

A horizontal bar with a yellow segment on the left and a red segment on the right.

# Neural Language Model

NLP II 2025

Assoc. Prof. Attapol Thamrongrattanarit

# ChatGPT: Optimizing Language Models for Dialogue

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests. ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

[TRY CHATGPT ↗](#)




# Problems

if this is zero, then probability is zero.  
we must smooth the count

$$P(\text{right} \mid \text{officials deny the}) = \frac{\text{count}(\text{officials deny the right})}{\text{count}(\text{officials deny the})}$$

we have to store  
millions of counts

if this zero, then divide by 0 = Infinity!  
we must backoff to the lower order  
model such as 4gram → trigram

A horizontal bar with a gold segment on the left and a red segment on the right.

## Review: Drawback of n-gram language model

- N-gram models struggle with choosing how much context we should consider
  - Too much context and the model performs poorly because we do not have enough data. (sparsity problem)
  - Too little context and the model performs poorly because of long-range dependencies in language.

A horizontal bar with a gold segment on the left and a red segment on the right.

## History of neural language model

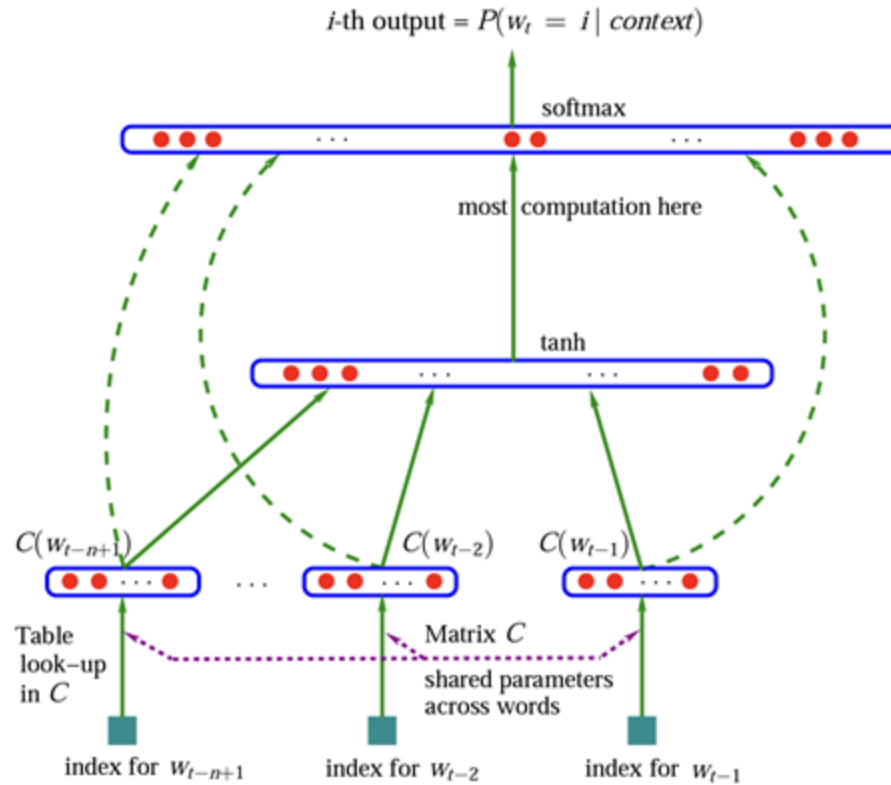
- 2003 - feedforward neural language model
- 2016 - recurrent neural language model (RNN-LM)
- 2017 - transformer-based language model
- 2019 - large transformer-based language model (e.g. GPT-2)
- 2022 - massive transform-based language model fine-tuned for use as a chatbot (ChatGPT)



## Feedforward Neural Language Model

Assign each word an embedding and concatenate the embeddings to create an  $n$ -gram representation.

Add a hidden layer and connect it to a classifier to predict the next word.  $V$ -way classification where  $V$  is the vocabulary size.



Feedforward neural net language model

# The results are very promising

- What are the number of parameters of n-gram model vs n-gram neural LM?
- How does neural LM solve the sparsity problem?
- Does it solve long-range dependency problems?

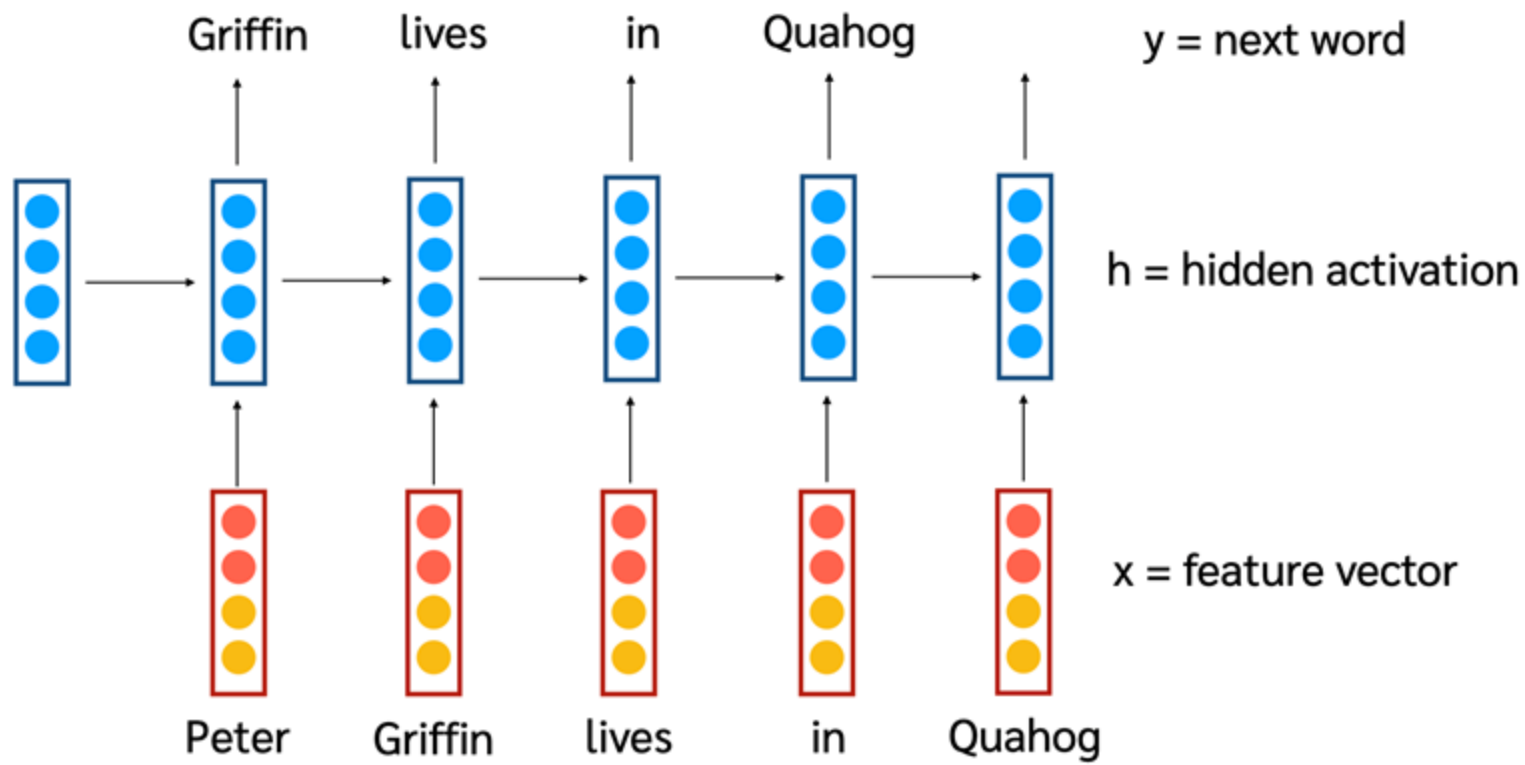
	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	<b>252</b>
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	<b>312</b>
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

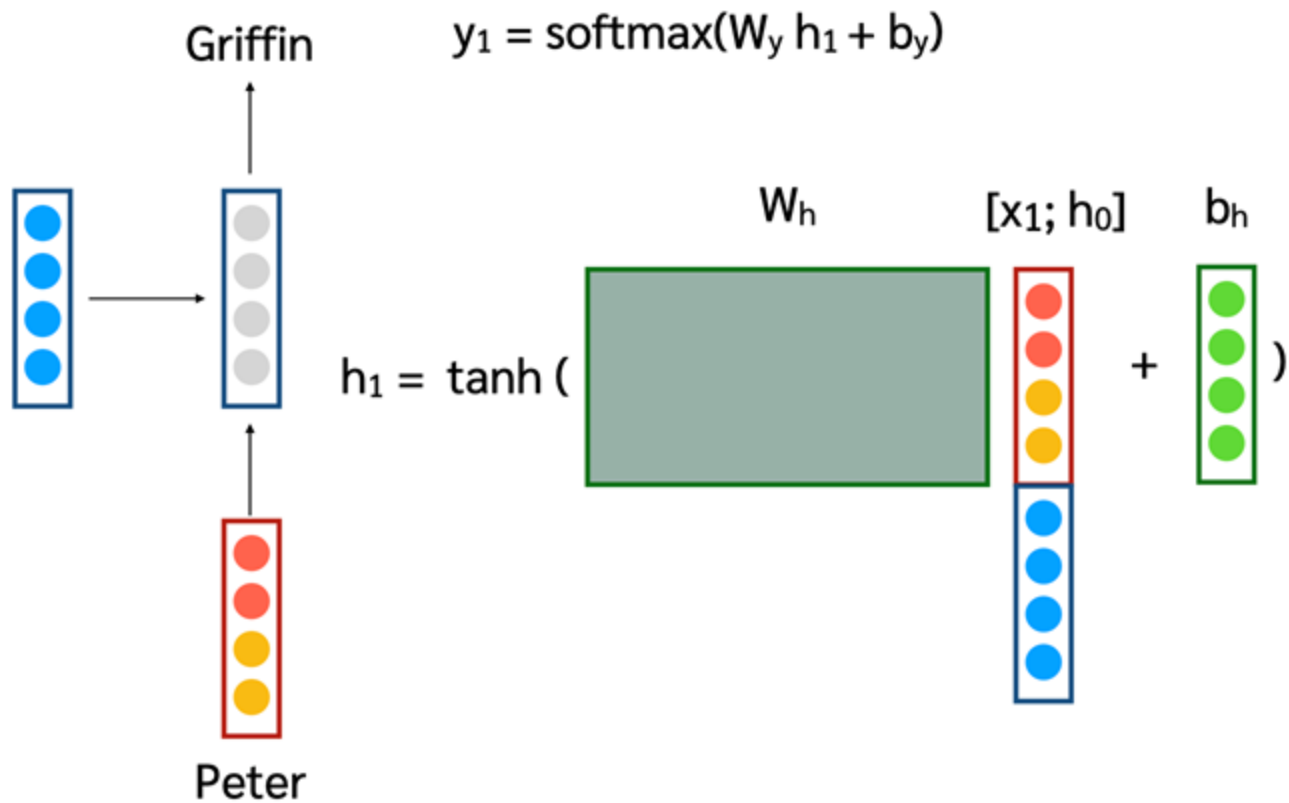


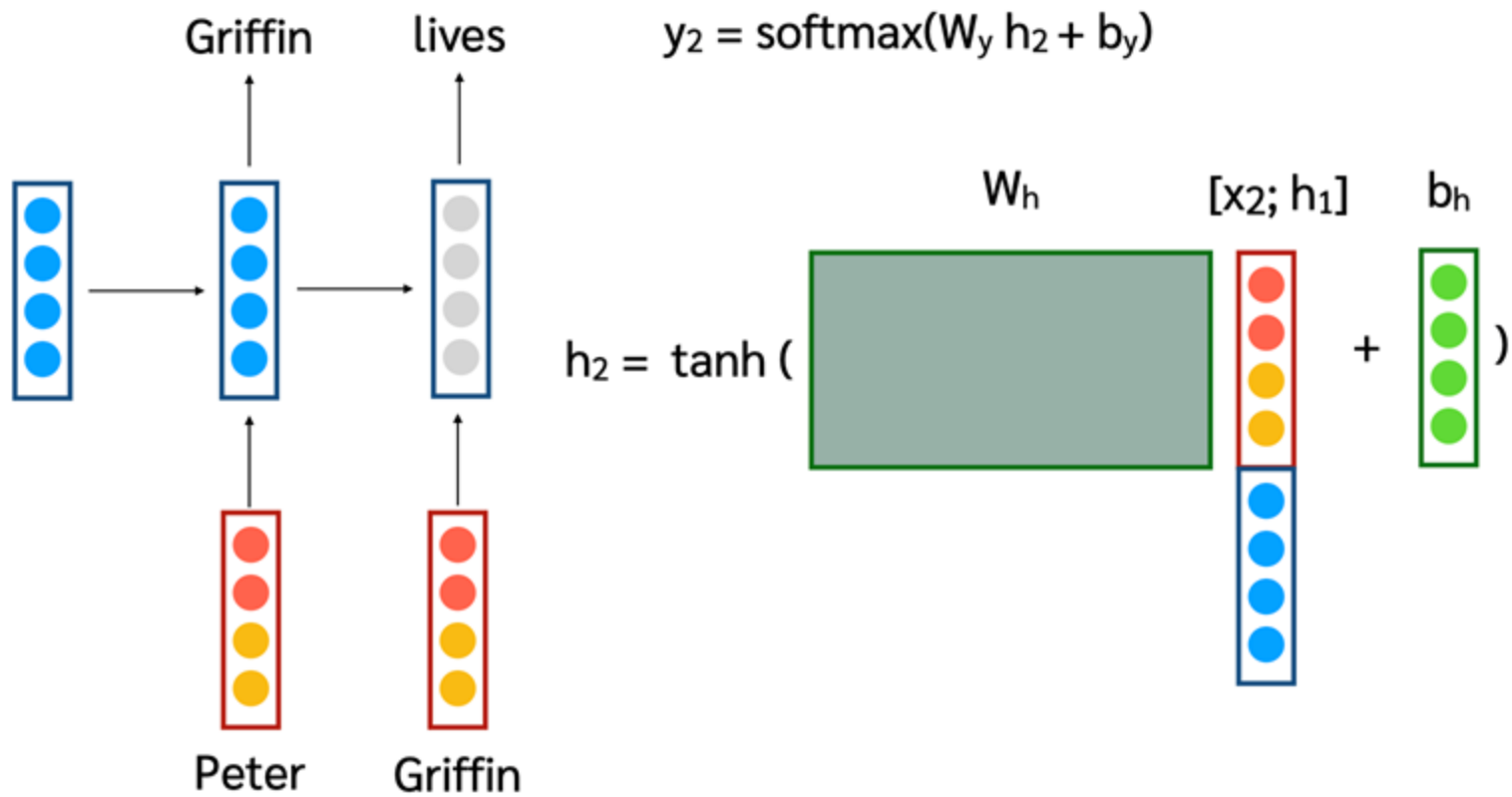
A horizontal bar with a gold segment on the left and a red segment on the right.

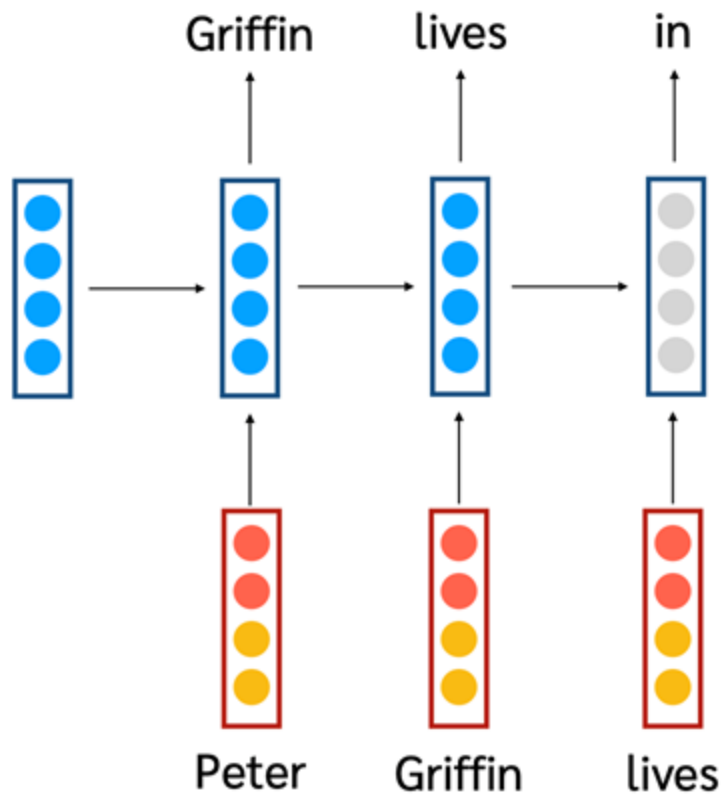
## History of neural language model

- 2003 - feedforward neural language model
- 2016 - recurrent neural language model (RNN-LM)
- 2017 - transformer-based language model
- 2019 - large transformer-based language model (e.g. GPT-2)
- 2022 - massive transformer-based language model fine-tuned for use as a chatbot (ChatGPT)



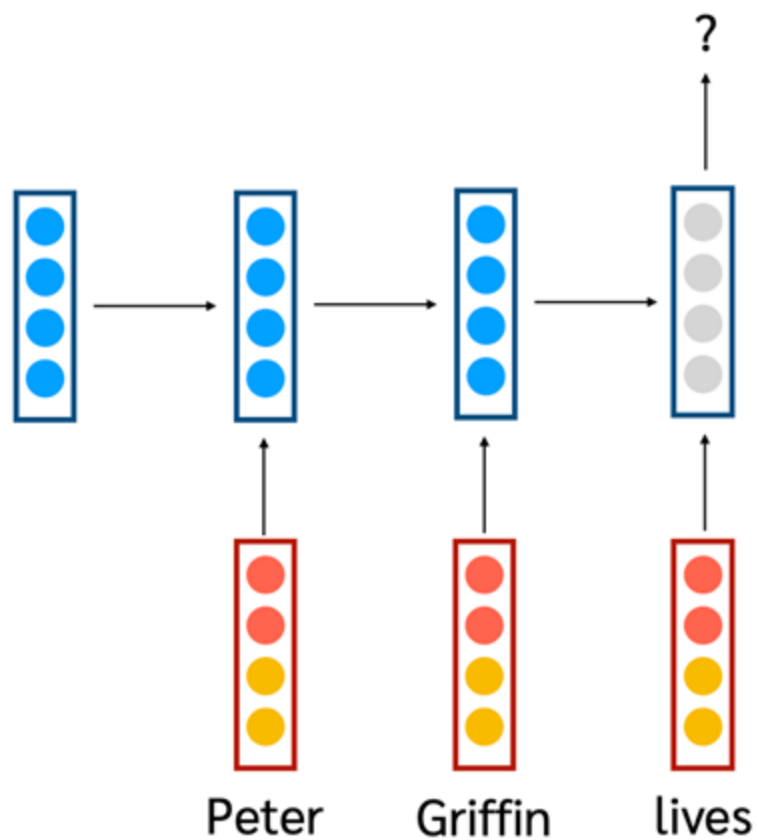






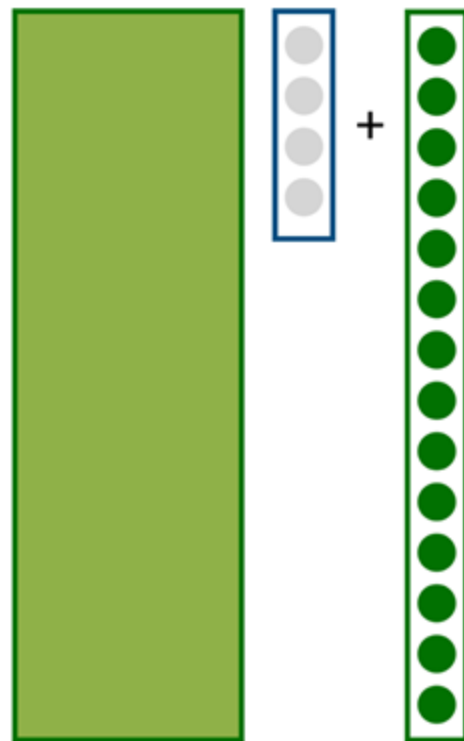
$$y_3 = \text{softmax}(W_y h_3 + b_y)$$

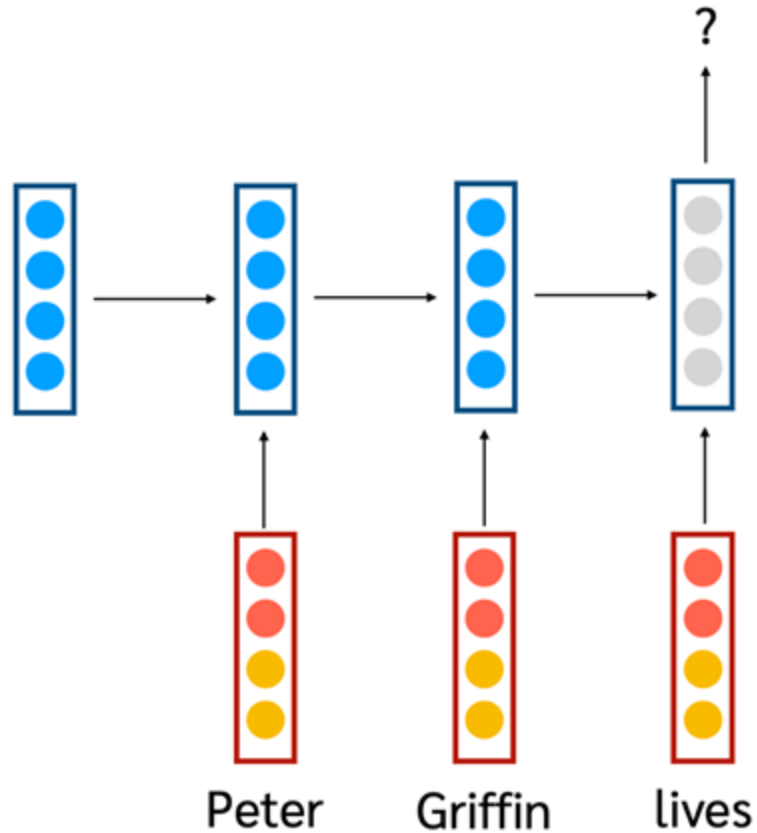
$$h_3 = \tanh \left( W_h [x_3; h_2] + b_h \right)$$



คำต่อไปคือคำว่าอะไร

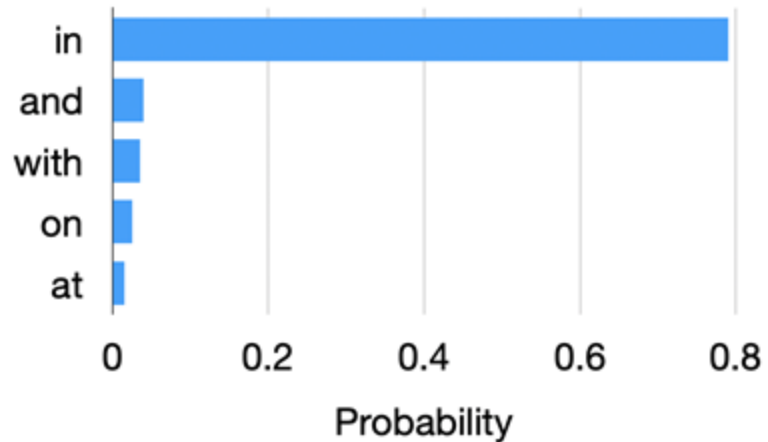
$$y_t = \text{softmax}(W_y \cdot h_t + b_y)$$

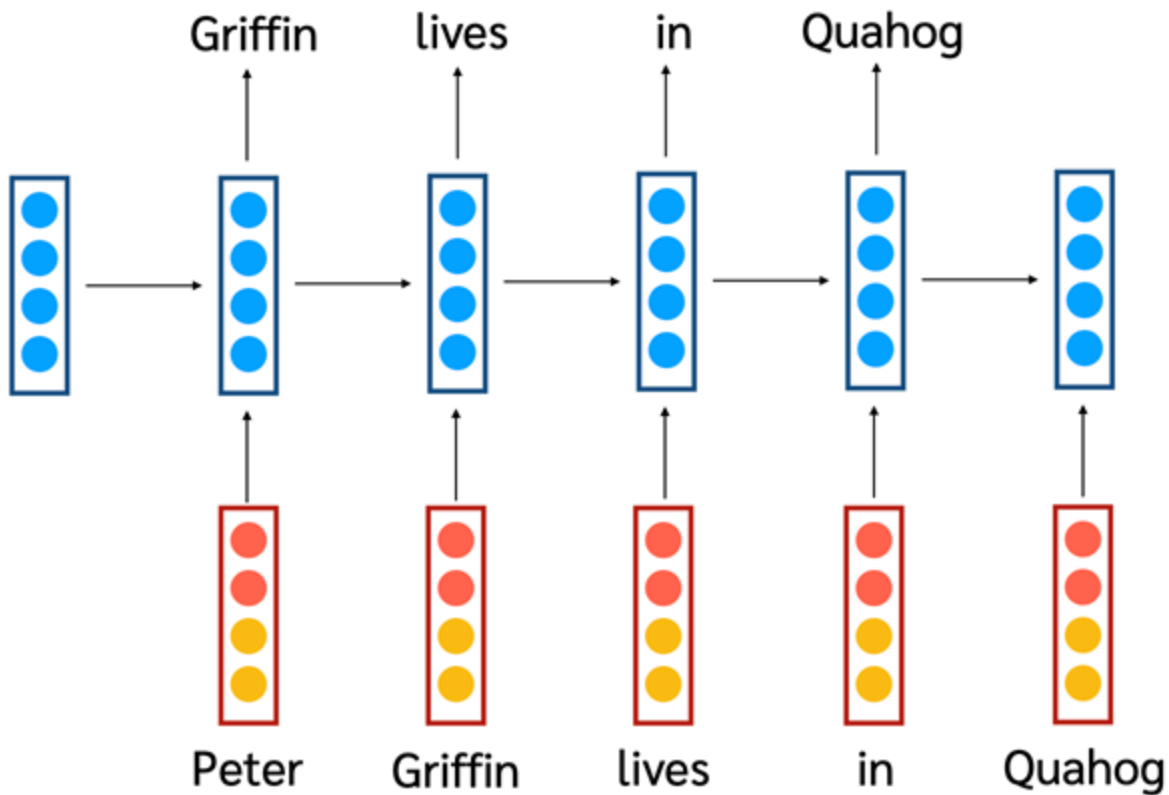




คำต่อไปคือคำว่าอะไร

$$y_t = \text{softmax}(W_y \cdot h_t + b_y)$$



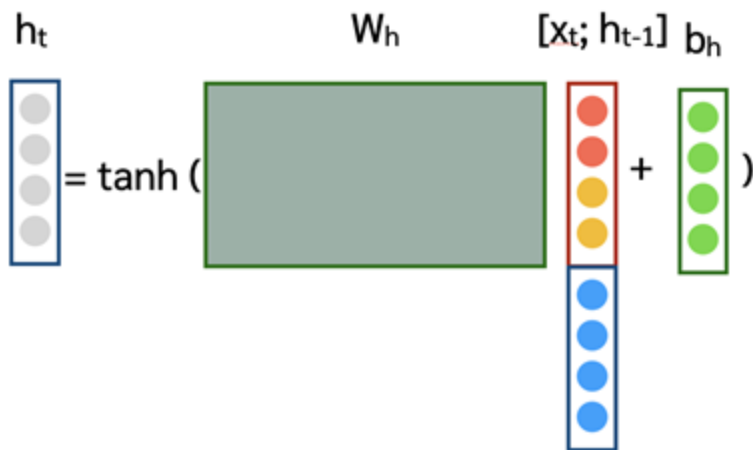


Teacher Forcing  
Use the actual gold standard data to train the next word prediction model

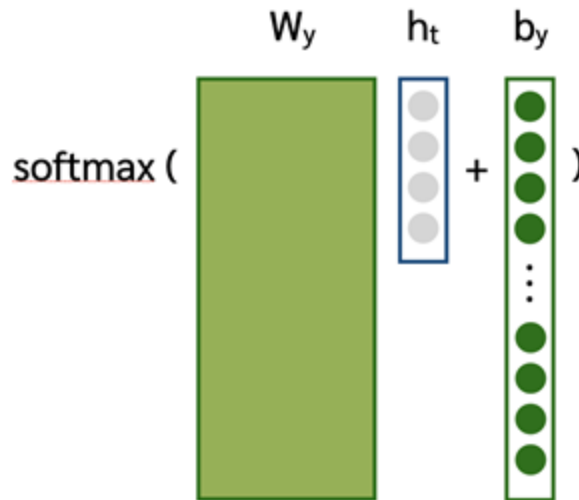


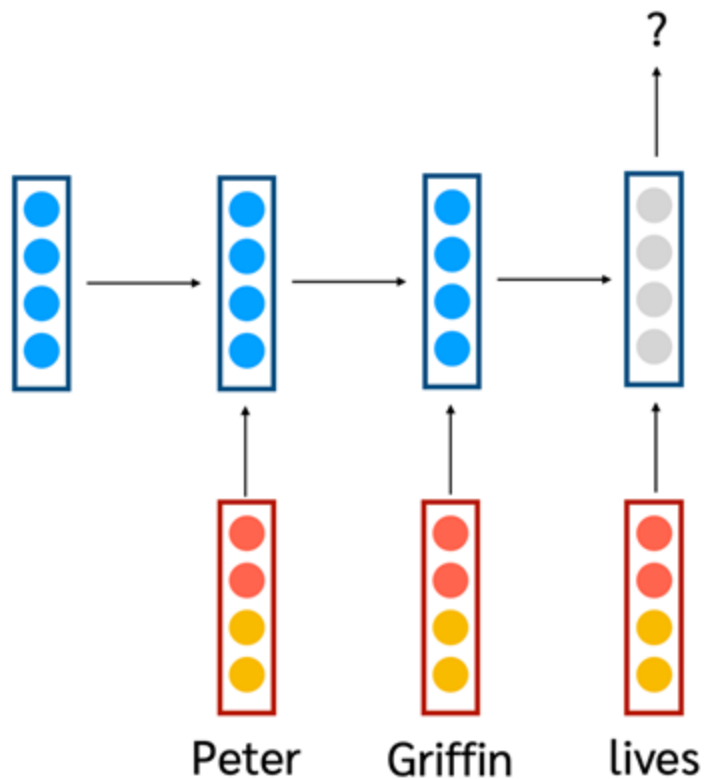
# Recurrent Neural Network LM parameters

$$h_t = \tanh(W_h \cdot [x_t; h_{t-1}]) + b_h$$

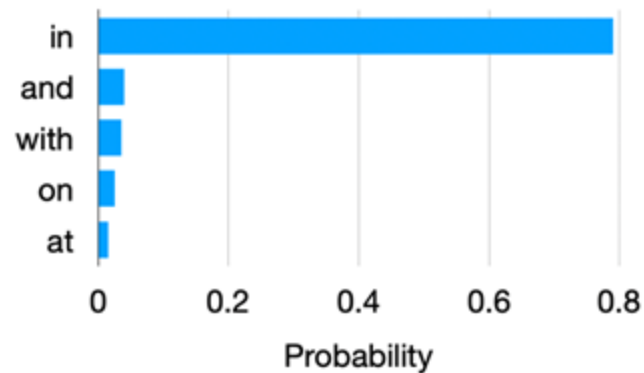


$$y_t = \text{softmax}(W_y \cdot h_t + b_y)$$

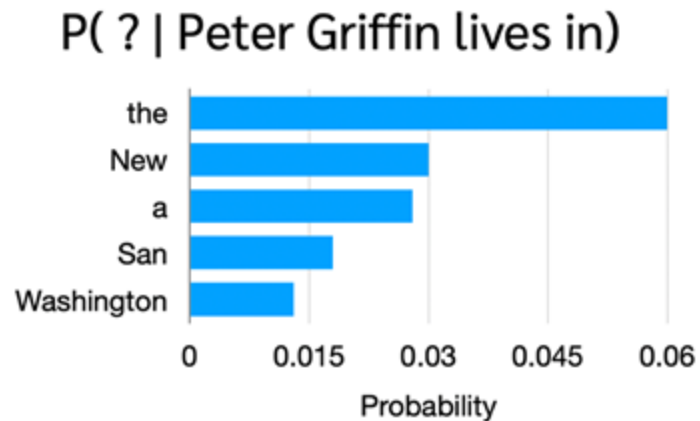
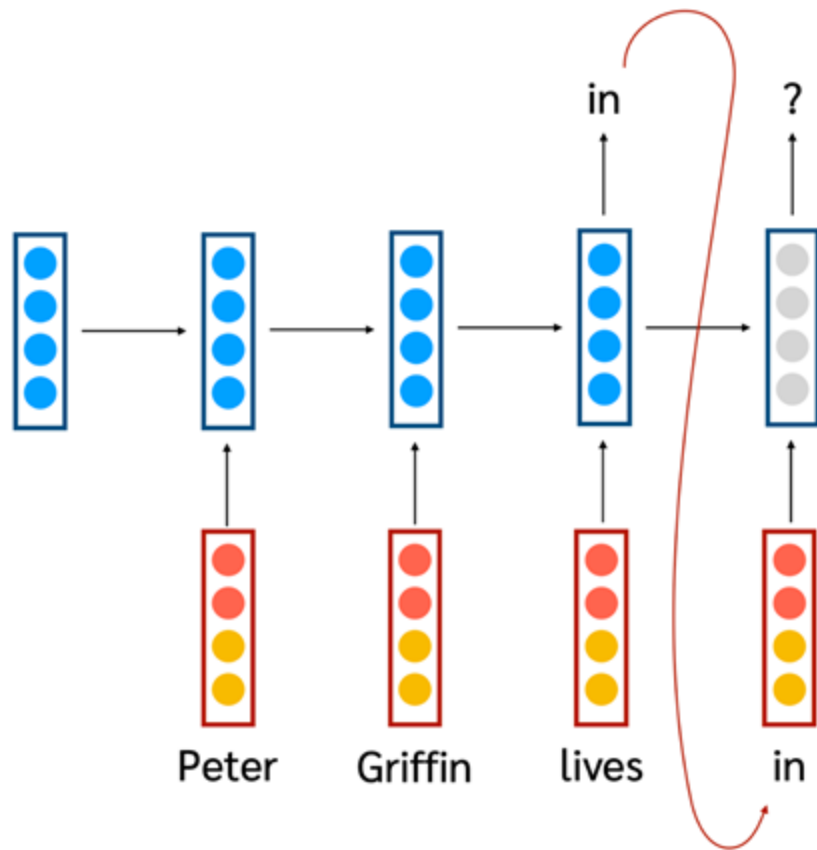




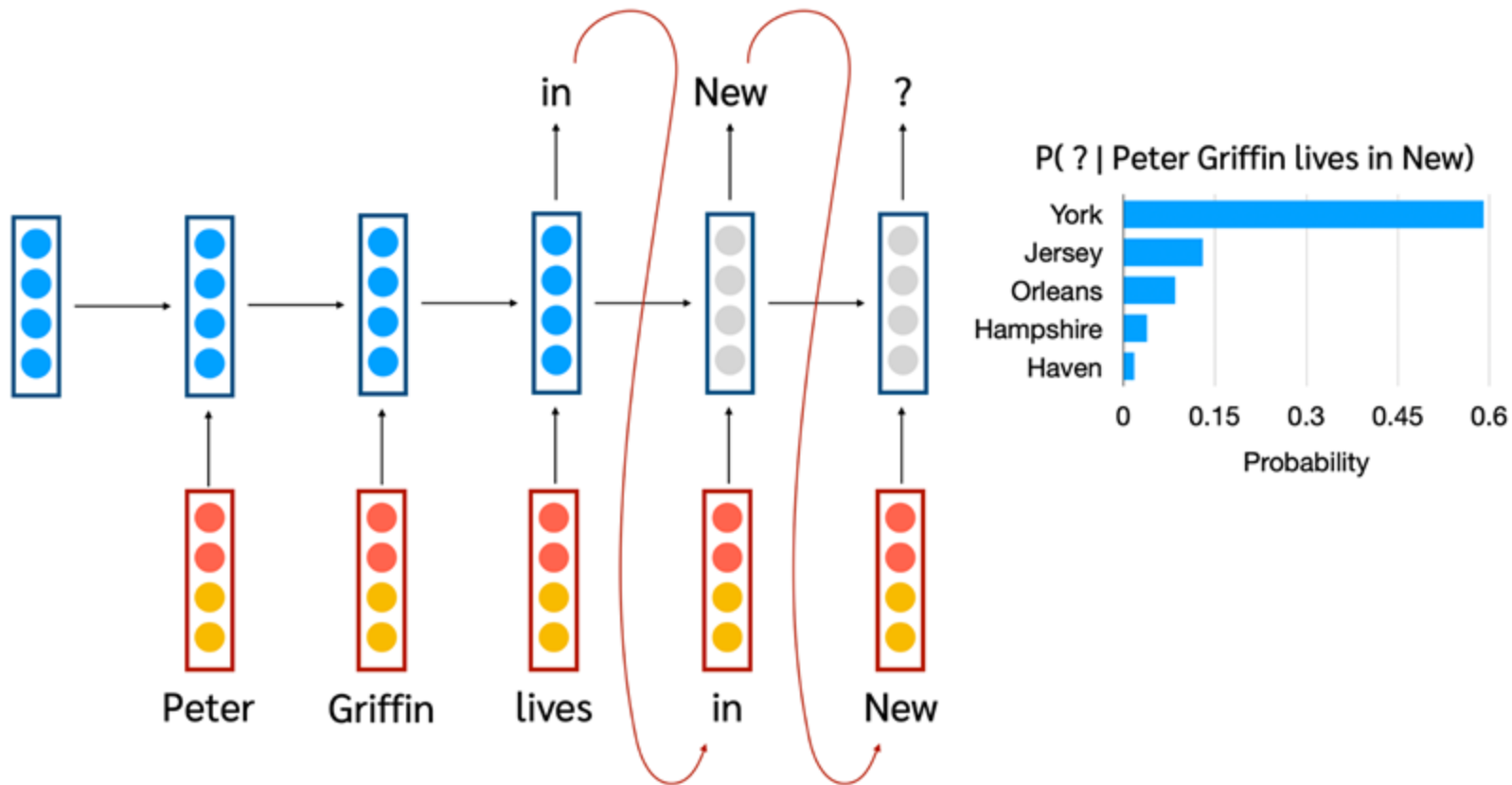
$P( ? \mid \text{Peter Griffin lives})$



Generate from the prediction of the next word



Use the output as the next input



And the next...



## Autoregressive model

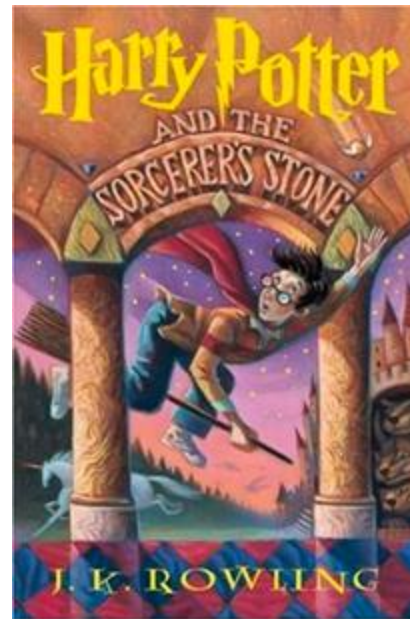
- Generate Text ได้โดยการสุ่มจาก Probability ที่โมเดลคำนวณมาให้
- คำต่อไปถูกทำนายจากบริบทที่อยู่ทางด้านซ้ายมือ
- คำที่ทำนายกลายเป็น input สำหรับ step ถัดไป

## Generating text from RNN-LM

*“The Malfoys!” said Hermione.*

*Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.*

*“I’m afraid I’ve definitely been suspended from power, no chance — indeed?” said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.*



## Generating text from RNN-LM

PANDARUS:

*Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.*

Second Senator:

*They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.*



## Generating text from RNN-LM

*RICHARD: Dammit. Dammit. Dammit. Routine of their room!*

*JARED: We're planning for current grid contacts, shortages, top values.*

*MONICA: Okay. Shoulda warned here!*

*RICHARD: Sorry.*

*ERLICH: How was a \$20,000 worth term reasonable, and then we vote to piss to everything. It was absolute. Time in safari with the board of us guys. She'd run a plumber?*







## Character RNN-LM vs word-level RNN-LM

- A character-level RNN-LM generates text one character at a time, allowing for more flexibility in generating new and creative combinations of characters. (Character-level n-gram LM stands no chance. Why?)
- A word-level RNN-LM generates text one word at a time and is better suited for generating text that is more structured and follows patterns present in the corpus.

## Evolution during training character RNN-LM

iteration	Sample text
100	tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrkd t o idoe ns,smtt h ne etie h,hregtrs niglike,aoaenns lng
300	"Tmont thithey" fomesscerliund Keushey. Thom here sheulke, anmerenith ol sivh l lalterthend Bleipile shuwly fil on aseterlome coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
500	we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt off pahtt thin wall. Which das stimn

## Evolution during training on Tolstoy (2)

iteration	Sample text
700	Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and offer.
1200	"Kite vouch!" he repeated by her door. "But I would be done and quarts, feeling, then, son is people...."
2000	"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre aking his soul came to the packs and drove up his father-in-law women.

## Stunning evaluation results

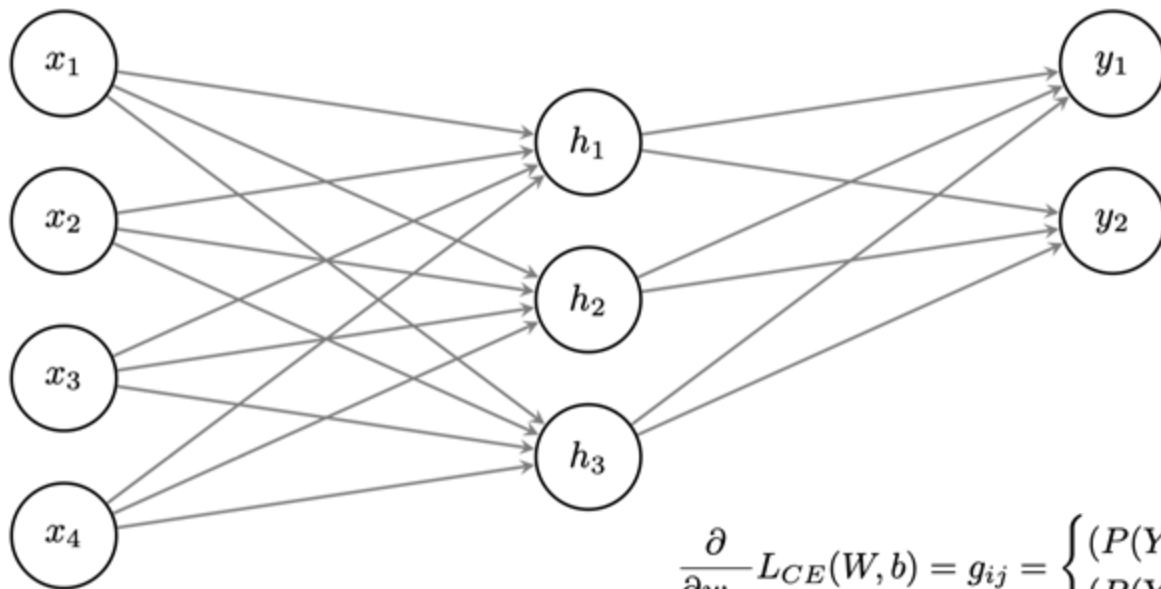
Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
<b>Ours small</b> (LSTM-2048)	43.9
<b>Ours large</b> (2-layer LSTM-2048)	39.8

A horizontal bar with a gold segment on the left and a red segment on the right.

## RNN Language model

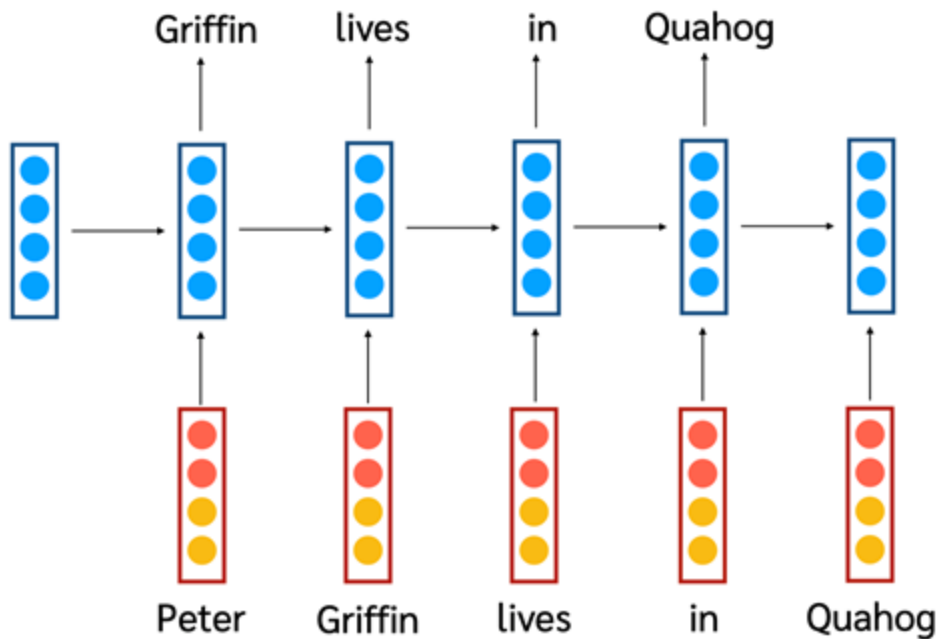
- No need to specify the window/context size (the  $n$  of  $n$ -gram)
- The model uses the context from many steps back (in theory)
- The context size does not affect the number of parameters

## Review: Backpropagation



$$\frac{\partial}{\partial w_{ij}} L_{CE}(W, b) = g_{ij} = \begin{cases} (P(Y = i|X, W, b) - 1)x_j & \text{true label} = i \\ (P(Y = i|X, W, b) - 0)x_j & \text{otherwise} \end{cases}$$

# Training RNN: Backpropagation Through Time



## Vanishing gradient

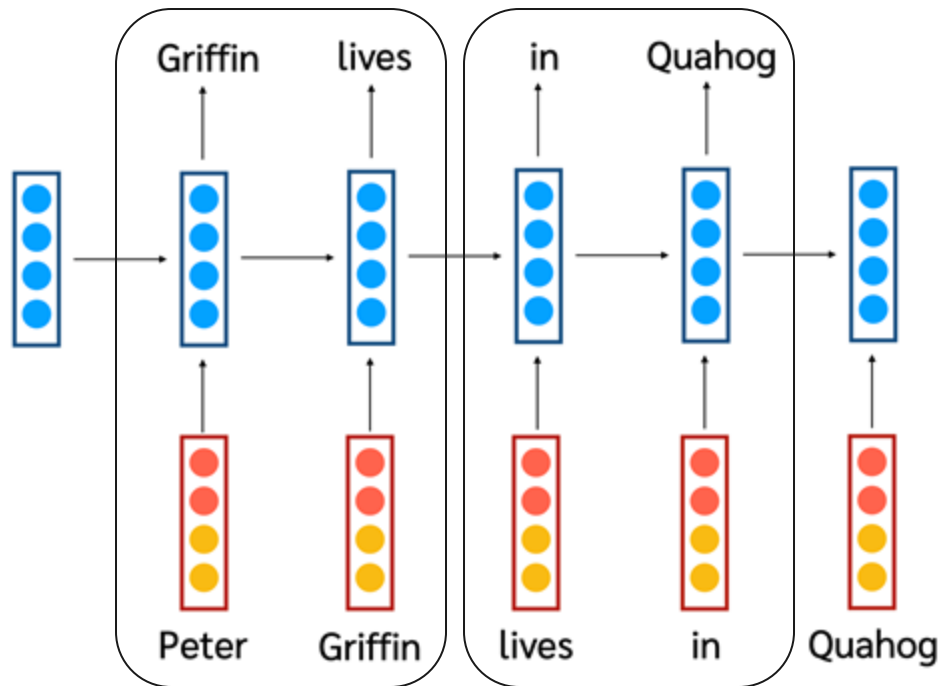
- $$\begin{aligned}
 h_5 &= W [h_4; x_5] + b \\
 &= W [W [h_3; x_4] + b; x_5] + b \\
 &= W [W [W [h_2; x_3] + b; x_4] + b; x_5] + b
 \end{aligned}$$

what if W is very small?
- The error that  $h_2$  will mostly comes from  $y_2$  and not  $y_5$  because it was scaled down by W many times.
- The error is much louder when near. Every time step in the network will listen to the error signal close by must more than the error signal from far away.



# Vanishing gradient

- Gradient (errors) gets scaled down exponentially over each step.
- Only short-range dependencies in the sentence are learned.





## Exploding gradient

- $$\begin{aligned}h_5 &= W [h_4; x_4] + b \\&= W [W [h_3; x_3] + b; x_4] + b \\&= W [W [W [h_2; x_2] + b; x_3] + b; x_4] + b\end{aligned}$$
- The update step becomes very large or infinity.

what if  $W$  is very

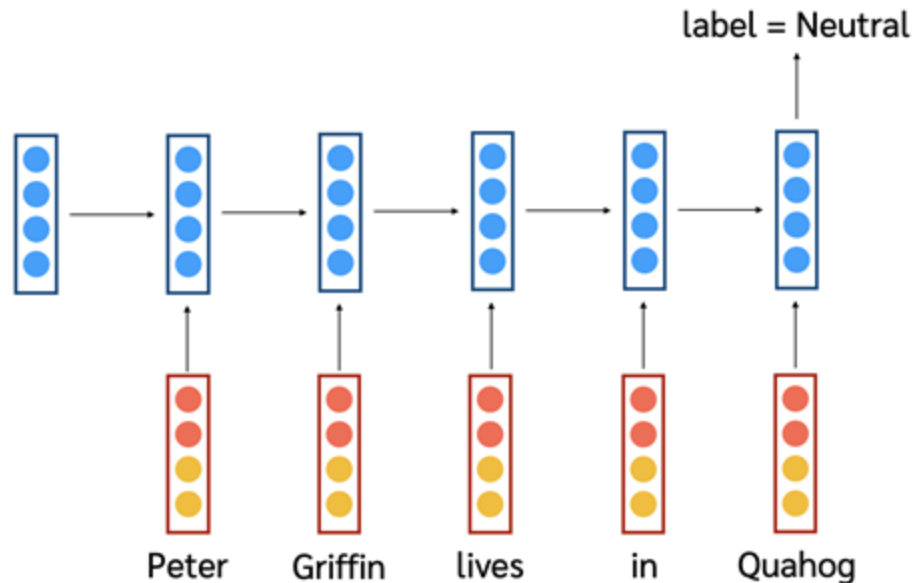


## Gradient clipping

- If the sum of squared gradient is greater than some threshold, then divide it so that it becomes smaller.
- Clipping gradient is an important and easy way to solve the problem of exploding gradient.

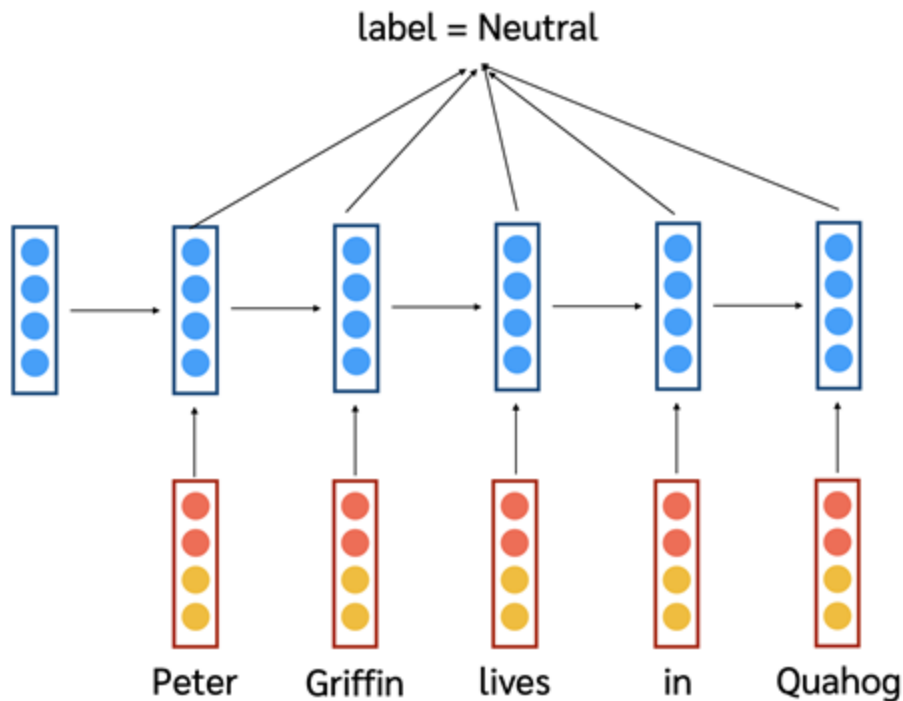
## RNN can be used for classification too.

- We can connect the last hidden vector to one output layer that predicts the probability of a classification label.
- We 'encode' a sentence with an RNN.



RNN can be used for classification too.

- We can also average or sum across all of the hidden vectors to 'encode' a sentence with an RNN.
- Actually quite effective and more effective than using just the last time step.



# Exploding and vanishing gradients

- The exploding gradient problem occurs when the gradients in a neural network become very large during training, causing the model's weights to update too aggressively and potentially leading to numerical instability or divergence.
- The vanishing gradient problem occurs because the gradients multiplication can cause the gradients to become exponentially smaller as they propagate backwards through time. When the gradients become very small, they no longer effectively update the weights of the earlier time steps in the sequence.



## How to solve vanishing gradient problem

- RNN cannot encode long-range dependencies in practice.
- The hidden vectors try to do two things simultaneously
  - Pass on the context to the future steps (memory)
  - Provide good features for the current step (feature vector)
- Let's separate the memory vector and feature vector.

A horizontal bar with a gold segment on the left and a red segment on the right.

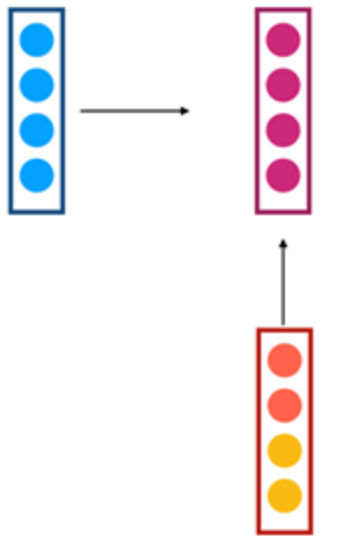
## GRU and LSTM

- The RNN we just discussed is usually called vanilla RNN.
- GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) are both types of recurrent neural networks (RNNs) that are designed to address the vanishing gradient problem.
- GRU is a simplified version of LSTM and can be just as effective as LSTM.



# RNN Cell

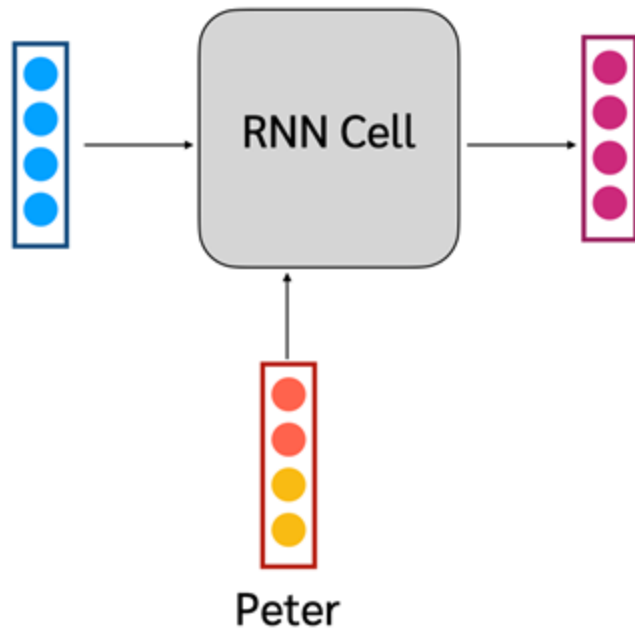
$$c_t = \tanh(W_c \cdot [c_{t-1}; x_t] + b_c)$$



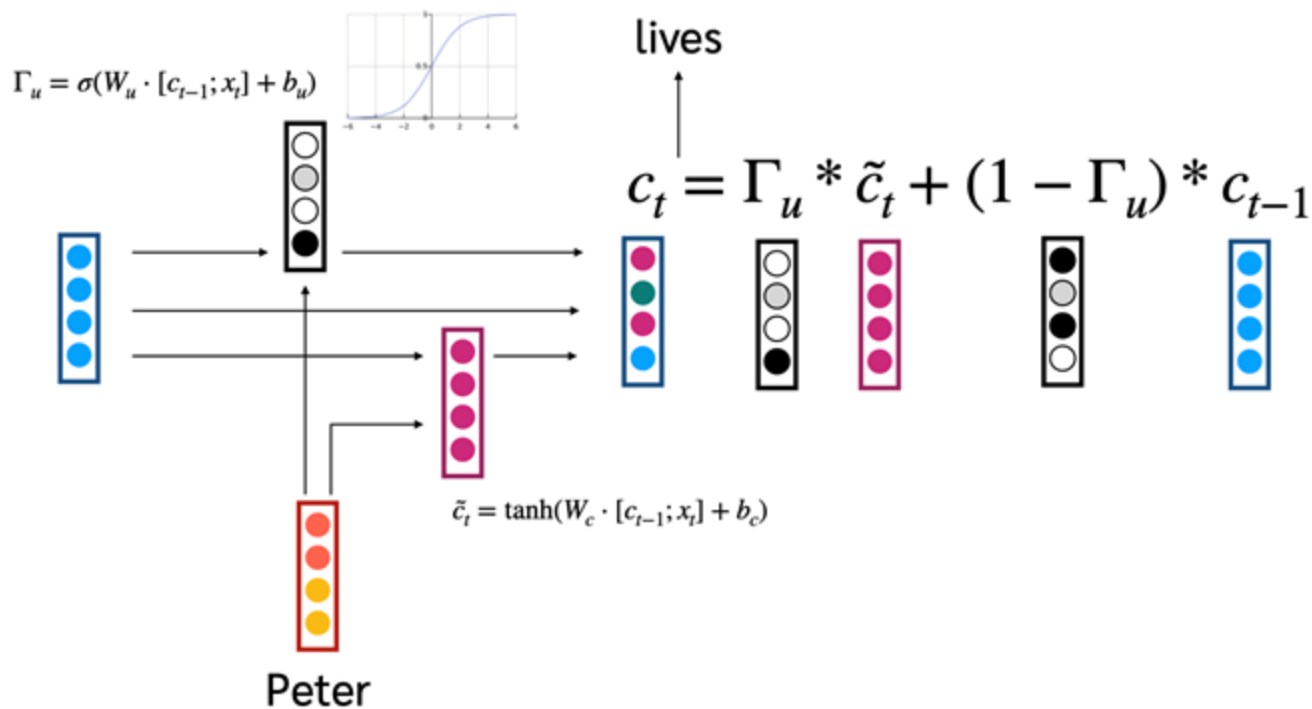
Peter

# RNN Cell

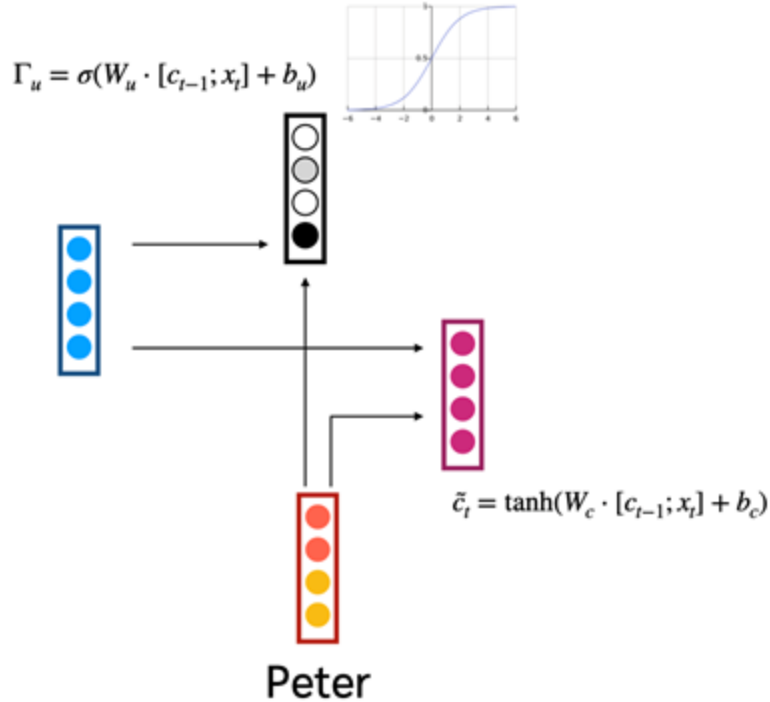
$$c_t = \tanh(W_c \cdot [c_{t-1}; x_t] + b_c)$$



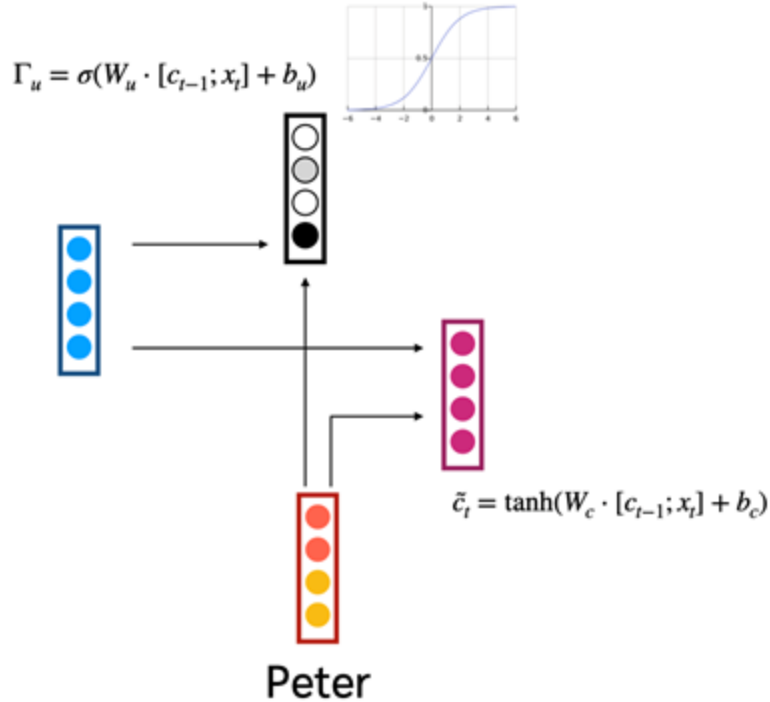
# (Simplified) Gated Recurrent Unit



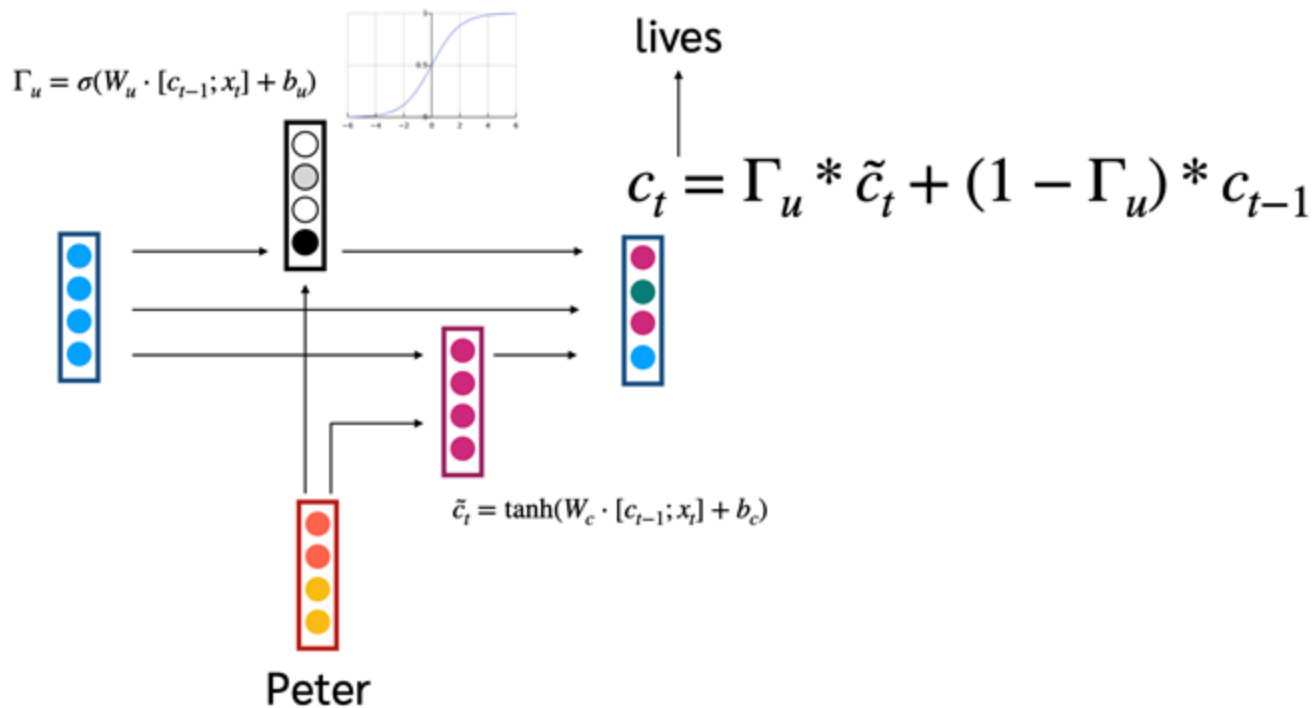
# (Simplified) Gated Recurrent Unit



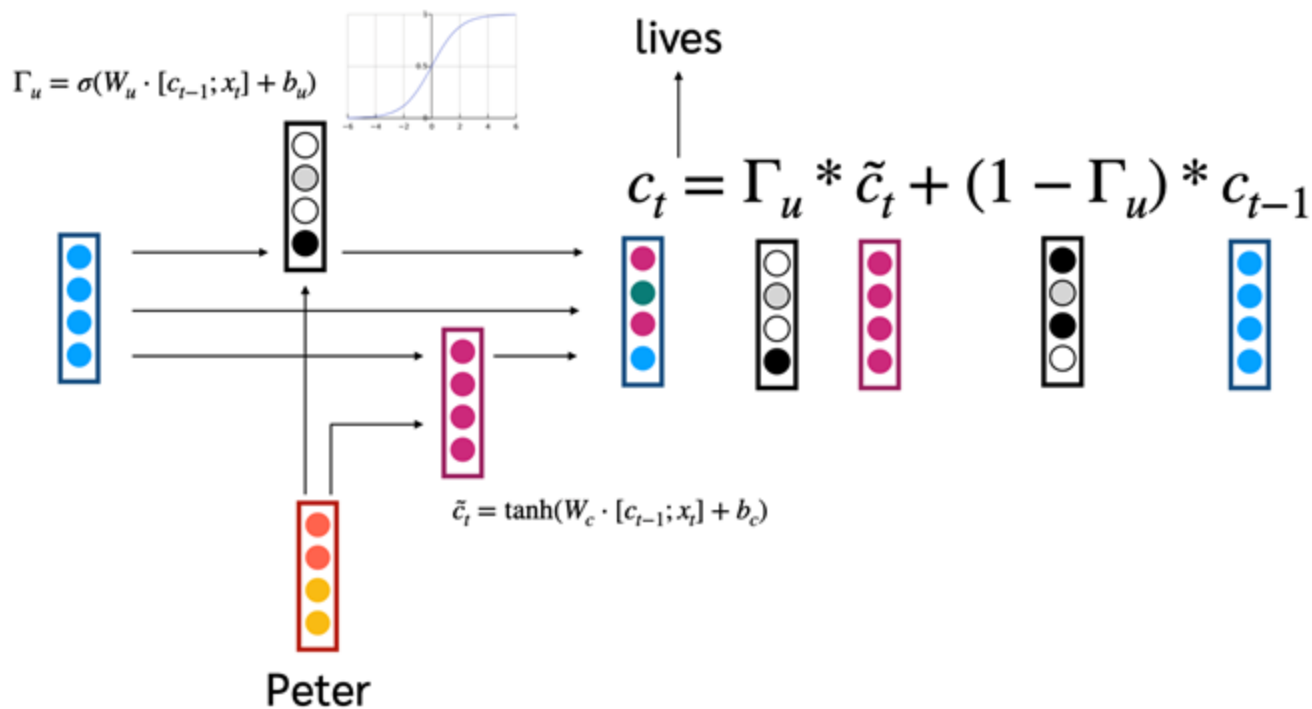
# (Simplified) Gated Recurrent Unit



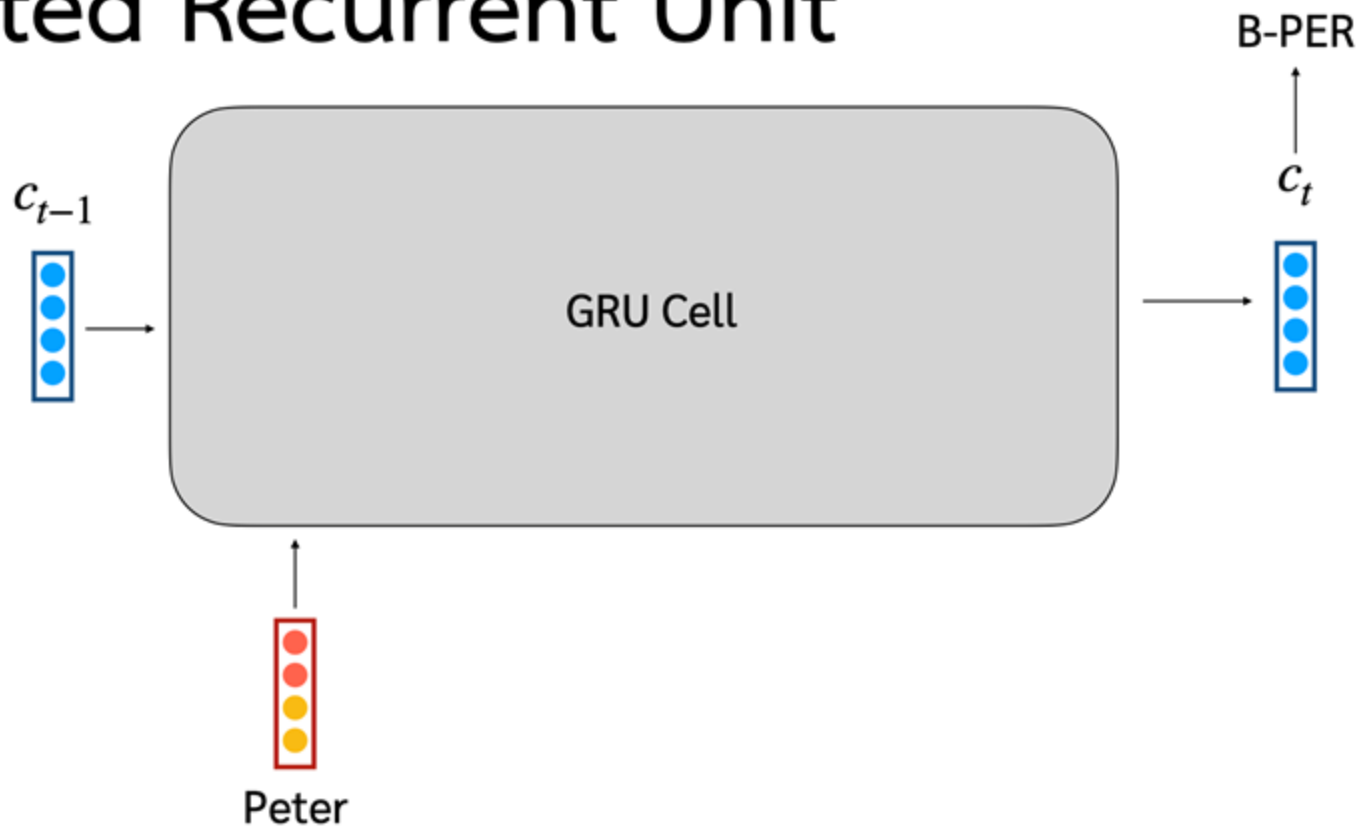
# (Simplified) Gated Recurrent Unit



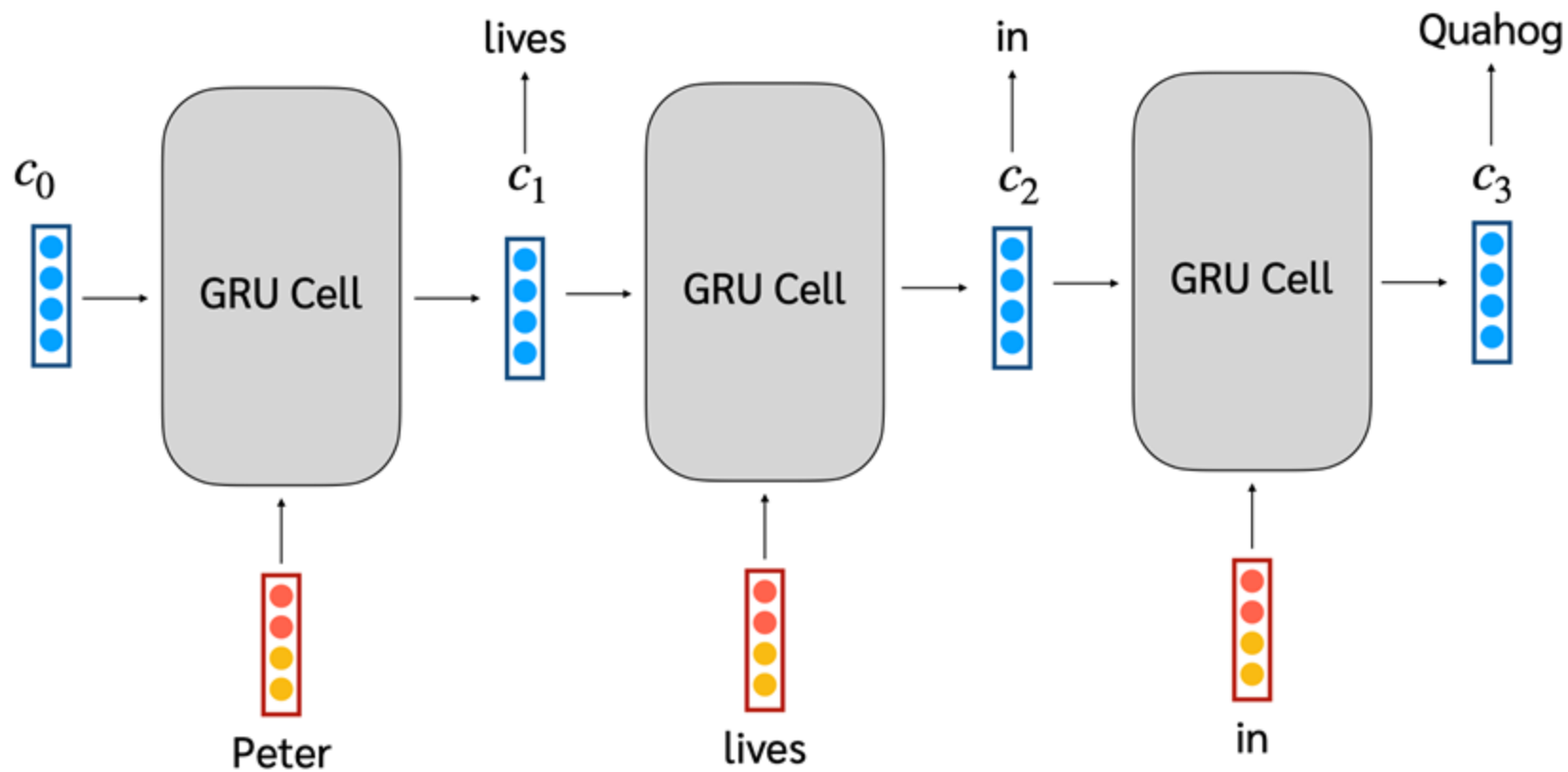
# (Simplified) Gated Recurrent Unit



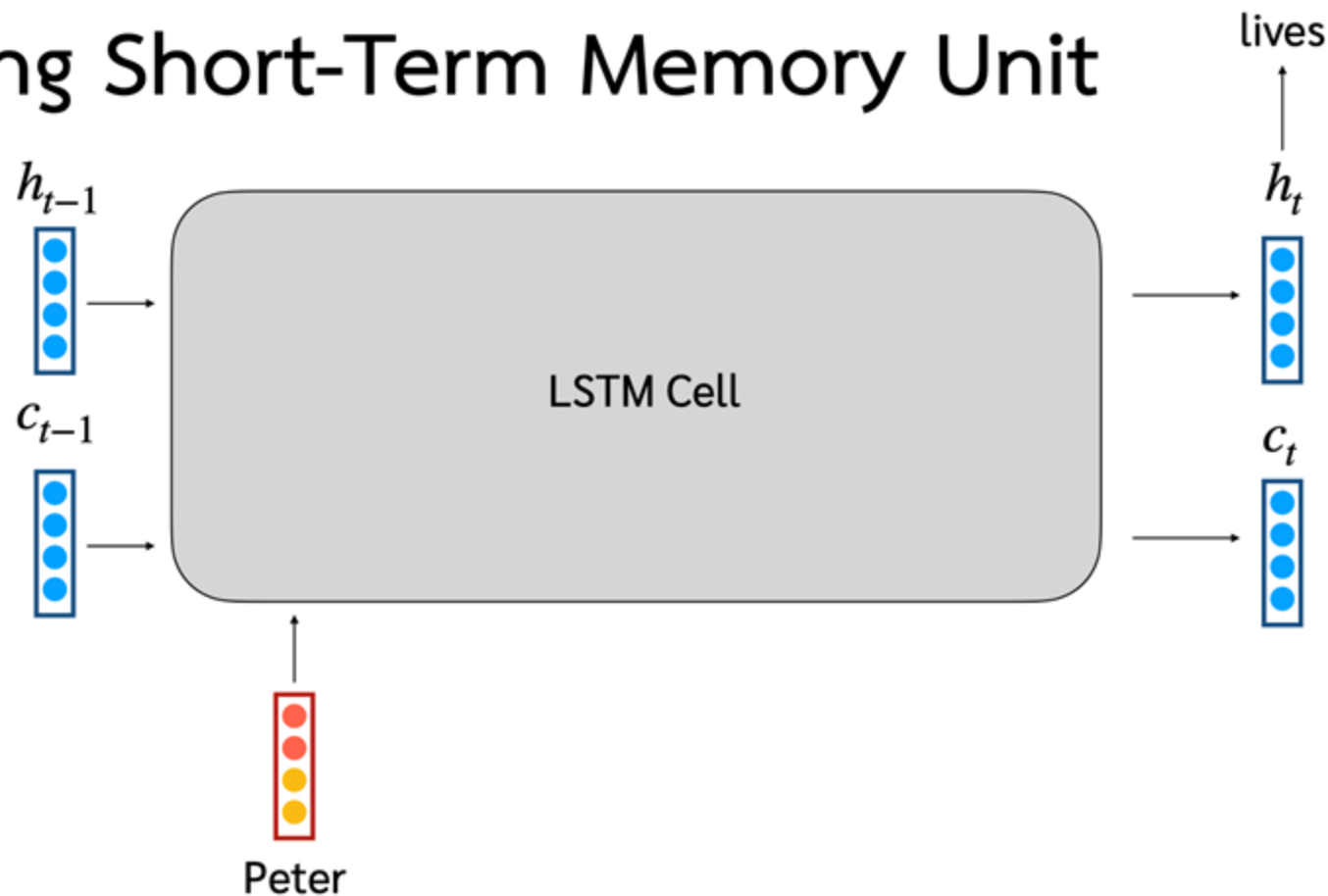
# Gated Recurrent Unit

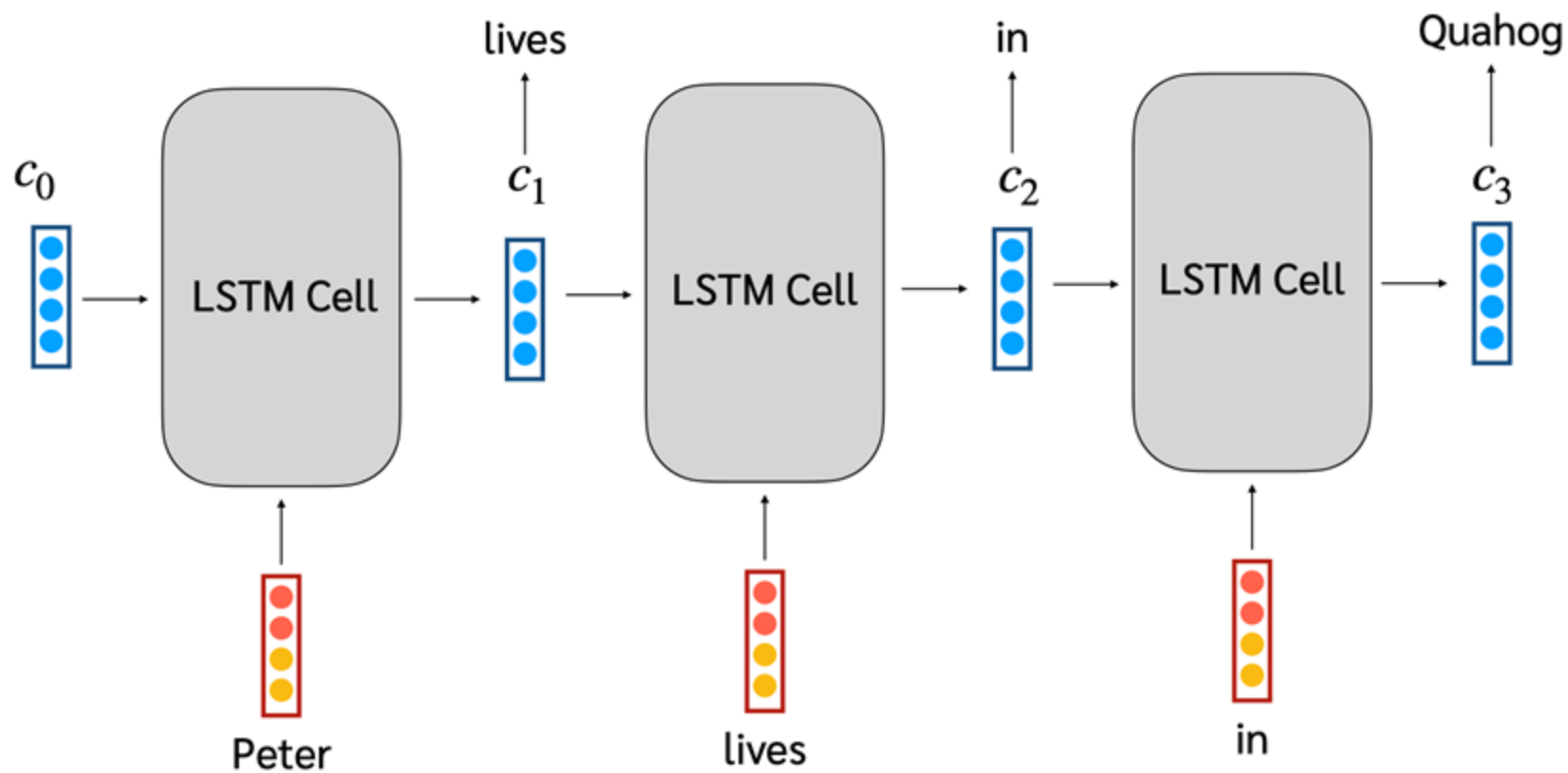


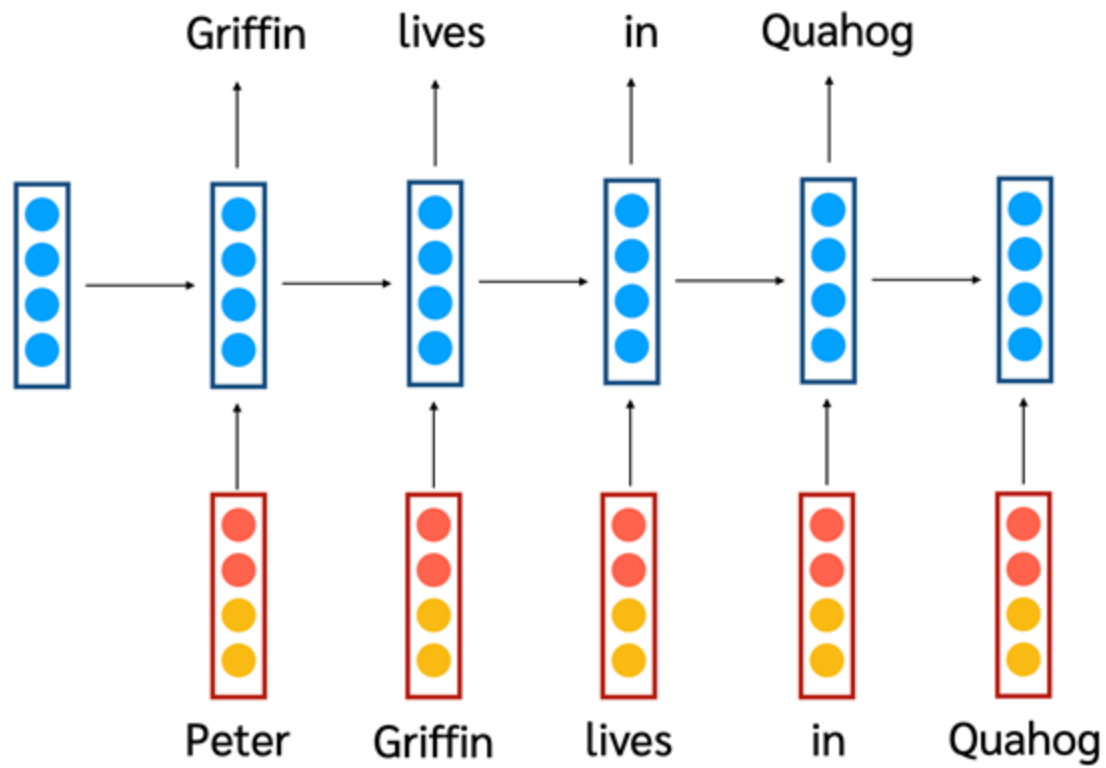


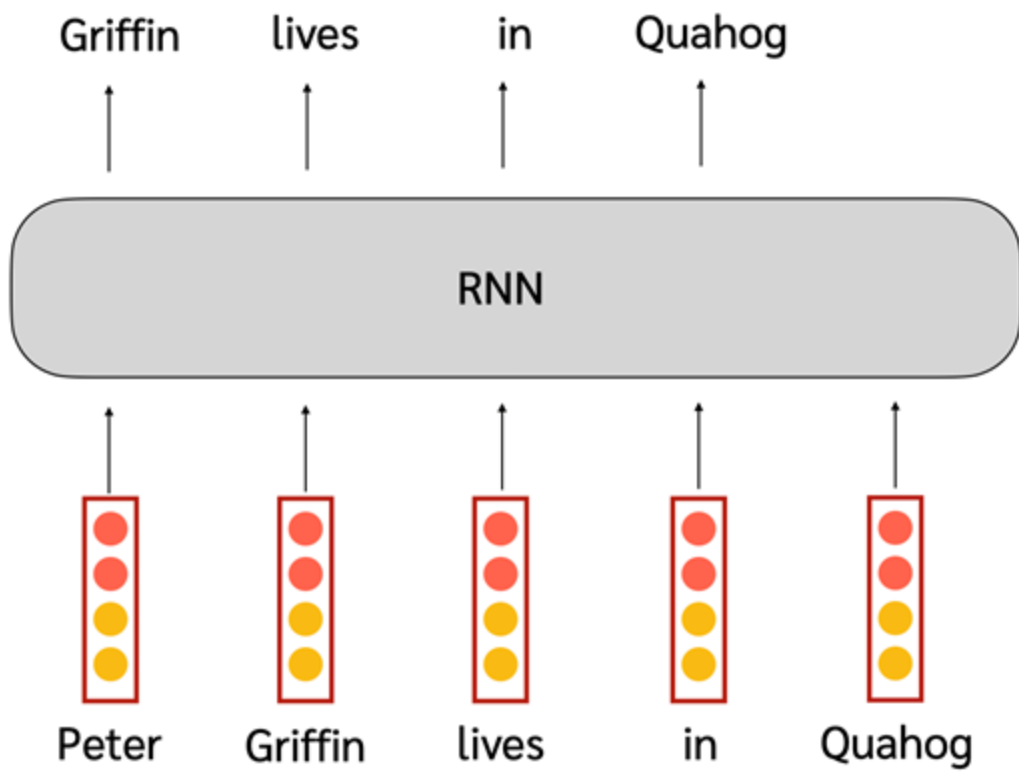


# Long Short-Term Memory Unit

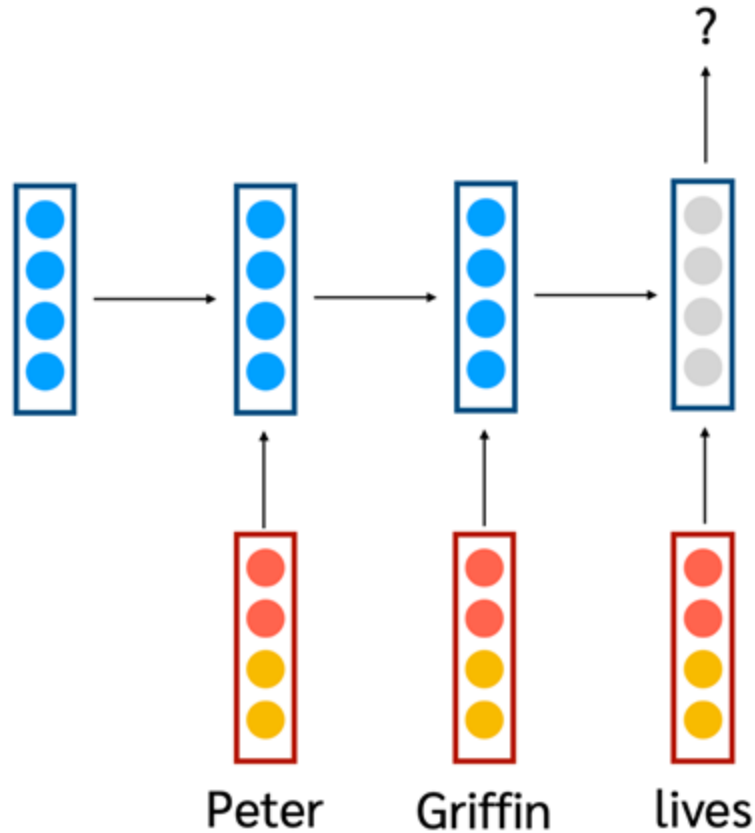






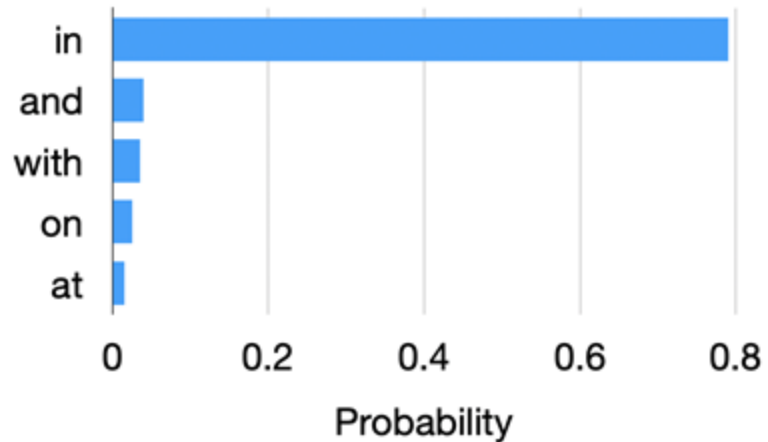






คำต่อไปคือคำว่าอะไร

$$y_t = \text{softmax}(W_y \cdot h_t + b_y)$$



A horizontal bar with a gold segment on the left and a red segment on the right.

# Data Preparation

1. Read data from raw text (word segment as necessary)
2. Create a vocabulary (Tokenizer)
3. Convert word strings into word indices
4. Pad sequences to be the same length
5. Offset the token sequences into label sequences



A horizontal bar with a gold segment on the left and a red segment on the right.

## 1. Read data from raw file

- A list of sentences (strings)
  - Each sentence should be tokens separated by space (in both English and Thai)



## Example - word segmentation

อร่อยจังเลย

ไม่รู้อร่อยตรงไหน

พอกินได้

กินจนพุงกาง

อร่อย จัง เลย

ไม่ รู้ อร่อย ตรง ไหน

พอ กิน ได้

กิน จน พุง กาง



## 2. Create a vocabulary

- Tokenizer class in keras learns from the list of sentences
  - word\_index
  - index\_word
  - 0 is for padding
- The indices are sorted by frequencies.
  - If an index is greater than the vocab size, then it's converted to the same index for out-of-vocabulary index (= vocab size)

## Example vocabulary - Vocab size = 10

index	word
0	<PADDING>
1	อรั๋ย
2	กิน
3	จั้ง
4	เลย
5	ม่
6	รู้
7	ตรง
8	ไหน
9	พอ
10	ได้
11	จน
12	พว

index	word
0	<PADDING>
1	อรั๋ย
2	กิน
3	จั้ง
4	เลย
5	ม่
6	รู้
7	ตรง
8	ไหน
9	พอ

A horizontal bar with a gold segment on the left and a red segment on the right.

### 3. Convert strings to indices

- Tokenizer class uses the vocabulary (`word_index` and `index_word`) to convert each word string into its index
- We get a sequence of indices

## Example

อร่อย จัง เลย

1 3 4

ไม่รู้ อร่อย ตรง ไหน

5 6 1 7 8

พอ กิน ได้

9 2 10

กิน จน พุง กาง

2 10 10 10

index	word
0	<PADDING>
1	อร่อย
2	กิน
3	จัง
4	เลย
5	ไม่
6	รู้
7	ตรง
8	ไหน
9	พอ

A horizontal bar with a gold segment on the left and a red segment on the right.

## 4. Pad sequences to the same length

- Each list of indices must be of the same length.
  - Pad empty spots with 0
  - Truncate the sequence to the same length. Why is this bad?
- We will run RNN on the whole list (sequence) regardless of the actual length of the sentence. But the timesteps with 0 will be 'masked.'
  - Choose the sequence length carefully. What if it's too long?

## Example - Set the sequence length to 8

อร่อย จัง เลย

1 3 4  
0 0 0 0 0 1 3 4

ไม่รู้ อร่อย ตรง ไหน

5 6 1 7 8  
0 0 0 5 6 1 7 8

พอ กิน ได้

9 2 10  
0 0 0 0 0 9 2 10

กิน จน พุง กาง

2 10 10 10  
0 0 0 0 2 10 10 10

index	word
0	<PADDING>
1	อร่อย
2	กิน
3	จัง
4	เลย
5	ไม่
6	รู้
7	ตรง
8	ไหน
9	พอ



## Example - Set the sequence length to 8

อร่อย จัง เลย

1 3 4  
0 0 0 0 0 1 3 4

ไม่รู้ อร่อย ตรง ไหน  
5 6 1 7 8  
0 0 0 5 6 1 7 8

พอ กิน ได้  
9 2 10  
0 0 0 0 0 9 2 10

กิน จน พุง กาง  
2 10 10 10  
0 0 0 0 2 10 10 10

We get the training data matrix:

```
0 0 0 0 0 1 3 4
0 0 0 5 6 1 7 8
0 0 0 0 0 9 2 10
0 0 0 0 2 10 10 10
```



## 5. Offset the tokens to create labels

- Offset = ทำให้มันเยื้องกัน
- For a token at timestep  $t$ , use the token at timestep  $t+1$  as label

word X				ไม่	รู้	อรรอย	ตรง	ไหน
index X	0	0	0	5	6	7	1	8
label Y				รู้	อรรอย	ตรง	ไหน	
index Y	0	0	5	6	7	1	8	

The actual sequence length is the length minus 1

word X				ไม่	รู้	อ้อย	ตรง	ไหน
index X	0	0	0	5	6	7	1	8
label Y				รู้	อ้อย	ตรง	ไหน	
index Y	0	0	5	6	7	1	8	

word X				ไม่	รู้	อ้อย	ตรง	ไหน
index X	<b>0</b>	<b>0</b>	<b>0</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>1</b>	
index Y		<b>0</b>	<b>0</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>1</b>	<b>8</b>

The actual sequence length is the length minus 1

## Example - actual data matrices

We get the training data matrix:

```
0 0 0 0 0 1 3 4
0 0 0 5 6 1 7 8
0 0 0 0 0 9 2 10
0 0 0 0 2 10 10 10
```

Actual X - clip the back

```
0 0 0 0 0 1 3
0 0 0 5 6 1 7
0 0 0 0 0 9 2
0 0 0 0 2 10 10
```

Label - clip the front

```
0 0 0 0 1 3 4
0 0 5 6 1 7 8
0 0 0 0 9 2 10
0 0 0 2 10 10 10
```

A horizontal bar with a gold segment on the left and a red segment on the right.

## Playing with RNN

- [https://colab.research.google.com/drive/1Hyc21QU7vS49jy-\\_Qf0fMvsHtHkyc4oh?usp=sharing](https://colab.research.google.com/drive/1Hyc21QU7vS49jy-_Qf0fMvsHtHkyc4oh?usp=sharing)
- Add your own pattern of strings
- Questions
  - What if you reduce the training set size?
  - What if you switch out vanilla RNN or GRU or LSTM?
  - What if you reduce the hidden layer size?

---

# Large Language Models

A horizontal bar with a gold segment on the left and a red segment on the right.

## History of neural language model

- 2003 - feedforward neural language model
- 2016 - recurrent neural language model (RNN-LM)
- 2017 - transformer-based language model
- 2019 - large transformer-based language model (e.g. GPT-2)
- 2022 - massive transformer-based language model fine-tuned for use as a chatbot (ChatGPT)





## Lessons learned from previous models

- N-gram models: Predicting the next word is an excellent way to learn languages and knowledge about the world from data.
- Neural n-gram models: Embeddings help with the sparsity problem.
- RNN-LM: Vanishing gradients occur because each timestep can only communicate through a memory cell.
- GRU and LSTM: Separating a vector for encoding context and a feature vector helps with vanishing gradient.

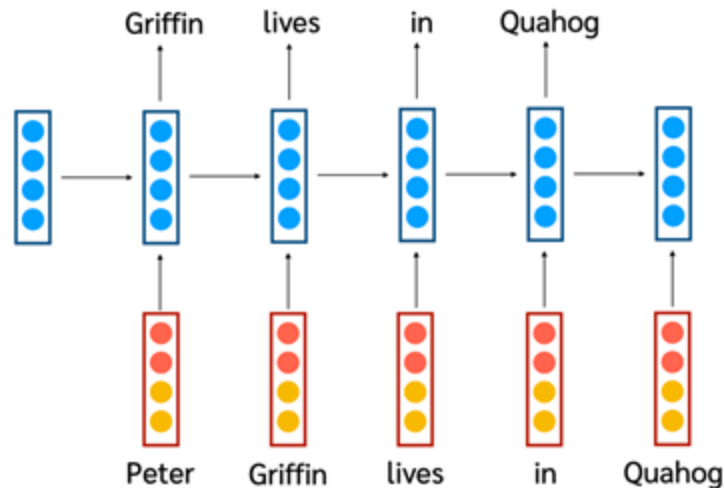
A horizontal bar with a gold segment on the left and a red segment on the right.

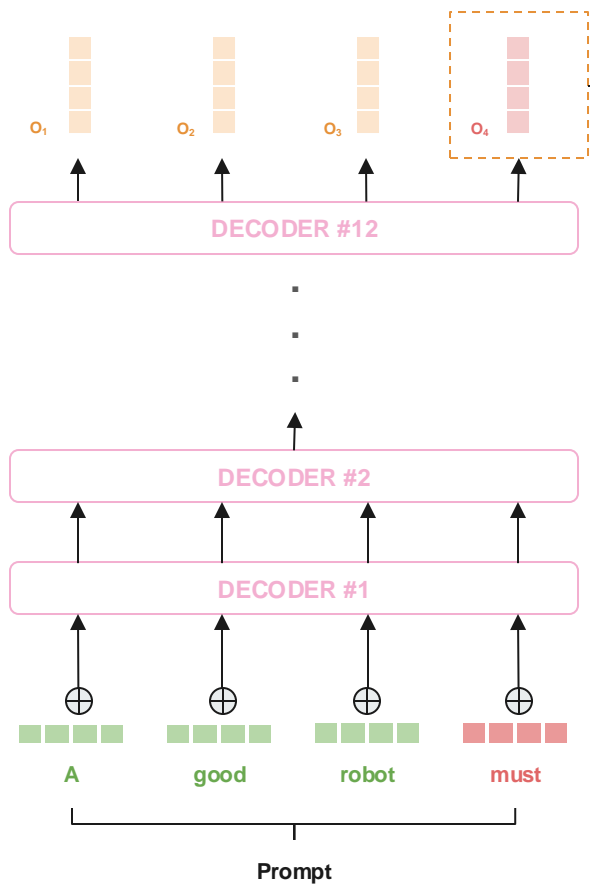
# GPT

- Generative Pretrained Transformer: Transformers are pre-trained to generate text (predicting the next word)
- GPT achieves an excellent results on perplexity and moderate results on other downstream tasks, including text classification

# Transformer

- We want each time step should be connected to the output more directly. The current time step should be computed from the previous steps directly without the memory cells.
- 'Attention is all you need' (2016) introduces 'attention mechanism'



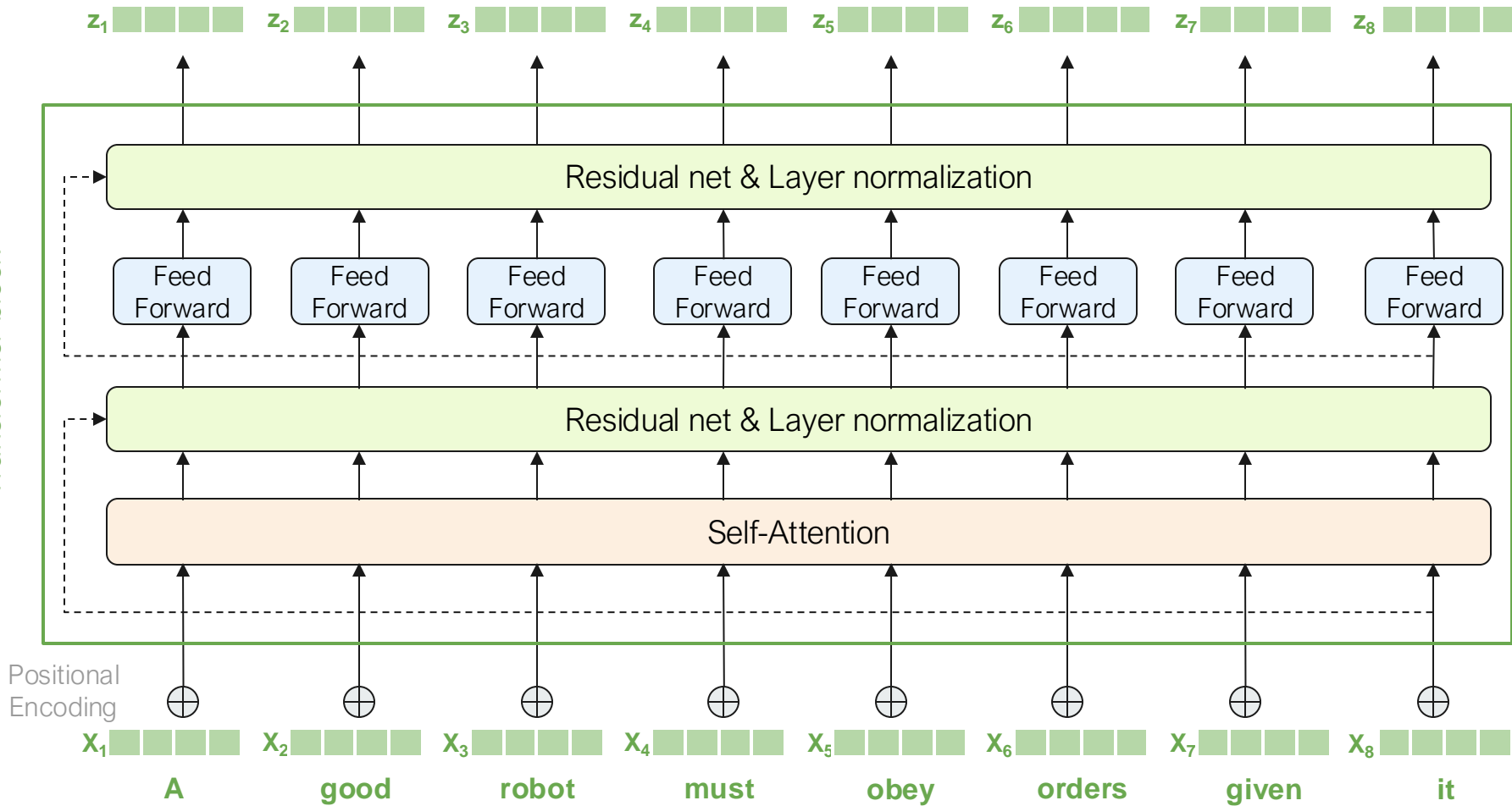


softmax (

$$\begin{matrix}
 \begin{bmatrix} \text{blue grid} \\ \vdots \\ \text{blue grid} \end{bmatrix} & \cdot & \begin{bmatrix} \text{red vector} \\ \vdots \\ \text{red vector} \end{bmatrix} & + & \begin{bmatrix} \text{blue vector} \\ \vdots \\ \text{blue vector} \end{bmatrix} & = \\
 W_l & & o_1 & & b_l
 \end{matrix}$$

0	a
0	and
0	but
0.07	ethically
<b>0.41</b>	<b>follow</b>
<b>0.36</b>	<b>obey</b>
0.02	orders
...	...
0.08	rules
0.06	set
0	strictly
0	the
0	without

Transformer block





# Attention Mechanism

Attention mechanism is the formula for computing a word embedding from the other words in the sentence by paying attention to different words the different amount.

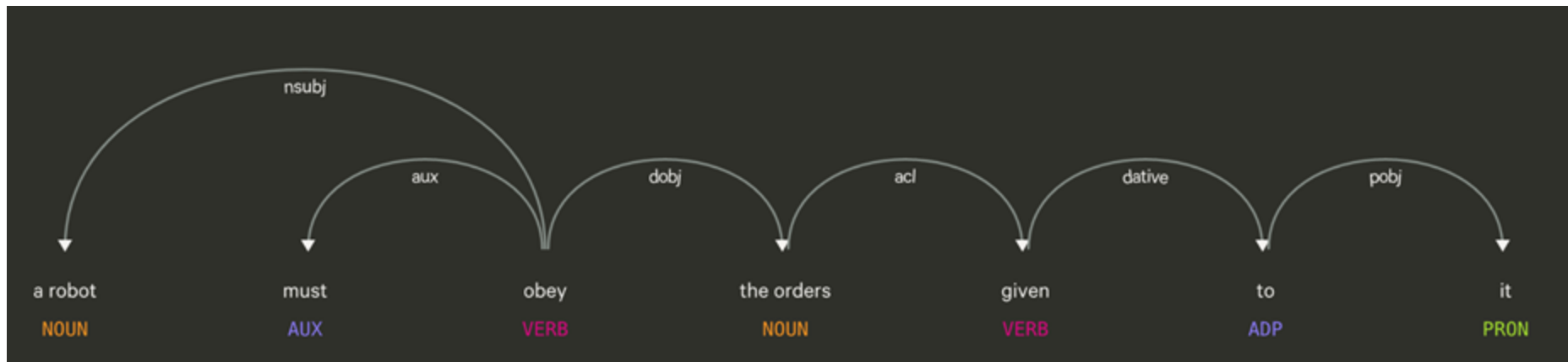


A horizontal bar with a gold segment on the left and a red segment on the right.

Consider this sentence

A robot must obey orders given to it.



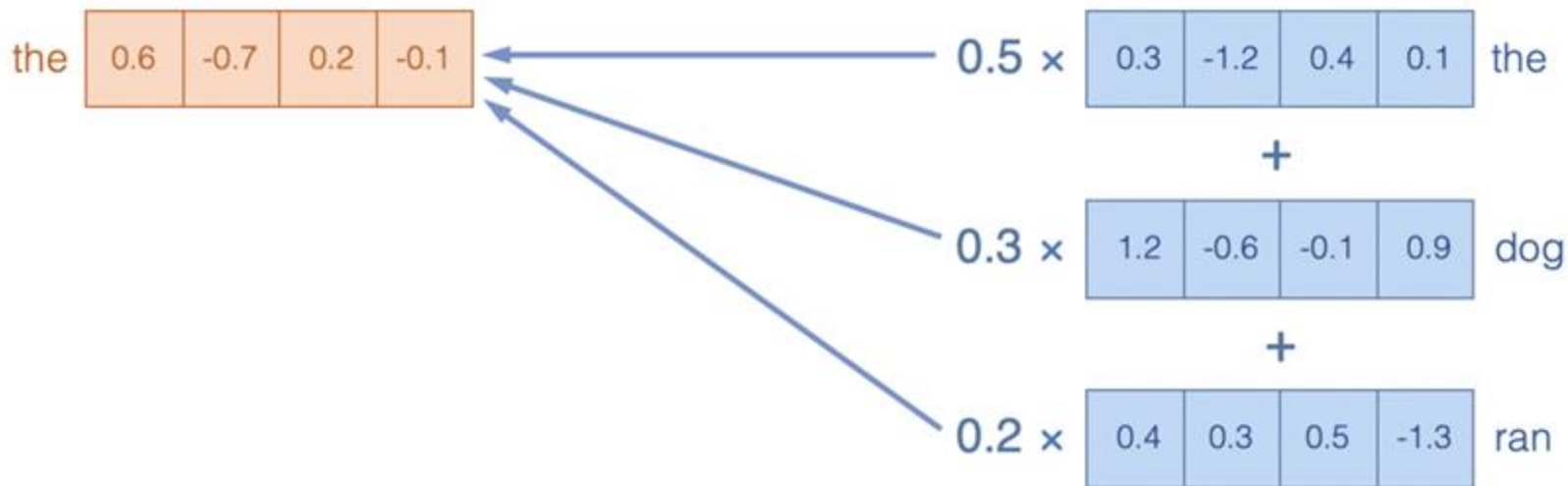




# Attention Mechanism

- Attention mechanism is the formula for computing a word embedding from the other words in the sentence by paying attention to different words the different amount. = relationship of words in sentences.
- It's implemented as a weighted average.

# Attention is a weighted average

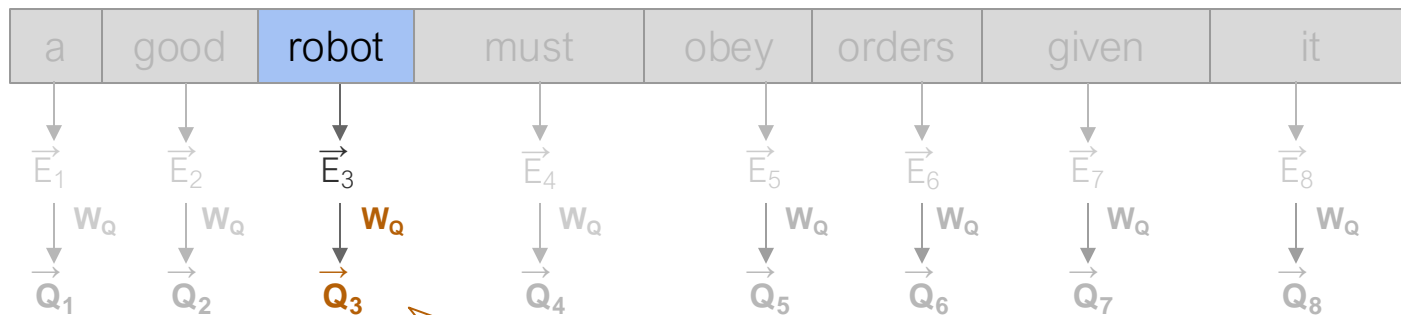


A horizontal bar with a gold segment on the left and a red segment on the right.

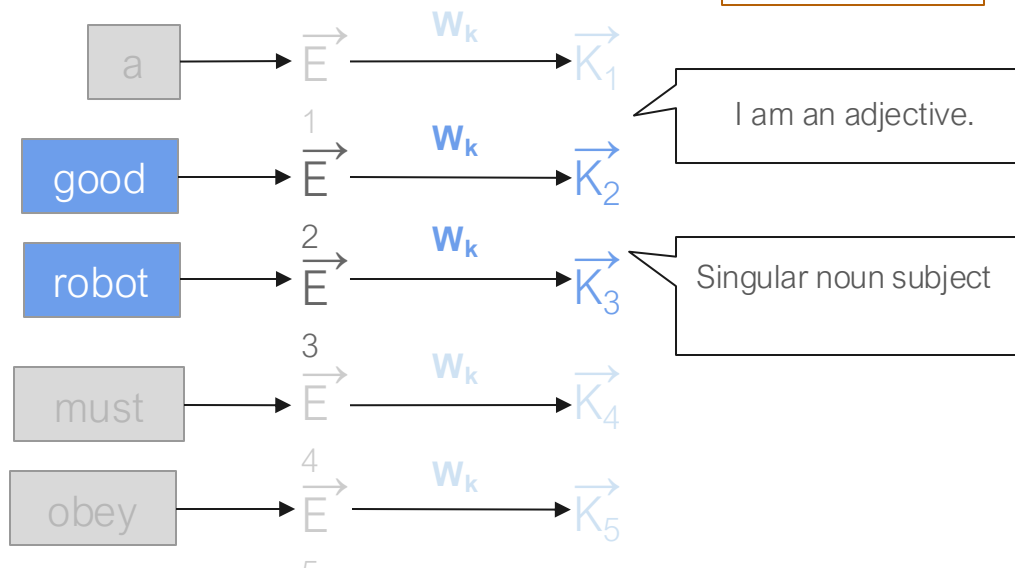
# Self-attention

- Query - the vector for the word that we want to compute the representation for
- Key - the vector for the word that we want to know how much attention we should give
- Value - the vector for the word that we use to compute the average





Any adjectives  
in front of me?



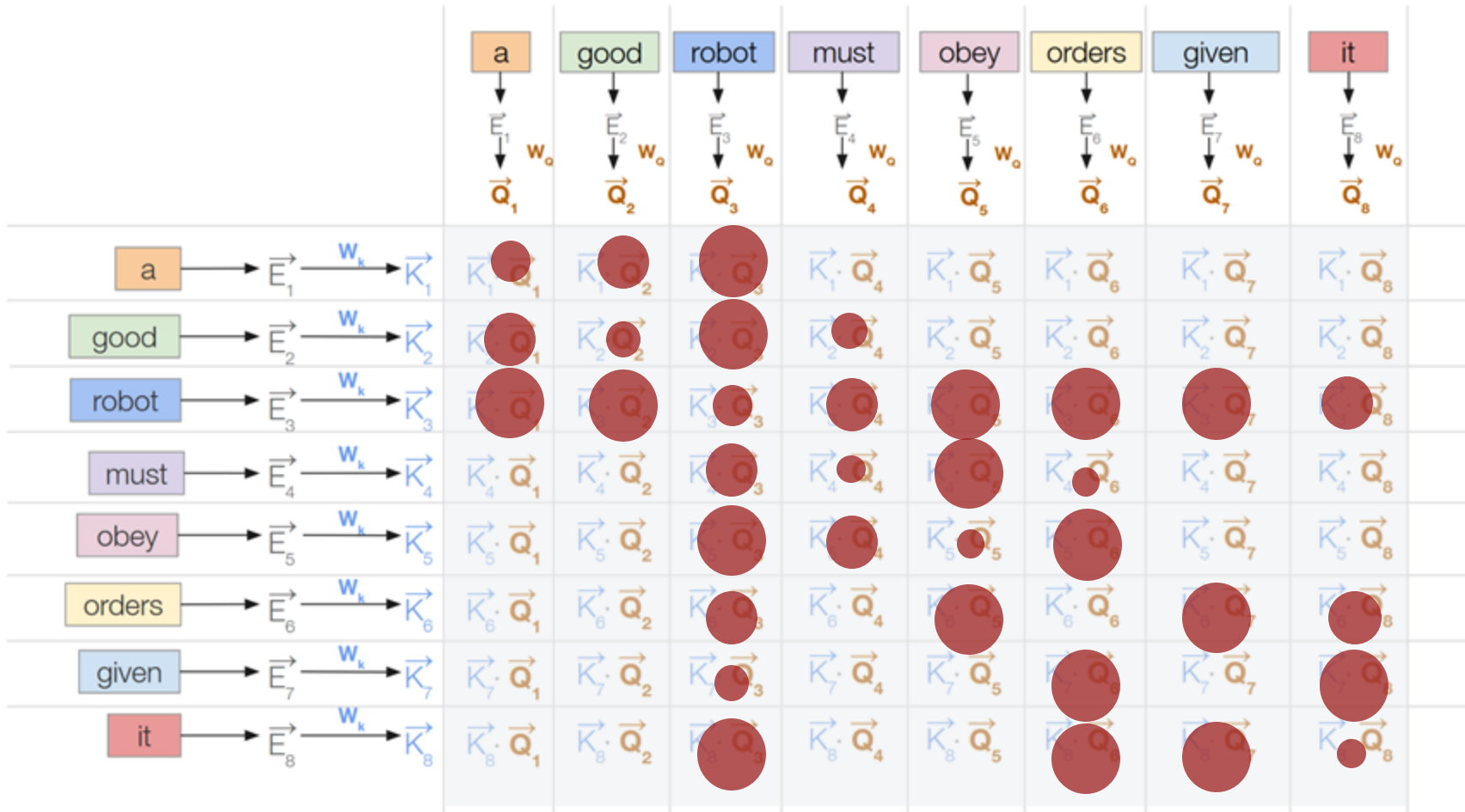
A horizontal bar with a gold segment on the left and a red segment on the right.

## Self-attention in Transformer

Self-attention can be thought of as a way for the model to pay attention to the most relevant parts of the input sequence for each step in the prediction process. Instead of processing the entire input sequence in the same way, self-attention allows the model to dynamically adjust its focus based on the context of the current step.

		<div>a</div> $\vec{E}_1 \xrightarrow{w_o} \vec{Q}_1$	<div>good</div> $\vec{E}_2 \xrightarrow{w_o} \vec{Q}_2$	<div>robot</div> $\vec{E}_3 \xrightarrow{w_o} \vec{Q}_3$	<div>must</div> $\vec{E}_4 \xrightarrow{w_o} \vec{Q}_4$	<div>obey</div> $\vec{E}_5 \xrightarrow{w_o} \vec{Q}_5$	<div>orders</div> $\vec{E}_6 \xrightarrow{w_o} \vec{Q}_6$	<div>given</div> $\vec{E}_7 \xrightarrow{w_o} \vec{Q}_7$	<div>it</div> $\vec{E}_8 \xrightarrow{w_o} \vec{Q}_8$
<div>a</div>	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{K}_1 \cdot \vec{Q}_1$	$\vec{K}_1 \cdot \vec{Q}_2$	$\vec{K}_1 \cdot \vec{Q}_3$	$\vec{K}_1 \cdot \vec{Q}_4$	$\vec{K}_1 \cdot \vec{Q}_5$	$\vec{K}_1 \cdot \vec{Q}_6$	$\vec{K}_1 \cdot \vec{Q}_7$	$\vec{K}_1 \cdot \vec{Q}_8$
<div>good</div>	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{K}_2 \cdot \vec{Q}_1$	$\vec{K}_2 \cdot \vec{Q}_2$	$\vec{K}_2 \cdot \vec{Q}_3$	$\vec{K}_2 \cdot \vec{Q}_4$	$\vec{K}_2 \cdot \vec{Q}_5$	$\vec{K}_2 \cdot \vec{Q}_6$	$\vec{K}_2 \cdot \vec{Q}_7$	$\vec{K}_2 \cdot \vec{Q}_8$
<div>robot</div>	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{K}_3 \cdot \vec{Q}_1$	$\vec{K}_3 \cdot \vec{Q}_2$	$\vec{K}_3 \cdot \vec{Q}_3$	$\vec{K}_3 \cdot \vec{Q}_4$	$\vec{K}_3 \cdot \vec{Q}_5$	$\vec{K}_3 \cdot \vec{Q}_6$	$\vec{K}_3 \cdot \vec{Q}_7$	$\vec{K}_3 \cdot \vec{Q}_8$
<div>must</div>	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{K}_4 \cdot \vec{Q}_1$	$\vec{K}_4 \cdot \vec{Q}_2$	$\vec{K}_4 \cdot \vec{Q}_3$	$\vec{K}_4 \cdot \vec{Q}_4$	$\vec{K}_4 \cdot \vec{Q}_5$	$\vec{K}_4 \cdot \vec{Q}_6$	$\vec{K}_4 \cdot \vec{Q}_7$	$\vec{K}_4 \cdot \vec{Q}_8$
<div>obey</div>	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{K}_5 \cdot \vec{Q}_1$	$\vec{K}_5 \cdot \vec{Q}_2$	$\vec{K}_5 \cdot \vec{Q}_3$	$\vec{K}_5 \cdot \vec{Q}_4$	$\vec{K}_5 \cdot \vec{Q}_5$	$\vec{K}_5 \cdot \vec{Q}_6$	$\vec{K}_5 \cdot \vec{Q}_7$	$\vec{K}_5 \cdot \vec{Q}_8$
<div>orders</div>	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{K}_6 \cdot \vec{Q}_1$	$\vec{K}_6 \cdot \vec{Q}_2$	$\vec{K}_6 \cdot \vec{Q}_3$	$\vec{K}_6 \cdot \vec{Q}_4$	$\vec{K}_6 \cdot \vec{Q}_5$	$\vec{K}_6 \cdot \vec{Q}_6$	$\vec{K}_6 \cdot \vec{Q}_7$	$\vec{K}_6 \cdot \vec{Q}_8$
<div>given</div>	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{K}_7 \cdot \vec{Q}_1$	$\vec{K}_7 \cdot \vec{Q}_2$	$\vec{K}_7 \cdot \vec{Q}_3$	$\vec{K}_7 \cdot \vec{Q}_4$	$\vec{K}_7 \cdot \vec{Q}_5$	$\vec{K}_7 \cdot \vec{Q}_6$	$\vec{K}_7 \cdot \vec{Q}_7$	$\vec{K}_7 \cdot \vec{Q}_8$
<div>it</div>	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$	$\vec{K}_8 \cdot \vec{Q}_1$	$\vec{K}_8 \cdot \vec{Q}_2$	$\vec{K}_8 \cdot \vec{Q}_3$	$\vec{K}_8 \cdot \vec{Q}_4$	$\vec{K}_8 \cdot \vec{Q}_5$	$\vec{K}_8 \cdot \vec{Q}_6$	$\vec{K}_8 \cdot \vec{Q}_7$	$\vec{K}_8 \cdot \vec{Q}_8$





robot	must	obey	orders	given	it
$\vec{E}_3$	$\vec{E}_4$	$\vec{E}_5$	$\vec{E}_6$	$\vec{E}_7$	$\vec{E}_8$
$\vec{Q}_3$	$\vec{Q}_4$	$\vec{Q}_5$	$\vec{Q}_6$	$\vec{Q}_7$	$\vec{Q}_8$
79.6	+20.3	+10.7	+5.1	+3.4	-12.4
89.7	+13.3	+2.2	+5.2	+2.8	-20
78.8	+20.6	+25.3	+10.8	+3.8	43.3
-25.2	We want these to act like weights				18.9
88.4	+20.7	+50.8	+30.3	+10.7	-23.5
-6.9	+15.4	+30.8	+40.3	+35.8	25.6
-20.2	+10.2	+25.4	+30.1	+45.7	41.9
78.2	+5.8	+10.1	+10.0	+30.0	34.7

It





Value Matrix $W_v$	a ↓ $E_1$	good ↓ $E_2$	robot ↓ $E_3$	must ↓ $E_4$	obey ↓ $E_5$	orders ↓ $E_6$	given ↓ $E_7$	it ↓ $E_8$
a → $\vec{E}_1 \xrightarrow{W_v} \vec{V}_1$					0.00			
good → $\vec{E}_2 \xrightarrow{W_v} \vec{V}_2$					0.00			
robot → $\vec{E}_3 \xrightarrow{W_v} \vec{V}_3$					0.14			
must → $\vec{E}_4 \xrightarrow{W_v} \vec{V}_4$					0.32			
obey → $\vec{E}_5 \xrightarrow{W_v} \vec{V}_5$					0.00			
orders → $\vec{E}_6 \xrightarrow{W_v} \vec{V}_6$					0.54			
given → $\vec{E}_7 \xrightarrow{W_v} \vec{V}_7$					0.00			
it → $\vec{E}_8 \xrightarrow{W_v} \vec{V}_8$					0.00			

Value Matrix $W_v$	a ↓ $E_1$	good ↓ $E_2$	robot ↓ $E_3$	must ↓ $E_4$	obey ↓ $E_5$	orders ↓ $E_6$	given ↓ $E_7$	it ↓ $E_8$
a → $E_1 \xrightarrow{w_v} \vec{V}_1$					0.00 · $\vec{V}_1$			
good → $E_2 \xrightarrow{w_v} \vec{V}_2$					0.00 · $\vec{V}_2$			
robot → $E_3 \xrightarrow{w_v} \vec{V}_3$					<b>0.14</b> · $\vec{V}_3$			
must → $E_4 \xrightarrow{w_v} \vec{V}_4$					<b>0.32</b> · $\vec{V}_4$			
obey → $E_5 \xrightarrow{w_v} \vec{V}_5$					0.00 · $\vec{V}_5$			
orders → $E_6 \xrightarrow{w_v} \vec{V}_6$					<b>0.54</b> · $\vec{V}_6$			
given → $E_7 \xrightarrow{w_v} \vec{V}_7$					0.00 · $\vec{V}_7$			
it → $E_8 \xrightarrow{w_v} \vec{V}_8$					0.00 · $\vec{V}_8$			

Value Matrix $W_v$	a $\downarrow$ $\vec{E}_1$	good $\downarrow$ $\vec{E}_2$	robot $\downarrow$ $\vec{E}_3$	must $\downarrow$ $\vec{E}_4$	obey $\downarrow$ $\vec{E}_5$	orders $\downarrow$ $\vec{E}_6$	given $\downarrow$ $\vec{E}_7$	it $\downarrow$ $\vec{E}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_v} \vec{V}_1$					$0.00 \cdot \vec{V}_1$			
good $\rightarrow \vec{E}_2 \xrightarrow{W_v} \vec{V}_2$					$0.00 \cdot \vec{V}_2$			
robot $\rightarrow \vec{E}_3 \xrightarrow{W_v} \vec{V}_3$					$0.14 \cdot \vec{V}_3$			
must $\rightarrow \vec{E}_4 \xrightarrow{W_v} \vec{V}_4$					$0.32 \cdot \vec{V}_4$			
obey $\rightarrow \vec{E}_5 \xrightarrow{W_v} \vec{V}_5$					$0.00 \cdot \vec{V}_5$			
orders $\rightarrow \vec{E}_6 \xrightarrow{W_v} \vec{V}_6$					$0.54 \cdot \vec{V}_6$			
given $\rightarrow \vec{E}_7 \xrightarrow{W_v} \vec{V}_7$					$0.00 \cdot \vec{V}_7$			
it $\rightarrow \vec{E}_8 \xrightarrow{W_v} \vec{V}_8$					$0.00 \cdot \vec{V}_8$			

$$\begin{array}{cccccccc}
 \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0.14 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0.32 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0.54 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & + 0 & \begin{array}{|c|} \hline \\ \hline \\ \hline \\ \hline \\ \hline \end{array} & = \vec{E}'_5 \\
v1 & & v2 & & v6 & & v7 & & v3 & & v8 & & v4 & & v5 & & 
\end{array}$$

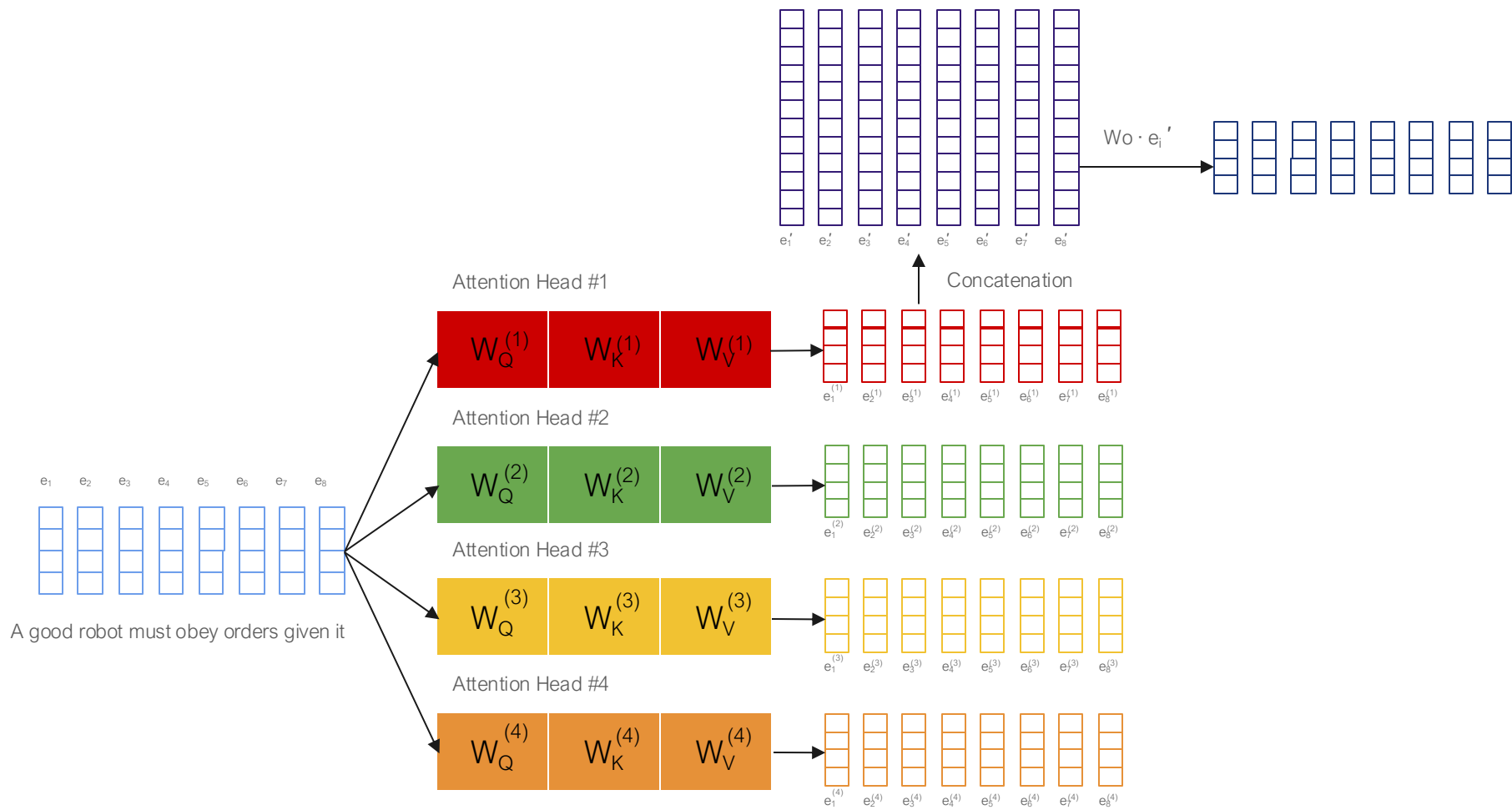
Value Matrix $W_v$	<div>a</div> <div>↓</div> $\vec{E}_1$	<div>good</div> <div>↓</div> $\vec{E}_2$	<div>robot</div> <div>↓</div> $\vec{E}_3$	<div>must</div> <div>↓</div> $\vec{E}_4$	<div>obey</div> <div>↓</div> $\vec{E}_5$	<div>orders</div> <div>↓</div> $\vec{E}_6$	<div>given</div> <div>↓</div> $\vec{E}_7$	<div>it</div> <div>↓</div> $\vec{E}_8$
<div>a</div> <div>→</div> $\vec{E}_1$ <div>→ <math>W_v</math></div> $\vec{V}_1$	$0.16 \cdot \vec{V}_1$	$0.23 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$	$0.00 \cdot \vec{V}_1$
<div>good</div> <div>→</div> $\vec{E}_2$ <div>→ <math>W_v</math></div> $\vec{V}_2$	$0.32 \cdot \vec{V}_2$	$0.12 \cdot \vec{V}_2$	$0.79 \cdot \vec{V}_2$	$0.00 \cdot \vec{V}_2$	$0.00 \cdot \vec{V}_2$	$0.00 \cdot \vec{V}_2$	$0.00 \cdot \vec{V}_2$	$0.00 \cdot \vec{V}_2$
<div>robot</div> <div>→</div> $\vec{E}_3$ <div>→ <math>W_v</math></div> $\vec{V}_3$	$0.52 \cdot \vec{V}_3$	$0.65 \cdot \vec{V}_3$	$0.00 \cdot \vec{V}_3$	$0.37 \cdot \vec{V}_3$	$0.14 \cdot \vec{V}_3$	$0.22 \cdot \vec{V}_3$	$0.00 \cdot \vec{V}_3$	$0.80 \cdot \vec{V}_3$
<div>must</div> <div>→</div> $\vec{E}_4$ <div>→ <math>W_v</math></div> $\vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.32 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$	$0.00 \cdot \vec{V}_4$
<div>obey</div> <div>→</div> $\vec{E}_5$ <div>→ <math>W_v</math></div> $\vec{V}_5$	$0.00 \cdot \vec{V}_5$	$0.00 \cdot \vec{V}_5$	$0.21 \cdot \vec{V}_5$	$0.63 \cdot \vec{V}_5$	$0.00 \cdot \vec{V}_5$	$0.34 \cdot \vec{V}_5$	$0.00 \cdot \vec{V}_5$	$0.00 \cdot \vec{V}_5$
<div>orders</div> <div>→</div> $\vec{E}_6$ <div>→ <math>W_v</math></div> $\vec{V}_6$	$0.00 \cdot \vec{V}_6$	$0.00 \cdot \vec{V}_6$	$0.00 \cdot \vec{V}_6$	$0.00 \cdot \vec{V}_6$	$0.54 \cdot \vec{V}_6$	$0.00 \cdot \vec{V}_6$	$0.48 \cdot \vec{V}_6$	$0.00 \cdot \vec{V}_6$
<div>given</div> <div>→</div> $\vec{E}_7$ <div>→ <math>W_v</math></div> $\vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.37 \cdot \vec{V}_7$	$0.00 \cdot \vec{V}_7$	$0.20 \cdot \vec{V}_7$
<div>it</div> <div>→</div> $\vec{E}_8$ <div>→ <math>W_v</math></div> $\vec{V}_8$	$0.00 \cdot \vec{V}_8$	$0.00 \cdot \vec{V}_8$	$0.00 \cdot \vec{V}_8$	$0.00 \cdot \vec{V}_8$	$0.00 \cdot \vec{V}_8$	$0.07 \cdot \vec{V}_8$	$0.52 \cdot \vec{V}_8$	$0.00 \cdot \vec{V}_8$
	 $\vec{E}'_1$	 $\vec{E}'_2$	 $\vec{E}'_3$	 $\vec{E}'_4$	 $\vec{E}'_5$	 $\vec{E}'_6$	 $\vec{E}'_7$	 $\vec{E}'_8$

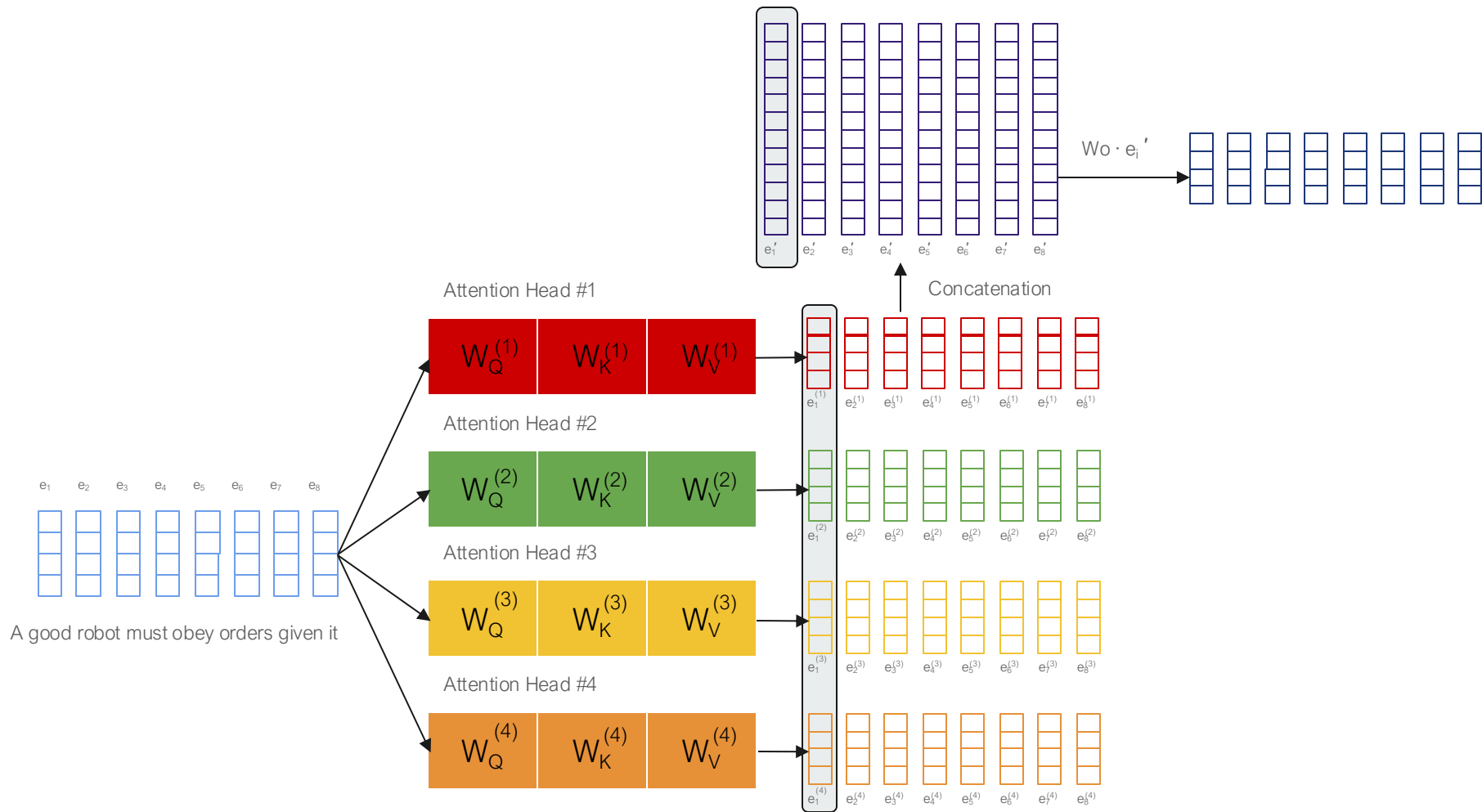


## Multiheaded self-attention

- One attention head learns one set of relationships between words.
- Why not more than one set? One head = one set of  $K$   $Q$   $V$







A horizontal bar with a gold segment on the left and a red segment on the right.

## Recap: Self-attention

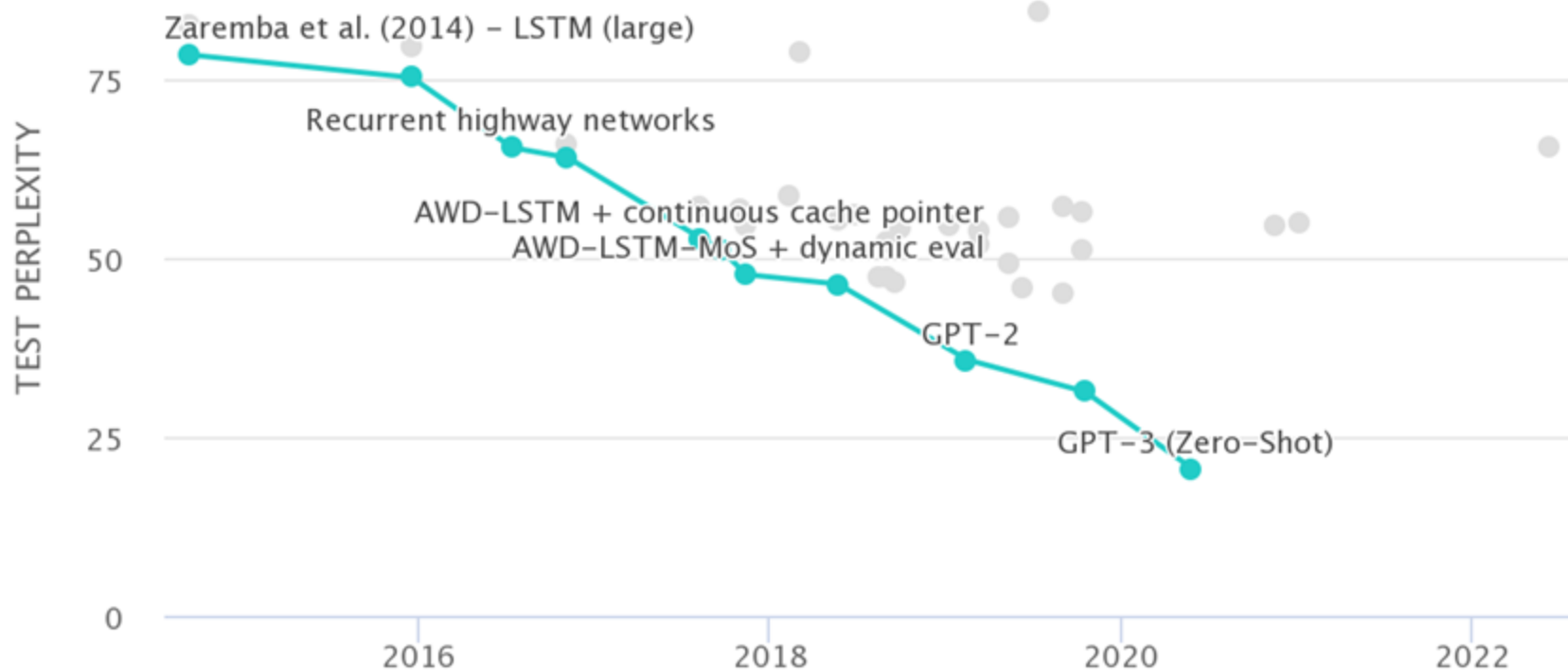
- Attention mechanism learns the relationship between the query word and the surrounding words. The relationship is modeled with query and key vectors.
- The new representation uses that relationship and value vectors.

A horizontal bar with a gold segment on the left and a red segment on the right.

## History of neural language model

- 2003 - feedforward neural language model
- 2016 - recurrent neural language model (RNN-LM)
- 2017 - transformer-based language model
- 2019 - large transformer-based language model (e.g. GPT-2)
- 2022 - massive transformer-based language model fine-tuned for use as a chatbot (ChatGPT)

# Penn Treebank Benchmark



---

# Language Models are Unsupervised Multitask Learners

---

GPT-2

Alec Radford \*<sup>1</sup> Jeffrey Wu \*<sup>1</sup> Rewon Child<sup>1</sup> David Luan<sup>1</sup> Dario Amodei \*\*<sup>1</sup> Ilya Sutskever \*\*<sup>1</sup>

---

## Language Models are Few-Shot Learners

---

GPT-3

[cs.CL] 22 Jul 2020

Tom B. Brown*	Benjamin Mann*	Nick Ryder*	Melanie Subbiah*	
Jared Kaplan <sup>†</sup>	Prafulla Dhariwal	Arvind Neelakantan	Pranav Shyam	Girish Sastry
Amanda Askell	Sandhini Agarwal	Ariel Herbert-Voss	Gretchen Krueger	Tom Henighan
Rewon Child	Aditya Ramesh	Daniel M. Ziegler	Jeffrey Wu	Clemens Winter
Christopher Hesse	Mark Chen	Eric Sigler	Mateusz Litwin	Scott Gray
Benjamin Chess	Jack Clark	Christopher Berner		
Sam McCandlish	Alec Radford	Ilya Sutskever	Dario Amodei	

OpenAI



## Findings from OpenAI papers

- GPT-2 (2019) 1.5B parameters = 1,500 Million parameters
  - Trained on 2 billion words scraped from all over the internet
  - Scaling up models and training data continue improve the performance, and there does not seem to be a limit.
  - Zero-shot learning is an emerging capability. (Task prompts are not in the training set.)
- GPT-3 (2020) 200B parameters = 200,000 Million parameters
  - Scaling up models and training data continue improve the performance.
  - Few-shot learning is an emerging capability.

## Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 cheese => ..... ← prompt
```

---

## One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← example
3 cheese => ..... ← prompt
```

## Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French: ← task description
2 sea otter => loutre de mer ← examples
3 peppermint => menthe poivrée ←
4 plush girafe => girafe peluche ←
5 cheese => ..... ← prompt
```



# Emergent properties

Naturally occurring training data contain the demonstration pattern.

---

"I'm not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile** [I'm not a fool].

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: "**Mentez mentez, il en restera toujours quelque chose**," which translates as, "**Lie lie and something will always remain**."

"I hate the word '**perfume**,'" Burr says. 'It's somewhat better in French: '**parfum**.'

If listened carefully at 29:55, a conversation can be heard between two guys in French: "**-Comment on fait pour aller de l'autre coté? -Quel autre coté?**", which means "**- How do you get to the other side? - What side?**".

If this sounds like a bit of a stretch, consider this question in French: **As-tu aller au cinéma?**, or **Did you go to the movies?**, which literally translates as Have-you to go to movies/theater?

**"Brevet Sans Garantie Du Gouvernement"**, translated to English: **"Patented without government warranty"**.

---

*Table 1.* Examples of naturally occurring demonstrations of English to French and French to English translation found throughout the WebText training set.

A horizontal bar with a gold segment on the left and a red segment on the right.

## Breakthroughs from GPT-3

- Massive scale pre-training enable language models to be more than just a language model, which predicts the next word or computes probabilities of sentences.
- The more data the better. And there seems to be no limit here.

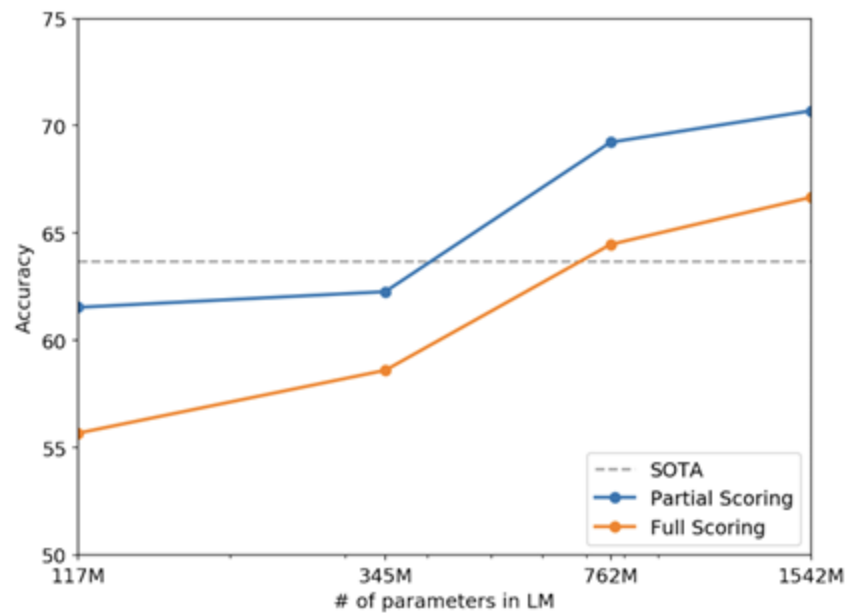


Figure 3. Performance on the Winograd Schema Challenge as a function of model capacity.

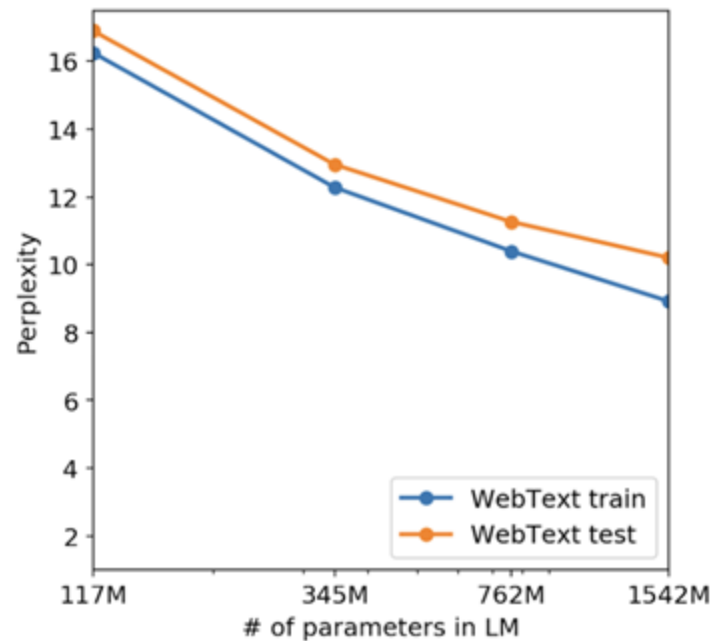


Figure 4. The performance of LMs trained on WebText as a function of model size.

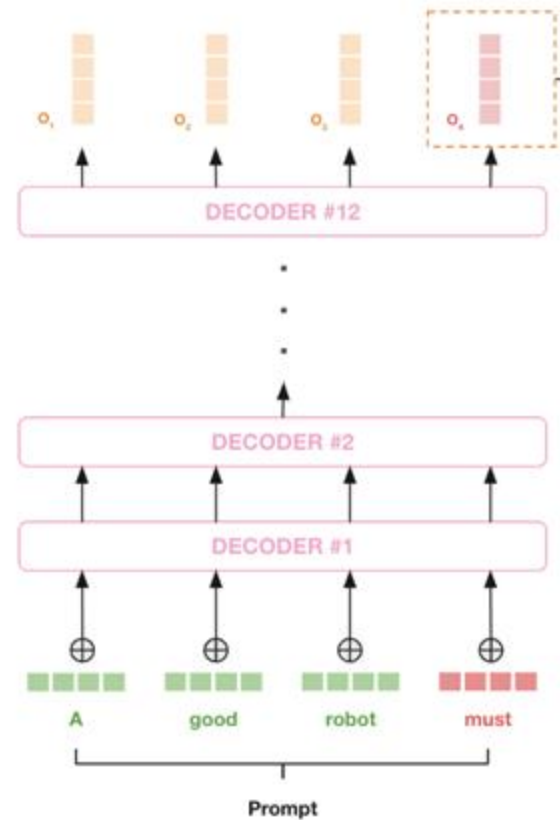
A horizontal bar with a gold segment on the left and a red segment on the right.

## History of neural language model

- 2003 - feedforward neural language model
- 2016 - recurrent neural language model (RNN-LM)
- 2017 - transformer-based language model
- 2019 - large transformer-based language model (e.g. GPT-2)
- 2022 - massive transform-based language model fine-tuned for use as a chatbot (ChatGPT)

# Large Language Model - ChatGPT (3.5)

- 175,000 M parameters
- 96 layers
- 12,288-dimensional embeddings
- 96 attention heads
- Context length of 2048 tokens



# Pretraining process and data

- Pre-training (a large language model) refers to training the model on the pretraining loss (next word prediction loss) usually on a massive amount of data.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

# Pre-trained LLMs are not good assistants at all

## Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:

- (A) They will discuss the reporter's favorite dishes
- (B) They will discuss the chef's favorite dishes
- (C) Ambiguous

A: Let's think step by step.

## PaLM 540B output

The reporter and the chef will discuss their favorite dishes.  
The reporter and the chef will discuss the reporter's favorite dishes.  
The reporter and the chef will discuss the chef's favorite dishes.  
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ (doesn't answer question)

## Model input (Snarks)

Q: Which statement is sarcastic?

Options:

- (A) Wow thats crazy, I've never had a takeout meal that used hardwood containers before!
- (B) Wow thats crazy, I've never had a takeout meal that used plastic containers before!

A: Let's think step by step.

## PaLM 540B output

What is the speaker trying to say?  
The speaker is trying to say that he has never had a takeout meal that used plastic containers before.  
What is the definition of sarcasm?  
the use of irony to mock or convey contempt.  
What is the definition of irony?  
the expression of one's meaning by using language that normally signifies the opposite [...]

✖ (never stops generating)

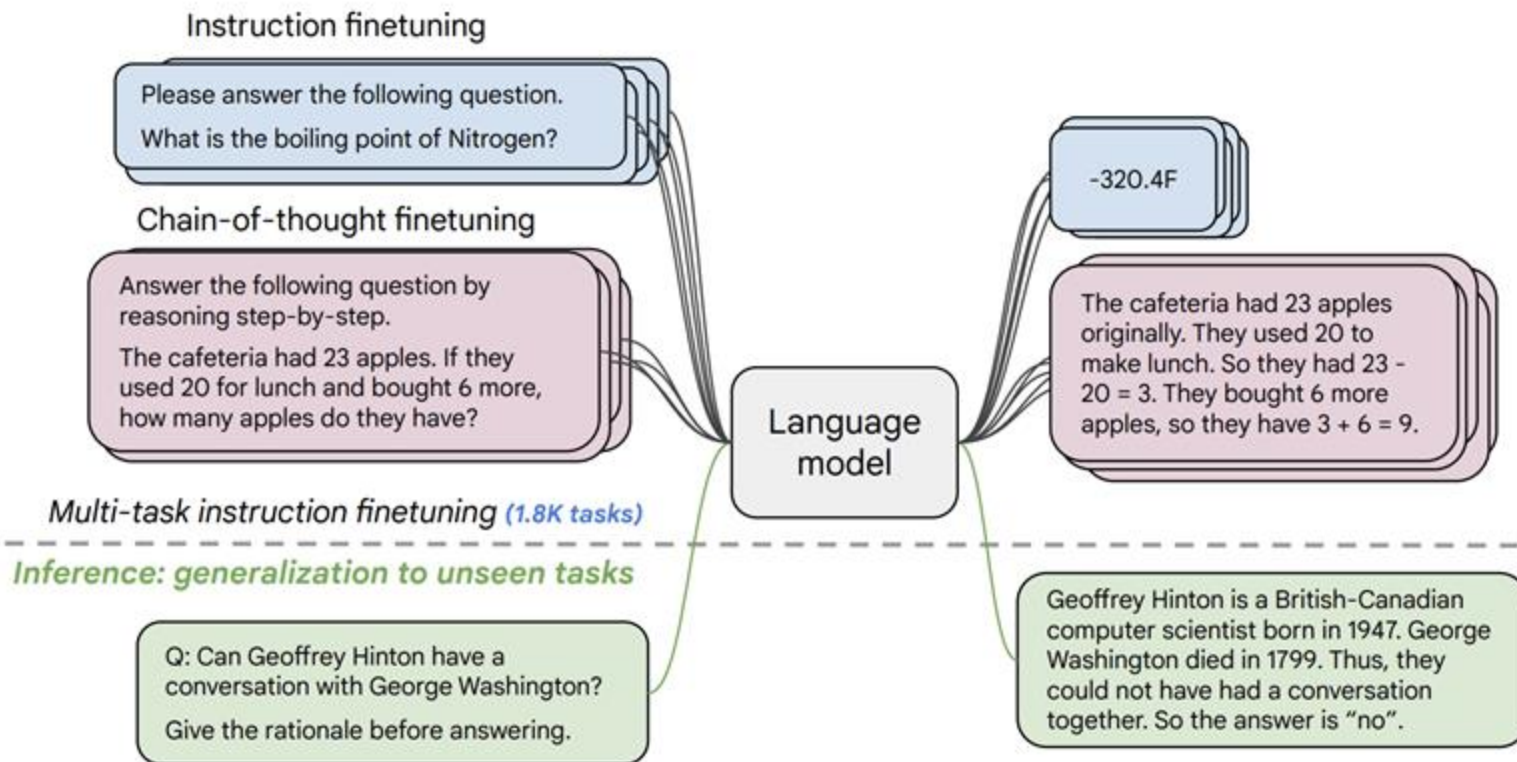
A horizontal bar with a gold segment on the left and a red segment on the right.

## Post-training

- LLM is not an assistant yet.
- Post-training is the process of training the LLMs to follow instructions and give the outputs that meet the expectations of the user
  - Instruction fine-tuning or supervised fine-tuning (SFT)
  - Alignment with human feedback



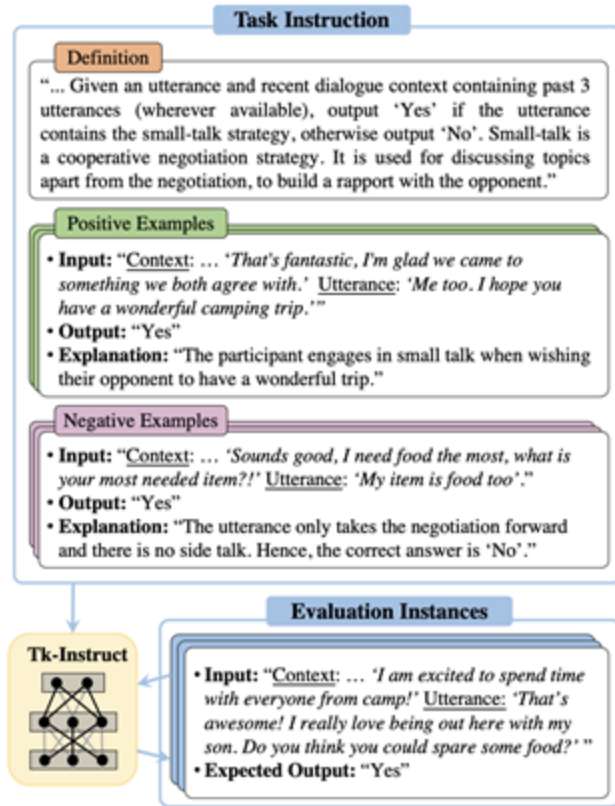
# Scaling up fine-tuning



# SuperNaturalInstructions (Wang et al. 2022)



(a) SUP-NATINST (this work)



### Step 1

**Collect demonstration data,  
and train a supervised policy.**

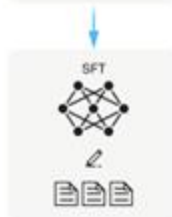
A prompt is  
sampled from our  
prompt dataset.



A labeler  
demonstrates the  
desired output  
behavior.



This data is used  
to fine-tune GPT-3  
with supervised  
learning.



### Step 2

**Collect comparison data,  
and train a reward model.**

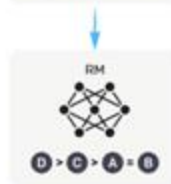
A prompt and  
several model  
outputs are  
sampled.



A labeler ranks  
the outputs from  
best to worst.



This data is used  
to train our  
reward model.



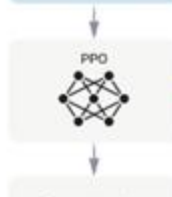
### Step 3

**Optimize a policy against  
the reward model using  
reinforcement learning.**

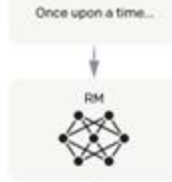
A new prompt  
is sampled from  
the dataset.



The policy  
generates  
an output.



The reward model  
calculates a  
reward for  
the output.

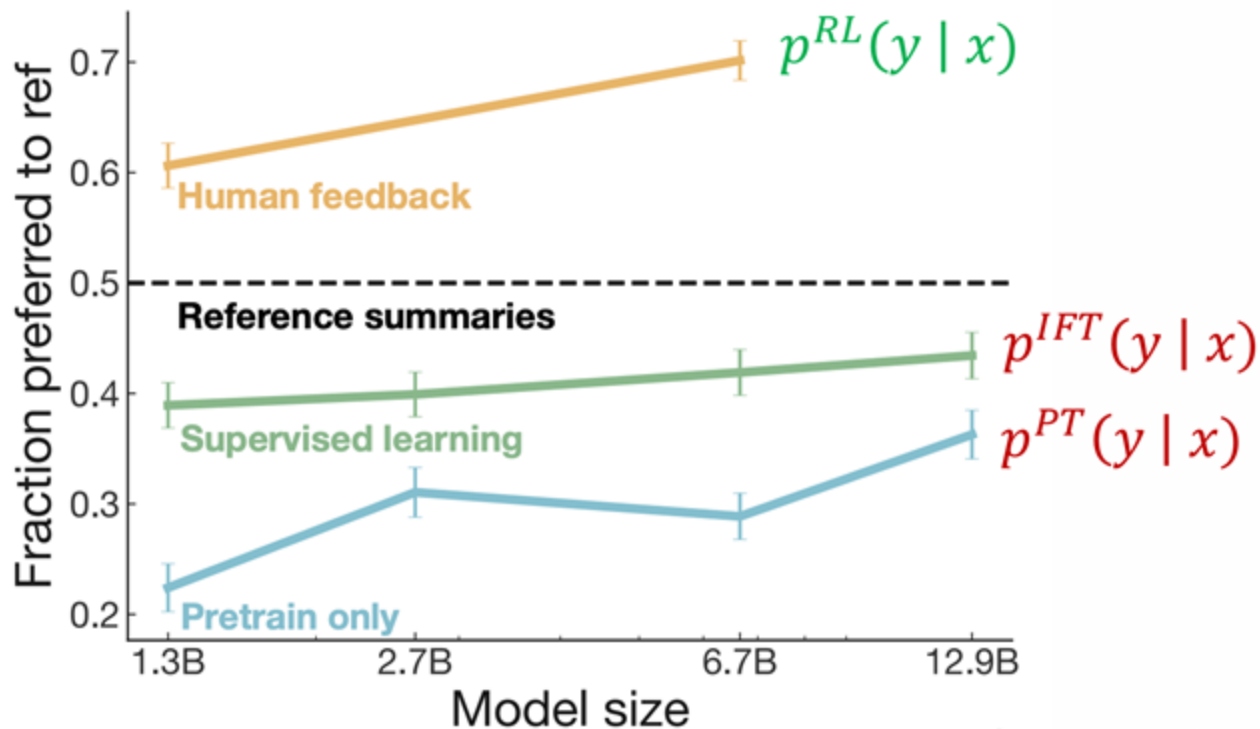


The reward is  
used to update  
the policy  
using PPO.



Reinforcement Learning with Human Feedback (Ouyang et al., 2022)

# RLHF improves pretrained and fine-tuned models



```
def analyze_sentiment(text):
    sentiment_analysis_prompt = f"""
    Analyze the sentiment of the text delimited by <text> and </text> tags.
    The sentiment must be one of the following: positive, negative, or neutral.

    {text}

    Output must be json with keys:
    |     reason: the reason for the sentiment
    |     sentiment: the sentiment label
    """
    messages = [
        {"role": "user", "content": sentiment_analysis_prompt}
    ]
    response = openai.chat.completions.create(
        model="gpt-4o-mini",
        messages=messages,
        response_format={ "type": "json_object" }
    )
    json_response = json.loads(response.choices[0].message.content)
    return json_response

sentiment = analyze_sentiment("บริษัทจะต้องเน้นการสร้างความเชื่อมั่นให้กับลูกค้าให้มากขึ้น")
```

```
def correct_grammatical_errors(text):  
    gec_prompt = f"""  
    Correct the grammatical errors in the text below. Output the corrected text  
  
    {text}  
    """  
    messages = [  
        {"role": "user", "content": gec_prompt}  
    ]  
    response = openai.chat.completions.create(  
        model="gpt-4o-mini",  
        messages=messages  
    )  
    return response.choices[0].message.content
```

```
corrected = correct_grammatical_errors("After meeting over, the staffs had to  
returned to work promptly. Other meeting attendees go to the airport.")  
print(corrected)
```

```
def academify(text):
    academify_prompt = f"""
    Act as a computer science professor.
```

First, turn the paragraph below into a paragraph for an academic journal manuscript. The language must be academic English. Use strong verbs. Use we. Add sentences as necessary to improve logical flow and clarity.

Second, critique the logical flow of the generated paragraph. And rewrite or add sentences to improve logical flow

The output must be in JSON with keys:

```
    'academic_text'
    'critique'
    'improved_text'
```

```
{text}
```

```
"""
```

```
    messages = [
        {"role": "user", "content": academify_prompt}
    ]
    response = openai.chat.completions.create(
        model="gpt-4o-mini",
        messages=messages,
        response_format={ "type": "json_object" }
    )
    json_response = json.loads(response.choices[0].message.content)
    return json_response
```

```
text = """
```

Large language model is an NLP model that can do many many tasks. Examples are grammatical error correction, question answering, and summarization. We don't have to train it on each task. They can do these tasks quite well. But they are not perfect.

```
"""
```

```
response = academify(text)
```



# LLM have internal knowledge and skills

Before:

*Large language model is an NLP model that can do many many tasks. Examples are grammatical error correction, question answering, and summarization. We don't have to train it on each task. They can do these tasks quite well. But they are not perfect.*

After:

*Large language models (LLMs) represent a significant advancement in natural language processing (NLP), displaying remarkable proficiency in a diverse array of tasks, including but not limited to grammatical error correction, question answering, and text summarization. These models operate on a principle of generalization that allows them to perform effectively without the need for exhaustive training on each specific task, thereby streamlining the deployment process across various applications. Moreover, the versatility of LLMs enables them to adapt to different contexts, making them invaluable tools for practitioners in the field. However, it is crucial to acknowledge that, despite their impressive capabilities, LLMs are not infallible; they may face limitations in accuracy, contextual understanding, and the generation of coherent results. Overall, this balance between their broad applicability and inherent shortcomings underscores the complexity of leveraging LLMs in practical NLP tasks.*



# Conclusion - Neural Language Model

- Neural language models are powerful tools for processing natural language and have achieved state-of-the-art results on a wide range of NLP tasks.
- Feedforward neural language models are a basic type of language model that can model dependencies between adjacent words in a sentence, but do not capture long-term dependencies.
- Recurrent neural network language models (RNN-LMs) are a type of language model that can capture long-term dependencies between words.
- Transformers are a newer type of neural language model that use a self-attention mechanism to model dependencies between all tokens in a sequence, allowing for modeling of long-term dependencies.
- Large Language Models are transformer-based language models that are pretrained and fine-tuned to be more than just a language model.