, y) + \Delta(y, y_i)
\]
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 \((x_i, y_i)\):

\[
\forall y \quad w^T \phi(x_i, y_i) \geq w^T \phi(x_i, y) + \Delta(y, y_i)
\]

- 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 \((x_i, y_i)\):

\[
\forall y \quad w^T \phi(x_i, y_i) \geq w^T \phi(x_i, y) + \Delta(y, y_i)
\]

- 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 \((x_i, y_i)\):

\[
\forall y \quad w^T \phi(x_i, y_i) \geq w^T \phi(x_i, y) + \Delta(y, y_i)
\]

- 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
    \[ y'_i = \arg\max_{y \in \mathcal{Y}} w^T \phi (x_i, y) \]
  - **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
    - $y_i' = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y)$
  - **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
      - \(y'_i = \arg\max_{y \in Y} w^T \phi(x_i, y)\)

    - 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
    - \(y'_i = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y)\)
  - **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
    \[ y'_i = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y) \]
  - **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

For each example \((x_i, y_i)\)

Do:

- **Predict**: perform Inference with the current weight vector
  \[ y_i' = \arg\max_{y \in \mathcal{Y}} w_{\text{EASY}}^T \phi_{\text{EASY}}(x_i, y) + w_{\text{HARD}}^T \phi_{\text{HARD}}(x_i, y) \]

- **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

Solution I: decompose the scoring function to EASY and HARD parts
Structured Prediction: Learning Algorithm

- For each example \((x_i, y_i)\)
- Do:
  - **Predict**: perform Inference with the current weight vector
    - \(y_i' = \arg\max_{y \in Y} w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y)\)
  - **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

**Solution I**: decompose the scoring function to \(EASY\) and \(HARD\) parts

**\(EASY\)**: could be feature functions that correspond to an HMM, a linear CRF, or even \(\phi_{EASY}(x, y) = \phi(x)\), omitting 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

- For each example \((x_i, y_i)\)
- Do:
  - **Predict**: perform Inference with the current weight vector
    \[ y_i' = \arg\max_{y \in \mathcal{Y}} w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y) \]
  - **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

Solution II: Disregard some of the dependencies: assume a simple model.
Structured Prediction: Learning Algorithm

- For each example \((x_i, y_i)\)
- Do:
  - **Predict**: perform Inference with the current weight vector
    \[ y'_i = \arg\max_{y \in Y} w_{EASY}^T \phi_{EASY} (x_i, y) + w_{HARD}^T \phi_{HARD} (x_i, y) \]
  - **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

Solution II: Disregard some of the dependencies: assume a simple model.
Structured Prediction: Learning Algorithm

- For each example \( (x_i, y_i) \)
- Do:
  - **Predict**: perform Inference with the current weight vector
    \[ y_i' = \arg\max_{y \in \mathcal{Y}} \ w_{EASY}^T \phi_{EASY} (x_i, y) + w_{HARD}^T \phi_{HARD} (x_i, y) \]
  - **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
    \[ y_i' = \arg\max_{y \in \mathcal{Y}} w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y) \]
  - **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**

\[ y_i' = \arg\max_{y \in \mathcal{Y}} w_{EASY}^T \phi_{EASY}(x_i, y) + w_{HARD}^T \phi_{HARD}(x_i, y) \]
Structured Prediction: Learning Algorithm

- For each example \((x_i, y_i)\)
- Do:
  - Predict: perform Inference with the current weight vector
    \[ y'_i = \arg\max_{y \in \mathcal{Y}} w_{\text{EASY}}^T \phi_{\text{EASY}} (x_i, y) + w_{\text{HARD}}^T \phi_{\text{HARD}} (x_i, y) \]
  - 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

\[ y'_i = \arg\max_{y \in \mathcal{Y}} w_{\text{EASY}}^T \phi_{\text{EASY}} (x_i, y) + w_{\text{HARD}}^T \phi_{\text{HARD}} (x_i, y) \]

This is the most commonly used solution in NLP today

Solution III: Disregard some of the dependencies during learning; take into account at decision time
y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)
y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)
Constrained Conditional Models

\[ y = \text{argmax}_{y \in \mathcal{Y}} \ w^T \phi(x, y) \]

Features, classifiers; log-linear models (HMM, CRF) or a combination
y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y)

Weight Vector for “local” models

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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

Knowledge component: (Soft) constraints

Weight Vector for “local” models

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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

- **Knowledge component:** (Soft) constraints
- **Penalty for violating the constraint.**
- **Features, classifiers; log-linear models (HMM, CRF) or a combination**
- **How far \( y \) is from a “legal/expected” assignment**
- **Weight Vector for “local” models**
**Constrained Conditional Models**

\[
y = \arg\max_{y \in \mathcal{Y}} \ w^T \phi(x, y) + u^T \! C(x, y)
\]

- **Training:** learning the objective function \((w, u)\)

  - Decouple? Decompose? Force \(u\) to model hard constraints?

- **Penalty for violating the constraint.**

- **Knowledge component:** (Soft) constraints

- **How far \(y\) is from a “legal/expected” assignment**

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

- **Weight Vector for “local” models**
Constrained Conditional Models

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))]

\[
y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y)
\]
Constrained Conditional Models

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

- **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))]

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

\[
\arg \max_{y \in Y} \sum_{p \in \Gamma_x} 1[y_p = y_p] w^T \Phi_p (x, y_p) + u^T C (x, y_p)
\]

- **Knowledge component**: (Soft) constraints
- **Features, classifiers; log-linear models (HMM, CRF) or a combination**
- **How far \(y\) is from a “legal/expected” assignment**

- **Weight Vector for “local” models**
Constrained Conditional Models

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)]

Variables are “parts”

\[
\arg \max_{y \in Y} \sum_{p \in \Gamma_x} 1_{[y_p = y_p]} w^T \Phi_p (x, y_p) \]

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

Roth+ 04, Taskar 04

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

Page 28
Constrained Conditional Models

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**.

\[
\arg\max_{y \in Y} \sum_{p \in \Phi_x} 1[y_p = y_p] w^T \Phi_p(x, y_p)
\]
Examples: CCM Formulations

\[ y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) + u^T C(x, y) \]
Examples: CCM Formulations

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

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.
Examples: CCM Formulations

\[ y = \arg\max_{y \in \mathcal{Y}} w^T \phi(x, y) + u^T C(x, y) \]

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.
Examples: CCM Formulations

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

$$y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y)$$

The second part of the tutorial is on how to learn.
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.

The second part of the tutorial is on how to learn.
y = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \phi(x, y) + \mathbf{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.
Examples: CCM Formulations

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.

\[
y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y)
\]

The second part of the tutorial is on how to learn.

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

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

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)

Sequential Prediction

HMM/CRF based:

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

Knowledge/Linguistics Constraints

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

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

The second part of the tutorial is on how to learn.
Examples: CCM Formulations

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

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)

Knowledge/Linguistics Constraints

- If a modifier chosen, include its head
- If verb is chosen, include its arguments

The second part of the tutorial is on how to learn

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

Sentence Compression/Summarization:
Language Model based:
$$\text{Argmax} \sum \lambda_{ijk} x_{ijk}$$

Knowledge/Linguistics Constraints
If a modifier chosen, include its head
If verb is chosen, include its arguments
Examples: CCM Formulations

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

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.

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

The second part of the tutorial is on how to learn.

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.

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

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

I left my pearls to my daughter in my will.
Semantic Role Labeling (SRL)

I left my pearls to my daughter in my will.

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

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

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

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

\[
\arg\max \sum_{a,t} y_{a,t} c_{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}
\]

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
    
    \[
    \arg\max \sum_{a,t} y_{a,t} c_{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}
    \]

    Subject to:
    - One label per argument: \( \sum_t y_{a,t} = 1 \)
    - No overlapping or embedding
    - Relations between verbs and arguments,....

Variable \( y_{a,t} \) indicates whether candidate argument \( a \) is assigned a label \( t \).
\( c_{a,t} \) is the corresponding model score
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

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

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 by the argument classifier

\[
\text{argmax} \sum_{a,t} y_{a,t} c_{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}
\]

Subject to:
- One label per argument: \( \sum_t y_{a,t} = 1 \)
- No overlapping or embedding
- Relations between verbs and arguments, ....

Variable \( y_{a,t} \) indicates whether candidate argument \( a \) is assigned a label \( t \).
\( c_{a,t} \) is the corresponding model score.

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

I left my nice pearls to her
**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

\[
\text{argmax} \sum_{a,t} y_{a,t} c_{a,t} = \sum_{a,t} 1_{a=t} c_{a=t}
\]

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

- 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

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

- No duplicate argument classes
  - Any Boolean rule can be encoded as a set of linear inequalities.
  - If there is an Reference-Ax phrase, there is an Ax

- Reference-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
- Reference-Ax
- Continuation-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

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

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

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

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

- No duplicate argument classes

**Reference-Ax**

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

**Continuation-Ax**

- If there is a 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

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.

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

- No duplicate argument classes

\[ \forall y \in Y, \sum_{i=0}^{n-1} 1 \{ y_i = y \} \leq 1 \]

- Reference-Ax

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

- Continuation-Ax

If there is a 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

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

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

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

- No duplicate argument classes
  \[
  \forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1\{y_i = y\} \leq 1
  \]

- Reference-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

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

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

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

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

- No duplicate argument classes

\[ \forall y \in Y, \sum_{i=0}^{n-1} 1\{y_i = y\} \leq 1 \]

- Reference-Ax

\[ \forall y \in Y_R, \sum_{i=0}^{n-1} 1\{y_i = y = \text{“R-Ax”}\} \leq \sum_{i=0}^{n-1} 1\{y_i = \text{“Ax”}\} \]

- Continuation-Ax

\[ \forall j, y \in Y_C, 1\{y_j = y = \text{“C-Ax”}\} \leq \sum_{i=0}^{j} 1\{y_i = \text{“Ax”}\} \]

- 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

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

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

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

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

The tutorial web page will point to material on how to write down linear inequalities for various logical expressions.
SRL: Posing the Problem

maximize \( \sum_{i=0}^{n-1} \sum_{y \in Y} \lambda_{x_i, y} \mathbb{1}_{y_i = y} \)

where \( \lambda_{x, y} = \lambda \cdot F(x, y) = \lambda_y \cdot F(x) \)

subject to

<table>
<thead>
<tr>
<th>A</th>
<th>bomb [A1]</th>
<th>killer [A0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bomb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>that</td>
<td>bomb</td>
<td></td>
</tr>
<tr>
<td>(Reference)</td>
<td>[R-A1]</td>
<td></td>
</tr>
<tr>
<td>exploded</td>
<td>V: explode</td>
<td></td>
</tr>
<tr>
<td>outside</td>
<td>location</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>[AM-LOC]</td>
<td></td>
</tr>
<tr>
<td>U.S.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>military</td>
<td>temporal</td>
<td></td>
</tr>
<tr>
<td>base</td>
<td>[AM-TMP]</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>location</td>
<td></td>
</tr>
<tr>
<td>Benijji</td>
<td>[AM-LOC]</td>
<td></td>
</tr>
<tr>
<td>killed</td>
<td>V: kill</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>corpse [A1]</td>
</tr>
<tr>
<td>Iraqi citizens</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SRL: Posing the Problem

maximize \( \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{x_i,y} \mathbb{1}_{\{y_i=y\}} \)

where \( \lambda_{x,y} = \lambda \cdot F(x, y) = \lambda_y \cdot F(x) \)

subject to \( \forall i, \sum_{y \in \mathcal{Y}} \mathbb{1}_{\{y_i=y\}} = 1 \)

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

\( \forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} \mathbb{1}_{\{y_i=y=\text{"R-Ax"}\}} \leq \sum_{i=0}^{n-1} \mathbb{1}_{\{y_i=\text{"Ax"}\}} \)

\( \forall j, y \in \mathcal{Y}_C, \mathbb{1}_{\{y_j=y=\text{"C-Ax"}\}} \leq \sum_{i=0}^{j} \mathbb{1}_{\{y_i=\text{"Ax"}\}} \)
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i) \]

\[ P(y_0) \quad P(y_1 | y_0) \quad P(y_2 | y_1) \quad P(y_3 | y_2) \quad P(y_4 | y_3) \]

\[ P(x_0 | y_0) \quad P(x_1 | y_1) \quad P(x_2 | y_2) \quad P(x_3 | y_3) \quad P(x_4 | y_4) \]
Example 2: Sequence Tagging

\[ y^* = \arg\max_{y} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

Here, \( y \)'s are labels; \( x \)'s are observations.
**Example 2: Sequence Tagging**

HMM:

\[
    y^* = \arg\max_{y \in \mathcal{Y}} P(y_0) P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1}) P(x_i|y_i)
\]

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: \[ y^* = \arg\max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

Here, \( y \)'s are labels; \( x \)'s are observations.

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

Example: the man saw the dog
Example 2: Sequence Tagging

HMM:

\[
y^* = \arg \max_{y \in Y} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i)
\]

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:

\[ y^* = \arg \max_{y \in Y} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

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:

\[ y^* = \arg \max_{y \in Y} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

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 \( y_0 = D \land y_1 = ? \).
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

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 \( y_0 = D \land y_1 = ? \).

Every assignment to the \( y \)'s is a path.
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg\max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

Example: the man saw the dog
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i) \]

As an ILP:

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

subject to
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1}) P(x_i|y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \quad \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\ 
\text{subject to} & \quad \lambda_{0,y} = \log(P(y)) + \log(P(x_0|y)) \\ & \quad \lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))
\end{align*}
\]
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} & \\
\sum_{y \in \mathcal{Y}} \lambda_{0,y} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} = 1 \\
\lambda_{0,y} &= \log(P(y)) + \log(P(x_0 | y)) \\
\lambda_{i,y,y'} &= \log(P(y | y')) + \log(P(x_i | y))
\end{align*}
\]
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1}) P(x_i|y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \quad \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} & \quad \\
\end{align*}
\]

\[
\begin{align*}
\lambda_{0,y} &= \log(P(y)) + \log(P(x_0|y)) \\
\lambda_{i,y,y'} &= \log(P(y|y')) + \log(P(x_i|y))
\end{align*}
\]
Example 2: Sequence Tagging

HMM:
\[
y^* = \arg\max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)
\]

As an ILP:
maximize
\[
\sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\}
\]
subject to
\[
\lambda_{0,y} = \log(P(y)) + \log(P(x_0|y)) \quad \lambda_{i,y,y'} = \log(P(y|y')) + \log(P(x_i|y))
\]
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in Y} P(y_0)P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1})P(x_i | y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \quad \sum_{y \in Y} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in Y} \sum_{y' \in Y} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} & \quad \lambda_{0,y} = \log(P(y)) + \log(P(x_0 | y)) \\
& \quad \lambda_{i,y,y'} = \log(P(y | y')) + \log(P(x_i | y)) \\
& \quad 1\{y_0 = \text{“NN”}\} = 1 \\
& \quad 1\{y_0 = \text{“VB”}\} = 1 \\
& \quad 1\{y_0 = \text{“JJ”}\} = 1
\end{align*}
\]
Example 2: Sequence Tagging

**HMM:**

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

**As an ILP:**

\[
\begin{align*}
\text{maximize} \quad & \sum_{y \in \mathcal{Y}} \lambda_{0,y} \cdot 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} \cdot 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} \quad & \sum_{y \in \mathcal{Y}} 1\{y_0 = y\} = 1
\end{align*}
\]

\[
\begin{align*}
\lambda_{0,y} &= \log(P(y)) + \log(P(x_0|y)) \\
\lambda_{i,y,y'} &= \log(P(y|y')) + \log(P(x_i|y))
\end{align*}
\]

*Unique label for each word*
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg\max_{y \in \mathcal{Y}} \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

As an ILP:

maximize \[ \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0=y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i=y \land y_{i-1}=y'\} \]

subject to \[ \sum_{y \in \mathcal{Y}} 1\{y_0=y\} = 1 \]

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

Unique label for each word
HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0)P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1})P(x_i | y_i) \]

As an ILP:

\[
\text{maximize } \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to } \sum_{y \in \mathcal{Y}} 1\{y_0 = y\} = 1
\]

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

Unique label for each word

\[
1\{y_0 = "NN"\} = 1 \\
1\{y_0 = "DT" \land y_1 = "JJ" \} = 1 \\
1\{y_1 = "NN" \land y_2 = "VB" \} = 1
\]
Example 2: Sequence Tagging

**HMM:**

\[
y^* = \arg \max_{y \in Y} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i)
\]

As an ILP:

maximize \[
\sum_{y \in Y} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in Y} \sum_{y' \in Y} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\}
\]

subject to

\[
\sum_{y \in Y} 1\{y_0 = y\} = 1
\]

\[
\forall y, \ 1\{y_0 = y\} = \sum_{y' \in Y} 1\{y_0 = y \land y_0 = y'\}
\]

\[
\forall y, i > 1 \sum_{y' \in Y} 1\{y_{i-1} = y' \land y_i = y\} = \sum_{y'' \in Y} 1\{y_i = y \land y_{i+1} = y''\}
\]

Unique label for each word

Edges that are chosen must form a path
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg\max_{y \in \mathcal{Y}} P(y_0) P(x_0 | y_0) \prod_{i=1}^{n-1} P(y_i | y_{i-1}) P(x_i | y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \quad \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} & \quad \sum_{y \in \mathcal{Y}} 1\{y_0 = y\} = 1 \\
& \quad \forall y, 1\{y_0 = y\} = \sum_{y' \in \mathcal{Y}} 1\{y_0 = y \land y_1 = y'\} \\
& \quad \forall y, i > 1 \sum_{y' \in \mathcal{Y}} 1\{y_{i-1} = y' \land y_i = y\} = \sum_{y'' \in \mathcal{Y}} 1\{y_i = y \land y_{i+1} = y''\}
\end{align*}
\]

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

Unique label for each word

Edges that are chosen must form a path
Example 2: Sequence Tagging

HMM:

\[
y^* = \arg \max_{y \in \mathcal{Y}} P(y_0) P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1}) P(x_i|y_i)
\]

As an ILP:

\[
\begin{align*}
\text{maximize} & \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0=y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i=y \land y_{i-1}=y'\} \\
\text{subject to} & \sum_{y \in \mathcal{Y}} 1\{y_0=y\} = 1 \\
& \forall y, \ 1\{y_0=y\} = \sum_{y' \in \mathcal{Y}} 1\{y_0=y \land y_1=y'\} \\
& \forall y, i > 1 \ \ \sum_{y' \in \mathcal{Y}} 1\{y_{i-1}=y' \land y_i=y\} = \sum_{y'' \in \mathcal{Y}} 1\{y_i=y \land y_{i+1}=y''\}
\end{align*}
\]

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

Unique label for each word

Edges that are chosen must form a path
Example 2: Sequence Tagging

HMM:

\[
y^* = \arg \max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i)
\]

As an ILP:

maximize \[ \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0=y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i=y \land y_{i-1}=y'\} \]

subject to

\[ \sum_{y \in \mathcal{Y}} 1\{y_0=y\} = 1 \]

\[ \forall y, \ 1\{y_0=y\} = \sum_{y' \in \mathcal{Y}} 1\{y_0=y \land y_1=y'\} \]

\[ \forall y, i > 1 \sum_{y' \in \mathcal{Y}} 1\{y_{i-1}=y' \land y_i=y\} = \sum_{y'' \in \mathcal{Y}} 1\{y_i=y \land y_{i+1}=y''\} \]

\[ 1\{y_0=“V”\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} 1\{y_{i-1}=y \land y_i=“V”\} \geq 1 \]

Example: the man saw the dog

Edges that are chosen must form a path

Unique label for each word

There must be a verb!
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg \max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

As an ILP:

\[
\begin{align*}
\text{maximize} & \quad \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0 = y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i = y \land y_{i-1} = y'\} \\
\text{subject to} & \quad \sum_{y \in \mathcal{Y}} 1\{y_0 = y\} = 1 \\
& \quad \forall y, 1\{y_0 = y\} = \sum_{y' \in \mathcal{Y}} 1\{y_0 = y \land y_1 = y'\} \\
& \quad \forall y, i > 1, \sum_{y' \in \mathcal{Y}} 1\{y_{i-1} = y' \land y_i = y\} = \sum_{y'' \in \mathcal{Y}} 1\{y_i = y \land y_{i+1} = y''\} \\
& \quad 1\{y_0 = "V"\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} 1\{y_{i-1} = y \land y_i = "V"\} \geq 1
\end{align*}\n\]

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

Example: the man saw the dog

Edges that are chosen must form a path.

Unique label for each word.

There must be a verb!
Example 2: Sequence Tagging

HMM:

\[ y^* = \arg\max_{y \in \mathcal{Y}} P(y_0)P(x_0|y_0) \prod_{i=1}^{n-1} P(y_i|y_{i-1})P(x_i|y_i) \]

As an ILP:

\[
\begin{align*}
&\text{maximize} & \sum_{y \in \mathcal{Y}} \lambda_{0,y} 1\{y_0=y\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} \sum_{y' \in \mathcal{Y}} \lambda_{i,y,y'} 1\{y_i=y \land y_{i-1}=y'\} \\
&\text{subject to} & \sum_{y \in \mathcal{Y}} 1\{y_0=y\} = 1 \\
& & \forall y, \sum_{y' \in \mathcal{Y}} 1\{y_0=y \land y_1=y'\} = \sum_{y'' \in \mathcal{Y}} 1\{y_i=y \land y_{i+1}=y''\} \\
& & 1\{y_0=“V”\} + \sum_{i=1}^{n-1} \sum_{y \in \mathcal{Y}} 1\{y_{i-1}=y \land y_i=“V”\} \geq 1
\end{align*}
\]

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

Example: the man saw the dog

Edges that are chosen must form a path

Unique label for each word

There must be a verb!

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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]
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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]
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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]
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

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]

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

The second part of the tutorial is on how to learn
maximize \[ \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{x_i, y} \mathbb{1}_{\{y_i = y\}} \]

where \[ \lambda_{x, y} = \lambda \cdot F(x, y) = \lambda_y \cdot F(x) \]

subject to

\[ \text{A} \quad \text{bom [A1]} \quad \text{killer [A0]} \]

\[ \text{car} \quad \text{bom} \quad \text{bom [R-A1]} \]

\[ \text{that} \quad \text{(Reference)} \quad \text{explicated [R-A1]} \]

\[ \text{V: explode} \quad \text{location [AM-LOC]} \]

\[ \text{outside} \quad \text{the} \quad \text{U.S.} \]

\[ \text{military} \quad \text{base} \quad \text{in} \]

\[ \text{Benjji} \quad \text{killed} \quad \text{11} \]

\[ \text{Iraqi} \quad \text{citizens} \]

\[ \text{V: kill} \quad \text{corpse [A1]} \]
Verb SRL

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=0}^{n-1} \sum_{y \in \mathcal{Y}} \lambda_{x_i, y} 1\{y_i = y\} \\
\text{where} & \quad \lambda_{x, y} = \lambda \cdot F(x, y) = \lambda_y \cdot F(x) \\
\text{subject to} & \quad \forall i, \sum_{y \in \mathcal{Y}} 1\{y_i = y\} = 1 \\
& \quad \forall y \in \mathcal{Y}, \sum_{i=0}^{n-1} 1\{y_i = y\} \leq 1 \\
& \quad \forall y \in \mathcal{Y}_R, \sum_{i=0}^{n-1} 1\{y_i = y \text{“R-Ax”}\} \leq \sum_{i=0}^{n-1} 1\{y_i = \text{“Ax”}\} \\
& \quad \forall j, y \in \mathcal{Y}_C, 1\{y_j = y \text{“C-Ax”}\} \leq \sum_{i=0}^{j} 1\{y_i = \text{“Ax”}\}
\end{align*}
\]
John, a fast-rising politician, slept on the train to Chicago.

Verb Predicate: sleep
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
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?
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
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?
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
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] 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
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
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
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.
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

Verb SRL is not Sufficient

- 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 Challenges

- Predict the preposition relations
  - [EMNLP, ‘11]
- Identify the relation’s arguments
- 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
The touchdown scored by Bradford cemented the victory of the Eagles.

<table>
<thead>
<tr>
<th>Verb</th>
<th>Nominalization</th>
<th>Preposition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicate:</strong> score</td>
<td><strong>Predicate:</strong> win</td>
<td>Sense: 11(6)</td>
</tr>
<tr>
<td><strong>A0:</strong> Bradford (scorer)</td>
<td><strong>A0:</strong> the Eagles (winner)</td>
<td>“the object of the preposition is the object of the underlying verb of the nominalization”</td>
</tr>
<tr>
<td><strong>A1:</strong> The touchdown (points scored)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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

- 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

- 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)
  - Done via global inference via CCM

Verb SRL is not Sufficient

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

Input & relation

Argument & their types
Joint inference (CCMs)

Verb arguments

$$\max_y \sum_t \sum_a y^{a,t} c^{a,t}$$
Joint inference (CCMs)

Verb arguments

\[
\max_y \sum_t \sum_a y^{a,t} c^{a,t}
\]

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

$$\max_y \sum_t \sum_a y_{a,t} c_{a,t}$$

Argument candidates

Each argument label
Joint inference (CCMs)

Verb arguments

\[
\max_y \sum_t \sum_a y^{a,t} c^{a,t}
\]

Constraints:

Verb SRL constraints

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

**Verb arguments**

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

**Preposition relations**

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

**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

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

Constraints:

Verb SRL constraints

Preposition relations

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

Preposition relation label

Preposition
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**

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

**Preposition relations**

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

**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

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

Preposition relations

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.

Constraints:

Verb SRL constraints

Preposition SRL Constraints

+$\text{Joint constraints between tasks; easy with ILP formulations}$
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**

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

**Preposition relations**

**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.

Constraints:

Verb arguments

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

Preposition relations

Verb SRL constraints

Preposition SRL Constraints

Joint constraints between tasks; easy with ILP formulations

Joint Inference – no (or minimal) joint learning
The bus was heading to Nairobi in Kenya.
Joint inference over phenomena–specific models to enforce consistency

Models trained with latent structure: senses, types, arguments
Extended SRL [Demo]

<table>
<thead>
<tr>
<th>SRL</th>
<th>Preposition</th>
<th>Preposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>leader [A0]</td>
<td></td>
</tr>
<tr>
<td>bus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>was</td>
<td></td>
<td></td>
</tr>
<tr>
<td>heading</td>
<td>V: head</td>
<td>Governor</td>
</tr>
<tr>
<td>to</td>
<td></td>
<td>Destination</td>
</tr>
<tr>
<td>Nairobi</td>
<td>Destination [A1]</td>
<td>Object</td>
</tr>
<tr>
<td>in</td>
<td></td>
<td>Object</td>
</tr>
<tr>
<td>Kenya</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Joint inference over phenomena–specific models to enforce consistency

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

Models trained with latent structure: senses, types, arguments
<table>
<thead>
<tr>
<th>AUTHOR</th>
<th>Lars Ole Andersen</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>Program analysis and specialization for the C Programming language. PhD thesis. DIKU, University of Copenhagen, May 1994</td>
</tr>
<tr>
<td>EDITOR</td>
<td></td>
</tr>
<tr>
<td>BOOKTITLE</td>
<td></td>
</tr>
<tr>
<td>TECH-REPORT</td>
<td></td>
</tr>
<tr>
<td>INSTITUTION</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
</tbody>
</table>

Prediction result of a trained HMM

[AUTHOR] Lars Ole Andersen. Program analysis and specialization for the C Programming language.
[EDITOR] DIKU, University of Copenhagen.
Prediction result of a trained HMM

Prediction result of a trained HMM

Information Extraction without Output Expectations


\[
\arg \max_y \lambda \cdot F(x, y)
\]

Prediction result of a trained HMM


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?

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

\[
\arg\max_y \lambda \cdot F(x, y)
\]

[AUTHOR] Lars Ole Andersen.
[TITLE] Program analysis and specialization for the C Programming language.
[INSTITUTION] DIKU, University of Copenhagen.
Adding constraints, we get **correct results!**

- Without changing the model

\[
\arg\max_y \lambda \cdot F(x, y)
\]

- **[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

\[ \arg \max_y \lambda \cdot F(x, y) - \sum_{i=1}^{K} \rho_i d(y, 1_{C_i(x)}) \]

[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

Seed examples → Model → Un-labeled Data
Guiding (Semi-Supervised) Learning with Constraints

Seed examples → Model → Un-labeled Data
In traditional Semi-Supervised learning the model can drift away from the correct one.
In traditional Semi-Supervised learning the model can drift away from the correct one.
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.
For N iterations do

\( T = \emptyset \)

For each \( x \) in unlabeled dataset

\[ h \leftarrow \arg\max_y w^T \phi(x,y) - \sum \rho d_C(x,y) \]

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

\((w, \rho) = \gamma (w, \rho) + (1 - \gamma) \text{learn}(T)\)
Constraints Driven Learning (CoDL)

For N iterations do

\( T = \phi \)

For each \( x \) in unlabeled dataset

\[
    h \leftarrow \mathop{\text{argmax}}_{y} \ w^{T} \ \phi(x, y) - \sum \ \rho \ \delta_{C}(x, y)
\]

\[
    T = T \cup \{(x, h)\}
\]

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

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

Supervised learning algorithm parameterized by \((w, \rho)\). [LATER]
For N iterations do

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

\[ T = \phi \]

For each \( x \) in unlabeled dataset

\[ h \leftarrow \arg\max_y w^T \phi(x, y) - \sum \rho \ d_C(x, y) \]

\[ T = T \cup \{(x, h)\} \]

\[ (w, \rho) = \gamma (w, \rho) + (1 - \gamma) \text{learn}(T) \]
(w, ρ) = learn(L)

For N iterations do

T = φ

For each x in unlabeled dataset

h ← argmax_y w^T φ(x, y) - ∑ ρ d_C(x, y)

T = T ∪ {(x, h)}

(w, ρ) = γ (w, ρ) + (1 - γ) learn(T)

Supervised learning algorithm parameterized by (w, ρ). [LATER]

Inference with constraints: augment the training set

Learn from new training data
Weigh supervised & unsupervised models.

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

\[ T = \emptyset \]

For each \( x \) in unlabeled dataset

\[ h \leftarrow \arg\max_y w^T \phi(x,y) - \sum \rho \, d_C(x,y) \]

\[ T = T \cup \{(x, h)\} \]

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

Constraints Driven Learning (CoDL)

Archetypical Semi/un-supervised learning: A constrained EM

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

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

Inference with constraints: augment the training set

Learn from new training data
Weigh supervised & unsupervised models.
For N iterations do

\[ T = \phi \]

For each \( x \) in unlabeled dataset

\[ h \leftarrow \arg\max_y w^T \phi(x, y) - \sum \rho d_c(x, y) \]

\[ T = T \cup \{(x, h)\} \]

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

Excellent Experimental Results showing the advantages of using constraints, especially with small amounts of labeled data [Chang et al, Others]
Objective function: \[ f_{\Phi,C}(x, y) = \sum w_i \phi_i(x, y) - \sum \rho_i d_{C_i}(x, y). \]
Value of Constraints in Semi-Supervised Learning

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

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

Learning w/o Constraints: 300 examples. 
Learning w 10 Constraints

# of available labeled examples
Value of Constraints in Semi-Supervised Learning

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

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.

Learning w/o Constraints: 300 examples.
Learning w/ 10 Constraints

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
  \[ h = y^* = \arg\max_y P_w(y|x) \]
- 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
  \[ h = y^* = \operatorname{argmax}_y P_w(y|x) \]
- 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:
  \[ y^* = \operatorname{argmax}_{y : U_y \leq b} P_w(y|x) \]
Constrained EM: Two Versions

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

\[ y^* = \text{argmax}_{y: U_y \leq b} P_w(y|x) \]

... It is possible to derive a constrained version of EM:

To do that, constraints are relaxed into expectation constraints on the posterior probability \( q \):
While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:

\[ y^* = \arg\max_{y : Uy \leq b} P_w(y | x) \]

... It is possible to derive a constrained version of EM:

To do that, constraints are relaxed into expectation constraints on the posterior probability \( q \):

\[ E_q[Uy] \leq b \]
While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:

\[ y^* = \arg\max_{y: U_y \leq b} P_w(y|x) \]

... It is possible to derive a constrained version of EM:

To do that, constraints are relaxed into expectation constraints on the posterior probability \( q \):

\[ E_q[U_y] \leq b \]

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

\[ q' = \arg\min_{q: q(y) \geq 0, E_q[U_y] \leq b, \sum_y q(y) = 1} KL(q(y)||P(y|x, w)) \]
Constrained EM: Two Versions

- While Constrained-Driven Learning [CODL; Chang et al, 07,12] is a constrained version of hard EM:

\[ y^* = \arg\max_{y: Uy \leq b} P_w(y|x) \]

- 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]

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

- 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

- 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?

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 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
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 algorithms
  - Including approaches that make use of declarative constraints to minimize the level of supervision using constraints
  - [Chang et.al. ICML’10, NAACL’10,...]

\[ y = \arg\max_{y \in Y} w^T \phi(x, y) + u^T C(x, y) \]
For each example \((x_i, y_i)\)

Do: (with the current weight vector \(w\))

- **Predict**: perform Inference with the current weight vector

  \[ y'_i = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y) \]

- **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
  \[ y'_i = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y) \]

- **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
    \[ y_i' = \arg\max_{y \in \mathcal{Y}} w^T \phi(x_i, y) \]
  - **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
    
    \[
    y'_i = \arg\max_{y \in Y} w^T \phi(x_i, y)
    \]

  - **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
  \[ y'_i = \arg\max_{y \in Y} \ w^T \phi(x_i, y) \]

- **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.
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?
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:
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:

$$\begin{align*}
\text{Max } c \cdot x \\
Ax &\leq b \\
x &\in \{0,1\}
\end{align*}$$
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
Number of examples of given size

Thousands

Number of Tokens

The Hope: POS Tagging on Gigaword
The Hope: POS Tagging on Gigaword

Number of examples of a given size

Number of unique POS tag sequences

Number of structures is much smaller than the number of sentences
Number of examples of a given size

Number of unique Dependency Trees

Number of structures is much smaller than the number of sentences
How skewed is the distribution of the structures?
POS Tagging on Gigaword

How skewed is the distribution of the structures?
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
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).
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).
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 lifetime of the learning & Inference program?

We give conditions on the objective functions (for all objectives with the same number of 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 \( \text{CONDITION}\) (problem cache, new problem)
then (no need to call the solver)

\[ \text{SOLUTION(new problem)} = \text{old solution} \]

Else

Call base solver and update cache

End
Speedup & Accuracy

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

Amortization schemes [EMNLP’12, ACL’13, AAAI’15]
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]
By decomposing the objective function, building on the fact that “smaller structures” are more redundant, it is possible to get even better results.

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

Solve only one in six problems!

Amortization schemes [EMNLP’12, ACL’13, AAAI’15]
By decomposing the objective function, building on the fact that “smaller structures” are more redundant, it is possible to get even better results.

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

Even stronger results can be shown when ILP inference is used within the Structured Learning loop. (after the break)

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 INFEERENCE 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