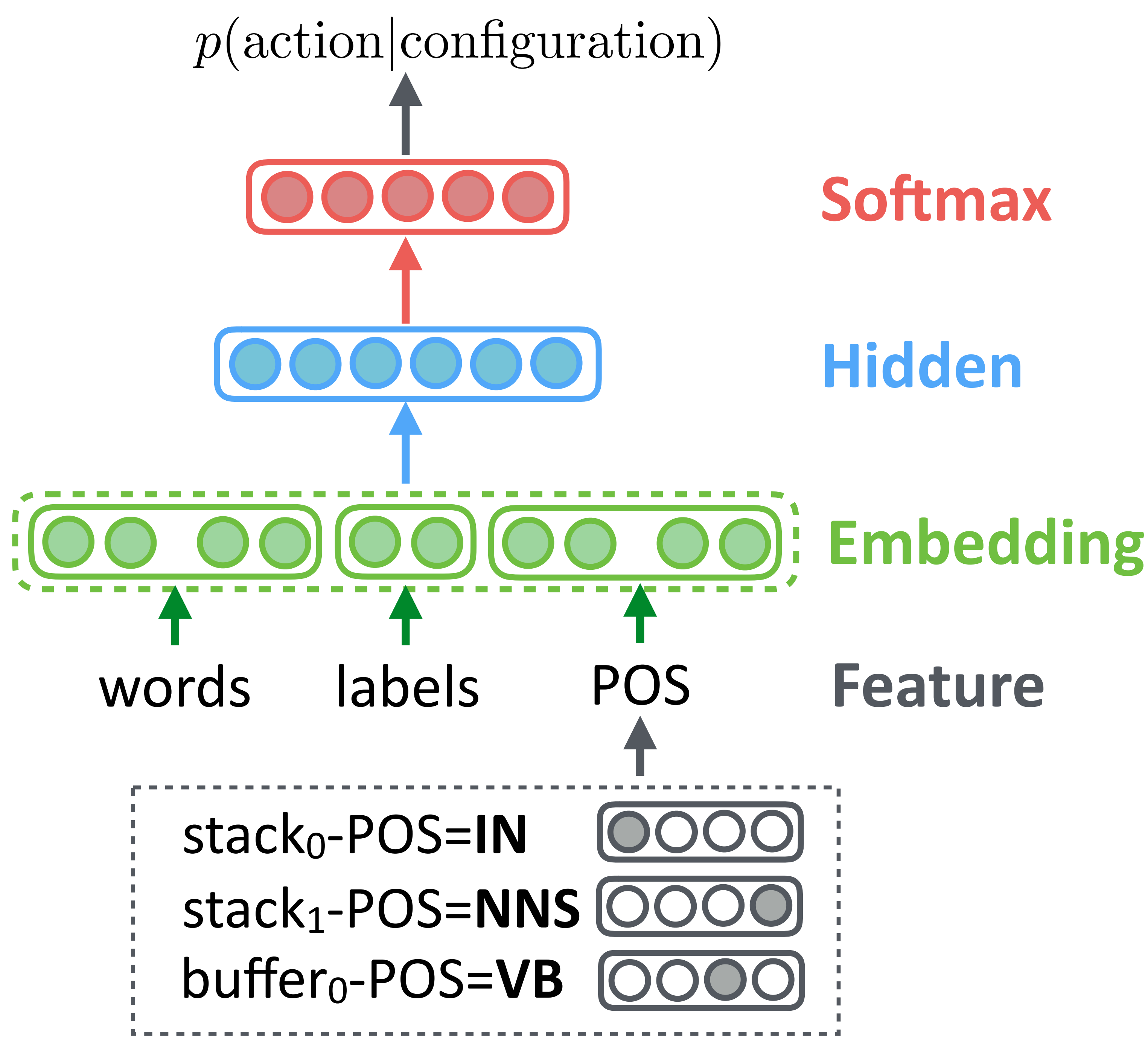
Stack-propagation: Improved Representation Learning for Syntax

Yuan Zhang, David Weiss

MIT, Google

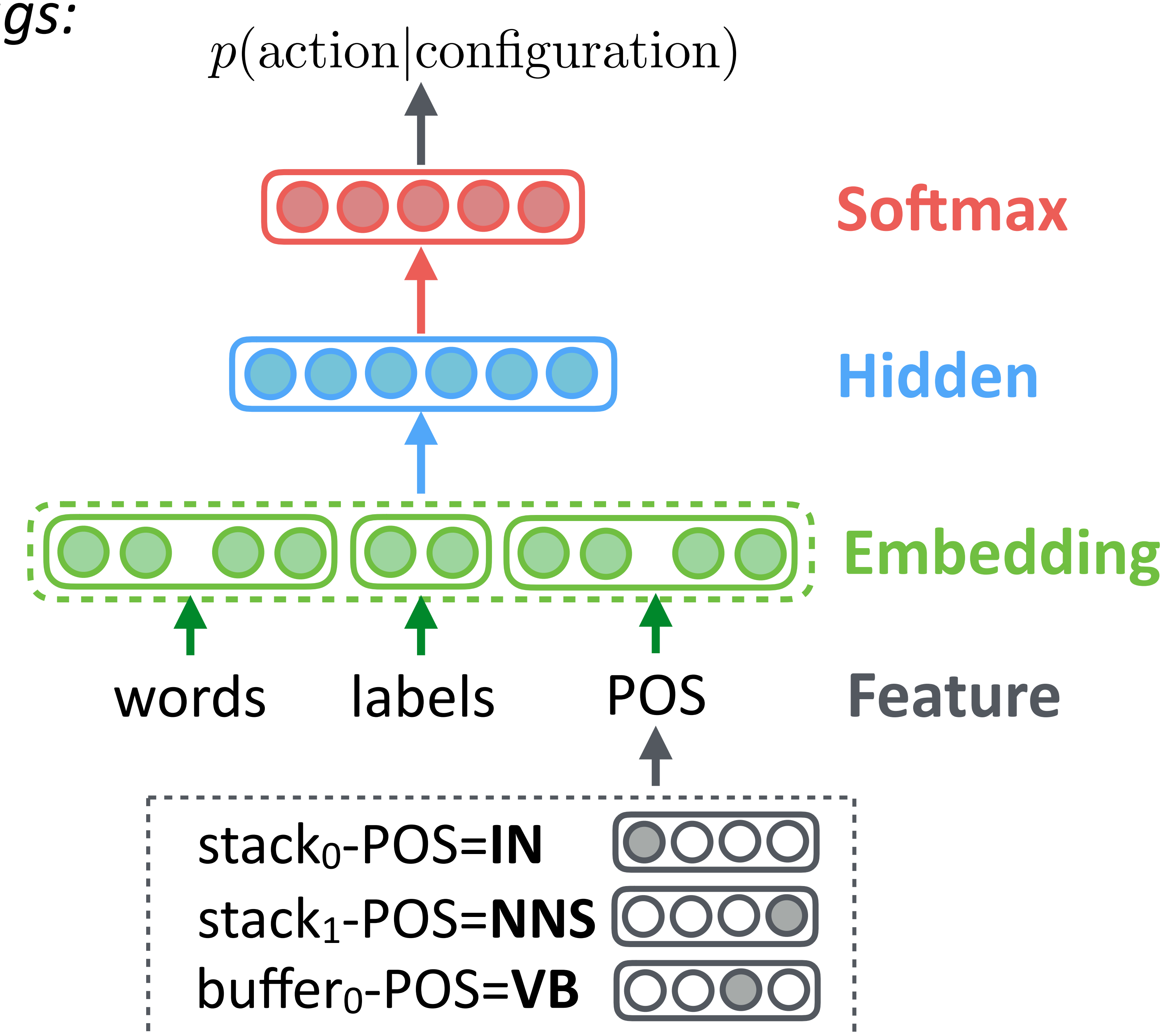
Transition-based Neural Network Parser



Transition-based Neural Network Parser

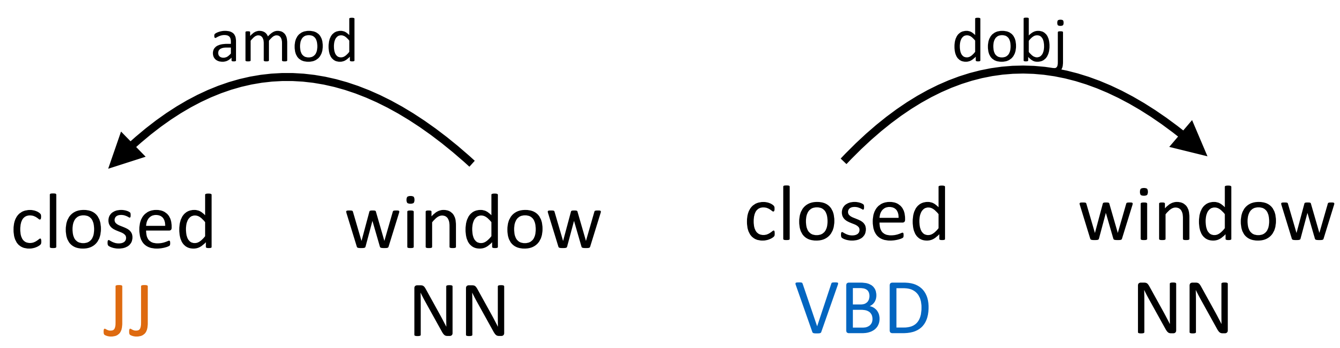
Prior methods on using POS tags:
Traditional stacking (Pipeline)
(Alberti'15, Weiss'15, Chen'14)

- Train an independent POS tagger
- Use predicted POS tags as sparse features



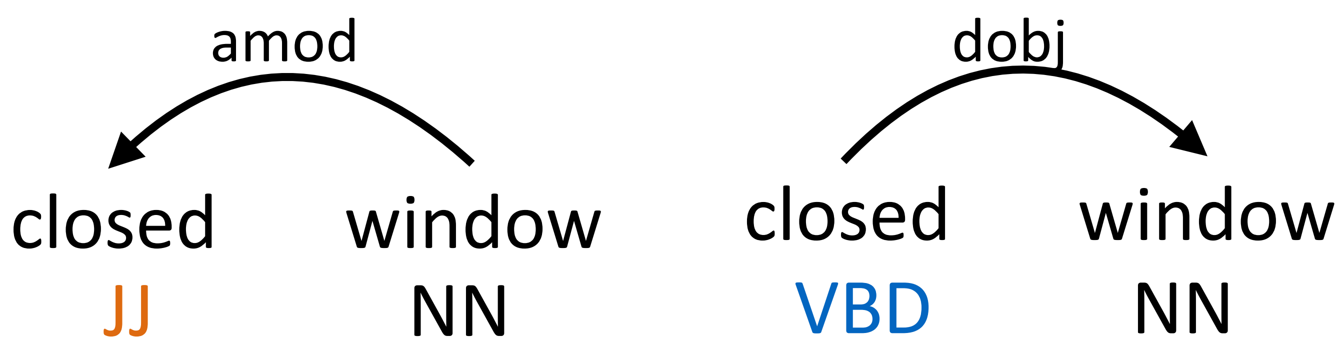
Issues of Traditional Stacking

- Error propagation from predicted POS tags
  - ✦ Predicted (■) vs. Gold (■)

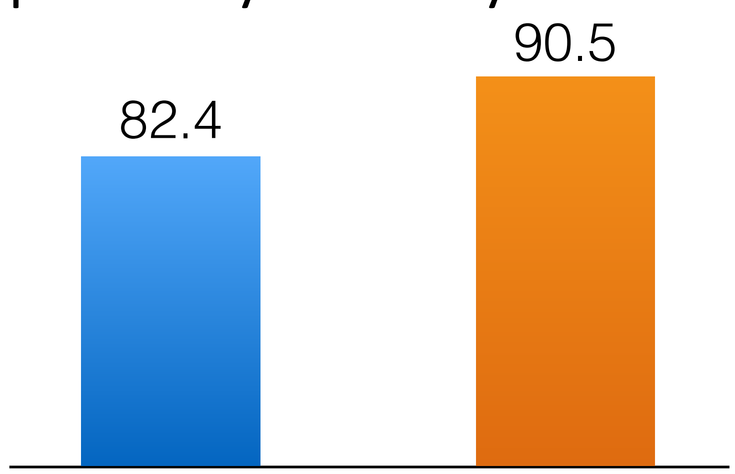


Issues of Traditional Stacking

- Error propagation from predicted POS tags
  - ✦ Predicted (■) vs. Gold (■)

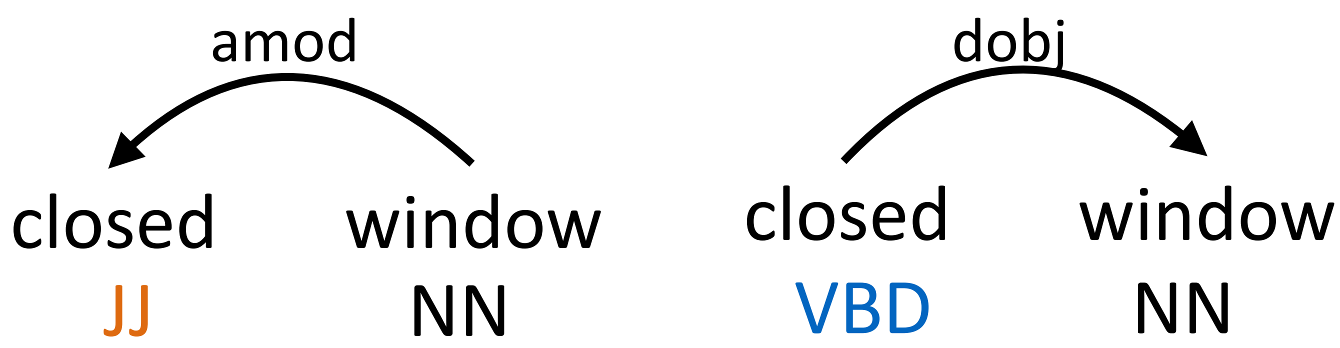


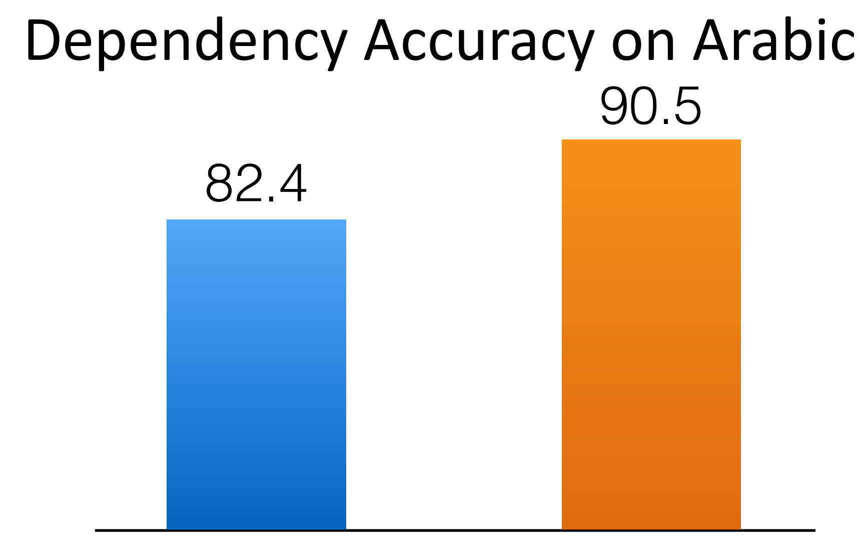
Dependency Accuracy on Arabic



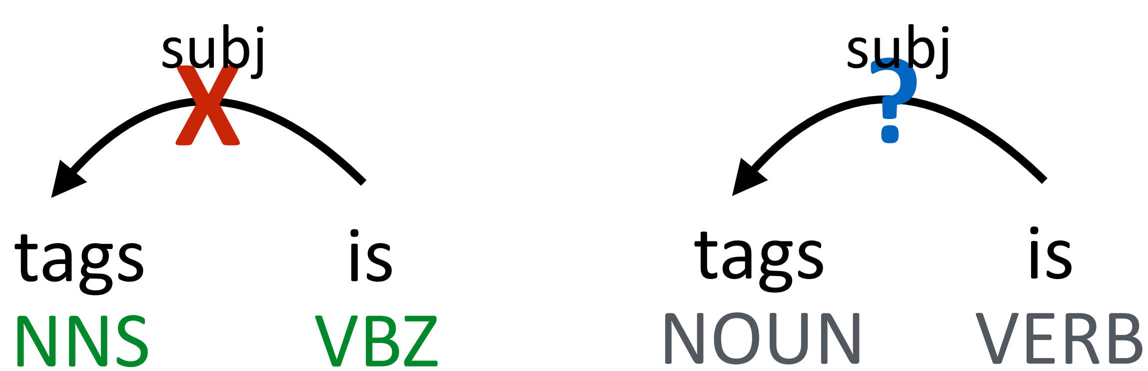
Issues of Traditional Stacking

- Error propagation from predicted POS tags
  - ✦ Predicted (■) vs. Gold (■)



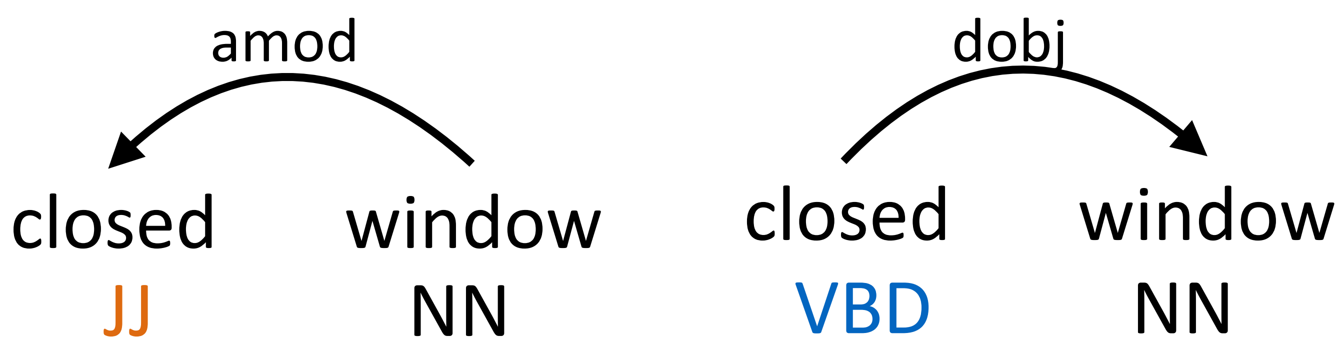


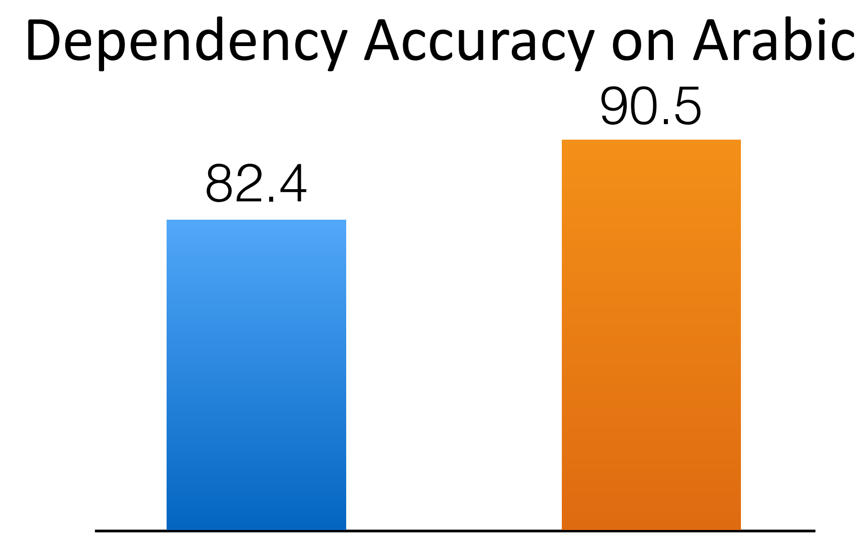
- Limitation of using discrete POS representation
  - ✦ Coarse (■ 12 tags) vs. Fine tagset (■ 45 tags)



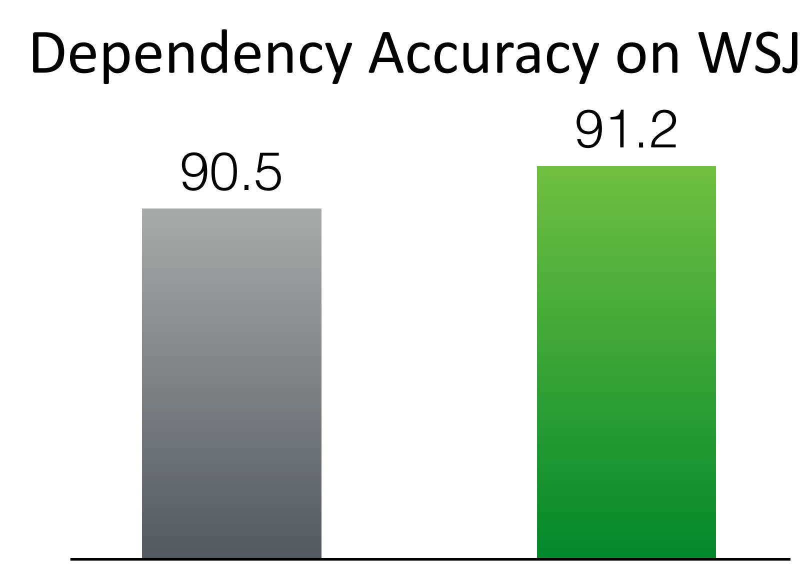
Issues of Traditional Stacking

- Error propagation from predicted POS tags
  - ✦ Predicted ( ) vs. Gold ( )



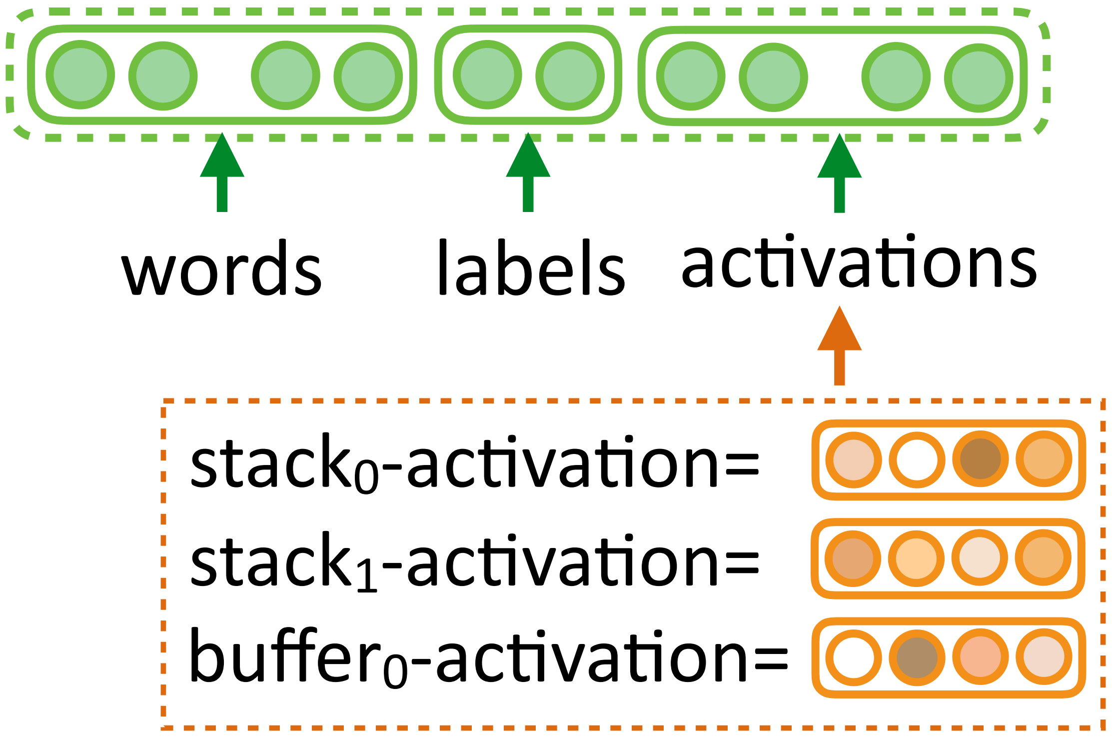


- Limitation of using discrete POS representation
  - ✦ Coarse (12 tags) vs. Fine tagset (45 tags)



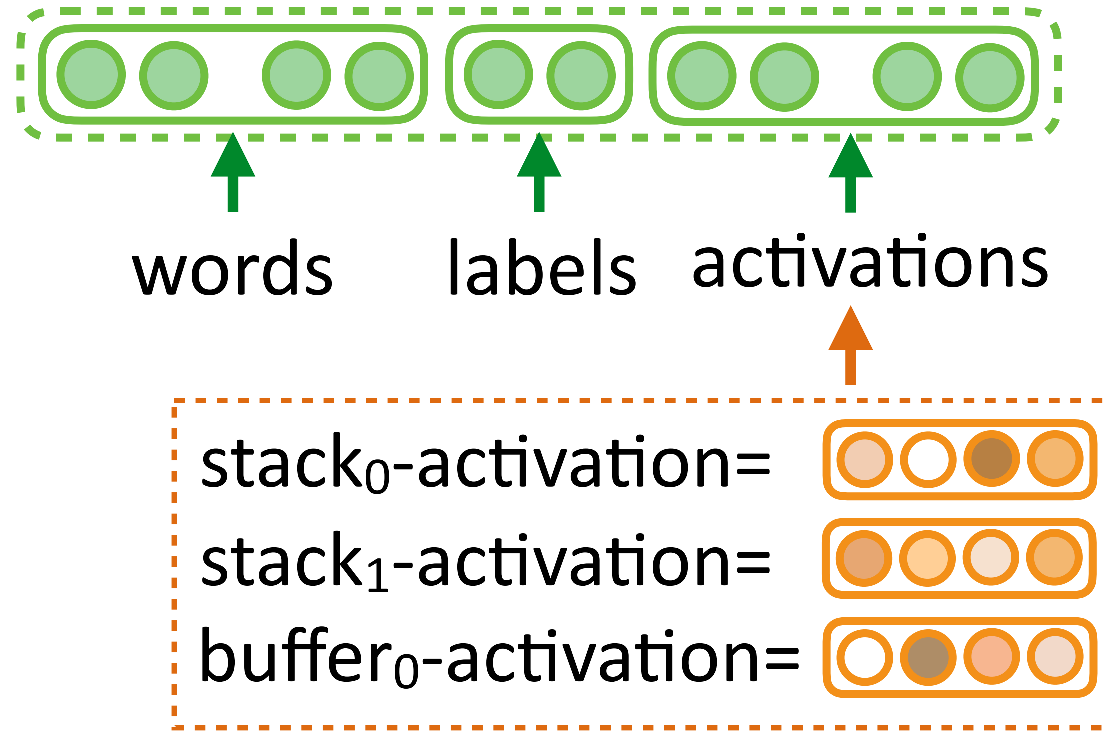
Core Idea of Our Stacking Methods

- Stack-propagation: Replace discrete POS features with hidden layer activations of a tagger

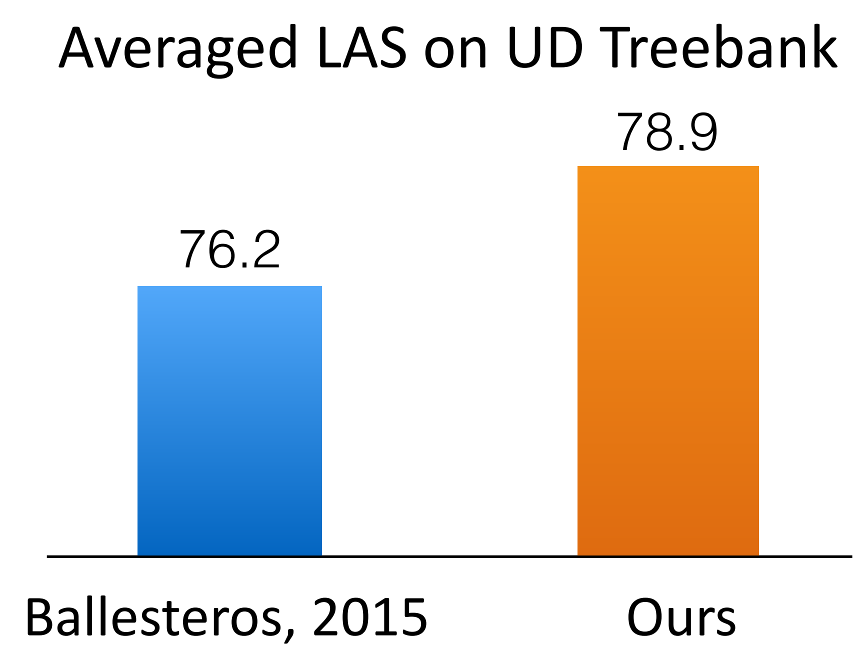


Core Idea of Our Stacking Methods

- Stack-propagation: Replace discrete POS features with hidden layer activations of a tagger

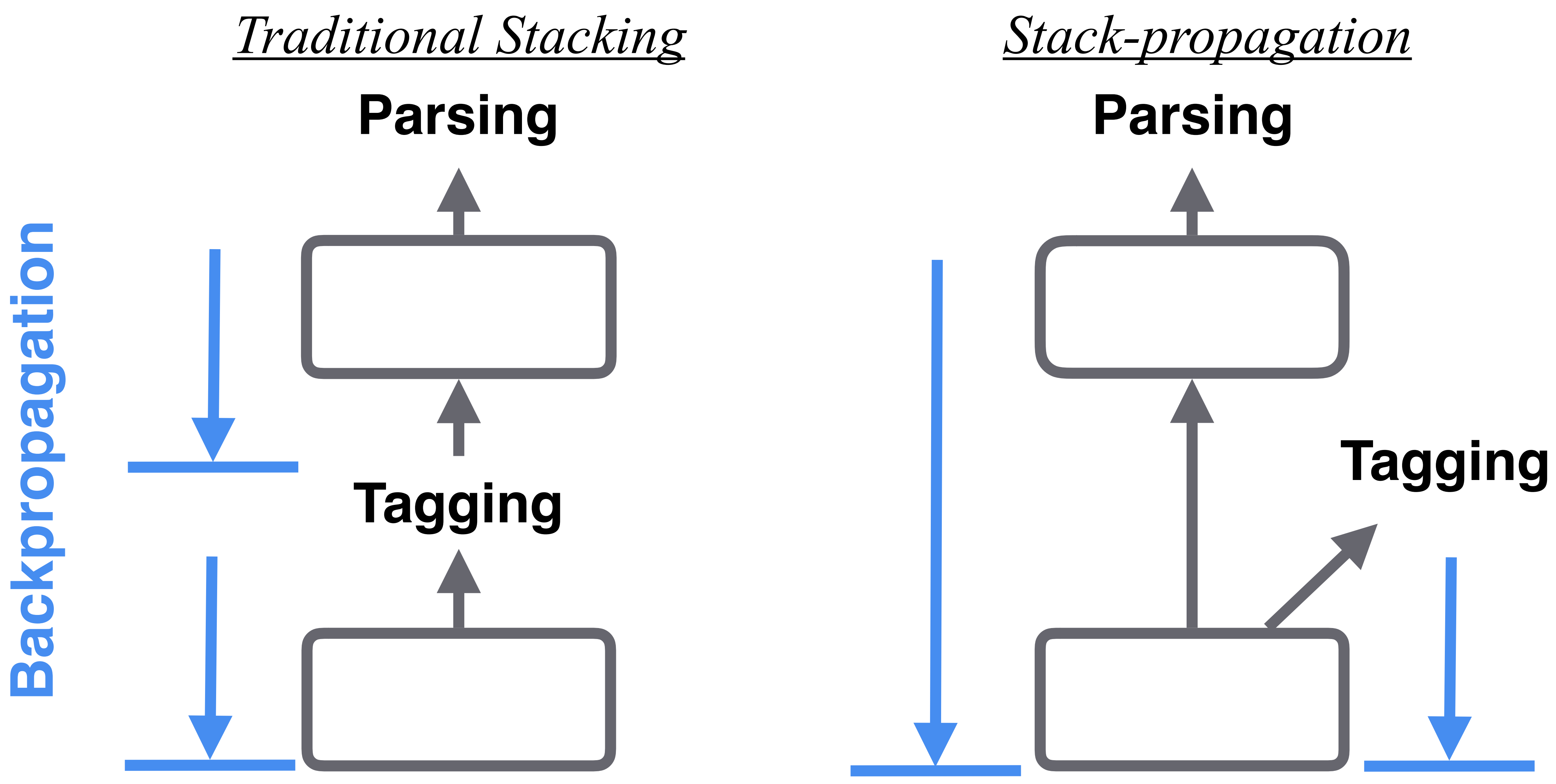


- Advantages:
  - ✦ Joint training for parsing and tagging
  - ✦ Robust to POS errors
  - ✦ Better feature representations than discrete POS features



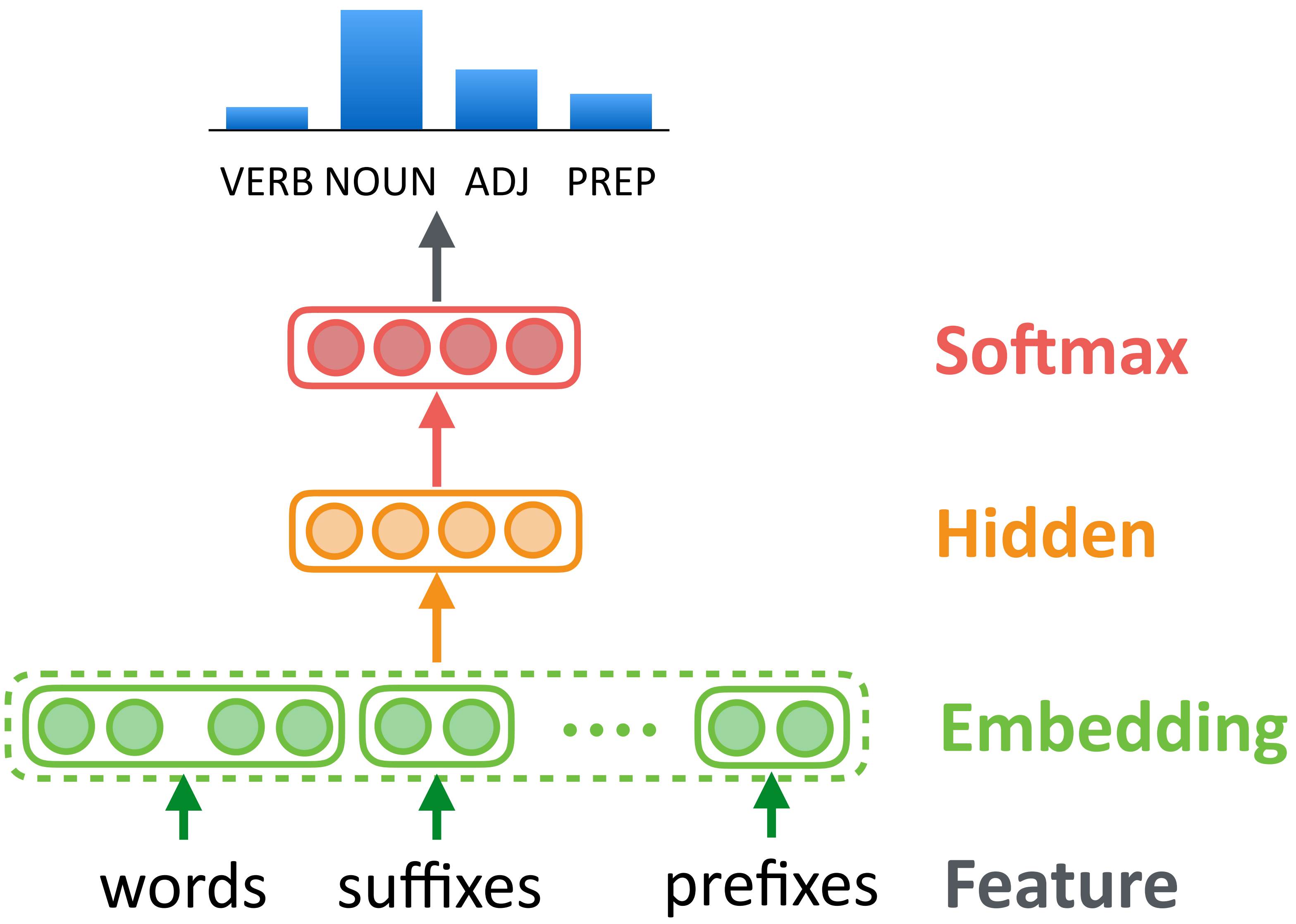
Overview of Network Architecture

- Two networks: a parser and a tagger
- Shared component: tagger except for softmax layers



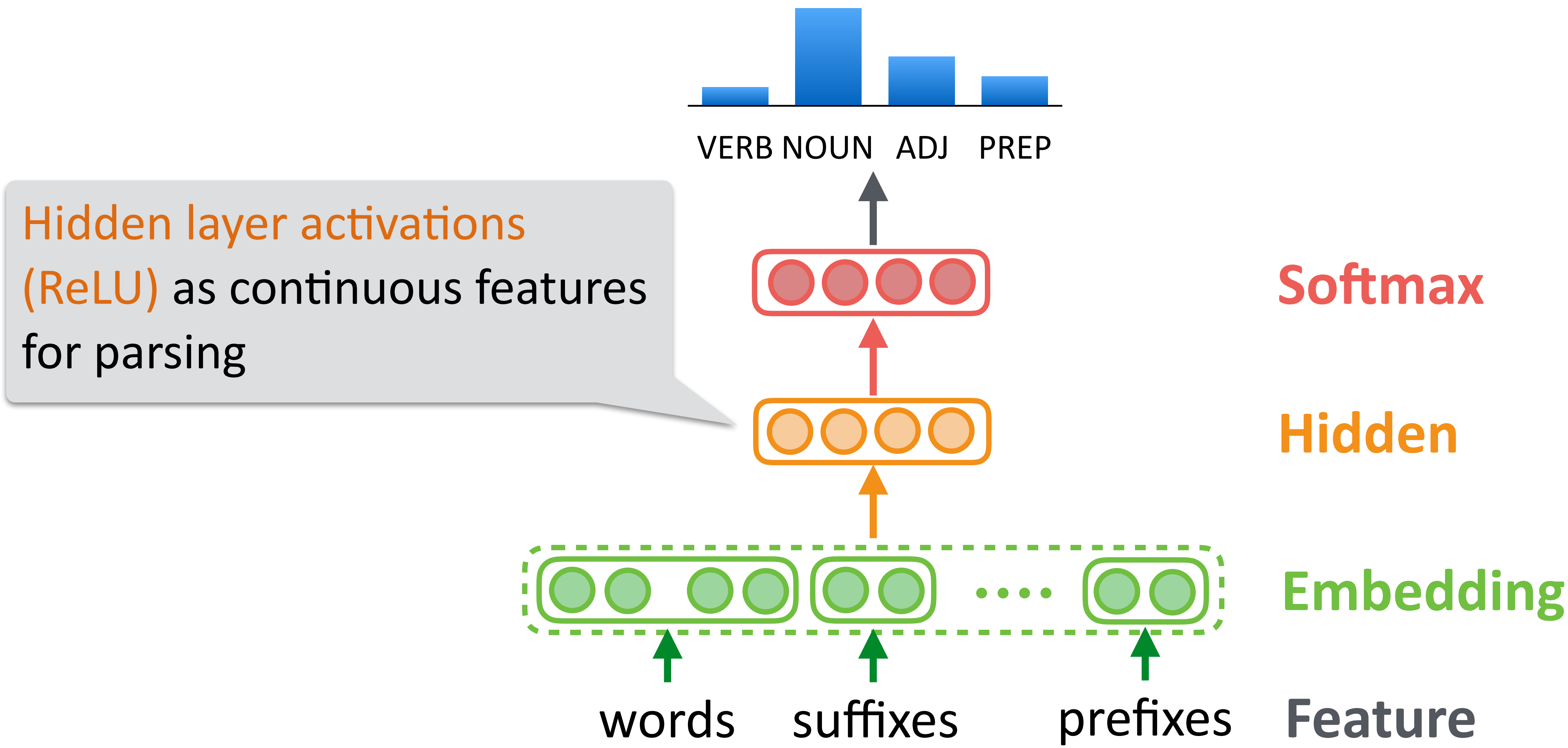
Tagger Network

- Standard window-based NN classifier for tagging



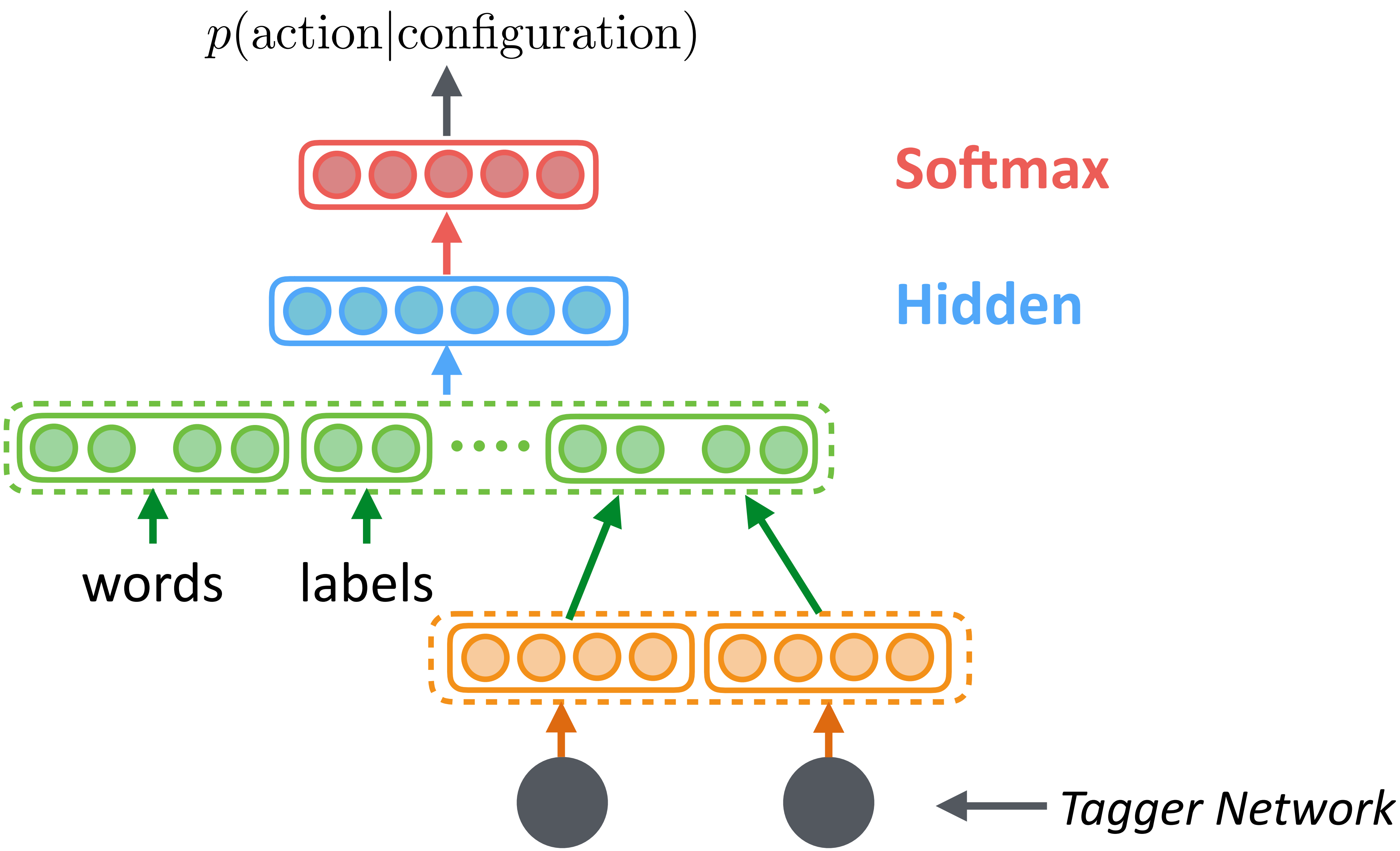
Tagger Network

- Standard window-based NN classifier for tagging



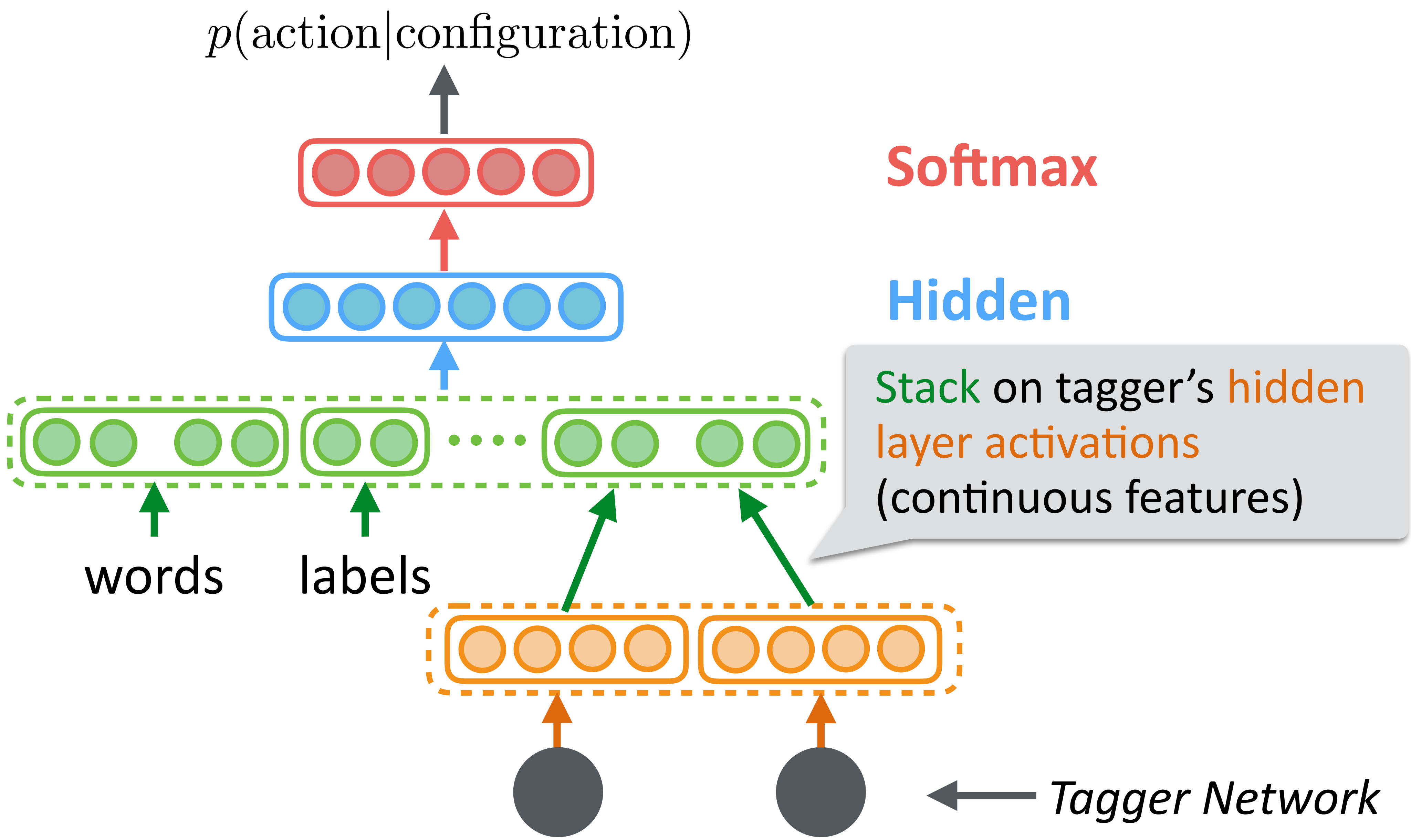
Parser Network

- Transition-based NN parser (Weiss'15, Chen'15) *stacked* on hidden layer activations of the tagger



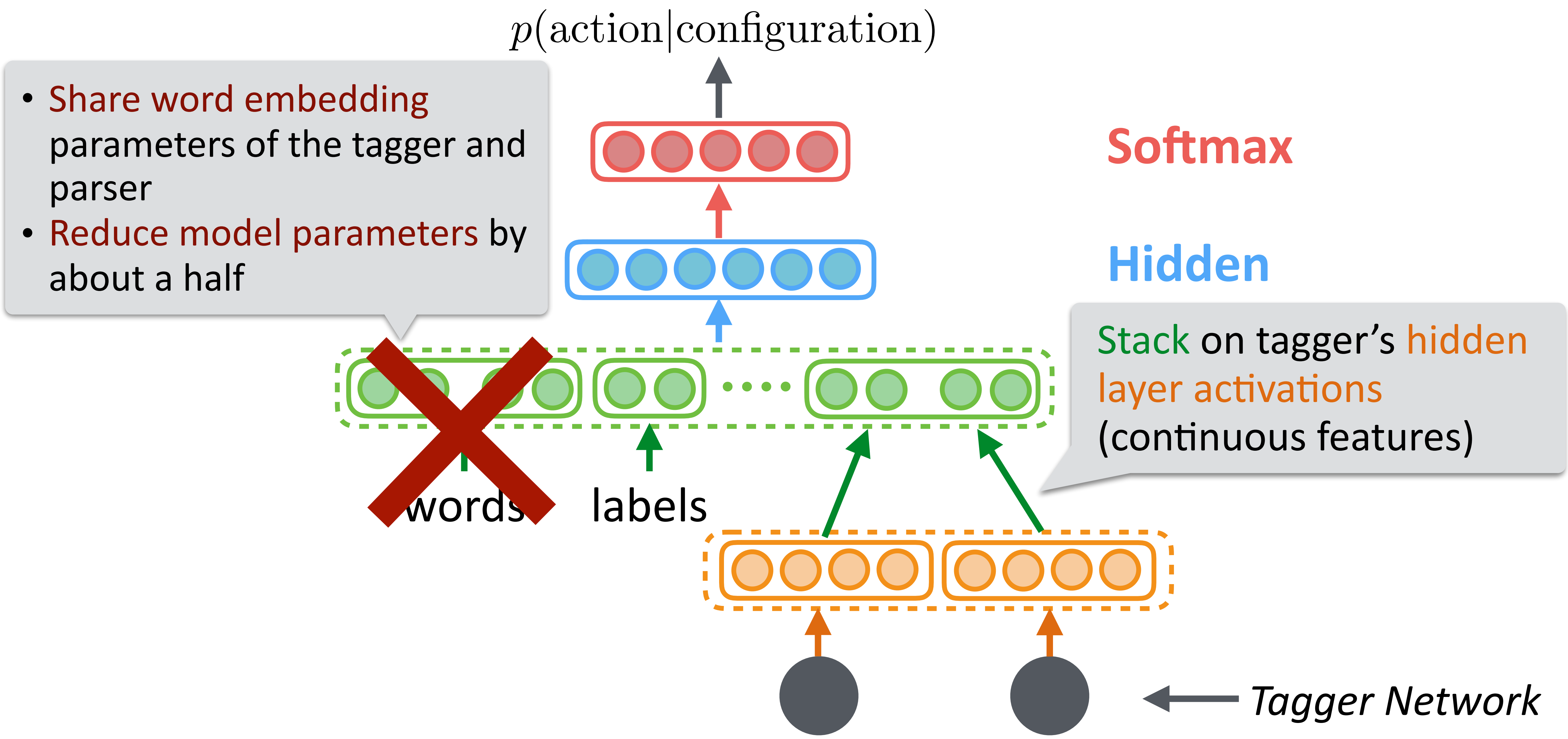
Parser Network

- Transition-based NN parser (Weiss'15, Chen'15) *stacked* on hidden layer activations of the tagger



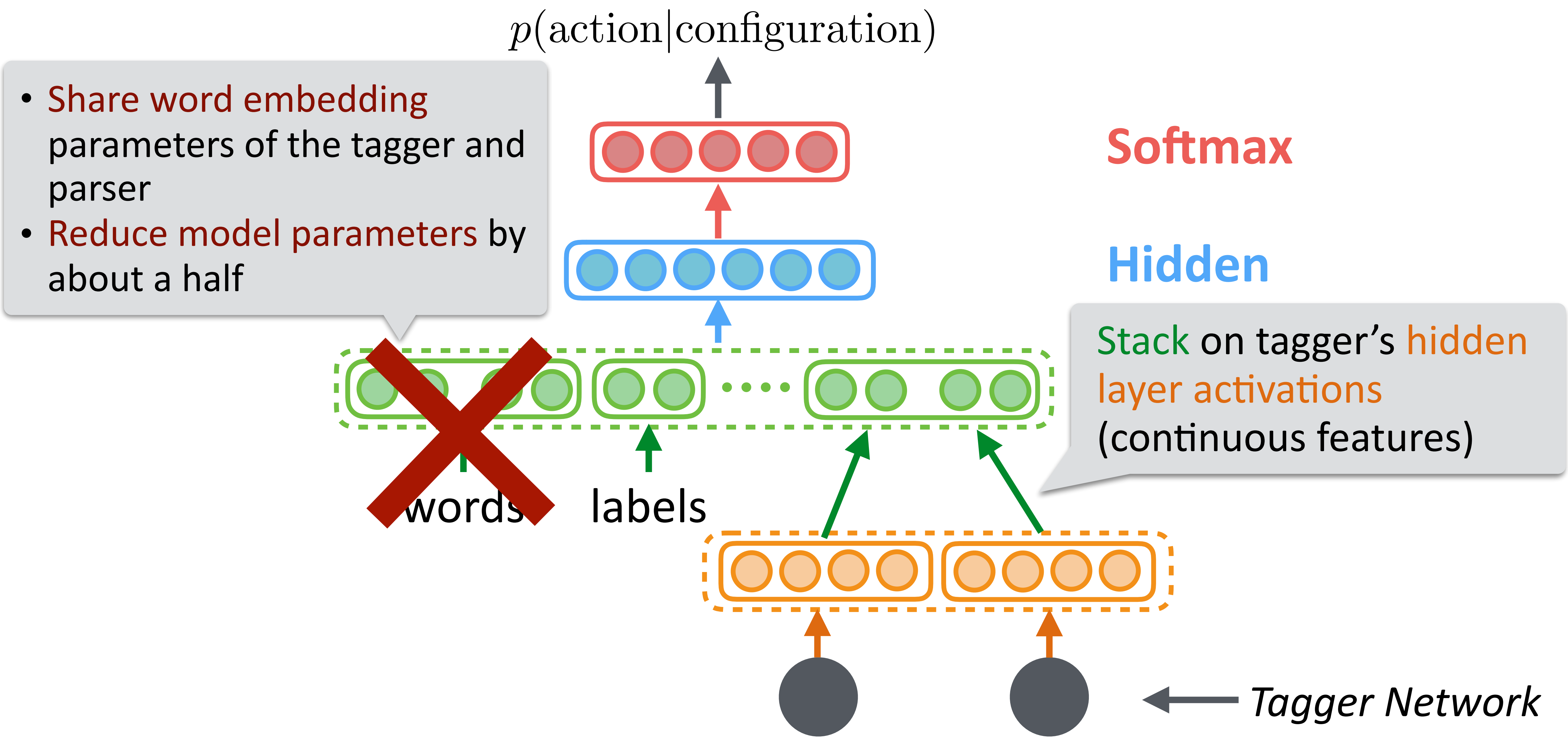
Parser Network

- Transition-based NN parser (Weiss'15, Chen'15) *stacked* on hidden layer activations of the tagger



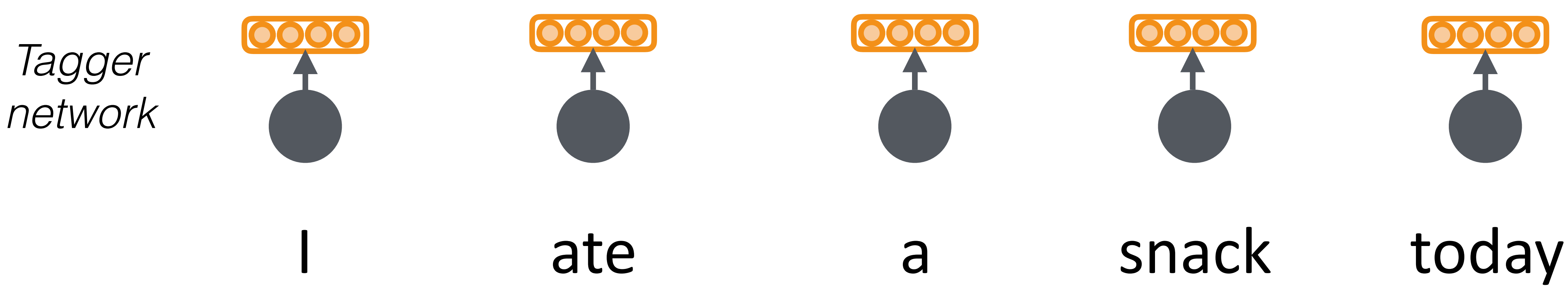
Parser Network

- Transition-based NN parser (Weiss'15, Chen'15) *stacked* on hidden layer activations of the tagger

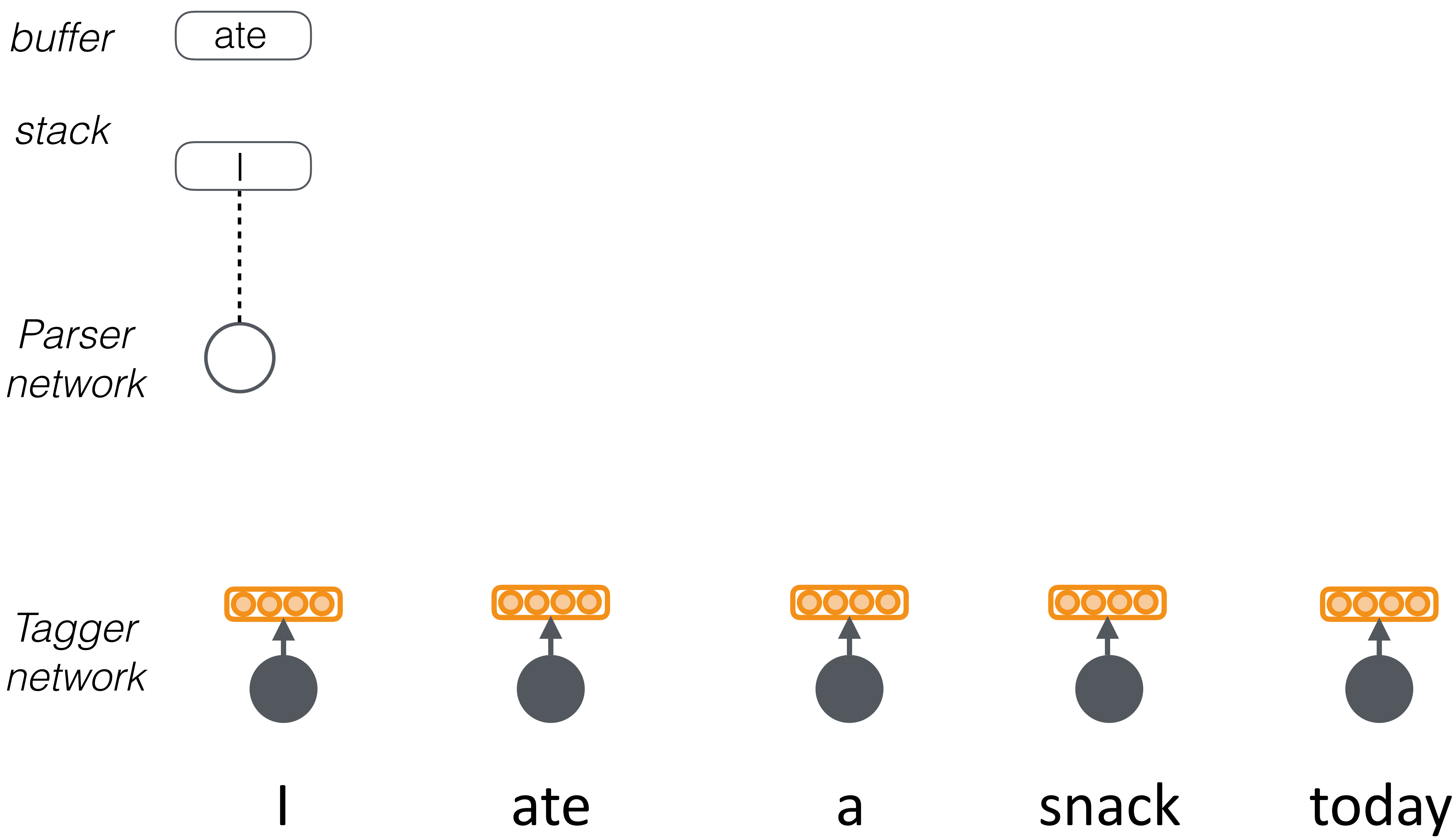


- More complex architecture after unrolling parser transitions

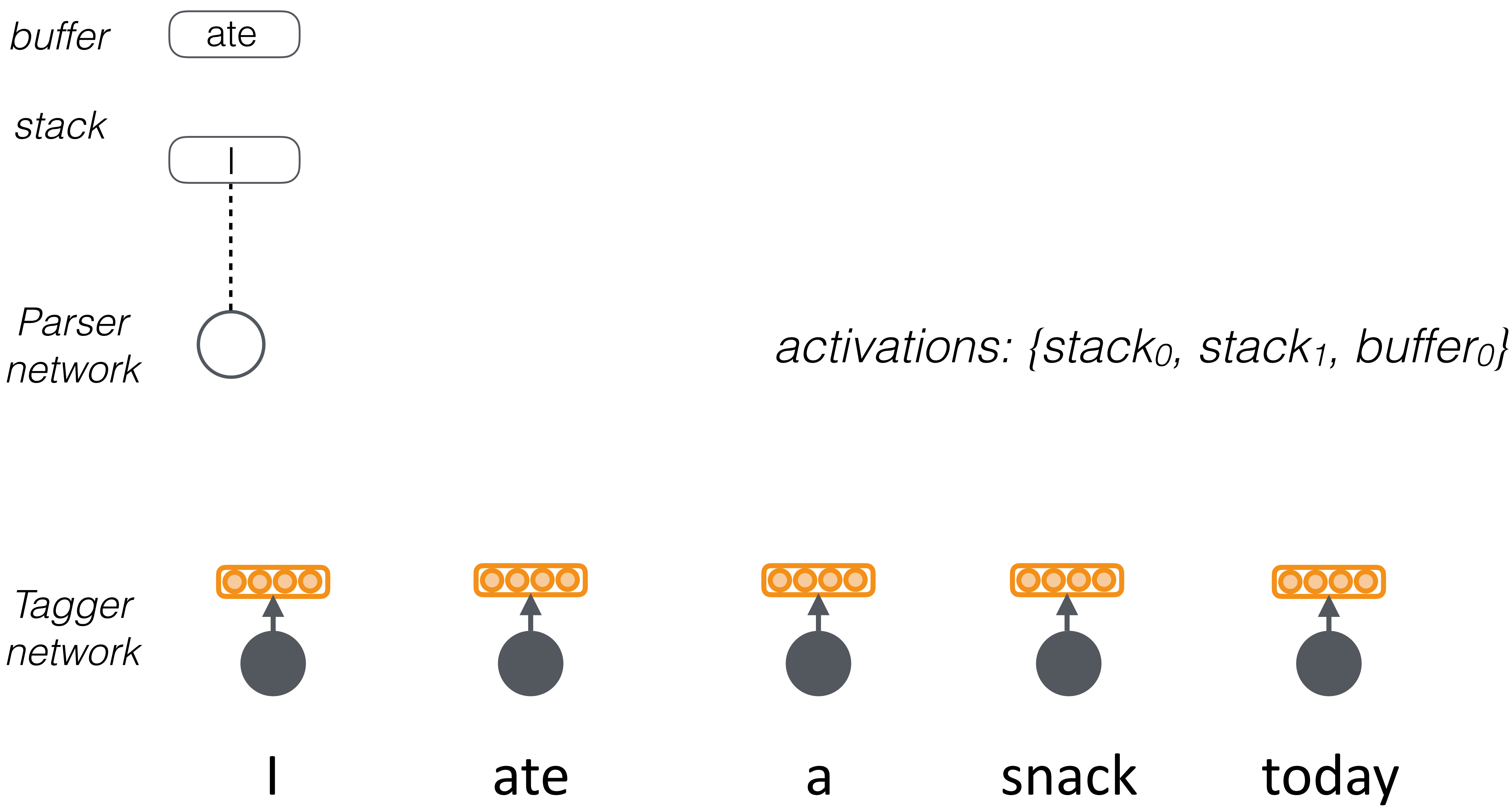
Unrolled NN Architecture (Inference)



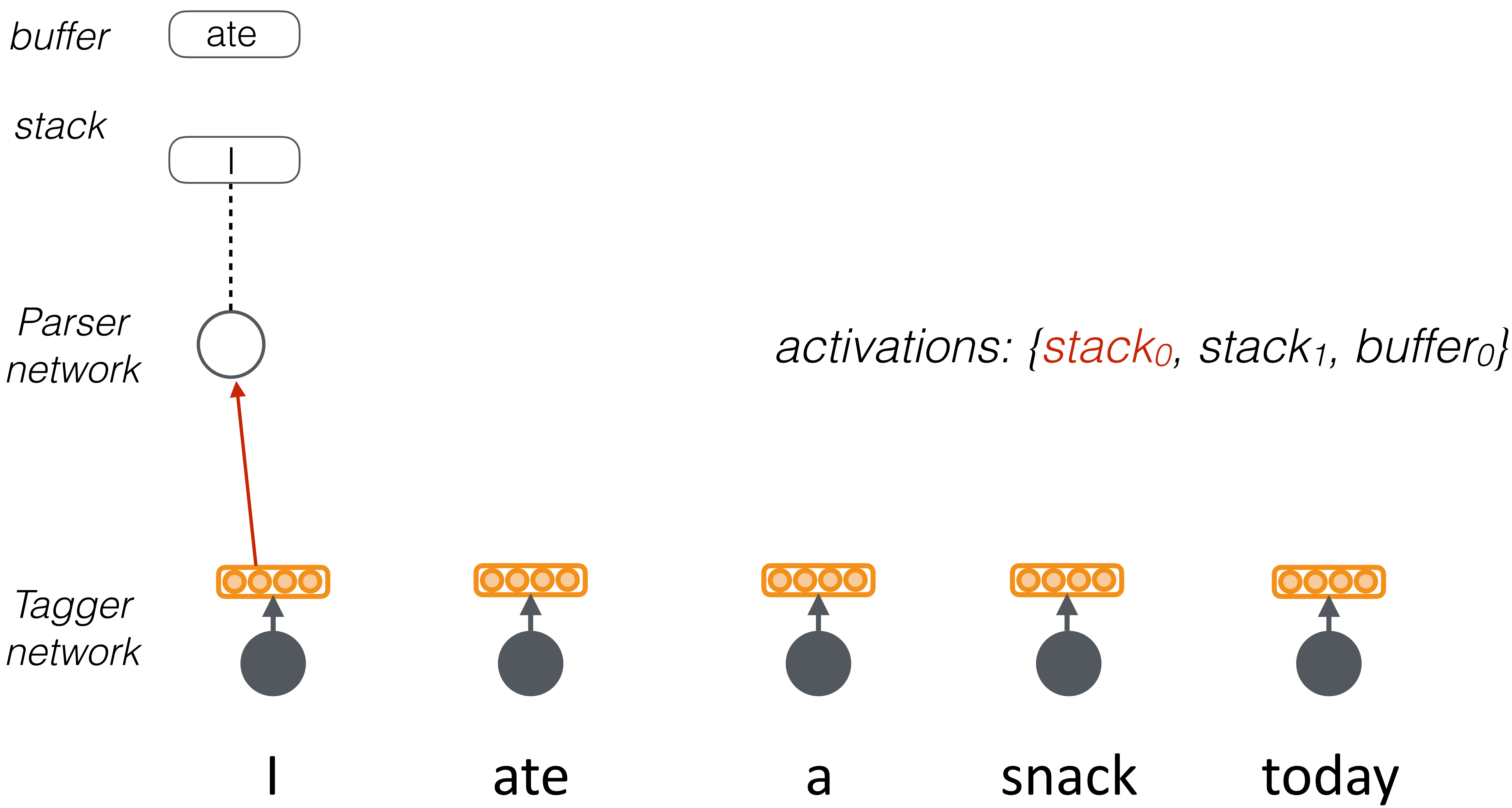
Unrolled NN Architecture (Inference)



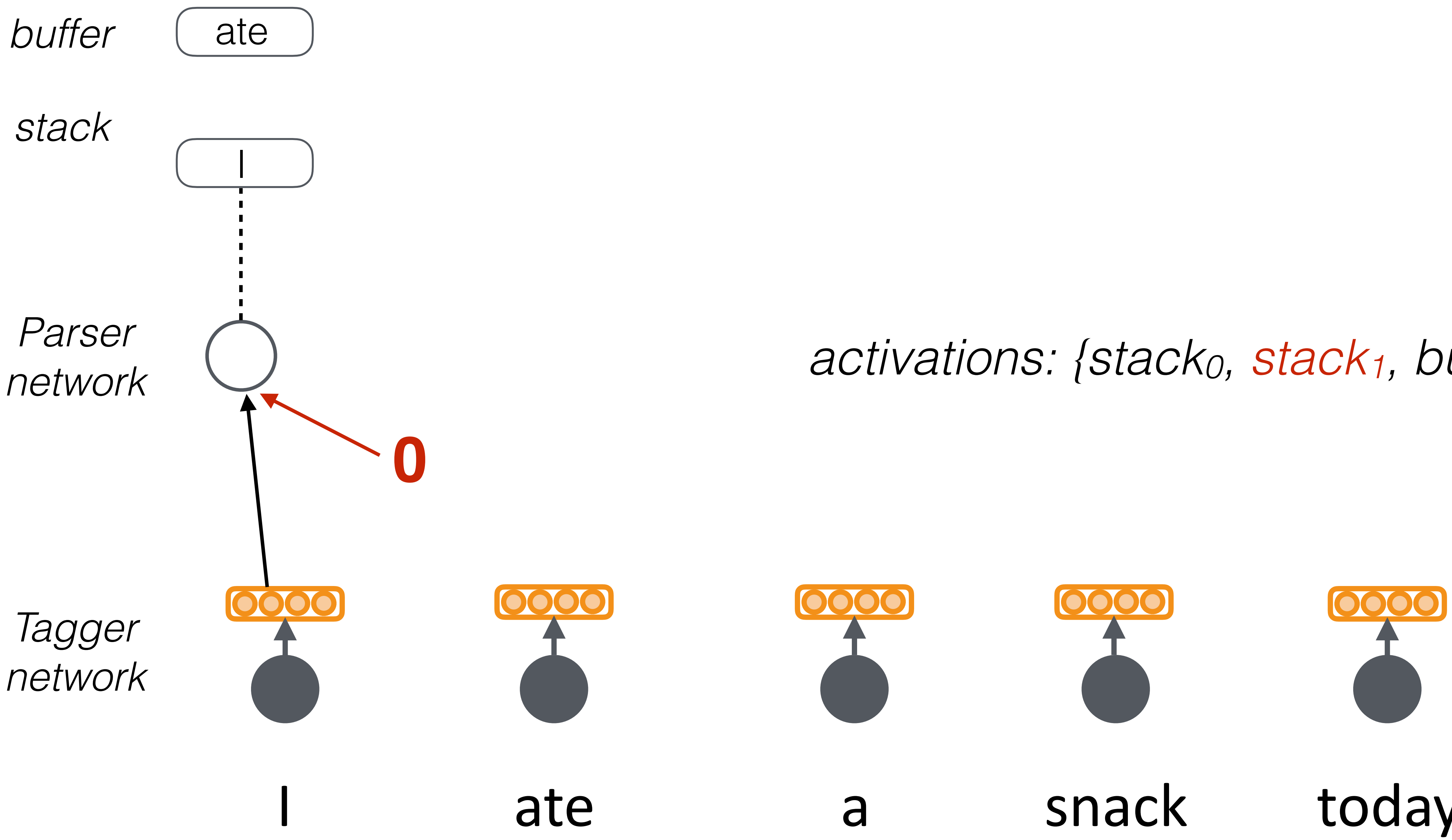
Unrolled NN Architecture (Inference)



Unrolled NN Architecture (Inference)

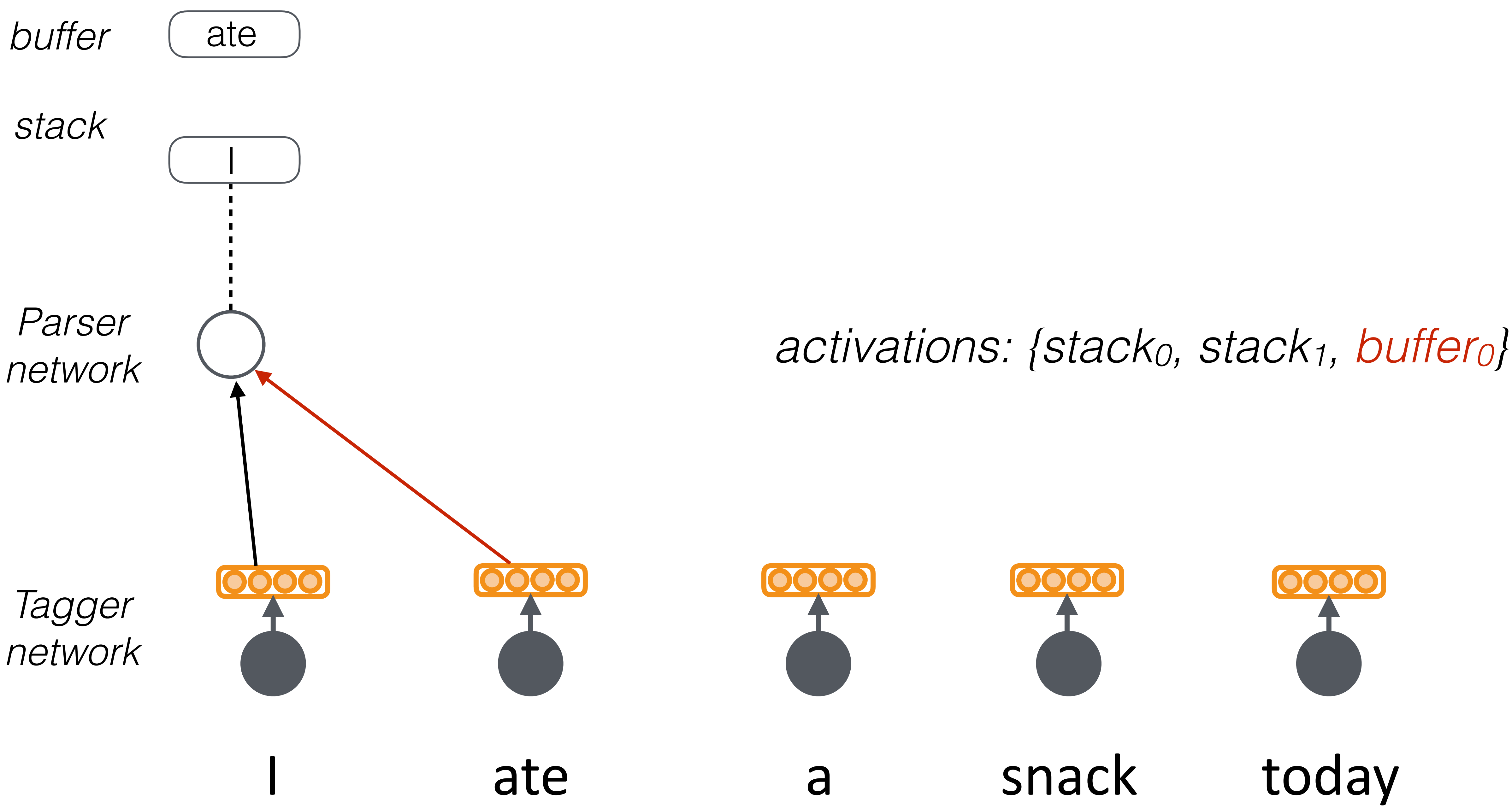


Unrolled NN Architecture (Inference)

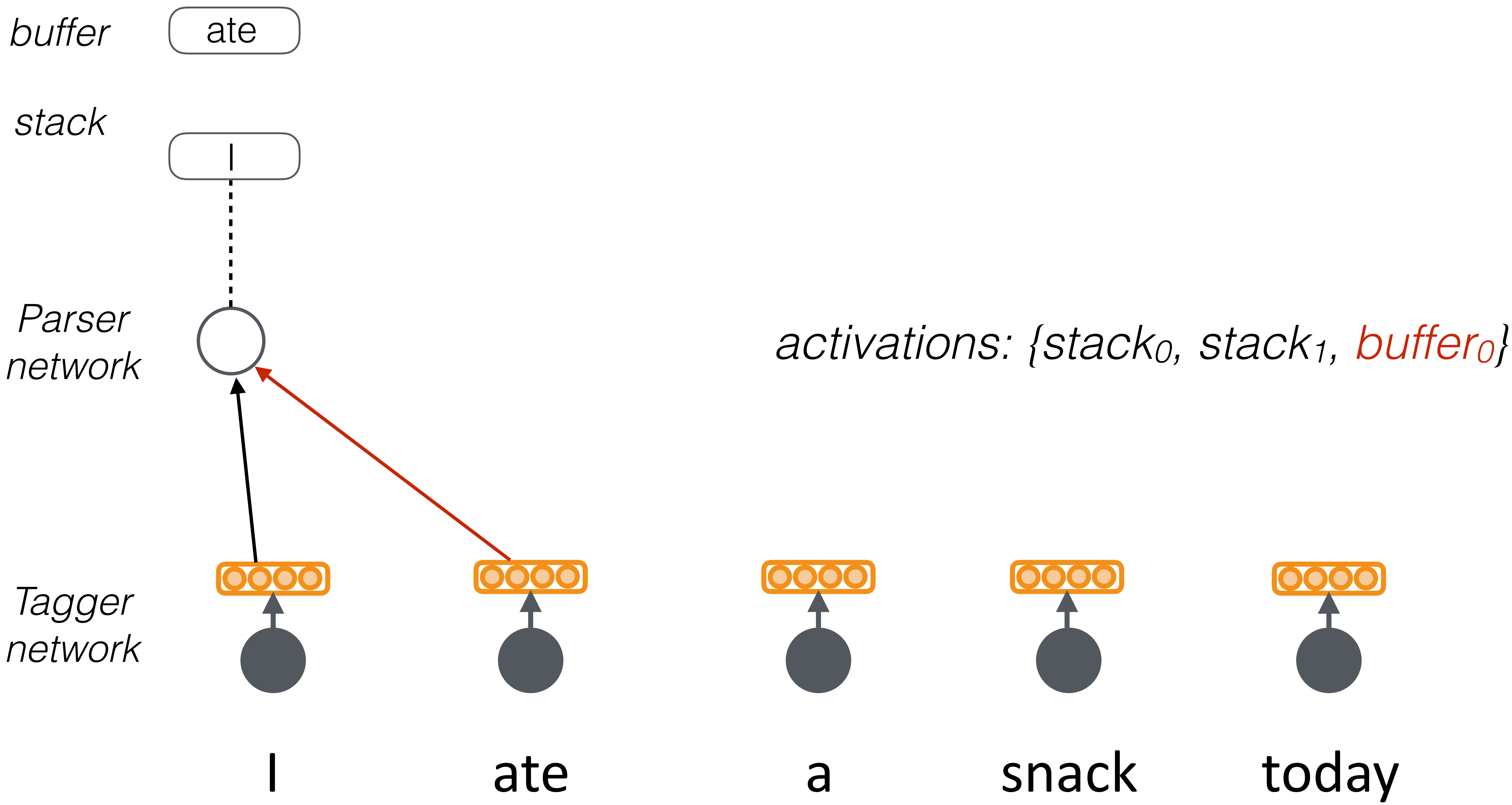


activations: {stack$_0$, stack$_1$, buffer$_0$}

Unrolled NN Architecture (Inference)

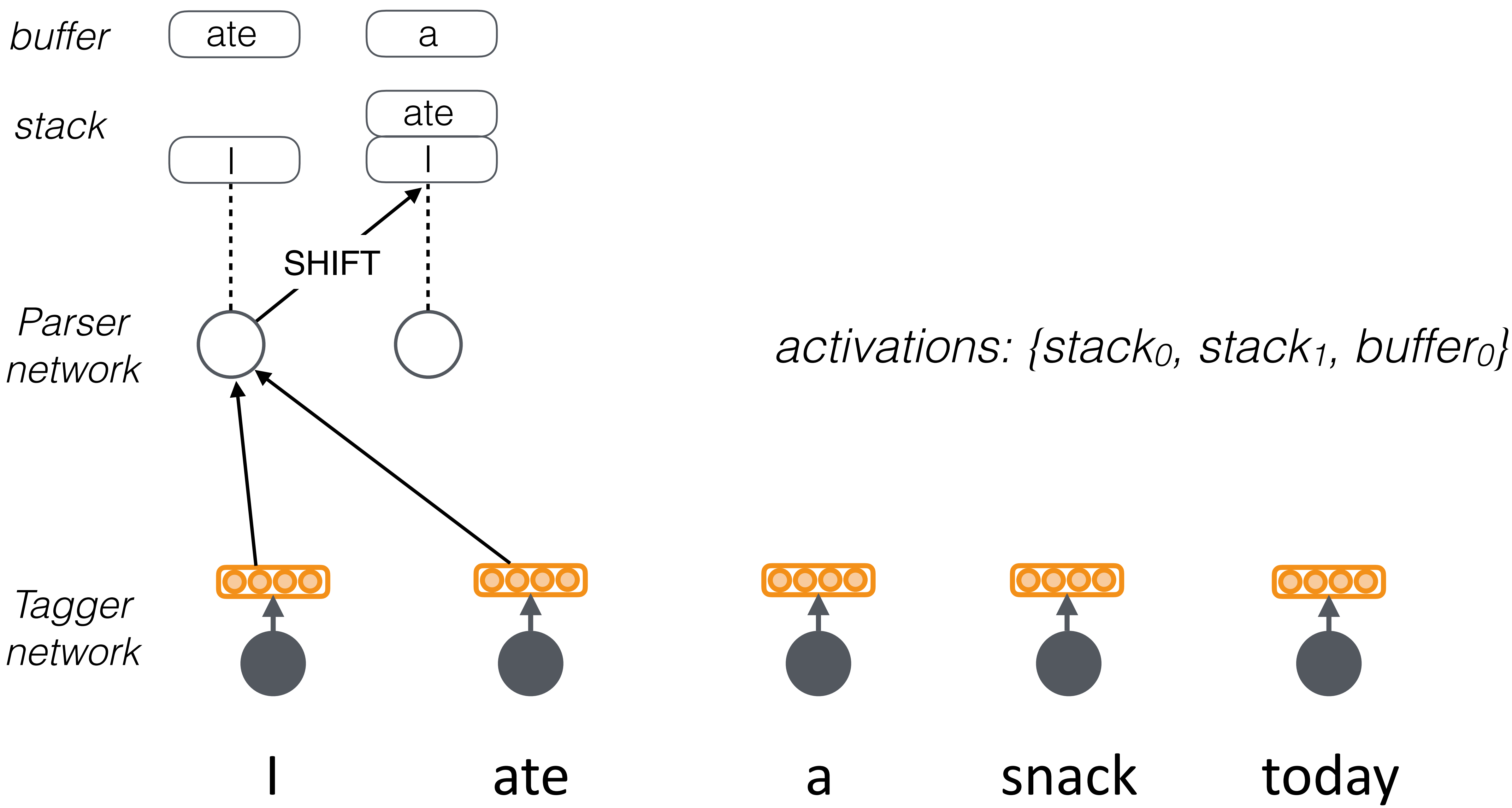


Unrolled NN Architecture (Inference)



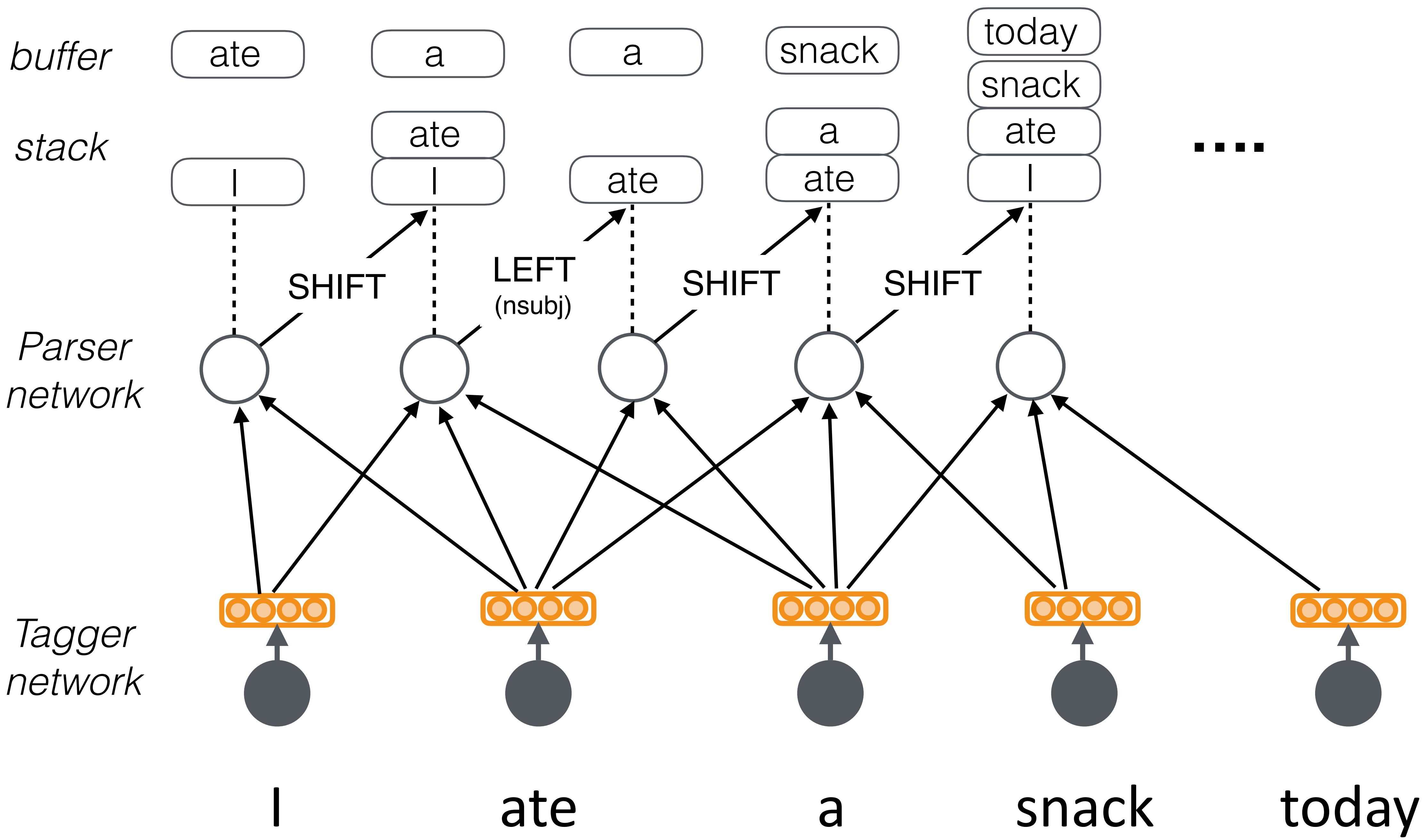
- In practice, the parser looks up activations from 20 locations
- No need to predict POS tags

Unrolled NN Architecture (Inference)



- In practice, the parser looks up activations from 20 locations
- No need to predict POS tags

Unrolled NN Architecture (Inference)



- In practice, the parser looks up activations from 20 locations
- No need to predict POS tags

Multi-task Style Learning

- Objective: maximize log-likelihood for parsing and tagging

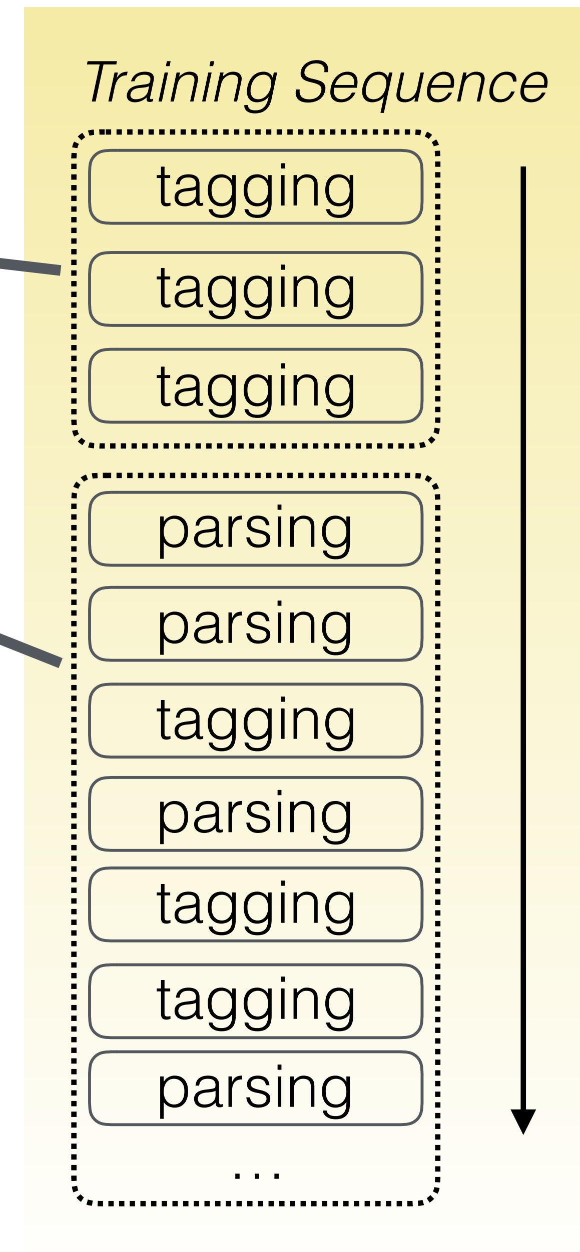
$$\max_{\Theta} \boxed{\lambda \sum_{x,y \in \mathcal{T}} \log(P_\Theta(y|x))} + \boxed{\sum_{c,a \in \mathcal{P}} \log(P_\Theta(a|c))}$$

Multi-task Style Learning

- Objective: maximize log-likelihood for parsing and tagging

$$\max_{\Theta} \lambda \sum_{x,y\in\mathcal{T}} \log(P_{\Theta}(y|x)) + \sum_{c,a\in\mathcal{P}} \log(P_{\Theta}(a|c))$$

- Multi-task style learning
  - ✦ Pre-train tagger for one epoch
  - ✦ Randomly alternate between updating with parsing/tagging examples

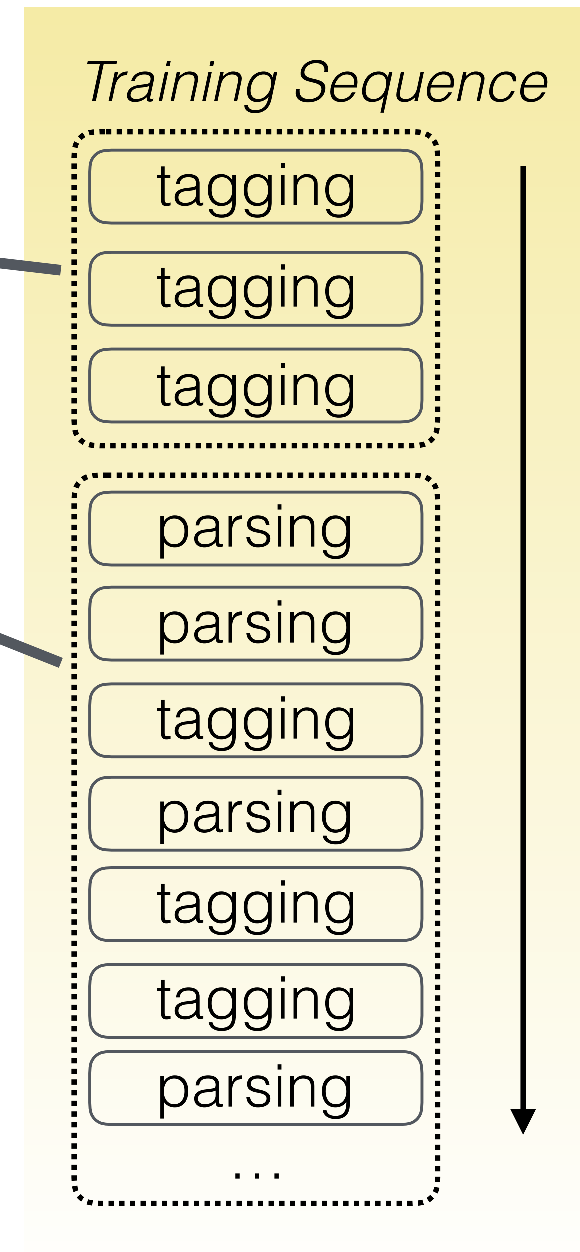


Multi-task Style Learning

- Objective: maximize log-likelihood for parsing and tagging

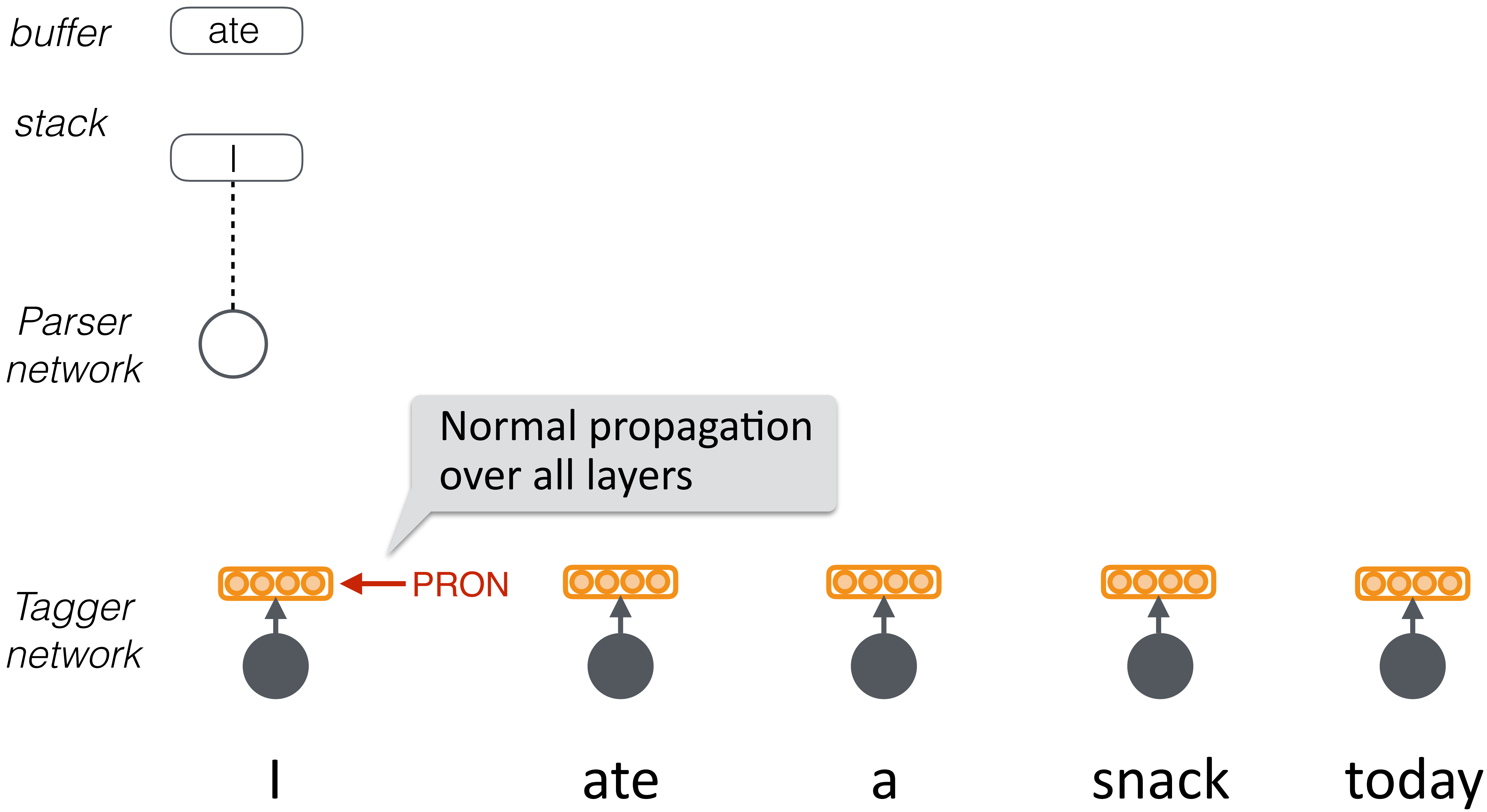
$$\max_{\Theta} \lambda \sum_{x,y \in \mathcal{T}} \log(P_\Theta(y|x)) + \sum_{c,a \in \mathcal{P}} \log(P_\Theta(a|c))$$

- Multi-task style learning
  - ✦ Pre-train tagger for one epoch
  - ✦ Randomly alternate between updating with parsing/tagging examples

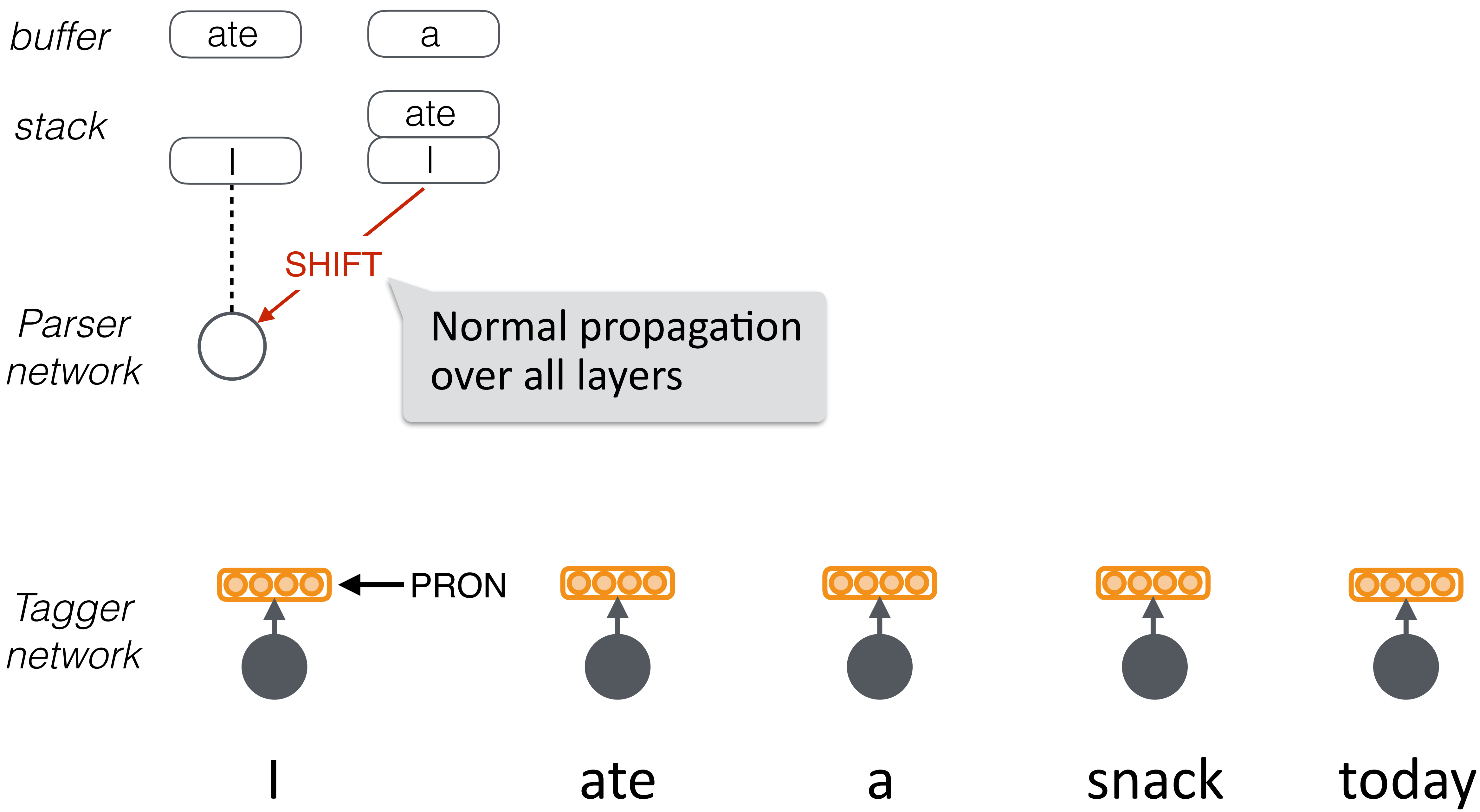


- Updating schema
  - ✦ Tagging: update tagger
  - ✦ Parsing: update parser & tagger except for softmax

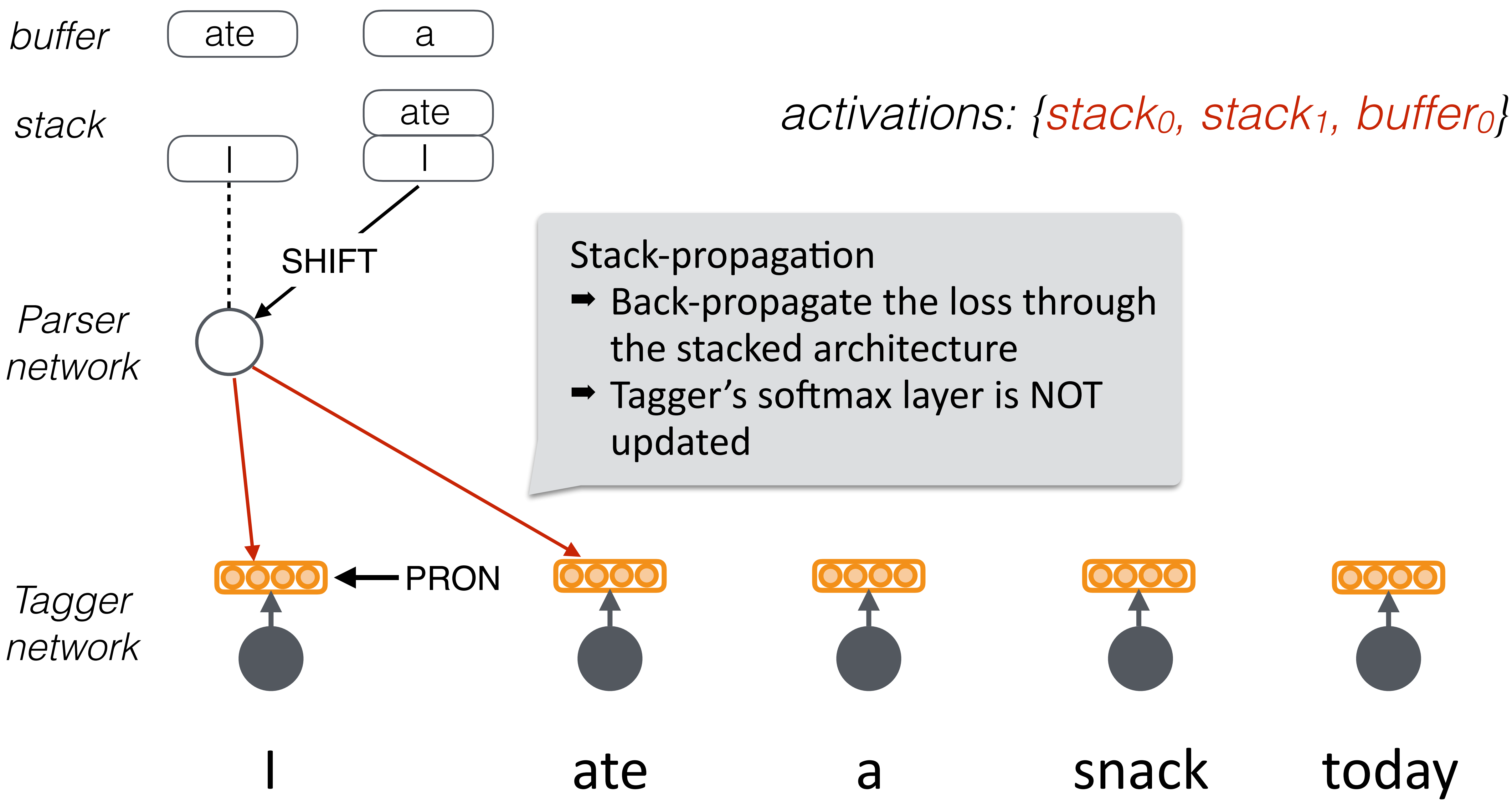
Unrolled NN Architecture (Update Tagger)



Unrolled NN Architecture (Update Parser)

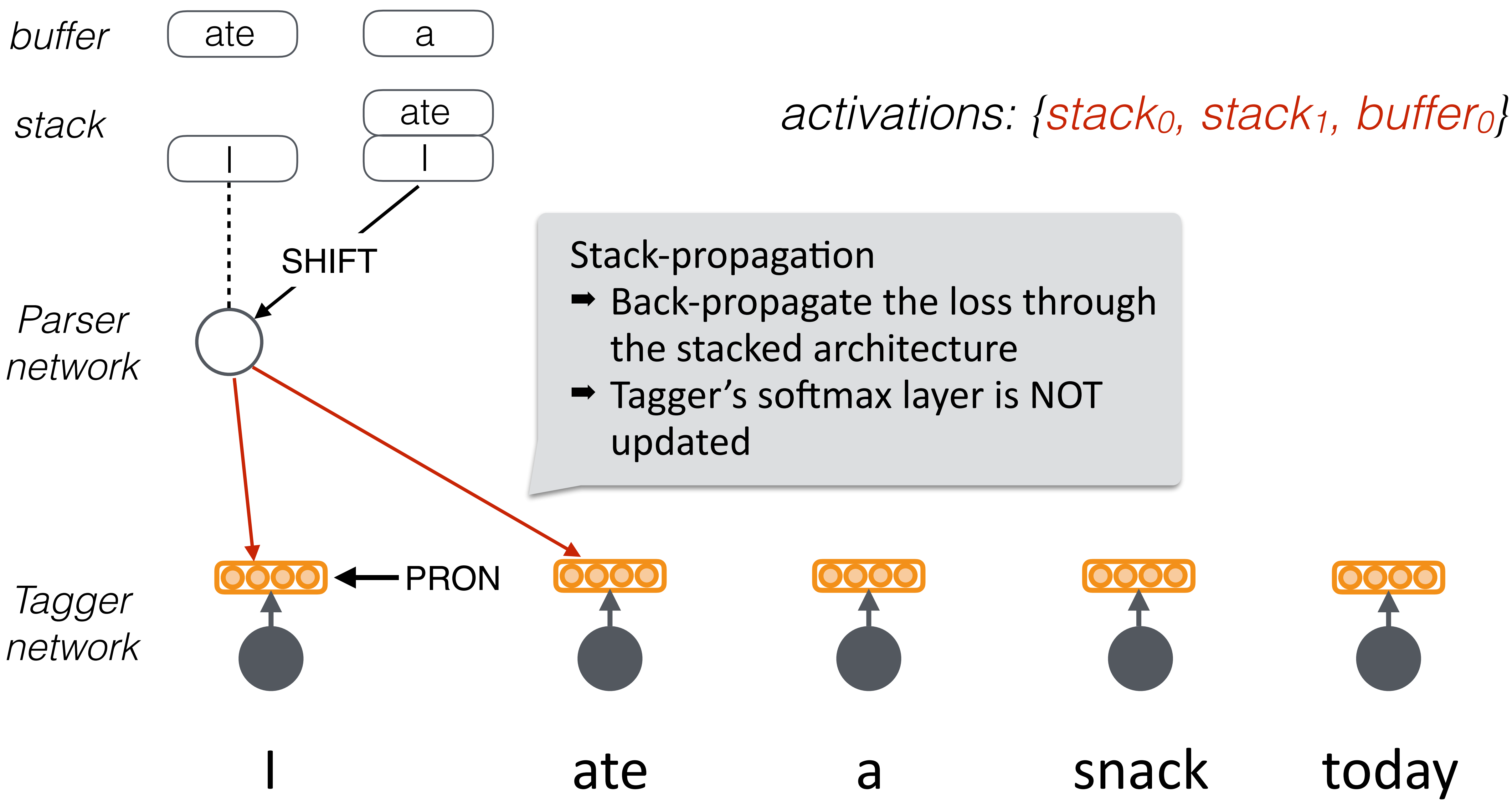


Unrolled NN Architecture (Update Parser)



Stack-propagation
➡ Back-propagate the loss through the stacked architecture
➡ Tagger's softmax layer is NOT updated

Unrolled NN Architecture (Update Parser)



Stack-propagation
➡ Back-propagate the loss through the stacked architecture
➡ Tagger's softmax layer is NOT updated

- Other learning recipes borrowed from Weiss'15

Experimental Setup

- Dataset
    - ✦ 19 languages from the Universal Dependencies 1.2
    - ✦ Wall Street Journal (Stanford converter 3.3.0)

Experimental Setup

- Dataset
  - ✦ 19 languages from the Universal Dependencies 1.2
  - ✦ Wall Street Journal (Stanford converter 3.3.0)

- Transition system
  - ✦ Arc-standard with greedy decoding
  - ✦ Exception for Dutch, adding SWAP action

Experimental Setup

- Dataset
  - ✦ 19 languages from the Universal Dependencies 1.2
  - ✦ Wall Street Journal (Stanford converter 3.3.0)

- Transition system
  - ✦ Arc-standard with greedy decoding
  - ✦ Exception for Dutch, adding SWAP action

- Word embeddings
  - ✦ No pre-trained embeddings for UD
  - ✦ Pre-trained embeddings for WSJ

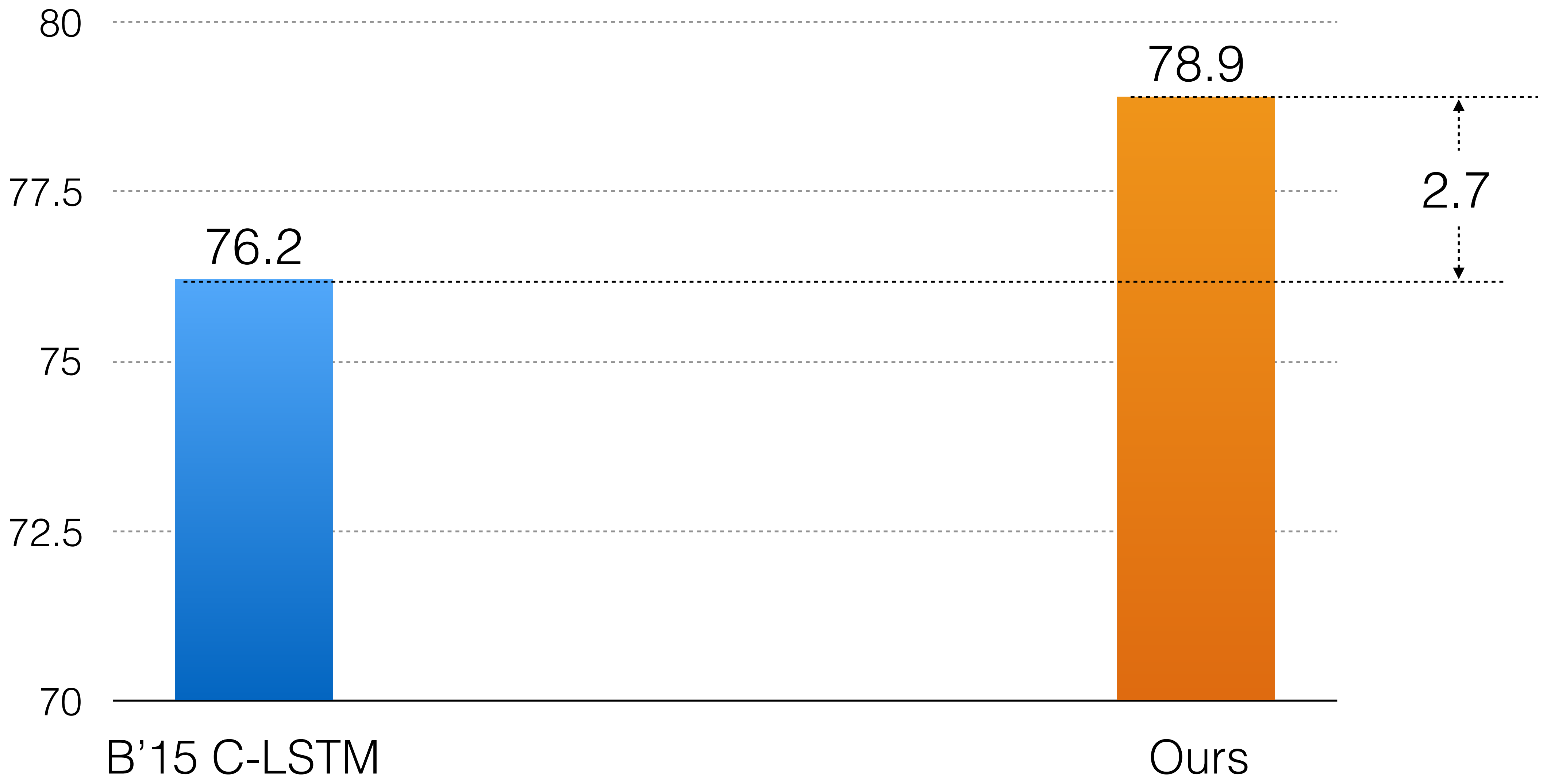
Baselines

- Transition-based NN system with greedy decoding
  - ✦ D'15 W-LSTM (Dyer'15): word-based LSTM
  - ✦ B'15 C-LSTM (Ballesteros'15): character-based LSTM
  - ✦ A'15 Joint (Alberti'15): joint parsing & tagging system

- Graph-based system
  - ✦ RBGParser (Lei'14, Zhang'14): state-of-the-art graph-based parser

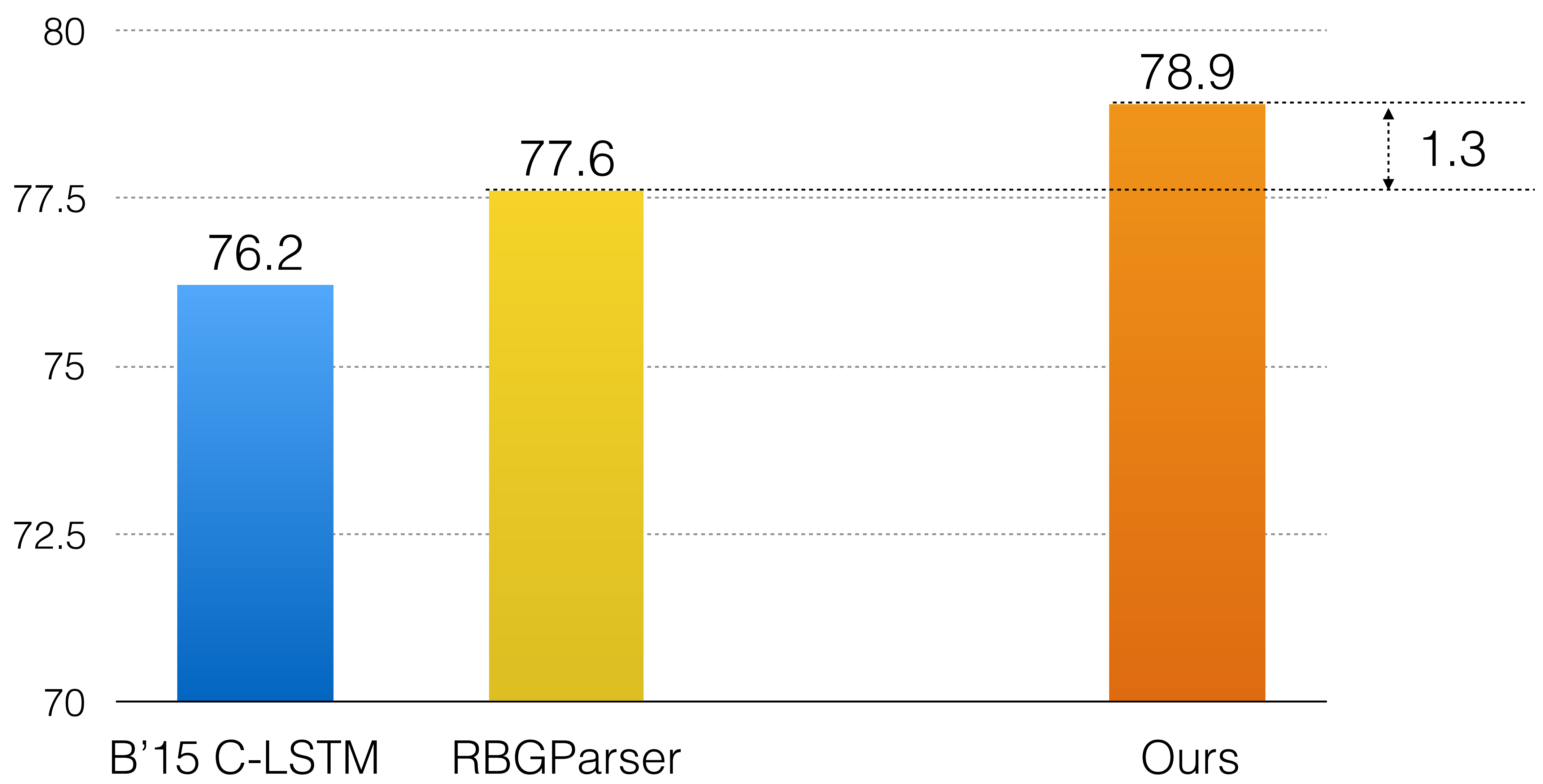
Results on Universal Dependencies

Averaged Labeled Attachment Score (LAS) on 19 Languages

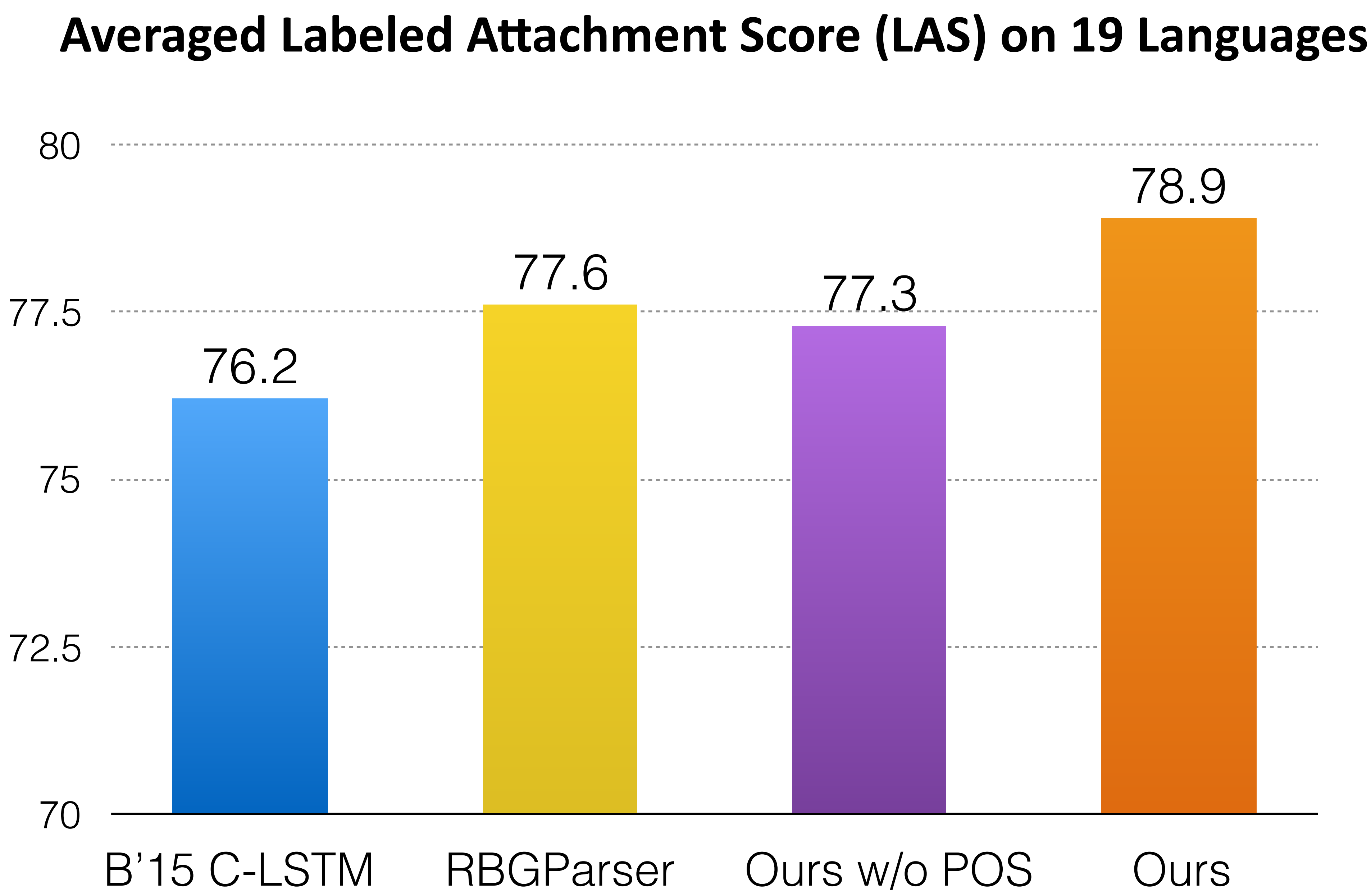


Results on Universal Dependencies

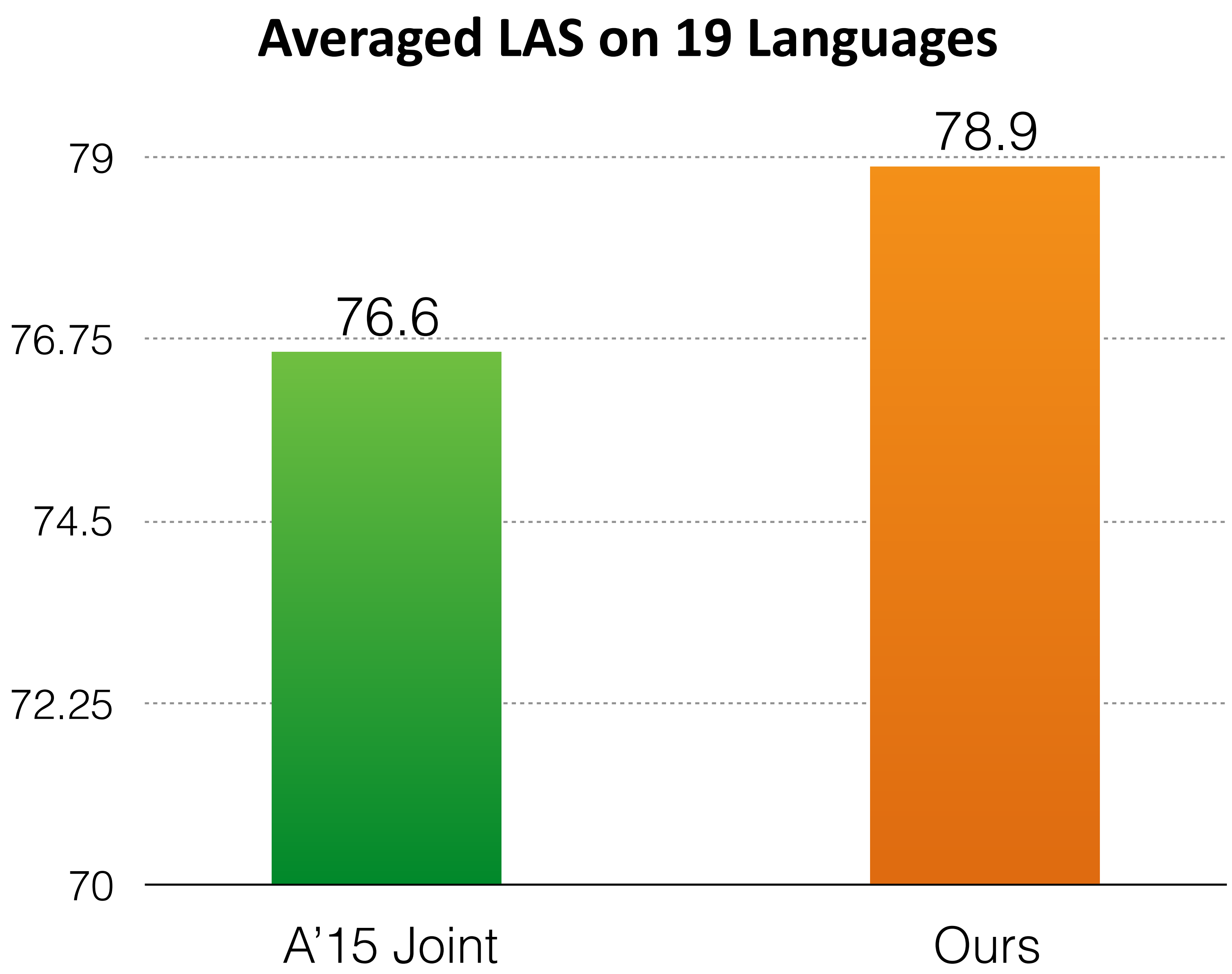
Averaged Labeled Attachment Score (LAS) on 19 Languages



Results on Universal Dependencies



Stackprop vs. Joint Modeling



- Our learned representations are better than discrete feature vectors

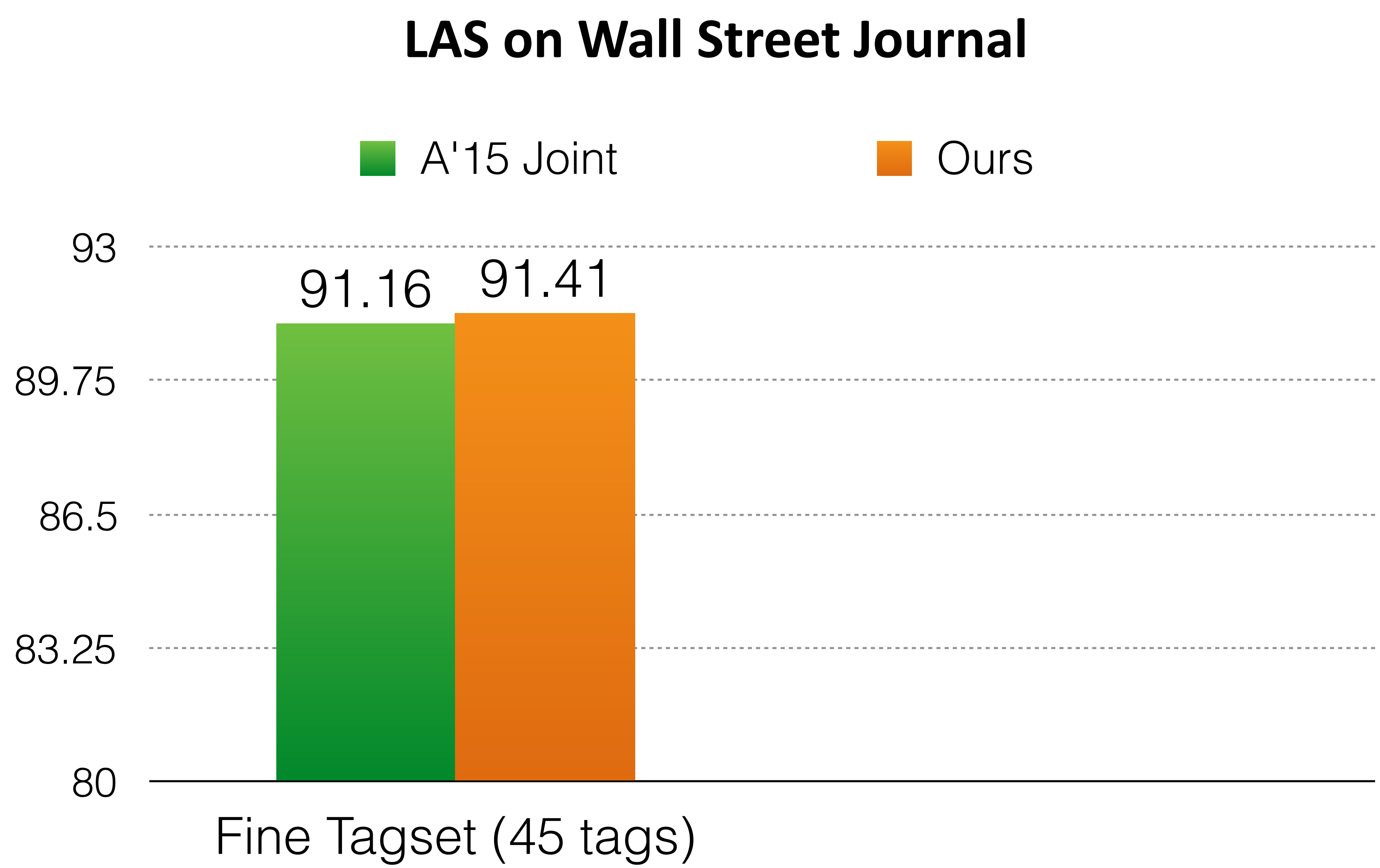
Results on Wall Street Journal

Method	WSJ §23 UAS	LAS	
D'15 W-LSTM	93.10	90.90	
A'15 Joint	93.10	91.16	Beam=1
Ours	**93.43**	**91.41**	

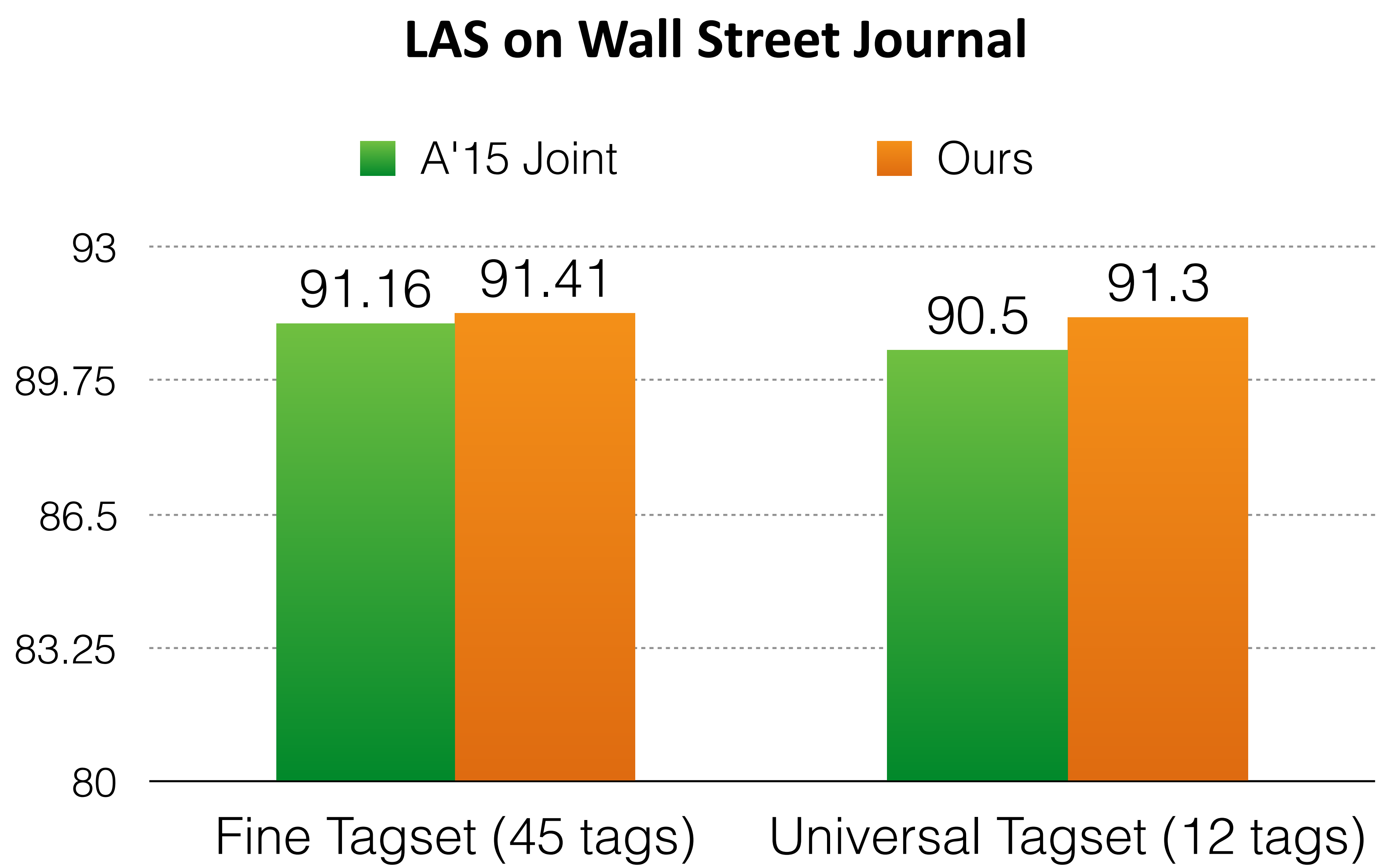
Results on Wall Street Journal

Method	WSJ §23 UAS	WSJ §23 LAS	
D'15 W-LSTM	93.10	90.90	Beam=1
A'15 Joint	93.10	91.16	
Ours	**93.43**	**91.41**	
Weiss et al. (2015)	93.99	92.05	Beam=8
Alberti et al. (2015)	94.23	92.36	

Impact of Tagset Size

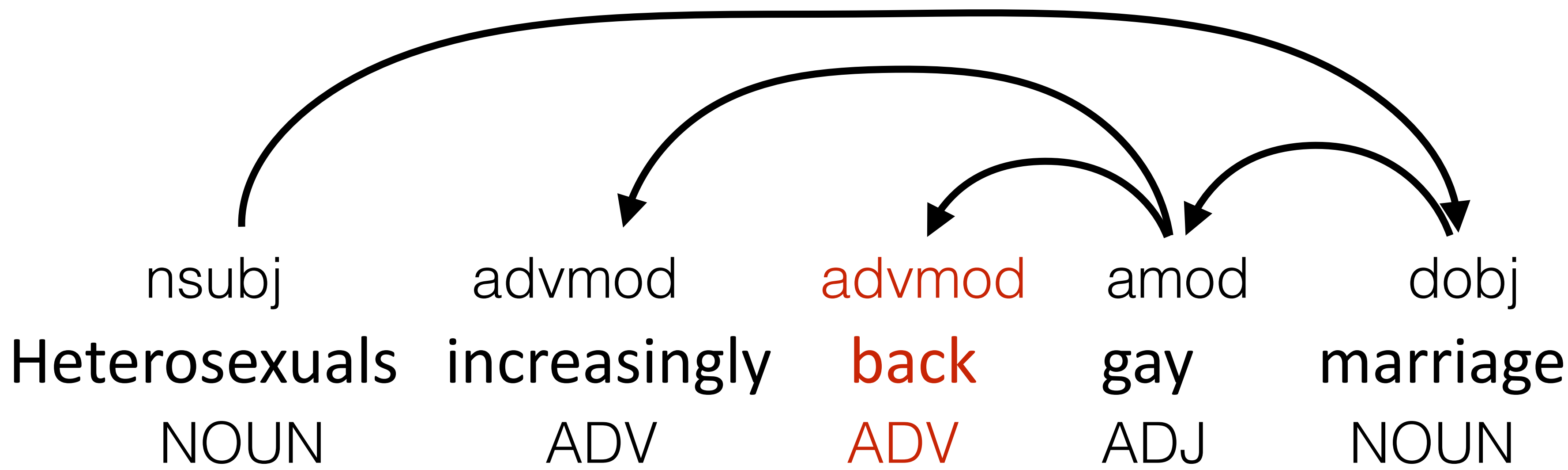


Impact of Tagset Size

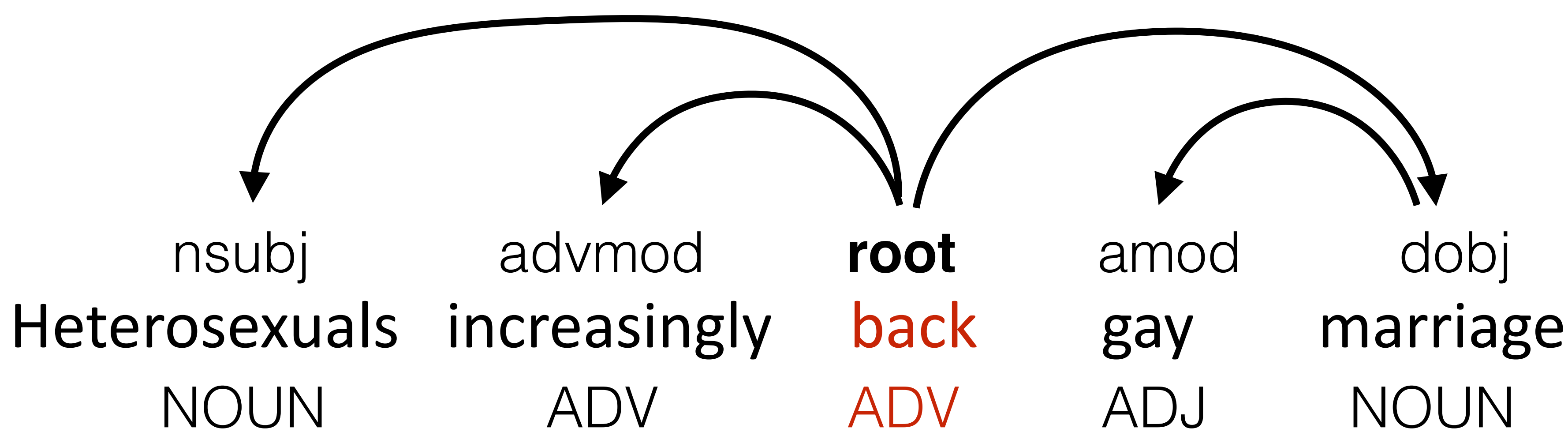


- More gains when only coarse tags are available

Reducing Error Propagations



Tree by a pipeline model



Tree by Stackprop model

- 10.9% LAS gain (34.1% vs 45.0%) on tokens with wrong POS tags

Conclusions

- *Modeling:* we present a stacking neural network model for dependency parsing and tagging

Conclusions

- *Modeling:* we present a stacking neural network model for dependency parsing and tagging

- *Performance:* our model outperforms all baselines when evaluated on 19 languages of the UD treebank and outperforms other greedy models on the WSJ

Conclusions

- *Modeling:* we present a stacking neural network model for dependency parsing and tagging
- *Performance:* our model outperforms all baselines when evaluated on 19 languages of the UD treebank and outperforms other greedy models on the WSJ
- *Opportunities and Challenges:* we hope to apply this stacking idea to other structures and NLP problems.

Thank You!