Recent Advances in Structured Prediction

Jana Doppa WASHINGTON STATE UNIVERSITY

Liping Liu Tufts UNIVERSITY

Chao Ma Oregon State University

Tutorial at AAAI Conference on Artificial Intelligence (AAAI), 2018

Dedication: Ben Taskar (1977-2013)

- Ben made fundamental contributions to the area of structured prediction
- We dedicate this tutorial to him

Outline of Tutorial

- **Different frameworks for structured prediction [Jana]**
  - Cost function learning framework and recent advances
  - Control knowledge learning framework (greedy and beam search)
  - HC-Search: A Unifying framework
- **Integrating deep learning and structured prediction [Liping]**
  - Deep learning ∩ cost function learning
  - Deep learning ∩ control knowledge learning
- **Multi-task structured prediction [ChaoMa]**
  - Graphical models approach
  - Search based learning and inference architectures

Part 1: Introduction

Introduction

- **Structured Prediction problems are very common**
  - Natural language processing
  - Computer vision
  - Computational biology
  - Planning
  - Social networks
  - ….

Natural Language Processing Examples

NLP Examples: POS Tagging and Parsing

- **POS Tagging**

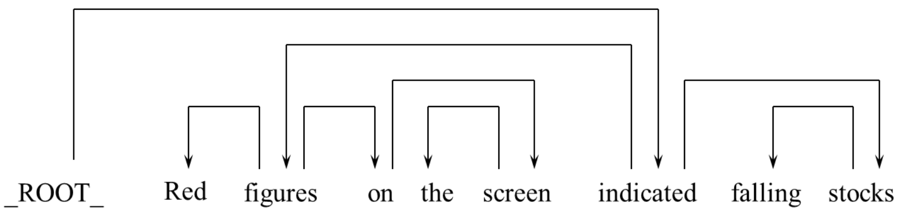
x = "The cat ran" y = *<article> <noun> <verb>*

- **Parsing**

x

"Red figures on the screen indicated falling stocks"

y



NLP Examples: Coreference and Translation

- **Co-reference Resolution**

x

*“**Barack Obama** nominated **Hillary Clinton** as his **secretary of state** on Monday. **He** chose **her** because **she** had foreign affair experience as a former **First Lady**.”*

y

*“**Barack Obama** nominated **Hillary Clinton** as his **secretary of state** on Monday. **He** chose **her** because **she** had foreign affair experience as a former **First Lady**.”*

- **Machine Translation**

x = “The man bit the dog”

y = 该男子咬狗

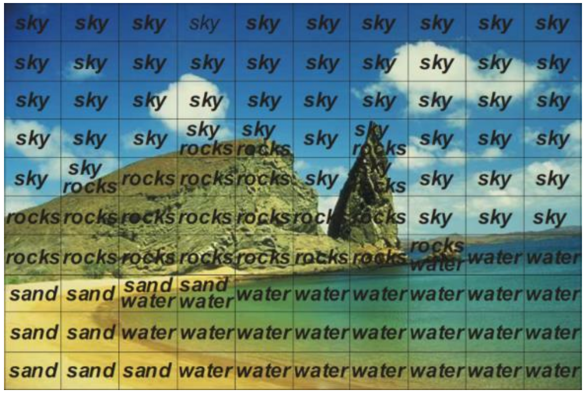
Examples of Bad Prediction

Computer Vision Examples

Scene Labeling

Image

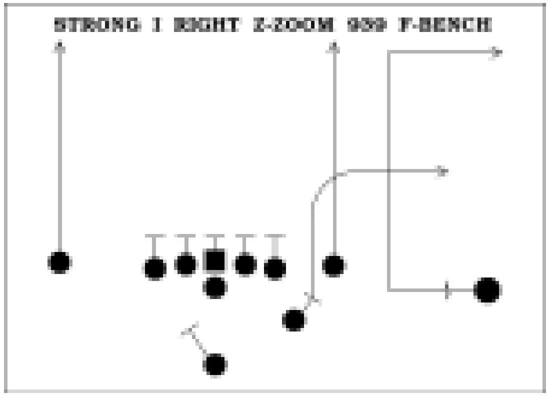
Labeling



The OSU Digital Scout Project

Objective: compute semantic interpretations of football video

Raw video



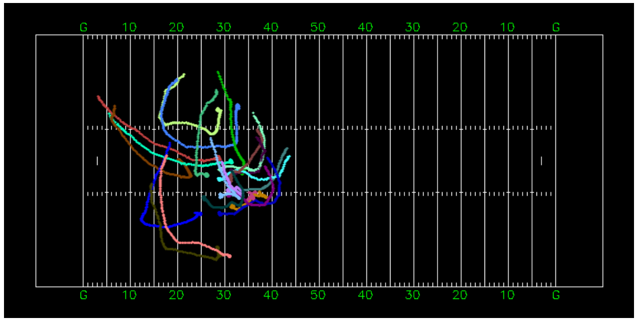
High-level interpretation of play

- Help automate tedious video annotation done by pro/college/HS teams
  - Working with hudl (hudl.com)
- Requires advancing state-of-the-art in computer vision, including:
  - registration, multi-object tracking, event/activity recognition

Multi-Object Tracking in Videos

Video

Player Trajectories



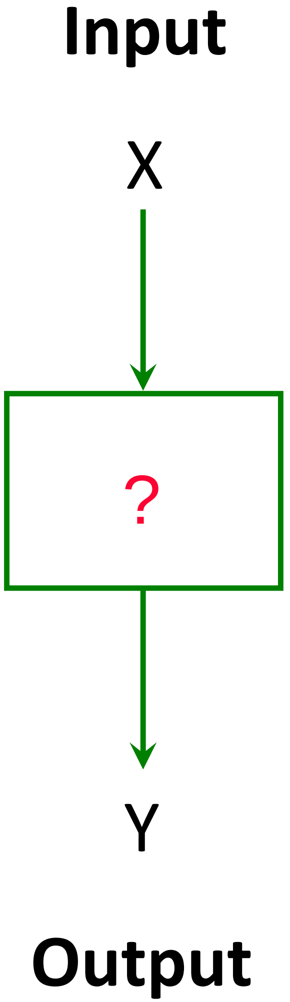
Common Theme

- POS tagging, Parsing, Co-reference resolution, detecting parts of biological objects
  - **Inputs and outputs are highly structured**

- Studied under a sub-field of machine learning called "**Structured Prediction**"
  - Generalization of standard classification
  - Exponential no. of classes (e.g., all POS tag sequences)

Classification to Structured Prediction

Learning a Classifier



Learning a Classifier

?

male

<u>Example problem:</u>

X - image of a face

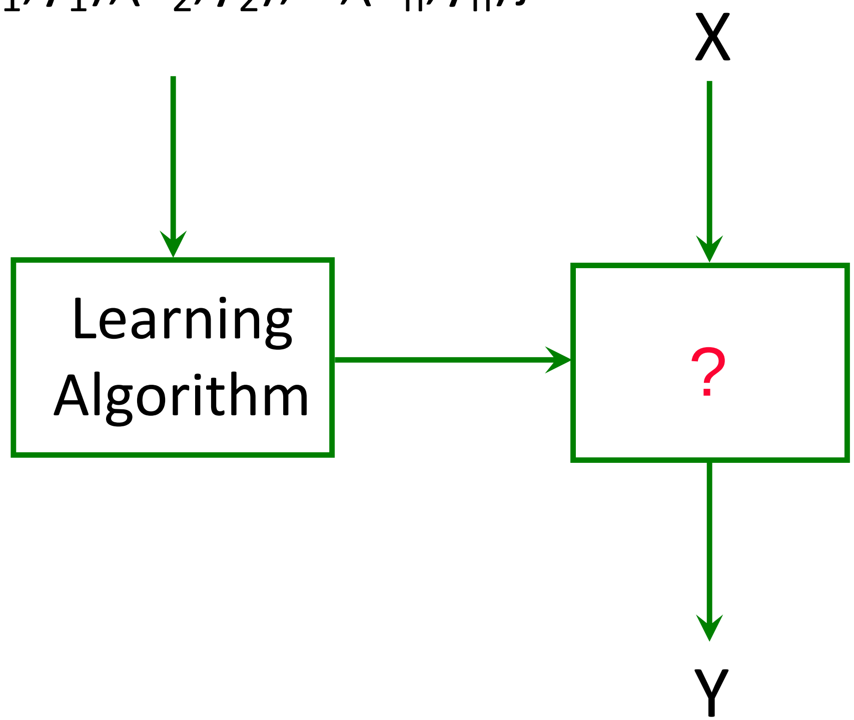
$Y \in$ {male, female}

Learning a Classifier

( , male)

Training Data

$\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$

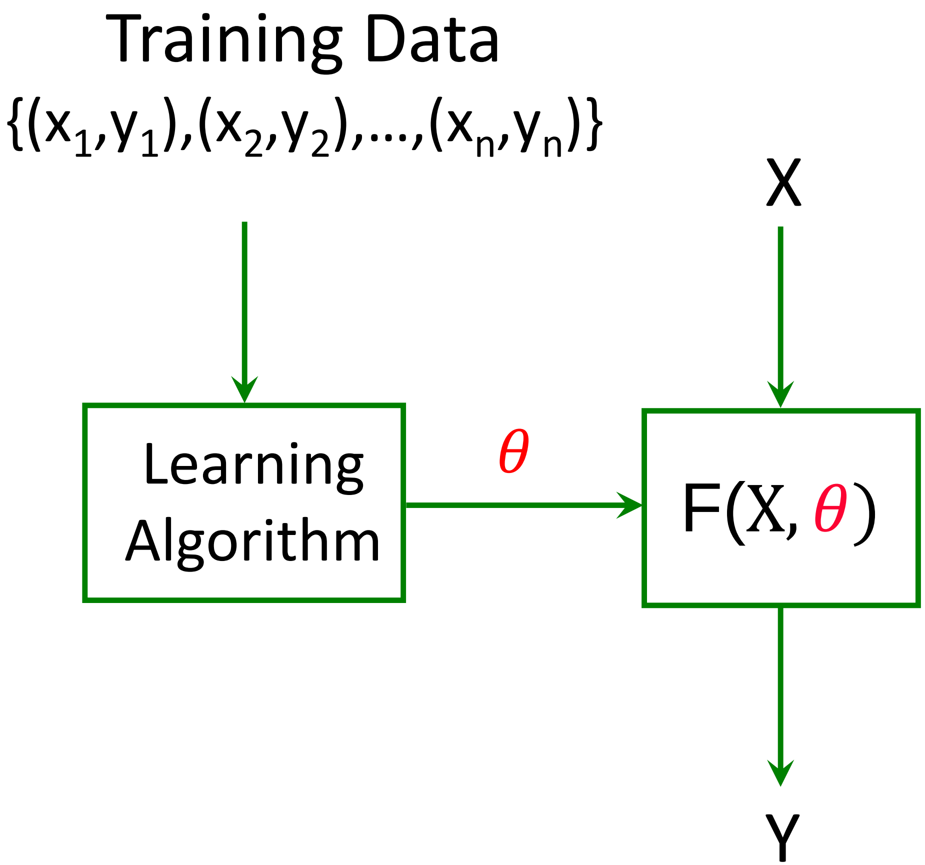


Example problem:

X - image of a face

$Y \in$ {male, female}

Learning a Classifier

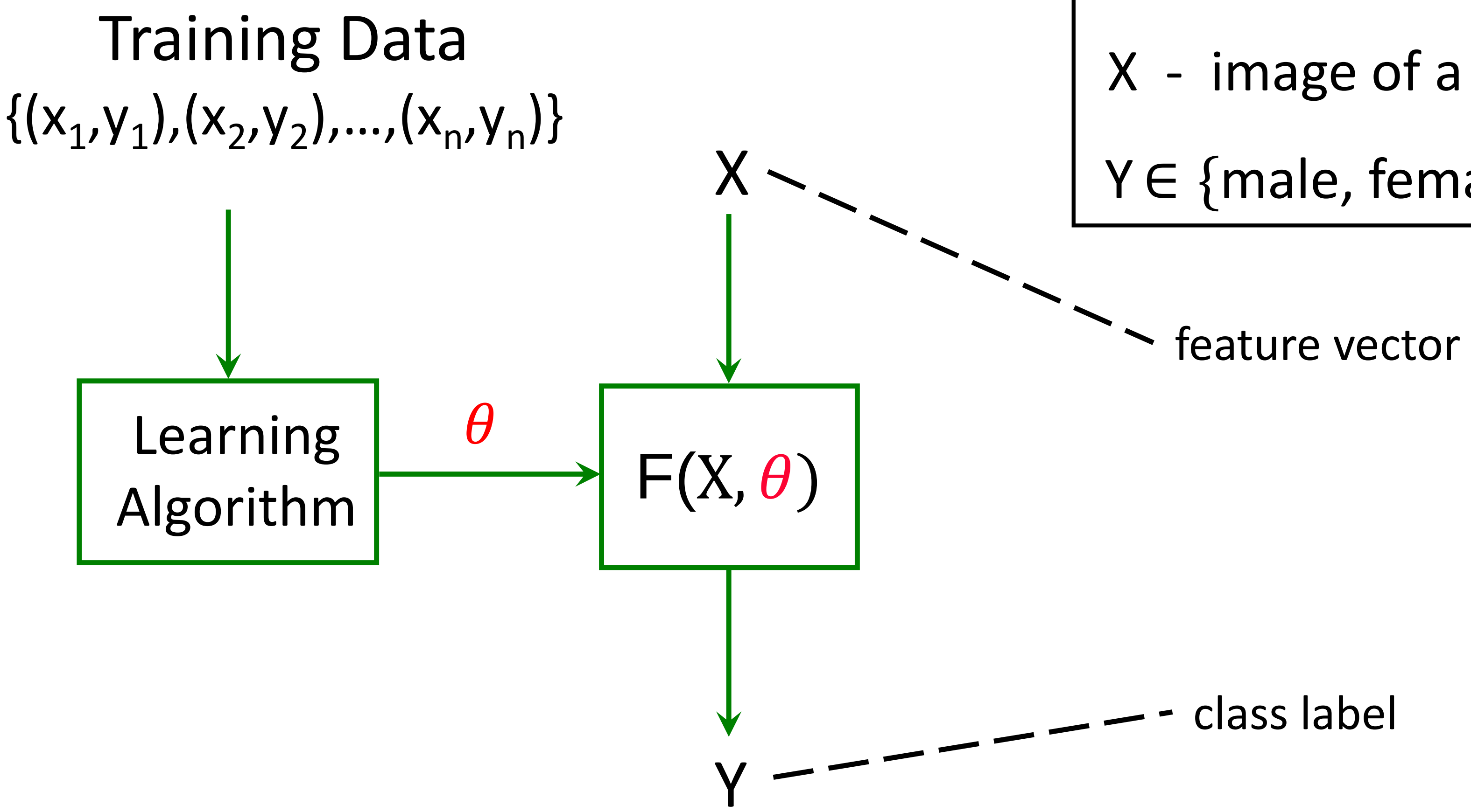


Example problem:

X - image of a face

$Y \in$ {male, female}

Learning for Simple Outputs



<u>Example problem:</u>

X - image of a face

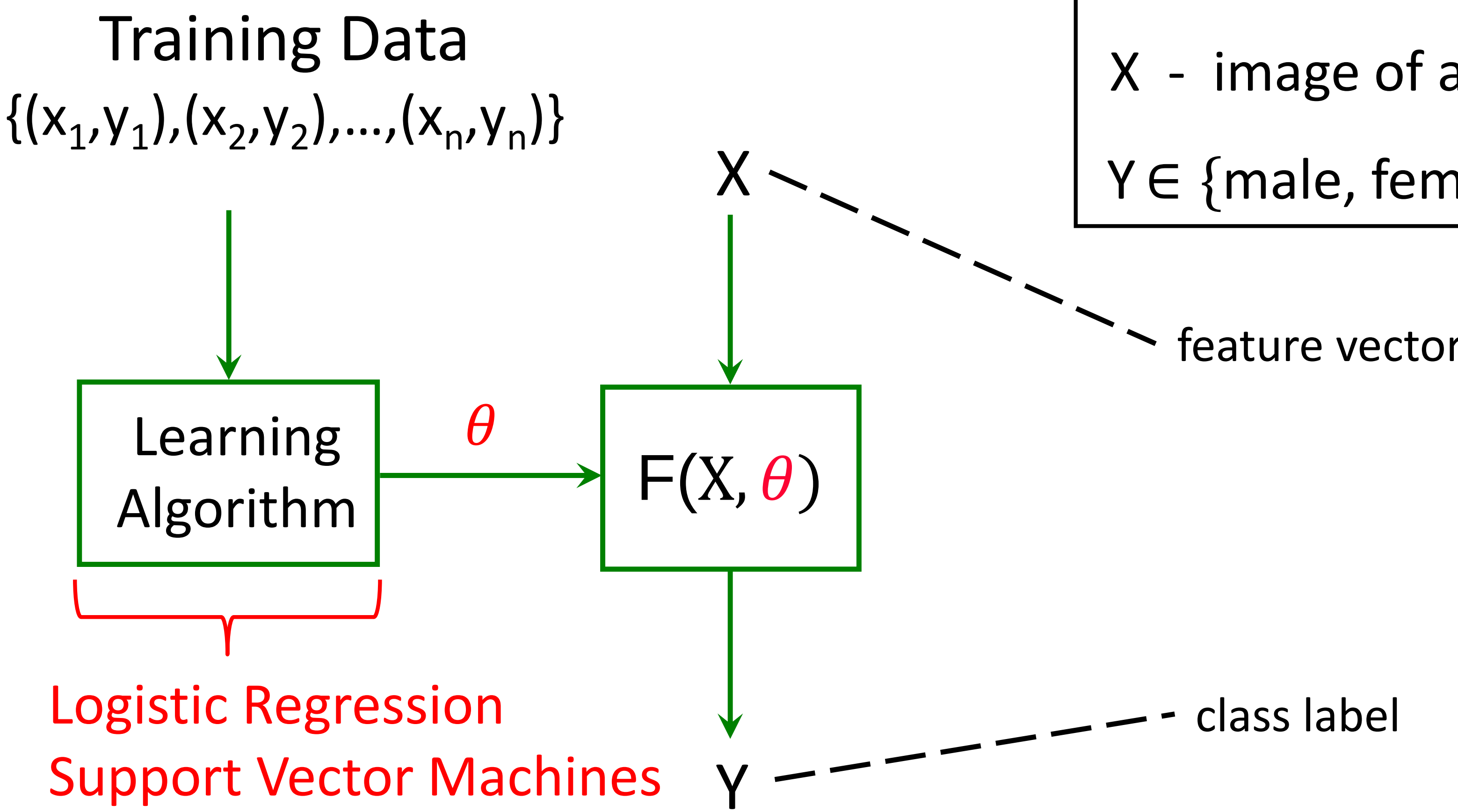
$Y \in$ {male, female}

Learning for Simple Outputs

Example problem:

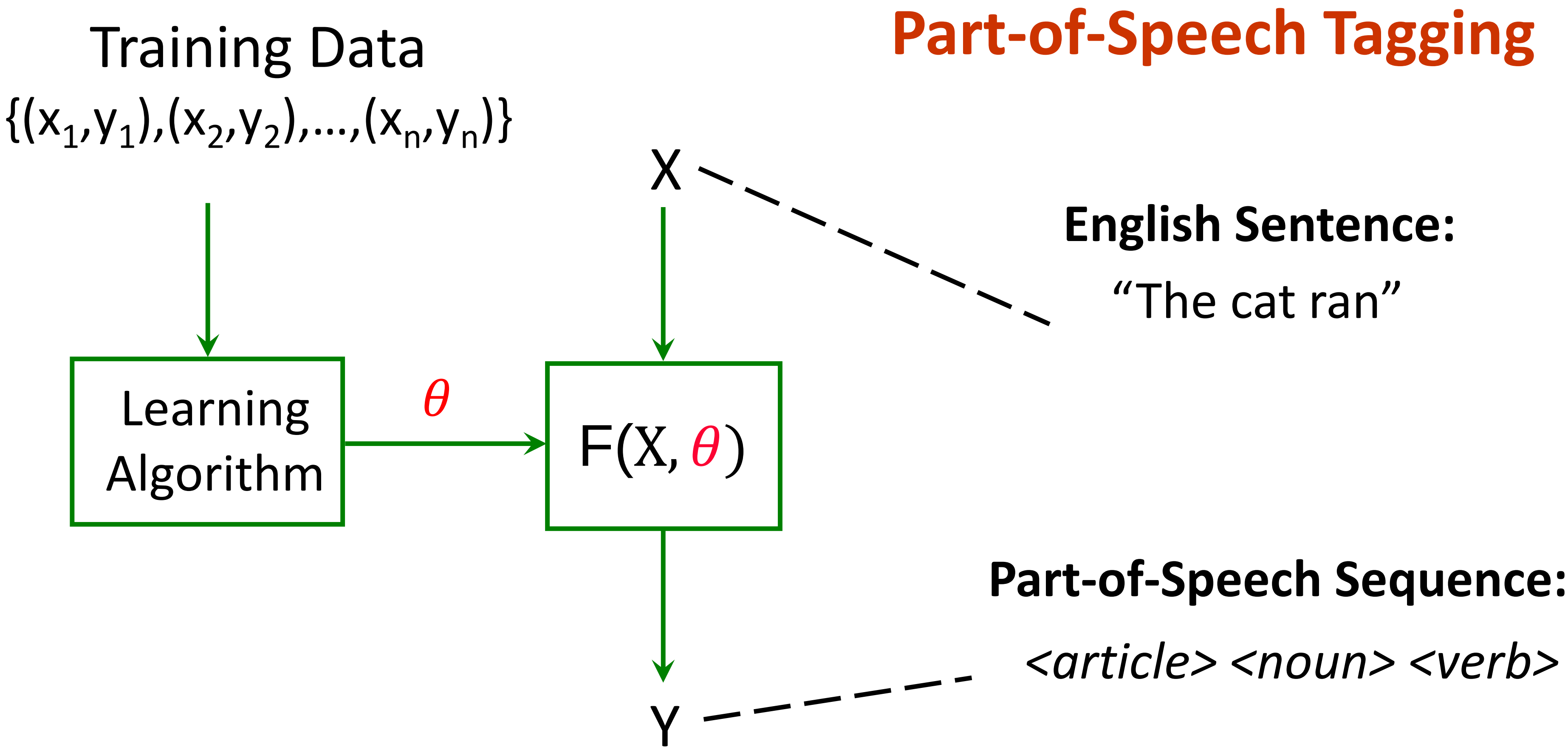
X - image of a face

$Y \in$ {male, female}



- Logistic Regression
- Support Vector Machines
- K Nearest Neighbor
- Decision Trees
- Neural Networks

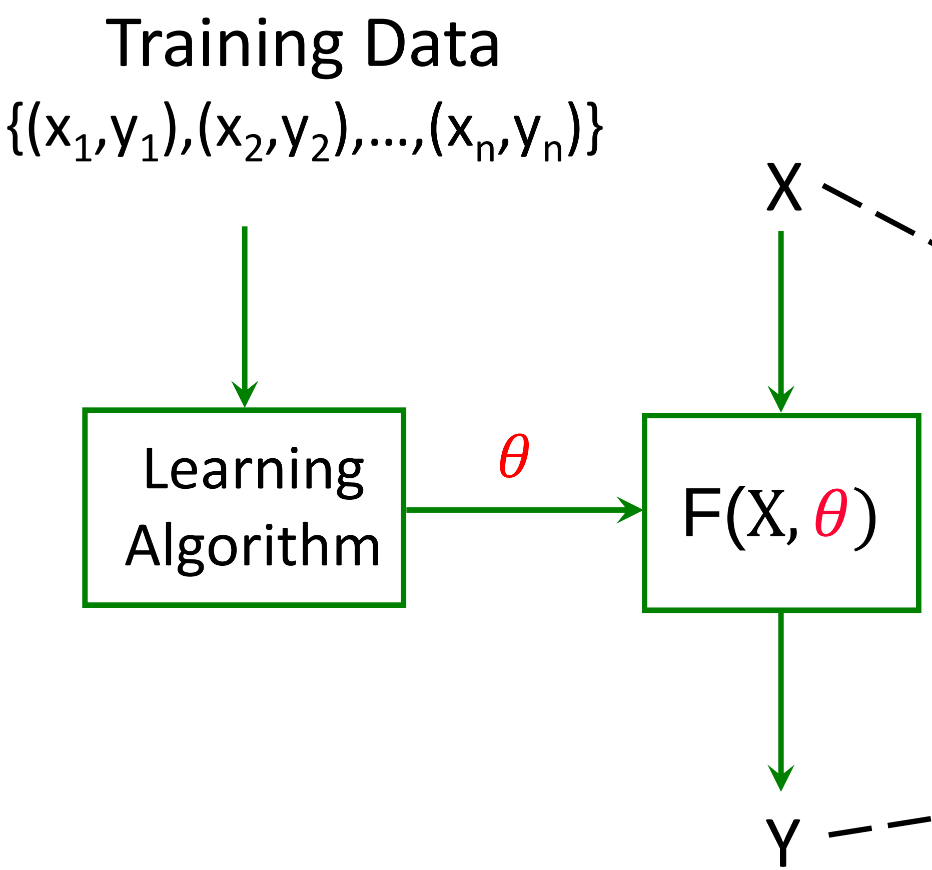
Learning for Structured Outputs



$\boldsymbol{Y} =$ set of all possible POS tag sequences

Exponential !!

Learning for Structured Outputs



$\boldsymbol{Y} =$ set of all possible clusterings

Exponential !!

Co-reference Resolution

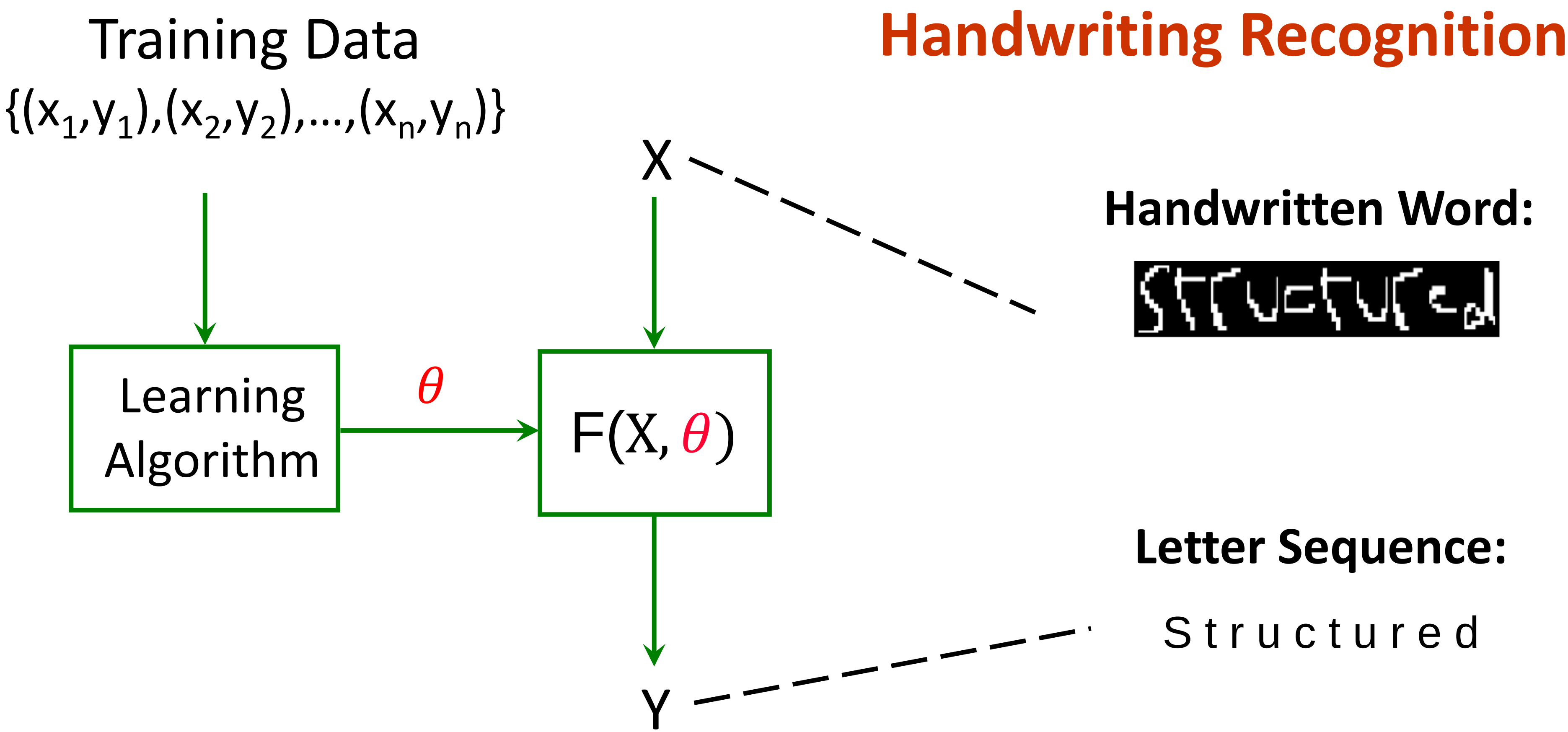
Text with input mentions:

*"**Barack Obama** nominated **Hillary Clinton** as his **secretary of state** on Monday. **He** chose **her** because **she** had foreign affair experience as a former **First Lady**."*

Co-reference Output:

*"**Barack Obama** nominated **Hillary Clinton** as his **secretary of state** on Monday. **He** chose **her** because **she** had foreign affair experience as a former **First Lady**."*

Learning for Structured Outputs



$\boldsymbol{Y} =$ set of all possible letter sequences

Exponential !!

Part 2: Cost Function Learning Framework and Argmin Inference Challenge

Cost Function Learning Approaches: Inspiration

- Generalization of traditional ML approaches to structured outputs
  - SVMs $\Rightarrow$ Structured SVM [Tsochantaridis et al., 2004]
  - Logistic Regression $\Rightarrow$ Conditional Random Fields [Lafferty et al., 2001]
  - Perceptron $\Rightarrow$ Structured Perceptron [Collins 2002]

Cost Function Learning: Approaches

- Most algorithms learn parameters of linear models
  - $\phi(x, y)$ is n-dim feature vector over input-output pairs
  - w is n-dim parameter vector

$$\mathbf{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

Cost Function Learning: Approaches

- Most algorithms learn parameters of linear models
  - $\phi(x, y)$ is n-dim feature vector over input-output pairs
  - w is n-dim parameter vector

$$\mathbf{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

<u>Example</u>: Part-of-Speech Tagging

x = “The cat ran” *y = <article> <noun> <verb>*

$\phi(x, y)$ may have unary and pairwise features

unary feature: e.g. # of times ‘the’ is paired with <article>

pairwise feature: e.g. # of times <article> followed by <verb>

Key challenge: “Argmin” Inference

$$F(x) = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

Exponential size of output space !!

Key challenge: “Argmin” Inference

$$\mathrm{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$$

- Time complexity of inference depends on the dependency structure of features $\phi(x, y)$

Key challenge: “Argmin” Inference

$$F(x) = \arg \min_{y \in Y} w \cdot \phi(x, y)$$

- Time complexity of inference depends on the dependency structure of features $\phi(x, y)$
  - NP-Hard in general
  - Efficient ``exact’’ inference algorithms exist only for simple features
  - Approximate inference techniques are employed in practice and they work reasonably well

Cost Function Learning: Key Elements

- **Joint Feature Function**
  - How to encode a structured input (x) and structured output (y) as a fixed set of features $\phi(x, y)$?
- **(Loss Augmented) Argmin Inference Solver**
  - $\mathbf{F(x)} = \arg\min_{y \in Y} w \cdot \phi(x, y)$
  - Viterbi algorithm for sequence labeling
  - CKY algorithm for parsing
  - (Loopy) Belief propagation for Markov Random Fields
  - Sorting for ranking
- **Optimization algorithm for learning weights**
  - (sub) gradient descent, cutting plane algorithm ...

Cost Function Learning: Generic Template

- **Training goal:**
  - Find weights w s.t
  - For each input x, the cost of the correct structured output y is lower than all wrong structured outputs

Exponential size of output space !!

- **repeat**
  - For every training example (x, y)
  - **Inference:** $\hat{y} = \arg min_{y \in Y} \; w \cdot \varphi(x, y)$
  - If mistake $y \neq \hat{y}$,

    Learning: online or batch weight update
- **until** *convergence* or *max. iterations*

Expensive Training Process

- **Main Reason**
  - repeated calls to “Argmin inference solver” (computationally expensive) on all the training examples

- **Recent Solutions**
  - **Amortized Inference:** Kai-Wei Chang, Shyam Upadhyay, Gourab Kundu, Dan Roth: *Structural Learning with Amortized Inference*. AAAI 2015
  - **Decomposed Learning:** Rajhans Samdani, Dan Roth: *Efficient Decomposed Learning for Structured Prediction*. ICML 2012

Amortized Inference and Speedup Learning

- We need to solve many inference problems during both training and testing
  - Computationally expensive!

- Can we improve the speed of solving new problems based on past problem-solving experience?
  - Yes, amortized Inference!
  - Highly related to ``speedup learning'' **[Fern, 2010]**

Amortized Inference with ILP Formulation

- Inference can be formulated as ILP **[Roth and Yih, 2004]**
- Imagine that you already solved many inference problems
  - Your algorithmic solution method doesn't matter
- How can we exploit this fact to save inference cost?
  - After solving n inference problems, can we make the (n+1)th one faster?
- Conditions under which the solution of a new problem Q is the same as the one of P (which we already cached)

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

The Theorem Must Fire a Lot

- Inference formulation provides a new level of abstraction for amortization
- Modulo renaming
  - Dan gave a talk
  - Vinod ate a pizza
  - Heng read a book
- Have same POS tag structure, Parse Tree, Semantic Parse
- **Pigeon Hole Principle**
  - Many different instances have to be mapped into identical inference outcomes
  - Often, saves 85% of the computation.

Amortized ILP Inference: References

- Vivek Srikumar, Gourab Kundu, Dan Roth: *On Amortizing Inference Cost for Structured Prediction*. **EMNLP** 2012
- Gourab Kundu, Vivek Srikumar, Dan Roth: *Margin-based Decomposed Amortized Inference*. **ACL** 2013
- Kai-Wei Chang, Shyam Upadhyay, Gourab Kundu, Dan Roth: *Structural Learning with Amortized Inference*. **AAAI** 2015
- **PAC Theory for ILP Inference:** The behavior of ILP inference (integrality of relaxed solutions) on training examples generalize to testing examples
  - Ofer Meshi, Mehrdad Mahdavi, Adrian Weller, David Sontag: *Train and Test Tightness of LP Relaxations in Structured Prediction*. **ICML** 2016

Decomposed Learning (DecL)

- **Key Idea:** Inference over a smaller structured output space
  - All structured outputs that have a hamming accuracy of k from the ground truth structured output: DecL(k)

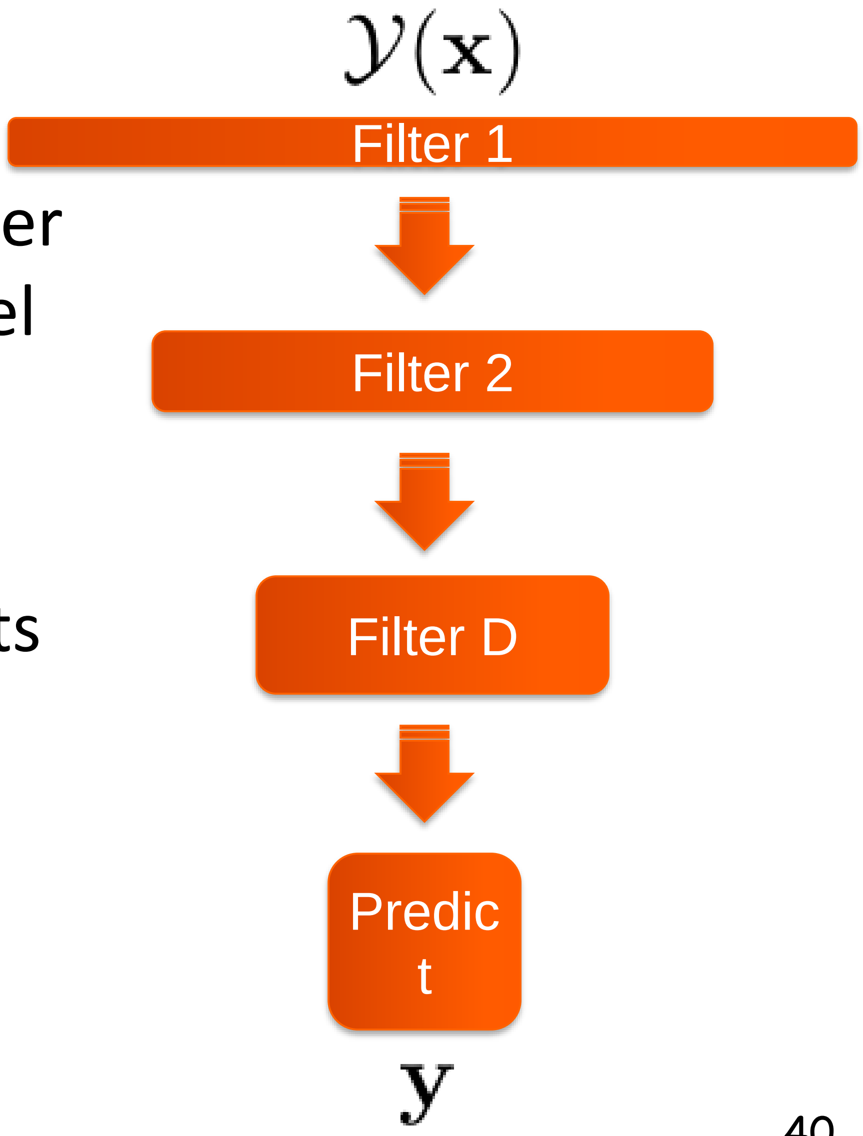
- As k increases, learning approaches standard learning
  - Theoretical guarantees on when DecL will behave similar to standard learning [Samdani and Roth, 2012]

- Special case (k=1):
  - Pseudo-max training [Sontag et al., 2010]

Structured Prediction Cascades [Weiss and Taskar, 2010]

- **Accuracy:** Minimize the number of errors incurred by each level

- **Efficiency**: Maximize the number of filtered assignments at each level



Cost Function Learning: "Exact" vs. "Approximate" Inference Solver

- **Most theory works for "Exact" Inference**
- **Theory breaks with "Approximate" Inference**
  - Alex Kulesza, Fernando Pereira: *Structured Learning with Approximate Inference*. NIPS 2007
  - Thomas Finley, Thorsten Joachims: *Training structural SVMs when exact inference is intractable*. ICML 2008: 304-311
- **Active Research Topic:** Interplay between (approximate) inference and learning
  - Veselin Stoyanov, Alexander Ropson, Jason Eisner: *Empirical Risk Minimization of Graphical Model Parameters Given Approximate Inference, Decoding, and Model Structure*. AISTATS 2011
  - Justin Domke: *Structured Learning via Logistic Regression*. NIPS 2013
  - Tamir Hazan, Alexander G. Schwing, Raquel Urtasun: Blending Learning and Inference in Conditional Random Fields. JMLR-2016
  - ...

Search-based Structured Prediction

- Integrating "Learning" and "Search" two fundamental branches of AI to solve structured prediction problems

- **Key Idea:**
  - **Learning "with Inference" vs. Learning "for Inference"**
  - Select a computationally bounded search architecture for making predictions
  - Optimize the parameters of that procedure to produce accurate outputs using training data

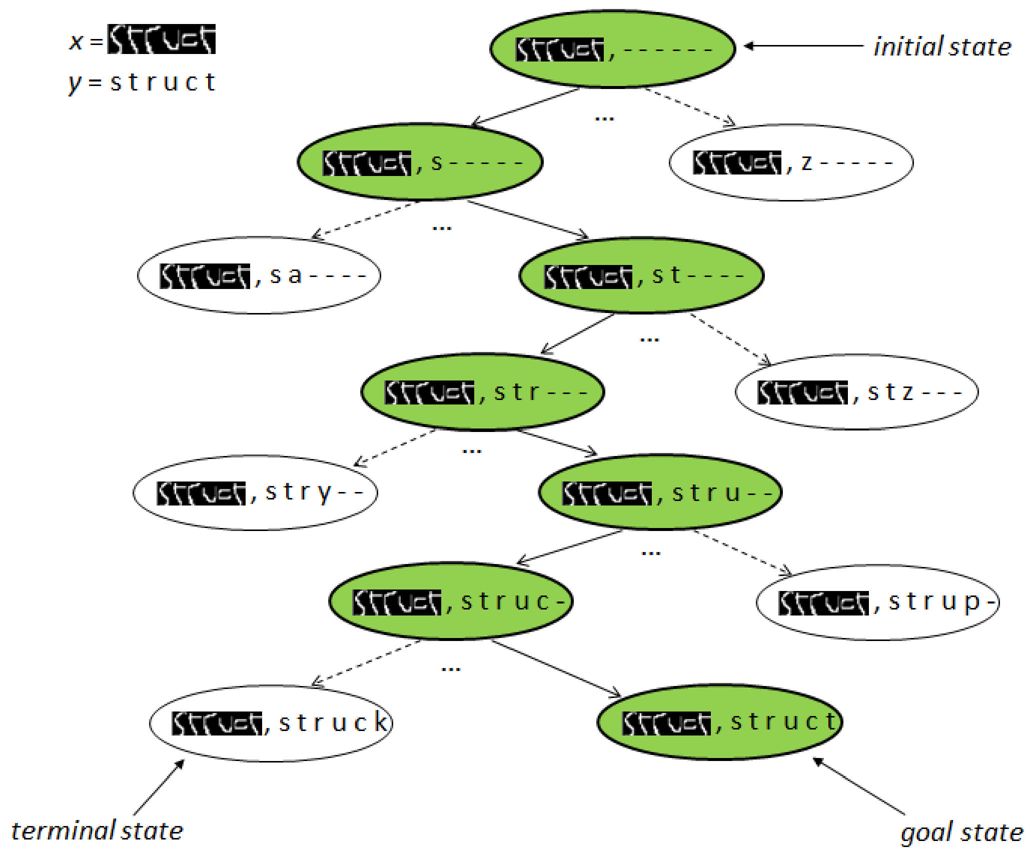
Part 3: Control Knowledge Learning Framework: Greedy Methods

Greedy Control Knowledge Learning

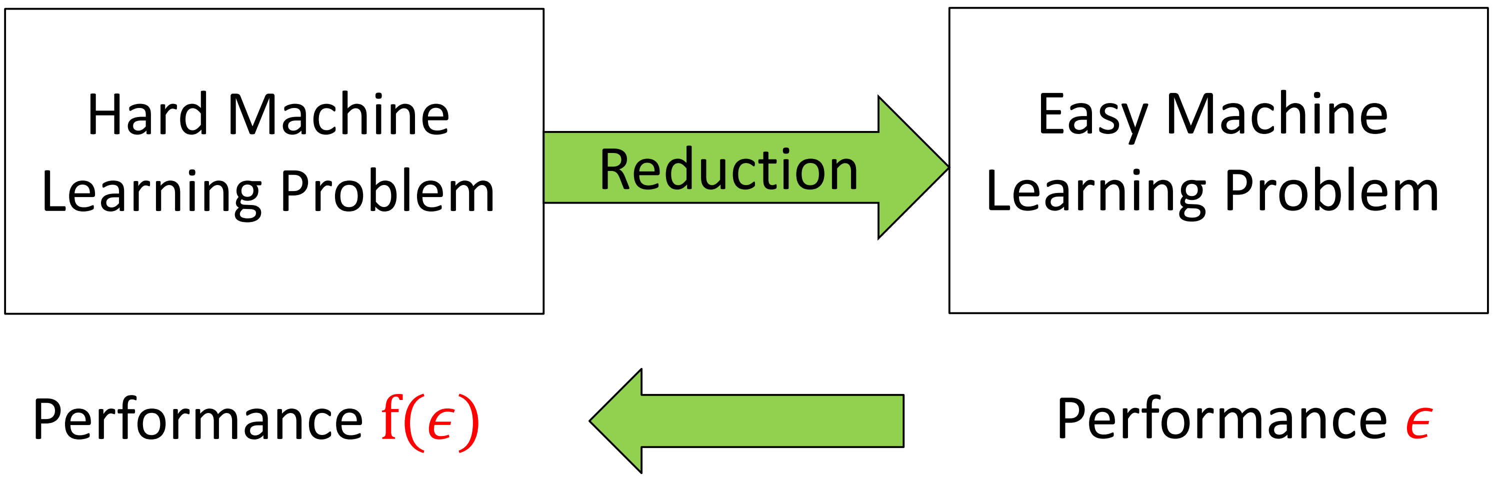
- **Given**
  - Search space definition
  - Training examples (input-output pairs)
- **Learning Goal**
  - Learn a policy or classifier to that directly predicts good structured outputs (no inference needed!)
- **Key Idea:**
  - Training examples can be seen as expert demonstrations
  - Equivalent to "Imitation Learning" or "Learning from Demonstration"
  - Reduction to classifier or rank learning

Classifier-based Structured Prediction

- Reduction to classifier learning
  - 26 classes
- IL Algorithms
  - Exact-Imitation
  - SEARN
  - DAgger
  - AggreVaTe
  - LOLS



Aside: Reductions in Machine Learning



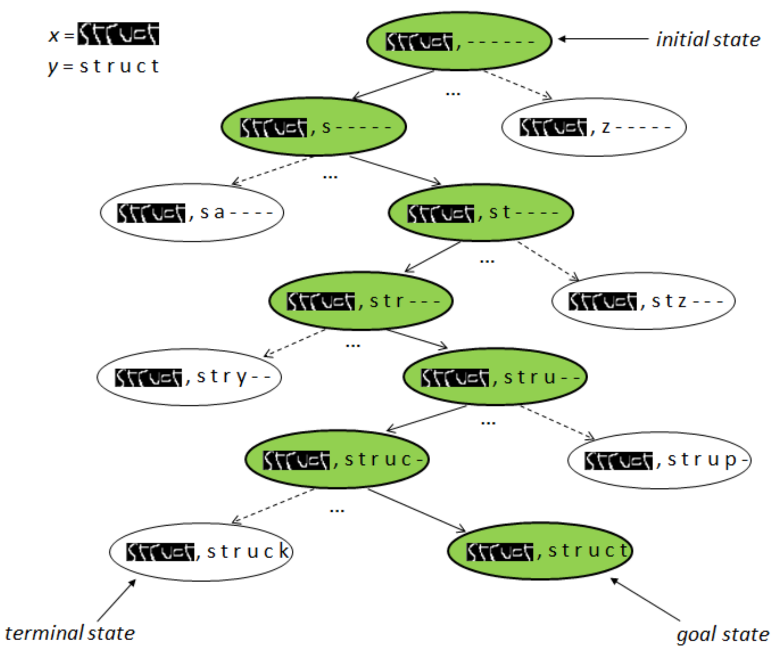
- Reduce complex problem to simpler problem(s)
- A better algorithm for simpler problem means a better algorithm for complex problem
- Composability, modularity, ease-of-implementation

Imitation Learning Approach

- **Expert demonstrations**
  - each training example (input-output pair) can be seen as a "expert" demonstration for sequential decision-making
- **Collect classification examples**
  - Generate a multi-class classification example for each of the decisions
  - Input: f(n), features of the state n
  - Output: y_n, the correct decision at state n
- **Classifier Learning**
  - Learn a classifier from all the classification examples

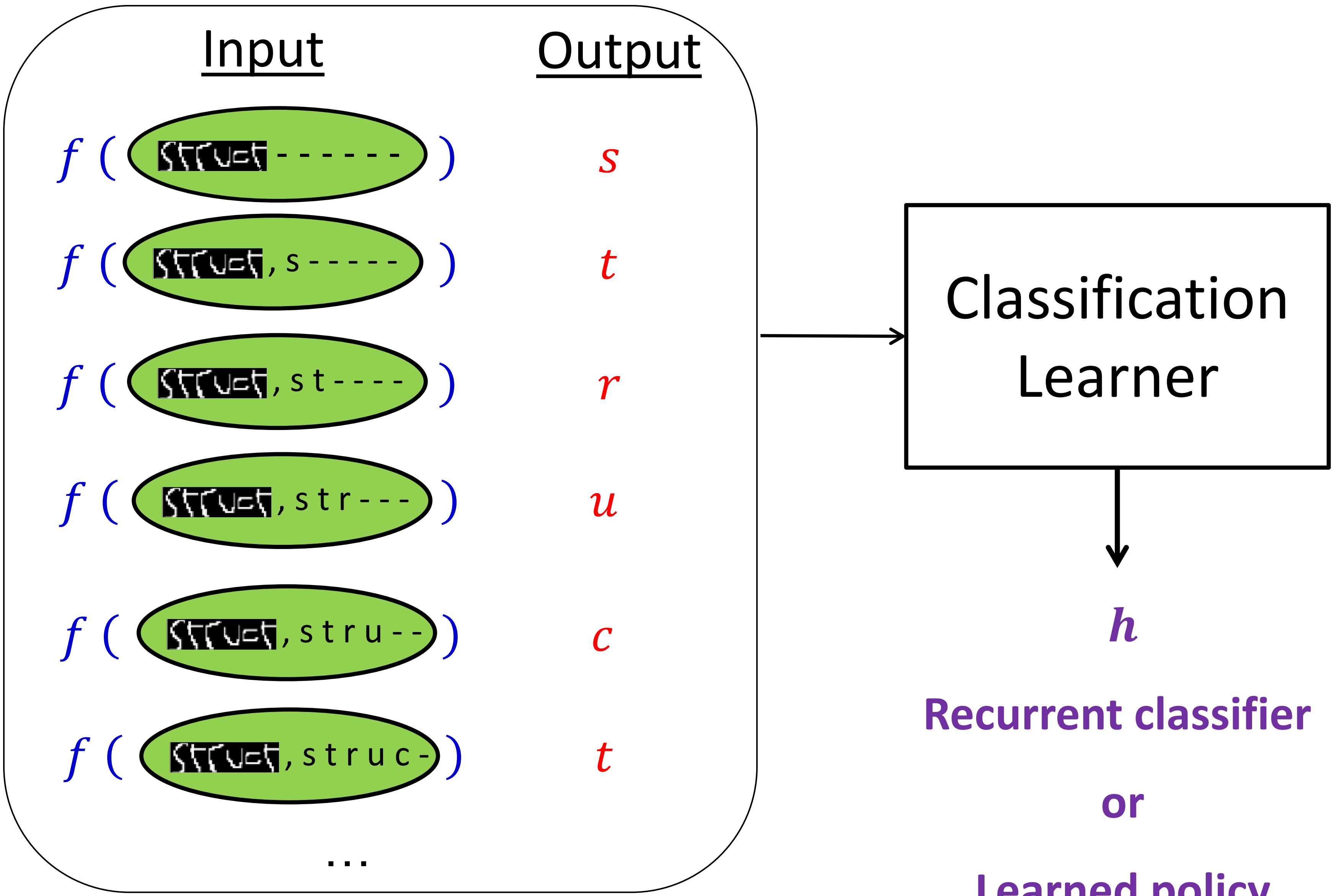
Exact Imitation: Classification examples

- For each training example

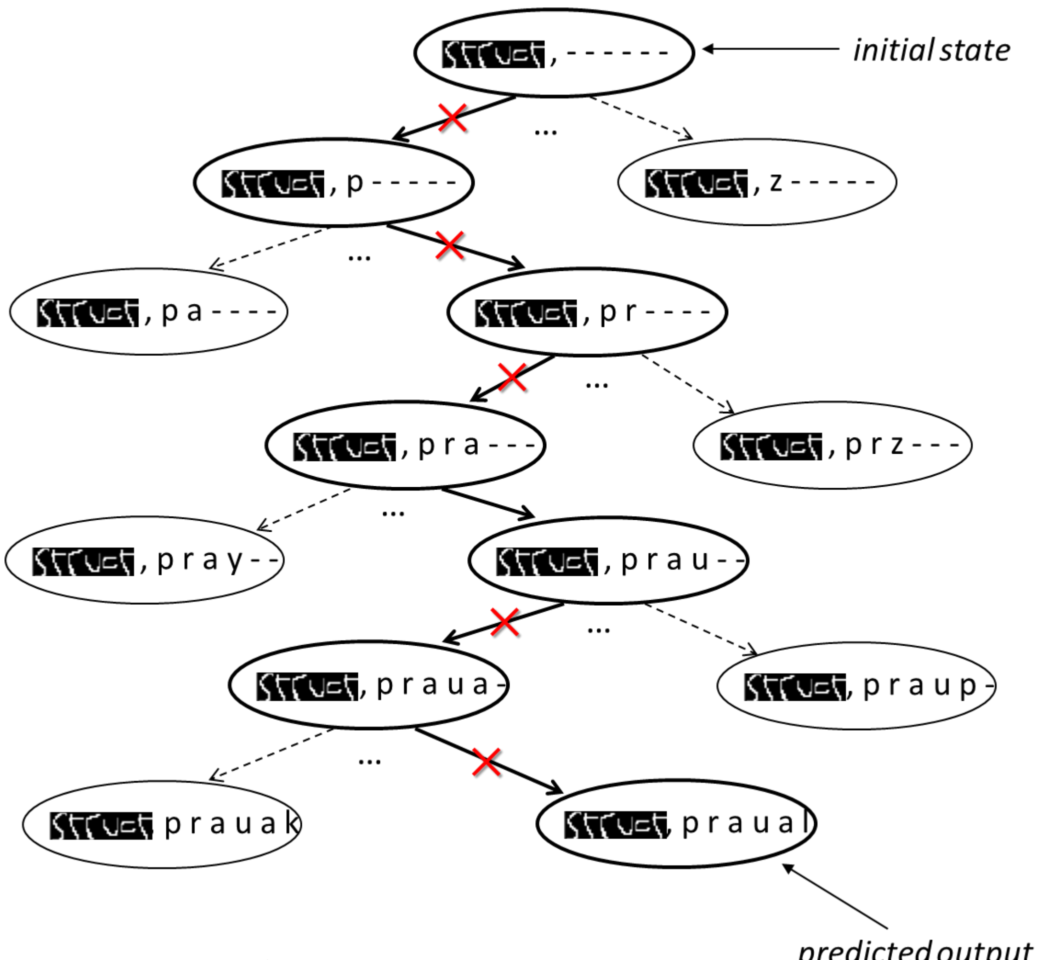


Input	Output
$f($ - - - - - - $)$	s
$f($, s - - - - - $)$	t
$f($, s t - - - - $)$	r
$f($, s t r - - - $)$	u
$f($, s t r u - - $)$	c
$f($, s t r u c - $)$	t

Exact Imitation: Classifier Learning



Learned Recurrent Classifier: Illustration



- **Error propagation:**
  - errors in early decisions propagate to down-stream decisions

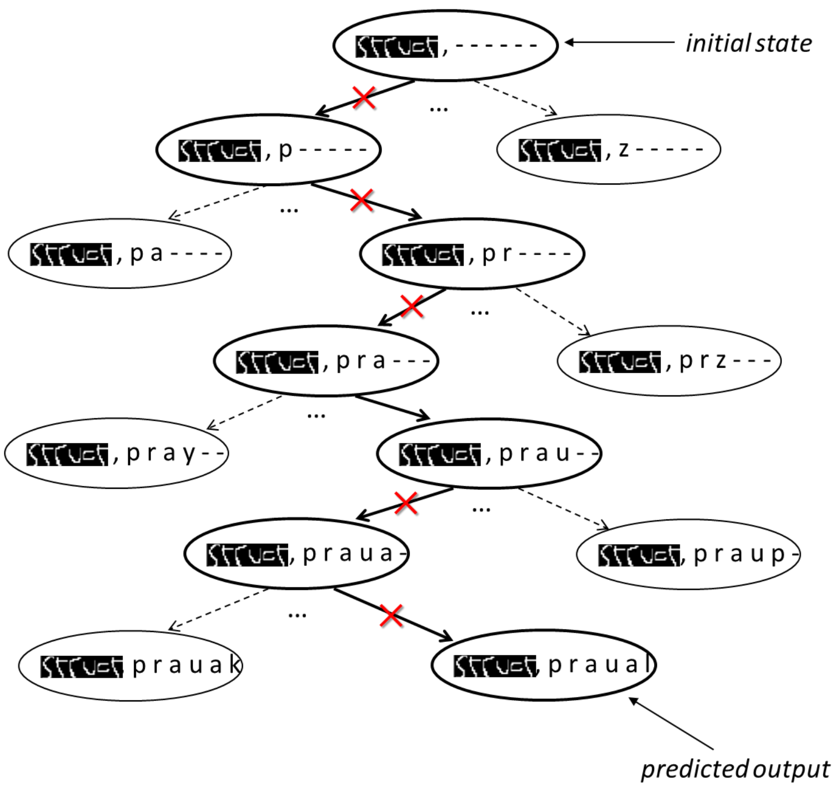
Recurrent Error

- Can lead to poor global performance
- Early mistakes propagate to downstream decisions: $\mathrm{f}(\epsilon) = O(\epsilon T^2)$, where ϵ is the probability of error at each decision and T is the number of decision steps [Kaariainen 2006] [Ross & Bagnell 2010]
- Mismatch between training (IID) and testing (non-IID) distribution
- Is there a way to address error propagation?

Addressing Error Propagation

- **<u>Rough Idea:</u>** Iteratively observe current policy and augment training data to better represent important states

- Several variations on this idea [Fern et al., 2006], [Daume et al., 2009], [Xu & Fern 2010], [Ross & Bagnell 2010], [Ross et al. 2011, 2014], [Chang et al., 2015]



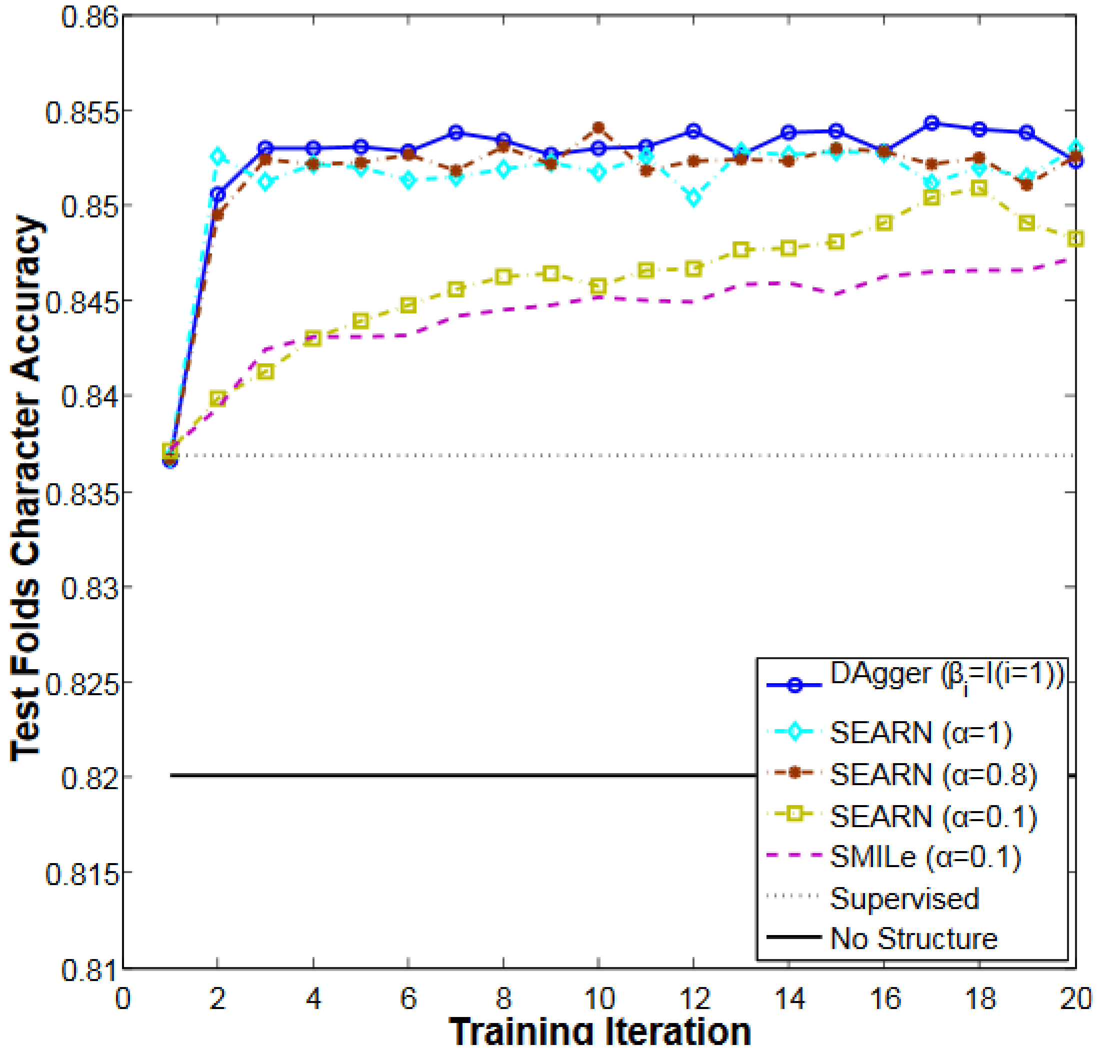
- Generate trajectories using current policy (or some variant)
- Collect additional classification examples using optimal policy (via ground-truth output)

DAgger Algorithm [Ross et al., 2011]

- Collect initial training set D of N trajectories from reference policy π^*
- Repeat until done
  - $\pi \leftarrow \text{LearnClassifier}(D)$
  - Collect set of states S that occur along N trajectories of π
  - For each state $s \in S$
    - $D \leftarrow D \cup \{(s, \pi^*(s))\}$ *// add state labeled by expert or reference policy*
- Return π

Each iteration increases the amount of training data (data aggregation)

DAgger for Handwriting Recognition



- Source: [Ross et al., 2011]

Easy-First Approach: Big Picture

- **Drawbacks of classifier-based structured prediction**
  - Need to define an ordering over the output variables (e.g., left-to-right in sequence labeling)
  - Which order is good? How do you find one?
  - Some decisions are hard to make if you pre-define a fixed order over the output variables

- **Easy-First Approach: Key Idea**
  - Make easy decisions first to constrain the harder decisions
  - Learns to dynamically order the decisions
  - Analogous to constraint satisfaction algorithms

Easy-First Learning as Imitation Learning

- Imitation learning with a non-deterministic oracle policy
  - multiple good decisions (actions) at a state
- Ties are broken with the learned policy (scoring function)
- NLP researchers employ imitation learning ideas and call them "training with exploration"
  - Miguel Ballesteros, Yoav Goldberg, Chris Dyer, Noah A. Smith: *Training with Exploration Improves a Greedy Stack-LSTM Parser*. CoRR abs/1603.03793 (2016)
- Imitation learning ideas are also employed in training recurrent neural networks (RNNs) under the name "scheduled sampling"
  - Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer: *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. NIPS 2015

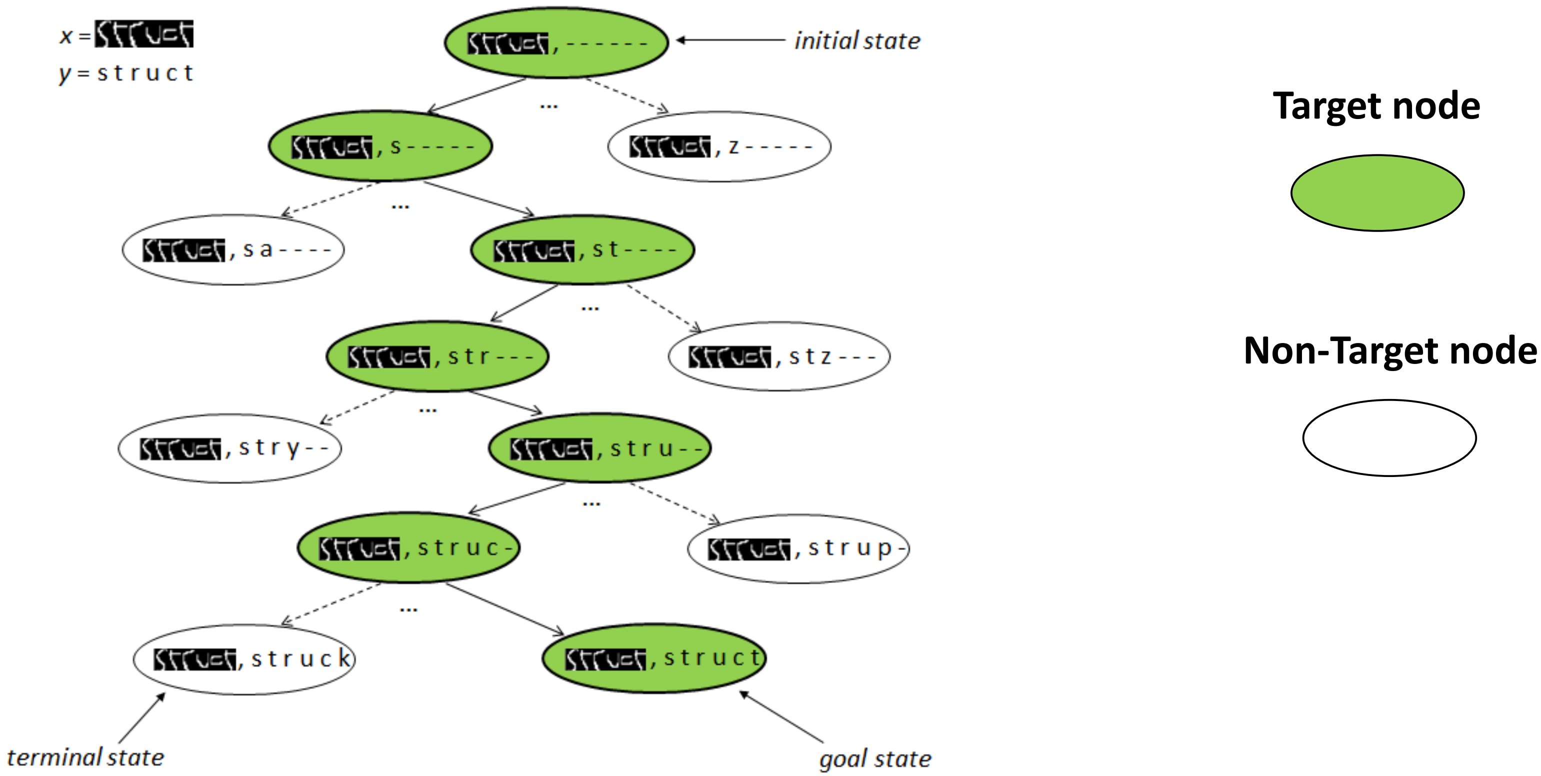
Part 4: Control Knowledge Learning: Beam Search Methods

Beam Search Framework

- **Given**
  - Search space definition (ordered or unordered)
  - Training examples (input-output pairs)
  - Beam width B (>1)
- **Learning Goal**
  - Learn a heuristic function to quickly guide the search to the correct "complete" output
- **Key Idea:**
  - Structured prediction as a search problem in the space of partial outputs
  - Training examples define target paths from initial state to the goal state (correct structured output)

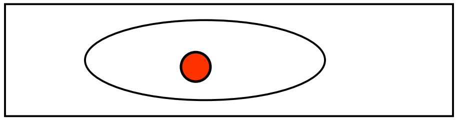
Beam Search Framework: Key Elements

- 1) Search space; 2) Search procedure; 3) **Heuristic function**

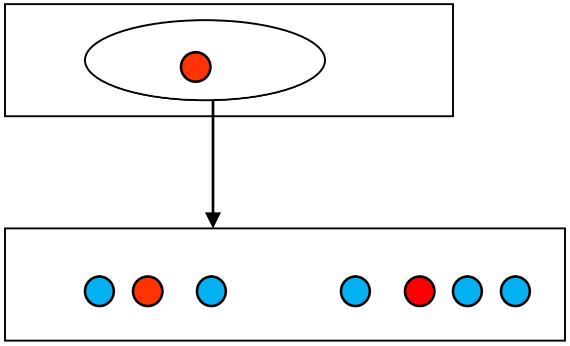


- Represent heuristic function as a linear function
  - $H(n) = w \cdot \psi(n)$, where $\psi(n)$ stands for features of node n

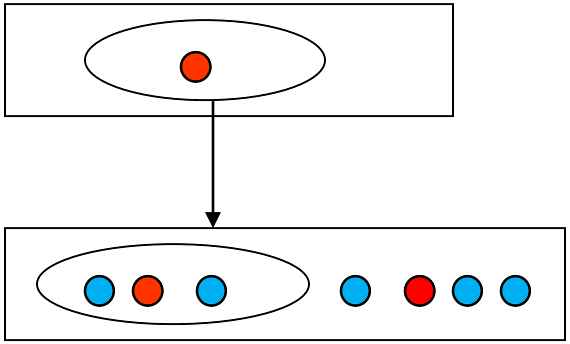
Beam Search: Illustration



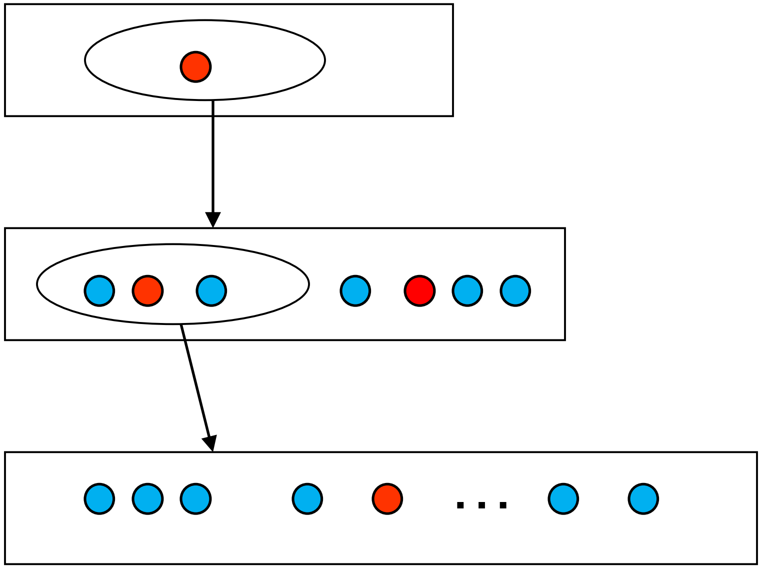
Beam Search: Illustration



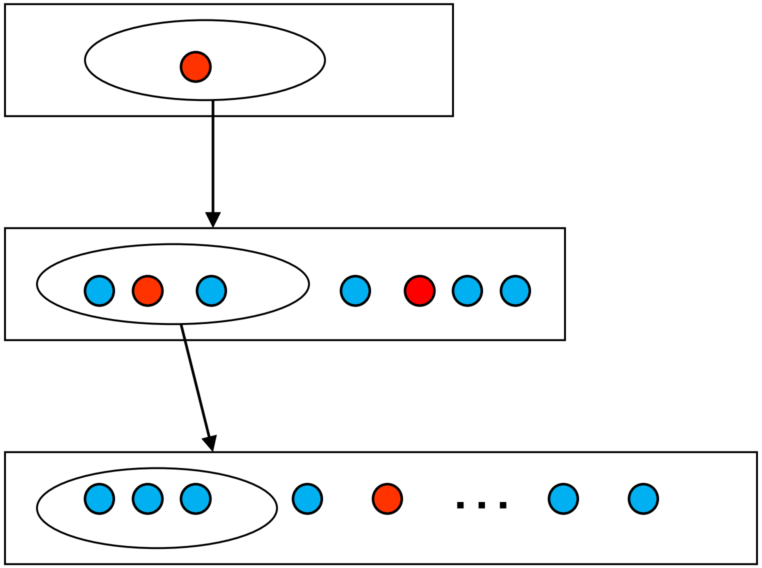
Beam Search: Illustration



Beam Search: Illustration



Beam Search: Illustration



Beam Search Framework: Inference

- **Input:** learned weights w; beam width B; structured input x
- **repeat**
  - Perform search with heuristic $H(n) = w \cdot \psi(n)$
- **until** *reaching a terminal state*
- **Output:** the complete output y corresponding to the terminal state

Beam Search Framework: Generic Learning Template

- **Three design choices**
  - How to define the notion of "search error"?
  - How to "update the weights" of heuristic function when a search error is encountered?
  - How to "update the beam" after weight update?

Beam Search Framework: Learning Instantiations

- Early update
  [Collins and Roark, 2004]

- Max-violation update
  [Huang et al., 2012]

- Learning as Search Optimization (LaSO)
  [Daume et al., 2005], [Xu et al., 2009]

Beam Search Framework: LaSO

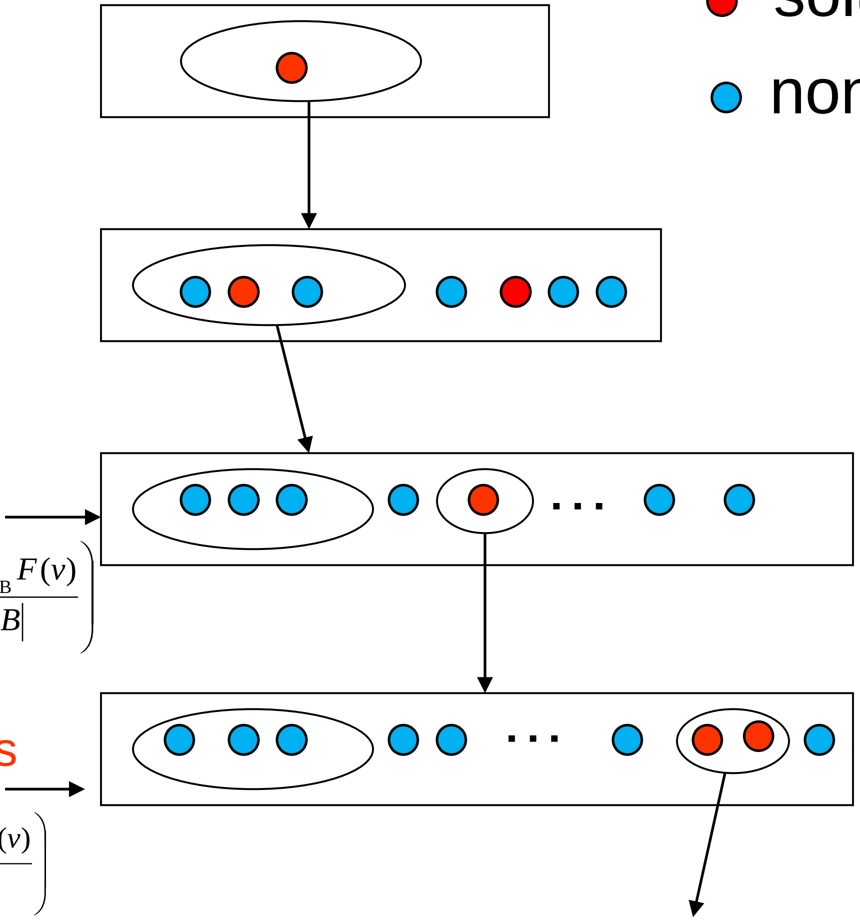
- **Search error:** NO target node in the beam
  - We cannot reach the goal node (correct structured output)
- **Weight update:** perceptron update
  - $w_{new} = w_{old} + \alpha \cdot (\psi_{avg}(target) - \psi_{avg}(non-target))$
  - $\psi_{avg}(target)$ = Average features of all target nodes in the candidate set
  - $\psi_{avg}(non-target)$= Average features of all non-target nodes in the candidate set
  - **Intuition:** increase the score of target nodes and decrease the score of the non-target nodes
- **Beam update:** reset beam with target nodes in the candidate set

LaSO Training: Illustration

Basic Idea: repeatedly conduct search on training examples update weights when error occurs

solution node

non-solution node



An error occurs

$$w = w + \alpha \times \left(\frac{\sum_{v^* \in P_{i,j} \cap C} F(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} F(v)}{|B|} \right)$$

An error occurs

$$w = w + \alpha \times \left(\frac{\sum_{v^* \in P_{i,j} \cap C} F(v^*)}{|P_{i,j} \cap C|} - \frac{\sum_{v \in B} F(v)}{|B|} \right)$$

Beam Search Framework: LaSO

- **repeat**
  - For every training example (x, y)
    - Perform search with current heuristic (weights)
    - If search error, update weights
    - Reset beam with target nodes in the candidate set
    - Continue search
- **until** *convergence* or *max. iterations*

LaSO Convergence Results

- Under certain assumptions, LaSO-BR converges to a weight vector that solves all training examples in a finite number of iterations

- **Interesting convergence result**
  - Mistake bound depends on the beam width
  - Formalizes the intuition that learning becomes easier as we increase the beam width (increase the amount of search)
  - First formal result of this kind

Part 5: HC-Search: A Unifying Framework for Cost Function and Control Knowledge Learning

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

HC-Search: A Unifying View

- **Cost Function Learning Approaches**
  - Don’t learn search control knowledge
- **Control Knowledge Learning Approaches**
  - Don’t learn cost functions
- **HC-Search Learning Framework**
  - Unifies the above two frameworks and has many advantages
  - Without H, degenerates to cost function learning
  - Without C, degenerates to control knowledge learning
  - Supports learning to improve both speed and accuracy of structured prediction

HC-Search framework: Inspiration

Traditional AI Search for combinatorial optimization

+

Learning

HC-Search Framework

HC-Search Framework: Overview

- **Key Idea:**
  - Generate high-quality candidate outputs by conducting a time-bounded search guided by a learned heuristic ***H***
  - Score the candidate outputs using a learned cost function ***C*** to select the least cost output as prediction

- **Heuristic Learning**
  - can be done in primitive space (e.g., IJCAI'16 paper on incremental parsing)
  - OR complete output space

IJCAI'16 paper on computing M-Best Modes via Heuristic Search

HC-Search framework: Overview

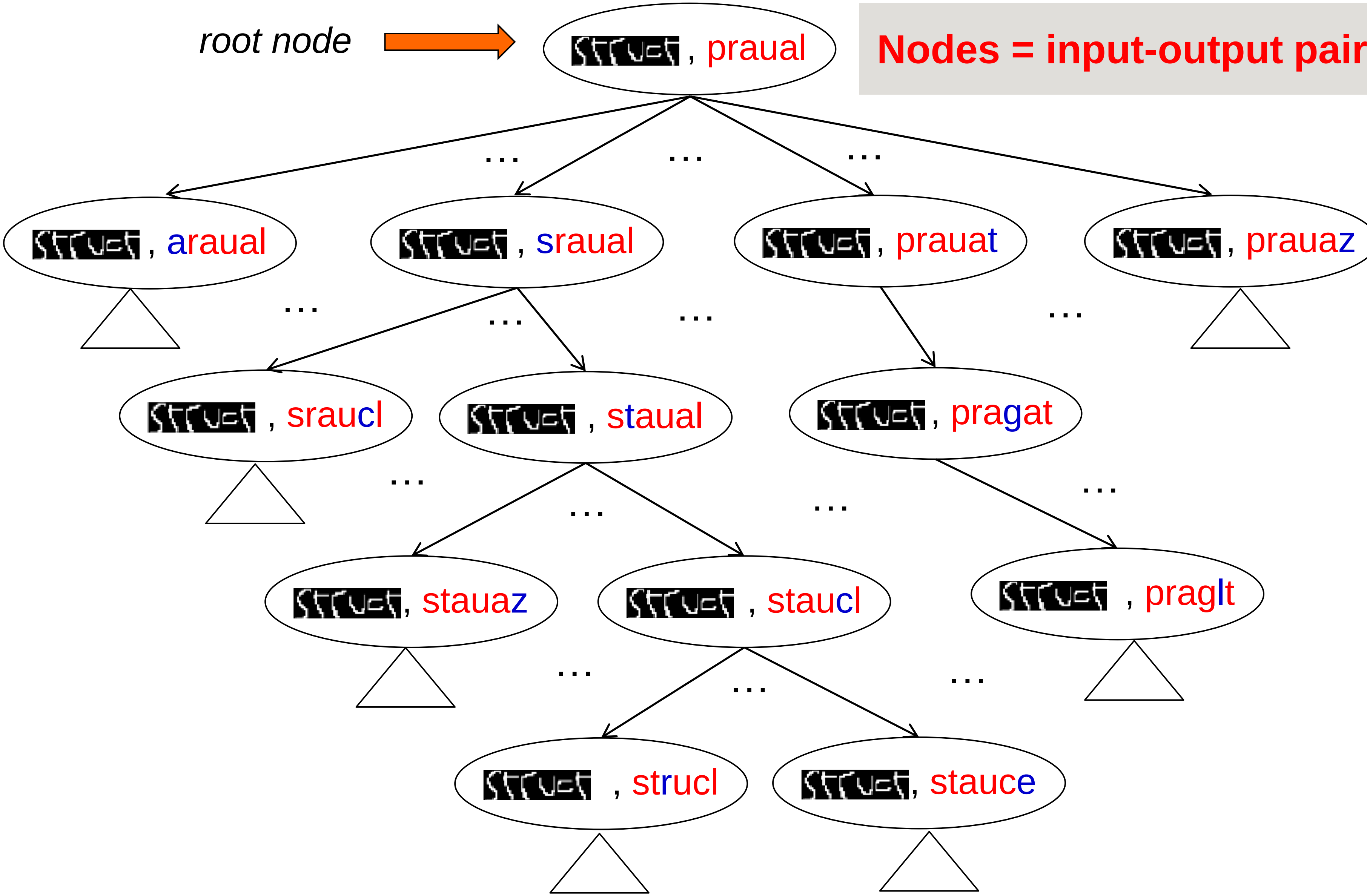
Our approach:

- Structured Prediction as a search process in the combinatorial space of outputs

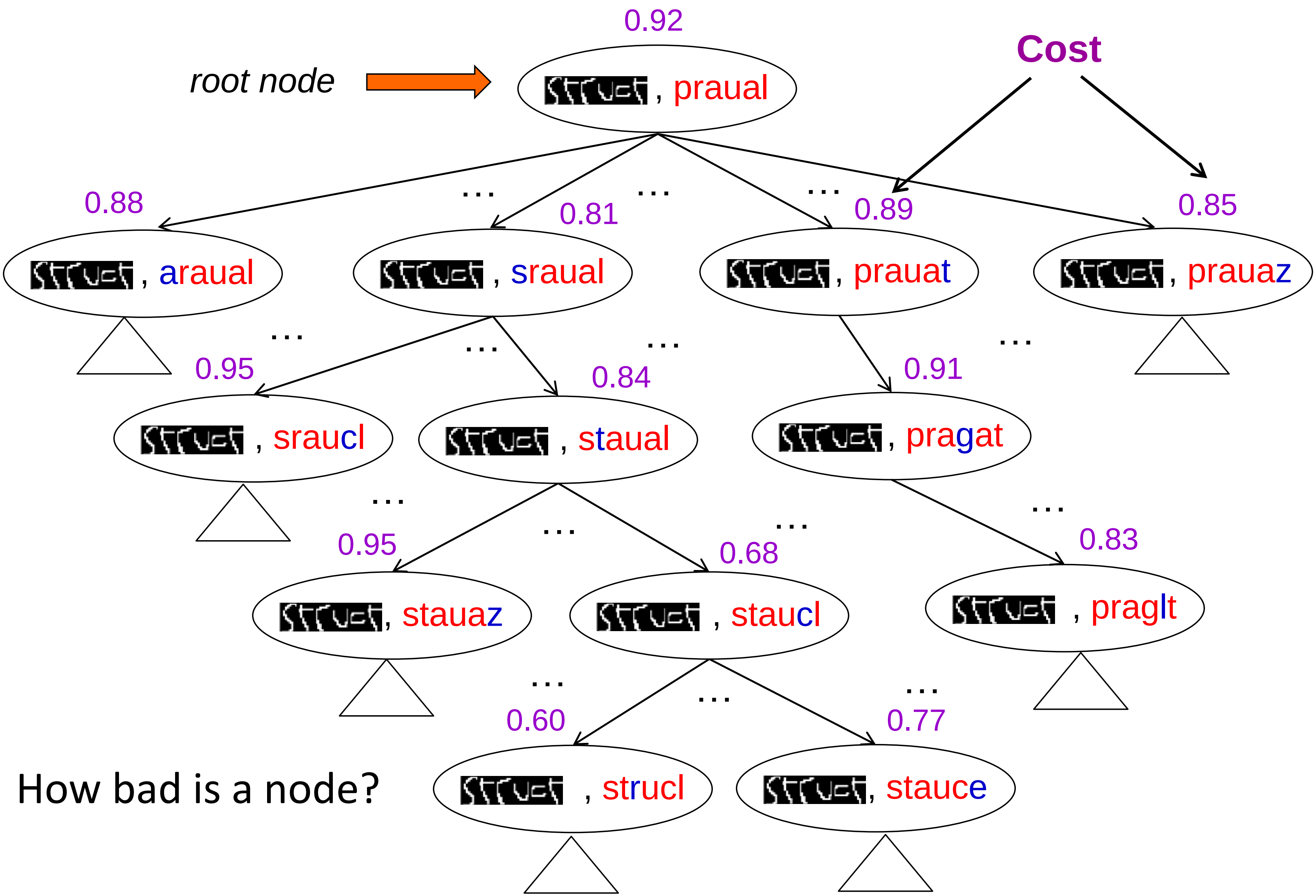
- **Key Ingredients:**
  - Define a search space over structured outputs
  - Learn a cost function $\boldsymbol{C}$ to score potential outputs
  - Use a search algorithm to find low cost outputs
  - Learn a heuristic function $\boldsymbol{H}$ to make search efficient

HC-Search Illustration: Search Space

Nodes = input-output pairs

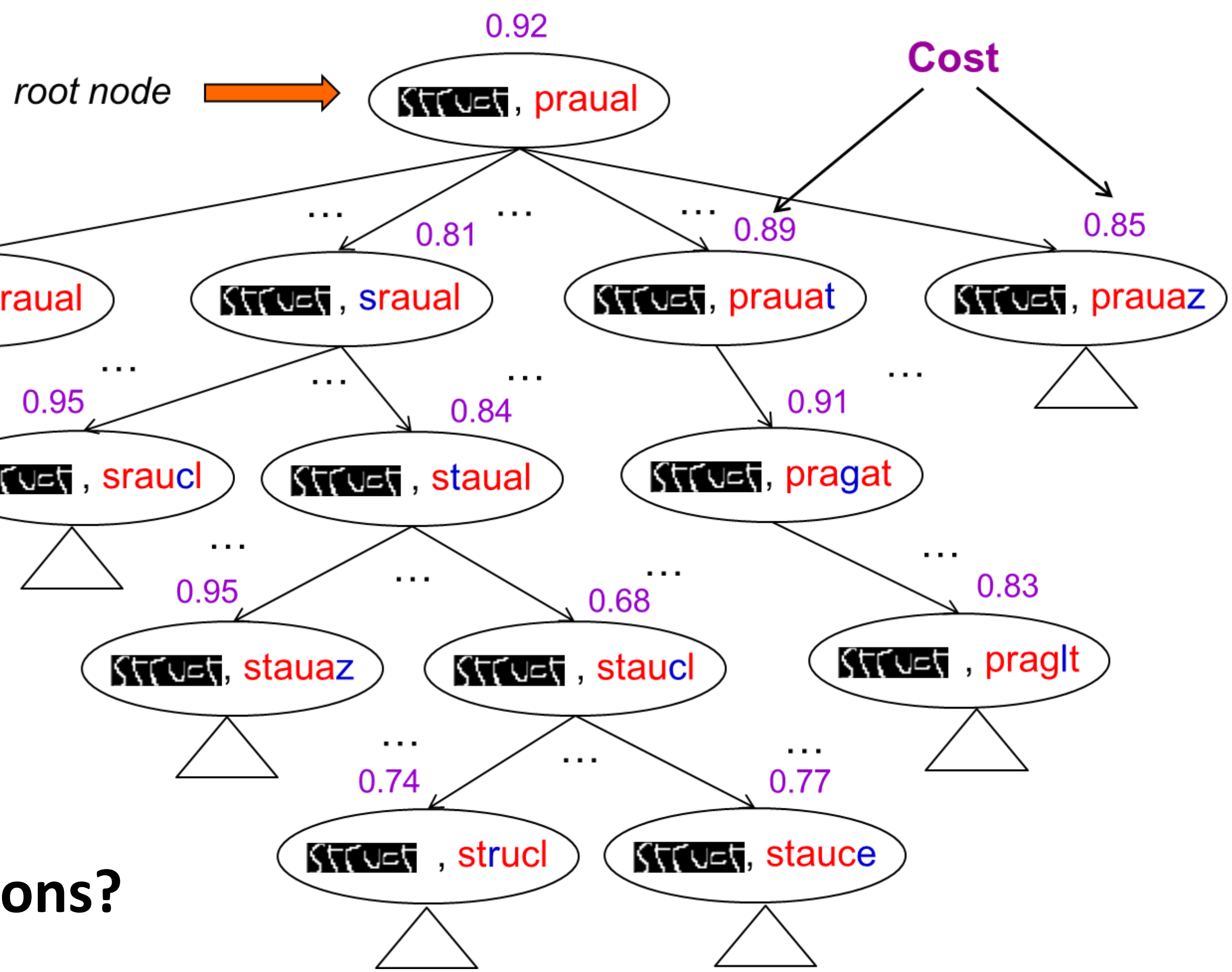


HC-Search Illustration: Cost Function



How bad is a node?

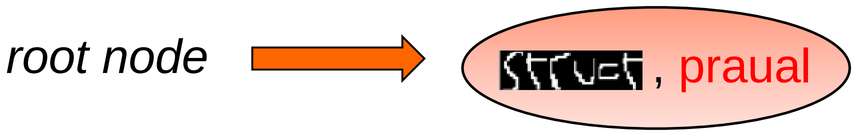
HC-Search Illustration: Making Predictions



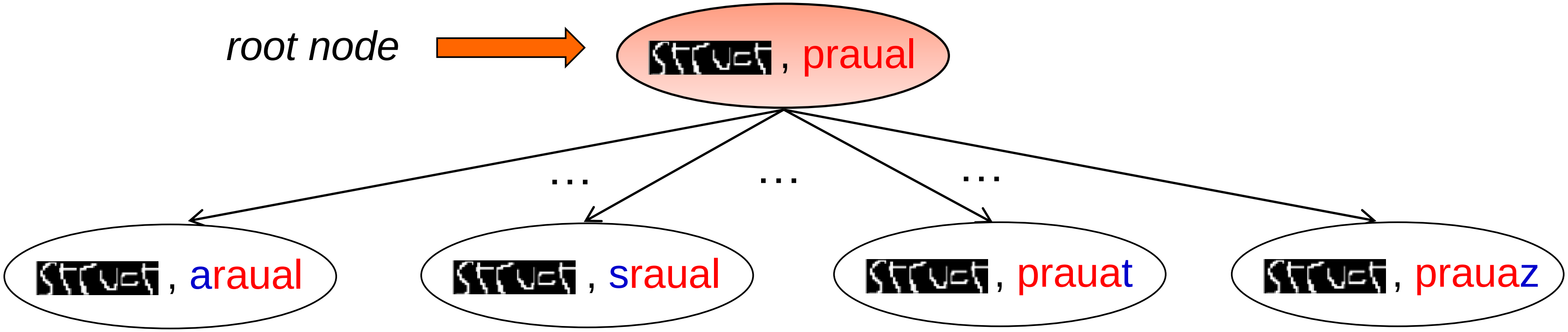
Assume we have a good cost function.

How to make predictions?

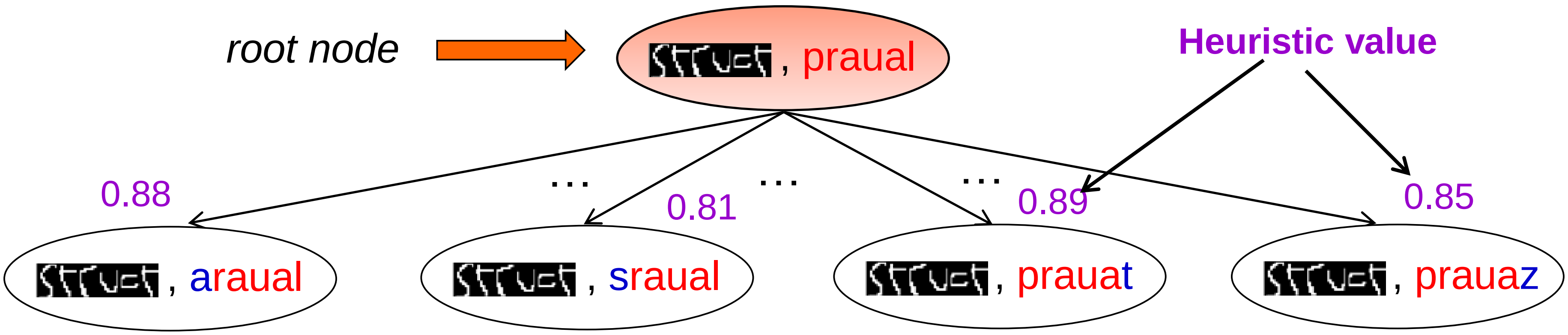
HC-Search Illustration: Greedy Search



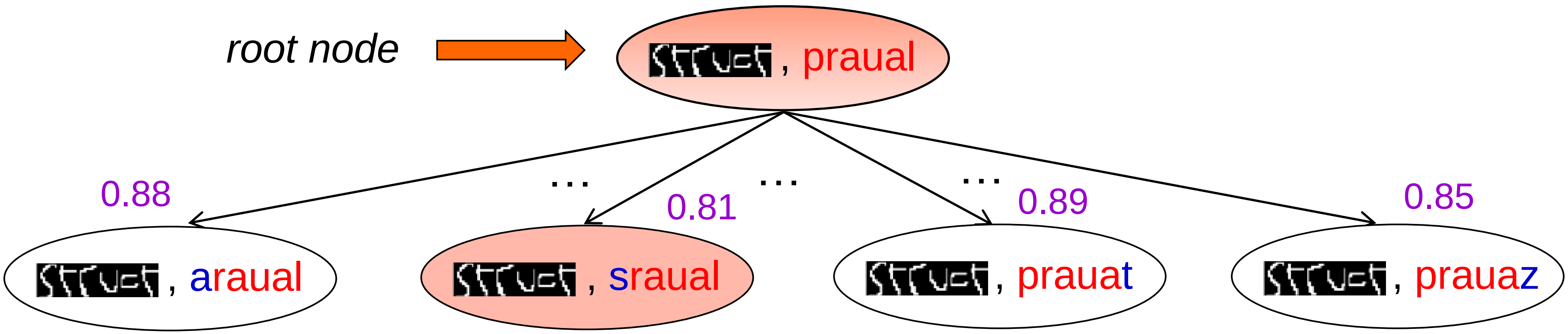
HC-Search Illustration: Greedy Search



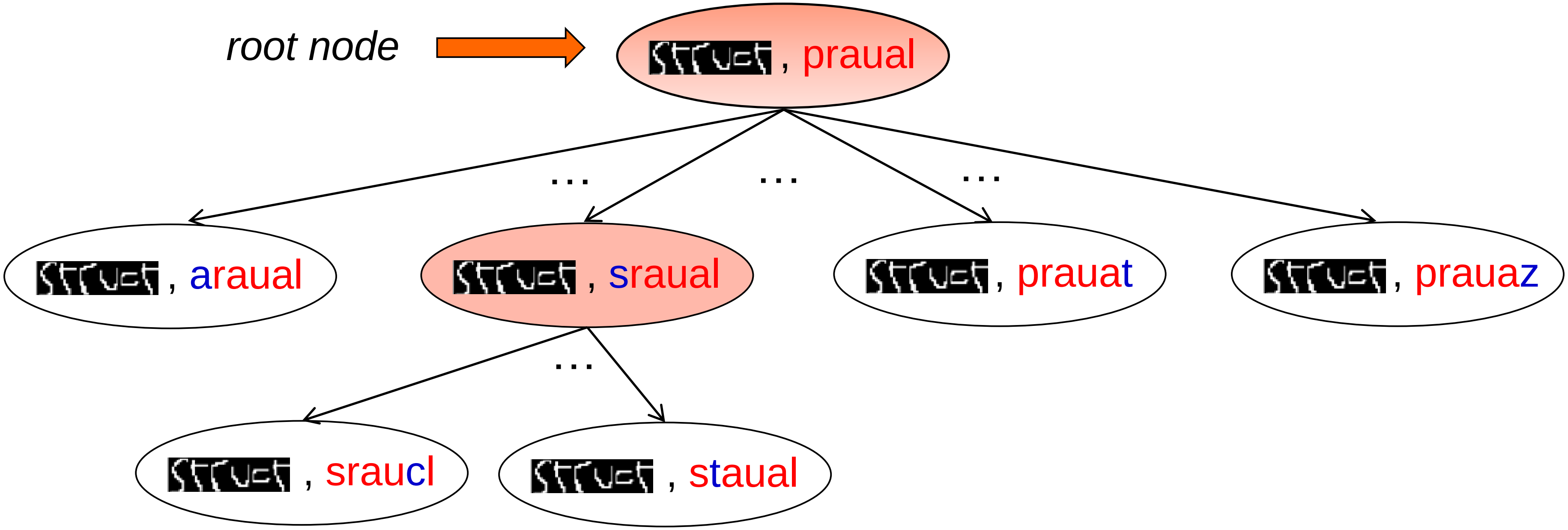
HC-Search Illustration: Greedy Search



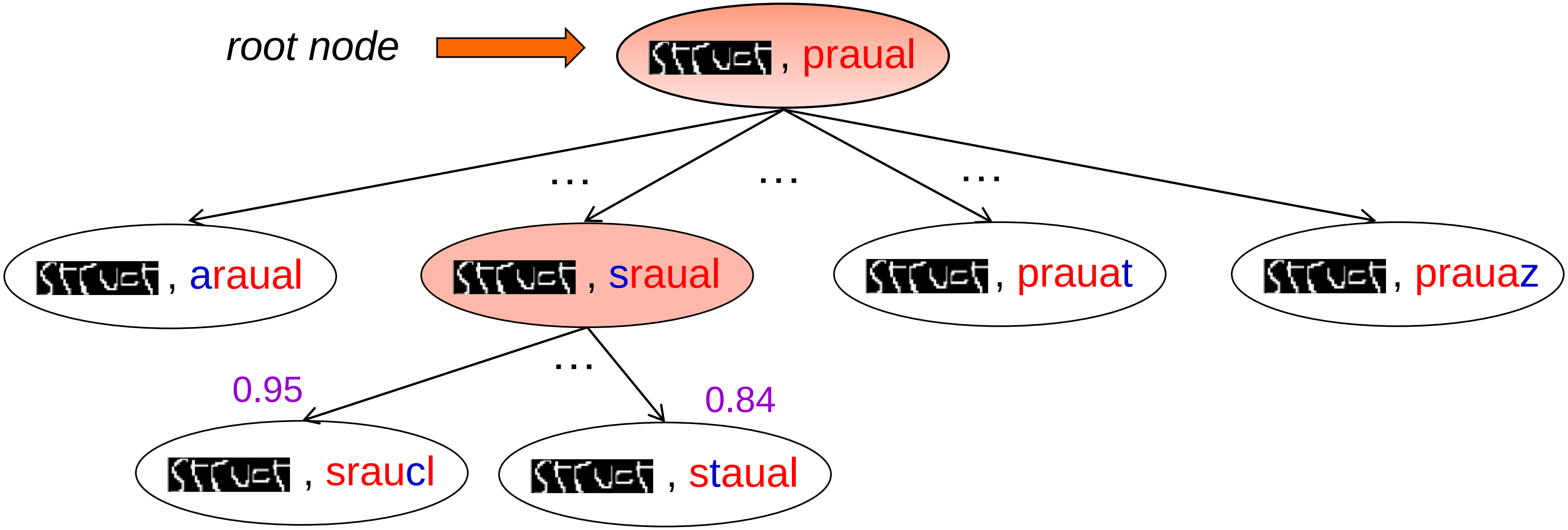
HC-Search Illustration: Greedy Search



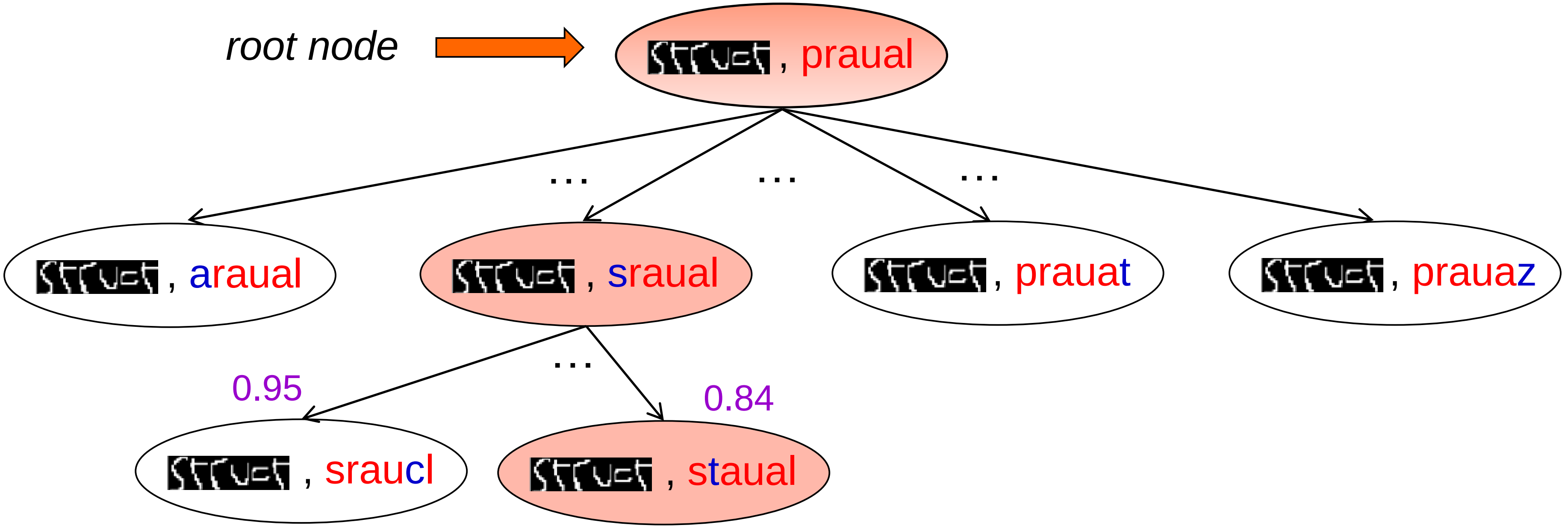
HC-Search Illustration: Greedy Search



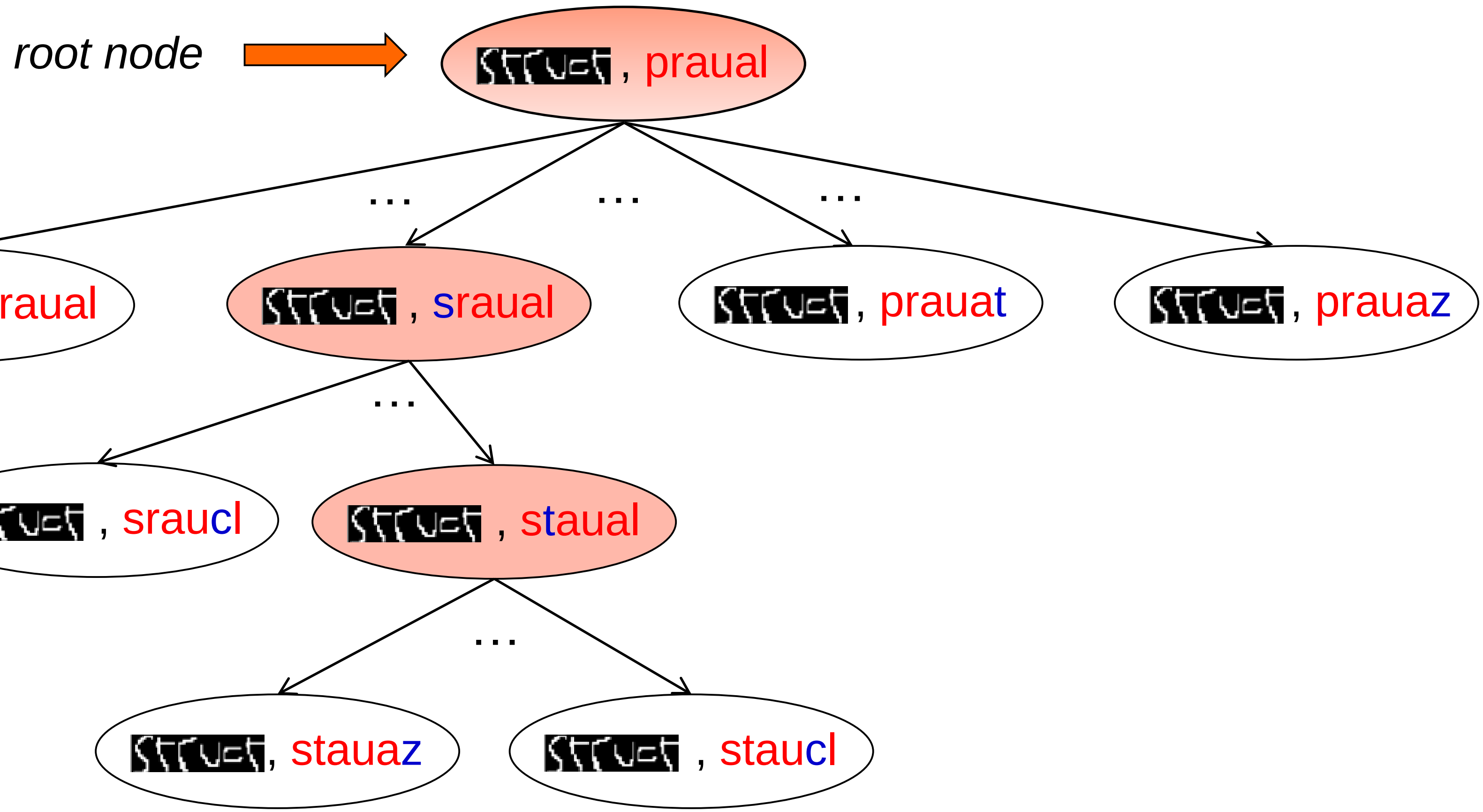
HC-Search Illustration: Greedy Search



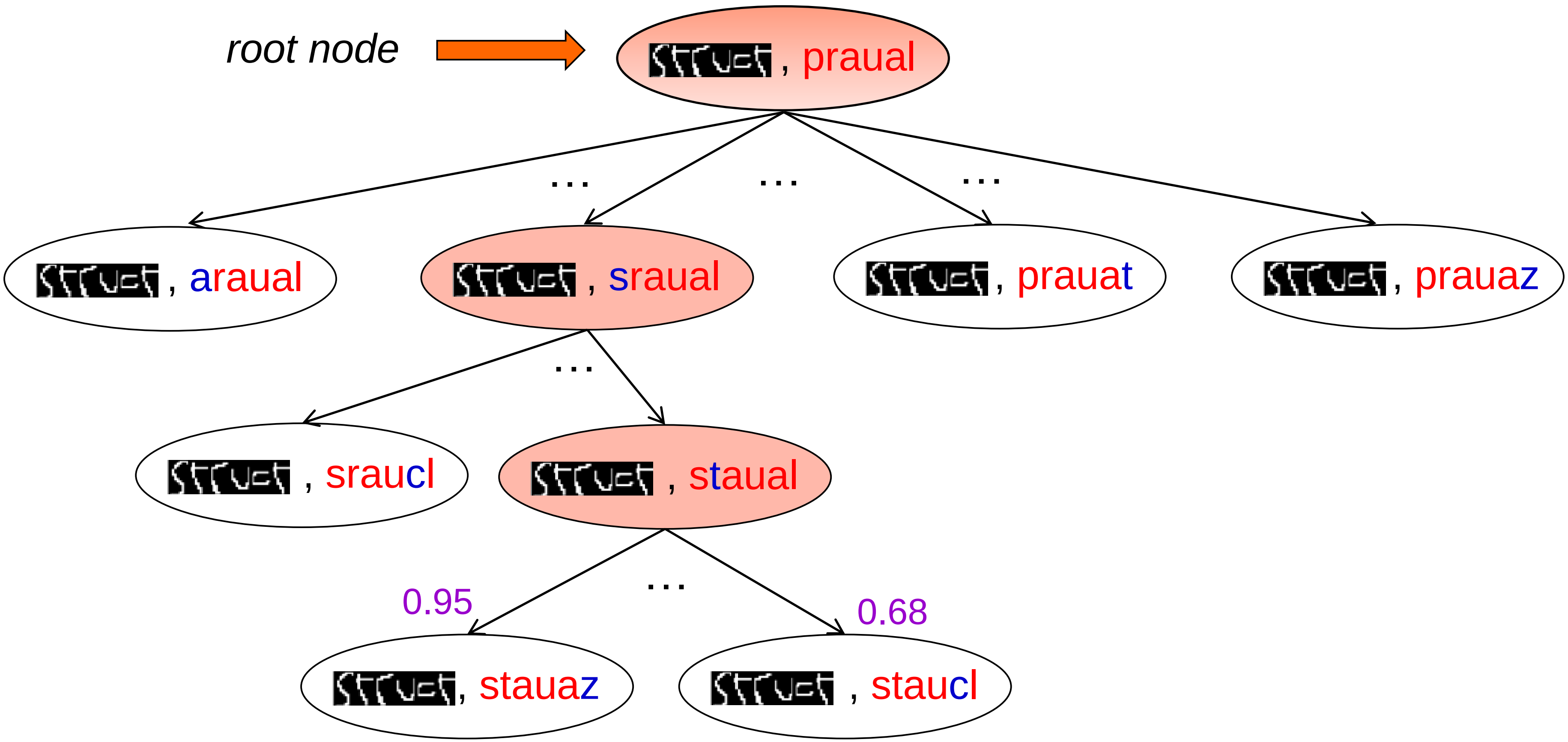
HC-Search Illustration: Greedy Search



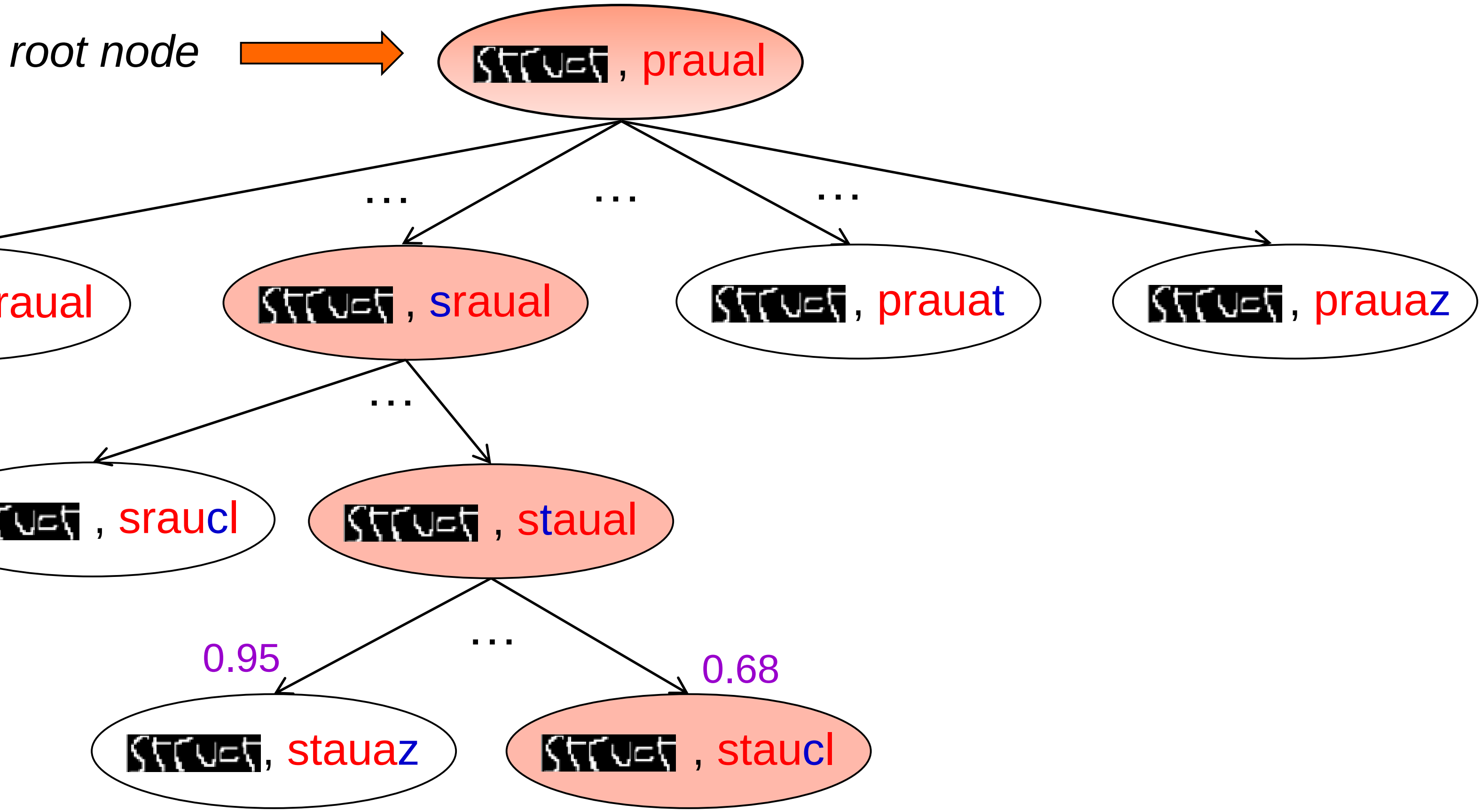
HC-Search Illustration: Greedy Search



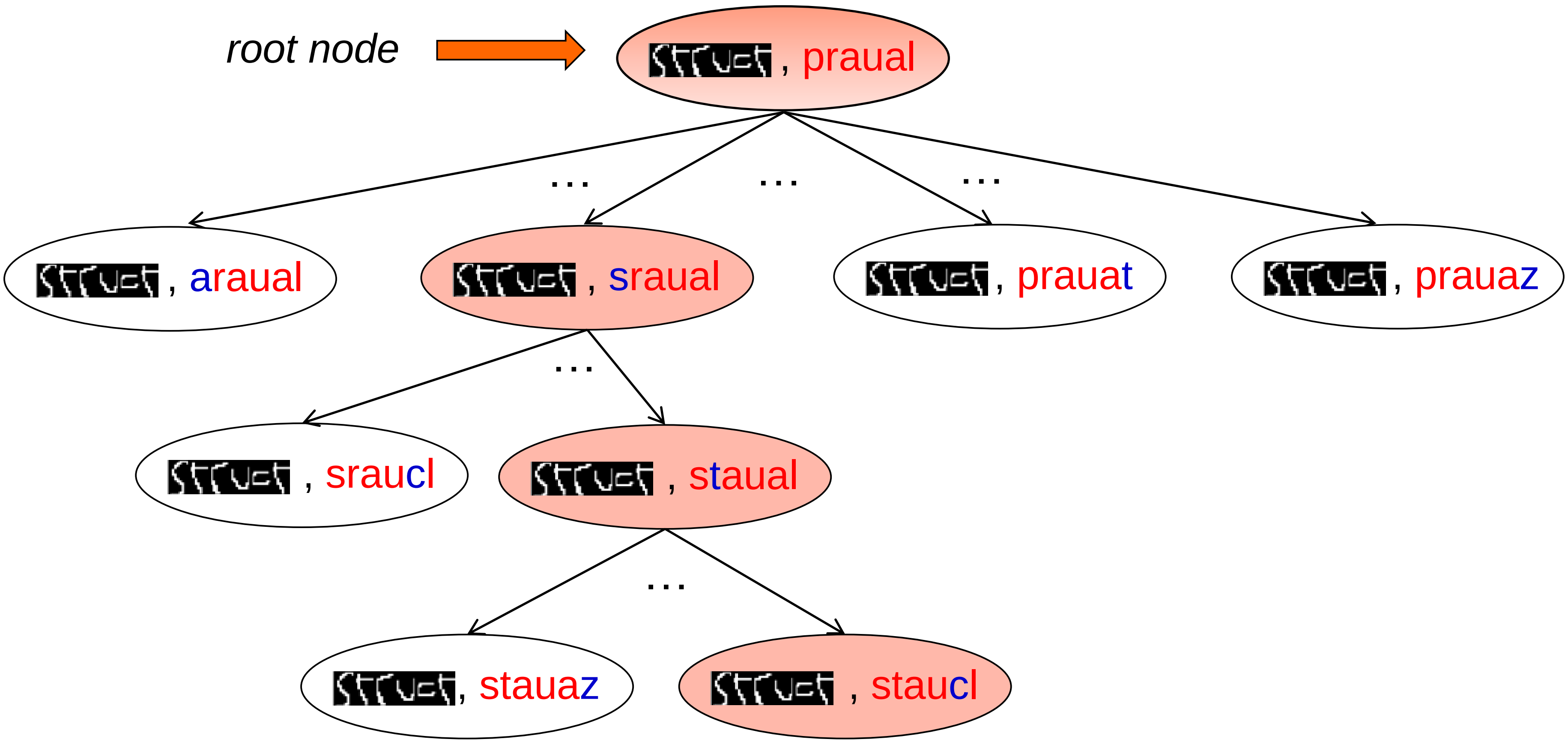
HC-Search Illustration: Greedy Search



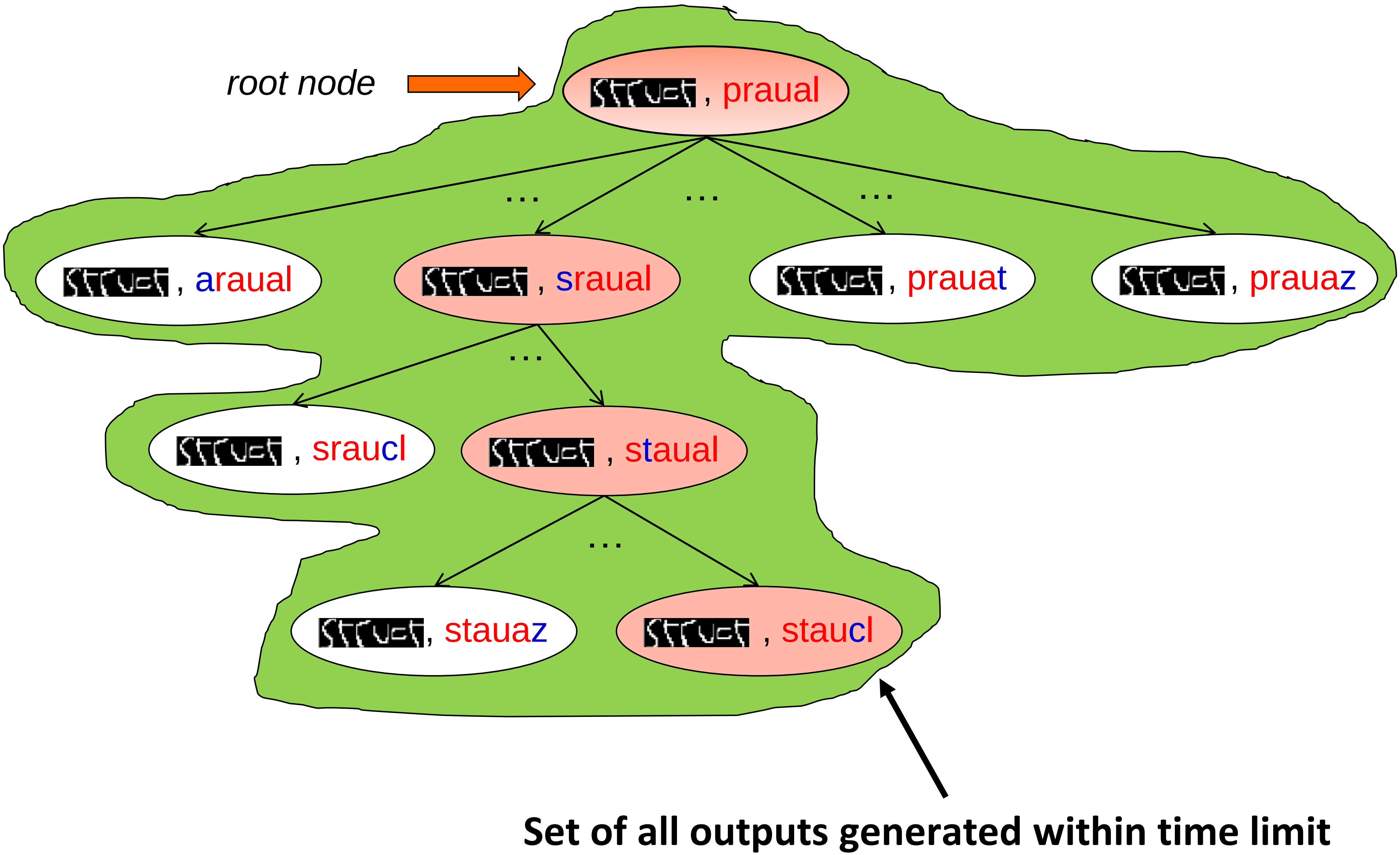
HC-Search Illustration: Greedy Search



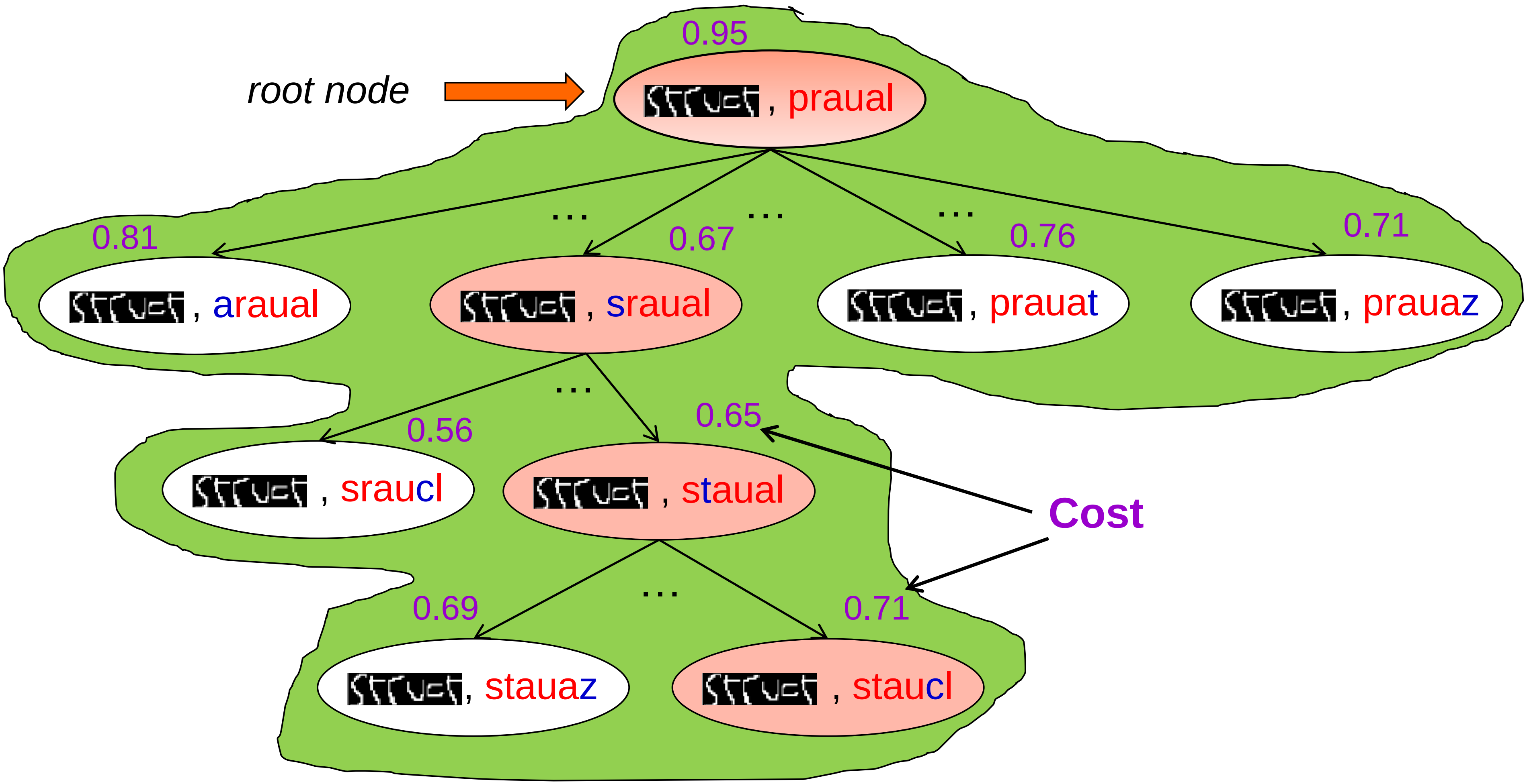
HC-Search Illustration: Greedy Search



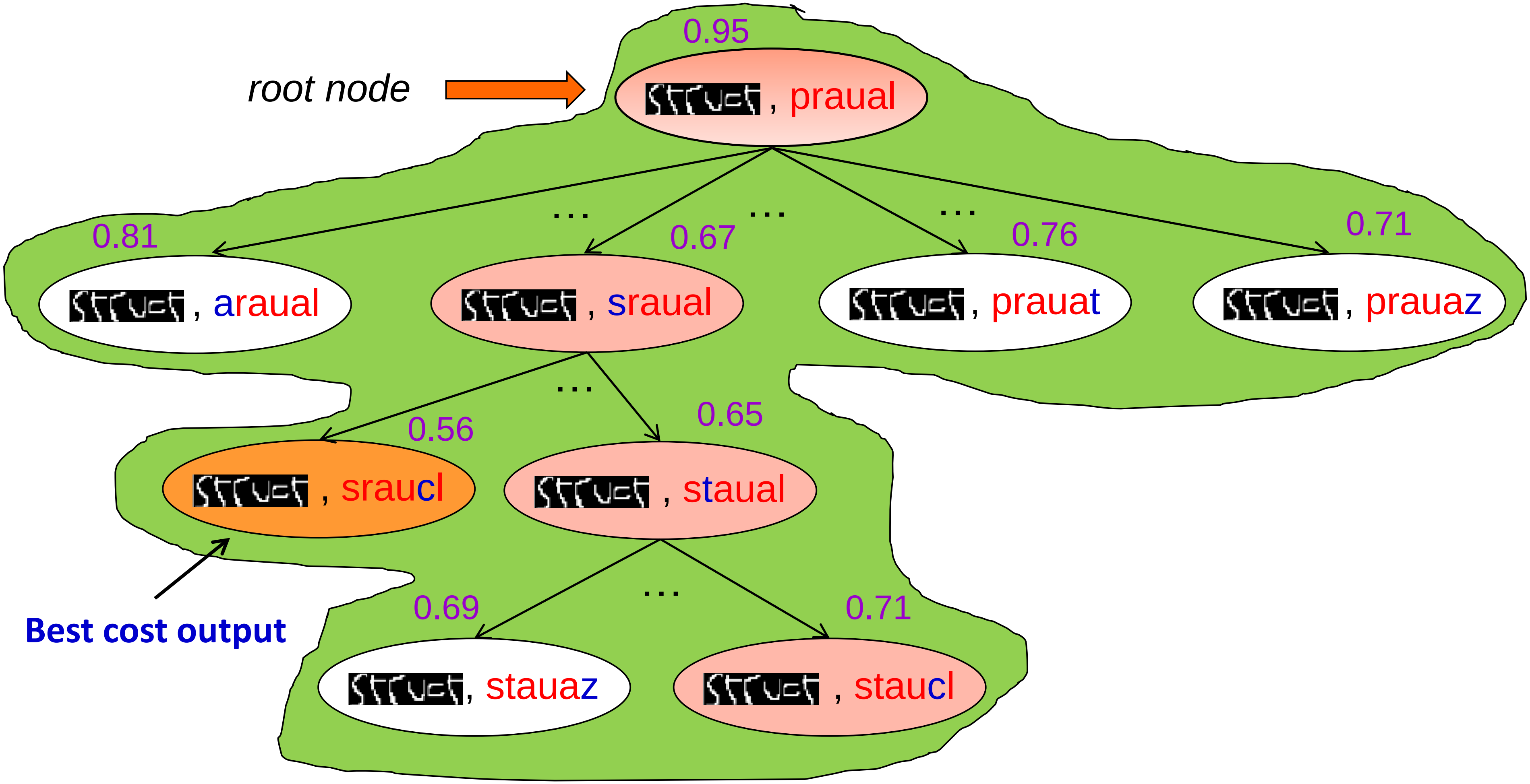
HC-Search Illustration: Greedy Search



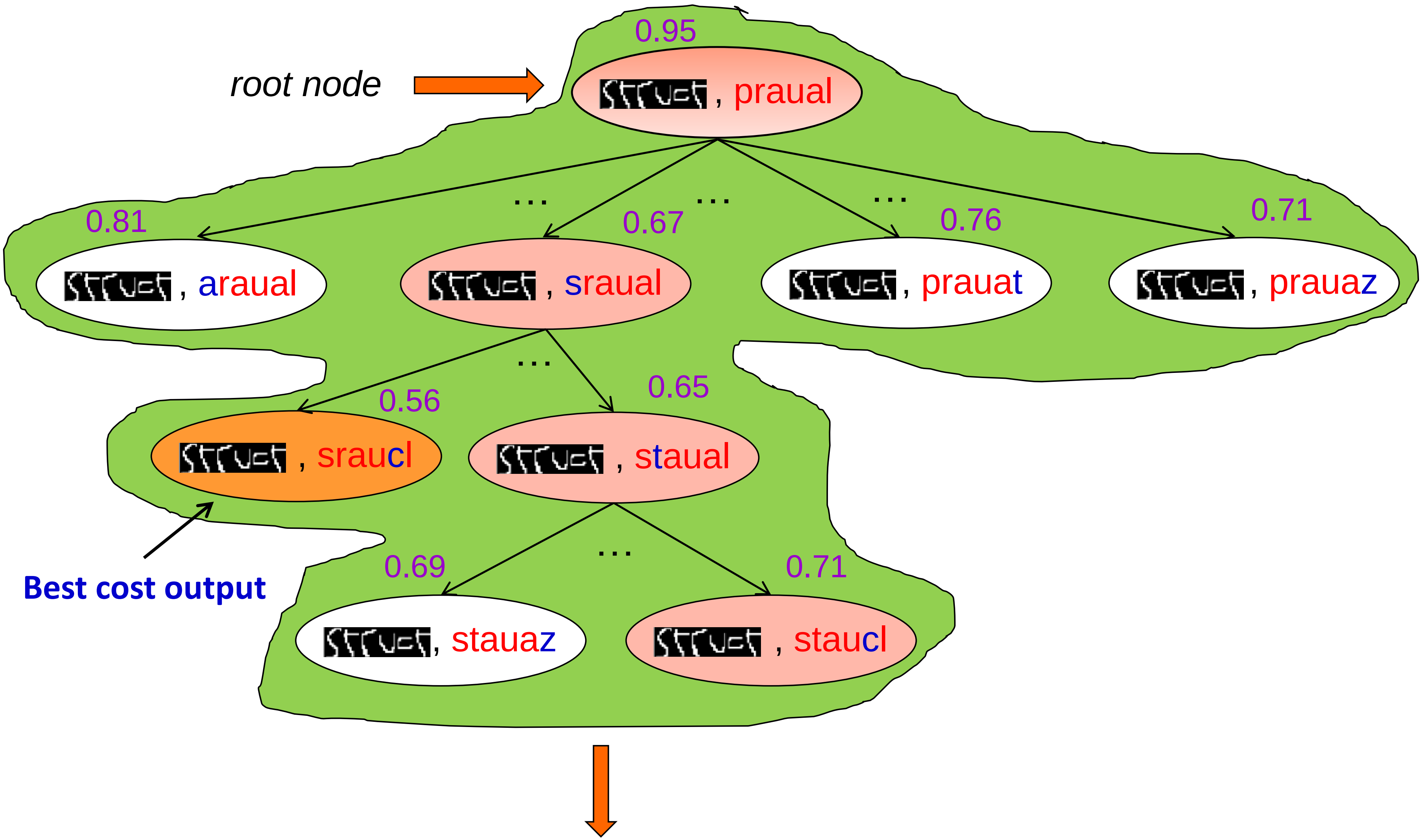
HC-Search Illustration: Greedy Search



HC-Search Illustration: Greedy Search



HC-Search Illustration: Greedy Search



$\hat{y}$ = sraucl

HC-Search: Properties

- **Anytime predictions**
  - Stop the search at any point and return the best cost output
- **Minimal restrictions on the complexity of heuristic and cost functions**
  - Only needs to be evaluated on complete input-output pairs
  - Can use higher-order features with negligible overhead
- **Can optimize non-decomposable loss functions**
  - e.g., F1 score
- **Error Analysis:** Heuristic error + Cost function error
  - engineering methodology guided by the error decomposition

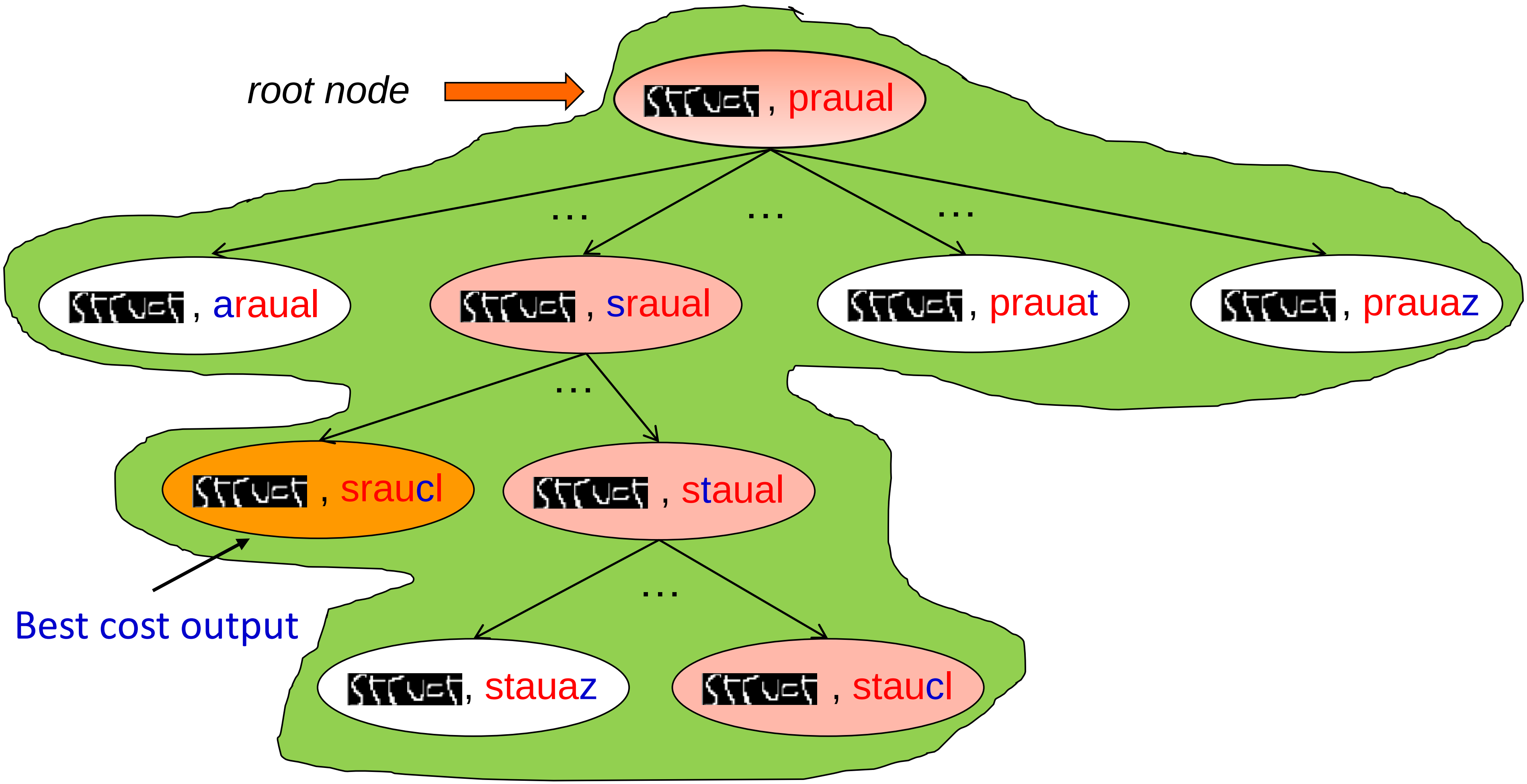
HC-Search: Key Learning Challenges

- **Search Space Design:**
  - How can we automatically define high-quality search spaces ?
- **Heuristic Learning:**
  - How can we learn a heuristic function to guide the search to generate high-quality outputs ?
- **Cost Function Learning:**
  - How can we learn a cost function to score the outputs generated by the heuristic function ?

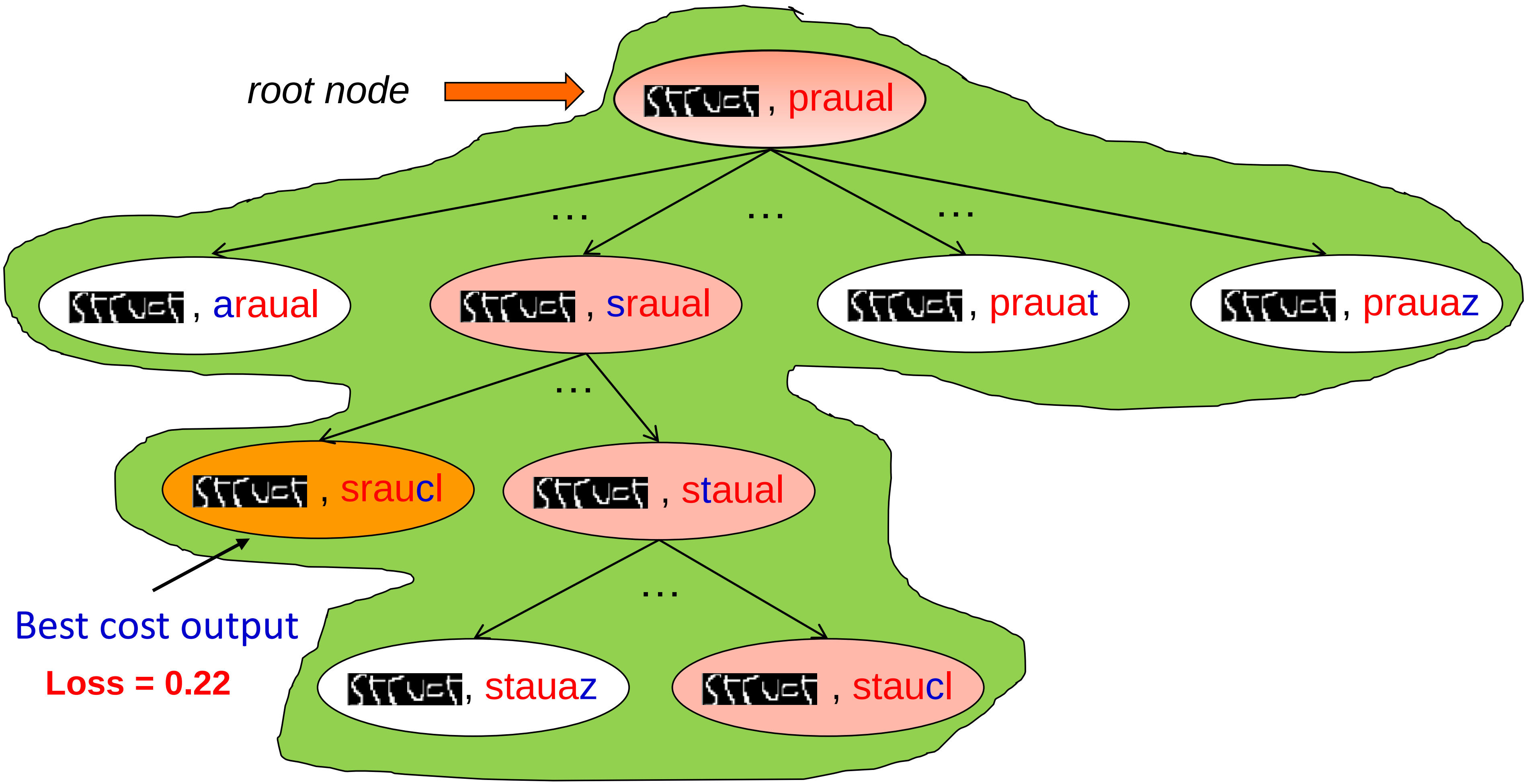
Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

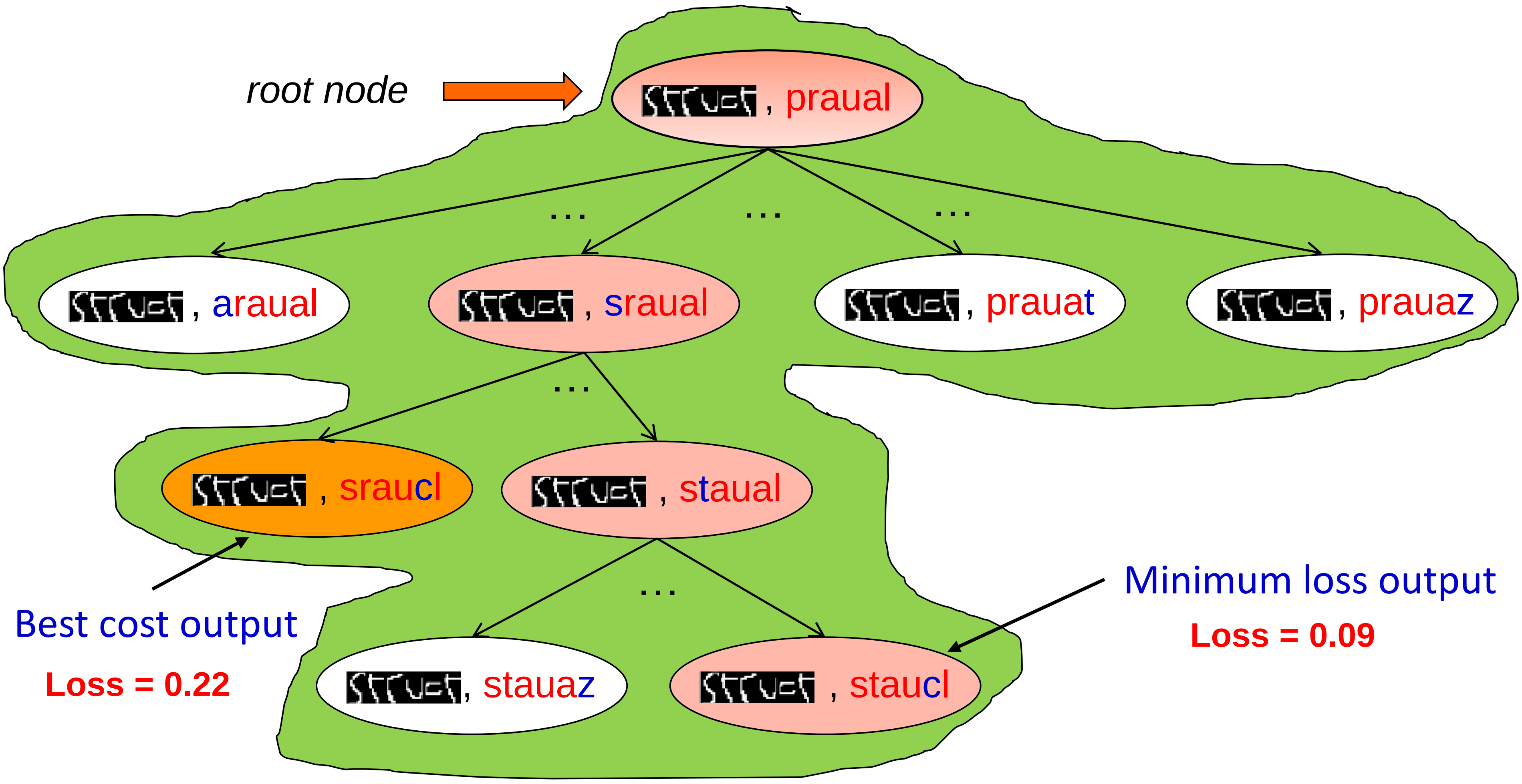
HC-Search: Loss Decomposition



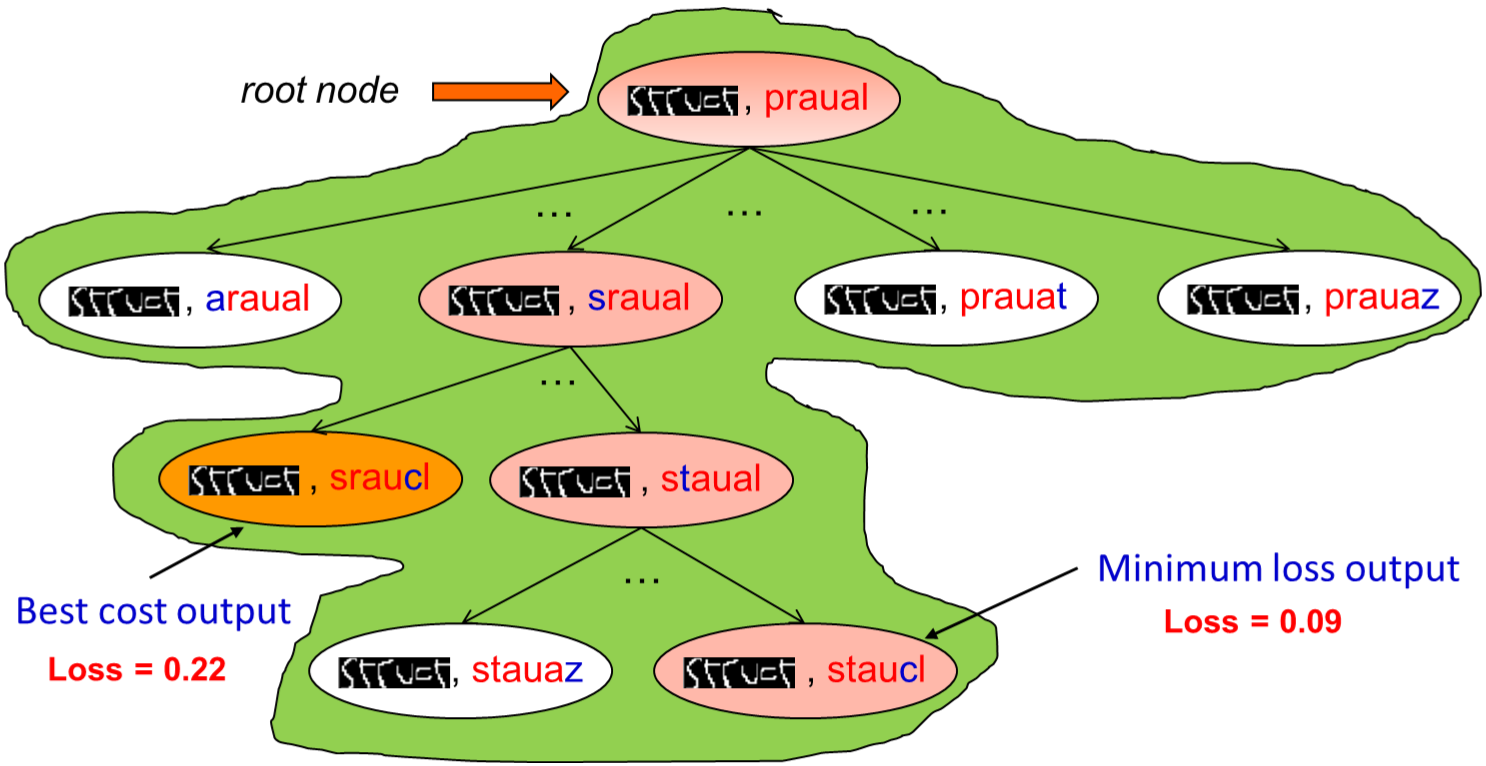
HC-Search: Loss Decomposition



HC-Search: Loss Decomposition



HC-Search: Loss Decomposition



Overall loss ϵ = 0.22

Generation loss ϵ_H = 0.09

(Heuristic function)

Selection loss ϵ_C = 0.22 – 0.09

(Cost function)

HC-Search: Loss Decomposition

$$C(x, y) = w_c \cdot \phi_H(x, y)$$
$$H(x, y) = w_H \cdot \phi_C(x, y)$$

$$\epsilon = \epsilon_H + \epsilon_{C|H}$$

Overall expected loss

Generation loss (Heuristic function)

Selection loss (Cost function)

HC-Search: Learning

$$\epsilon = \epsilon_H + \epsilon_{C|H}$$

Overall loss

Generation loss
(Heuristic function)

Selection loss
(Cost function)

- **Key idea:** Greedy stage-wise minimization guided by the loss decomposition

Doppa, J.R., Fern, A., Tadepalli, P. HC-Search: A Learning Framework for Search-based Structured Prediction. *Journal of Artificial Intelligence Research (JAIR)* 2014.

HC-Search: Learning

$$\epsilon = \epsilon_H + \epsilon_{C|H}$$

Overall loss

Generation loss
(Heuristic function)

Selection loss
(Cost function)

- **<u>Key idea:</u>** Greedy stage-wise minimization guided by the loss decomposition
  - **Step 1:** $\hat{H} = \arg min_{H \in \boldsymbol{H}} \; \epsilon_H$ (heuristic training)

HC-Search: Learning

$$\epsilon = \epsilon_{H} + \epsilon_{C|H}$$

Overall loss

Generation loss
(Heuristic function)

Selection loss
(Cost function)

- **<u>Key idea:</u>** Greedy stage-wise minimization guided by the loss decomposition
  - **Step 1:** $\hat{H} = \arg min_{H \in \boldsymbol{H}}\ \epsilon_{H}$ (heuristic training)
  - **Step 2:** $\hat{C} = \arg min_{C \in \boldsymbol{C}}\ \epsilon_{C|\hat{H}}$ (cost function training)

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

HC-Search: Heuristic learning

- **Learning Objective:**
  - Guide the search quickly towards high-quality (low loss) outputs

HC-Search: Heuristic Learning

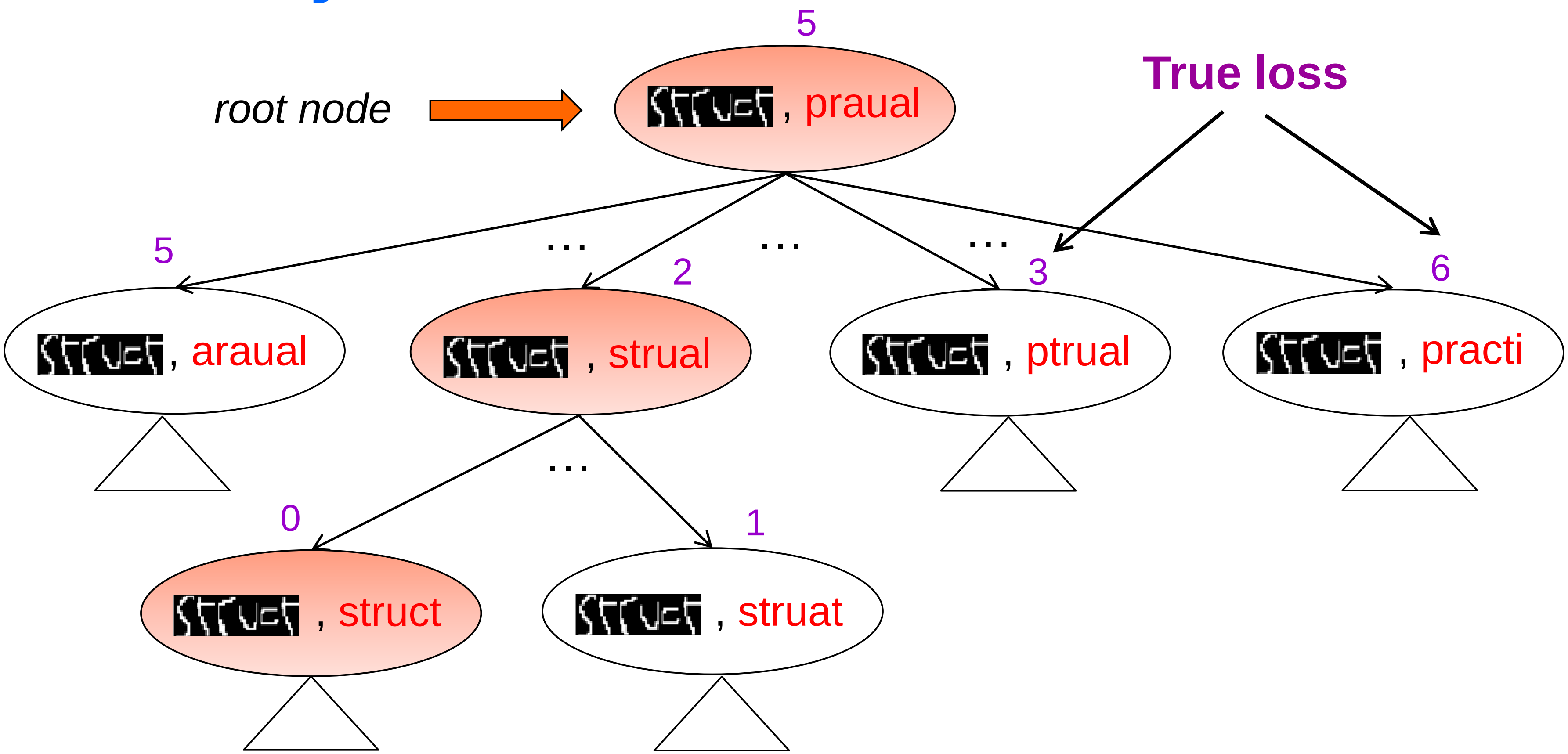
- Given a search procedure (e.g., greedy search)

- **Key idea: Imitation of true loss function**
  - Conduct searches on training example using the true loss function as a heuristic

    (generally is a good way to produce good outputs)

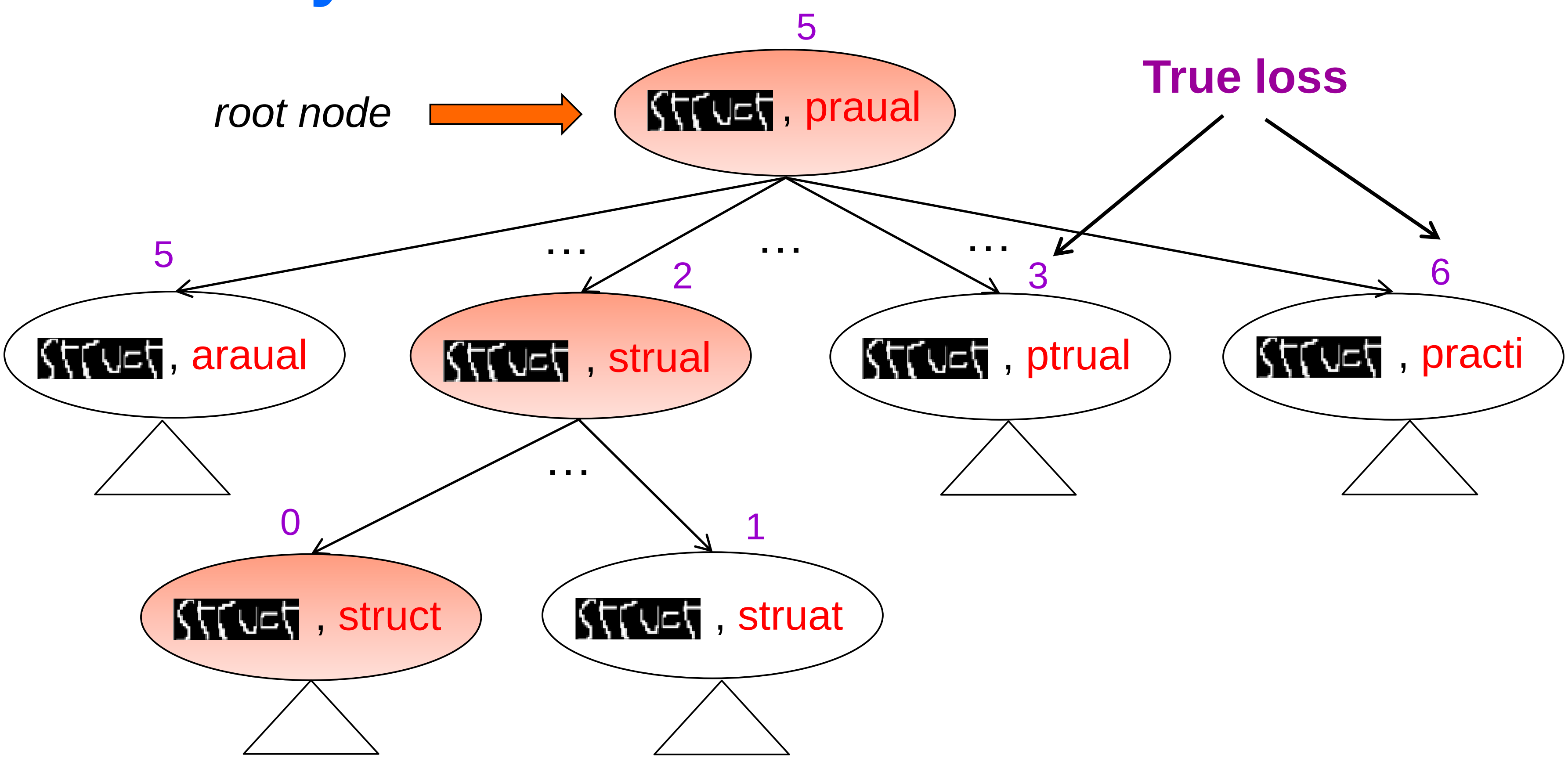
  - Learn a heuristic function that tries to imitate the observed search behavior

Greedy Search: Imitation with true loss



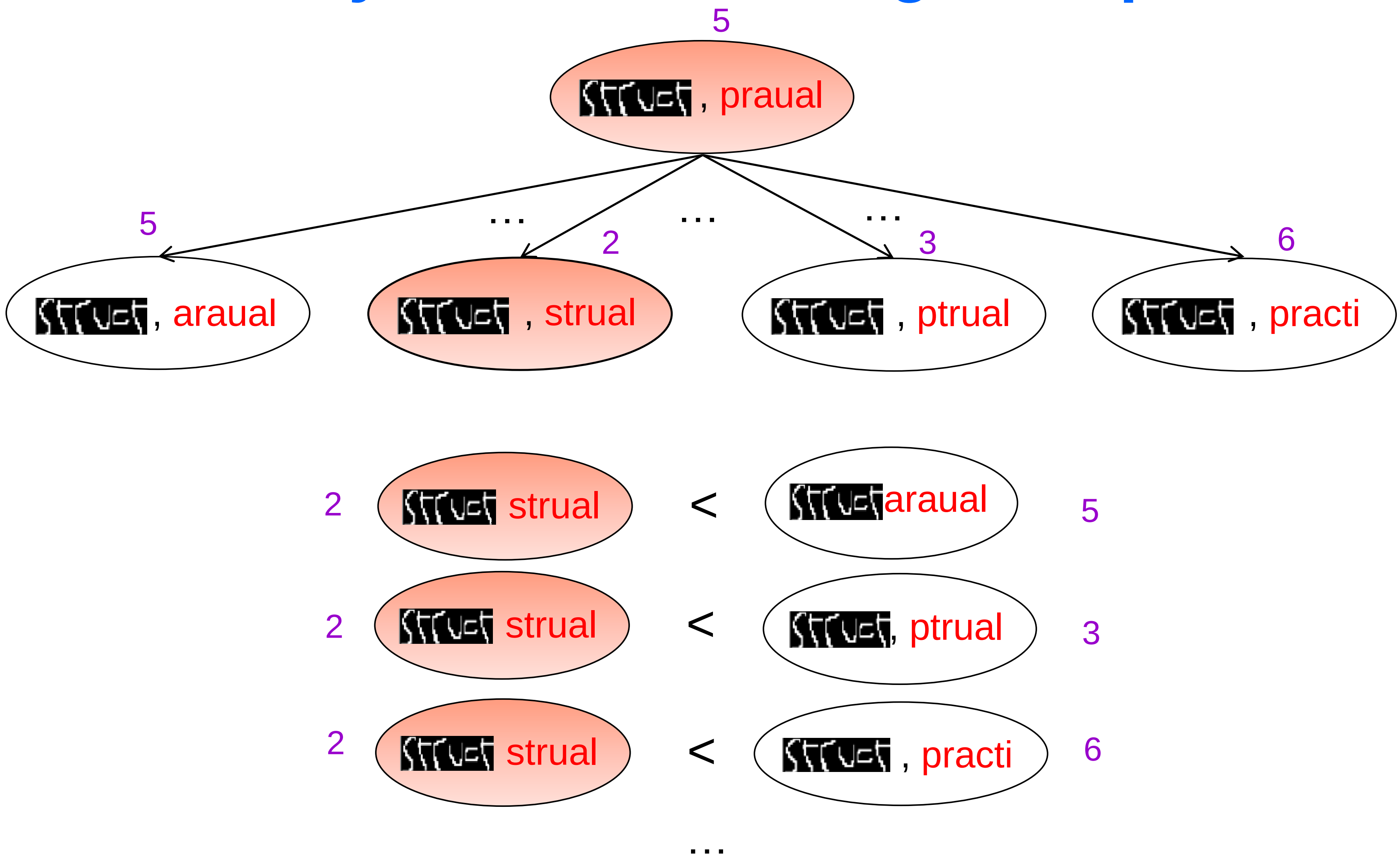
Hamming Loss ( , strual , struct) = 2

Greedy Search: Imitation with true loss

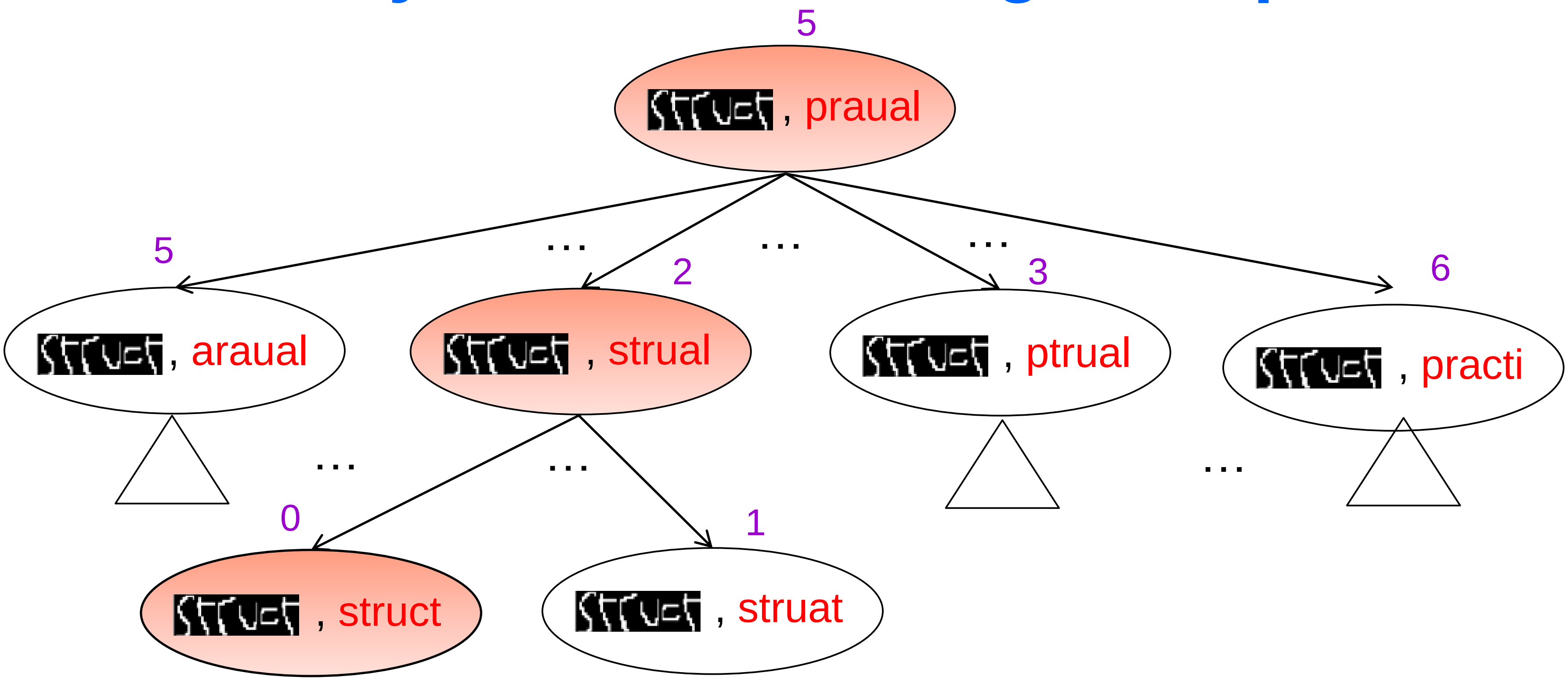


Generation loss $\epsilon_{H^*} = 0$

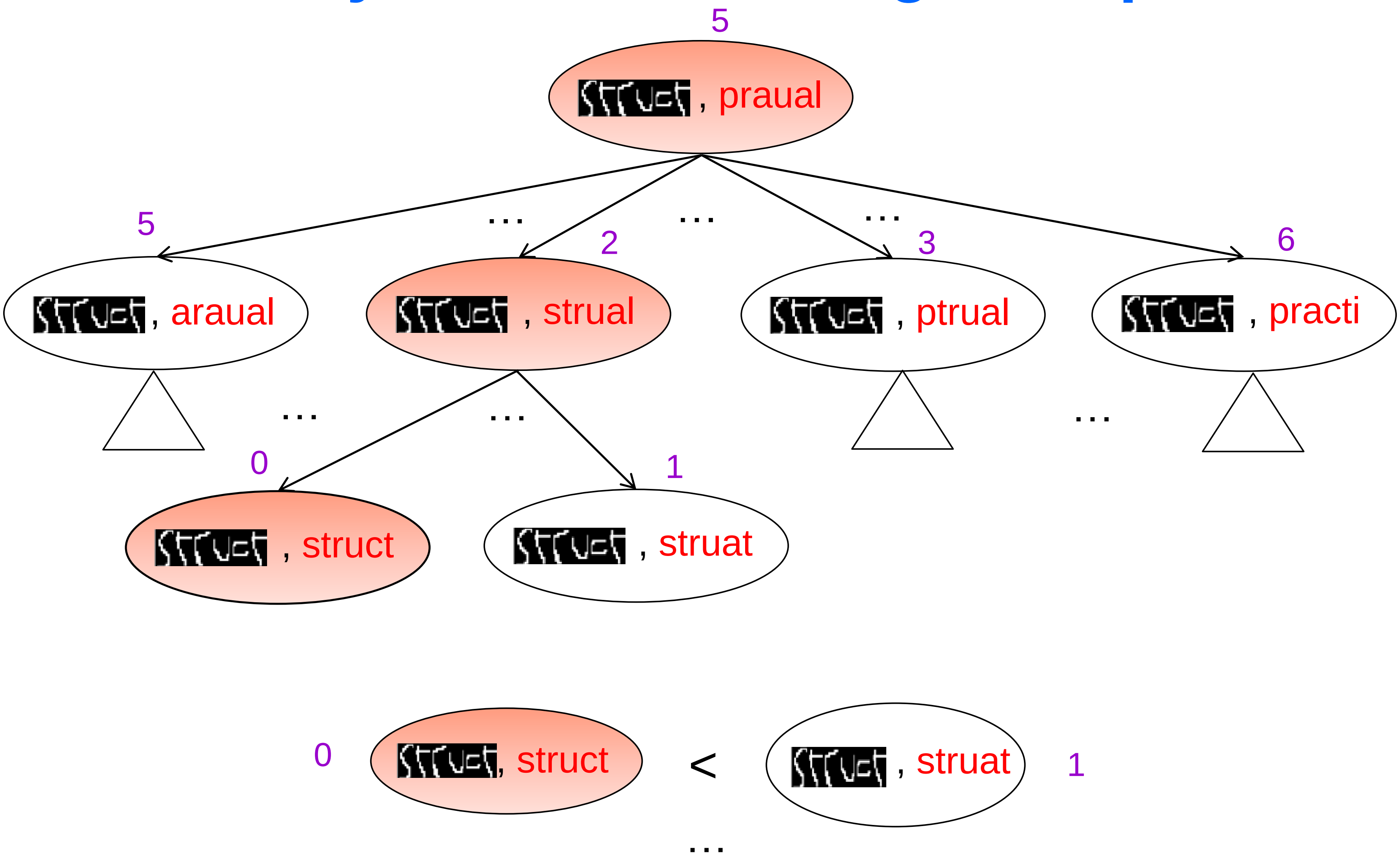
Greedy Search: Ranking examples



Greedy Search: Ranking examples

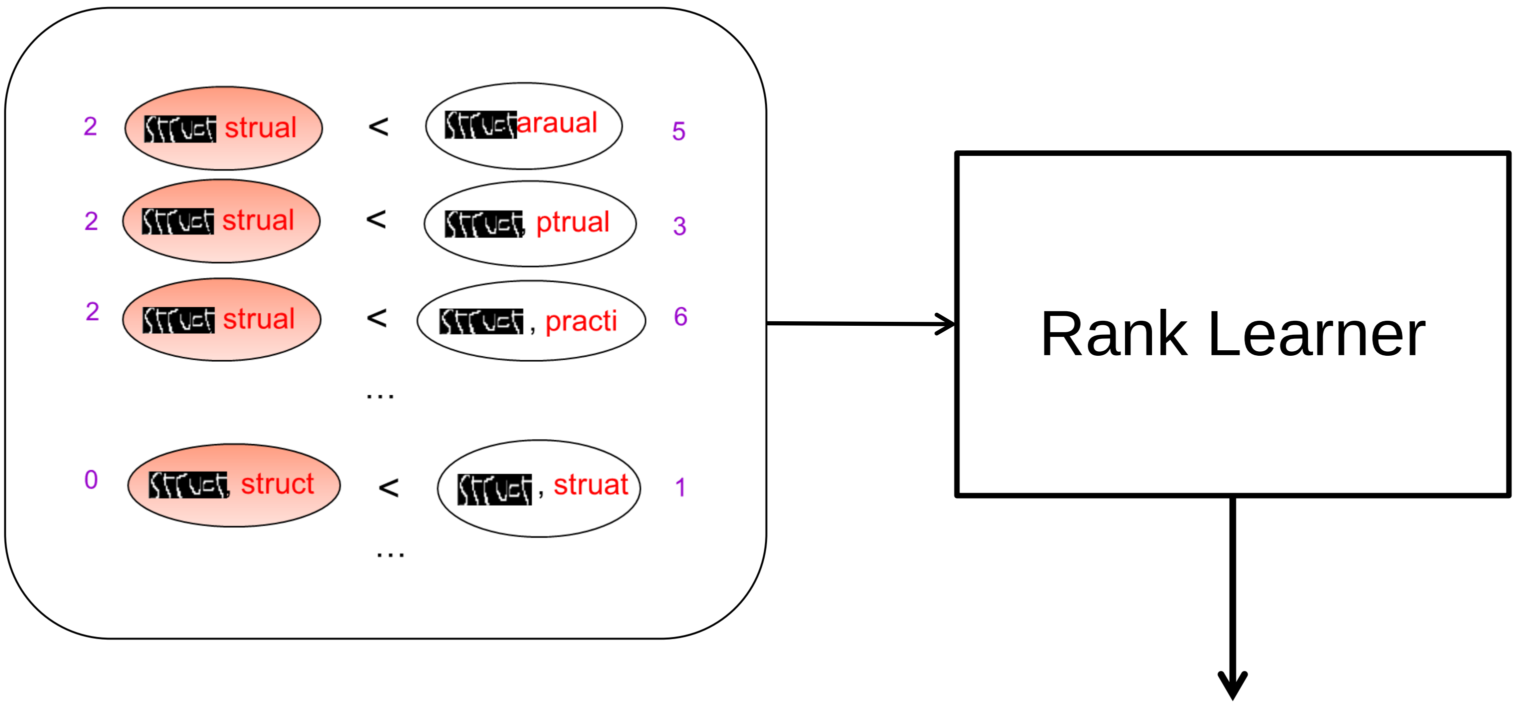


Greedy Search: Ranking examples



HC-Search: Heuristic Function Learning

Ranking examples



Heuristic function $\hat{H}$

Can prove generalization bounds on learned heuristic

[Doppa et al., 2012]

HC-Search: Learning

$$\epsilon = \epsilon_{H} + \epsilon_{C|H}$$

Overall loss

Generation loss
(Heuristic function)

Selection loss
(Cost function)

- **Key idea:** Greedy stage-wise minimization guided by the loss decomposition
  - **Step 1:** $\hat{H} = \arg min_{H \in \mathbf{H}}\ \epsilon_H$ (heuristic training)
  - **Step 2:** $\hat{C} = \arg min_{C \in \mathbf{C}}\ \epsilon_{C|\hat{H}}$ (cost function training)

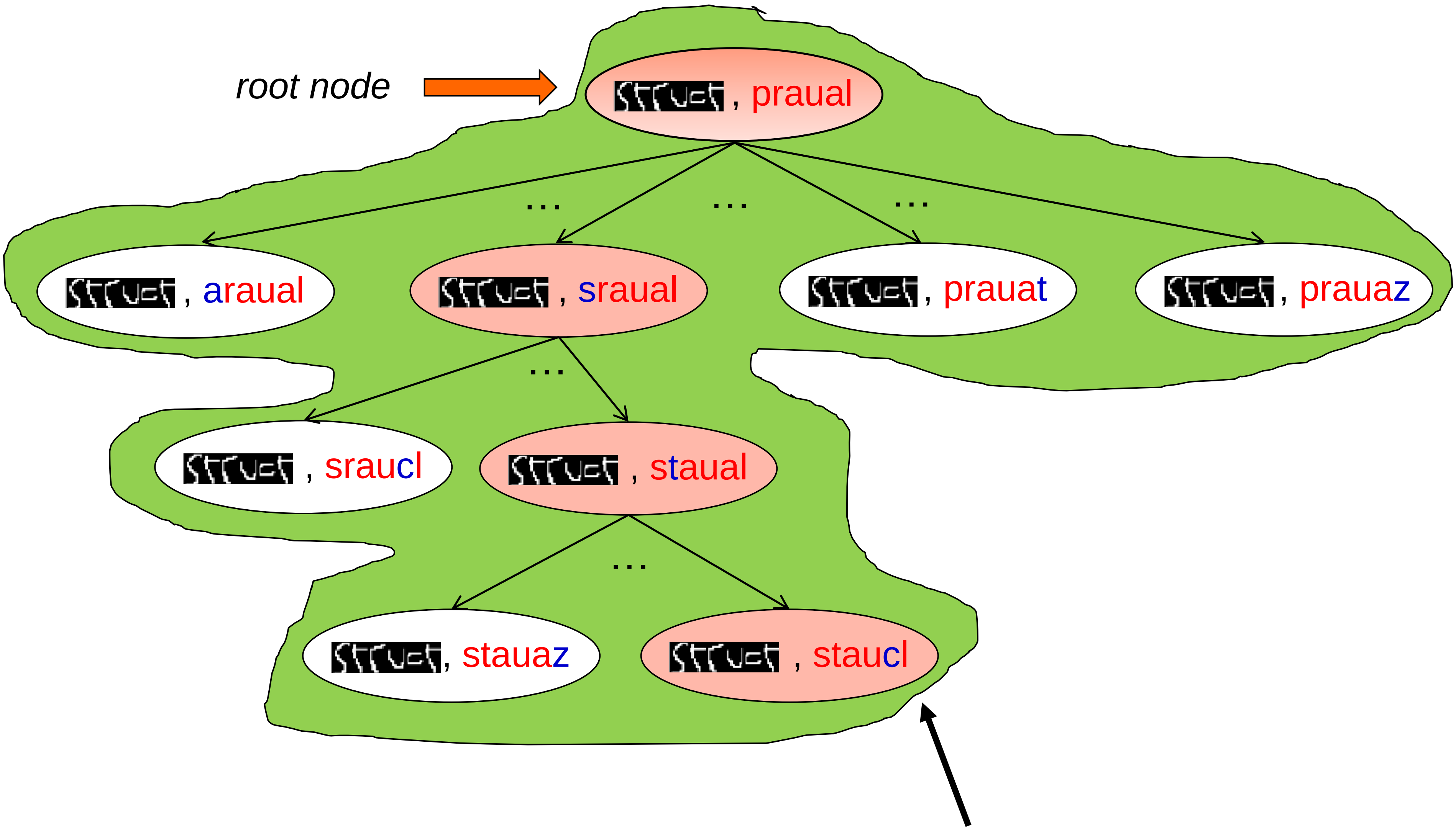
Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

HC-Search: Cost Function Learning

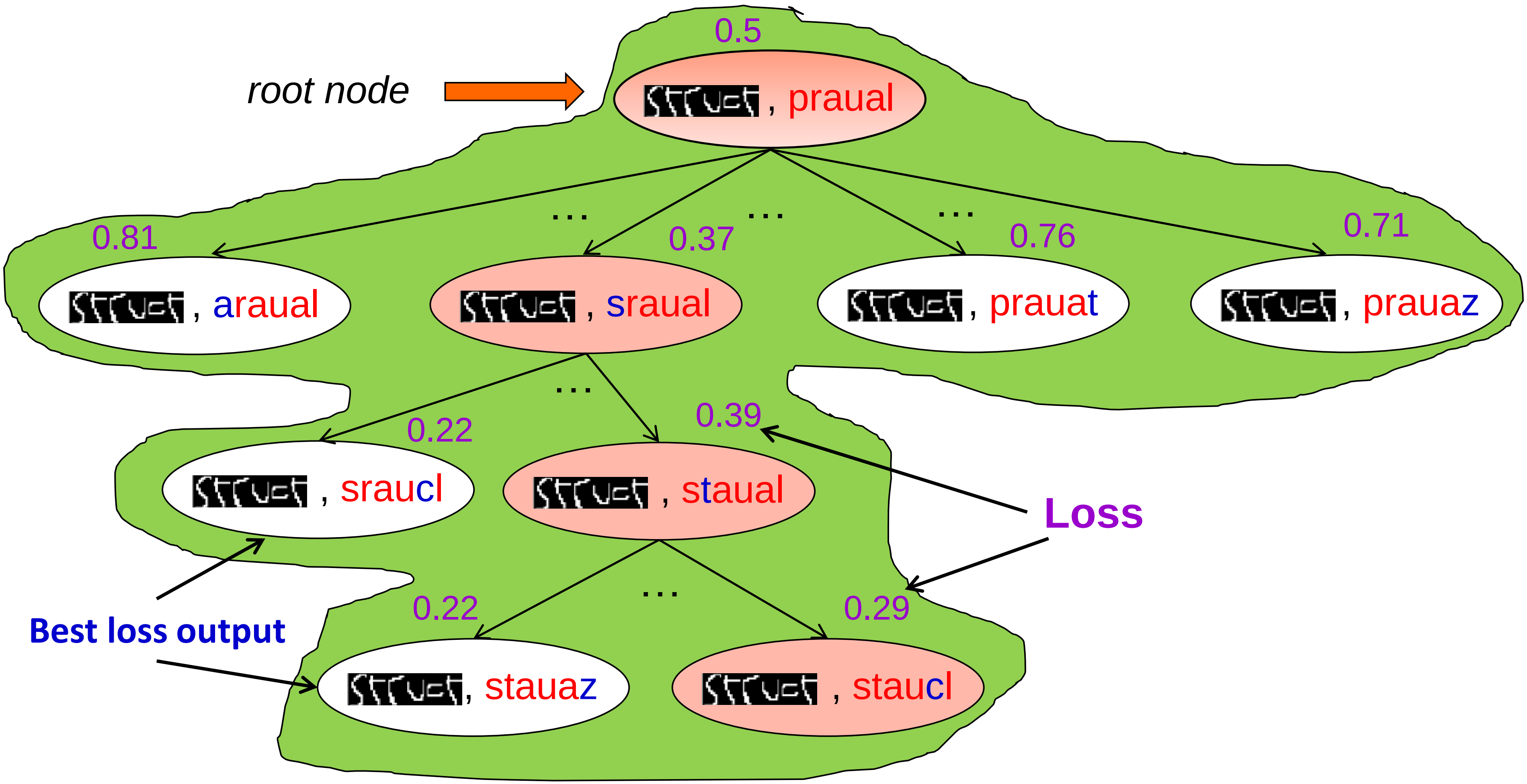
- **Learning Objective:**
  - Correctly score the outputs generated by the heuristic as per their losses

HC-Search: Cost function Learning



Set of all outputs generated by the heuristic $\hat{H}$

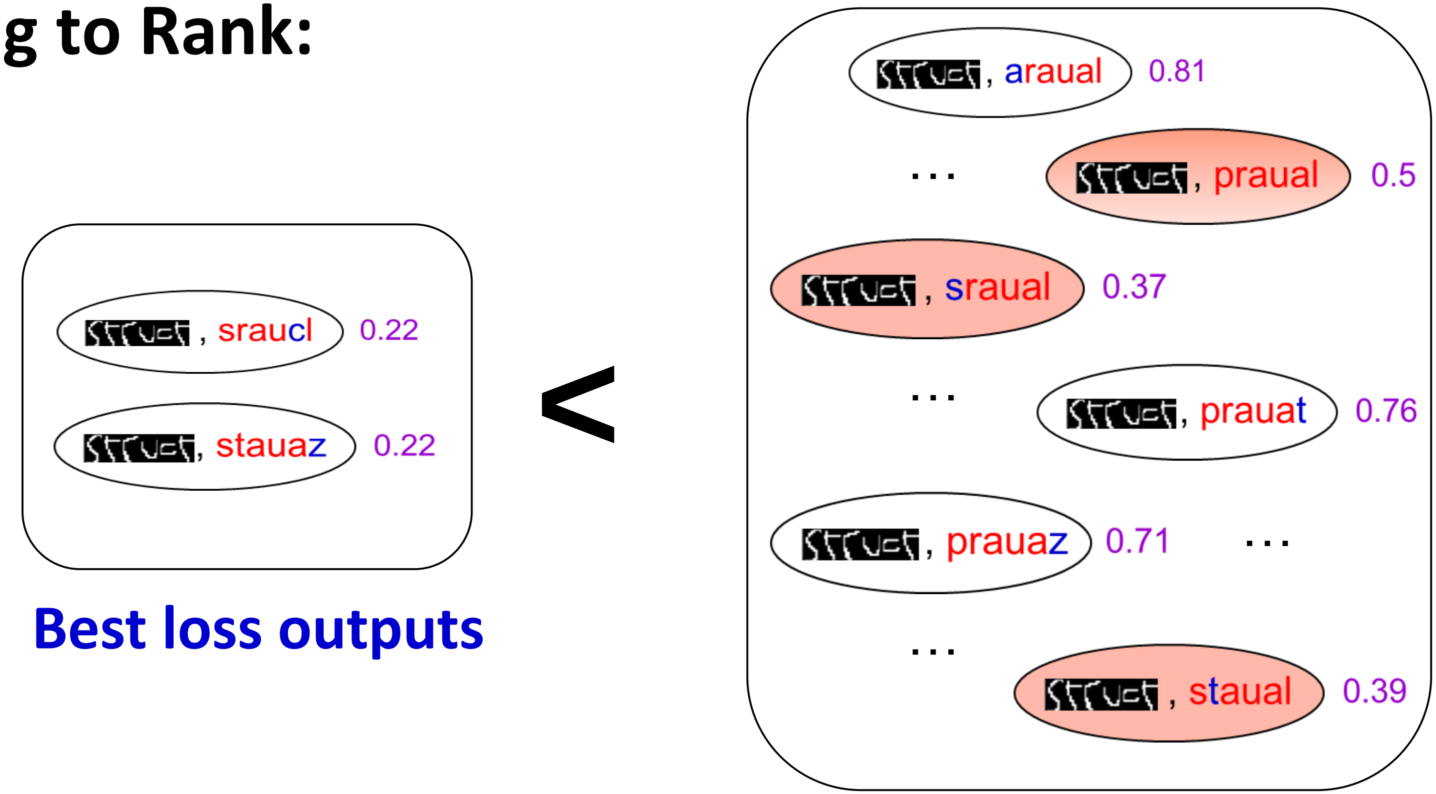
HC-Search: Cost function Learning



- **Key Idea:** Learn to rank the outputs generated by the learned heuristic function $\widehat{H}$ as per their losses

HC-Search: Cost function Learning

- **Learning to Rank:**



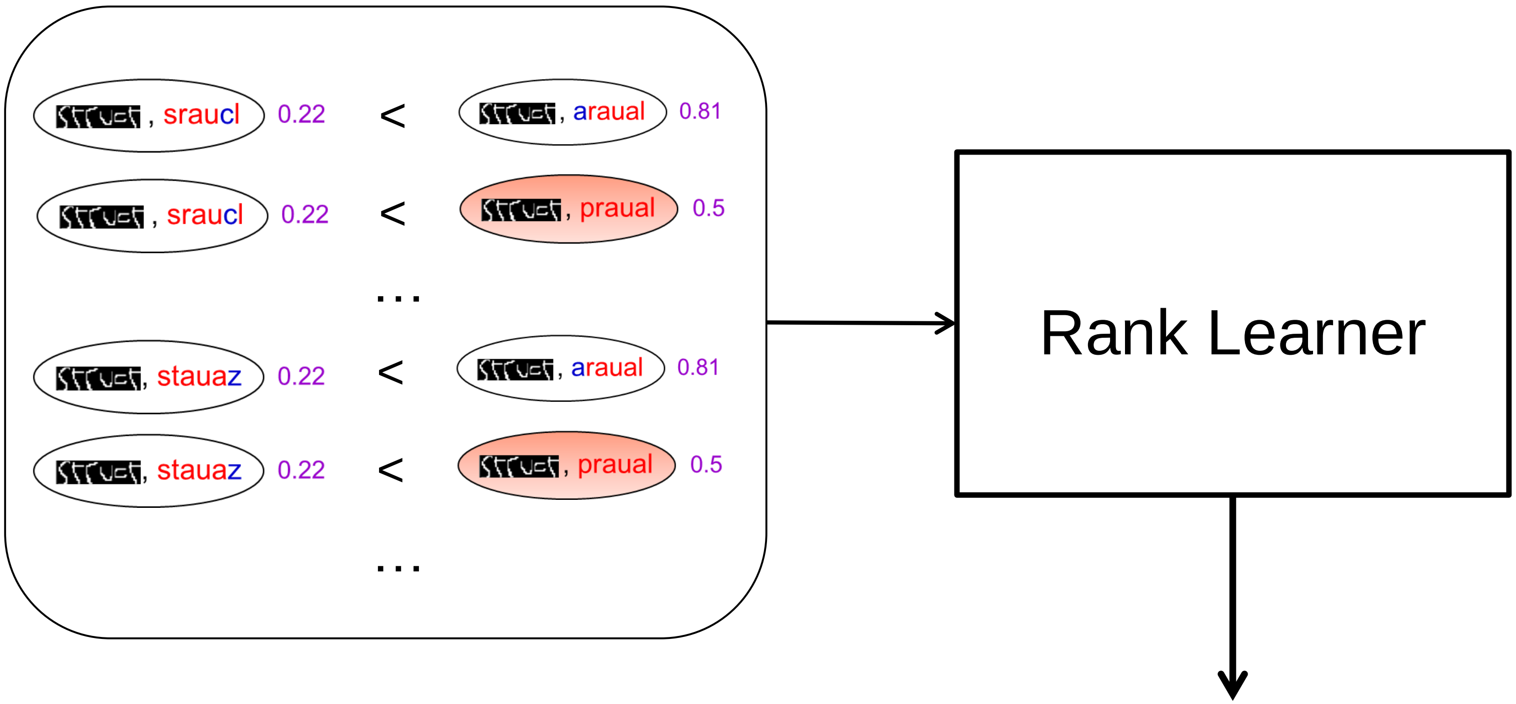
Best loss outputs

Non-best loss outputs

- Create a ranking example between every pair of outputs (y_{best}, y) such that: $C(x, y_{best}) < C(x, y)$

HC-Search: Cost function Learning

Ranking examples



Cost function $\widehat{C}$

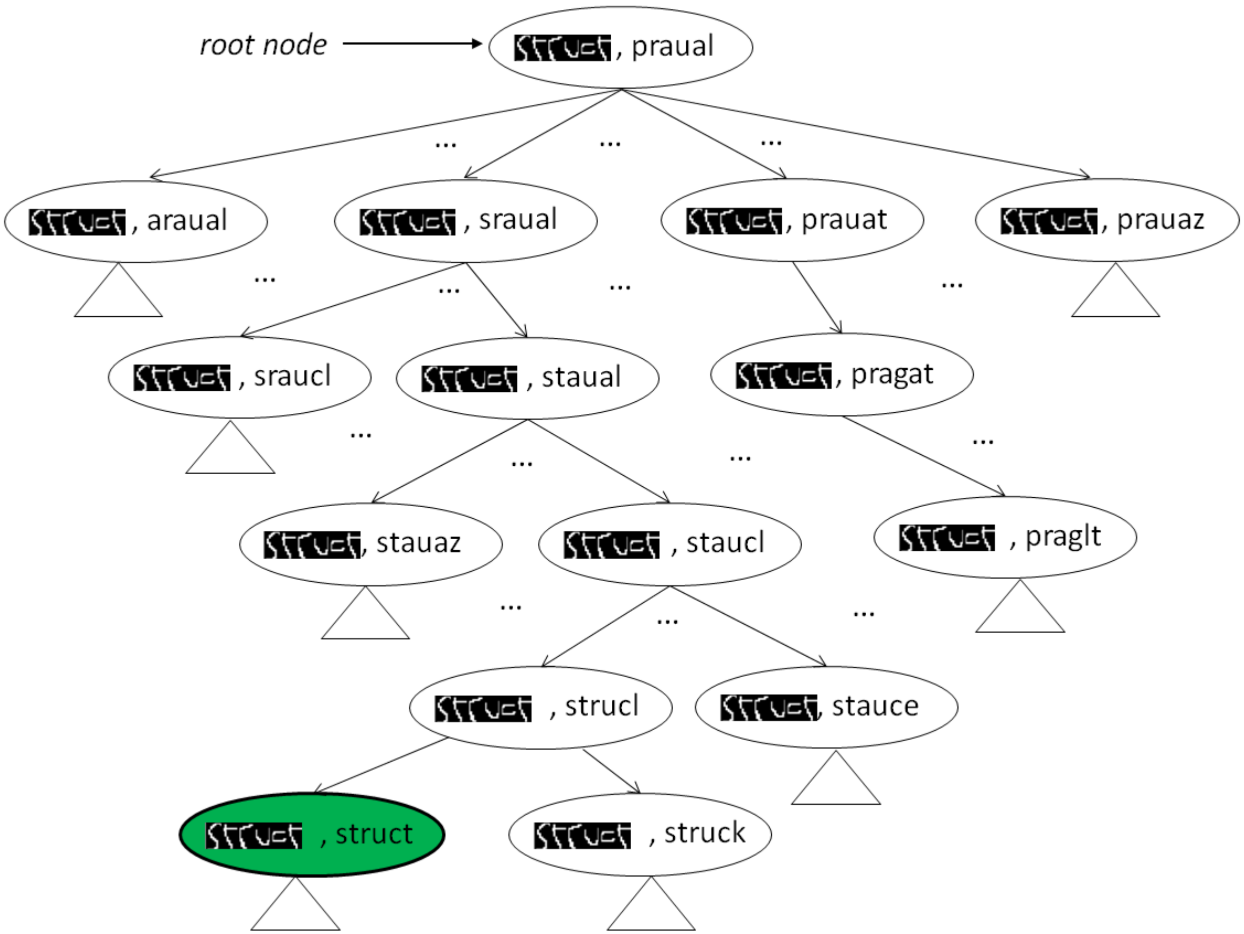
Can borrow generalization bounds from rank-learning literature
[Agarwal and Roth, 2005 & Agarwal and Niyogi, 2009]

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

HC-Search: Search Space Design

- **Objective:**
  - High-quality outputs can be located at small depth

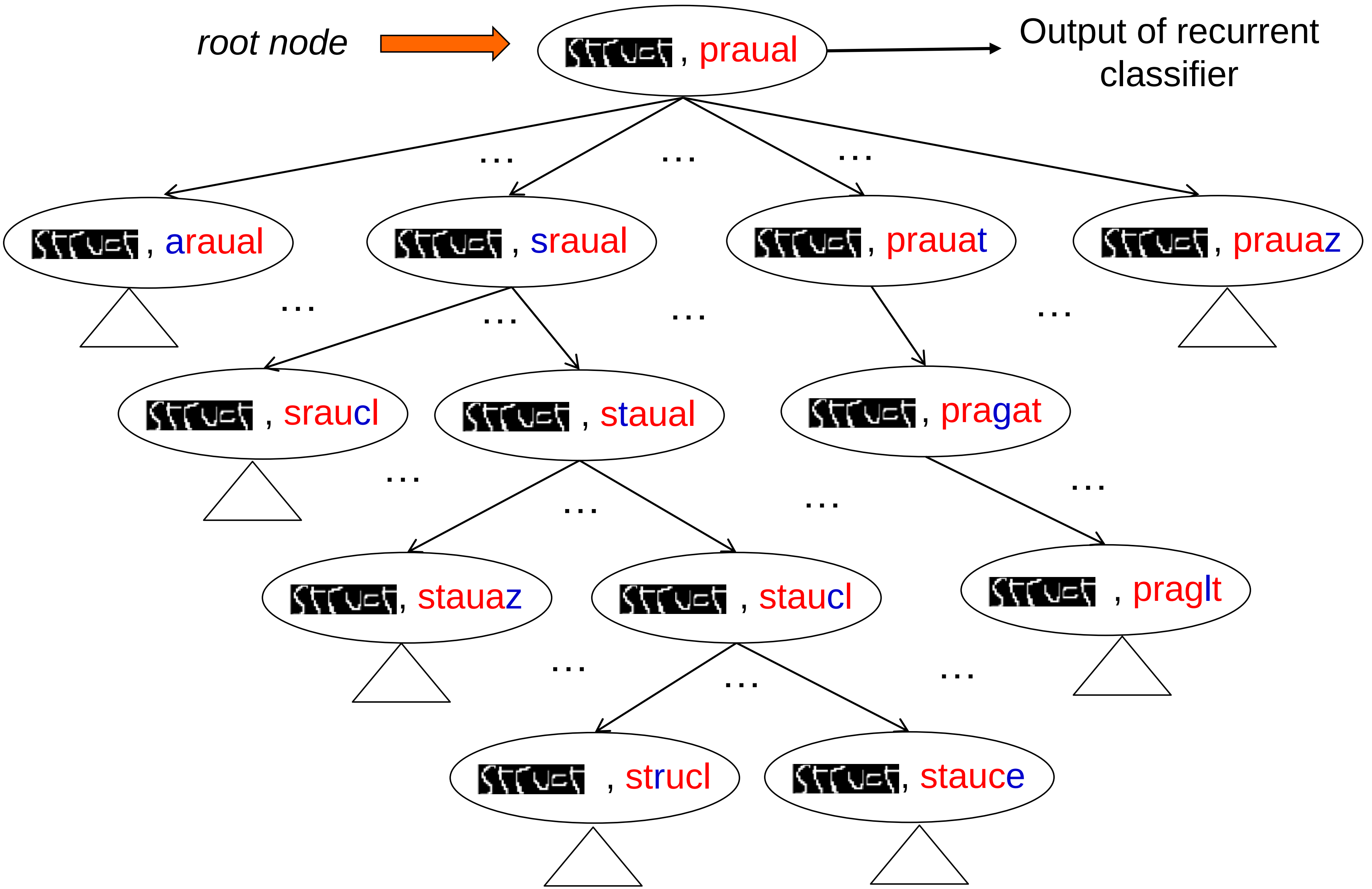


Target depth = 5

HC-Search: Search Space Design

- **Objective:**
  - High-quality outputs can be located at small depth
- **Solution #1:**
  - Flipbit Search Space [JMLR, 2014]
- **Solution #2:**
  - Limited Discrepancy Search (LDS) Space [JMLR, 2014]
  - Defined in terms of a greedy predictor or policy
- **Solution #3:**
  - Segmentation Search Space for computer vision tasks [CVPR, 2015]

Flip-bit Search Space



Multi-Label Prediction: Problem

Input

Output

0	computer
0	chair
1	sky
…	
1	water
1	sand
0	mountains
…	

Multi-Label Prediction: Problem

- Commonly arises in various domains
  - **Biology** – predict functional classes of a protein/gene
  - **Text** – predict email tags or document classes
  - …

Multi-Label Prediction

- Benchmark data

Dataset	Domain	#TR	#TS	#F	#L	$E[d]$
Scene	image	1211	1196	294	6	1.07
Emotions	music	391	202	72	6	1.86
Medical	text	333	645	1449	45	1.24
Genbase	biology	463	199	1185	27	1.25
Yeast	biology	1500	917	103	14	4.23
Enron	text	1123	579	1001	53	3.37
LLog	text	876	584	1004	75	1.18
Slashdot	text	2269	1513	1079	22	2.15

Multi-Label Prediction

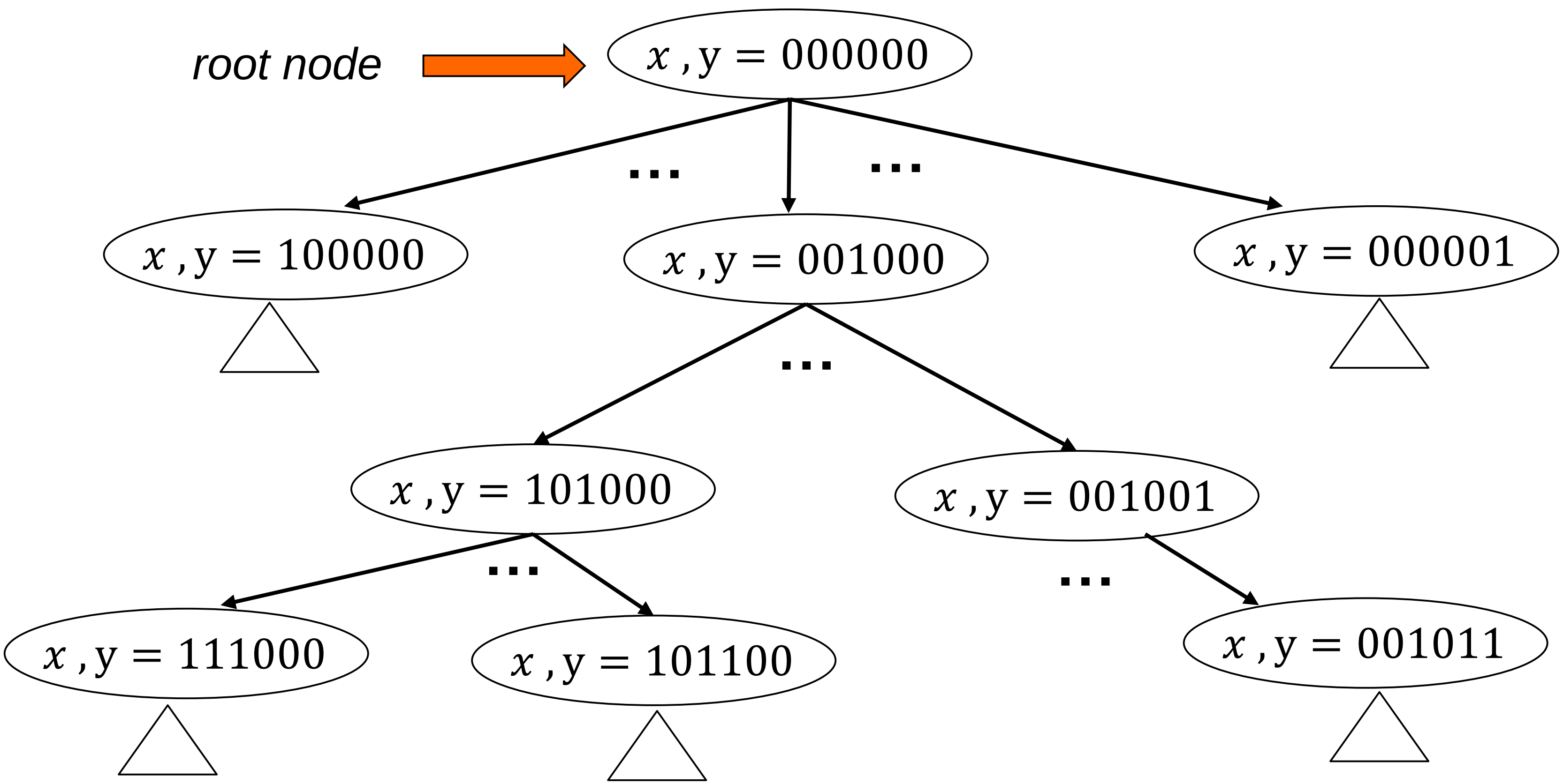
- Benchmark data

Dataset	Domain	#TR	#TS	#F	#L	$E[d]$
Scene	image	1211	1196	294	6	1.07
Emotions	music	391	202	72	6	1.86
Medical	text	333	645	1449	45	1.24
Genbase	biology	463	199	1185	27	1.25
Yeast	biology	1500	917	103	14	4.23
Enron	text	1123	579	1001	53	3.37
LLog	text	876	584	1004	75	1.18
Slashdot	text	2269	1513	1079	22	2.15

Label vectors are highly sparse

Multi-Label Prediction via HC-Search

- **HC-Search**
  - Exploit the sparsity property (Null vector + flip bits)



Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

Engineering Methodology

- **Select a time-bounded search architecture**
  - High-quality search space (e.g., LDS space or its variant)
  - Search procedure
  - Time bound
  - Effectiveness can be measured by performing LL-Search (loss function as both heuristic and cost function)
- **Training and Debugging**
  - Overall error = generation error (heuristic) + selection error (cost function)
  - Take necessary steps to improve the appropriate error guided by the decomposition

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Experiments and Results
- Engineering Methodology for applying HC-Search
- Relation to Alternate Methods

HC-Search vs. CRF/SSVM

- **Inference in CRF/SSVM**
  - Cost function needs to score exponential no. of outputs

$$\mathbf{F(x)} = \arg \min_{y \in Y(x)} C(x, y)$$

- **Inference in HC-Search**
  - Cost function needs to score only the outputs generated by the search procedure guided by heuristic H

$$\mathbf{F(x)} = \arg \min_{y \in Y_H(x)} C(x, y)$$

HC-Search vs. Re-Ranking Algorithms

- **Re-Ranking Approaches**
  - k-best list from a generative model

    Michael Collins: *Ranking Algorithms for Named Entity Extraction: Boosting and the Voted Perceptron*. ACL 2002: 489-496
  - Diverse M-best modes of a probabilistic model

    Payman Yadollahpour, Dhruv Batra, Gregory Shakhnarovich: Discriminative Re-ranking of Diverse Segmentations. CVPR 2013: 1923-1930
  - No guarantees on the quality of generated candidate set
- **HC-Search**
  - Candidate set is generated via generic search in high-quality search spaces guided by the learned heuristic
  - Minimal restrictions on the representation of heuristic
  - PAC guarantees on the quality of candidate set

HC-Search: A "Divide-and-Conquer" Solution

- **HC-Search is a "Divide-and-Conquer'' solution with procedural knowledge injected into it**
  - All components have clearly pre-defined roles
  - Every component is contributing towards the overall goal by making the role of other components easier

HC-Search: A “Divide-and-Conquer” Solution

- Every component is contributing towards the overall goal by making the role of other components easier

  - **LDS space** leverages greedy classifiers to reduce the target depth to make the heuristic learning easier

  - **Heuristic** tries to make the cost function learning easier by generating high-quality outputs with as little search as possible

Important References

- **Advances in cost function learning**
  - **Amortized inference and learning with ILP:**

Vivek Srikumar, Gourab Kundu, Dan Roth: On Amortizing Inference Cost for Structured Prediction. EMNLP 2012

Gourab Kundu, Vivek Srikumar, Dan Roth: Margin-based Decomposed Amortized Inference. ACL 2013

Kai-Wei Chang, Shyam Upadhyay, Gourab Kundu, Dan Roth: Structural Learning with Amortized Inference. AAAI 2015

  - **PAC theory for ILP inference:**

Ofer Meshi, Mehrdad Mahdavi, Adrian Weller, David Sontag: Train and Test Tightness of LP Relaxations in Structured Prediction. ICML 2016

  - **Decomposed learning:**

Rajhans Samdani, Dan Roth: Efficient Decomposed Learning for Structured Prediction. ICML 2012

  - **Structured prediction cascades:**

David J. Weiss, Benjamin Taskar: Structured Prediction Cascades. AISTATS 2010: 916-923

Important References

- **Classifier-based structured Prediction**
  - **Recurrent classifier:**

Thomas G. Dietterich, Hermann Hild, Ghulum Bakiri: A Comparison of ID3 and Backpropagation for English Text-to-Speech Mapping. Machine Learning 18(1): 51-80 (1995)

  - **PAC Results and Error Propagation:**

Roni Khardon: Learning to Take Actions. Machine Learning 35(1): 57-90 (1999)

Alan Fern, Sung Wook Yoon, Robert Givan: Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. J. Artif. Intell. Res. (JAIR) 25: 75-118 (2006)

Umar Syed, Robert E. Schapire: A Reduction from Apprenticeship Learning to Classification. NIPS 2010

Stéphane Ross, Drew Bagnell: Efficient Reductions for Imitation Learning. AISTATS 2010: 661-668

  - **Advanced Imitation Learning Algorithms:**

SEARN: Hal Daumé III, John Langford, Daniel Marcu: Search-based structured prediction. Machine Learning 75(3): 297-325 (2009)

DAgger: Stéphane Ross, Geoffrey J. Gordon, Drew Bagnell: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. AISTATS 2011: 627-635

AggreVaTe: Stéphane Ross, J. Andrew Bagnell: Reinforcement and Imitation Learning via Interactive No-Regret Learning. CoRR abs/1406.5979 (2014)

LOLS: Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, John Langford: Learning to Search Better than Your Teacher. ICML 2015: 2058-2066

Yuehua Xu, Alan Fern, Sung Wook Yoon: Iterative Learning of Weighted Rule Sets for Greedy Search. ICAPS 2010: 201-208

Alan Fern, Sung Wook Yoon, Robert Givan: Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. J. Artif. Intell. Res. (JAIR) 25: 75-118 (2006)

Important References

- **Easy-first approach for structured Prediction**

Yoav Goldberg, Michael Elhadad: An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing. HLT-NAACL 2010

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nate Chambers, Mihai Surdeanu, Dan Jurafsky, Christopher D. Manning: A Multi-Pass Sieve for Coreference Resolution. EMNLP 2010

Lev-Arie Ratinov, Dan Roth: Learning-based Multi-Sieve Co-reference Resolution with Knowledge. EMNLP-CoNLL 2012

Veselin Stoyanov, Jason Eisner: Easy-first Coreference Resolution. COLING 2012

Jun Xie, Chao Ma, Janardhan Rao Doppa, Prashanth Mannem, Xiaoli Z. Fern, Thomas G. Dietterich, Prasad Tadepalli: Learning Greedy Policies for the Easy-First Framework. AAAI 2015

- **Learning Beam search heuristics for structured prediction**

Michael Collins, Brian Roark: Incremental Parsing with the Perceptron Algorithm. ACL 2004

Hal Daumé III, Daniel Marcu: Learning as search optimization: approximate large margin methods for structured prediction. ICML 2005

Yuehua Xu, Alan Fern, Sung Wook Yoon: Learning Linear Ranking Functions for Beam Search with Application to Planning. Journal of Machine Learning Research 10: 1571-1610 (2009)

Liang Huang, Suphan Fayong, Yang Guo: Structured Perceptron with Inexact Search. HLT-NAACL 2012

Important References

- **HC-Search Framework for structured Prediction**

Janardhan Rao Doppa, Alan Fern, Prasad Tadepalli: HC-Search: A Learning Framework for Search-based Structured Prediction. J. Artif. Intell. Res. (JAIR) 50: 369-407 (2014)

Janardhan Rao Doppa, Alan Fern, Prasad Tadepalli: Structured prediction via output space search. Journal of Machine Learning Research 15(1): 1317-1350 (2014)

Janardhan Rao Doppa, Jun Yu, Chao Ma, Alan Fern, Prasad Tadepalli: HC-Search for Multi-Label Prediction: An Empirical Study. AAAI 2014

Michael Lam, Janardhan Rao Doppa, Sinisa Todorovic, Thomas G. Dietterich: HC-Search for structured prediction in computer vision. CVPR 2015

Part 6: Integrating Deep Learning and Structured Prediction

Liping Liu

Tufts University

Motivation

- **Deep models as non-linear functions**
  - mapping from the input to the output
  - non-linear
  - need fast training
- **How about replacing functions with deep models?**
  - potential function for CRF
  - search function for search based predicting models
  - attention model for structured prediction

Motivation

- **Deep models as non-linear functions**
  - mapping from the input to the output
  - non-linear
  - need fast training
- **How about replacing functions with deep models?**
  - potential function for CRF
  - search function for search based predicting models
  - attention model for structured prediction

Conditional Random Field (CRF)

- The basic form
  - $P(y \mid x;\ w) = \frac{1}{Z(w)} \exp(\sum_k w_k \cdot \phi_k(x, y))$

    partition function $Z(w) = \sum_{y \in \mathcal{Y}} \exp(\sum_k w_k \cdot \phi_k(x, y))$

  - The function $\phi_k(x, y) = \phi_k(x, y_{i_k})$often defines the potential of a single label or a pair of labels

CRF extensions with Deep models

- Deep structured models [Chen et al. ICML 2015]
  - Replace linear potential $w_k \phi_k(\cdot)$ with a deep function $f_k(x, y; w)$ to extract information from complex object x
- Structured Prediction Energy Network (SPEN) [Belanger et al. ICML 2016, 2017]
  - Replace $\sum_k w_k \cdot \phi_k(x, y)$ with a deep function $F(x, y; w)$, so $P(y \mid x;\ w) = \frac{1}{Z(x,w)} \exp\big(F(x, y; w)\big)$
- Deep Value Network (DVN) [Gygli et al. ICML 2017]
  - Learn a deep model $v(x, y; w)$ to fit the negative loss (DVN)

Deep Structured Models

- Deep structured models [Chen et al. ICML 2015]
  - The potential $F(x, y; w)$ is decomposable by nodes or node pairs,

$$F(x, y; w) = \sum_k f_k(x, y_{i_k}; w)$$

  - $f_k(x, y; w)$ is still a single or a pairwise potential

- DSM approximates the partition function with loopy belief propagation

$$\log Z(x, w) = \max_p \; E_p\left[F\left(x, y_{i_k}; w\right)\right] + H[p]$$

(Treat w as a constant here)

- Approximate marginal of dist p by local beliefs

Structured Prediction Energy Network

- SPEN [Belanger et al. ICML 2016, 2017] allows high order label interactions through non-linear transformation of label vectors

$$\min_{y} E_x(y) \quad s.t. \; y \in \{0, 1\}^L$$

- Training in the same way as structured SVM
  - Minimize the hinge loss

$$\min_{w} \max_{y} \left[\Delta(y_i, y) - E_{x_i}(y) + E_{x_i}(y_i)\right]_+$$

  - Inner optimization problem is solved by LP relaxation, relaxing the space of discrete labels to a continuous one

Deep Value Networks (DVN)

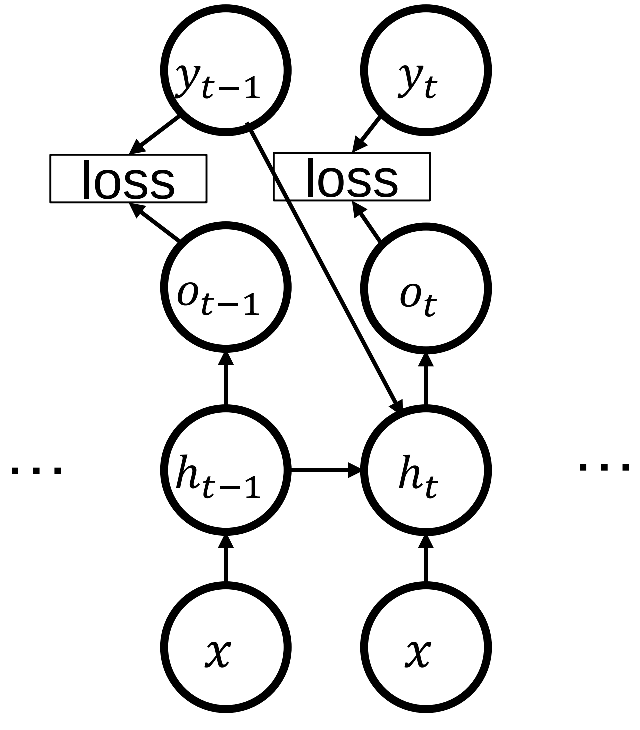
- DVN [Gygli et al. ICML 2017] fit negative loss values
  - Train a network $v(x, y; \theta)$ such that
$$v(x, y; \theta) \approx -loss(y, y^*)$$
- Trained with samples $(x, y', -loss(y', y^*))$, with y' being
  - Training label y^*
  - Inference result $\hat{y} = \arg\max v(x, y; \theta)$
  - Random samples
  - Adversarial samples
- Inference is done by optimization of y in the continuous space

Motivation

- **Deep models as non-linear functions**
  - mapping from the input to the output
  - non-linear
  - need fast training
- **How about replacing functions with deep models?**
  - potential function for CRF
  - search function for search based predicting models
  - attention model for structured prediction

RNN for Structured Prediction

- RNN can output predictions with structures
  - Input x, y_{t-1}
  - Output y_t at time t



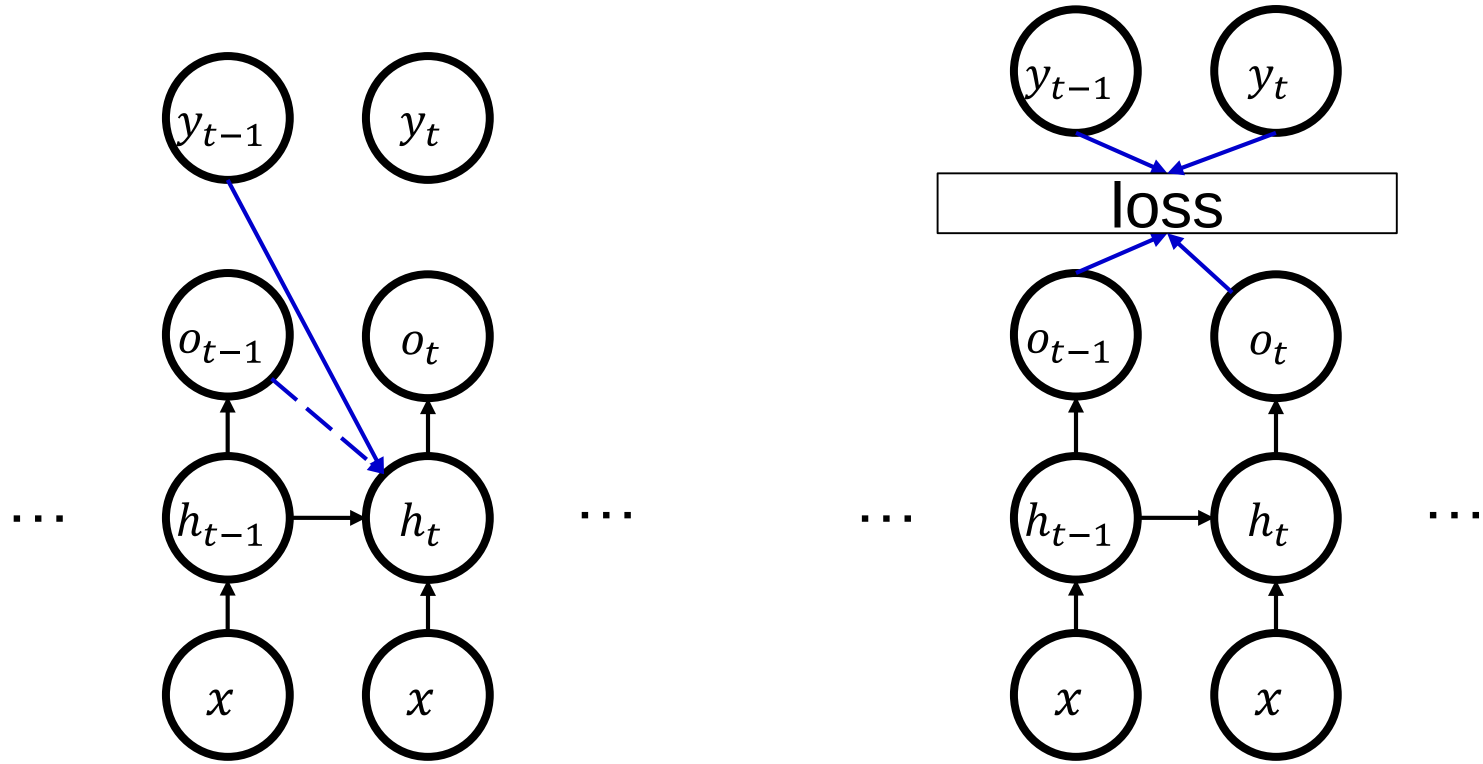
RNN for Structured Prediction

- RNN can output predictions with structures
  - Input x, y_{t-1}
  - Output y_t at time t

- Considerations for structured prediction
  - How to avoid exposure bias (i.e. teacher forcing makes training and testing different)?
  - How to include loss function in training?

RNN for Structured Prediction

- Two issues
  - Exposure bias (teacher enforcing)
  - Loss-evaluation mismatch

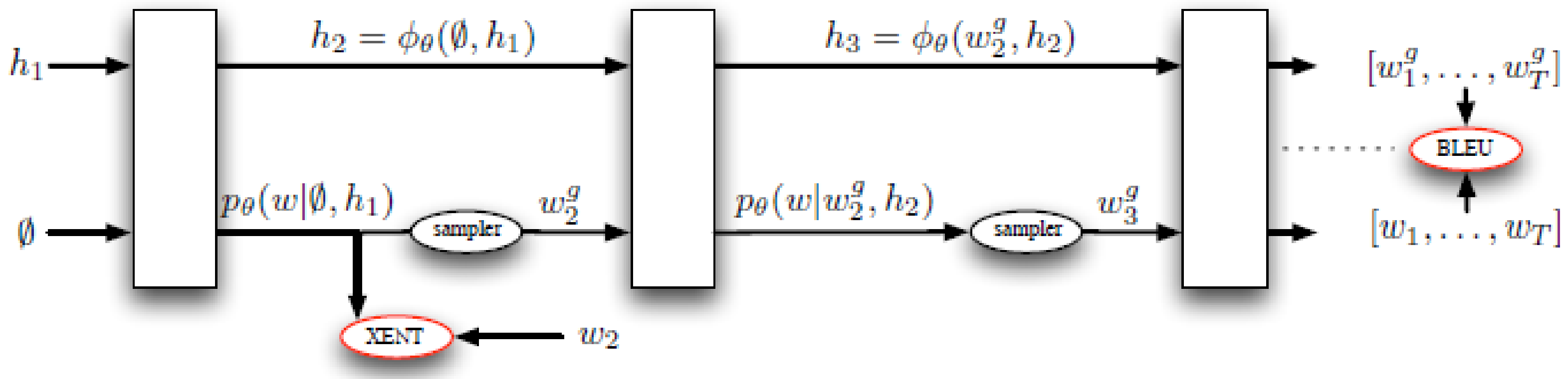


Structured Prediction as an RL Problem

- Formulation as reinforcement learning
  - (x, o_{t-1}, h_t) as a state
  - Negative loss as reward
    - Reward is given at the last step
    - Zero reward for intermediate steps
  - Output y_t at each step as action
  - RNN as a policy
- Tackle two issues together
  - Minimize loss by maximize reward
  - Learning naturally corrects exposure bias

Training RNN with policy gradient

- Learn RNN with MIXER [Ranzato, ICLR 2016]
  - First time steps are trained by maximize likelihood
  - The last few steps are trained by REINFORCE
    - REINFORCE is a one policy gradient algorithm
    - Use a single sample from RNN to estimated expected reward



Actor-Critic Algorithm for RNN Learning

- An actor-critic algorithm for sequence prediction [Bahdanau et al. ICLR 2017]
  - Actor: RNN(θ)
  - Critic: another network to estimated Q function
- Learning Procedure
  - Update actor/RNN with gradient,

$$\frac{dV}{d\theta} = E_{y\sim RNN(\theta)}\left[\sum_t \sum_{y'_t} \frac{d\, p(y'_t|y_{t-1}, h_t\,)}{d\theta} \hat{Q}(y'_t, y_{1:t-1}))\right]$$

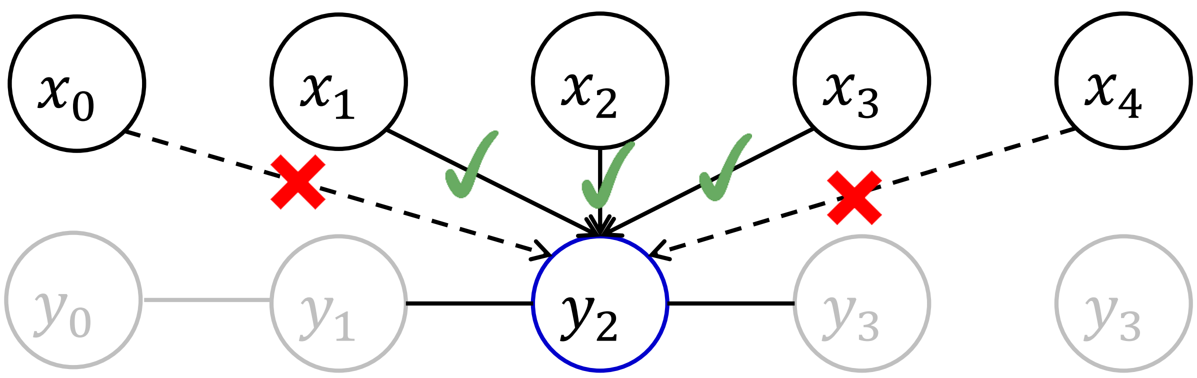
  - Update critic/estimation of $\hat{Q}$

Motivation

- **Deep models as non-linear functions**
  - mapping from the input to the output
  - non-linear
  - need fast training
- **How about replacing functions with deep models?**
  - potential function for CRF
  - search function for search based predicting models
  - attention model for structured prediction

Structure design of structured prediction

- A single label does not need all inputs
- Let the model to decide which to use



Attention Model

- Attention model for image captioning [Xu et al. ICML 2015]
  - RNN model for image captioning
  - Output: a sequence of words
  - Input: feature vectors extracted from selected image locations at different time steps

A woman is throwing a frisbee in a park.

Image is from the paper [Xu et al., ICML 2015]

Structured Attention Model

- Assume attention sequence has structures
  - Define attention sequence as CRF

$$p(attention\ z \mid output\ y_t, input\ x) = softmax(\theta_c(z_c))$$

- End-to-end training
  - Gradient calculation is propagated through the inference procedure

Structured Prediction with Deep Models – the Trend

- Less hand crafted model design
  - Function design
  - Structure design
- More Utilization of existing network structures
  - Need to consider the propagation of gradients

Structured Prediction with Deep Models – Summary

- Large data amount calls for flexible models that support fast training (*aka* deep models)
- Batch training and stochastic gradient are important ingredients – as in other deep learning models
- Slow inference methods become less favored
- Disclaimer:
  - Only a small portion of the recent literature is covered here due to time limit.
  - Many more papers worth reading.

Important References

- ## Classifier-based structured Prediction

CRF & Deep models :

Charles Sutton and Andrew McCallum. An Introduction to Conditional Random Fields. Foundations and Trends in Machine Learning: Vol. 4: No. 4, pp 267-373 (2012).

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. ICLR (2015).

Liang-Chieh Chen, Alexander G. Schwing, Alan L. Yuille, and Raquel Urtasun. Learning Deep Structured Models. ICML (2015).

David Belanger and Andrew McCallum. Structured Prediction Energy Networks. ICML (2016).

Michael Gygli, Mohammad Norouzi, and Anelia Angelova. Deep Value Networks Learn to Evaluate and Iteratively Refine Structured Outputs. ICML (2017).

Rahul G. Krishnan, Uri Shalit, and David Sontag. Structured Inference Networks for Nonlinear State Space Models. AAAI (2017).

RNN

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press (2016).

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. SEARNN: Training RNNs with global-local losses. ICML Workshop (2017).

Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba . Sequence Level Training with Recurrent Neural Networks. ICLR (2016).

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An Actor-Critic Algorithm for Sequence Prediction. ICLR (2017).

Sam Wiseman and Alexander M. Rush. Sequence-to-Sequence Learning as Beam-Search Optimization. EMNLP (2016).

Important References

- **Classifier-based structured Prediction**
  - **Attention models:**

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML (2015).

Yoon Kim, Carl Denton, Luong Hoang and Alexander M. Rush. Structured Attention Networks. ICLR (2017).

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing Multimedia Content using Attention-based Encoder-Decoder Networks. In IEEE Transactions on Multimedia, 2015.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In Proceedings of NIPS, 2015.

Multi-Task Structured Prediction

Chao Ma

Oregon State University

Entity Analysis in Language Processing

Many NLP tasks process mentions of entities – things, people, organizations, etc.

- **Named Entity Recognition**
- **Coreference Resolution**
- **Entity Linking**
- Semantic Role Labeling
- Entity Relation Extraction

……

We focus on three of them in this work

Coreference Resolution

***i* = 1** ***i* = 2**

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

Coreference Resolution

$\boldsymbol{i} = \mathbf{1}$ $\boldsymbol{i} = \mathbf{2}$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

Coreference:

$y_{\mathrm{i}} = \{1, 2 \ldots i\}$

co-referent link

co-referent link

$\boldsymbol{y}_{\text{coref}} = (\text{Columbia}, \text{Columbia University}, \ldots)$

Coreference Resolution

$i = 1$ **$i = 2$**

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link

Columbia

co-referent link

Columbia University

Coreference:

$y_{\mathrm{i}} = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

m_1	m_2	m_3	m_4	m_5	m_6	m_7

$$\boldsymbol{y}_{\text{coref}} = (\; ? \; , \; ? \; , \; ? \; , \; ? \; , \; ? \; , \; ? \; , \; ? \;)$$

Coreference Resolution

$\boldsymbol{i} = \mathbf{1}$ $\boldsymbol{i} = \mathbf{2}$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link

Columbia

co-referent link

Columbia University

Coreference:

$y_{\mathrm{i}} = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

m_1 m_2 m_3 m_4 m_5 m_6 m_7

$\boldsymbol{y}_{\mathrm{coref}} = (\; 1 \; , \; ? \; , \; ? \; , \; ? \; , \; ? \; , \; ? \; , \; ? \;)$

Coreference Resolution

$i = 1$ $i = 2$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link

Columbia

co-referent link

Columbia University

Coreference:

$y_i = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

m_1 m_2 m_3 m_4 m_5 m_6 m_7

$y_{coref} = (\ 1\ ,\ 1\ ,\ ?\ ,\ ?\ ,\ ?\ ,\ ?\ ,\ ?\ )$

Coreference Resolution

***i* = 1** **i = 2**

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link

co-referent link

Columbia

Columbia University

Coreference:

$y_{\mathrm{i}} = \{1, 2 \dots i\}$

Left-linking Tree formulation for coreference resolution:

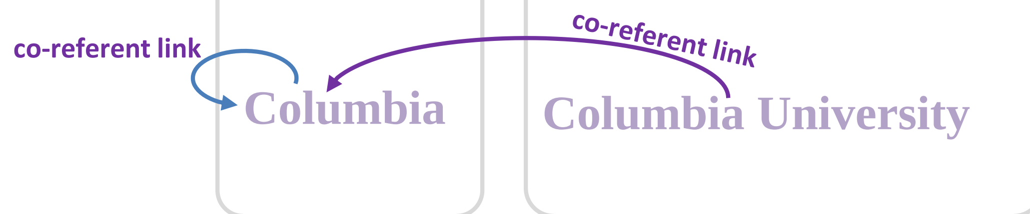
m_1 m_2 m_3 m_4 m_5 m_6 m_7

$y_{\mathrm{coref}} = (\; 1 \; , \; 1 \; , \; 2 \; , \; ? \; , \; ? \; , \; ? \; , \; ? \;)$

Coreference Resolution

$i = 1$ $i = 2$

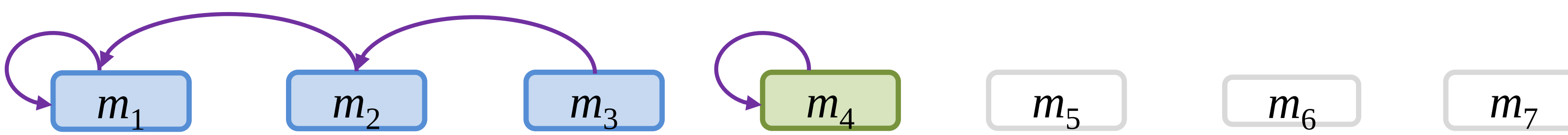
He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…



Coreference:

$y_i = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:



$\boldsymbol{y}_{\text{coref}} = (1, 1, 2, 4, ?, ?, ?)$

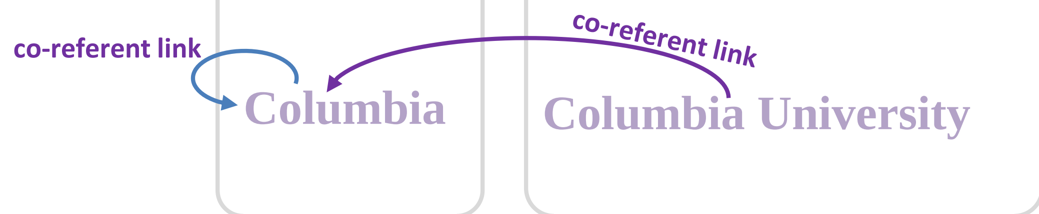
Coreference Resolution

i = 1 i = 2

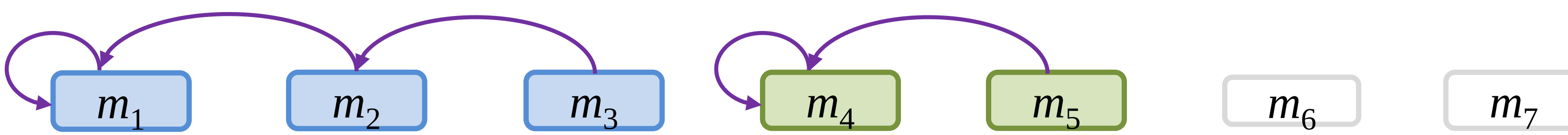
He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

Coreference:

$y_i = \{1, 2 \ldots i\}$



Left-linking Tree formulation for coreference resolution:

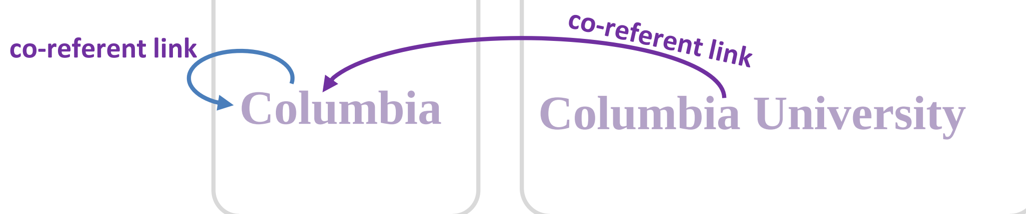


$y_{\text{coref}} = ($ 1 , 1 , 2 , 4 , 5 , ? , ? $)$

Coreference Resolution

$i = 1$ $i = 2$

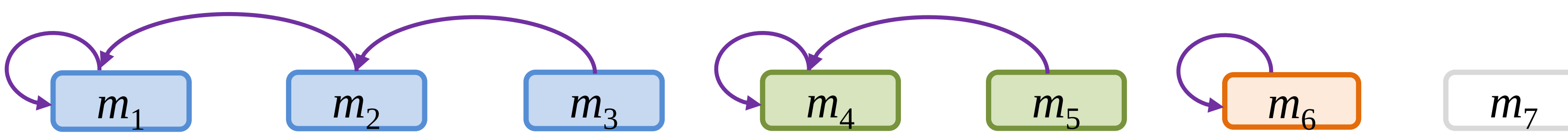
He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…



Coreference:

$y_i = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

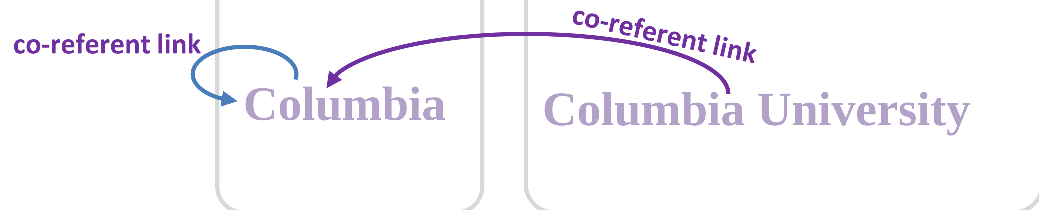


$$\boldsymbol{y}_{\text{coref}} = (\ 1,\ 1,\ 2,\ 4,\ 5,\ 6,\ ?\ )$$

Coreference Resolution

$i = 1$ $\quad$ $i = 2$

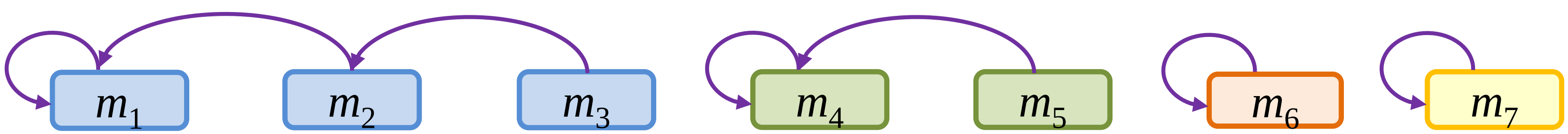
He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…



Coreference:

$y_i = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:

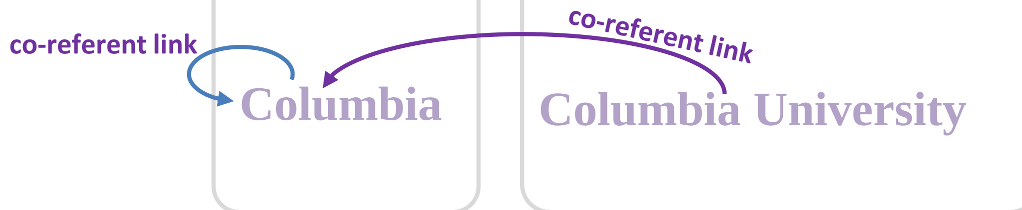


$$\boldsymbol{y}_{\text{coref}} = (\ 1\ ,\ 1\ ,\ 2\ ,\ 4\ ,\ 5\ ,\ 6\ ,\ 7\ )$$

Coreference Resolution

$\boldsymbol{i} = \mathbf{1}$ $\boldsymbol{i} = \mathbf{2}$

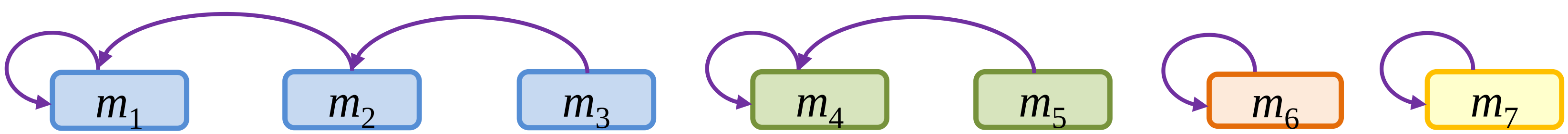
He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…



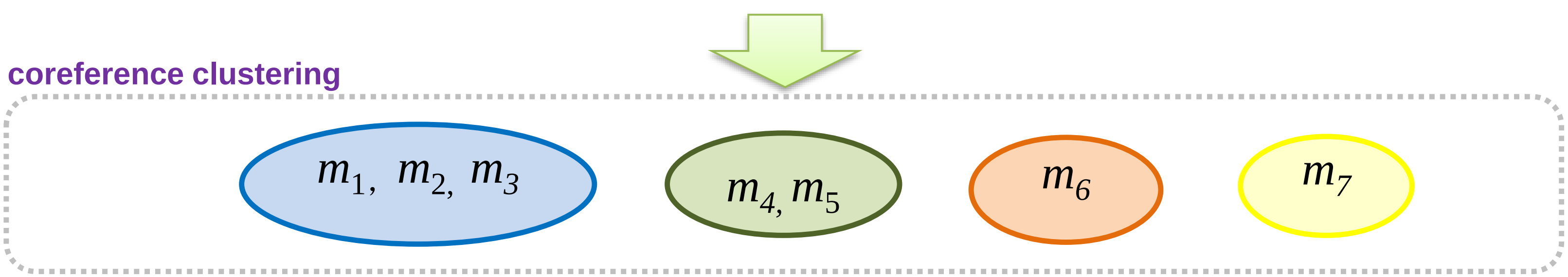
Coreference:

$y_{\mathrm{i}} = \{1, 2 \ldots i\}$

Left-linking Tree formulation for coreference resolution:



$\boldsymbol{y}_{\mathrm{coref}} = (\ 1,\ 1,\ 2,\ 4,\ 5,\ 6,\ 7\ )$



Coreference Resolution

$\boldsymbol{i} = \mathbf{1}$ $\boldsymbol{i} = \mathbf{2}$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

Coreference:

$y_{\mathrm{i}} = \{1, 2 \ldots i\}$

co-referent link

co-referent link

$\boldsymbol{y}_{\text{coref}} =$ (**Columbia** , **Columbia University** , …)

Named Entity Recognition

$i = 1$ $\quad$ $i = 2$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link $\quad$ co-referent link

Coreference: $\quad y_{\text{coref}} = ($ Columbia , $\quad$ Columbia University , $\quad \ldots)$

$y_i = \{1, 2 \ldots i\}$

Named Entity Recognition : $\quad y_{\text{ner}} = ($ ORG, $\quad$ ORG, $\quad \ldots)$

$y_i = \{$ORG, PER, GPE, LOC, FAC, VEL, WEA$\}$

Entity Linking

$i = 1$ $i = 2$

He left [**Columbia**] in 1983 with a BA degree, ... after graduating from [**Columbia University**], he worked as a community organizer in Chicago…

co-referent link co-referent link

Coreference: $y_{\text{coref}} =$ (Columbia , Columbia University , …)

$y_i = \{1, 2 \ldots i\}$

Named Entity Recognition : $y_{\text{ner}} =$ (ORG, ORG, …)

$y_i = \{$ORG, PER, GPE, LOC, FAC, VEL, WEA$\}$

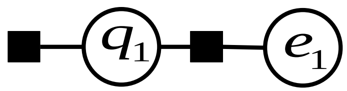
Entity Linking: $y_{\text{link}} =$ (https://en.wikipedia.org/wiki/Columbia_University , https://en.wikipedia.org/wiki/Columbia_University , …)

$y_i = \{$
https://en.wikipedia.org/wiki/Columbia_University,
https://en.wikipedia.org/wiki/Columbia_District,
https://en.wikipedia.org/wiki/Columbia,_British_Columbia,
https://en.wikipedia.org/wiki/Columbia_College,_Columbia_University,
…
$\}$

Graphic Model: Joint Entity Linking, Typing, and Coreference Task [Greg Durrett and Dan Klein. TACL 2014]

Insolate models

Entity Linking



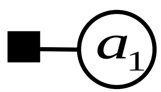
q **is a key word from the mention to query the KB to get an ranked candidate list. Then *e* is the best linking output from the list.**

m = "Oregon State University"

q = "Oregon State" — KB → { Oregon_State, Oregon_City, University_of_Oregon, OSU }

e = OSU (id = EL34233)

Coreference



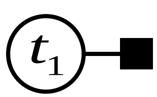
The value of coreference variable indicates the index of its coreferent antecedent mention.

$m(i)$ = "Oregon State University"

$m(j)$ = "OSU located at Corvallis"

$a(j) = i$

NER



Each mention has a corresponding entity typing tag.

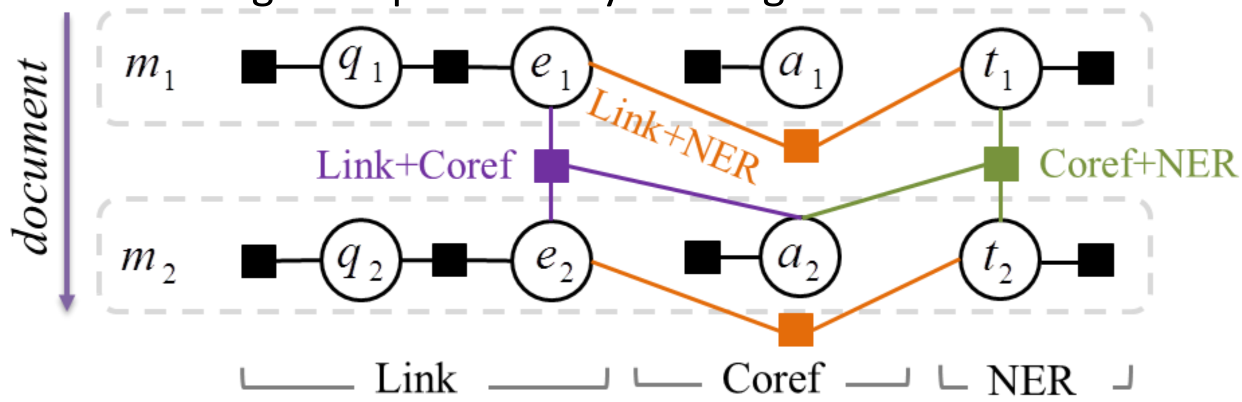
m = "Oregon State University"

$t = ORG$

Graphic Model: Joint Entity Linking, Typing, and Coreference Task [Greg Durrett and Dan Klein. TACL 2014]

This figure is provided by the original author

Joint Model



Learning

- The objective can be optimized using *AdaGrad* algorithm.

Inference

- ***Belief propagation*** is still the best choice, but not efficient enough.
  - **Solution**: use a threshold to prune away most of bad links in for coreference variables, but keep only ***k*** remaining.

Summary of Graphic Model Approaches

Advantages

- The powerful capability of representation.
- Easy to deal with missing labels.

Challenges

- The main learning difficulty in these graphic models is its complicate structure
  - Graph decomposition during learning and inference by ignoring some other parts of the graph.
- Huge number of (hidden) variables and parameters.
  - Pruning candidate values;
  - Fixing some of variable values at early stages

Search-based inference for Structured Prediction

x → $f: \mathcal{X}, \mathcal{Y} \longrightarrow \mathcal{R}$ → y

Input

Scoring function

Output

Learning

$$f(x, y) = w \cdot \underbrace{\phi(x, y)}_{\text{Feature Vector}}$$

Inference

$$\hat{y} = \underbrace{\underset{y}{argmax}\, f(x, y)}_{\text{Intractable in most cases}}$$

Candidate Methods:

- Structured Perceptron
- $\boxed{\text{Structured SVM}}$

……

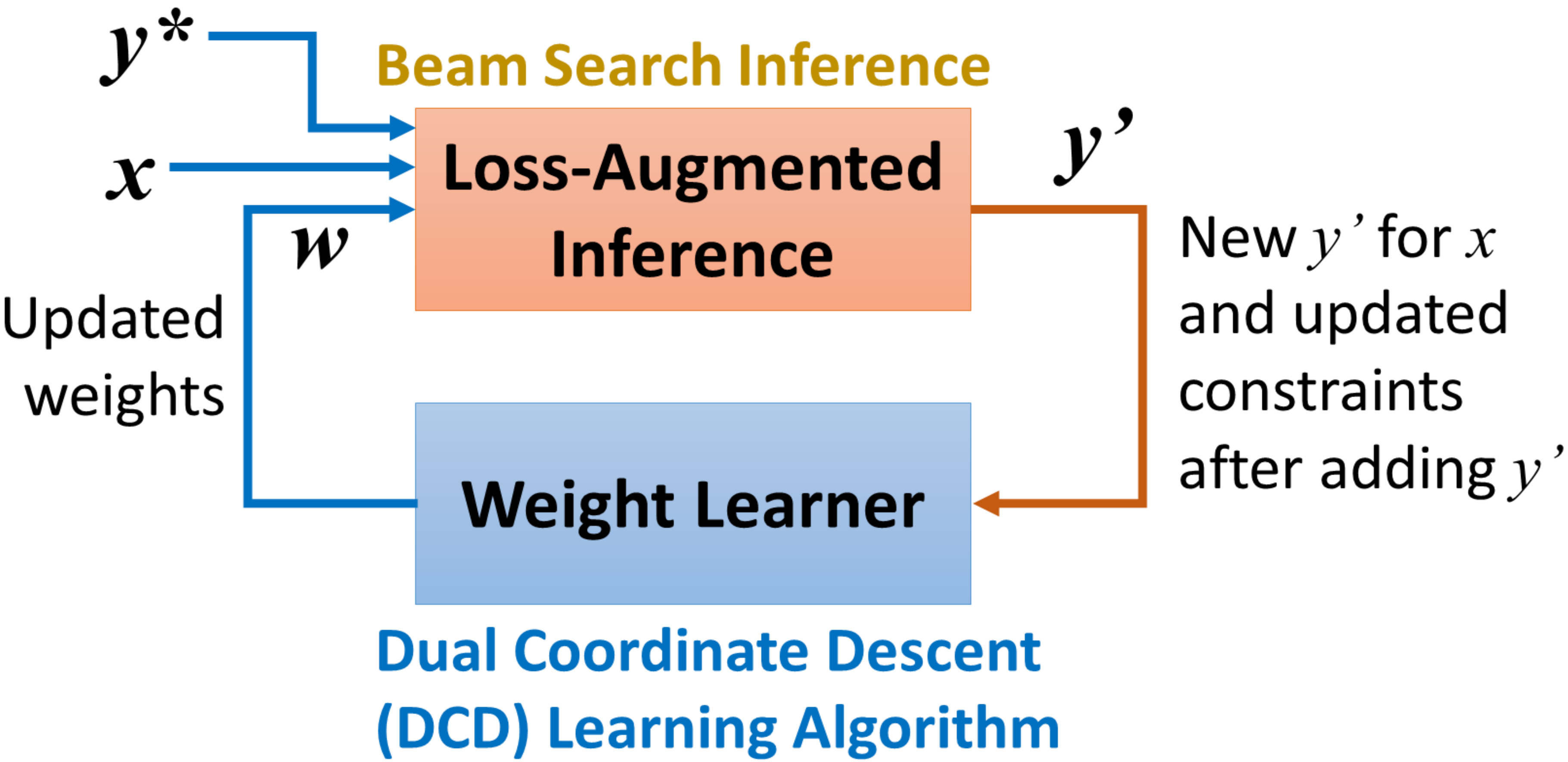
Candidate Methods:

- $\boxed{\text{Beam Search}}$

……

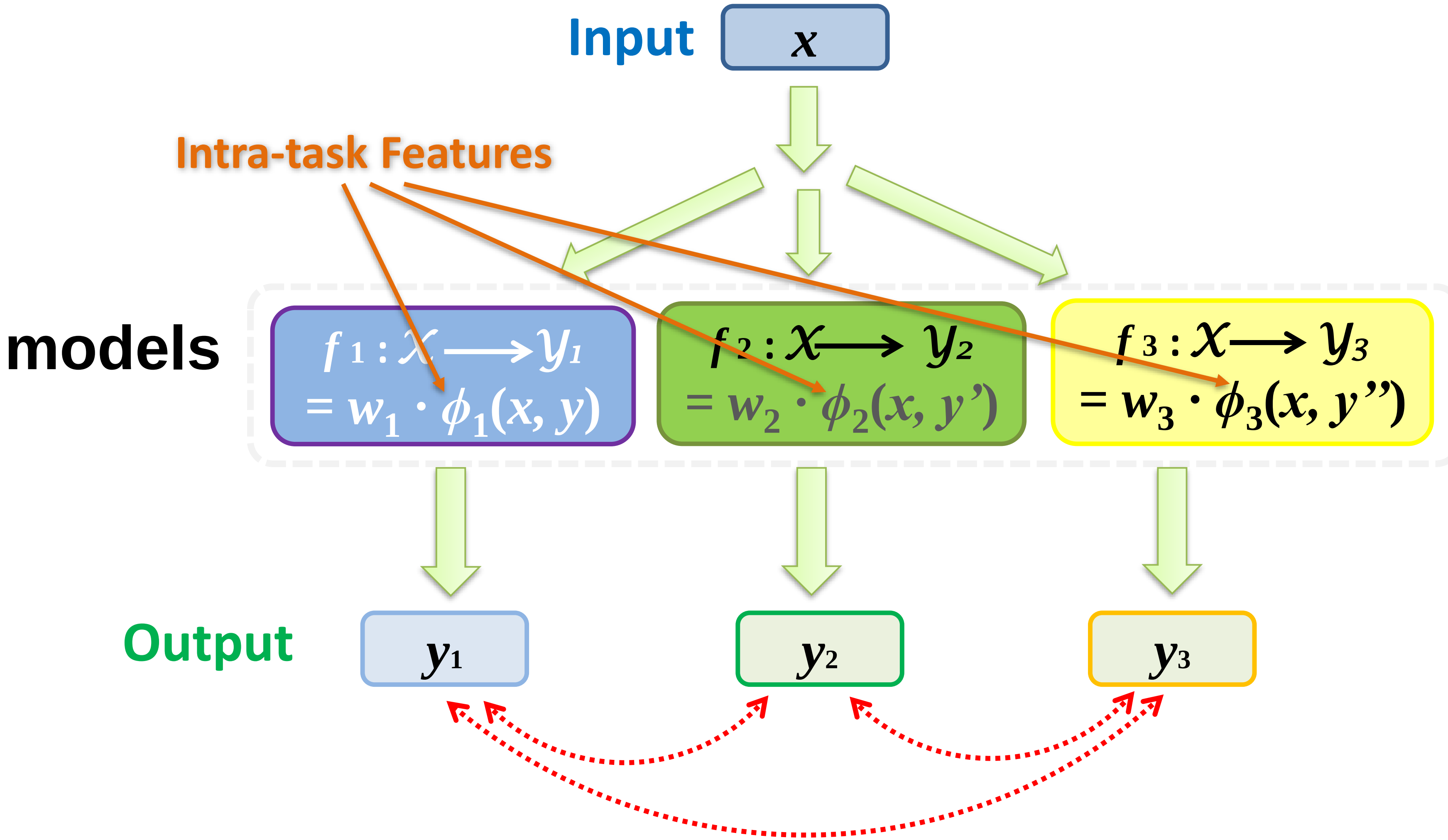
This Work

Structured SVM Learning with Search-based Inference



Multi-Task Structured Prediction

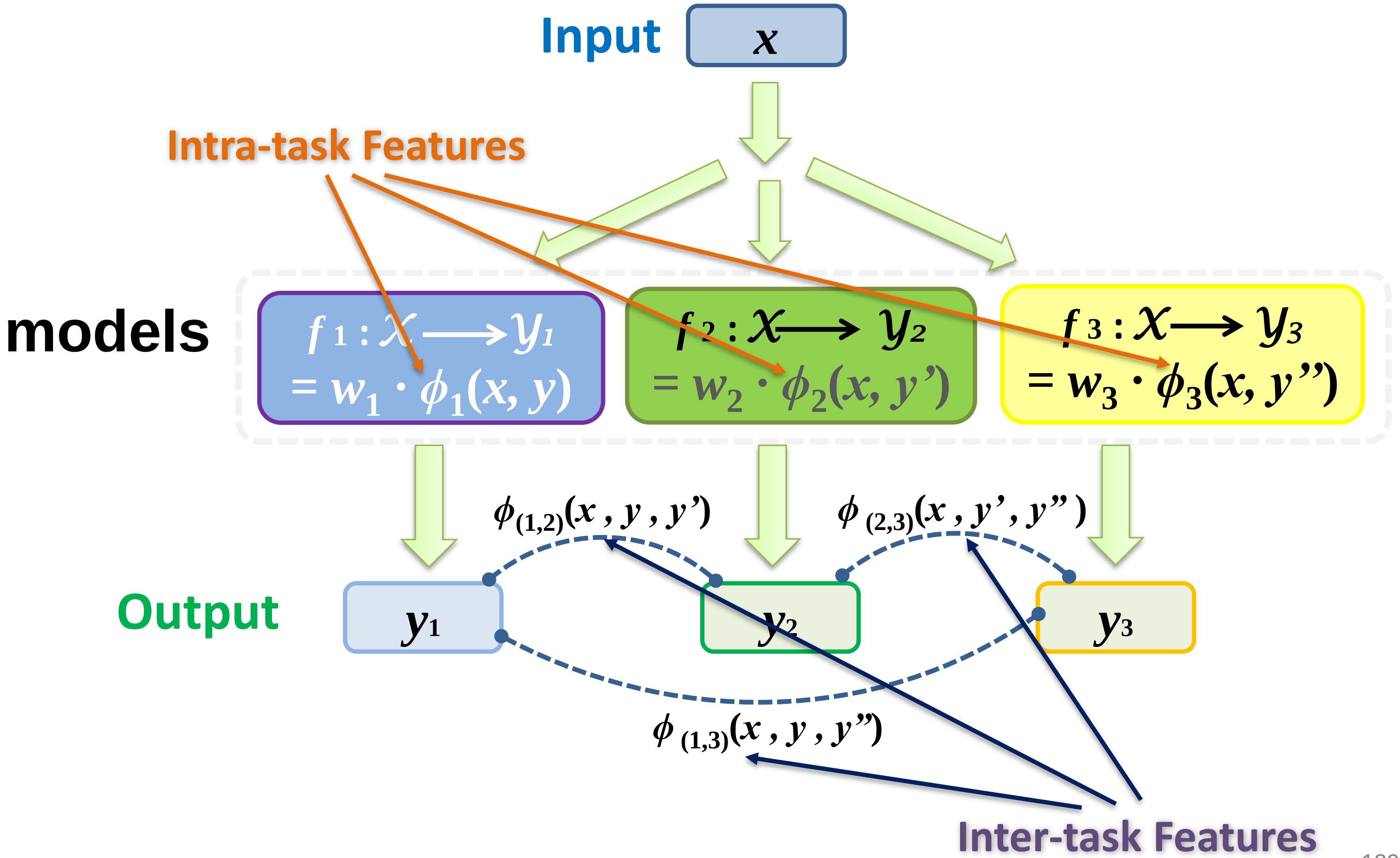
Multi-Task Structured Prediction (MTSP):



- *How to exploit the interdependencies between tasks?*

Multi-Task Structured Prediction

Introduce Inter-task Features:



Pipeline Architecture

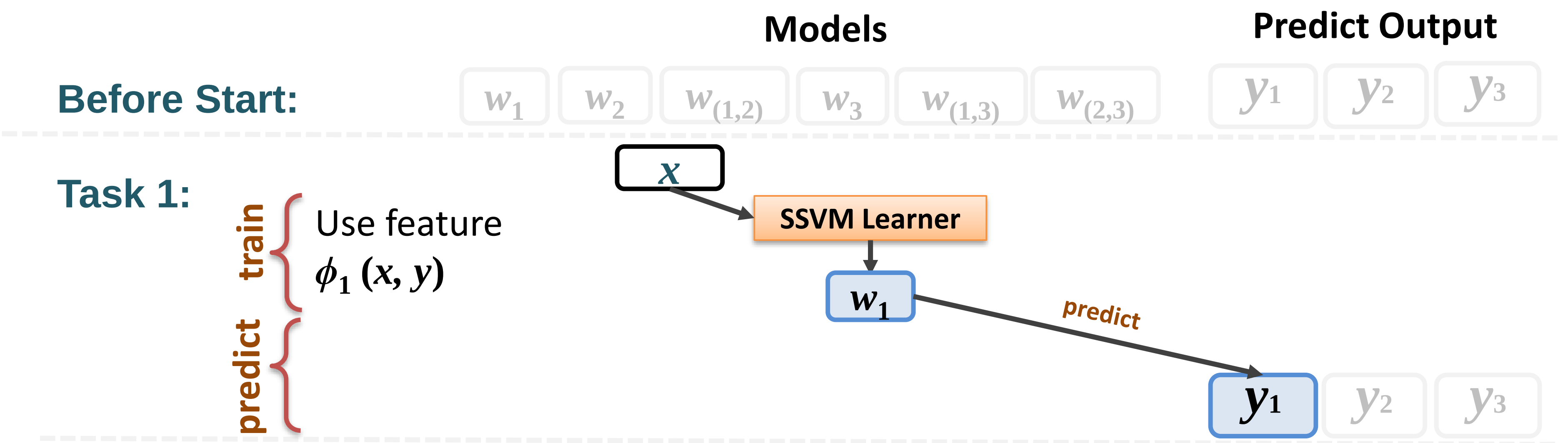
Learning k (= 3) independent models, one after another;

	Models						Predict Output		
Before Start:	w_1	w_2	$w_{(1,2)}$	w_3	$w_{(1,3)}$	$w_{(2,3)}$	y_1	y_2	y_3

Define a order: Task 1 → Task 2 → Task 3

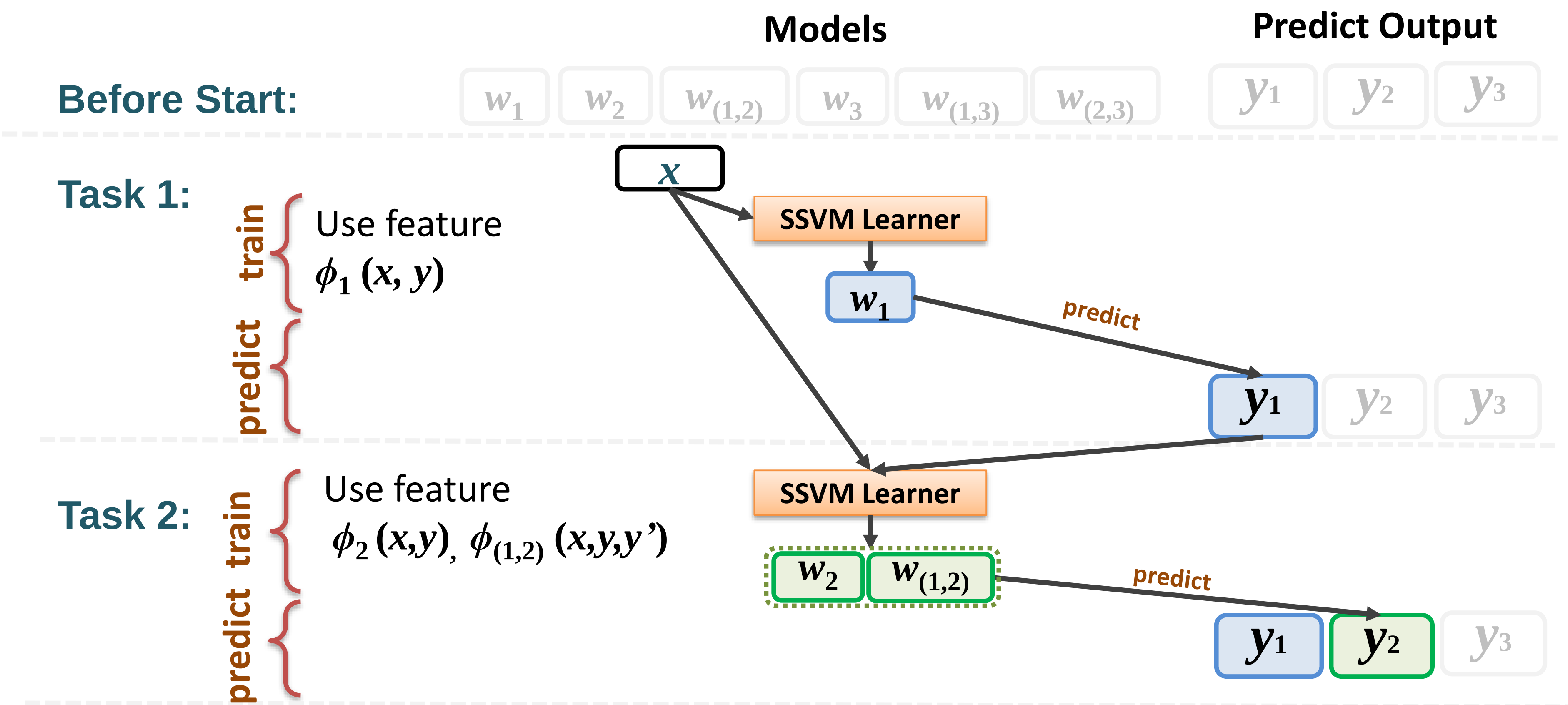
Pipeline Architecture

Learning k (= 3) independent models, one after another;



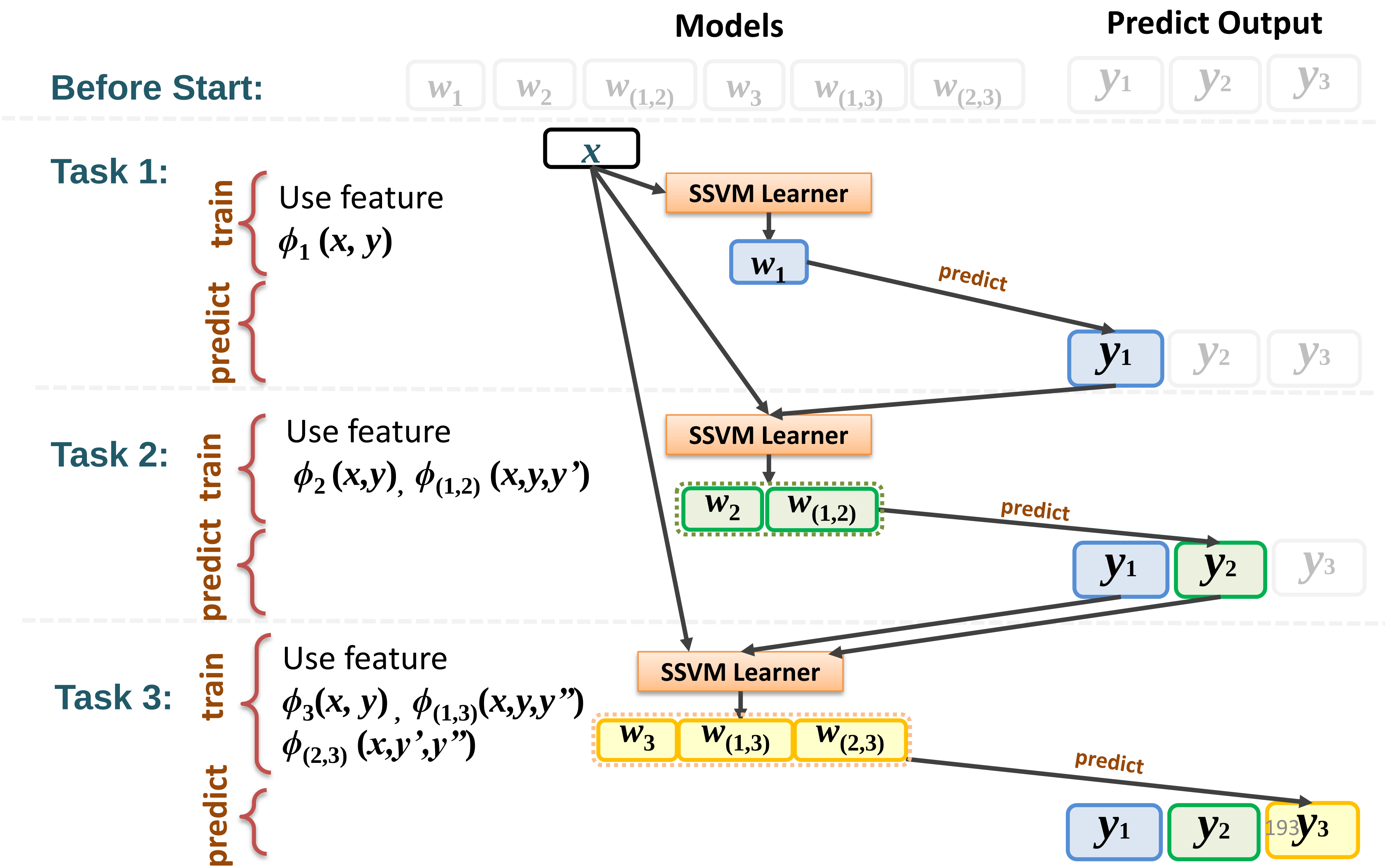
Pipeline Architecture

Learning k (= 3) independent models, one after another;

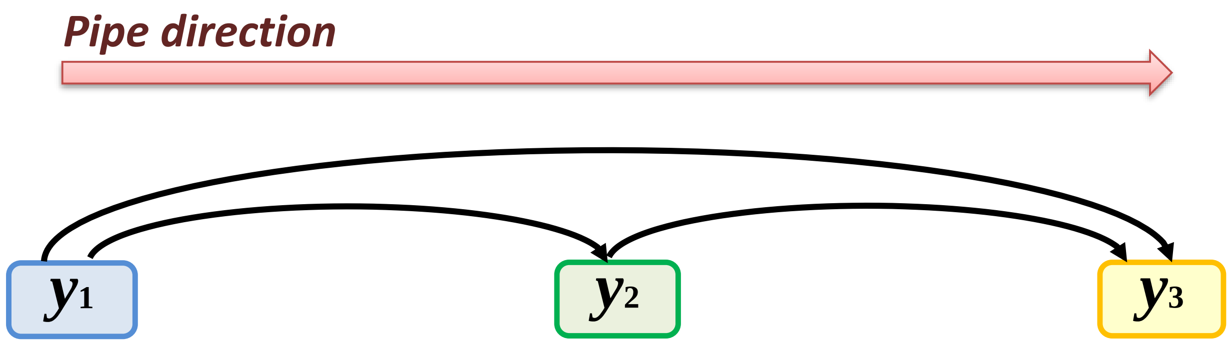


Pipeline Architecture

Learning k (= 3) independent models, one after another;



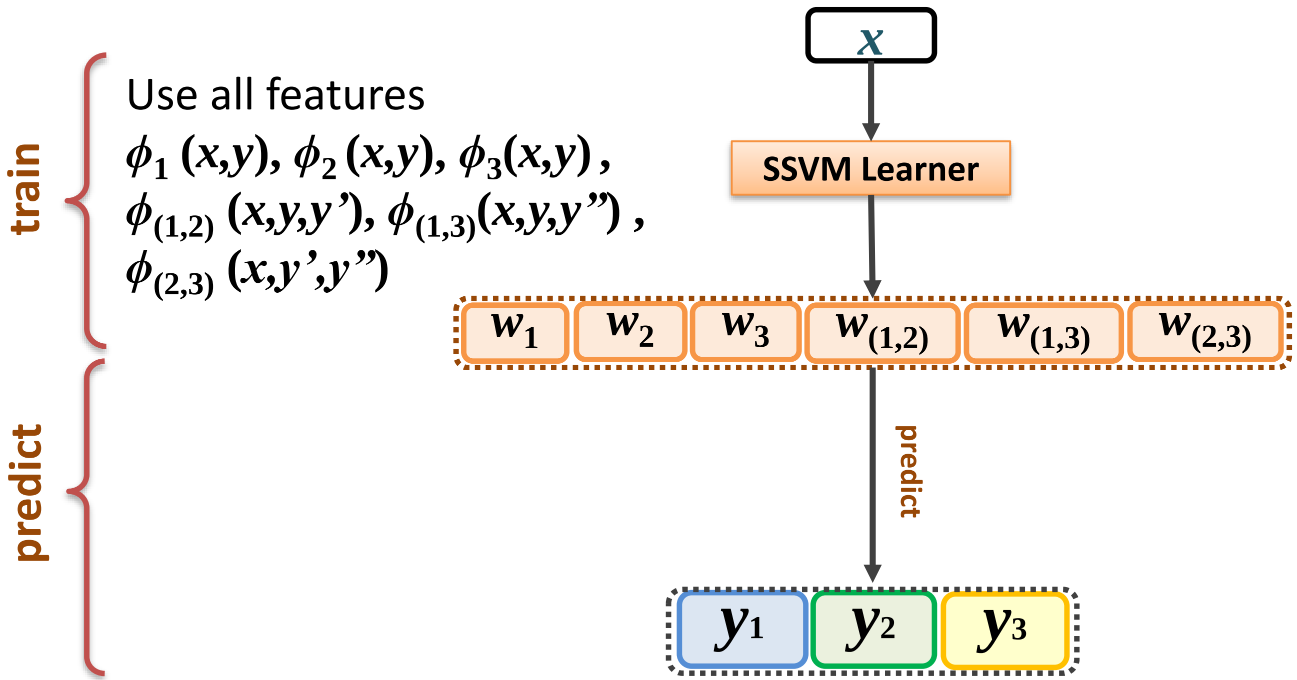
Pipeline Performance Depends on Task Order



- ❑ The task performs better when it is placed last in order.
- ❑ There is **no** ordering that allows the pipeline to reach peak performance on all the three tasks.

Joint Architecture

Task 1 & 2 & 3:



$$\phi = \phi_1(x,y) \circ \phi_2(x,y) \circ \phi_3(x,y) \circ \phi_{(1,2)}(x,y,y') \circ \phi_{(1,3)}(x,y,y'') \circ \phi_{(2,3)}(x,y',y'')$$

Vector concatenation

Big Problem: Huge branching factor for search

Pruning

A pruner is a classifier to prune the domain of each variable using state features.

Score-agnostic Pruning

Training Data $\xrightarrow{\text{Pruner training \& predicting}}$ *Pruned Data* $\xrightarrow{\text{SSVM Learner}}$ *Cost function* $\xrightarrow{\text{testing}}$ *Testing Results*

- Can accelerate the training time;
- May or may not improve the testing accuracy;

Score-sensitive Pruning

Training Data $\xrightarrow{\text{SSVM Learner}}$ *Cost function* $\xrightarrow{\text{Pruner training \& predicting}}$ *Testing Results*

- Can improve the testing accuracy;
- No training speedup, but evaluation does speed up.

Cyclic Architecture

Pipeline architecture

Task 1 → Task 2 → Task 3

Connect the tail of pipeline to the head?

Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: **Define a order: Task 1 → Task 2 → Task 3**

Step 2: **Predict initial outputs:** $\boxed{y_1}$ $\boxed{y_2}$ $\boxed{y_3}$

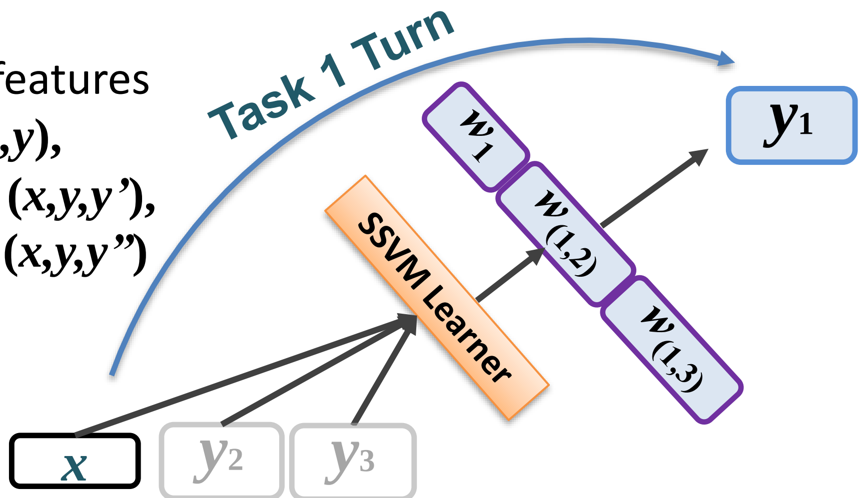
Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: **Define a order: Task 1 → Task 2 → Task 3**

Step 2: **Predict initial outputs:** y_1 y_2 y_3

Use features
$\phi_1\,(x,y)$,
$\phi_{(1,2)}\,(x,y,y')$,
$\phi_{(1,3)}\,(x,y,y'')$

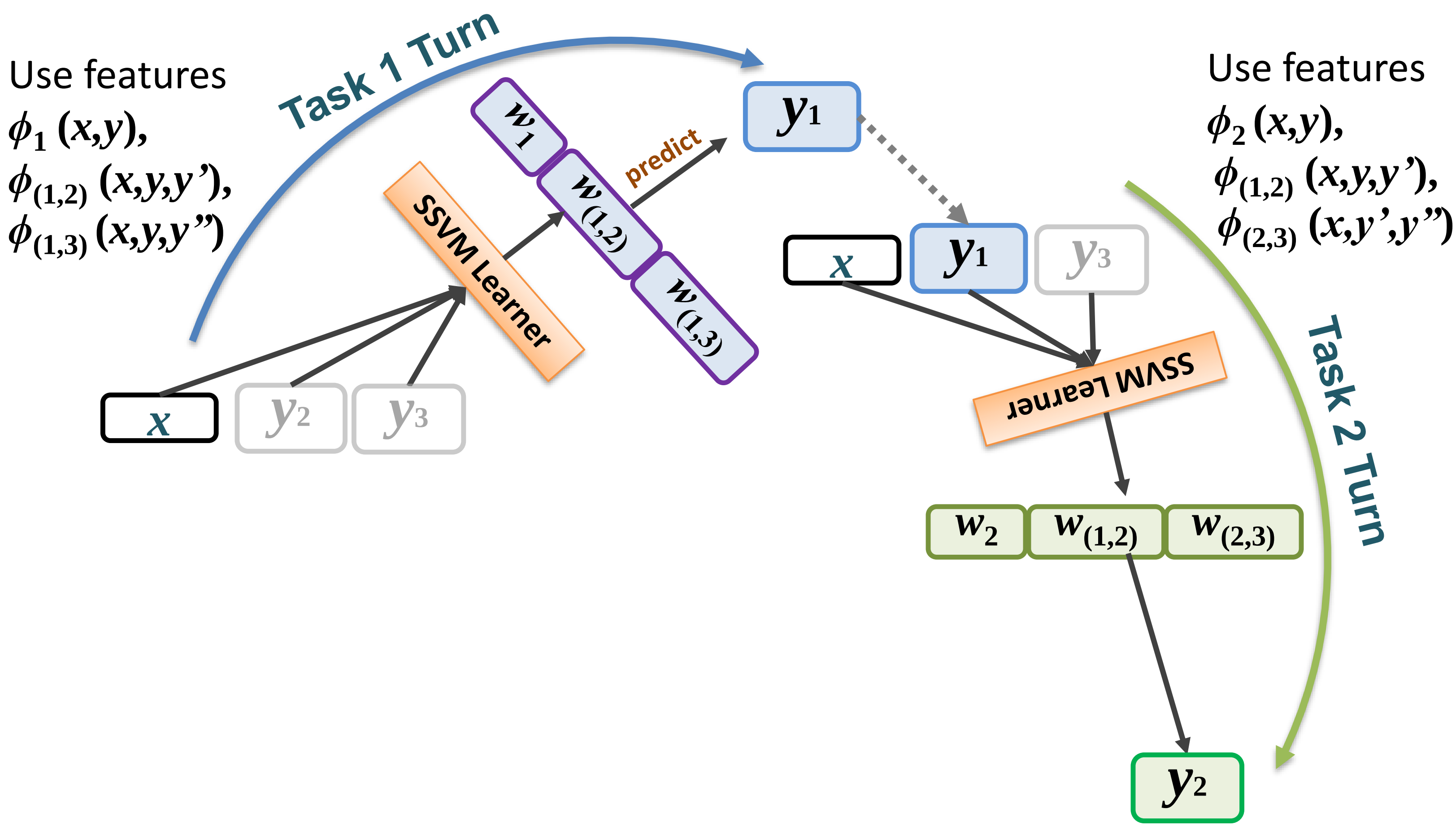


Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: **Define a order: Task 1 → Task 2 → Task 3**

Step 2: **Predict initial outputs:** y_1 y_2 y_3

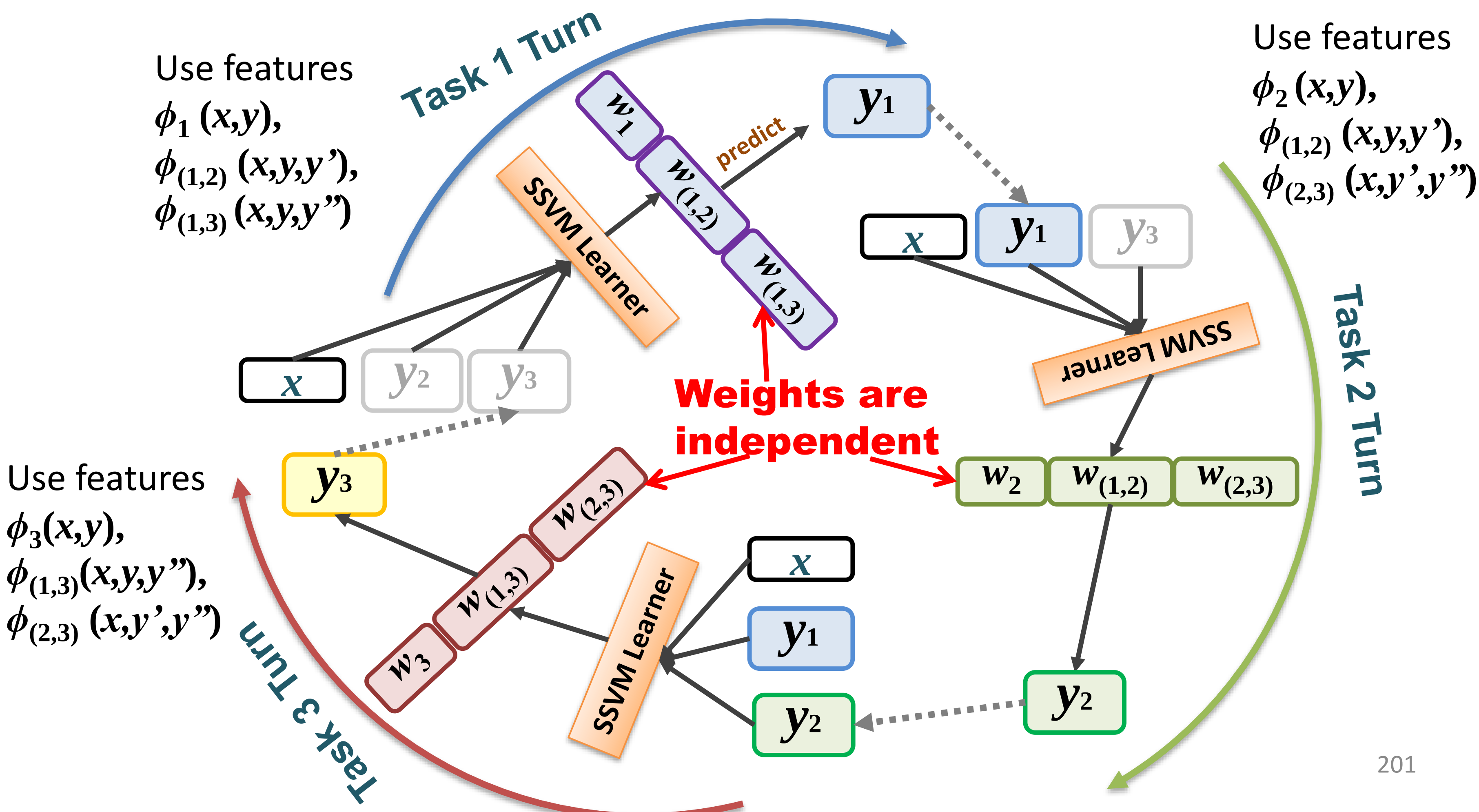


Cyclic Architecture

Unshared-Weight-Cyclic Training

Step 1: **Define a order: Task 1 → Task 2 → Task 3**

Step 2: **Predict initial outputs:** y_1 y_2 y_3



Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: y_1 y_2 y_3

x y_1 y_2 y_3

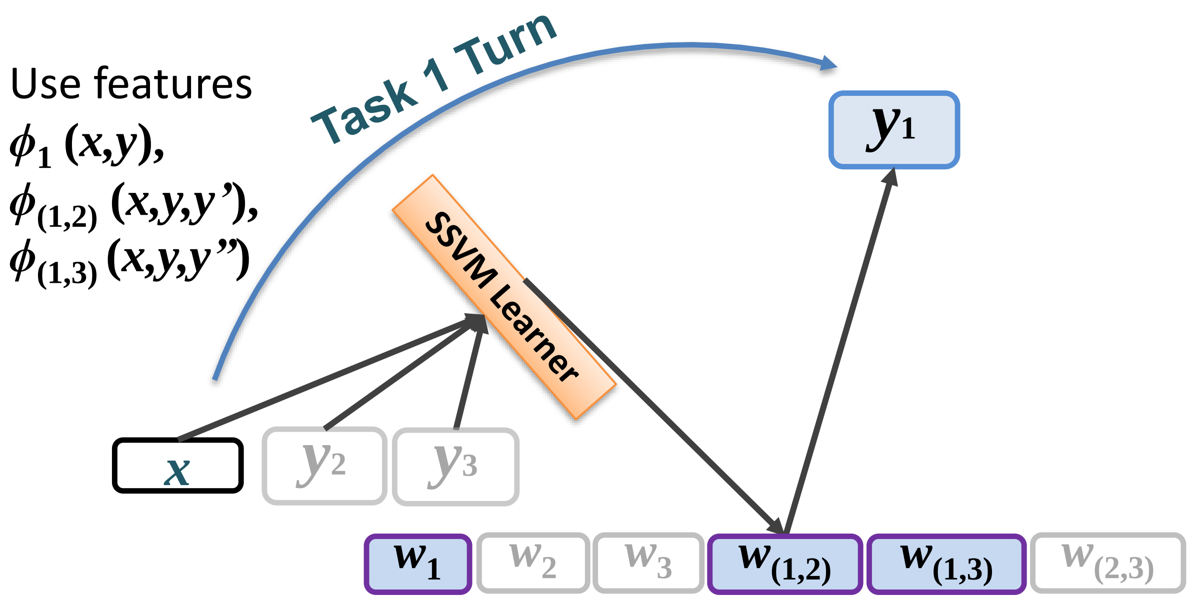
w_1 w_2 w_3 $w_{(1,2)}$ $w_{(1,3)}$ $w_{(2,3)}$

Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: **Define a order: Task 1 → Task 2 → Task 3**

Step 2: **Predict initial outputs:** y_1 y_2 y_3

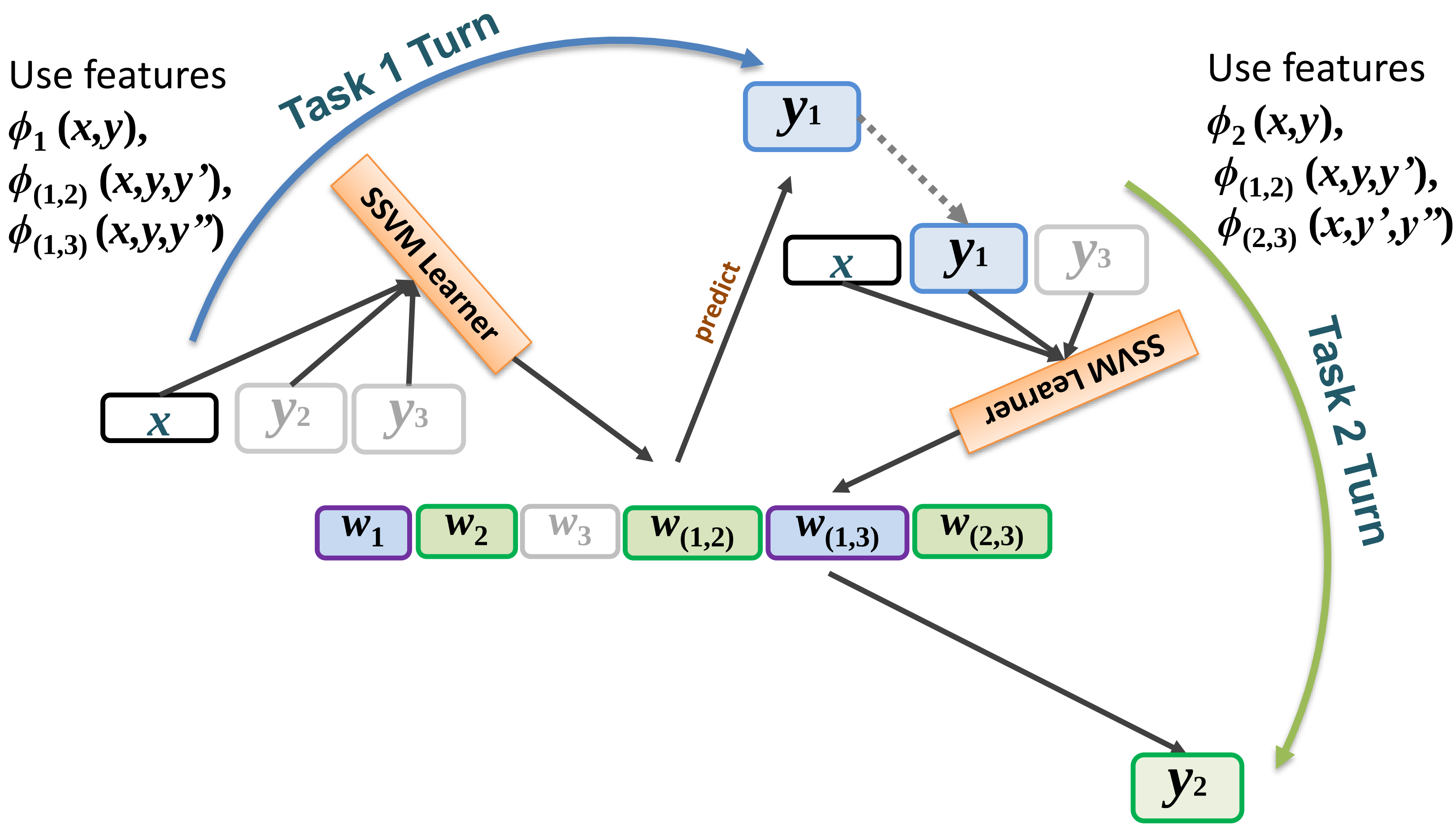


Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: y_1 y_2 y_3

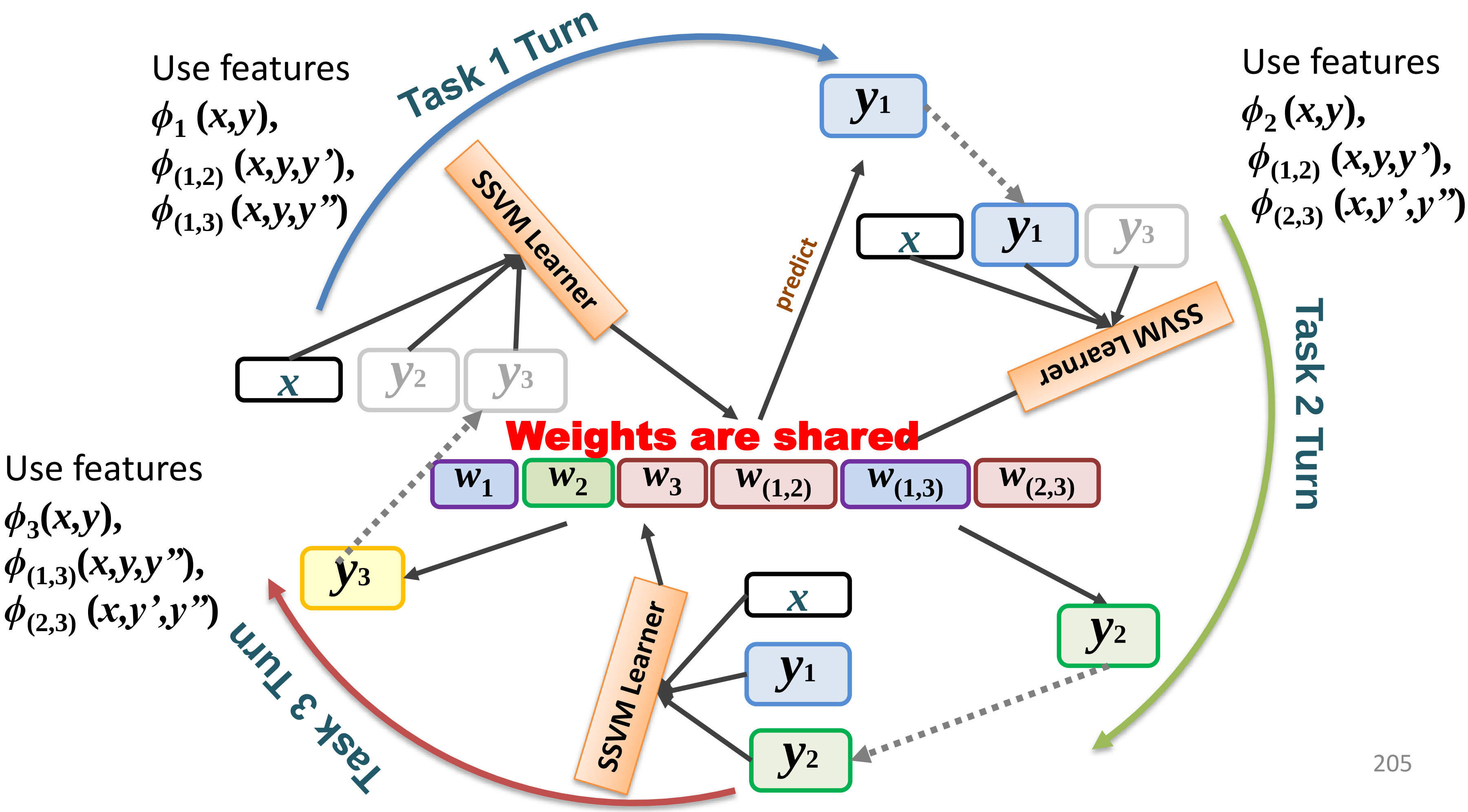


Cyclic Architecture

Shared-Weight-Cyclic Training

Step 1: Define a order: Task 1 → Task 2 → Task 3

Step 2: Predict initial outputs: y_1 y_2 y_3



Summary of Search-based Approaches

1. MTSP outperform the STSP by exploiting interdependency, which is captured by inter-task features.
2. Search-based inference for large structured prediction problems suffers from local optima and is mitigated by a good initialization.
3. ***Pipeline*** architecture is the fastest on both training and testing, but low accuracy; ***Joint*** architecture is good on accuracy, but slow speed; ***Cyclic*** is a trade-off between these two.
4. Score-sensitive pruning of joint MTSP performs the best and takes most time.
5. Unshared weights usually performs better than shared weights.

Summary

- **Different frameworks for structured prediction [Jana]**
  - Cost function learning framework and recent advances
  - Control knowledge learning framework (greedy and beam search)
  - HC-Search: A Unifying framework
- **Integrating deep learning and structured prediction [Liping]**
  - Deep learning ∩ cost function learning
  - Deep learning ∩ control knowledge learning
- **Multi-task structured prediction [ChaoMa]**
  - Graphical models approach
  - Search based learning and inference architectures

Future Directions

- Design and optimization of search spaces for complex structured prediction problems
  - very under-studied problem
- Leveraging deep learning advances to improve the performance of structured prediction approaches
  - Loose vs. tight integration
- Learning to trade-off speed and accuracy of structured prediction
  - Active research topic, but relatively less work
- What architectures are more suitable for "Anytime" predictions? How to learn for anytime prediction?

Future Directions

- Theoretical analysis: sample complexity and generalization bounds
  - Lot of room for this line of work in the context of "learning" + "search" approaches
- Understanding and analyzing structured predictors in the context of integrated applications
  - Pipelines in NLP and Vision among others
- Amortized inference or speedup learning for other inference formulations
- (Multi-task) structured prediction with weak supervision
  - Dan Roth: Incidental Supervision: Moving beyond Supervised Learning. AAAI 2017