Statistical Machine Translation LING-462/COSC-482 Week 9: Neural machine translation

Achim Ruopp

achim.ruopp@Georgetown.edu

Agenda

- Language in ten minutes: Yan Yang - French
- Neural machine translation with sequence-to-sequence models

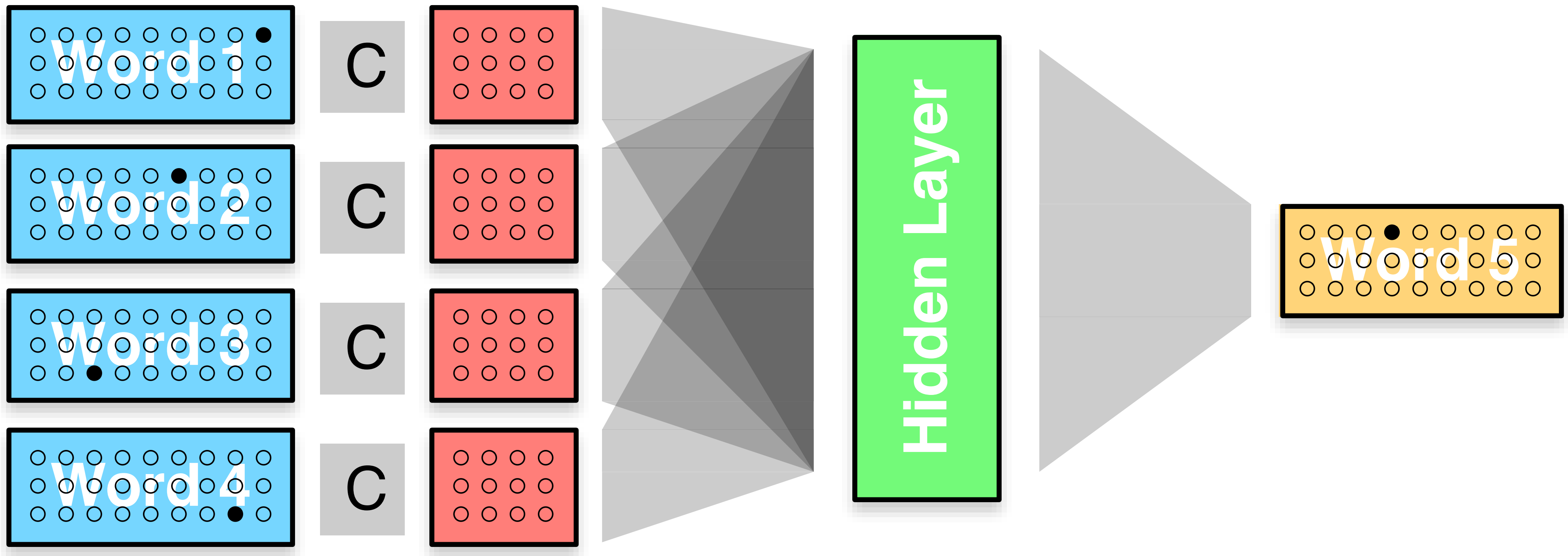
- Break -

- Neural machine translation by jointly learning to align and translate

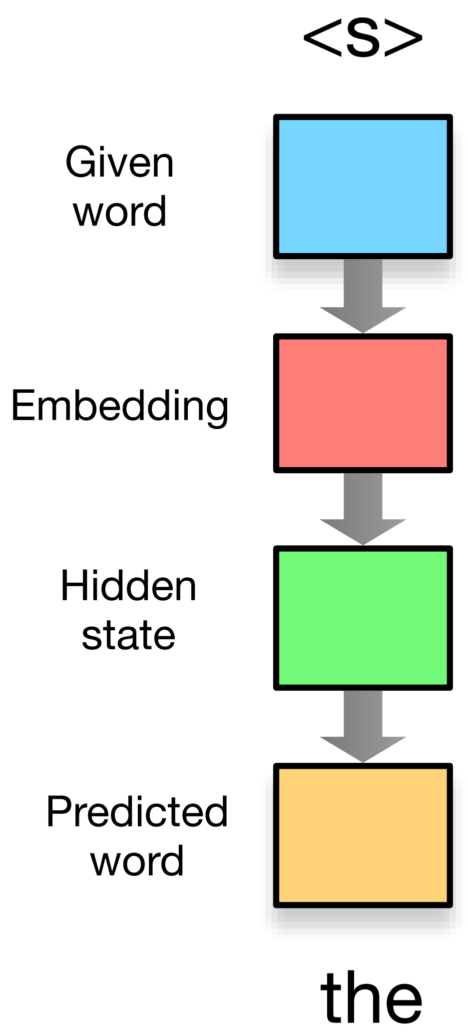
Language Models

- Modeling variants
  - feed-forward neural network
  - recurrent neural network
  - long short term memory neural network
- May include input context

Feed Forward Neural Language Model



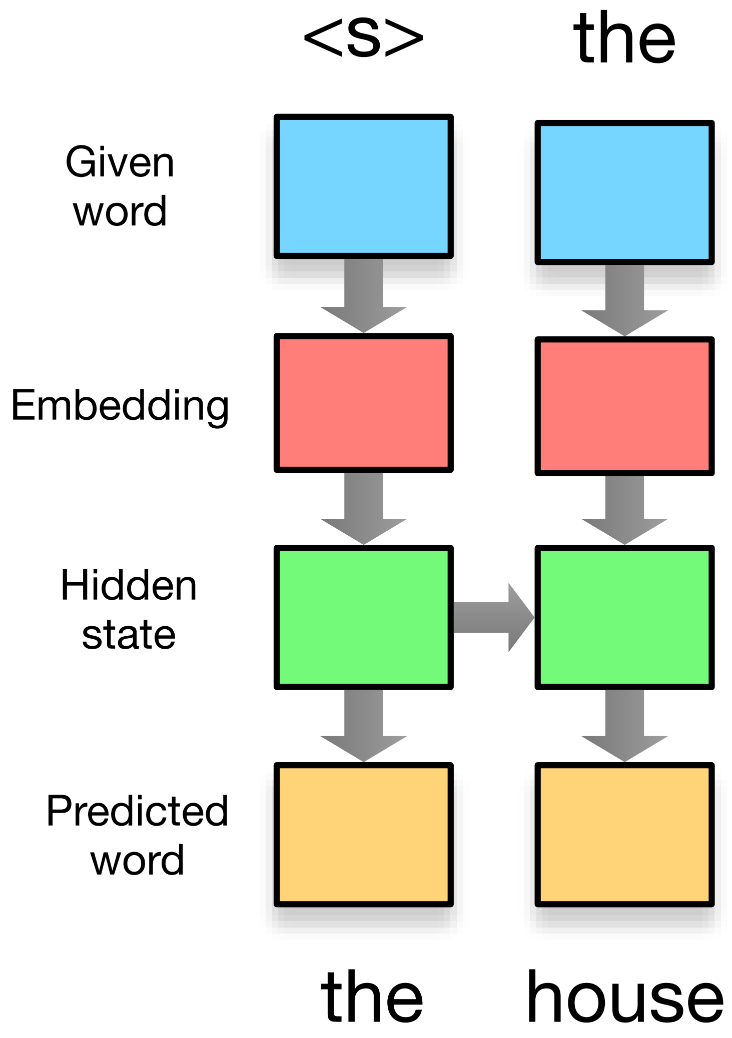
Recurrent Neural Language Model



Predict
the first word
of a sentence

Same as before,
just drawn top-down

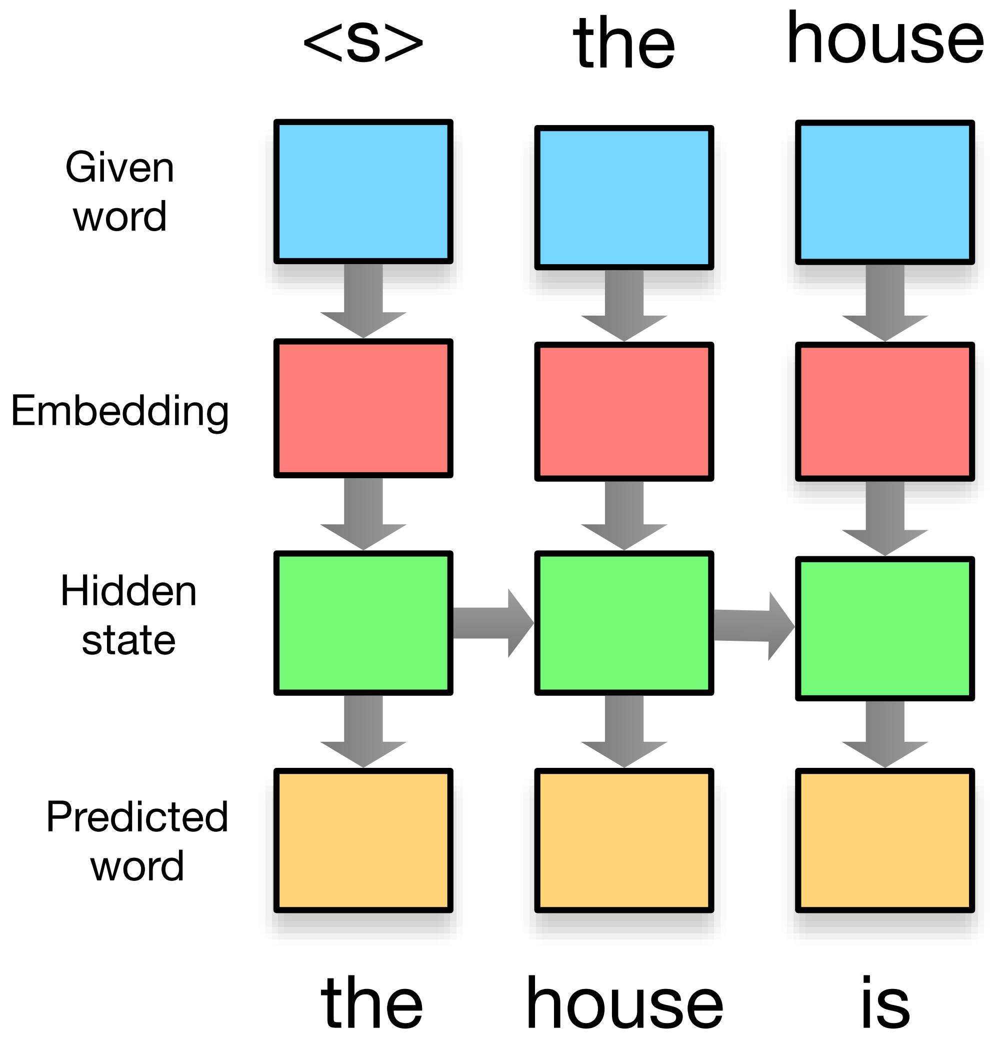
Recurrent Neural Language Model



Predict
the second word
of a sentence

Re-use hidden state
from
first word prediction

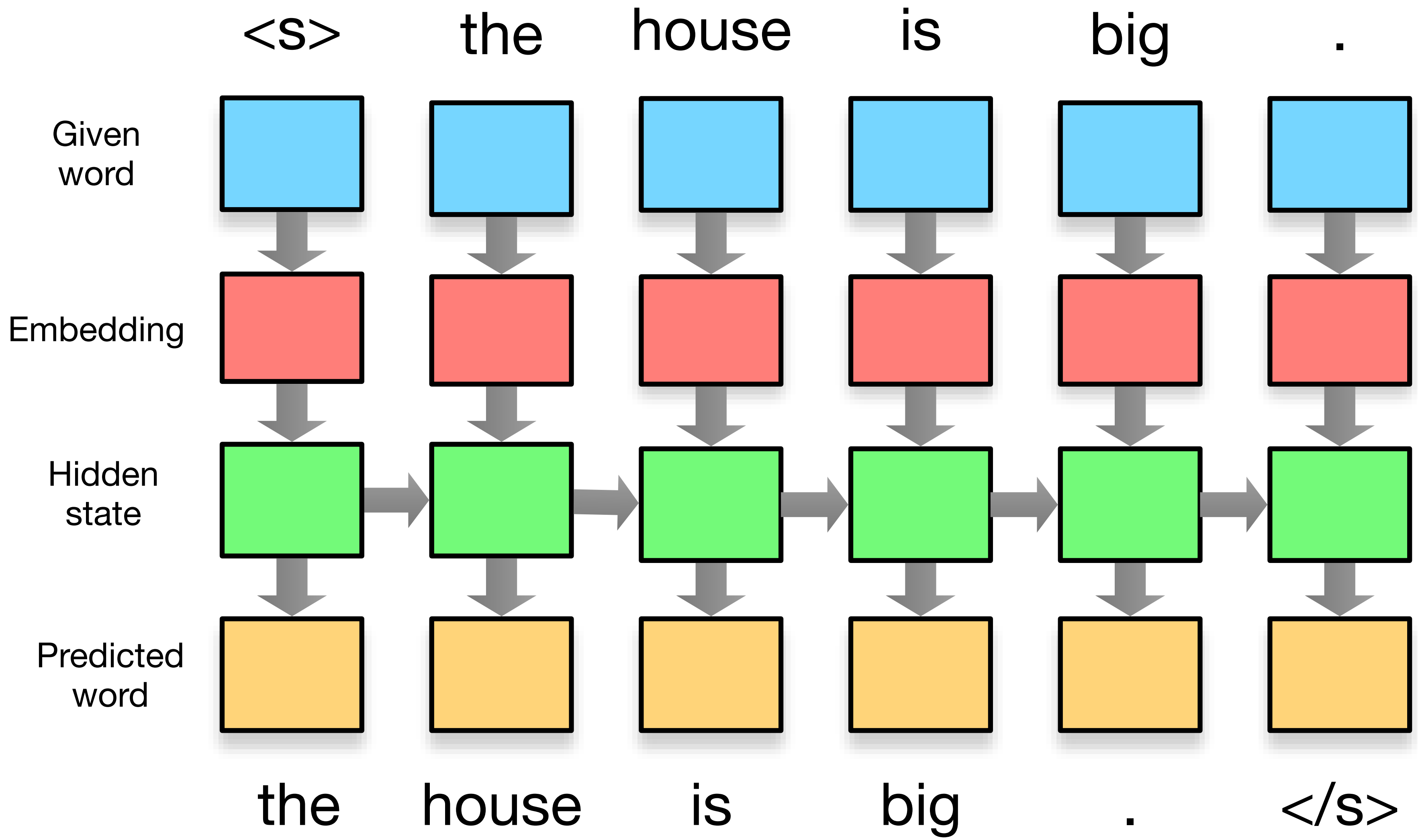
Recurrent Neural Language Model



Predict
the third word
of a sentence

... and so on

Recurrent Neural Language Model

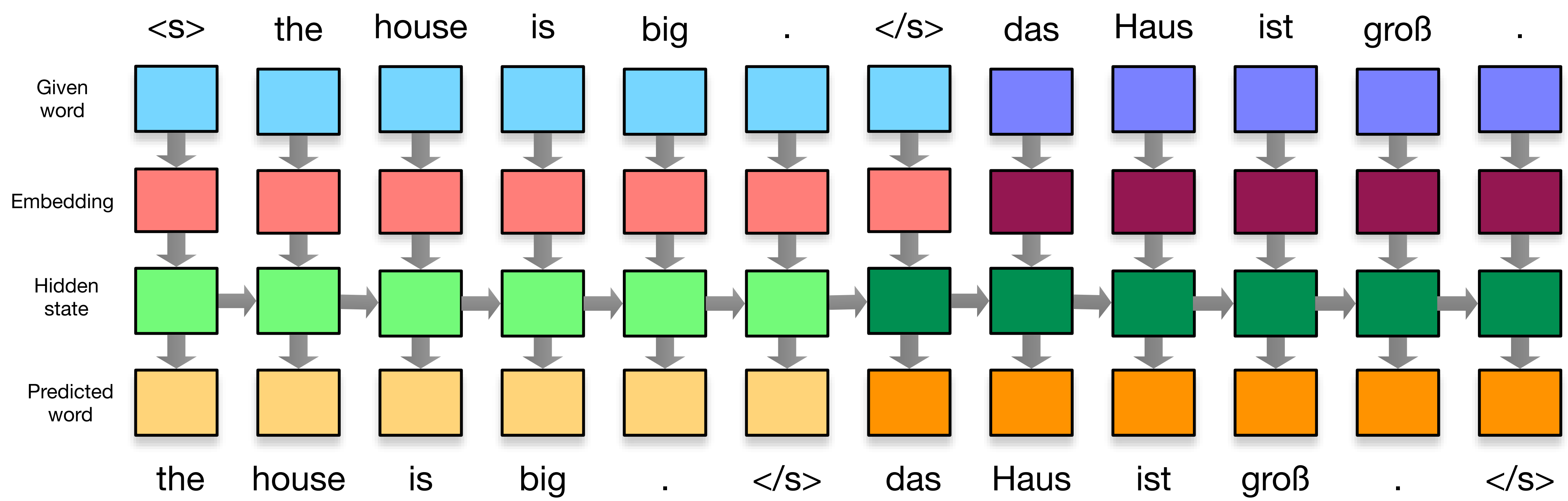


Recurrent Neural Translation Model

- We predicted the words of a sentence

- Why not also predict their translations?

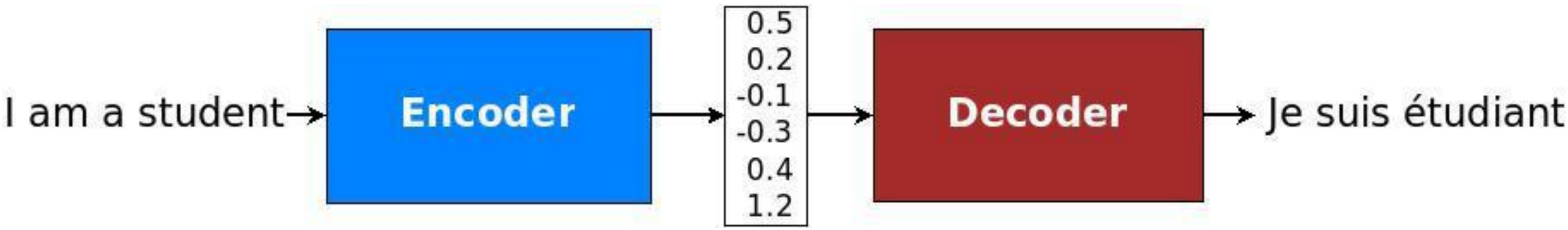
Encoder-Decoder Model



- Obviously madness

- Proposed by Google (Sutskever et al. 2014)

Encoder-decoder architecture

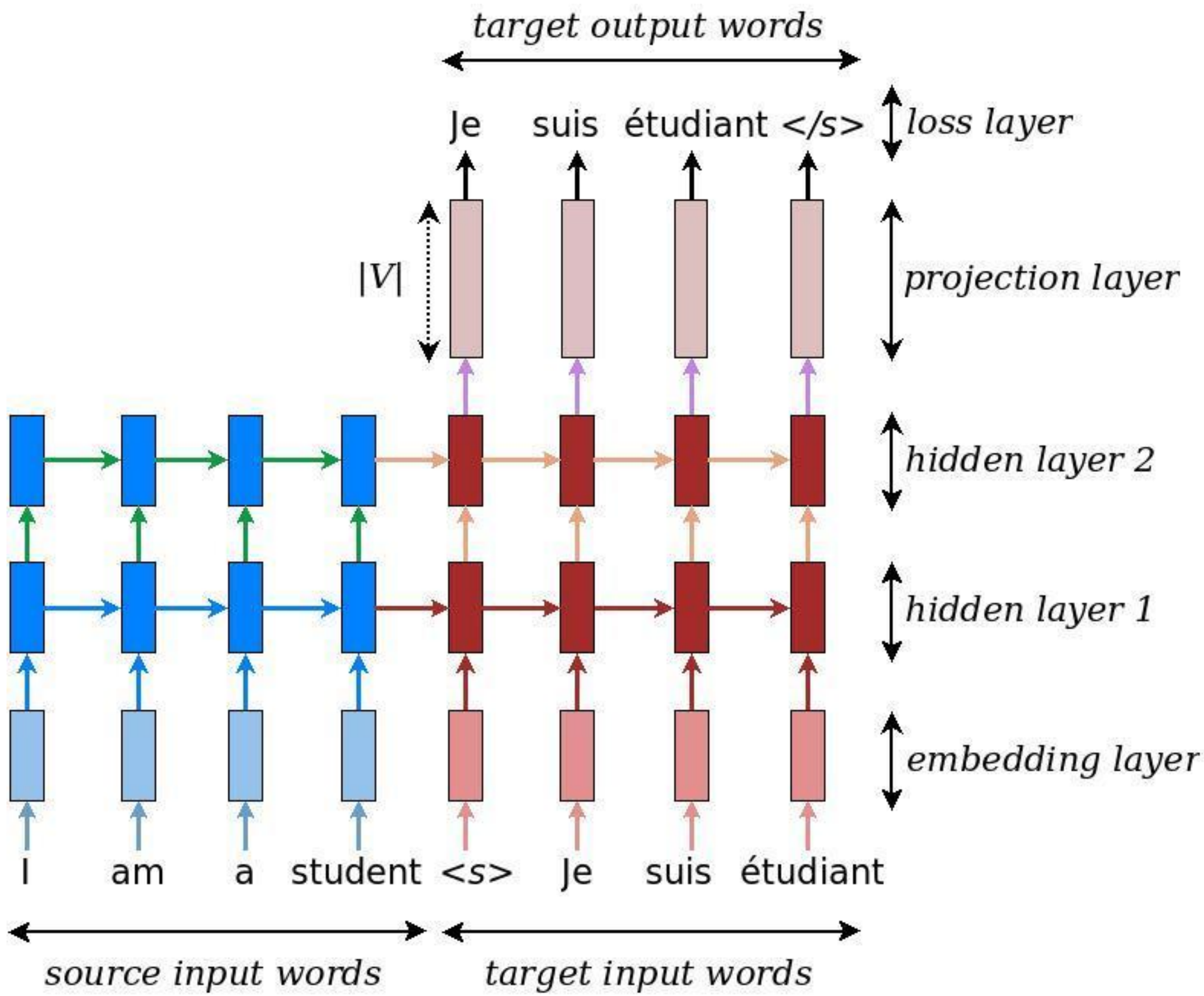


- Converting a source sentence into a “meaning” vector
- Decoding the “meaning” vector into a target sentence
- Addresses local translation problem
  - Long distance dependencies
  - Syntactic structures
  - Agreement
  - Fluency
  - …

Encoder-decoder RNNs

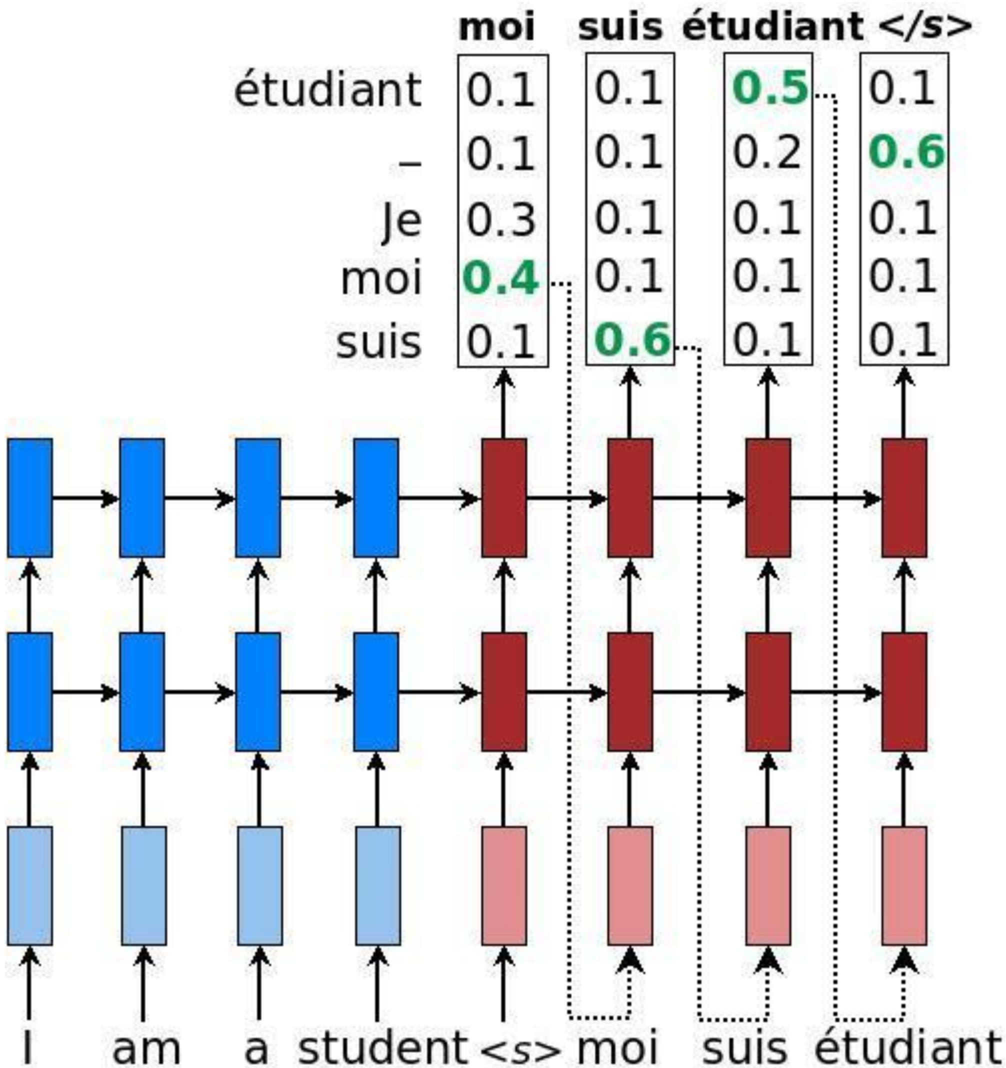
- Can vary in
- Directionality
  - Unidirectional vs bidirectional
- Type
  - SimpleRNN
  - LSTM
  - GRU
- Depth

Deep multi-layer RNN with LSTM



Google Neural Machine Translation (seq2seq) Tutorial

Decoding with Seq2Seq NMT



Google Neural Machine Translation (seq2seq) Tutorial

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

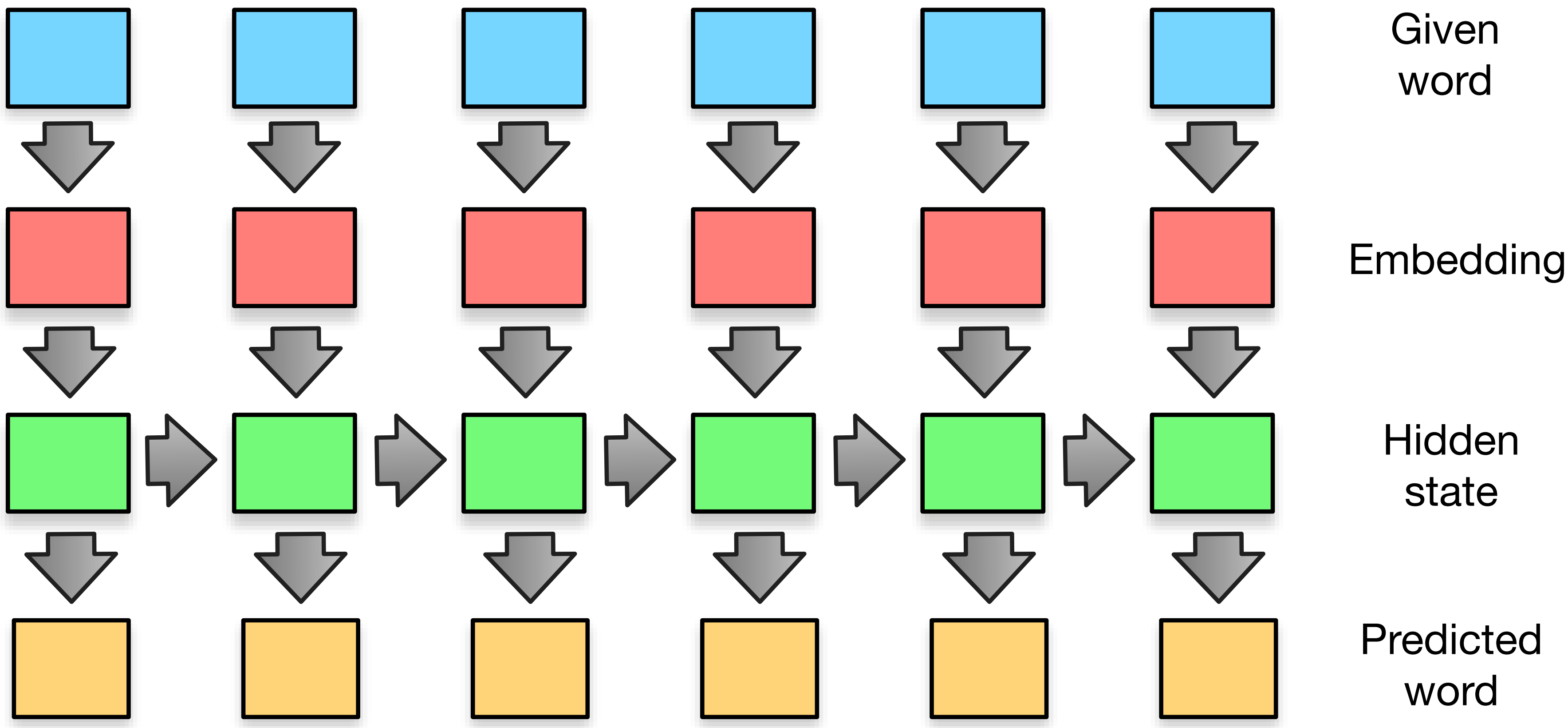
What is missing?

- Alignment of input words to output words

$\Rightarrow$ Solution: attention mechanism

neural translation model with attention

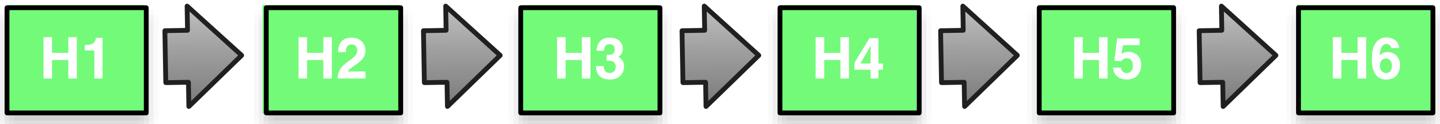
Input Encoding



- Inspiration: recurrent neural network language model on the input side

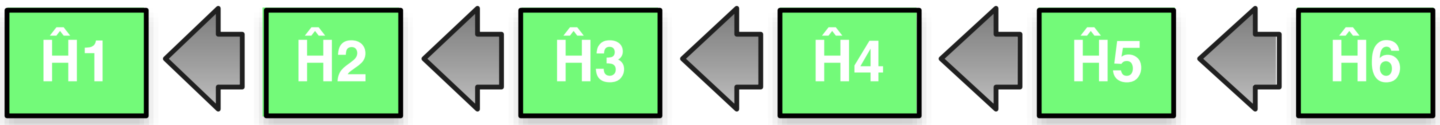
Hidden Language Model States

- This gives us the hidden states

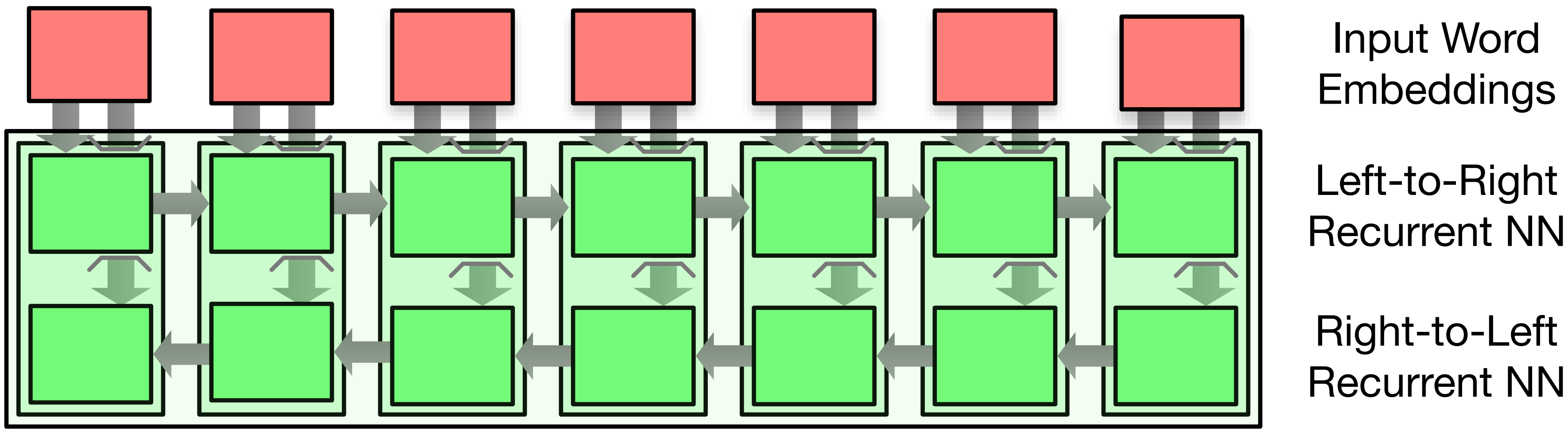


- These encode left context for each word

- Same process in reverse: right context for each word

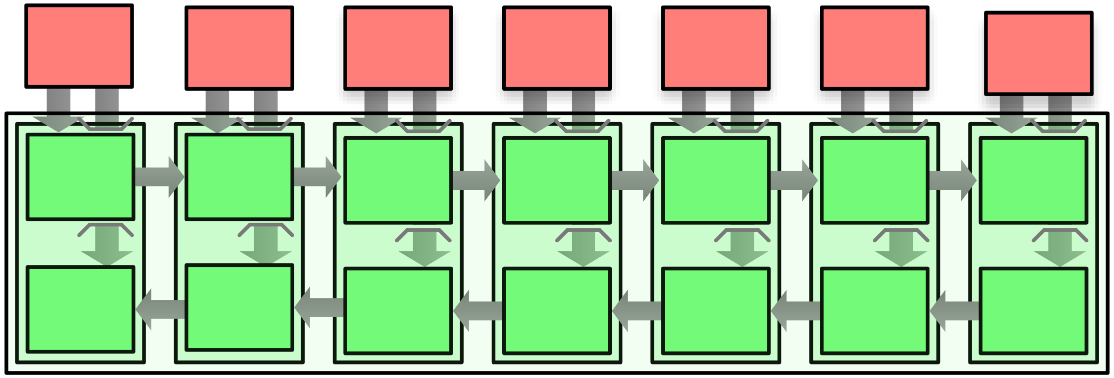


Input Encoder



- Input encoder: concatenate bidirectional RNN states
- Each word representation includes full left and right sentence context

Encoder: Math



- Input is sequence of words x_j, mapped into embedding space $\bar{E}\ x_j$

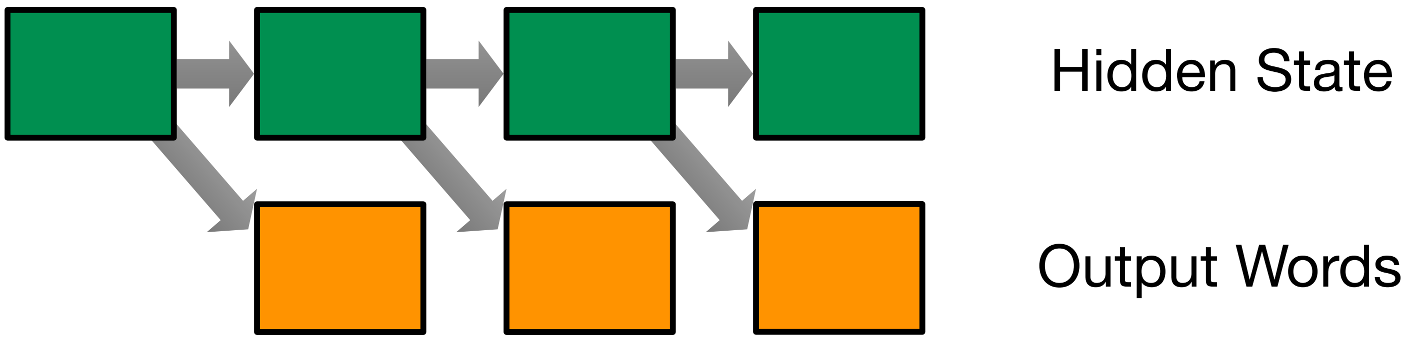
- Bidirectional recurrent neural networks

$$\overleftarrow{h_j} = f(\overleftarrow{h_{j+1}}, \bar{E}\ x_j)$$
$$\overrightarrow{h_j} = f(\overrightarrow{h_{j-1}}, \bar{E}\ x_j)$$

- Various choices for the function $f()$: feed-forward layer, GRU, LSTM, ...

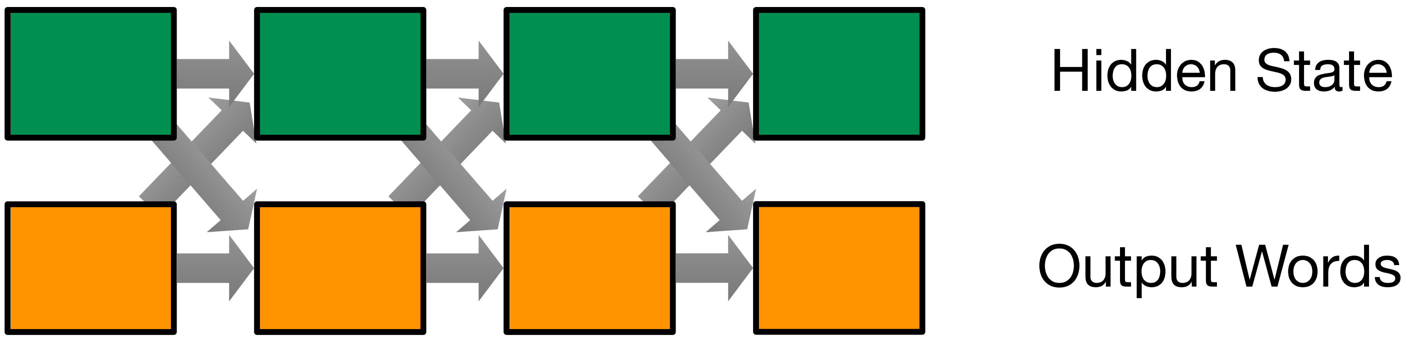
Decoder

- We want to have a recurrent neural network predicting output words



Decoder

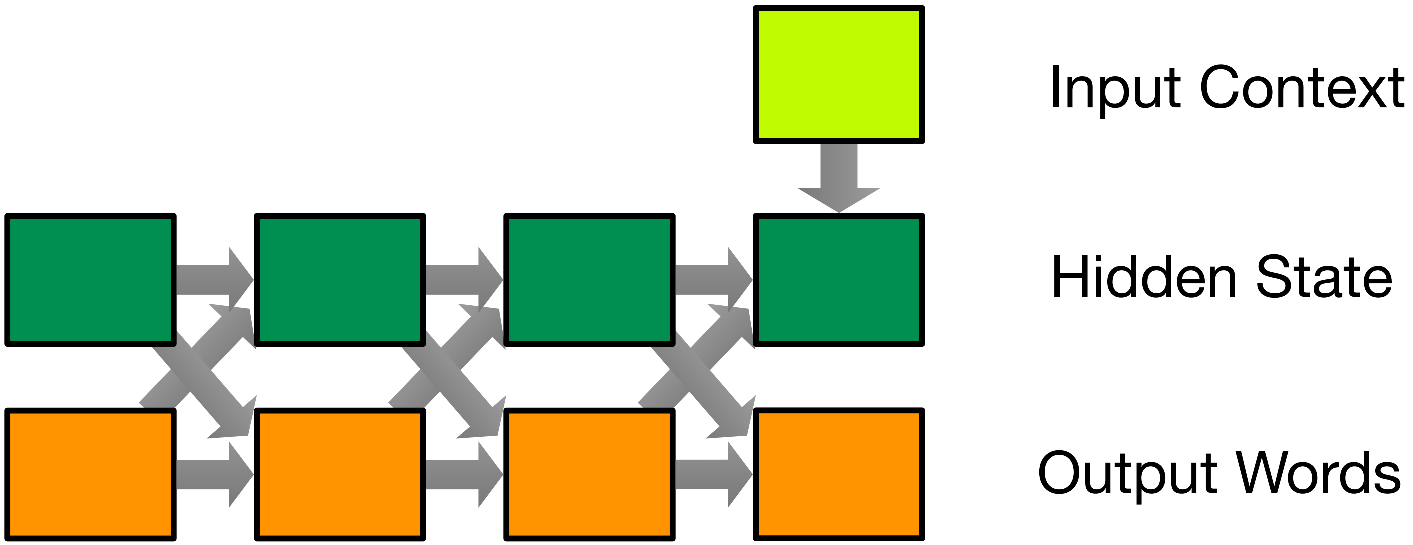
- We want to have a recurrent neural network predicting output words



- We feed decisions on output words back into the decoder state

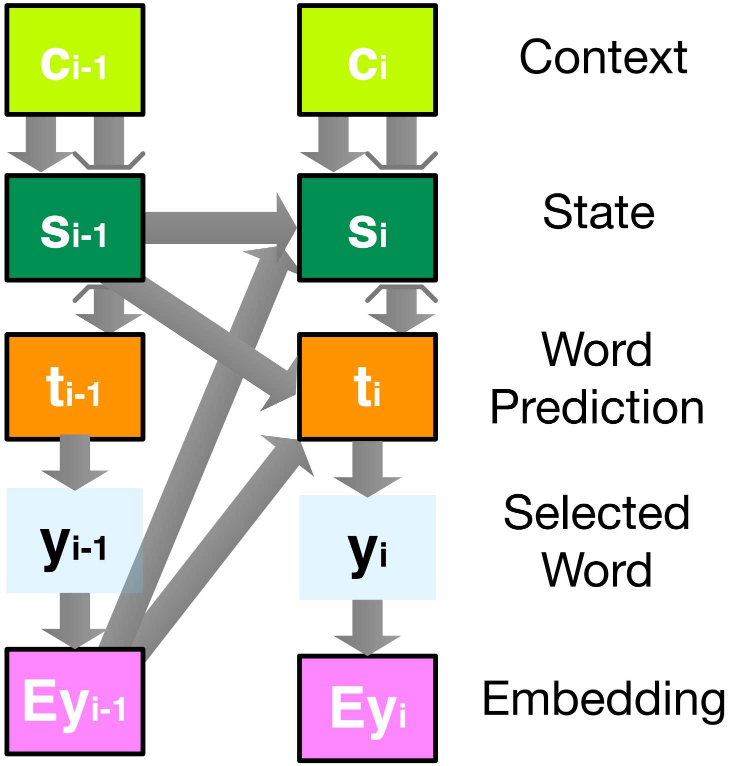
Decoder

- We want to have a recurrent neural network predicting output words



- We feed decisions on output words back into the decoder state
- Decoder state is also informed by the input context

More Detail



- Decoder is also recurrent neural network over sequence of hidden states s_i

$$s_i = f(s_{i-1},\ Ey_{-1}, c_i)$$

- Again, various choices for the function $f()$: feed-forward layer, GRU, LSTM, ...

- Output word y_i is selected by computing a vector t_i (same size as vocabulary)

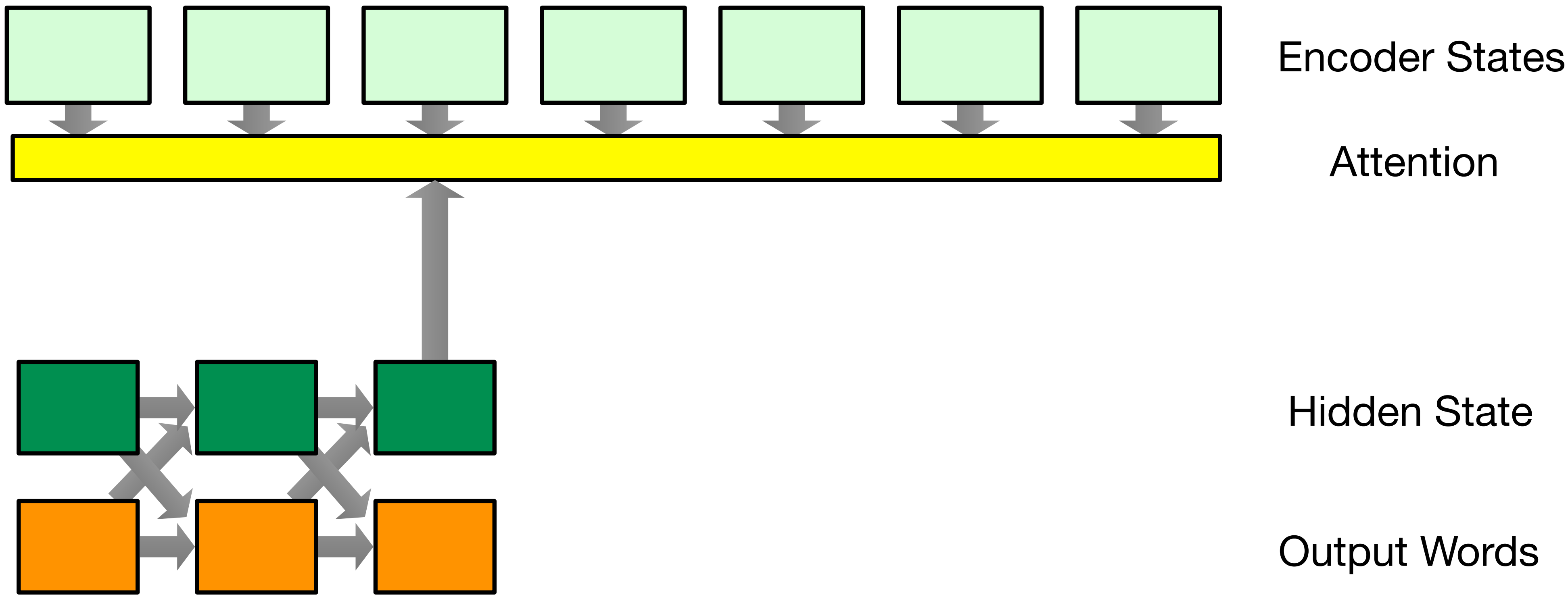
$$t_i = W(Us_{i-1} + VEy_{i-1} + Cc_i)$$

then finding the highest value in vector t_i

- If we normalize t_i, we can view it as a probability distribution over words

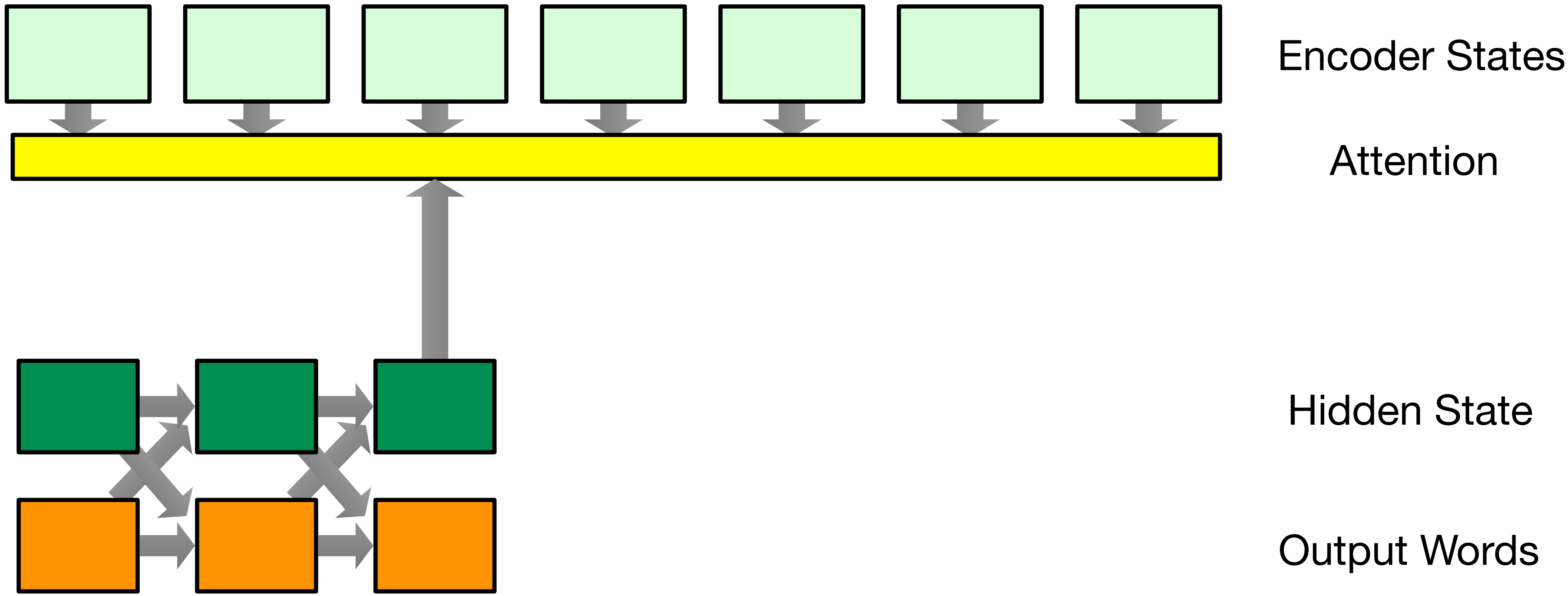
- Ey_i is the embedding of the output word y_i

Attention



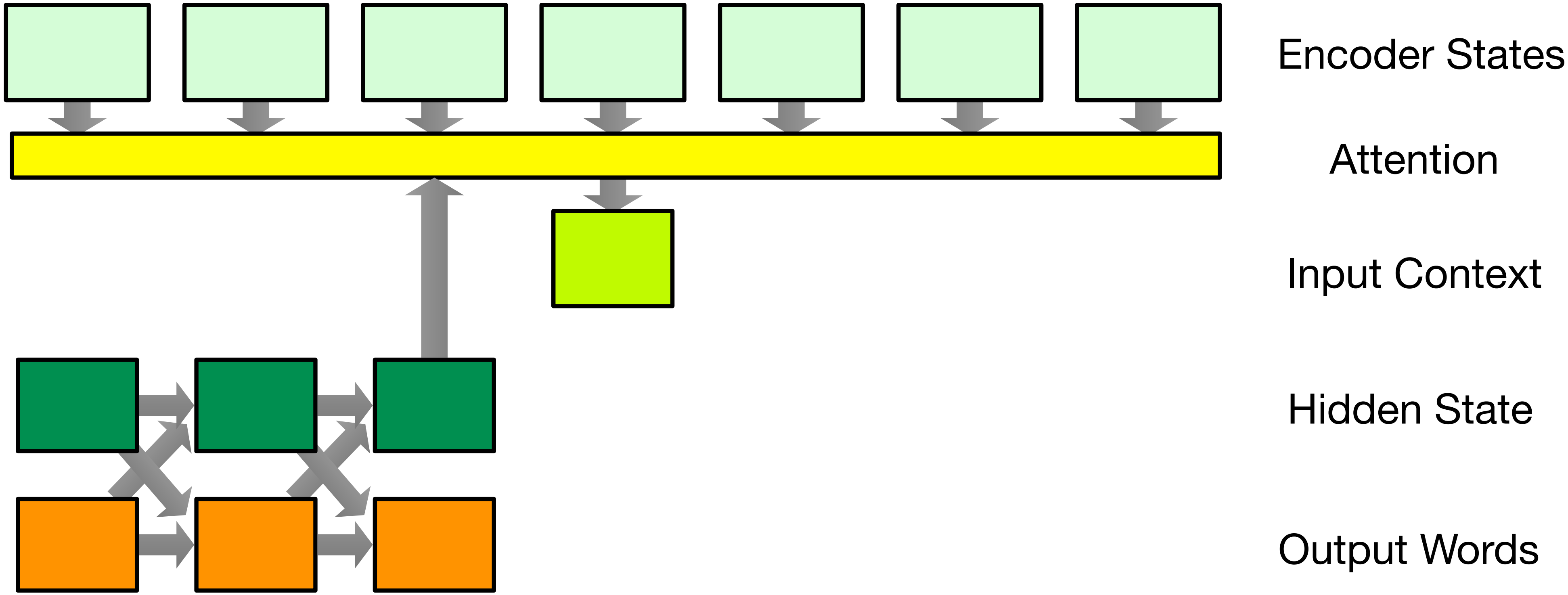
- Given what we have generated so far (decoder hidden state)
- ... which words in the input should we pay attention to (encoder states)?

Attention



- Given: – the previous hidden state of the decoder s_{i-1}
  – the representation of input words $h_j = (\overleftarrow{h_j}, \overrightarrow{h_j})$
- Predict an alignment probability $a(s_{i-1}, h_j)$ to each input word j (modeled with with a feed-forward neural network layer)

Attention

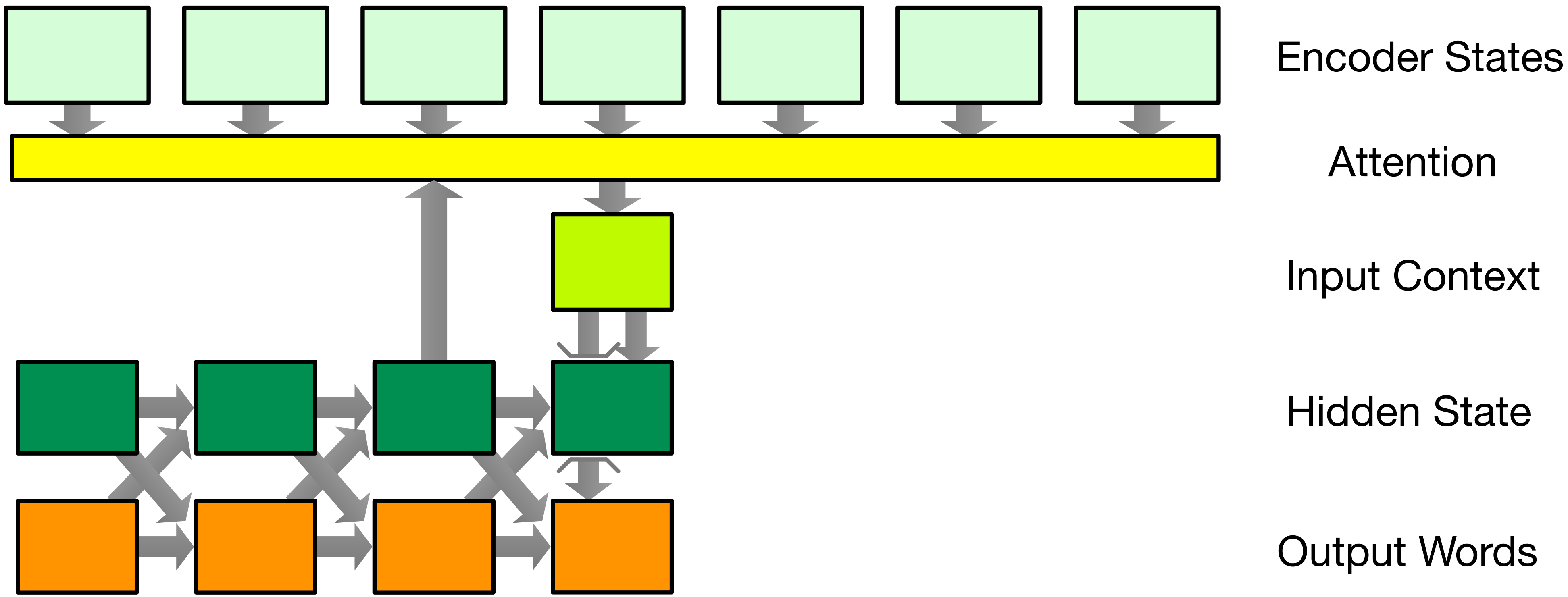


- Normalize attention (softmax)

$$\alpha_{ij} = \frac{\exp(a(s_{i-1}, h_j))}{\sum_k \exp(a(s_{i-1}, h_k))}$$

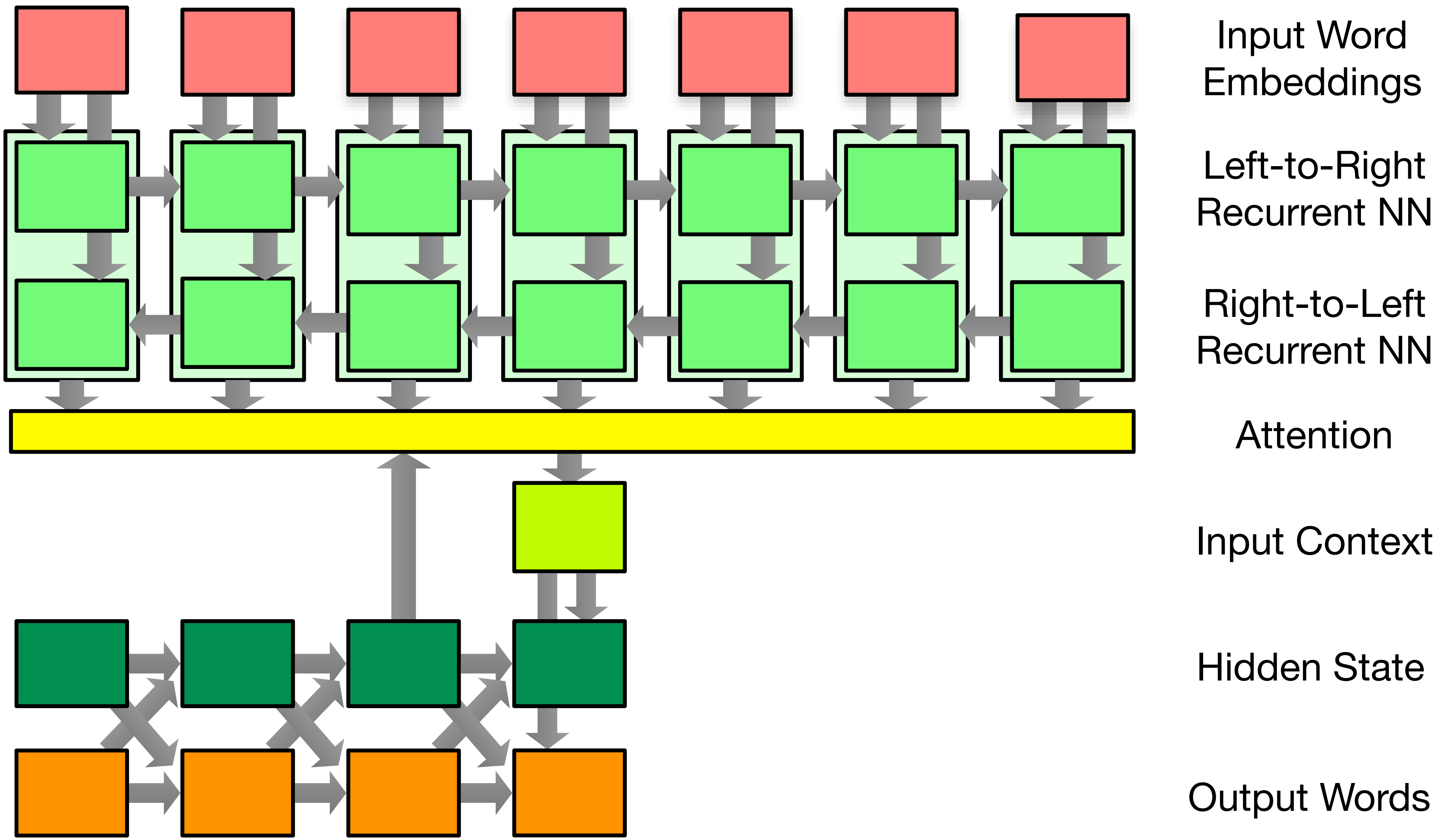
- Relevant input context: weigh input words according to attention: $c_i = \sum_j \alpha_{ij} h_j$

Attention



- Use context to predict next hidden state and output word

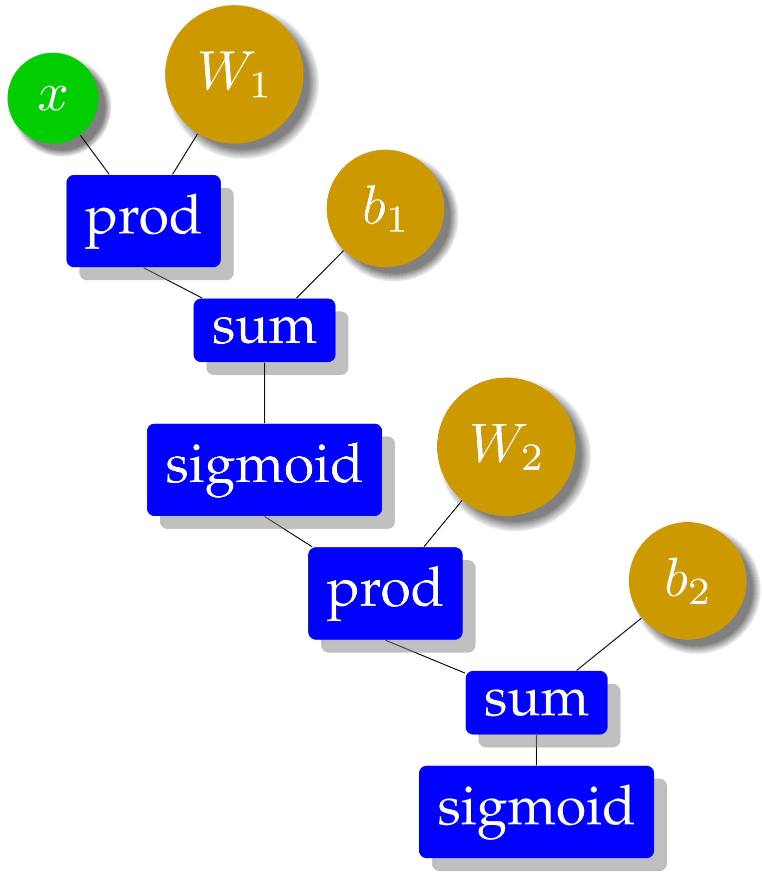
Encoder-Decoder with Attention



training

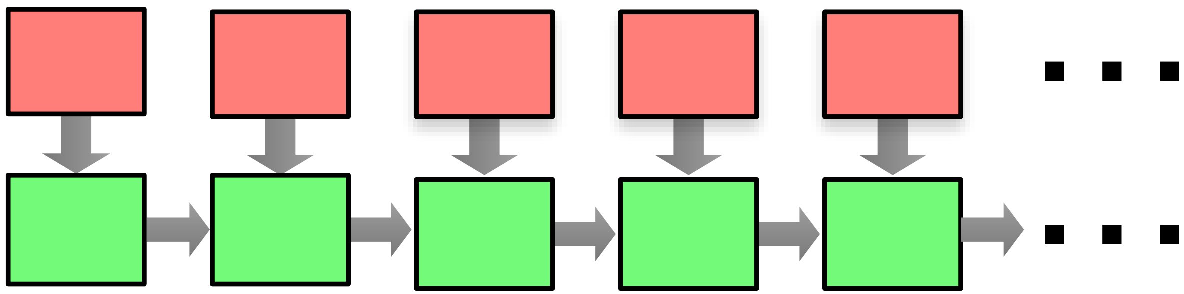
Computation Graph

- Math behind neural machine translation defines a computation graph
- Forward and backward computation to compute gradients for model training



Problem: Recurrent Neural Networks

- RNNs imply dynamically sized graph

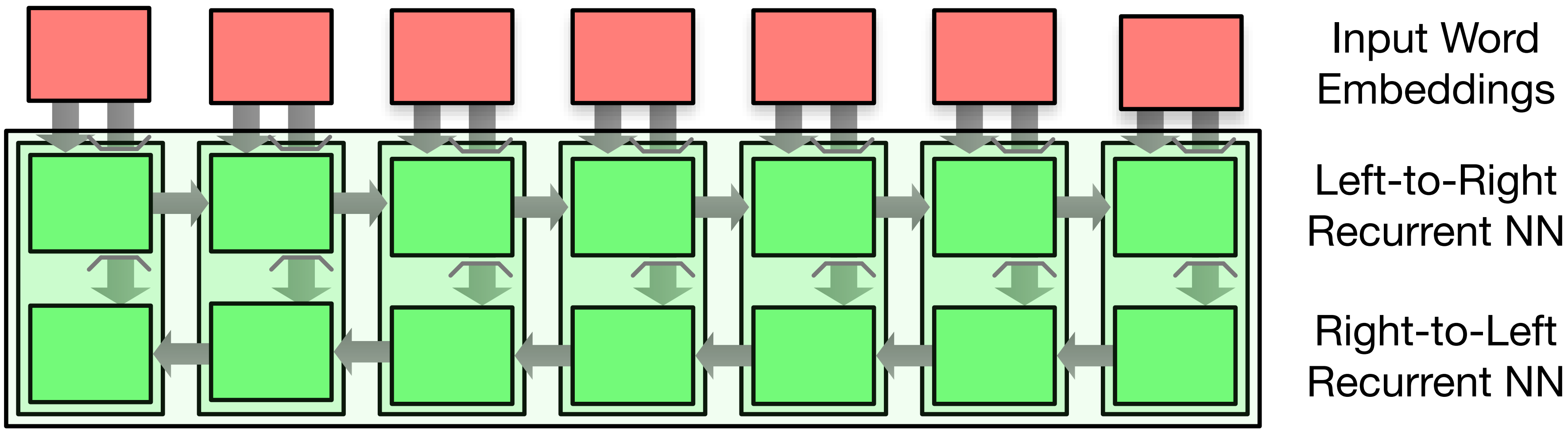


- Size of graph depends on length, of input and output sentence

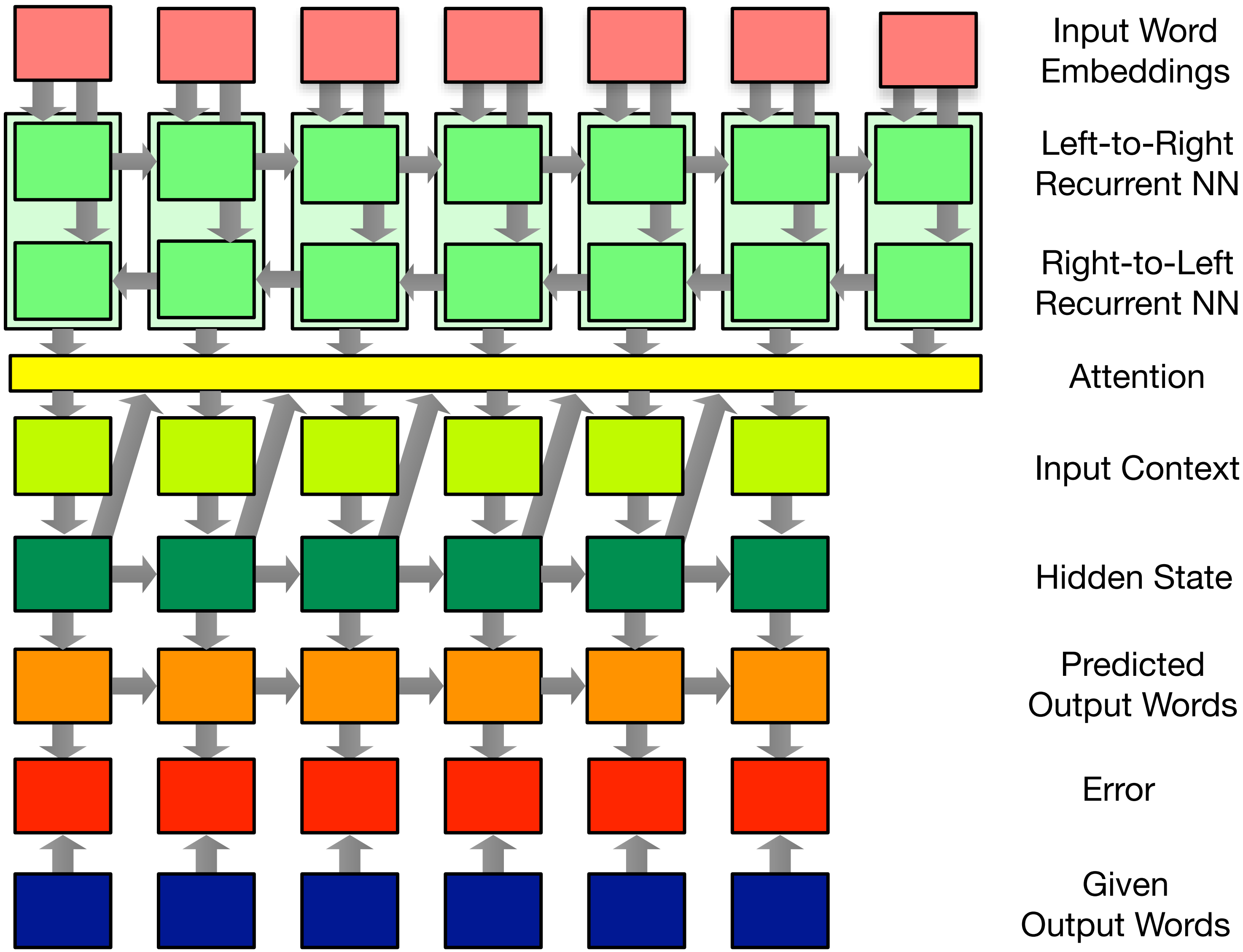
Unrolling RNNs

- For a given training example, length of input and output sentence known

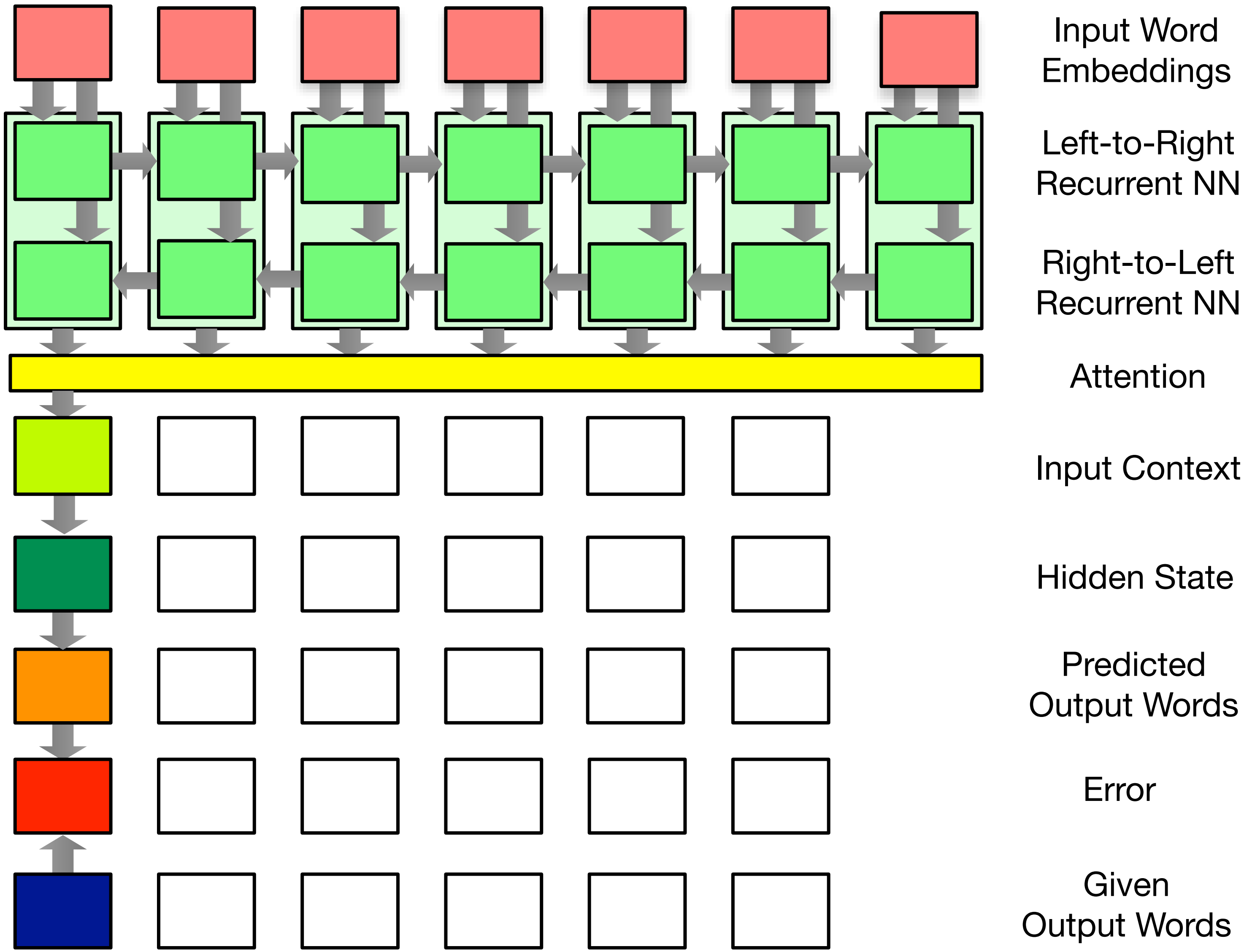
$\Rightarrow$ Build out the entire computation graph



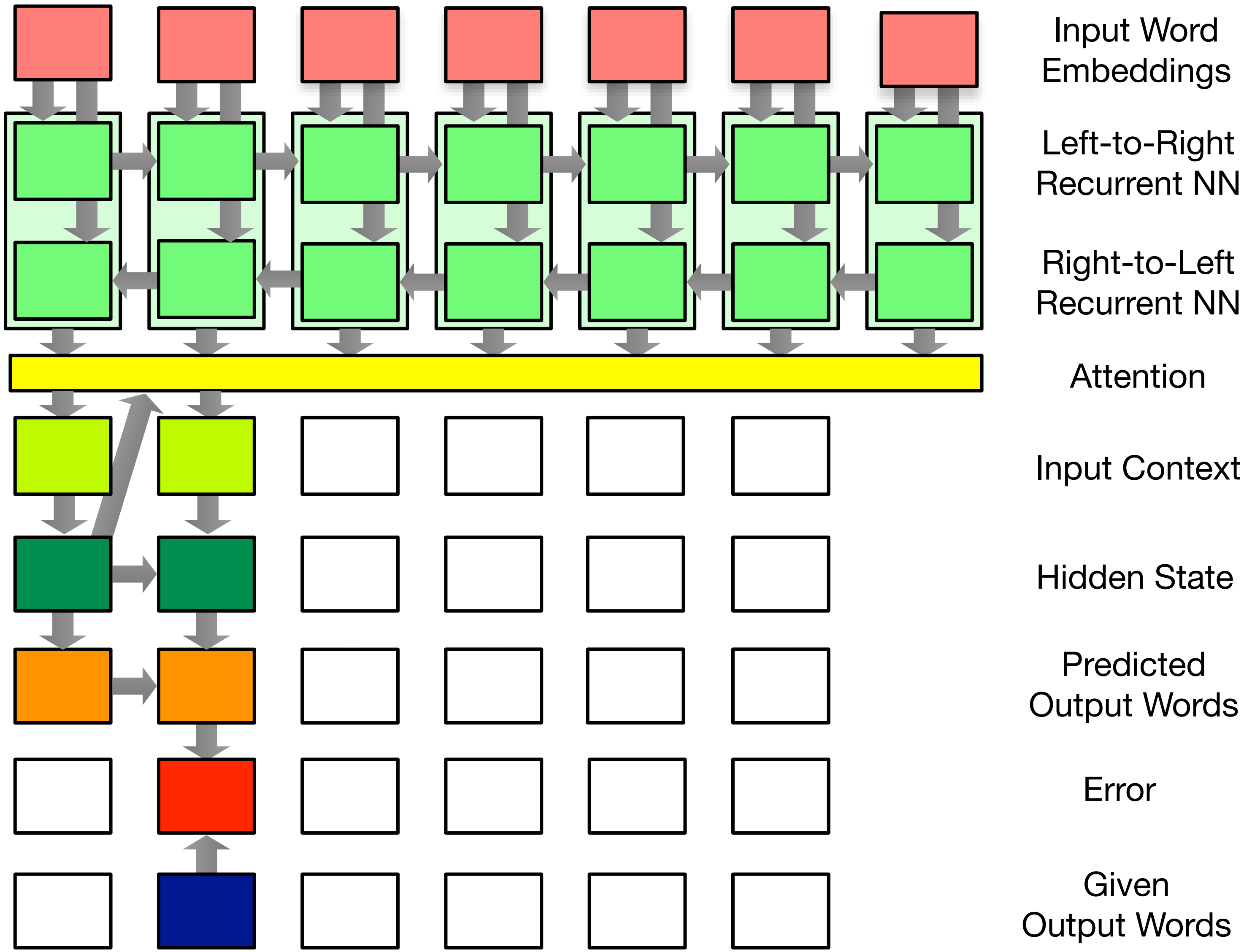
Fully Computed Graph



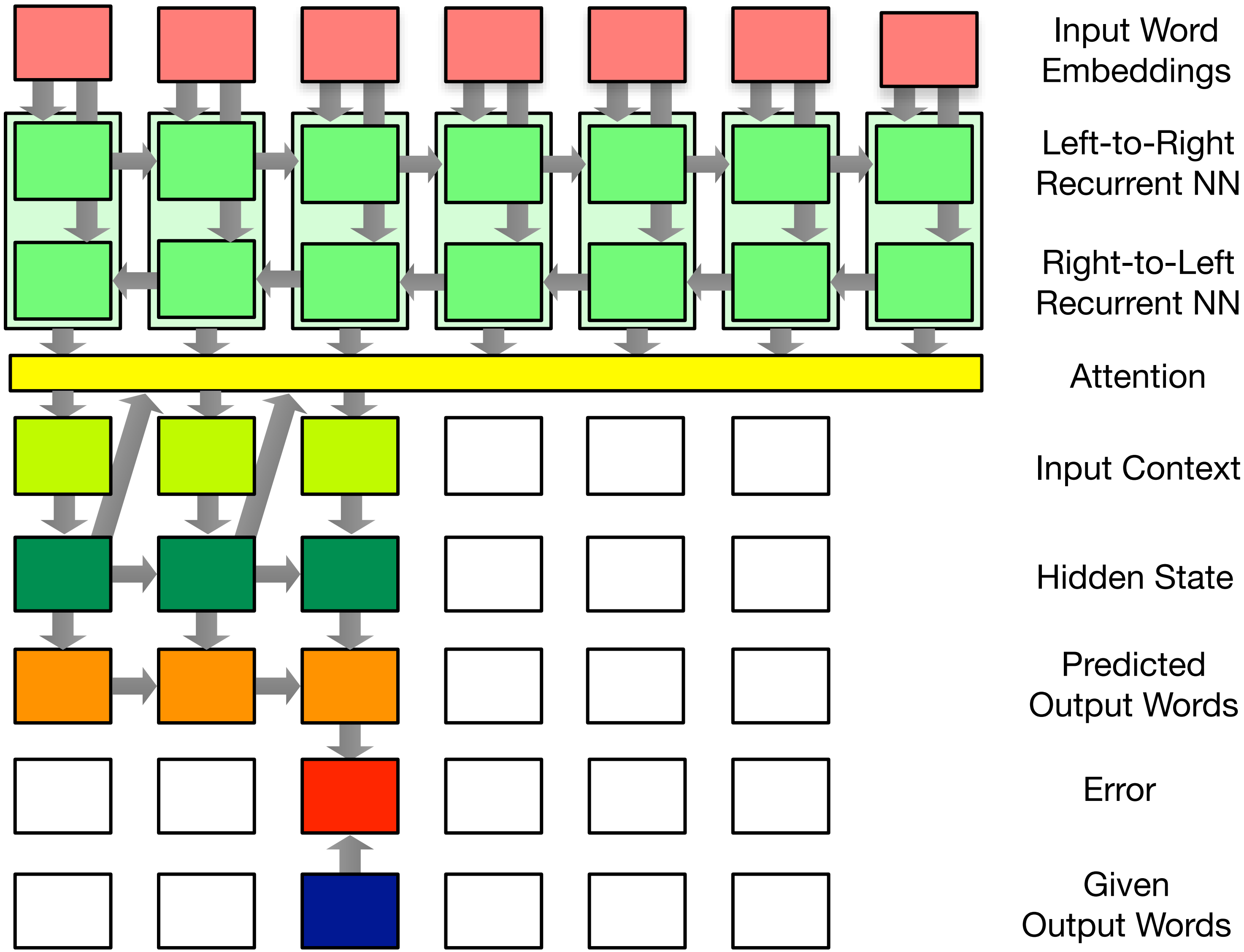
Update from Word 1



Update from Word 2



Update from Word 3

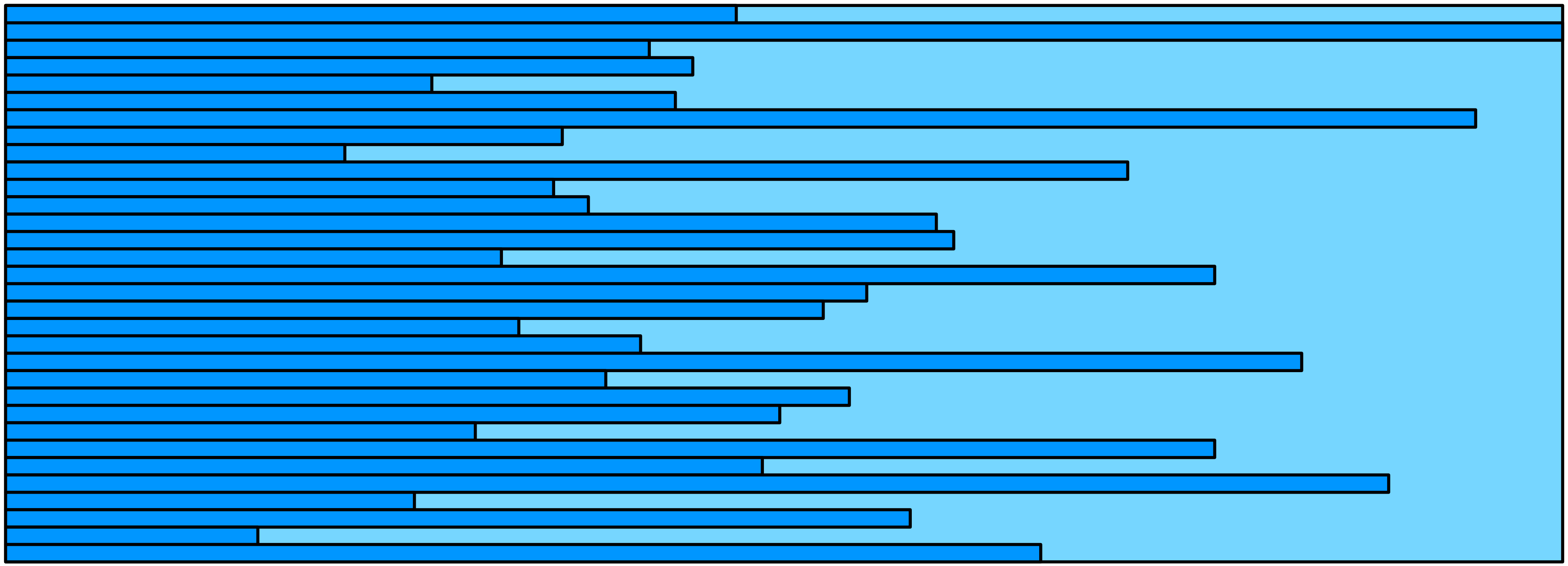


Batching

- Already large degree of parallelism
  - most computations on vectors, matrices
  - efficient implementations for CPU and GPU
- Further parallelism by batching
  - processing several sentence pairs at once
  - scalar operation $\rightarrow$ vector operation
  - vector operation $\rightarrow$ matrix operation
  - matrix operation $\rightarrow$ 3d tensor operation
- Typical batch sizes 50–100 sentence pairs

Batches

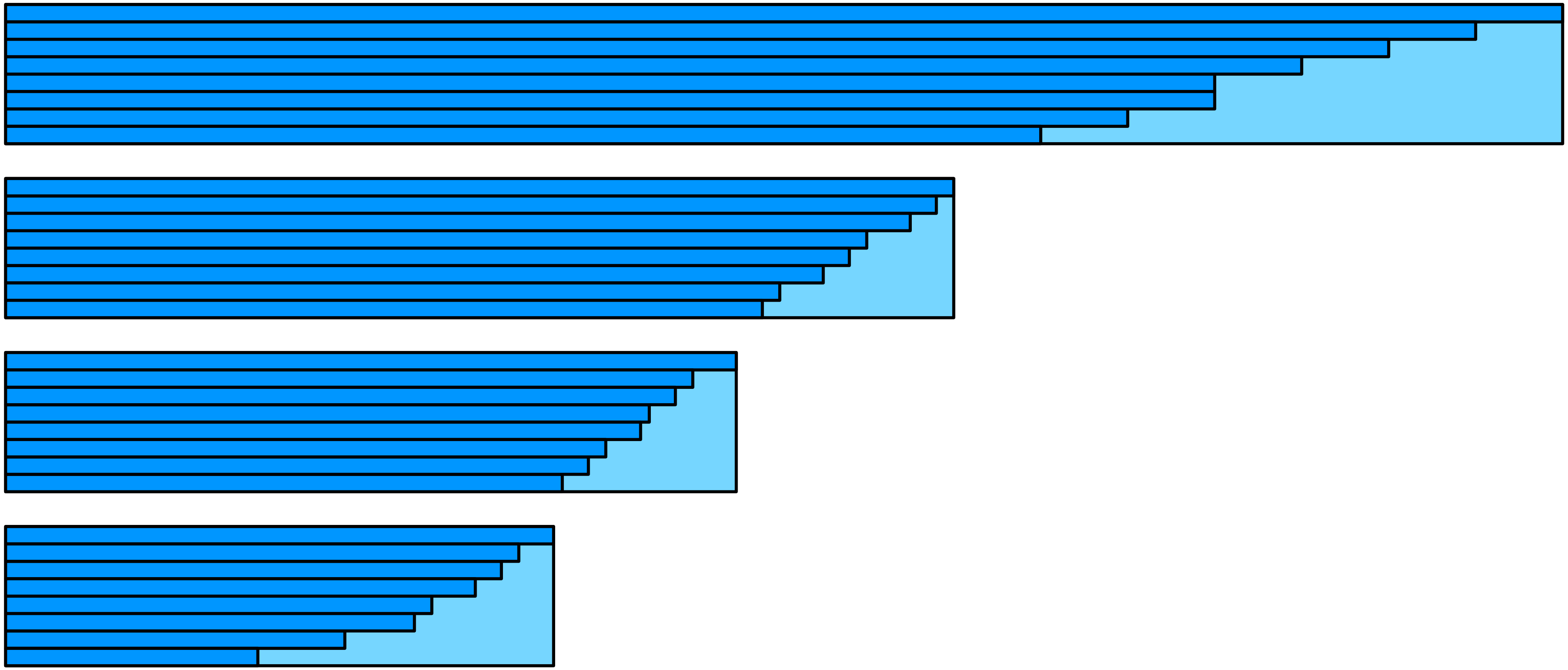
- Sentences have different length
- When batching, fill up unneeded cells in tensors



$\Rightarrow$ A lot of wasted computations

Mini-Batches

- Sort sentences by length, break up into mini-batches



- Example: Maxi-batch 1600 sentence pairs, mini-batch 80 sentence pairs

Overall Organization of Training

- Shuffle corpus
- Break into maxi-batches
- Break up each maxi-batch into mini-batches
- Process mini-batch, update parameters
- Once done, repeat
- Typically 5-15 epochs needed (passes through entire training corpus)

inference

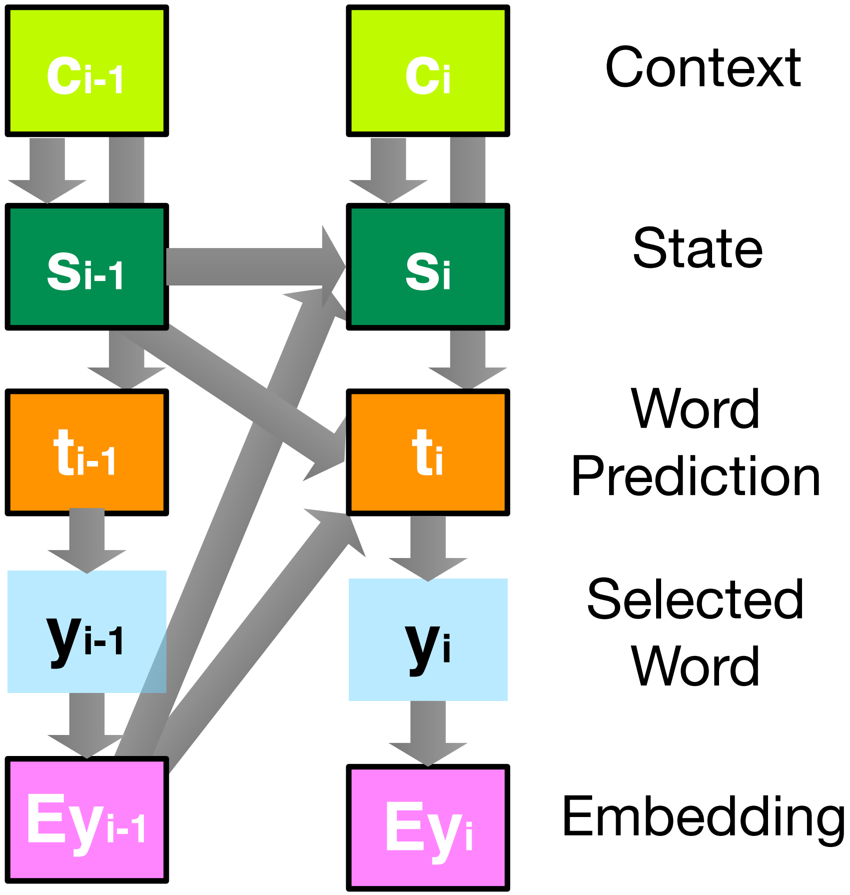
Inference

- Given a trained model

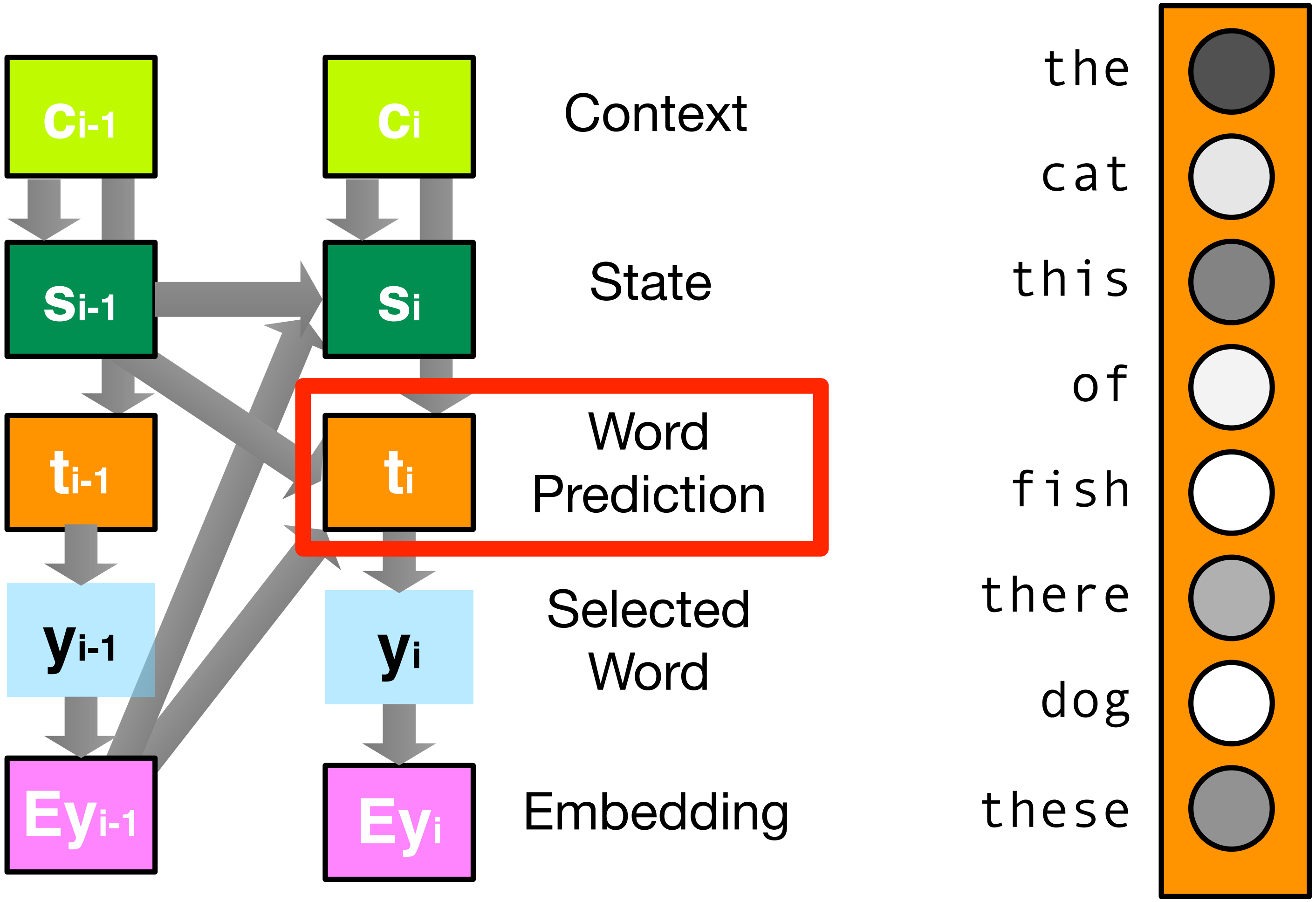
  ... we now want to translate test sentences

- We only need execute the ”forward” step in the computation graph

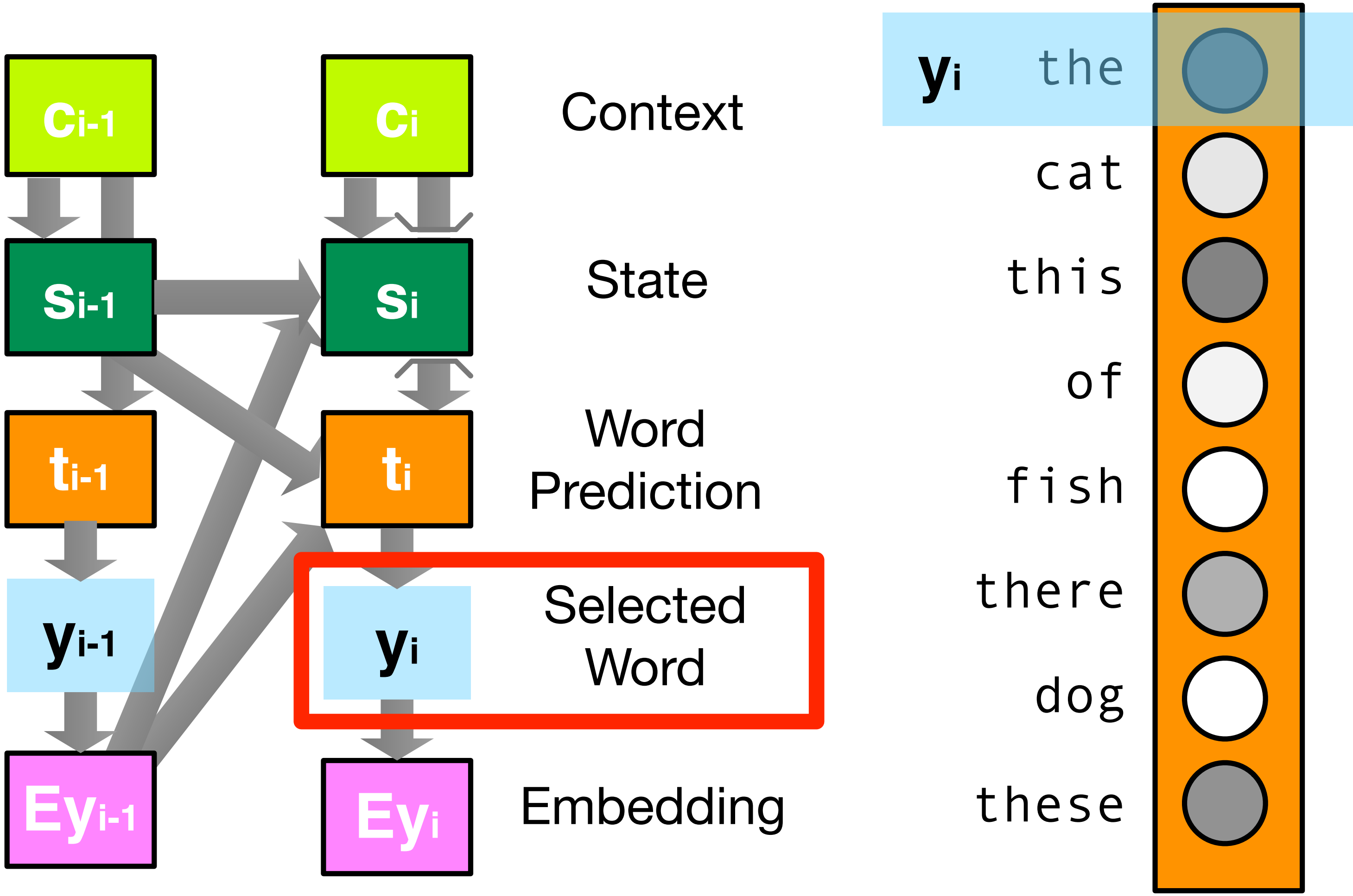
Word Prediction



Selected Word

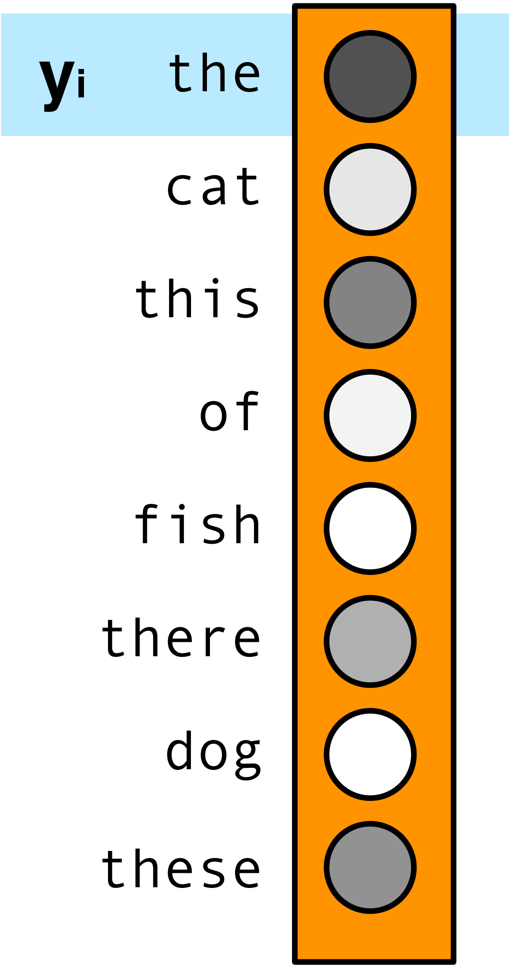


Embedding

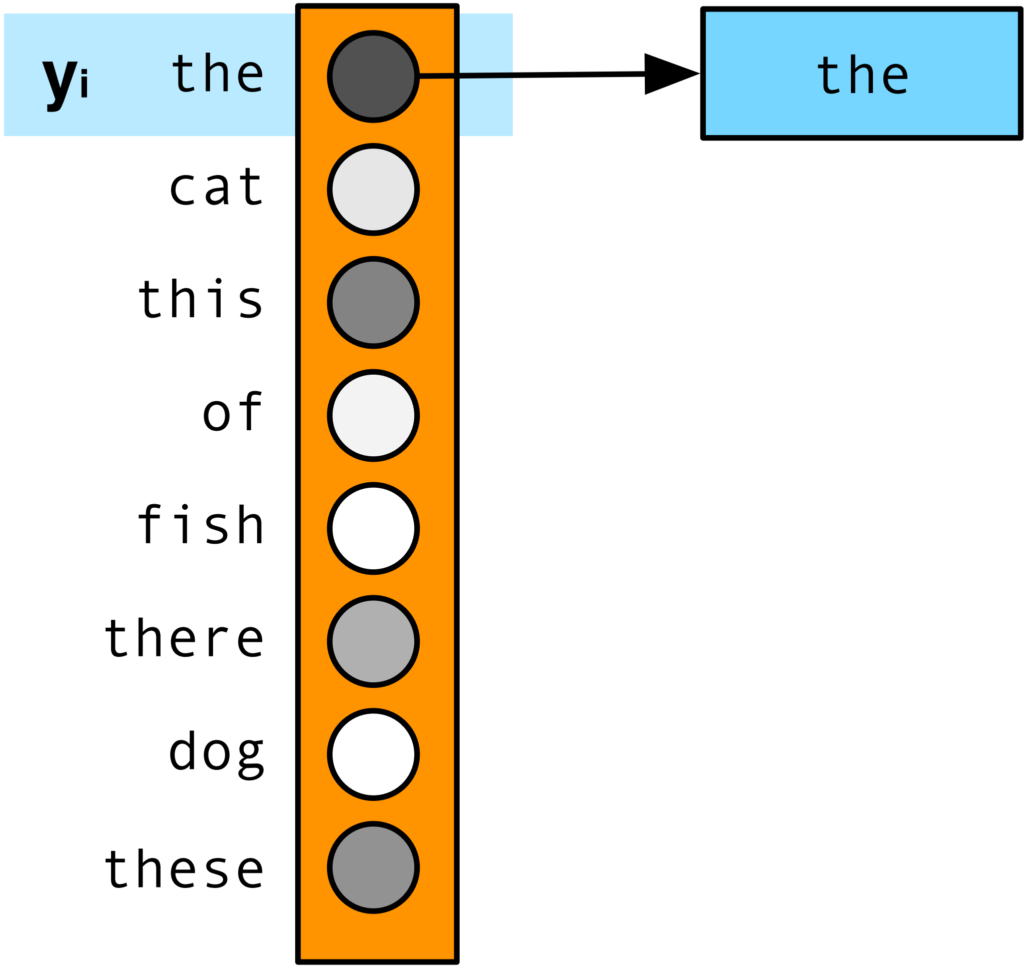


ci-1
ci
Context
si-1
si
State
ti-1
ti
Word
Prediction
yi-1
yi
Selected
Word
Eyi-1
Eyi
Embedding
yi
the
cat
this
of
fish
there
dog
these

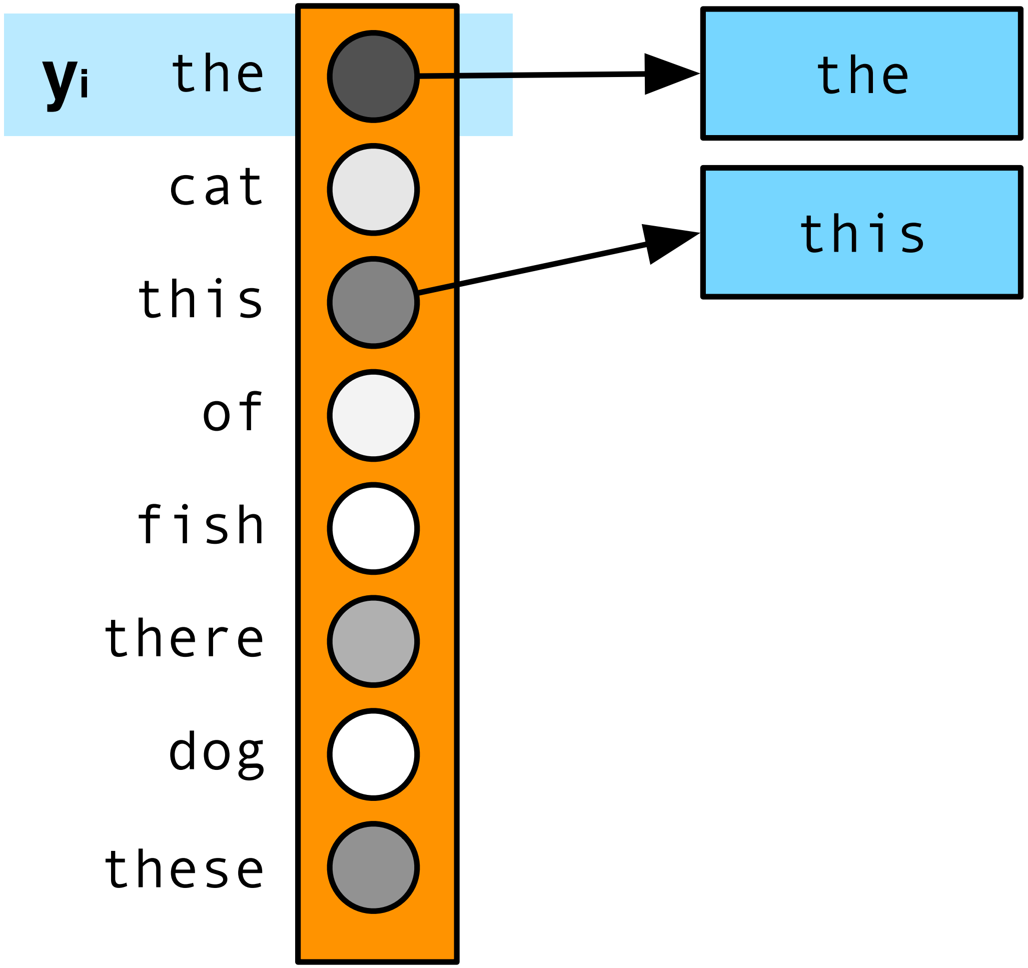
Distribution of Word Predictions



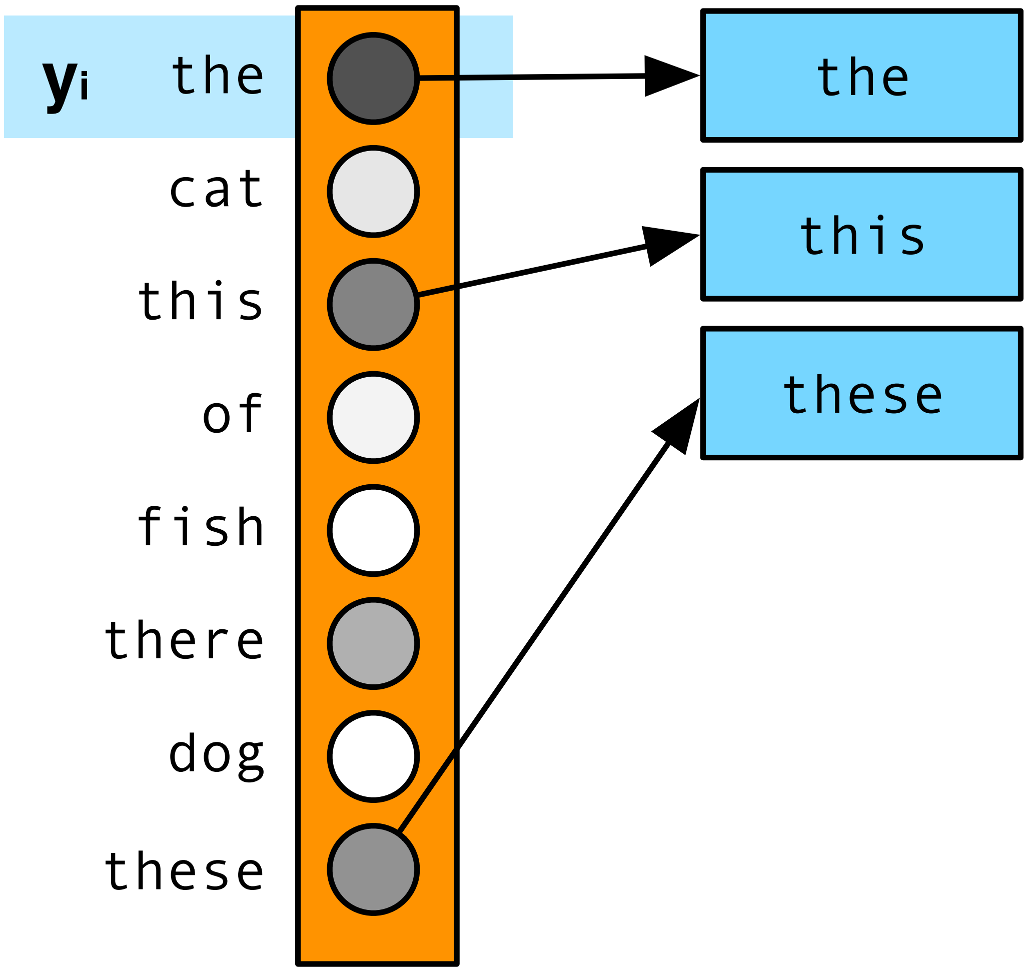
Select Best Word



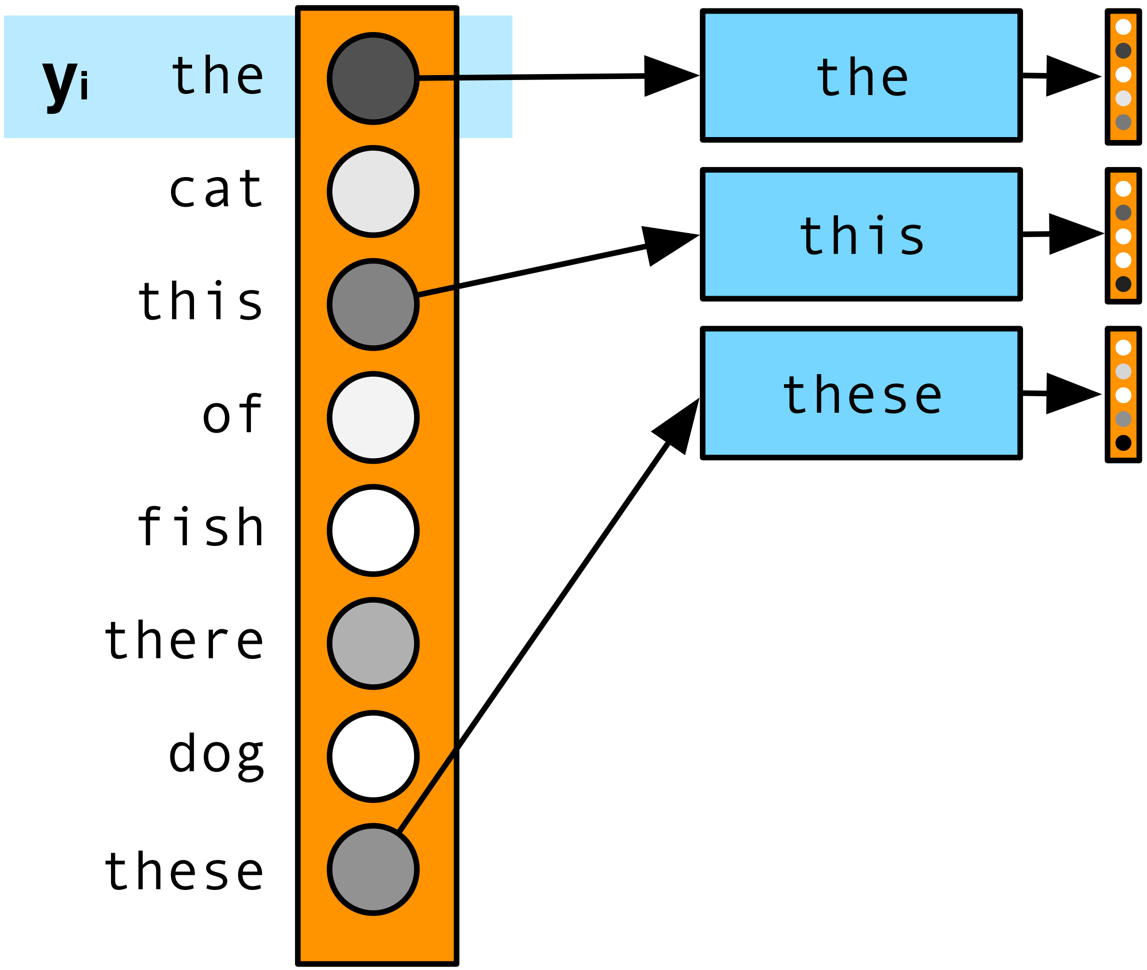
Select Second Best Word



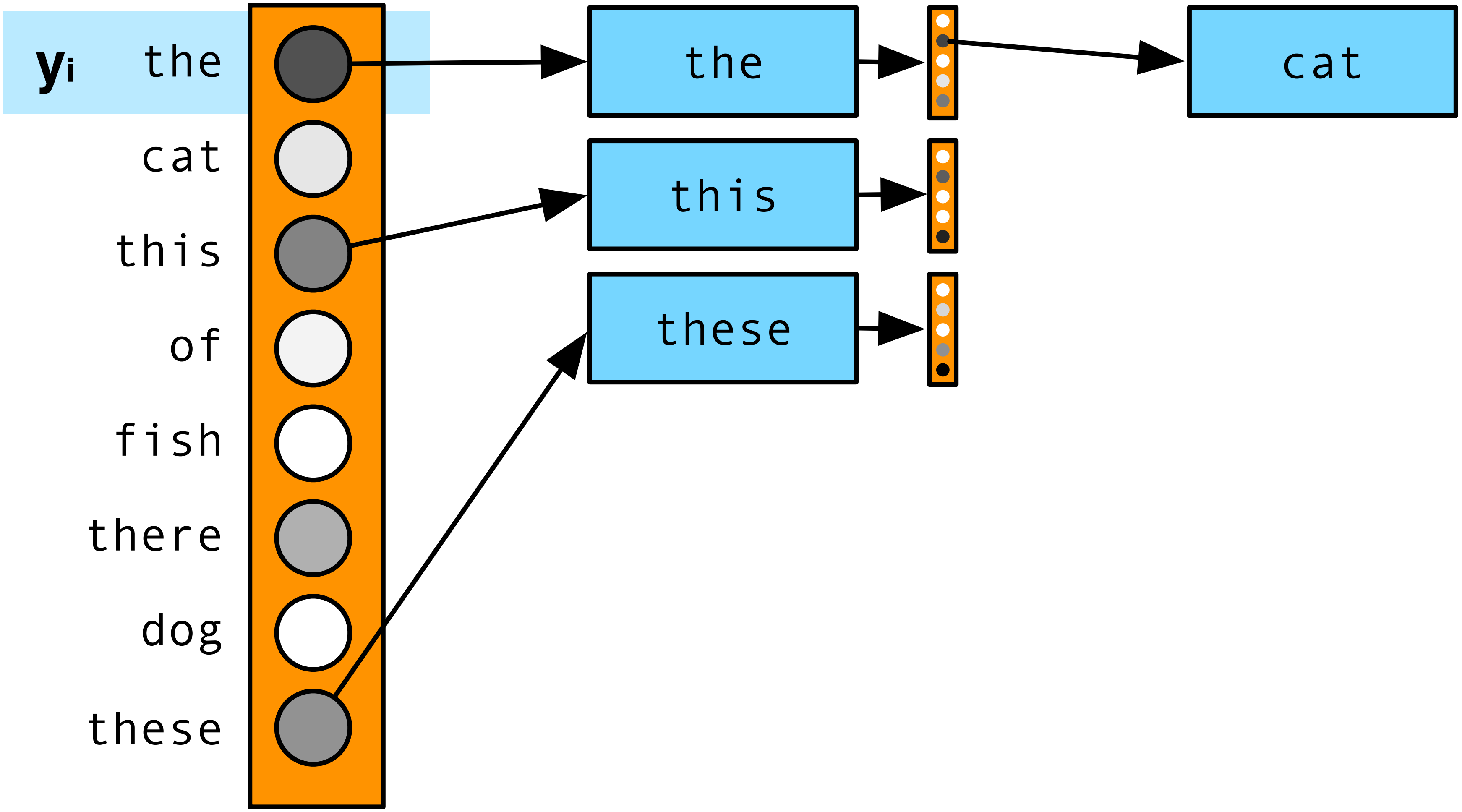
Select Third Best Word



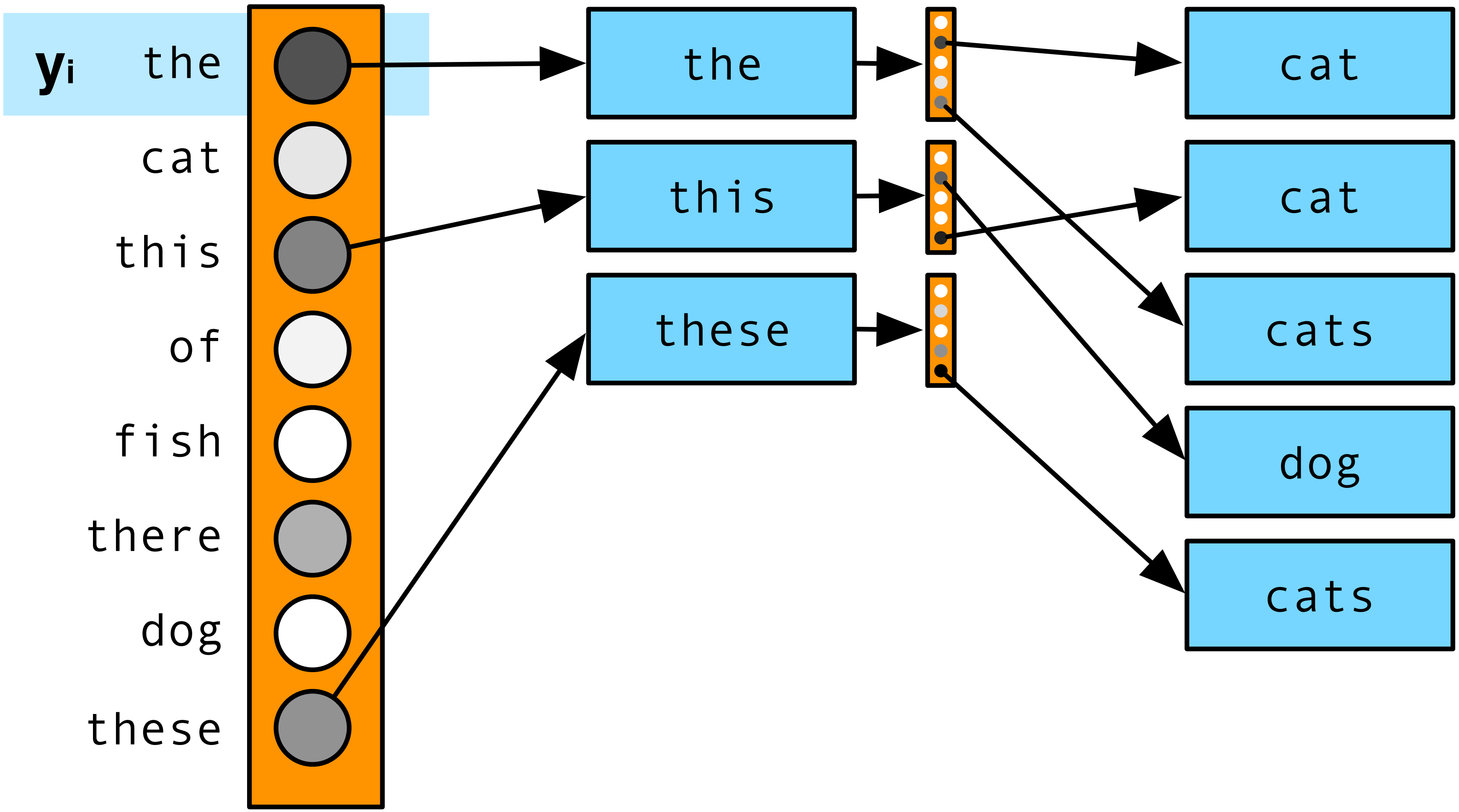
Use Selected Word for Next Predictions



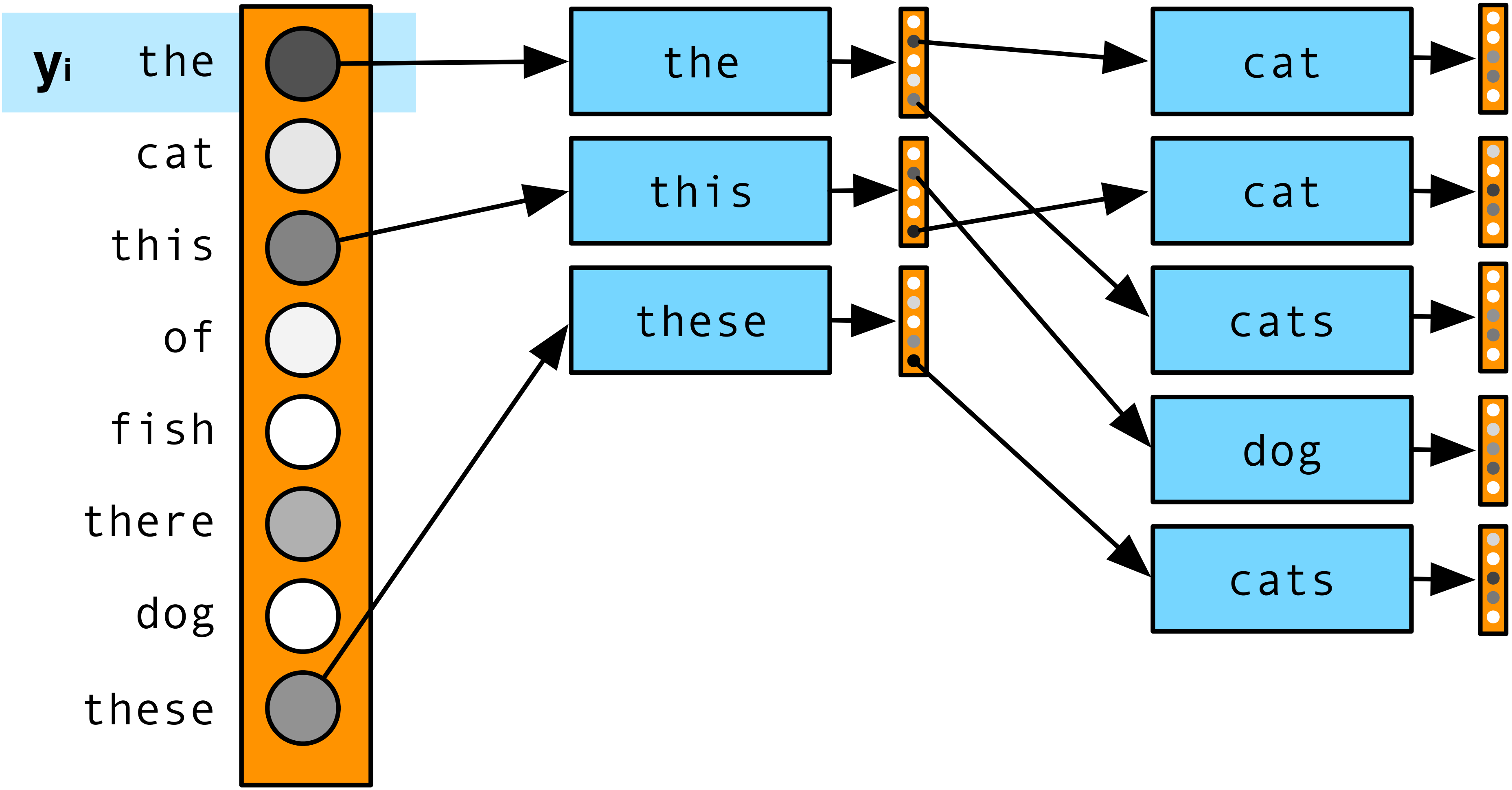
Select Best Continuation



Select Next Best Continuations

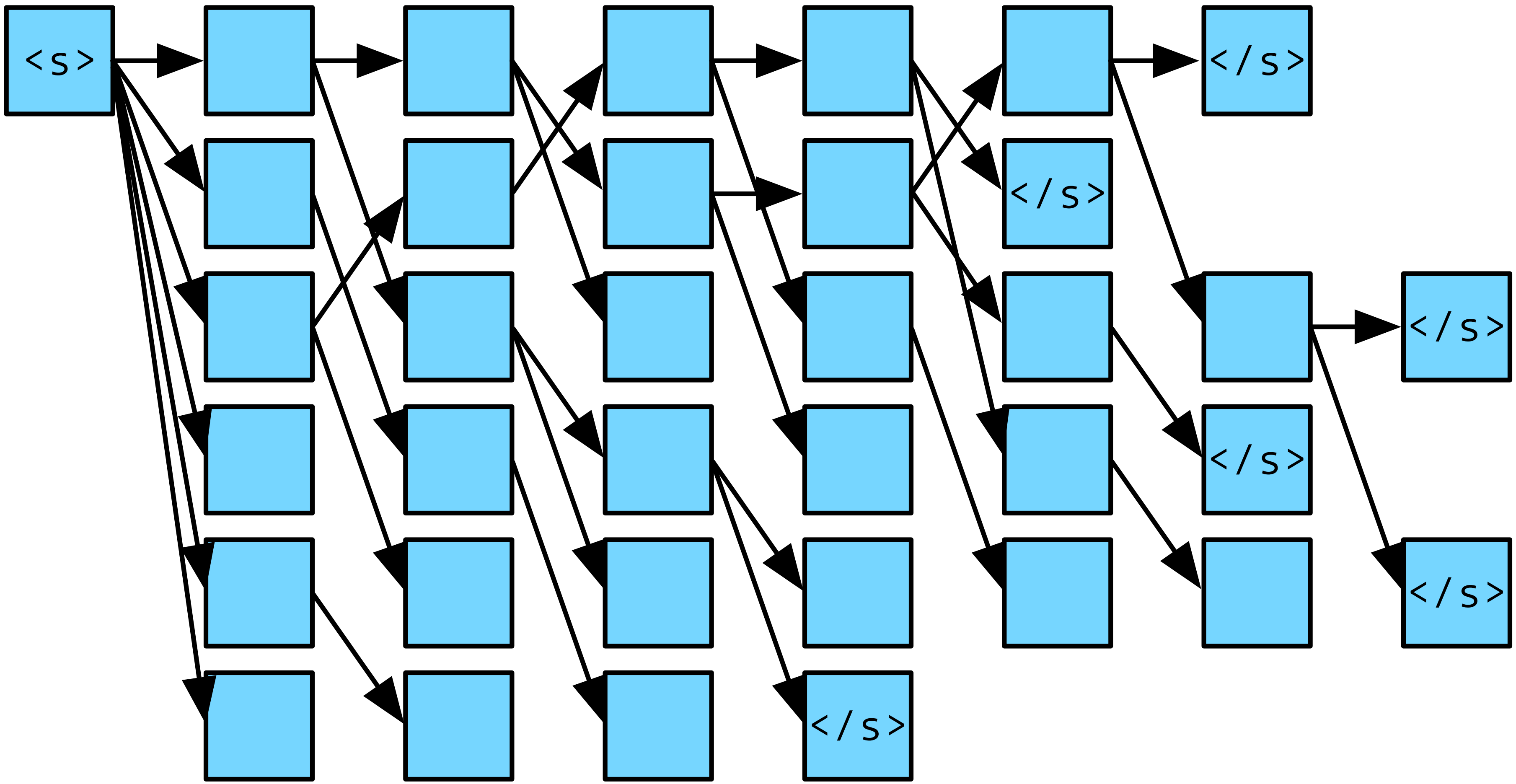


Continue...

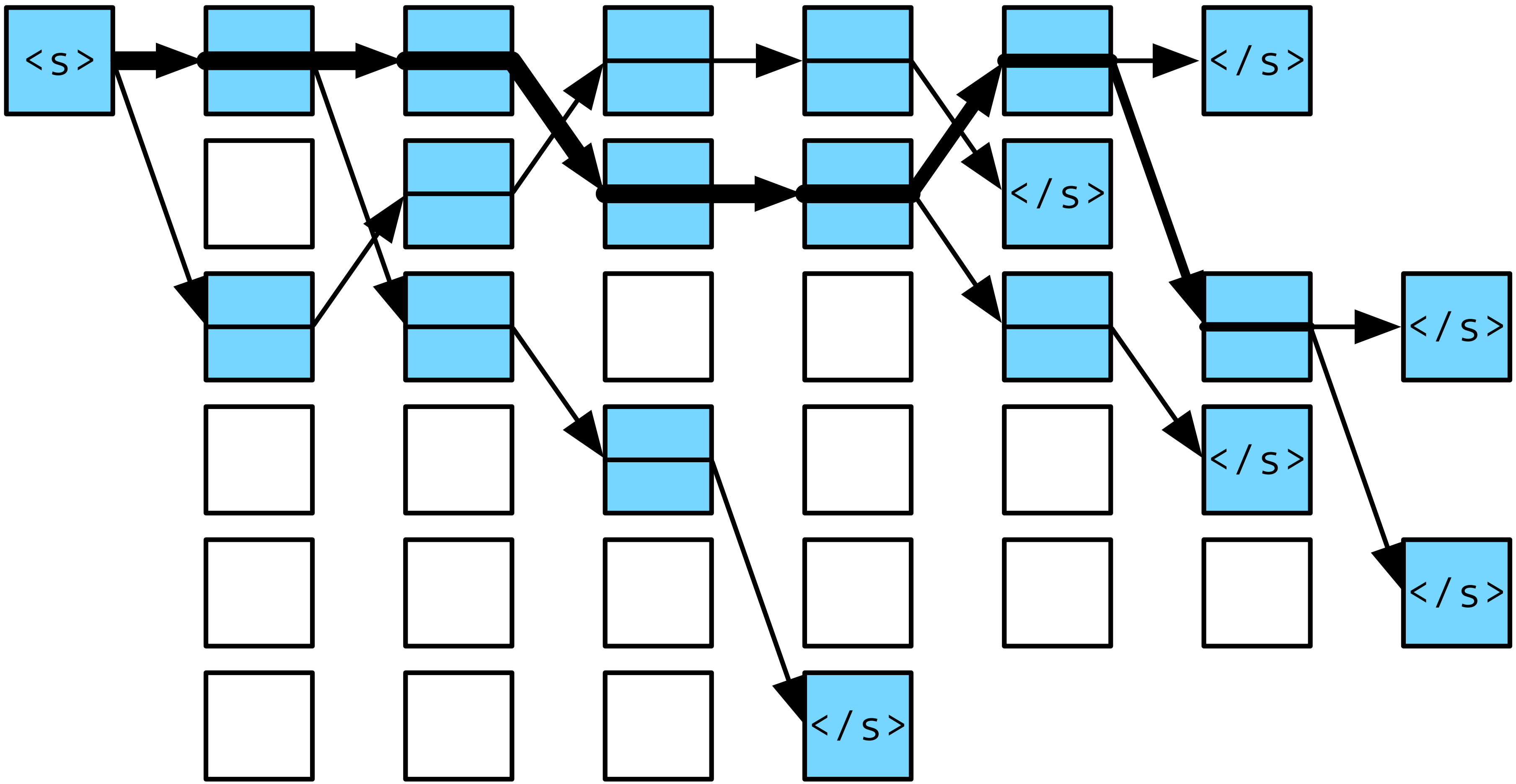


y_i
the
cat
this
of
fish
there
dog
these
the
this
these
cat
cat
cats
dog
cats

Beam Search



Best Paths



Beam Search Details

- Normalize score by length
- No recombination (paths cannot be merged)

Output Word Predictions

Input Sentence: *ich glaube aber auch , er ist clever genug um seine Aussagen vage genug zu halten , so dass sie auf versci Weise interpretiert werden können .*

Best		**Alternatives**
but	(42.1%)	*however* (25.3%), *I* (20.4%), *yet* (1.9%), *and* (0.8%), *nor* (0.8%), ...
I	(80.4%)	*also* (6.0%), *,* (4.7%), *it* (1.2%), *in* (0.7%), *nor* (0.5%), *he* (0.4%), ...
also	(85.2%)	*think* (4.2%), *do* (3.1%), *believe* (2.9%), *,* (0.8%), *too* (0.5%), ...
believe	(68.4%)	*think* (28.6%), *feel* (1.6%), *do* (0.8%), ...
he	(90.4%)	*that* (6.7%), *it* (2.2%), *him* (0.2%), ...
is	(74.7%)	*'s* (24.4%), *has* (0.3%), *was* (0.1%), ...
clever	(99.1%)	*smart* (0.6%), ...
enough	(99.9%)	
to	(95.5%)	*about* (1.2%), *for* (1.1%), *in* (1.0%), *of* (0.3%), *around* (0.1%), ...
keep	(69.8%)	*maintain* (4.5%), *hold* (4.4%), *be* (4.2%), *have* (1.1%), *make* (1.0%), ...
his	(86.2%)	*its* (2.1%), *statements* (1.5%), *what* (1.0%), *out* (0.6%), *the* (0.6%), ...
statements	(91.9%)	*testimony* (1.5%), *messages* (0.7%), *comments* (0.6%), ...
vague	(96.2%)	*v@@* (1.2%), *in* (0.6%), *ambiguous* (0.3%), ...
enough	(98.9%)	*and* (0.2%), ...
so	(51.1%)	*,* (44.3%), *to* (1.2%), *in* (0.6%), *and* (0.5%), *just* (0.2%), *that* (0.2%), ...
they	(55.2%)	*that* (35.3%), *it* (2.5%), *can* (1.6%), *you* (0.8%), *we* (0.4%), *to* (0.3%), ...
can	(93.2%)	*may* (2.7%), *could* (1.6%), *are* (0.8%), *will* (0.6%), *might* (0.5%), ...
be	(98.4%)	*have* (0.3%), *interpret* (0.2%), *get* (0.2%), ...
interpreted	(99.1%)	*interpre@@* (0.1%), *constru@@* (0.1%), ...
in	(96.5%)	*on* (0.9%), *differently* (0.5%), *as* (0.3%), *to* (0.2%), *for* (0.2%), *by* (0.1%), ...
different	(41.5%)	*a* (25.2%), *various* (22.7%), *several* (3.6%), *ways* (2.4%), *some* (1.7%), ...
ways	(99.3%)	*way* (0.2%), *manner* (0.2%), ...
.	(99.2%)	</s> (0.2%), *,* (0.1%), ...
</s>	(100.0%)	

Open Source NMT Toolkits

- OpenNMT (SYSTRAN/Harvard)
  - http://opennmt.net/
  - Really three toolkits: OpenNMT-lua, OpenNMT-py, OpenNMT-tf
- Nematus (University of Edinburgh)
  - https://github.com/EdinburghNLP/nematus
- Marian (University of Pozan)
  - https://marian-nmt.github.io/
- Sockeye (Amazon)
  - https://github.com/awslabs/sockeye

Open Source NMT Toolkits 2

- Tensor2Tensor (Google)
  - https://github.com/tensorflow/tensor2tensor#translation
- Facebook
  - https://github.com/facebookresearch/fairseq
- ModernMT
  - http://modernmt.eu
- Neural Monkey (Charles University, Prague)
  - https://github.com/ufal/neuralmonkey
- XNMT (CMU)
  - https://github.com/neulab/xnmt
- SGNMT (Cambridge)
  - https://github.com/ucam-smt/sgnmt

NMT Toolkits

- Number of toolkits still increasing
- From experience with SMT toolkits (Moses)
  - Toolkit that can build the best community will prevail
  - New iterative improvements in NMT will be integrated
  - Can be maintained for years to come
  - Will **not** integrate use case-specific technologies (or only to small degree)

Tutorials

- Tensorflow seq2seq
  – https://www.tensorflow.org/tutorials/seq2seq
- ACL 2016
  – https://sites.google.com/site/acl16nmt/home