CS11-747 Neural Networks for NLP

Advanced Search Algorithms

Daniel Clothiaux

https://phontron.com/class/nn4nlp2017/

Carnegie Mellon University
Language Technologies Institute

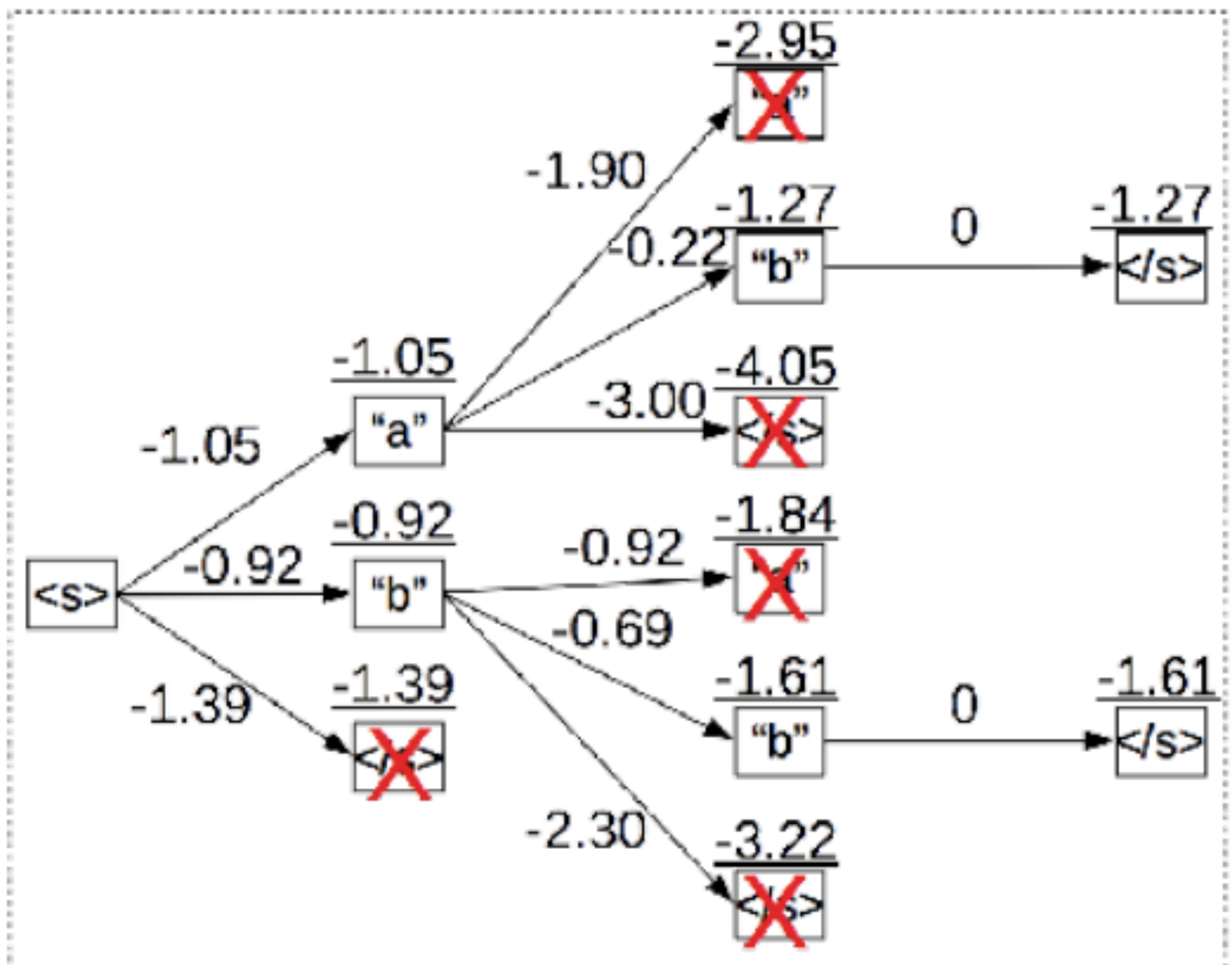
Why search?

- 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 top n paths from the expanded set



<s>
-1.05
-0.92
-1.39
"a"
"b"
-1.90
-0.22
-3.00
-0.92
-0.69
-2.30
-2.95
-1.27
"b"
-4.05
-1.84
-1.61
"b"
-3.22
0
0
-1.27
</s>
-1.61
</s>

Why will this help

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

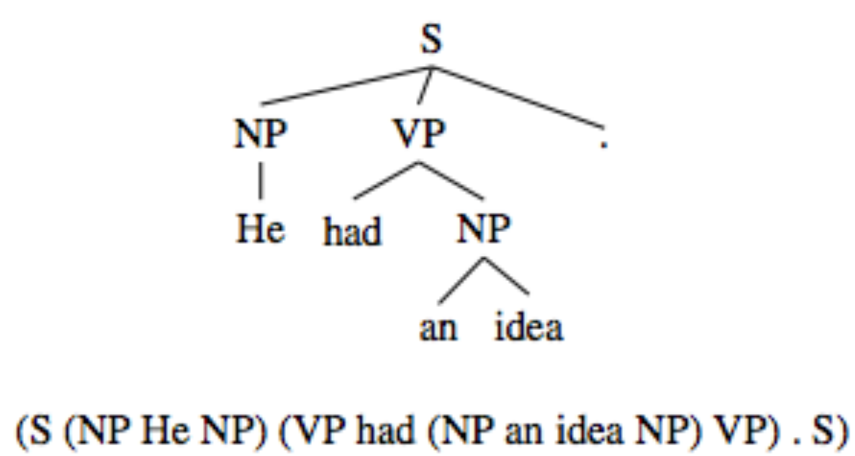
Potential Problems

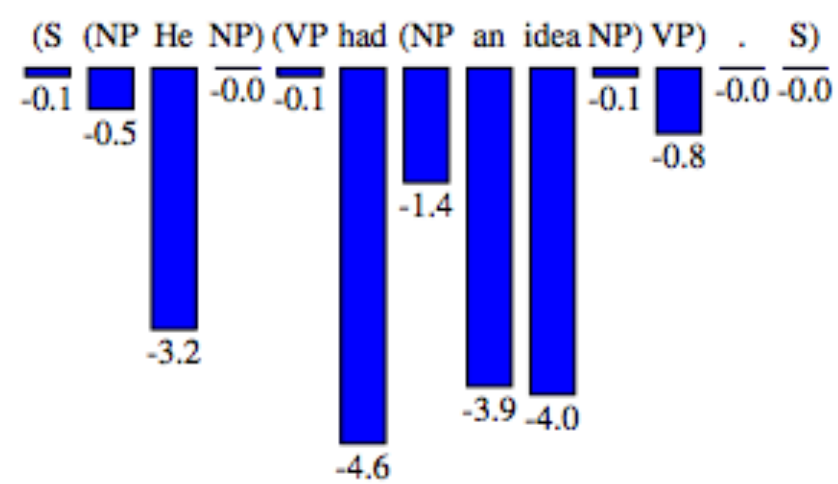
- Unbalanced action sets
- Larger beam sizes may be significantly slower
- Lack of diversity in beam
- Outputs of Variable length
  - Will not always improve evaluation metric

Dealing with disparity in actions

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

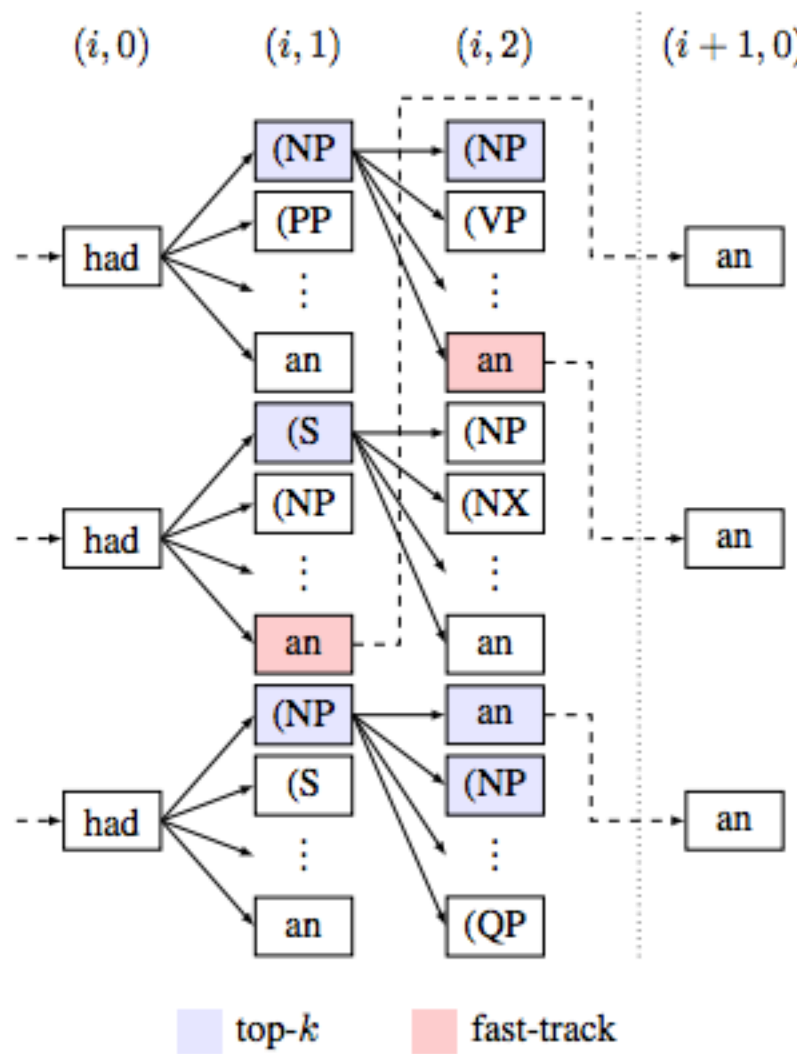
- In generative parsing there are Shifts (or Generates) equal to the vocabulary size
- Opens equal to # of labels





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



(i, 0)
(i, 1)
(i, 2)
(i + 1, 0)
had
(NP
(PP
an
(S
(NP
(VP
(NX
(QP
top-k
fast-track

Pruning

- Expanding each path with large beams is slow
- Pruning the search tree speeds things up
    - Remove paths from the tree
    - Predict what paths to expand

Threshold based pruning

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

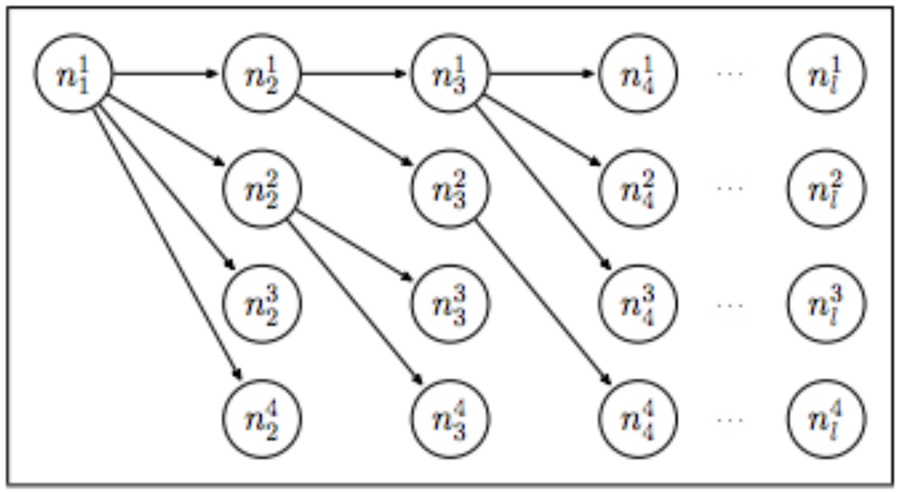
- Compare the path score with best path score
- Compare expanded node score with best node
  - If either falls beneath threshold, drop them

Predict what nodes to expand

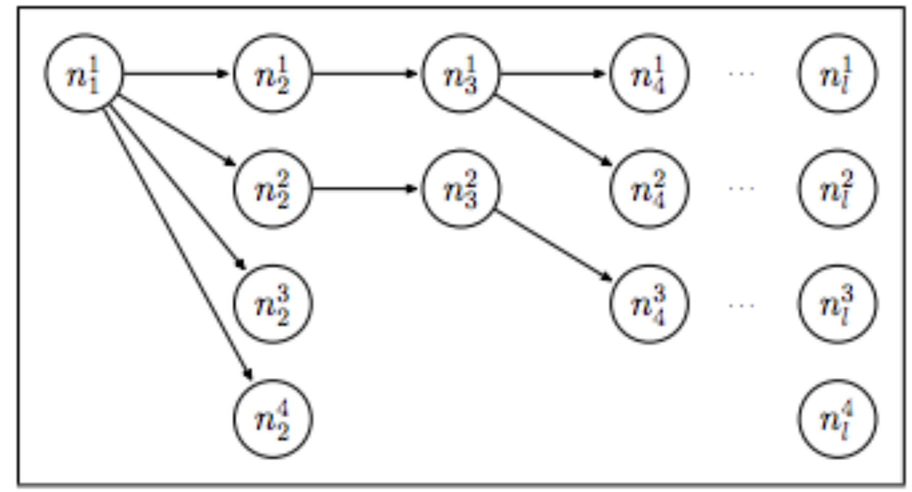
- Effective Inference for Generative Neural Parsing (Stern et al., 2017):
    - a simple feed forward network predicts actions to prune
    - This works well in parsing, as most of the possible actions are Open, vs. a few Closes and one Shift
- Transition-Based Dependency Parsing with Heuristic Backtracking
    - Early cutoff based off of single Stack LSTM

Backtrack to points most likely to be wrong

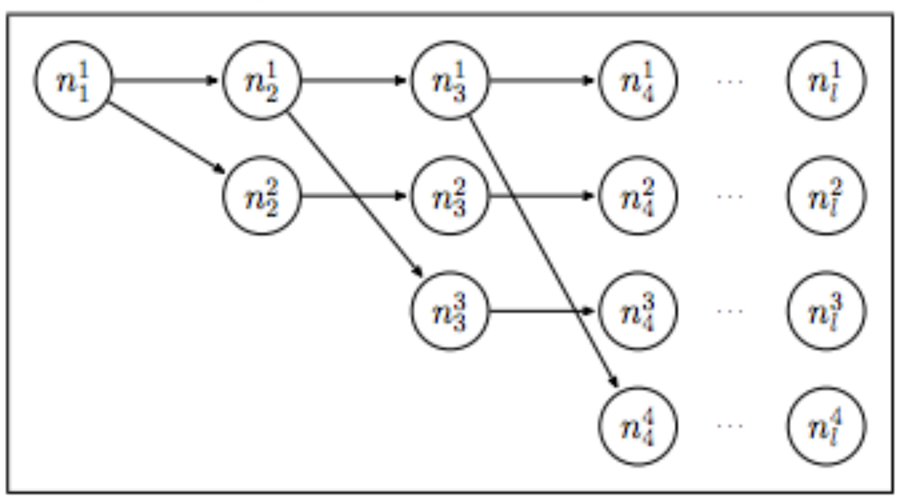
Transition-Based Dependency Parsing with Heuristic Backtracking (Buckman et al, 2016)



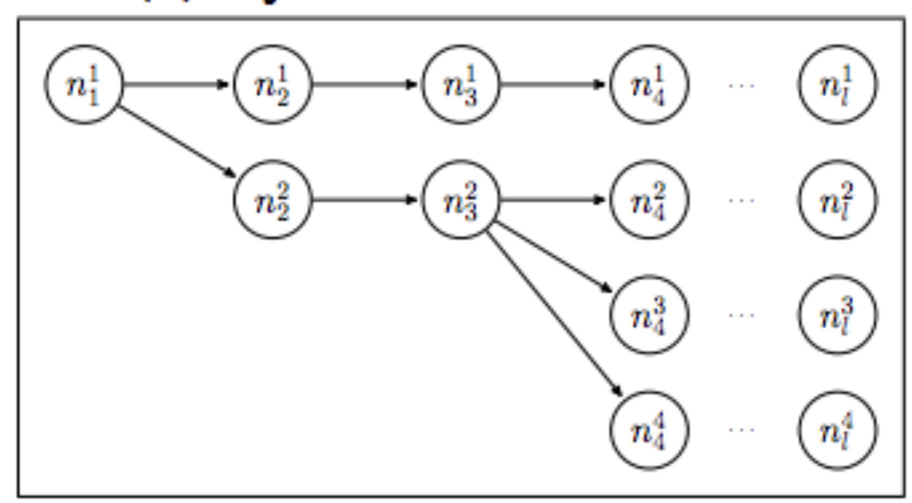
(a) Beam Search



(b) Dynamic Beam Search



(c) Selectional Branching

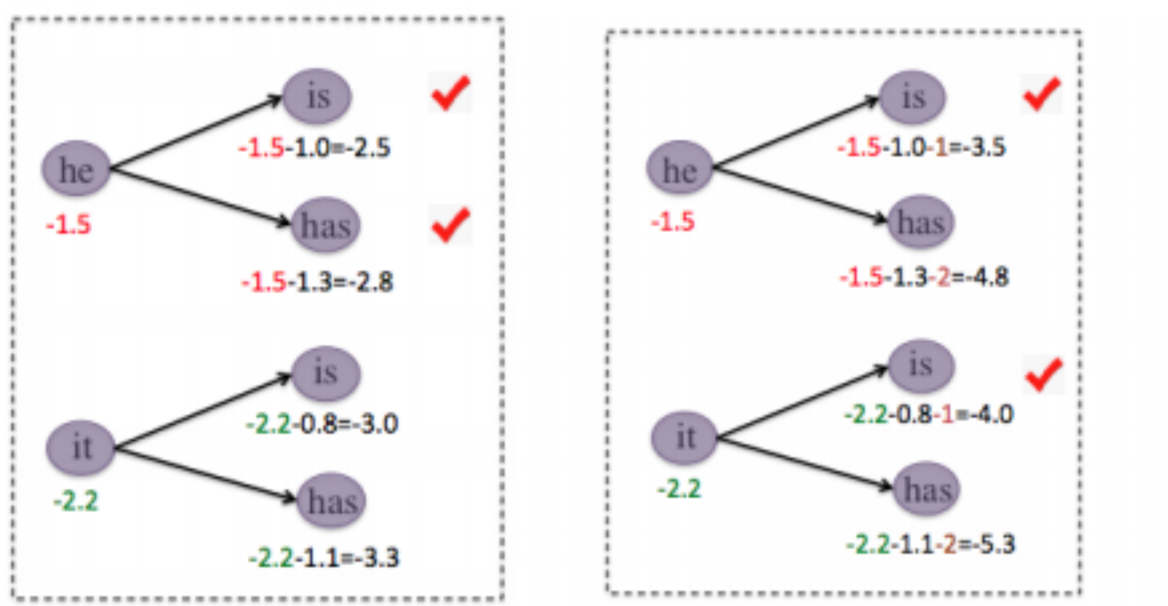


(d) Heuristic Backtracking

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



Standard Beam Search

Diversity Promoting Beam Search (γ set to 1)

Improving Diversity through Sampling

Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models (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

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

Tree-to-Sequence Attentional Neural Machine Translation
(Eriguchi et al. 2016)

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

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

$$L_{\boldsymbol{x},\boldsymbol{y}} = \log p(len(\boldsymbol{y}) | len(\boldsymbol{x})),$$

What beam size should I use?

- Larger beam sizes will be slower, and may not give better results
- Mostly done empirically-experiment!
- Many papers use less than 15, but I’ve seen as high as 1000

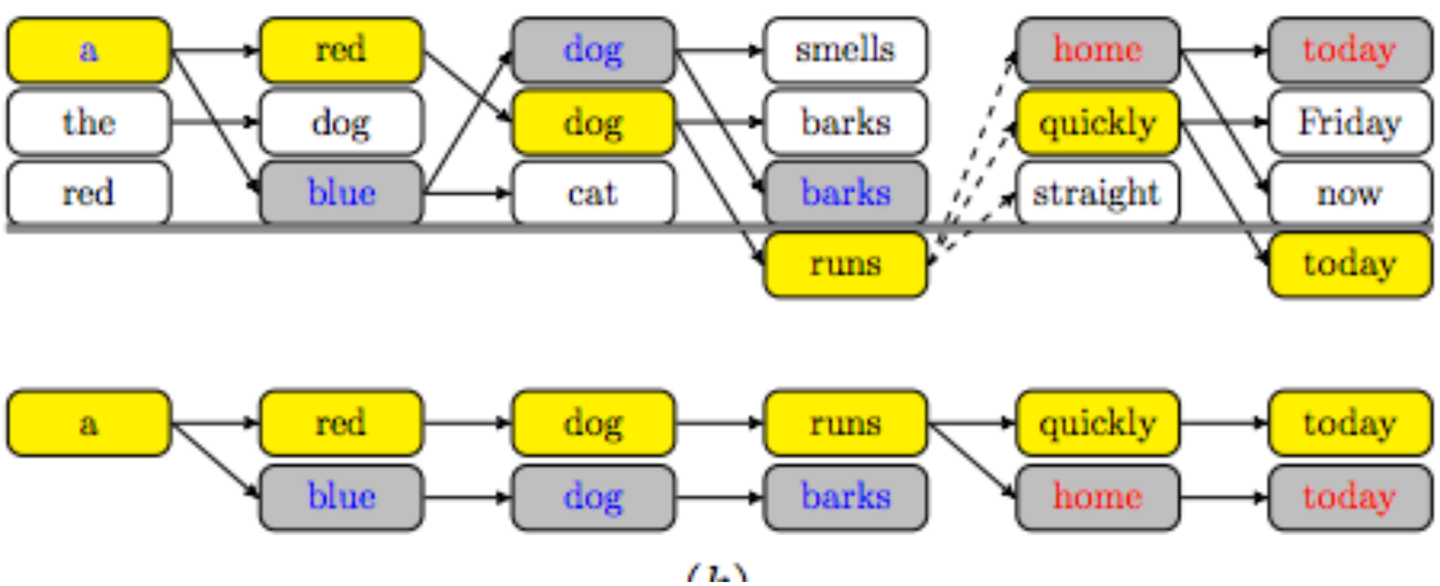
Beam Search-Benefits and Drawbacks

- Benefits:
    - Generally easy to implement off of an existing model
    - Guaranteed to not decrease model score
        - Otherwise, something's wrong
- Drawbacks
    - Larger beam sizes may be significantly slower
    - Will not always improve evaluation metric
    - Depending on how complicated you want to get, there will be a few more hyper-parameters to tune

Using beam search in training

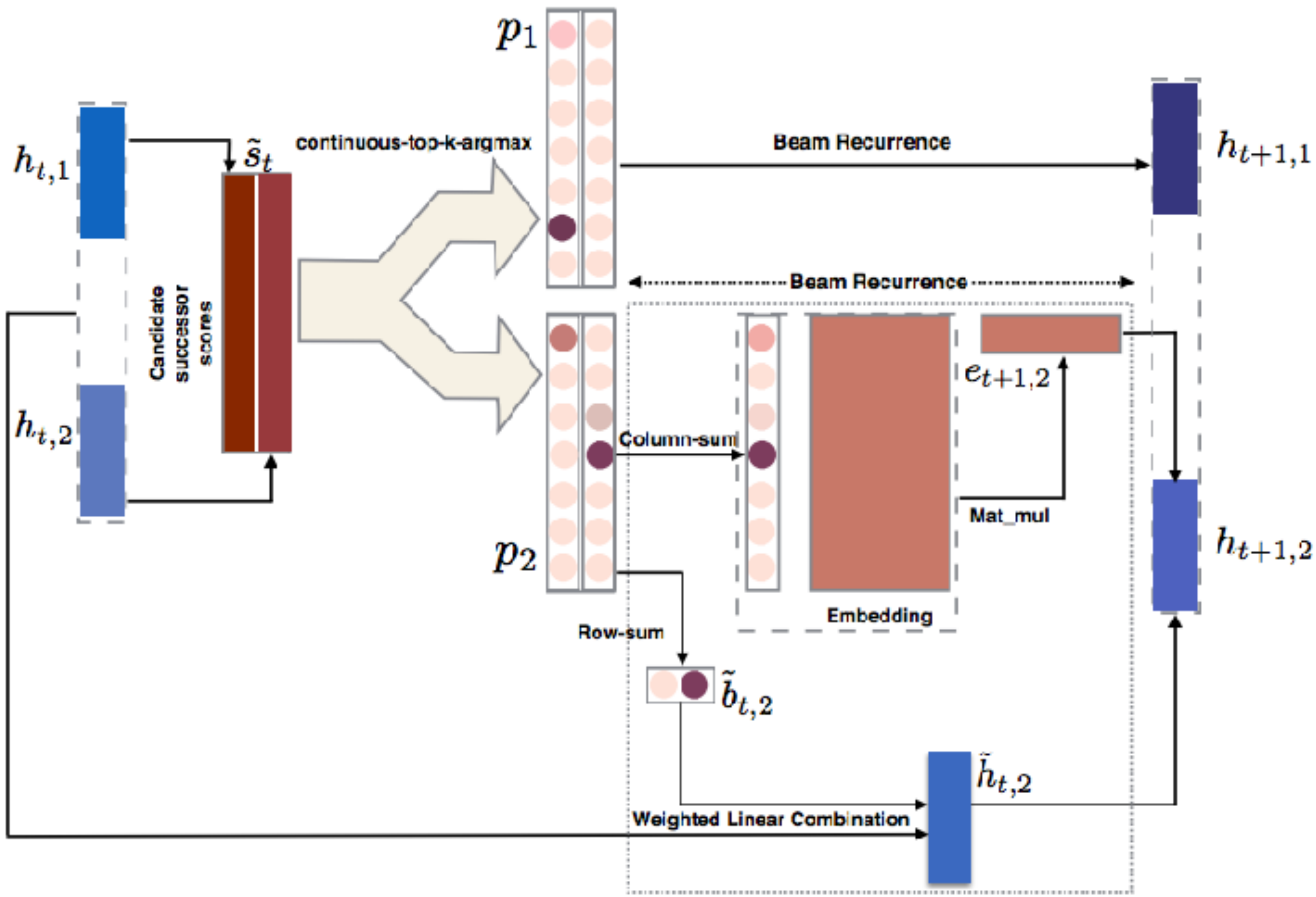
Sequence-to-Sequence Learning
as Beam-Search Optimization (Wiseman et al., 2016)

- 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* 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

A* Parsing: Fast Exact Viterbi Parse Selection (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

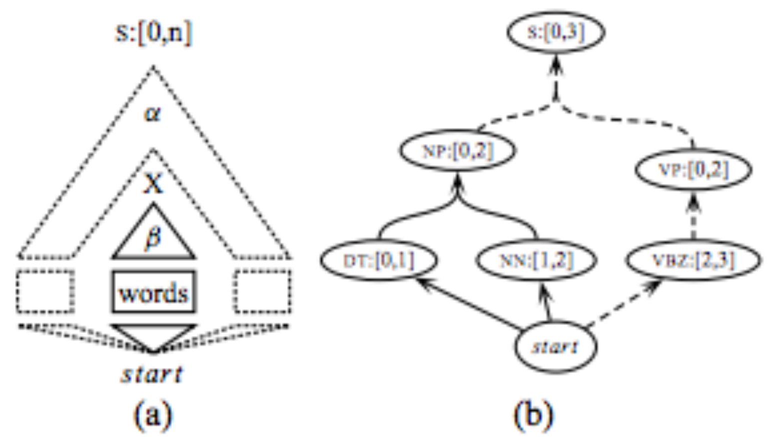
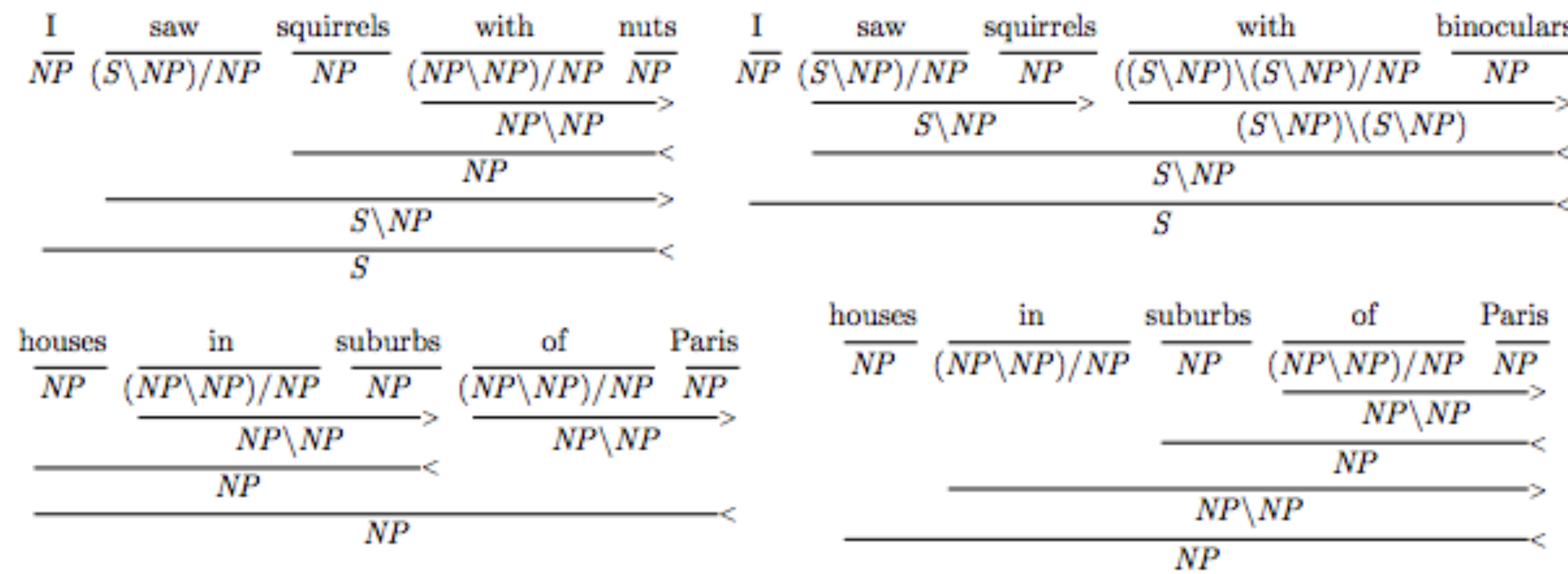


Figure 1: A* edge costs. (a) The cost of an edge X is a combination of the cost to build the edge (the Viterbi inside score β) and the cost to incorporate it into a root parse (the Viterbi outside score α). (b) In the corresponding hypergraph, we have exact values for the inside score from the explored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

Adoption with neural networks: CCG Parsing

LSTM CCG Parsing (Lewis et al. 2014)

CCG Parsing:

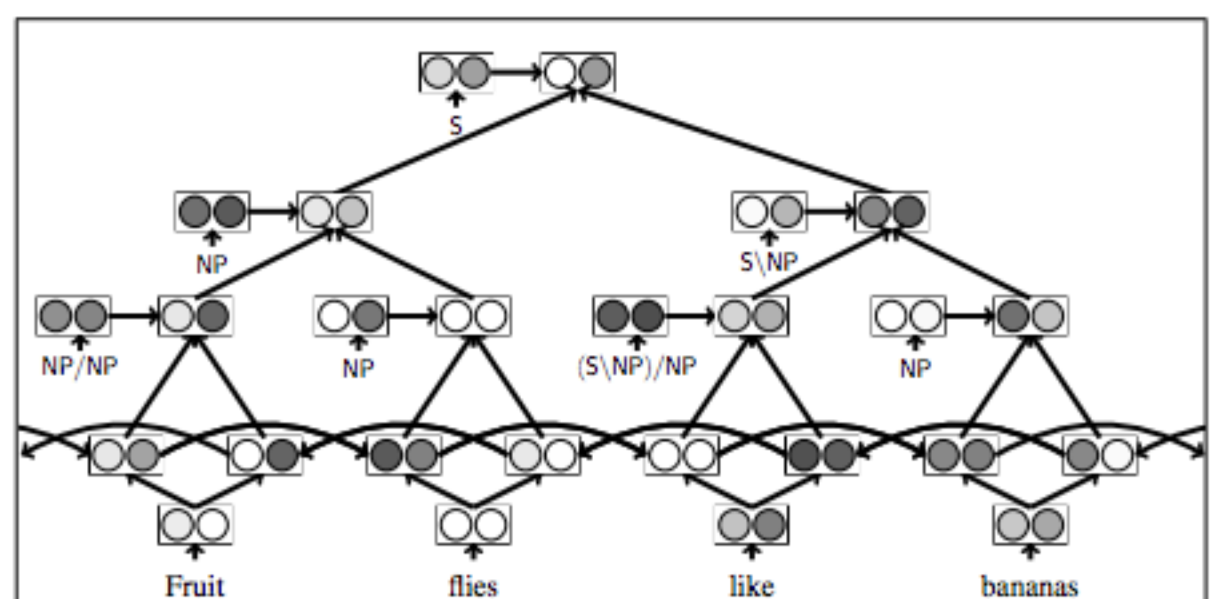


- A* for parsing
  - g(n): sum of encoded LSTM scores over current span
  - h(n): sum of maximum encoded scores for each constituent outside of current span

Is the heuristic admissible?

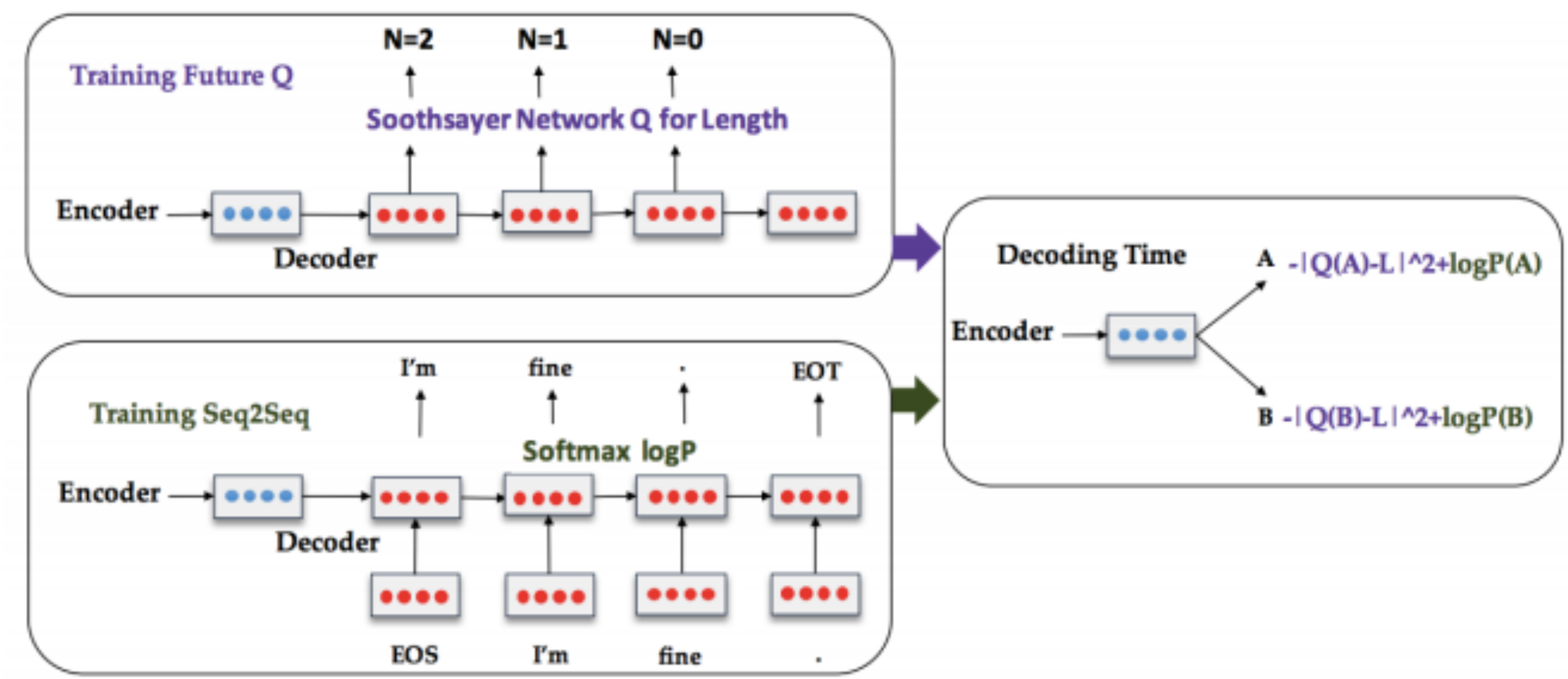
Global Neural CCG Parsing with Optimality Guarantees (Lee et al. 2016)

- 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

Learning to Decode for Future Success (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

Other search algorithms

Particle Filters

A Bayesian Model for Generative Transition-based Dependency Parsing (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

Recurrent Neural Network Grammars
(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

Monte-Carlo Tree Search

Human-like Natural Language Generation Using Monte Carlo Tree Search