Cognitive Computation Group
University of Illinois at Urbana-Champaign

# Learning Based Programming:

## Facilitating the Programming of Data Driven Software Systems

Dan Roth

Department of Computer Science

University of Illinois at Urbana—Champaign

**June 2014**
**ICML AutoML Workshop**

Cognitive Computation Group
University of Illinois at Urbana-Champaign

# Learning Based Programming:

## Facilitating the Programming of Data Driven Software Systems


Dan Roth

Department of Computer Science

University of Illinois at Urbana—Champaign



With thanks to:

Collaborators: **Nick Rizzolo, Ming-Wei Chang, Kai-Wei Chang, Scott Yih, Parisa Kordjamshidi; Many others**

Funding: NSF; DHS; NIH; DARPA; IARPA, ARL, ONR
DASH Optimization (Xpress-MP); Gurobi.

# A Hypothetical Surveillance Program

```
Image i = captureImage()
for (Person p in i):
      if isMasked(p) and (isRunning(p) or hasGun(p)):
            soundAlarm()
```

# A Hypothetical Surveillance Program

```
Image i = captureImage()
for (Person p in i):
      if isMasked(p) and (isRunning(p) or hasGun(p)):
            soundAlarm()
```

- Simple! …except we have to
  - **detect people**
  - **determine if they are masked**
  - **determine if they are running**
  - **determine if they have a gun**

# A Hypothetical Surveillance Program

```
Image i = captureImage()
for (Person p in i):
      if isMasked(p) and (isRunning(p) or hasGun(p)):
            soundAlarm()
```

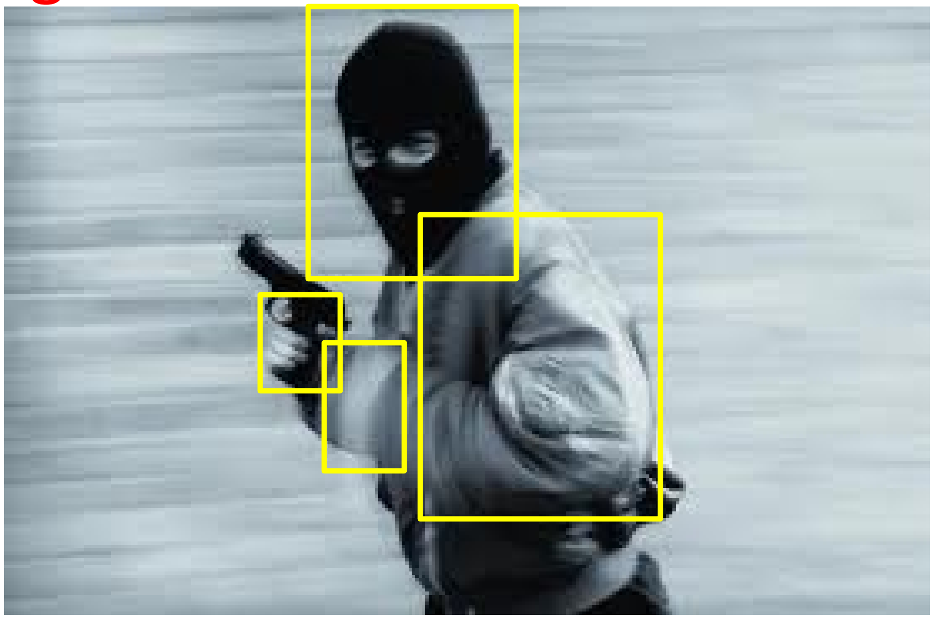
- Simple!  …except we have to
  - **detect people**
  - **determine if they are masked**
  - **determine if they are running**
  - **determine if they have a gun**

All hard problems

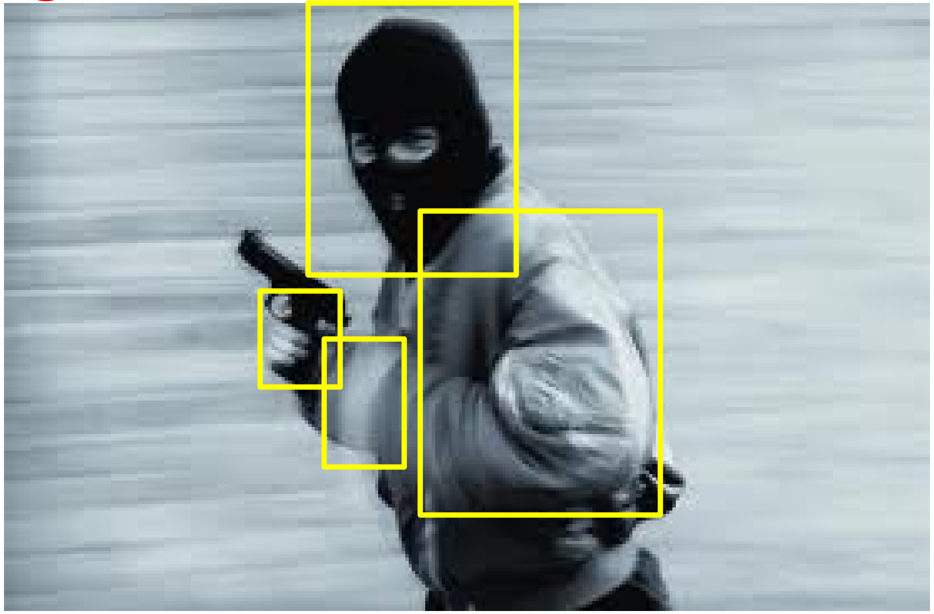
All just predicates
that break problem down

# Common Approach: Breaking it Down

- Person detection:
  - **Detect head**
  - **Detect arms**
  - **Detect hands**
  - **Detect legs**



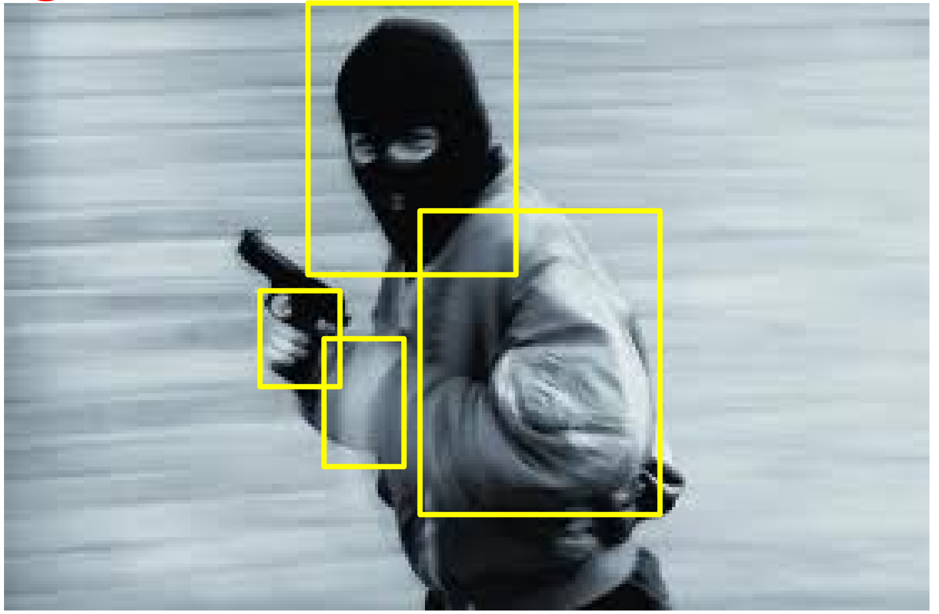
# Common Approach: Breaking it Down



- Person detection:
  - **Detect head**
  - **Detect arms**
  - **Detect hands**
  - **Detect legs**

- How to write detectors?
  - **If we could hard-code, we would**
  - **But heuristics perform poorly**
  - **Machine learning to the rescue**
    - ***Functions defined via data***

# Common Approach: Breaking it Down



- Person detection:
  - **Detect head**
  - **Detect arms**
  - **Detect hands**
  - **Detect legs**
- How to write detectors?
  - **If we could hard-code, we would**
  - **But heuristics perform poorly**
  - **Machine learning to the rescue**
    - ***Functions defined via data***
- A PhD Thesis…
- Download some libraries
  - **Learning algorithms**
  - **Inference algorithms**
  - **Other researchers' detectors**
- Write some feature extractors
- Write some scripts to run everything

# A (Realistic) Knowledge Management Program

```
Corpus c = ReadCollection()
List   LikedPeople = ReadPeople()
List   DisLikedBPeople = ReadPeople()
for (Email e in c):
      for (Person p in body(e)):
            Like = isLike(p)
            if Like and (not in LikedPeople)
                  LikedPeople +v p
            if (not Like) and (not in DisLikedPeople)
                  DisLikedPeople +v p
Update Likedpeople, DisLikedPeople
```

COGNITIVE COMPUTATION GROUP
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

# A (Realistic) Knowledge Management Program

```
Corpus c = ReadCollection()
List   LikedPeople = ReadPeople()
List   DisLikedBPeople = ReadPeople()
for (Email e in c):
      for (Person p in body(e)):
            Like = isLike(p)
            if Like and (not in LikedPeople)
                  LikedPeople +v p
            if (not Like) and (not in DisLikedPeople)
                  DisLikedPeople +v p
Update Likedpeople, DisLikedPeople
```

- Simple! …except we have to
  - **recognize people in text**
  - **determine if they are Liked in the text**
  - **determine if they are new to the list**

# A (Realistic) Knowledge Management Program

```
Corpus c = ReadCollection()
List   LikedPeople = ReadPeople()
List   DisLikedBPeople = ReadPeople()
for (Email e in c):
      for (Person p in body(e)):
            Like = isLike(p)
            if Like and (not in LikedPeople)
                  LikedPeople +v p
            if (not Like) and (not in DisLikedPeople)
                  DisLikedPeople +v p
Update Likedpeople, DisLikedPeople
```

- Simple! …except we have to
  - **recognize people in text**
  - **determine if they are Liked in the text**
  - **determine if they are new to the list**

All hard problems

All just predicates that break problem down

# Discriminative Example: Semantic Role Labeling

[Punyakanok, et.al., CL'08]

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

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

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

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

# Our Learning Based Programming Thesis

- Existing programming languages are not designed to deal with real-world messy data, and to describe the central components of modern learning-based programs:
  - **constrained optimization problems whose objective function are derived from data**

# Our Learning Based Programming Thesis

- Existing programming languages are not designed to deal with real-world messy data, and to describe the central components of modern learning-based programs:
  - **constrained optimization problems whose objective function are derived from data**
- Present the Constrained Conditional Model (CCM), a computation model for learning and inference that is
  - **Expressive enough to capture a large class of problems**
  - **Provides the abstraction for our language**

# Our Learning Based Programming Thesis

- Existing programming languages are not designed to deal with real-world messy data, and to describe the central components of modern learning-based programs:
  - **constrained optimization problems whose objective function are derived from data**
- Present the Constrained Conditional Model (CCM), a computation model for learning and inference that is
  - **Expressive enough to capture a large class of problems**
  - **Provides the abstraction for our language**
- Demonstrate 2 CCM-based LBP languages that compile their efficient implementation from data.
  - **LBJava http://cogcomp.cs.illinois.edu/page/software_view/11**
  - **Our 2nd generation language, CCMP**

# Principles of Learning Based Programming

- An LBP language provides:
  - **High level primitives**
    - **for feature extraction, learning, inference, and their combinations**
  - **Relational features (a.k.a. *structure*)**
    - **Features involving multiple output variables**
  - **Infinite feature space**
    - **Cannot assume a priori how many or which features will be present**
  - **Customizable objective function**
    - **Model can't be a black box**

# Principles of Learning Based Programming (2)

- An LBP language provides:
  - **Model composability**
    - **Encapsulate model in a name; re-use in larger models**
  - **Training & Inference decomposability**
    - **Facilitate tailored inference solutions via access to structure**
    - **In particular, support of reusability of models, pipelines, etc.**
  - **Algorithm independence**

# Roadmap

- Introduction
  - **Desiderata**
- Constrained Conditional Models
  - **A general, discriminative inference framework**
- Learning Based Java
  - **A discriminative modeling language**
- CCMP: Constraint Conditional Model Processing Language
  - **LBP with structure**
  - **Developing flexible programs over models**
    - **Example**
    - **Program structure: all you need is the paper…**

# Constrained Conditional Models [Roth & Yih ‘04, 07; Chang, et.al.,’08,’12]

# Constrained Conditional Models [Roth & Yih ‘04, 07; Chang, et.al.,’08,’12]

- Prediction function: assign values that maximize objective

$$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} z(\mathbf{x}, \mathbf{y})$$

# Constrained Conditional Models [Roth & Yih ‘04, 07; Chang, et.al.,’08,’12]

- Prediction function: assign values that maximize objective

$$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} z(\mathbf{x}, \mathbf{y})$$

- Objective is linear in features and constraints

$$z(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

  - **Both have free reign over input and output variables**

# Constrained Conditional Models [Roth & Yih '04, 07; Chang, et.al.,'08,'12]

- Prediction function: assign values that maximize objective

$$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} z(\mathbf{x}, \mathbf{y})$$

- Objective is linear in features and constraints

$$z(\mathbf{x}, \mathbf{y}) = \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

  - **Both have free reign over input and output variables**
- Accommodates both probabilistic *and* discriminative techniques

# Constrained Conditional Models

$$\text{argmax}_{\text{y}} \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

# Constrained Conditional Models

$$\text{argmax}_{\text{y}} \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

# Constrained Conditional Models

$$\text{argmax}_{\mathrm{y}} \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

Weight Vector for "local" models

# Constrained Conditional Models

$$\text{argmax}_y \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

Weight Vector for "local" models

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

# Constrained Conditional Models

$$\text{argmax}_y \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

(Soft) constraints component

Weight Vector for "local" models

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

# Constrained Conditional Models

Penalty for violating the constraint.

$$\text{argmax}_y \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

(Soft) constraints component

Weight Vector for "local" models

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

How far y is from a "legal" assignment

# Constrained Conditional Models

Penalty for violating the constraint.

$$\text{argmax}_y \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

(Soft) constraints component

Weight Vector for "local" models

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

How far y is from a "legal" assignment

**Inference (Prediction)**

- This is an Integer Linear Program.
- Any discrete MPE problem can be formulated this way.
- Solution: Use ILP packages for an exact solution; Cutting Planes, Dual Decomposition & other search techniques for approximate solutions

# Constrained Conditional Models

Penalty for violating the constraint.

$$\text{argmax}_y \left( \sum_{i=1}^{n_\phi} w_i \phi_i(\mathbf{x}, \mathbf{y}) \right) - \left( \sum_{j=1}^{n_c} \rho_j c_j(\mathbf{x}, \mathbf{y}) \right)$$

(Soft) constraints component

Weight Vector for "local" models

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

How far y is from a "legal" assignment

## Inference (Prediction)

- This is an Integer Linear Program.
- Any discrete MPE problem can be formulated this way.
- Solution: Use ILP packages for an exact solution; Cutting Planes, Dual Decomposition & other search techniques for approximate solutions

## Learning

- Training is learning the objective function
- **Decoupling** Left from Right allows incorporating independently trained models, pipelines, etc.
- Model decomposition facilitates more flexible and efficient computation.

# Examples: CCM Formulations

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

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

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

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

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

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

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

**Sequential Prediction**

HMM/CRF based:

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

**Linguistics Constraints**

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

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

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

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

Sentence Compression/Summarization:

Language Model based:

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

Linguistics Constraints

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

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

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

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

Sentence Compression/Summarization:

Language Model based:

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

Linguistics Constraints

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

# Examples: CCM Formulations

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

CCMs can be viewed as a general interface to easily combine declarative domain knowledge with data driven statistical models

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

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

Constrained Conditional Models Allow:

- Learning a simple model (or multiple; or pipelines)
- Make decisions with a more complex model
- Accomplished by directly incorporating constraints to bias/re-rank global decisions composed of simpler models' decisions
- More sophisticated algorithmic approaches exist to bias the output
  [CoDL: Cheng et. al 07,12; PR: Ganchev et. al. 10; DecL, UEM: Samdani et. al 12]

# Discriminative Example: Semantic Role Labeling

[Punyakanok, et.al., CL'08]

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

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

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

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

Cognitive Computation Group
University of Illinois at Urbana-Champaign

# SRL: Discriminative Decomposition

- Identify argument candidates
  - Pruning Heuristics
  - Argument Identifier
    - Binary classification
- Classify argument candidates
  - Multi-class classification
  - Can choose to "trust" output of identifier
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output

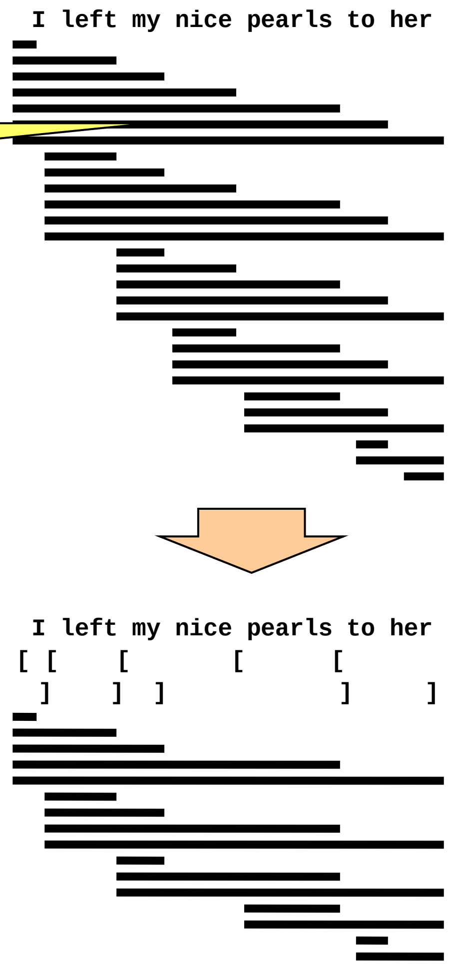
# SRL: Discriminative Decomposition

candidate arguments

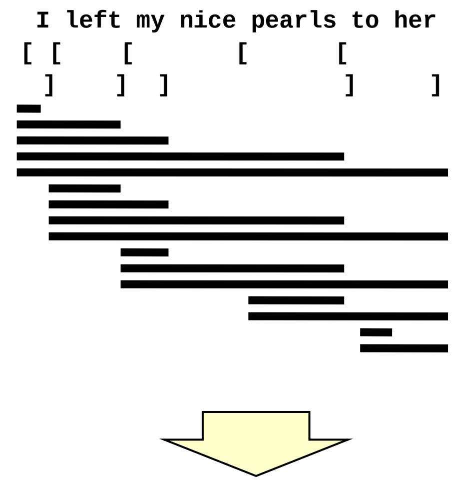
- **Identify** argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**

$$\phi_i^F(\mathbf{x},\mathbf{y}) = \sum_{a=1}^{\mathcal{A}} y_a^F \ x_i^F$$

- **Classify** argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**
- **Inference**
  - **Use the estimated probability distribution given by the argument classifier**
  - **Use structural and linguistic constraints**
  - **Infer the optimal global output**



# SRL: Discriminative Decomposition




- **Identify** argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- **Classify** argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**

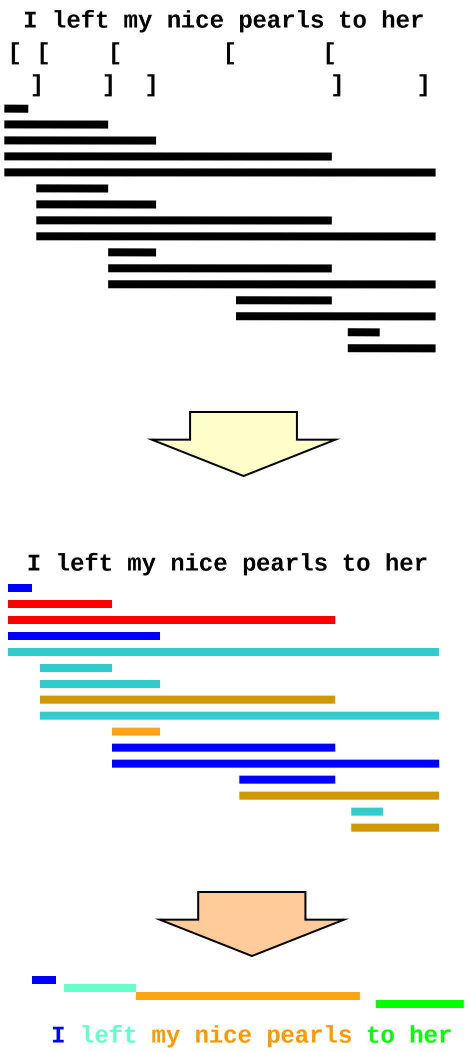
$$\phi^{T}_{t,i}(\mathbf{x},\mathbf{y}) = \sum_{a=1}^{\mathcal{A}} y^{T}_{a,t} x^{T}_{i}$$

$$c^{D}_{a}(\mathbf{x},\mathbf{y}) = I\left\{ \sum_{t}^{\mathcal{T}\cup\{\text{null}\}} y^{T}_{a,t} = 1 \right\}$$


- **Inference**
  - **Use the estimated probability distribution given by the argument classifier**
  - **Use structural and linguistic constraints**
  - **Infer the optimal global output**

# SRL: Discriminative Decomposition

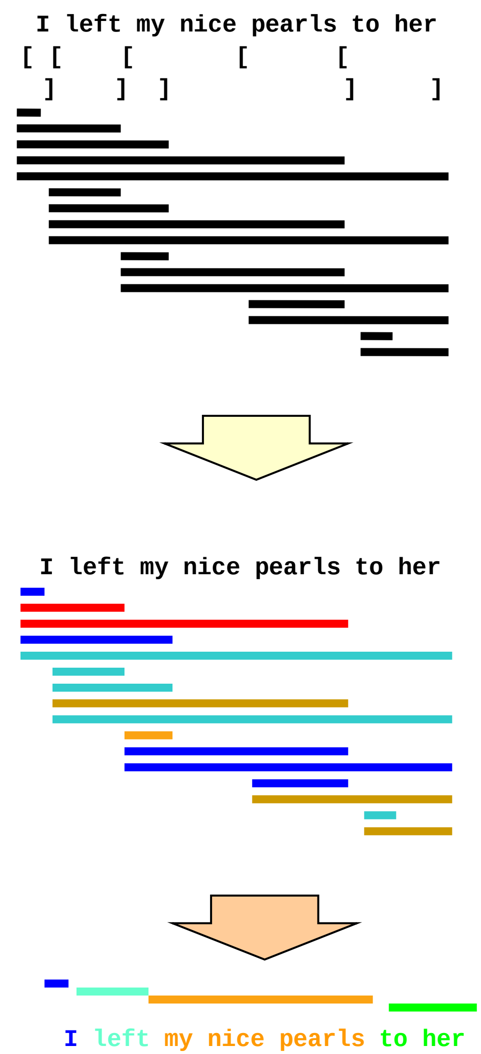
- Identify argument candidates
  - Pruning Heuristics
  - Argument Identifier
    - Binary classification
- Classify argument candidates
  - Multi-class classification
  - Can choose to “trust” output of identifier
- Inference
  - Use the estimated probability distribution given by the argument classifier
  - Use structural and linguistic constraints
  - Infer the optimal global output



# SRL: Discriminative Decomposition

- Identify argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- Classify argument candidates
  - **Multi-class classification**
  - **Can choose to “trust” output of identifier**
- Inference
  - **Use the estimated probability distribution given ... ssifier**
  - **Use st ... tic constraints**
  - **Infer the optimal global output**

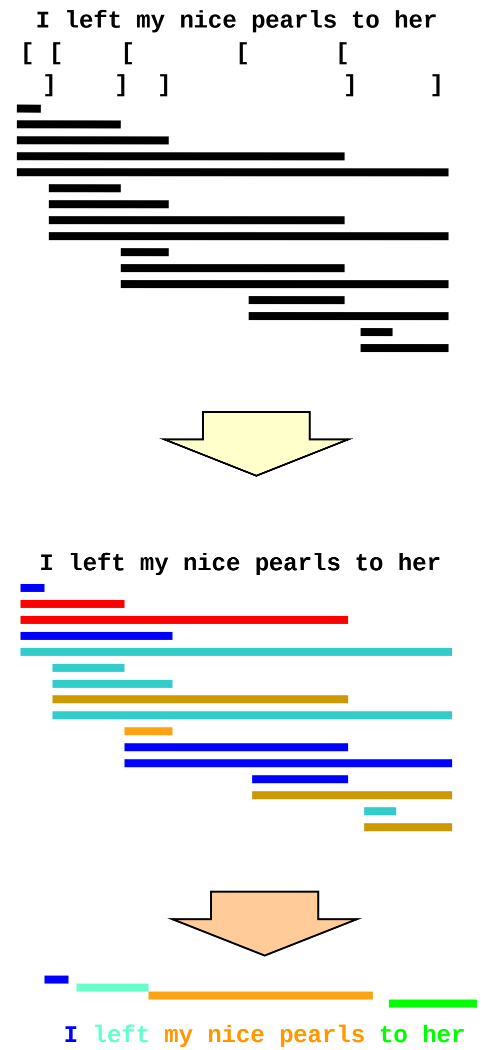
One inference problem for each verb predicate.



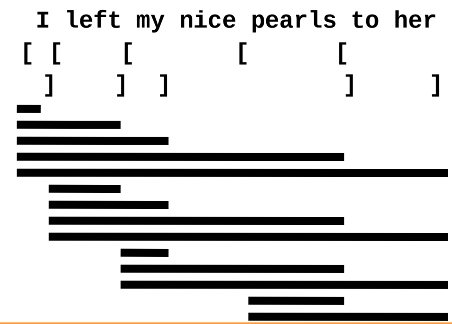
# SRL: Discriminative Decomposition

- Identify argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- Classify argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**
- 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,....

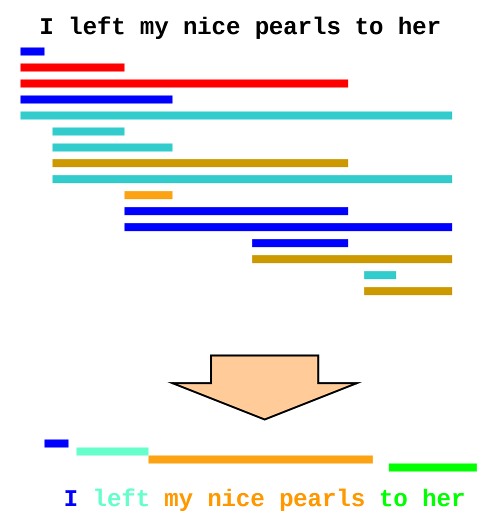


# SRL: Discriminative Decomposition




- Identify argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- Classify argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**

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




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

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

# SRL: Discriminative Decomp

No duplicate argument classes $\forall i, \sum_{y \in \mathcal{Y}} 1_{\{y_i = y\}} = 1$

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

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

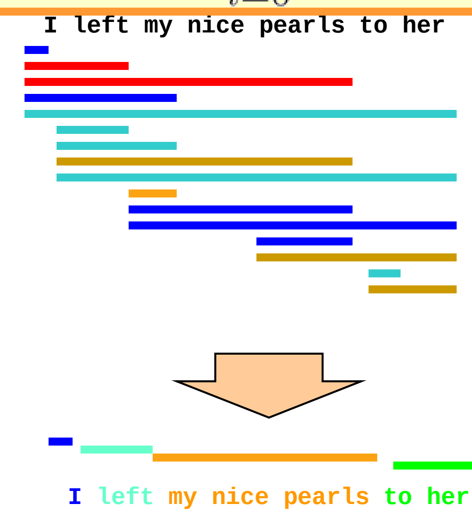
$$\forall j, y \in \mathcal{Y}_C, 1_{\{y_j = y = \text{“C-Ax”}\}} \le \sum_{i=0}^{j} 1_{\{y_i = \text{“Ax”}\}}$$

- Identify argument candid
  - Pruning Heuristics
  - Argument Identifier
    - Binary classification
- Classify argument candida
  - Multi-class classification
  - Can choose to “trust” output of identifier

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

Subject to:

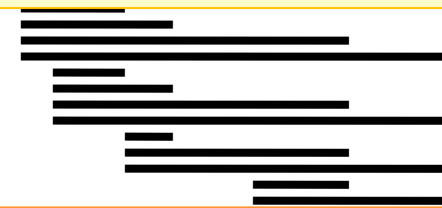
- One label per argument: $\sum_t y^{a,t} = 1$
- No overlapping or embedding
- Relations between verbs and arguments,....



# SRL: Discriminative Decomposition

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

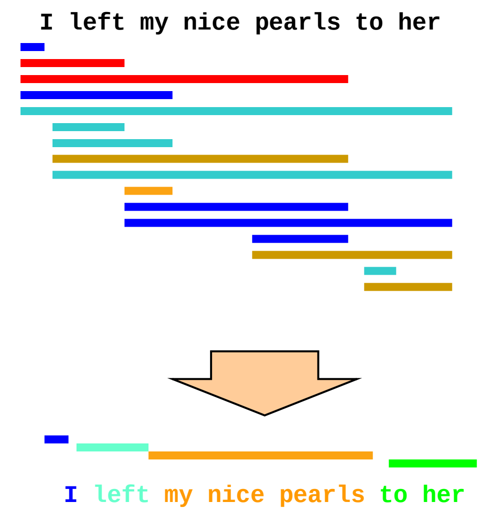
- Identify argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- Classify argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**



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

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

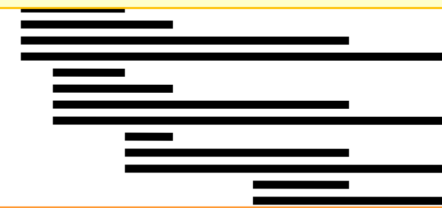
  Subject to:
  - One label per argument: $\sum_t y^{a,t} = 1$
  - No overlapping or embedding
  - Relations between verbs and arguments,….



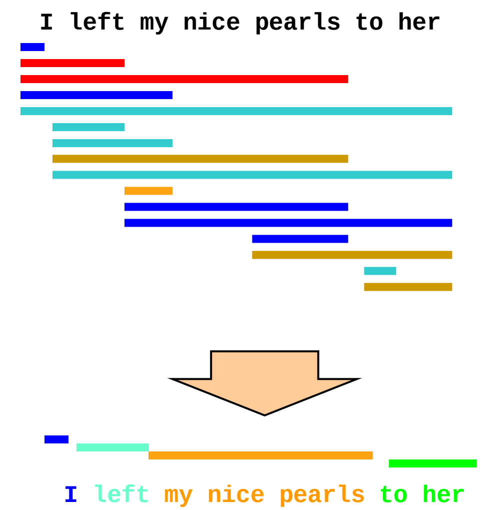
# SRL: Discriminative Decomposition

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

- Identify argument candidates
  - **Pruning Heuristics**
  - **Argument Identifier**
    - **Binary classification**
- Classify argument candidates
  - **Multi-class classification**
  - **Can choose to "trust" output of identifier**



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




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

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

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

# 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
  - **[Chang, Ratinov & Roth, Machine Learning Journal 2012]**
- **Summary of work & a bibliography: http://L2R.cs.uiuc.edu/tutorials.html**

# Roadmap

- Introduction
  - **Desiderata**
- Constrained Conditional Models
  - **A general, discriminative inference framework**
- Learning Based Java
  - **A discriminative modeling language**
- CCMP: Constraint Conditional Model Processing Language
  - **LBP with structure**
  - **Developing flexible programs over models**
    - **Example**
    - **Program structure: all you need is the paper…**

# Learning Based Java [Rizzolo & Roth, ICSC'07, LREC'10]

- LBP design principles:
  - **High level primitives**
  - **Relational features**
  - **Infinite feature space**
  - **Customizable objective function**
  - **Model composability**
  - **Inference decomposability** *(not flexible enough)*
  - **Algorithm independence** *(learning; not inference)*

# Learning Based Java [Rizzolo & Roth, ICSC'07, LREC'10]

- LBP design principles:
  - ☑ **High level primitives**
  - ☐ **Relational features**
  - ☑ **Infinite feature space**
  - ☐ **Customizable objective function**
  - ☑ **Model composability**
  - ☑ **Inference decomposability** *(not flexible enough)*
  - ☑ **Algorithm independence** *(learning; not inference)*

# Learning Based Java [Rizzolo & Roth, ICSC'07]

- Describes a particular type of CCM
  - **Collection of (optionally normalized) multi-class CCMs**
  - **User-defined feature functions**

$$\mathbf{y}_k \equiv \{y_t \mid t \in \mathcal{T}_k,\ y_t \in \{0,1\}\}$$

$$z_k(\mathbf{x},\mathbf{y}) = \left(\sum_{t \in \mathcal{T}_k} \sigma_k(\mathbf{w}_t, \Phi_k(\mathbf{x}))y_t\right) - \infty\, c_k^D(\mathbf{x},\mathbf{y})$$

$$\sigma_k(\mathbf{w},\mathbf{x}) = g_k(\mathbf{w} \cdot \mathbf{x})$$

  - **User-defined constraints (only hard constraints)**
- How to represent constraints / perform inference?
  - **First order logic → translate to ILP**
- How to integrate with user's application?

COGNITIVE COMPUTATION GROUP
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

# Example: Semantic Role Labeling

```
discrete{false, true} ArgumentIdentifier(Argument a) <-
learn ArgumentIdentifierLabel
  using CandidateFeatures,
        PredicatePOS && PhraseType, PredicatePOS && HeadWordAndTag,
        PredicatePOS && ParsePosition, VerbNegated && LinearPosition,
        VerbNegated && Path, ContainsModal && LinearPosition,
        ContainsModal && Path
  with new SparseAveragedPerceptron(.1, 0, 4)
  from new FilterParser(Constants.chunkTrainingData) 50 rounds
end
```

- Classifiers take user's objects as input; produce features
  - **Can be hard-coded or learned**
  - **Learned classifiers use other classifiers to extract features**
    - **Those can be learned too: model composability**
- LBJava compiler:
  - **Indexes features for fast training / testing**
  - **Generates a Java class for every classifier**

# Constraints

```
constraint References(SRLSentence sentence)
{
  for (int i = 0; i < sentence.verbCount(); ++i)
  {
    ParseTreeWord verb = sentence.getVerb(i);
    LinkedList forVerb = sentence.getCandidates(verb);

    (exists (Argument a in forVerb) ArgumentTypeLearner(a) :: "R-A0")
      => (exists (Argument a in forVerb) ArgumentTypeLearner(a) :: "A0");
    (exists (Argument a in forVerb) ArgumentTypeLearner(a) :: "R-A1")
      => (exists (Argument a in forVerb) ArgumentTypeLearner(a) :: "A1");
```

$$(\exists a \in \mathcal{A},\ y_{a,\text{“R-A0”}}) \Rightarrow (\exists a \in \mathcal{A},\ y_{a,\text{“A0”}})$$

“If there’s a reference to an A0, there must be an A0.”

- Declarative, FOL-style constraints
  - **Learned classifiers appear as functions**
  - **Applied directly over user’s Java objects**
  - **Interspersed with arbitrary Java code**
  - **New quantifiers: `atleast` and `atmost`**

# Inference Problems

```
inference SRLInference head SRLSentence sentence
{
  Argument a      { return a.getConstituent().getSentence(); }
  ParseTreeWord w { return w.getSentence(); }
  subjectto
  {
     @NoOverlaps(sentence) /\ @NoDuplicates(sentence) /\ @VA1CV(sentence)
       /\ @References(sentence) /\ @Continuences(sentence)
       /\ @LegalArguments(sentence);
  }
  with new GLPK()
}

discrete ArgumentType(Argument a) <- SRLInference(ArgumentTypeLearner)
```

- "Head" object represents entire inference problem
- At run-time
  - **Constraints translated to linear inequalities**
  - **ILP inference solves problem**
    - **Used broadly in NLP applications**

# LBJava: Success Stories

- Multiple state-of-the-art Natural Language Processing Tools
  - **Part-of-speech tagger; Named Entity Recognition**
  - **Co-Reference Resolution; Relation and Event Extraction,…**
  - **Recognizing authority in dialogue [Mayfield & Rose, ACL'11]**

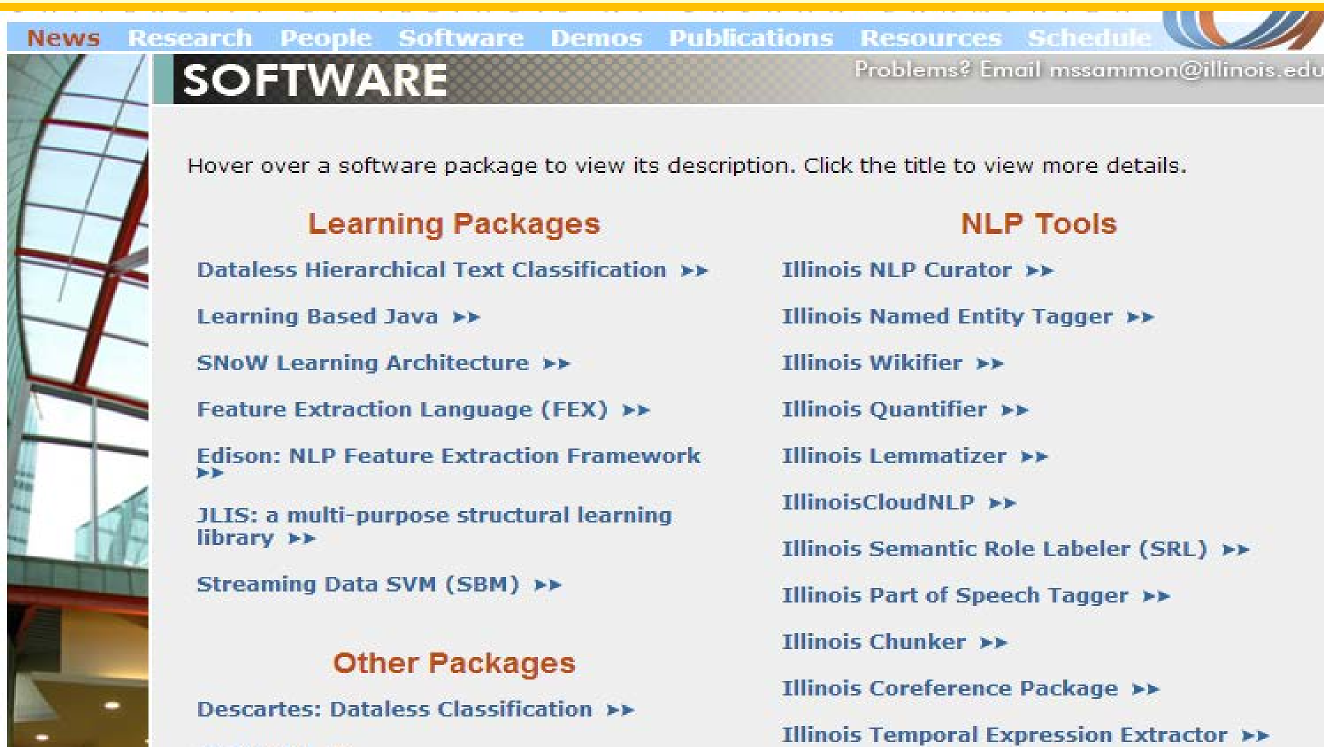
# LBJava: Success Stories

- Multiple state-of-the-art Natural Language Processing Tools
  - **Part-of-speech tagger; Named Entity Recognition**
  - **Co-Reference Resolution; Relation and Event Extraction,…**
  - **Recognizing authority in dialogue [Mayfield & Rose, ACL'11]**

# LBJava: Success Stories

- Multiple state-of-the-art Natural Language Processing Tools
  - **Part-of-speech tagger; Named Entity Recognition**
  - **Co-Reference Resolution; Relation and Event Extraction,…**
  - **Recognizing authority in dialogue [Mayfield & Rose, ACL'11]**

**Developing a state-of-the-art NER takes ~half a day**



# LBJava: Success Stories

- Multiple state-of-the-art Natural Language Processing Tools
  - **Part-of-speech tagger; Named Entity Recognition**
  - **Co-Reference Resolution; Relation and Event Extraction,…**
  - **Recognizing authority in dialogue [Mayfield & Rose, ACL'11]**

**Developing a state-of-the-art NER takes ~half a day**

# Roadmap

- Introduction
  - **Desiderata**
- Constrained Conditional Models
  - **A general, discriminative inference framework**
- Learning Based Java
  - **A discriminative modeling language**
- CCMP: Constraint Conditional Model Processing Language
  - **LBP with structure**
  - **Developing flexible programs over models**
    - **Example**
    - **Program structure: all you need is the paper…**

# 2nd Generation: From LBJava to CCMP

- What's missing?

- Expressivity:
  - **Structures over output variables**

- Ease of use: from paper to program
  - **Declarative definition of models**
    - **Declarative ways to define training and inference preferences**
  - **Procedural building of an application**

# Constrained Conditional Model Processing (CCMP)

- General purpose language; Turing complete
- Fully supports CCMs
- Modular design, decomposed and reusable models
- Flexible and expressive training and inference paradigms
- LBP design principles:
  - **High level primitives**
  - **Relational features**
  - **Infinite feature space**
  - **Customizable objective function**
  - **Model composability**
  - **Inference decomposability**
  - **Algorithm independence**

# CCMP's Unified Formalism

- *Features*, *sparse vectors*, *examples*, and *models* are all primitive data types.
- Provided operators break them down and build them up.
- Models are modular
  - **Previously learned models can be imported, constrained, etc.**
  - **Instances store**
    - **learned parameters**
    - **feature functions**
    - **pointers to other models**
- *Supports a variety of learning and inference protocols*

# From Paper* to Program

- The goal of CCMP is to (almost) automatically generate a program from the application/model described in your paper
- Some code is generated automatically
  - **But can be modified by the programmer**

# From Paper* to Program

Information crucial to the development of an application is often omitted; CCMP abstractions reveal these gaps.

- The goal of CCMP is to (almost) automatically generate a program from the application/model described in your paper
- Some code is generated automatically
  - **But can be modified by the programmer**

# From Paper* to Program

Information crucial to the development of an application is often omitted; CCMP abstractions reveal these gaps.

- The goal of CCMP is to (almost) automatically generate a program from the application/model described in your paper
- Some code is generated automatically
  - **But can be modified by the programmer**
- The program has five components:
  - **Data**
  - **Y Space Definition  (the variables you want to assign values to)**
  - **Representation (features; constraints)**
  - **Prediction (inference) Paradigm**
  - **Training Paradigm**

# From Paper* to Program

Information crucial to the development of an application is often omitted; CCMP abstractions reveal these gaps.

- The goal of CCMP is to (almost) automatically generate a program from the application/model described in your paper
- Some code is generated automatically
  - **But can be modified by the programmer**
- The program has five components:
  - **Data**
  - **Y Space Definition (the variables you want to assign values to)**
  - **Representation (features; constraints)**
  - **Prediction (inference) Paradigm**
  - **Training Paradigm**
- Decoupling decision time prediction and training facilitates reusable models, various decompositions, and pipelines

# Recognizing Entities and Relations [Roth&Yih'04,07]

$\boxed{\text{Dole}}$ 's wife, $\boxed{\text{Elizabeth}}$ , is a native of $\boxed{\text{N.C.}}$

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

Dole 's wife, Elizabeth , is a native of N.C.

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

Dole 's wife, Elizabeth , is a native of N.C.

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

$\boxed{\text{Dole}}$ 's wife, $\boxed{\text{Elizabeth}}$ , is a native of $\boxed{\text{N.C.}}$

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

Dole 's wife, Elizabeth , is a native of N.C.

$E1 \xrightarrow{R_{12}} E2 \xrightarrow{R_{23}} E3$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

$\boxed{\text{Dole}}$ 's wife, $\boxed{\text{Elizabeth}}$ , is a native of $\boxed{\text{N.C.}}$

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

$\boxed{\text{Dole}}$ 's wife, $\boxed{\text{Elizabeth}}$ , is a native of $\boxed{\text{N.C.}}$

$$E1 \xrightarrow{R_{12}} E2 \xrightarrow{R_{23}} E3$$

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

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

# Recognizing Entities and Relations [Roth&Yih'04,07]

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

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

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

Dole 's wife, Elizabeth , is a native of N.C.

$E_1 \xrightarrow{R_{12}} E_2 \xrightarrow{R_{23}} E_3$

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

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

**Is the problem well defined?**
Models for variables could be learned separately; constraints may come up only at decision time.

# Prediction Paradigms

$$\min \sum_{E\in\mathcal{E}} \sum_{e\in\mathcal{L}_\mathcal{E}} c_E(e)\cdot x_{\{E,e\}} + \sum_{R\in\mathcal{R}} \sum_{r\in\mathcal{L}_\mathcal{R}} c_R(r)\cdot x_{\{R,r\}}$$

$$+ \sum_{\substack{E_i,E_j\in\mathcal{E}\\ E_i\neq E_j}} \left[ \sum_{r\in\mathcal{L}_\mathcal{R}} \sum_{e_1\in\mathcal{L}_\mathcal{E}} d^1(r,e_1)\cdot x_{\{R_{ij},r,E_i,e_1\}} + \sum_{r\in\mathcal{L}_\mathcal{R}} \sum_{e_2\in\mathcal{L}_\mathcal{E}} d^2(r,e_2)\cdot x_{\{R_{ij},r,E_j,e_2\}} \right]$$

subject to:

$$\sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{E,e\}} = 1 \qquad \forall E\in\mathcal{E} \tag{2}$$

$$\sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r\}} = 1 \qquad \forall R\in\mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r,E,e\}} \qquad \forall E\in\mathcal{E} \text{ and } \forall R\in\{R: E=\mathcal{N}^1(R) \text{ or } R: E=\mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{R,r,E,e\}} \qquad \forall R\in\mathcal{R} \text{ and } \forall E=\mathcal{N}^1(R) \text{ or } E=\mathcal{N}^2(R) \tag{5}$$

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R},\ E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{8}$$

# Prediction Paradigms

Variable indicating E takes value e.

Variable indicating R takes value r.

$$\min \sum_{E \in \mathcal{E}} \sum_{e \in \mathcal{L}_\mathcal{E}} c_E(e) \cdot x_{\{E,e\}} + \sum_{R \in \mathcal{R}} \sum_{r \in \mathcal{L}_\mathcal{R}} c_R(r) \cdot x_{\{R,r\}}$$

$$+ \sum_{\substack{E_i, E_j \in \mathcal{E} \\ E_i \neq E_j}} \left[ \sum_{r \in \mathcal{L}_\mathcal{R}} \sum_{e_1 \in \mathcal{L}_\mathcal{E}} d^1(r, e_1) \cdot x_{\{R_{ij}, r, E_i, e_1\}} + \sum_{r \in \mathcal{L}_\mathcal{R}} \sum_{e_2 \in \mathcal{L}_\mathcal{E}} d^2(r, e_2) \cdot x_{\{R_{ij}, r, E_j, e_2\}} \right]$$

Variable relating E and R

subject to:

$$\sum_{e \in \mathcal{L}_\mathcal{E}} x_{\{E,e\}} = 1 \qquad \forall E \in \mathcal{E} \tag{2}$$

$$\sum_{r \in \mathcal{L}_\mathcal{R}} x_{\{R,r\}} = 1 \qquad \forall R \in \mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r \in \mathcal{L}_\mathcal{R}} x_{\{R,r,E,e\}} \qquad \forall E \in \mathcal{E} \text{ and } \forall R \in \{R : E = \mathcal{N}^1(R) \text{ or } R : E = \mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e \in \mathcal{L}_\mathcal{E}} x_{\{R,r,E,e\}} \qquad \forall R \in \mathcal{R} \text{ and } \forall E = \mathcal{N}^1(R) \text{ or } E = \mathcal{N}^2(R) \tag{5}$$

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E \in \mathcal{E}, e \in \mathcal{L}_\mathcal{E} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R \in \mathcal{R}, r \in \mathcal{L}_\mathcal{R} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R \in \mathcal{R}, r \in \mathcal{L}_\mathcal{R},\ E \in \mathcal{E}, e \in \mathcal{L}_\mathcal{E} \tag{8}$$

# Prediction Paradigms

Variable indicating E takes value e.

Variable indicating R takes value r.

$$\min \sum_{E\in\mathcal{E}}\sum_{e\in\mathcal{L}_\mathcal{E}} c_E(e)\cdot x_{\{E,e\}} + \sum_{R\in\mathcal{R}}\sum_{r\in\mathcal{L}_\mathcal{R}} c_R(r)\cdot x_{\{R,r\}}$$

Model weight

$$+ \sum_{\substack{E_i,E_j\in\mathcal{E}\\ E_i\neq E_j}} \left[\sum_{r\in\mathcal{L}_\mathcal{R}}\sum_{e_1\in\mathcal{L}_\mathcal{E}} d^1(r,e_1)\cdot x_{\{R_{ij},r,E_i,e_1\}} + \sum_{r\in\mathcal{L}_\mathcal{R}}\sum_{e_2\in\mathcal{L}_\mathcal{E}} d^2(r,e_2)\cdot x_{\{R_{ij},r,E_j,e_2\}}\right]$$

Constraint weight

Variable relating E and R

subject to:

$$\sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{E,e\}} = 1 \qquad \forall E\in\mathcal{E} \tag{2}$$

$$\sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r\}} = 1 \qquad \forall R\in\mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r,E,e\}} \qquad \forall E\in\mathcal{E} \text{ and } \forall R\in\{R : E=\mathcal{N}^1(R) \text{ or } R : E=\mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{R,r,E,e\}} \qquad \forall R\in\mathcal{R} \text{ and } \forall E=\mathcal{N}^1(R) \text{ or } E=\mathcal{N}^2(R) \tag{5}$$

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R},\ E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{8}$$

# Prediction Paradigms

Variable indicating E takes value e.

Variable indicating R takes value r.

$$\min \sum_{E\in\mathcal{E}}\sum_{e\in\mathcal{L}_{\mathcal{E}}} c_E(e)\cdot x_{\{E,e\}} + \sum_{R\in\mathcal{R}}\sum_{r\in\mathcal{L}_{\mathcal{R}}} c_R(r)\cdot x_{\{R,r\}}$$

Model weight

$$+ \sum_{\substack{E_i,E_j\in\mathcal{E}\\ E_i\neq E_j}} \left[\sum_{r\in\mathcal{L}_{\mathcal{R}}}\sum_{e_1\in\mathcal{L}_{\mathcal{E}}} d^1(r,e_1)\cdot x_{\{R_{ij},r,E_i,e_1\}} + \sum_{r\in\mathcal{L}_{\mathcal{R}}}\sum_{e_2\in\mathcal{L}_{\mathcal{E}}} d^2(r,e_2)\cdot x_{\{R_{ij},r,E_j,e_2\}}\right]$$

Constraint weight

Variable relating E and R

subject to:

$$\sum_{e\in\mathcal{L}_{\mathcal{E}}} x_{\{E,e\}} = 1 \qquad \forall E\in\mathcal{E} \tag{2}$$

$$\sum_{r\in\mathcal{L}_{\mathcal{R}}} x_{\{R,r\}} = 1 \qquad \forall R\in\mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r\in\mathcal{L}_{\mathcal{R}}} x_{\{R,r,E,e\}} \qquad \forall E\in\mathcal{E} \text{ and } \forall R\in\{R: E=\mathcal{N}^1(R) \text{ or } R: E=\mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e\in\mathcal{L}_{\mathcal{E}}} x_{\{R,r,E,e\}} \qquad \forall R\in\mathcal{R} \text{ and } \forall E=\mathcal{N}^1(R) \text{ or } E=\mathcal{N}^2(R) \tag{5}$$

Consistency Constraints

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_{\mathcal{R}},\ E\in\mathcal{E}, e\in\mathcal{L}_{\mathcal{E}} \tag{8}$$

# CCMP: Declarative Specification of Problem and Solution

- 1. Data
  - **Readers into CCMP data structures; Edison for NLP**
- 2. Defining the output space (Y)
  - **The variables we need to assign values to**
  - **Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;**
  - **Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }**
- 3. Representation
  - **Features and Constraints**
  - **Most are generated automatically, but can be modified**
- 4. Learning
  - **Defining the decomposition in Training**
- 5. Inference
  - **Decision Time Prediction**

# CCMP: Declarative Specification of Problem and Solution

- 1. Data
  - Readers into CCMP data structures; Edison for NLP
- 2. Defining the output space (Y)
  - The variables we need to assign values to
  - Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
  - Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }
- 3. Representation
  - Features and Constraints
  - Most are generated automatically, but can be modified
- 4. Learning
  - Defining the decomposition in Training
- 5. Inference
  - Decision Time Prediction

**The ability to define Learning and Inference paradigm independently is key in CCMP**

# Learning Paradigms

- There are multiple ways to train models for this problem

# Learning Paradigms

- There are multiple ways to train models for this problem

- {E(Phrase)}; {R(Phrase, Phrase)}
  - **Train independent models for Entities and Relations**

## Learning Paradigms

- There are multiple ways to train models for this problem

- **{**E(Phrase)**}**; **{**R(Phrase, Phrase)**}**
  - **Train independent models for Entities and Relations**

- **{**E(Phrase)**}**; **{**R(Phrase, Phrase, E)**}**
  - **Pipeline E decisions as input to learning R. (Variations possible).**

# Learning Paradigms

- There are multiple ways to train models for this problem

- **{**E(Phrase)**}**; **{**R(Phrase, Phrase)**}**
  - **Train independent models for Entities and Relations**

- **{**E(Phrase)**}**; **{**R(Phrase, Phrase, E)**}**
  - **Pipeline E decisions as input to learning R. (Variations possible).**

- **{**E(Phrase), R(Phrase, Phrase)**}**
  - **Train E and R jointly**

# Learning Paradigms

- There are multiple ways to train models for this problem

- {E(Phrase)}; {R(Phrase, Phrase)}
  - **Train independent models for Entities and Relations**

- {E(Phrase)}; {R(Phrase, Phrase, E)}
  - **Pipeline E decisions as input to learning R. (Variations possible).**

- {E(Phrase), R(Phrase, Phrase)}
  - **Train E and R jointly**

- $\sum_e W_e X_e + \sum_r W_r X_r$
- Subject to
  - $X_{r=LivesIn} \rightarrow X_{e1=PER}$
  - ....

# Inference Paradigms

- There are multiple ways to assign values to target variables

- {E(Phrase)}; {R(Phrase, Phrase)}
  - **Mode decisions with respect to individual variables.**

- {E(Phrase)}; {R(Phrase, Phrase, E)}
  - **Pipeline E decisions as input to inferring R.**

- {E(Phrase), R(Phrase, Phrase)}
  - **Global Inference**

# Inference Paradigms

- There are multiple ways to assign values to target variables

- {E(Phrase)}; {R(Phrase, Phrase)}
  - **Mode decisions with respect to individual variables.**

- {E(Phrase)}; {R(Phrase, Phrase, E)}
  - **Pipeline E decisions as input to inferring R.**

- {E(Phrase), R(Phrase, Phrase)}
  - **Global Inference**

$$\min \sum_{E\in\mathcal{E}} \sum_{e\in\mathcal{L}_\mathcal{E}} c_E(e)\cdot x_{\{E,e\}} + \sum_{R\in\mathcal{R}} \sum_{r\in\mathcal{L}_\mathcal{R}} c_R(r)\cdot x_{\{R,r\}}$$

$$+ \sum_{\substack{E_i,E_j\in\mathcal{E}\\ E_i\neq E_j}} \left[ \sum_{r\in\mathcal{L}_\mathcal{R}} \sum_{e_1\in\mathcal{L}_\mathcal{E}} d^1(r,e_1)\cdot x_{\{R_{ij},r,E_i,e_1\}} + \sum_{r\in\mathcal{L}_\mathcal{R}} \sum_{e_2\in\mathcal{L}_\mathcal{E}} d^2(r,e_2)\cdot x_{\{R_{ij},r,E_j,e_2\}} \right]$$

subject to:

$$\sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{E,e\}} = 1 \qquad \forall E\in\mathcal{E} \tag{2}$$

$$\sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r\}} = 1 \qquad \forall R\in\mathcal{R} \tag{3}$$

$$x_{\{E,e\}} = \sum_{r\in\mathcal{L}_\mathcal{R}} x_{\{R,r,E,e\}} \qquad \forall E\in\mathcal{E} \text{ and } \forall R\in\{R: E=\mathcal{N}^1(R) \text{ or } R: E=\mathcal{N}^2(R)\} \tag{4}$$

$$x_{\{R,r\}} = \sum_{e\in\mathcal{L}_\mathcal{E}} x_{\{R,r,E,e\}} \qquad \forall R\in\mathcal{R} \text{ and } \forall E=\mathcal{N}^1(R) \text{ or } E=\mathcal{N}^2(R) \tag{5}$$

$$x_{\{E,e\}} \in \{0,1\} \qquad \forall E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{6}$$

$$x_{\{R,r\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R} \tag{7}$$

$$x_{\{R,r,E,e\}} \in \{0,1\} \qquad \forall R\in\mathcal{R}, r\in\mathcal{L}_\mathcal{R},\ E\in\mathcal{E}, e\in\mathcal{L}_\mathcal{E} \tag{8}$$

# CCMP: Declarative Specification of a Solution

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05“
Use Reader "ACL-05-Reader"

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05“
Use Reader "ACL-05-Reader"

```
//Here is default input structure
  Corpus(id:Corpus)
  Document(id1:Document,id2:Corpus)
  Paragraph(id1:Paragraph,id2:Document)
  Sentence(id1:Sentence,id2:Paragraph)
  Phrase(id1:Phrase,id2:Word)
```

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05"
Use Reader "ACL-05-Reader"

Output Space:
Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }

# CCMP: Declarative Specification of a Solution

Data:

From Corpus "ACL-05"

Use Reader "ACL-05-Reader"

Output Space:

Y = { Entity(Phrase) $\in$ {PER, LO

Relation(Phrase, Phrase) 

//Here is default output definition

// Entity Aux Variables

$E_i = \{B_i, I_i, L_i, U_i, O\}$

// Phrase Construction Procedure:

Phrase = BIOLU(S)

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05“
Use Reader "ACL-05-Reader"

Output Space:
Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }

Representation:
Use FEX E-R
Use Const: If punc(w) → not E(w)

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05“
Use Reader "ACL-05-Reader"

Output Space:
Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }

Representation:
Use FEX E-R
Use Const: If punc(w) → not

//Here is default FEX
Lexical-form(id:phrase,[0,1])
parse-path(id1:phrase,id2:after(id1))
…..

# CCMP: Declarative Specification of a Solution

Data:
From Corpus "ACL-05“
Use Reader "ACL-05-Reader"

Output Space:
Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }

Representation:
Use FEX E-R
Use Const: If punc(w) → not E(w)

Training:
{E(Phrase)}; {R(Phrase, Phrase)}

# CCMP: Declarative Specification of a Solution

**Data:**
From Corpus "ACL-05"
Use Reader "ACL-05-Reader"

**Output Space:**
Y = { Entity(Phrase) $\in$ {PER, LOC, ORG} ;
Relation(Phrase, Phrase) $\in$ {LivesIn, WorksFor } }

**Representation:**
Use FEX E-R
Use Const: If punc(w) → not E(w)

**Training:**
{E(Phrase)}; {R(Phrase, Phrase)}

**Inference:**
{E(Phrase), R(Phrase, Phrase)}

# CCMP Also Provides Low Level Access

- Access to vector "slices" and individual elements
- Access to FGFs, constraints, and sub-models
  - **Enables learning and inference that was hard or impossible in LBJava**

- Break an FGF down to see operators and sub-formulae
  - **Enables translation to ILP**
  - **Will be useful for other inference algorithms as well**

# CCMP Status: 1st prototype define via Maude & K

- Maude: a language of *rewriting logic* [Meseguer, '92]
  - **Define logical functions and rewrite rules**
  - **Functions represent language syntax; rules give the semantics**
  - **Terms are programs + input; Maude deduces the output**
  - ***Executional semantics***
- K: semantics via *continuations* [Rosu & Serbanuta, '10]
  - **Arrange program state into a *configuration* of *cells***
  - **Arrange computation as stack of continuations**
  - ***Heating/cooling rules* bring next task to top of stack**
- CCMP is defined in 4500 lines of Maude
  - **Multiple applications using a variety of learning and inference paradigms have been coded, trained and tested**

# CCMP Status: 1st prototype define via Maude & K

- Maude: a language of *rewriting logic* [Meseguer, '92]
  - **Define logical functions and rewrite rules**
  - **Functions represent language syntax; rules give the semantics**
  - **Terms are programs + input; Maude deduces the output**
  - ***Executional semantics***
- K: semantics via *continuations* [Rosu & Serbanuta, '10]
  - **Arrange program state into a *configuration* of *cells***
  - **Arrange computation as stack of continuations**
  - ***Heating/cooling rules* bring next task to top of stack**
- CCMP is defined in 4500 lines of Maude
  - **Multiple applications using a variety of learning and inference paradigms have been coded, trained and tested**
- Current version is being implemented in Scala

# Before Conclusion: Cloud NLP

[Wu et. al. LREC'14]

# Before Conclusion: Cloud NLP

[Wu et. al. LREC'14]

**A related – save my time – effort**

# Before Conclusion: Cloud NLP
[Wu et. al. LREC'14]

A related – save my time – effort

- Researcher as well as small/medium-sized organizations sometimes need to analyze large document collections.
- They want to apply a lot of rich Natural Language Processing (NLP) analytics to the document text, but
  - **They don't have expertise developing them**
  - **They may not have peak-time computing power**
  - **They may not have expertise and time to install 3rd party versions.**
- How do you make it really easy to periodically process large sets of documents with rich NLP analytics…
  - **…in a short time**
  - **…at reasonable cost**
  - **…with minimal local compute power?**
- (HINT: IllinoisCloudNLP)

# Before Conclusion: Cloud NLP

[Wu et. al. LREC'14]

A related – save my time – effort

- Researcher as well as small/medium-sized organizations sometimes need to analyze large document collections.
- They want to apply a lot of rich Natural Language Processing (NLP) analytics to the document text, but
  - **They don't have expertise developing them**
  - **They may not have  peak-time computing power**
  - **They may not have expertise and time to install 3rd party versions.**

Illinois CloudNLP  (shortly on: http://cogcomp.cs.illinois.edu/page/software)

- **On-demand processing of large corpora**
- **Using state-of-the art Illinois NLP components**
- **Training and application of text classifiers**
- **Maximum user privacy & user control over data**
- **User runs client software from local machine**
- **Client software applies NLP analytics in the cloud**

# Conclusions

- Learning Based Programming
  - **LBP is the study of programming language abstractions for machine learning representations and techniques.**
  - **A platform for defining and combining decision making models.**
    - **Discriminative or probabilistic; Trained jointly or independently; Exact or approximate inference**
    - **All of these can be left to the programmer to decide**
- An LBP language makes the programmer's life easier
  - **Abstracts away details that distract from the main goal**
  - **Shortens the development cycle**
- Presented the case, and some details of two languages.
  - **LBJava: a mature, easy to use, language that supports learning individual models and joint inference at decision time**
  - **CCMP: an in development declarative language that support the whole LBP development cycle**

Thank You!

# Conclusions

- Learning Based Programming
  - **LBP is the study of programming language abstractions for machine learning representations and techniques.**
  - **A platform for defining and combining decision making models.**
    - **Discriminative or probabilistic; Trained jointly or independently; Exact or approximate inference**
    - **All of these can be left to the programmer to decide**
- An LBP language makes the programmer's life easier
  - **Abstracts away details that distract from the main goal**
  - **Shortens the development cycle**
- Presented the case, and some details of two languages.
  - **LBJava: a mature, easy to use, language that supports learning individual models and joint inference at decision time**
  - **CCMP: an in development declarative language that support the whole LBP development cycle**