

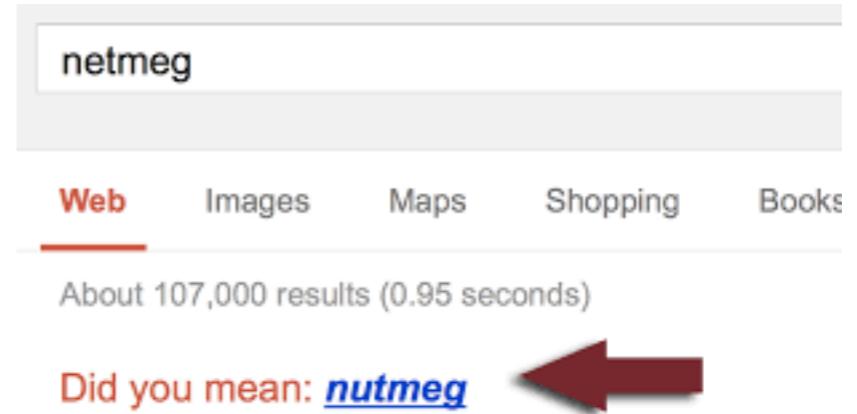
Neural Network Language Models & Large Language Models

Mittul Singh

Language Model Applications

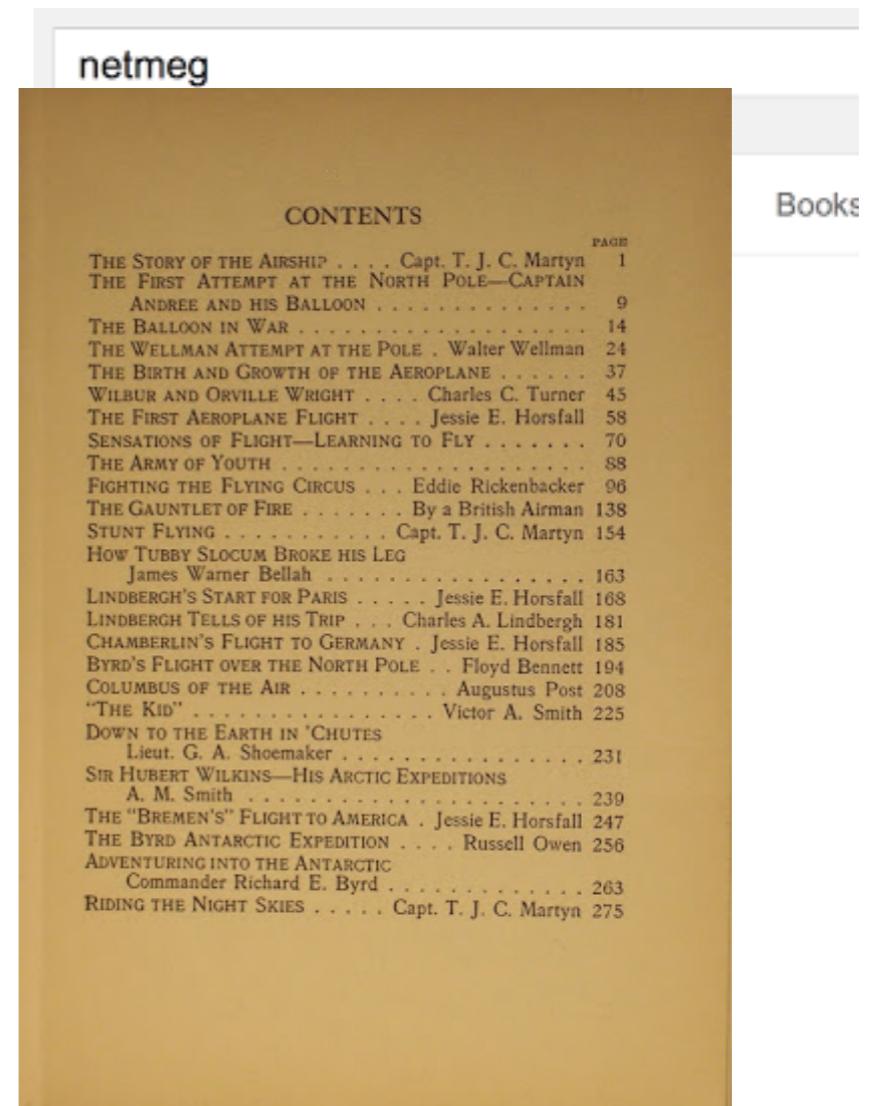
Language Model Applications

- Spelling correction, text input
 - Search Query Completion



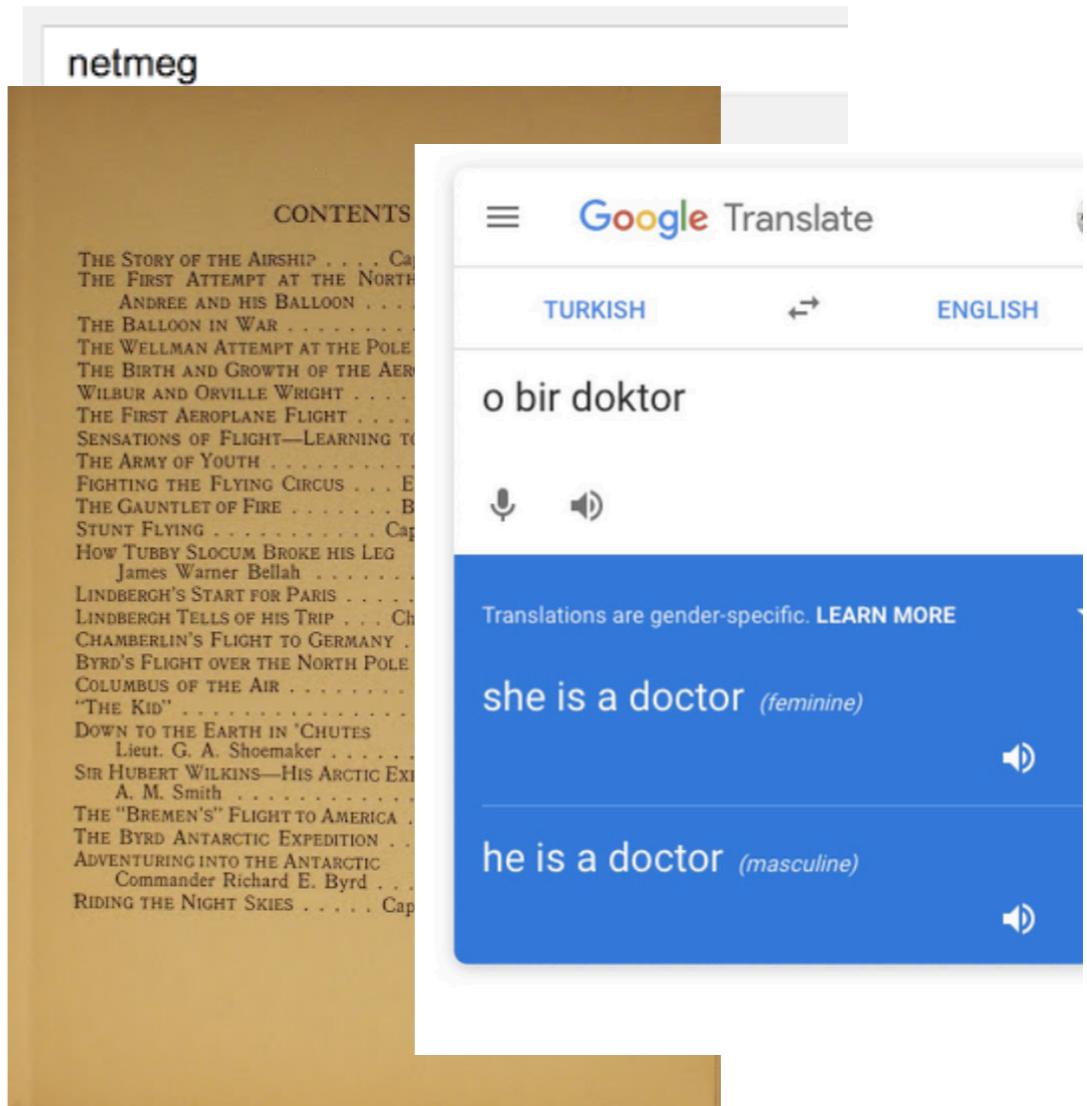
Language Model Applications

- Spelling correction, text input
 - Search Query Completion
- Optical character recognition
 - e.g. scanning old books



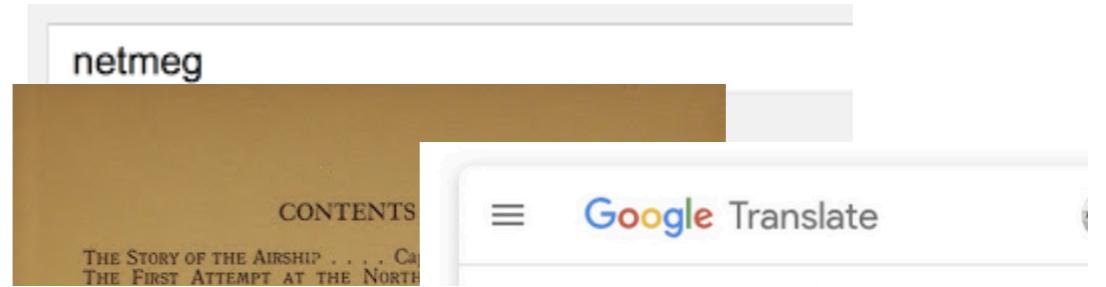
Language Model Applications

- Spelling correction, text input
 - Search Query Completion
- Optical character recognition
 - e.g. scanning old books
- Statistical machine translation



Language Model Applications

- Spelling correction, text input
 - Search Query Completion
- Optical character recognition
 - e.g. scanning old books
- Statistical machine translation
- Information retrieval
 - Question Answering



Passage Sentence

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity.

Question

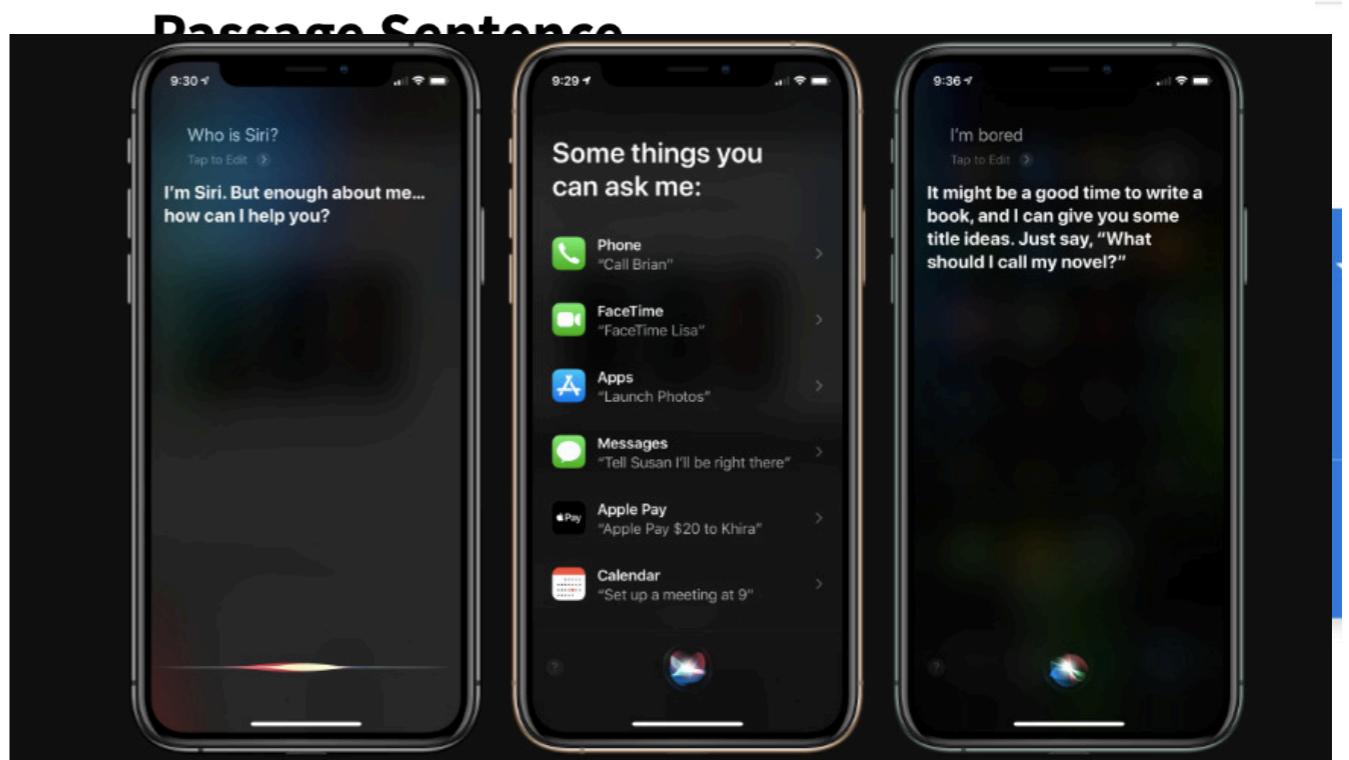
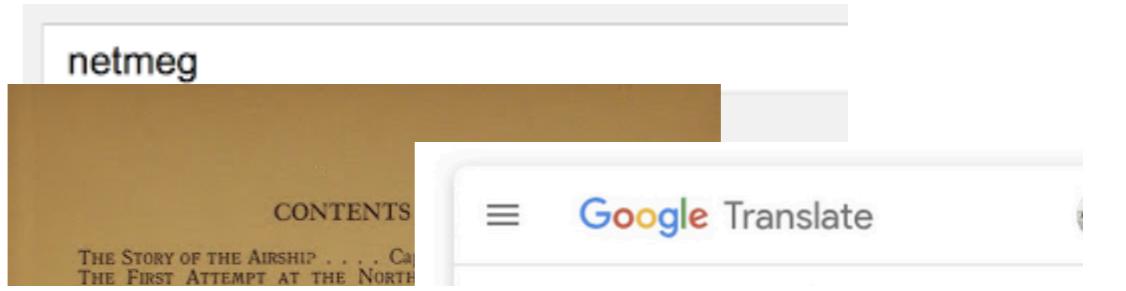
What causes precipitation to fall?

Answer Candidate

gravity

Language Model Applications

- Spelling correction, text input
 - Search Query Completion
- Optical character recognition
 - e.g. scanning old books
- Statistical machine translation
- Information retrieval
 - Question Answering
- Automatic speech recognition
- ...



Answer Candidate

gravity

Recap: N-gram Language Models

Recap: N-gram Language Models

- We wanted to calculate

$$p(W) = p(w_1, w_2, \dots, w_n) \quad (1)$$

$$p(w_i | w_{i-1}, w_{i-2}, \dots, w_{n-1}) \approx p(w_i | w_{i-1}, w_{i-2}, w_{i-3}, w_{i-4}) \quad (2)$$

Neural Network Classifier for Language Modelling

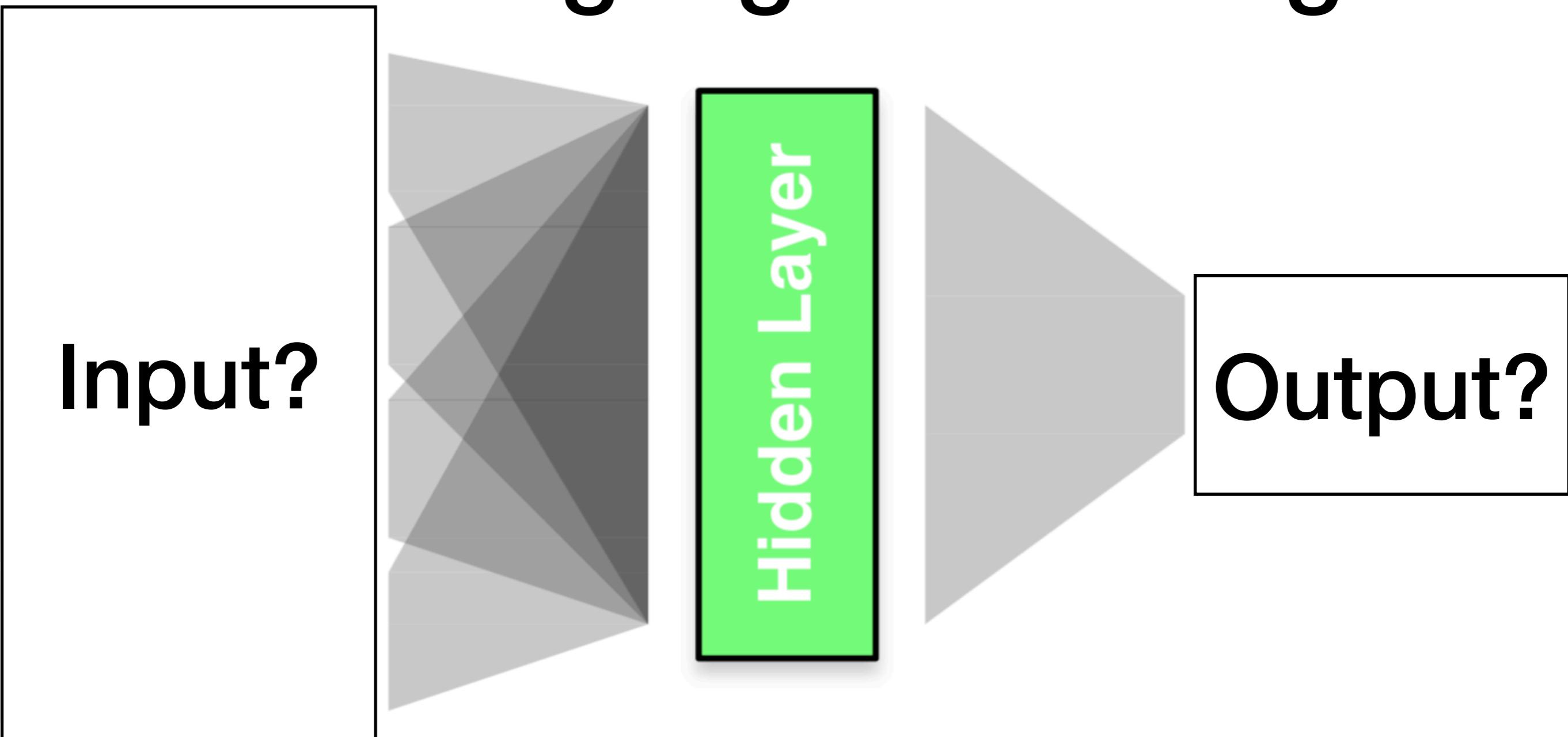


Image: <http://mt-class.org/jhu/slides/lecture-nn-lm.pdf>

Neural Network Classifier for Language Modelling

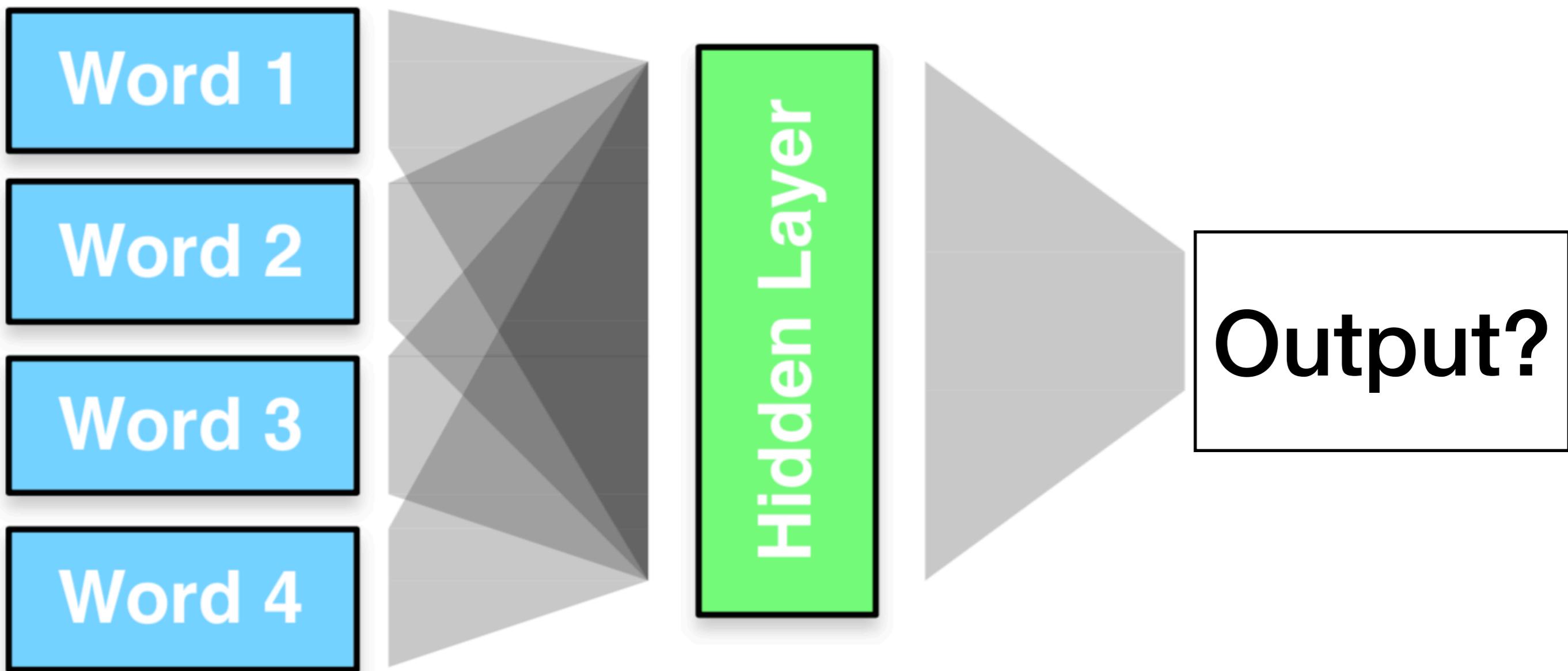


Image: <http://mt-class.org/jhu/slides/lecture-nn-lm.pdf>

Neural Network Classifier for Language Modelling

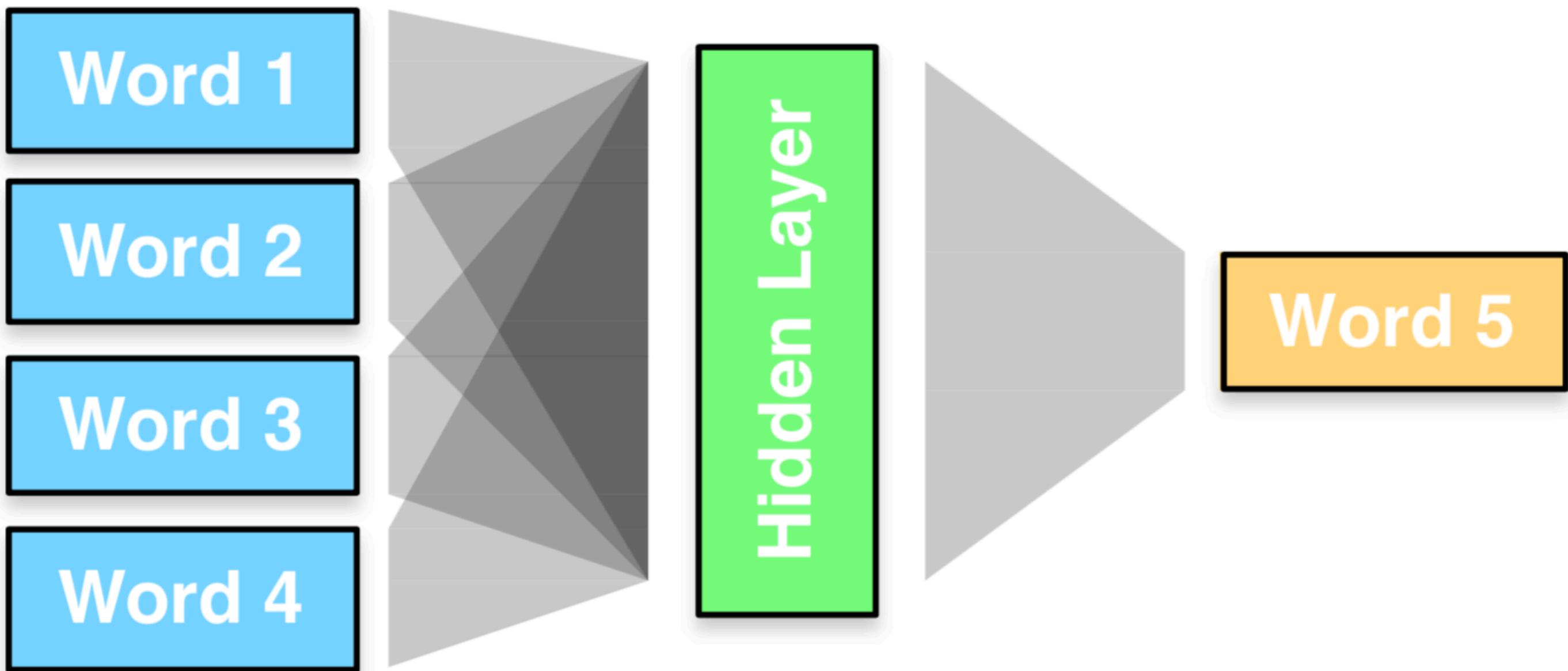


Image: <http://mt-class.org/jhu/slides/lecture-nn-lm.pdf>

Second Sketch

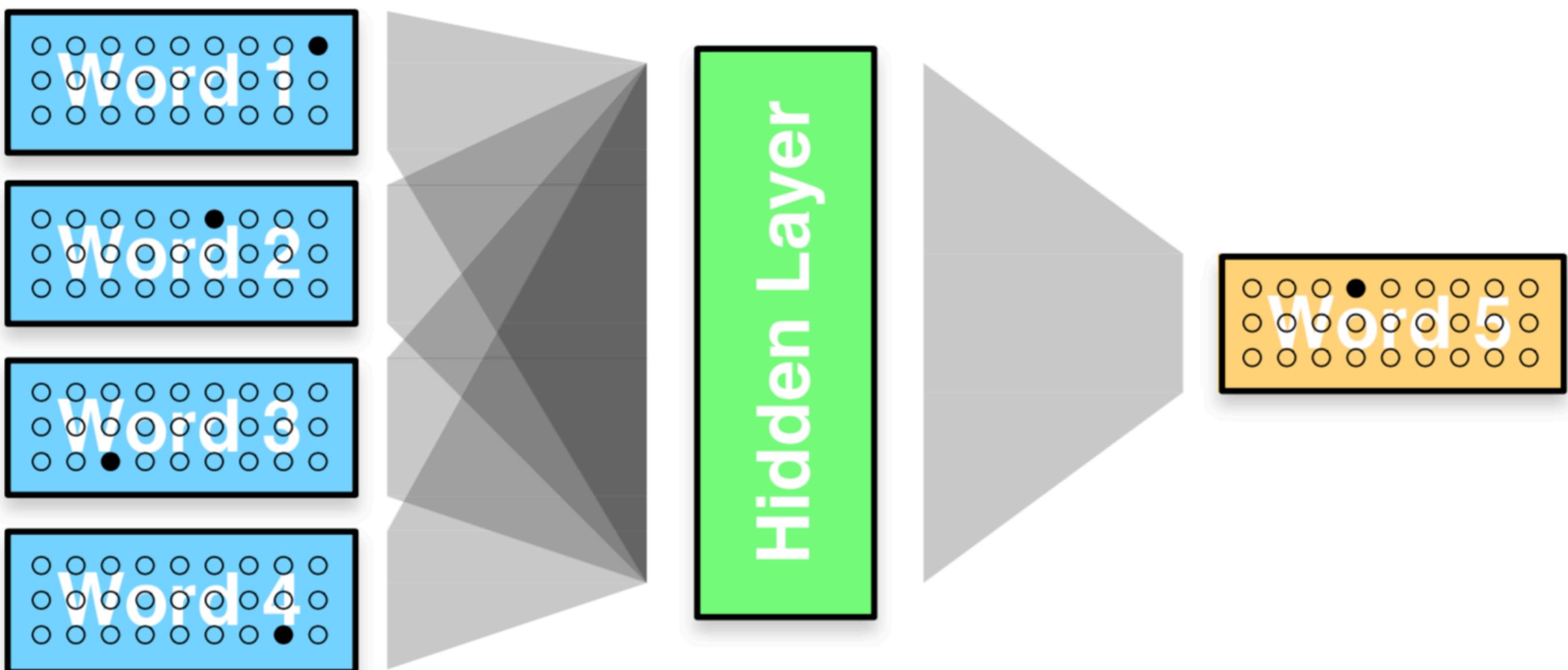
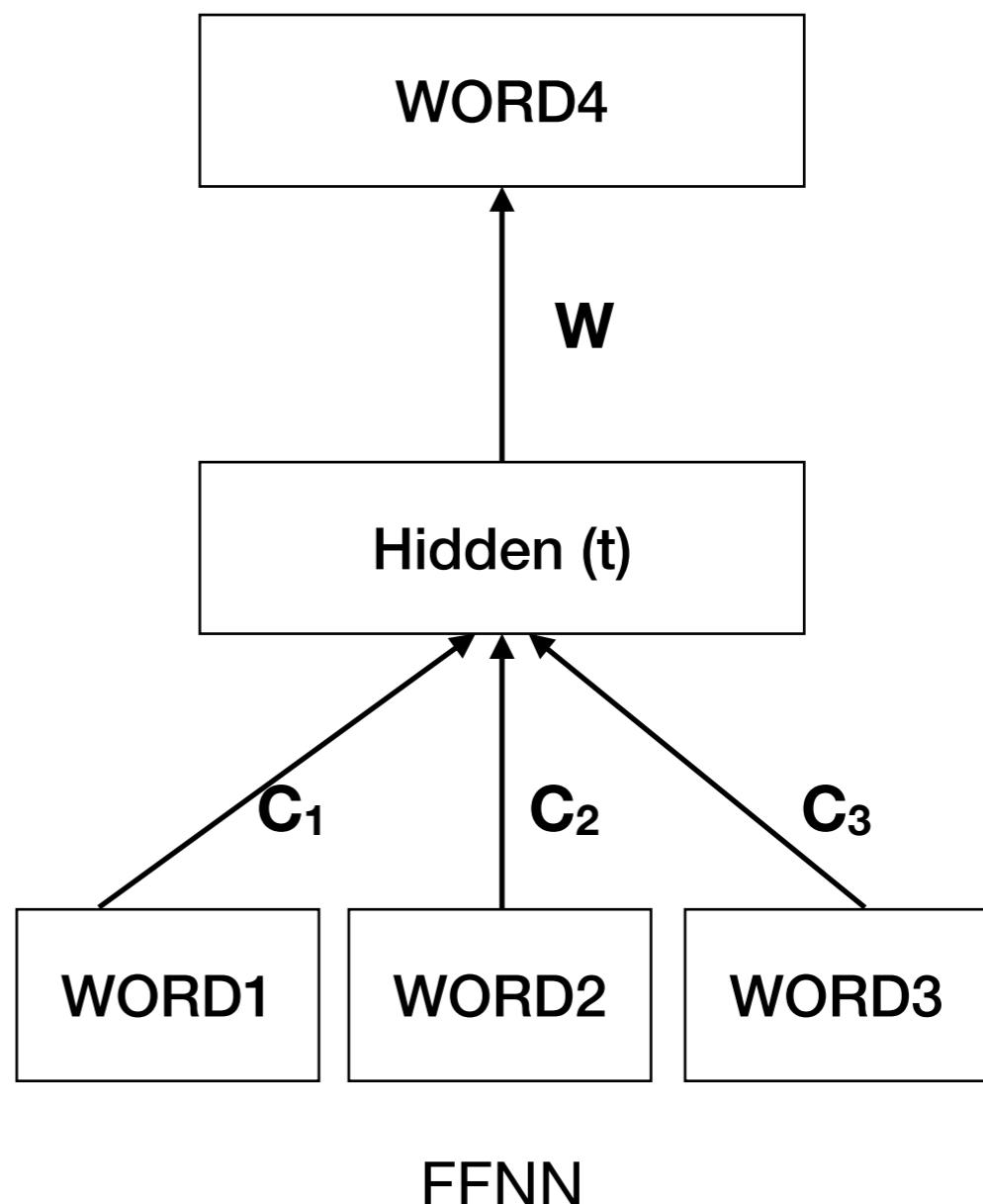


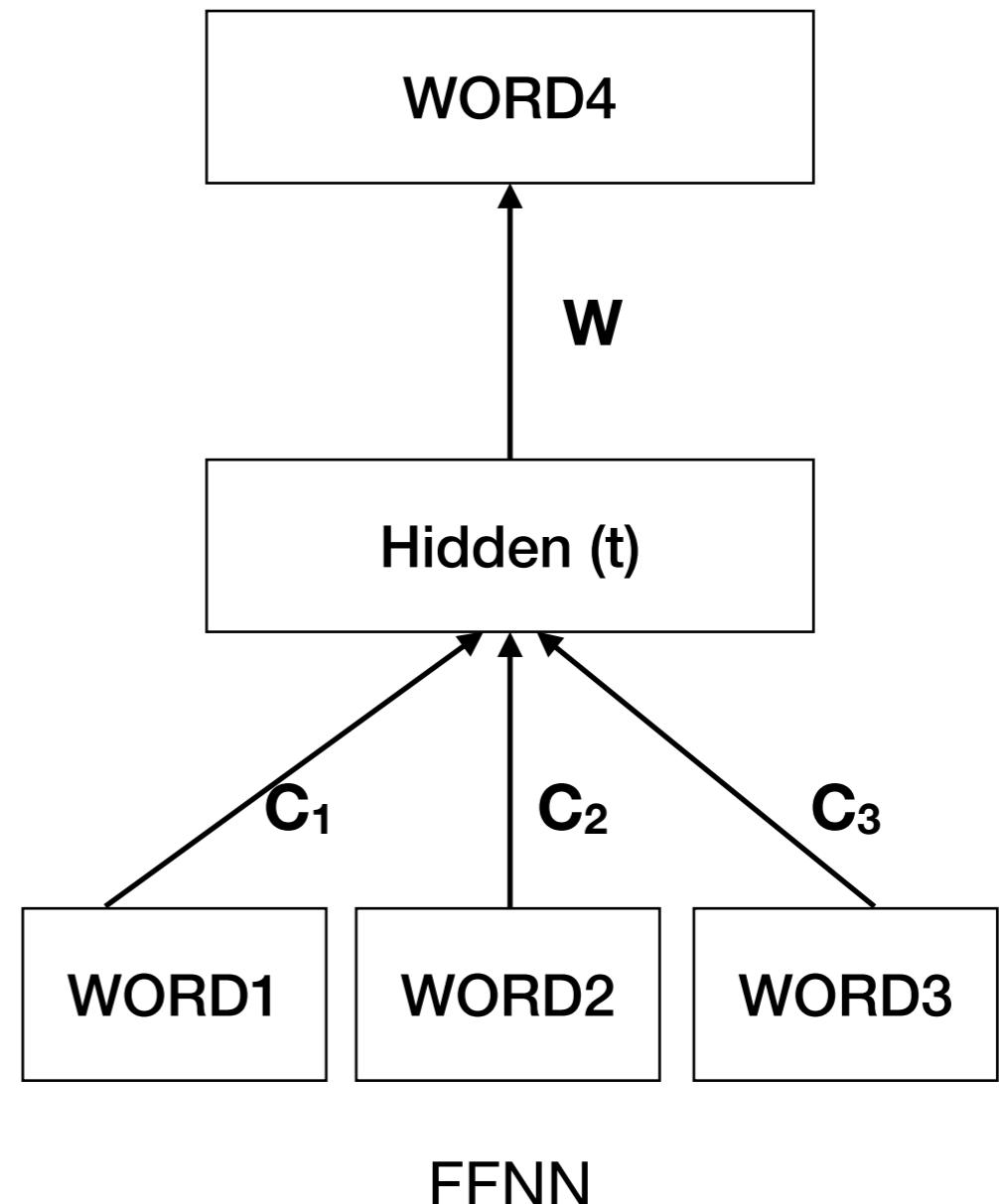
Image: <http://mt-class.org/jhu/slides/lecture-nn-lm.pdf>

Feedforward Neural Network LM (FFNN)



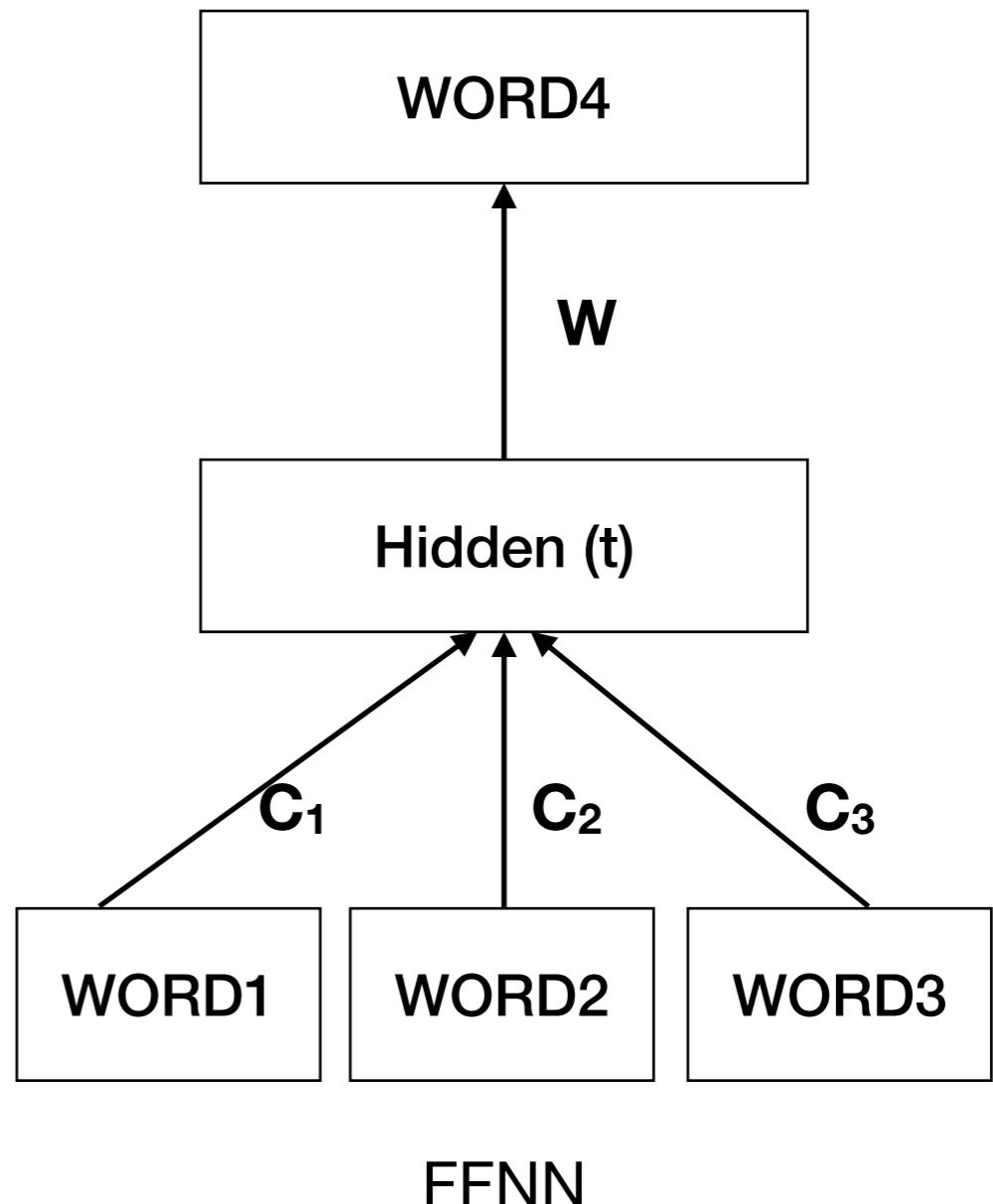
Feedforward Neural Network LM (FFNN)

- Loop through the entire corpus



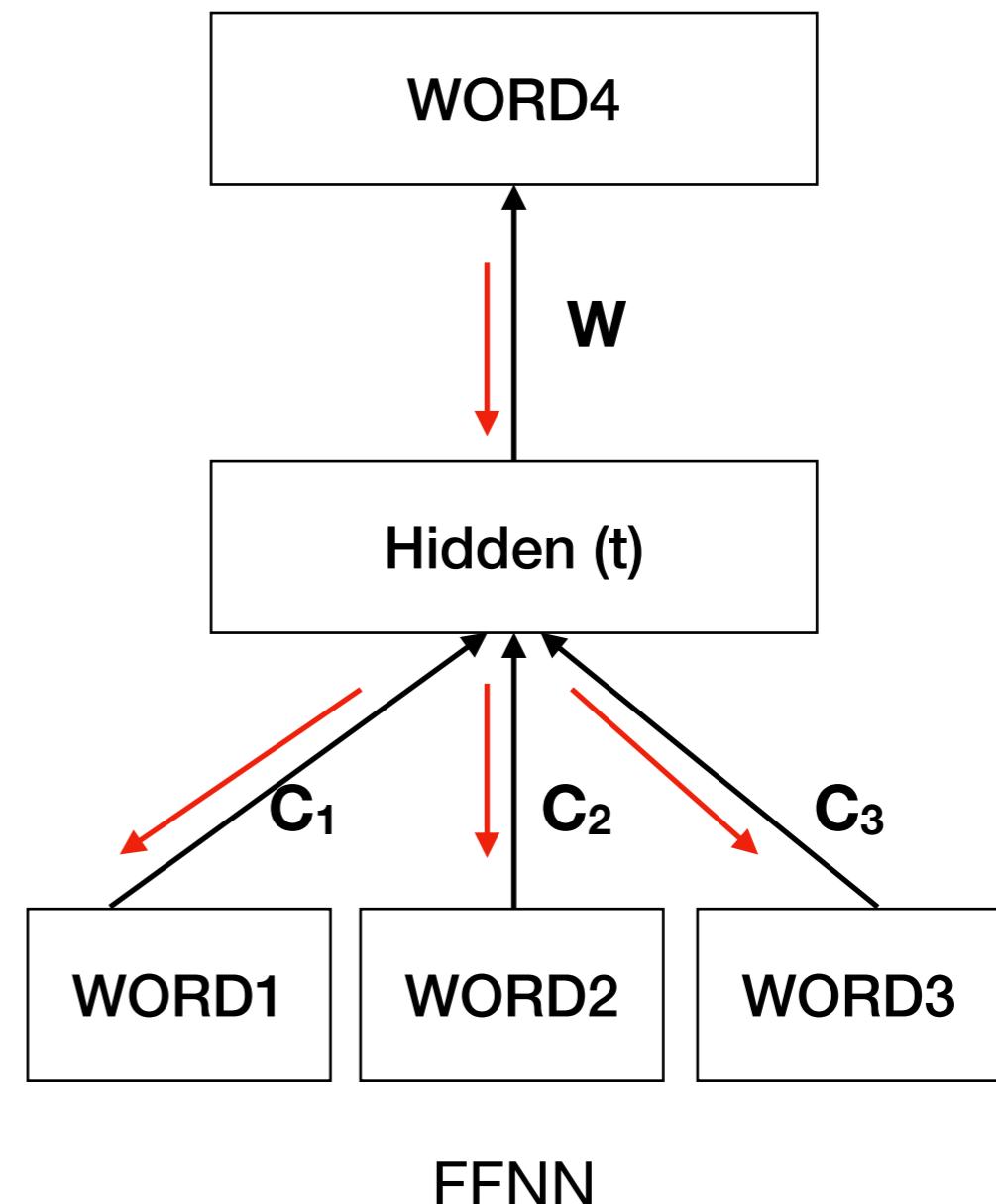
Feedforward Neural Network LM (FFNN)

- Loop through the entire corpus
- Calculate error or loss (cross-entropy loss)



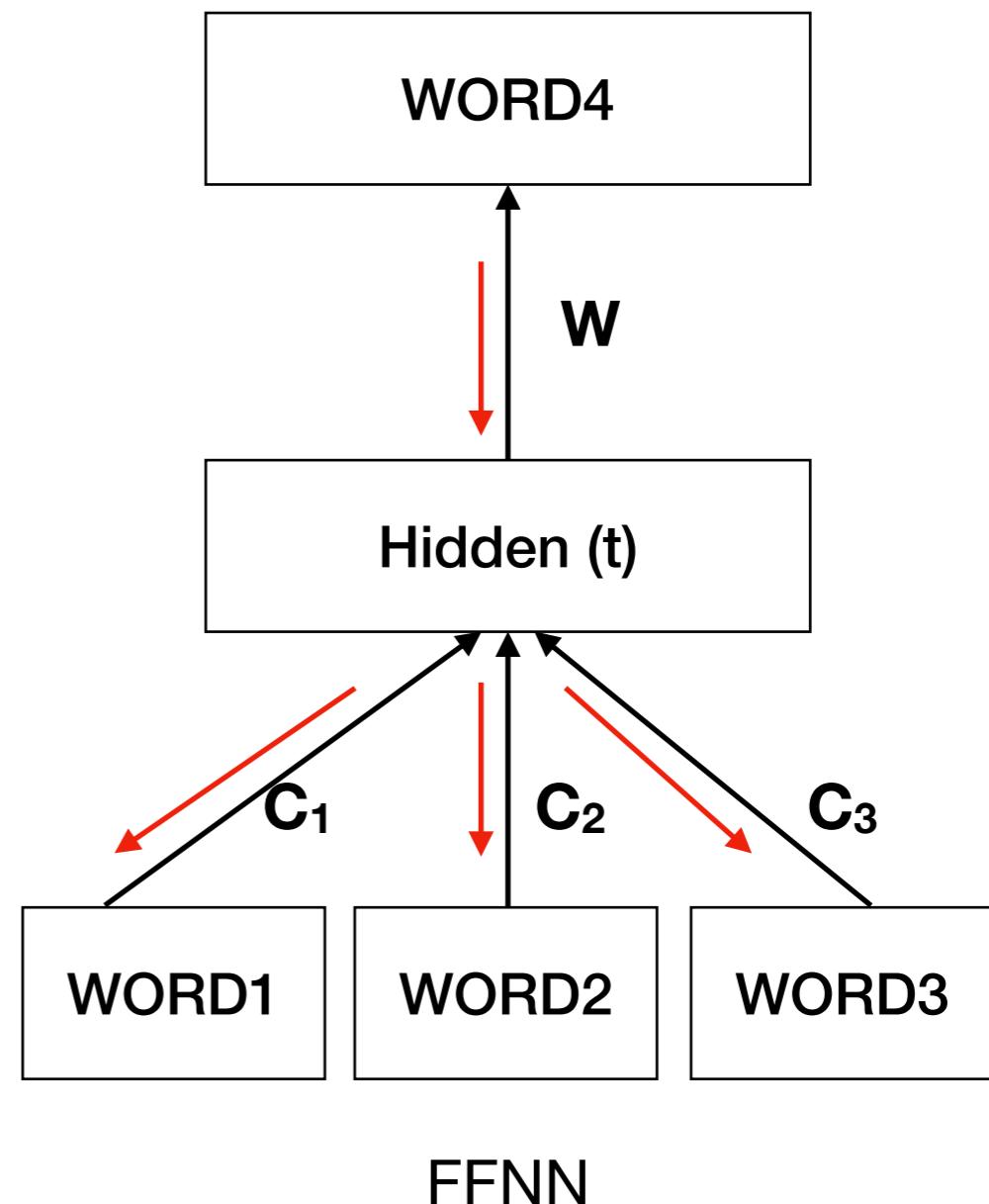
Feedforward Neural Network LM (FFNN)

- Loop through the entire corpus
- Calculate error or loss (cross-entropy loss)
- Propagate the **error** through network to update the weight matrices



Feedforward Neural Network LM (FFNN)

- Loop through the entire corpus
- Calculate error or loss (cross-entropy loss)
- Propagate the **error** through network to update the weight matrices
- Back Propagation



Why NNs for LMs

Why NNs for LMs

The cat is walking in the bedroom

A dog was running in a room

Why NNs for LMs

The **cat** is **walking** in the **bedroom**

A **dog** was **running** in a **room**

The **cat** is **running** in a **room**

=> A **dog** is **walking** in a **bedroom**

The **dog** was **walking** in the **room**

Why NNs for LMs

The **cat** is **walking** in the **bedroom**

A **dog** was **running** in a **room**

The **cat** is **running** in a **room**

=> A **dog** is **walking** in a **bedroom**

The **dog** was **walking** in the **room**

- NNLM generalizes in such a way that **similar** words have **similar** vectors

Why NNs for LMs

The **cat** is **walking** in the **bedroom**

A **dog** was **running** in a **room**

The **cat** is **running** in a **room**

=> A **dog** is **walking** in a **bedroom**

The **dog** was **walking** in the **room**

- NNLM generalizes in such a way that **similar** words have **similar** vectors
- Presence of only one such sentence in the training set helps improve the probability of its combinations

Types of NNLM

- Feedforward Neural Network Language Model
- Recurrent Neural Network Language Model
- Long-Short Term Memory LM
- Transformer-based LM
- ..

NNLM: Questions

- What might be some challenges that you might face while training or applying NNLMs?

NNLMs Challenges

NNLMs Challenges

- Long-Range Dependencies

NNLMs Challenges

- Long-Range Dependencies
- Training Speed

NNLMs Challenges

- Long-Range Dependencies
- Training Speed
- On-disk Size

NNLMs Challenges

- Long-Range Dependencies
- Training Speed
- On-disk Size
- Rare Context

NNLMs Challenges

- Long-Range Dependencies
- Training Speed
- On-disk Size
- Rare Context
- Bias

NNLMs Challenges

- Long-Range Dependencies
- Training Speed
- On-disk Size
- Rare Context
- Bias
- ...

Feedforward: Long-term information

- “I grew up in France... I speak fluent _____.”

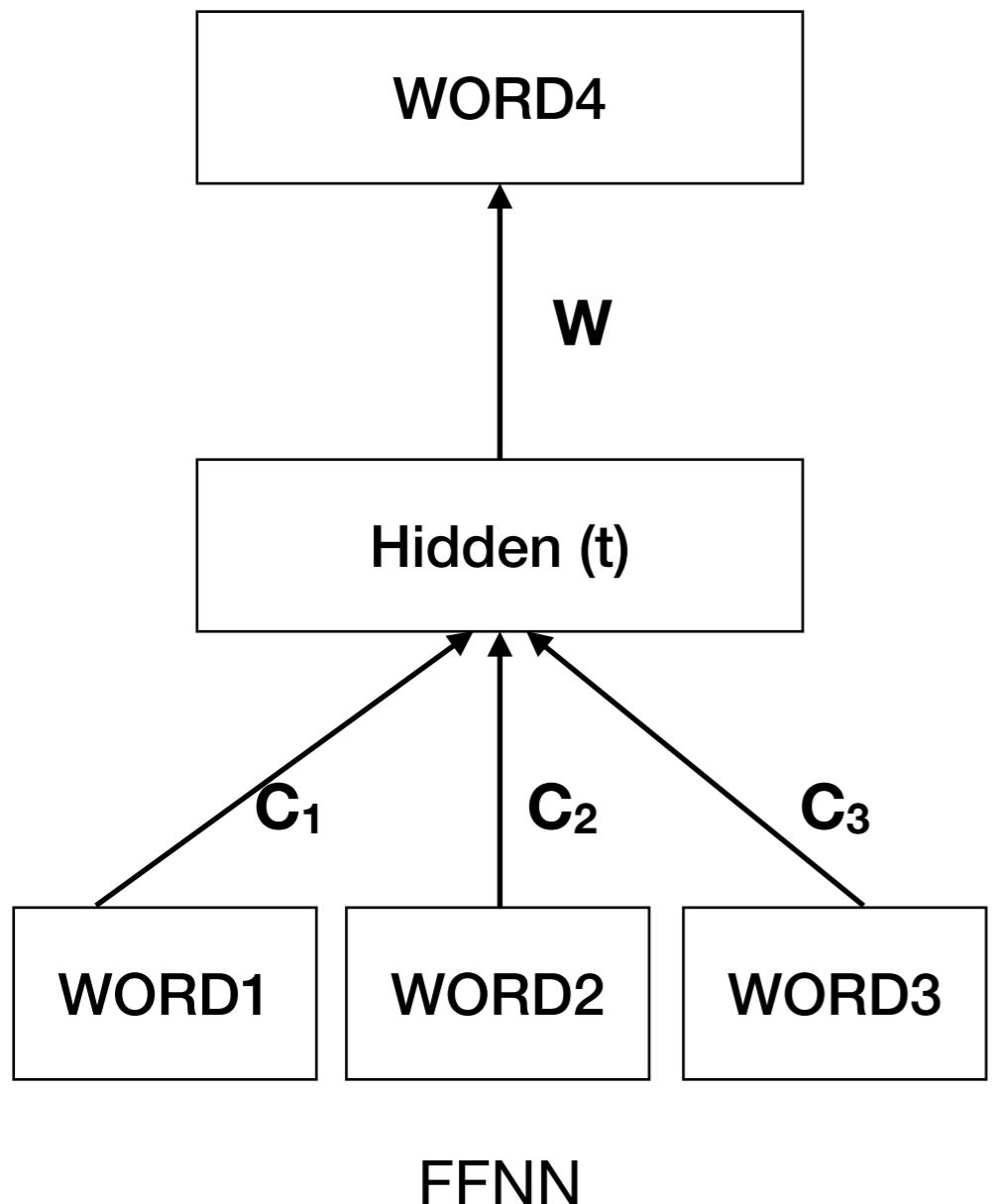
Feedforward: Long-term information

- “I grew up in France... I speak fluent French.”

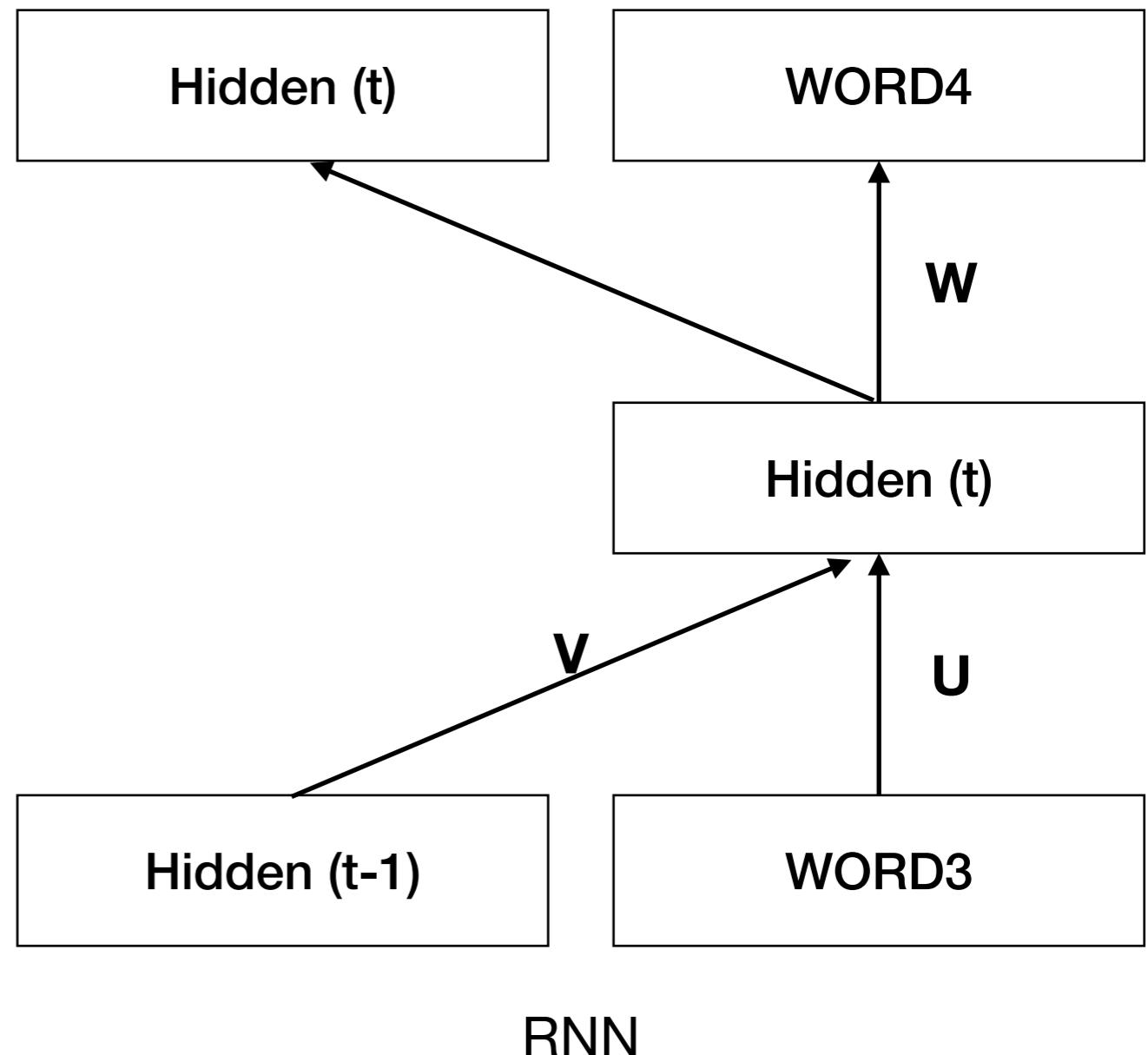
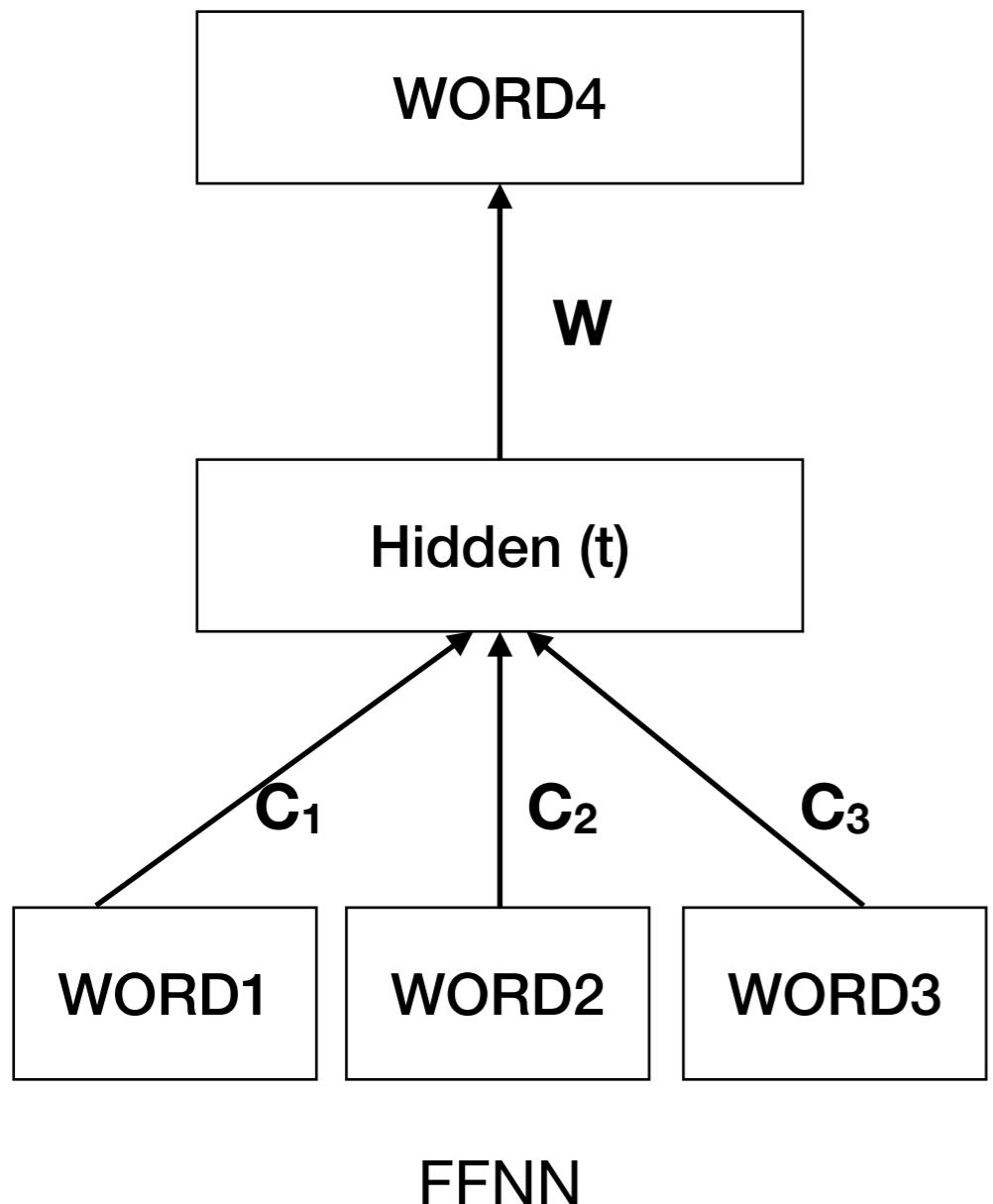
Feedforward: Long-term information

- “I grew up in France... I speak fluent *French*.”
- Feedforward Neural Network (FFNN) has limited context size

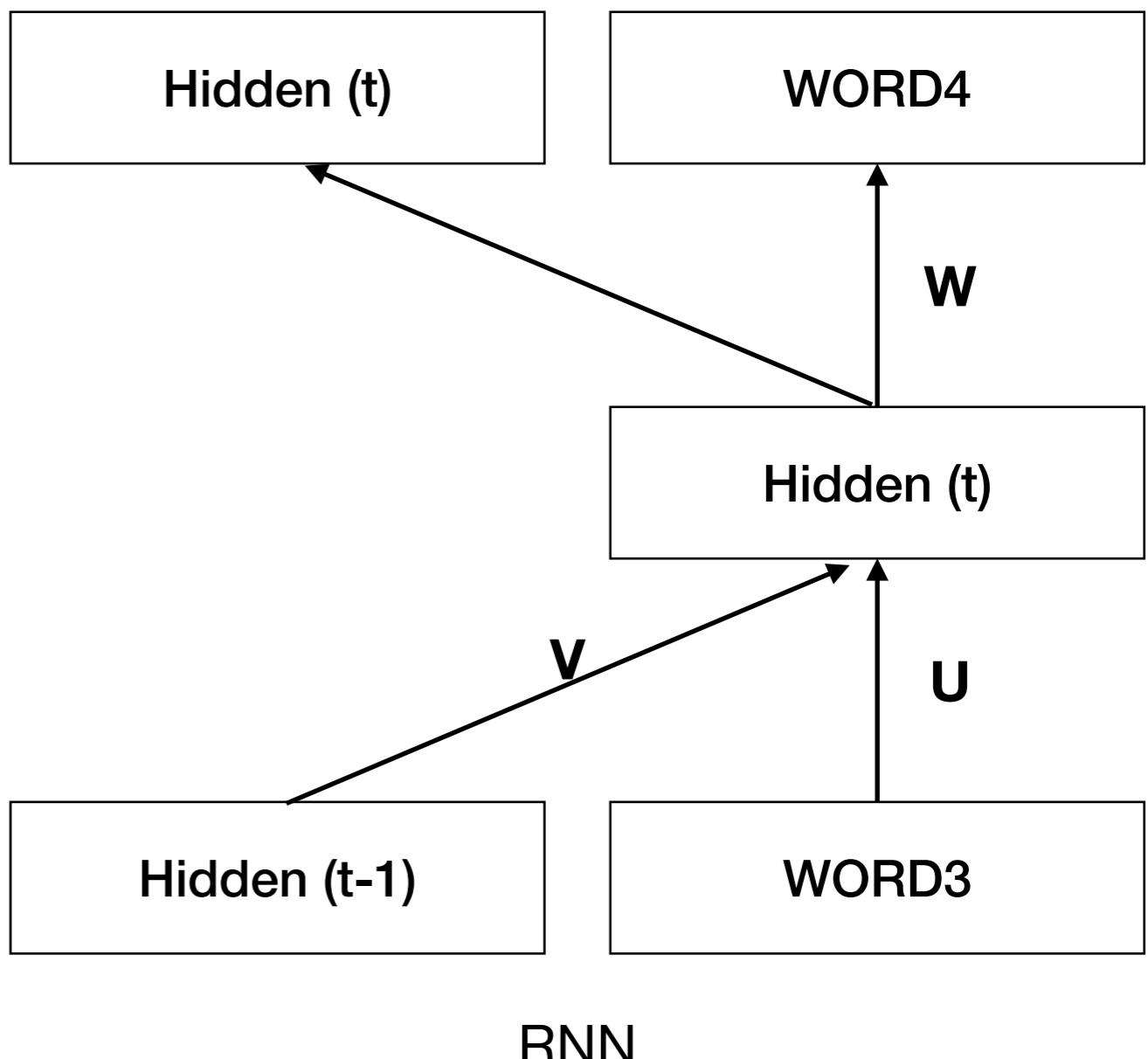
Recurrent Neural Networks (RNN)



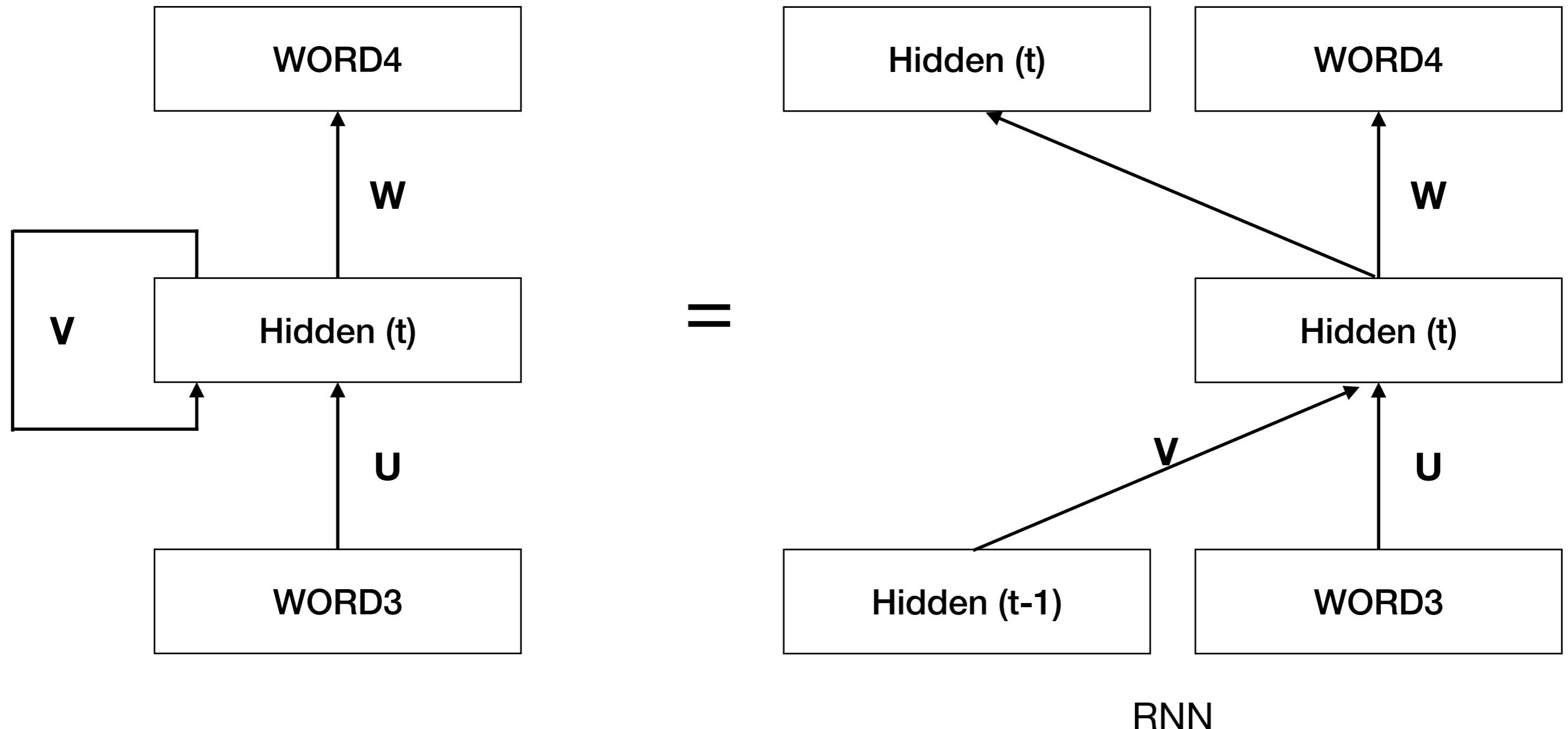
Recurrent Neural Networks (RNN)



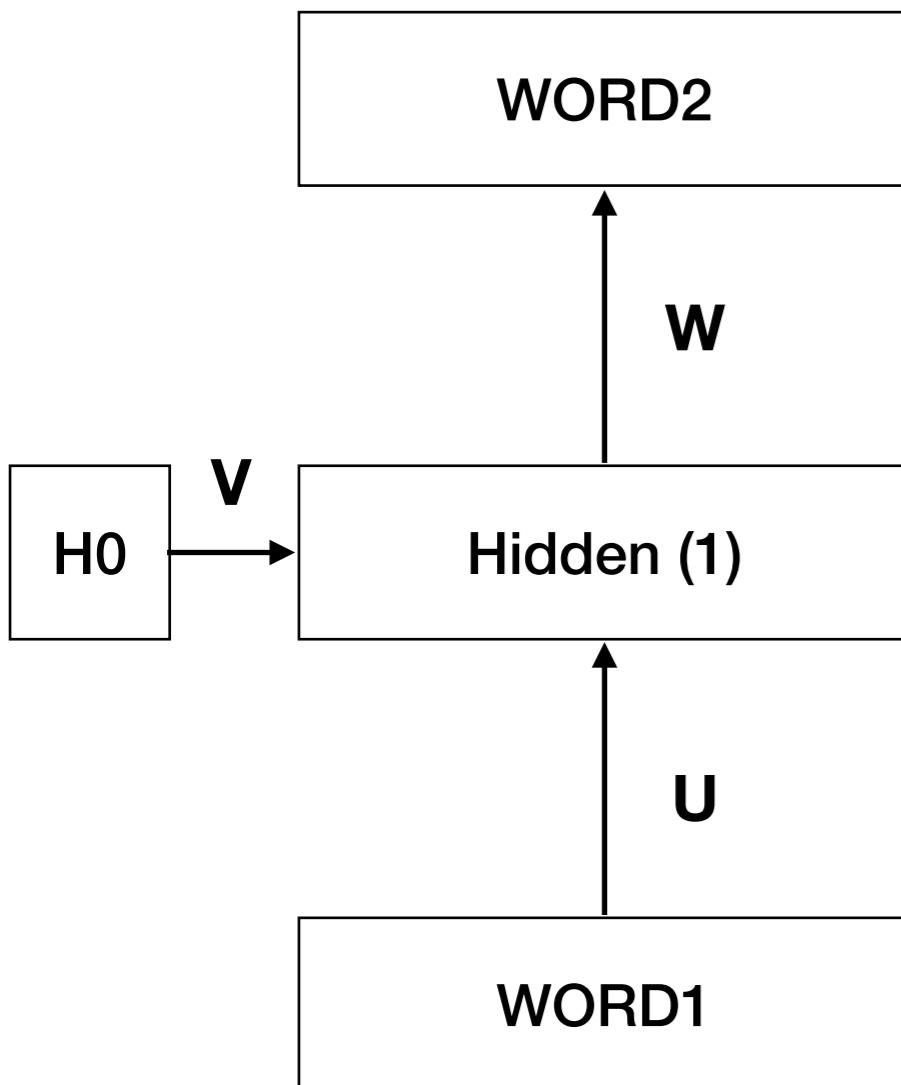
Recurrent Neural Networks (RNN)



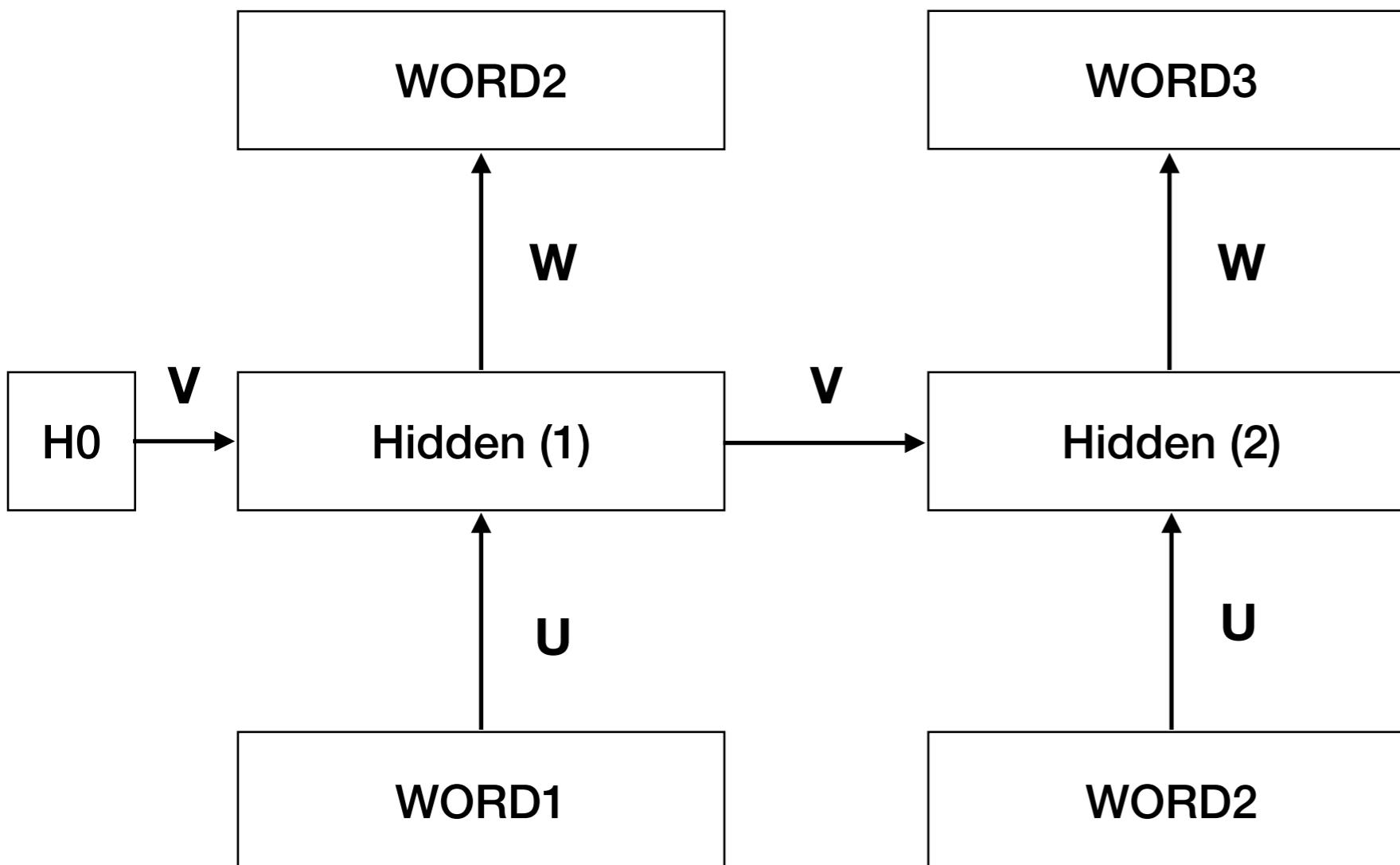
Recurrent Neural Networks (RNN)



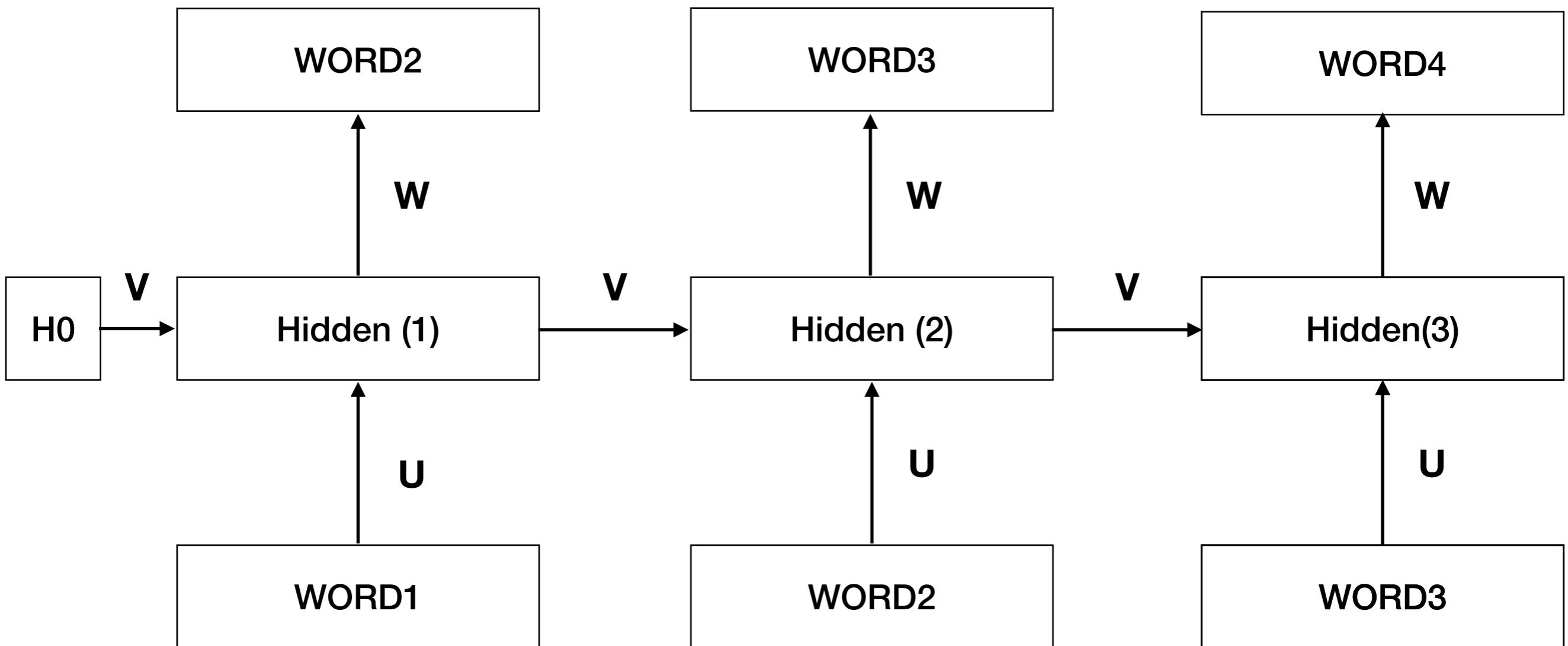
RNN: Timestep 1



RNN: Timestep 2



RNN: Timestep 3



Theoretically information from first step is available to the present timestep

RNN

- “I grew up in France... I speak fluent French.”

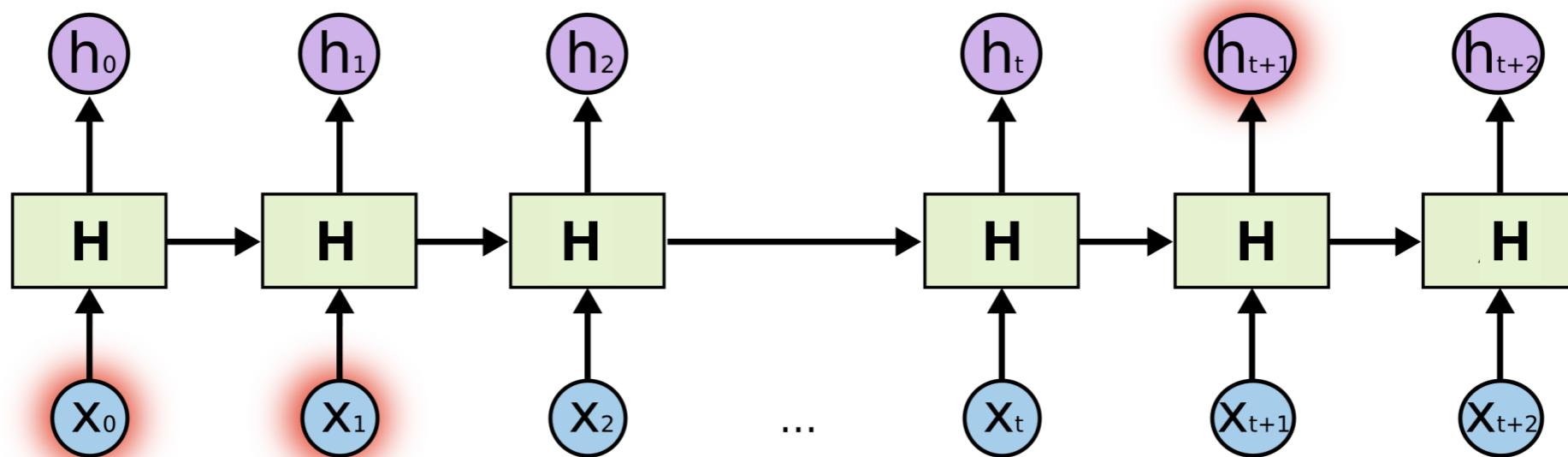


Image: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

RNN

- “I grew up in France... I speak fluent French.”
- As the gap grows, RNNs become unable to learn to connect information

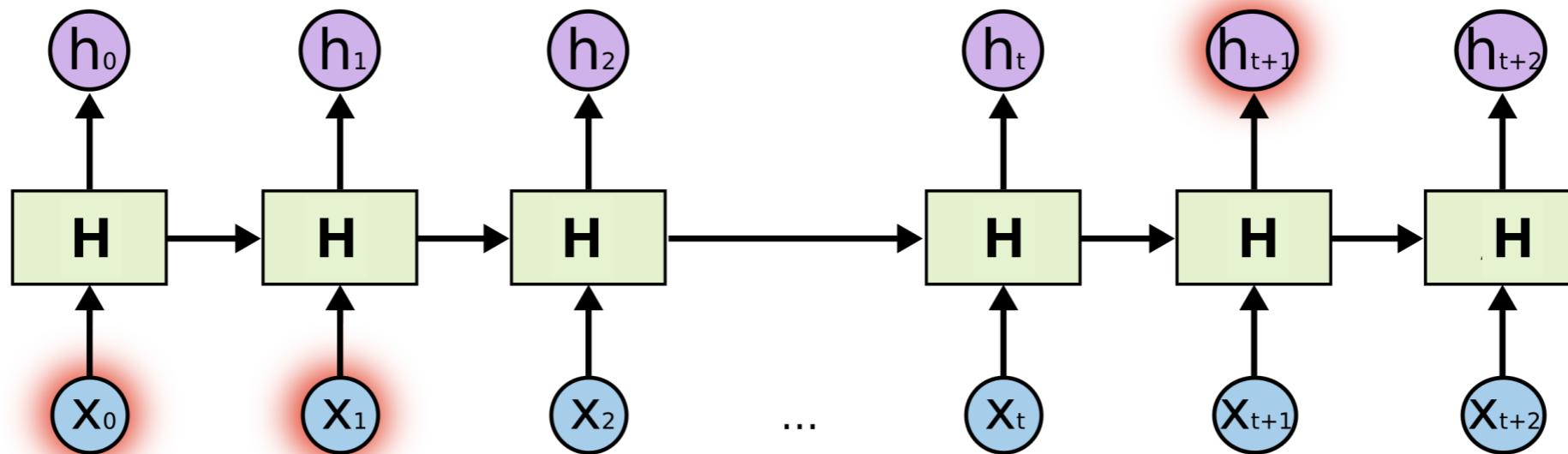
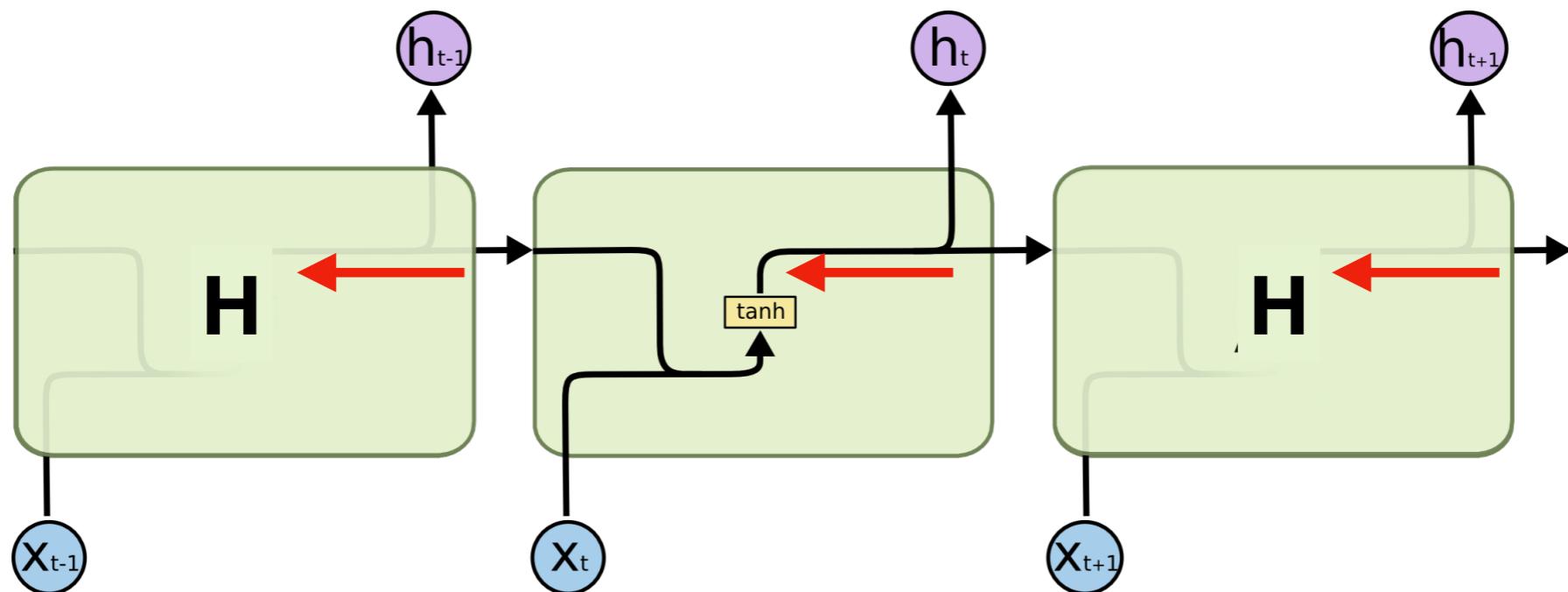


Image: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

RNN



- Error (red arrow) is passed through a chain of hidden states
- Error passing through multiple of these functions can vanish

Image: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

RNNs don't do long-distance well

- The main problem with RNNs is that gradients less than 1 become exponentially small over time

RNNs don't do long-distance well

- The main problem with RNNs is that gradients less than 1 become exponentially small over time
- Known as the vanishing gradient problem

RNNs don't do long-distance well

- The main problem with RNNs is that gradients less than 1 become exponentially small over time
- Known as the vanishing gradient problem
- Gradients greater than 1 become exponentially large over time (the exploding gradient problem)*

* The exploding gradient problem can be alleviated by clipping large gradient values to some maximum number

RNNs don't do long-distance well

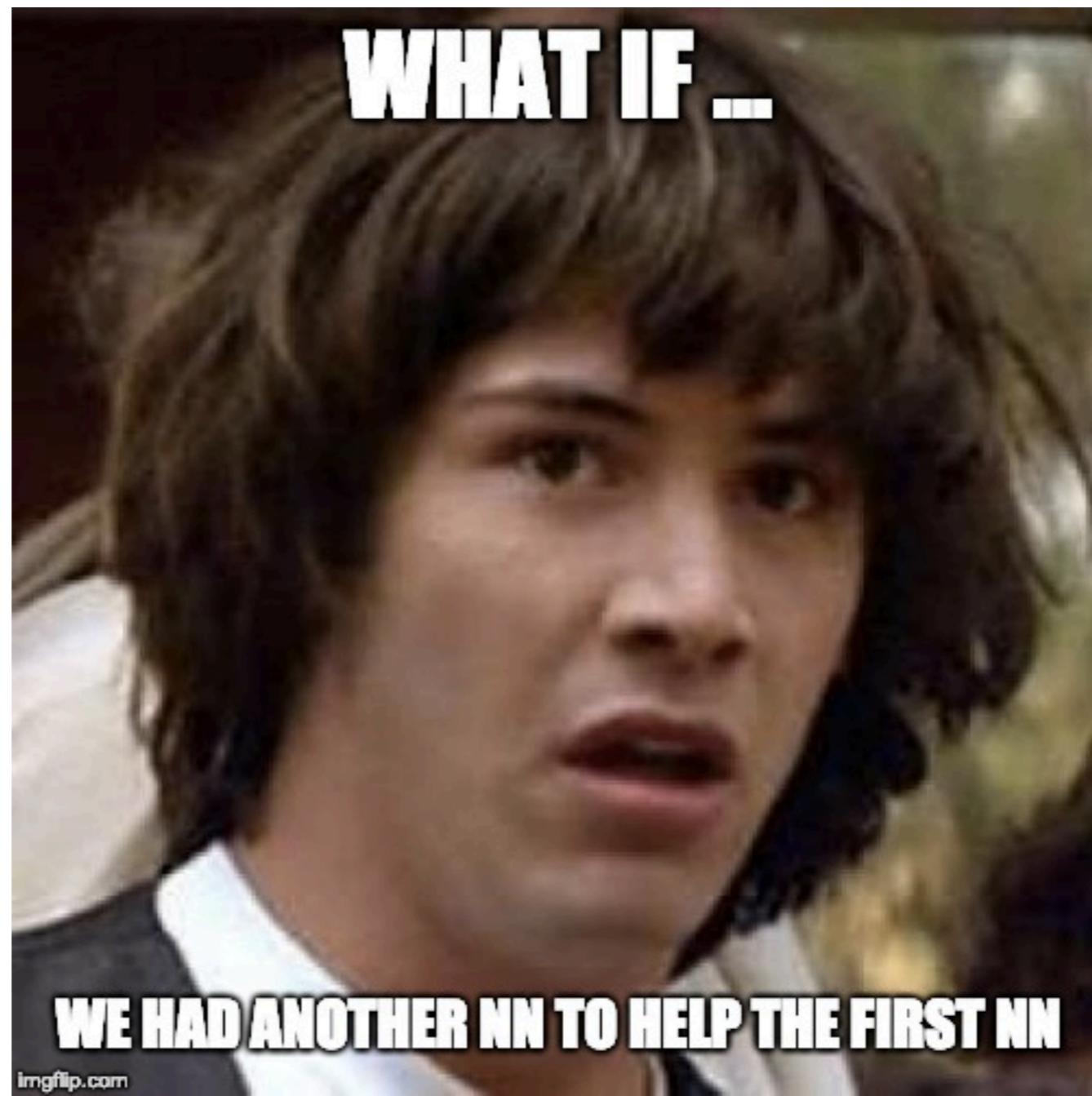
- The main problem with RNNs is that gradients less than 1 become exponentially small over time
- Known as the vanishing gradient problem
- Gradients greater than 1 become exponentially large over time (the exploding gradient problem)*
- This leads to training instability, and bad results

* The exploding gradient problem can be alleviated by clipping large gradient values to some maximum number

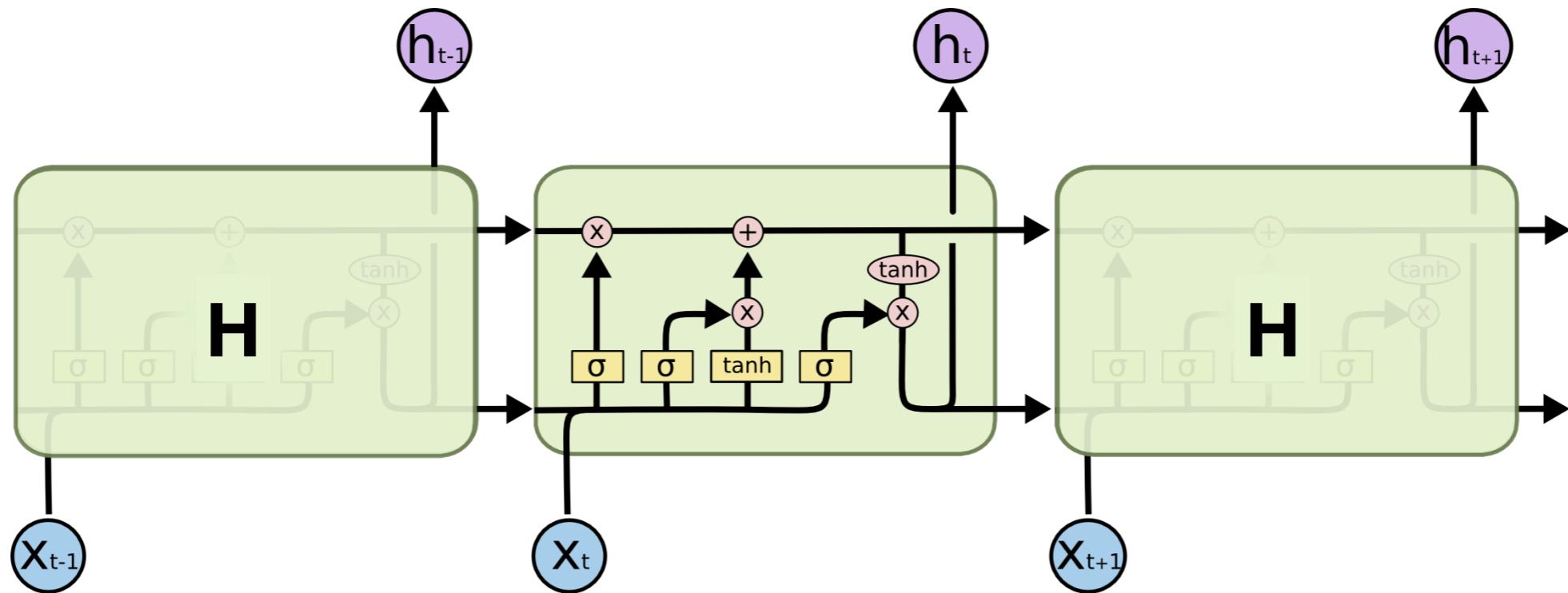
RNNs don't do long-distance well

- The main problem with RNNs is that gradients less than 1 become exponentially small over time
- Known as the vanishing gradient problem
- Gradients greater than 1 become exponentially large over time (the exploding gradient problem)*
- This leads to training instability, and bad results
- Sequence Modeling: <https://www.deeplearningbook.org/contents/rnn.html>

* The exploding gradient problem can be alleviated by clipping large gradient values to some maximum number



Long-Short Term Memory

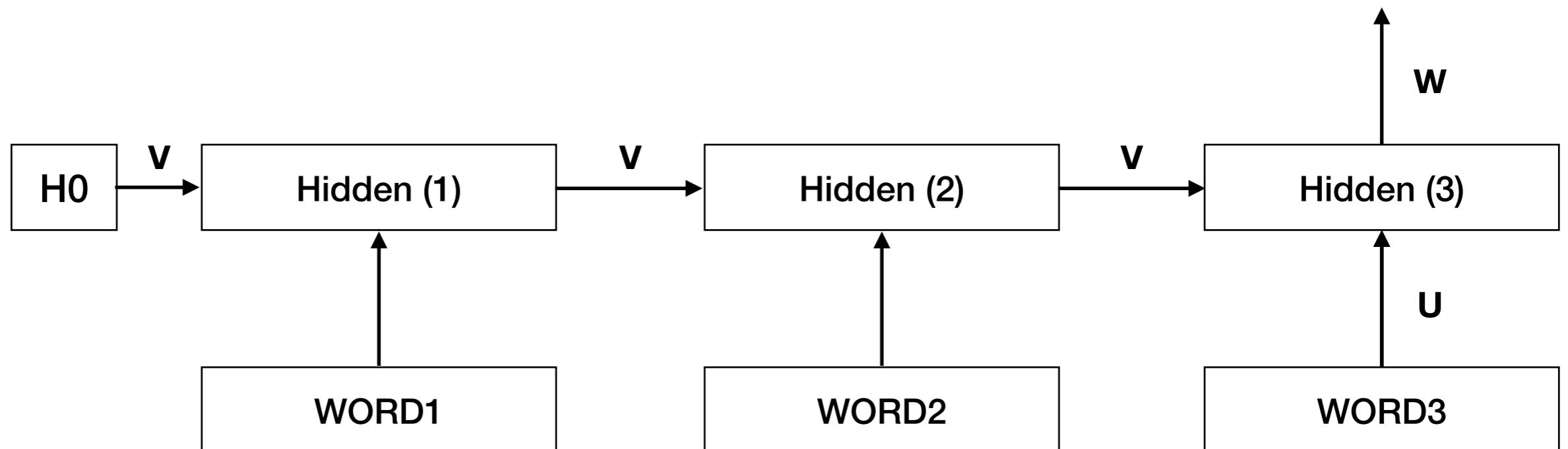


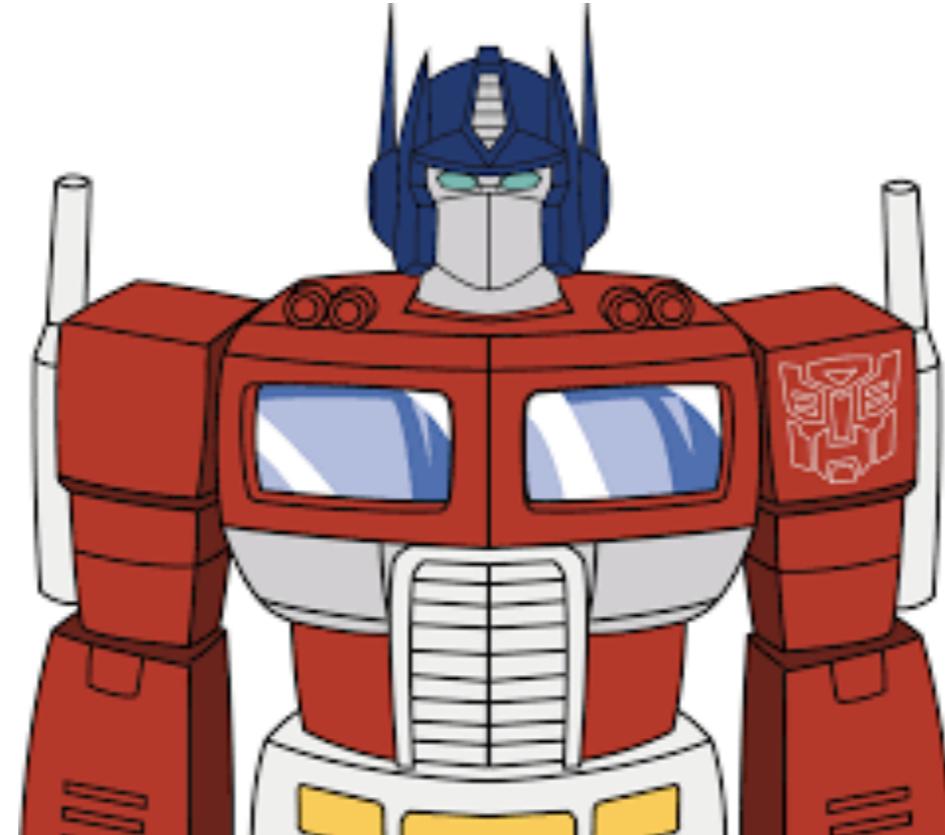
- Lets add another neural network help the first network learn long-term dependencies
- That's basically what we do when we add more weight matrices to a neural network

Image: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

LSTM Problems

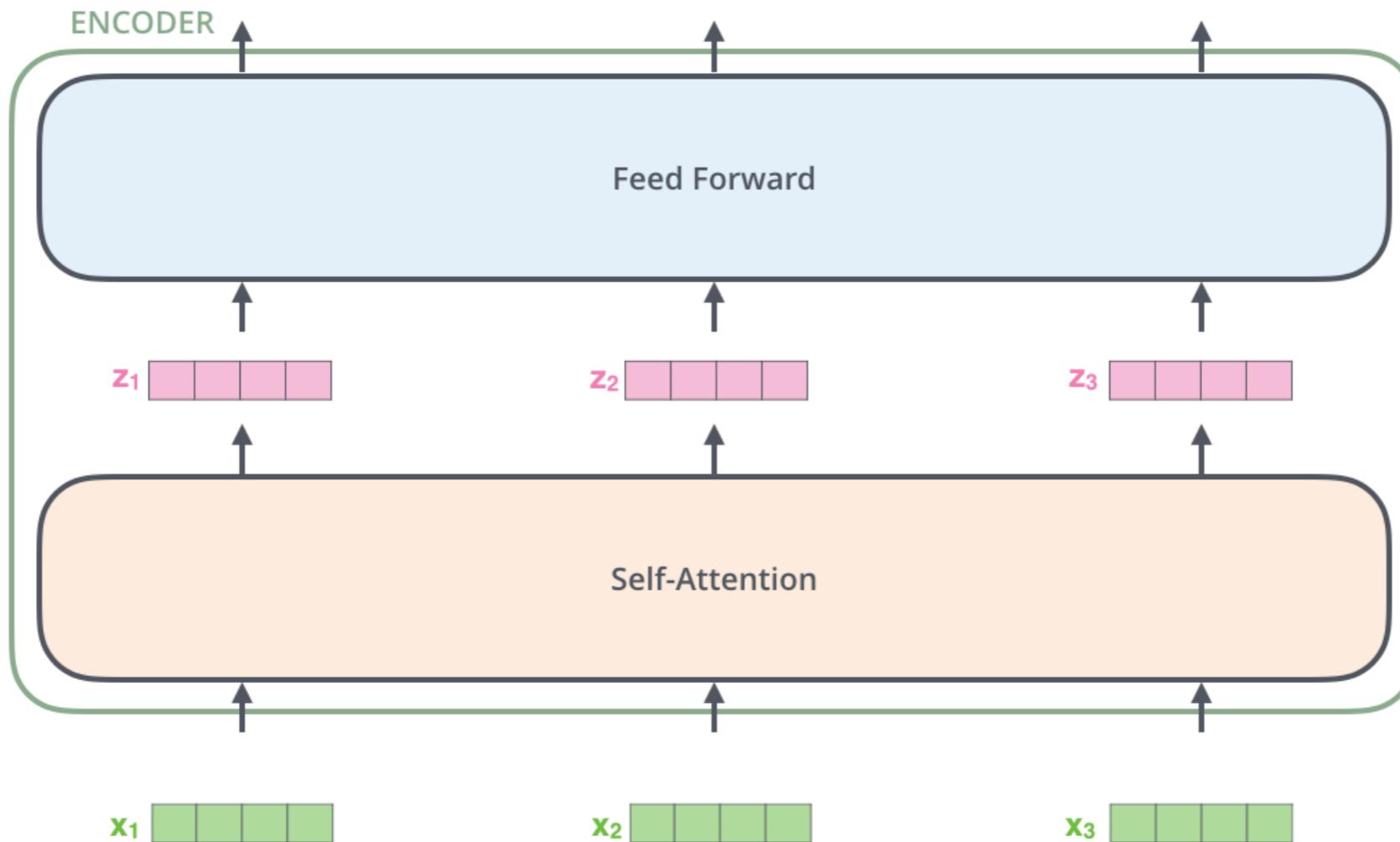
- Implicit representation of long-term information
 - Cell state and previous hidden state summarise the prior information





Transformers for Language Modelling

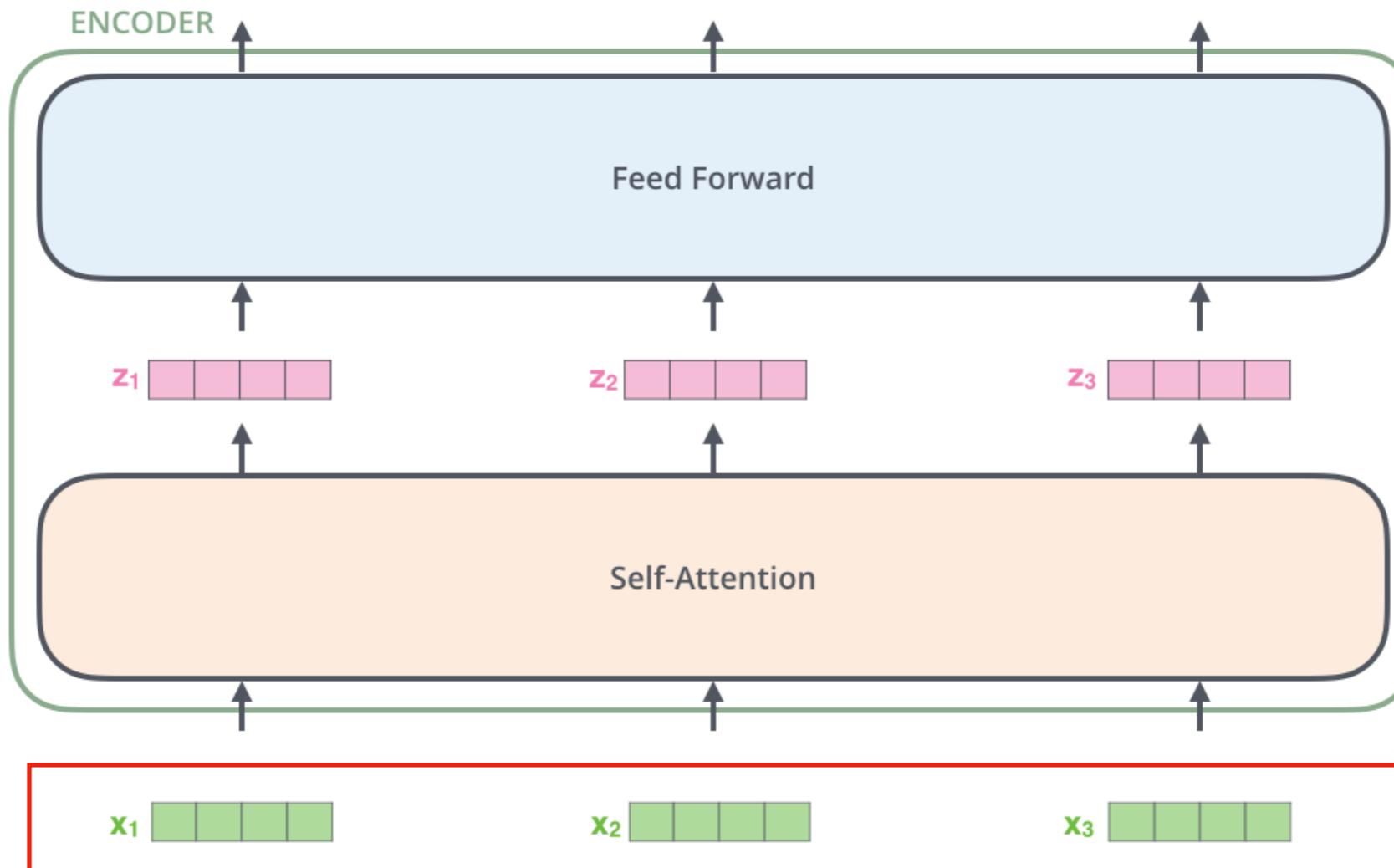
Transformers: Simplified



Multiple (50-90) such layers in a Transformer LM

Credit: <http://jalammar.github.io/illustrated-transformer/>

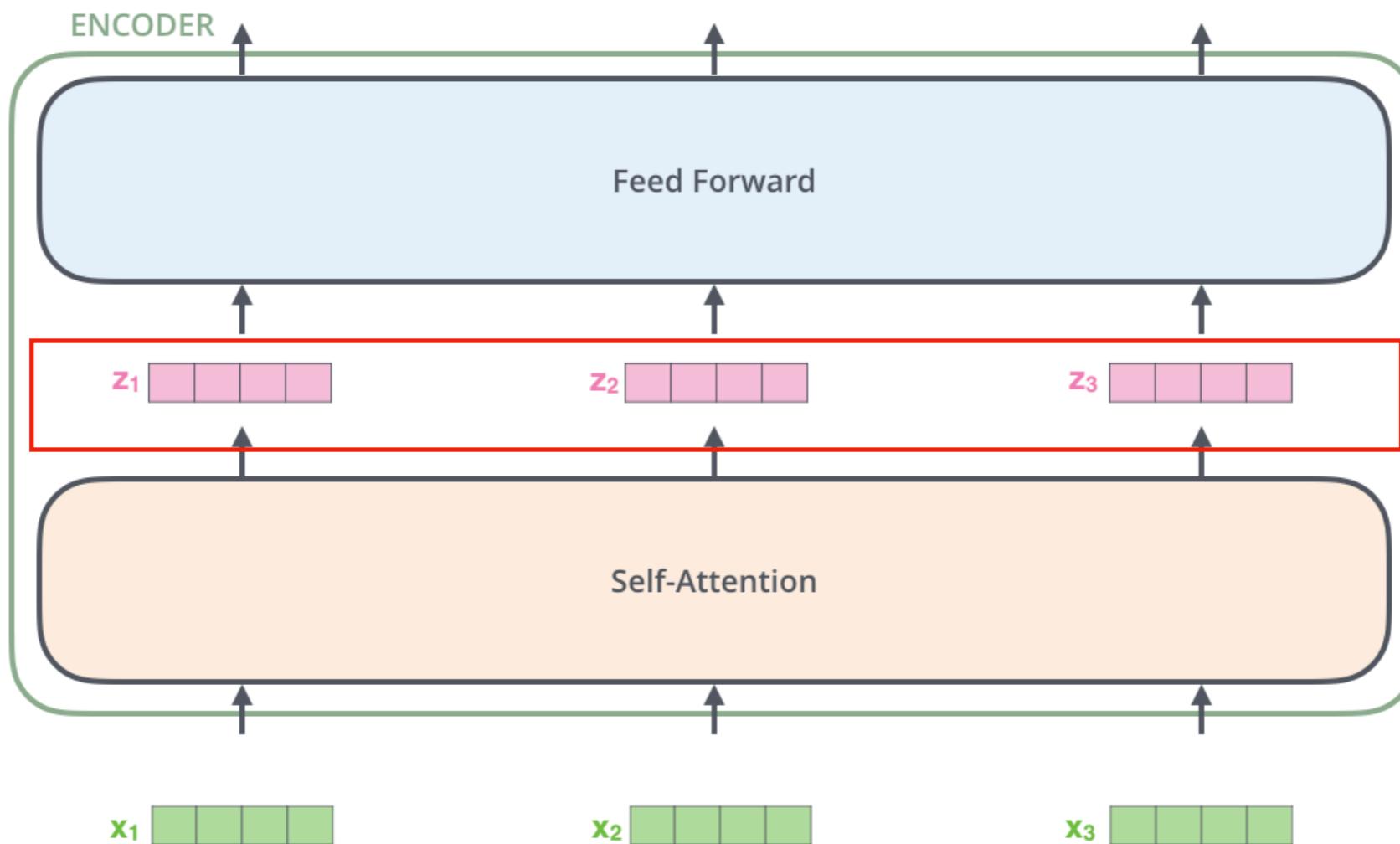
Transformers: Simplified



Multiple (50-90) such layers in a Transformer LM

Credit: <http://jalammar.github.io/illustrated-transformer/>

Transformers: Simplified



Multiple (50-90) such layers in a Transformer LM

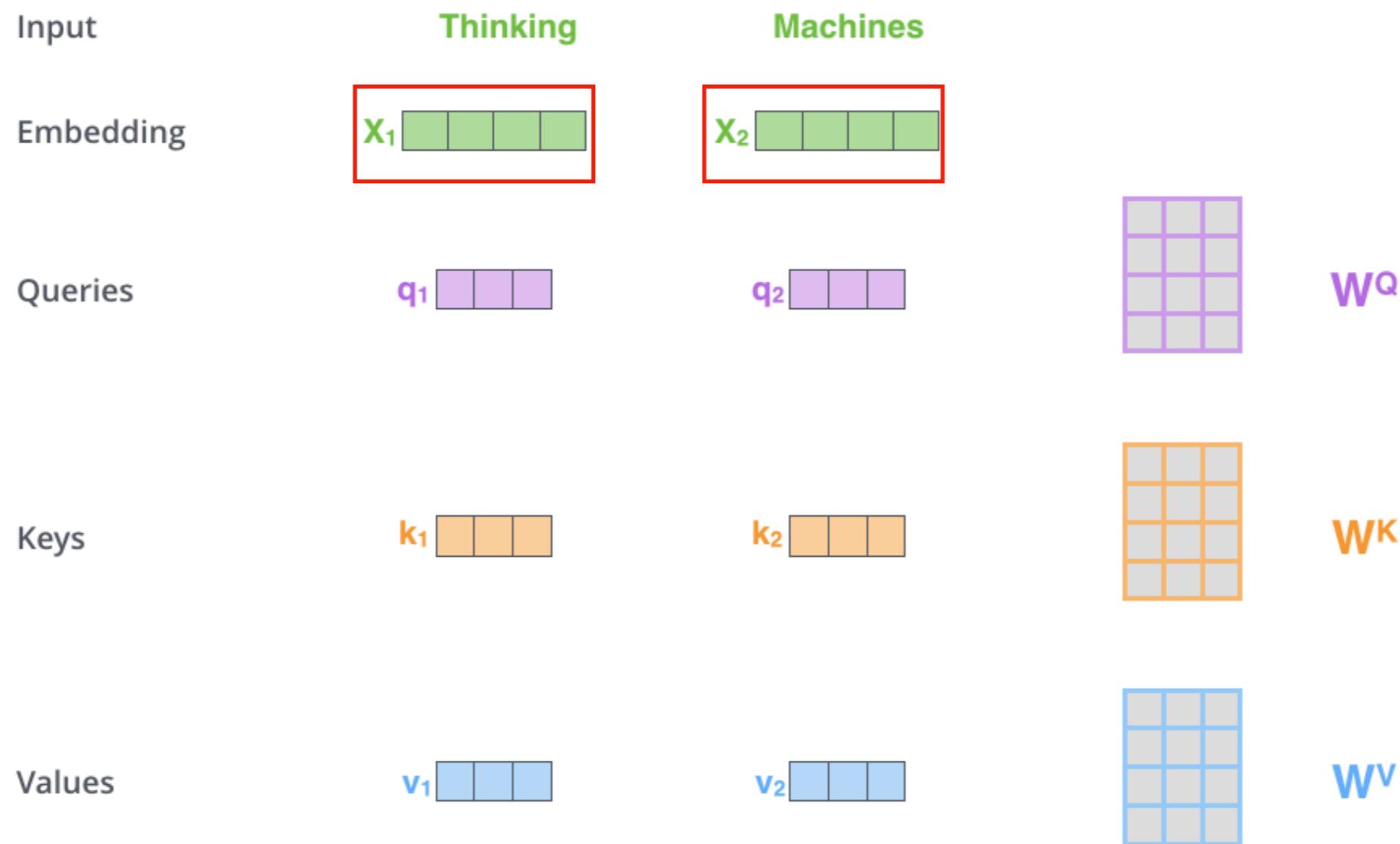
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention

- E.g. “The animal didn't cross the street because **it** was too tired”
- What does “**it**” refer to? “The animal” or “the street”
- Self-attention is the mechanism that helps LM associate:
 - “**it**” with “the animal”

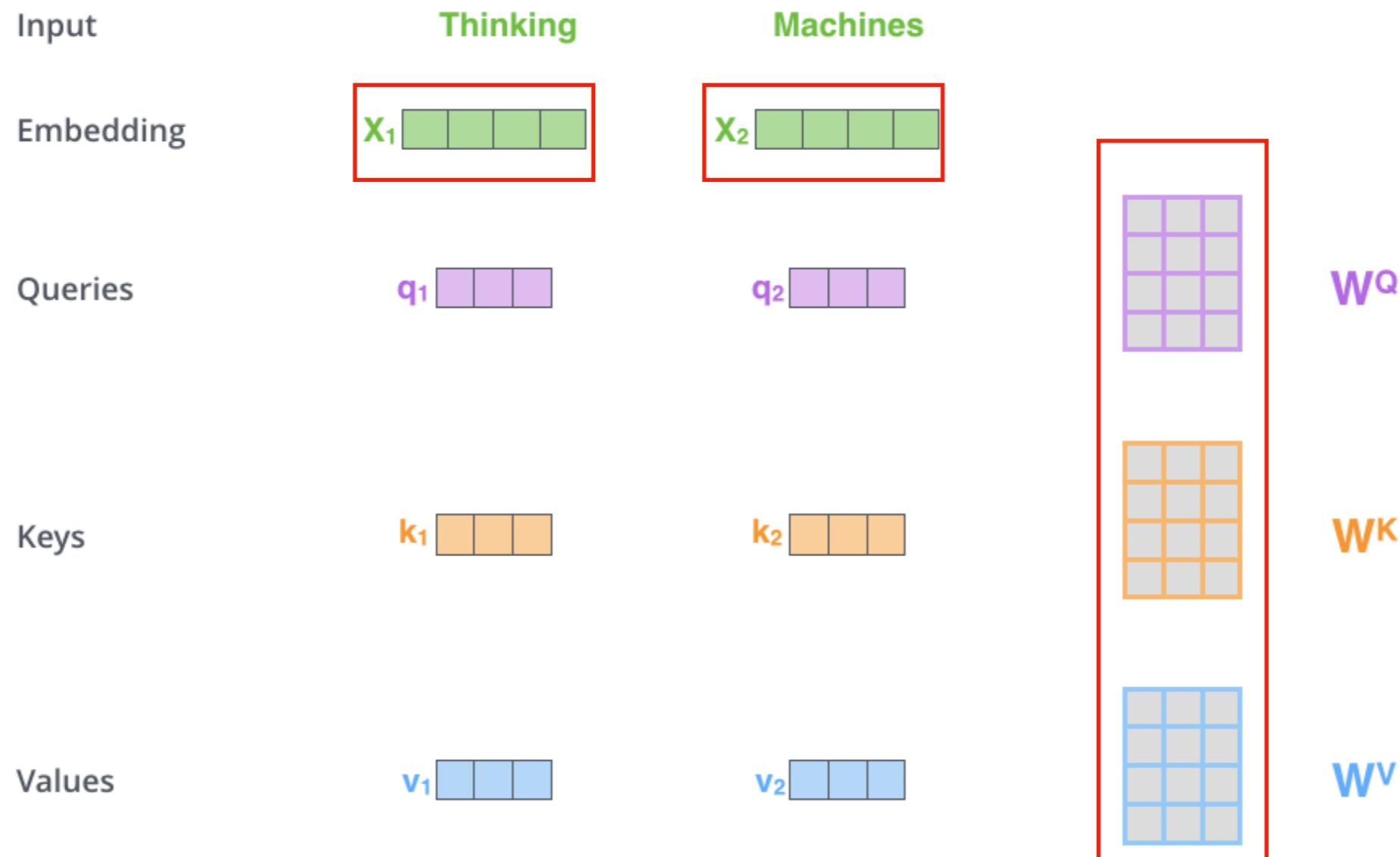
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



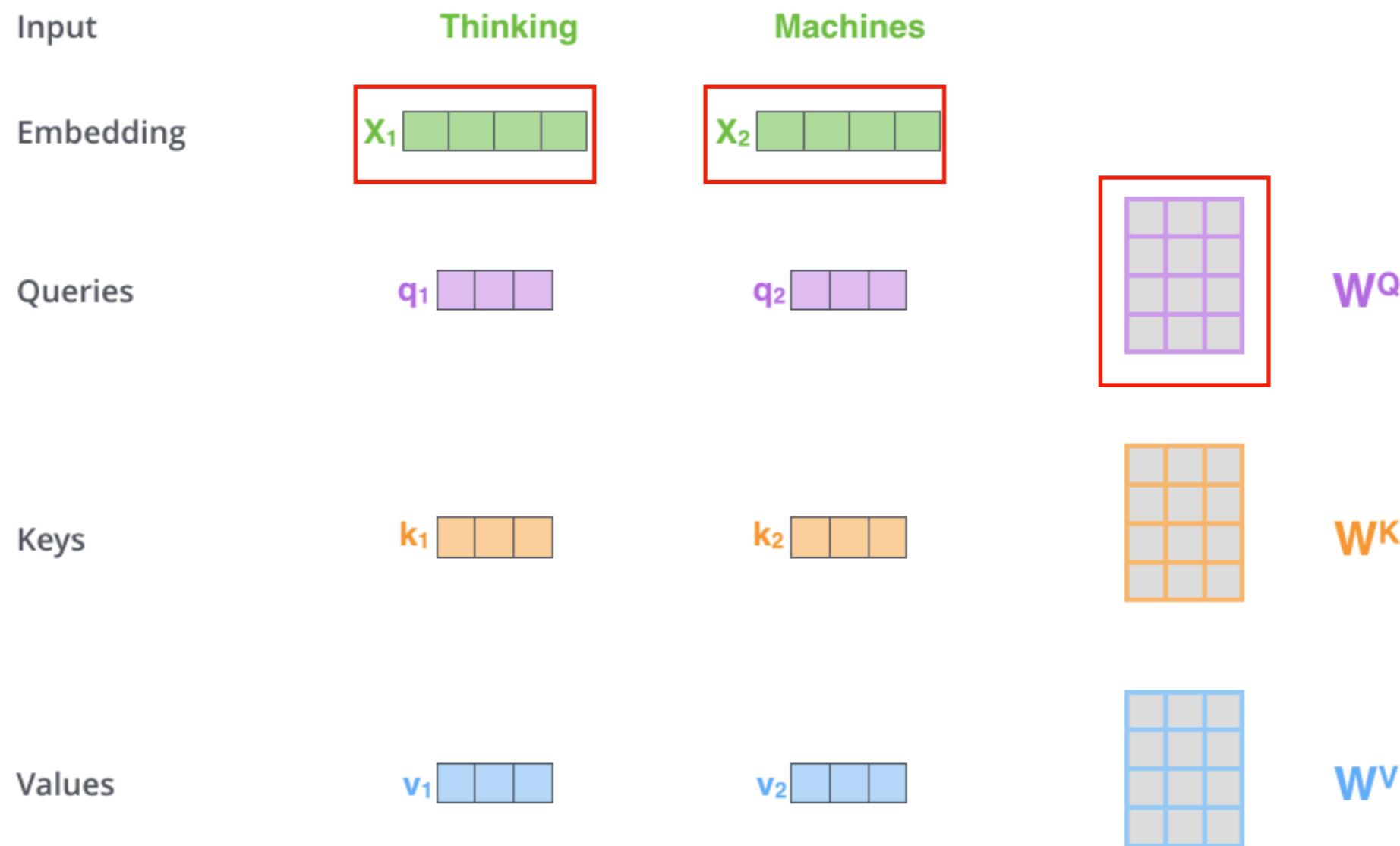
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



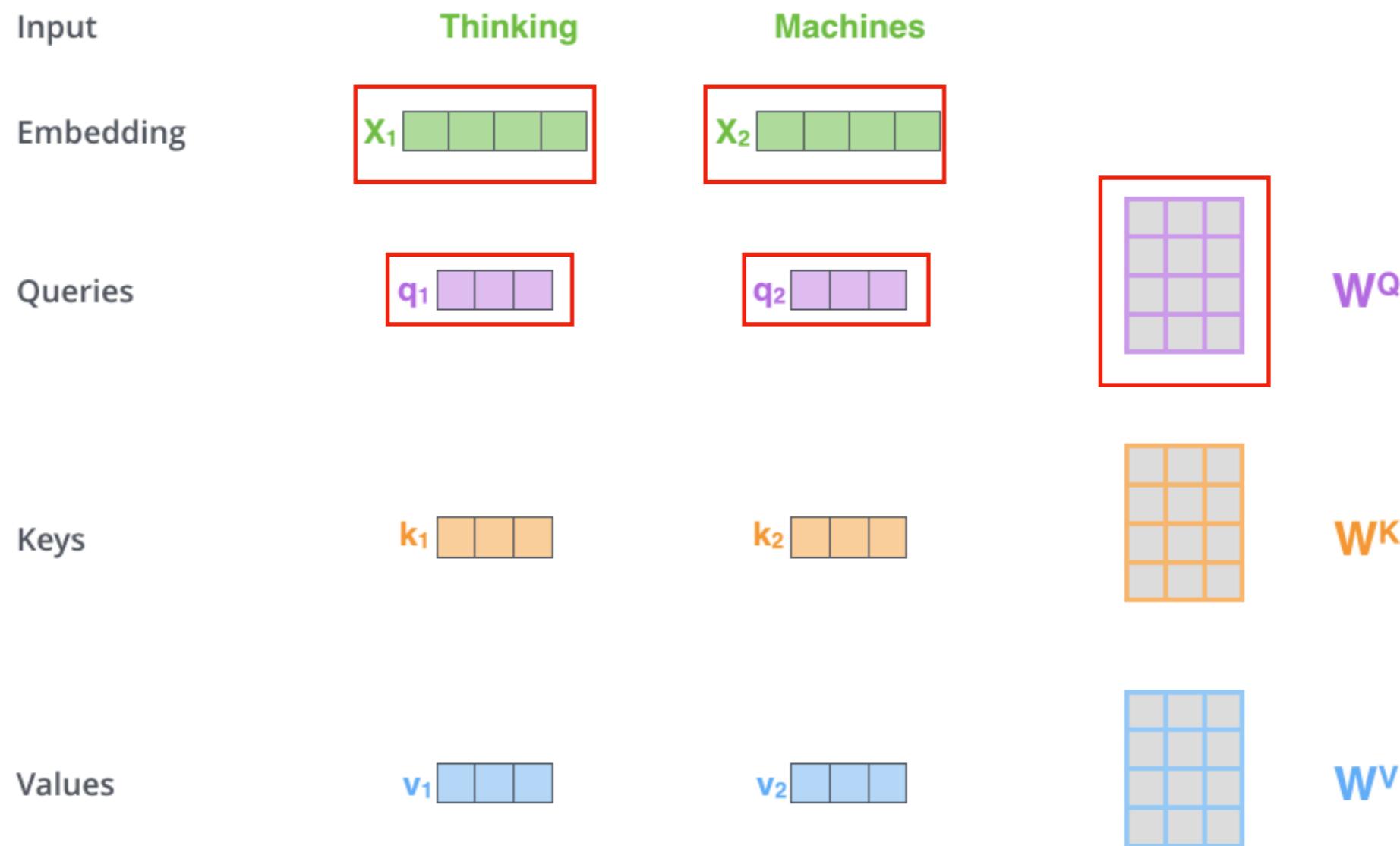
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



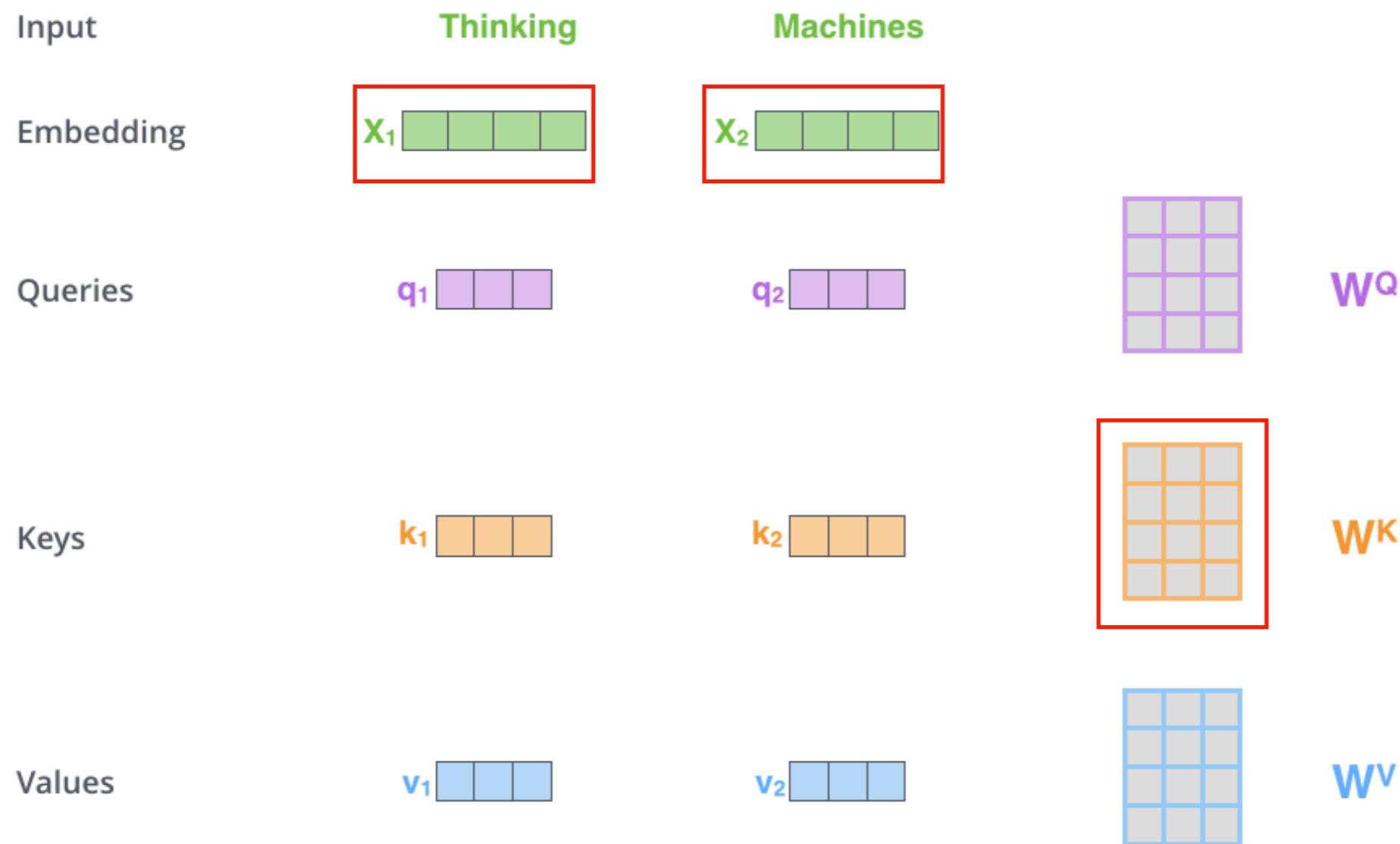
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



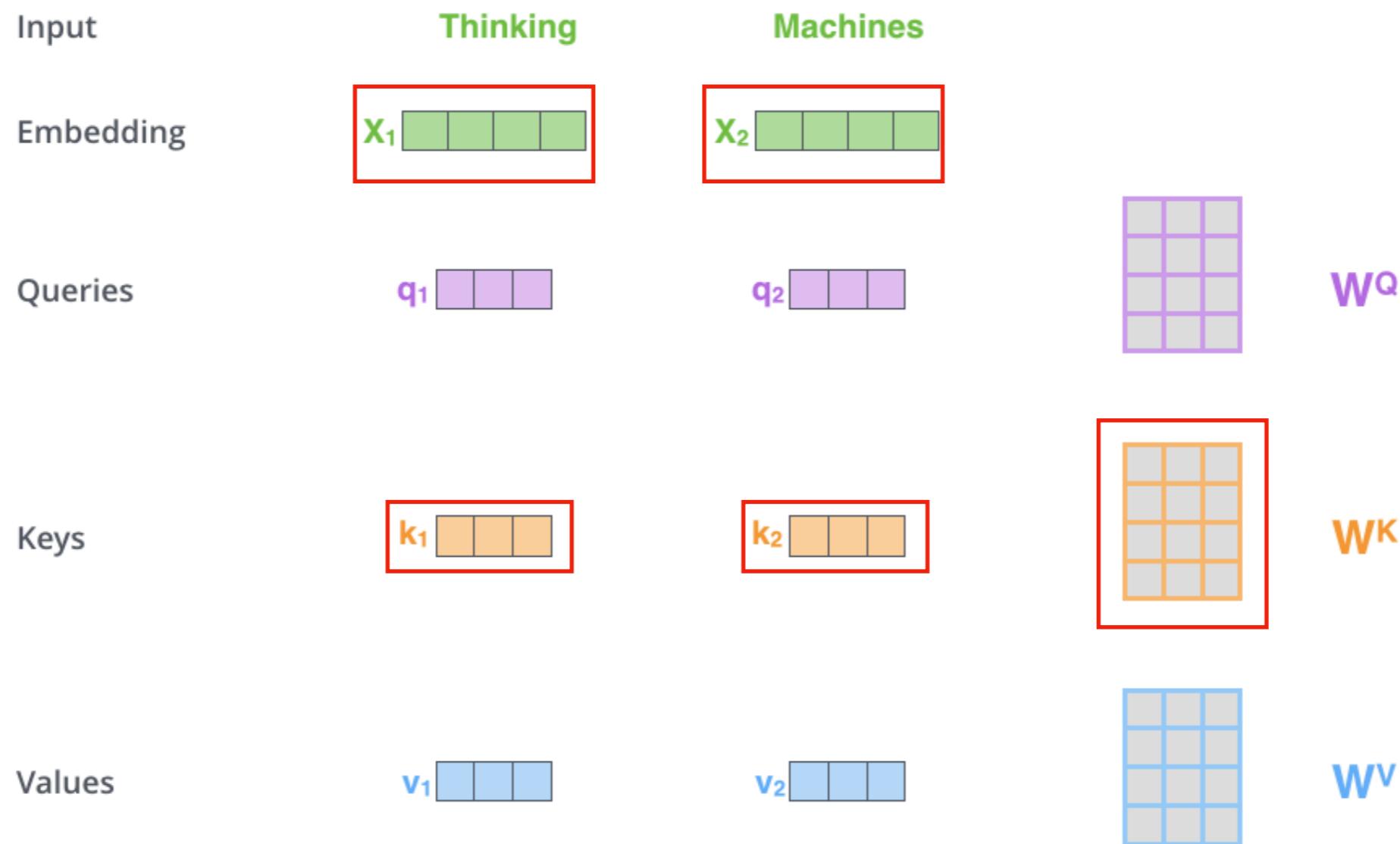
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



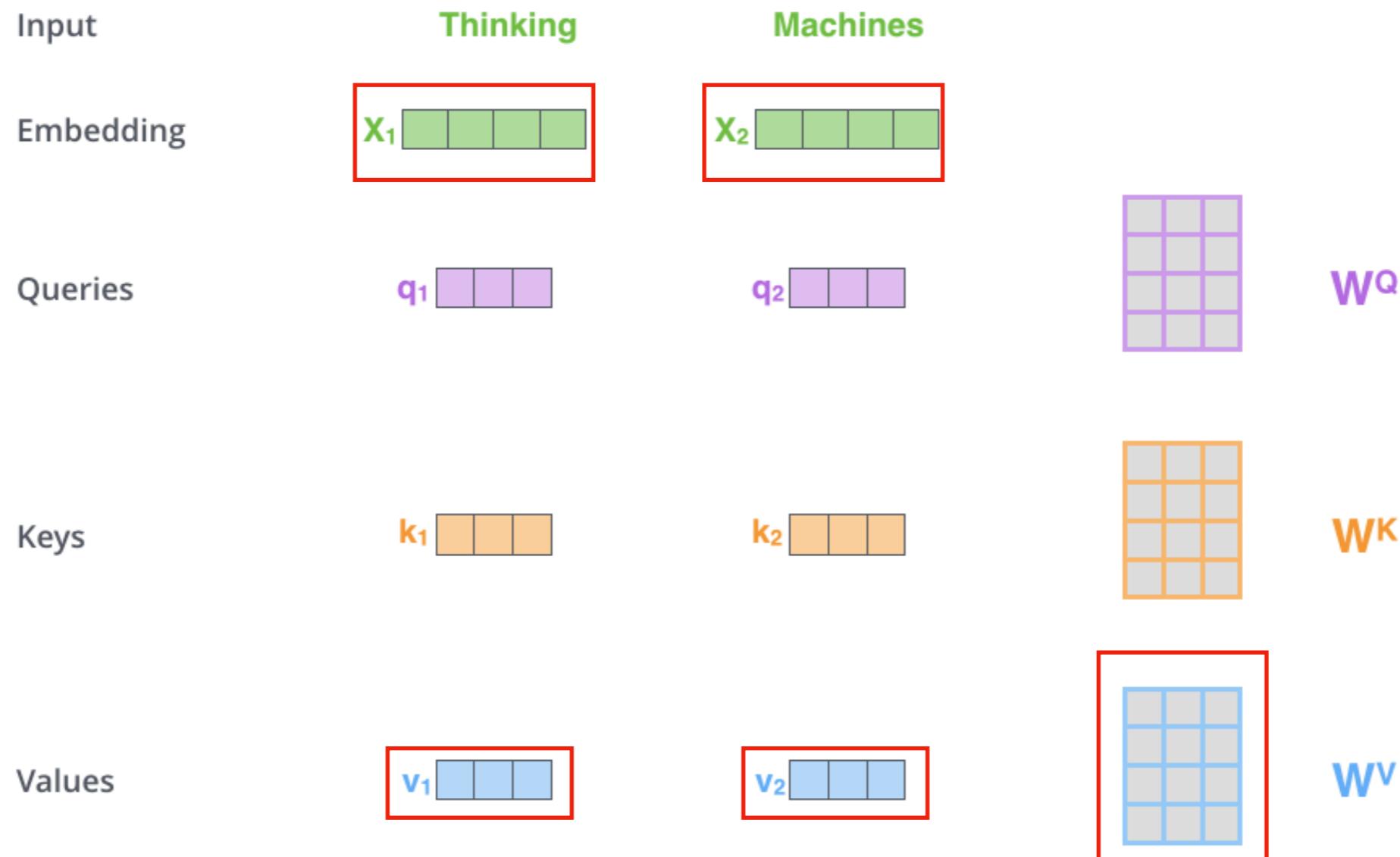
Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention: Step 0



Credit: <http://jalammar.github.io/illustrated-transformer/>

Input

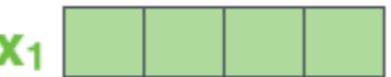
Embedding

Queries

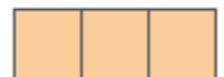
Keys

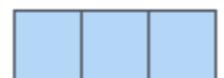
Values

Thinking

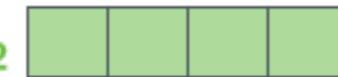
x_1 

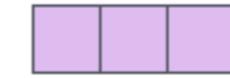
q_1 

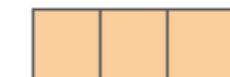
k_1 

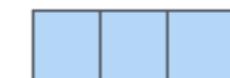
v_1 

Machines

x_2 

q_2 

k_2 

v_2 

Input

Embedding

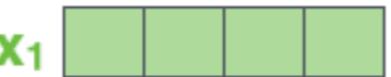
Queries

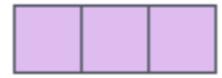
Keys

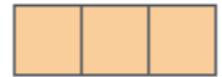
Values

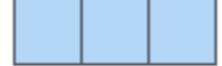
Score

Thinking

x_1 

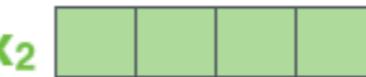
q_1 

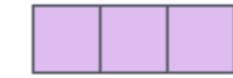
k_1 

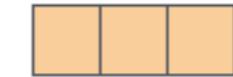
v_1 

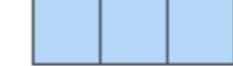
$$q_1 \cdot k_1 = 112$$

Machines

x_2 

q_2 

k_2 

v_2 

$$q_1 \cdot k_2 = 96$$

Input

Embedding

Queries

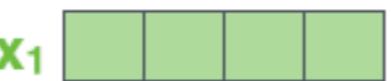
Keys

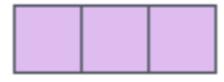
Values

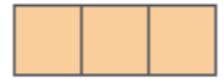
Score

Divide by 8 ($\sqrt{d_k}$)

Thinking

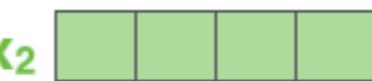
x_1 

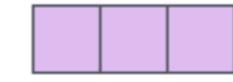
q_1 

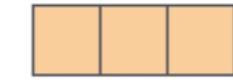
k_1 

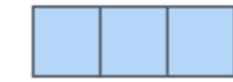
v_1 

Machines

x_2 

q_2 

k_2 

v_2 

$$q_1 \cdot k_1 = 112$$

14

$$q_1 \cdot k_2 = 96$$

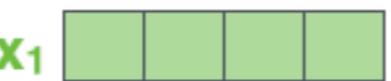
12

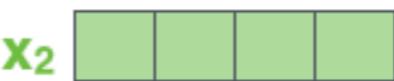
Input

Thinking

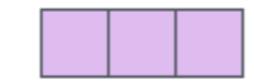
Machines

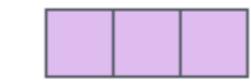
Embedding

x_1 

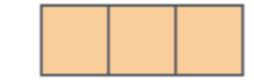
x_2 

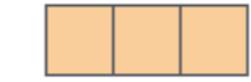
Queries

q_1 

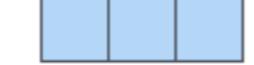
q_2 

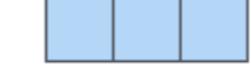
Keys

k_1 

k_2 

Values

v_1 

v_2 

Score

$$q_1 \cdot k_1 = 112$$

$$q_1 \cdot k_2 = 96$$

Divide by 8 ($\sqrt{d_k}$)

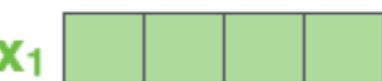
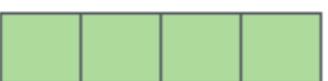
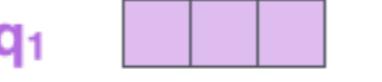
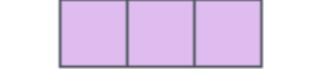
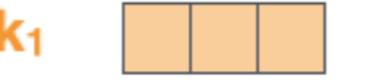
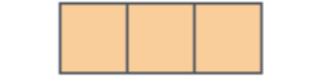
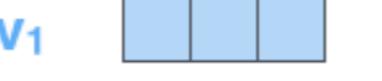
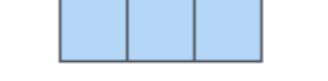
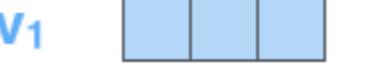
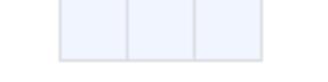
$$14$$

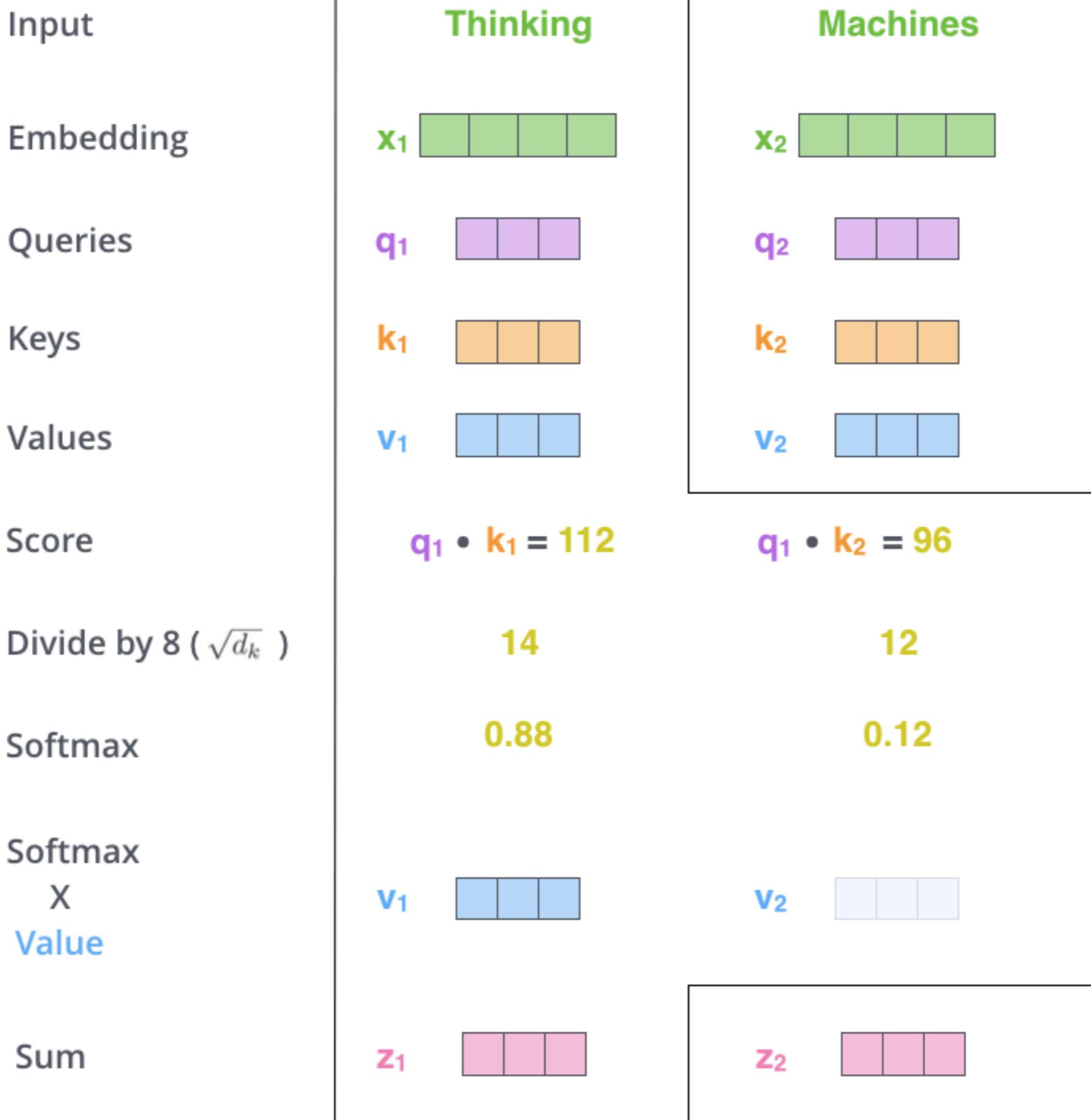
$$12$$

Softmax

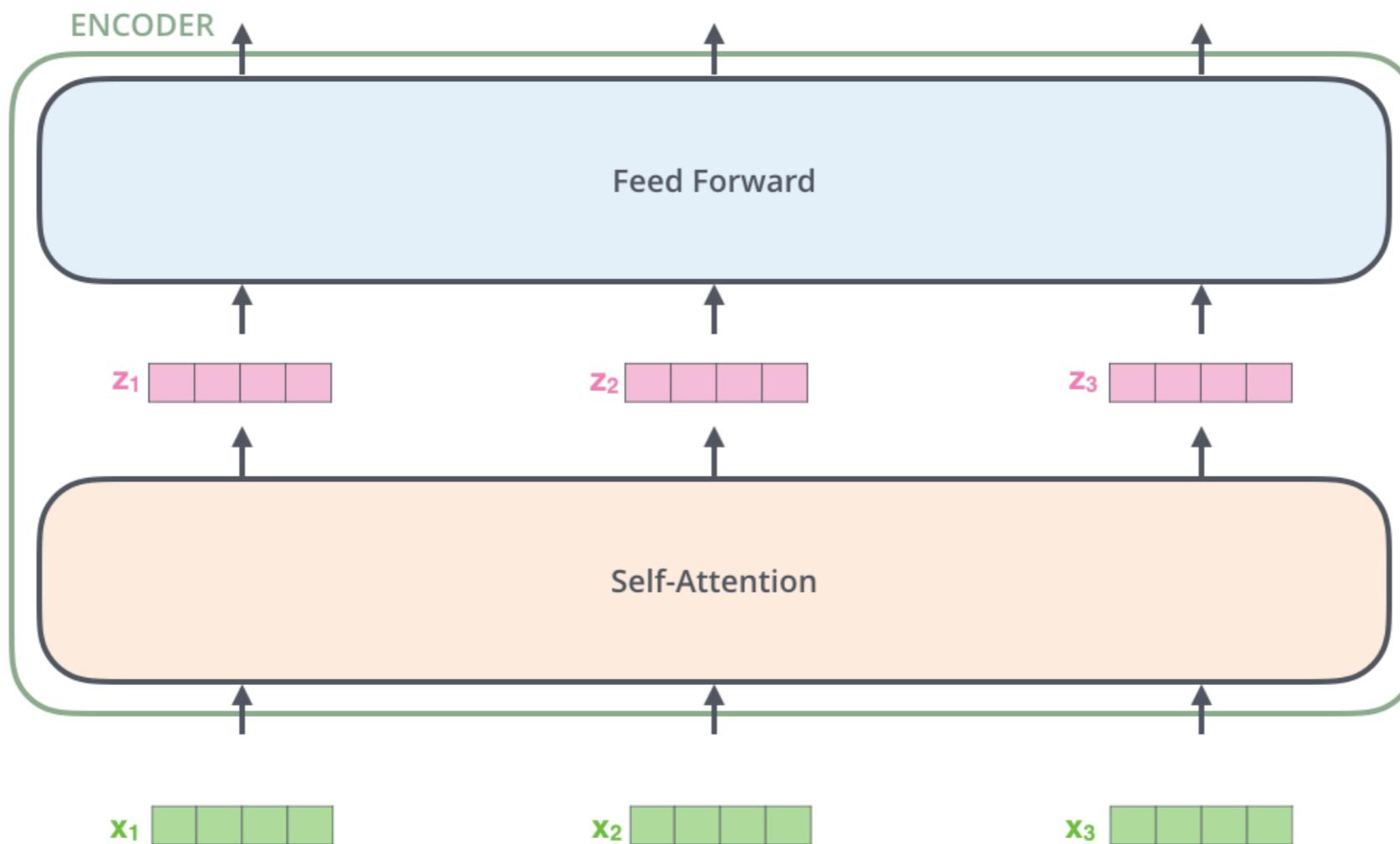
$$0.88$$

$$0.12$$

Input	Thinking x_1 		Machines x_2 
Embedding	q_1 		q_2 
Queries	k_1 		k_2 
Keys	v_1 		v_2 
Values	$q_1 \cdot k_1 = 112$ 14 0.88		$q_1 \cdot k_2 = 96$ 12 0.12
Score			
Divide by 8 ($\sqrt{d_k}$)			
Softmax			
Softmax X Value	v_1 		v_2 



Transformers: Simplified



Credit: <http://jalammar.github.io/illustrated-transformer/>

Self-Attention

- Self-Attention seems to be asking an association question

Self-Attention

- Self-Attention seems to be asking an association question
- Query ~ smaller word embedding

Self-Attention

- Self-Attention seems to be asking an association question
- Query ~ smaller word embedding
- Key & Value ~ Key is the hash key that maps to Value

Self-Attention

- Self-Attention seems to be asking an association question
- Query ~ smaller word embedding
- Key & Value ~ Key is the hash key that maps to Value
- The names Query, Key and Value come from retrieval parlance

Self-Attention

- Self-Attention seems to be asking an association question
- Query ~ smaller word embedding
- Key & Value ~ Key is the hash key that maps to Value
- The names Query, Key and Value come from retrieval parlance
 - you fire a query, you compare to a key vector and return the value

Self-attention: exercise

- “Computers are thinking machines”
- Compute z for machines
- $Q = K = V = \begin{bmatrix} 0.2 & 0.8 \\ -0.2 & 0.5 \\ -0.3 & -0.4 \\ 0.7 & 0.7 \end{bmatrix}$
- Computers = [1 0 0 0], are = [0 1 0 0], thinking = [0 0 1 0], machines = [0 0 0 1]
- Softmax

Self-attention: exercise

- “Computers are thinking machines”
- Compute z for machines
- $Q = K = V = \begin{bmatrix} 0.2 & 0.8 \\ -0.2 & 0.5 \\ -0.3 & -0.4 \\ 0.7 & 0.7 \end{bmatrix}$
- Computers = [1 0 0 0], are = [0 1 0 0], thinking = [0 0 1 0],
machines = [0 0 0 1]
- Softmax $z = [0.24 \quad 0.55]$

Input Computers are thinking machines

Embedding

Queries

Keys

Values

Score $q \cdot k$

Divide by $\sqrt{2(d_k)}$

Softmax

Softmax

X

Value

Sum

Input Computers are thinking machines

Embedding [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

Queries

Keys

Values

Score $q \cdot k$

Divide by $\sqrt{2}(\sqrt{d_k})$

Softmax

Softmax

X

Value

Sum

Input Computers are thinking machines

Embedding [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

Queries [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Keys

Values

Score $q \cdot k$

Divide by $\sqrt{2(d_k)}$

Softmax

Softmax

X

Value

Sum

Input Computers are thinking machines

Embedding [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

Queries [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Keys [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Values

Score $q \cdot k$

Divide by $\sqrt{2(d_k)}$

Softmax

Softmax

X

Value

Sum

Input Computers are thinking machines

Embedding [1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 0 1]

Queries [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Keys [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Values [0.2 0.8] [-0.2 0.5] [-0.3 -0.4] [0.7 0.7]

Score $q \cdot k$

Divide by $\sqrt{2(d_k)}$

Softmax

Softmax

X

Value

Sum

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score	$q \cdot k$	0.7	0.21	-0.49
	Divide by $\sqrt{2(d_k)}$			
	Softmax			
	Softmax			
	X			
	Value			
	Sum			

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score	$q \cdot k$	0.7	0.21	-0.49
	Divide by $\sqrt{2(d_k)}$			
	Softmax			
	Softmax			
	X			
	Value			
	Sum			

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2(d_k)}$				
Softmax				
Softmax				
X				
Value				
Sum				

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2(d_k)}$				
Softmax				
Softmax				
X				
Value				
Sum				

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2(d_k)}$				
Softmax				
Softmax				
X				
Value				
Sum				

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2(d_k)}$				
Softmax				
Softmax				
X				
Value				
Sum				

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score	$q \cdot k$	0.7	0.21	-0.49
	Divide by $\sqrt{2(d_k)}$			
	Softmax			
	Softmax			
	X			
	Value			
	Sum			

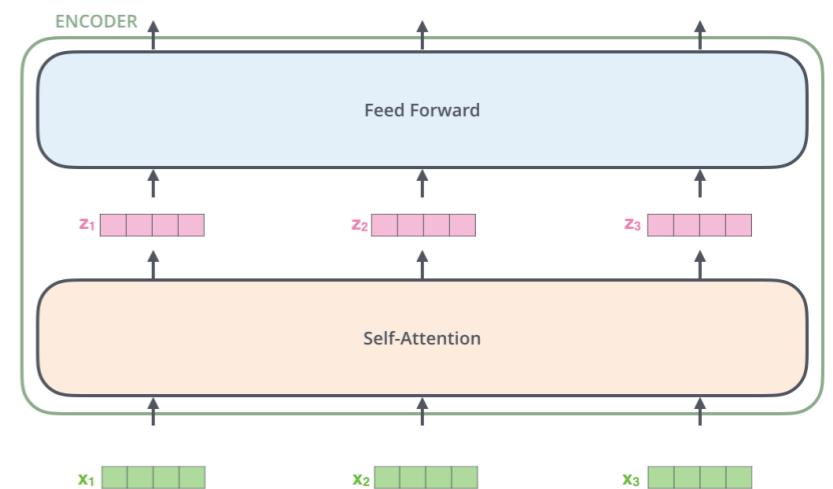
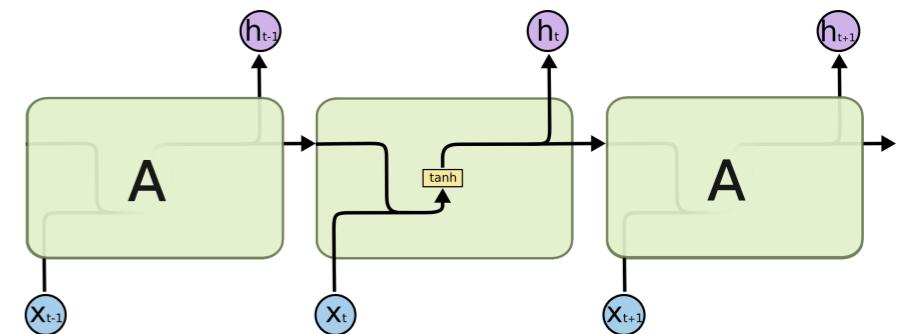
Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2(d_k)}$	0.49	0.15	-0.35	0.69
Softmax				
Softmax X Value				
Sum				

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score	$q \cdot k$	0.7	0.21	-0.49
	Divide by $\sqrt{2}(\sqrt{d_k})$	0.49	0.15	-0.35
	Softmax	0.30	0.21	0.13
	Softmax X Value			0.36
	Sum			

Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2}(\sqrt{d_k})$	0.49	0.15	-0.35	0.69
Softmax	0.30	0.21	0.13	0.36
Softmax x Value	[0.06 0.24]	[-0.04 0.10]	[-0.04 -0.05]	[0.25 0.25]
Sum				

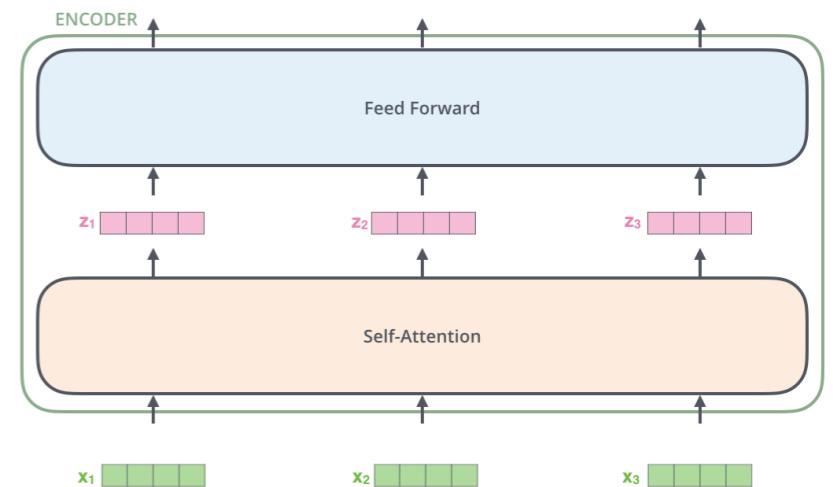
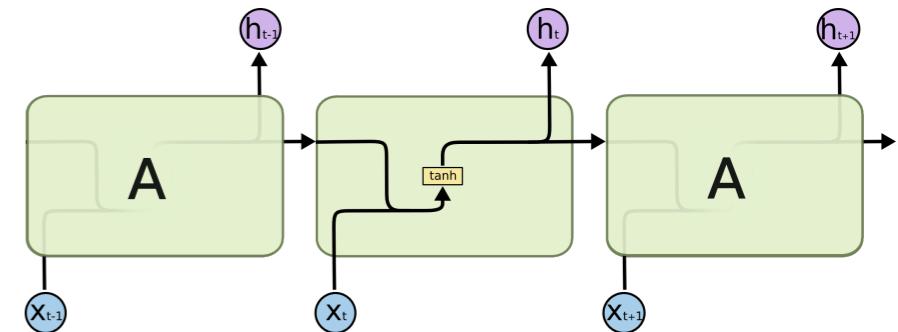
Input	Computers	are	thinking	machines
Embedding	[1 0 0 0]	[0 1 0 0]	[0 0 1 0]	[0 0 0 1]
Queries	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Keys	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Values	[0.2 0.8]	[-0.2 0.5]	[-0.3 -0.4]	[0.7 0.7]
Score q·k	0.7	0.21	-0.49	0.98
Divide by $\sqrt{2}(\sqrt{d_k})$	0.49	0.15	-0.35	0.69
Softmax	0.30	0.21	0.13	0.36
Softmax x Value	[0.06 0.24]	[-0.04 0.10]	[-0.04 -0.05]	[0.25 0.25]
Sum				[0.23 0.54]

Transformers for Language Modelling



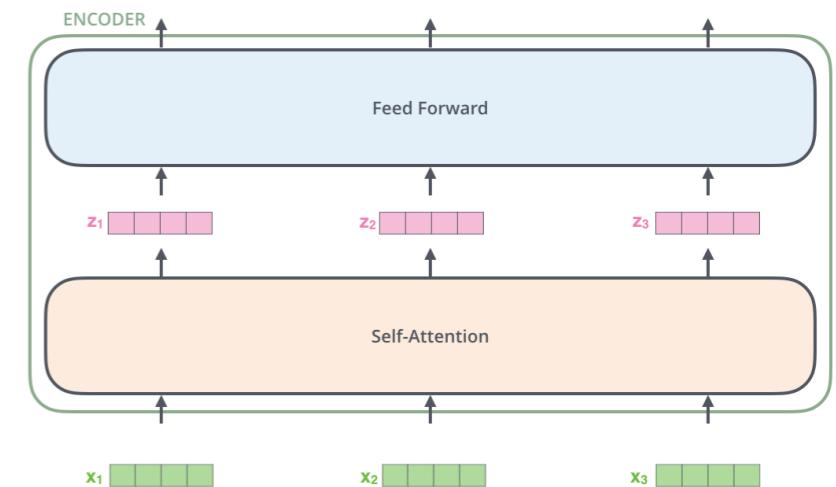
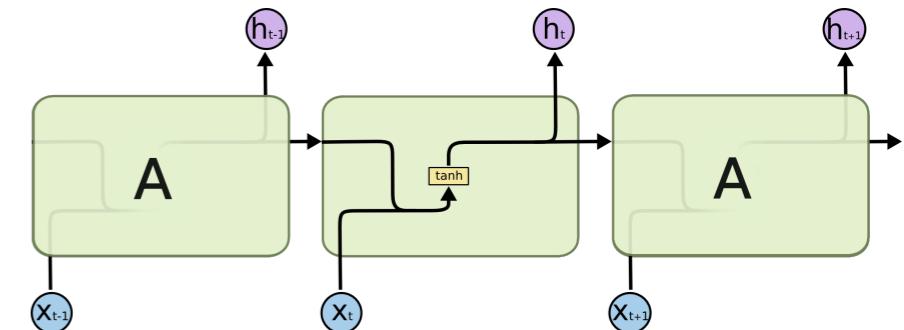
Transformers for Language Modelling

- RNNs: Process tokens one-by-one



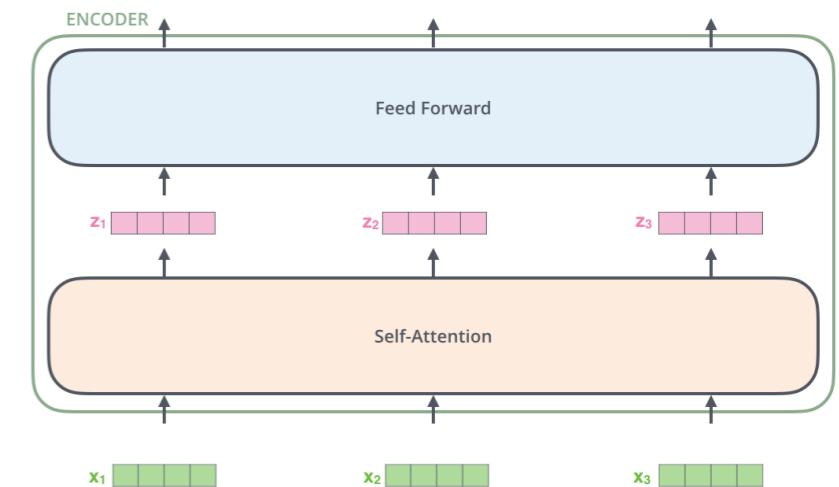
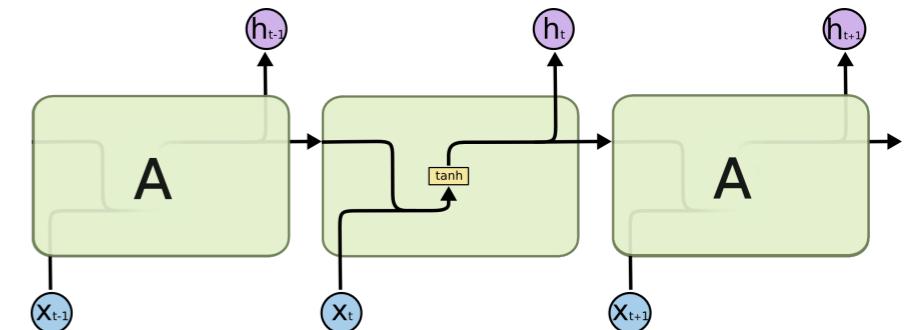
Transformers for Language Modelling

- RNNs: Process tokens one-by-one
 - Chain of dependencies built using a single token



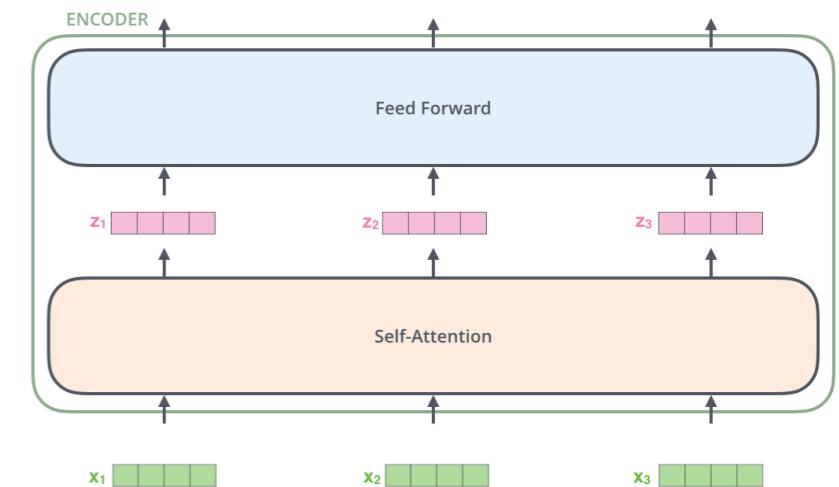
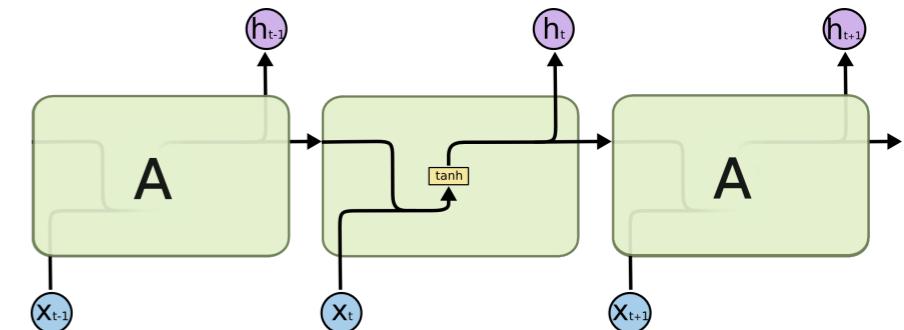
Transformers for Language Modelling

- RNNs: Process tokens one-by-one
 - Chain of dependencies built using a single token
- Transformers LM: Process a segment of tokens



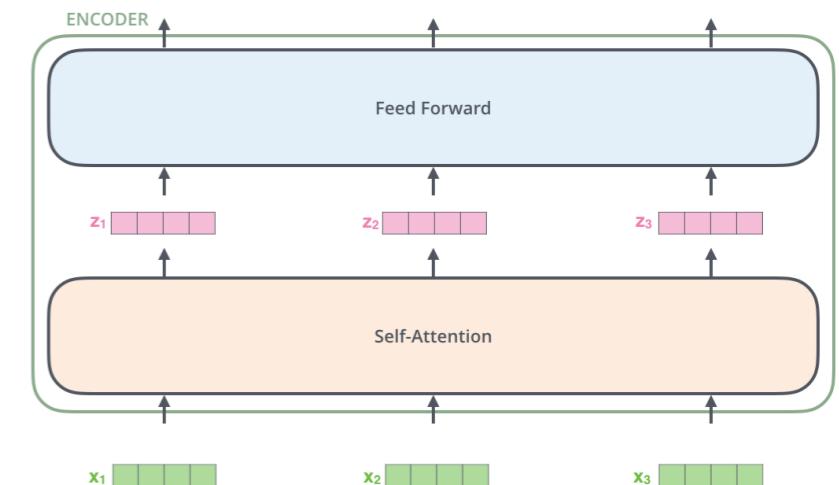
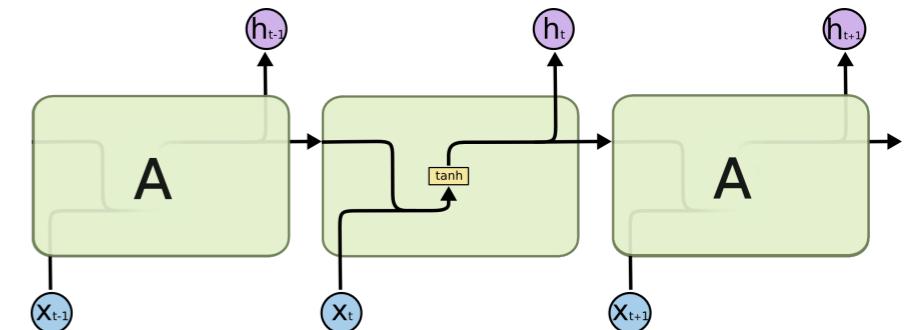
Transformers for Language Modelling

- RNNs: Process tokens one-by-one
 - Chain of dependencies built using a single token
- Transformers LM: Process a segment of tokens
 - Dependencies within the segment



Transformers for Language Modelling

- RNNs: Process tokens one-by-one
 - Chain of dependencies built using a single token
- Transformers LM: Process a segment of tokens
 - Dependencies within the segment
 - Within segment position is given by the positional encoding





[Jacob Devlin et al 2018]

Image credit: <https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/>

BERT: Bidirectional Encoder Representations from Transformers

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data
- Deep learning requires large amounts of annotated data

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data
- Deep learning requires large amounts of annotated data
- Language models for general purpose representations

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data
- Deep learning requires large amounts of annotated data
- Language models for general purpose representations
- Aim to **pretrain** general purpose representations

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data
- Deep learning requires large amounts of annotated data
- Language models for general purpose representations
- Aim to **pretrain** general purpose representations
- that can be **fine-tuned** using small task-specific dataset to obtain good performance on specialised tasks

BERT: Bidirectional Encoder Representations from Transformers

- For specialised tasks like named entity recognition, question answering there is a lack of training data
- Deep learning requires large amounts of annotated data
- Language models for general purpose representations
- Aim to **pretrain** general purpose representations
- that can be **fine-tuned** using small task-specific dataset to obtain good performance on specialised tasks
- Welcome BERT!



Transformers for Language Modelling

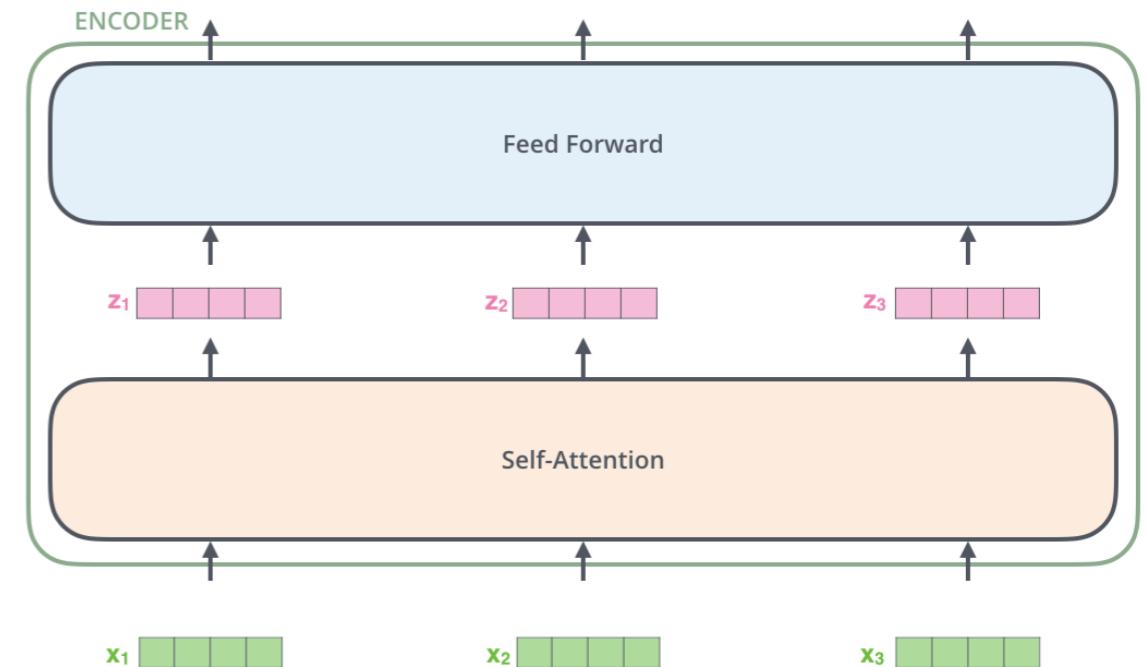


Image Credit: <https://arxiv.org/pdf/1706.03762.pdf>
Content Credit: [TransformerXL Explained & Al-Rfou et al. 2018](#)

Transformers for Language Modelling

- Transformers LM

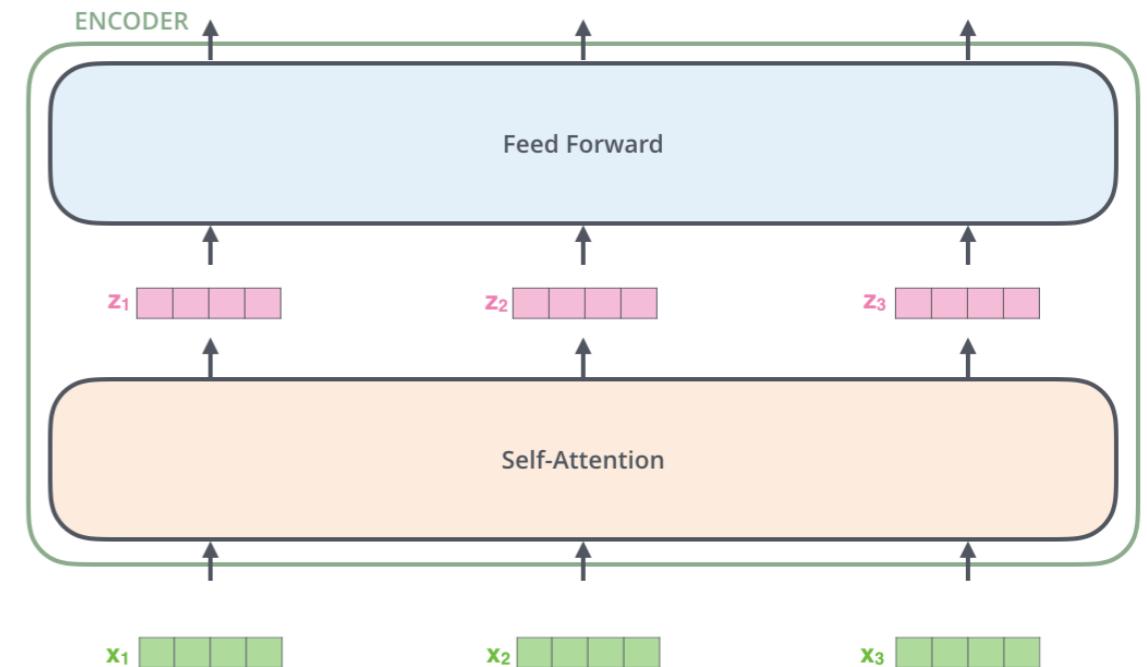


Image Credit: <https://arxiv.org/pdf/1706.03762.pdf>
Content Credit: [TransformerXL Explained & Al-Rfou et al. 2018](#)

Transformers for Language Modelling

- Transformers LM
 - Unidirectional

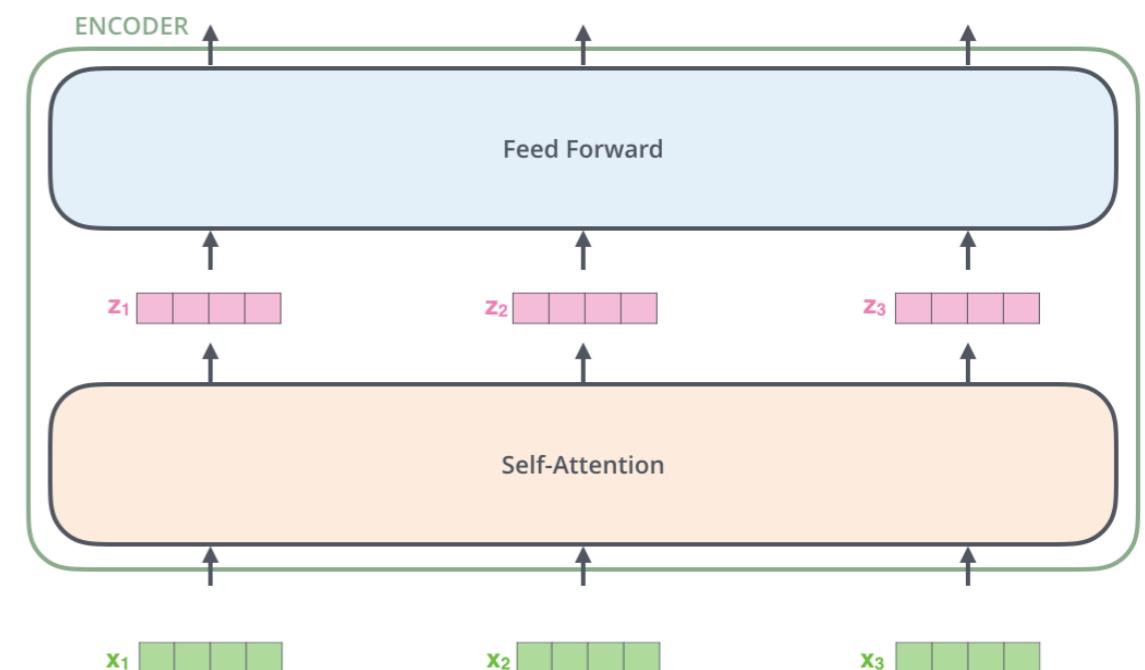


Image Credit: <https://arxiv.org/pdf/1706.03762.pdf>
Content Credit: [TransformerXL Explained & Al-Rfou et al. 2018](#)

Transformers for Language Modelling

- Transformers LM
 - Unidirectional
 - Segment of tokens

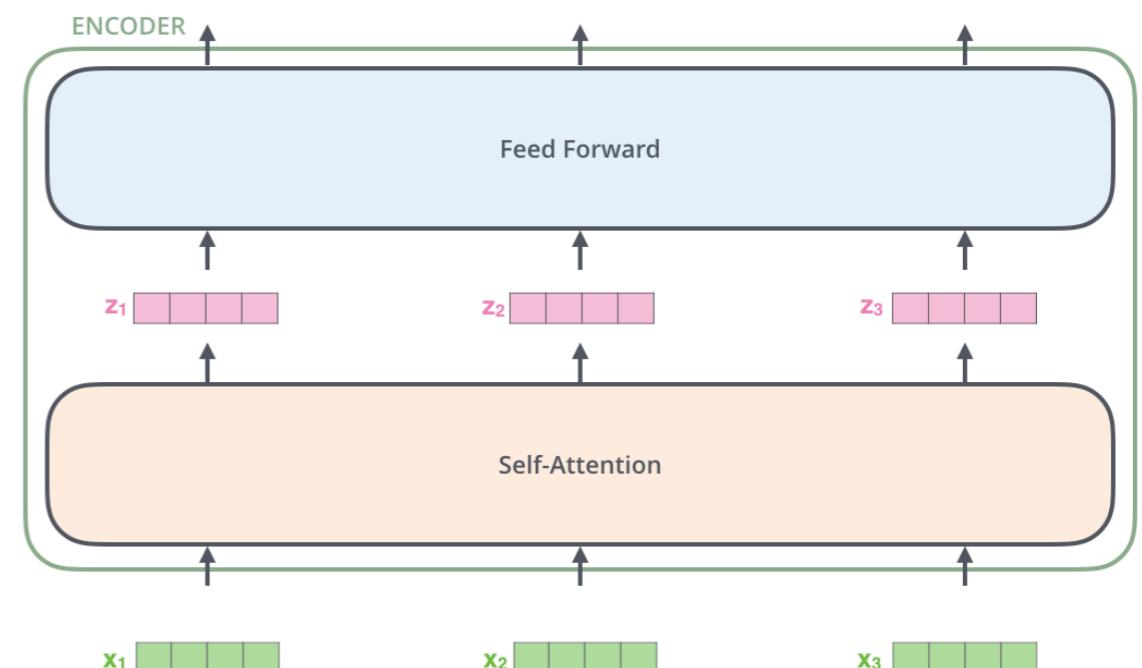


Image Credit: <https://arxiv.org/pdf/1706.03762.pdf>
Content Credit: [TransformerXL Explained & Al-Rfou et al. 2018](#)

Transformers for Language Modelling

- Transformers LM
 - Unidirectional
 - Segment of tokens
- Language Models predict the next word

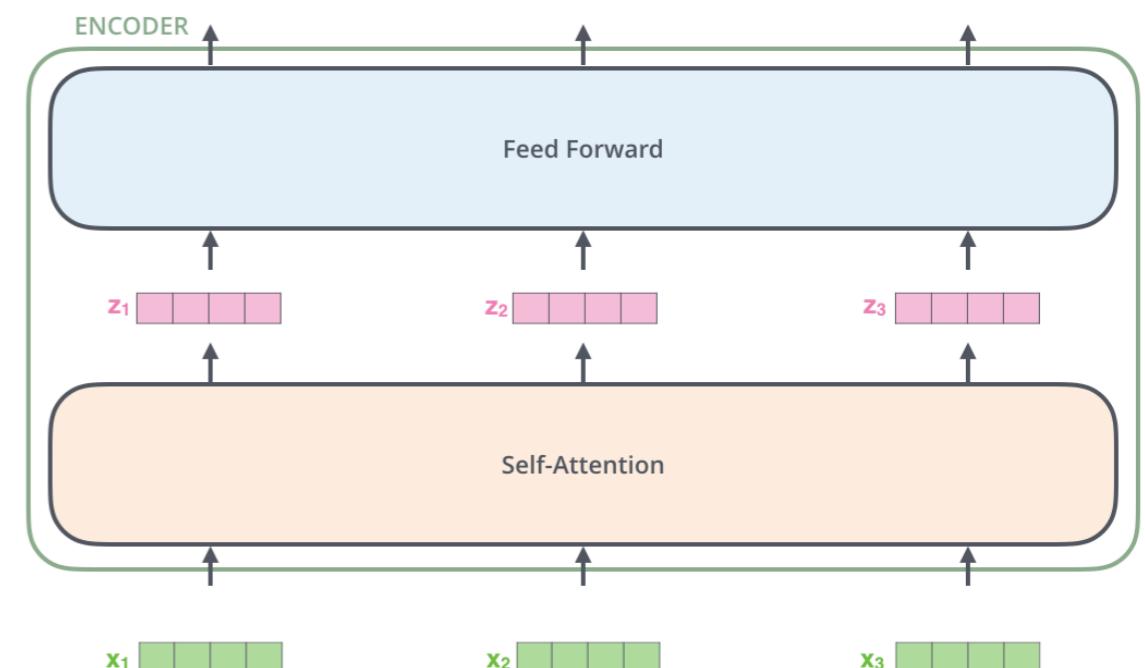


Image Credit: <https://arxiv.org/pdf/1706.03762.pdf>
Content Credit: [TransformerXL Explained & Al-Rfou et al. 2018](#)

Encoder Representations

- Require only the representations
- Forego of the output layer and only keep the encoder

Traditionally,

- Language Models predict the next word

Traditionally,

- Language Models predict the next word
- Or loosely, they “fill in the blank” based on the context

Traditionally,

- Language Models predict the next word
- Or loosely, they “fill in the blank” based on the context
- The man went to the store and bought a _____ of shoes

Traditionally,

- Language Models predict the next word
- Or loosely, they “fill in the blank” based on the context
- The man went to the store and bought a _____ of shoes
- Language models are mostly used as unidirectional tools

Traditionally,

- Language Models predict the next word
- Or loosely, they “fill in the blank” based on the context
- The man went to the store and bought a _____ of shoes
- Language models are mostly used as unidirectional tools
- Bidirectionality in this above example can help make a better judgement

Traditionally,

- Language Models predict the next word
- Or loosely, they “fill in the blank” based on the context
- The man went to the store and bought a _____ of shoes
- Language models are mostly used as unidirectional tools
- Bidirectionality in this above example can help make a better judgement
- In BERT, this bidirectionality is important to obtain good general purpose representations

BERT: Learning Setup

BERT: Learning Setup

- Pretraining

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences
 - Self-supervision

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences
 - Self-supervision
 - Masked LM

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences
 - Self-supervision
 - Masked LM
 - Next Sentence Prediction

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences
 - Self-supervision
 - Masked LM
 - Next Sentence Prediction
- Finetune

BERT: Learning Setup

- Pretraining
 - Takes lots and lots of sentences
 - Self-supervision
 - Masked LM
 - Next Sentence Prediction
- Finetune
 - Supervised using target task

Masked LM

Input



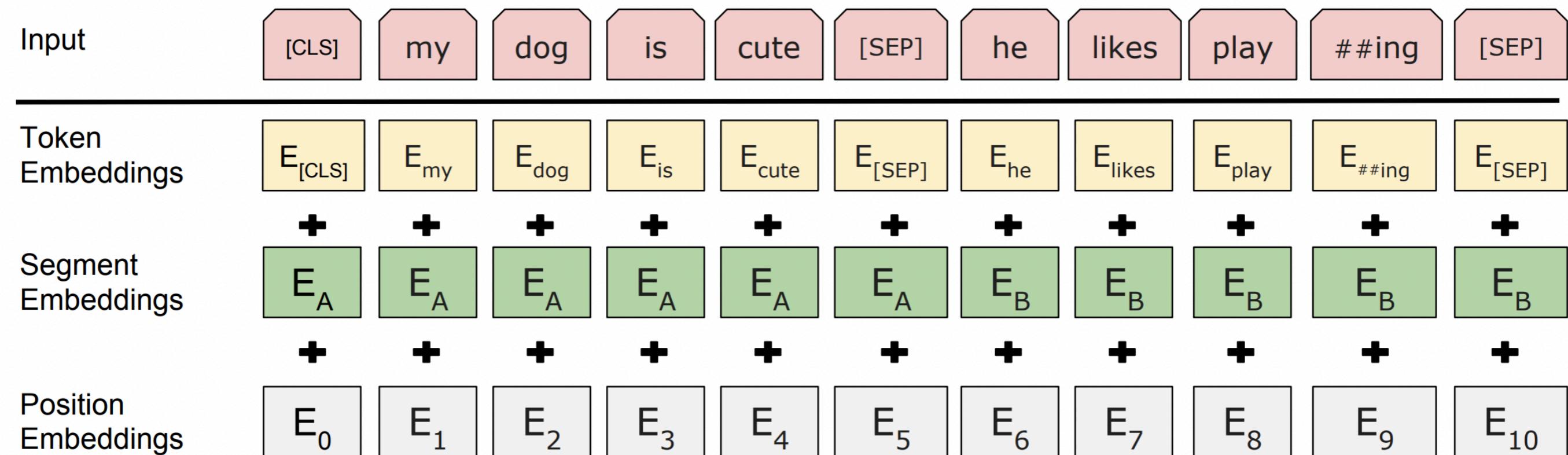
Masked LM

Input



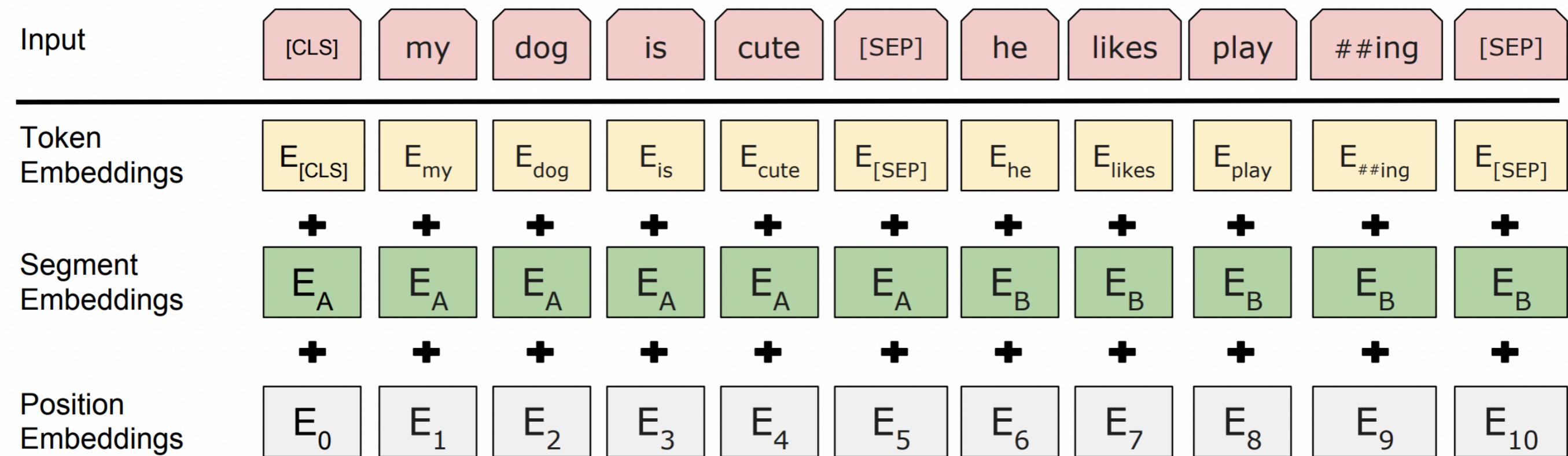
- Use specialised tokens CLS, SEP

Masked LM



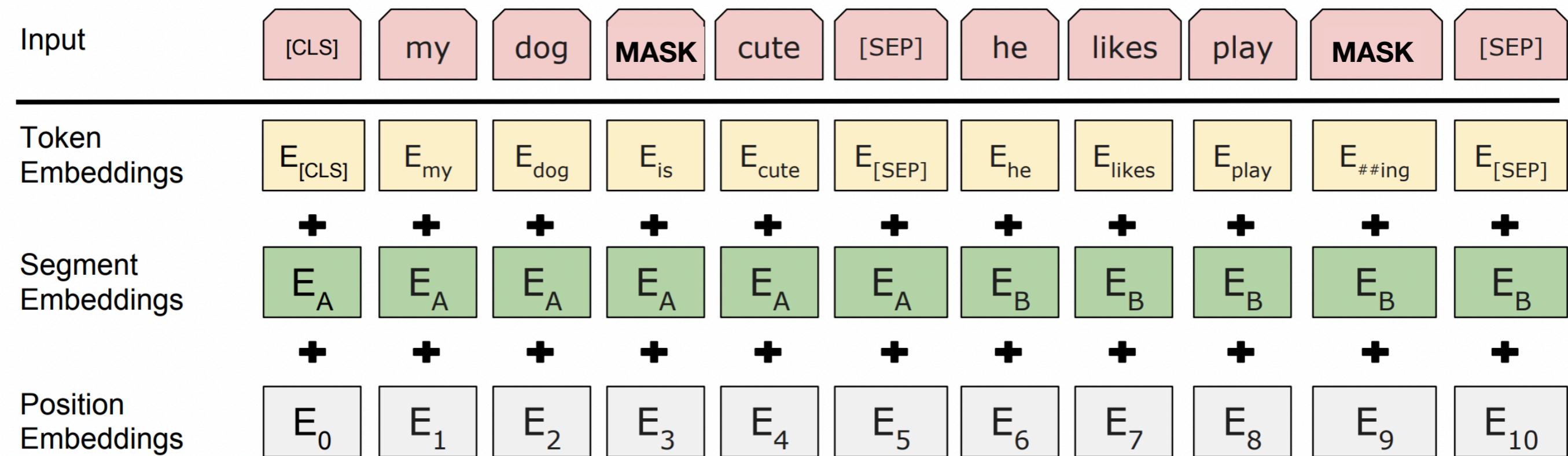
- Use specialised tokens CLS, SEP

Masked LM



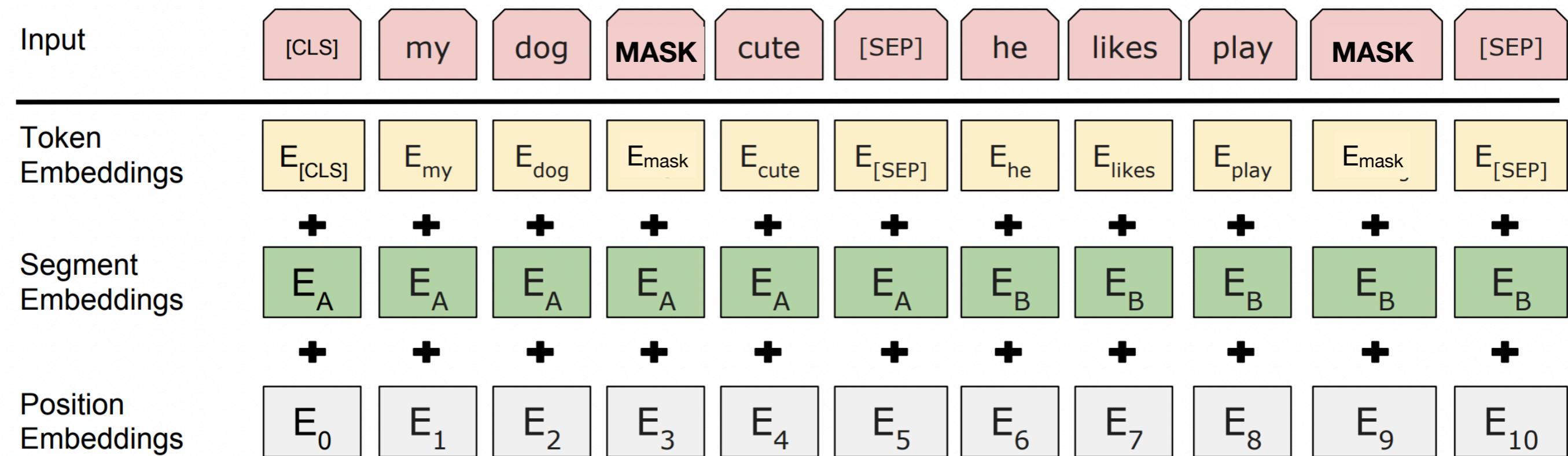
- Use specialised tokens CLS, SEP
- 15% of the tokens are randomly masked

Masked LM



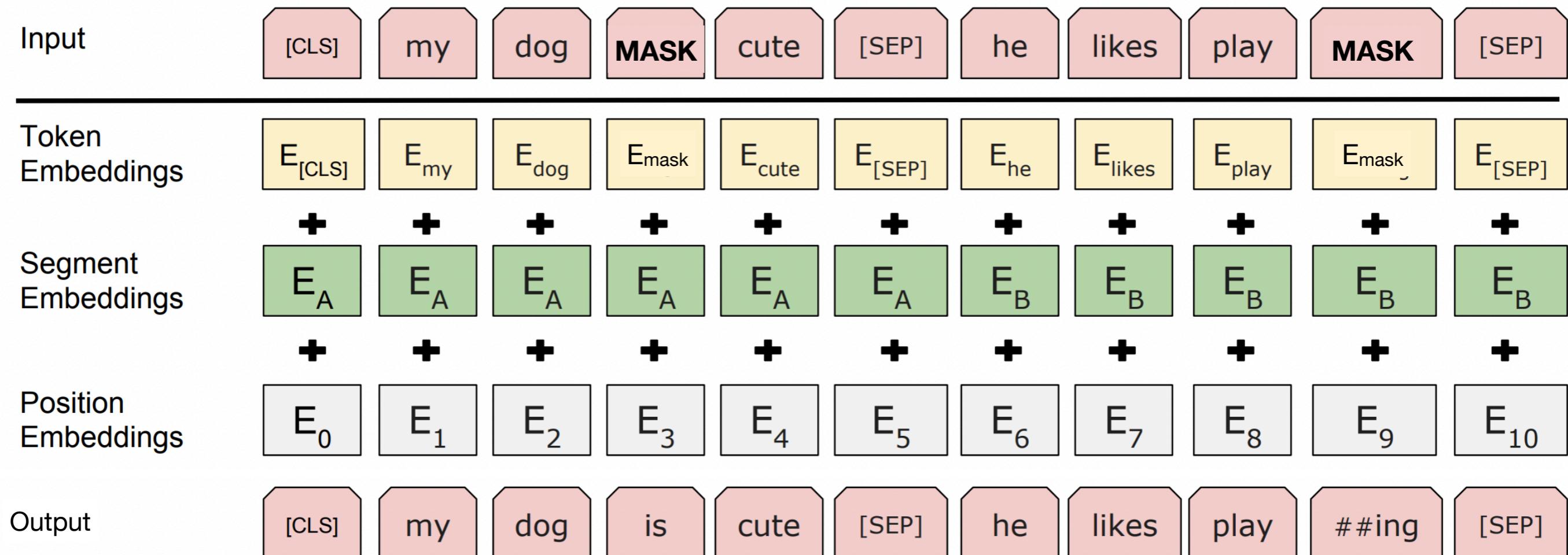
- Use specialised tokens CLS, SEP
- 15% of the tokens are randomly masked

Masked LM



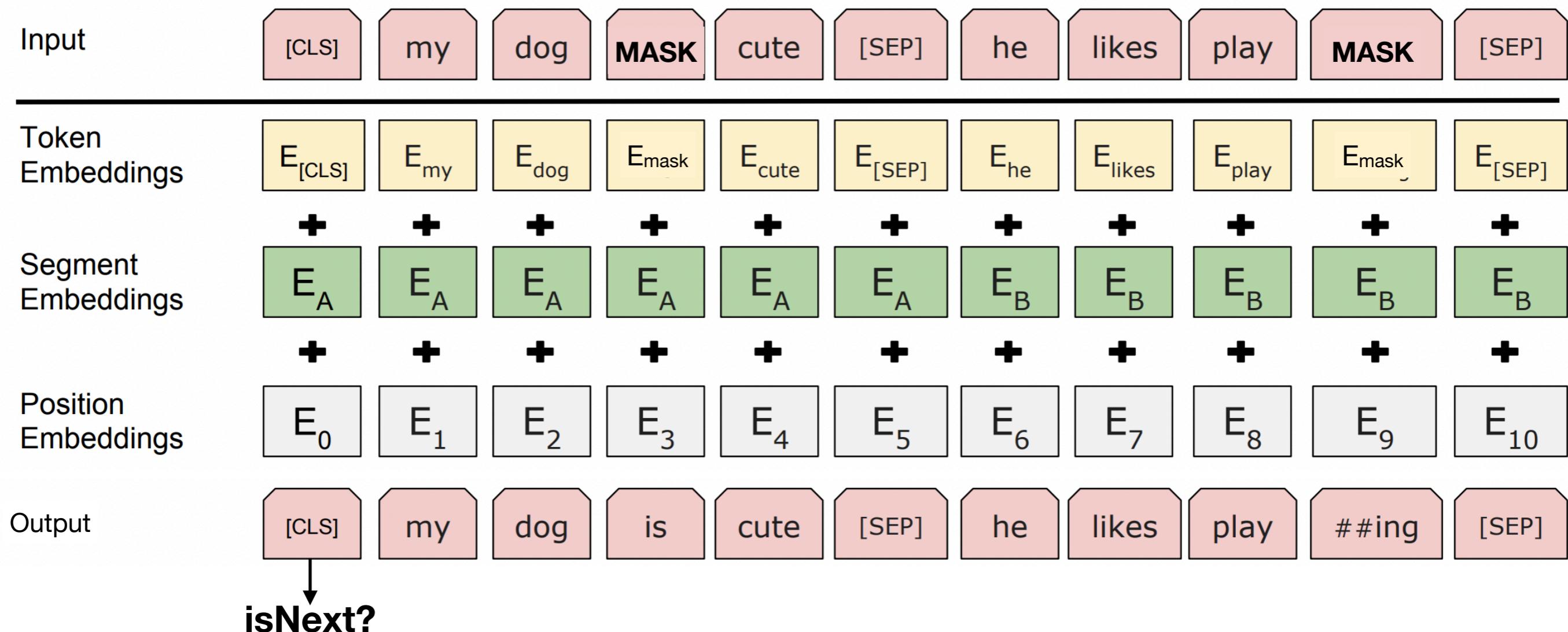
- Use specialised tokens CLS, SEP
- 15% of the tokens are randomly masked

Masked LM



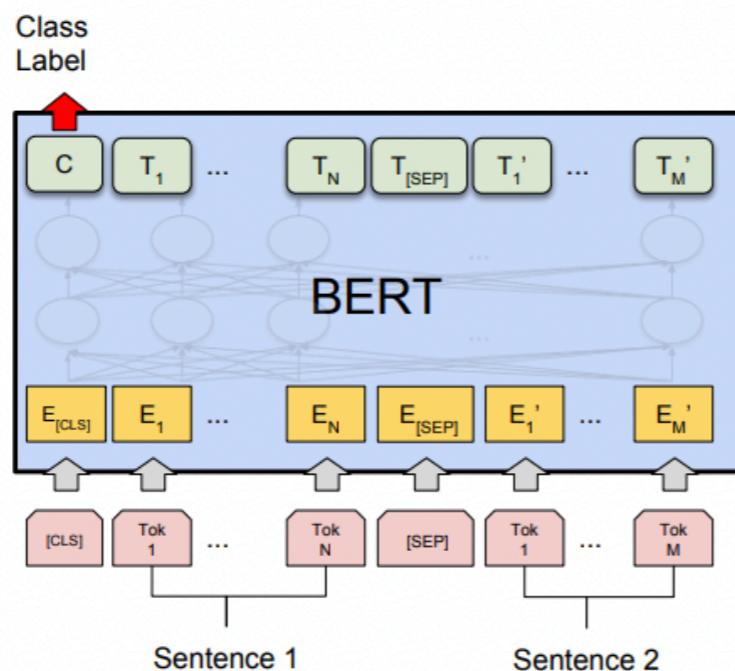
- Use specialised tokens CLS, SEP
- 15% of the tokens are randomly masked

Next Sentence Prediction

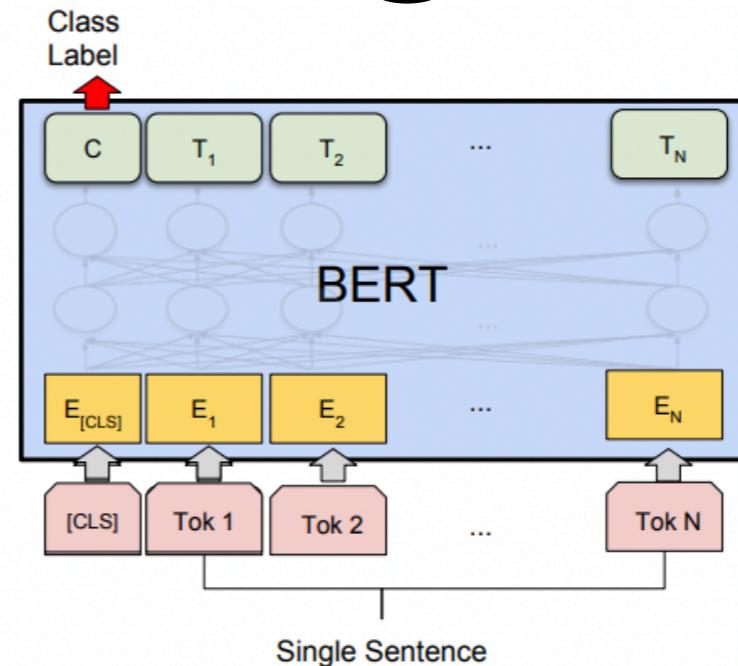


Use the CLS output embedding to predict is sentence B is the next sentence or not.

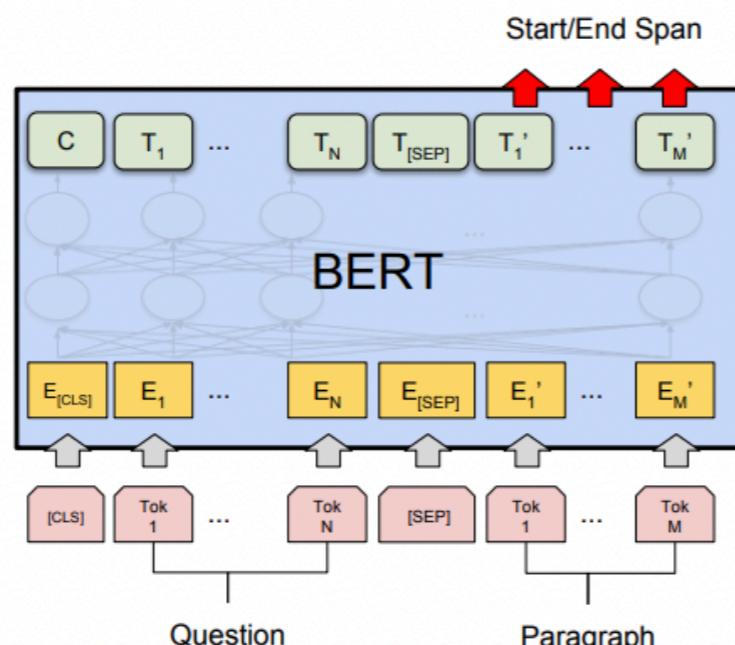
Fine-tuning



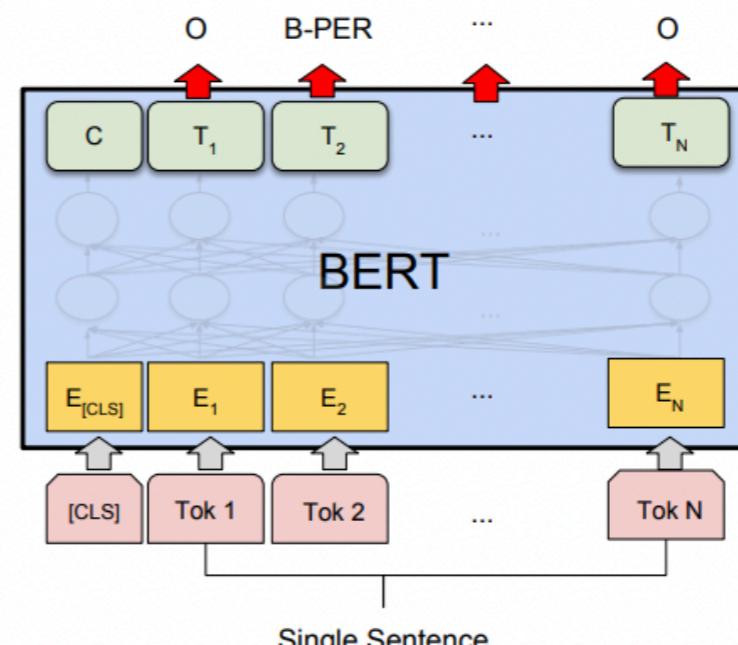
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Glue Test Results

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

[Jacob Devlin et al 2018]

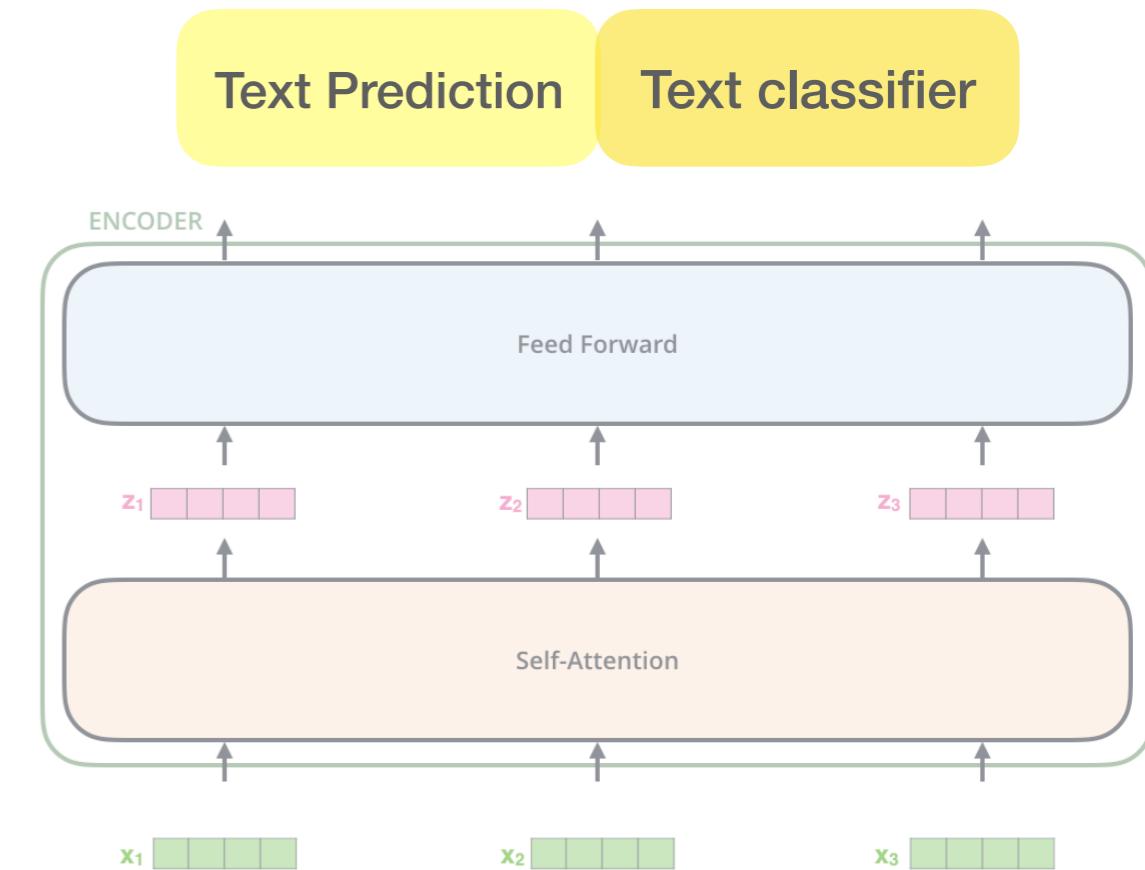
ChatGPT



CC: Adapted from slides by Dr. Stig-Arne Grönroos on ChatGPT

Generative Pretrained Transformer (GPT)

- Autoregressive
 - GPT: 12 layers, 117M
 - GPT-2: 48 layers, 1.5B
 - GPT-3: 96 layers, 175B
 - InstructGPT: GPT-3 + human feedback for a certain task
 - ChatGPT: GPT-3 + human feedback for being a good chatbot
- Text + Pos**
- 12x {



Loss drops smoothly with size

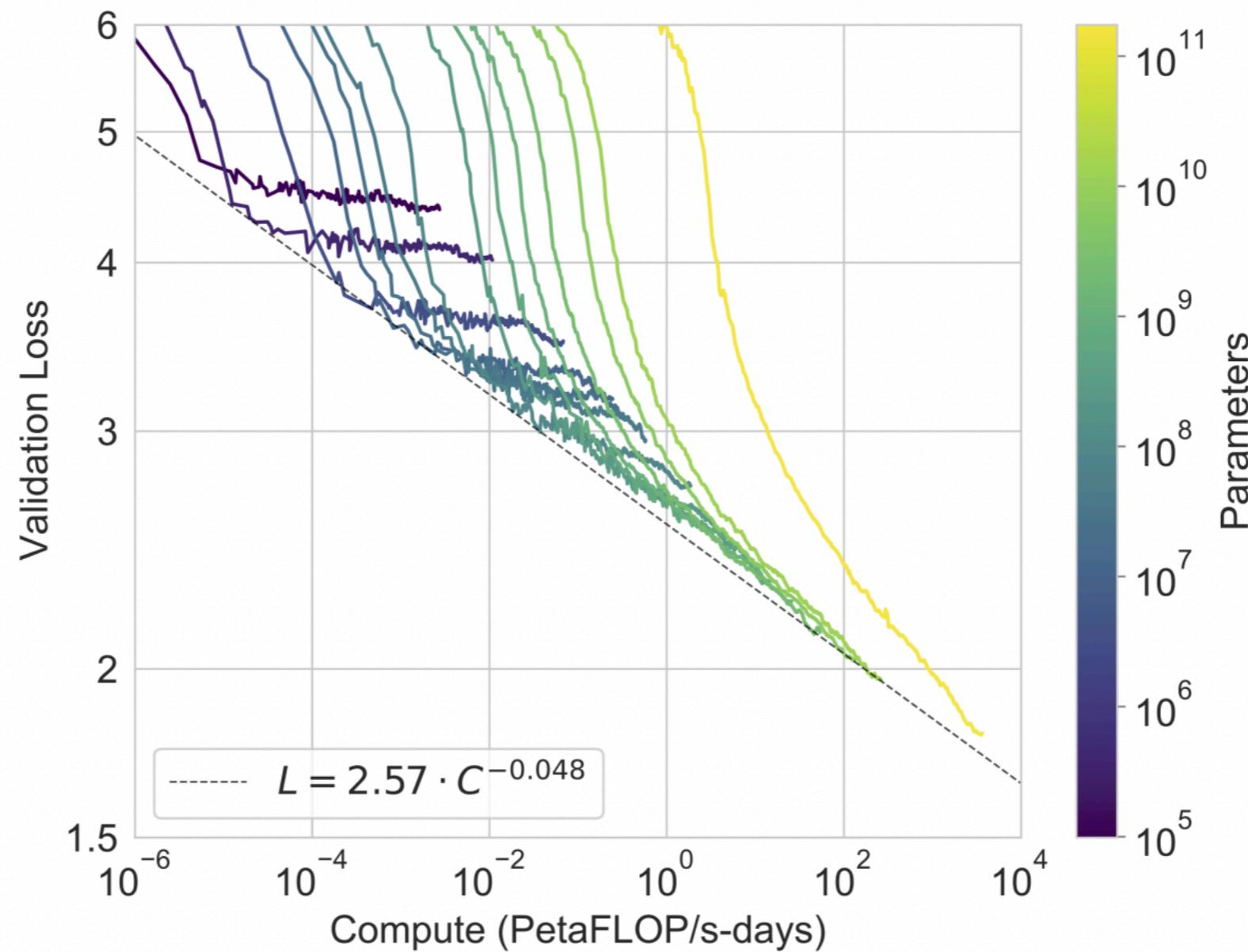


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH⁺20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

InstructGPT Training

Step 1

Collect demonstration data and train a supervised policy.

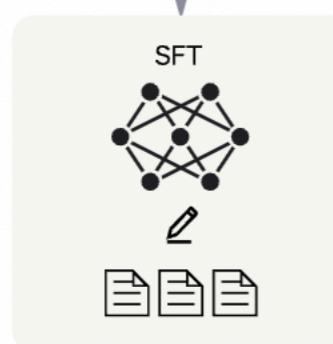
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

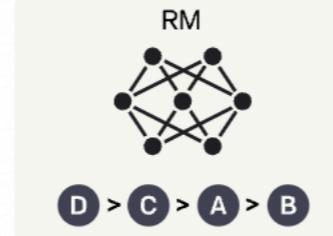
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



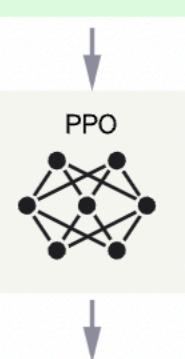
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

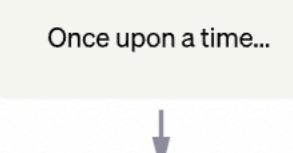
A new prompt is sampled from the dataset.



The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Performance

Performance

- The good

Performance

- The good
- Style transfer: write X in the style of Y

Performance

- The good
- Style transfer: write X in the style of Y
- Improved coherence: 2048 tokens as window size vs 512 for BERT

Performance

- The good
- Style transfer: write X in the style of Y
- Improved coherence: 2048 tokens as window size vs 512 for BERT
- Reduced toxicity

Performance

- The good
- Style transfer: write X in the style of Y
- Improved coherence: 2048 tokens as window size vs 512 for BERT
- Reduced toxicity
- Generate synthetic data

Performance

- The good
- Style transfer: write X in the style of Y
- Improved coherence: 2048 tokens as window size vs 512 for BERT
- Reduced toxicity
- Generate synthetic data
- The bad
- Logical and spatial reasoning
- Puns, word-play jokes
- The ugly
- Confidently states incorrect information

Spatial reasoning

ST

I will tell you a joke. Could you help me understand why it is funny? Here is the joke: "Two men walked on a bridge. The middle one wore a hat."



The joke is funny because it is a play on words. The word "middle" can refer to both the physical position of the man on the bridge, as well as his mental state. The joke suggests that the man wearing the hat is not quite right in the head, or that he is different in some way. The unexpected punchline, combined with the clever wordplay, makes the joke amusing.

Future of search?

 what new discoveries from the James Webb Space Telescope can I tell my 9 year old about?

Your 9-year old might like these recent discoveries made by The James Webb Space Telescope (JWST):

- In 2023, The JWST spotted a number of galaxies nicknamed “green peas.” They were given this name because they are small, round, and green, like peas.
- The telescope captured images of galaxies that are over 13 billion years old. This means that the light from these galaxies has been traveling for over 13 billion years to reach us.
- JWST took the very first pictures of a planet outside of our own solar system. These distant worlds are called “exoplanets.” Exo means “from outside.”

These discoveries can spark a child's imagination about the infinite wonders of the universe.

 ...

Summary

- NNLM:
 - RNNs
 - Transformers
 - Self Attention
 - BERT
 - ChatGPT
- Challenges
 - Long-Term Dependencies
 - Class-based output layer
 - Rare Words

Further Reading

- Neural Networks and Neural Language Models: <https://web.stanford.edu/~jurafsky/slp3/7.pdf>
- BERT Explained <https://medium.com/@samia.khalid/bert-explained-a-complete-guide-with-theory-and-tutorial-3ac9ebc8fa7c>
- GPT-3 paper: <https://arxiv.org/abs/2005.14165>
- ChatGPT: <https://gist.github.com/veekaybee/6f8885e9906aa9c5408ebe5c7e870698>