Integrating Learning and Search for Structured Prediction

Alan Fern

Oregon State University

Liang Huang

Oregon State University

Jana Doppa

Washington State University

Tutorial at International Joint Conference on Artificial Intelligence (IJCAI), 2016

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**

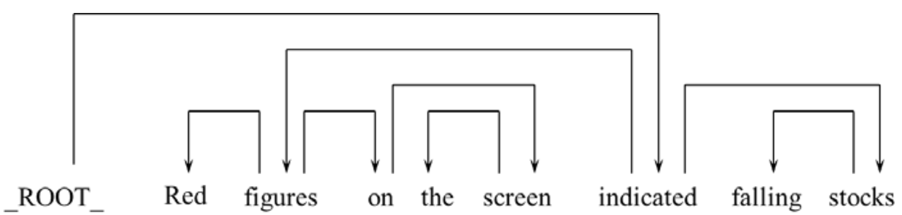
x = "The cat ran" y = *<article> <noun> <verb>*

- **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

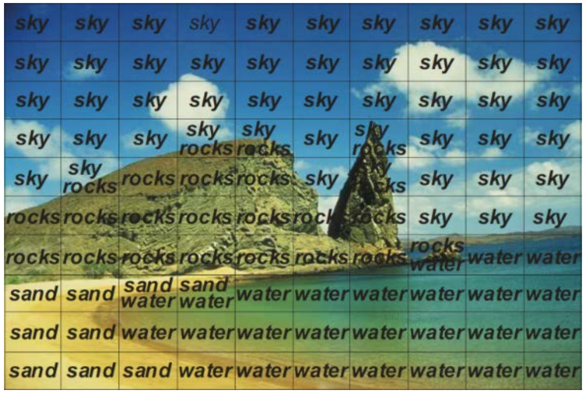
Examples of Bad Prediction

Computer Vision Examples

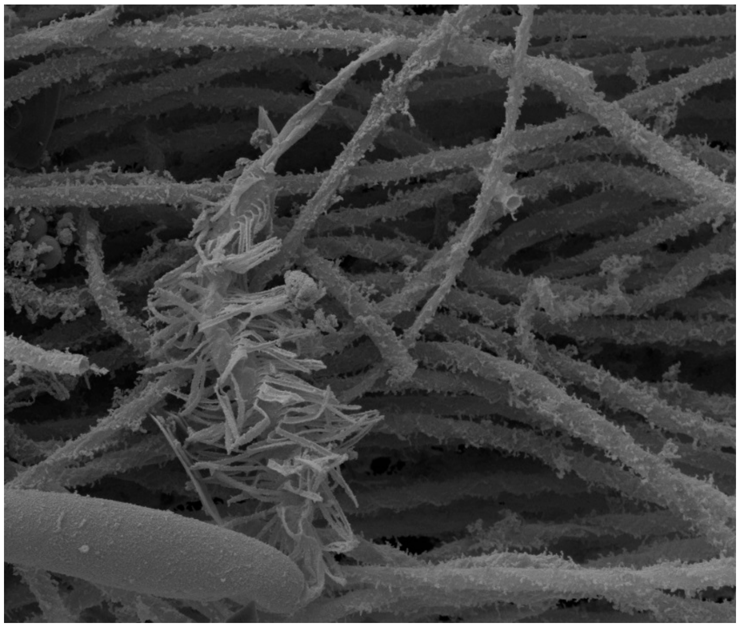
Scene Labeling

Image

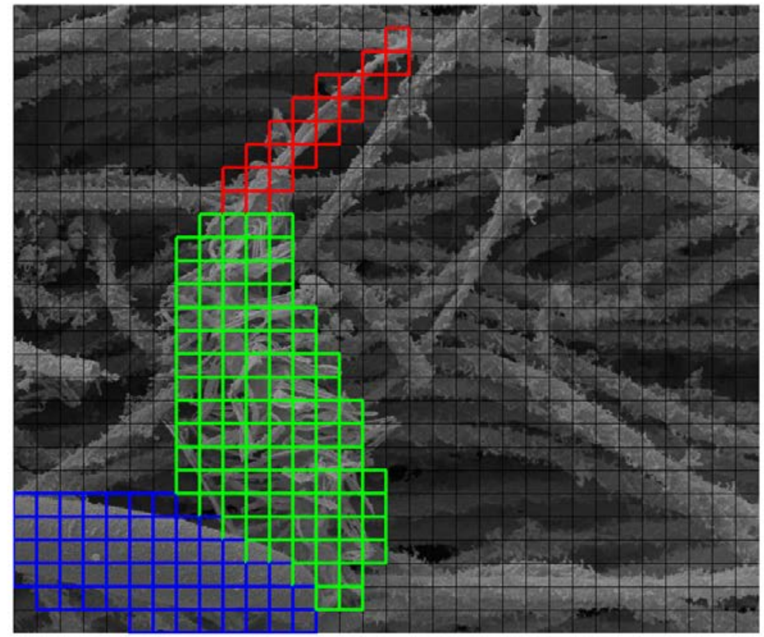
Labeling



Biological Image Analysis



Nematocyst Image

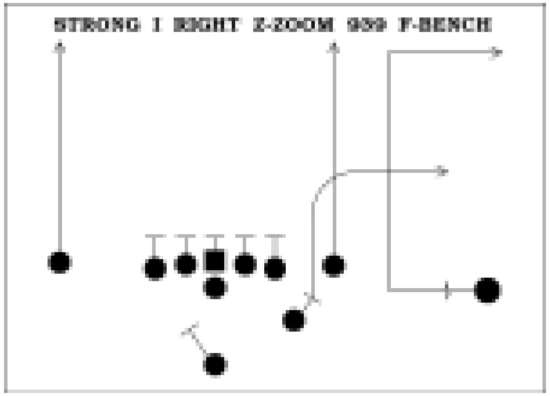


Body parts of the nematocyst

The OSU Digital Scout Project

Objective: compute semantic interpretations of football video

Raw video

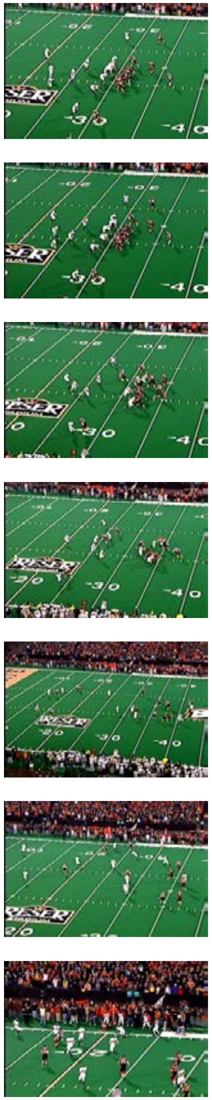


High-level interpretation of play

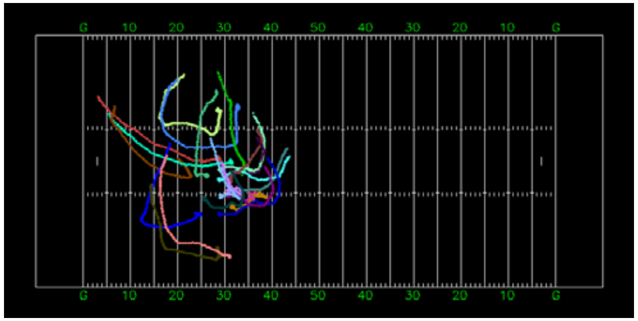
- Help automate tedious video annotation done by pro/college/HS teams
  - Working with hudl (hudl.com)
- Requires advancing state-of-the-art in computer vision, including:
  - registration, multi-object tracking, event/activity recognition

Multi-Object Tracking in Videos

Video



Player Trajectories



Automated Planning

Planning

A planning problem gives:

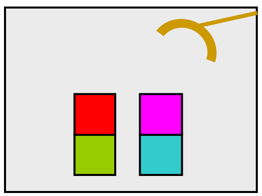
- an initial state
- a goal condition
- a list of actions and their semantics (e.g. STRIPS)

Initial State

?

Available actions:

Pickup(x)
PutDown(x,y)



Goal State

Objective: find action sequence from initial state to goal

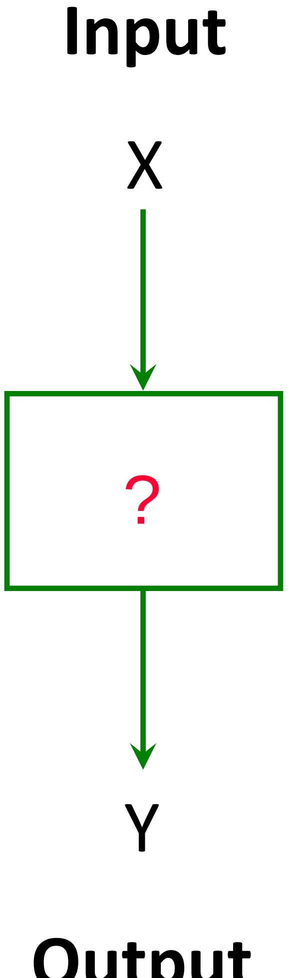
Common Theme

- 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>

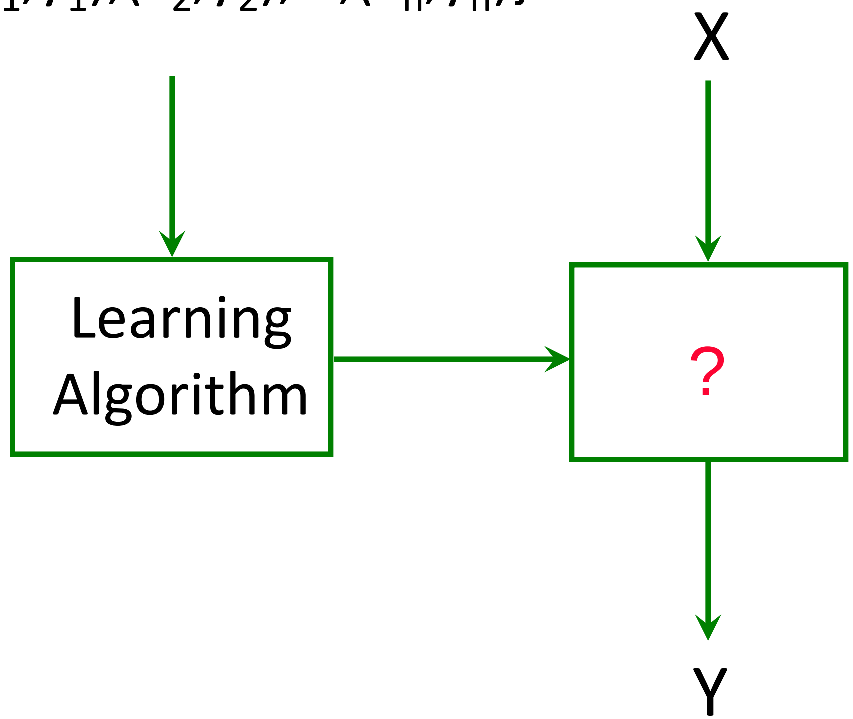
X - image of a face

$Y \in \{\text{male, female}\}$

Learning a Classifier

(, male)

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

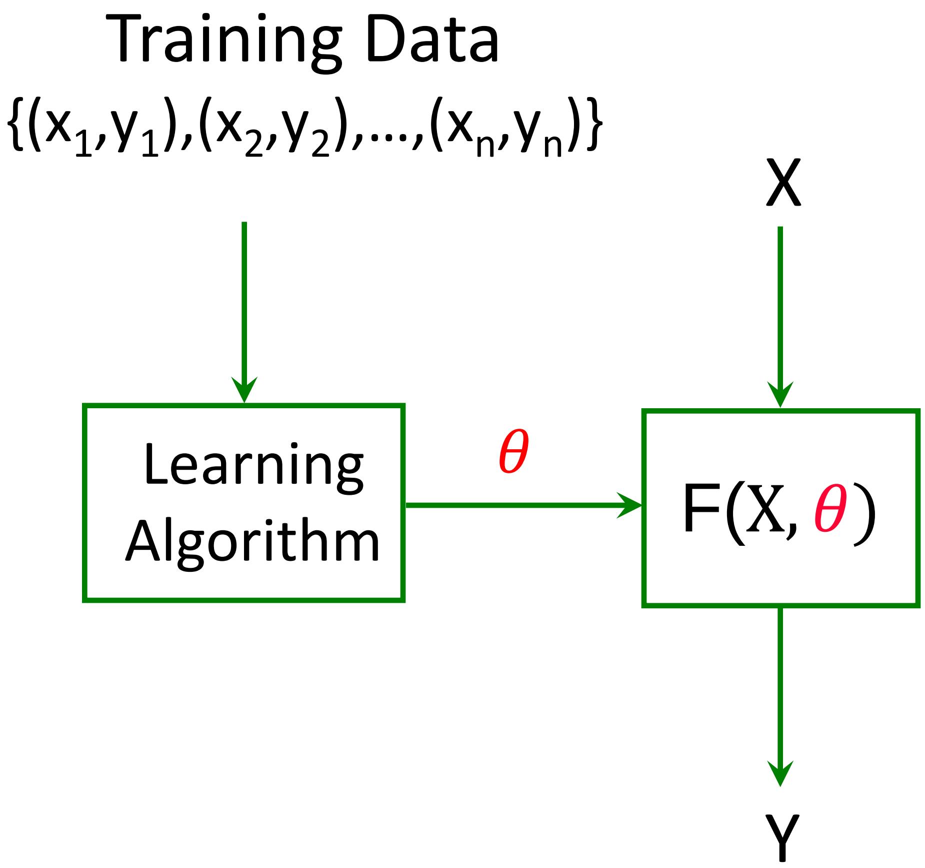


<u>Example problem:</u>

X - image of a face

$Y \in$ {male, female}

Learning a Classifier

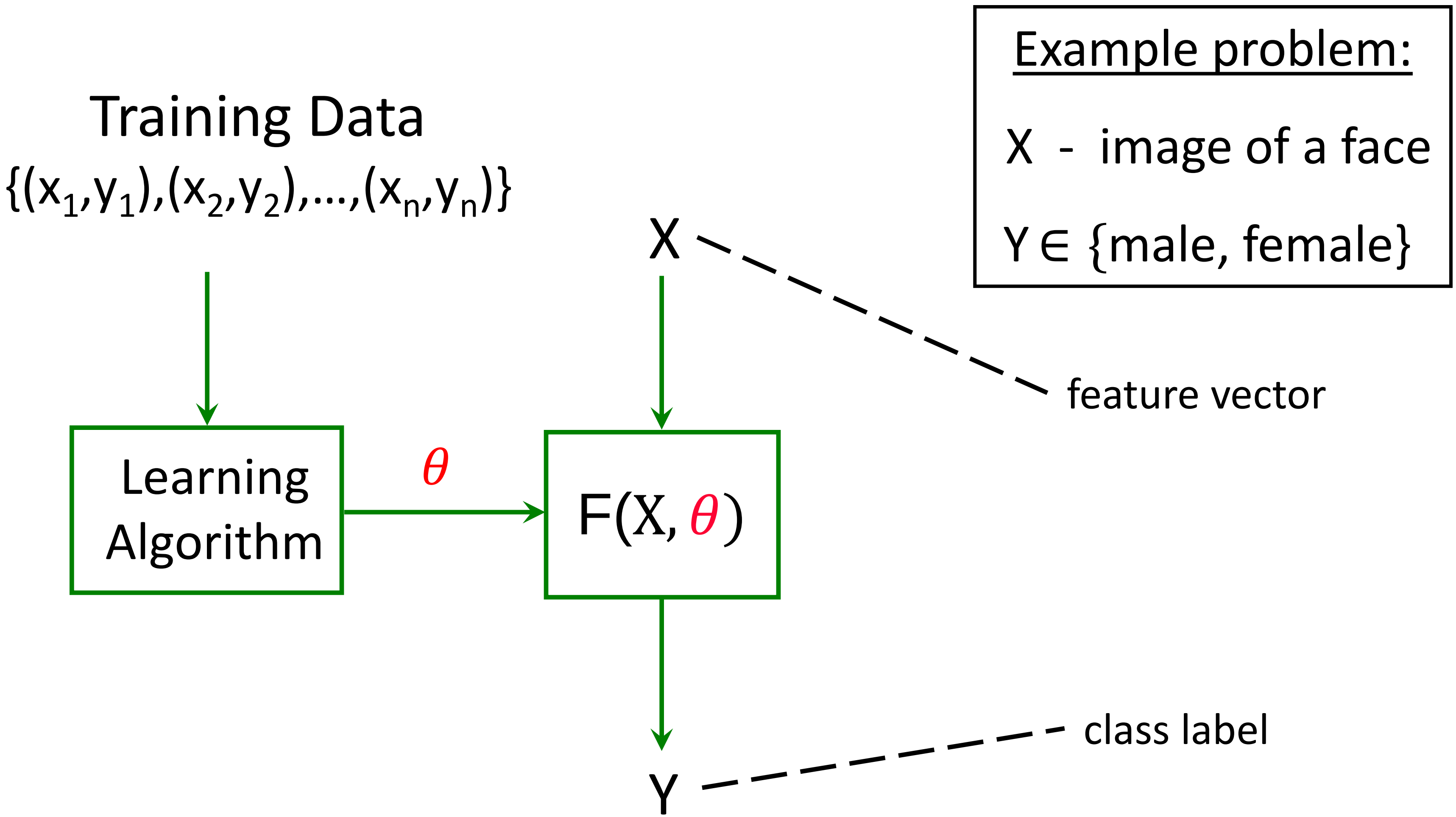


<u>Example problem:</u>

X - image of a face

$Y \in$ {male, female}

Learning for Simple Outputs

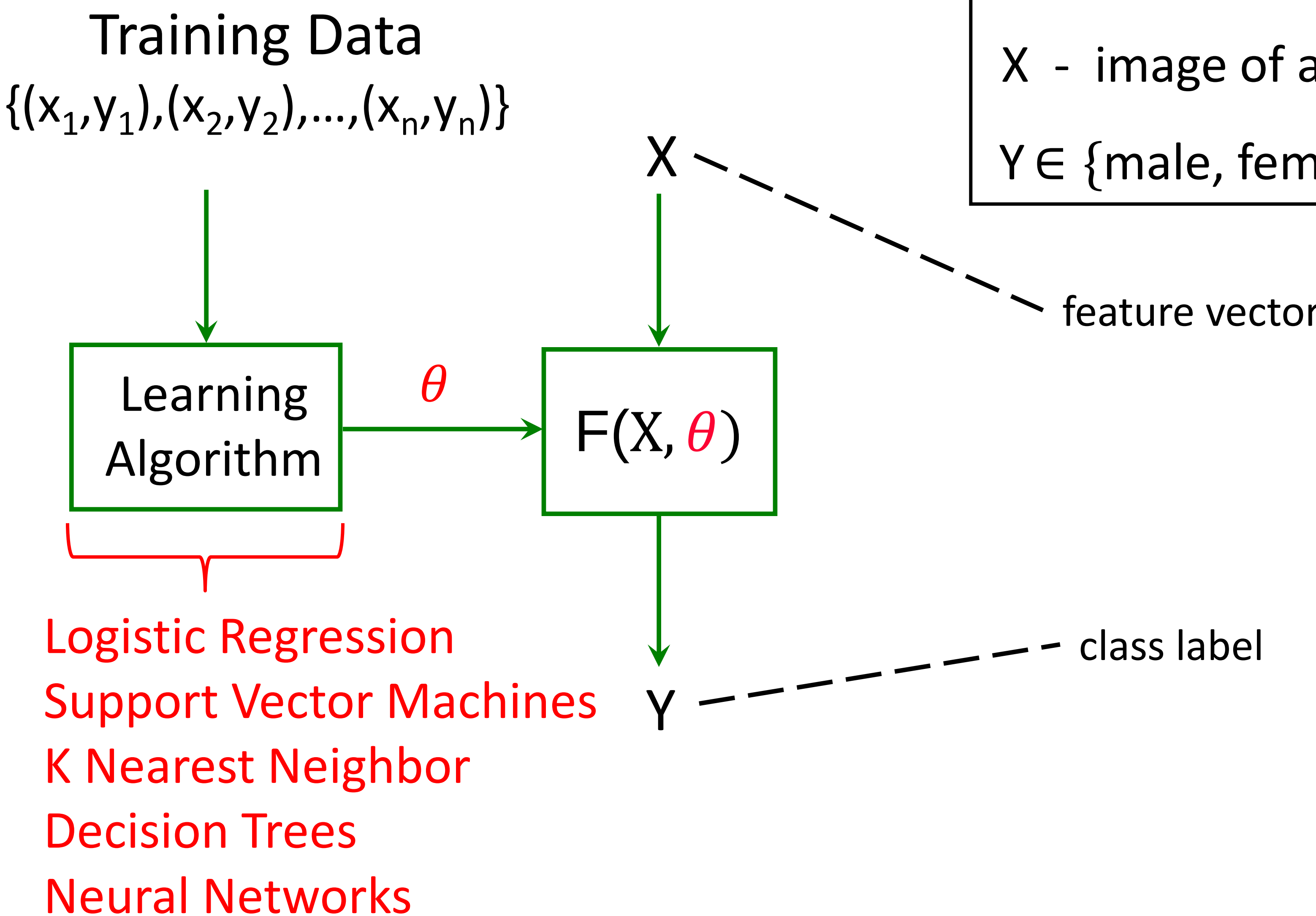


Learning for Simple Outputs

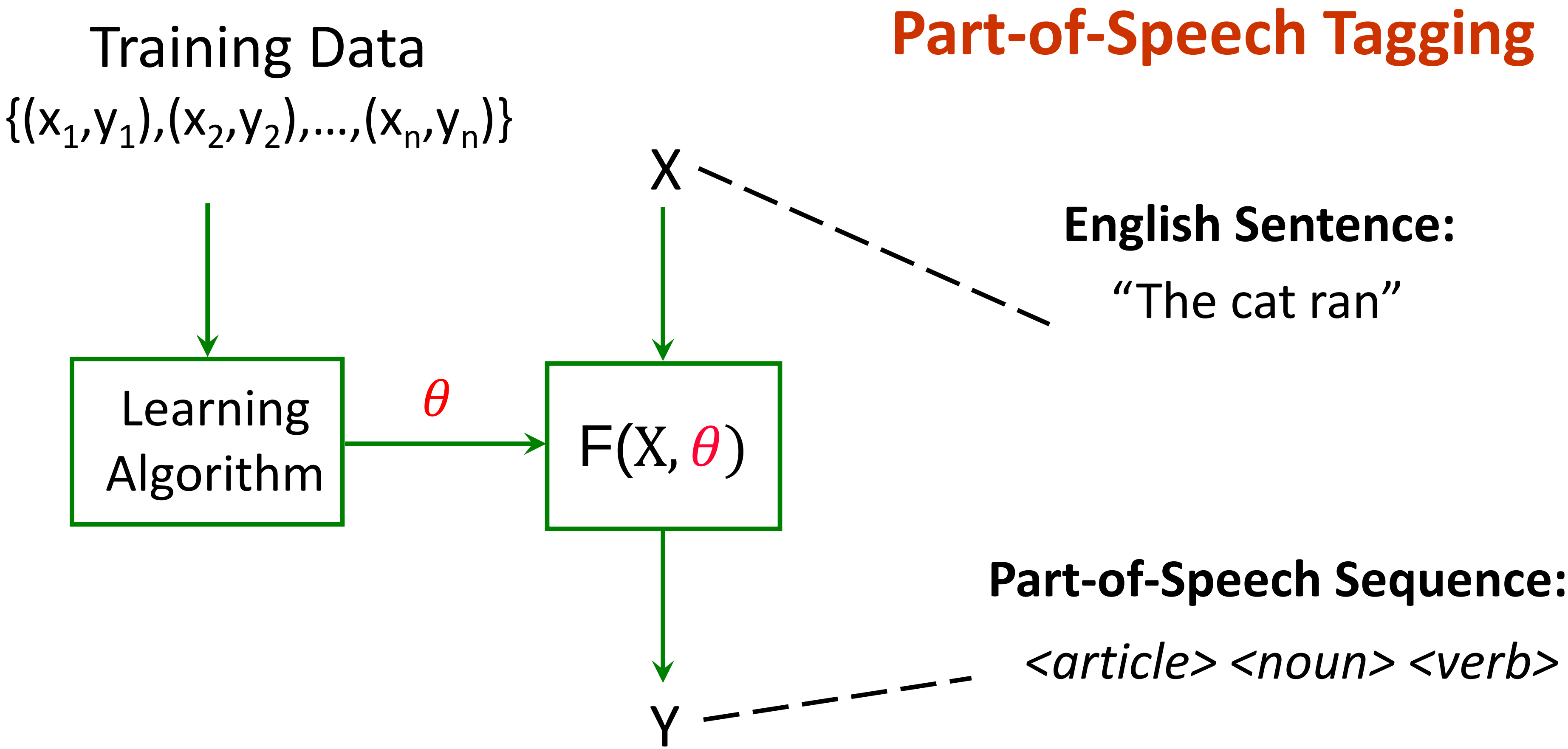
Example problem:

X - image of a face

$Y \in$ {male, female}



Learning for Structured Outputs



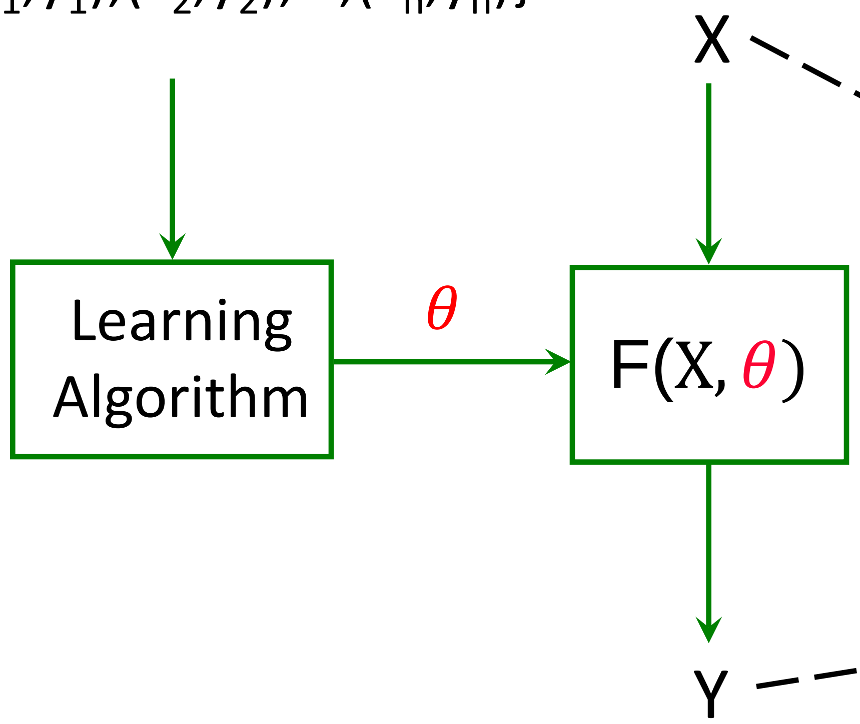
$\boldsymbol{Y} =$ set of all possible POS tag sequences

Exponential !!

Learning for Structured Outputs

Training Data

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



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**."*

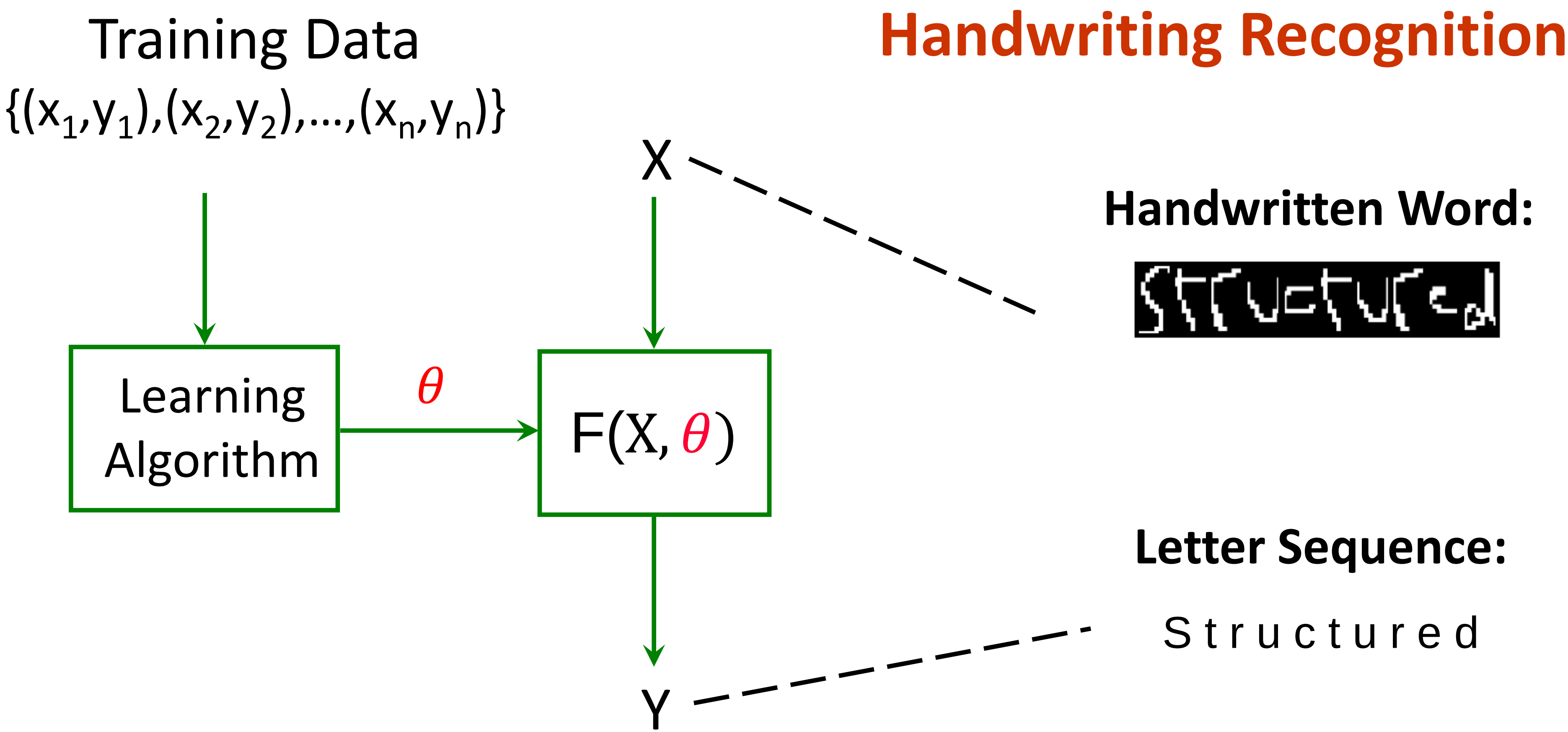
Co-reference Output:

*"**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**."*

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

Exponential !!

Learning for Structured Outputs



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

Exponential !!

Learning for Structured Outputs

Training Data

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

Learning Algorithm $\xrightarrow{\theta}$ $F(X, \theta)$

X

Y

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

Exponential !!

Image Labeling

Part 2: Cost Function Learning Framework and Argmin Inference Challenge

Cost Function Learning Approaches: Inspiration

- Generalization of traditional ML approaches to structured outputs

  - SVMs ⇒ Structured SVM [Tsochantaridis et al., 2004]
  - Logistic Regression ⇒ Conditional Random Fields [Lafferty et al., 2001]
  - Perceptron ⇒ 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

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

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

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

Exponential
size of output
space !!

Key challenge: "Argmin" Inference

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

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

Key challenge: "Argmin" Inference

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

- Time complexity of inference depends on the dependency structure of features $\phi(x, y)$
  - NP-Hard in general
  - Efficient inference algorithms exist only for simple features

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

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
  - …

Focus of Tutorial

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

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

Part 3: A Brief Overview of Search Concepts

Combinatorial Search: Key Concepts

- **Search Space**
  - Where to start the search?
  - How to navigate the space?

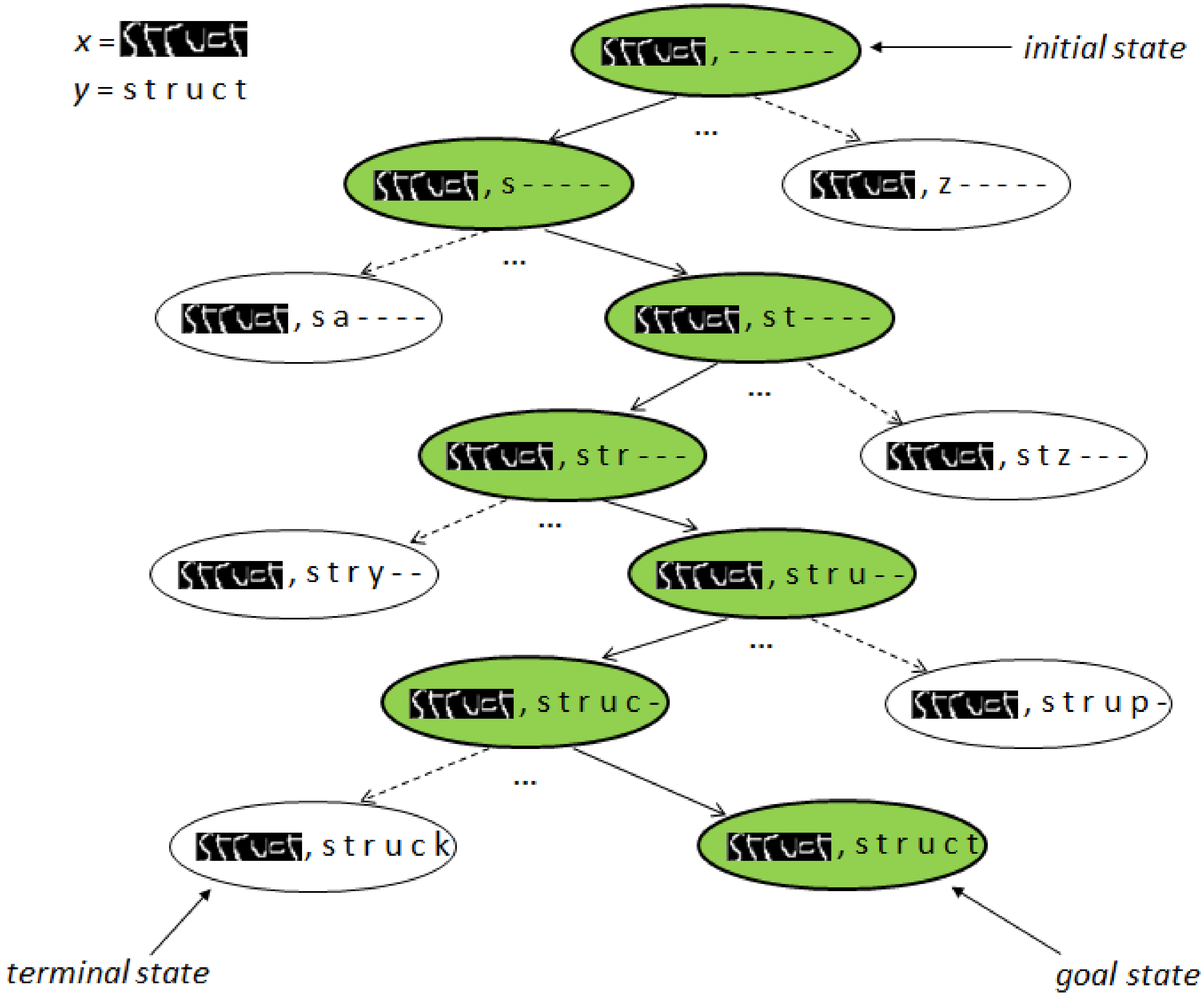
- **Search Procedure / Strategy**
  - How to conduct search?

- **Search Control Knowledge**
  - How to guide the search? (Intelligence)

Search Space Definition

- **Initial State Function:** I
  - Where to start the search?
- **Successor State Function:** S
  - What are the successor (next) states for a given state?
  - Generally, specified as a set of actions that modify the given state to compute the successor states
- **Terminal State Function:** T
  - When to stop the search?

(Ordered) Search Space: Example



Search Procedure

- **Search Tree (or Graph):** Instantiation of the search space. **How to navigate?**
- **Uninformed (Blind) Search Procedure**
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)
- **Informed (Intelligent) Search Procedure**
  - Greedy Search
  - Beam Search
  - Best-First Search
  - …

Informed Search Procedures

- Maintain an internal memory of a set of open nodes (M)
- **Intelligent search** guided by the control knowledge
- Algorithmic Framework for Best-First Search style search strategies:
  - **Selection:** score each open node in the memory M and select a subset of node(s) to expand
  - **Expansion:** expand each selected state using the successor function to generate the candidate set
  - **Pruning:** Retains a subset of all open nodes (update M) and prune away all the remaining nodes

Best-First Search Style Algorithms

- **Best-first Search** $(M = \infty)$
  - selects the best open node
  - no pruning

- **Greedy Search** $(M = 1)$
  - selection is trivial
  - prunes everything except for the best open node in the candidate set

Best-First Search Style Algorithms

- **Best-first Beam Search** $(M = B)$
  - selects the best open node
  - prunes everything except for the best B open nodes in the candidate set

- **Breadth-First Beam Search** $(M = B)$
  - selection is trivial – all B nodes
  - prunes everything except for the best B open nodes in the candidate set

Search Control Knowledge

- **Greedy Policies**
  - Classifier that selects the best action at each state
- **Heuristic Functions**
  - computes the score for each search node
  - heuristic scores are used to perform selection and pruning
- **Pruning Rules**
  - additional control knowledge to prune bad actions / states

- **Cost Function**
  - Scoring function to evaluate the terminal states

Part 4: Control Knowledge Learning Framework: Greedy Methods

Greedy Control Knowledge Learning

- **Given**
  - Search space definition (ordered or unordered)
  - Training examples (input-output pairs)
- **Learning Goal**
  - Learn a policy or classifier to make good predictions

- **Key Idea:**
  - Training examples can be seen as expert demonstrations
  - Equivalent to “Imitation Learning” or “Learning from Demonstration”
  - Reduction to classifier or rank learning

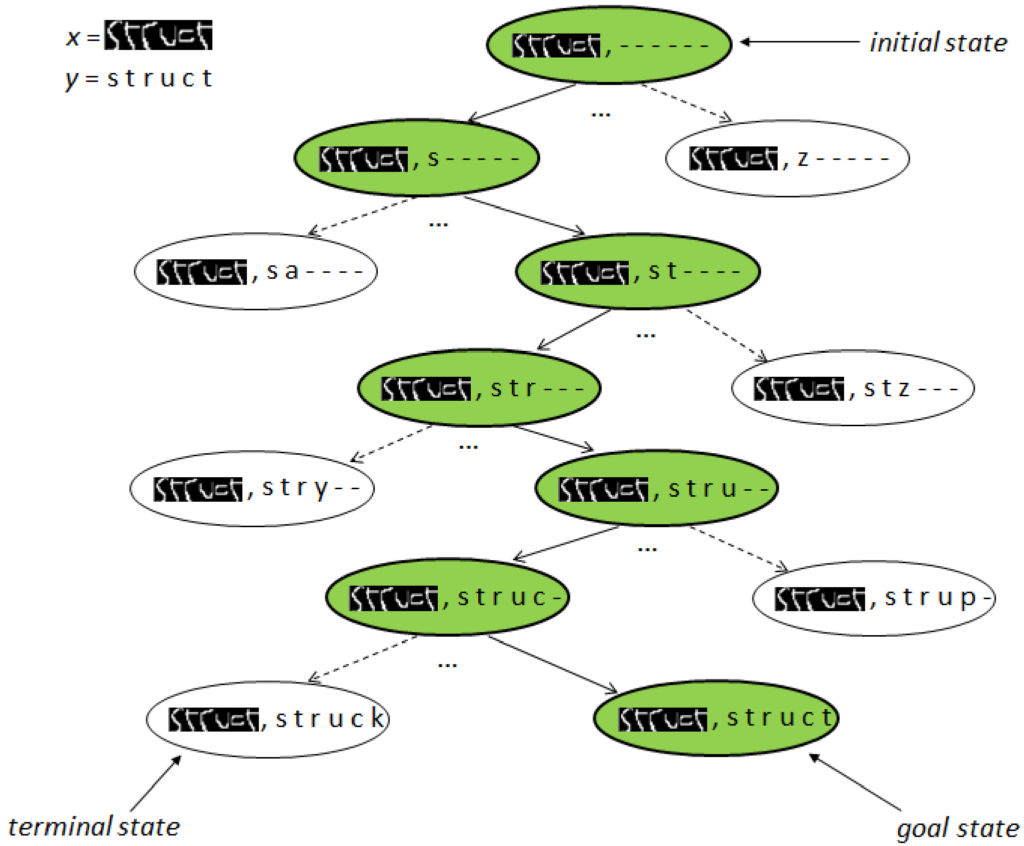
Ordered vs. Unordered Search Space

- **Ordered Search Space**
  - Fixed ordering of decisions (e.g., left-to-right in sequences)
  - Classifier based structured prediction
- **Unordered Search Space**
  - Learner dynamically orders the decisions
  - Easy-First approach

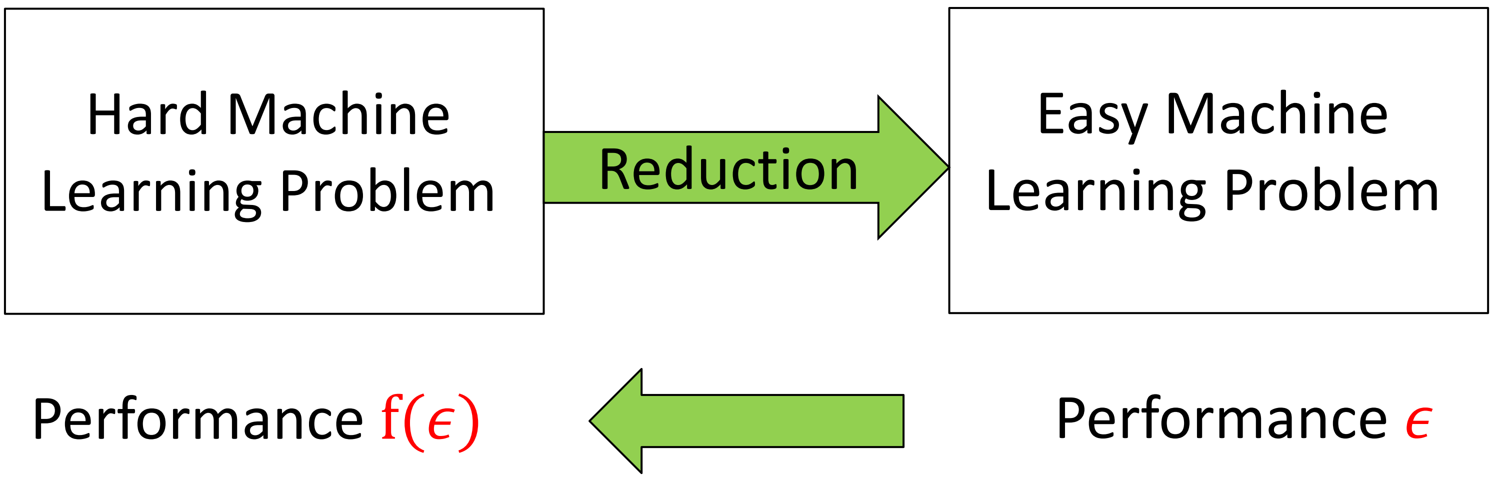
Classifier-based Structured Prediction

Classifier-based Structured Prediction

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



Aside: Reductions in Machine Learning



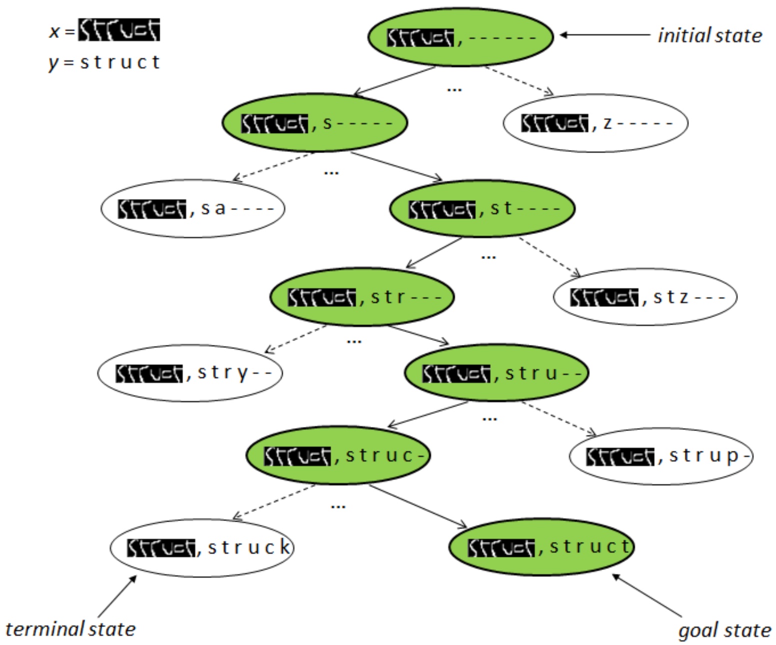
- Reduce complex problem to simpler problem(s)
- A better algorithm for simpler problem means a better algorithm for complex problem
- Composability, modularity, ease-of-implementation

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

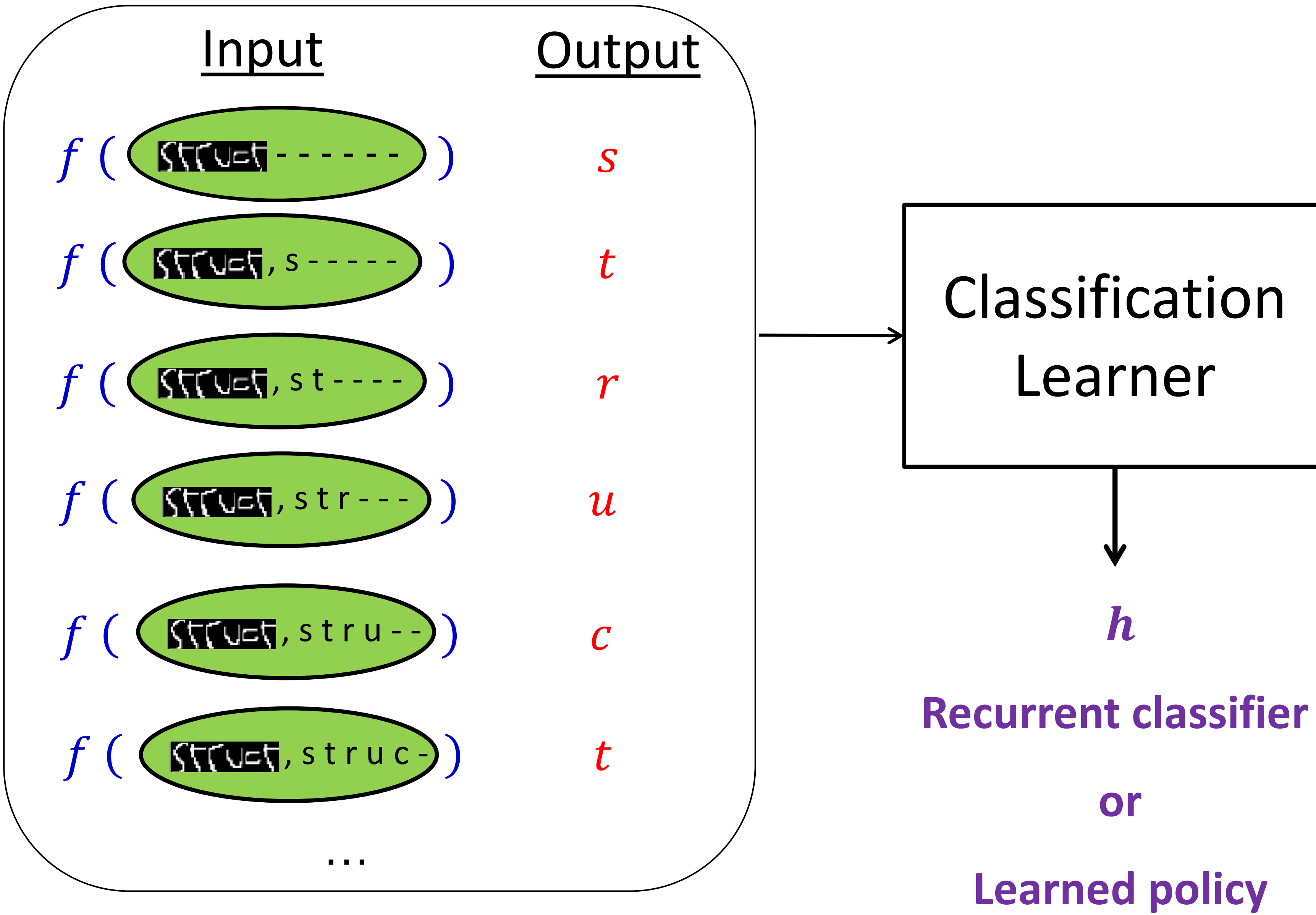
Exact Imitation: Classification examples

- For each training example

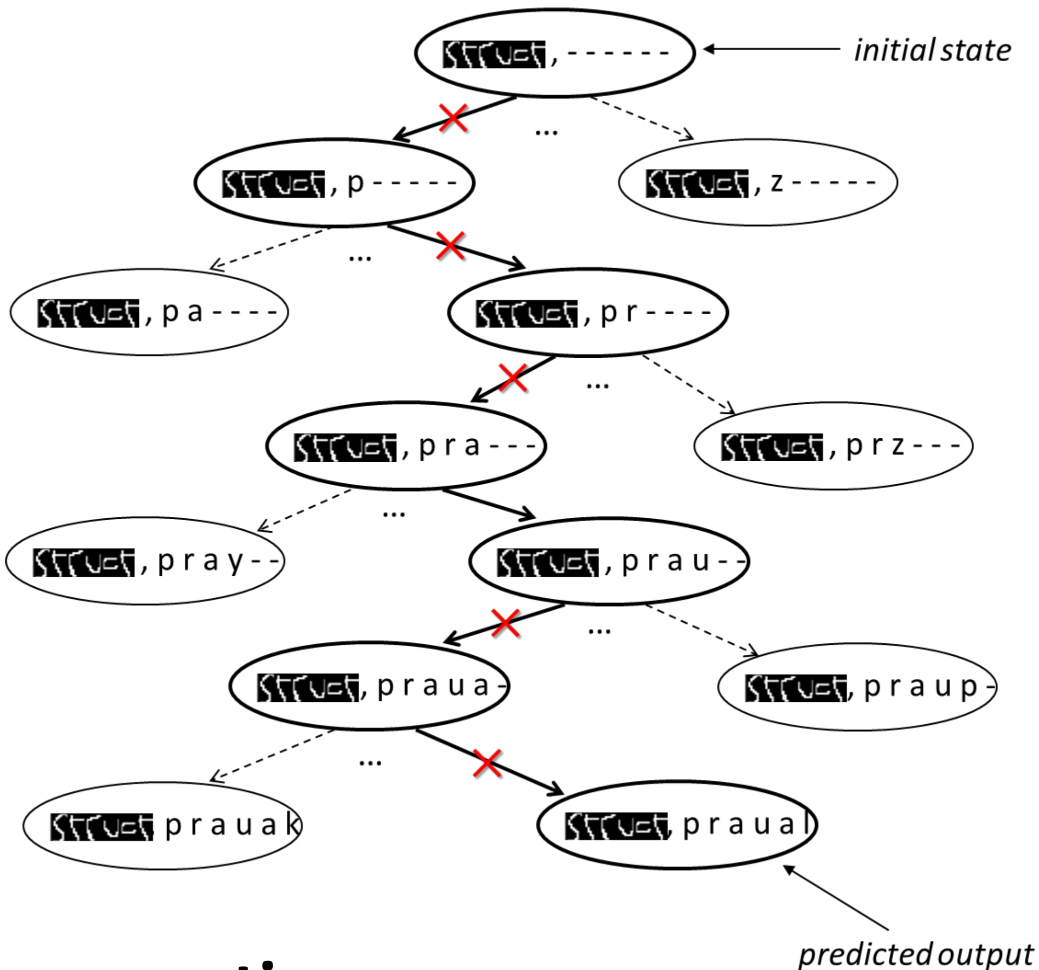


Input | Output

Exact Imitation: Classifier Learning



Learned Recurrent Classifier: Illustration



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

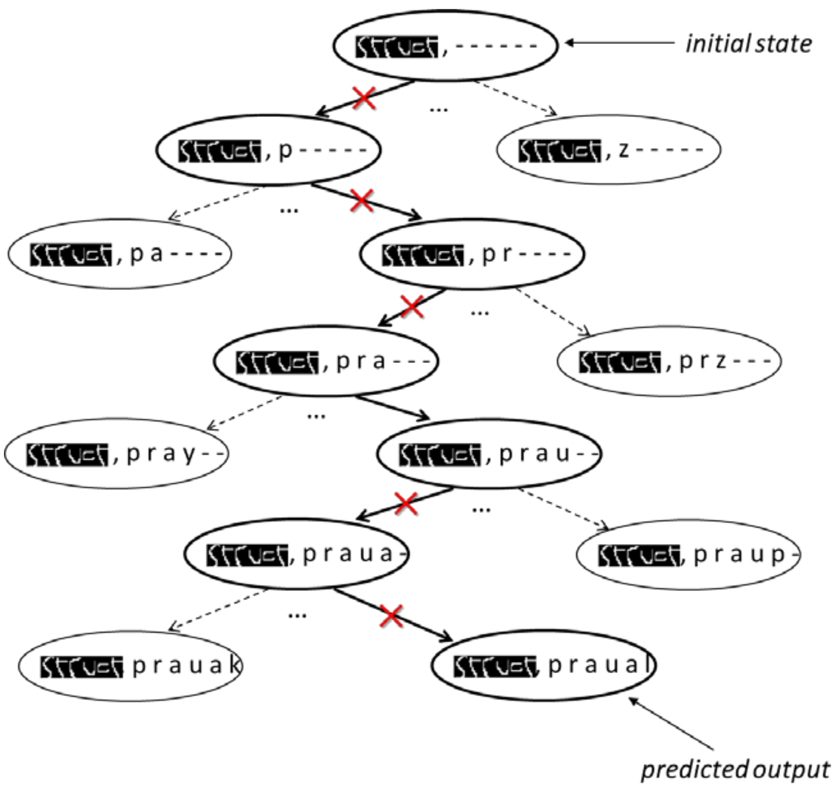
Recurrent Error

- 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]

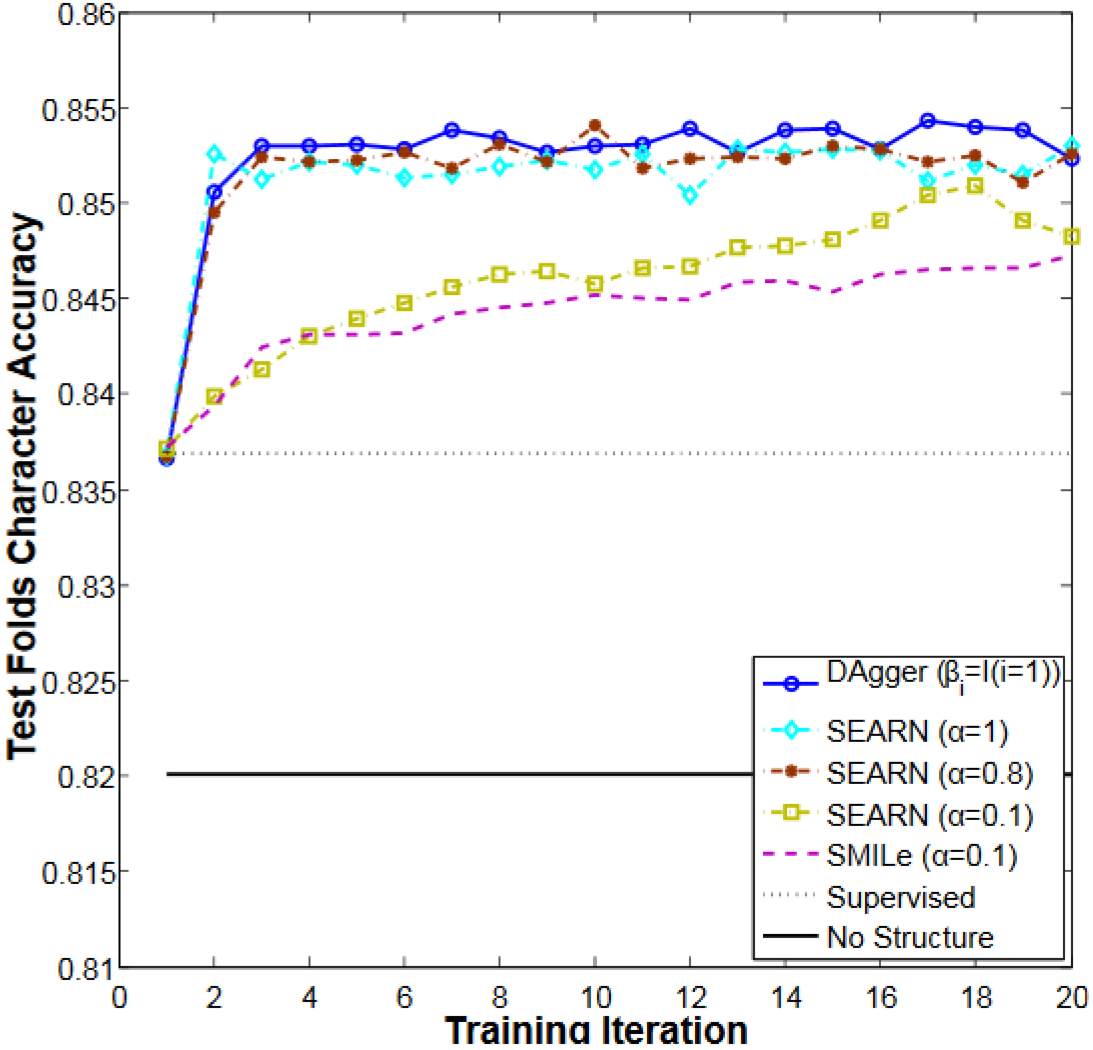
- Collect initial training set D of N trajectories from reference policy π^*

- 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]

Ordered vs. Unordered Search Space

- **Ordered Search Space**
  - Fixed ordering of decisions (e.g., left-to-right in sequences)
  - Classifier based structured prediction

- **Unordered Search Space**
  - Learner dynamically orders the decisions
  - **Easy-First approach**

Easy-First Approach for Structured Prediction

Easy-First Approach: Motivation

- **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

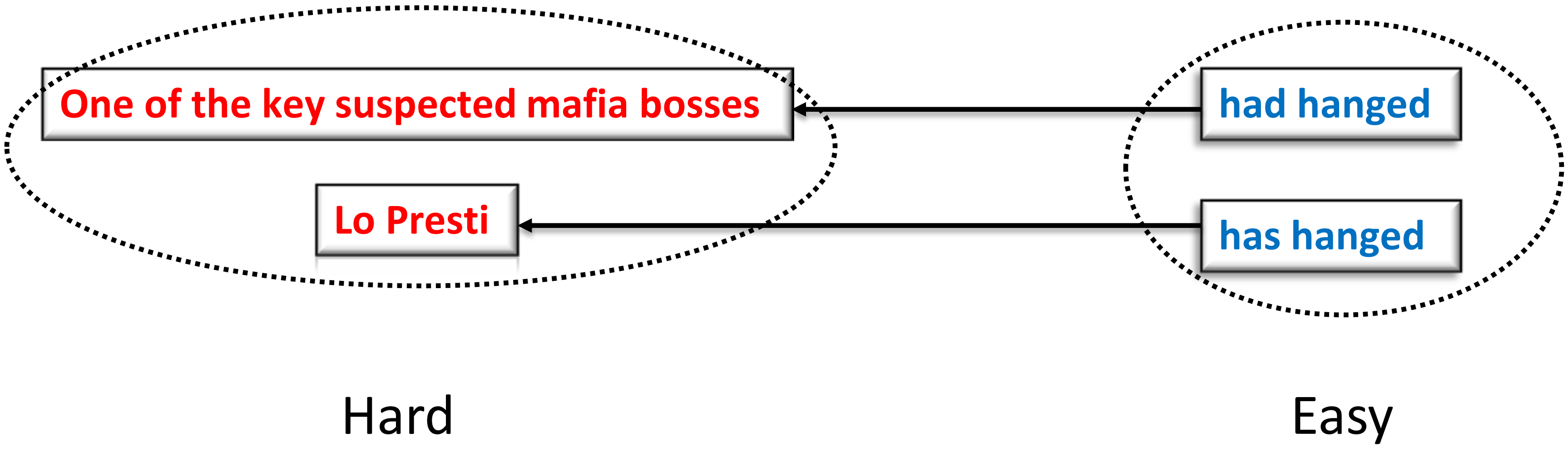
Example: Cross-Document Coreference

One of the key suspected mafia bosses arrested yesterday had hanged himself.

Doc 1

Police said Lo Presti has hanged himself.

Doc 2



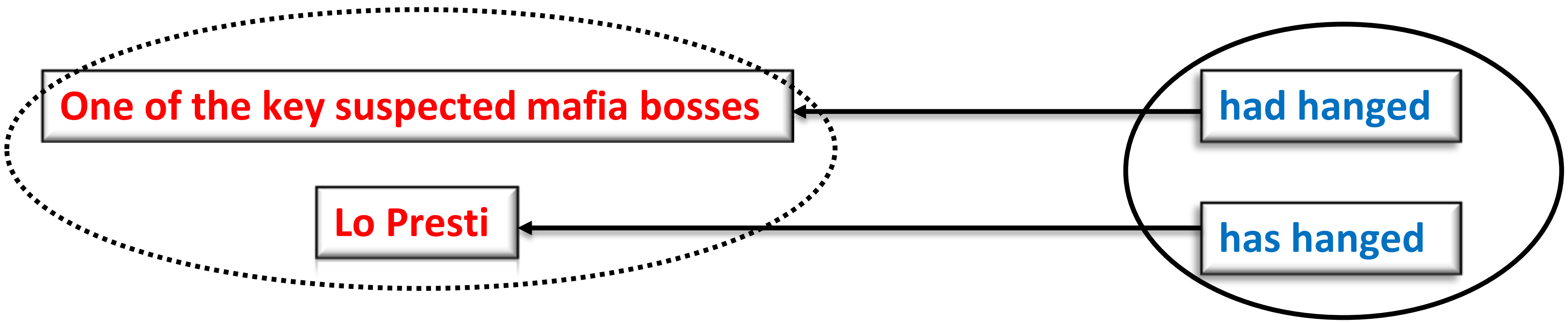
Example: Cross-Document Coreference

One of the key suspected mafia bosses arrested yesterday **had hanged** himself.

Doc 1

Police said **Lo Presti** **has hanged** himself.

Doc 2



- Once we decide that the two verbs are coreferent, the two noun mentions serve the same semantic role to the verb cluster
- Strong evidence for coreference

Easy-First Approach: Overview

- Consider a set of inter-dependent decisions in a sequential manner

- At each step, make the easiest decision first

- This allows us to accumulate more information to help resolve more challenging decisions later

Applications of Easy-First

- Cross-document joint entity and event co-reference
  - Lee et. al. EMNLP-CoNLL ’12
- Within-document co-reference Resolution
  - Stoyanov and Eisner, COLING’12
- Dependency parsing
  - Goldberg and Elhadad, HLT-NAACL’ 10

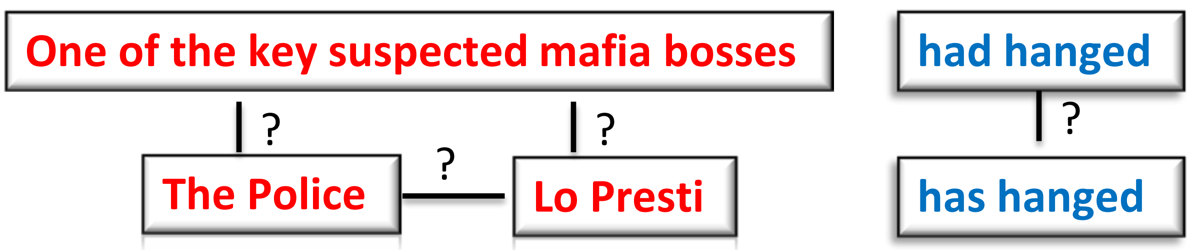
Easy-First Approach: Key Elements

- Search space
  - A state corresponds to a partial solution

Initial state: all mentions and verbs are in separate clusters

Easy-First Approach: Key Elements

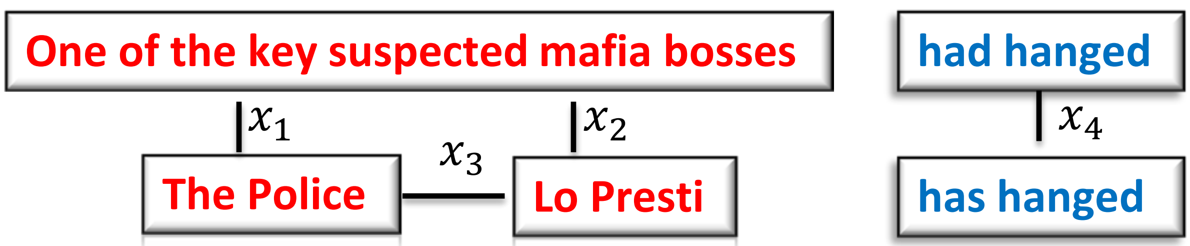
- Search space
  - A state corresponds to a partial solution
  - In each state, we consider a set of fixed possible actions



Four possible merge actions

Easy-First Approach: Key Elements

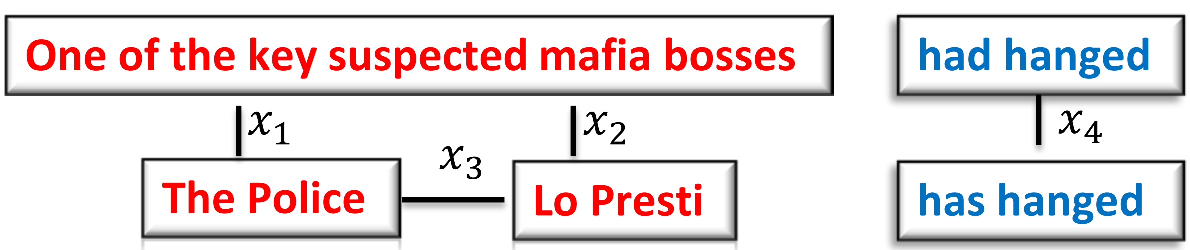
- Search space
  - A state corresponds to a partial solution
  - In each state, we consider a set of fixed possible actions
  - Each action is described by a feature vector $x \in R^d$



Four possible merge actions

Easy-First Approach: Key Elements

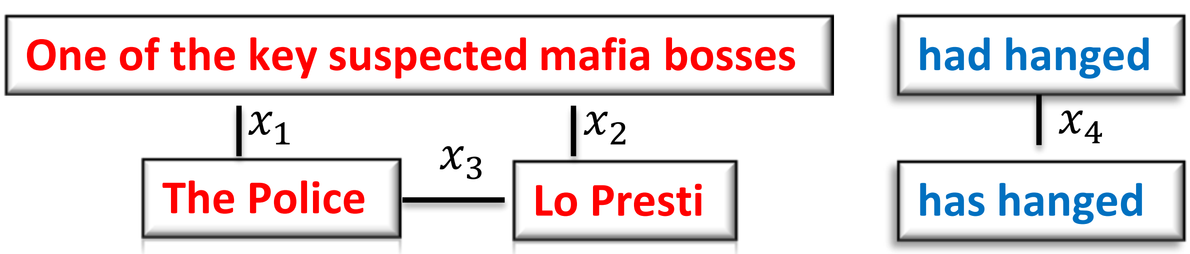
- Search space
  - A state corresponds to a partial solution
  - In each state, we consider a set of fixed possible actions
  - Each action is described by a feature vector $x \in R^d$
  - An action is defined to be ***good*** if it leads to an improved state



$x_2, x_4 \in G$ (good actions); $x_1, x_3 \in B$(bad actions)

Easy-First Approach: Key Elements

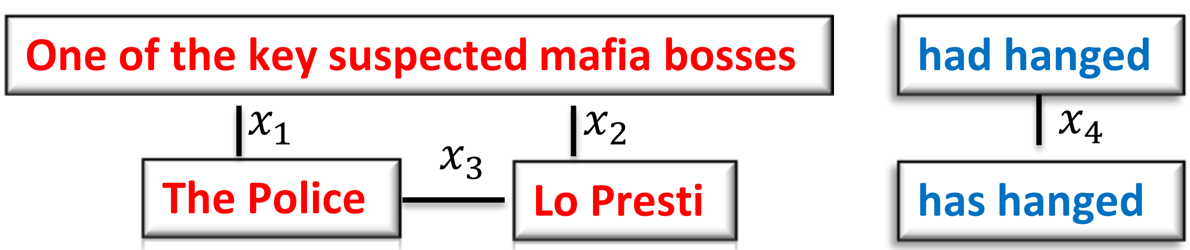
- Search space
- Scoring function $f: R^d \rightarrow R$
  - e.g., $f(x) = w \cdot x$
  - In each state, evaluate all possible actions



$f(x_1) = 0.05$ $f(x_2) = 0.08$ $f(x_3) = 0.057$ $f(x_4) = 0.75$

Easy-First Approach: Key Elements

- Search space
- Scoring function $f: R^d \rightarrow R$
  - e.g., $f(x) = w \cdot x$
  - In each state, evaluate all possible actions
  - Take the highest scoring action (easiest)



$f(x_1) = 0.05$ $f(x_2) = 0.08$ $f(x_3) = 0.057$ $f(x_4) = 0.75$

Scoring Function Learning

Possible goal: learn a scoring function such that:

in every state ~~all good actions~~ are ranked higher than all bad actions

A better goal: learn a scoring function such that in every state ***a good action*** is ranked higher than all bad actions

Alternate Methods

- In a training step, if the highest scoring action is bad, perform weight update

- Different update approaches
  - Best (highest scoring) good vs. best (highest scoring) bad
  - Average good vs. average bad

Issue: they do not directly optimize toward our goal!

Optimization Objective for Update

- **Goal:** find a linear function such that it ranks one good action higher than all bad actions
  – This can be achieved by a set of constraints

$$\max_{g \in G} w \cdot x_g > w \cdot x_b + 1$$

for all $b \in B$

- **Optimization Objective:**
  - Use hinge loss to capture the constraints
  - Regularization to avoid overly aggressive update

$$\boxed{\operatorname*{argmin}_{w} \frac{1}{|B|} \sum_{b \in B} \left(1 - \max_{g \in G} w \cdot x_g + w \cdot x_b\right)_+} + \boxed{\lambda \left\| w - w_c \right\|^2}$$

Optimization: Majorization-Minimization

[Xie et al., 2015]

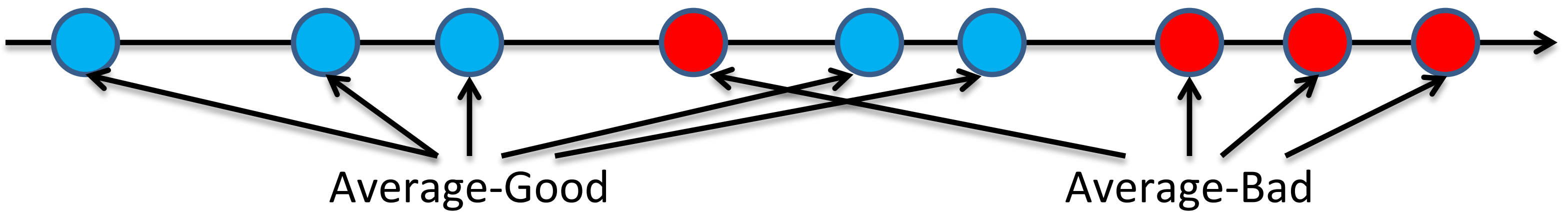
$$\boxed{\operatorname*{argmin}_{w} \frac{1}{|B|} \sum_{b \in B} \left(1 - \max_{g \in G} w \cdot x_g + w \cdot x_b\right)_+} + \boxed{\lambda \left\| w - w_c \right\|^2}$$

- It is non-convex
- Can be solved using a Majorization-Minimization (MM) algorithm to get local optima solution
- **In each MM iteration:**
  - Let x_g^* be the current highest scoring good action
  - Solve following convex objective (via subgradient descent):

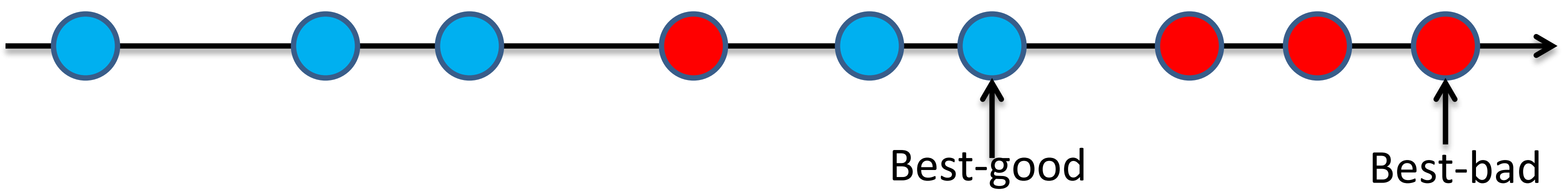
$$\operatorname*{argmin}_{w} \frac{1}{|B|} \sum_{b \in B} \left(1 - \underbrace{\cancel{\max_{g \in G} w \cdot x_g}}_{\boxed{w \cdot x_g^*}} + w \cdot x_b\right)_+ + \lambda \left\| w - w_c \right\|^2$$

Contrast with Alternate Methods

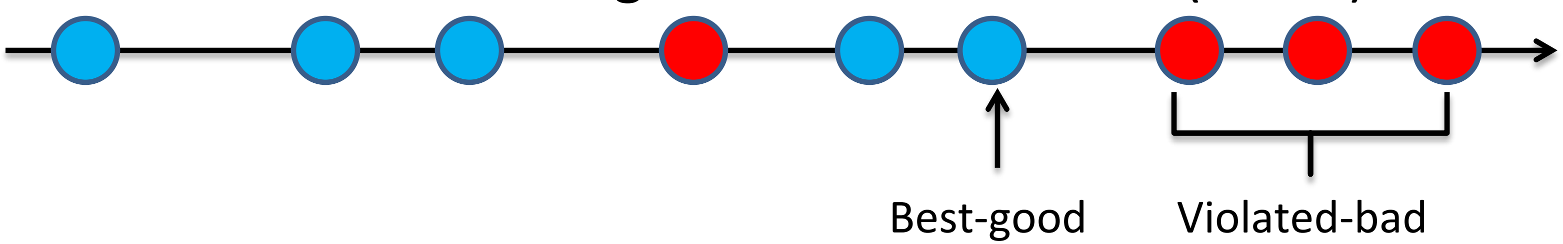
- Average-good vs. average-bad (AGAB) [Daume et al., 2005], [Xu et al., 2009]



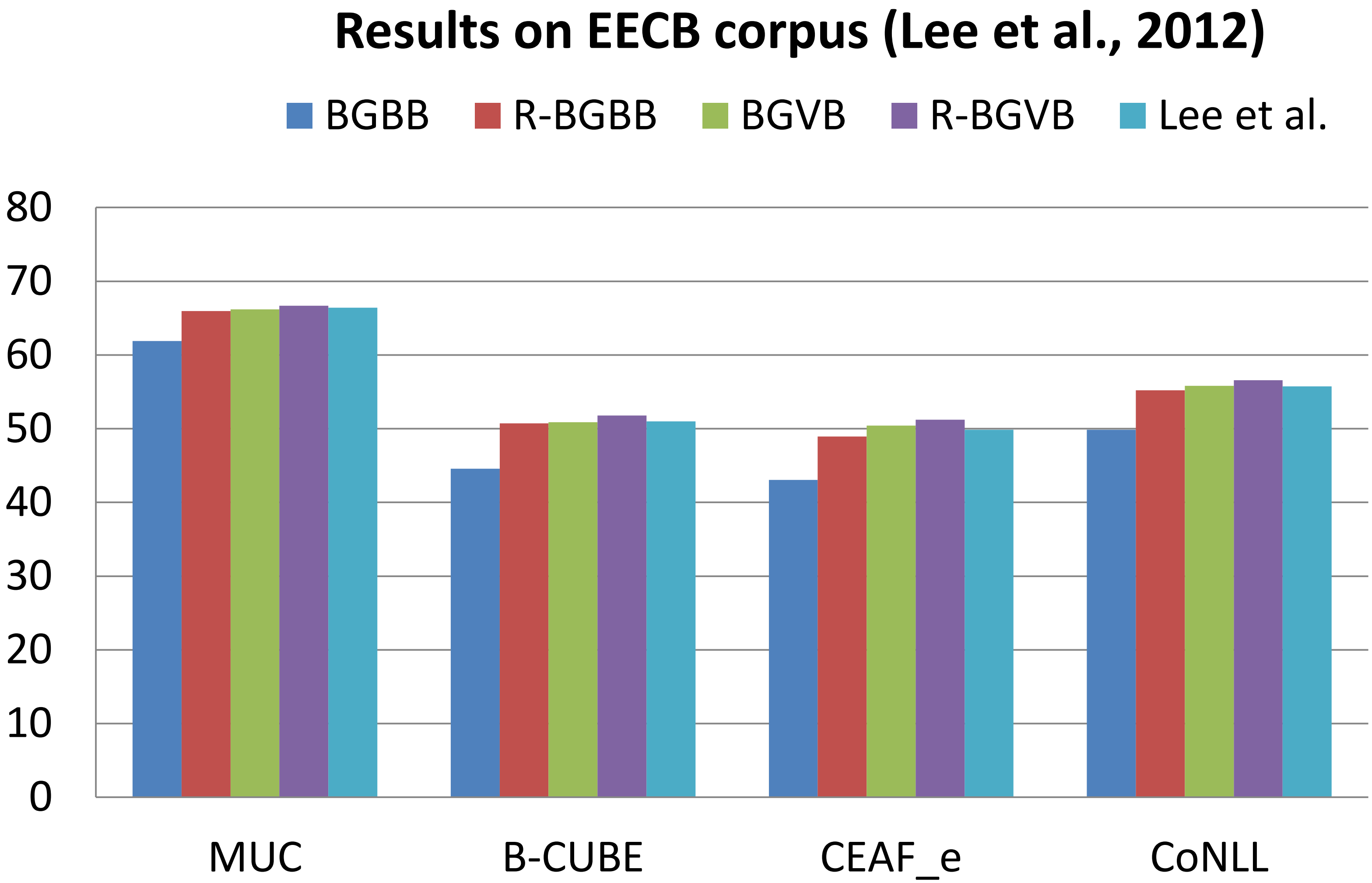
- Best-good vs. best-bad (BGBB) [Goldberg et al., 2010], [Stoyanov et al., 2012]



- Current method: Best-good vs. violated-bad (BGVB) [Xie et al., 2015]

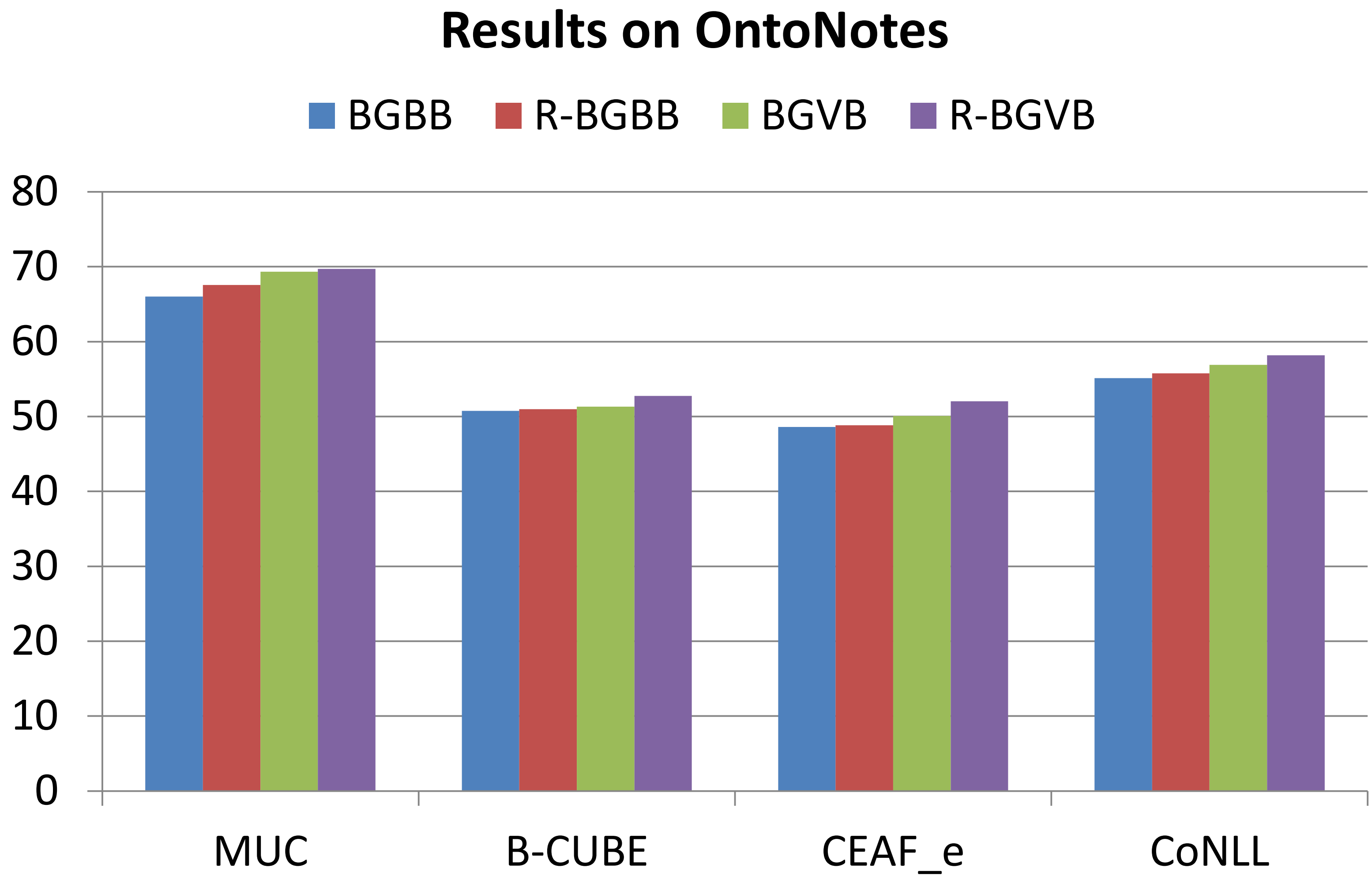


Experiment I: Cross-document entity and event Coreference



[Xie et al., 2015]

Experiment I: Within document Coreference



[Xie et al., 2015]

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

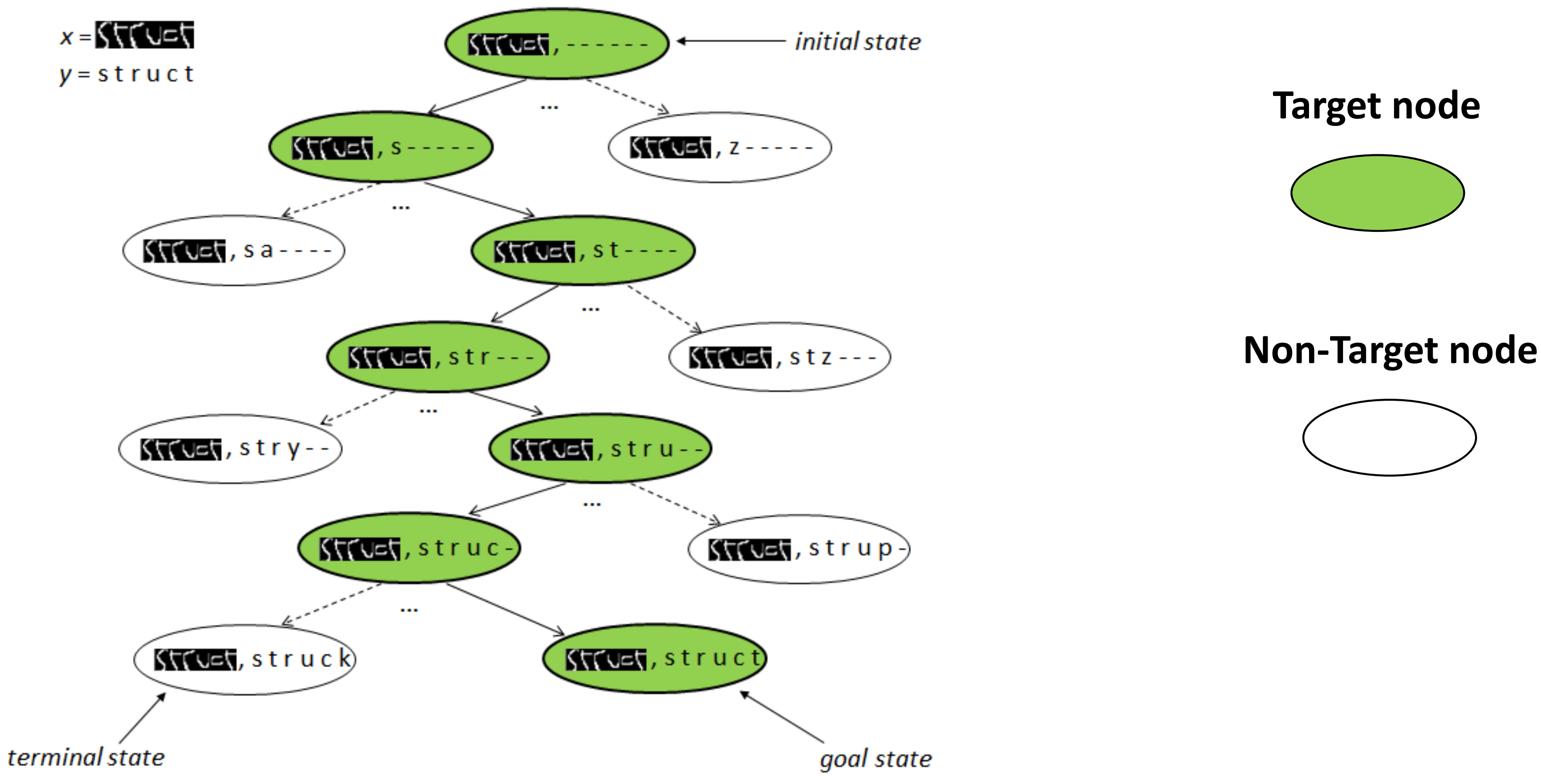
Part 5: Control Knowledge Learning: Beam Search Methods

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)

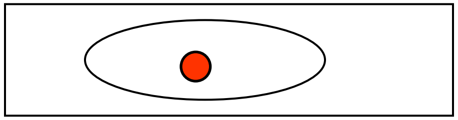
Beam Search Framework: Key Elements

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

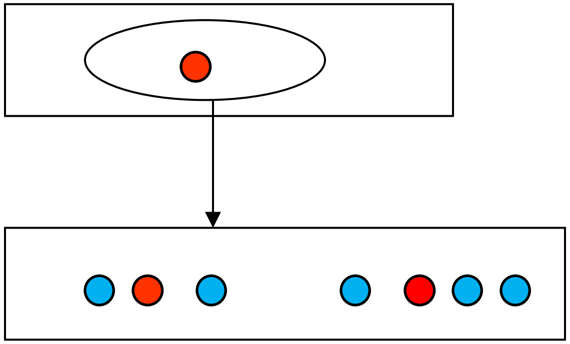


- Represent heuristic function as a linear function
  - $H(n) = w \cdot \psi(n)$, where $\psi(n)$ stands for features of node n

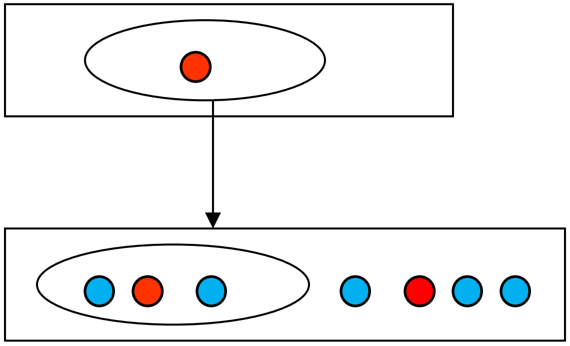
Beam Search: Illustration



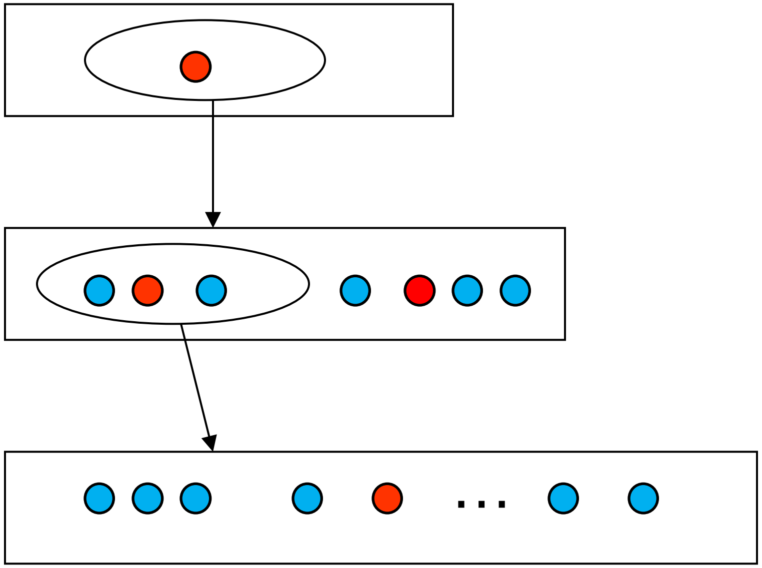
Beam Search: Illustration



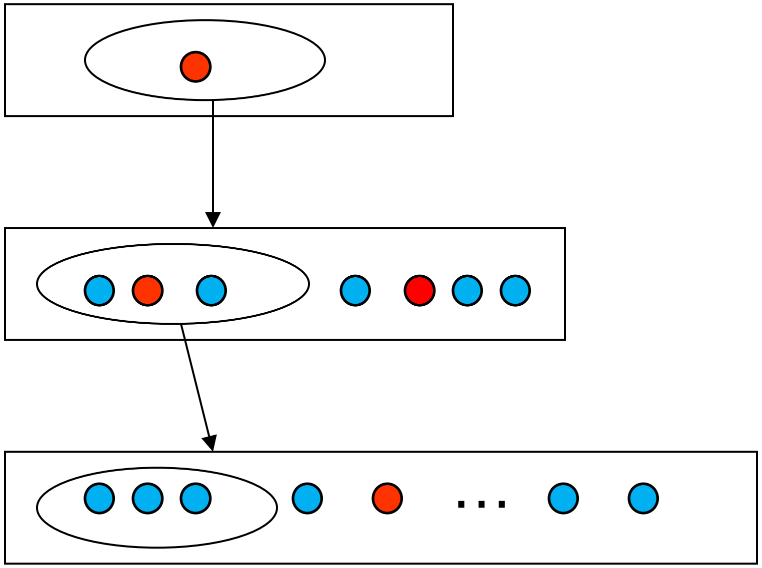
Beam Search: Illustration



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: Learning Instantiations

- Early update

- Max-violation update

- Learning as Search Optimization (LaSO)

Beam Search Framework: Early Update

- **Search error:** NO target node in the beam
  - We cannot reach the goal node (correct structured output)
- **Weight update:** standard structured perceptron
  - Score of correct output > score of bad output
- **Beam update:** reset beam with initial state OR discontinue search

Beam Search Framework: Early Update

- **repeat**
  - For every training example (x, y)
    - Perform search with current heuristic (weights)
    - If search error, update weights
    - Reset beam with initial state
    - (Dis)continue search
- **until** *convergence* or *max. iterations*

Beam Search Framework: Learning Instantiations

- Early update
- Max-violation update
- Learning as Search Optimization (LaSO)

Beam Search Framework: Max-Violation Update

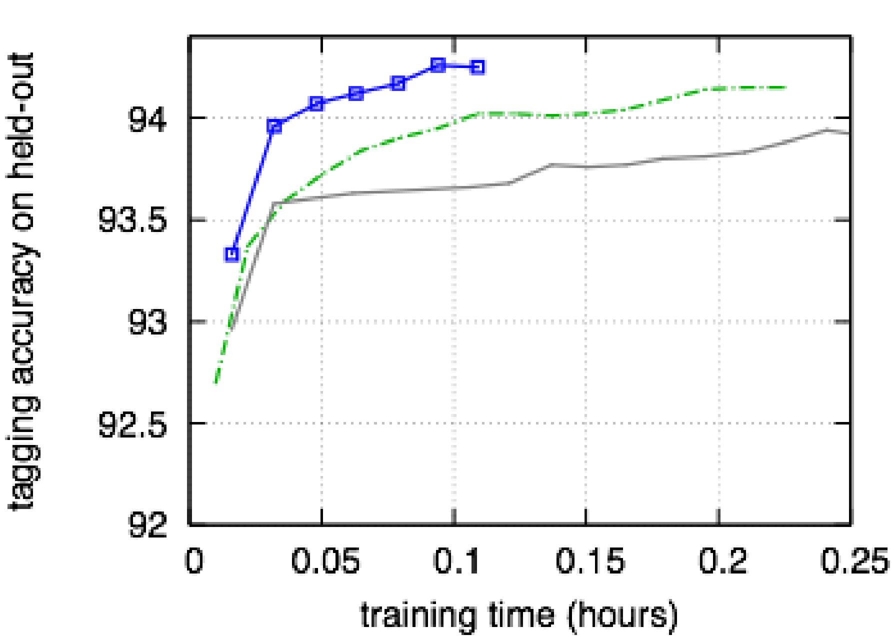
- Improves on the drawback of Early update
  - Slow learning: learns from only earliest mistake
- **Max-Violation fix**
  - Consider worst-mistake (maximum violation) instead of earliest-mistake for the weight update
  - More useful training data
  - Converges faster than early update

POS Tagging: Max-violation vs. Early vs. Standard

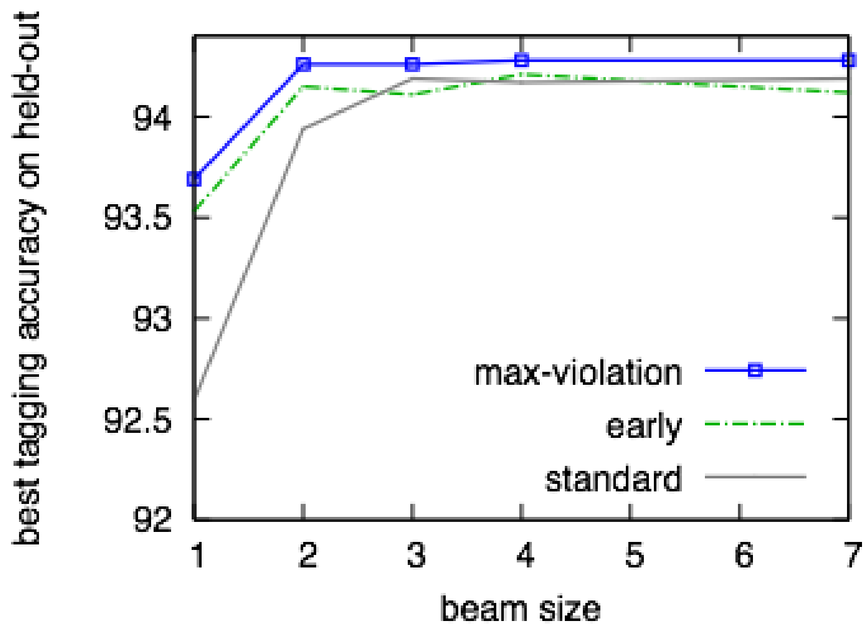
- **Early and Max-violation >> Standard at small beams**
  - Advantage shrinks as beam size increases
  - Max-violation converges faster than Early (and slightly better)

Source: Huang et al., 2012

Beam =2



Best accuracy vs. beam size



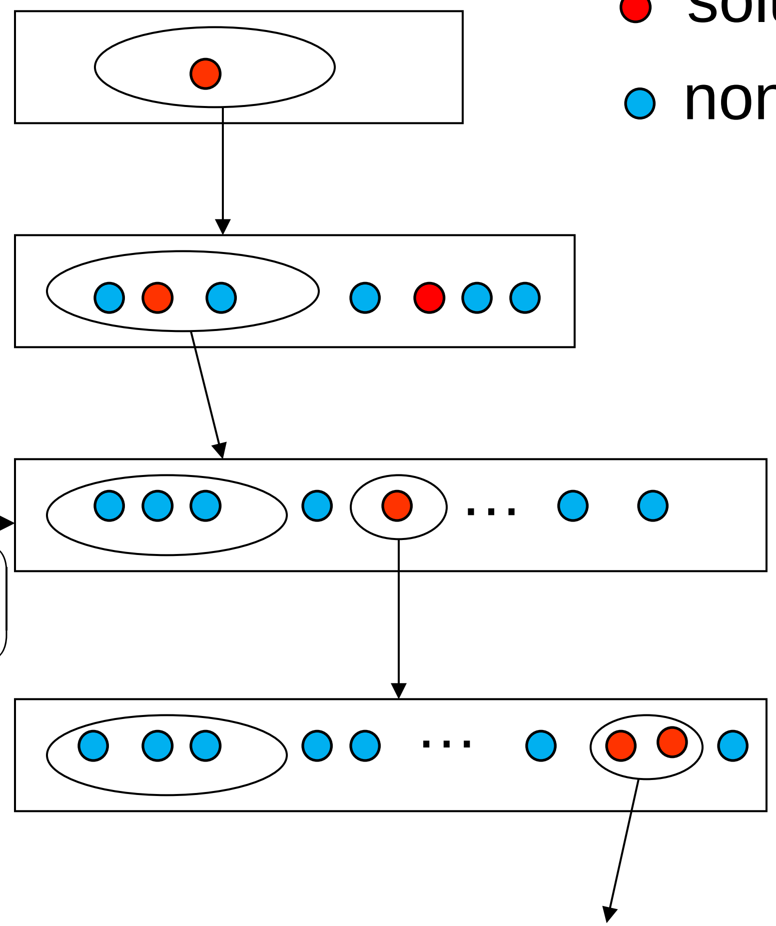
Beam Search Framework: LaSO

- **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 (red)
- non-solution node (blue)



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

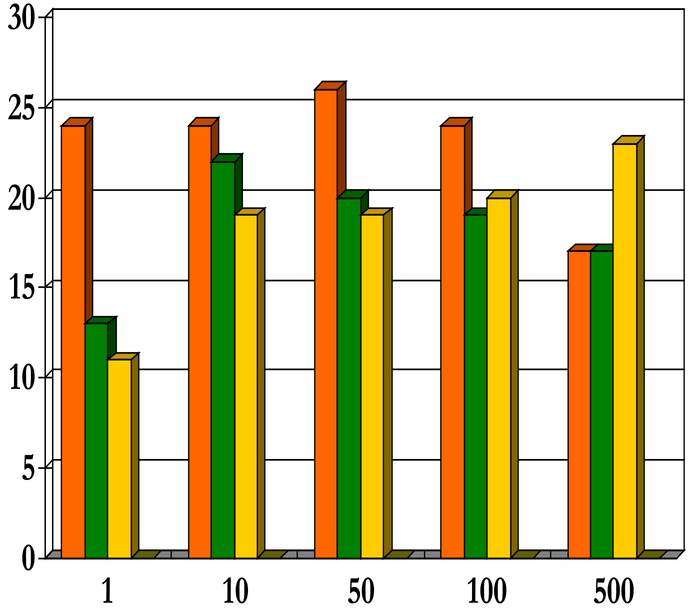
- **Interesting convergence result**
  - Mistake bound depends on the beam width
  - Formalizes the intuition that learning becomes easier as we increase the beam width (increase the amount of search)
  - First formal result of this kind

LaSO: Example Planning Results

- Blocksworld
  - 30 testing problems
  - Trained with beam width 10
  - Features: RPL heuristic and features induced in prior work

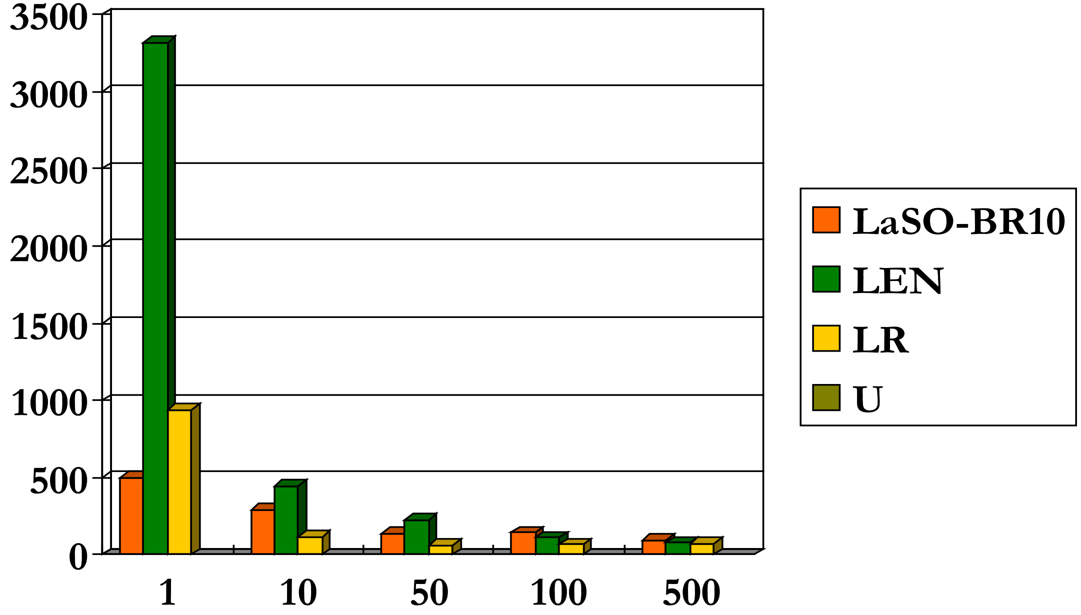
Source: Xu et al., 2009

Problems solved



Beam width

Median plan length



Beam width

Part 6: 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
- Experiments and Results
- 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
- Experiments and Results
- 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

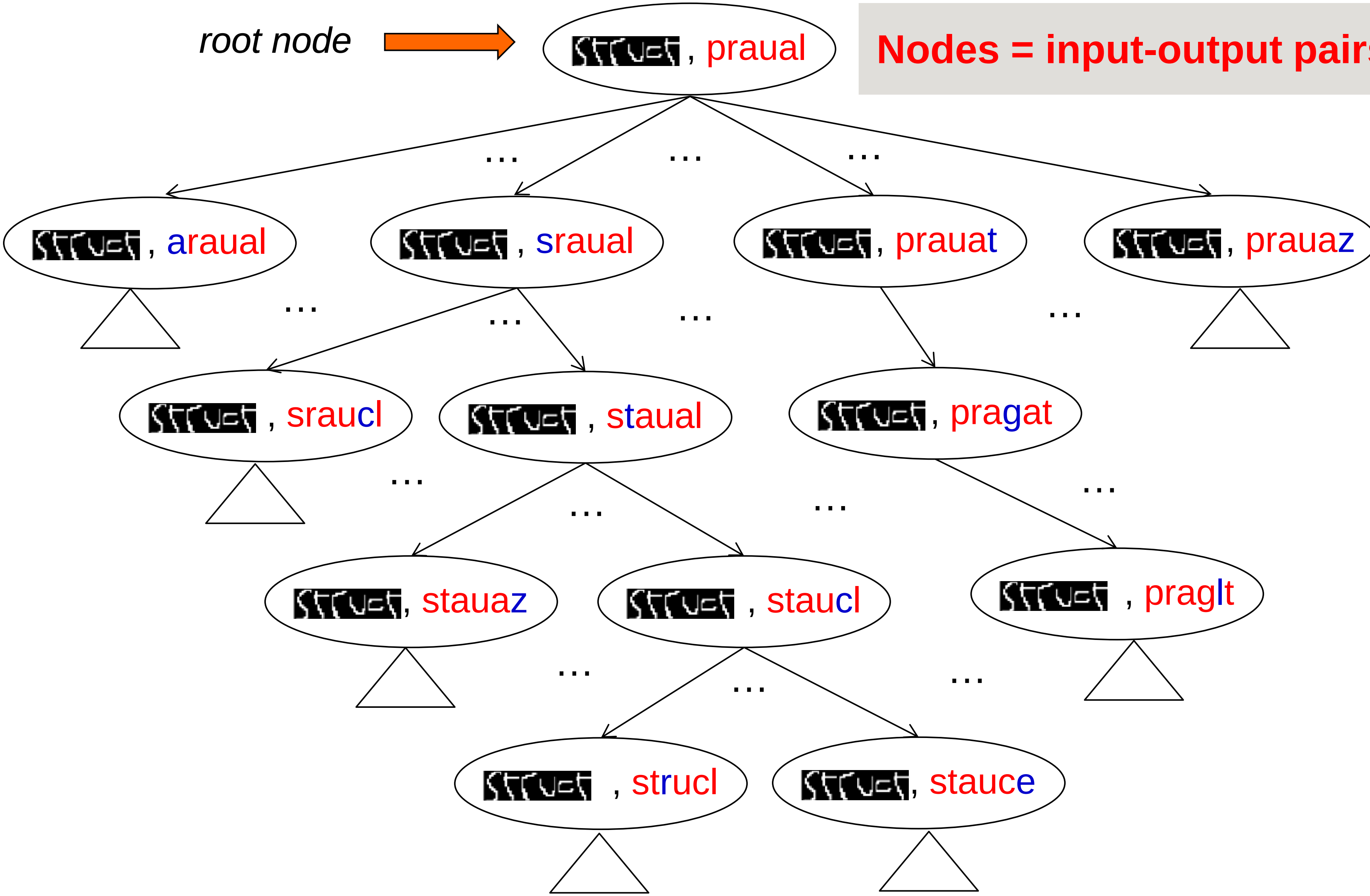
Our approach:

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

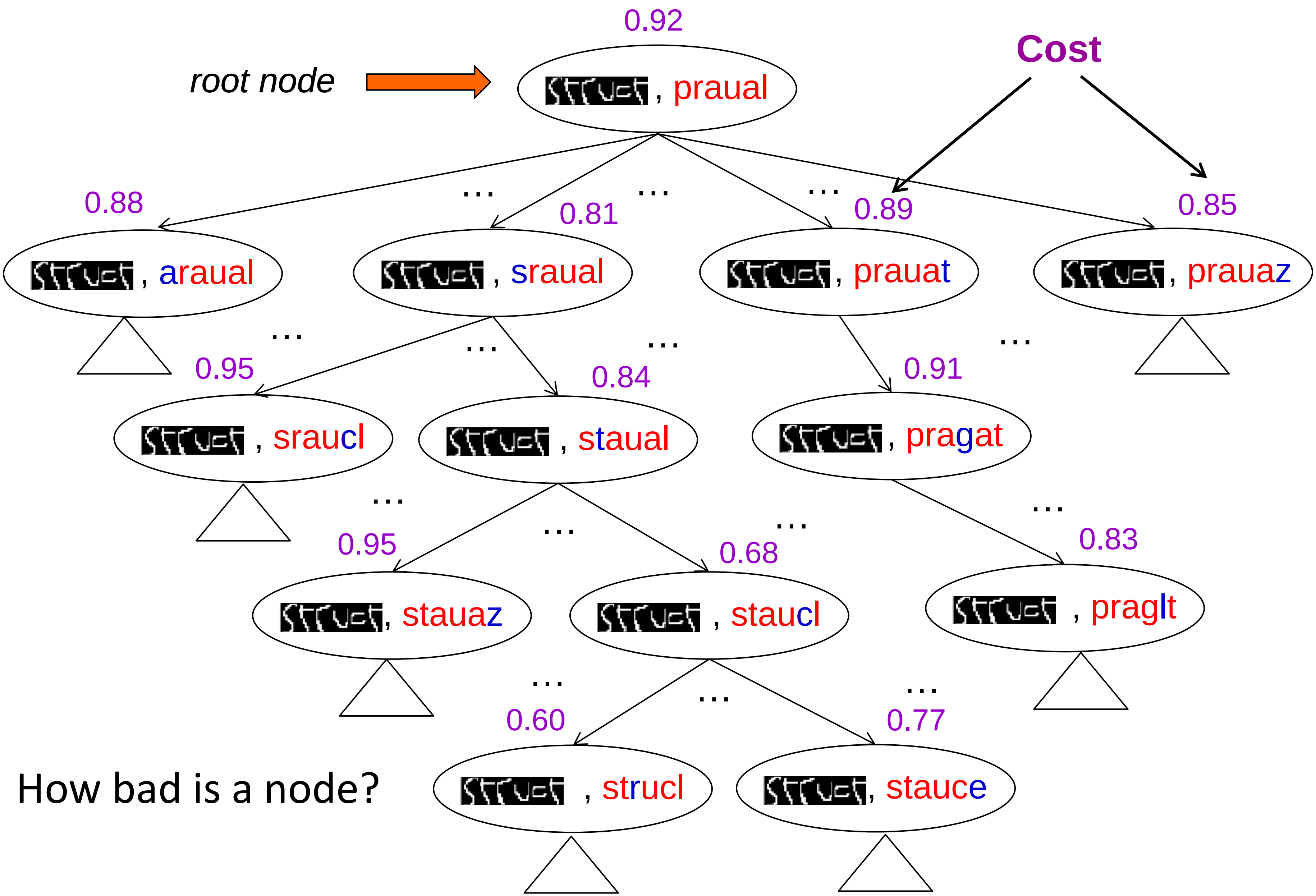
- **Key Ingredients:**
  - Define a search space over structured outputs
  - Learn a cost function $\boldsymbol{C}$ to score potential outputs
  - Use a search algorithm to find low cost outputs
  - Learn a heuristic function $\boldsymbol{H}$ to make search efficient

HC-Search Illustration: Search Space

Nodes = input-output pairs

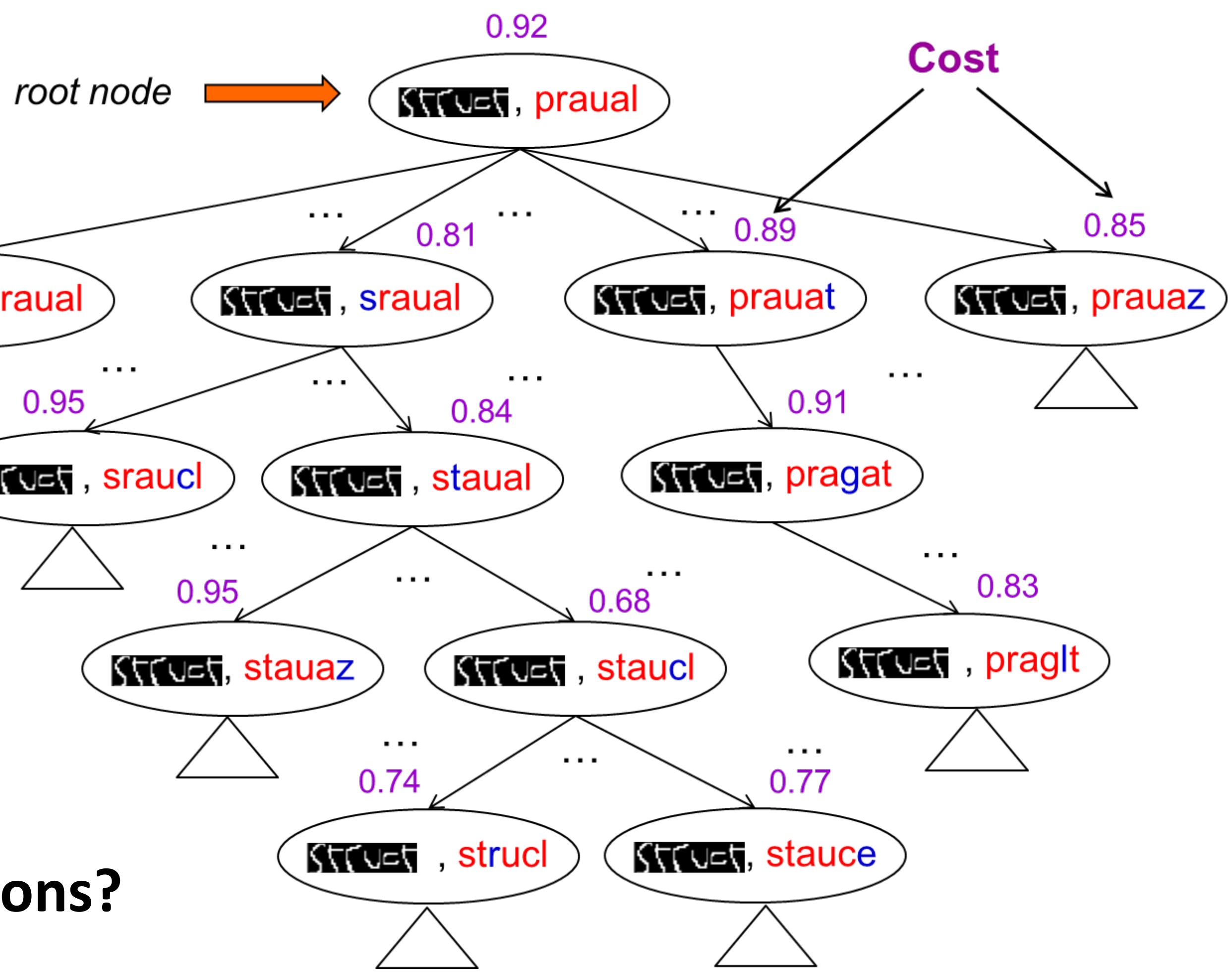


HC-Search Illustration: Cost Function



How bad is a node?

HC-Search Illustration: Making Predictions



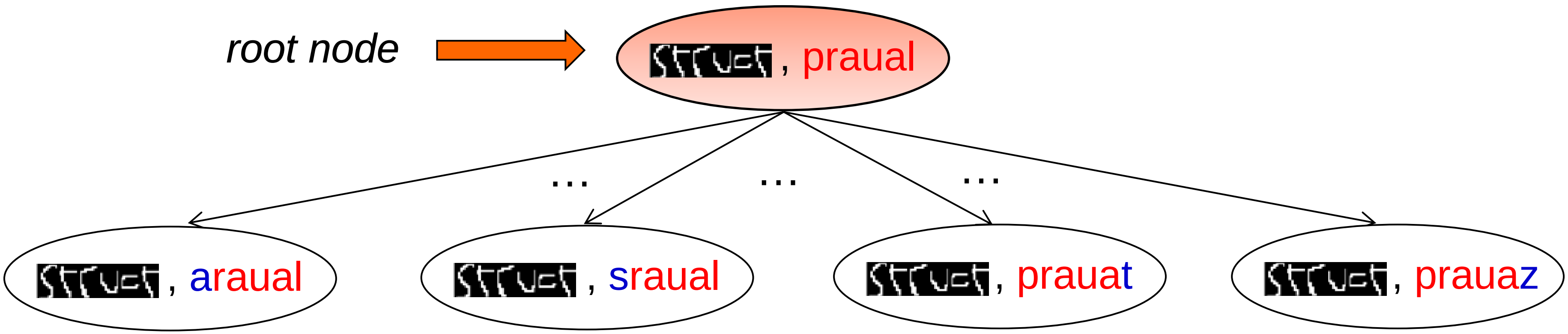
Assume we have a good cost function.

How to make predictions?

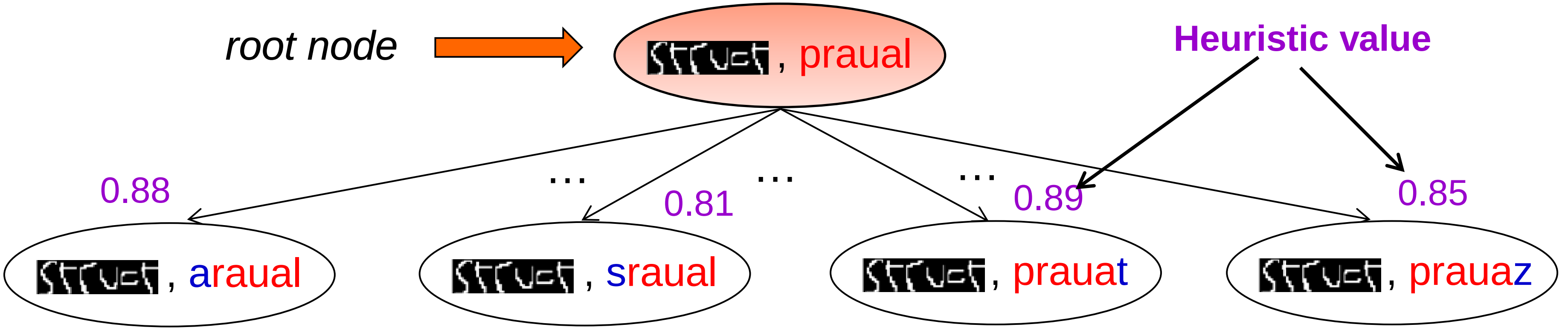
HC-Search Illustration: Greedy Search

root node →  , praual

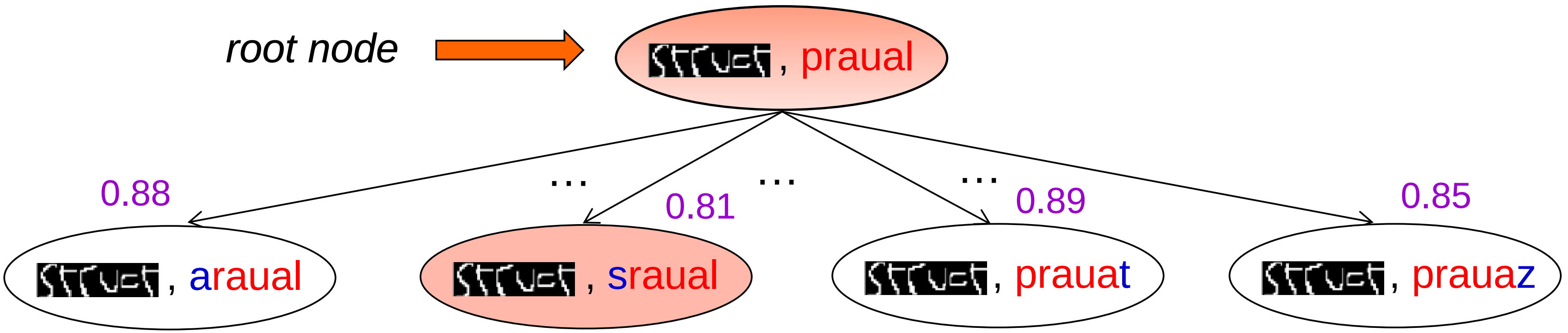
HC-Search Illustration: Greedy Search



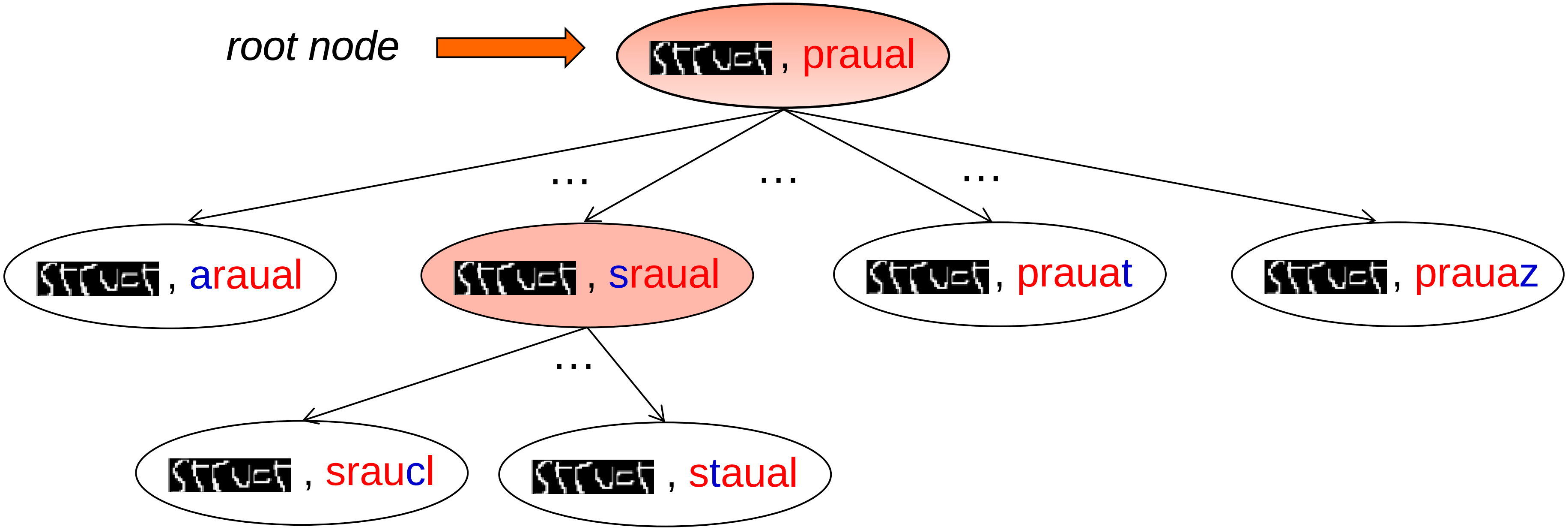
HC-Search Illustration: Greedy Search



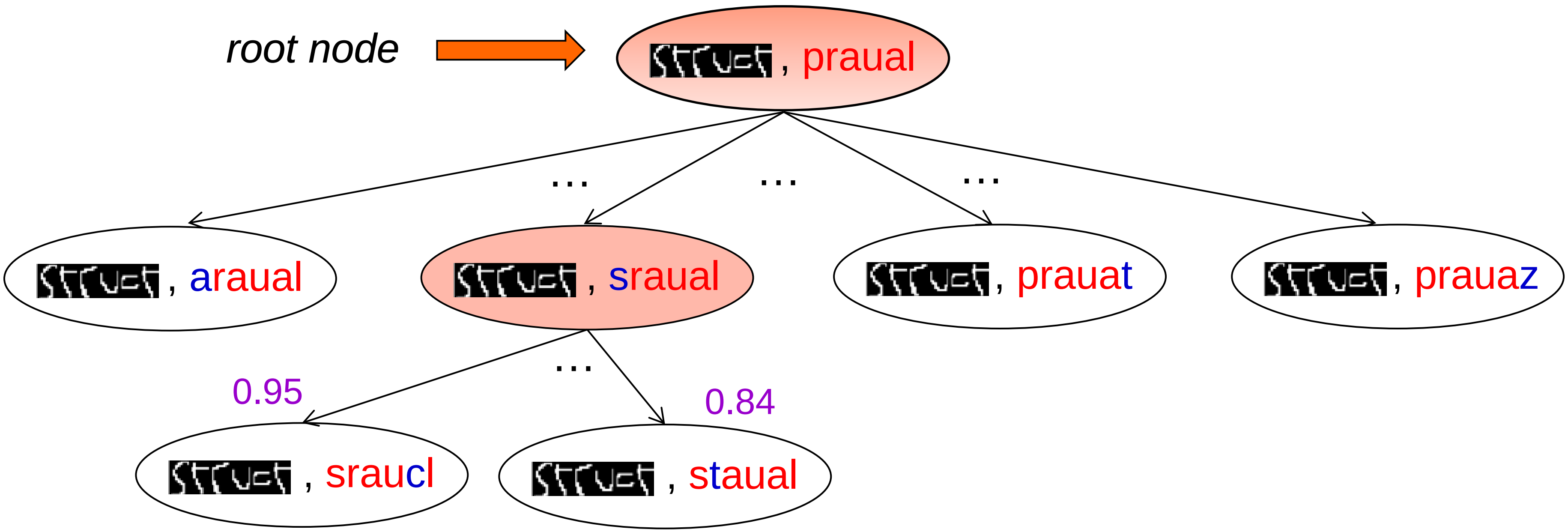
HC-Search Illustration: Greedy Search



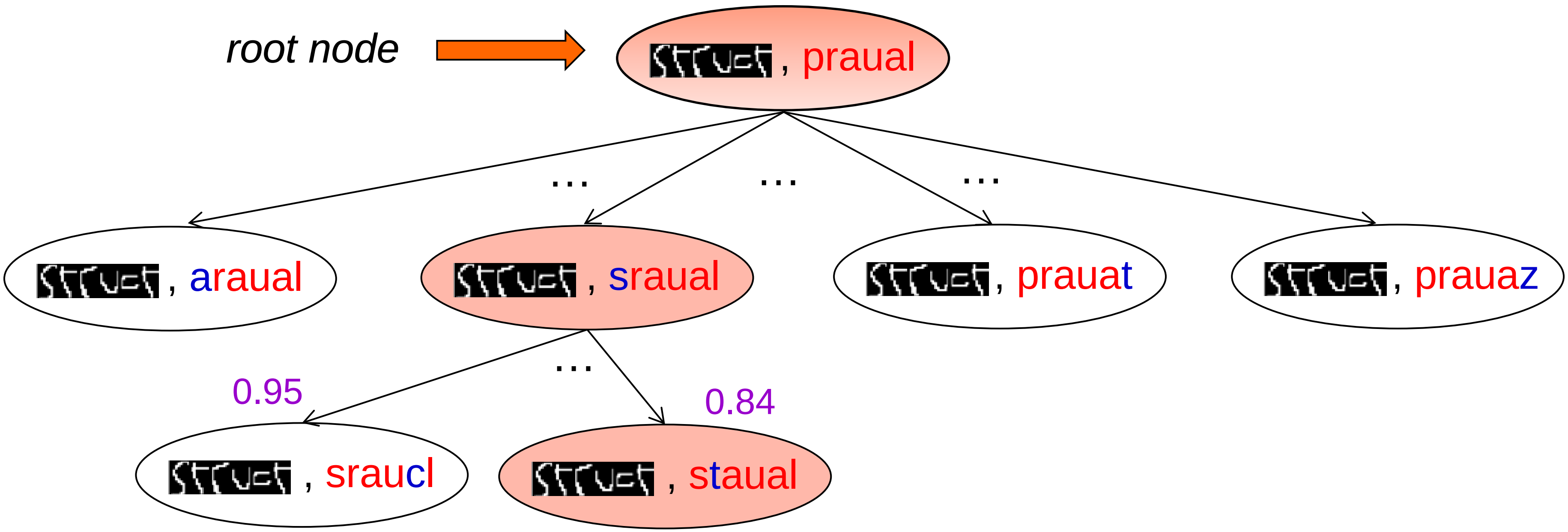
HC-Search Illustration: Greedy Search



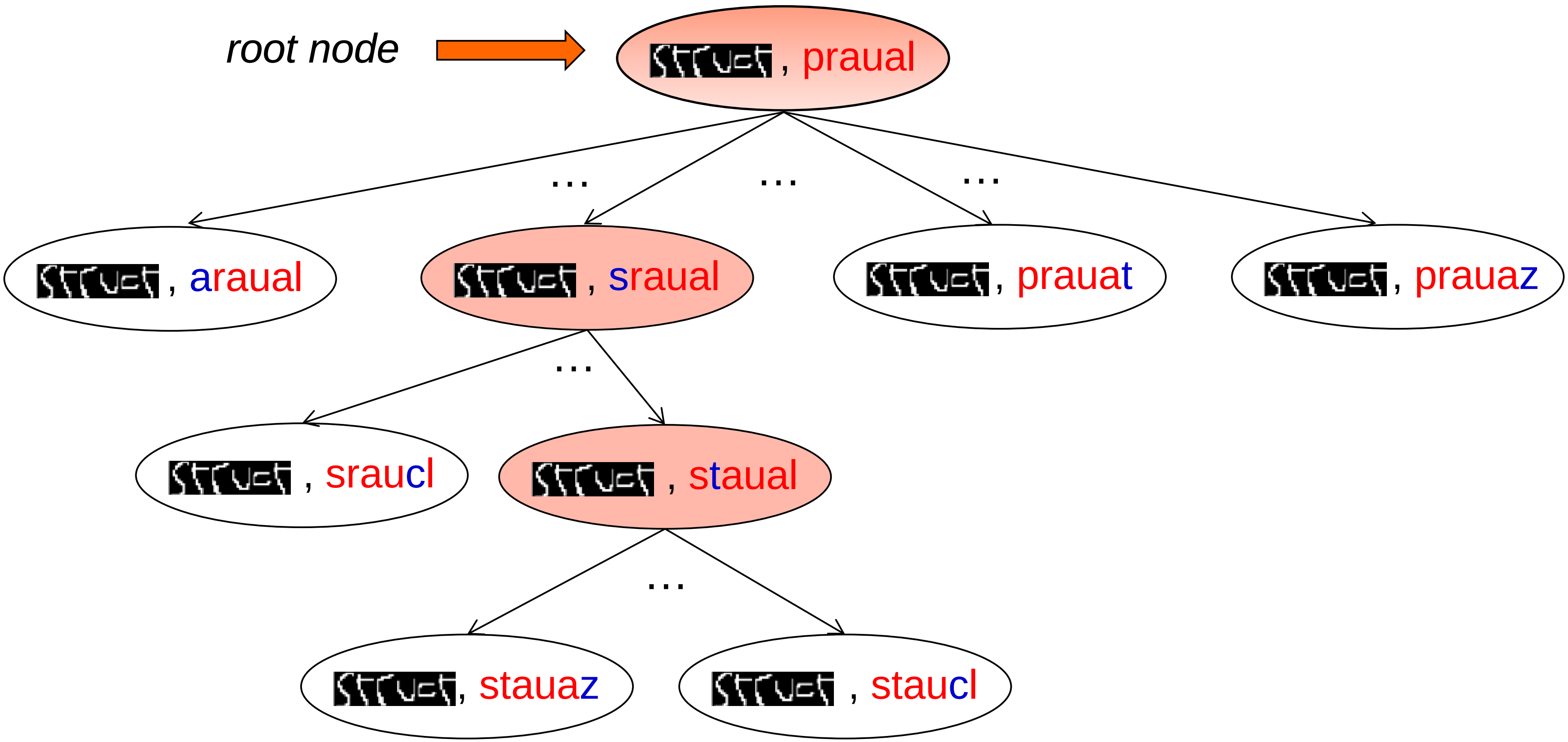
HC-Search Illustration: Greedy Search



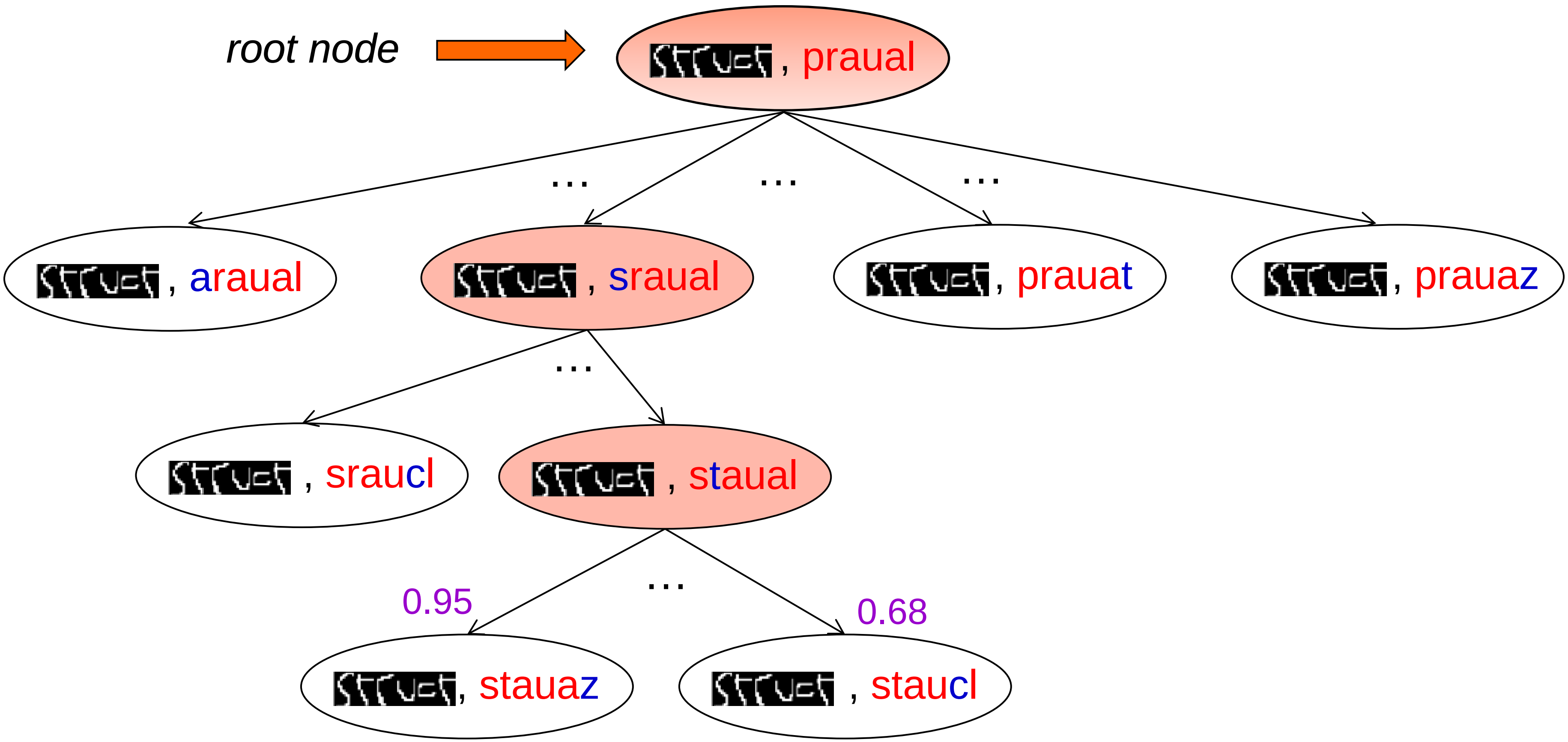
HC-Search Illustration: Greedy Search



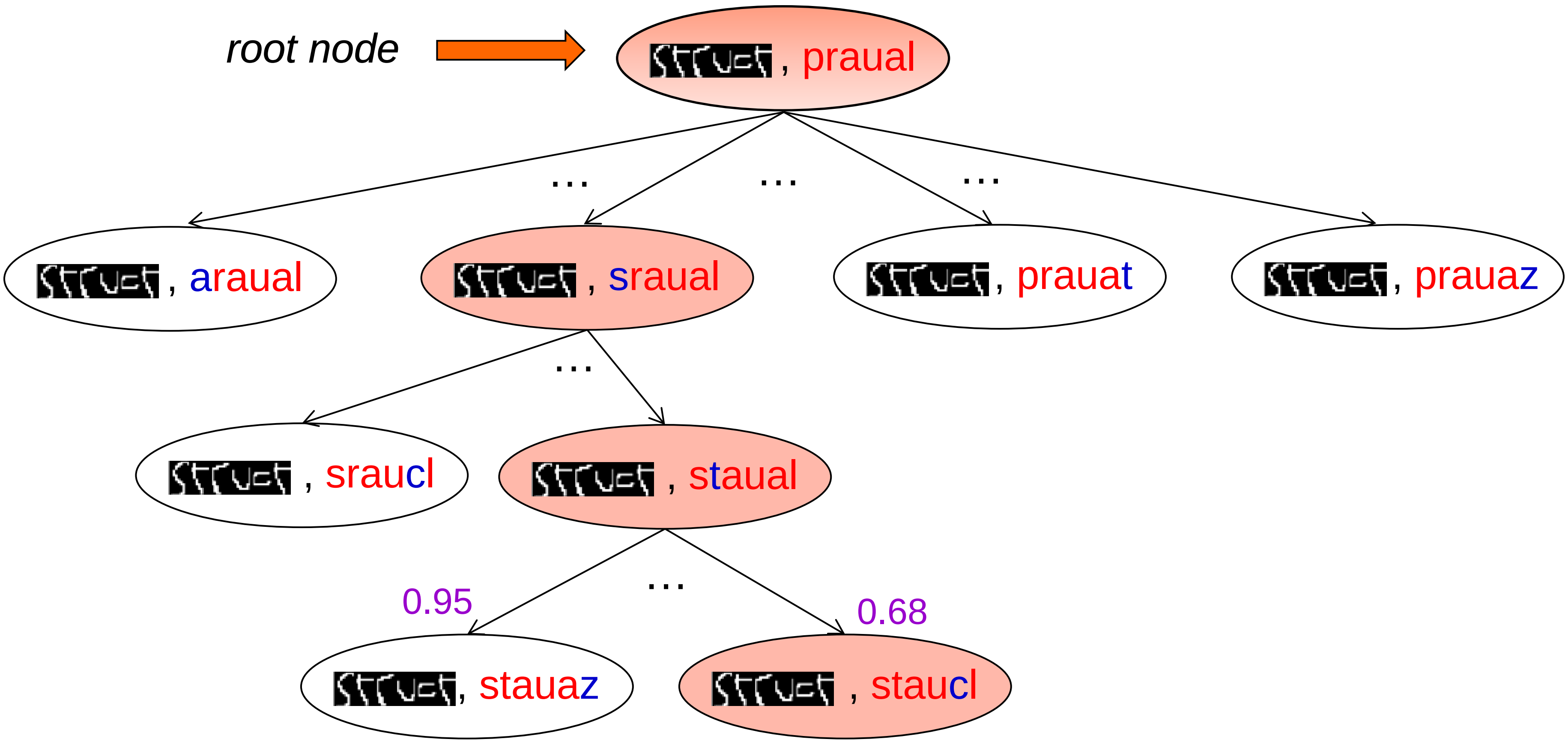
HC-Search Illustration: Greedy Search



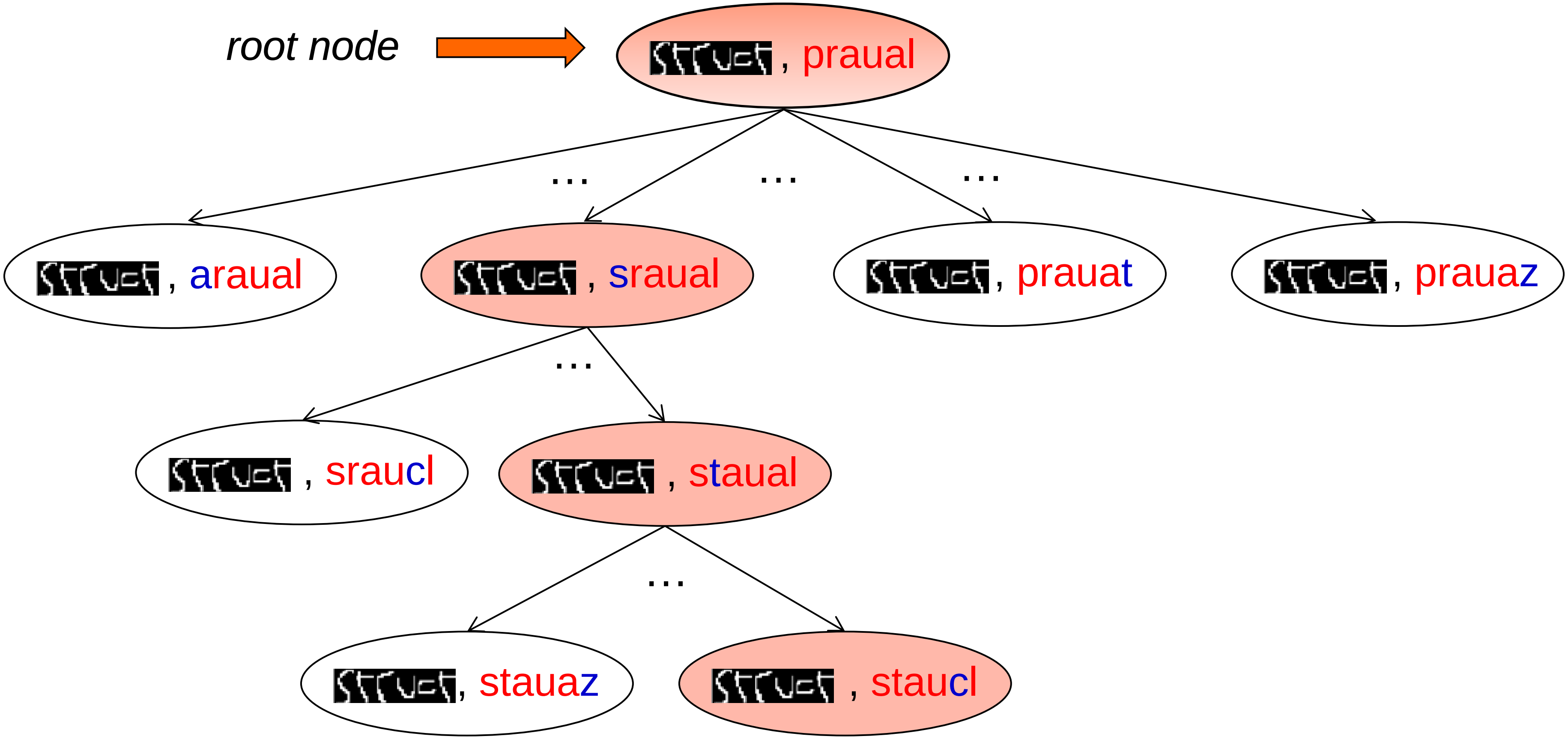
HC-Search Illustration: Greedy Search



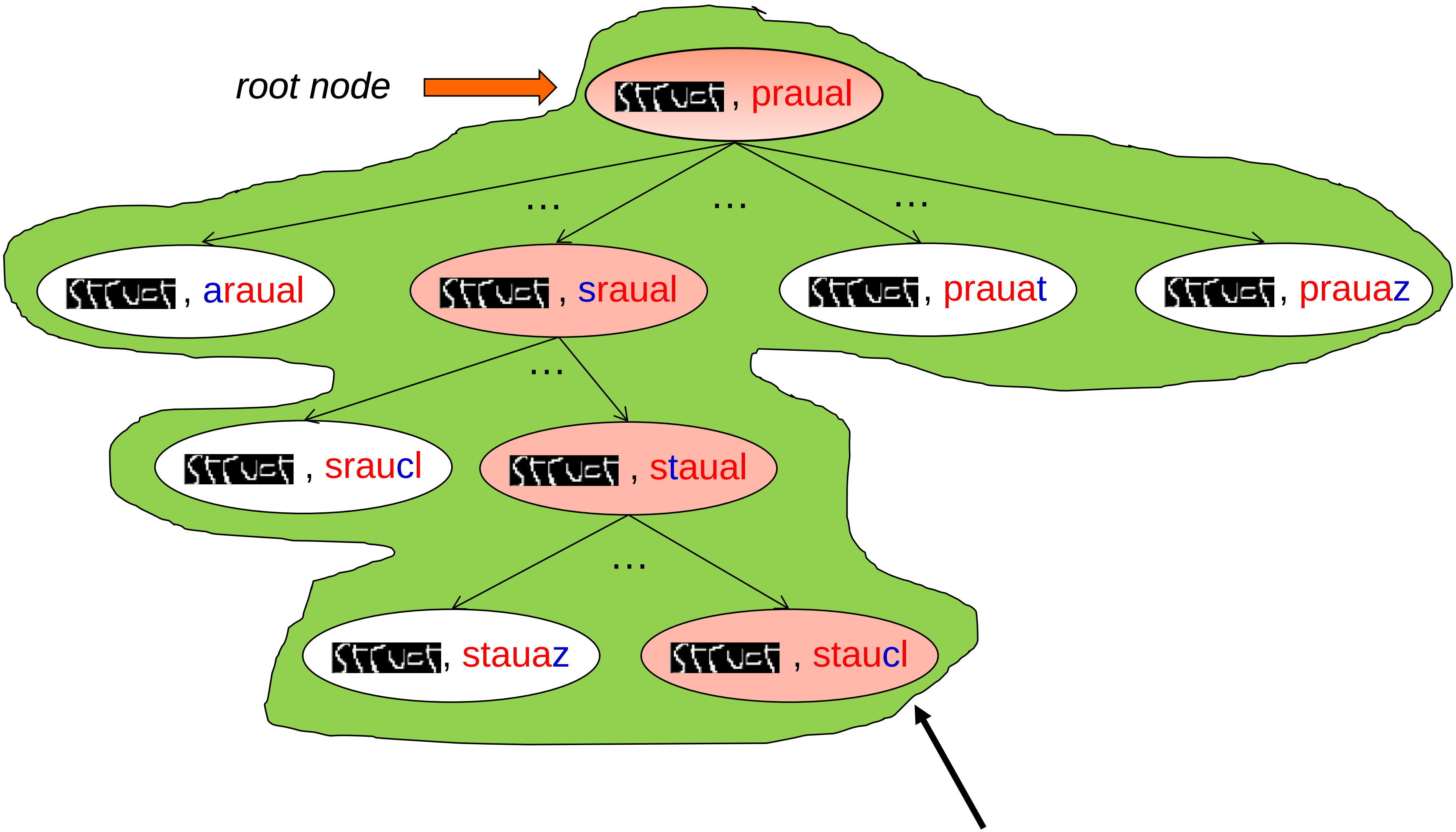
HC-Search Illustration: Greedy Search



HC-Search Illustration: Greedy Search

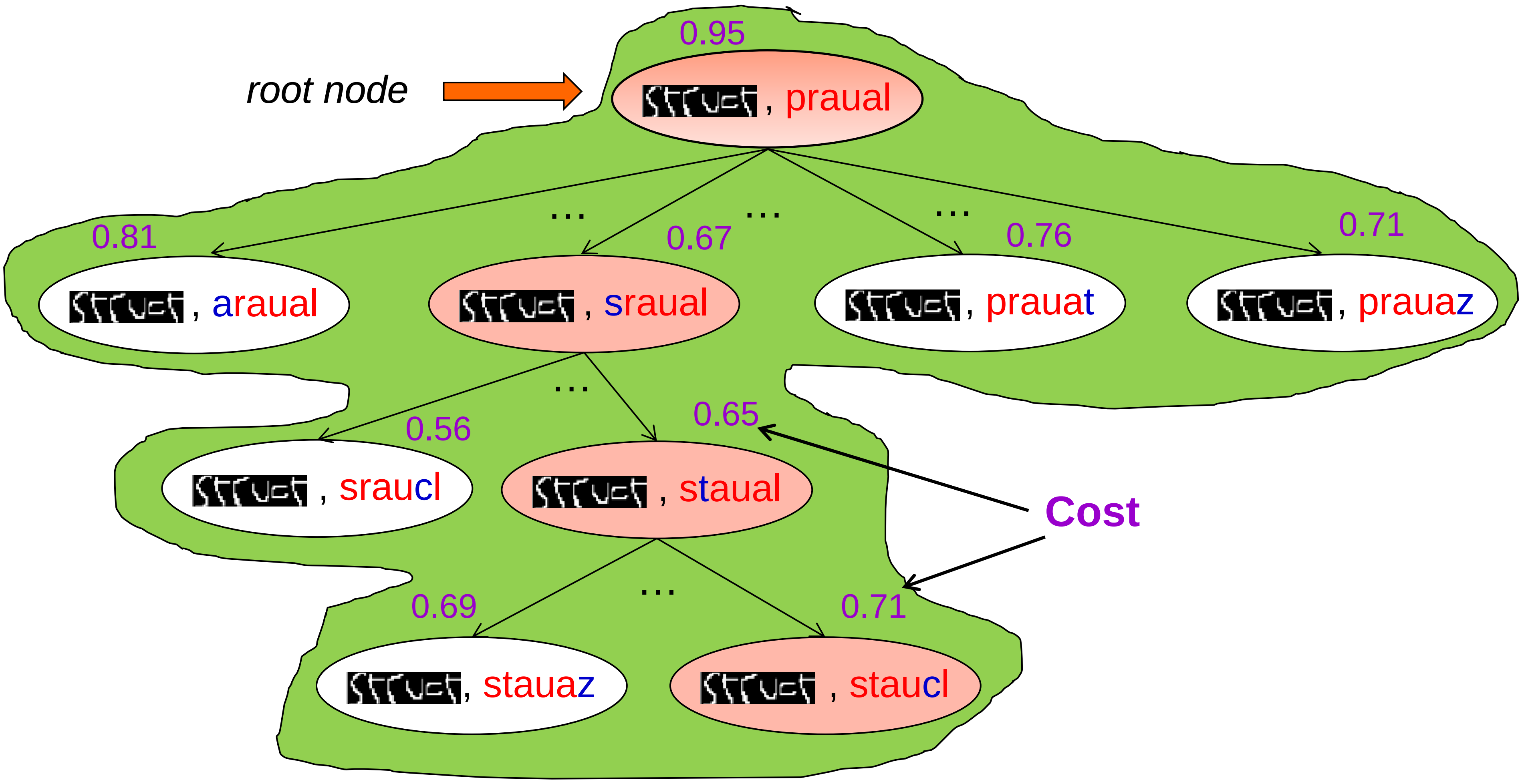


HC-Search Illustration: Greedy Search

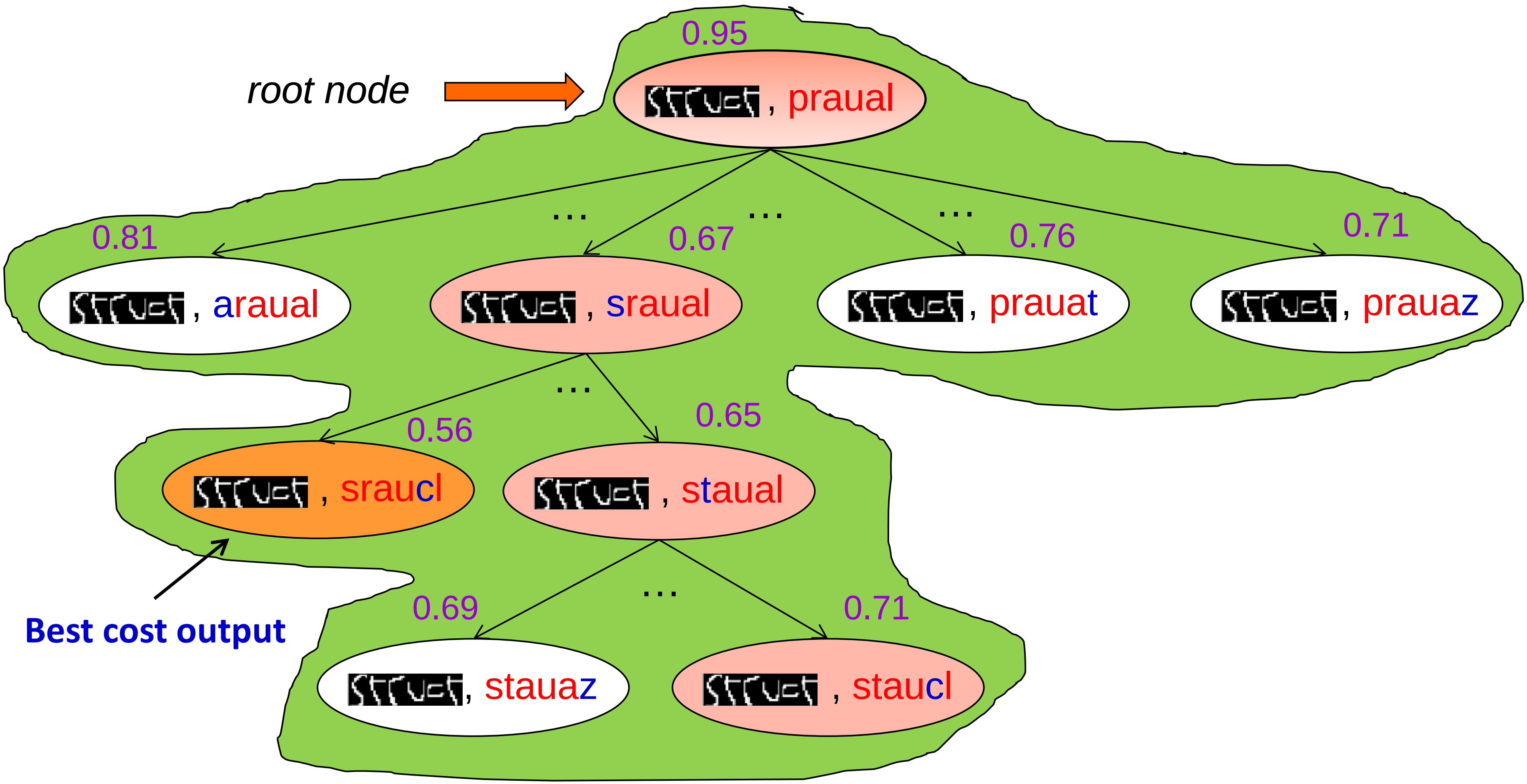


Set of all outputs generated within time limit

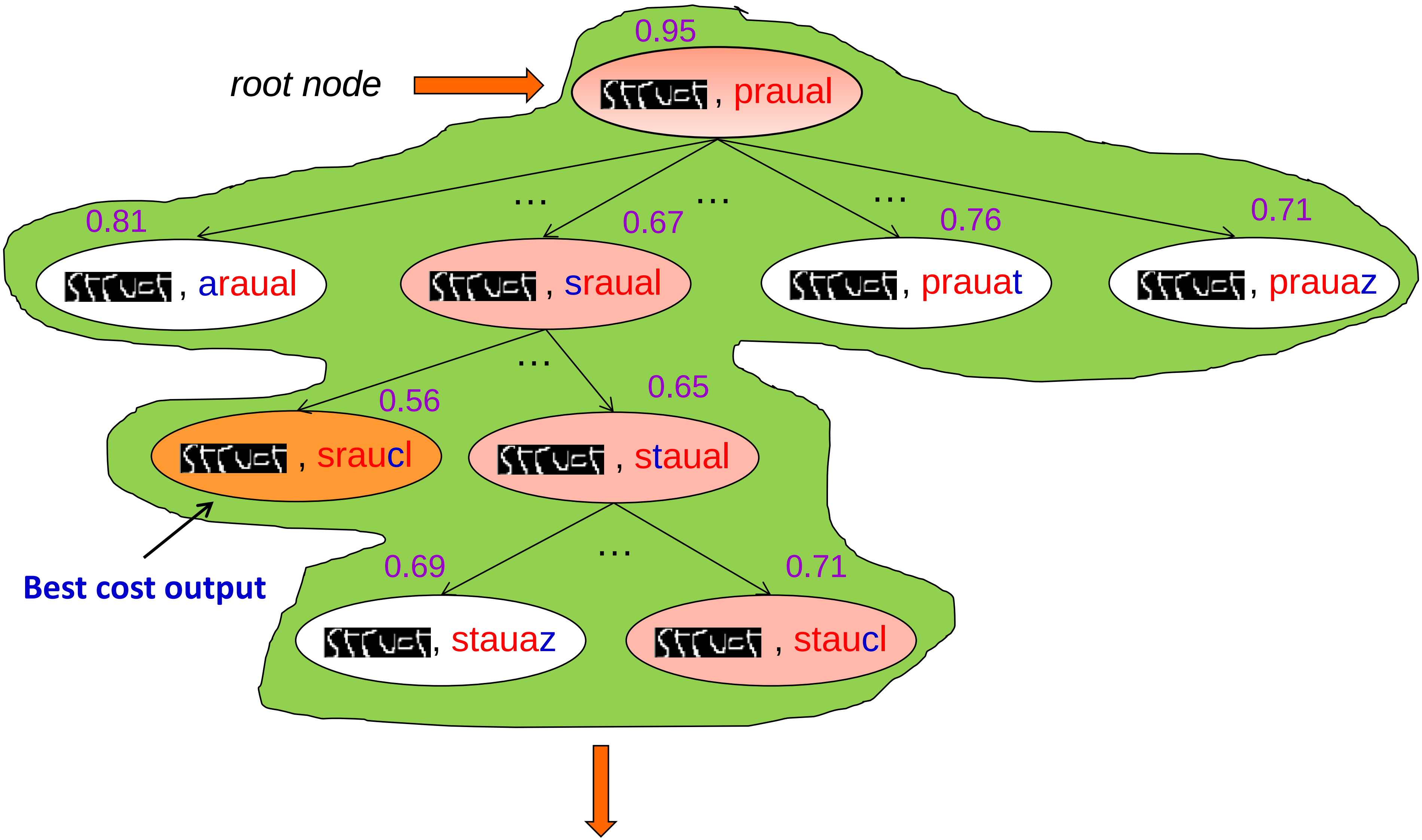
HC-Search Illustration: Greedy Search



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

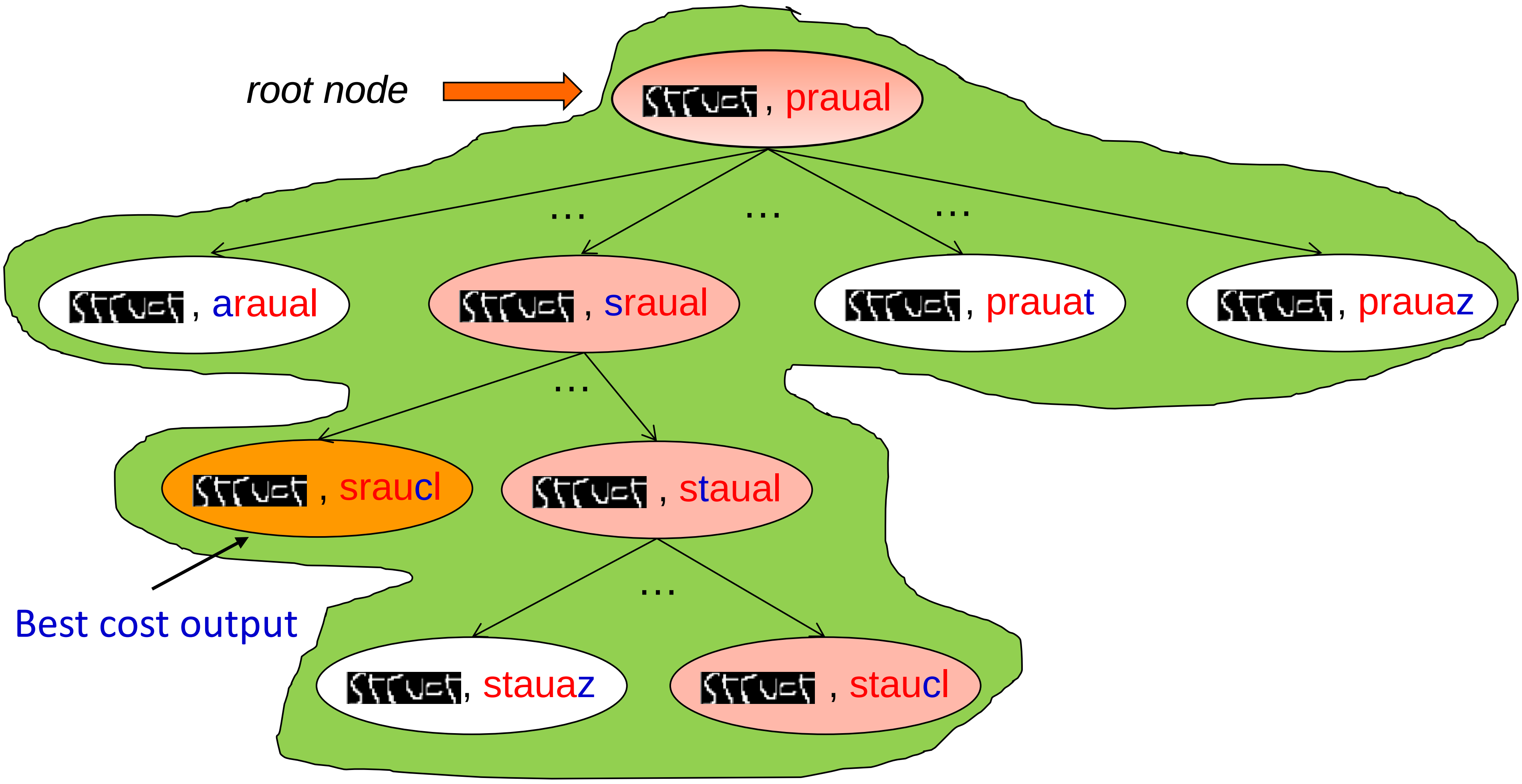
HC-Search: Key Learning Challenges

- **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 ?

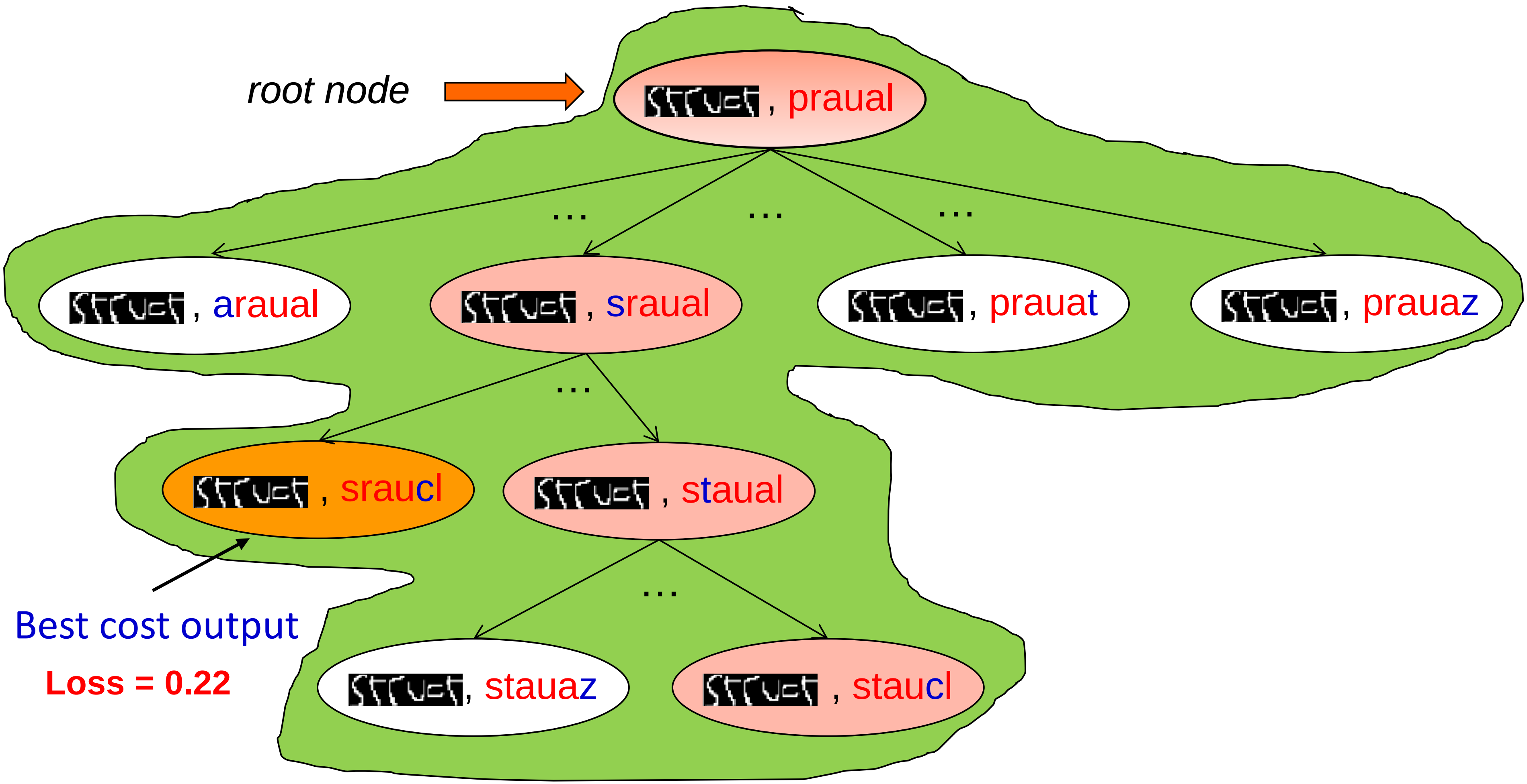
Outline of HC-Search Framework

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

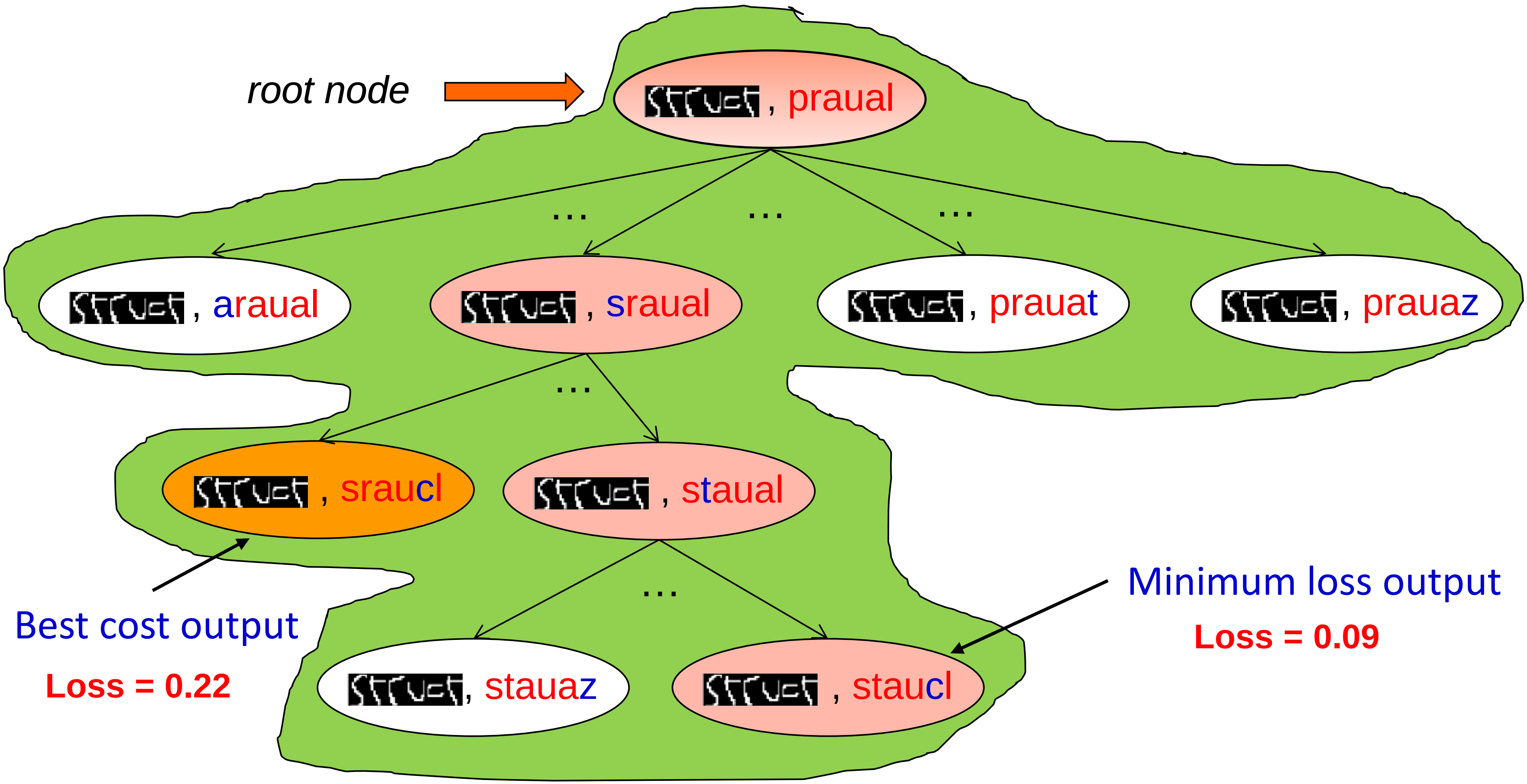
HC-Search: Loss Decomposition



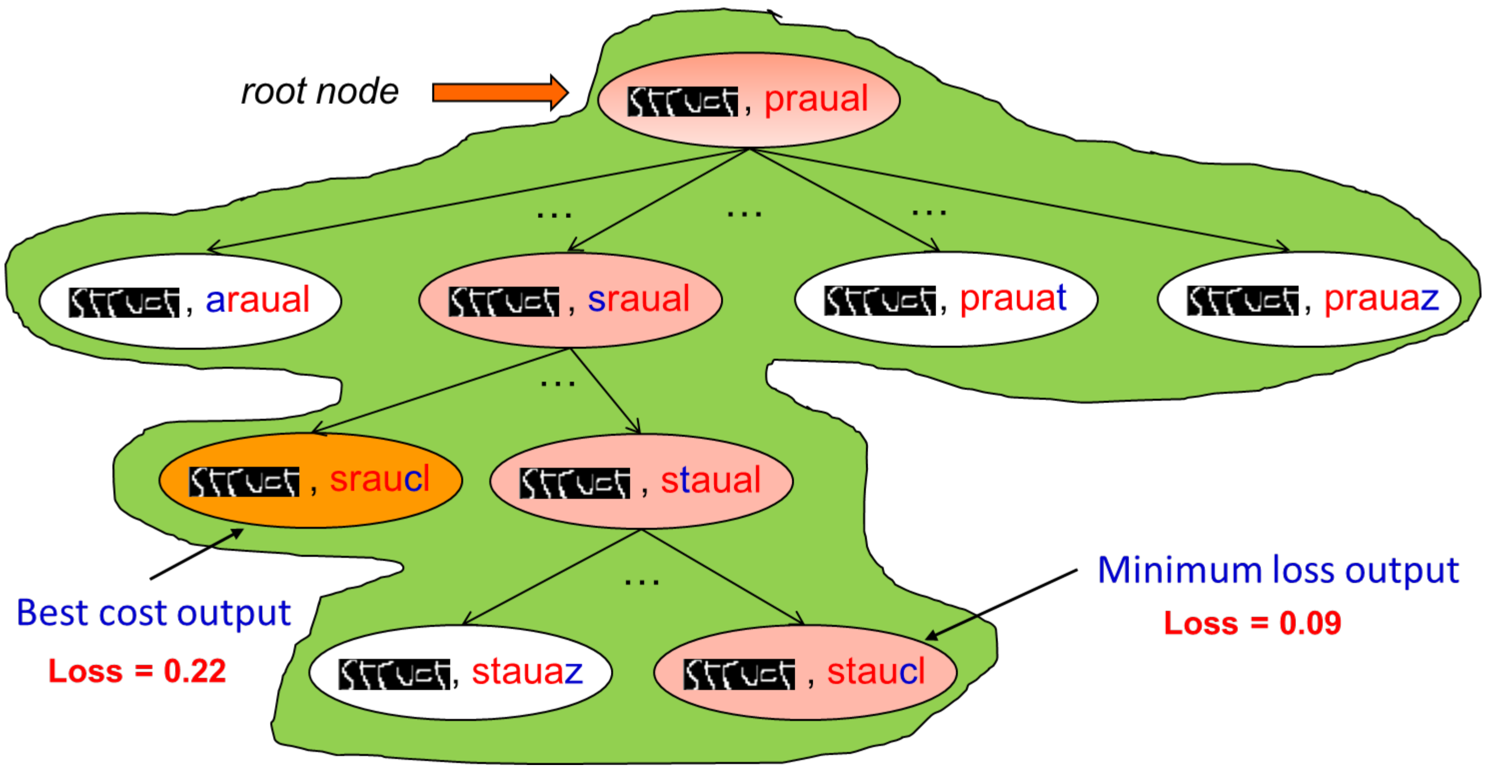
HC-Search: Loss Decomposition



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

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

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:** $\widehat{H} = \arg min_{H \in \boldsymbol{H}}\ \epsilon_H$ (heuristic training)

HC-Search: Learning

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

Overall loss

Generation loss
(Heuristic function)

Selection loss
(Cost function)

- **Key idea:** 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
- Experiments and Results
- 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

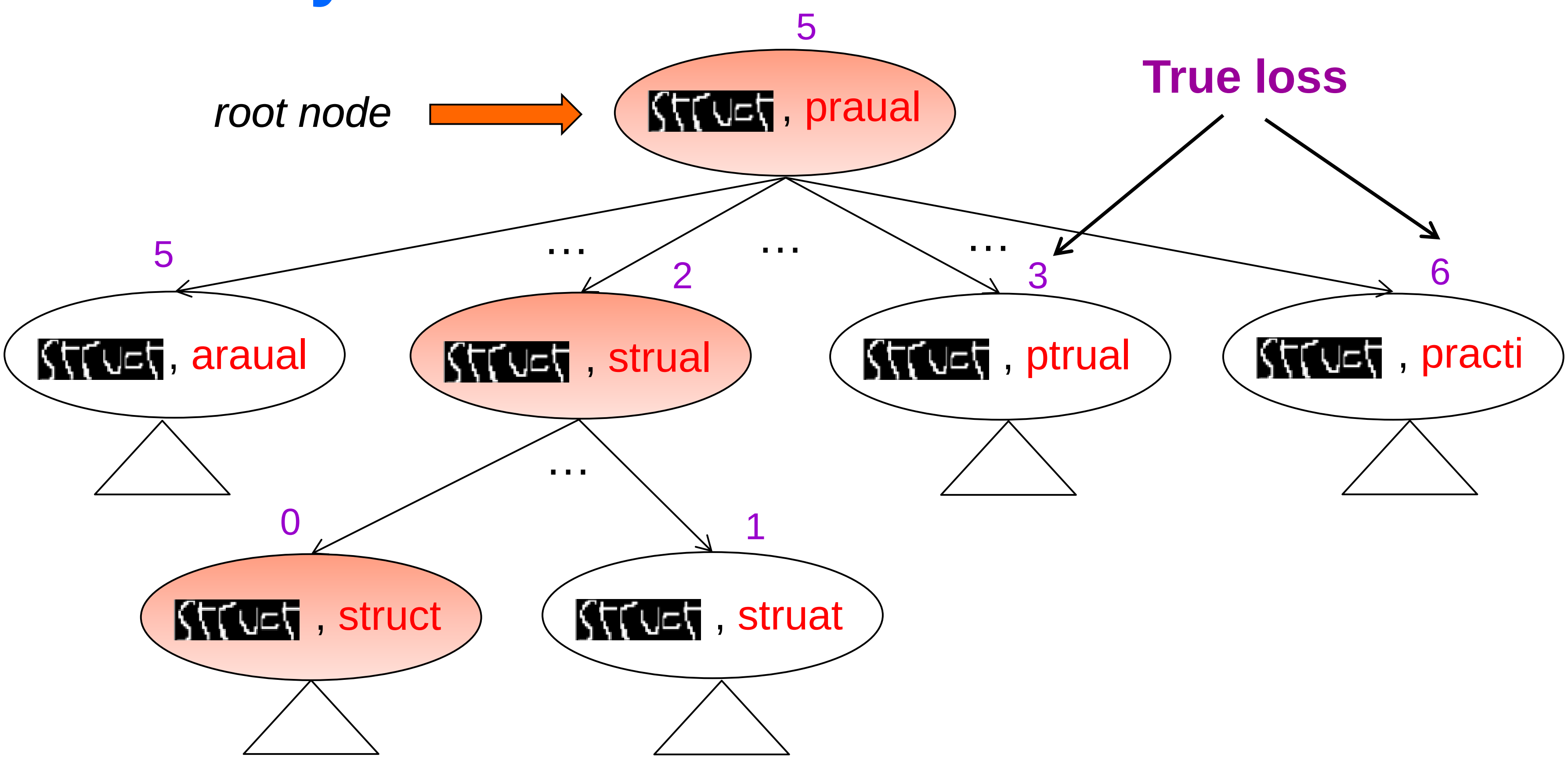
- Given a search procedure (e.g., greedy search)

- **<u>Key idea:</u> 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)

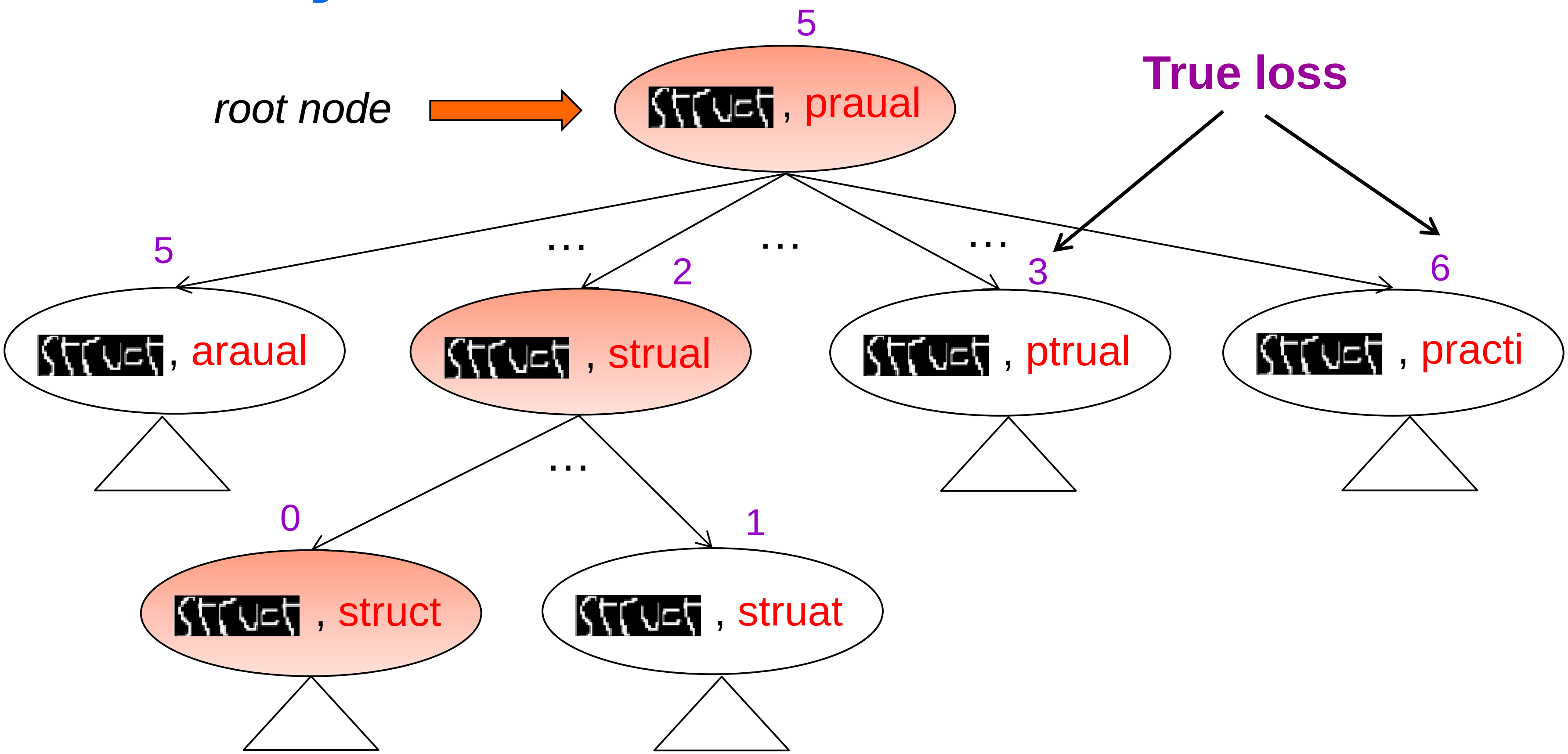
  - Learn a heuristic function that tries to imitate the observed search behavior

Greedy Search: Imitation with true loss



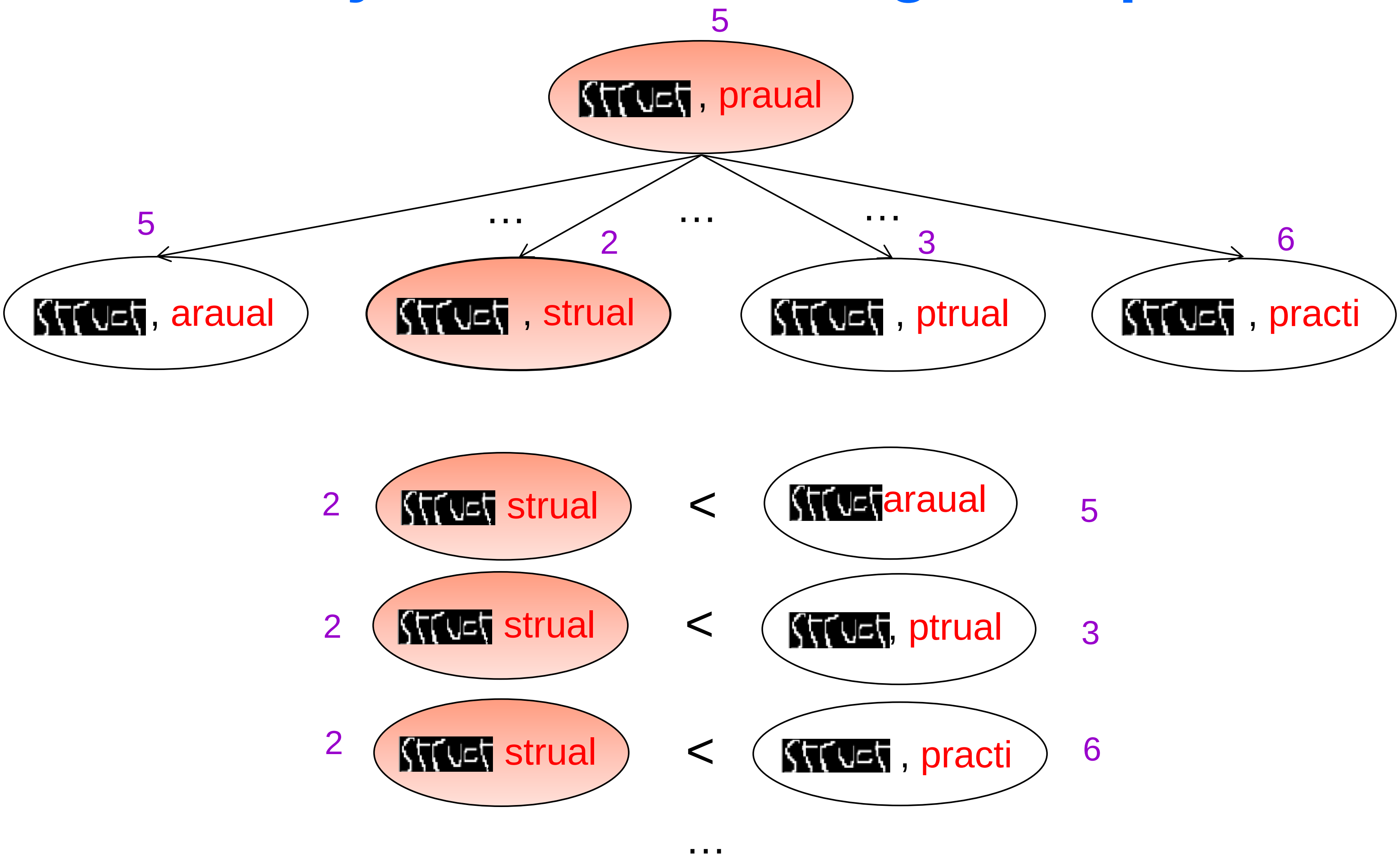
Hamming Loss ( , strual , struct) = 2

Greedy Search: Imitation with true loss

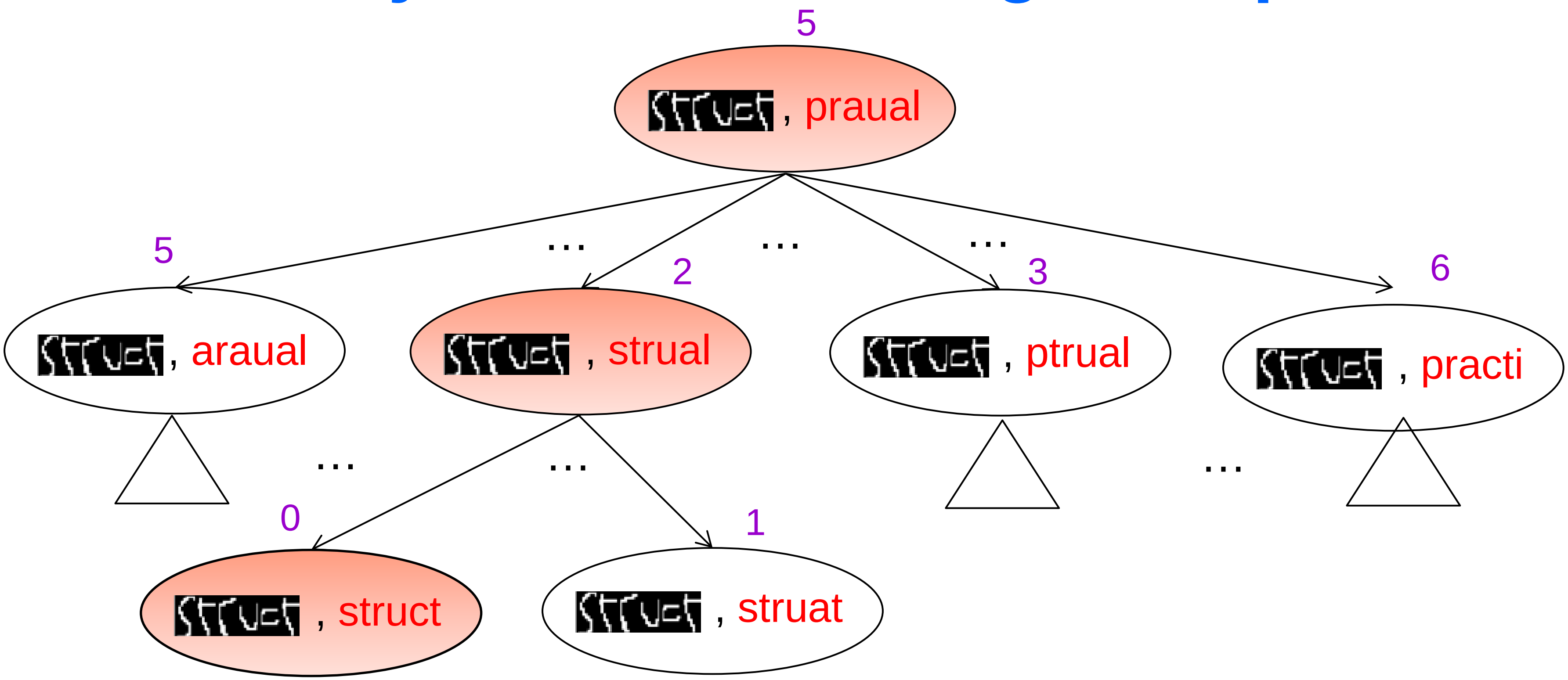


Generation loss $\epsilon_{H^*} = 0$

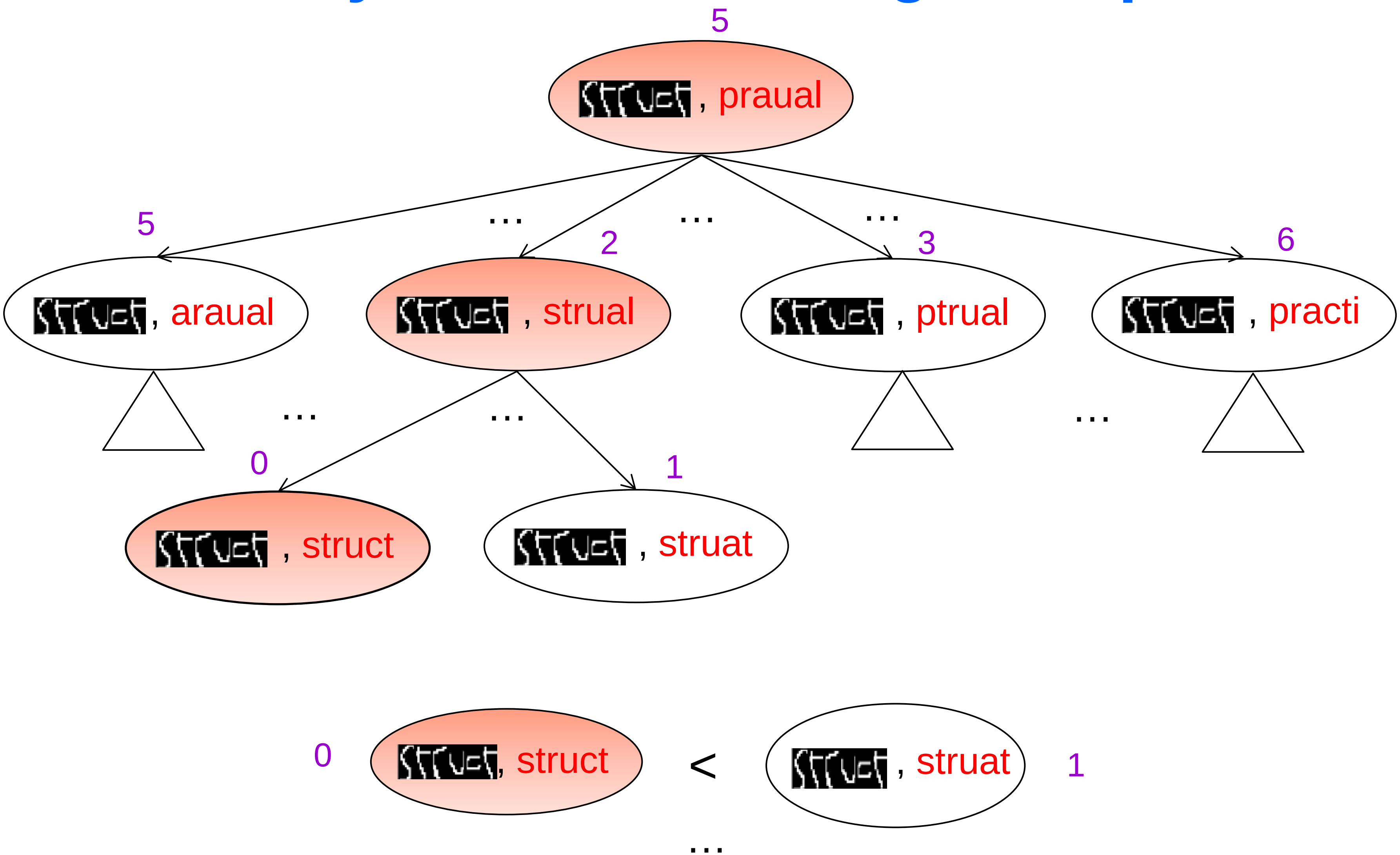
Greedy Search: Ranking examples



Greedy Search: Ranking examples

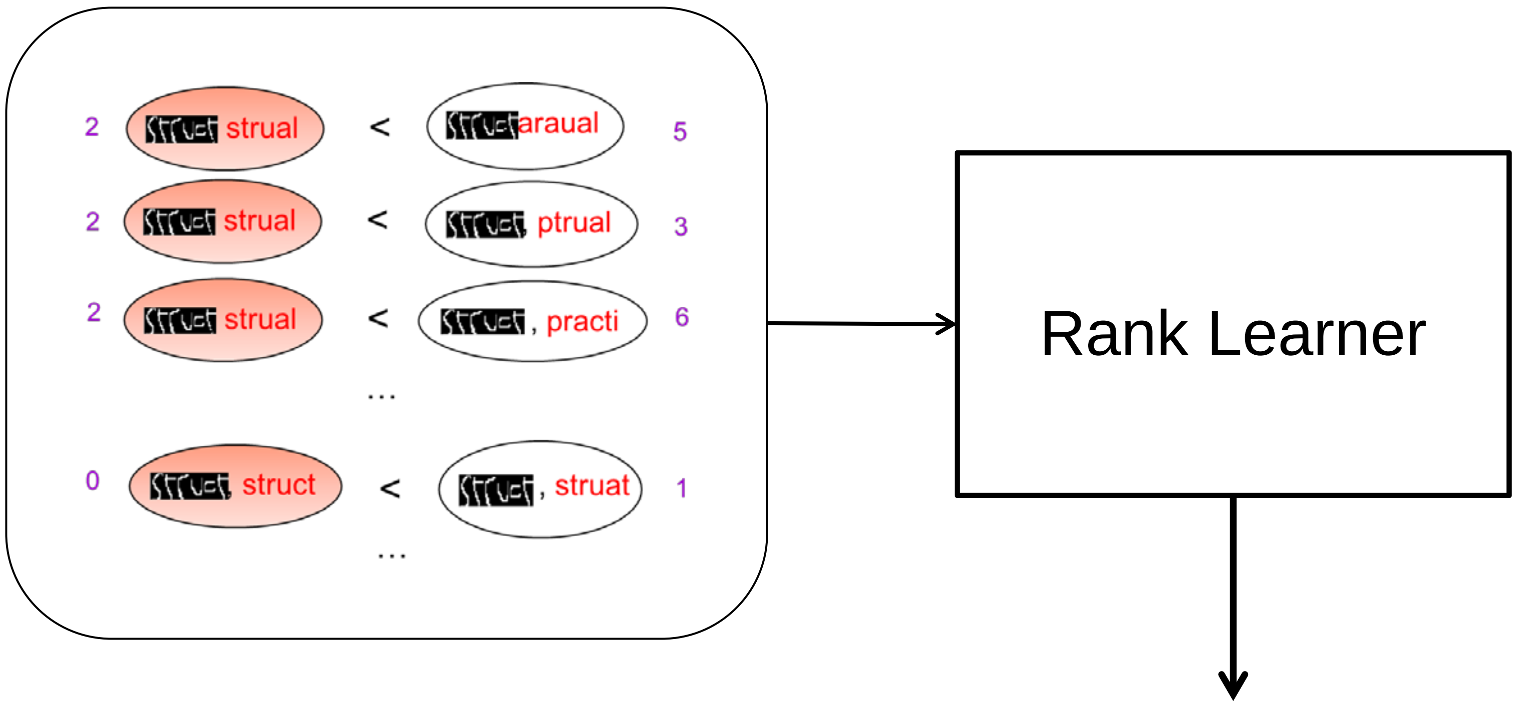


Greedy Search: Ranking examples



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)

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

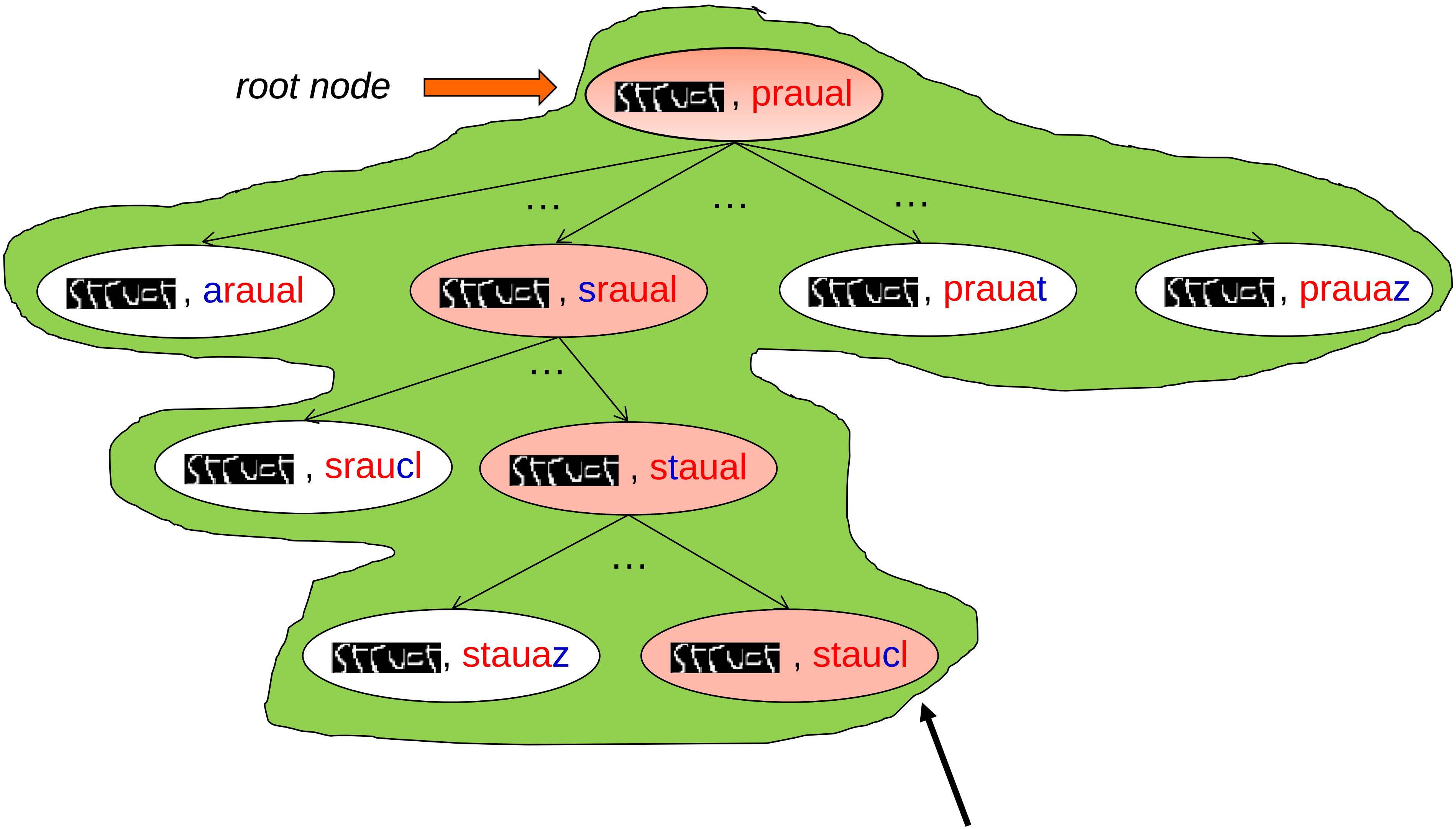
Outline of HC-Search Framework

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

HC-Search: Cost Function Learning

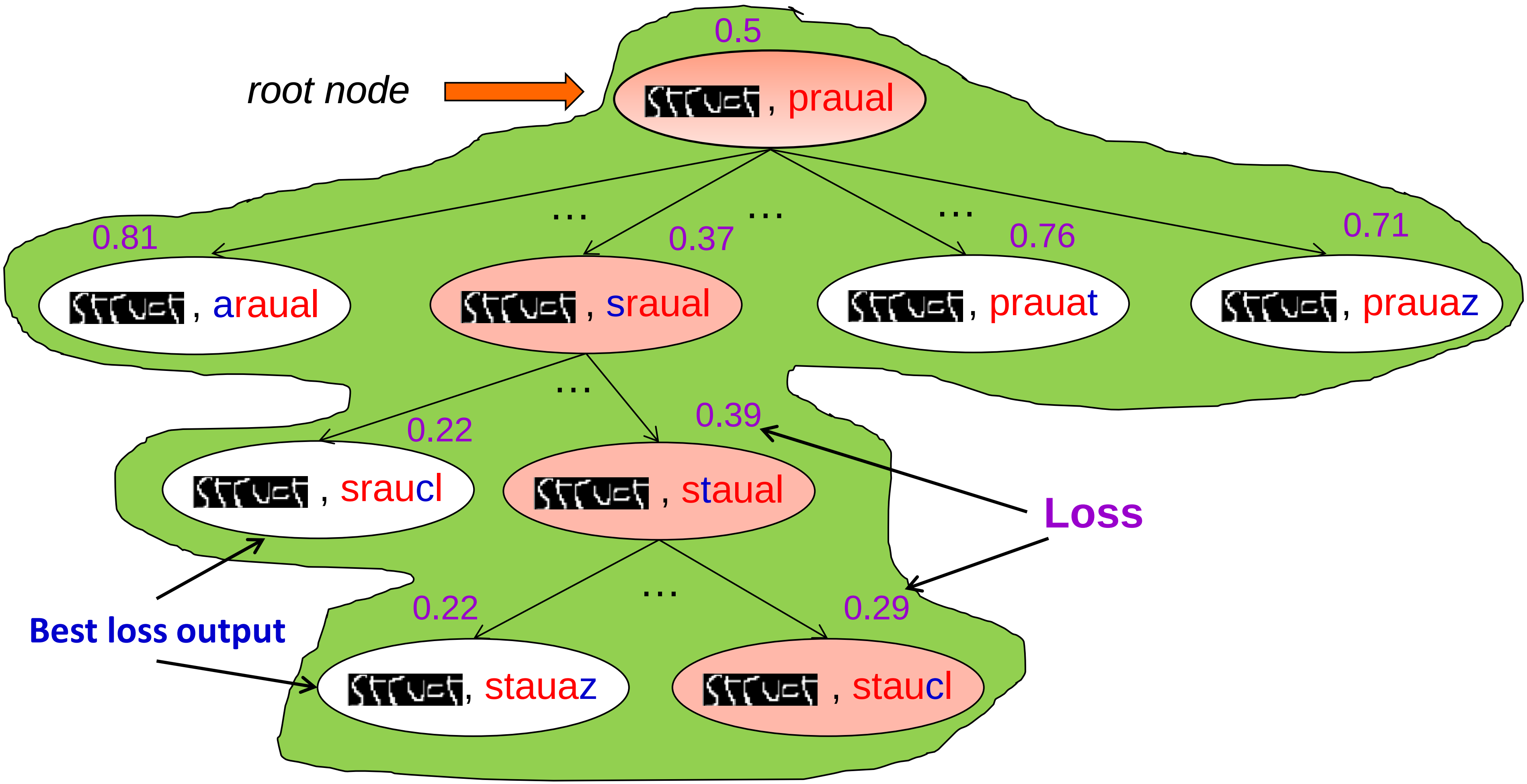
- **Learning Objective:**
  - Correctly score the outputs generated by the heuristic as per their losses

HC-Search: Cost function Learning



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

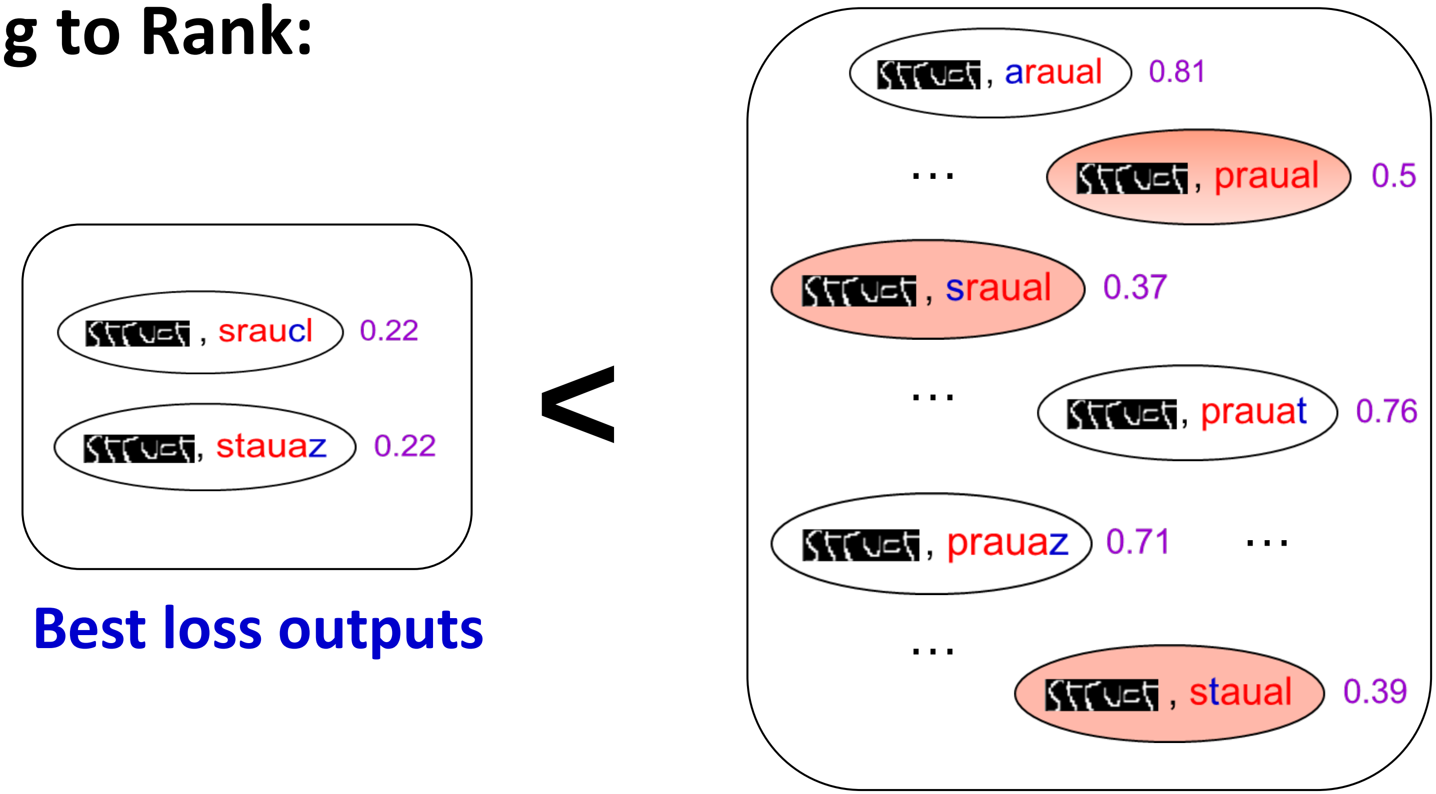
HC-Search: Cost function Learning



- **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:**



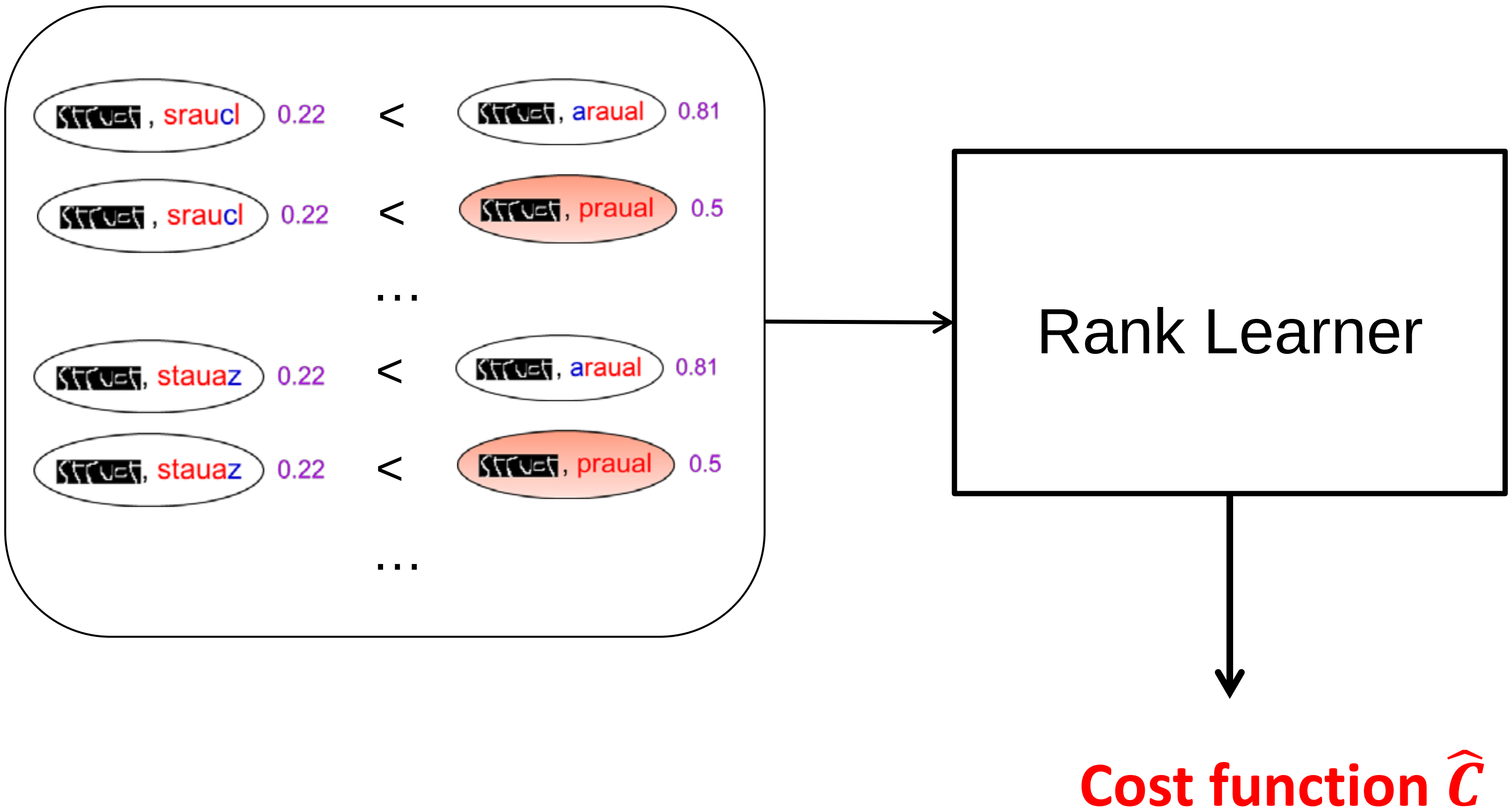
Best loss outputs

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



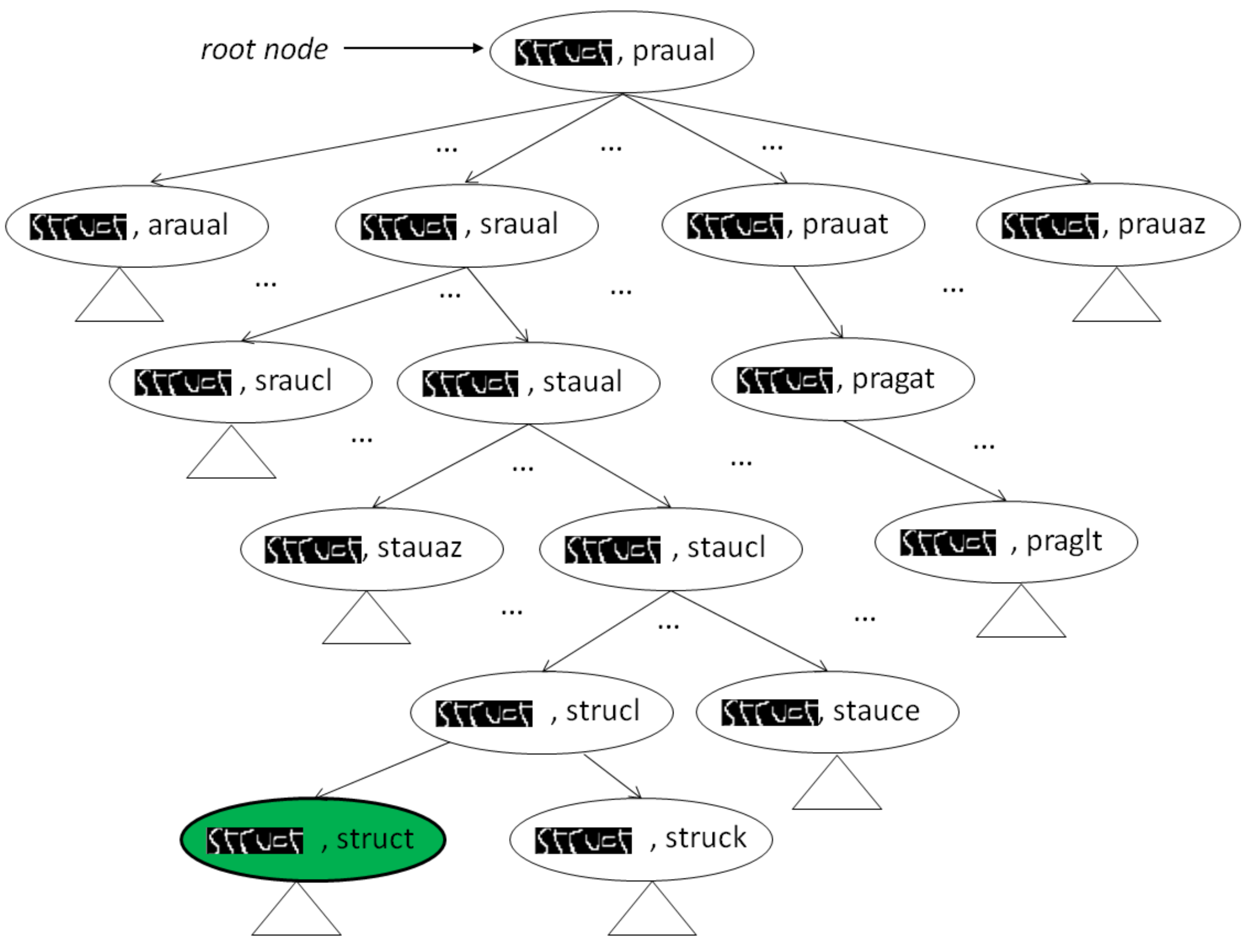
Can borrow generalization bounds from rank-learning literature
[Agarwal and Roth, 2005 & Agarwal and Niyogi, 2009]

Outline of HC-Search Framework

- Introduction
  - Unifying view and high-level overview
- Learning Algorithms
  - Heuristic learning
  - Cost function learning
- Search Space Design
- Experiments and Results
- 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

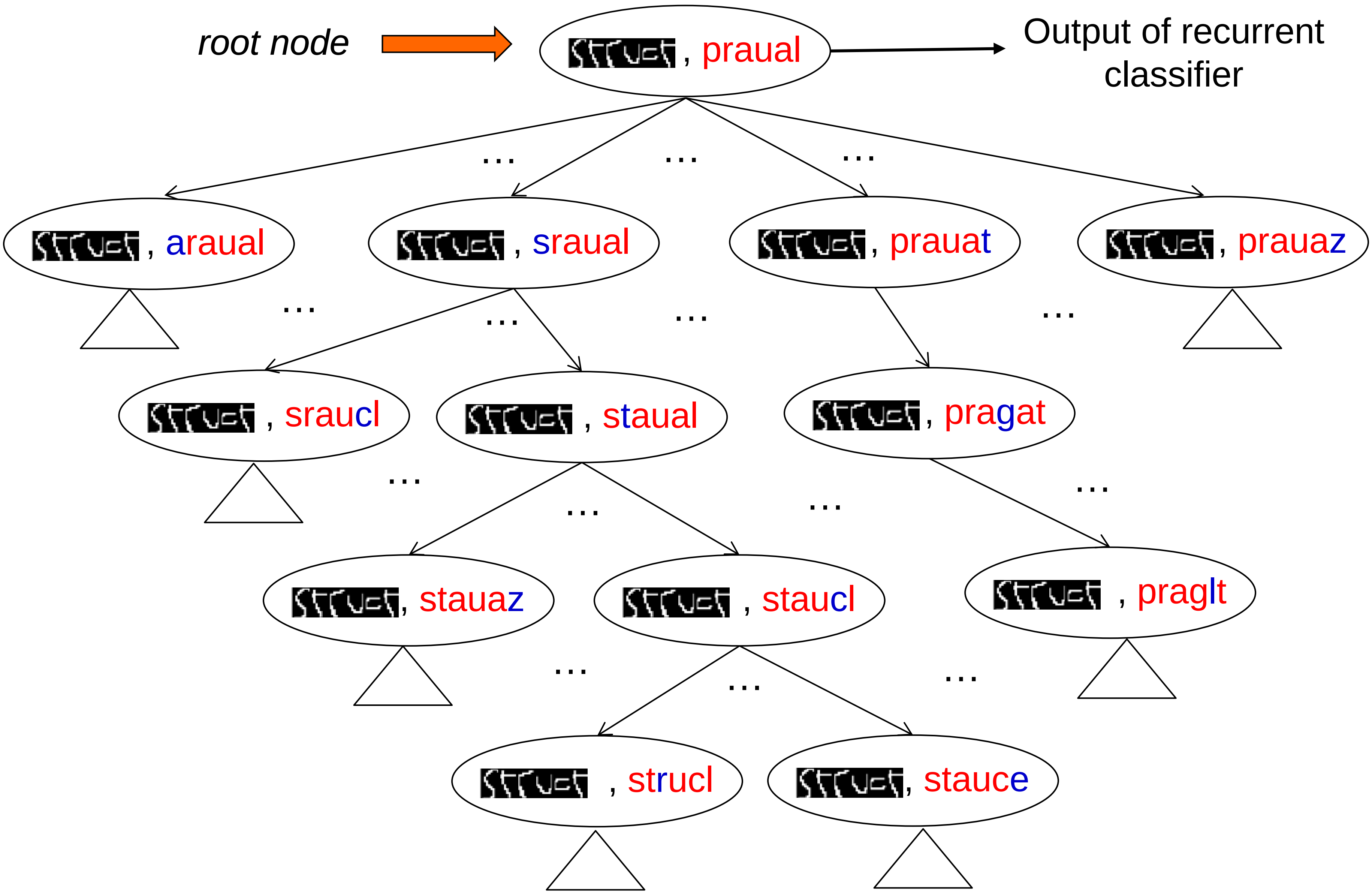
- **Objective:**
  - High-quality outputs can be located at small depth

- **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

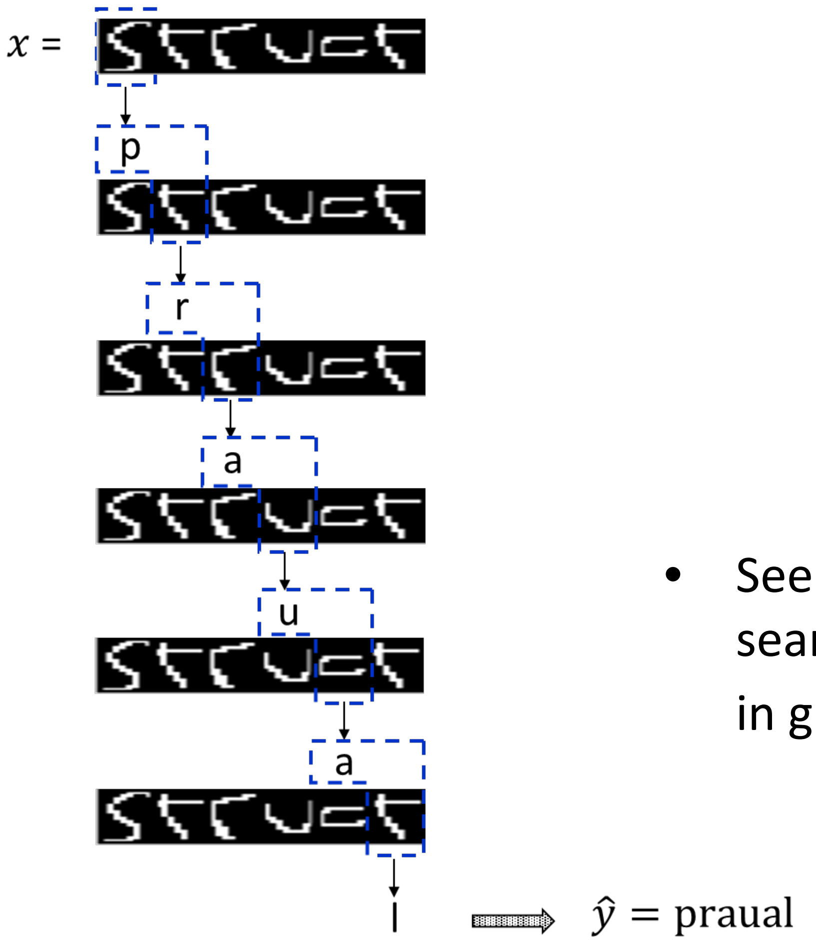
- **Solution #3:**
  - Segmentation Search Space for computer vision tasks [CVPR, 2015]

Flip-bit Search Space



Limited Discrepancy Search: Idea

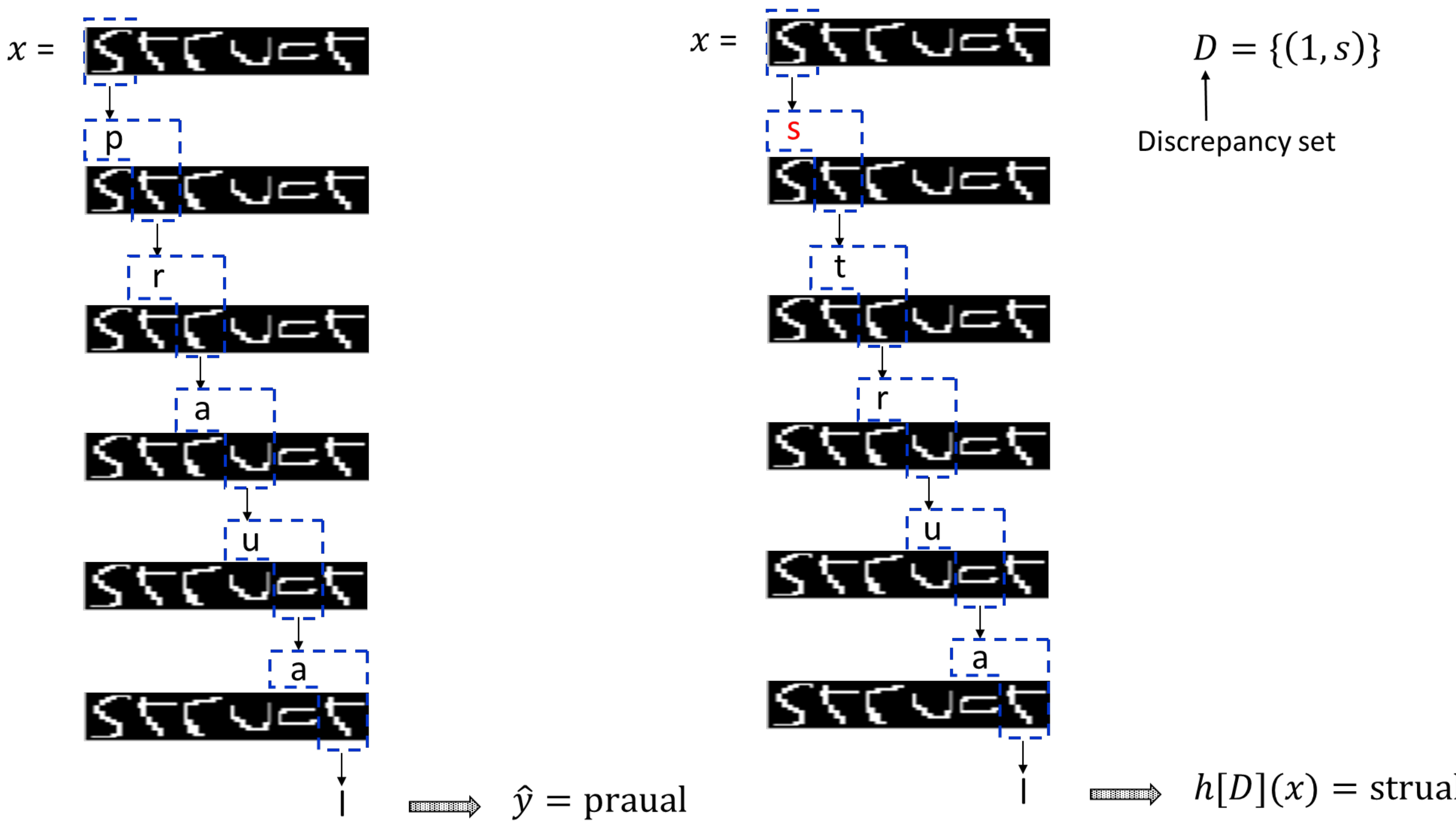
- **Limited Discrepancy Search** [Harvey and Ginsberg, 1995]
  - **Key idea:** correct the response of recurrent classifier at a small no. of critical errors to produce high-quality outputs



- See IJCAI'16 paper on LDS for AND/OR search w/ applications to optimization tasks in graphical models

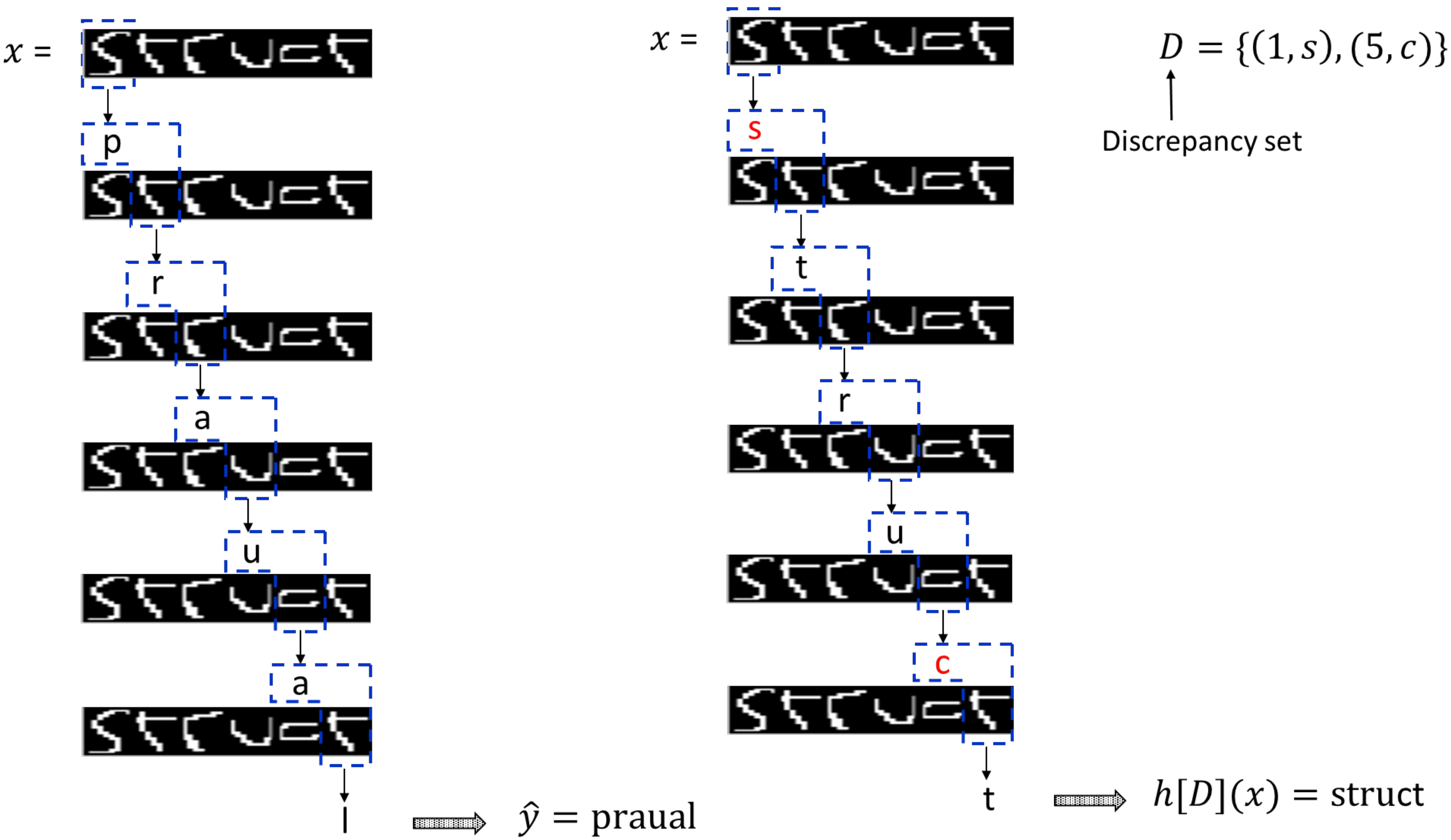
Limited Discrepancy Search: Illustration

- **Limited Discrepancy Search** [Harvey and Ginsberg, 1995]
  - **Key idea:** correct the response of recurrent classifier at a small no. of critical errors to produce high-quality outputs

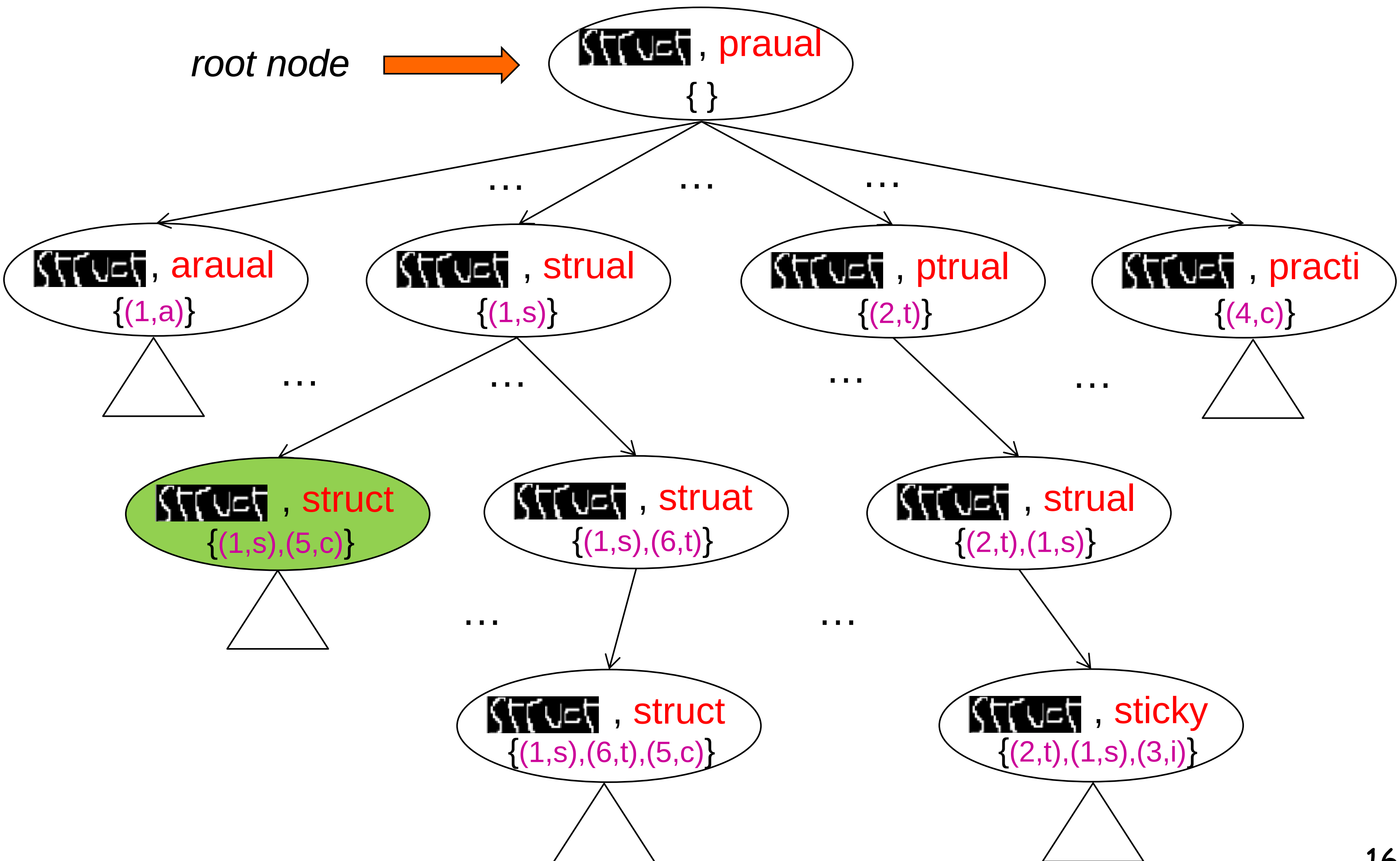


Limited Discrepancy Search: Illustration

- **Limited Discrepancy Search** [Harvey and Ginsberg, 1995]
  - **Key idea:** correct the response of recurrent classifier at a small no. of critical errors to produce high-quality outputs

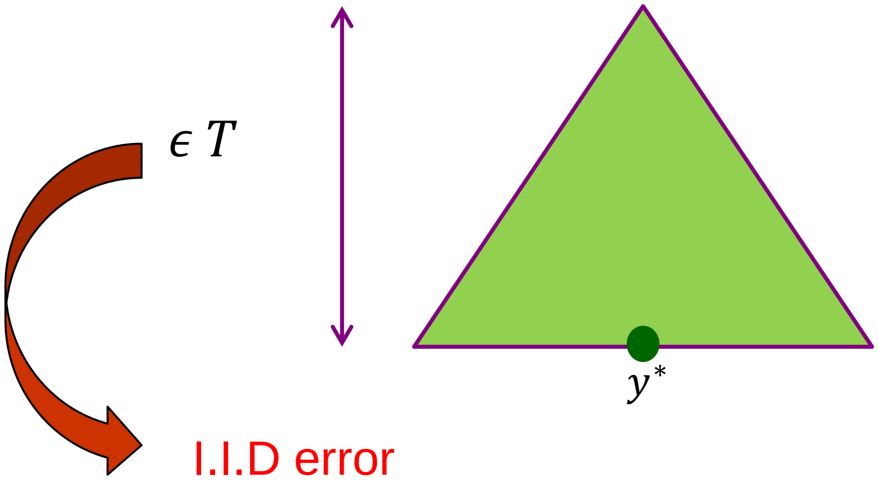


LDS Space: Illustration



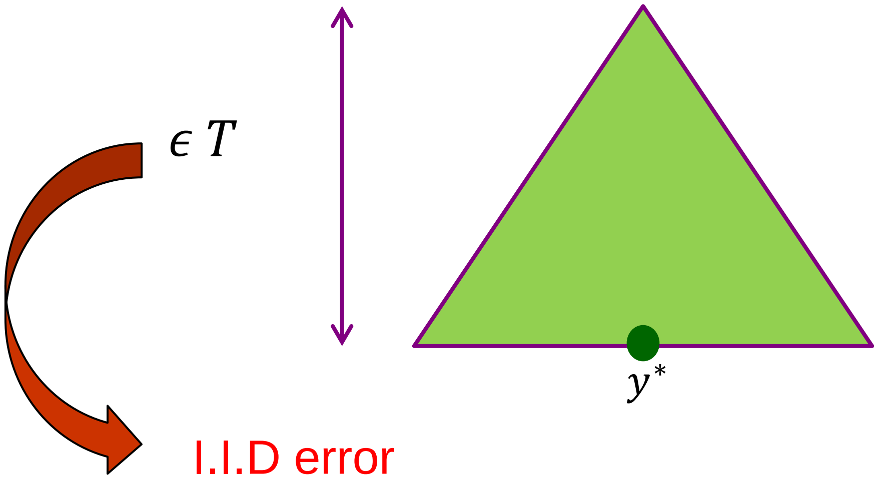
Quality of LDS Space

- Expected target depth



Quality of LDS Space

- Expected target depth

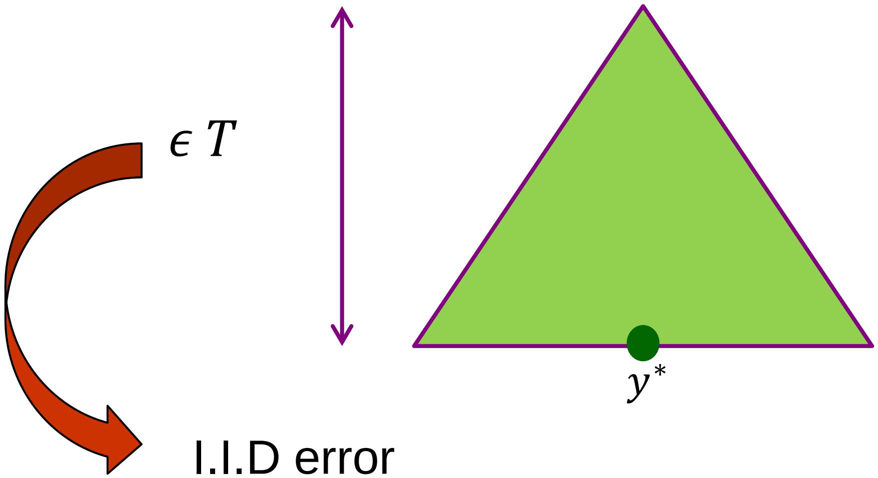


  - We can learn a classifier to optimize the I.I.D error ϵ

Doppa, J.R., Fern, A., Tadepalli, P. Structured Prediction via Output Space Search. *Journal of Machine Learning Research (JMLR), vol 15,* 2014.

Quality of LDS Space

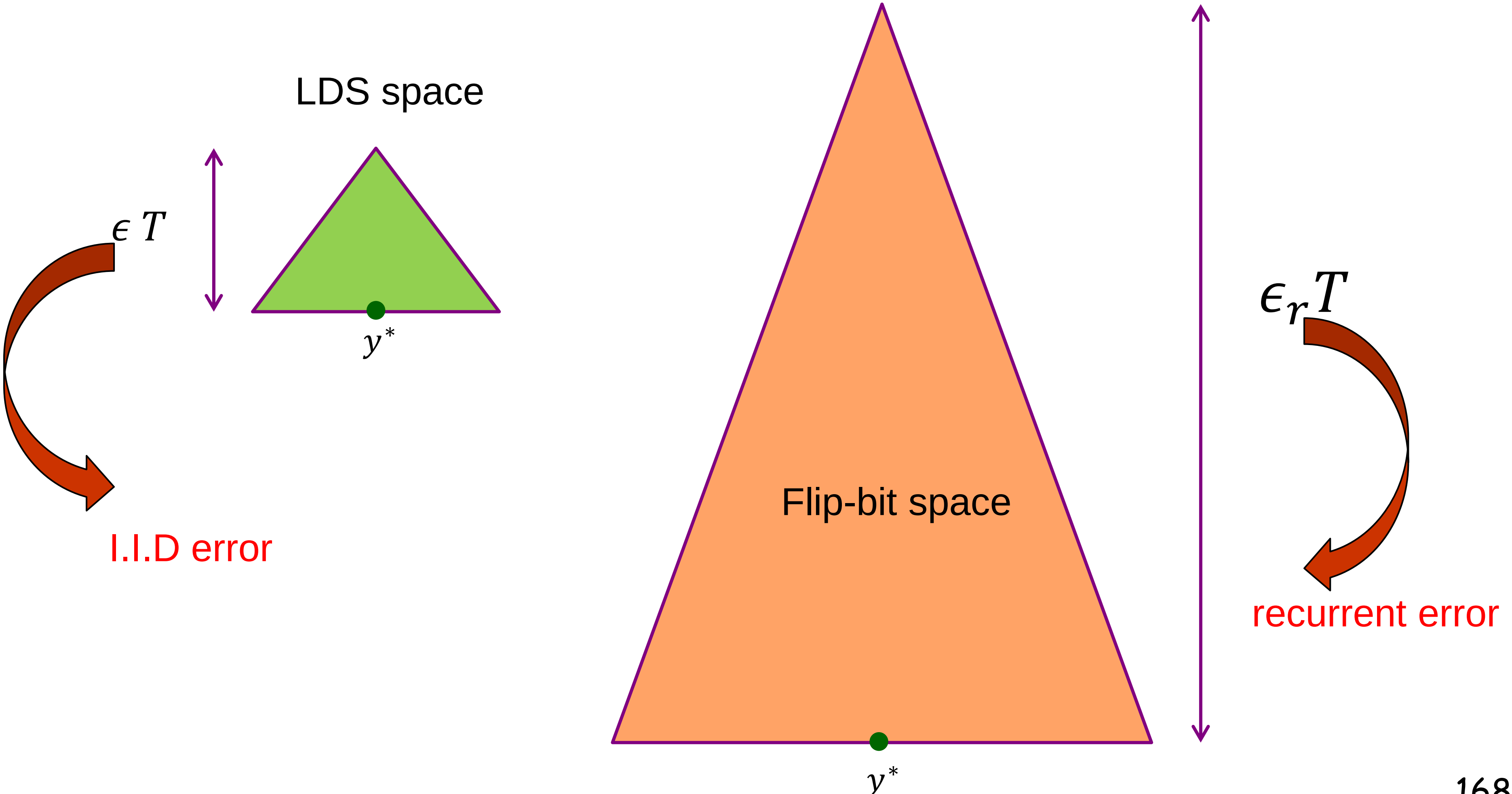
- Expected target depth



- o We can learn a classifier to optimize the I.I.D error ϵ
- o Important contribution that helped HC-Search achieve state-of-the-art results

Quality of Search Space: LDS vs. Flip-bit

- Expected target depth of a search space



Sparse LDS Space (k)

- **Complete LDS space is expensive**
  - each successor state generation requires running greedy policy with the given discrepancy set
  - # successors = $L.T$, where T is the size of the structured output and L is the number of labels
- **Sparse Search Space: Key Idea**
  - Sort discrepancies using recurrent classifier scores and pick top-k choices
  - # successors = $k.T$
  - Parameter k = # discrepancies for each variable controls the trade-off between speed and accuracy
  - In practice, very small k suffice
  - How can we deal with dependence on T?

Aside: Very simple HC-Search Instantiation

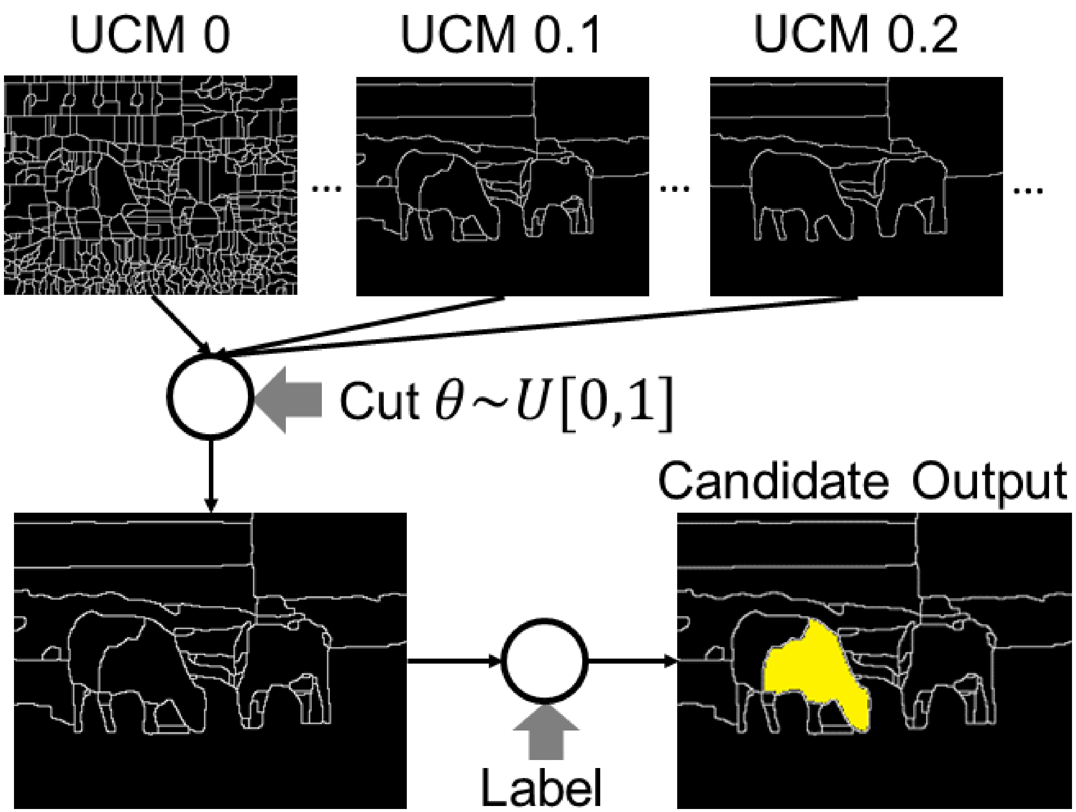
- **Heuristic function**
  - Greedy recurrent classifier (or policy)
- **Search procedure**
  - Depth-first or Breadth-first Limited Discrepancy Search w/ bounded depth
- **Cost function**
  - Score the outputs generated by search procedure

Computer Vision Tasks: Randomized Segmentation Space [Lam et al., 2015]

- **Key Idea:** probabilistically sample likely object configurations in the image from a hierarchical segmentation tree

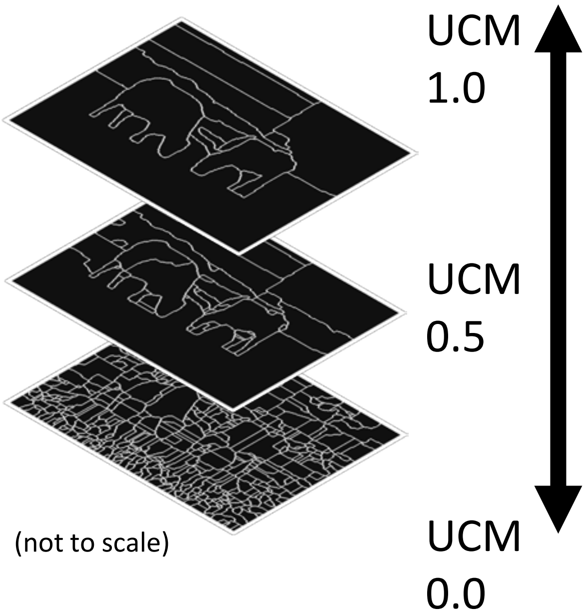
- Segmentation selection

- Candidate generation



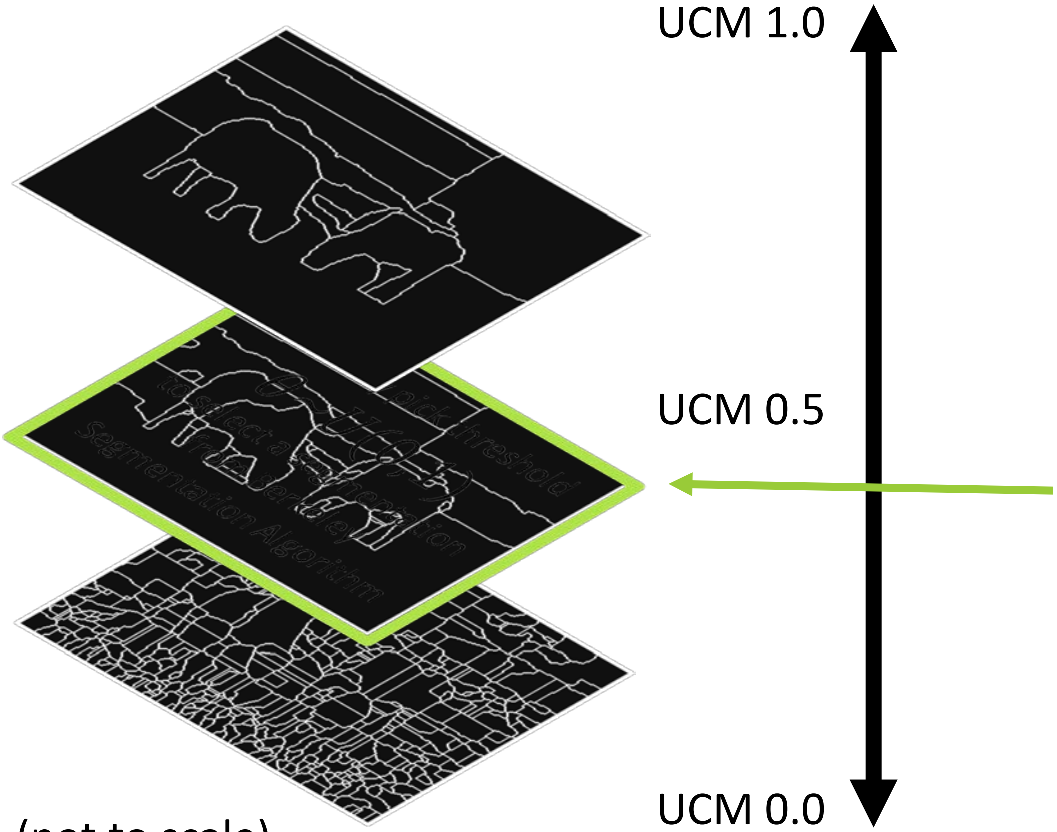
Pre-requisite: Hierarchical Segmentation Tree

- **Berkeley segmentation tree**
  - Regions are very robust
  - Regions are closed
  - UCM level 0 corresponds to all super-pixels



(not to scale)

Randomized Segmentation Space: Segmentation Selection



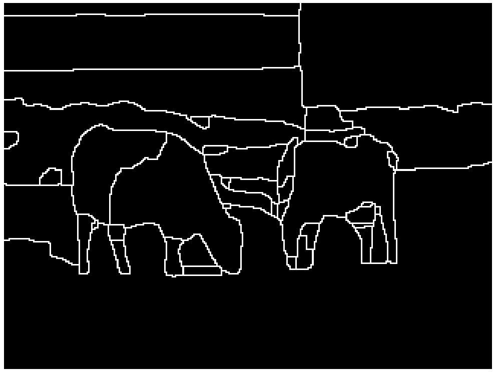
(not to scale)

Randomly pick threshold

$$\theta \sim U(0,1)$$

to select a segmentation from Berkeley Segmentation Algorithm

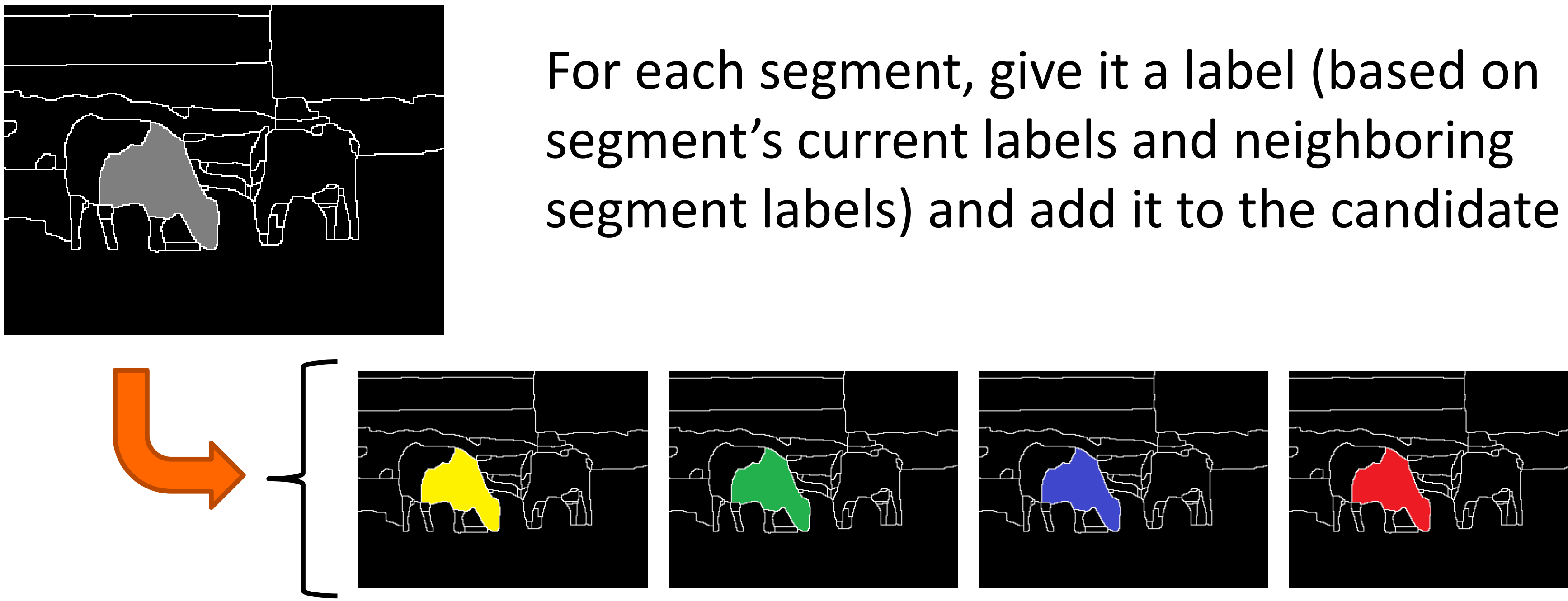
Randomized Segmentation Space: Candidate Generation



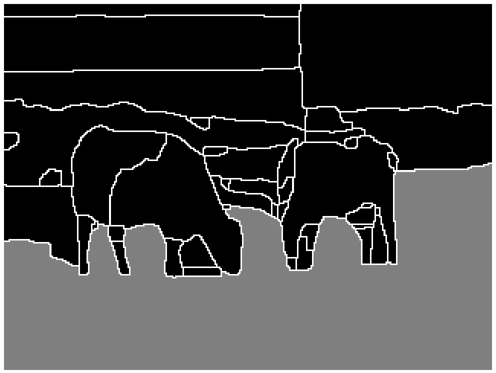
For each segment, give it a label (based on segment’s current labels and neighboring segment labels) and add it to the candidate set

Randomized Segmentation Space: Candidate Generation

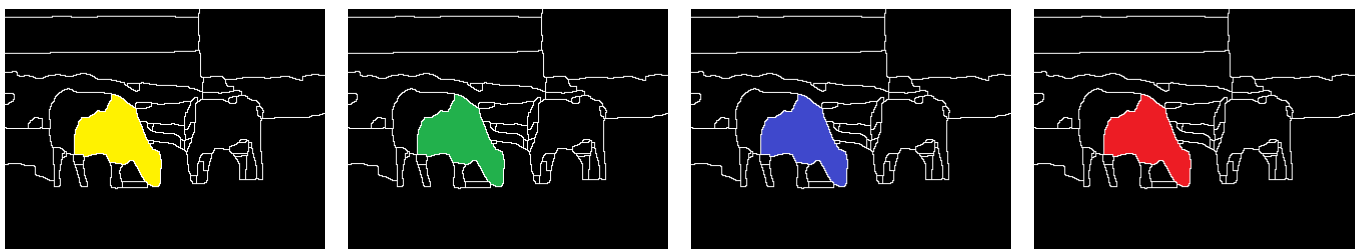
For each segment, give it a label (based on segment's current labels and neighboring segment labels) and add it to the candidate set

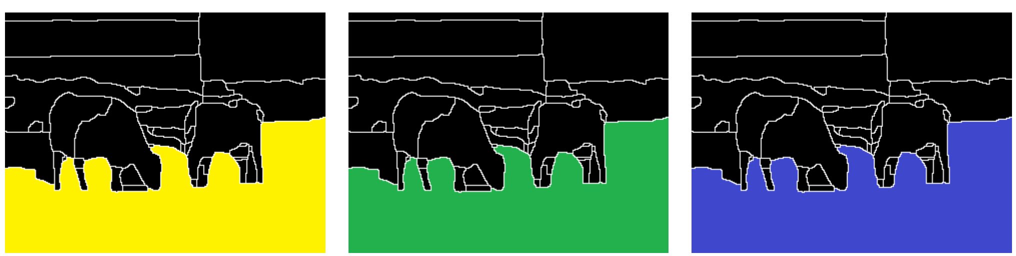


Randomized Segmentation Space: Candidate Generation



For each segment, give it a label (based on segment's current labels and neighboring segment labels) and add it to the candidate set





…etc…

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

Benchmark Domains

- **Handwriting recognition** [Taskar et al., 2003]
  - *HW-Small* and *HW-Large*

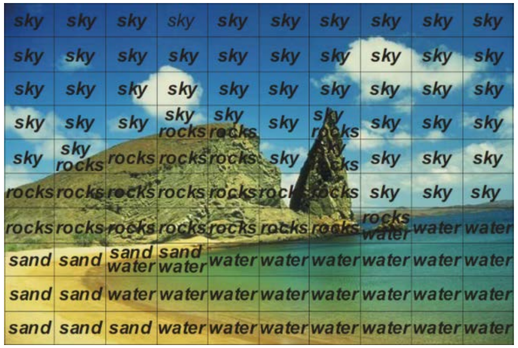
x =  y = s t r u c t u r e d

- **NET-Talk** [Sejnowski and Rosenberg, 1987]
  - *Stress* and *Phoneme* prediction

x = "photograph" y = /f-Ot@graf-/

- **Scene labeling** [Vogel et al., 2007]

x = 

y = 

Experimental Setup

- Search space: **LDS space**
- Search procedure: **Greedy search**
- Time bound: **15 steps** for sequences and **150** for scene labeling
- Loss function: **Hamming loss**
- Baselines
  - **Recurrent**
  - **CRFs**
  - **SVM-Struct**
  - **SEARN**
  - **CASCADES**
  - **C-Search**

Results: comparison to state-of-the-art

Error-rates of different structured prediction algorithms

	HW-Small	HW-Large	Phoneme	Scene labeling
HC-Search	**12.81**	**03.23**	**16.05**	**19.71**
C-Search	17.41	07.41	20.91	27.05
CRF	19.97	13.11	21.09	-
SVM-Struct	19.64	12.49	21.70	-
Recurrent	34.33	25.13	26.42	43.36
SEARN	17.88	09.42	22.74	37.69
CASCADES	**13.02**	**03.22**	17.41	-

Results: comparison to state-of-the-art

Error-rates of different structured prediction algorithms

	HW-Small	HW-Large	Phoneme	Scene labeling
HC-Search	**12.81**	**03.23**	**16.05**	**19.71**
C-Search	17.41	07.41	20.91	27.05
CRF	19.97	13.11	21.09	-
SVM-Struct	19.64	12.49	21.70	-
Recurrent	34.33	25.13	26.42	43.36
SEARN	17.88	09.42	22.74	37.69
CASCADES	13.02	03.22	17.41	-

Results: comparison to state-of-the-art

Error-rates of different structured prediction algorithms

	HW-Small	HW-Large	Phoneme	Scene labeling
HC-Search	**12.81**	**03.23**	**16.05**	**19.71**
C-Search	17.41	07.41	20.91	27.05
CRF	19.97	13.11	21.09	-
SVM-Struct	19.64	12.49	21.70	-
Recurrent	34.33	25.13	26.42	43.36
SEARN	17.88	09.42	22.74	37.69
CASCADES	13.02	03.22	17.41	-

- **HC-Search** outperforms all the other algorithms including C-Search (our prior approach that uses a single function C to serve the dual roles of heuristic and cost function)

Results: Loss Decomposition Analysis

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

Overall expected loss

Generation loss (Heuristic function)

Selection loss (Cost function)

Results: Loss decomposition analysis

	Phoneme			Scene labeling		
Error	ϵ	ϵ_H	$\epsilon_{C\|H}$	ϵ	ϵ_H	$\epsilon_{C\|H}$
HC-Search	16.05	03.98	**12.07**	19.71	5.82	**13.89**

Results: Loss decomposition analysis

	Phoneme			Scene labeling		
Error	ϵ	ϵ_H	$\epsilon_{C\|H}$	ϵ	ϵ_H	$\epsilon_{C\|H}$
HC-Search	16.05	03.98	**12.07**	19.71	5.82	**13.89**

- Selection loss $\boldsymbol{\epsilon_{C|H}}$ contributes more to the overall loss

Results: Loss decomposition analysis

	Phoneme			Scene labeling		
Error	ϵ	ϵ_H	$\epsilon_{C\|H}$	ϵ	ϵ_H	$\epsilon_{C\|H}$
HC-Search	16.05	03.98	12.07	19.71	05.82	13.89
C-Search	20.91	04.38	16.53	27.05	07.83	19.22

Results: Loss decomposition analysis

	Phoneme			Scene labeling		
Error	ϵ	ϵ_H	$\epsilon_{C\|H}$	ϵ	ϵ_H	$\epsilon_{C\|H}$
HC-Search	16.05	03.98	12.07	19.71	05.82	13.89
C-Search	20.91	04.38	16.53	27.05	07.83	19.22

- Improvement of HC-Search over C-Search is due to the improvement in the selection loss

Results: Loss decomposition analysis

	Phoneme			Scene labeling		
Error	ϵ	ϵ_H	$\epsilon_{C\|H}$	ϵ	ϵ_H	$\epsilon_{C\|H}$
HC-Search	16.05	03.98	12.07	19.71	05.82	13.89
C-Search	20.91	04.38	16.53	27.05	07.83	19.22

- Improvement of HC-Search over C-Search is due to the improvement in the selection loss
- **Clearly shows the advantage of separating the roles of heuristic and cost function**

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: Challenges

Input

Output

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

- **Joint prediction** of labels to exploit the relationships between labels
- **Automatically optimize the evaluation measure** of the real-world task

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

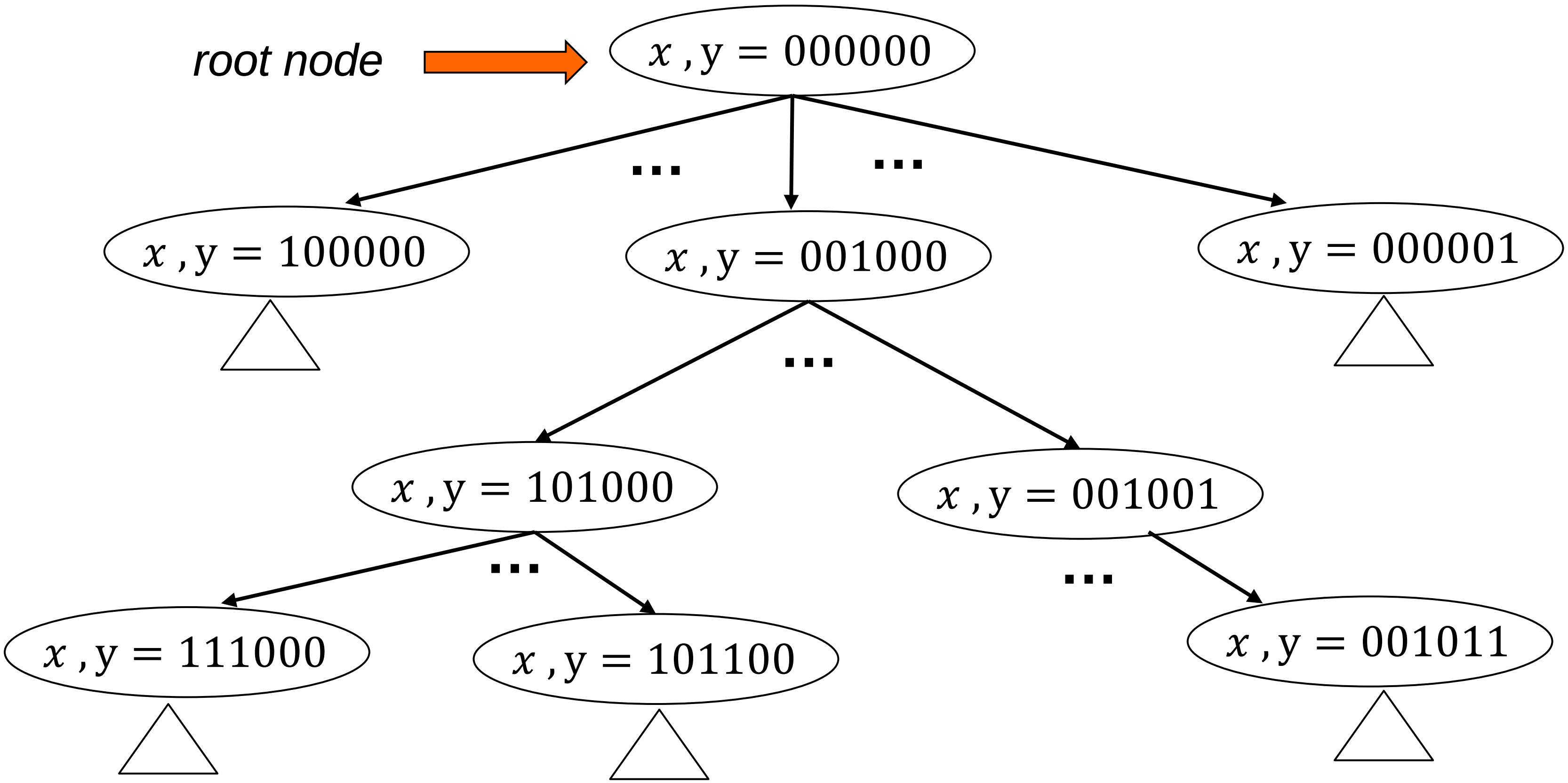
- Benchmark data

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)



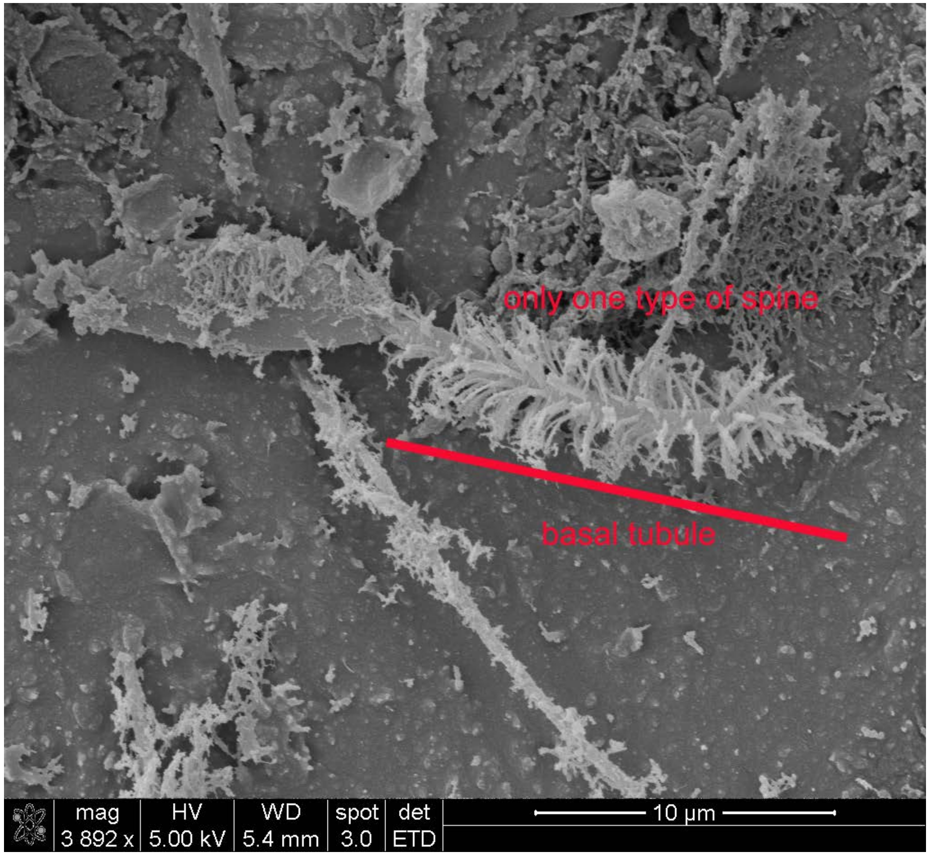
Multi-Label Prediction: Results

- **F1 Accuracy Results**

Algorithm	Scene	Emotions	Medical	Genbase	Yeast	Enron	LLog	Slashdot
BR	52.60	60.20	63.90	98.70	63.20	53.90	36.00	46.20
CC	59.10	57.50	64.00	99.40	63.20	53.30	26.50	44.90
ECC	68.00	62.60	65.30	99.40	64.60	59.10	32.20	50.20
M2CC	68.20	63.20	65.40	99.40	64.90	59.10	32.30	50.30
CLR	62.20	66.30	66.20	70.70	63.80	56.50	22.70	46.60
CDN	63.20	61.40	68.90	97.80	64.00	58.50	36.60	53.10
CCA	66.43	63.27	49.60	98.60	61.64	53.83	25.80	48.00
PIR	74.45	60.92	80.17	99.41	**65.47**	61.14	38.95	57.55
SML	68.50	64.32	68.34	**99.62**	64.32	57.46	34.95	55.73
RML	74.17	64.83	**80.73**	98.80	63.18	57.79	35.97	51.30
DecL	73.76	65.29	78.02	97.89	63.46	61.19	37.52	54.67
HC-Search	**75.89**	**66.17**	78.19	98.12	63.78	**62.34**	**39.76**	**57.98**

Doppa, J.R., Yu, J., Ma C., Fern, A., Tadepalli, P. HC-Search for Multi-Label Prediction: An Empirical Study. *American Association of Artificial Intelligence (AAAI) Conference* 2014.

Detecting Basal Tubules of Nematocysts

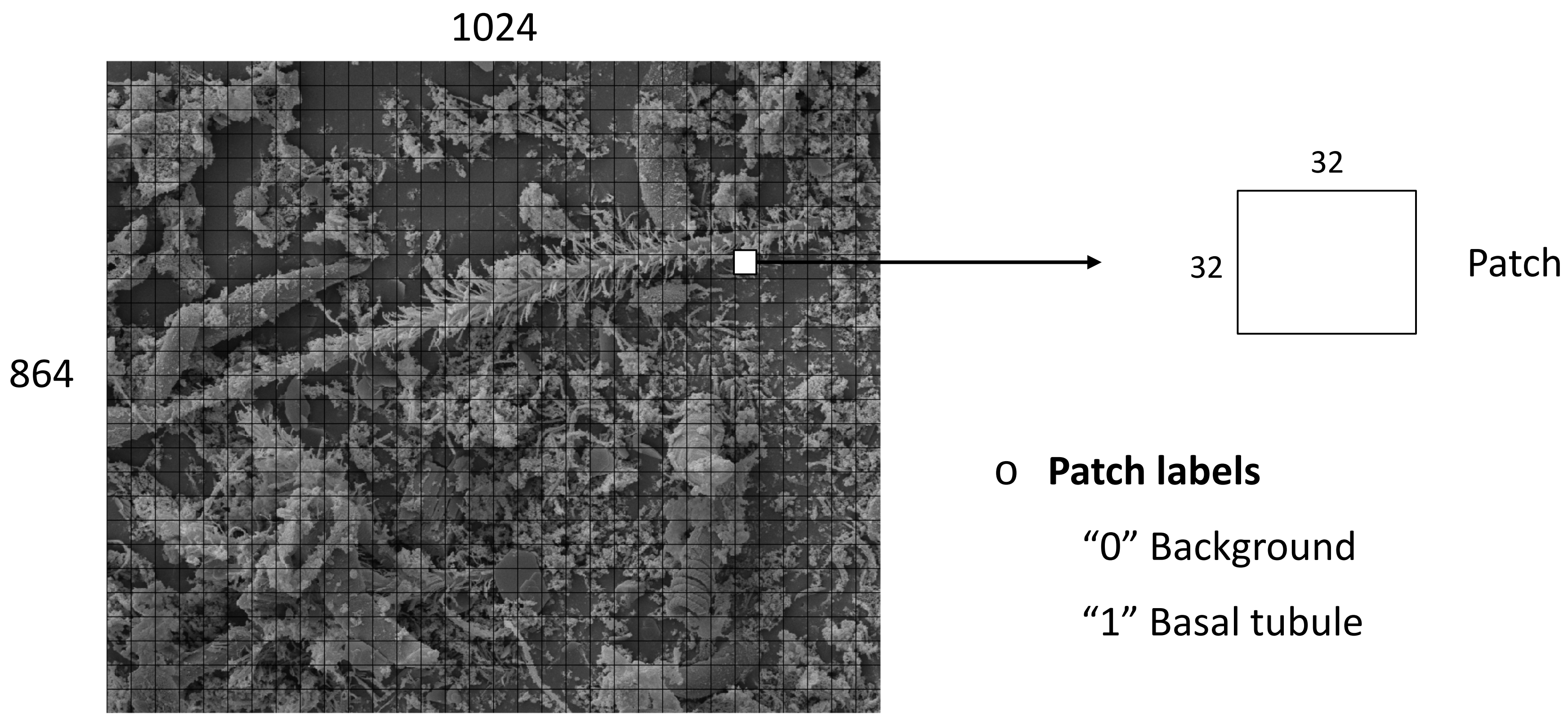


Challenges:

- Imaged against significant background clutter (unavoidable)
- Biological objects have highly-deformable parts

Detecting Basal Tubules of Nematocysts

- **Experimental Setup**
  - 80 images (training); 20 images (validation); 30 images (testing)



- **Patch labels**
  - "0" Background
  - "1" Basal tubule

Detecting Basal Tubules of Nematocysts

- **Baselines**
  - IID Classifier
  - Pairwise CRFs (w/ ICM, LBP, Graph-cuts)
- **HC-Search**
  - Flipbit space (IID classifier + flip patch labels)
  - Randomized Segmentation space

Basal Tubule Detection Results

Algorithm	Precision	Recall	F1
SVM	0.675	0.147	0.241
Logistic Regression	0.605	0.129	0.213

Basal Tubule Detection Results

Algorithm	Precision	Recall	F1
SVM	0.675	0.147	0.241
Logistic Regression	0.605	0.129	0.213
Pairwise CRF (w/ ICM)	0.432	0.360	**0.393**
Pairwise CRF (w/ LBP)	0.545	0.091	0.156
Pairwise CRF (w/ GC)	0.537	0.070	0.124

Basal Tubule Detection Results

Algorithm	Precision	Recall	F1
SVM	0.675	0.147	0.241
Logistic Regression	0.605	0.129	0.213
Pairwise CRF (w/ ICM)	0.432	0.360	0.393
Pairwise CRF (w/ LBP)	0.545	0.091	0.156
Pairwise CRF (w/ GC)	0.537	0.070	0.124
HC-Search (w/ Flipbit)	0.472	0.545	**0.506**

Lam, M., Doppa, J.R., Xu, S.H., Todorovic, S., Dietterich, T.G., Reft, A., Daly, M. Learning to Detect Basal Tubules of Nematocysts in SEM Images. *IEEE Workshop on Computer Vision for Accelerated Biosciences (CVAB)* 2013.

Basal Tubule Detection Results

Algorithm	Precision	Recall	F1
SVM	0.675	0.147	0.241
Logistic Regression	0.605	0.129	0.213
Pairwise CRF (w/ ICM)	0.432	0.360	0.393
Pairwise CRF (w/ LBP)	0.545	0.091	0.156
Pairwise CRF (w/ GC)	0.537	0.070	0.124
HC-Search (w/ Flipbit)	0.379	0.603	0.465
HC-Search (w/ Randomized)	0.831	0.651	**0.729**

Lam, M., Doppa, J.R., Todorovic, S., Dietterich, T.G. HC-Search for Structured Prediction in Computer Vision. *IEEE International Conference on Computer Vision (CVPR)* 2015.

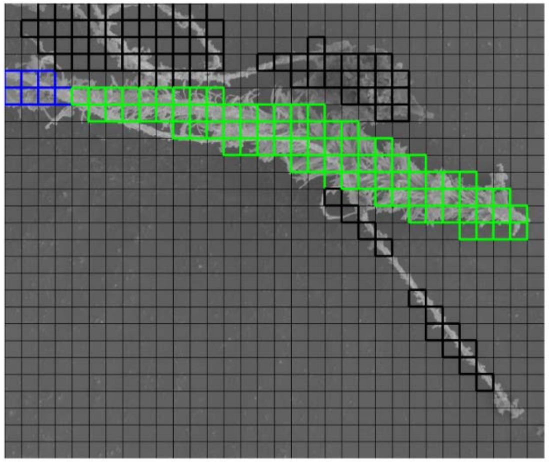
Basal Tubule Detection Results

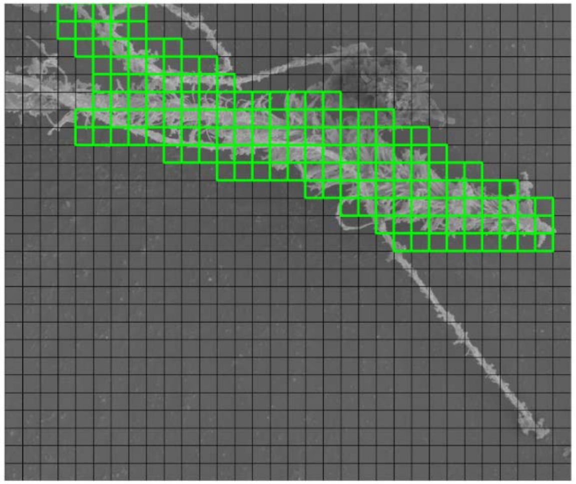
Algorithm	Precision	Recall	F1
SVM	0.675	0.147	0.241
Logistic Regression	0.605	0.129	0.213
Pairwise CRF (w/ ICM)	0.432	0.360	0.393
Pairwise CRF (w/ LBP)	0.545	0.091	0.156
Pairwise CRF (w/ GC)	0.537	0.070	0.124
HC-Search (w/ Flipbit)	0.379	0.603	0.465
HC-Search (w/ Randomized)	0.831	0.651	**0.729**

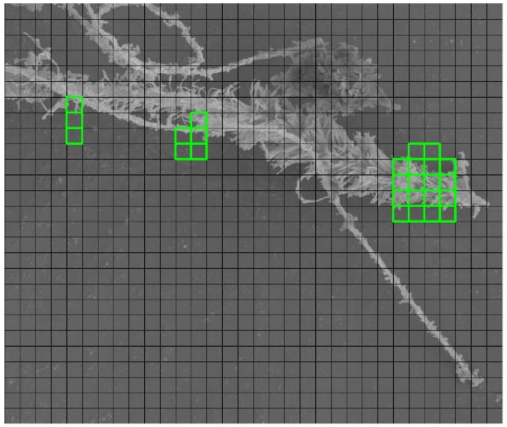
- **HC-Search** significantly outperforms all the other algorithms
- Performance critically depends on the **quality of the search space**

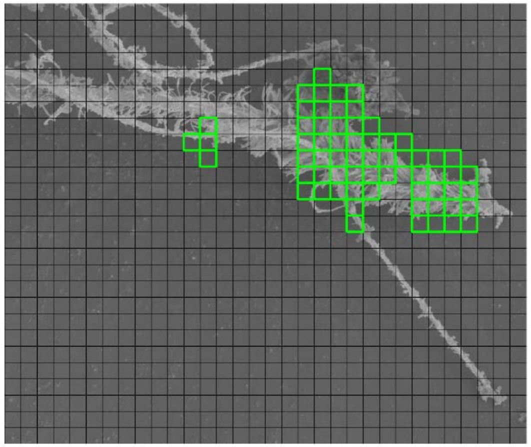
Basal Tubule Detection Results

- **Visual results:**


Ground-truth output


HC-Search


CRF w/ Graph cuts


CRF w/ LBP

Results: Stanford Background Dataset

- Benchmark for scene labeling in vision community

Method	Accuracy (%)
Region Energy	76.4
SHL	76.9
RNN	78.1
ConvNet	78.8
ConvNet + NN	80.4
ConvNet + CRF	81.4
Pylon (No Bnd)	81.3
Pylon	**81.9**
HC-Search (w/ Randomized)	81.4

- HC-Search without using features from deep learning

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

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

Part 7: Future Directions

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

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