CS11-747 Neural Networks for NLP

Advanced Search Algorithms

Graham Neubig
https://phontron.com/class/nn4nlp2018/

Carnegie Mellon University
Language Technologies Institute

(Some Slides by Daniel Clothiaux)

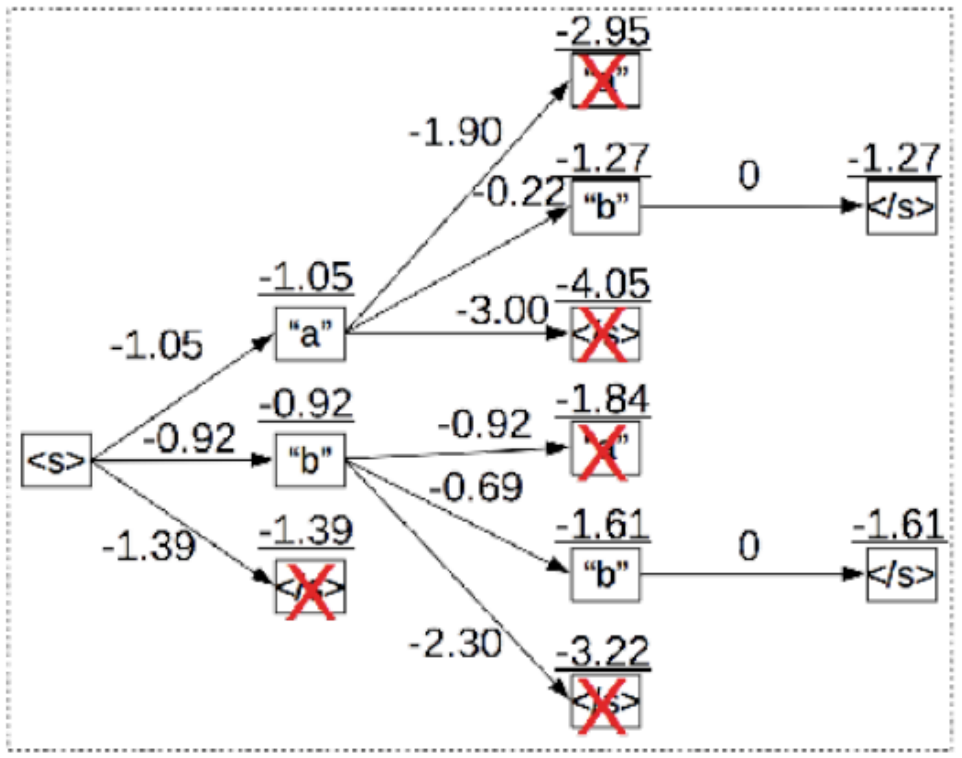
Why search?

- So far, decoding has mostly been greedy
    - Chose the most likely output from softmax, repeat
- Can we find a better solution?
- Oftentimes, yes!

Basic Search Algorithms

Beam Search

- Instead of picking the highest probability/score, maintain multiple paths
- At each time step
  - Expand each path
  - Choose a subset paths from the expanded set



Why will this help

Next word	P(next word)
Pittsburgh	0.4
New York	0.3
New Jersey	0.25
Other	0.05

Basic Pruning Methods
(Steinbiss et al. 1994)

- How to select which paths to keep expanding?
- **Histogram Pruning:** Keep exactly *k* paths at every time step
- **Score Threshold Pruning:** Keep all paths where score is within a threshold α of best score s_1

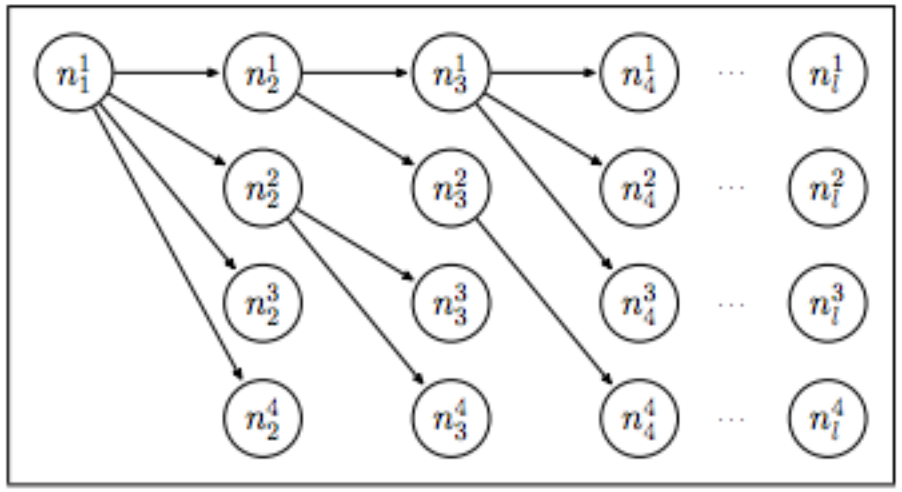
  $s_n + \alpha > s_1$

Prediction-based Pruning Methods (e.g. Stern et al. 2017)

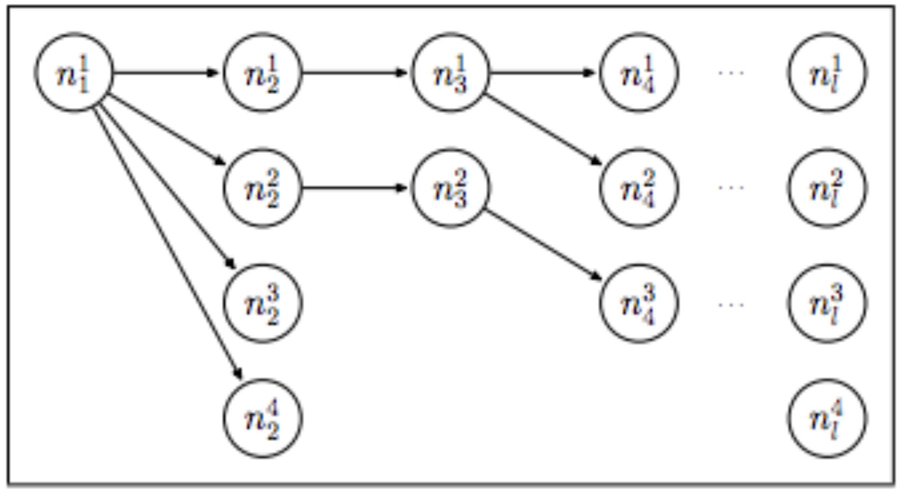
- A simple feed forward network predicts actions to prune
- This works well in parsing, as most of the possible actions are Open, vs. a few Closes and one Shift

Backtracking-based Pruning Methods

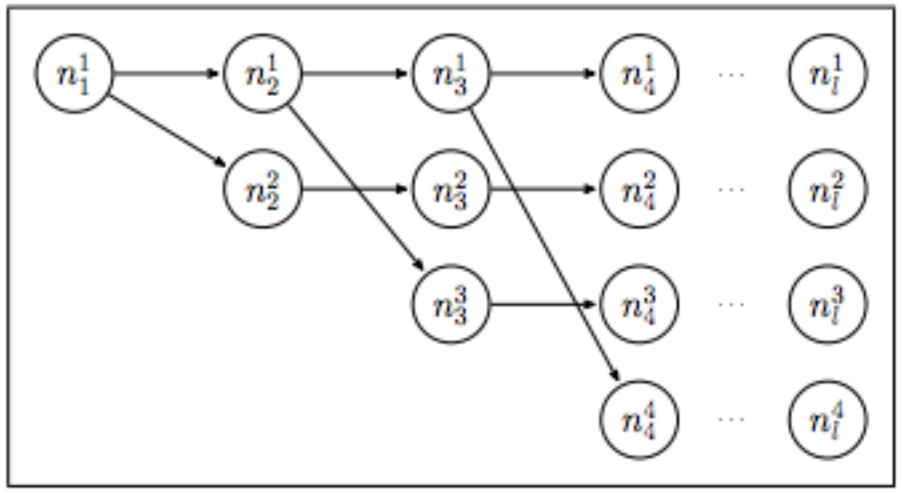
(Buckman et al, 2016)



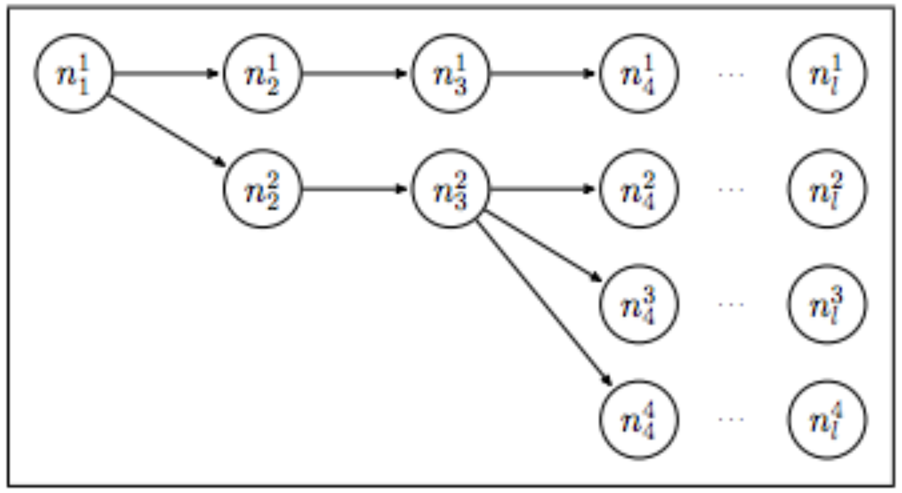
(a) Beam Search



(b) Dynamic Beam Search



(c) Selectional Branching



(d) Heuristic Backtracking

What beam size should I use?

- Larger beam sizes will be slower
- May not give better results
  - Sometimes result in shorter sequences
  - May favor high-frequency words
- Mostly done empirically -> experiment (range of 5-100?)

Variable length output sequences

- In many tasks (eg. MT), the output sequences will be of variable length
- Running beam search may then favor short sentences
- Simple idea:
    - Normalize by the length-divide by |N|
        - On the Properties of Neural Machine Translation: Encoder–Decoder (Cho et al., 2014)
- Can we do better?

More complicated normalization

‘Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation’ (Y Wu et al. 2016)

$$s(Y,X) = \log(P(Y|X))/lp(Y) + cp(X;Y)$$

$$lp(Y) = \frac{(5+|Y|)^\alpha}{(5+1)^\alpha}$$

$$cp(X;Y) = \beta * \sum_{i=1}^{|X|} \log(\min(\sum_{j=1}^{|Y|} p_{i,j}, 1.0)),$$

- X,Y: source, target sentence
- α: 0 < α < 1, normally in [0.6, 0.7]
- β: coverage penalty
- This is found empirically

Predict the output length
(Eriguchi et al. 2016)

- Add a penalty based off of length differences between sentences
- Calculate P(len(y) | len(x)) using corpus statistics

$$
\begin{aligned}
score(\boldsymbol{x}, \boldsymbol{y}) &= L_{\boldsymbol{x},\boldsymbol{y}} + \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}), \\
L_{\boldsymbol{x},\boldsymbol{y}} &= \log p(len(\boldsymbol{y}) | len(\boldsymbol{x})),
\end{aligned}
$$

Why do Bigger Beams Hurt, pt. 2

(Ott et. al. 2014)

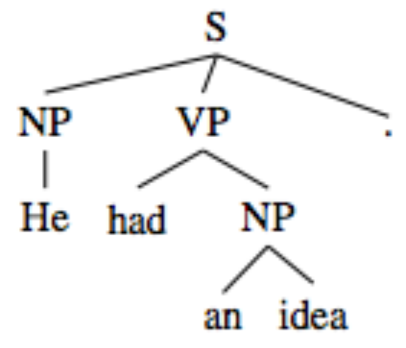
- They found that higher beam sizes:
  - Almost always lead to increased model loss
  - Often times lead to decreased evaluation score
- Why?
  - They theorize the model spreads it’s probability too much
  - Intrinsic (multiple translations can be good) and extrinsic uncertainty (bad training data, especially copies)
  - These combined mean individual good examples aren’t properly weighted, expanding beam compounds this problem

Beam Search for Disparate Action Spaces

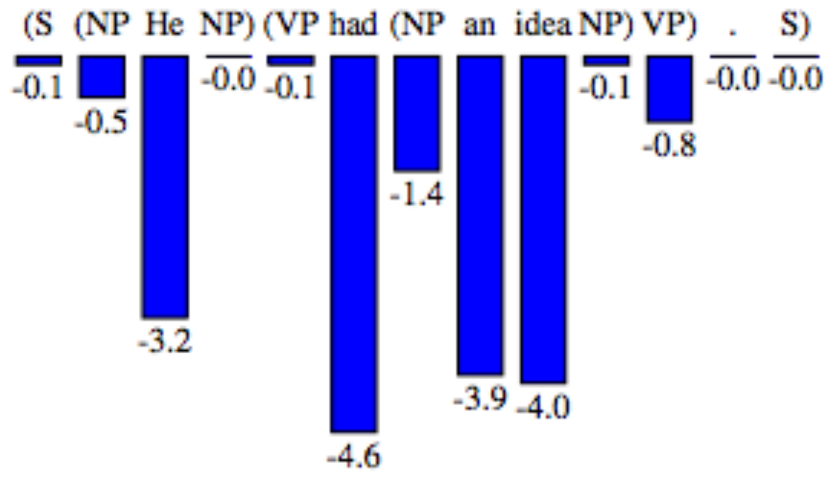
Dealing with disparity in actions

Effective Inference for Generative Neural Parsing
(Mitchell Stern et al., 2017)

- In generative parsing there are Shifts (or Generates) equal to the vocabulary size
- Opens equal to # of labels

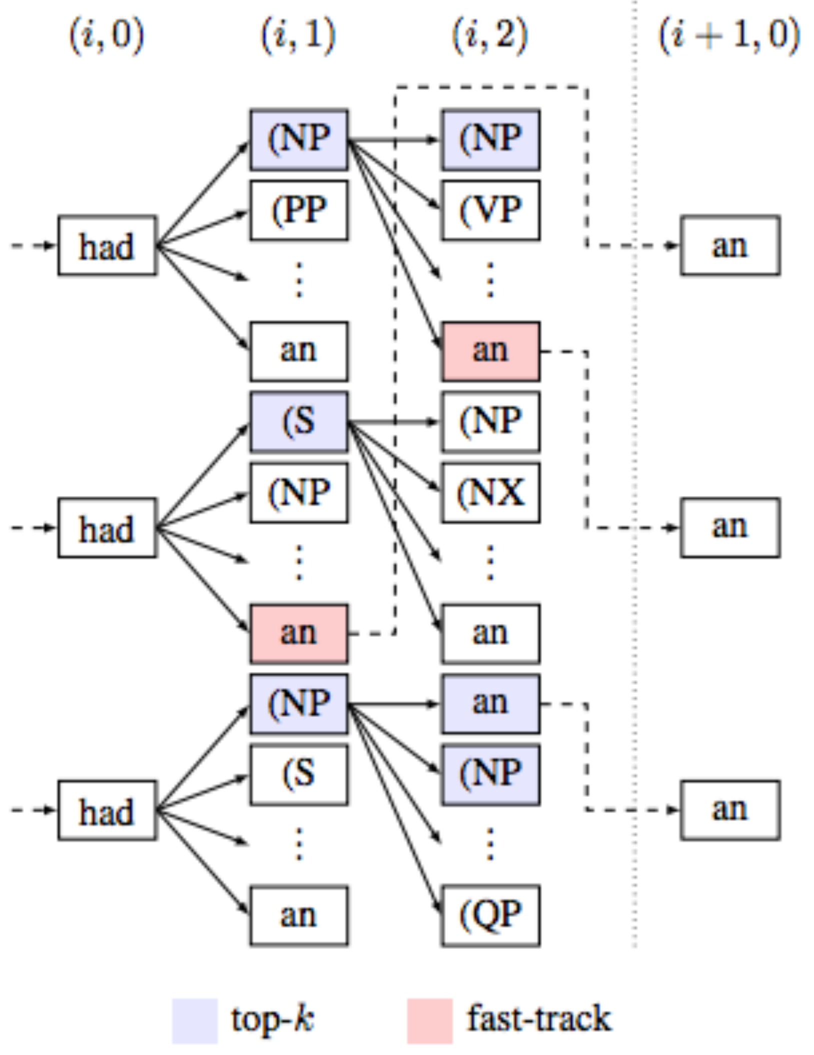


(S (NP He NP) (VP had (NP an idea NP) VP) . S)



Solution

- Group sequences of actions of the same length taken after the *i*th Shift.
- Create buckets based off of the number of Shifts and actions after the Shift
- Fast tracking:
    - To further reduce comparison bias, certain Shifts are immediately added to the next bucket

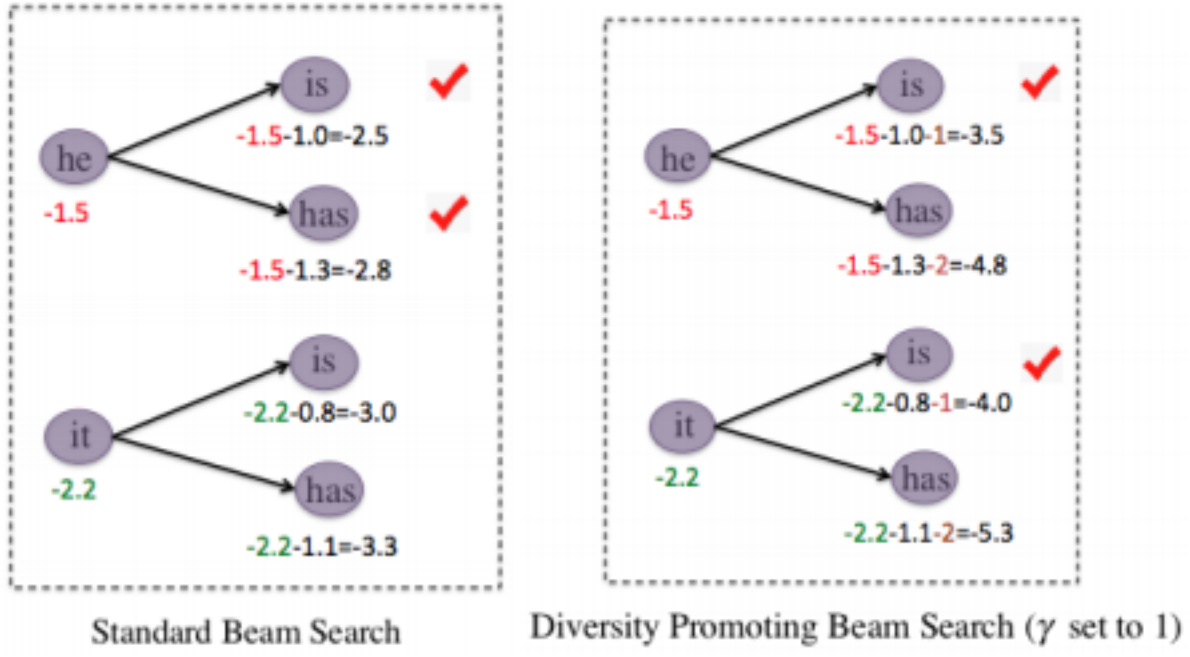


Improving Diversity in Search

Improving Diversity in top N Choices

Mutual Information and Diverse Decoding Improve Neural Machine Translation (Li et al., 2016)

- Entries in the beam can be very similar
- Improving the diversity of the top N list can help
- Score using source->target and target-> source translation models, language model



Improving Diversity through Sampling

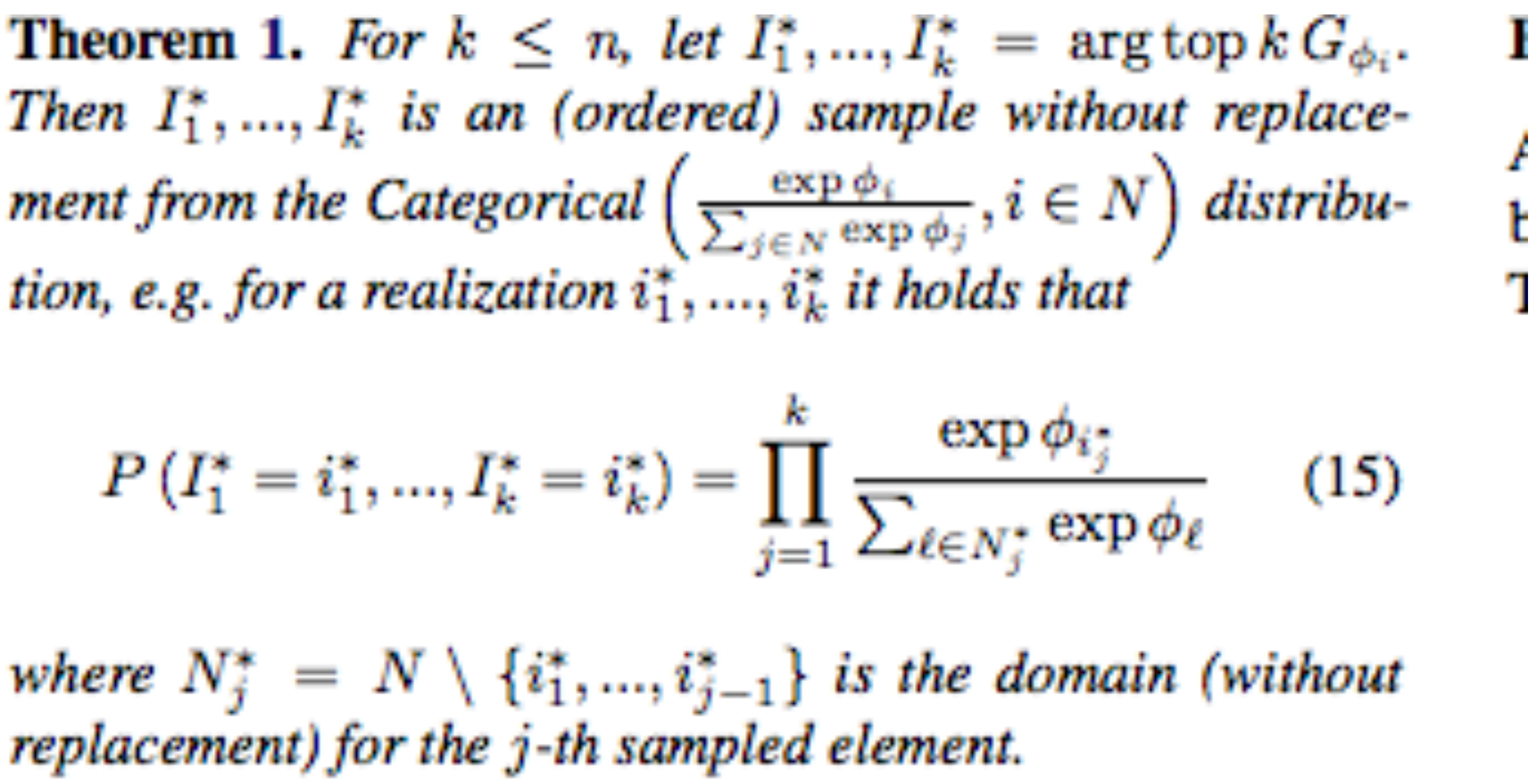
(Shao et al., 2017)

- Stochastically sampling from the softmax gives great diversity!
- Unlike in translation, the distributions in conversation are less peaky
    - This makes sampling reasonable

Sampling without Replacement

Stochastic Beams and Where to Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement (Kool et. al 2019)

- Gumbel distribution: If U is uniform(0,1)
  - G(ϕ) = ϕ – log(– log U)
- Perturbing log probabilities log-probabilities with Gumbel noise and finding the largest element is sampling from a categorical distribution without replacement
- A nice description of the Gumbel max trick can be found in the reading



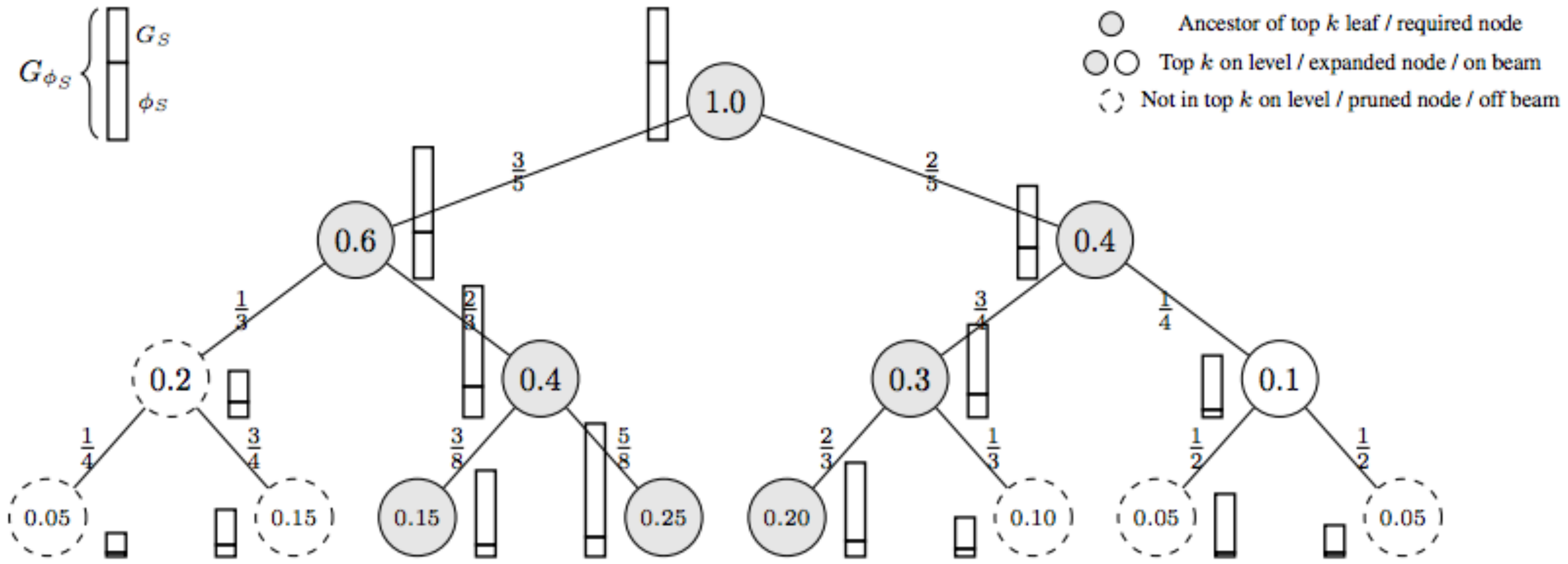
Theorem 1. *For $k \le n$, let $I_1^*, \dots, I_k^* = \arg\operatorname{top} k\, G_{\phi_i}$. Then $I_1^*, \dots, I_k^*$ is an (ordered) sample without replacement from the Categorical $\left(\frac{\exp \phi_i}{\sum_{j \in N} \exp \phi_j}, i \in N\right)$ distribution, e.g. for a realization $i_1^*, \dots, i_k^*$ it holds that*

$$P\left(I_1^* = i_1^*, \dots, I_k^* = i_k^*\right) = \prod_{j=1}^{k} \frac{\exp \phi_{i_j^*}}{\sum_{\ell \in N_j^*} \exp \phi_\ell} \qquad (15)$$

where $N_j^* = N \setminus \{i_1^*, \dots, i_{j-1}^*\}$ is the domain (without replacement) for the j-th sampled element.

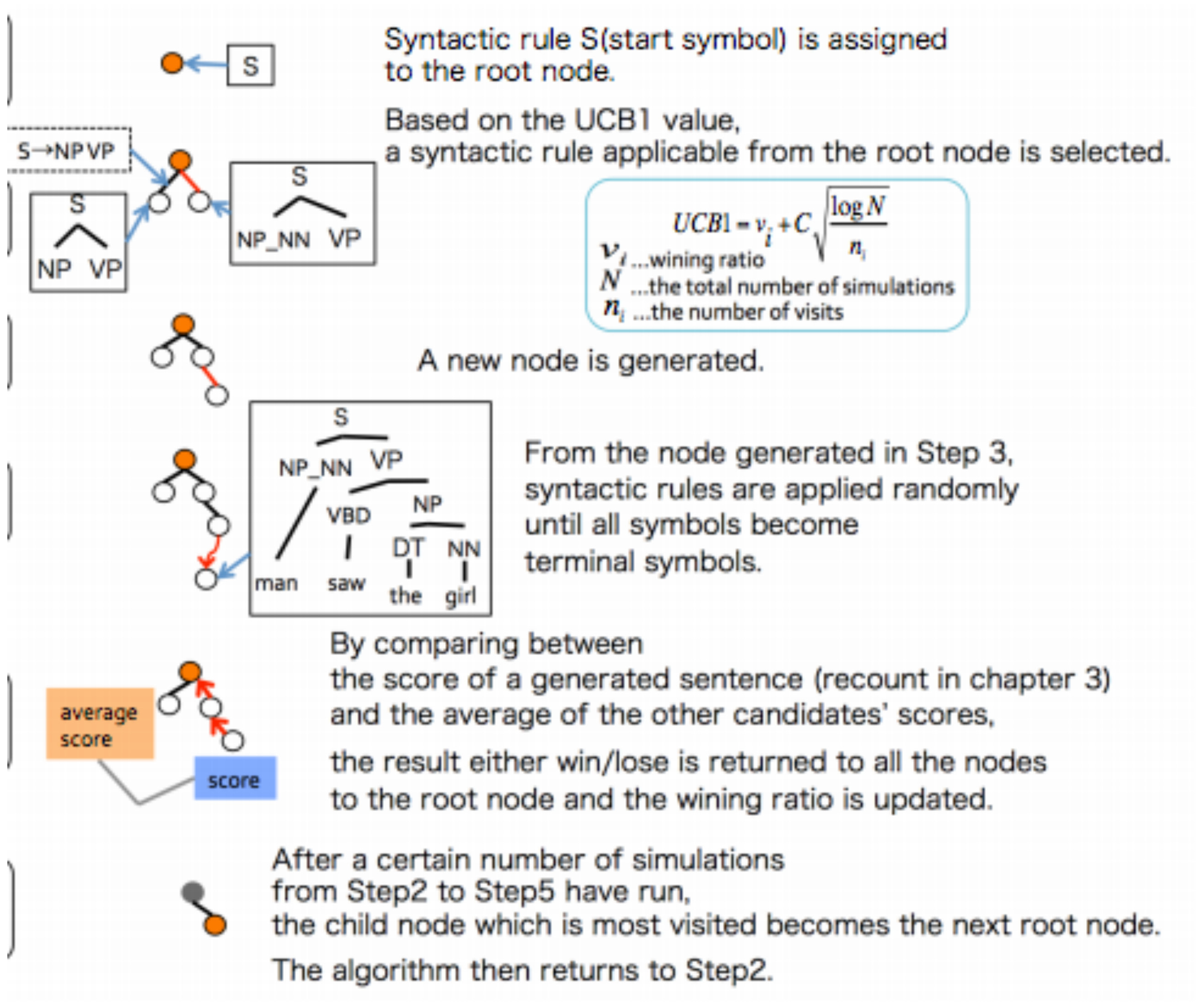
Sampling without Replacement (con't)

Stochastic Beams and Where to Find Them



Monte-Carlo Tree Search

Human-like Natural Language Generation Using Monte Carlo Tree Search



Syntactic rule S(start symbol) is assigned to the root node.

Based on the UCB1 value, a syntactic rule applicable from the root node is selected.

$$UCB1 = v_i + C\sqrt{\frac{\log N}{n_i}}$$

v_i ...wining ratio
N ...the total number of simulations
n_i ...the number of visits

A new node is generated.

From the node generated in Step 3, syntactic rules are applied randomly until all symbols become terminal symbols.

By comparing between the score of a generated sentence (recount in chapter 3) and the average of the other candidates' scores, the result either win/lose is returned to all the nodes to the root node and the wining ratio is updated.

After a certain number of simulations from Step2 to Step5 have run, the child node which is most visited becomes the next root node.

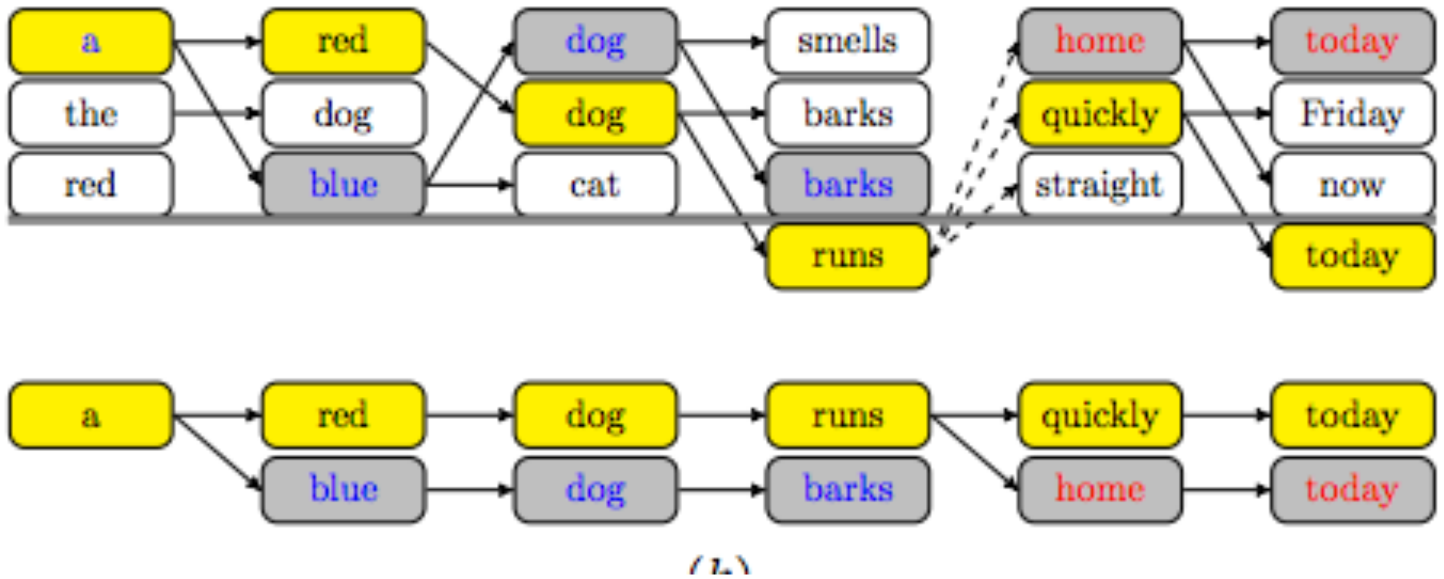
The algorithm then returns to Step2.

Incorporating Search in Training

Using beam search in training

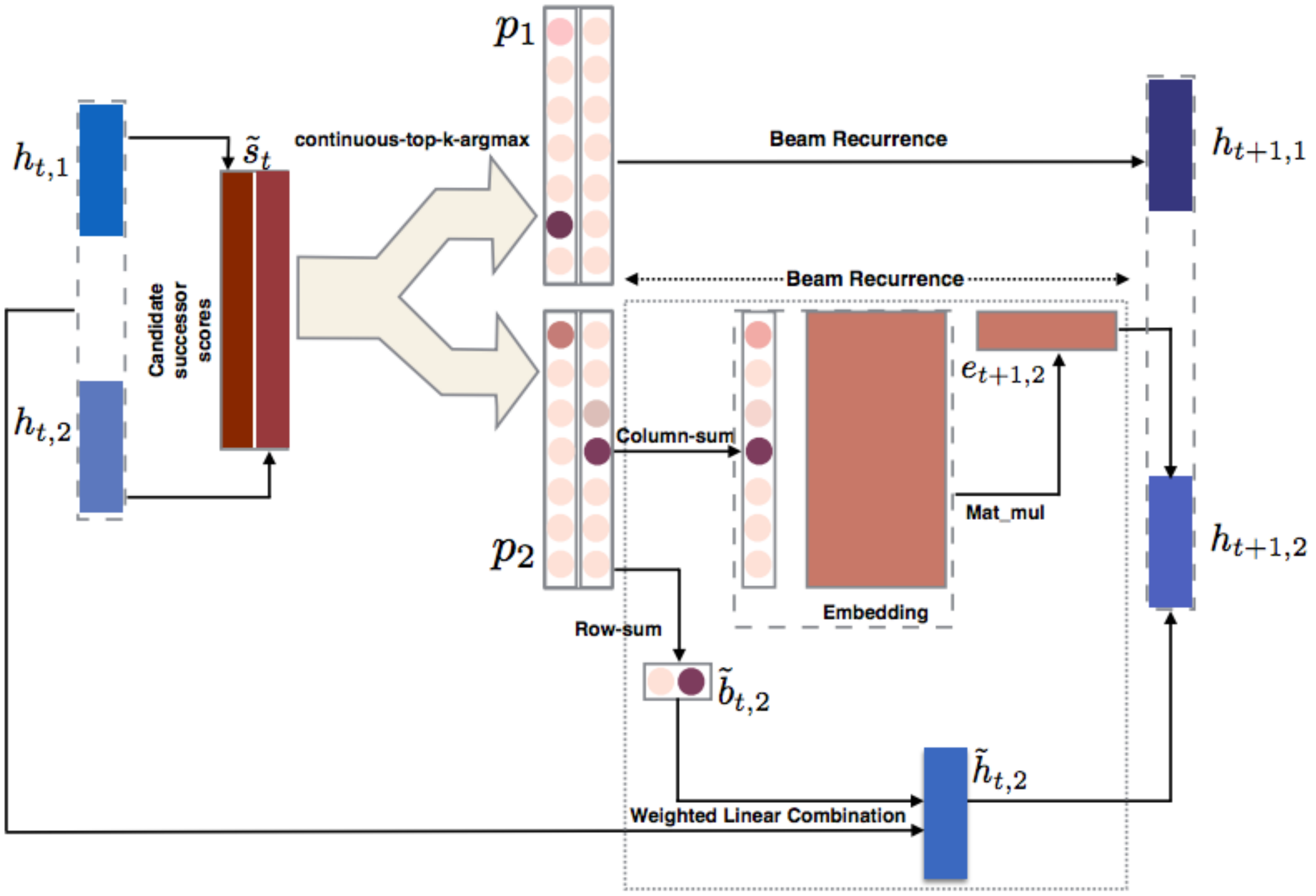
Sequence-to-Sequence Learning
as Beam-Search Optimization (Wiseman et al., 2016)

- Decoding with beam search has biases
  - Exposure: Model not exposed to errors during training
  - Label: scores are locally normalized
- Possible solution: train with beam search



More beam search in training

A Continuous Relaxation of Beam Search for End-to-end Training of Neural Sequence Models (Goyal et al., 2017)



A* and Look-ahead algorithms

A* search

- Basic idea:
  - Iteratively expand paths that have the cheapest total cost along the path
  - total cost = cost to current point + estimated cost to goal

- $f(n) = g(n) + h(n)$
    - $g(n)$: cost to current point
    - $h(n)$: estimated cost to goal
    - h should be admissible and consistent

Classical A* parsing
(Klein et al., 2003)

- PCFG based parser
- Inside (g) and outside (h) scores are maintained
    - Inside: cost of building this constituent
    - Outside: cost of integrating constituent with rest of tree

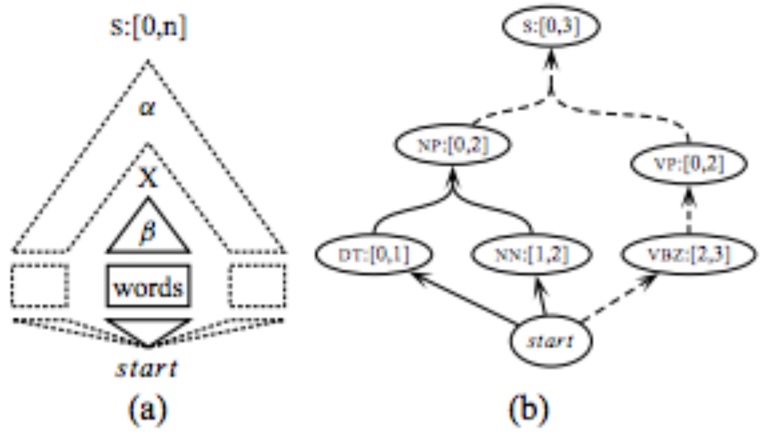
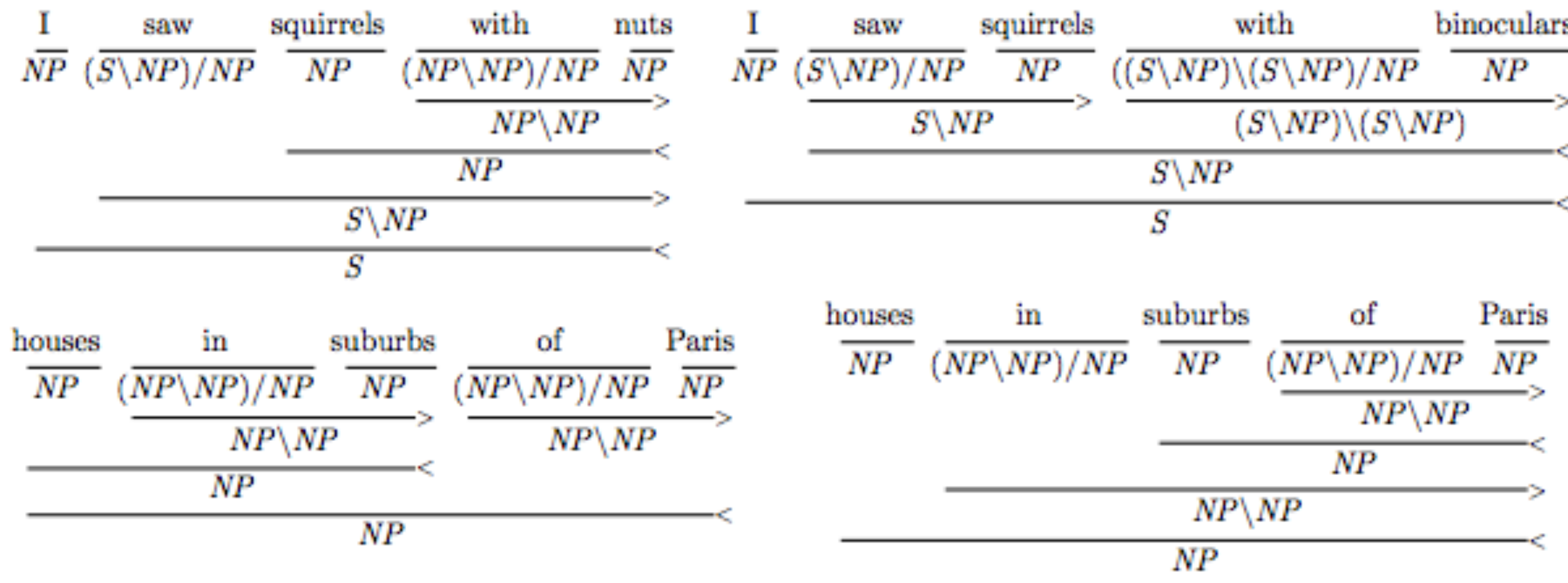


Figure 1: A* edge costs. (a) The cost of an edge X is a combination of the cost to build the edge (the Viterbi inside score β) and the cost to incorporate it into a root parse (the Viterbi outside score α). (b) In the corresponding hypergraph, we have exact values for the inside score from the explored hyperedges (solid lines), and use upper bounds on the outside score, which estimate the dashed hyperedges.

Adoption with neural networks: CCG Parsing

(Lewis et al. 2014)

CCG Parsing:

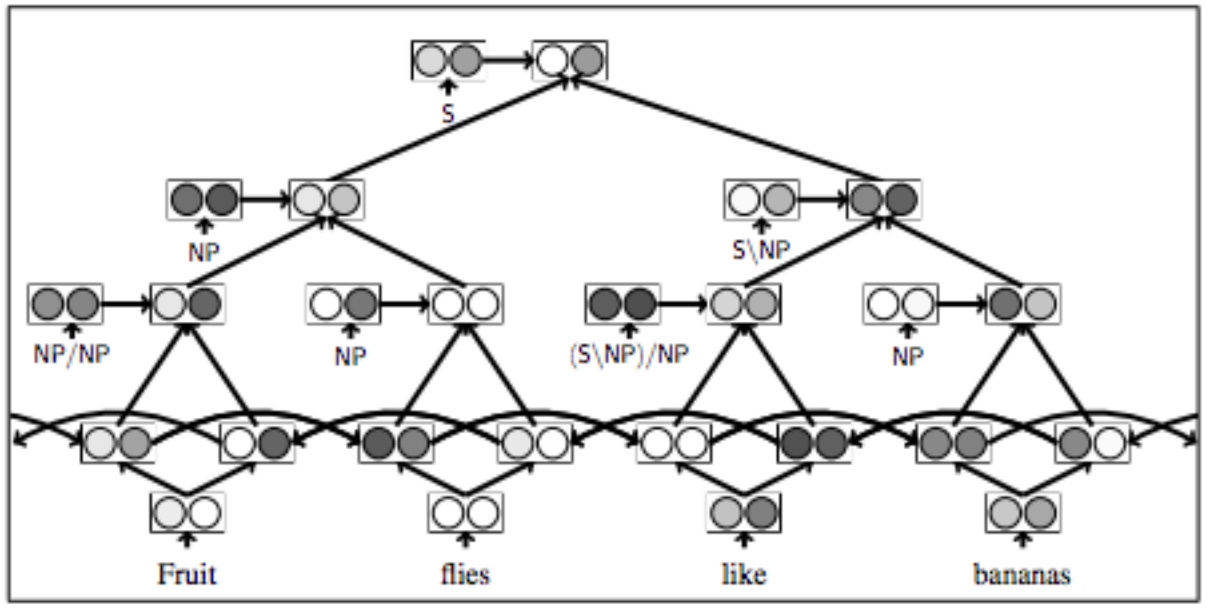

I saw squirrels with nuts
NP (S\NP)/NP NP (NP\NP)/NP NP
NP\NP
NP
S\NP
S
I saw squirrels with binoculars
NP (S\NP)/NP NP ((S\NP)\(S\NP))/NP NP
S\NP
(S\NP)\(S\NP)
S\NP
S
houses in suburbs of Paris
NP (NP\NP)/NP NP (NP\NP)/NP NP
NP\NP
NP\NP
NP
NP
houses in suburbs of Paris
NP (NP\NP)/NP NP (NP\NP)/NP NP
NP\NP
NP
NP\NP
NP

- A* for parsing
  - g(n): sum of encoded LSTM scores over current span
  - h(n): sum of maximum encoded scores for each constituent outside of current span

Is the heuristic admissible?

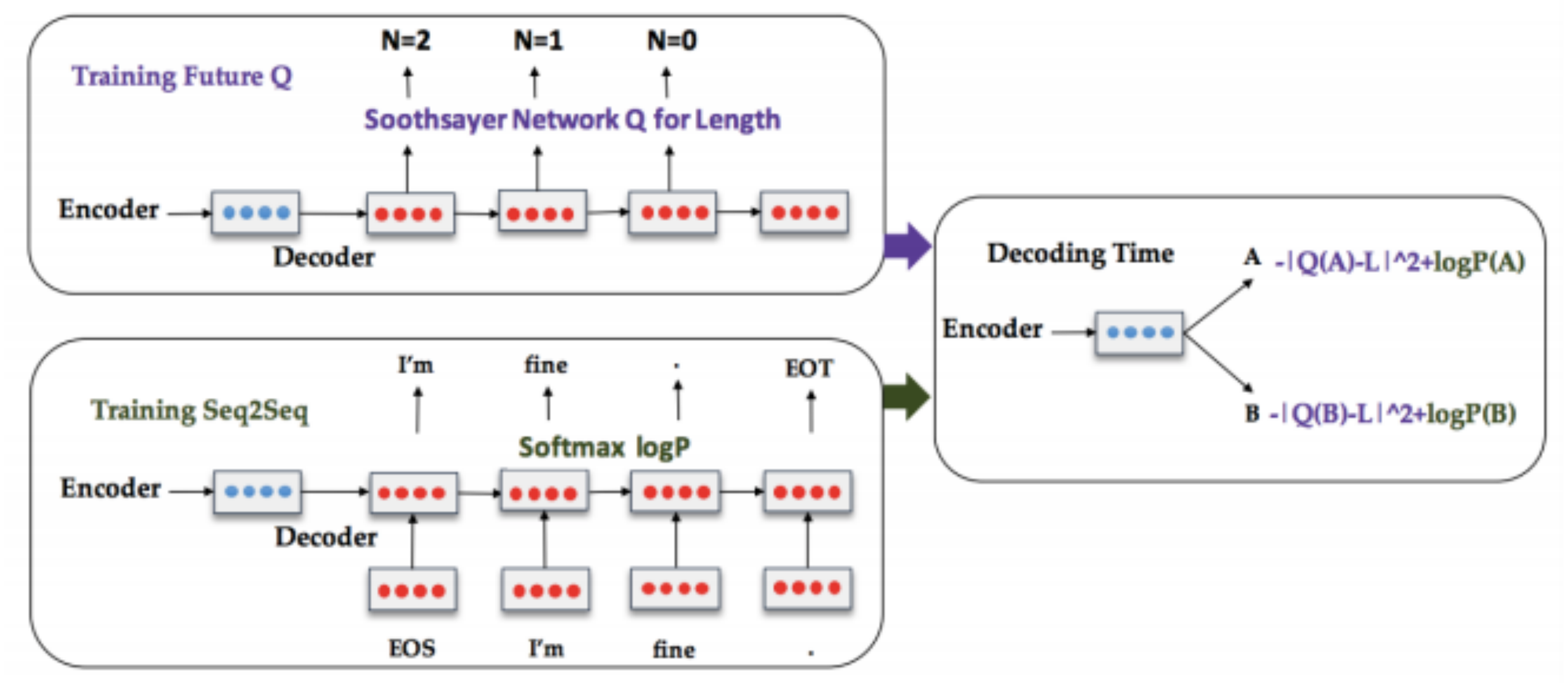
(Lee et al. 2016)

- No!
- Fix this by adding a global model score < 0 to the elements outside of the current span
  - This makes the estimated cost lower than the actual cost
- Global model: tree LSTM over completed parse
  - This is significantly slower than the embedding LSTM, so first evaluate g(n), then lazily expand good scores



Estimating future costs

Li et al., 2017)



A* search: benefits and drawbacks

- Benefits:
    - With heuristic, has nice optimality guarantees
    - Strong results in CCG parsing
- Drawbacks:
    - Needs more construction than beam search, can't easily throw on existing model
    - Requires a good heuristic for optimality guarantees

Actor Critic
(Bahdanau et. al., 2017)

- Basic idea:
  - Use Neural Model as an actor that predicts actions (say, the next word)
  - Use a critic to predict final reward (in this case, BLEU) for MT models
  - Actor trained similarly to REINFORCE, critic trained with TD

Actor Critic (continued)

Actor: $$\widehat{\frac{dV}{d\theta}} = \sum_{k=1}^{M} \sum_{t=1}^{T} \sum_{a \in \mathcal{A}} \frac{dp(a|\hat{Y}^k_{1\ldots t-1})}{d\theta} Q(a; \hat{Y}^k_{1\ldots t-1})$$

- T is the sequence, M in the set of examples, and a the potential next actions, Q reward

Critic: $$\frac{d}{d\phi}\left(\sum_{t=1}^{T}\left(\hat{Q}(\hat{y}_t; \hat{Y}_{1\ldots t-1}, Y) - q_t\right)^2 + \lambda_C C_t\right)$$

- C is a measure of reward over average reward similar to REINFORCE style algorithms

Other search algorithms

Particle Filters

(Buys et al., 2015)

- Similar to beam search
    - Think of it as beam search with a width that depends on certainty of it's paths
        - More certain, smaller, less certain, wider
- There are k total particles
- Divide particles among paths based off of probability of paths, dropping any path that would get <1 particle
- Compare after the same number of Shifts

Reranking
(Dyer et al. 2016)

- If you have multiple different models, using one to rerank outputs can improve performance
- Classically: use a target language language model to rerank the best outputs from an MT system
- Going back to the generative parsing problem, directly decoding from a generative model is difficult
- However, if you have both a generative model B and a discriminative model A
  - Decode with A then rerank with B
  - Results are superior to decoding then reranking with a separately trained B

Questions?