Parallelizable StackLSTM

Shuoyang Ding   Philipp Koehn

NAACL 2019 Structured Prediction Workshop
Minneapolis, MN, United States
June 7th, 2019

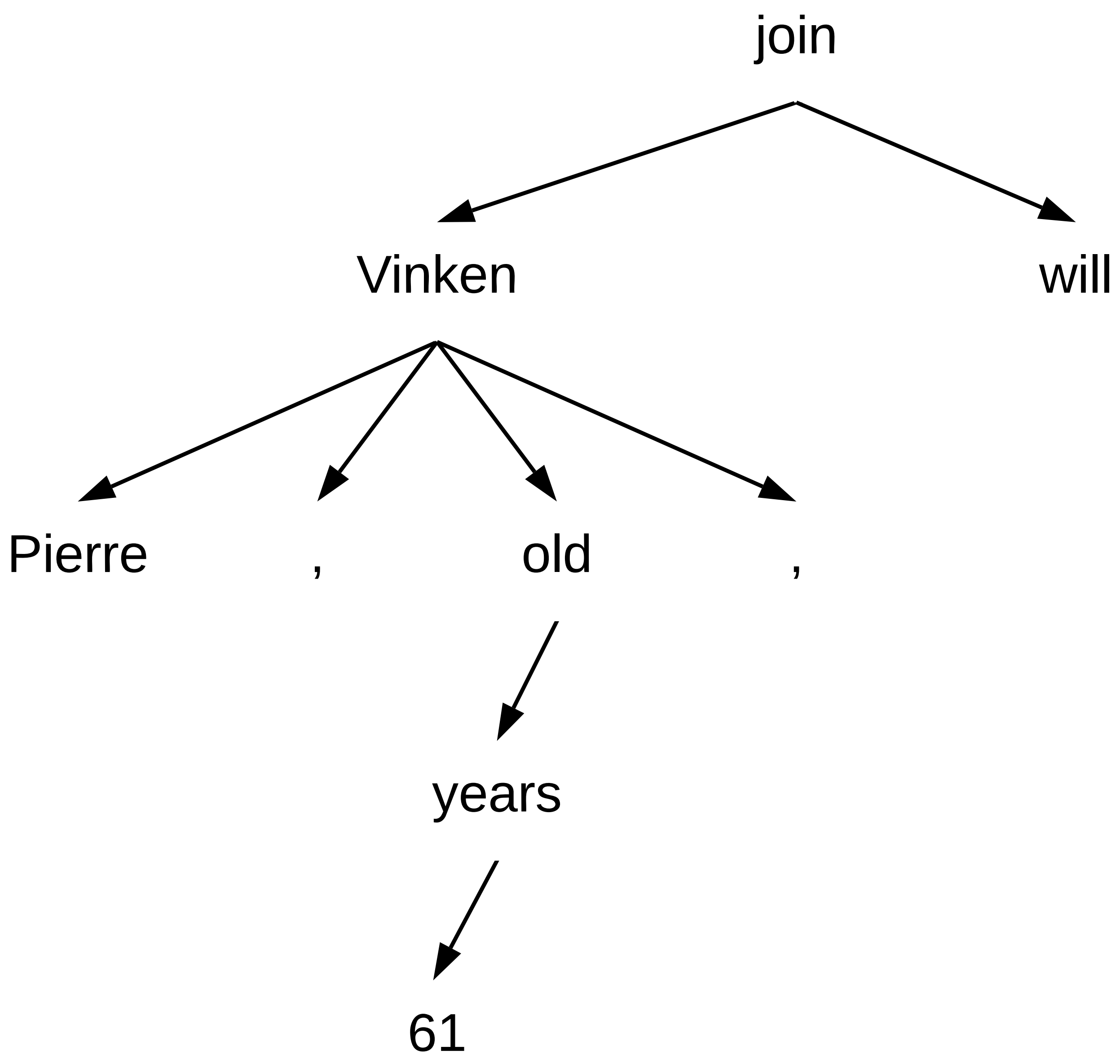
JOHNS HOPKINS
UNIVERSITY

Outline

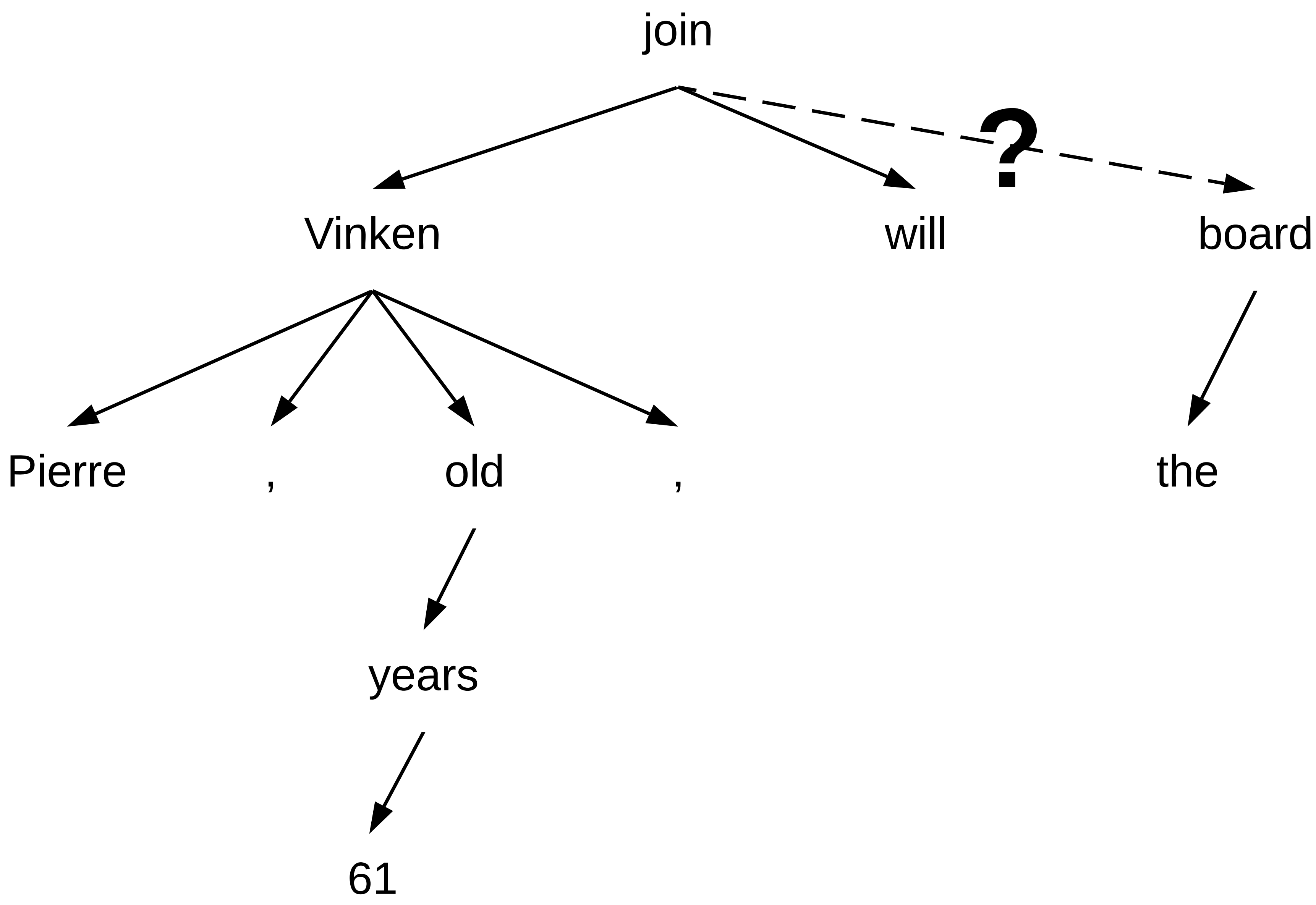
- What is StackLSTM?
- Parallelization Problem
- Homogenizing Computation
- Experiments

What is StackLSTM?

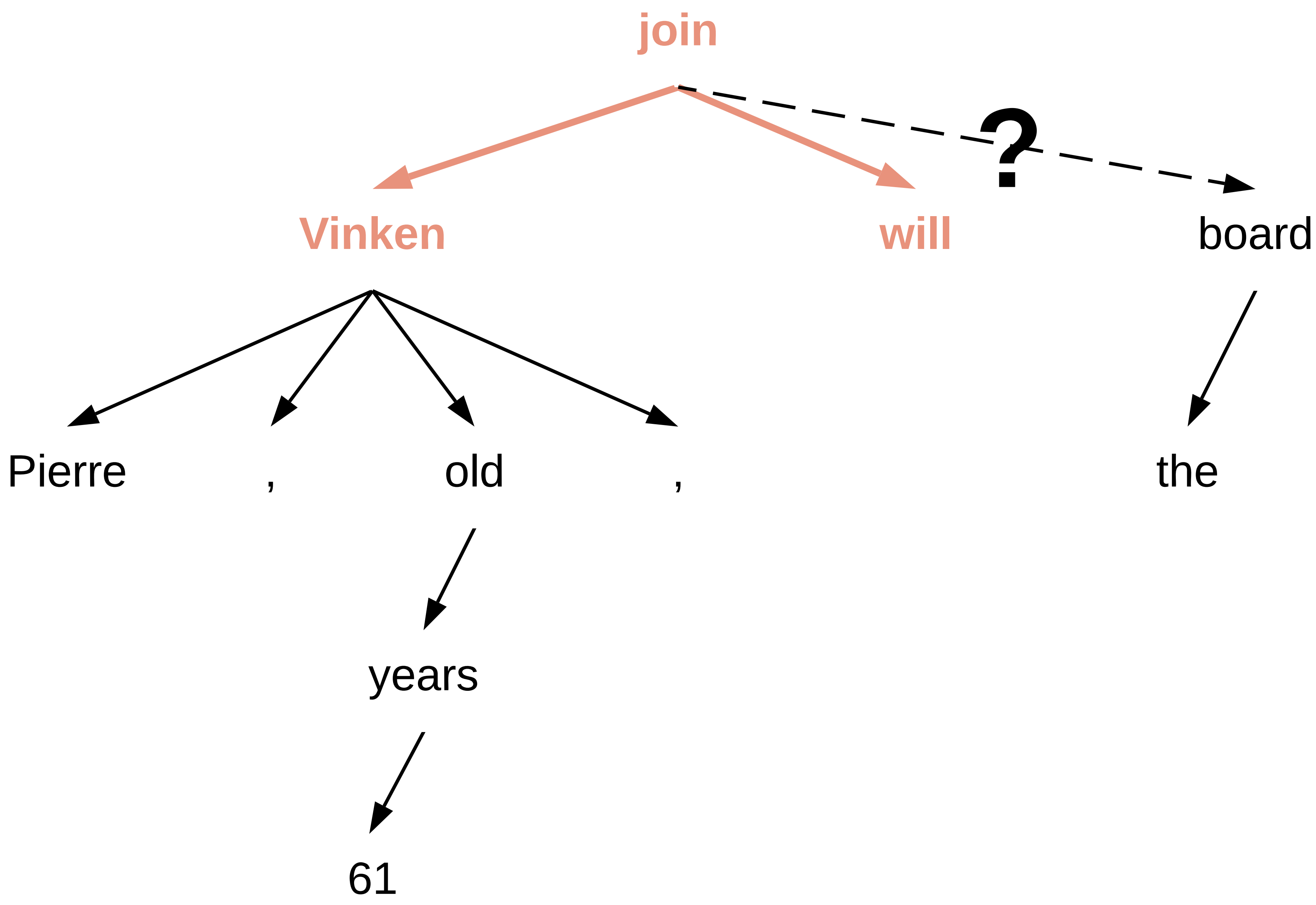
A Partial Tree



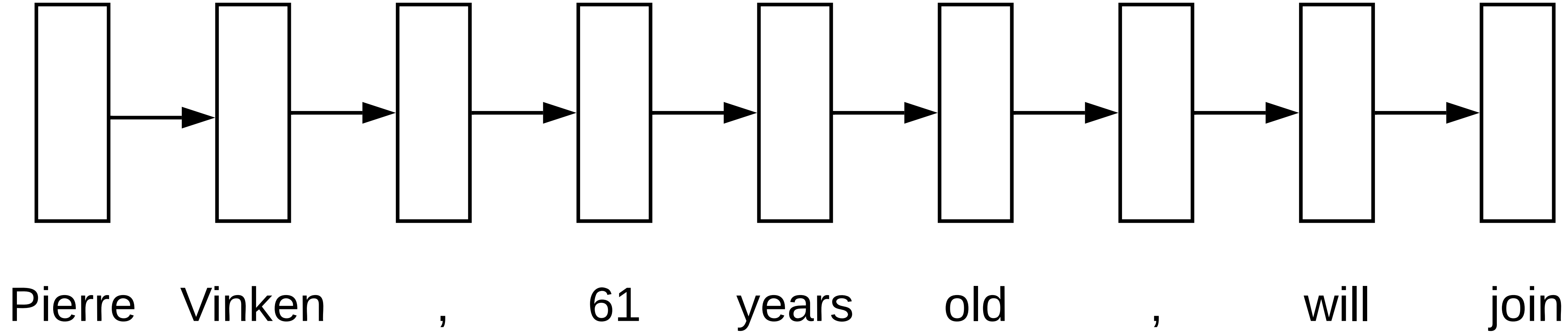
Good Edge?



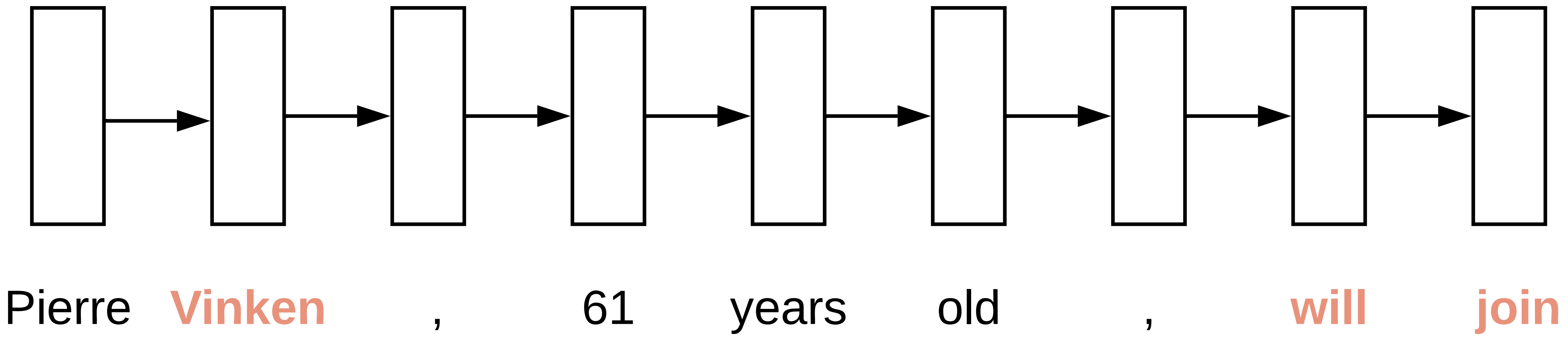
Good Edge?



LSTM?



:(



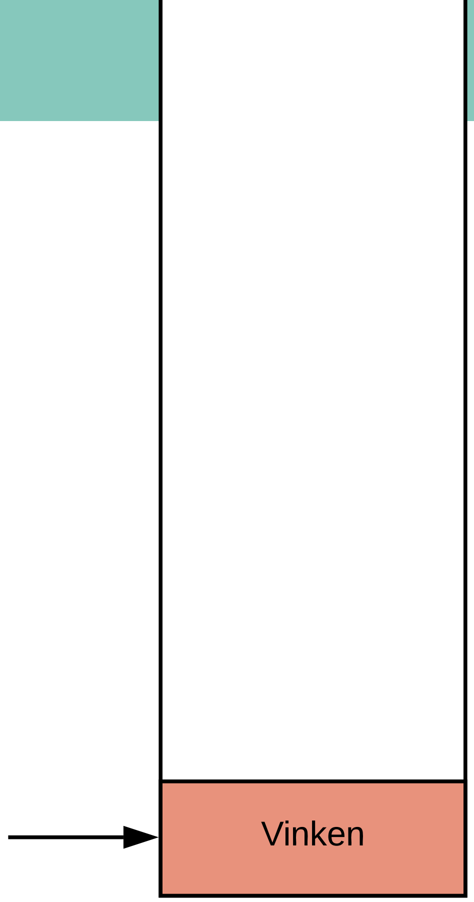
Pierre **Vinken** , 61 years old , **will** **join**

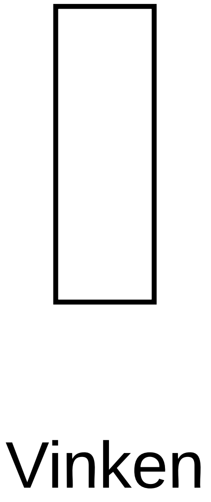
StackLSTM

- An LSTM whose states are stored in a stack
- Computation is conditioned on the stack operation

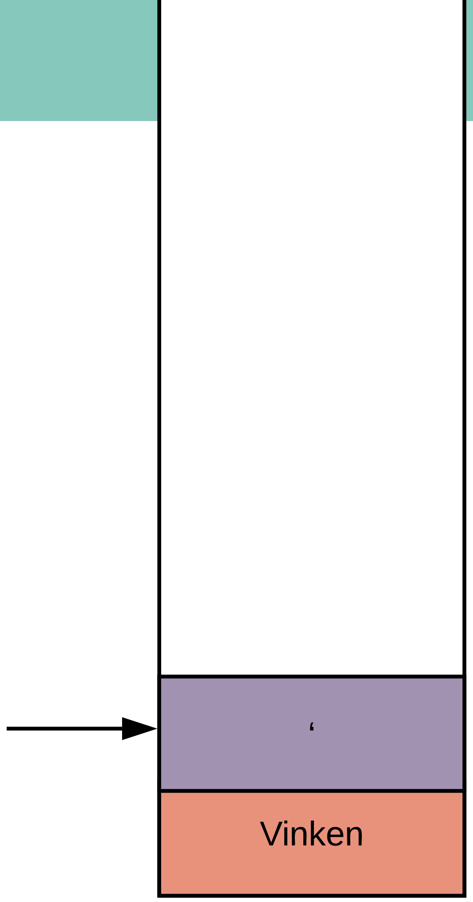
Dyer et al. (2015)
Ballesteros et al. (2017)

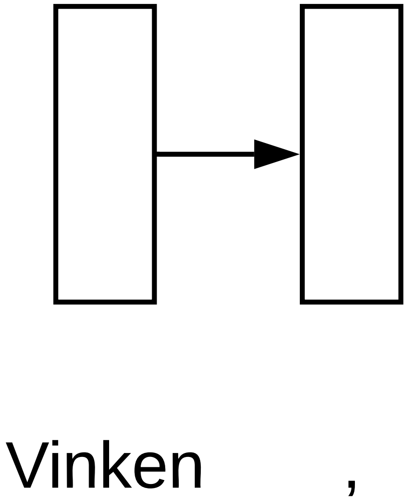
StackLSTM



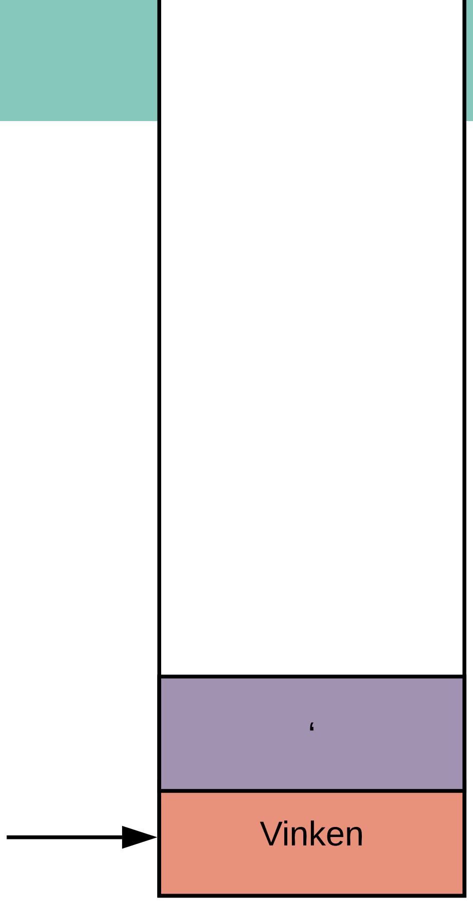


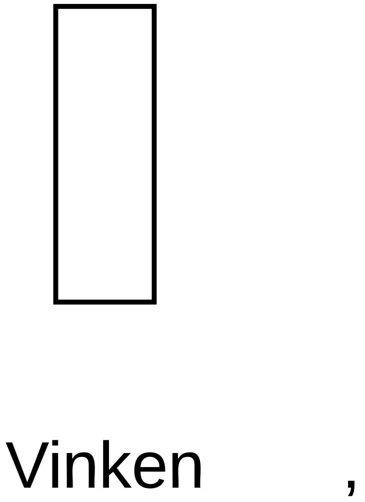
Push ,





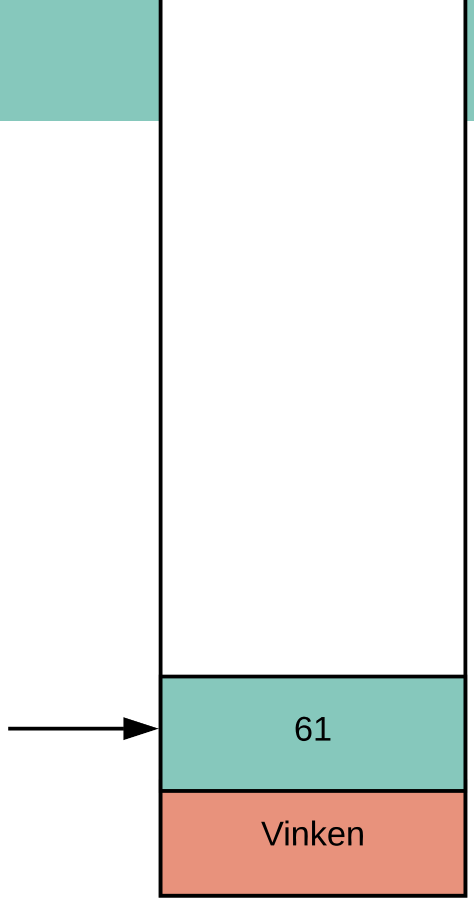
Pop





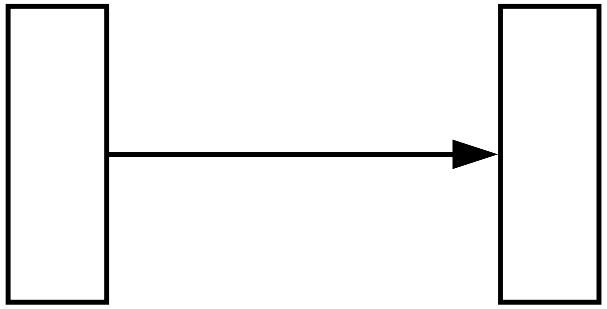
Vinken ,

Push 61



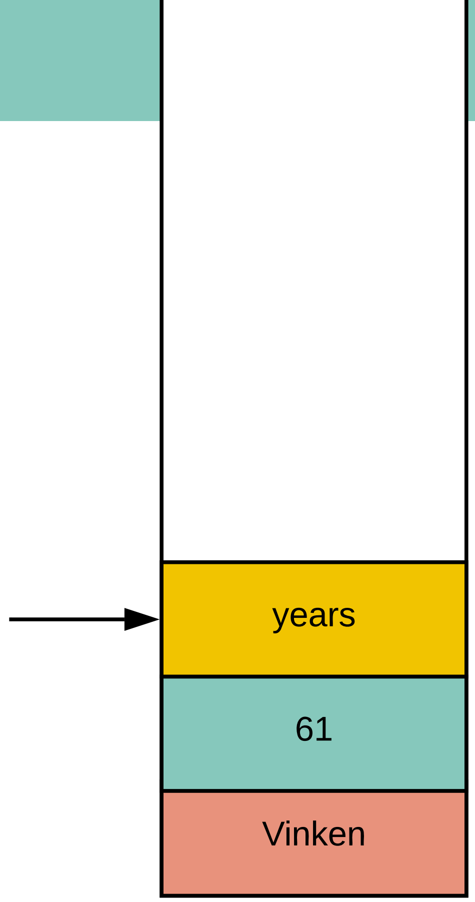
61

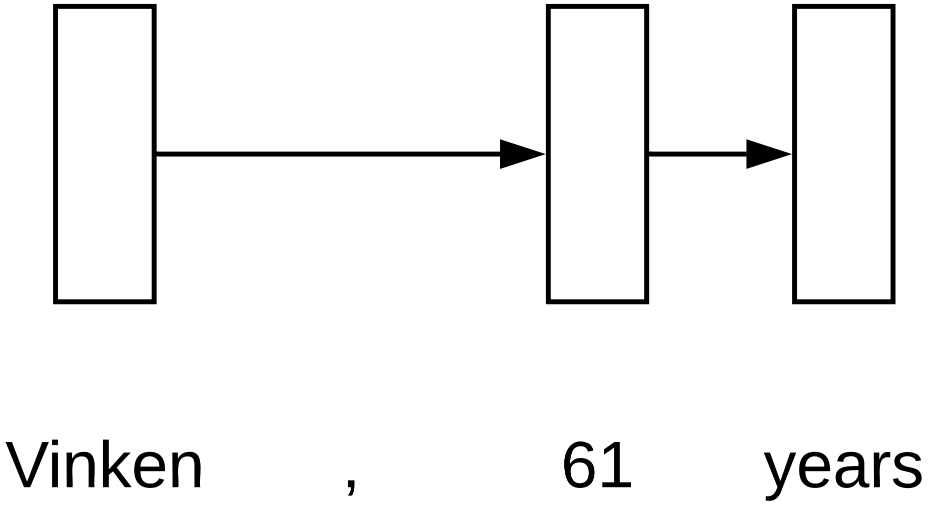
Vinken



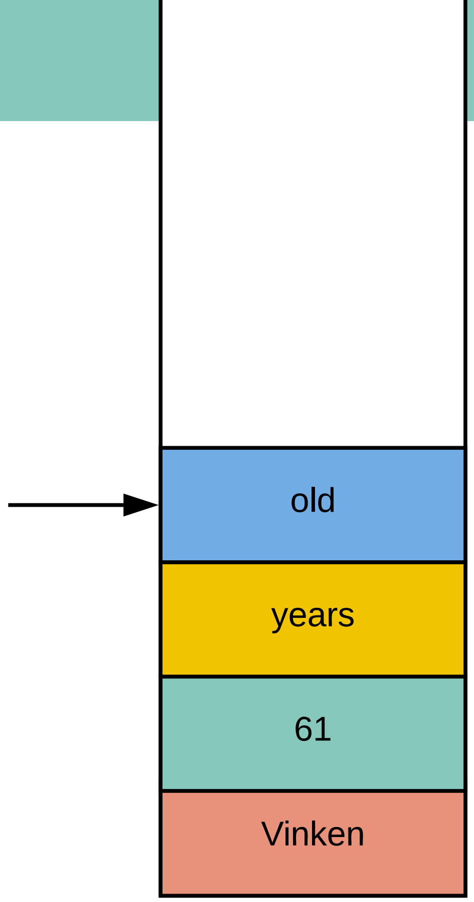
Vinken , 61

Push years





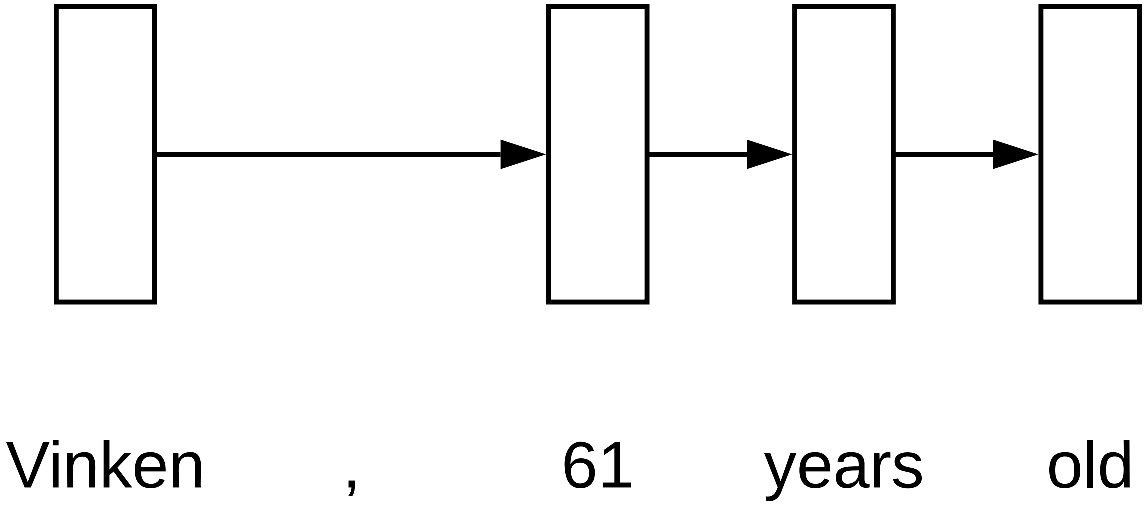
Push old


old

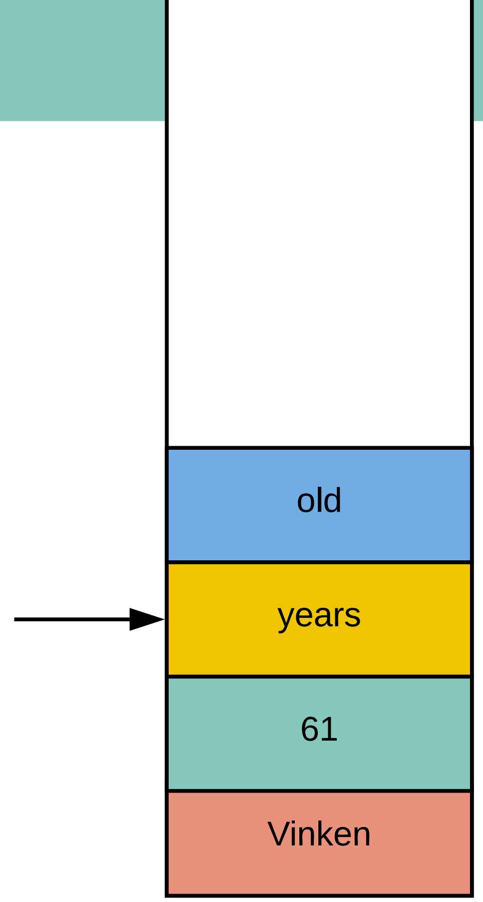
years

61

Vinken


Vinken , 61 years old

Pop

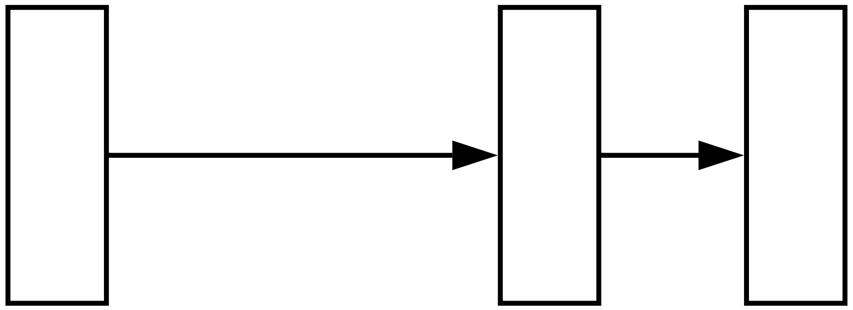


old

years

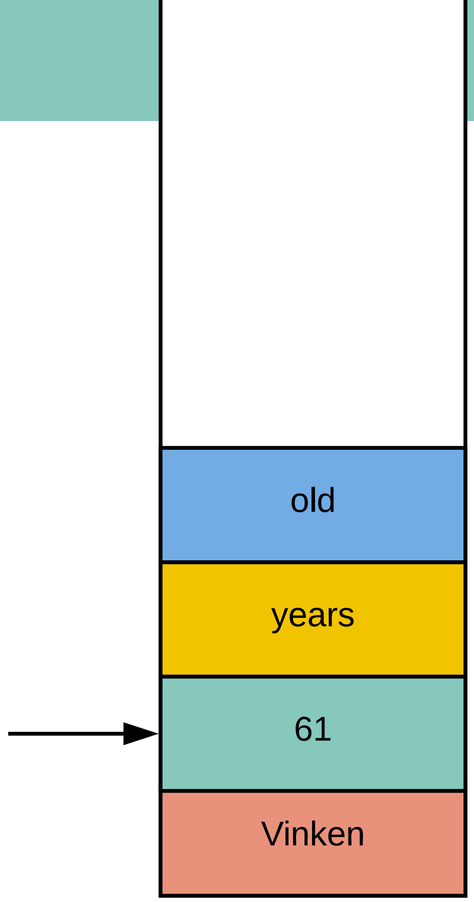
61

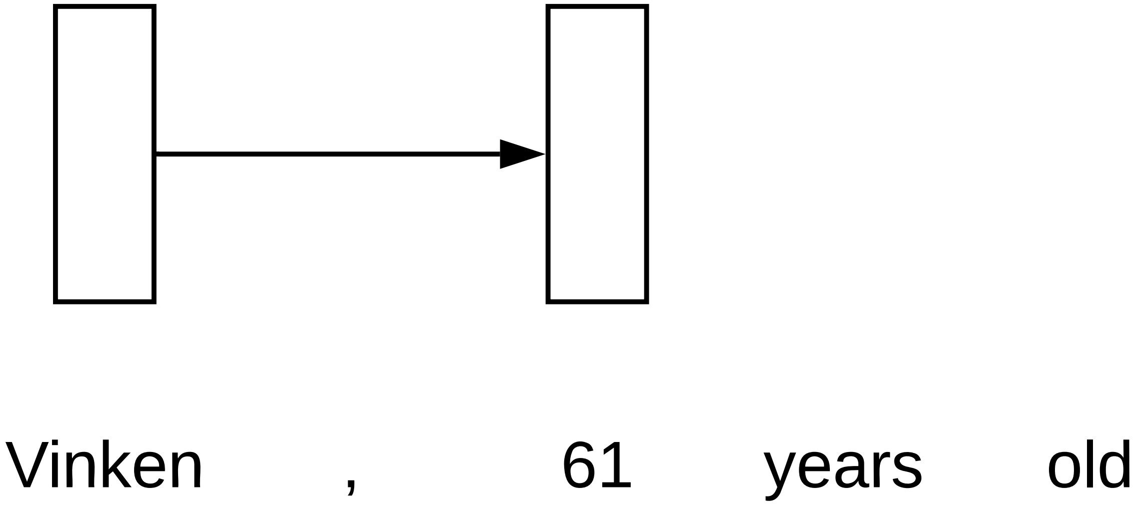
Vinken



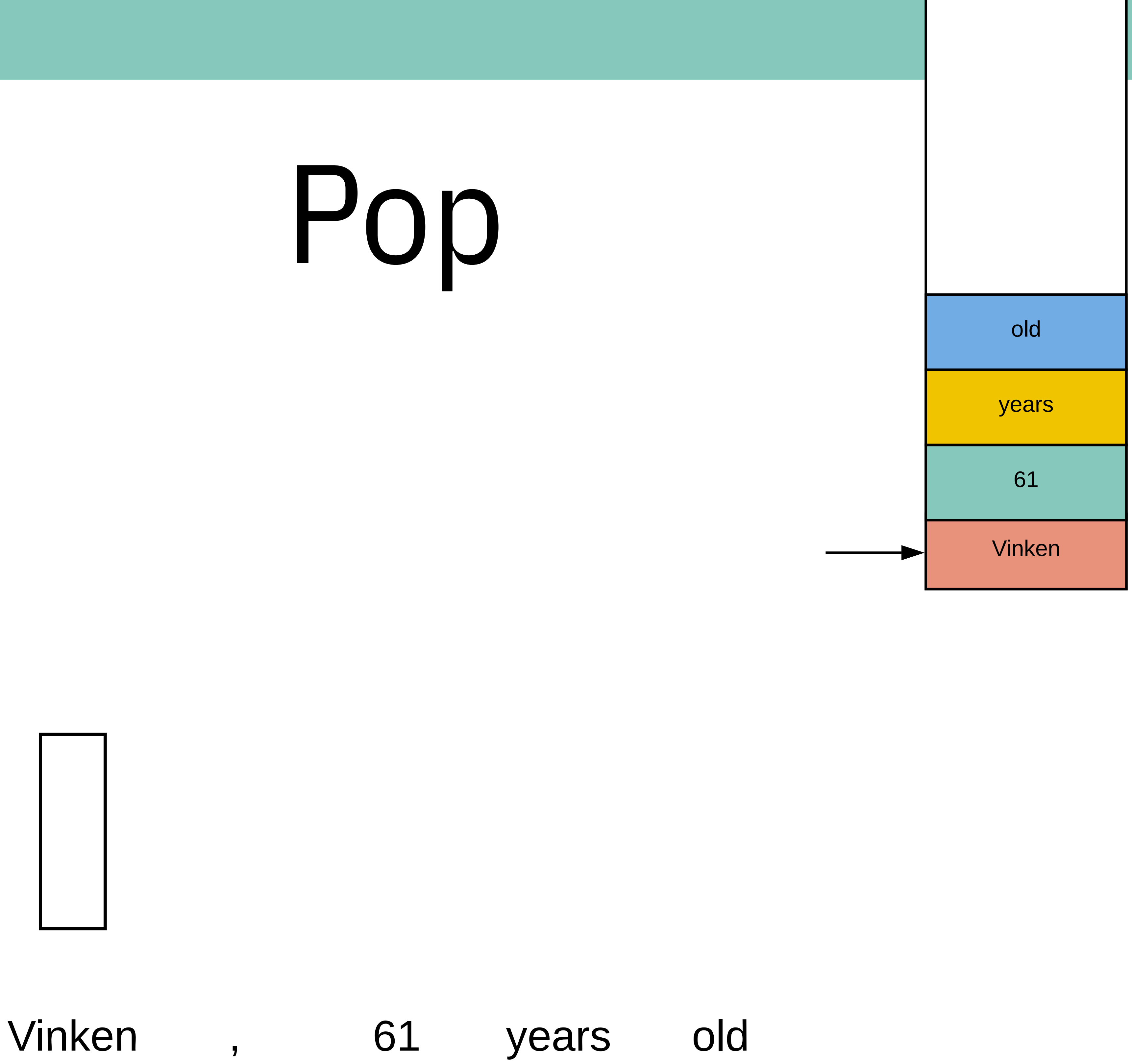
Vinken , 61 years old

Pop

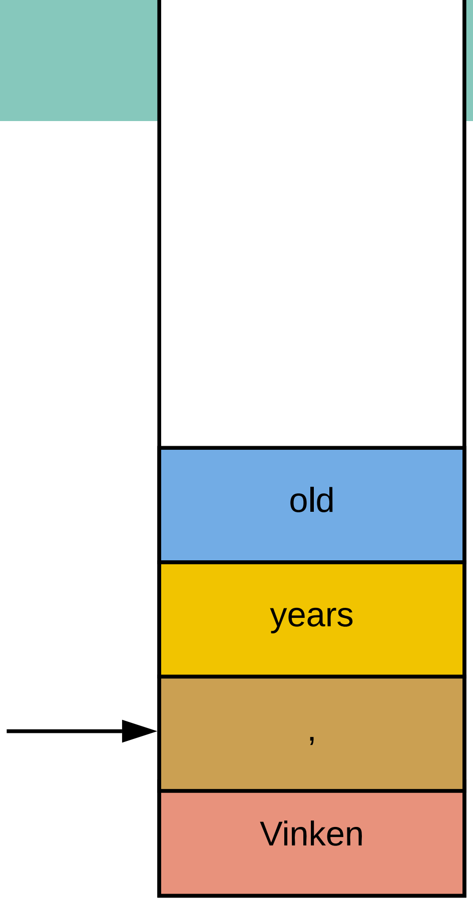


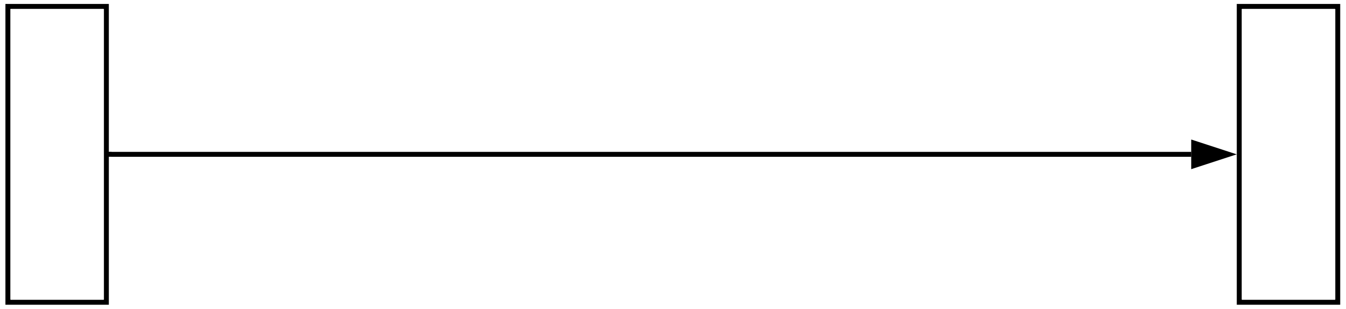


Pop



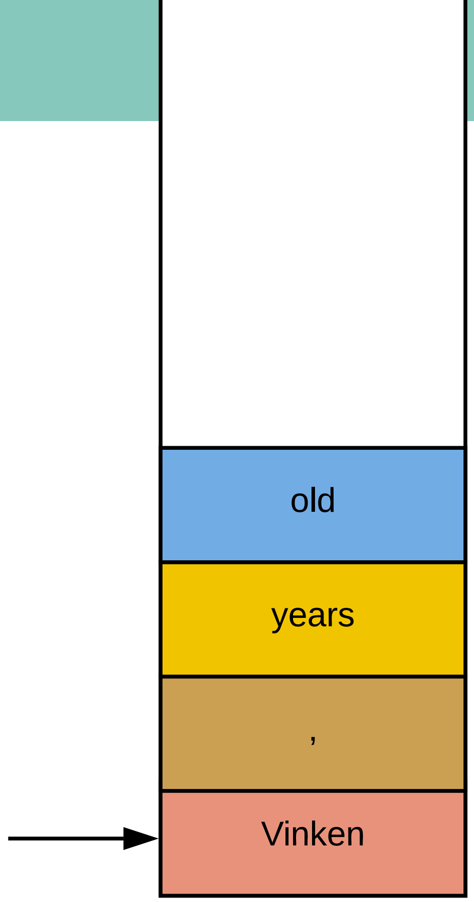
Push ,

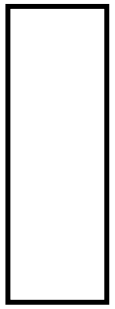




Vinken , 61 years old ,

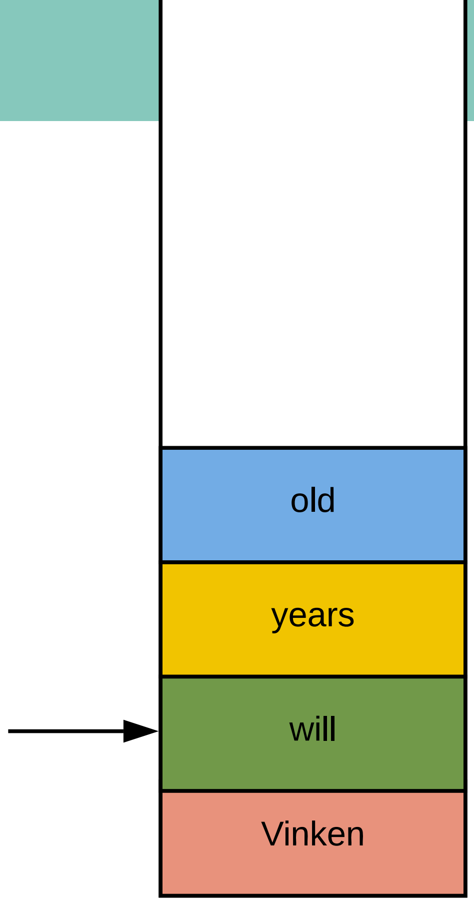
Pop

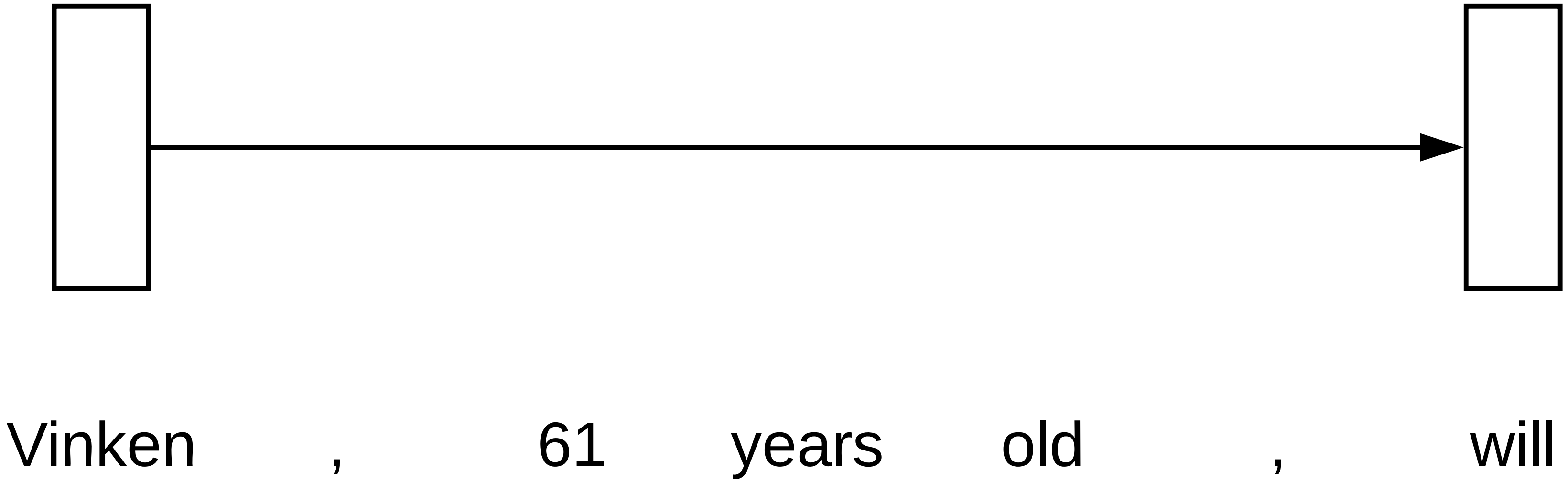




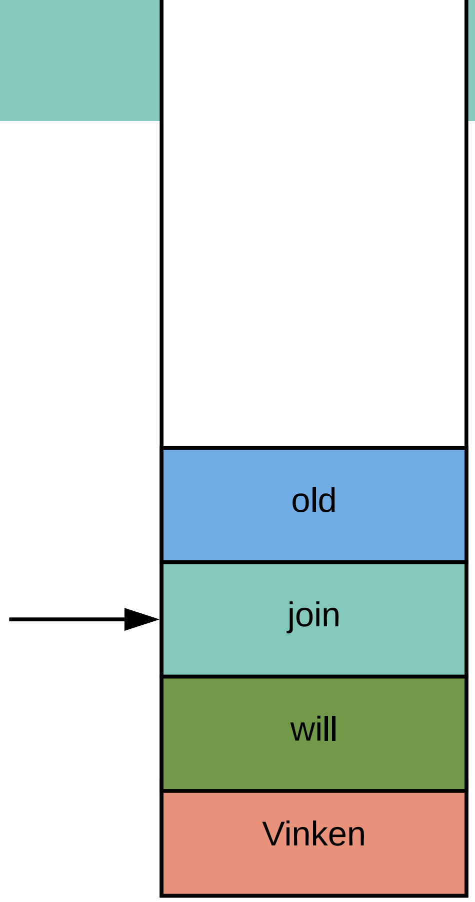
Vinken , 61 years old ,

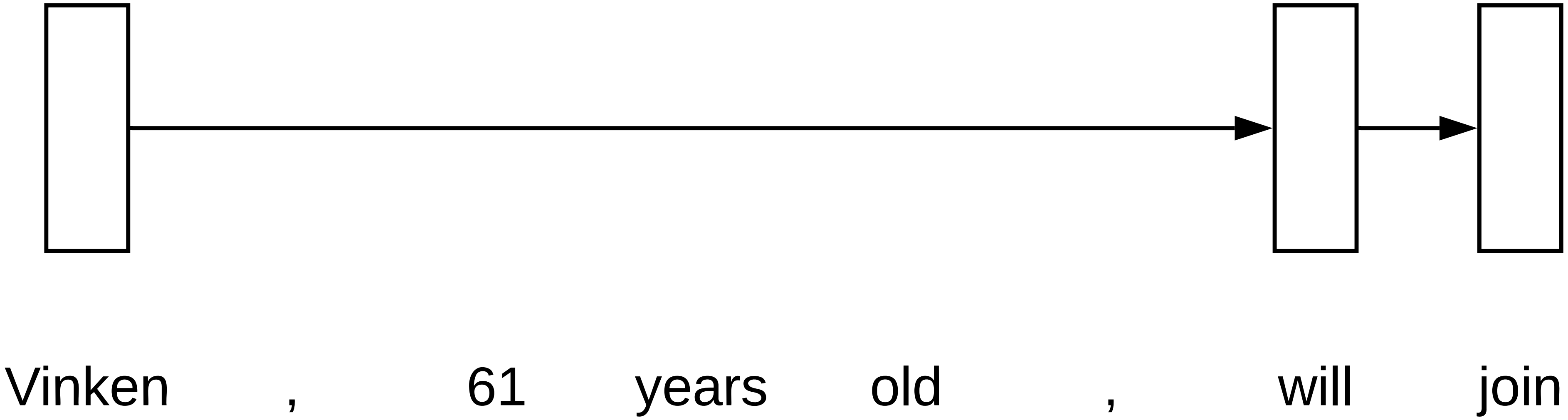
Push will



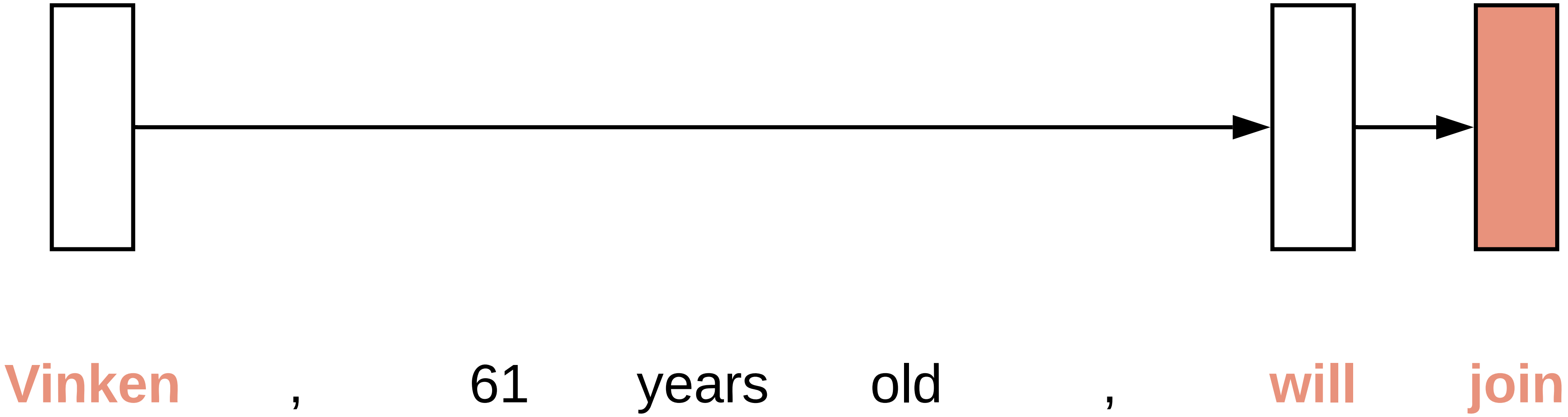


Push join





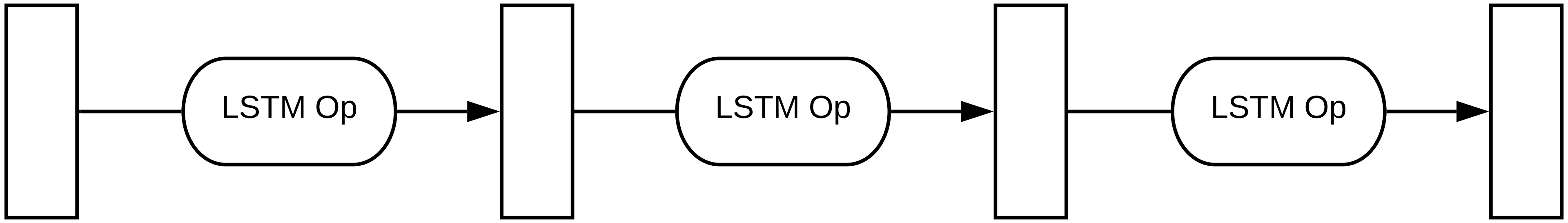
:)



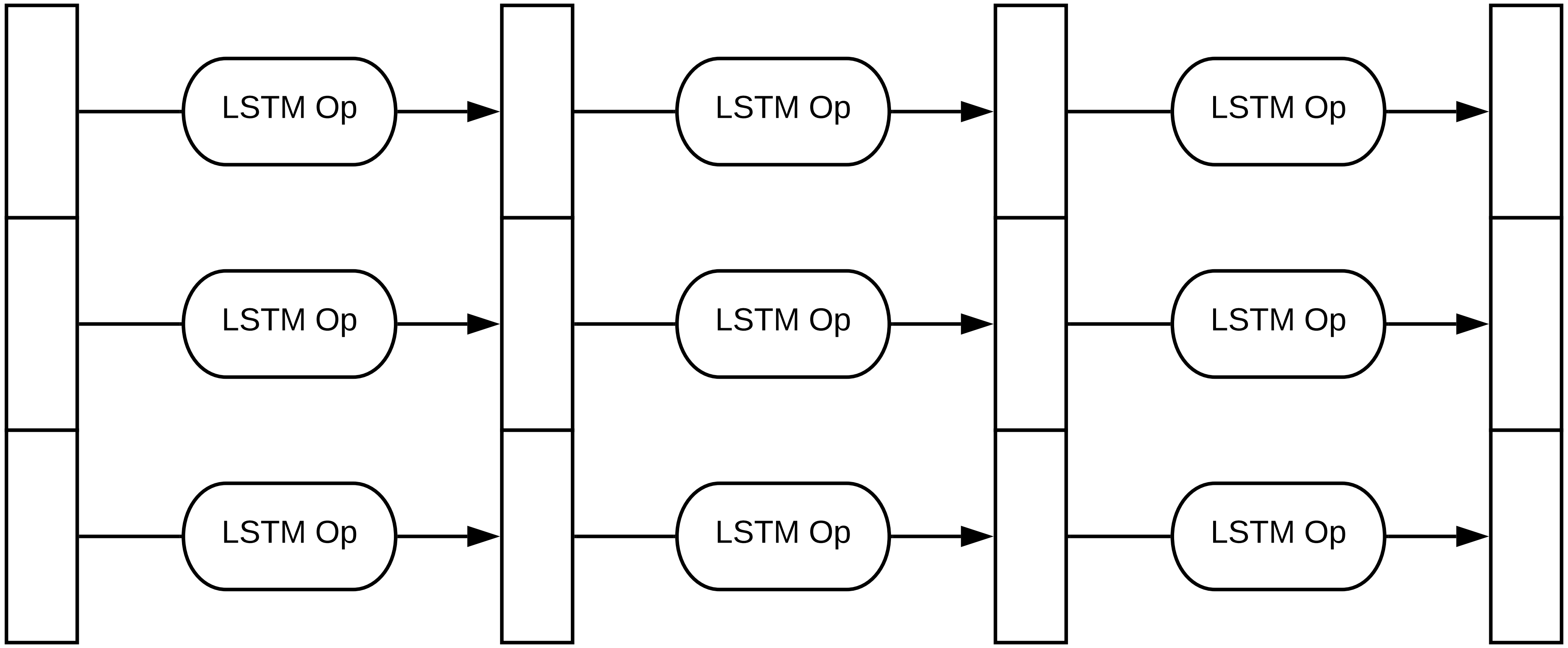
Vinken , 61 years old , will join

Parallelization Problem

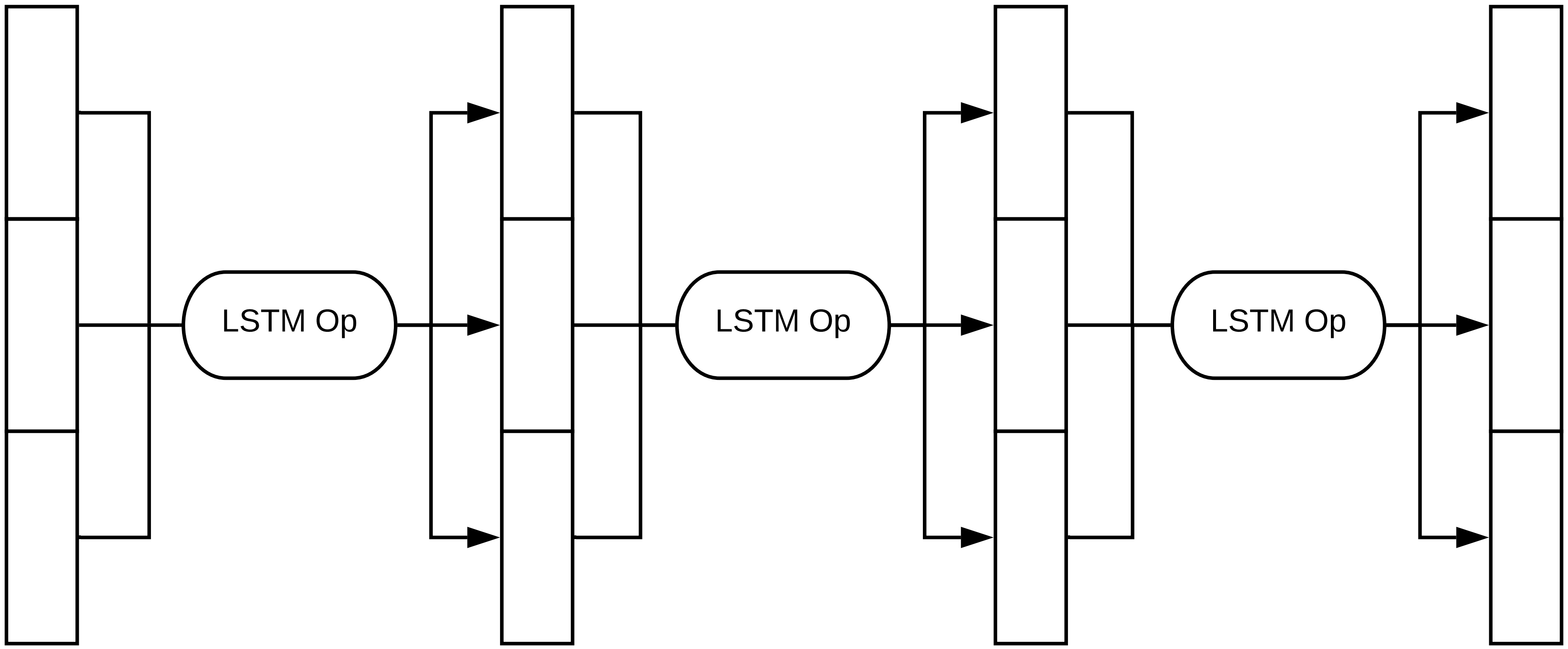
LSTM



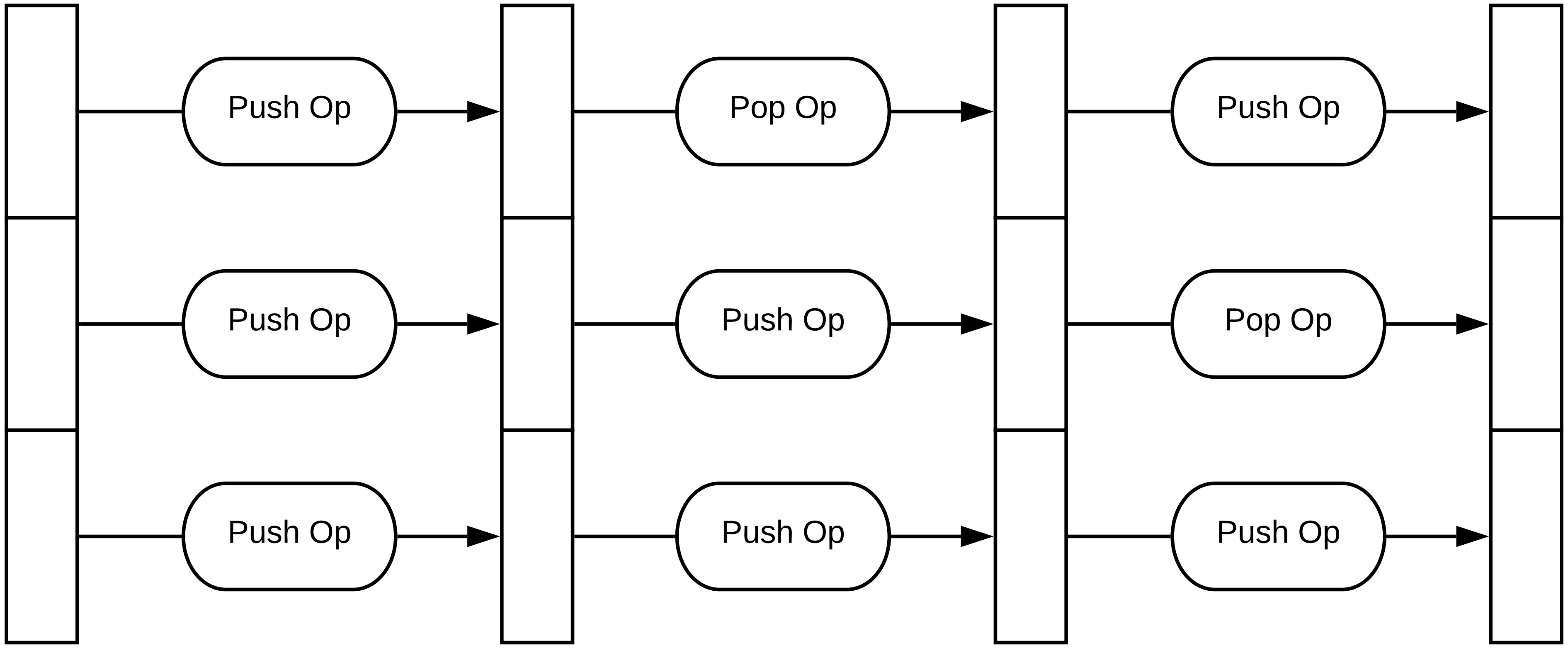
LSTM


LSTM Op
LSTM Op
LSTM Op
LSTM Op
LSTM Op
LSTM Op
LSTM Op
LSTM Op
LSTM Op

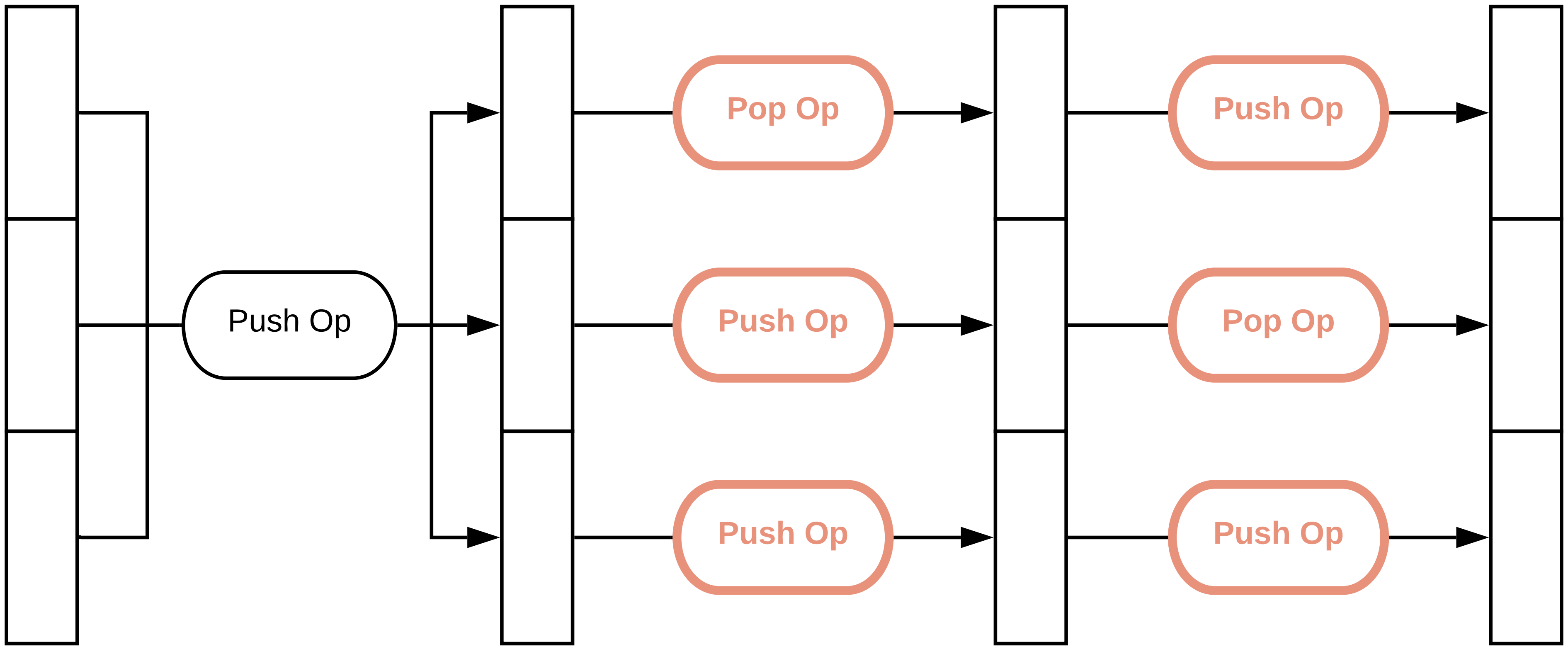
Batched LSTM



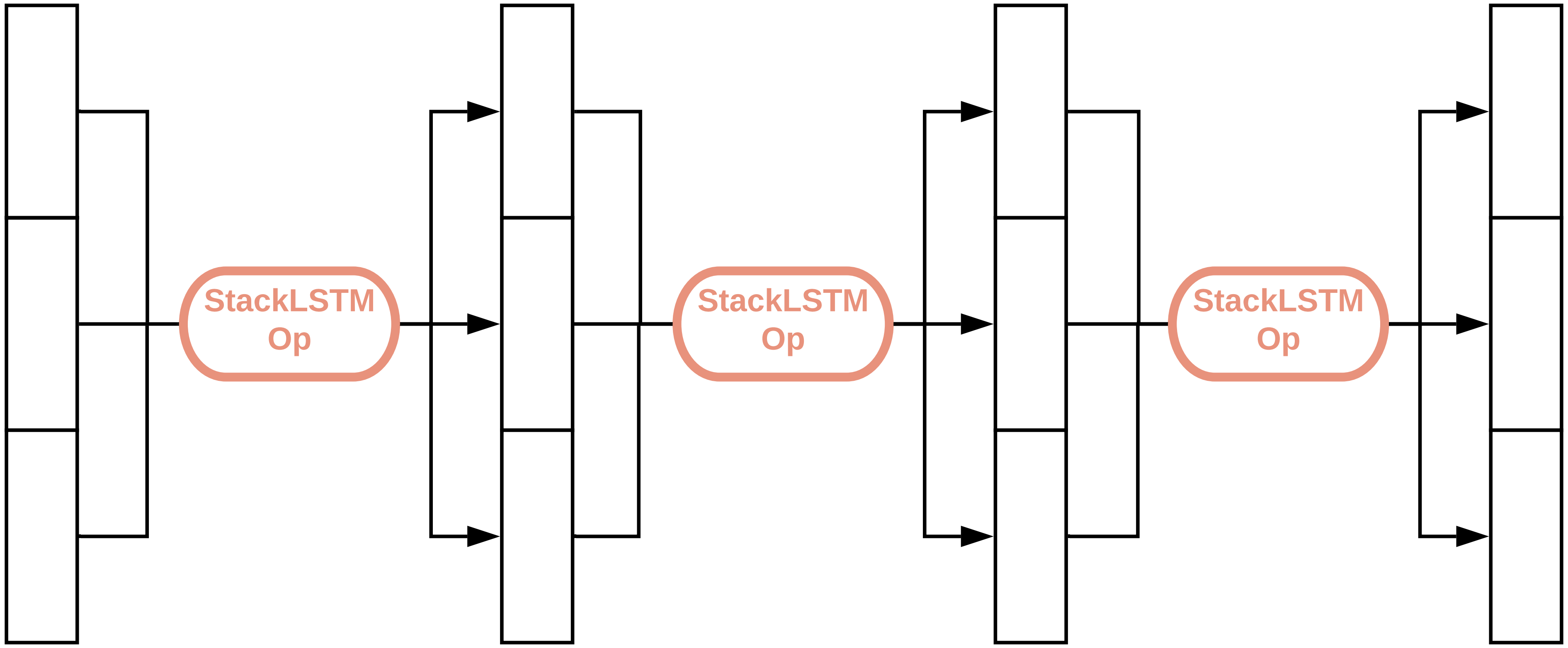
Batched… StackLSTM?



:(


Push Op
Pop Op
Push Op
Push Op
Push Op
Pop Op
Push Op

Wouldn’t it be nice if...



Homogenizing Computation

Push

- read the stack top hidden state h_{p(t)};

- perform LSTM forward computation with x(t) and h_{p(t)};

- write new hidden state to h_{p(t) + 1};

- update stack top pointer p(t+1) = p(t) + 1;

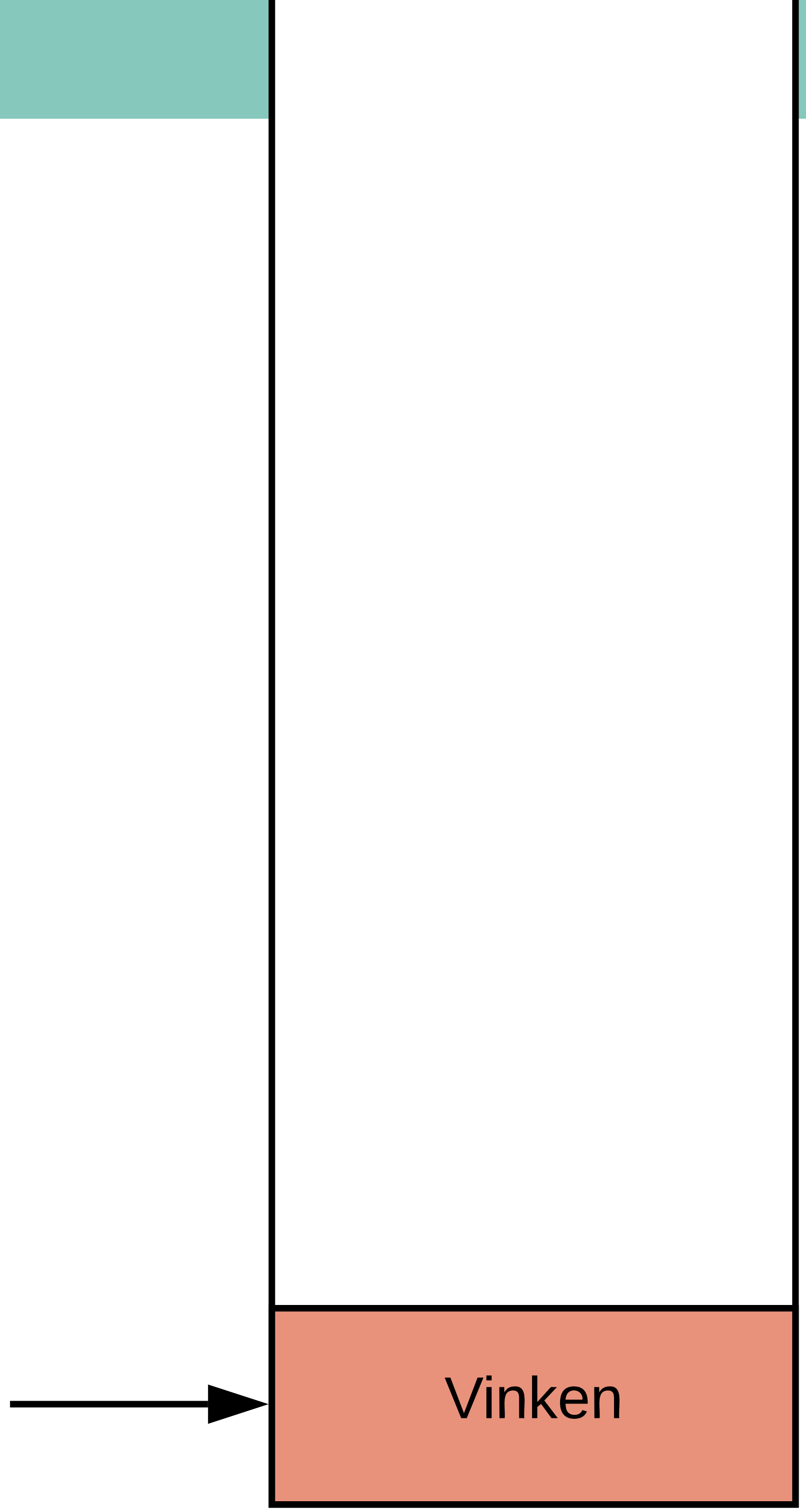
Push

- **read the stack top hidden state h_{p(t)};**

- perform LSTM forward computation with x(t) and h_{p(t)};

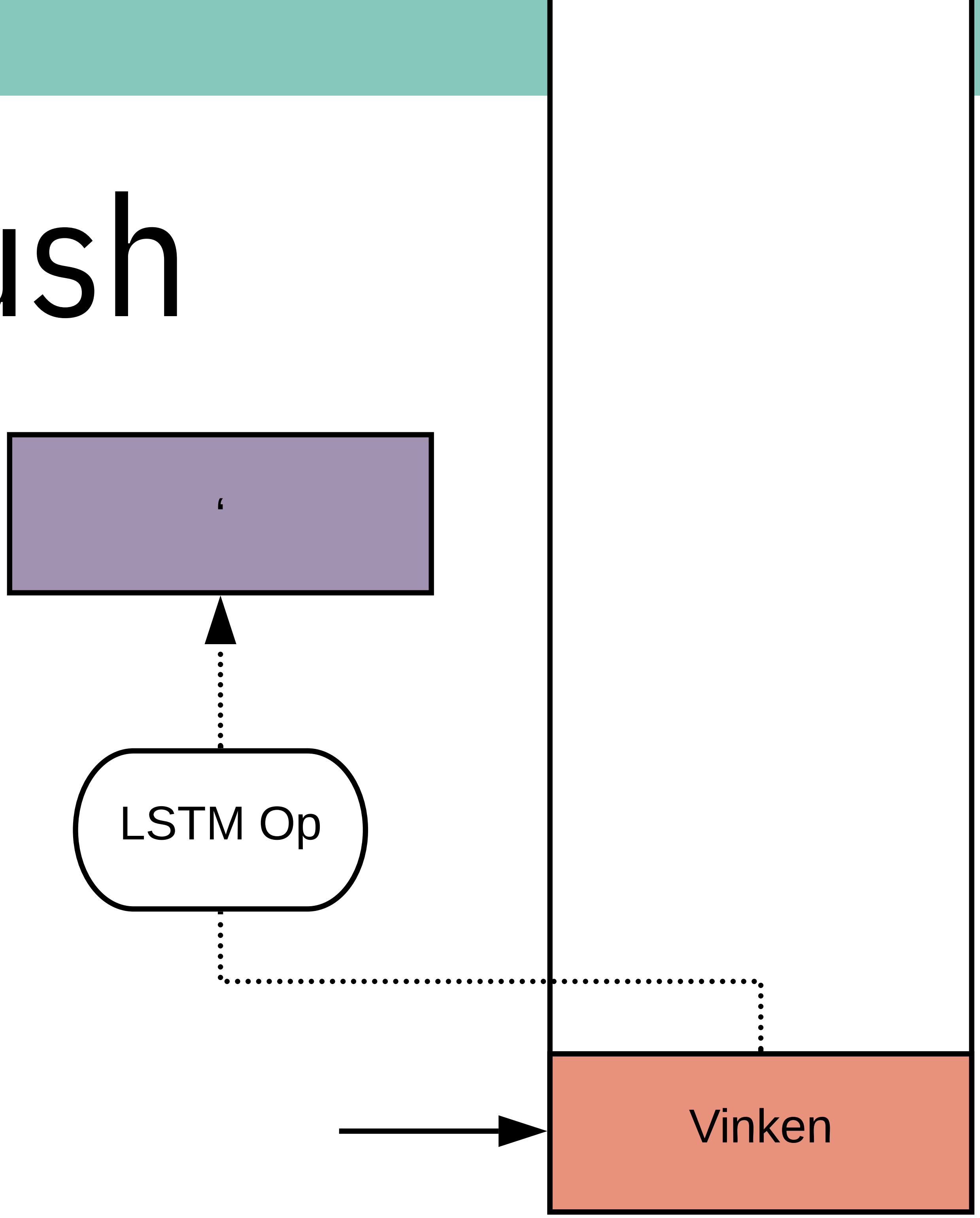
- write new hidden state to h_{p(t) + 1};

- update stack top pointer p(t+1) = p(t) + 1;



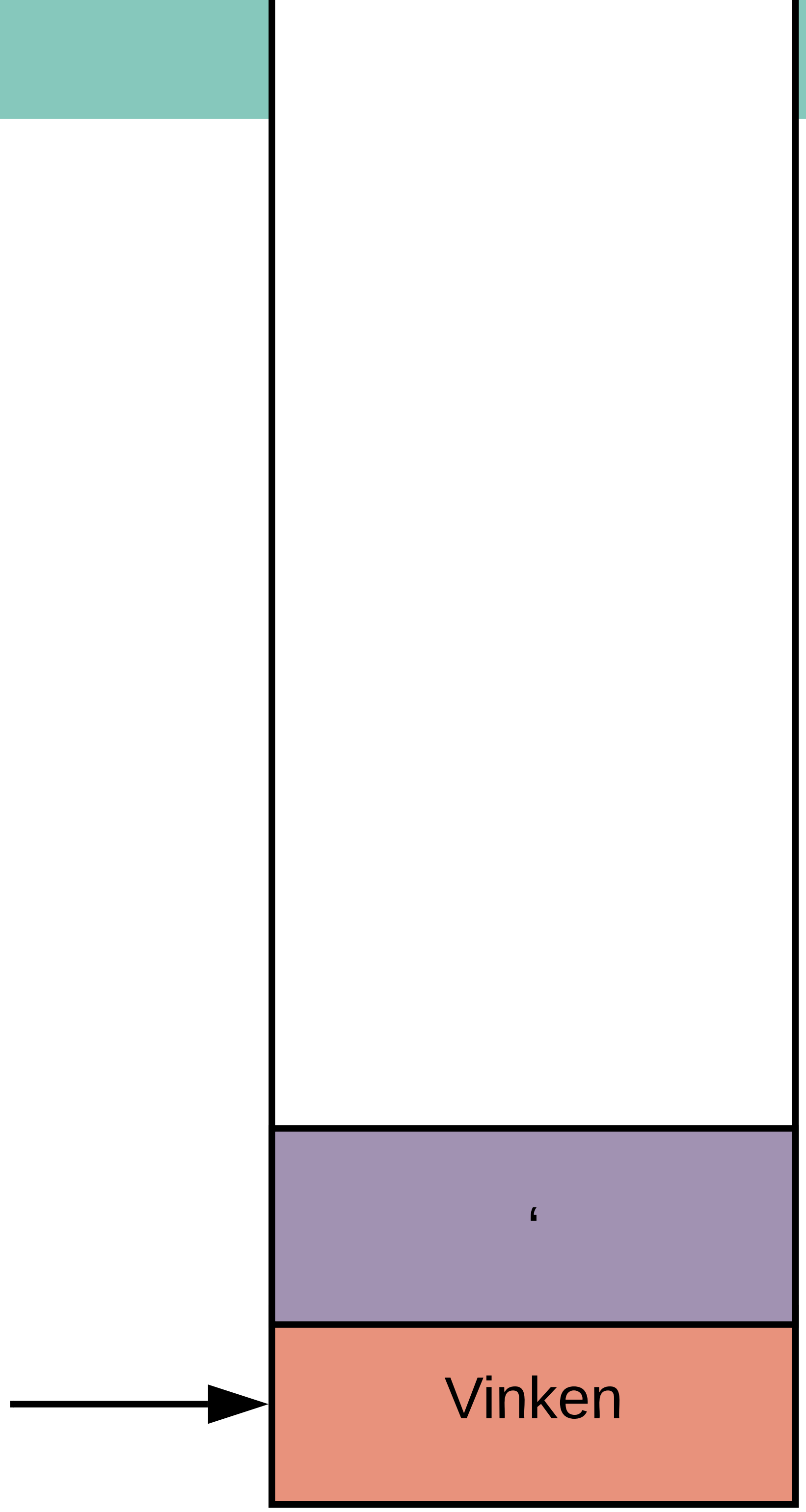
Push

- read the stack top hidden state h_{p(t)};
- **perform LSTM forward computation with x(t) and h_{p(t)};**
- write new hidden state to h_{p(t) + 1};
- update stack top pointer p(t+1) = p(t) + 1;



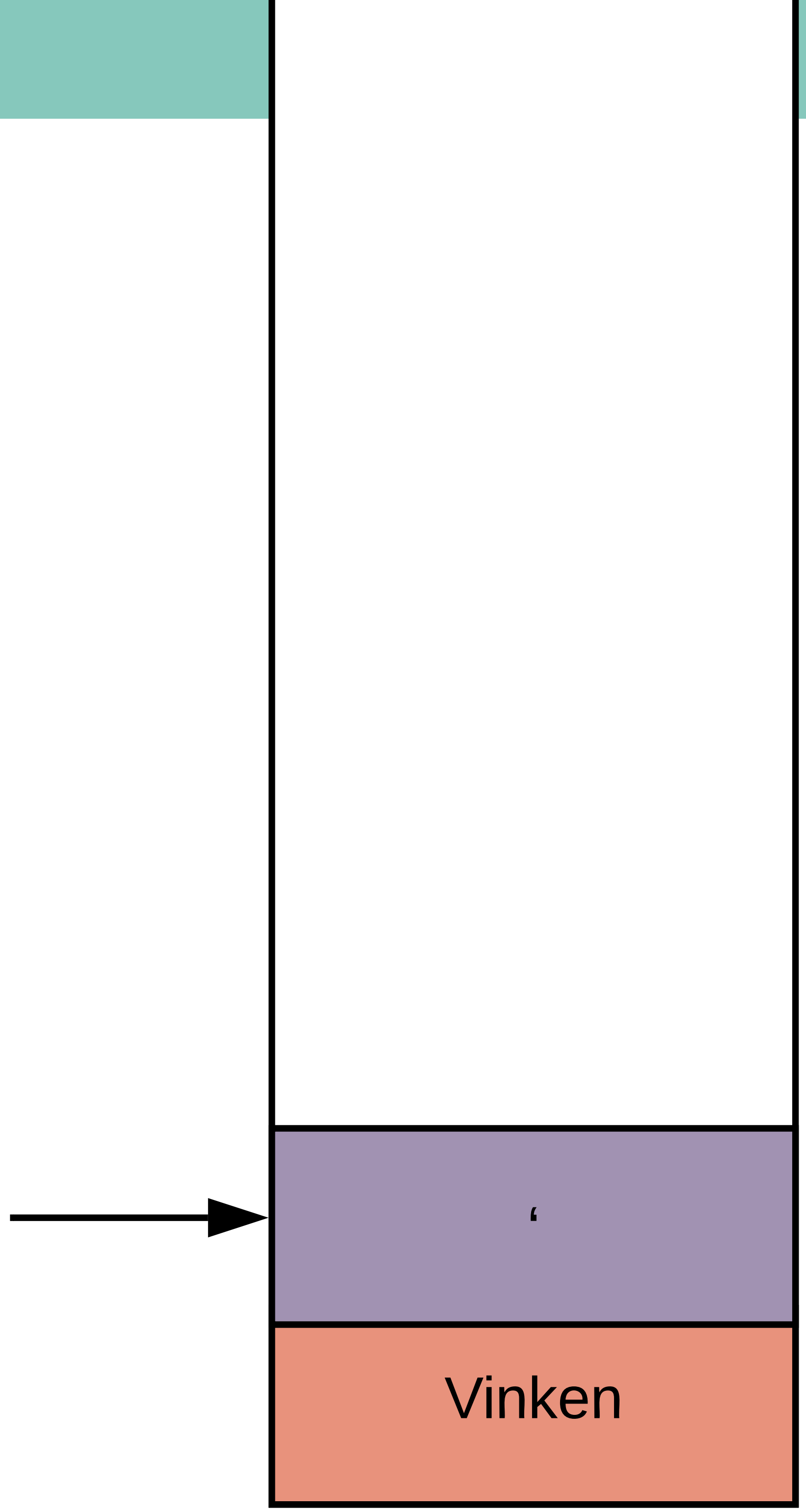
Push

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- **write new hidden state to h_{p(t) + 1};**
- update stack top pointer p(t+1) = p(t) + 1;



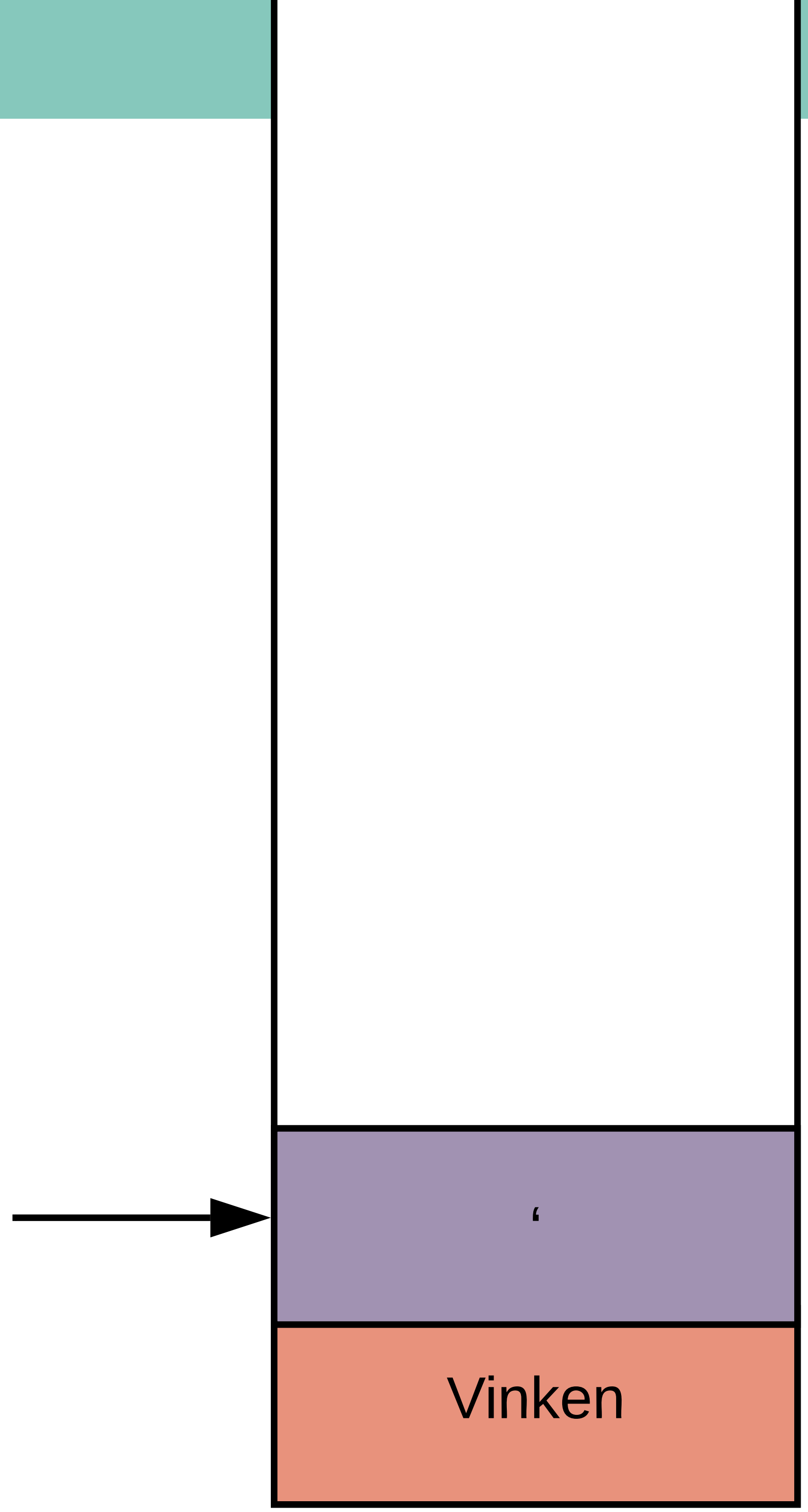
Push

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- write new hidden state to h_{p(t) + 1};
- **update stack top pointer p(t+1) = p(t) + 1;**



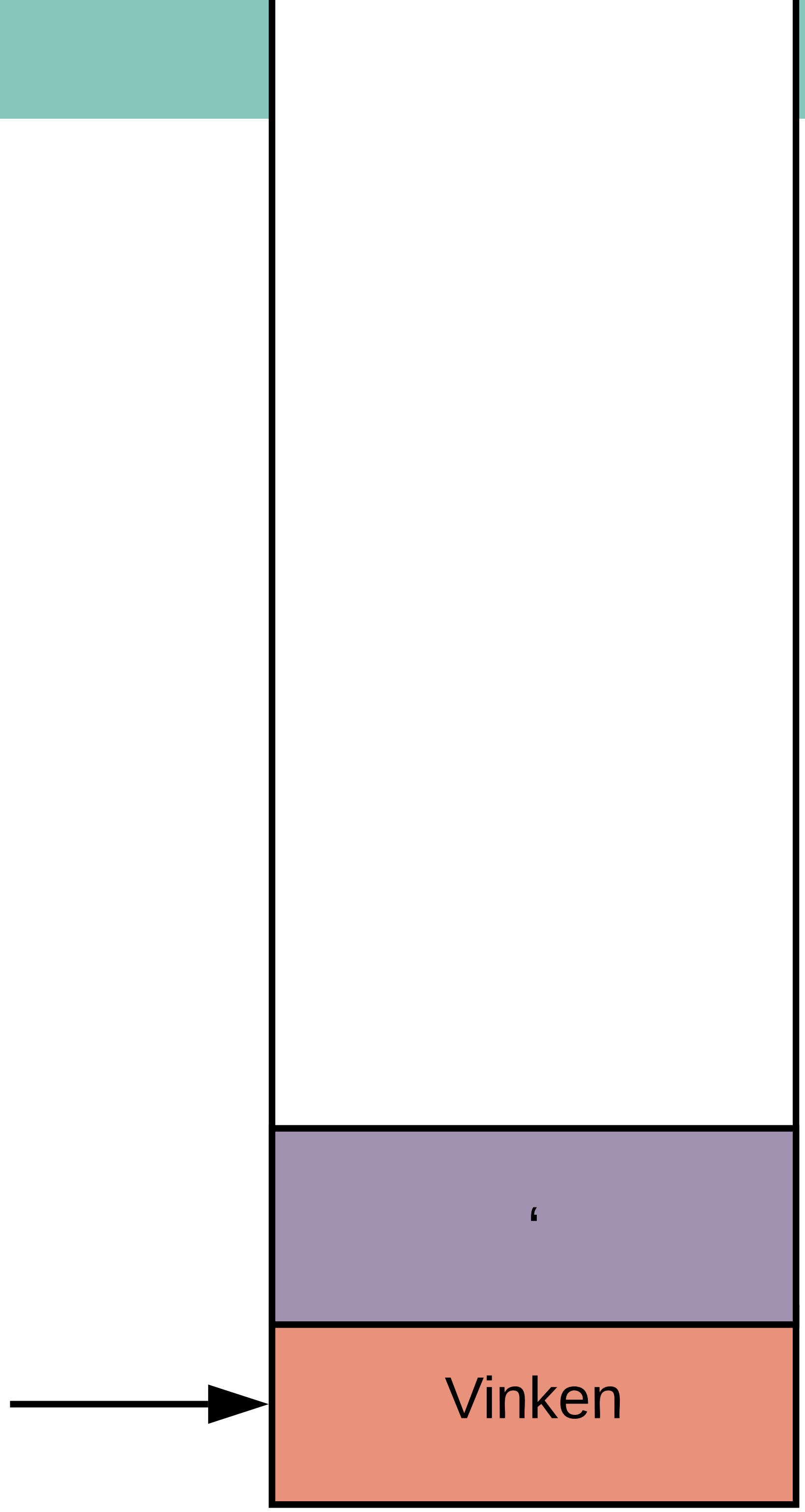
Pop

- update stack top pointer
  `p(t+1) = p(t) - 1;`



Pop

- `update stack top pointer`
  `p(t+1) = p(t) - 1;`



Observation 1

- read the stack top hidden state h_{p(t)};

- perform LSTM forward computation with x(t) and h_{p(t)};

- write new hidden state to h_{p(t) + 1};

- update stack top pointer p(t+1) = p(t) + 1;

- update stack top pointer p(t+1) = p(t) - 1;

Observation 1

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- write new hidden state to h_{p(t) + 1};
- update stack top pointer p(t+1) = p(t) **+ op**;

Use op = +1 for push and op = -1 for pop

- update stack top pointer p(t+1) = p(t) **+ op**;

Observation 1

The computation performed for Pop operation is a subset of Push operation.

Observation 2

Is it safe to do the other computations for push for pop as well?

Observation 2

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- write new hidden state to h_{p(t) + 1};
- update stack top pointer p(t+1) = p(t) + op;

- update stack top pointer p(t+1) = p(t) + op;

Observation 2

- read the stack top hidden state h_{p(t)};

- perform LSTM forward computation with x(t) and h_{p(t)};

- **write new hidden state to h_{p(t) + 1};**

- update stack top pointer p(t+1) = p(t) + op;

- update stack top pointer p(t+1) = p(t) + op;

Observation 2

A write will always happen before the stack top pointer advances.

Observation 2

If one wants to write anything in the higher position than the current stack top pointer…

Observation 2

If one wants to write anything in the higher position than the current stack top pointer...

Just do it!

Observation 2

- read the stack top hidden state h_{p(t)};

- perform LSTM forward computation with x(t) and h_{p(t)};

- write new hidden state to h_{p(t) + 1};

- update stack top pointer p(t+1) = p(t) + op;

- update stack top pointer p(t+1) = p(t) + op;

Observation 2

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- write new hidden state to h_{p(t) + 1};
- update stack top pointer p(t+1) = p(t) + op;

- read the stack top hidden state h_{p(t)};
- perform LSTM forward computation with x(t) and h_{p(t)};
- write new hidden state to h_{p(t) + 1};
- update stack top pointer p(t+1) = p(t) + op;

Done!

- `read the stack top hidden state h_{p(t)};`
- `perform LSTM forward computation with x(t) and h_{p(t)};`
- `write new hidden state to h_{p(t) + 1};`
- `update stack top pointer p(t+1) = p(t) + op;`

Experiments

Benchmark

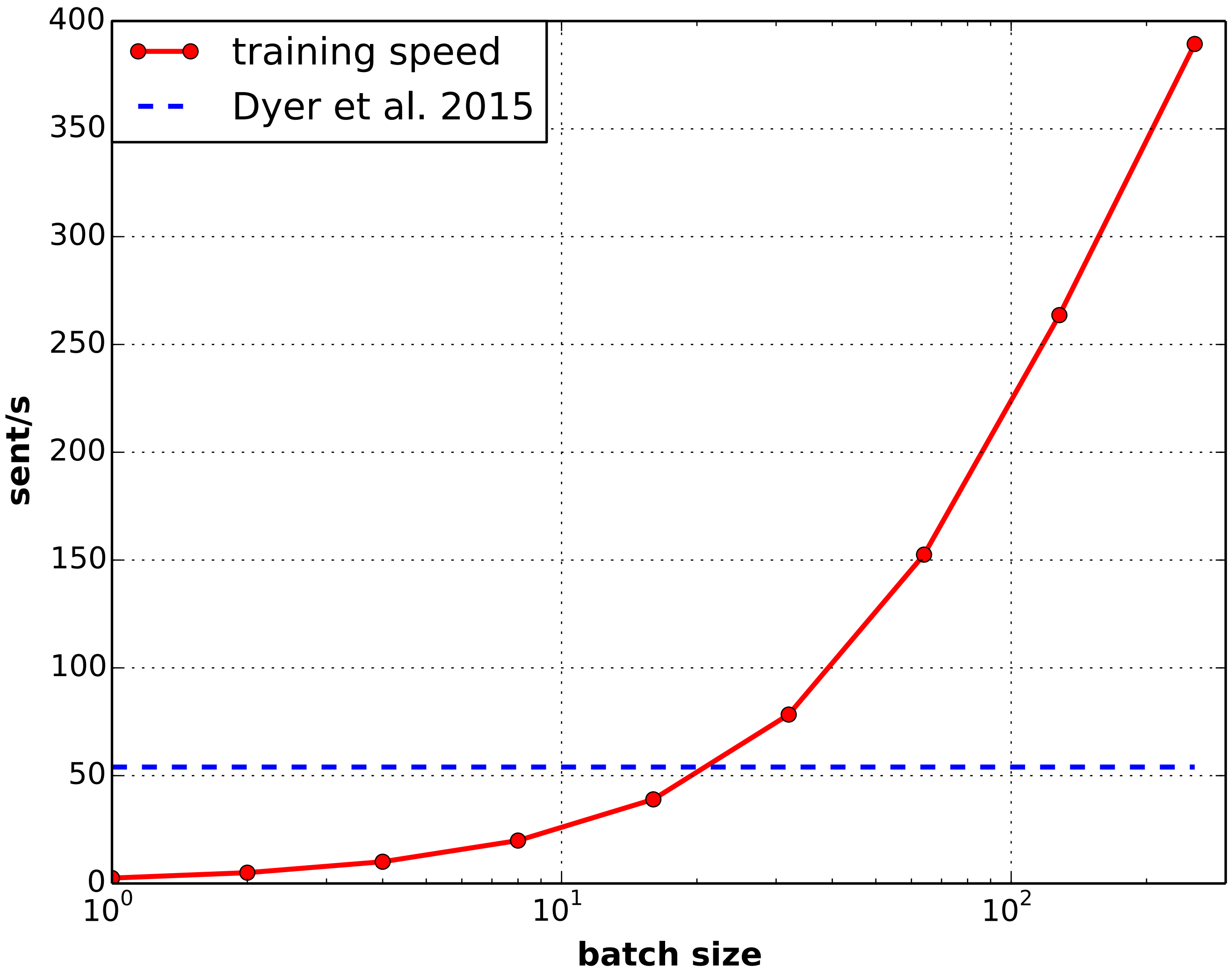
Transition-based dependency parsing on Stanford Dependency Treebank

PyTorch, Single K80 GPU

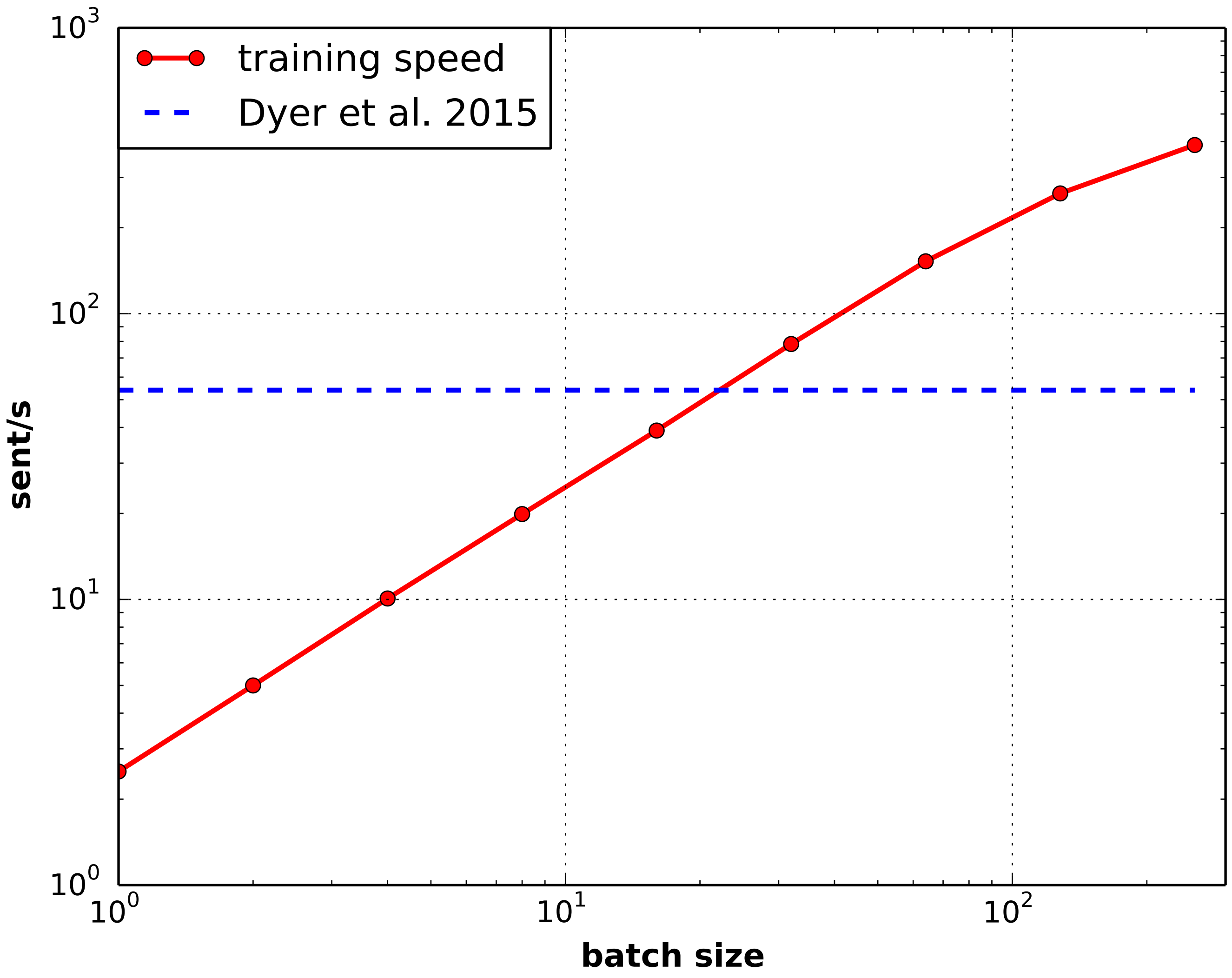
Hyperparameters

- Largely following Dyer et al. (2015); Ballesteros et al. (2017), except:
  - Adam w/ ReduceLROnPlateau and warmup
  - Arc-Hybrid w/o composition function
  - Slightly larger models (200 hidden, 200 state, 48 action embedding) perform better

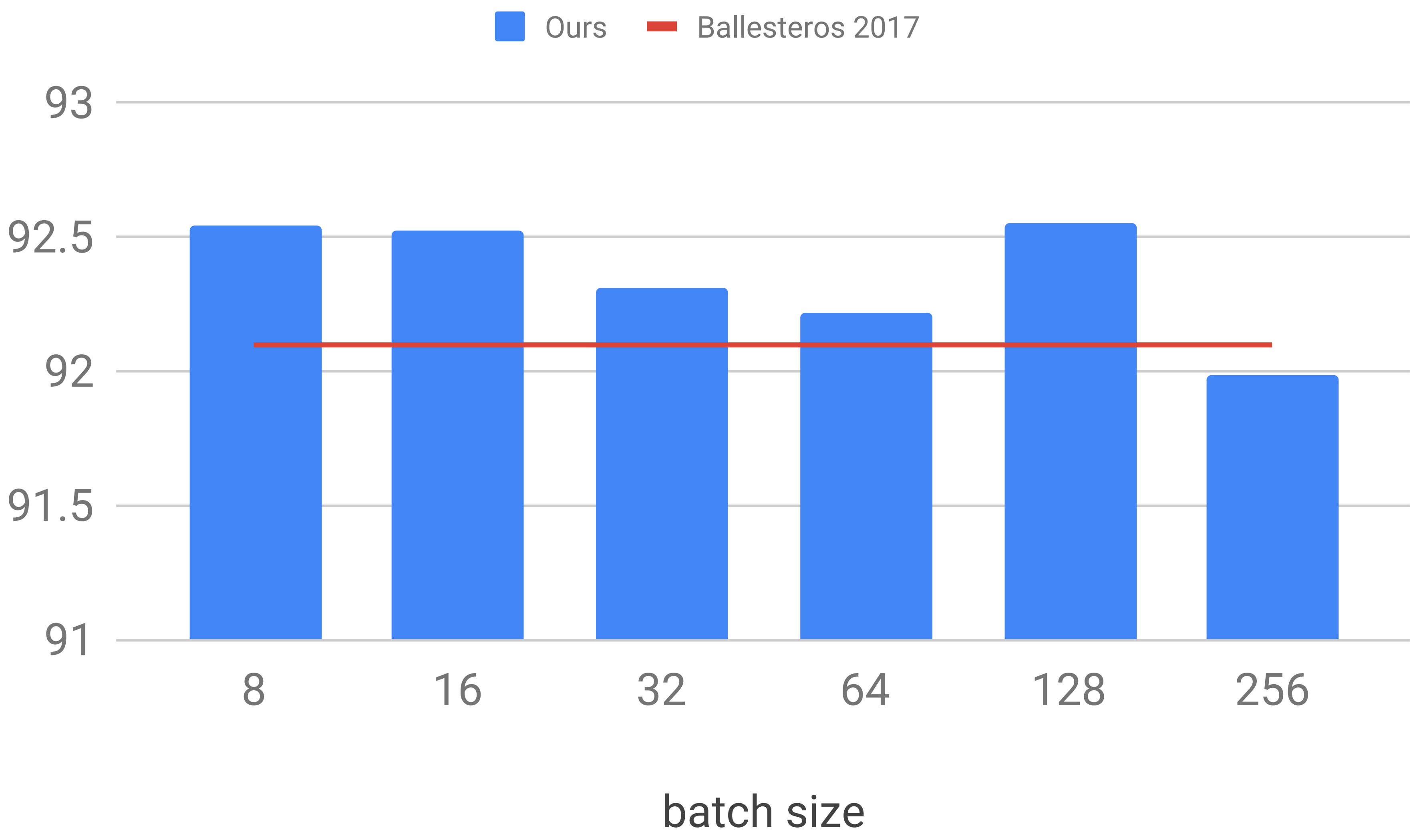
Speed


training speed
Dyer et al. 2015
sent/s
batch size
0
50
100
150
200
250
300
350
400
10^0
10^1
10^2

Speed


training speed
Dyer et al. 2015
sent/s
batch size
10^0
10^1
10^2
10^3
10^0
10^1
10^2

Performance



Conclusion

Conclusion

- We propose a parallelization scheme for StackLSTM architecture.
- Together with a different optimizer, we are able to train parsers of comparable performance within 1 hour.

paper
code
slides

```
https://github.com/shuoyangd/hoolock
```