Stack Attention

Brian DuSell and David Chiang

April 22nd, 2024

Seminars on Formal Languages and Neural Networks

UNIVERSITY OF NOTRE DAME

ETH zürich

FLaNN

Paper for This Talk

Published as a conference paper at ICLR 2024

STACK ATTENTION: IMPROVING THE ABILITY OF TRANSFORMERS TO MODEL HIERARCHICAL PATTERNS

Brian DuSell*
Department of Computer Science
ETH Zürich
brian.dusell@inf.ethz.ch

David Chiang
Department of Computer Science and Engineering
University of Notre Dame
dchiang@nd.edu

ABSTRACT

Attention, specifically scaled dot-product attention, has proven effective for natural language, but it does not have a mechanism for handling hierarchical patterns

ICLR To appear as a spotlight paper

Limitations of Transformers

- Cannot recognize the language of balanced brackets (Dyck-2) for arbitrary lengths and depths (Hahn, 2020)

```
( [ [ [ ( [ [ ( ( ( ) [ [ [ ( ( ( ) ) ) ] ] ] ) ) ] ] ) ] ] ( ) ] ) ( ( ( ) ) )
[ [ ( [ ( ) [ ( ( ) ) ] ] ) ] ] [ ( ( ( [ [ ( [ ( ( [ [ ] ] ) ) ] ) ] ] ) ) ) ]
( ( ( [ [ ( ( ( ( ( ( ) ( ( ( [ [ ] ] ) ) ) ) ) ) ) ) ] ] ) ) ) ( [ ( [ ] ) ] )
```

Limitations of Transformers

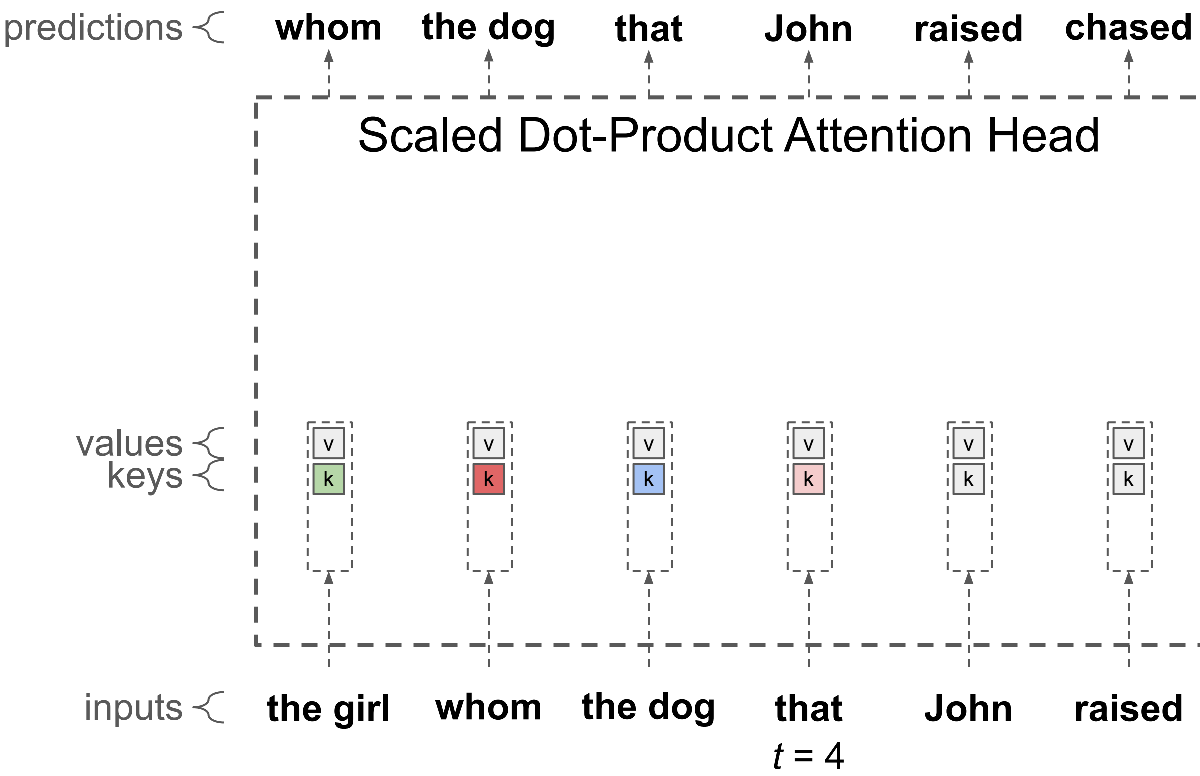
Recognizable by Transformers

- Dyck-2 with bounded depth

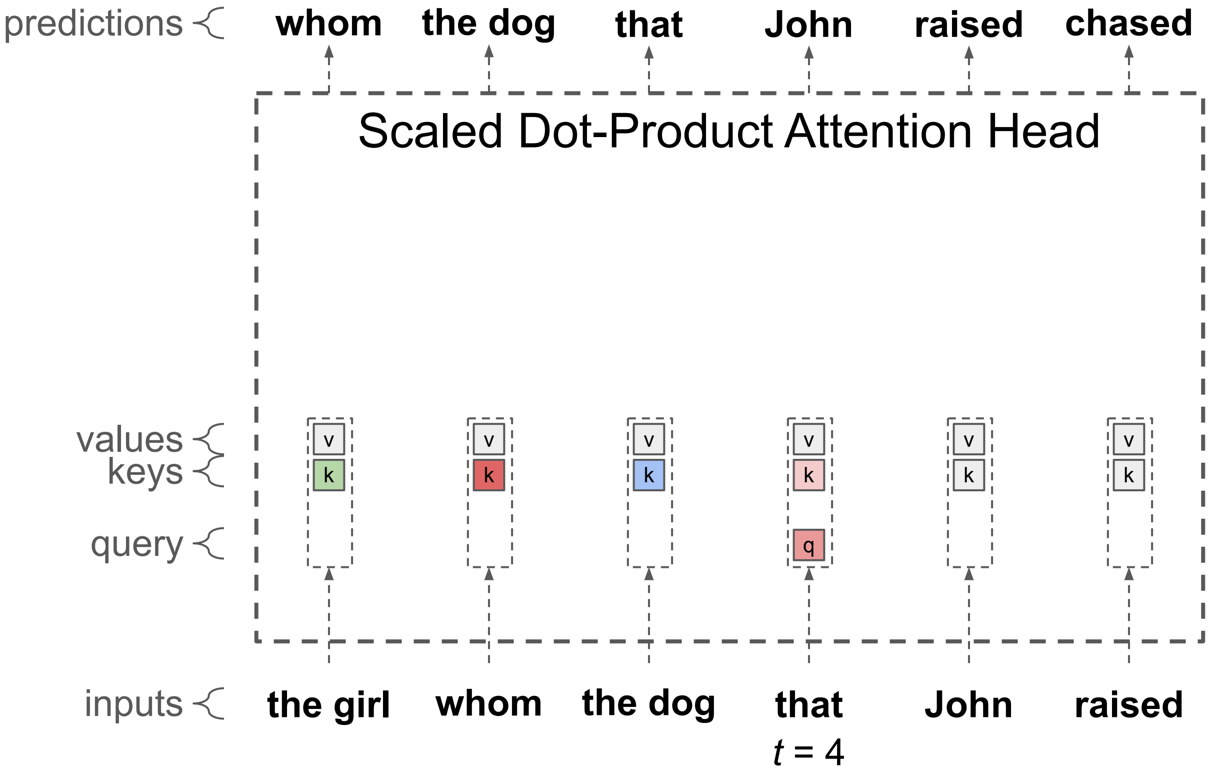
Not Recognizable by Transformers

- Dyck-2
- Natural Language?
- Code

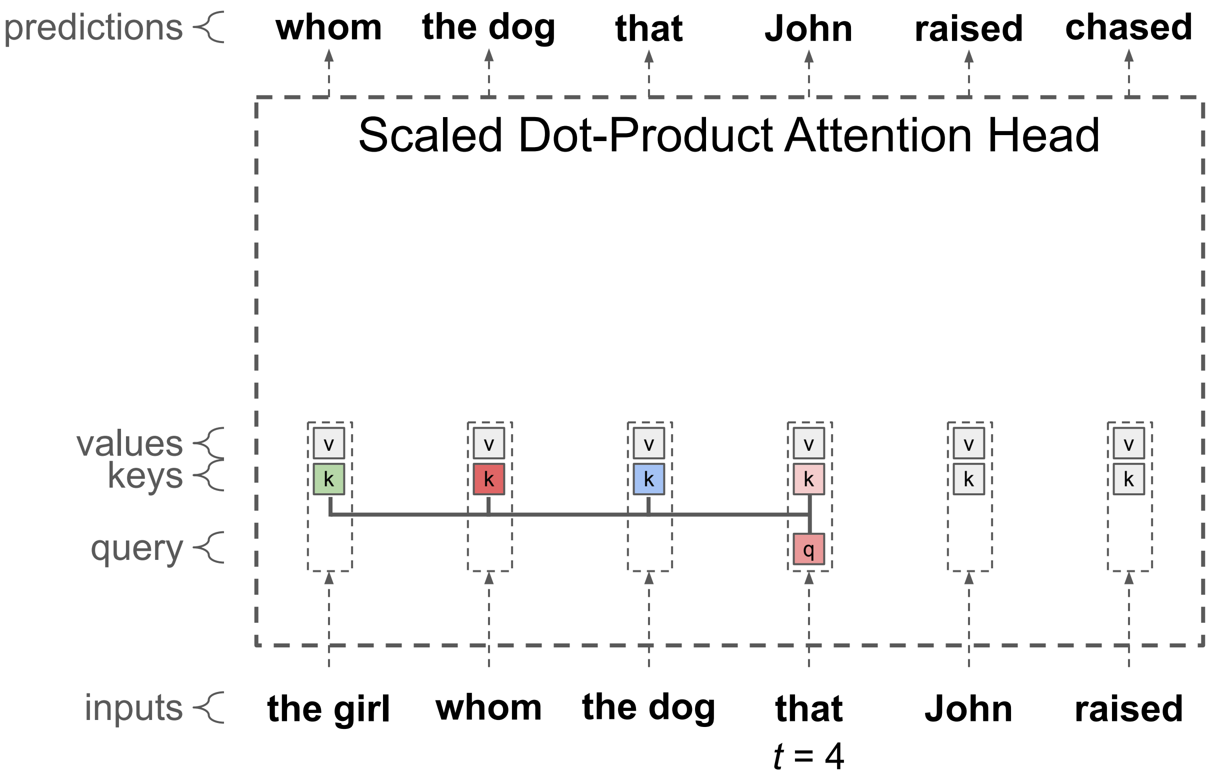
Self-Attention



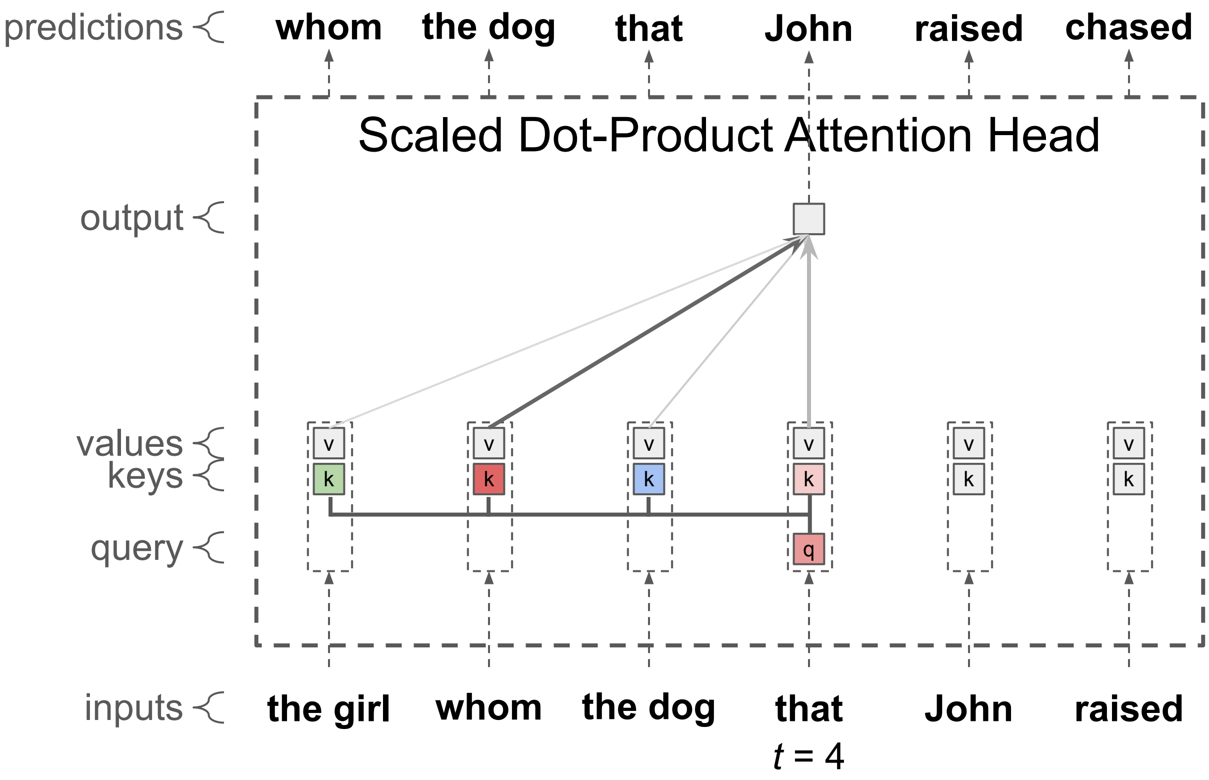
Self-Attention



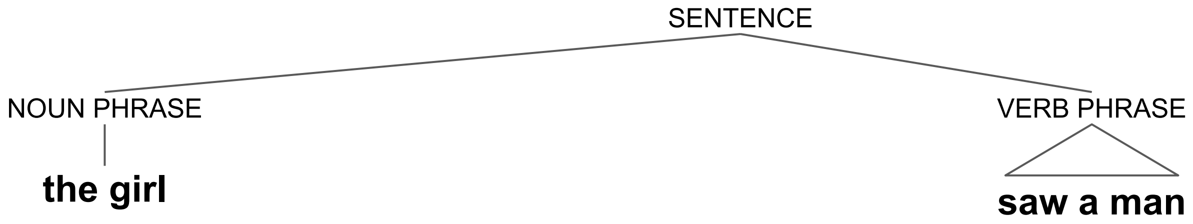
Self-Attention



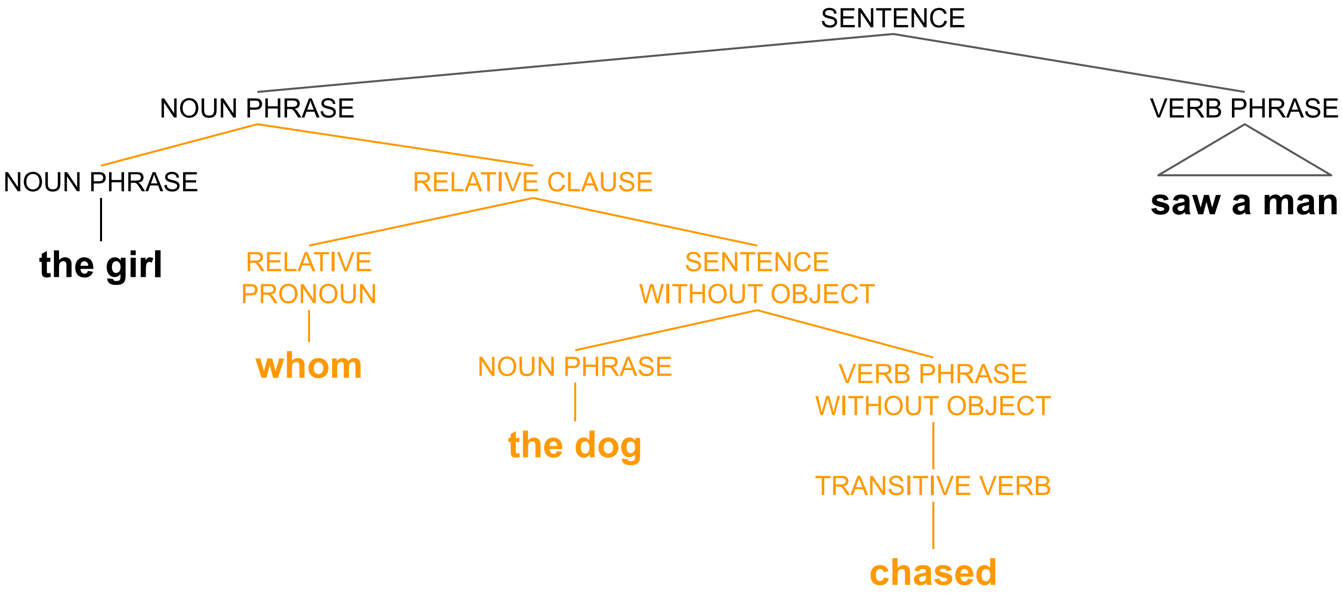
Self-Attention



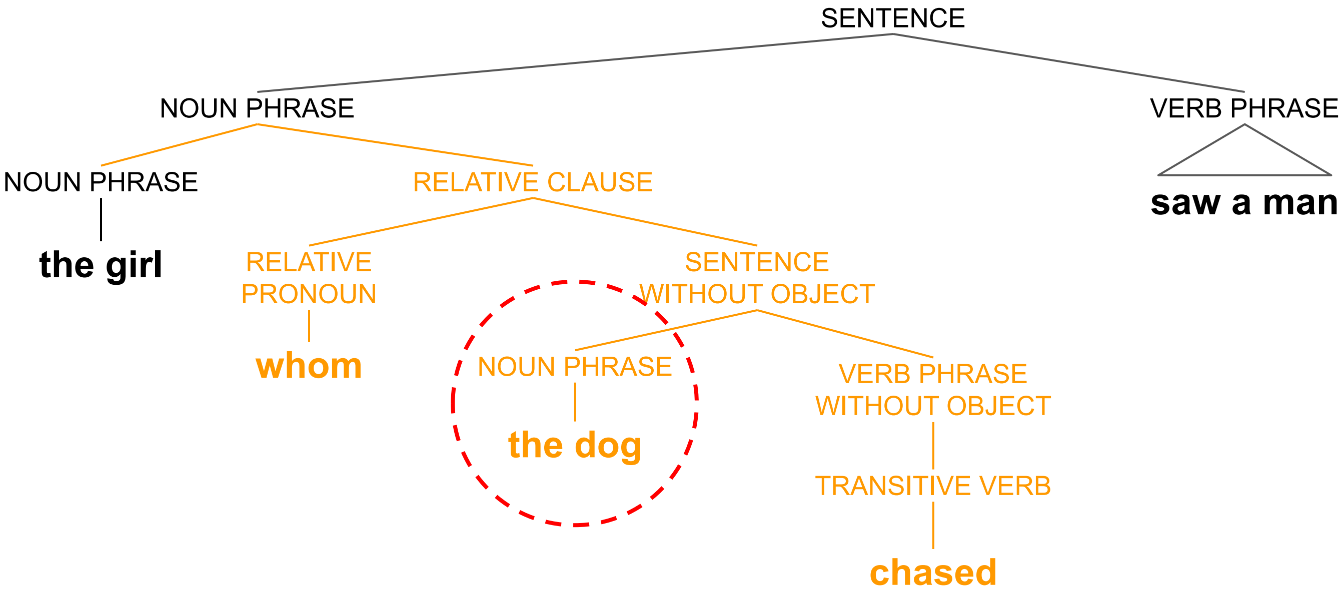
Recursion



Recursion

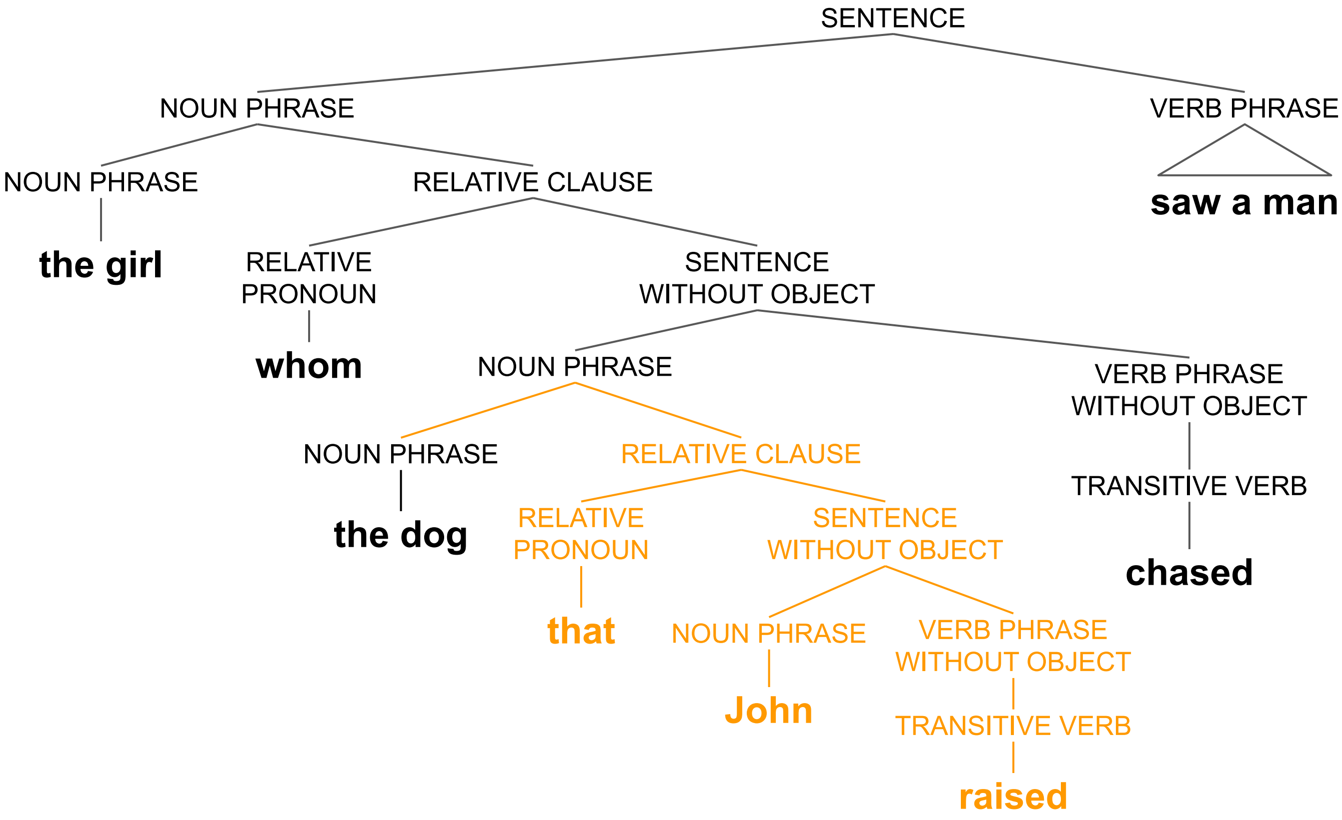

SENTENCE
NOUN PHRASE
VERB PHRASE
NOUN PHRASE
RELATIVE CLAUSE
saw a man
the girl
RELATIVE PRONOUN
SENTENCE WITHOUT OBJECT
whom
NOUN PHRASE
VERB PHRASE WITHOUT OBJECT
the dog
TRANSITIVE VERB
chased

Recursion



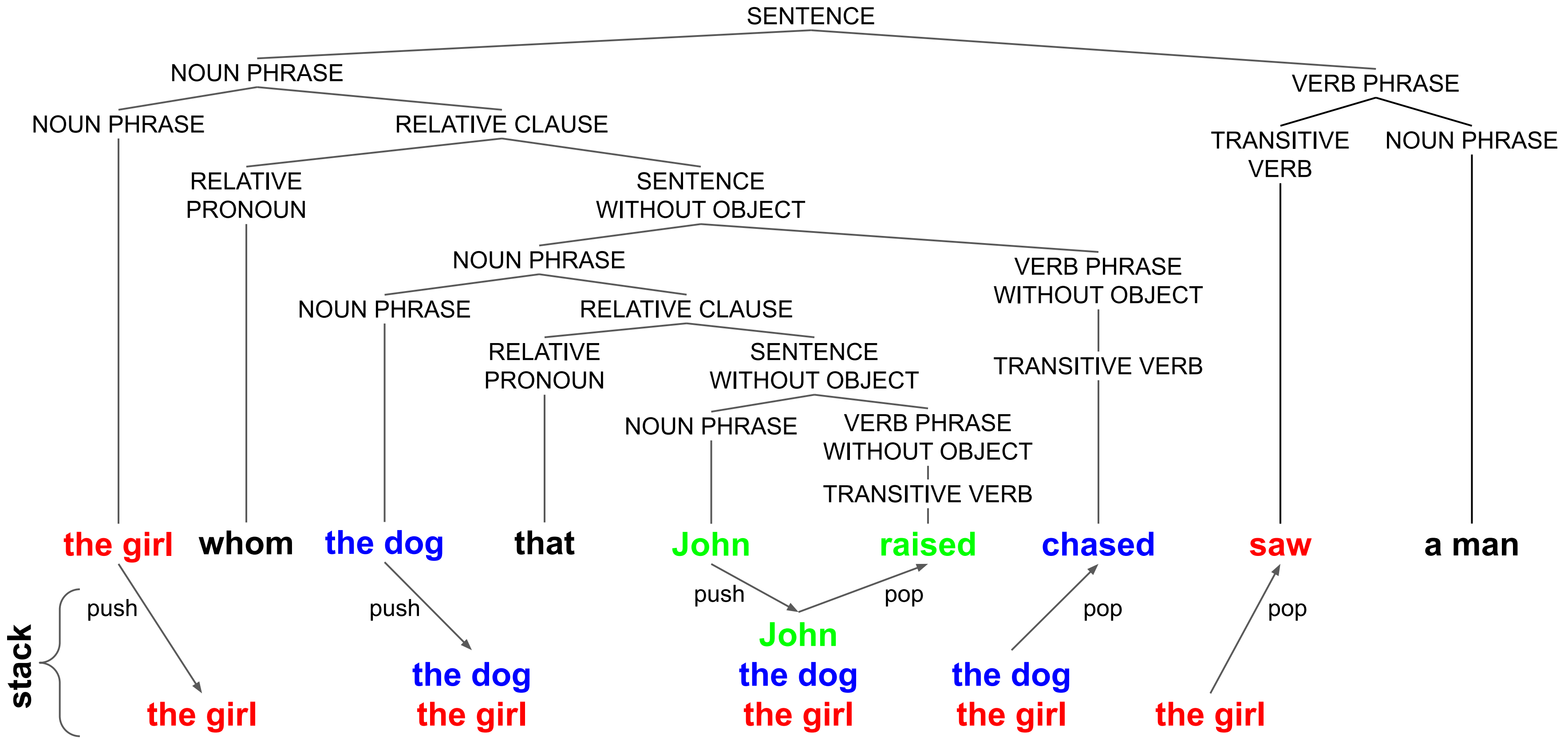
SENTENCE
NOUN PHRASE
VERB PHRASE
NOUN PHRASE
RELATIVE CLAUSE
saw a man
the girl
RELATIVE PRONOUN
SENTENCE WITHOUT OBJECT
whom
NOUN PHRASE
the dog
VERB PHRASE WITHOUT OBJECT
TRANSITIVE VERB
chased

Recursion

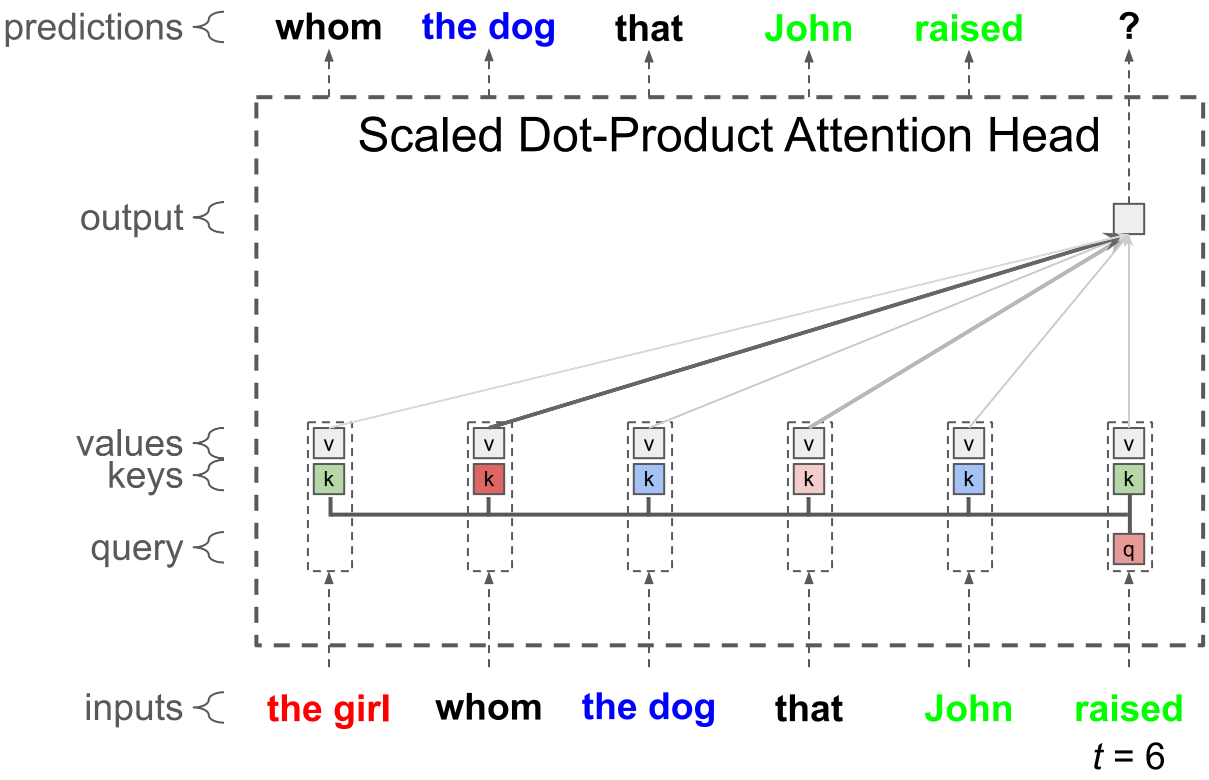


SENTENCE
NOUN PHRASE
VERB PHRASE
saw a man
NOUN PHRASE
the girl
RELATIVE CLAUSE
RELATIVE PRONOUN
whom
SENTENCE WITHOUT OBJECT
NOUN PHRASE
NOUN PHRASE
the dog
RELATIVE CLAUSE
RELATIVE PRONOUN
that
SENTENCE WITHOUT OBJECT
NOUN PHRASE
John
VERB PHRASE WITHOUT OBJECT
TRANSITIVE VERB
raised
VERB PHRASE WITHOUT OBJECT
TRANSITIVE VERB
chased

Stacks = Syntax



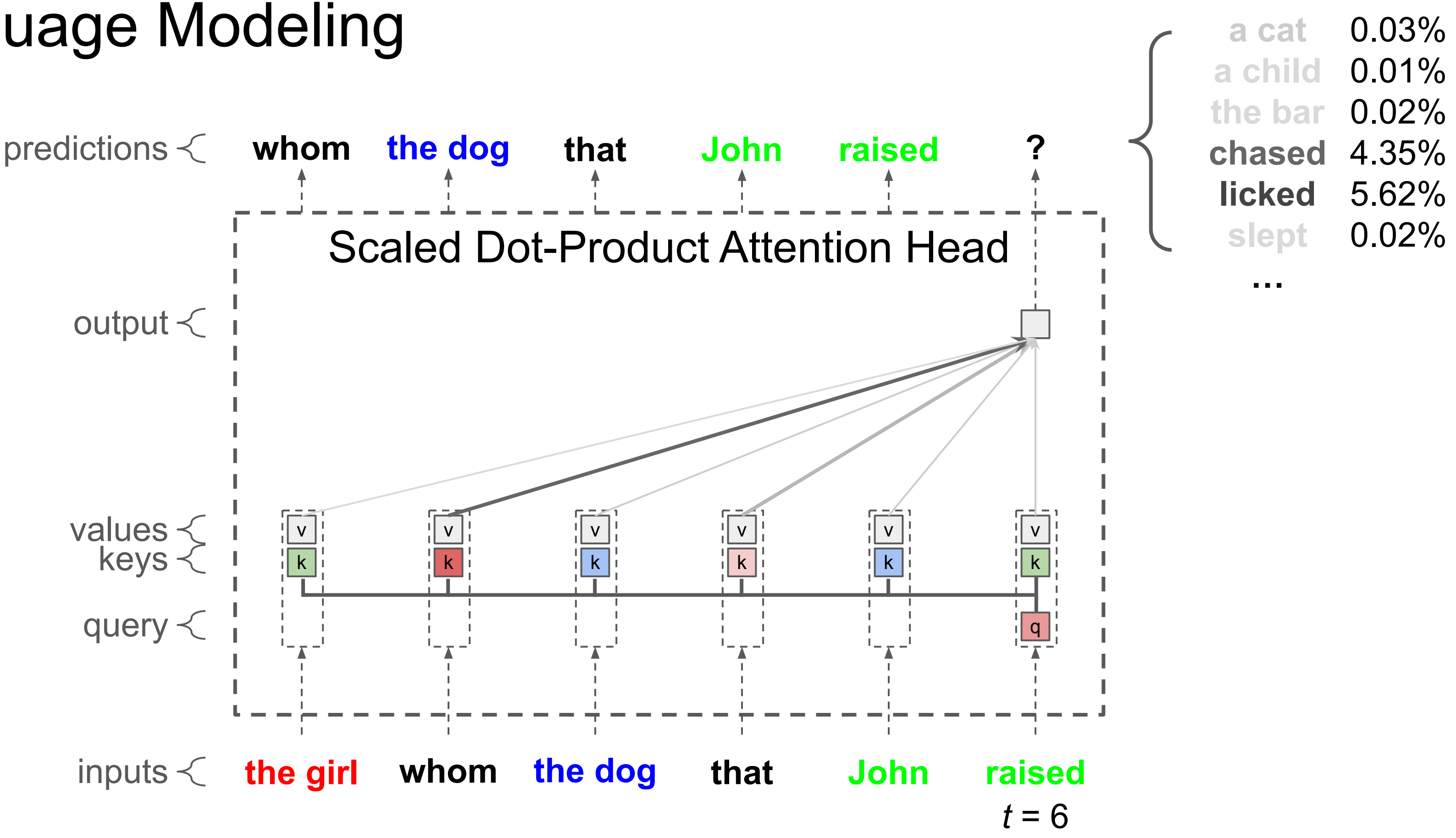
Language Modeling



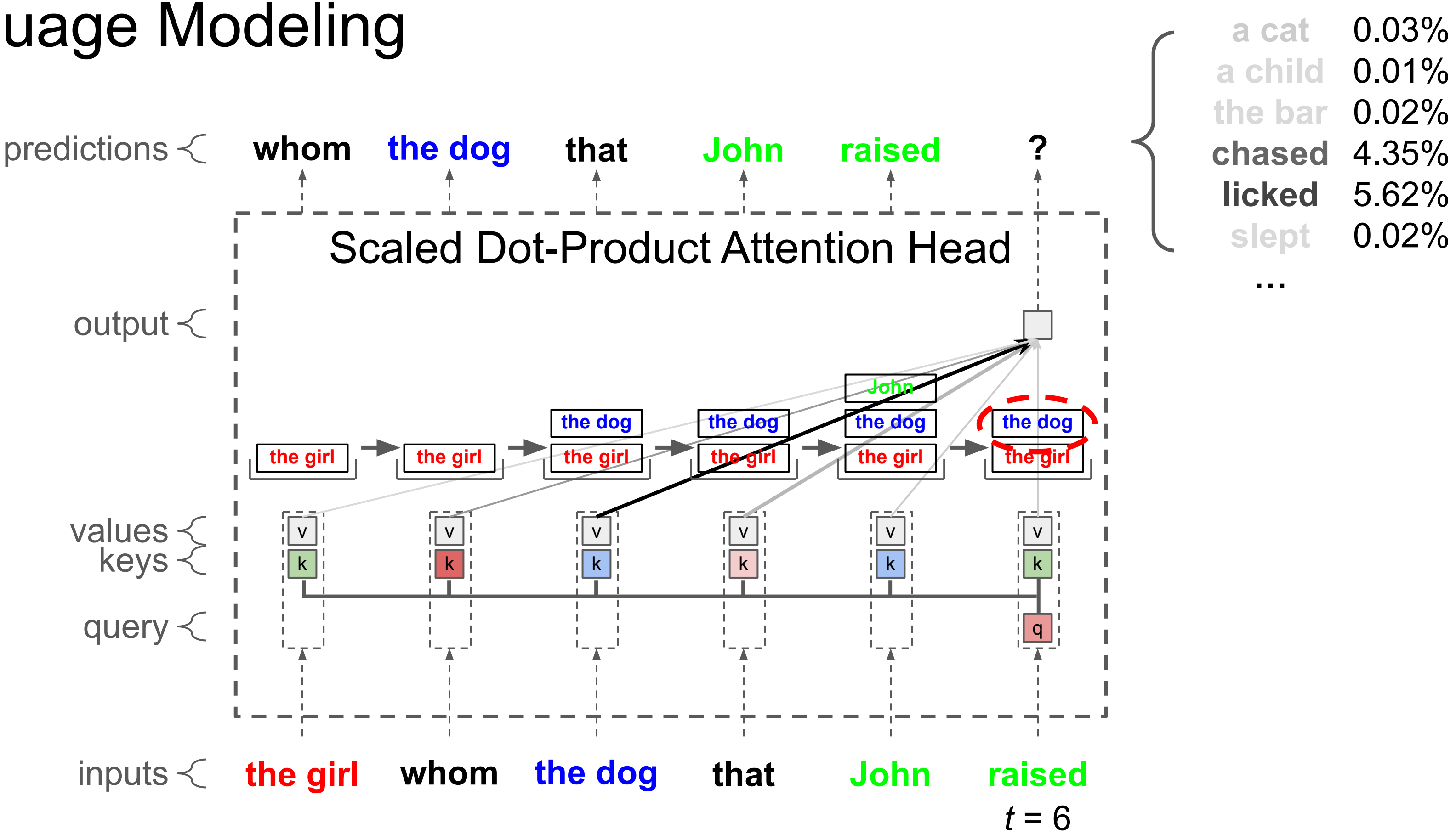
a cat	5.01%
a child	7.58%
the bar	6.12%
chased	0.02%
licked	0.03%
slept	0.02%
...	

Transformers struggle on this (Lakretz et al., 2022)

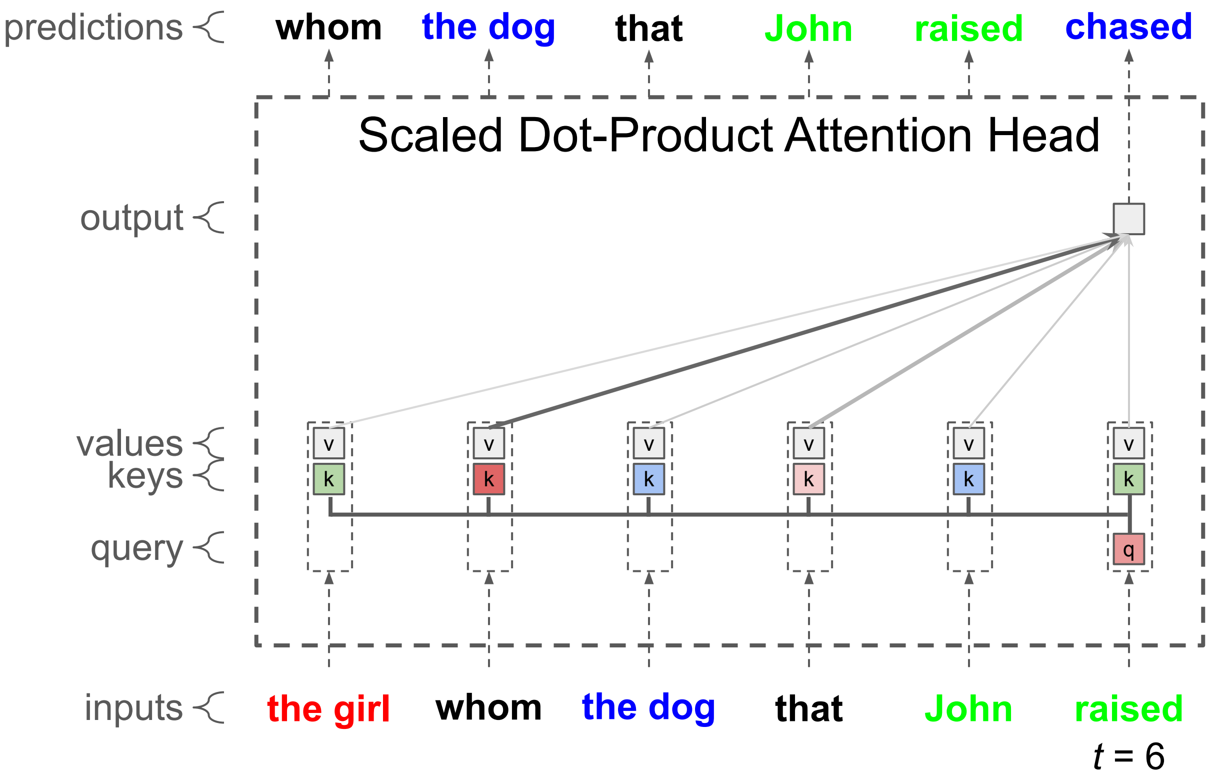
Language Modeling



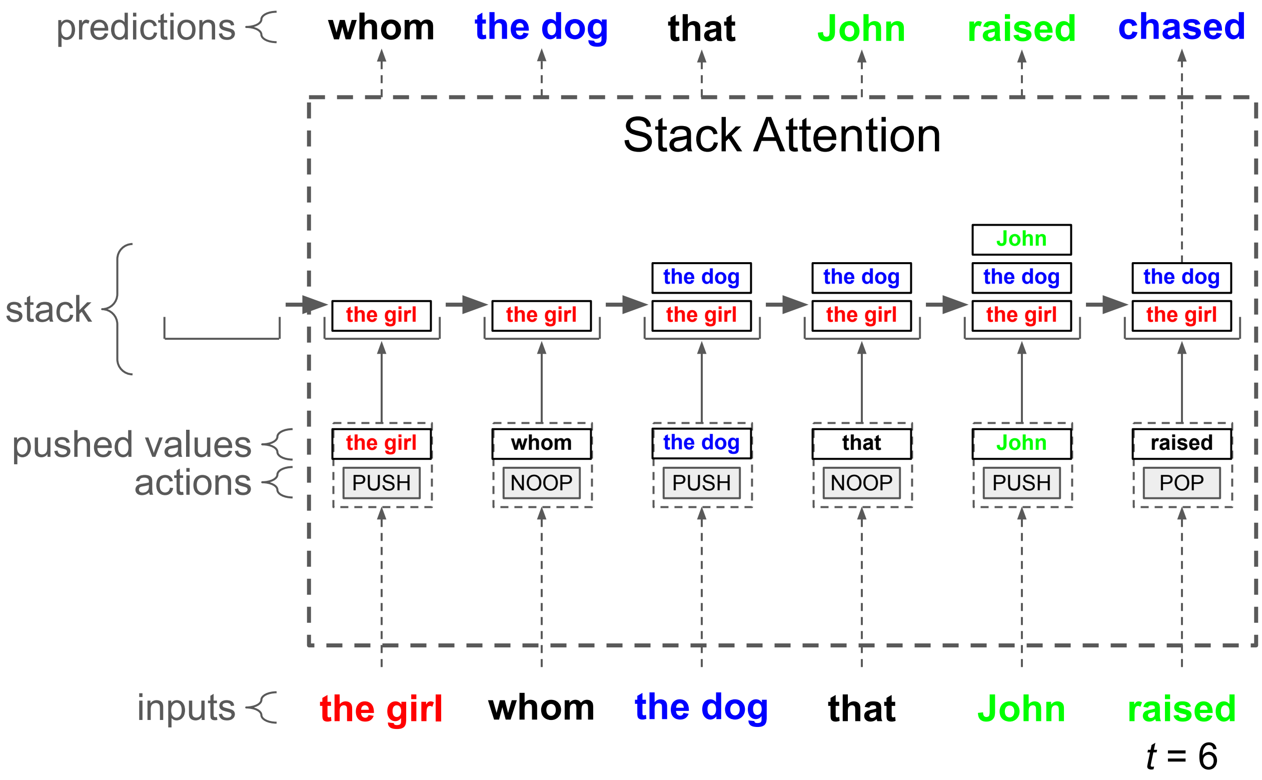
Language Modeling



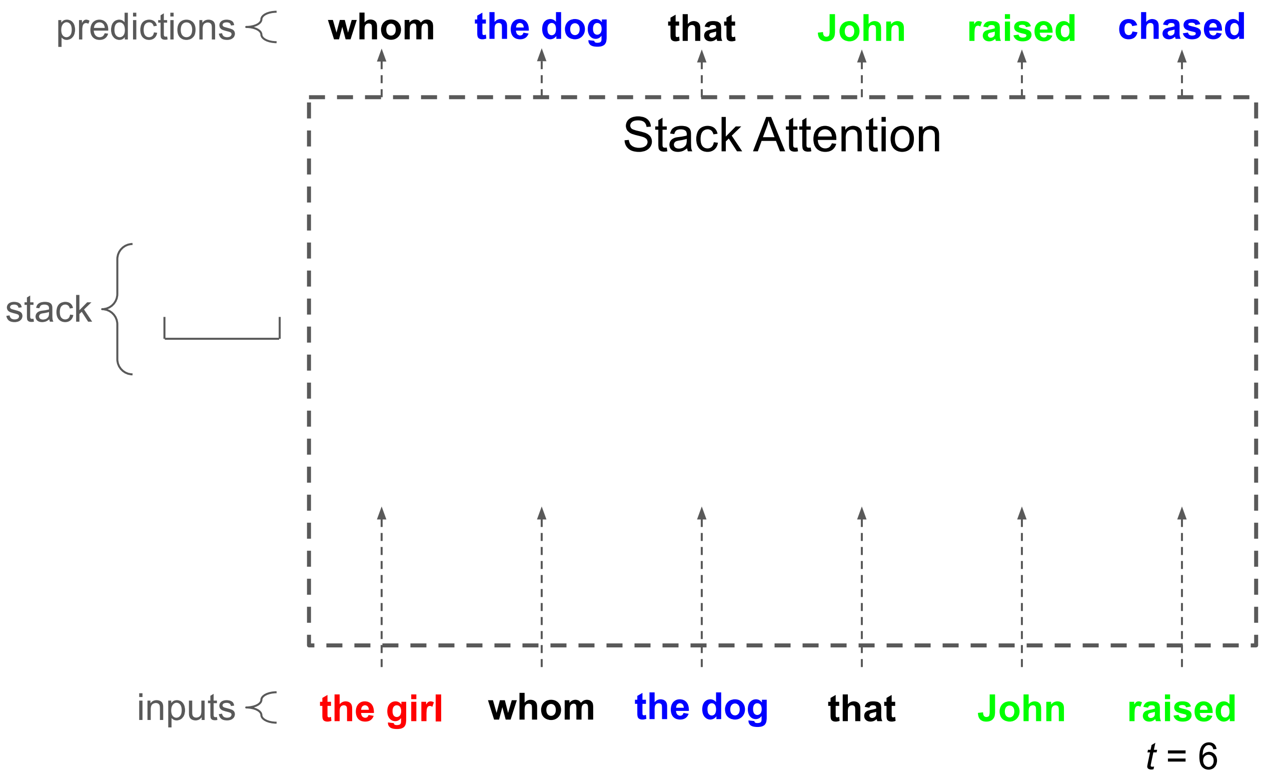
Self-Attention



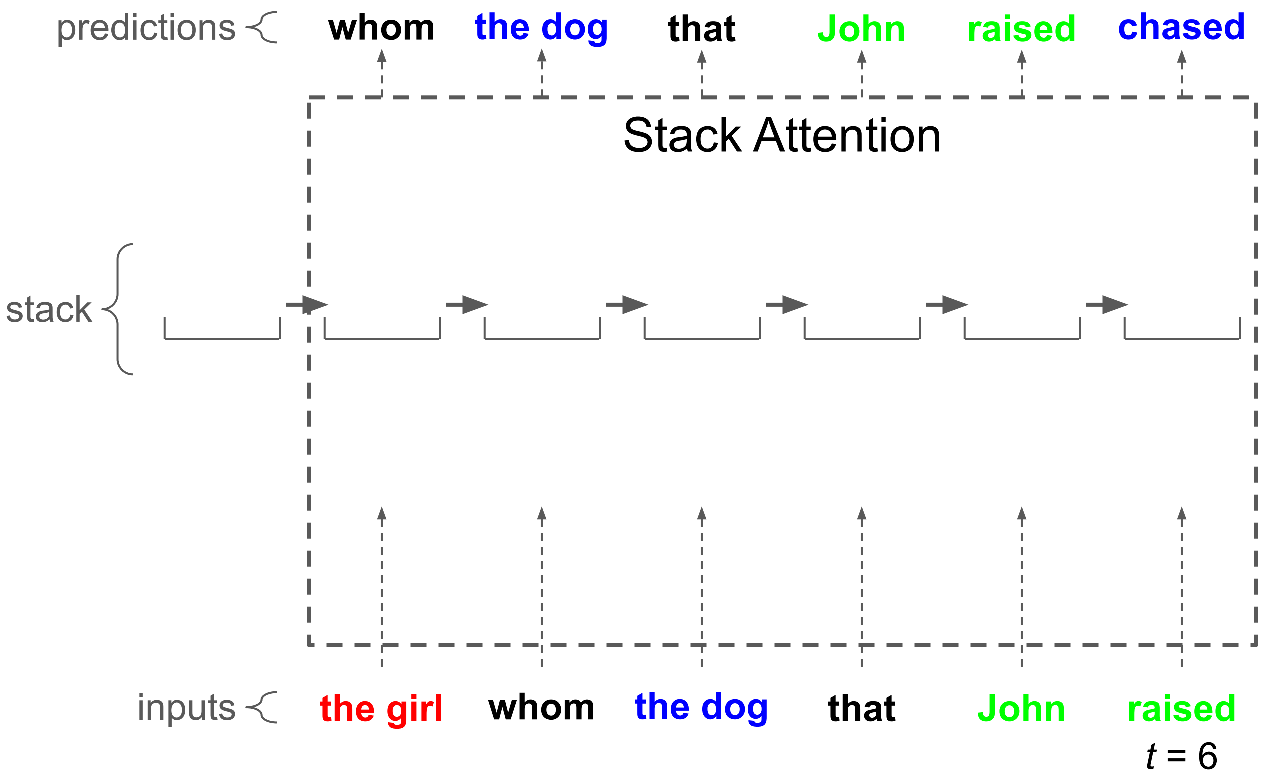
Our Method: Stack Attention



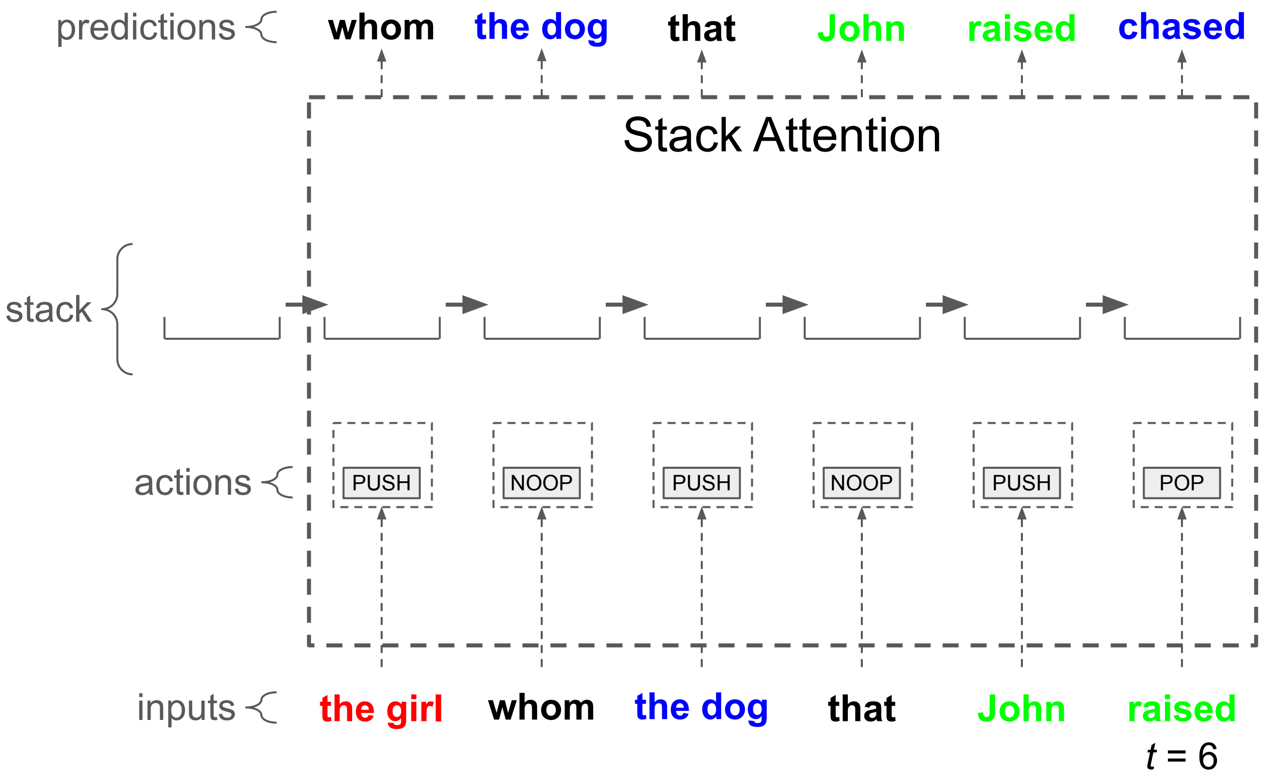
Our Method: Stack Attention



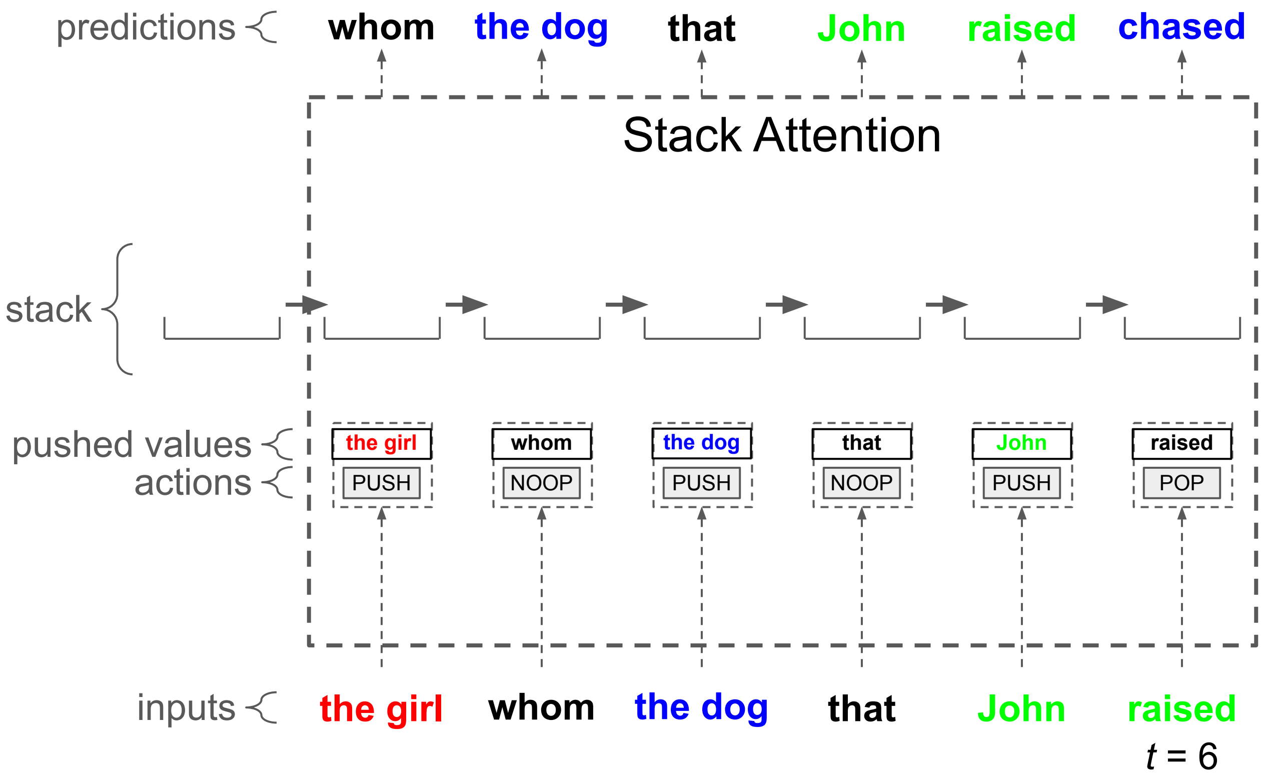
Our Method: Stack Attention



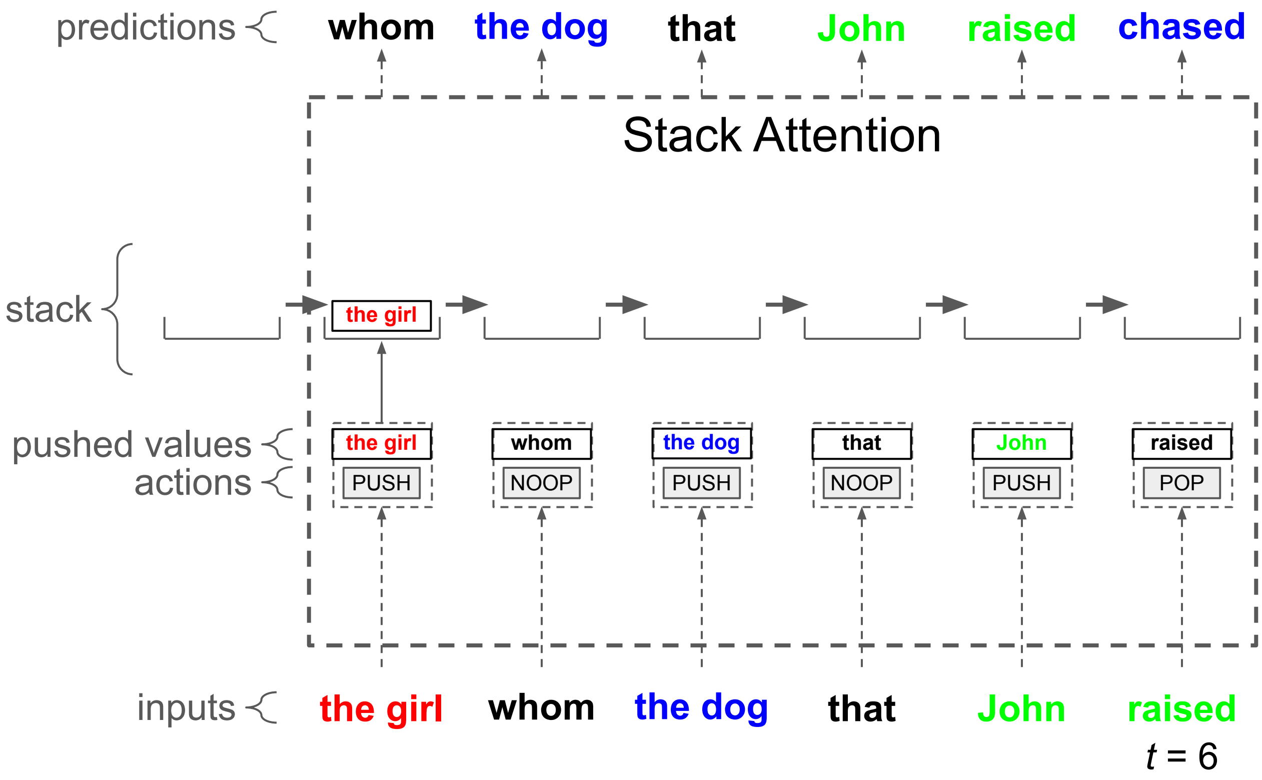
Our Method: Stack Attention



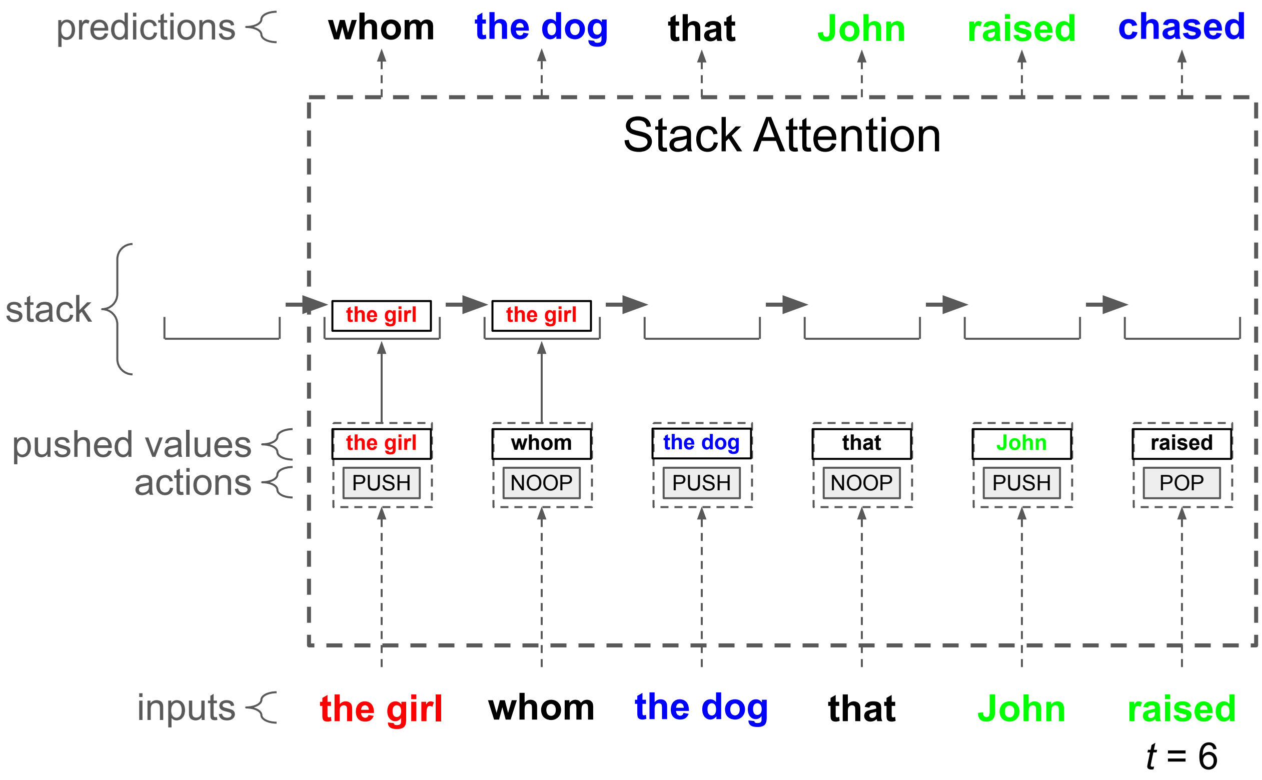
Our Method: Stack Attention



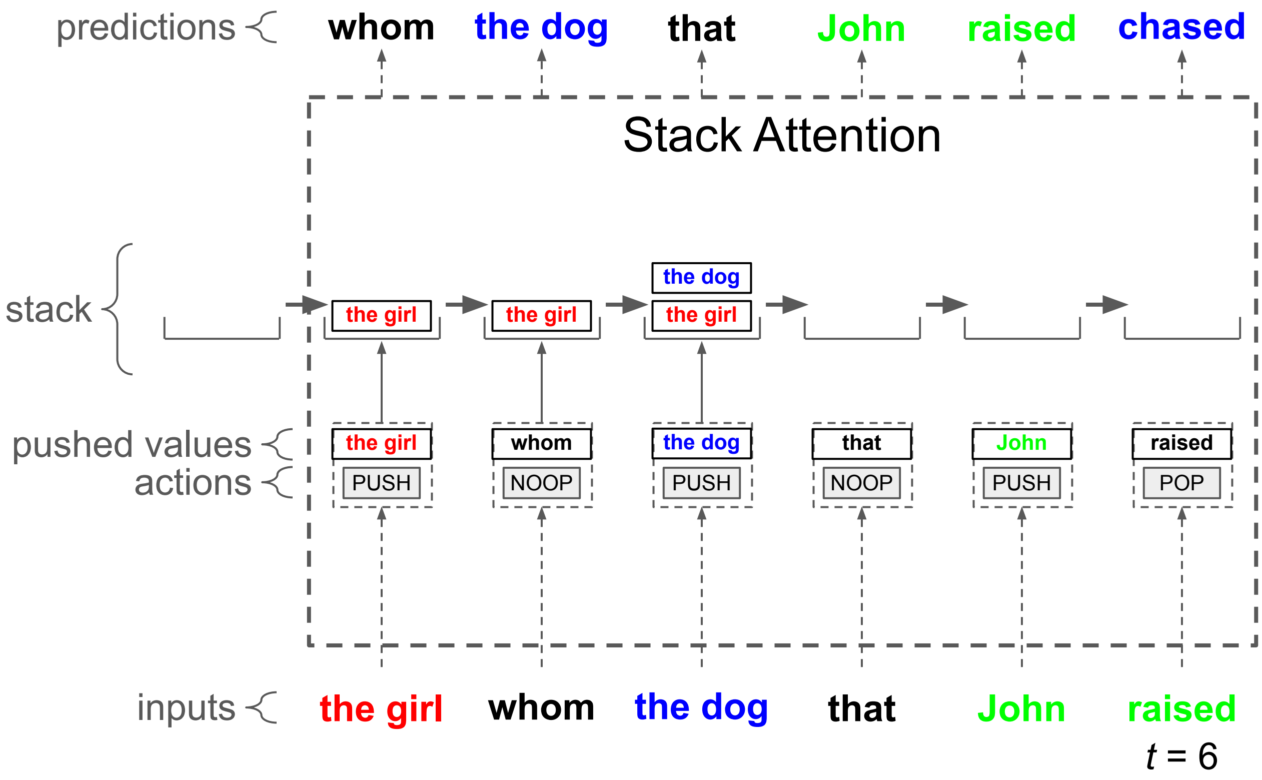
Our Method: Stack Attention



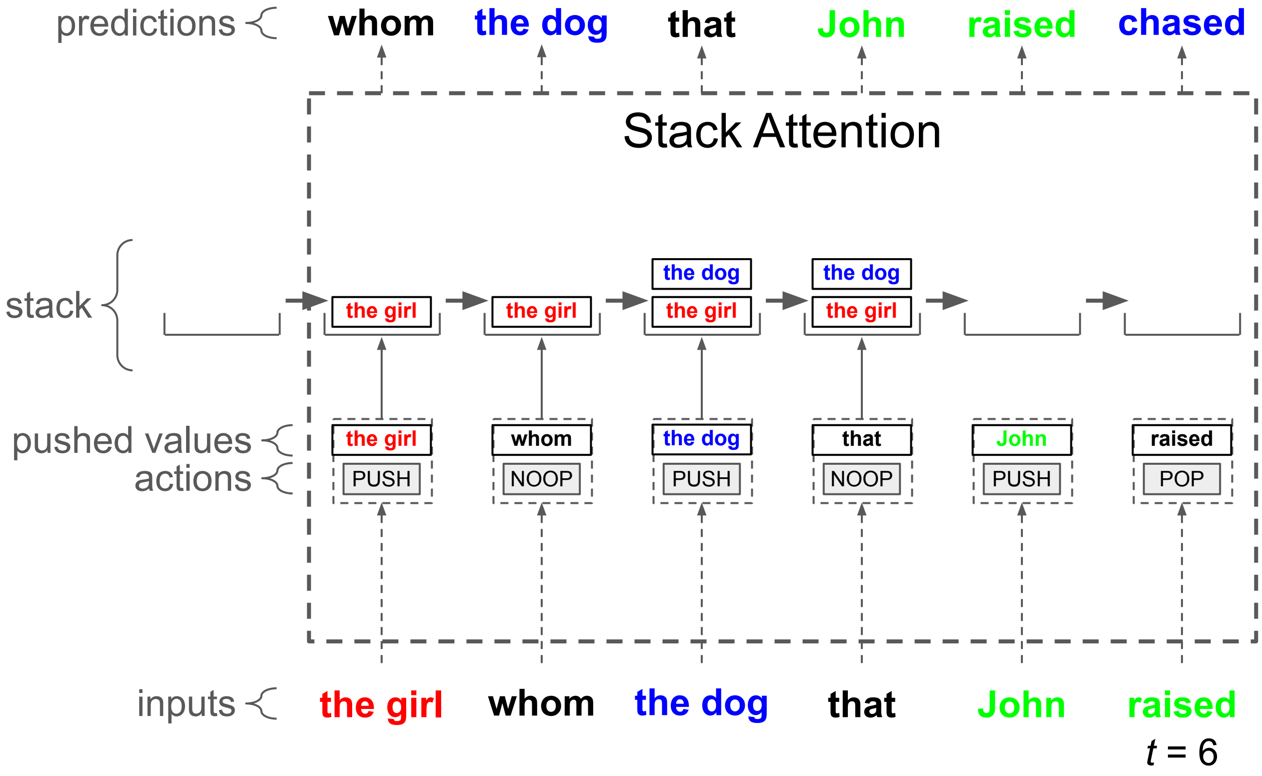
Our Method: Stack Attention



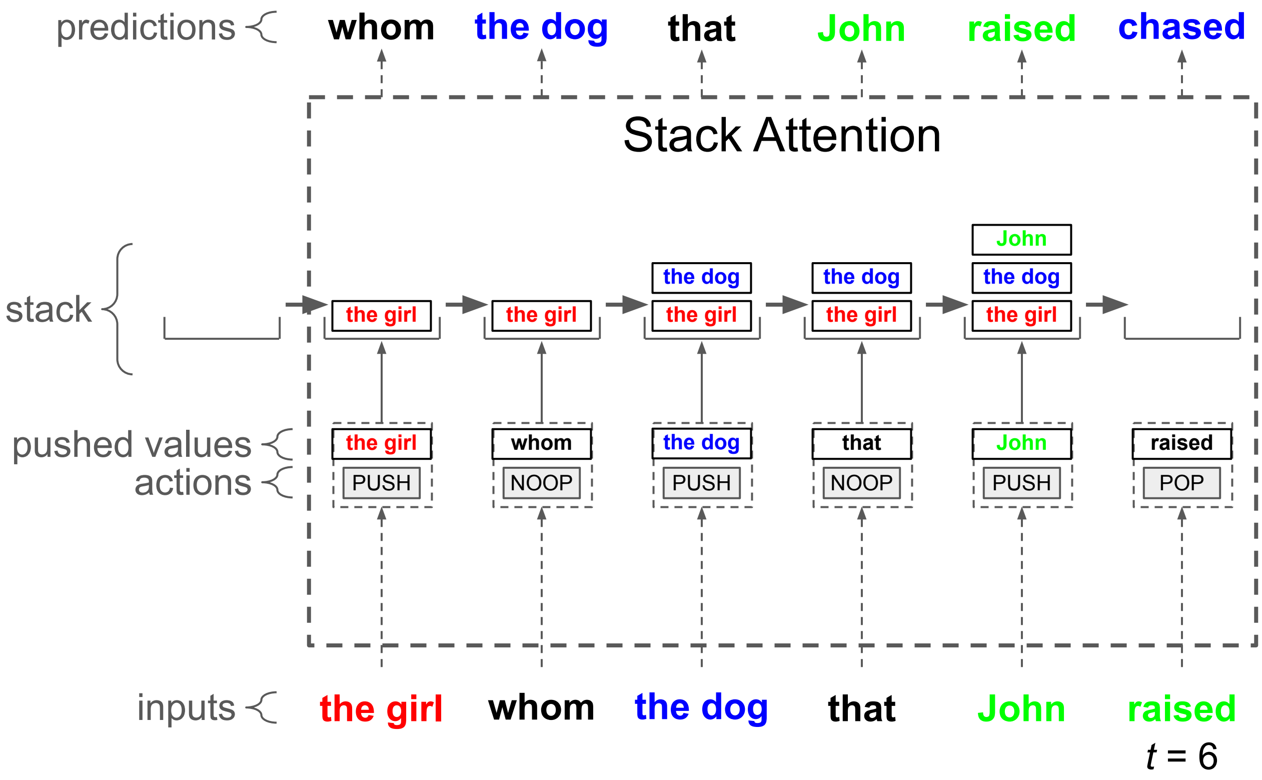
Our Method: Stack Attention



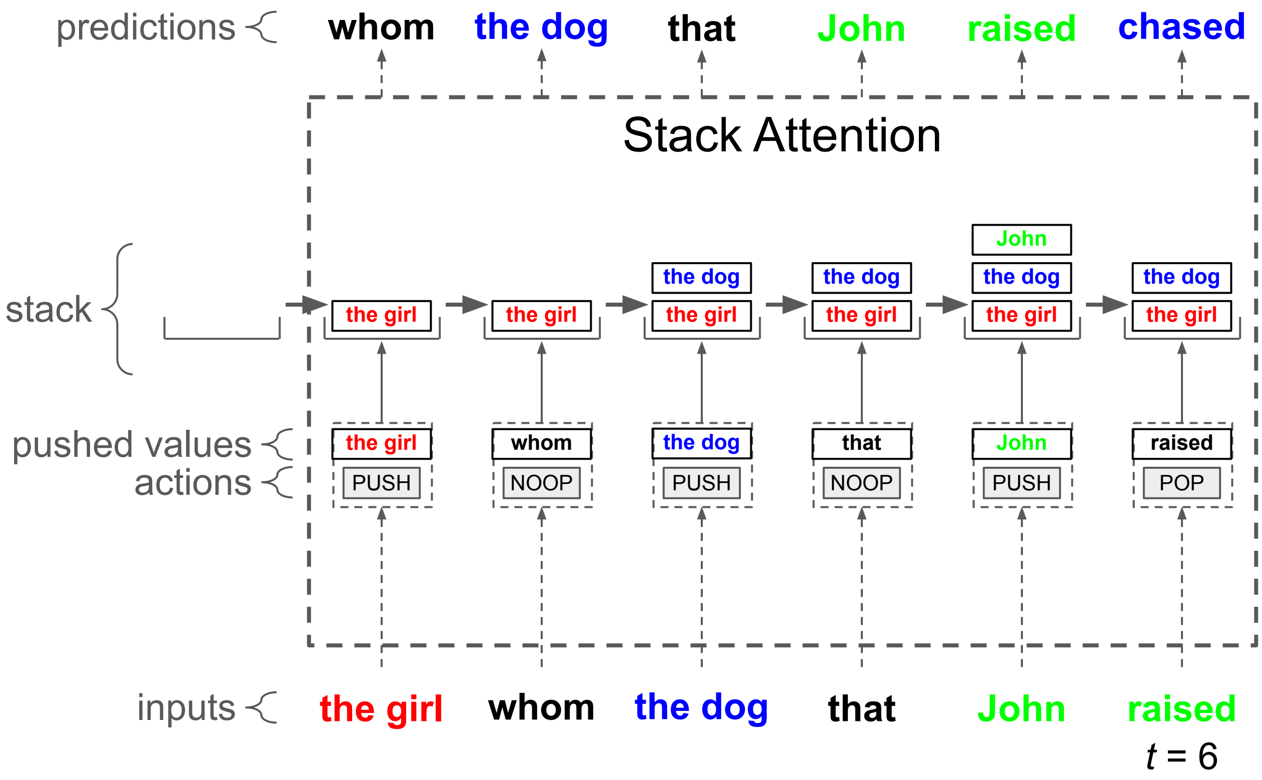
Our Method: Stack Attention



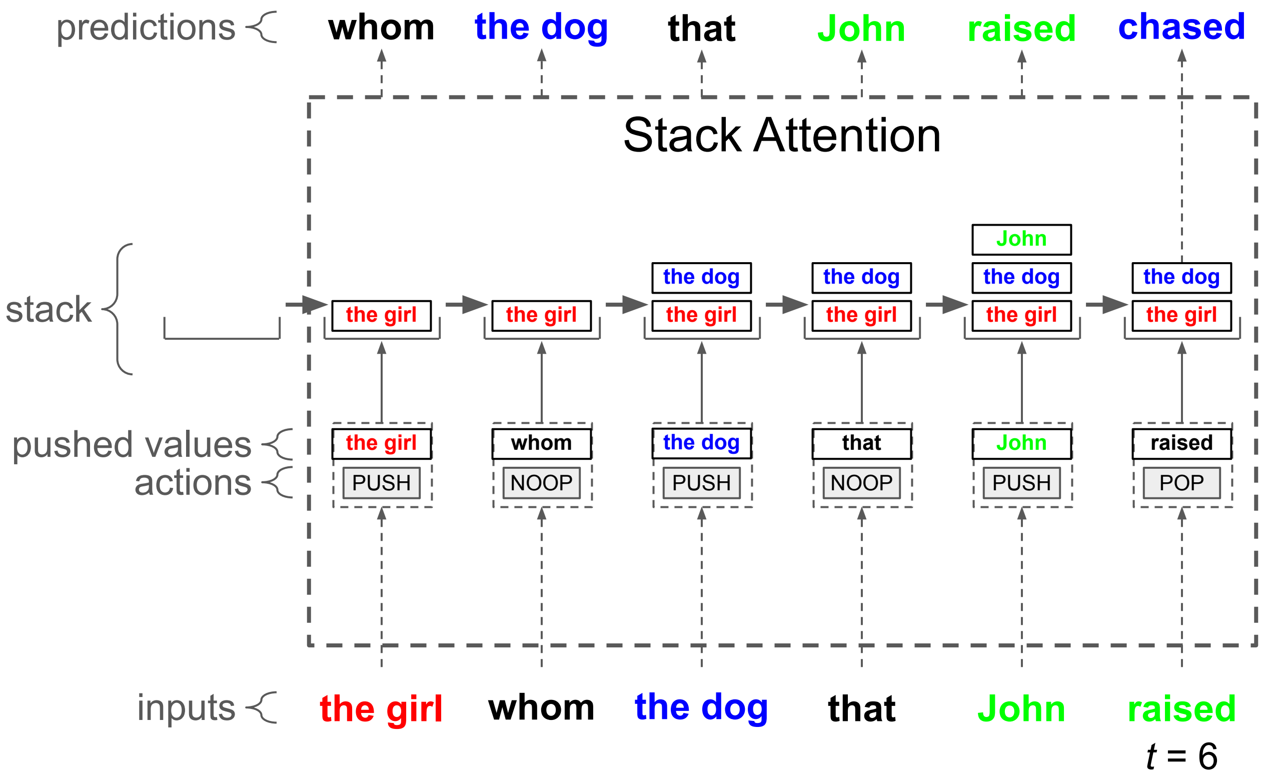
Our Method: Stack Attention



Our Method: Stack Attention



Our Method: Stack Attention



Benefits of This Work

- More expressive transformers
  - Able to recognize all context-free languages with no extra timesteps
- Better language modeling
  - Natural language
  - Context-free languages

Answering Two Questions

Multi-Head
Scaled Dot-Product Attention

RNNs + Differentiable Stacks
(Prior Work)

How do we improve
attention for syntax?

How do we connect stacks
to transformers?

Stack Attention
(This Work)

- Differentiable stack = attention over partial syntax trees
- Swap standard attention with a differentiable stack

Desiderata

1. No syntactic supervision required
2. Generative (unidirectional, not bidirectional)

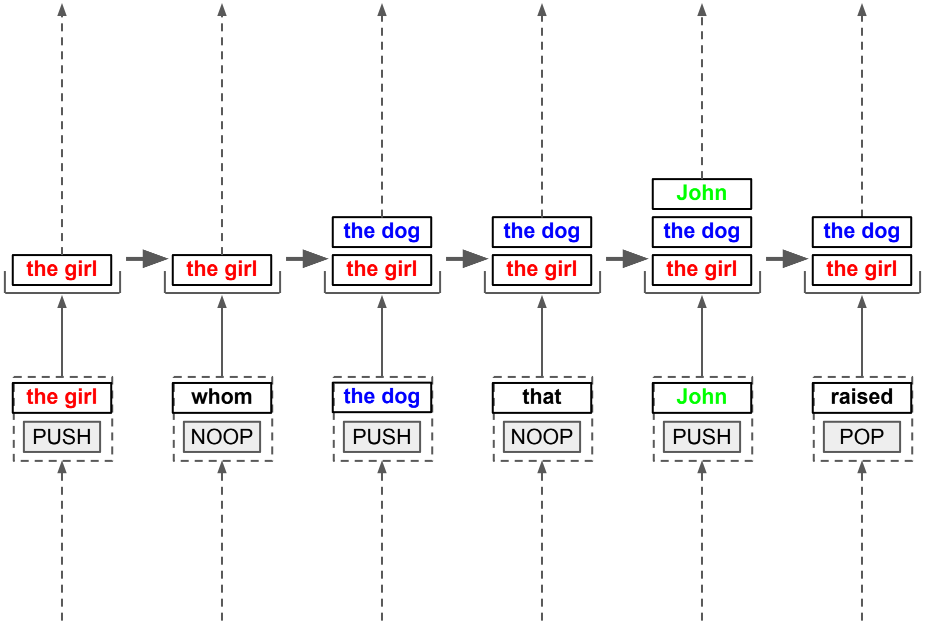
Serves as a drop-in replacement for standard transformers.

Prior Work on Syntax-Oriented Transformers

- Syntactically supervised
  - Positional encodings (Shiv & Quirk, 2019)
  - Attention masking (Deguchi et al., 2019; Zhang et al., 2020; McDonald & Chiang, 2021; Sartran et al., 2022)
  - Multi-task learning (Qian et al., 2021)
  - Stack depth-based attention (Murty et al., 2023)
- Unsupervised but bidirectional
  - Structured Attention: projective dependency trees, encoder-only, used for tree transduction (Kim et al., 2017)
  - Tree Transformer: BERT-style masked language modeling (Wang et al., 2019)
  - R2D2: differentiable CKY, bidirectional language modeling (Hu et al., 2021)

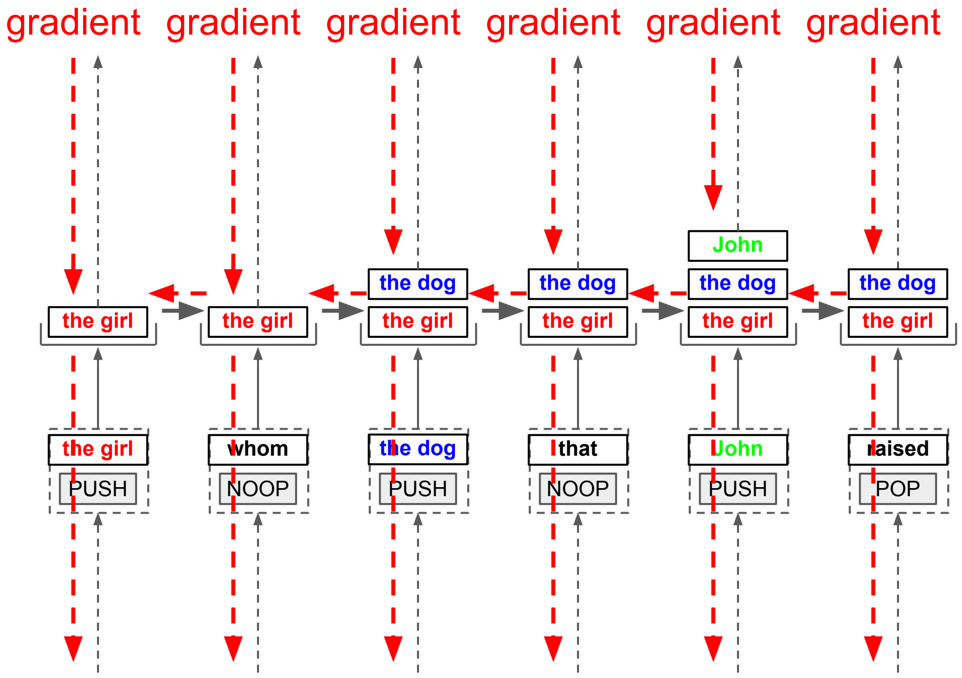
Differentiable Stack

- Continuous function that approximates the behavior of a stack
- Multiple kinds, not just one kind
- **Input:**
  - **Actions:** fractional stack action weights
  - **Pushed vector** associated with push action
- **Output:**
  - **Stack reading:** Approximation of new top stack vector
- Output (stack reading) is differentiable w.r.t. Input (actions and pushed vector)
- Unsupervised!
- Unidirectional!



Differentiable Stack

- Continuous function that approximates the behavior of a stack
- Multiple kinds, not just one kind
- **Input:**
  - **Actions:** fractional stack action weights
  - **Pushed vector** associated with push action
- **Output:**
  - **Stack reading:** Approximation of new top stack vector
- Output (stack reading) is differentiable w.r.t. Input (actions and pushed vector)
- Unsupervised!
- Unidirectional!



Differentiable Stacks

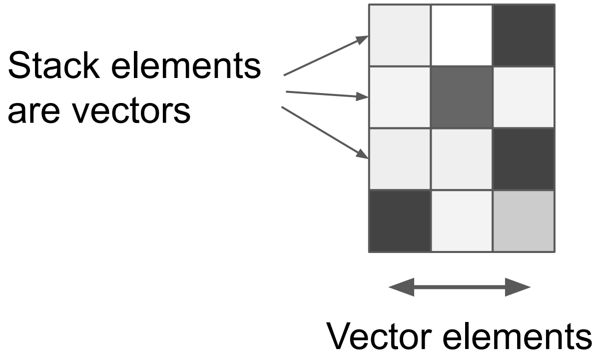
Two varieties used in this paper:

- **Superposition stack** (Joulin & Mikolov, 2015)
  - Computationally cheaper
  - Less expressive
- **Nondeterministic stack** (DuSell & Chiang, 2023)
  - Computationally more expensive
  - Able to recognize **all CFLs**

Two varieties of stack attention:

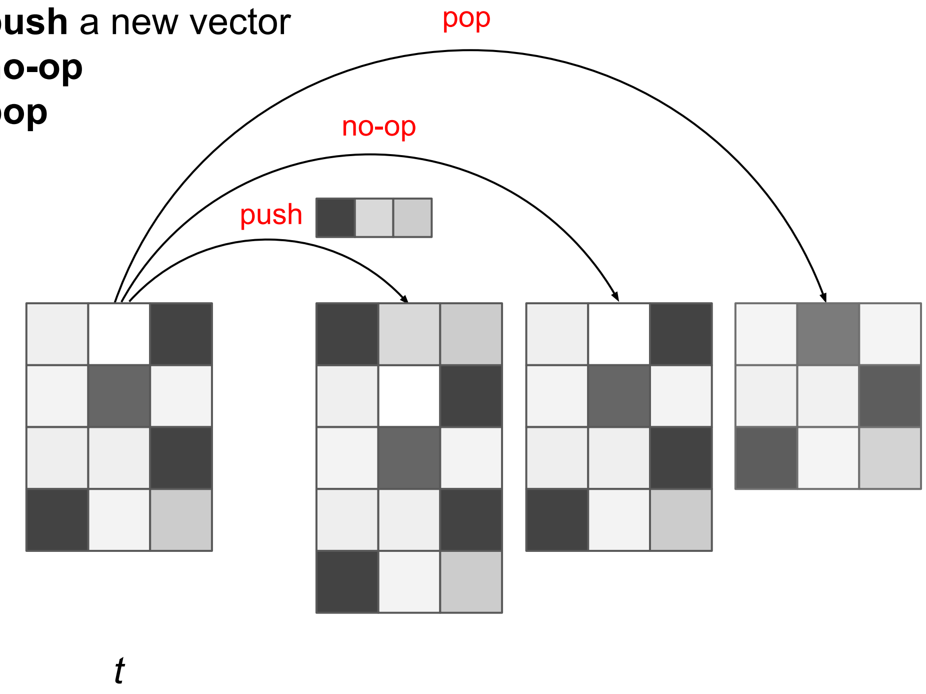
- **Superposition stack attention**
- **Nondeterministic stack attention**

Superposition Stack (Joulin & Mikolov, 2015)



Superposition Stack (Joulin & Mikolov, 2015)

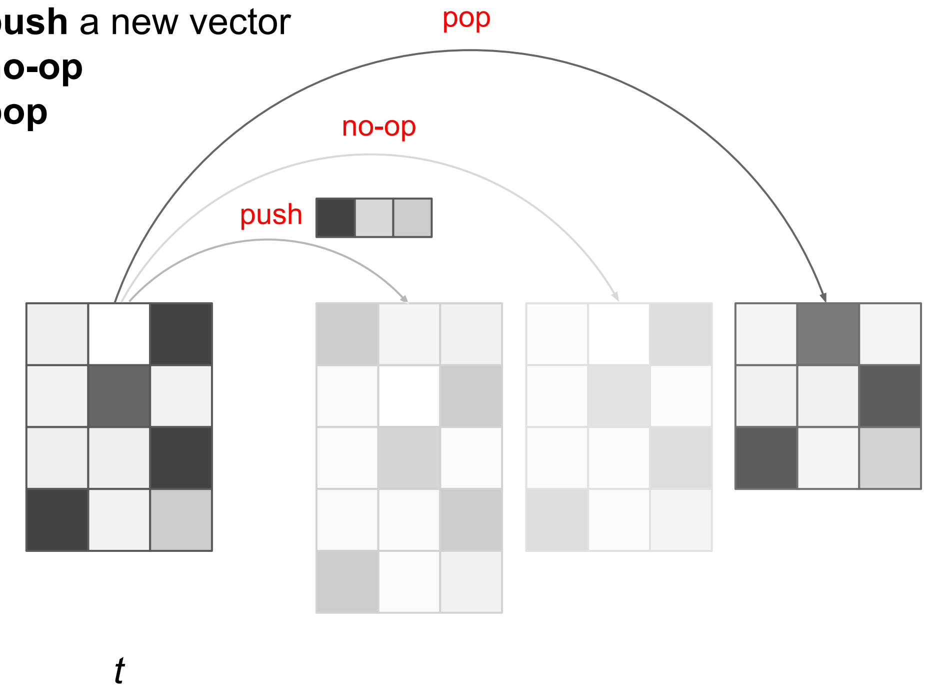
Actions:
Probability distribution over
- **push** a new vector
- **no-op**
- **pop**



Superposition Stack (Joulin & Mikolov, 2015)

Actions:
Probability distribution over

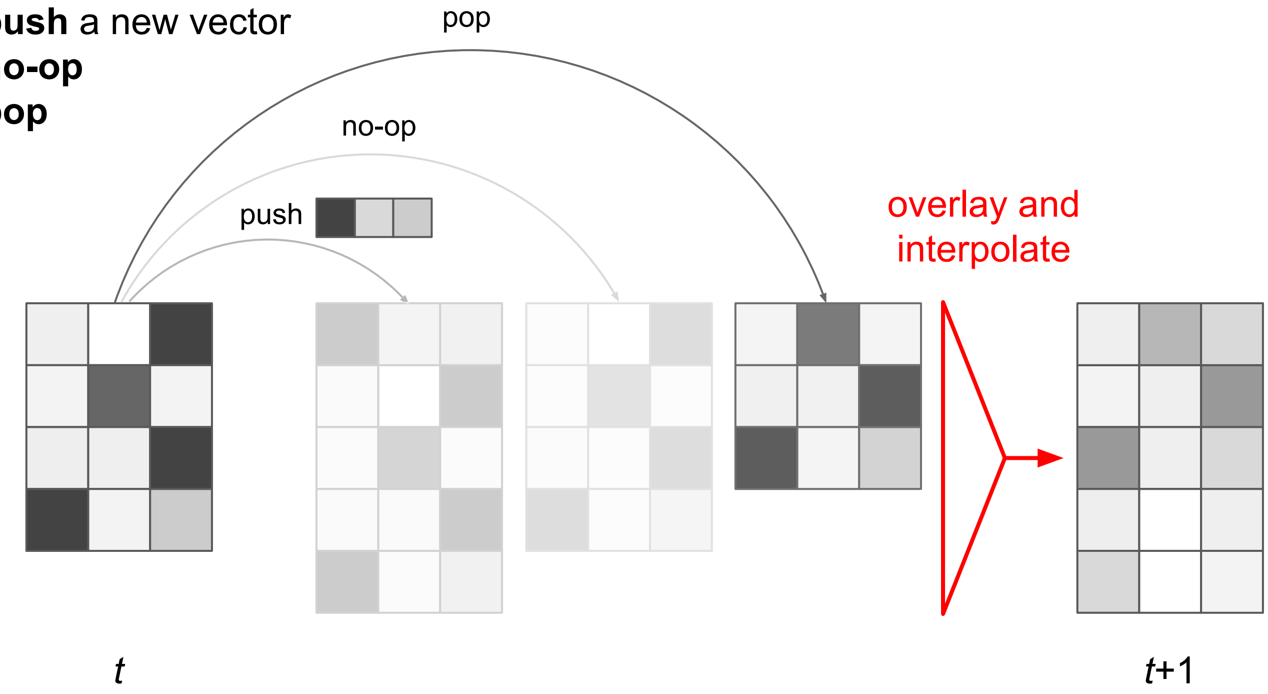
- **push** a new vector
- **no-op**
- **pop**



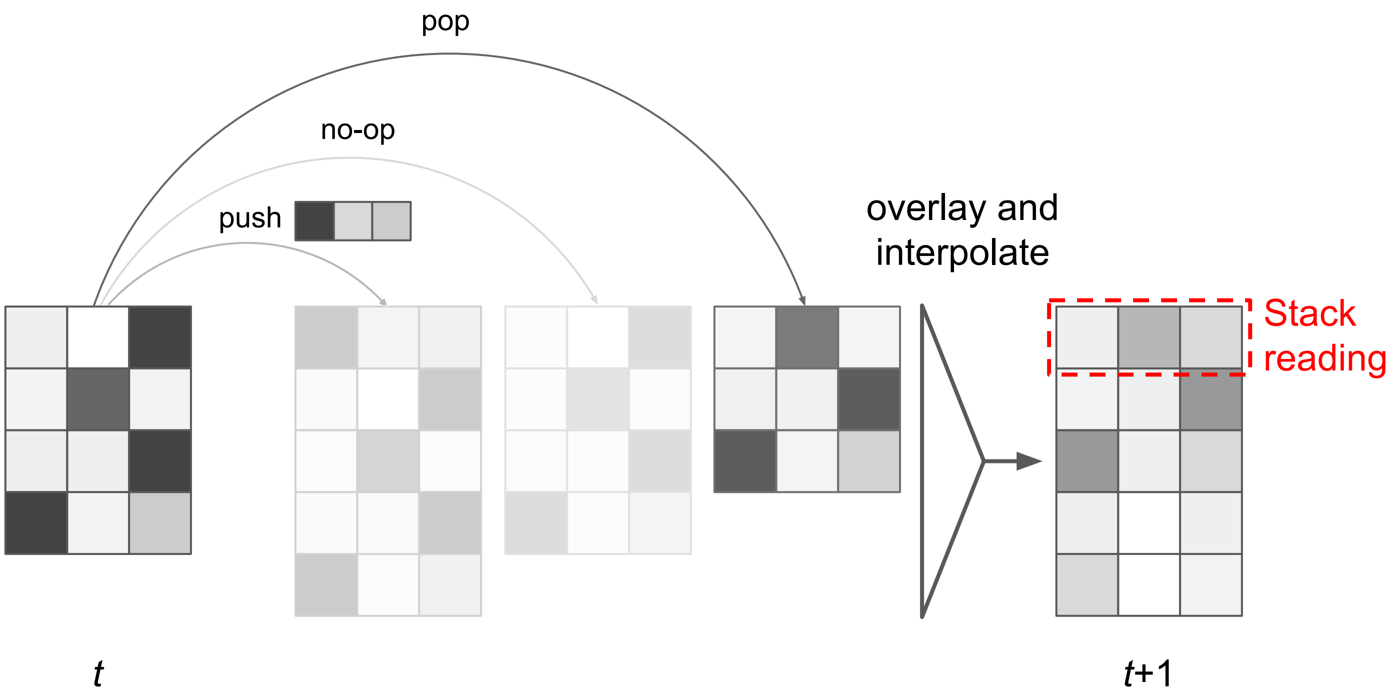
Superposition Stack (Joulin & Mikolov, 2015)

Actions:
Probability distribution over

- **push** a new vector
- **no-op**
- **pop**



Superposition Stack (Joulin & Mikolov, 2015)

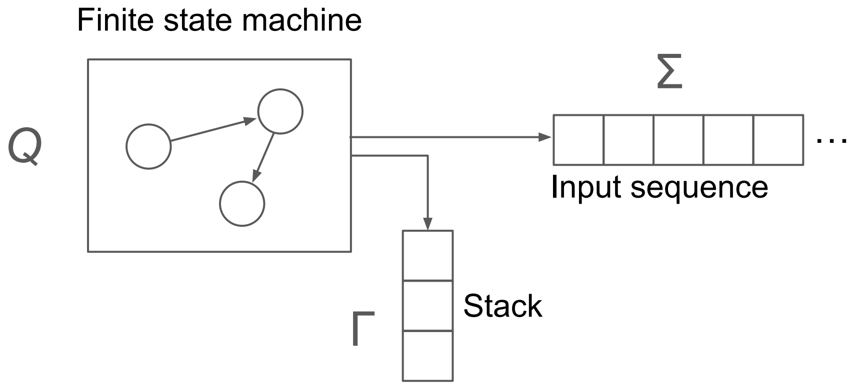


Nondeterministic Stack (DuSell & Chiang, 2023)

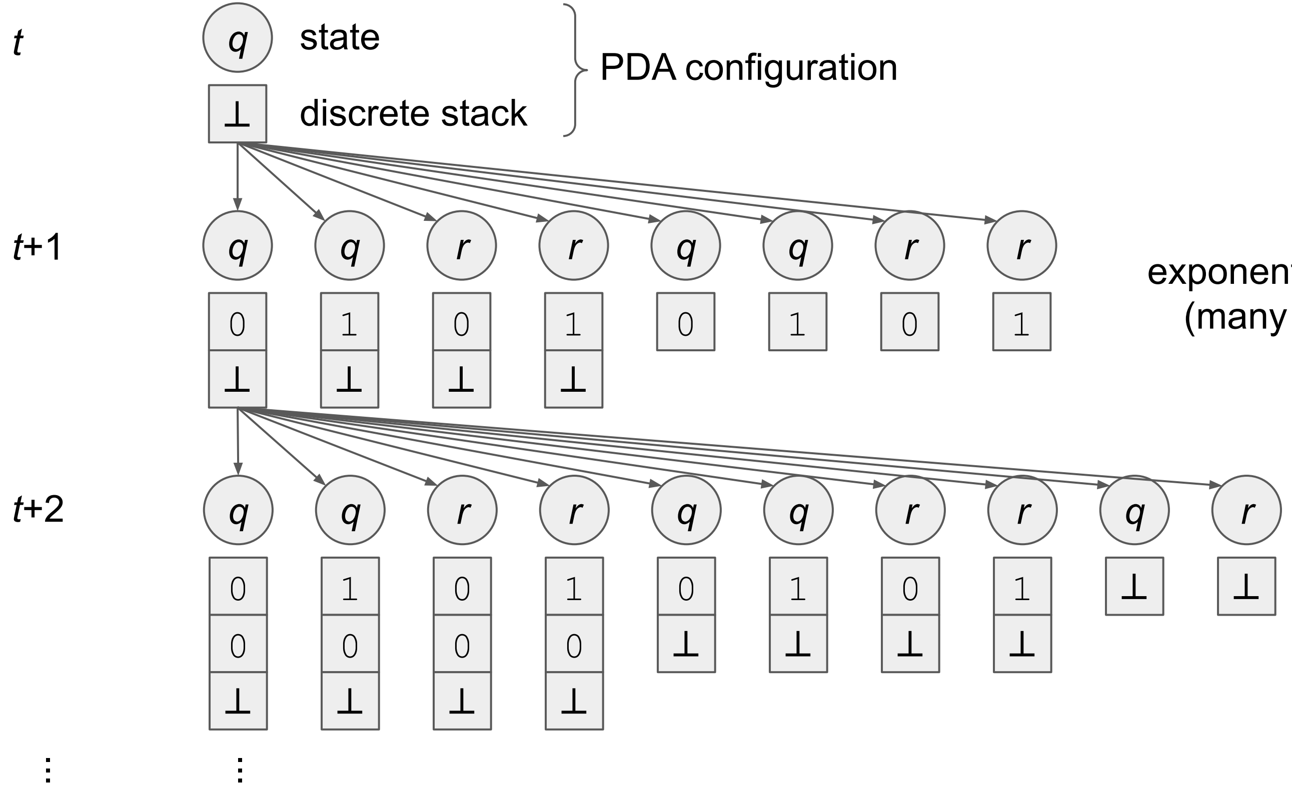
- Simulates a **nondeterministic pushdown automaton (PDA)**
- Uses an extension of PDA called the **Vector PDA (VPDA)**

Pushdown Automaton (PDA)

- Q: finite set of states
- Σ: finite alphabet of input symbols
- Γ: finite alphabet of stack symbols

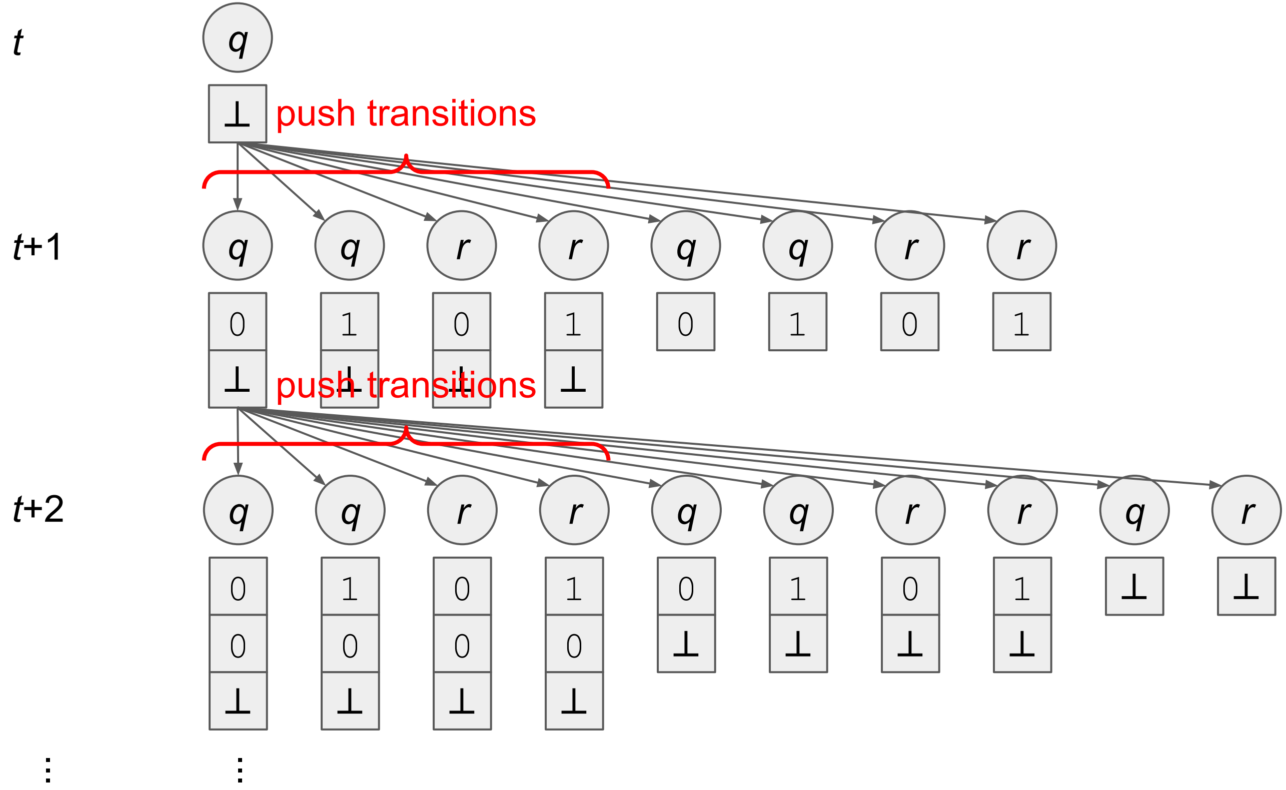


Nondeterministic PDA



exponential number of configurations
(many configurations not shown)

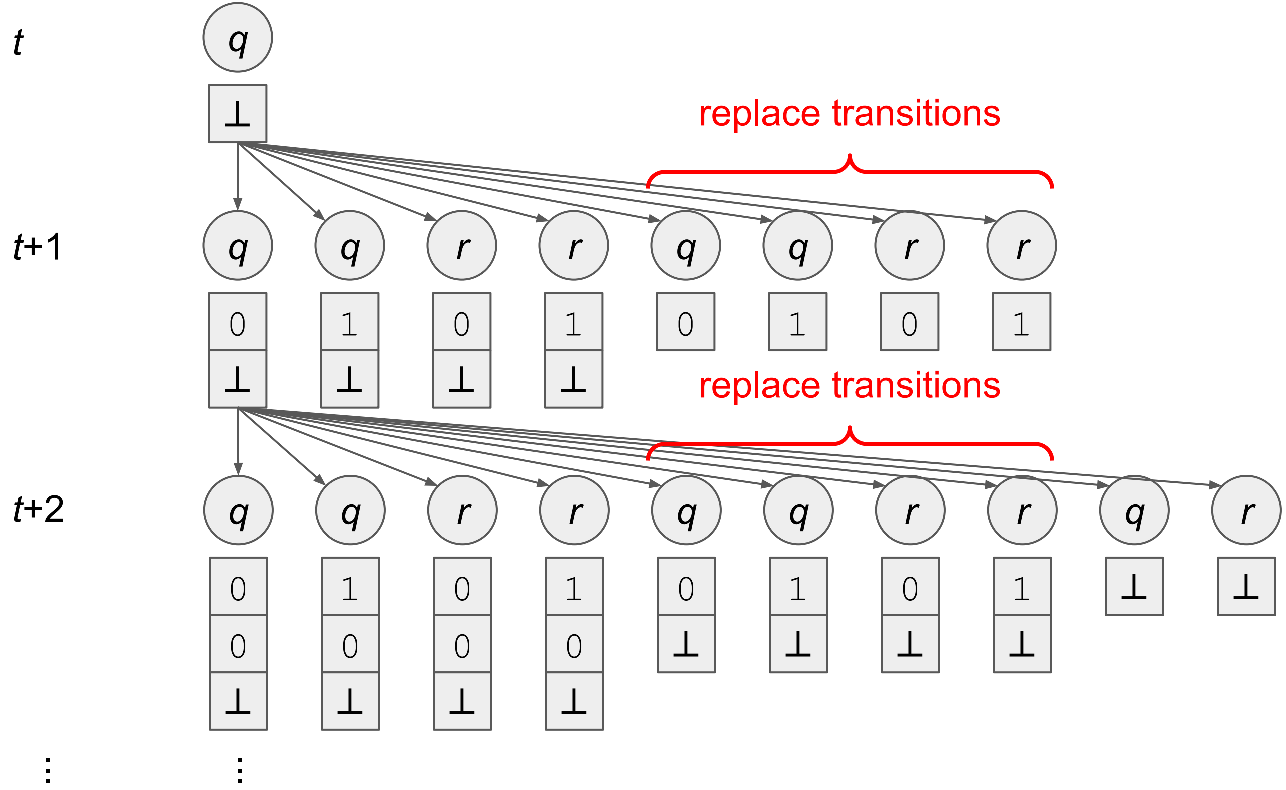
Nondeterministic PDA



<u>Transition types</u>

- **push** *y* on top of *x*
- **replace** *x* with *y*
- **pop** *x*

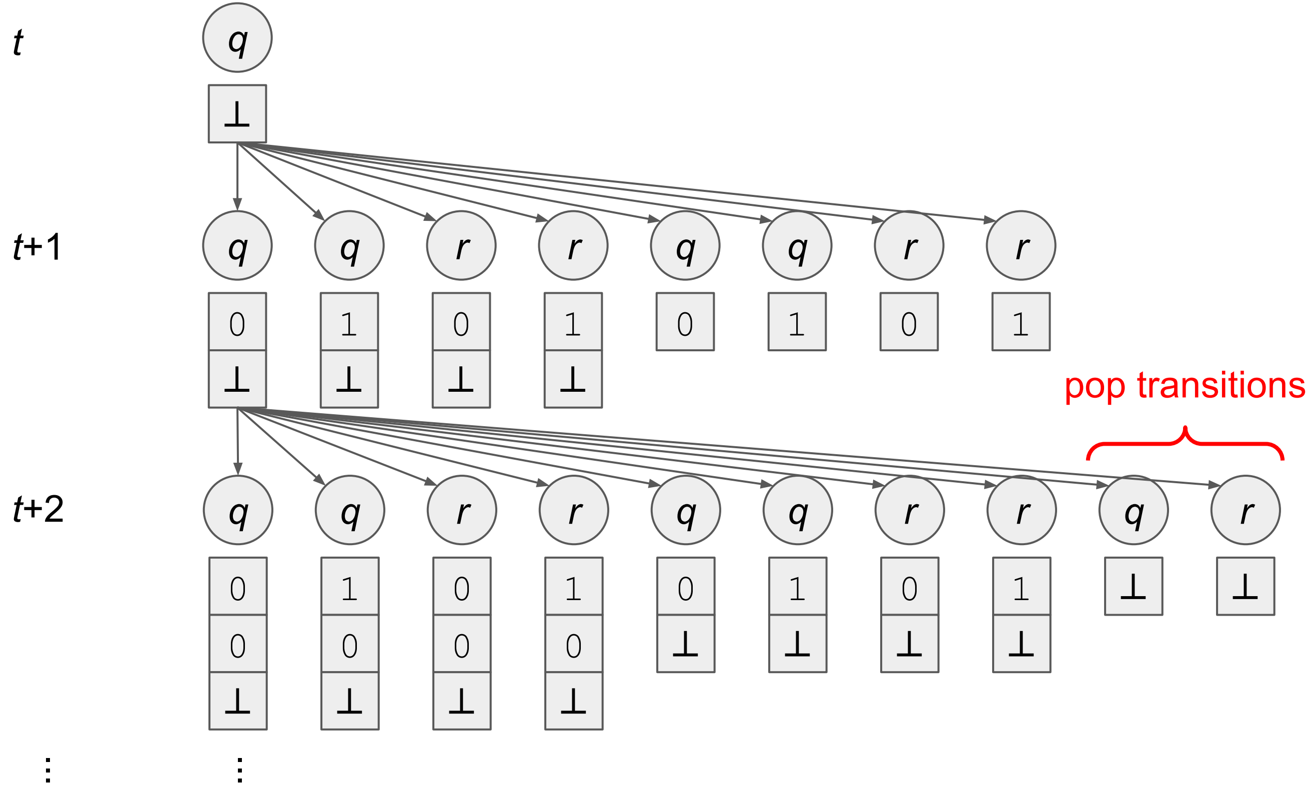
Nondeterministic PDA



<u>Transition types</u>

- **push** *y* on top of *x*
- **replace** *x* with *y*
- **pop** *x*

Nondeterministic PDA



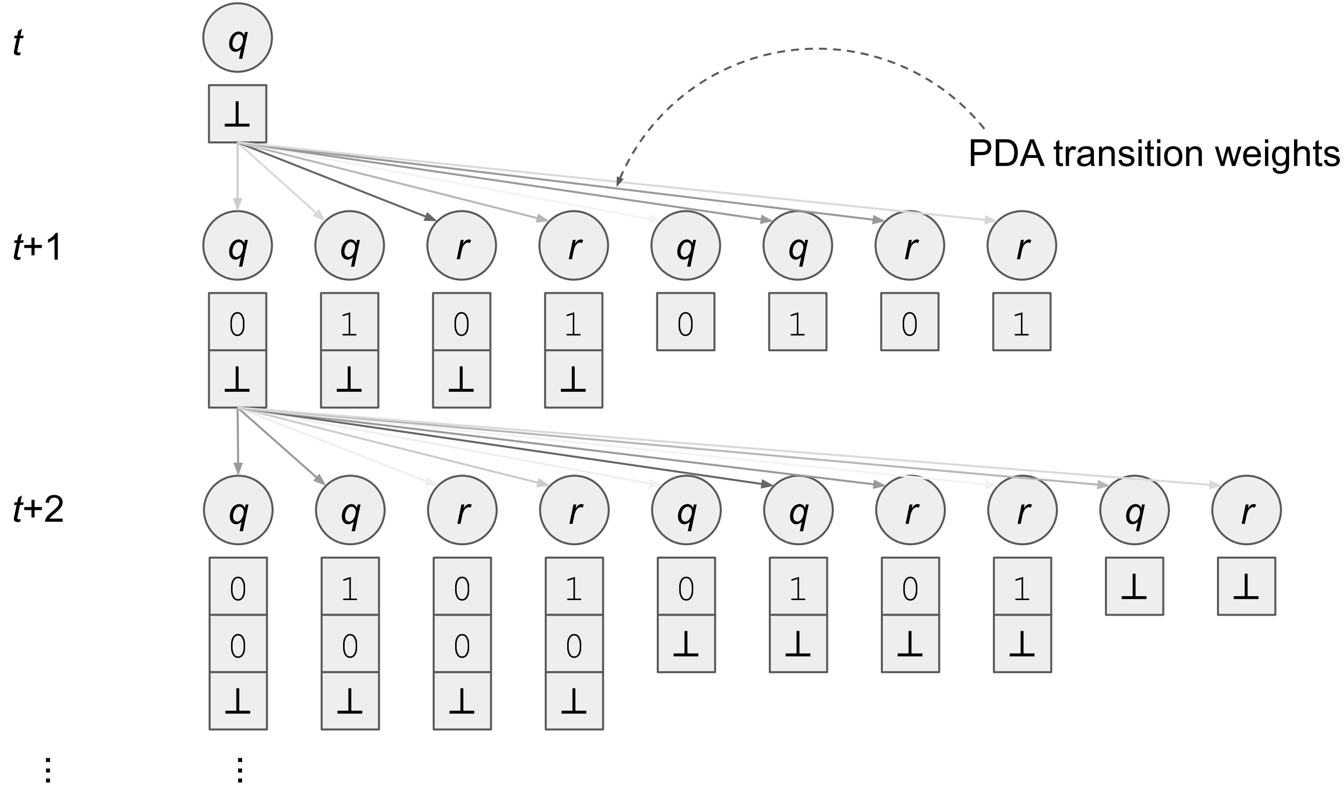
<u>Transition types</u>

- **push** y on top of x
- **replace** x with y
- **pop** x

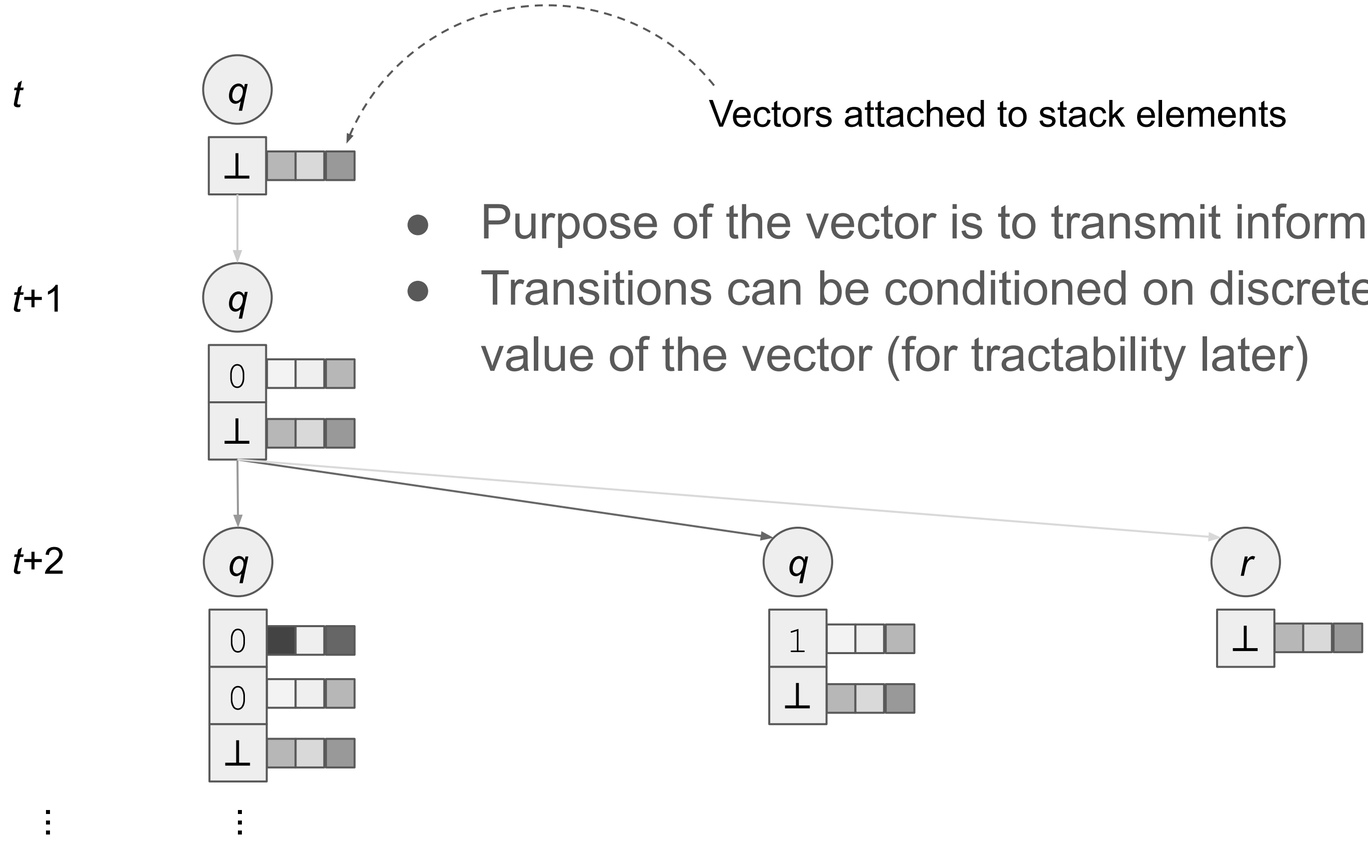
Weighted Pushdown Automaton (WPDA)

- Adds a non-negative **weight** to each transition
- The weight of a run is the product of its transition weights

Weighted PDA (WPDA)

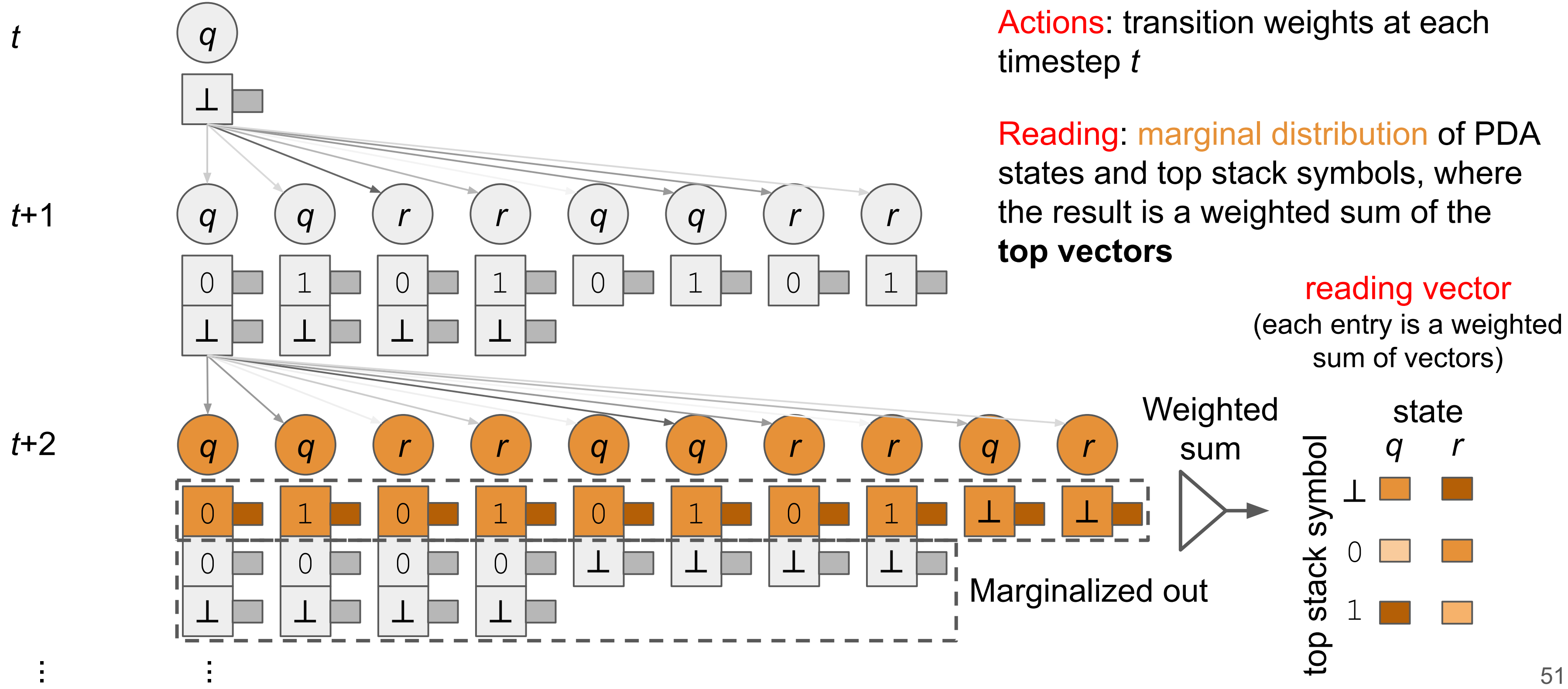


Vector PDA (VPDA)



- Purpose of the vector is to transmit information efficiently
- Transitions can be conditioned on discrete symbol but not the value of the vector (for tractability later)

Differentiable VPDA (dVPDA), aka Nondeterministic Stack



Actions: transition weights at each timestep t

Reading: marginal distribution of PDA states and top stack symbols, where the result is a weighted sum of the **top vectors**

Differentiable VPDA (dVPDA), aka Nondeterministic Stack

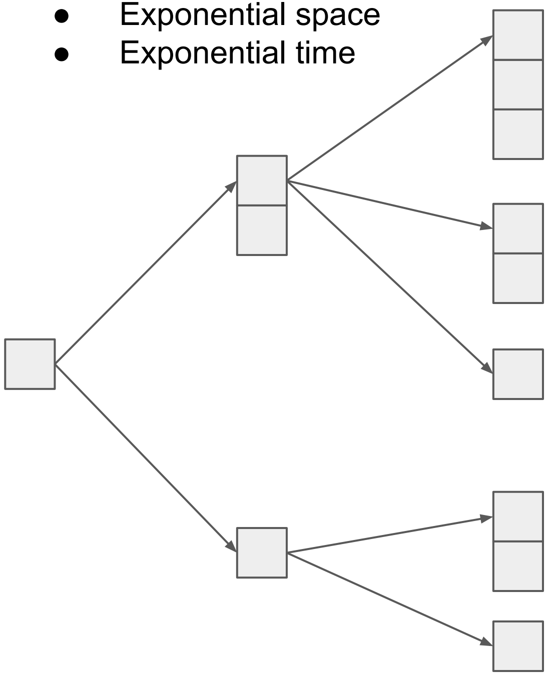
- t: current timestep
- $r \in Q$: PDA state
- $y \in \Gamma$: discrete stack symbol type

$$\text{reading}(t, r, y) = \frac{\sum_{\text{run } \pi \text{ ending in } t, r, y} \text{weight}(\pi) \cdot \textbf{top-vector}(\pi)}{\sum_{\text{run } \pi \text{ ending in } t} \text{weight}(\pi)}$$

Efficient Nondeterminism Using Lang's Algorithm (1974)

Nondeterministic Branches of Computation

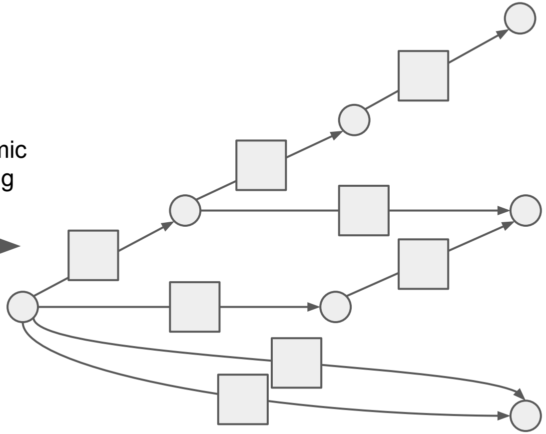
- Exponential space
- Exponential time



Lang's dynamic programming algorithm

Weighted Directed Graph

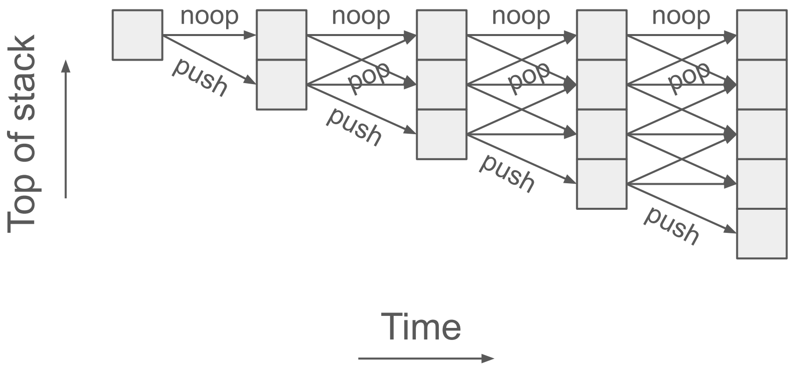
- Cubic time
- Quadratic space



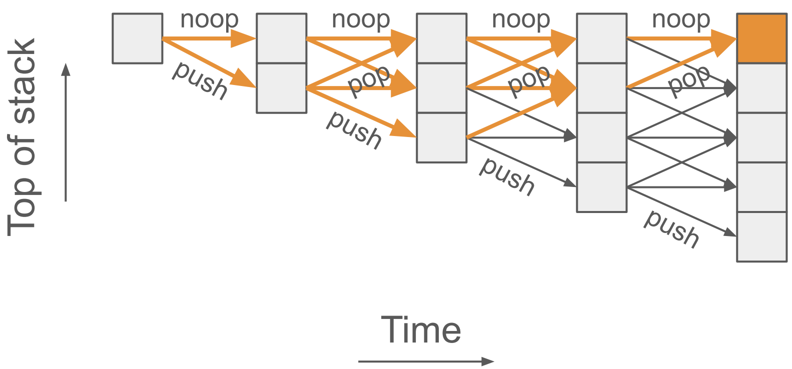
Nondeterministic Stack Attention Recognizes All CFLs

- PDAs in our normal form recognize all CFLs (DuSell & Chiang, 2023)
- VPDAs are a generalization of normal-form PDAs

Superposition Is a Special Case of Nondeterminism



Superposition Is a Special Case of Nondeterminism



noop × noop × noop × noop +
noop × noop × push × pop +
noop × push × pop × noop +
noop × push × noop × pop +
...

Differentiable Stacks as Attention

Scaled Dot-Product Attention

$$\text{output}(t) = \frac{1}{Z} \sum_{i=1}^{t} \text{weight}(i) \cdot \textbf{value}(i)$$

$$Z = \sum_{i=1}^{t} \text{weight}(i)$$

Differentiable Stack, aka Stack Attention

$$\text{reading}(t, r, y) = \frac{1}{Z} \sum_{\substack{\text{run } \pi \\ \text{ending in } t, r, y}} \text{weight}(\pi) \cdot \textbf{value}(\pi)$$

$$Z = \sum_{\substack{\text{run } \pi \\ \text{ending in } t}} \text{weight}(\pi)$$

Serial Time Complexity

Attention	Serial Time
SDPA	$O(n^2)$
Superposition	$O(n^2)$
Nondeterministic	$O(n^3)$

Parallel Time Complexity

Attention	Implemented	Parallel CKY	Theoretical
SDPA	$O(\log n)$	–	–
Superposition	$O(n)$	–	$O((\log n)^2)$
Nondeterministic	$O(n^2)$	$O(n \log n)$	$O((\log n)^2)$

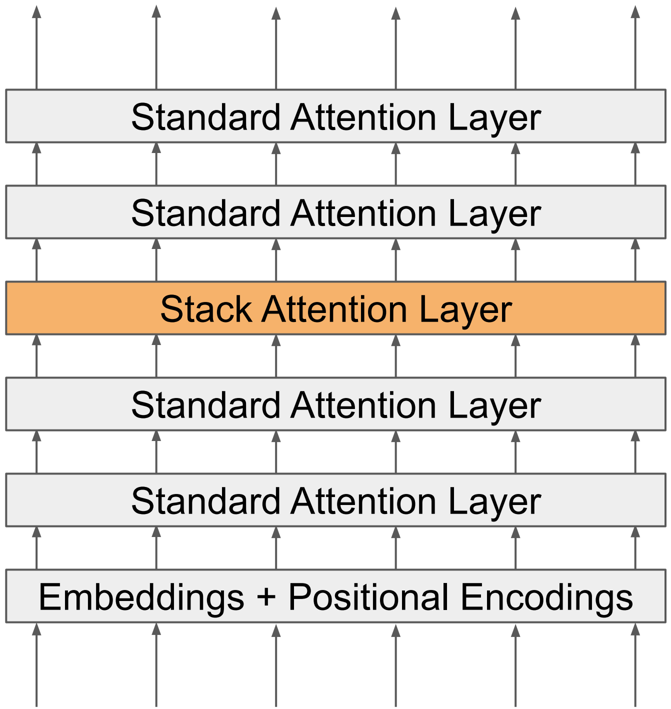
Wall-Clock Runtimes

Computational cost on a natural language modeling task

Model	Examples/s	Minutes/Epoch	GPU Memory
Tf	859	0.8	394 MB
Tf+Sup	345	1.9	397 MB
Tf+Nd	27	24.3	1.91 GB

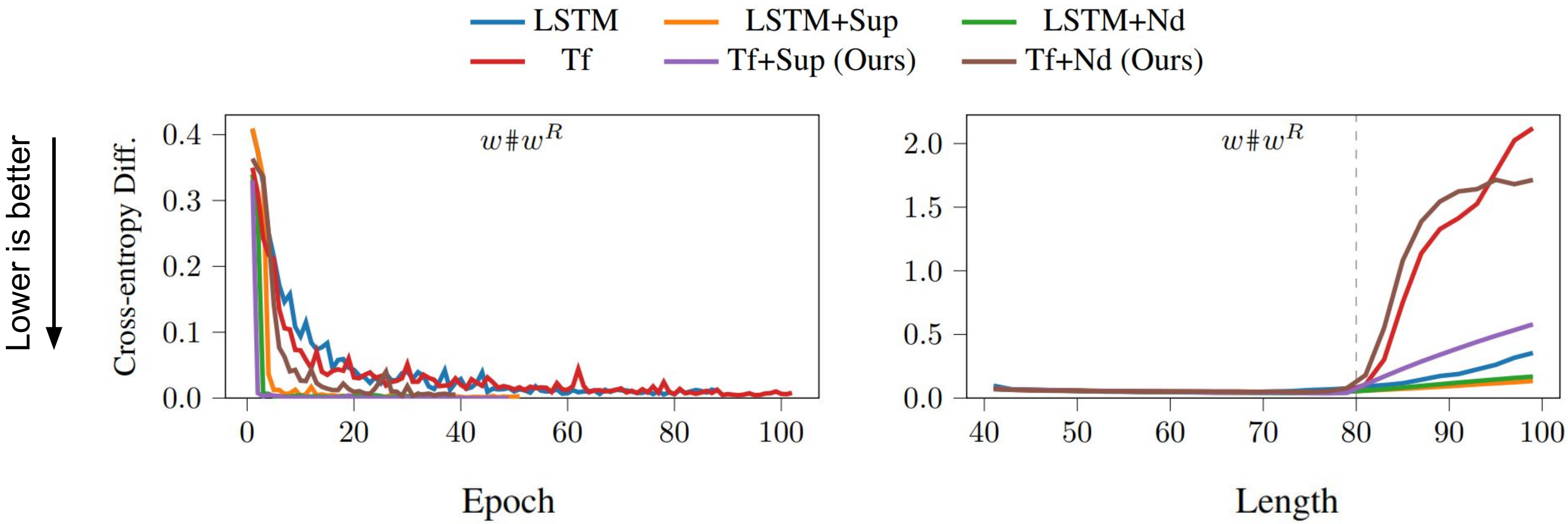
Language Modeling on Context-Free Languages

- All transformers have 5 layers
- Stack attention replaces scaled dot-product attention in the third (middle) layer
- 6 architectures:
  { Transformer, LSTM } ×
  { no stack, superposition, nondeterministic }

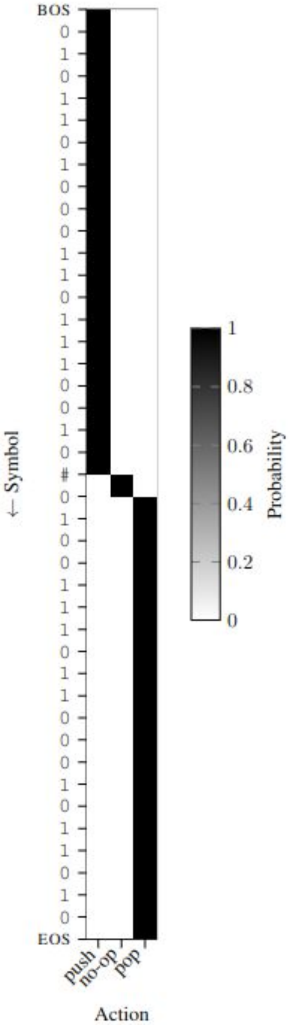


Marked Reversal $w\#w^R$

1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 # 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1
0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 # 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0
0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 # 0 1 0 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0



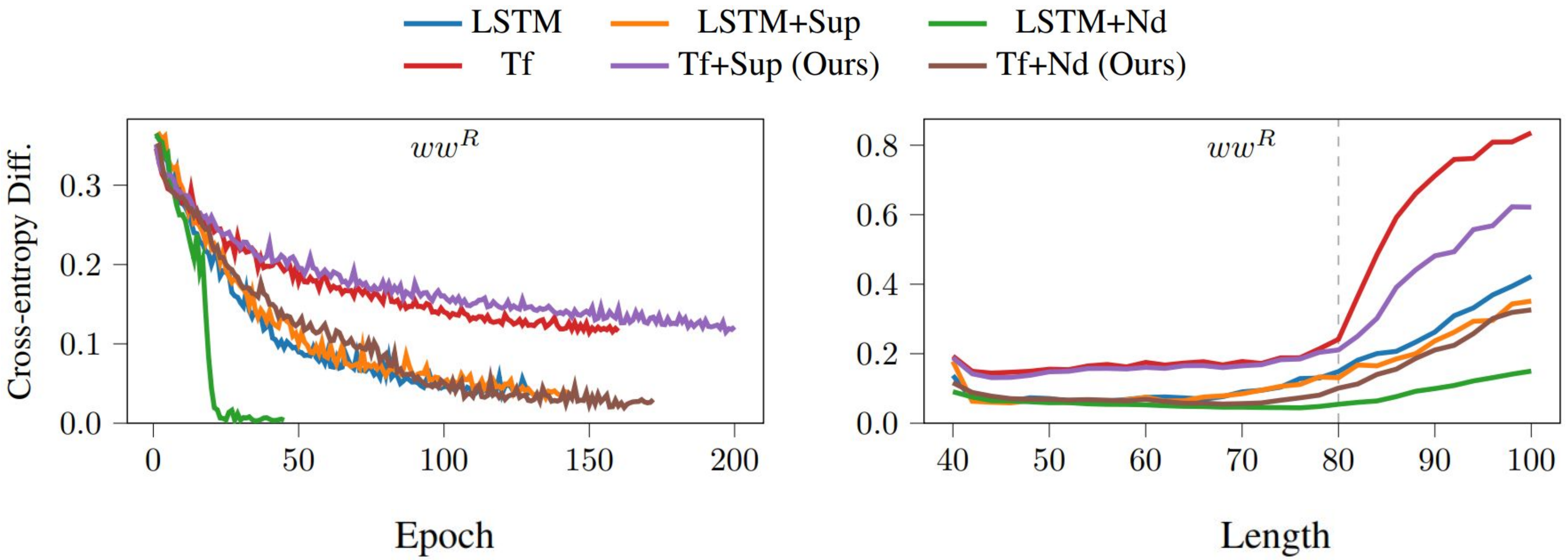
Learned Stack Actions



Unmarked Reversal ww^R

```
1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1
0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0
0 1 0 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0
```

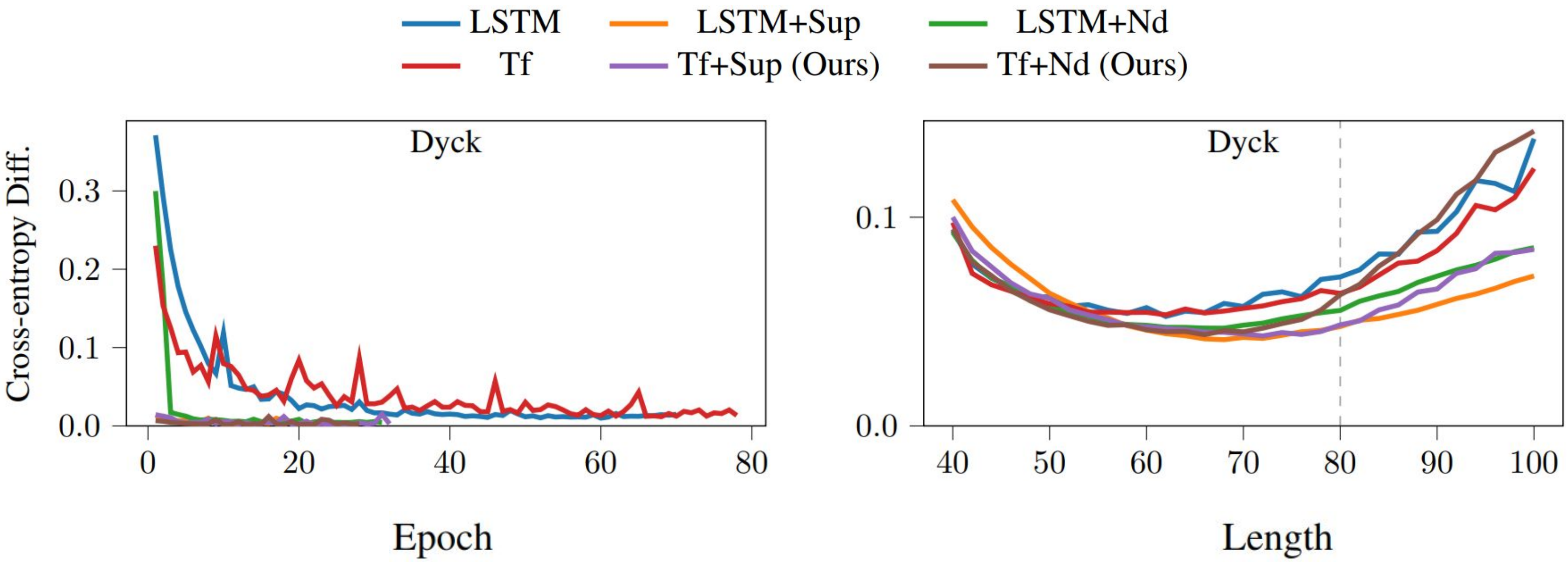
Lower is better



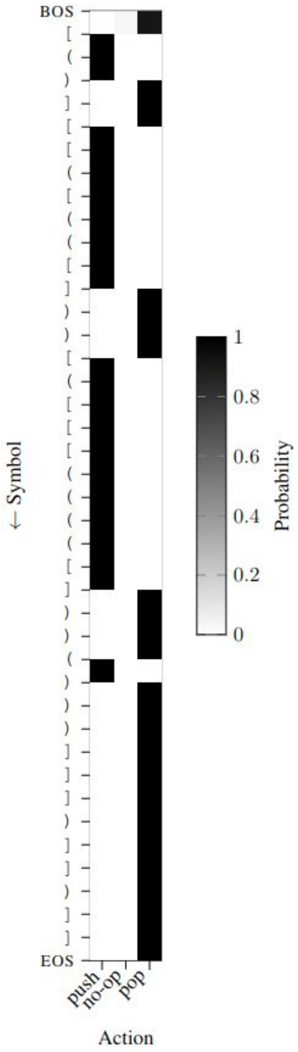
Dyck (Balanced Brackets)

([[[([[((( ) [[[((( )))]]]))]])]] ( )]) ((( )))
[[([( ) [(( ))]])]] [((([[([(([[ ]]))])]])))]
((([[(((((( ) ((([[ ]]))))))))]]))) ([([ ])])

Lower is better



Learned Stack Actions



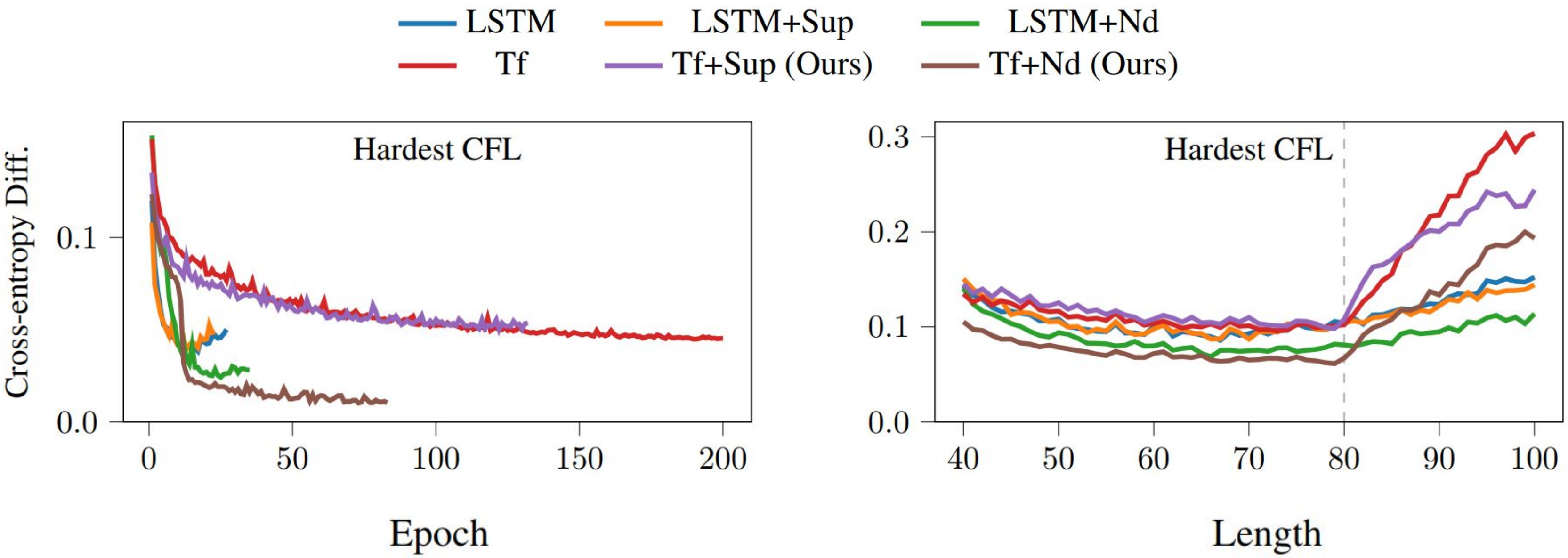
Hardest CFL (Greibach, 1973)

```
$,$,),[$$$,(,[;,[,(;[,],;,)($$[,[([],;,[,)],(),[;
,$,[);,[],;,[$)],[[(,;,)[,)]],(,)]],$[$[,],;
,$,;,[((([[]],[;$,),;,[,;,(],(),[,;,]$(],][]))],);
[,$(,][,(]$))]],$[(())(([()],$(,$]$;$(,)(,;],)),;,],(,],;
((,$[](,$);],[,(;,(,)[$)[,;,())],;,[],),;
,$,],;,$,]],[,$($(,;[)(,$[,(),(;$,],)]),;
[)(,$(())(()(,;],)(,$,$,];$,[]))(,(;(,[[,[,],;,]),[;
,$[[((,))[](]]$,;]$,[,;$((,$,)[,],$,);,))][]],]$$;
,$[,;,([(,;$,[(,[$,;)),)]()()),]),]$],;,])([]),$,;,][],;
,$([],;,[(([]),[;,[],;[,)[],;,])[[,;],]],([,$[,;
```

Hardest CFL

```
$,$,),[$$$,(,[;,[,(;[,],;,)($$[,[([],;,[,)],(),[;
,$,[);,[],;,[$)],[[(,;,)[,)]],(,)]],$[$[,],;
,$,;,[((([[]],[;$,),;,[,;,(],(),[,;,]$(],][])))],);
```

Lower is better

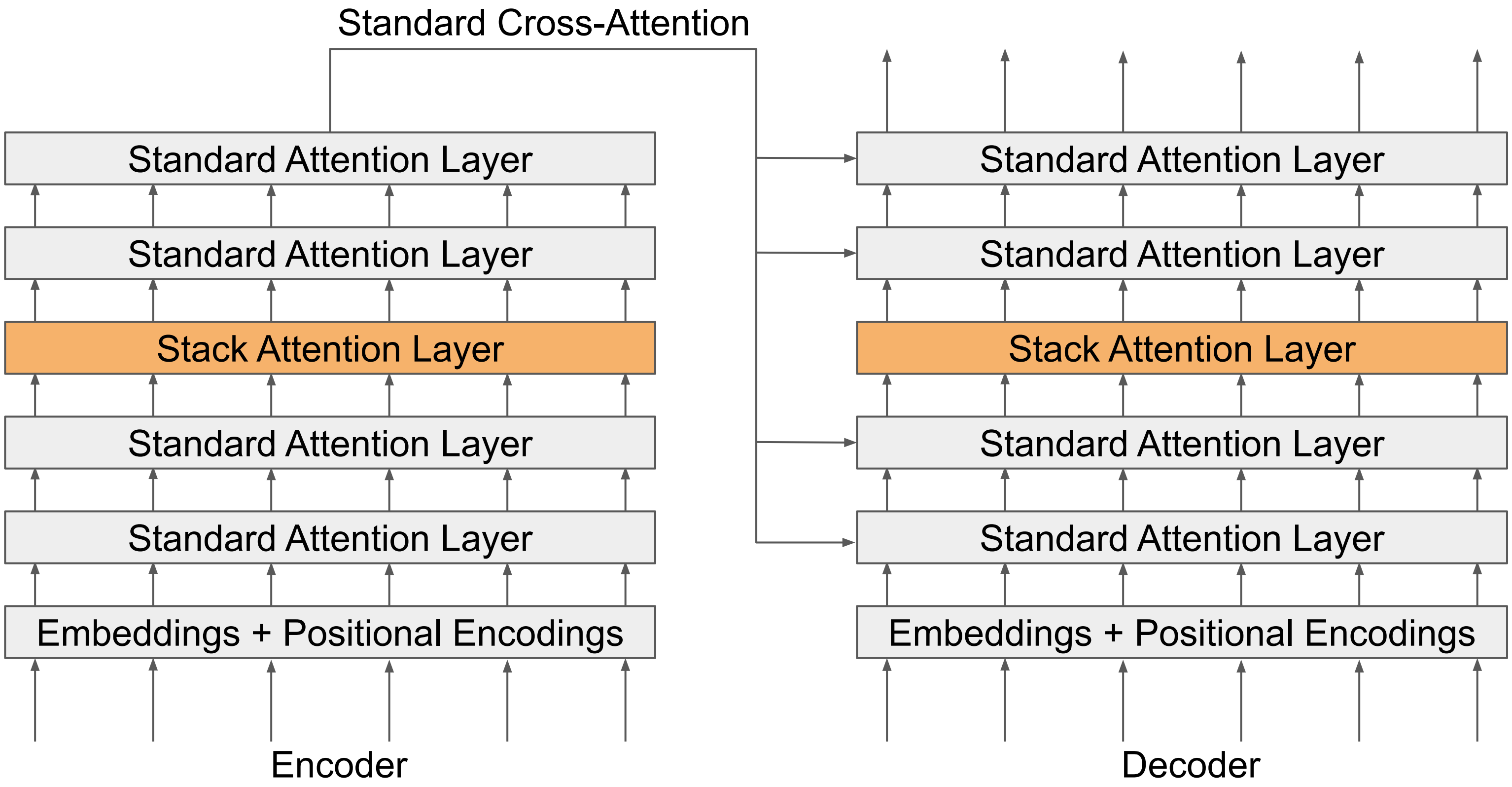


Language Modeling on Natural Language

- Perplexity on Penn Treebank
- Each result is best of 20 runs

Model	Params.	Val. ↓	Test ↓
Tf	10,051,072	115.11	92.84
Tf+Sup (Ours)	10,050,304	122.94	98.67
Tf+Nd (Ours)	9,861,898	**110.59**	**88.54**

Machine Translation



Machine Translation

- 100k training samples from German-English Europarl v7
- Limited to 150 characters each on source and target side

Model	d_{model}	Params.	Val. Perp. ↓	Test BLEU ↑
Tf	160	4,595,200	12.52	**12.21**
Tf+Sup (Ours)	160	4,492,480	**11.94**	12.03
Tf+Nd (Ours)	160	4,465,610	13.00	11.86
Tf	240	9,580,800	12.54	12.11
Tf+Sup (Ours)	240	9,349,920	**11.53**	**12.81**
Tf+Nd (Ours)	240	9,232,810	12.46	11.50
Tf	360	20,419,200	**11.80**	**12.69**
Tf+Sup (Ours)	360	19,900,080	12.03	12.03
Tf+Nd (Ours)	360	19,551,610	12.54	11.74

Summary of Contributions

- New self-attention with latent model of syntax
- Two variants: superposition and nondeterministic
- Unsupervised and generative, trainable with standard backprop, can be used wherever transformers are currently used
- Nondeterministic stack attention can recognize all CFLs
- Nondeterministic stack attention performs best on Hardest CFL and natural language modeling despite having the fewest parameters

Future Work

- Speed up nondeterministic stack attention by parallelizing across timestep dimension
- Interpretability for nondeterministic stack attention
- Evaluate data efficiency and hierarchical inductive bias

References

- Hiroyuki Deguchi, Akihiro Tamura, and Takashi Ninomiya. Dependency-based self-attention for transformer NMT. In Proc. RANLP, pp. 239–246, Varna, Bulgaria, September 2019. INCOMA Ltd.
- Brian DuSell and David Chiang. The surprising computational power of nondeterministic stack RNNs. In Proc. ICLR, Kigali, Rwanda, May 2023.
- Sheila A. Greibach. A new normal-form theorem for context-free phrase structure grammars. J. ACM, 12(1):42–52, January 1965.
- Michael Hahn. Theoretical limitations of self-attention in neural sequence models. Trans. ACL, 8: 156–171, January 2020.
- Xiang Hu, Haitao Mi, Zujie Wen, Yafang Wang, Yi Su, Jing Zheng, and Gerard de Melo. R2D2: Recursive Transformer based on Differentiable Tree for Interpretable Hierarchical Language Modeling. In Proc. ACL (Long Papers), pp. 4897–4908, Online, 2021. Association for Computational Linguistics.
- Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In Advances in NIPS, volume 28, Montreal, Canada, December 2015. Curran Associates, Inc.
- Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In Proc. ICLR, Toulon, France, April 2017.
- Yair Lakretz, Théo Desbordes, Dieuwke Hupkes, and Stanislas Dehaene. Can Transformers Process Recursive Nested Constructions, Like Humans?. In Proc. COLING, pp. 3226–3232, Gyeongju, Republic of Korea, 2022. International Committee on Computational Linguistics.
- Colin McDonald and David Chiang. Syntax-based attention masking for neural machine translation. In Proc. NAACL: Student Research Workshop, pp. 47–52, Online, June 2021. Association for Computational Linguistics.
- Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. Pushdown layers: Encoding recursive structure in transformer language models. In Proc. EMNLP, pp. 3233–3247, Singapore, December 2023. Association for Computational Linguistics.
- Peng Qian, Tahira Naseem, Roger Levy, and Ramon Fernandez Astudillo. Structural guidance for transformer language models. In Proc. ACL-IJCNLP (Long Papers), pp. 3735–3745, Online, August 2021. Association for Computational Linguistics.
- Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. Trans. ACL, 10:1423–1439, December 2022.
- Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. In Advances in NeurIPS, volume 32, Vancouver, Canada, 2019. Curran Associates, Inc.
- Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. Tree transformer: Integrating tree structures into self-attention. In Proc. EMNLP-IJCNLP, pp. 1061–1070, Hong Kong, China, November 2019. Association for Computational Linguistics.
- Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. SG-Net: Syntax-guided machine reading comprehension. In Proc. AAAI, New York, New York, USA, February 2020.

Thank You!

Questions?

Paper

Code