CSCE 638 Natural Language Processing Foundation and Techniques

Lecture 7: Transformers

Kuan-Hao Huang

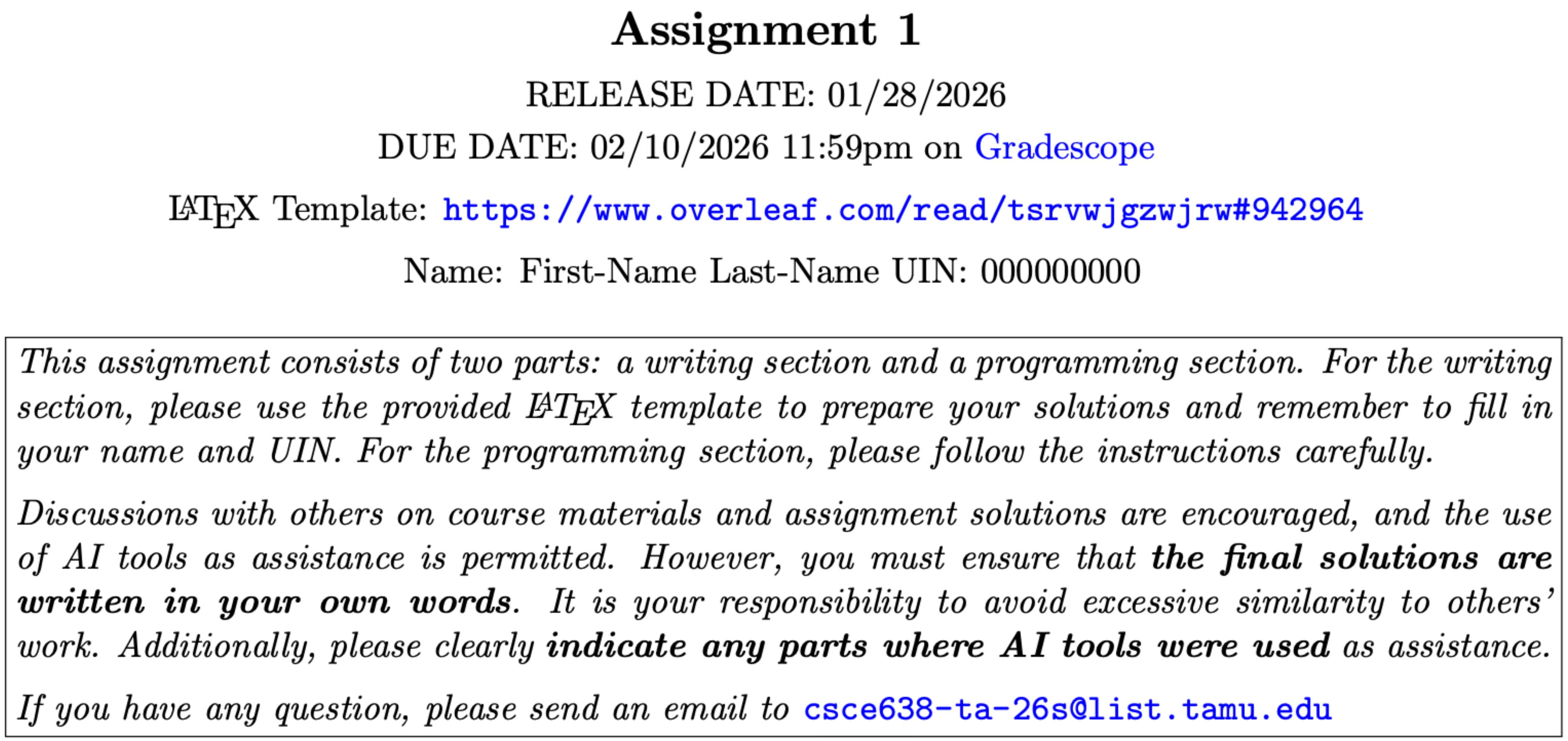
Spring 2026

(Some slides adapted from Chris Manning, Karthik Narasimhan, Danqi Chen, and Vivian Chen)

Assignment 0

- Grade was posted
- Contact csce638-ta-26s@list.tamu.edu if you have questions

Assignment 1



Assignment 1

RELEASE DATE: 01/28/2026

DUE DATE: 02/10/2026 11:59pm on Gradescope

LaTeX Template: `https://www.overleaf.com/read/tsrvwjgzwjrw#942964`

Name: First-Name Last-Name UIN: 000000000

This assignment consists of two parts: a writing section and a programming section. For the writing section, please use the provided LaTeX template to prepare your solutions and remember to fill in your name and UIN. For the programming section, please follow the instructions carefully.

Discussions with others on course materials and assignment solutions are encouraged, and the use of AI tools as assistance is permitted. However, you must ensure that ***the final solutions are written in your own words****. It is your responsibility to avoid excessive similarity to others' work. Additionally, please clearly* ***indicate any parts where AI tools were used*** *as assistance.*

If you have any question, please send an email to `csce638-ta-26s@list.tamu.edu`

Clarification on Participation Grading

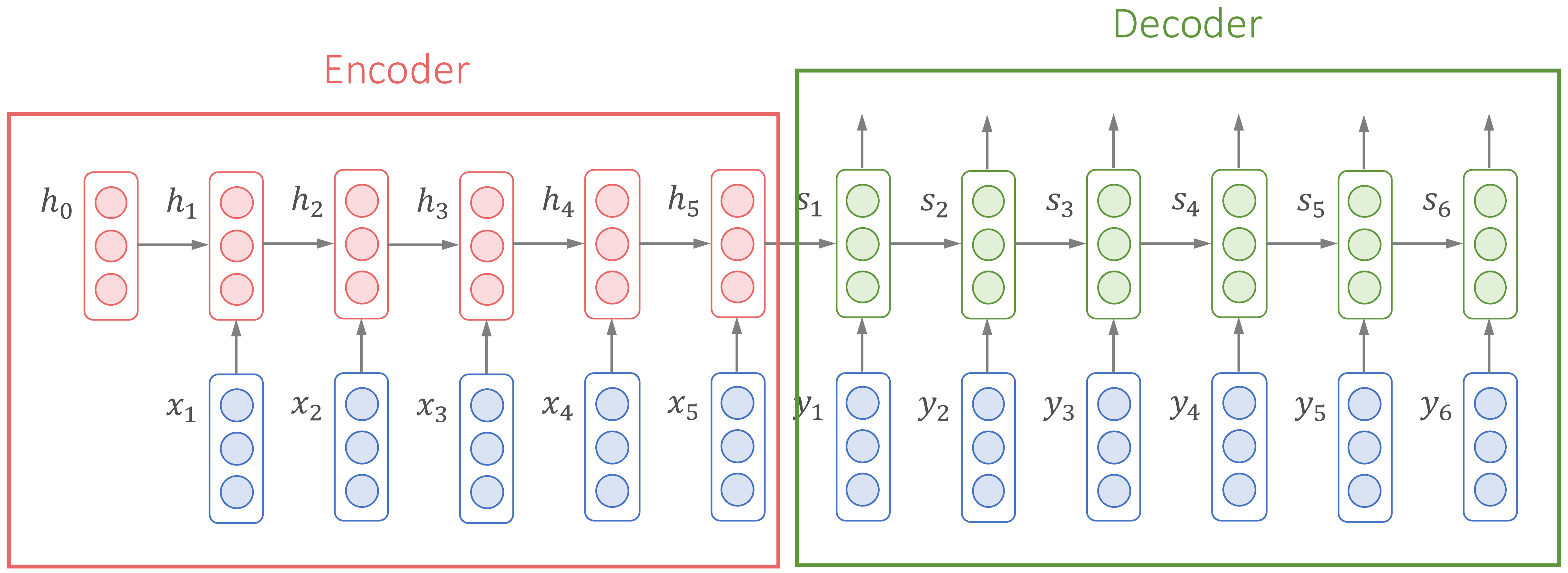
- Grade (100%)
  - Assignments (31%)
  - Quizzes (40%)
  - Course Project (26%)
  - Participation (3%)

Lecture Plan

- Transformers
  - Attention
  - Self-Attention
  - Transformer Encoder
  - Positional Encoding

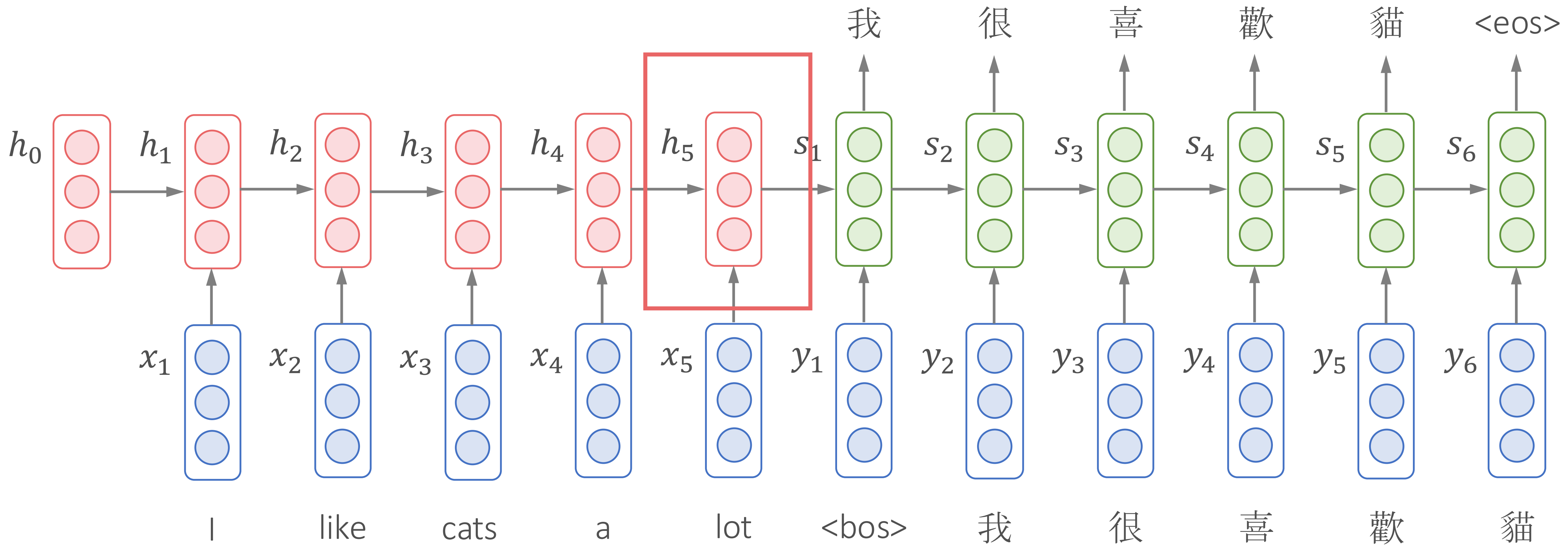
Recap: Sequence-to-Sequence Models (Seq2Seq)

- When we need understanding and generation at the same time



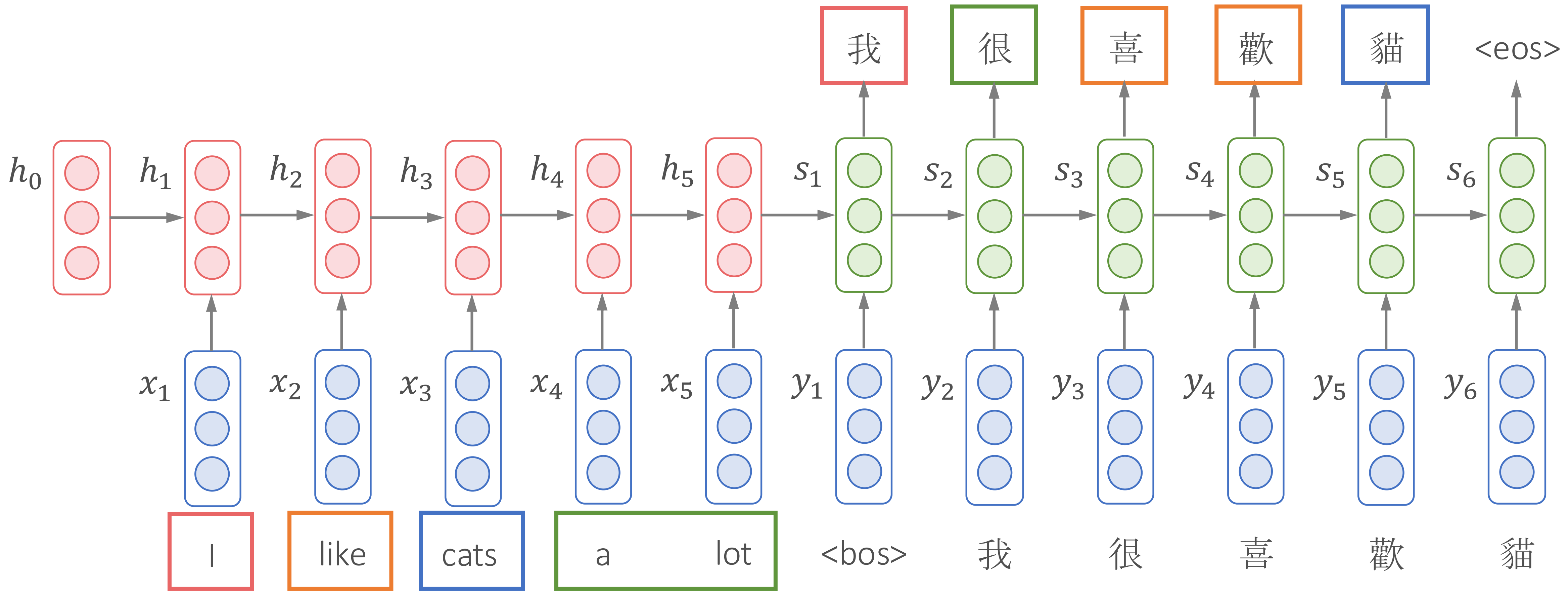
Recap: Seq2Seq: Bottleneck

- A single vector needs to capture all the information about source sentence
- Longer sequences can still lead to vanishing gradients

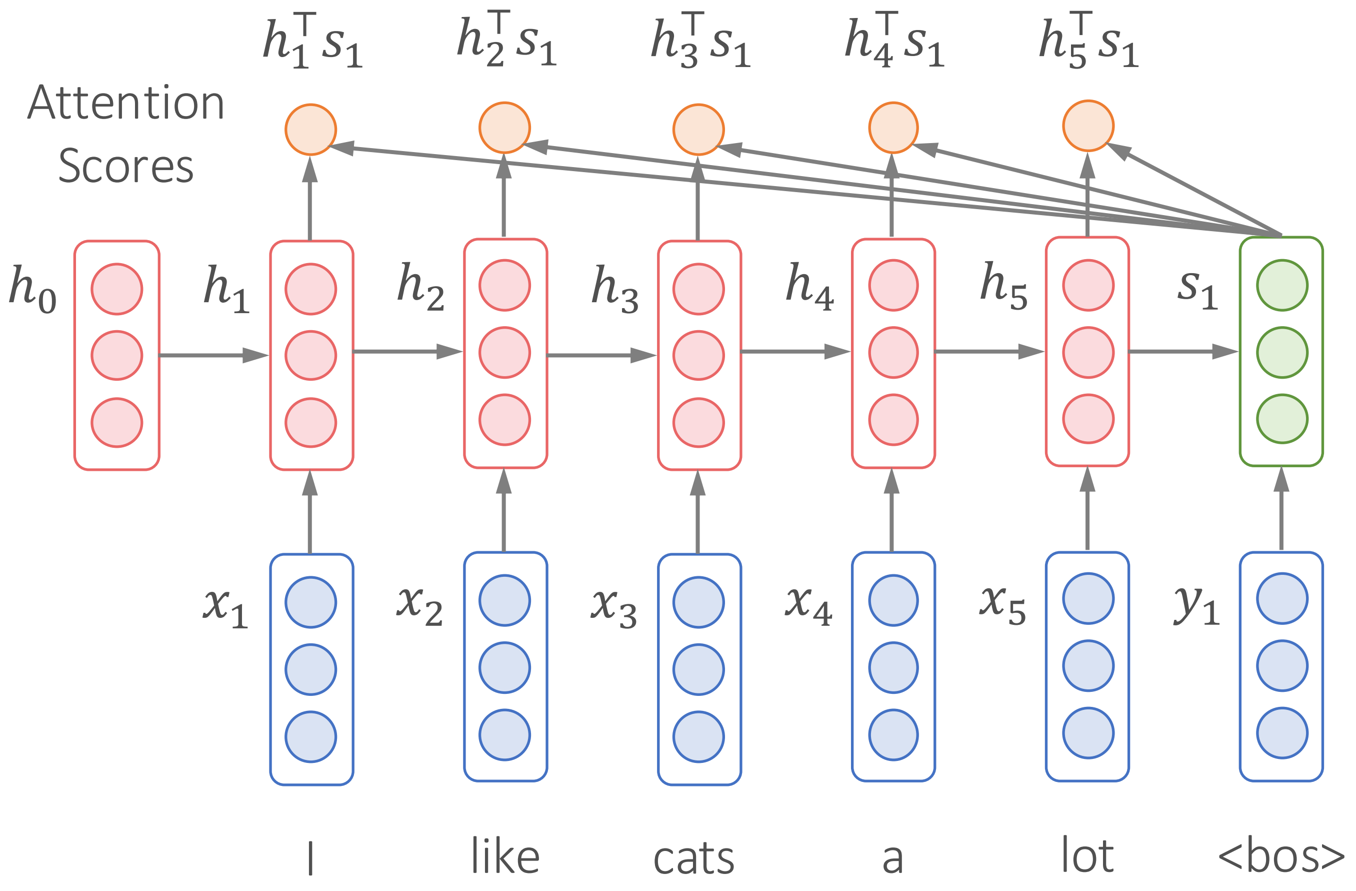


Recap: Focus on A Particular Part When Decoding

- Each token classification requires different part of information from source sentence

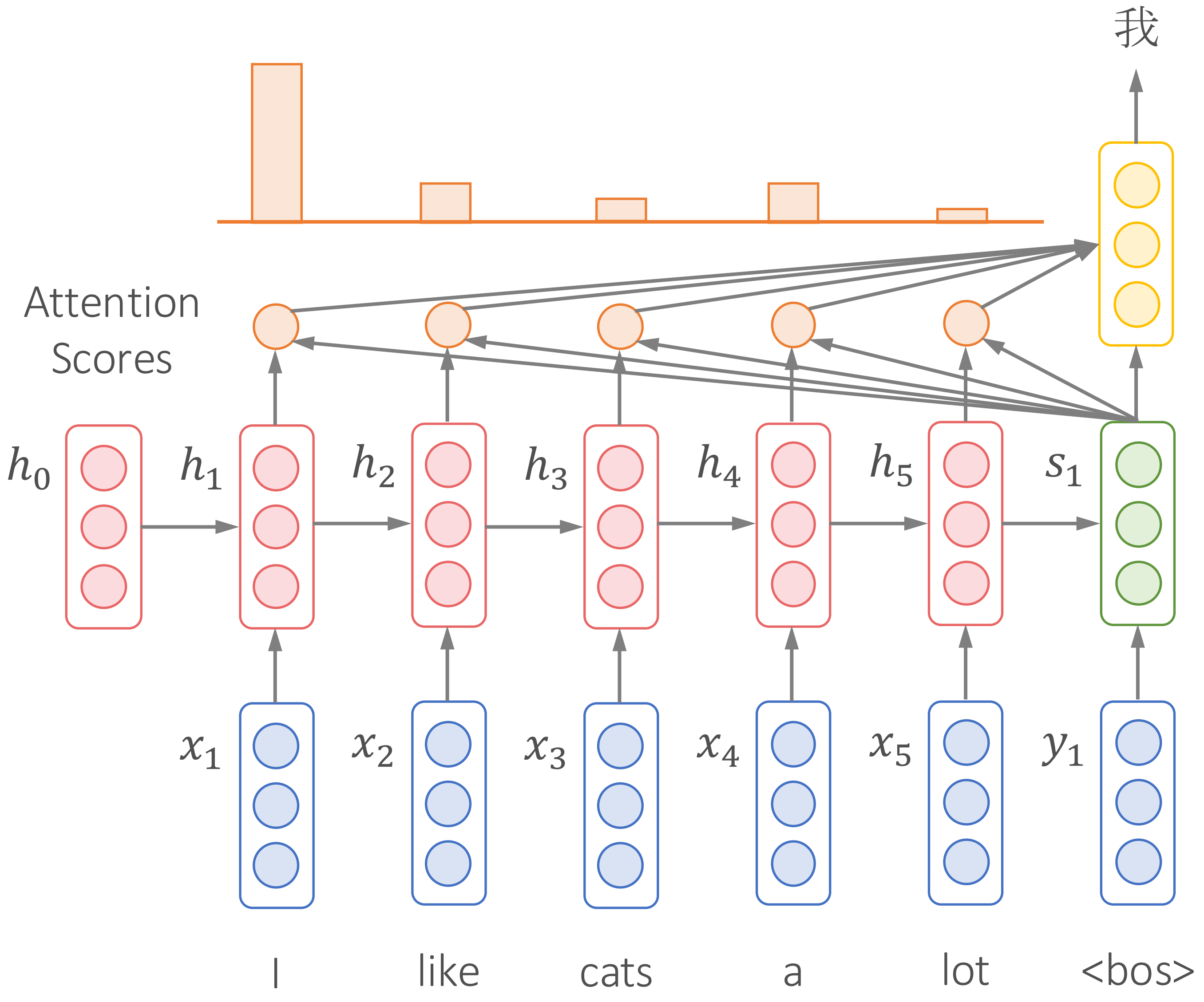


Recap: RNN with Attention



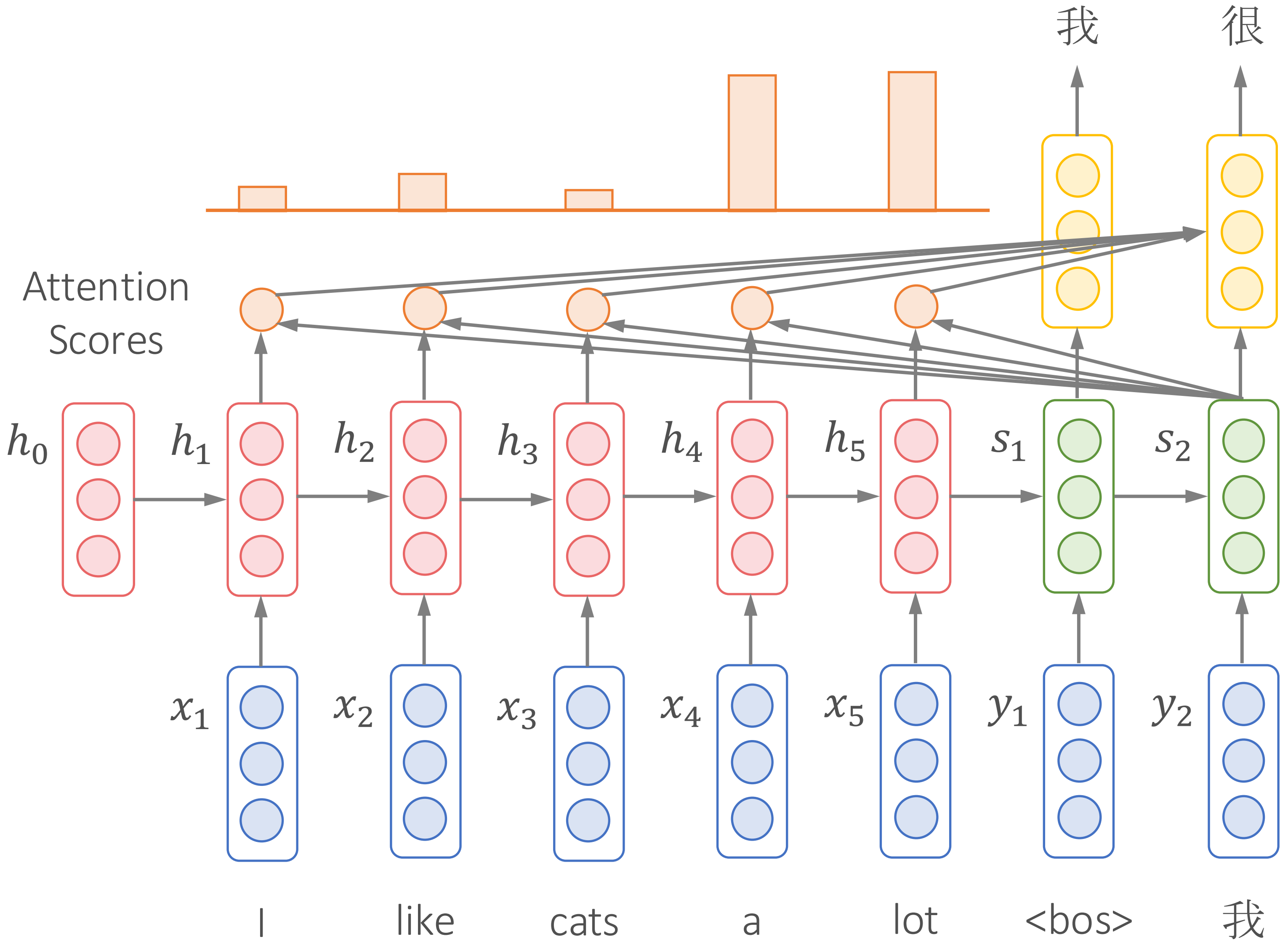
Attention Scores $\alpha_i = h_i^\top s_1$

Recap: RNN with Attention

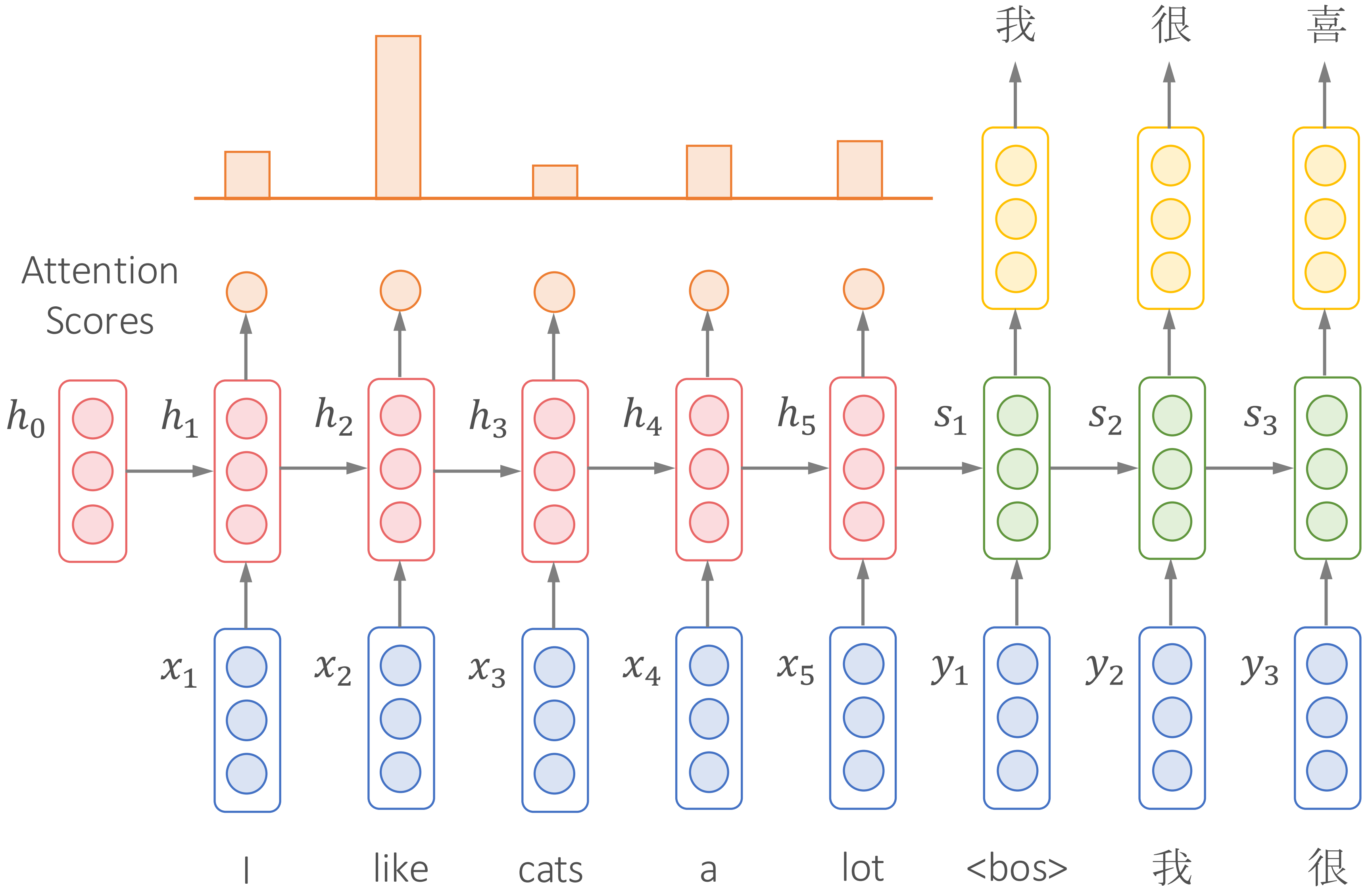


Attention Scores	$\alpha_i = h_i^{\top} s_1$
Normalized Attention Scores	$\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$
Weighted Sum	$a = \sum_i \hat{\alpha}_i h_i$
Attention Output	$\tanh(\mathbf{W}[a; s_1])$

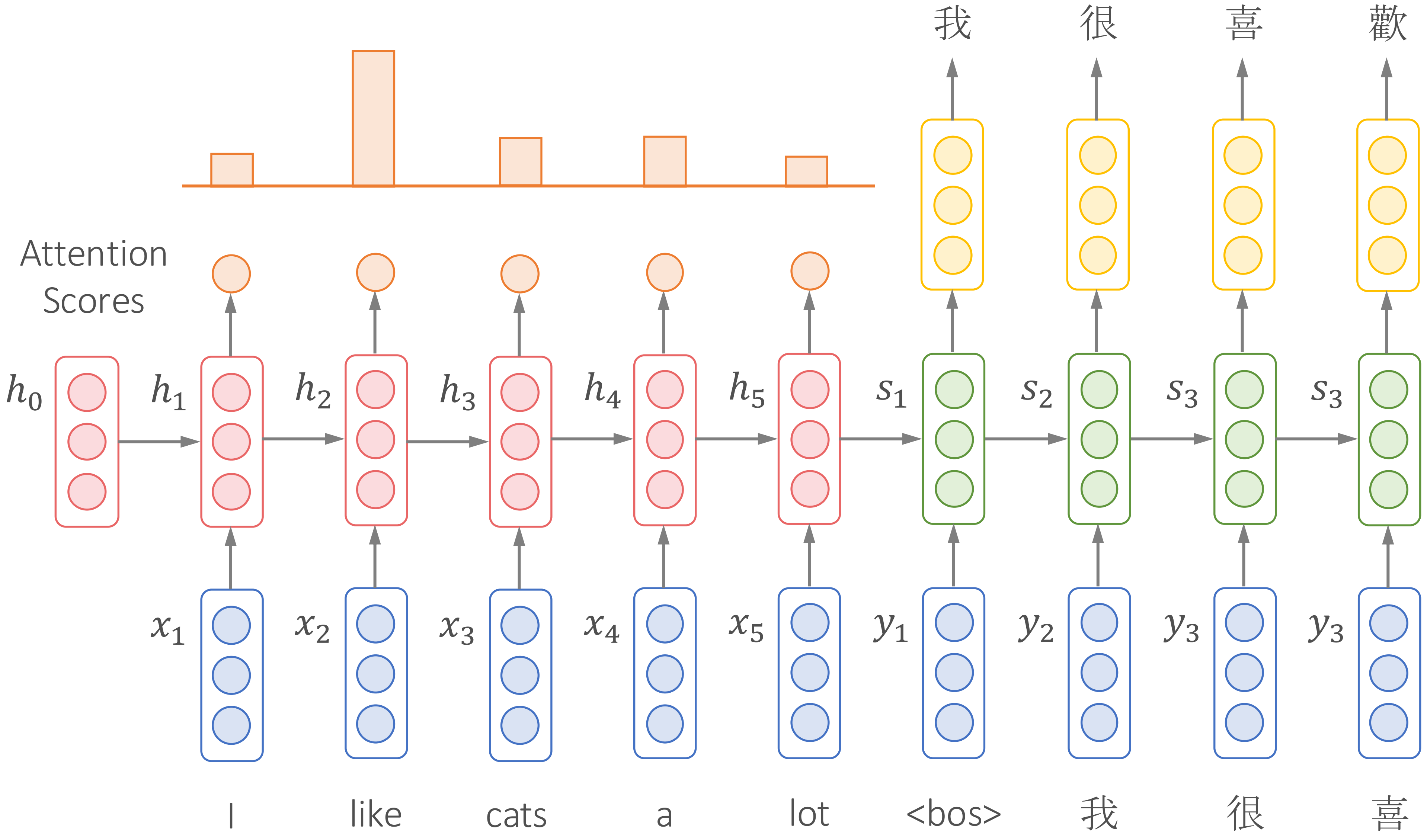
Recap: RNN with Attention



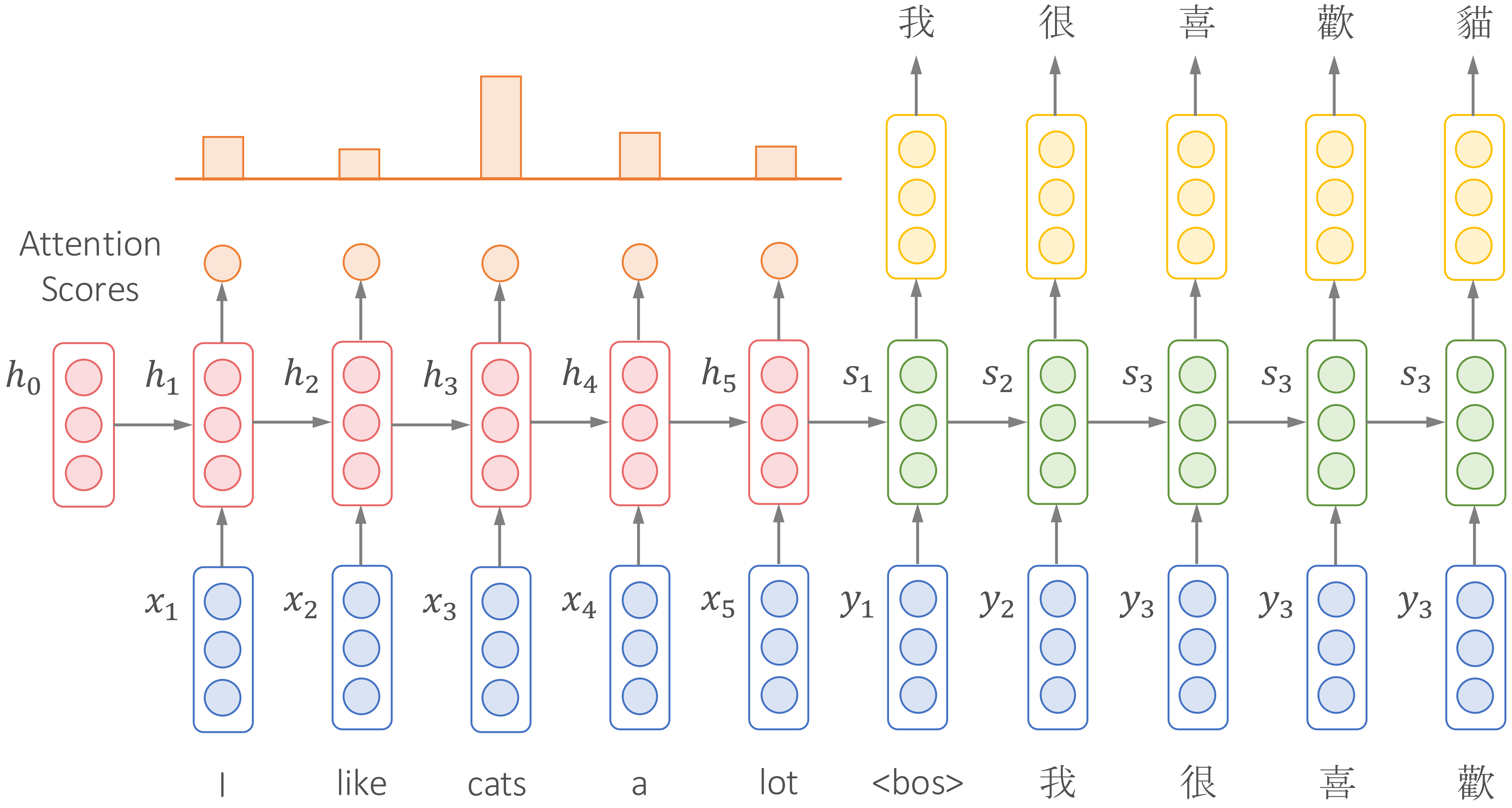
Recap: RNN with Attention



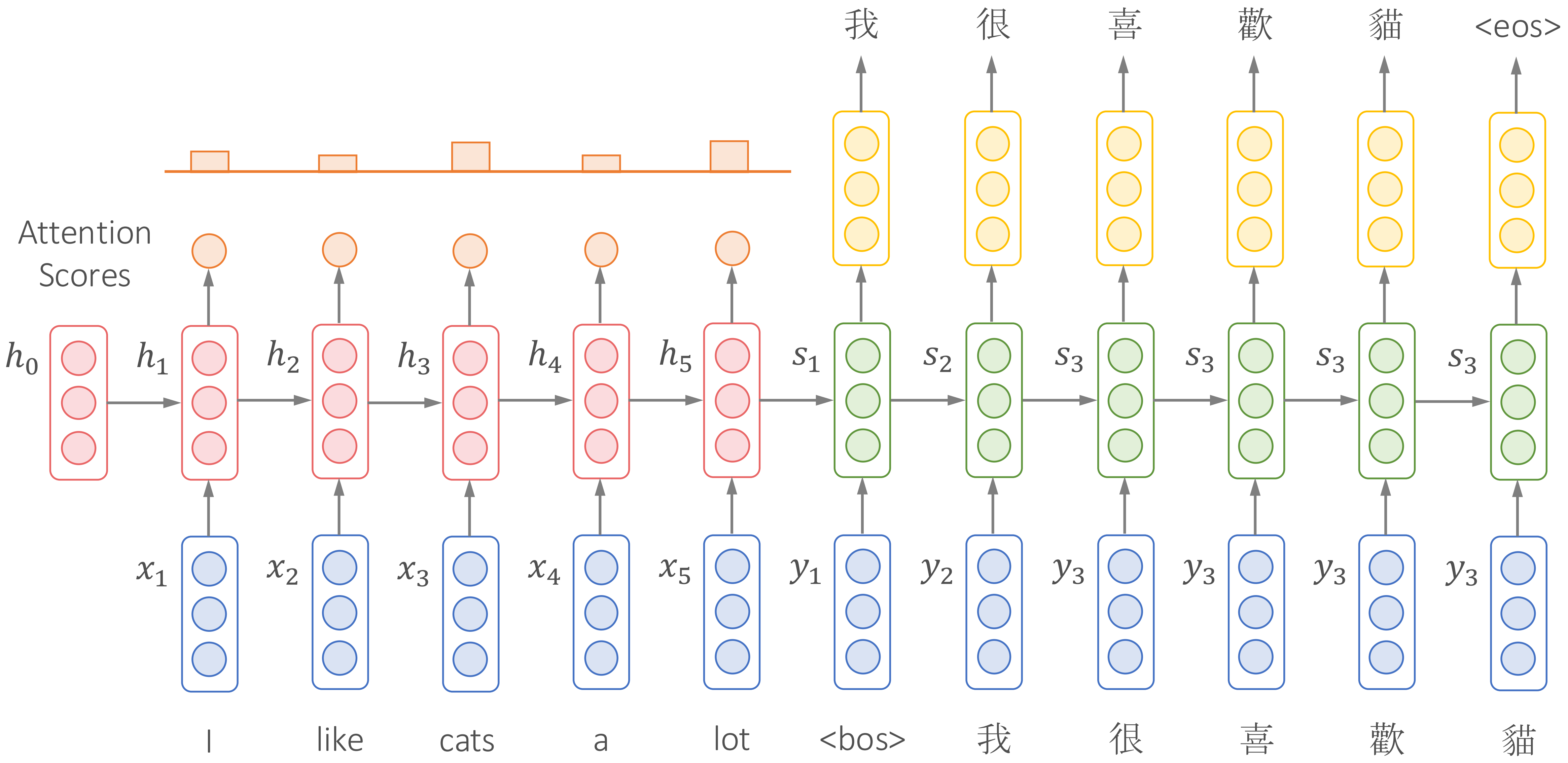
Recap: RNN with Attention



Recap: RNN with Attention



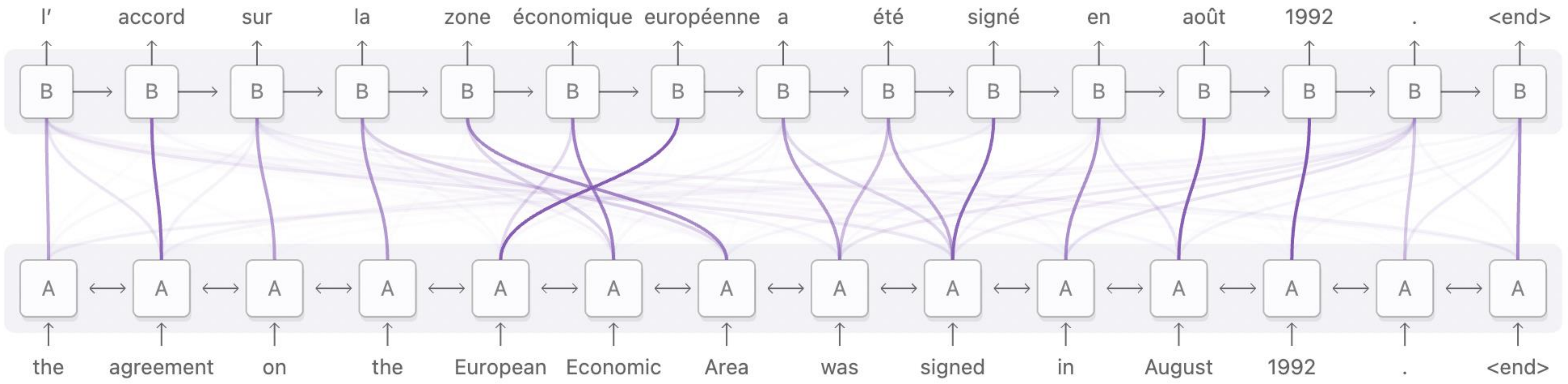
Recap: RNN with Attention



Different Types of Attention

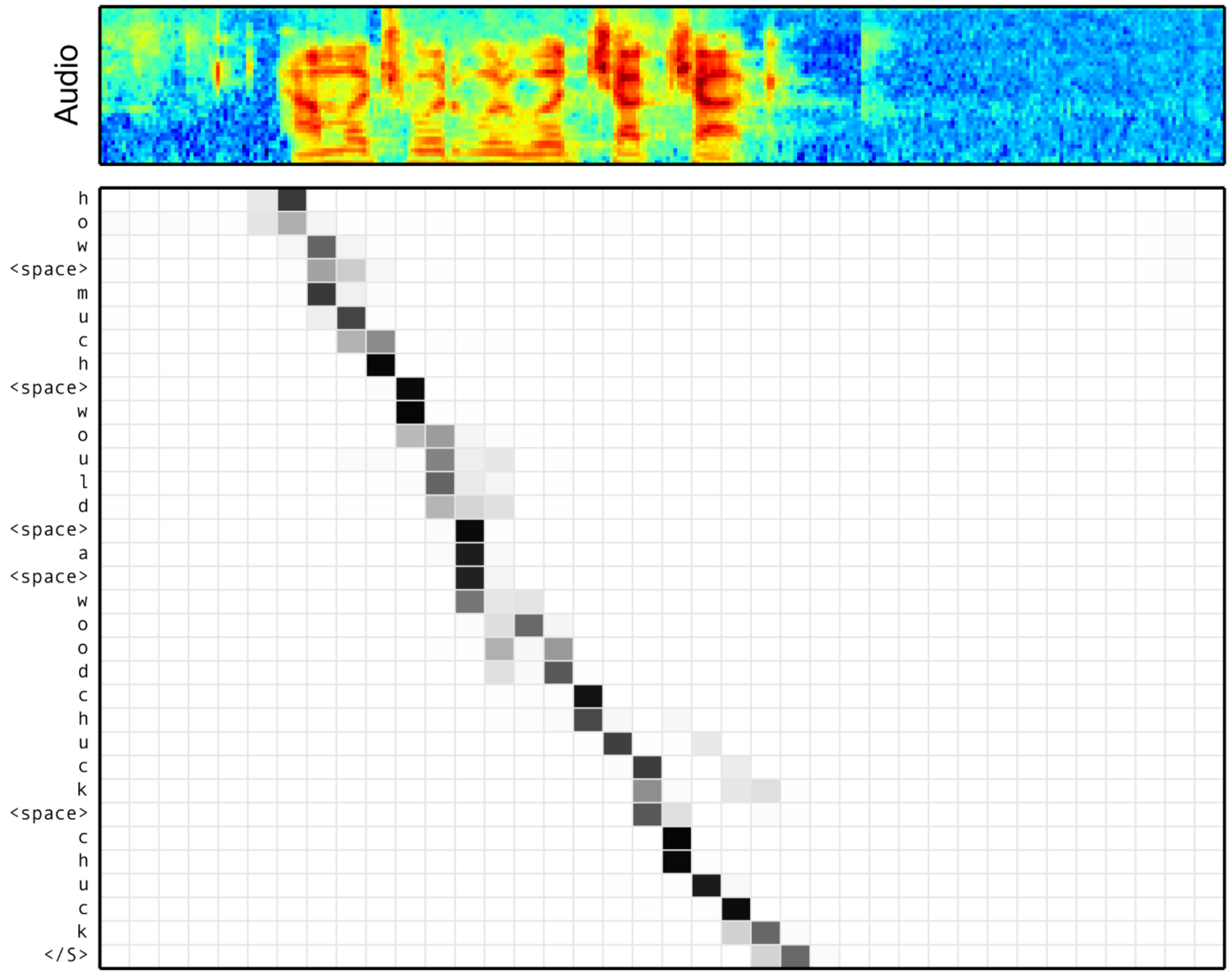
Dot-Product Attention	$h_i^\top s_j$
Multiplicative Attention	$h_i^\top W s_j$
Additive Attention	$v^\top \tanh(W_1 h_i + W_2 s_j)$

Machine Translation with Attention



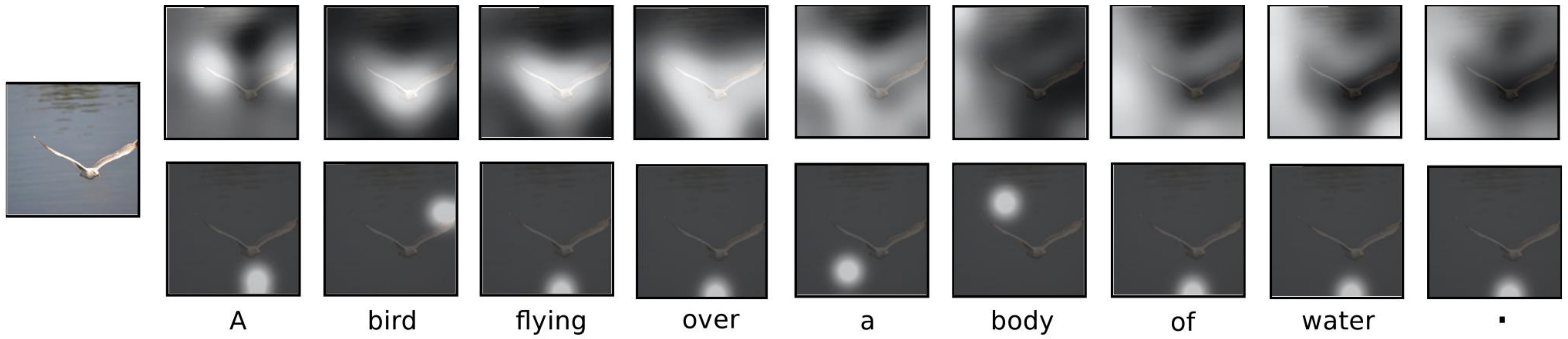
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

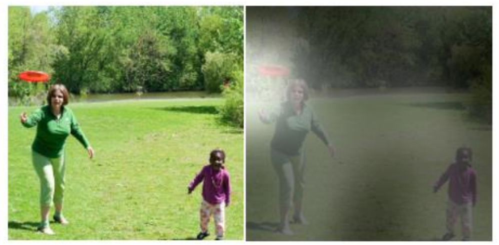
Speech Recognition with Attention

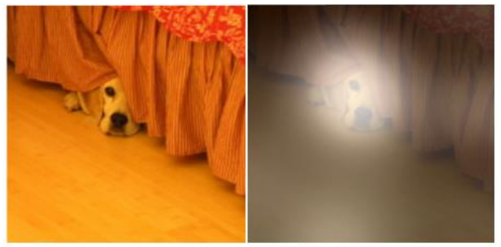


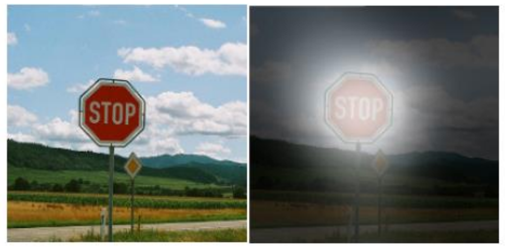
Listen, Attend and Spell, 2015

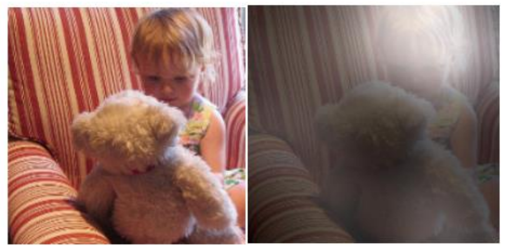
Image Captioning with Attention

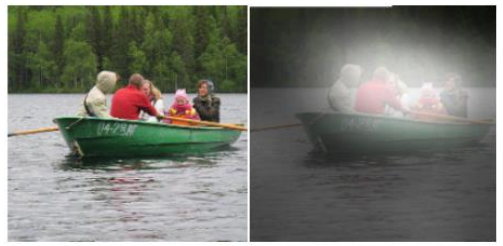


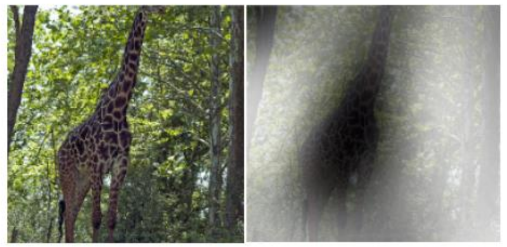

A woman is throwing a frisbee in a park.


A dog is standing on a hardwood floor.


A stop sign is on a road with a mountain in the background.

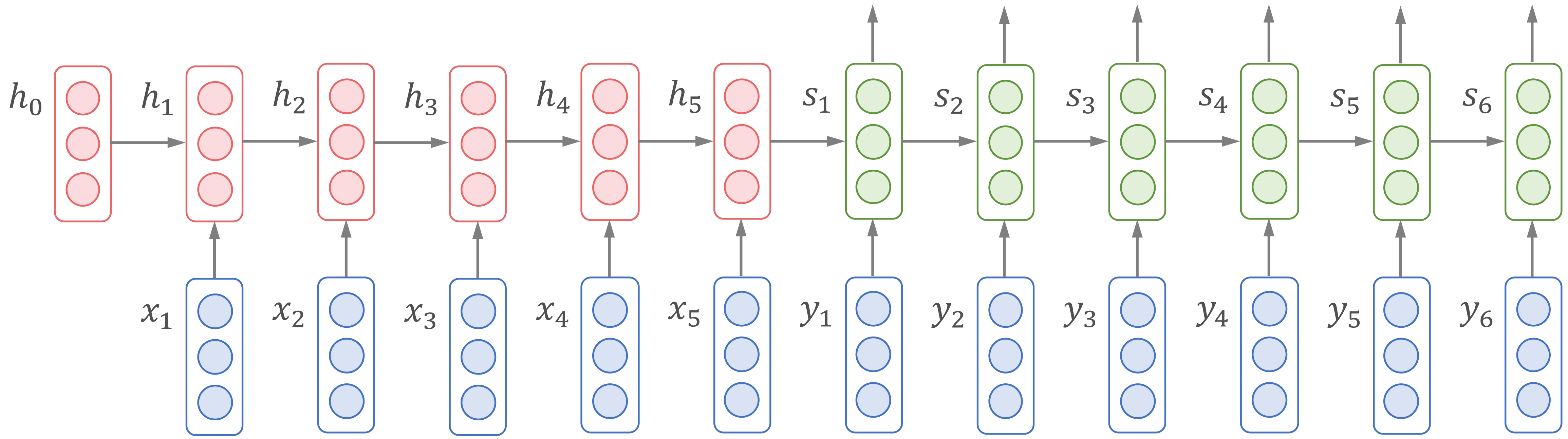

A little girl sitting on a bed with a teddy bear.


A group of people sitting on a boat in the water.

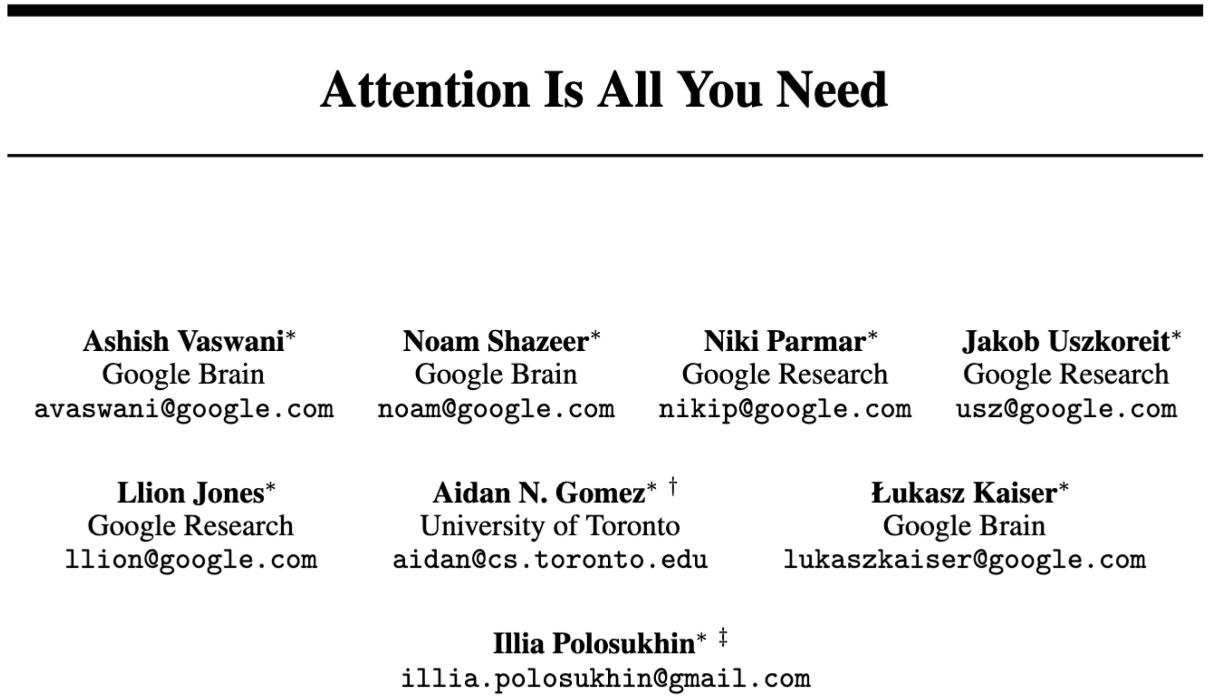

A giraffe standing in a forest with trees in the background.

Issues with RNN

- Longer sequences can lead to vanishing gradients → It is hard to capture long-distance information
- Lack parallelizability

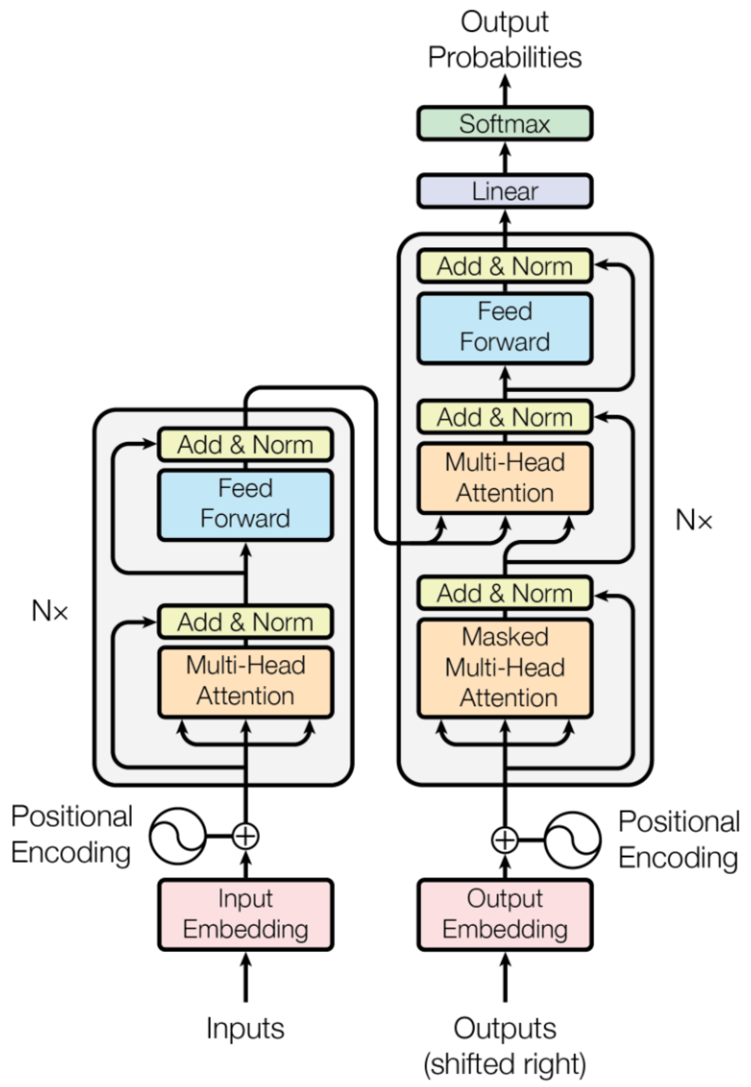


Transformers: Attention Is All You Need!



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



Attention Is All You Need, 2017

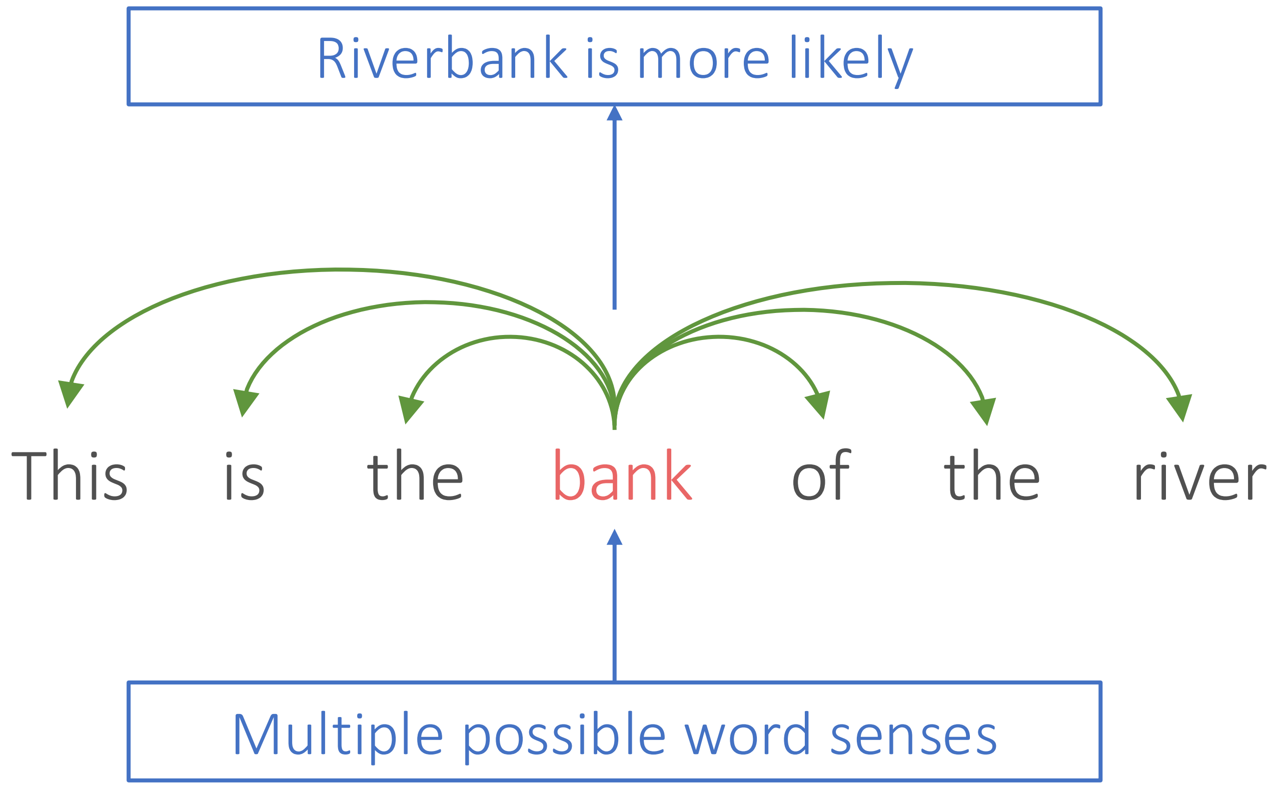
Context-Level Understanding

I went to the bank.

The dog ran to the bank of the river.

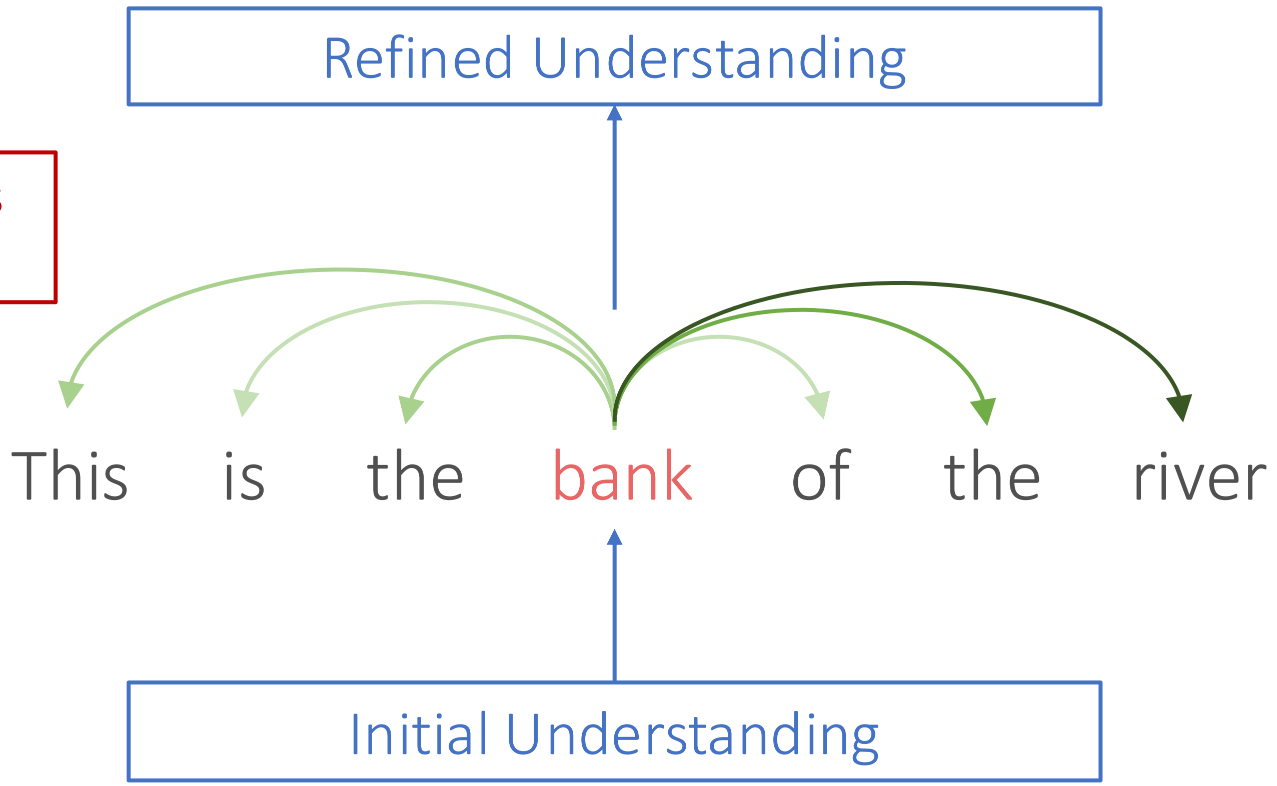
The plane began to bank.

How Do Human Decide Meaning?

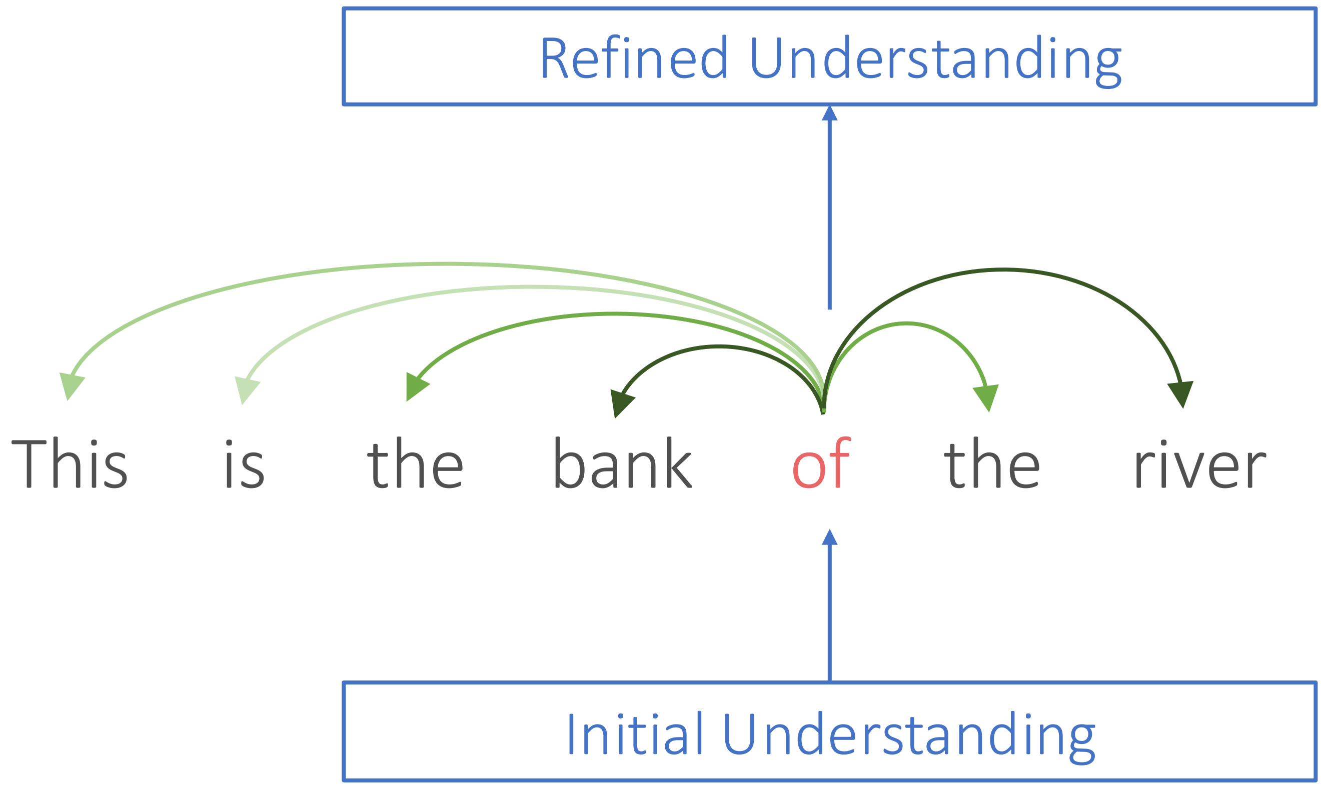


How Do Human Decide Meaning?

Different context words have different weights

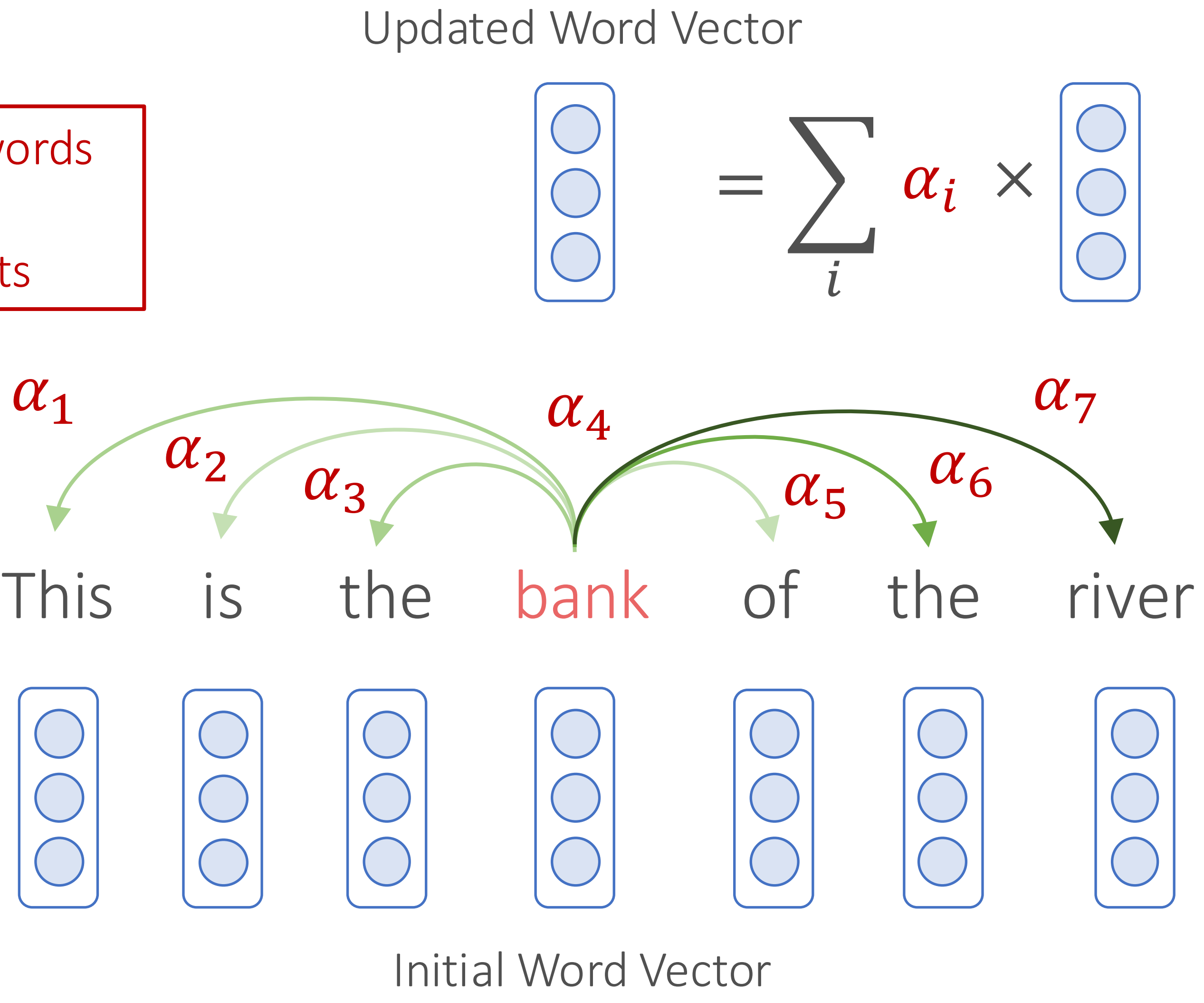


How Do Human Decide Meaning?



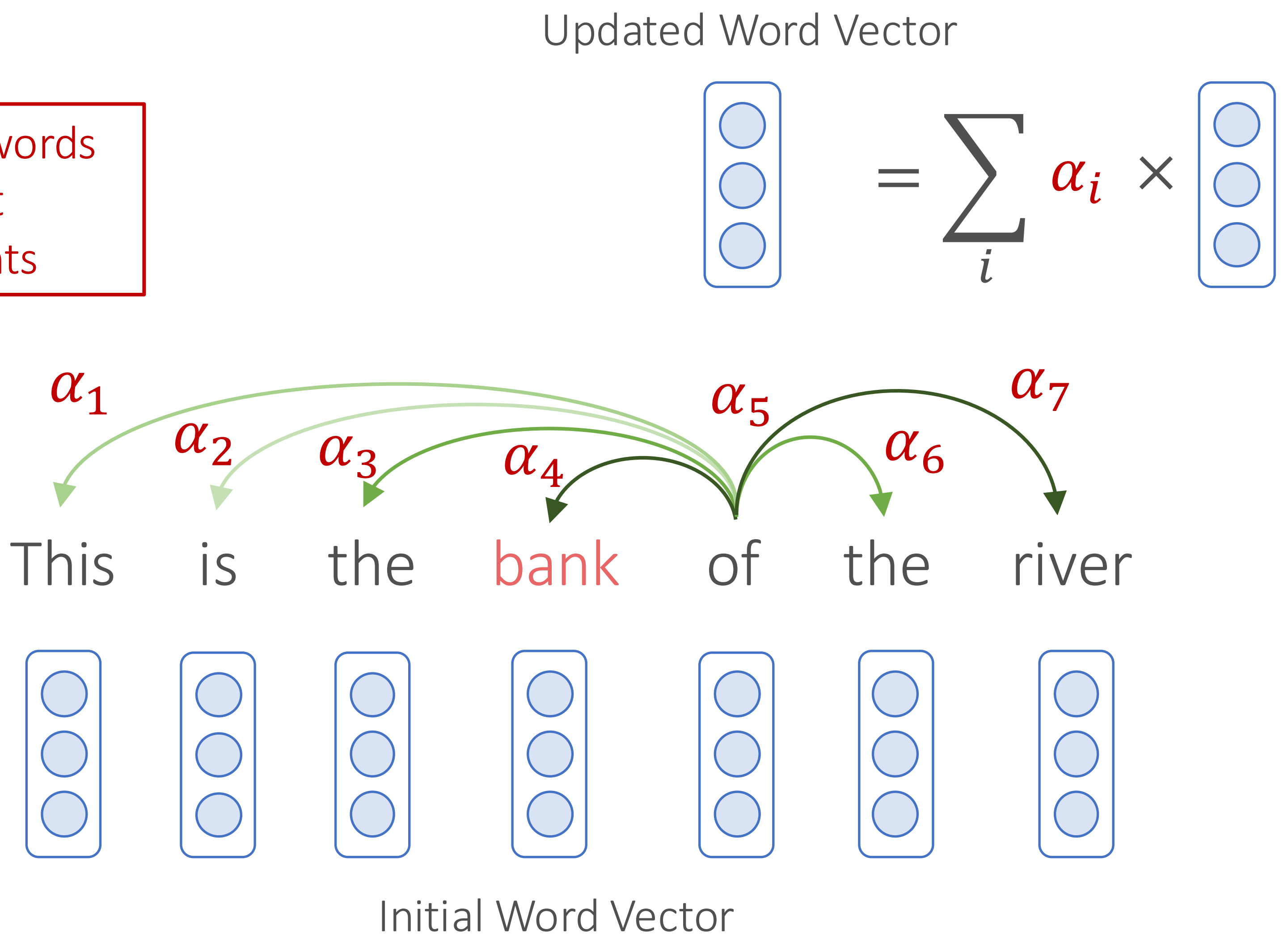
How Does Transformer Mimic Human Behavior?

Different context words have different **attention** weights

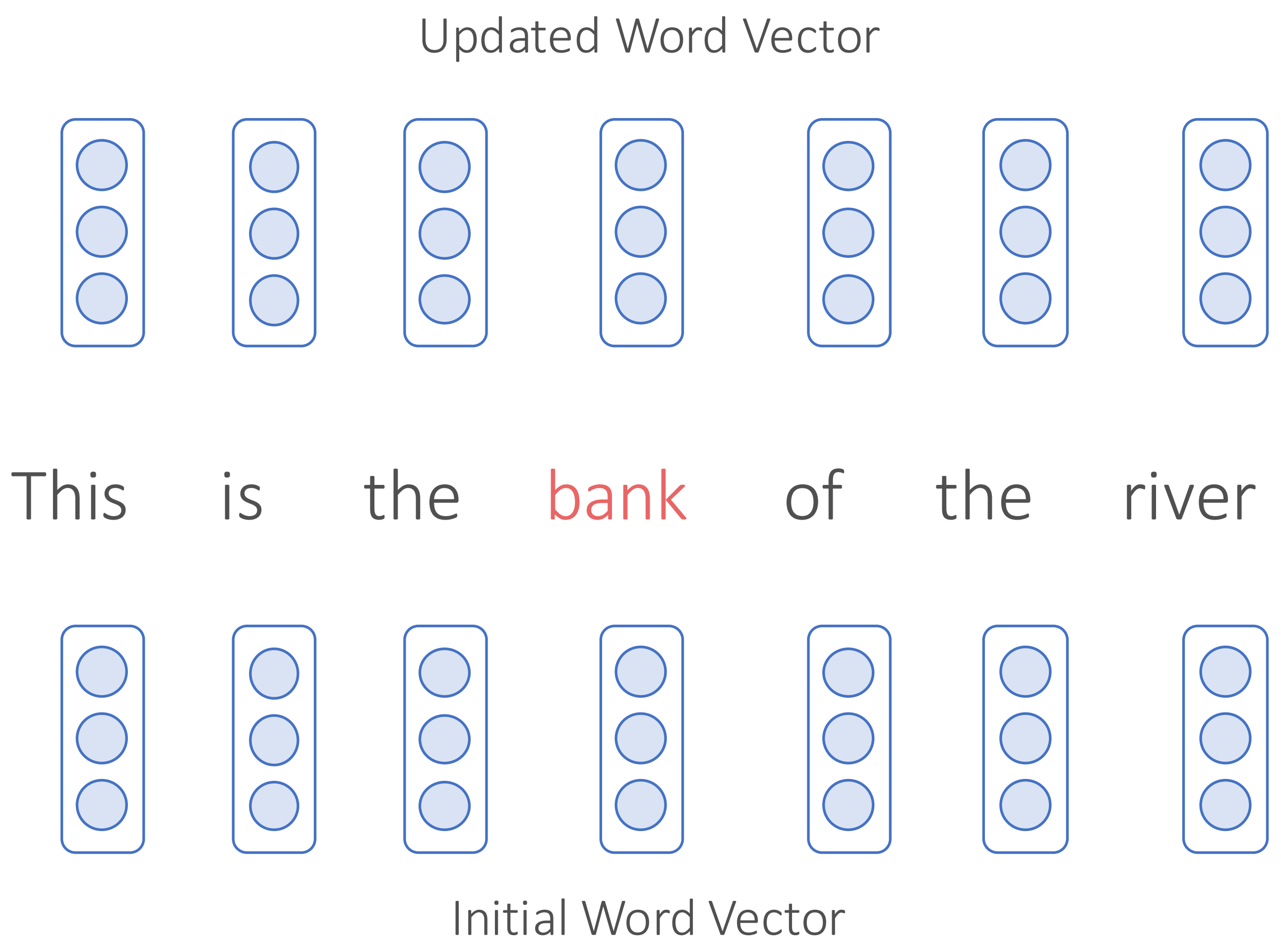


How Does Transformer Mimic Human Behavior?

Different context words have different **attention** weights

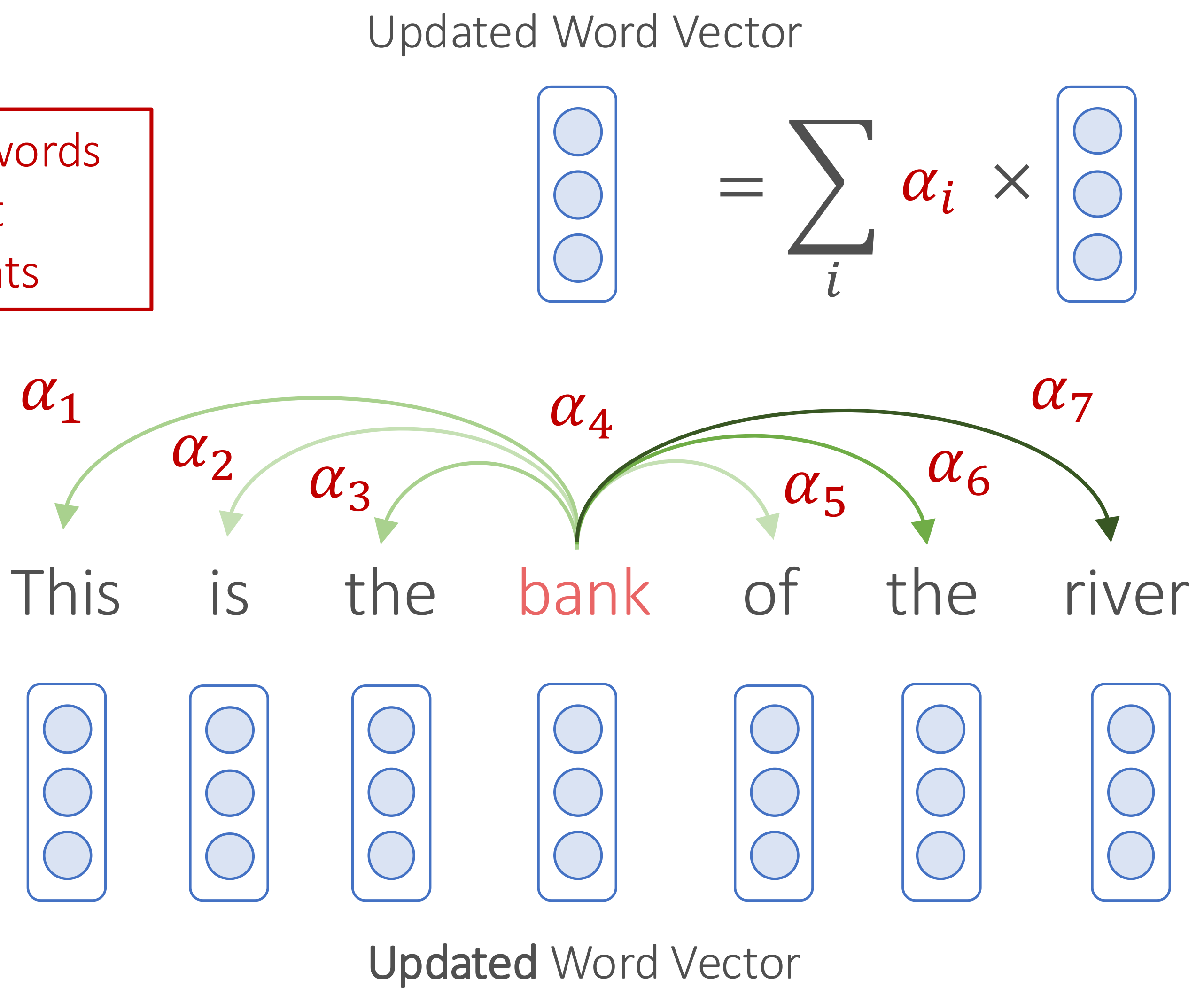


How Does Transformer Mimic Human Behavior?

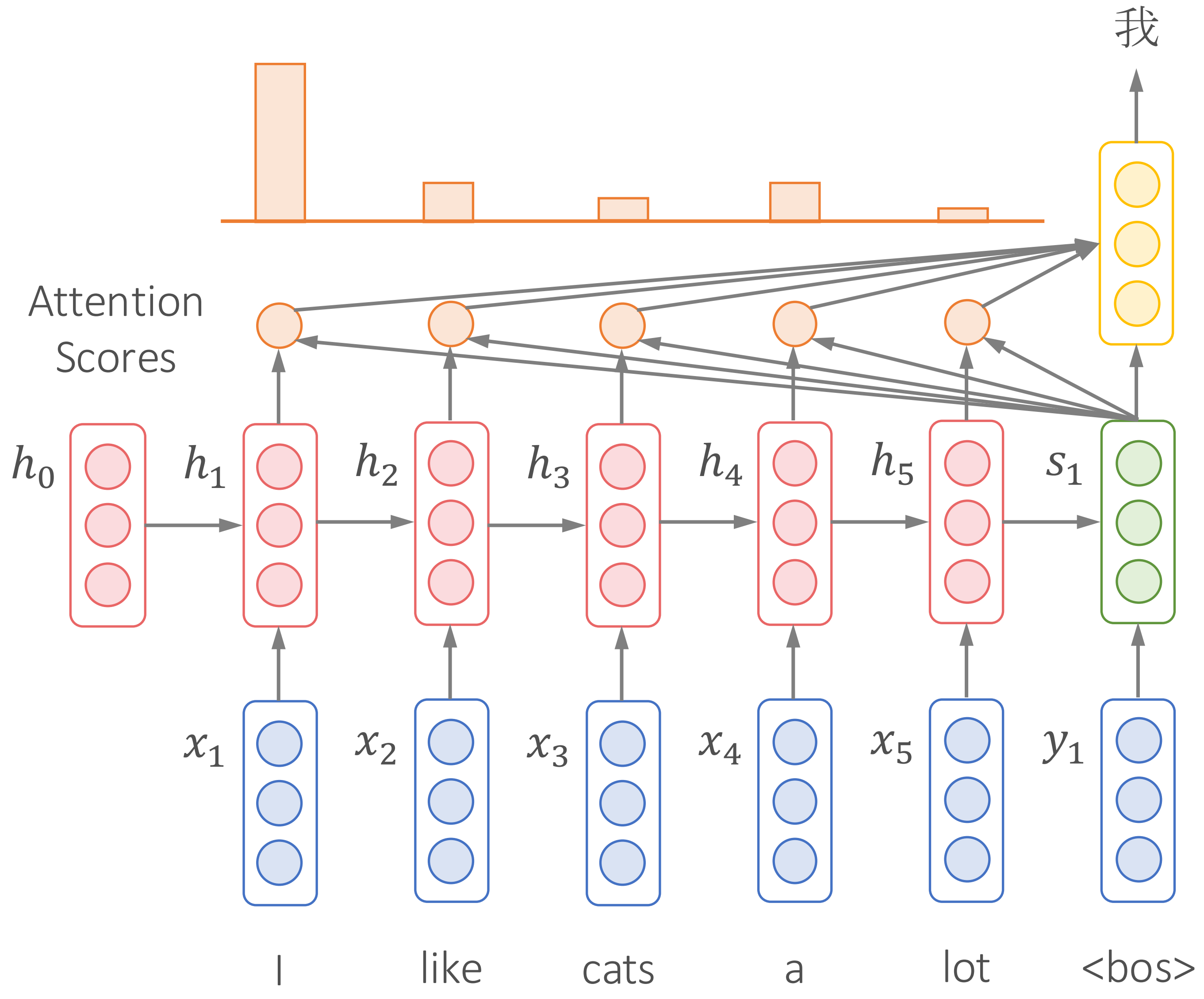


How Does Transformer Mimic Human Behavior?

Different context words have different **attention** weights



Look Back at RNN with Attention



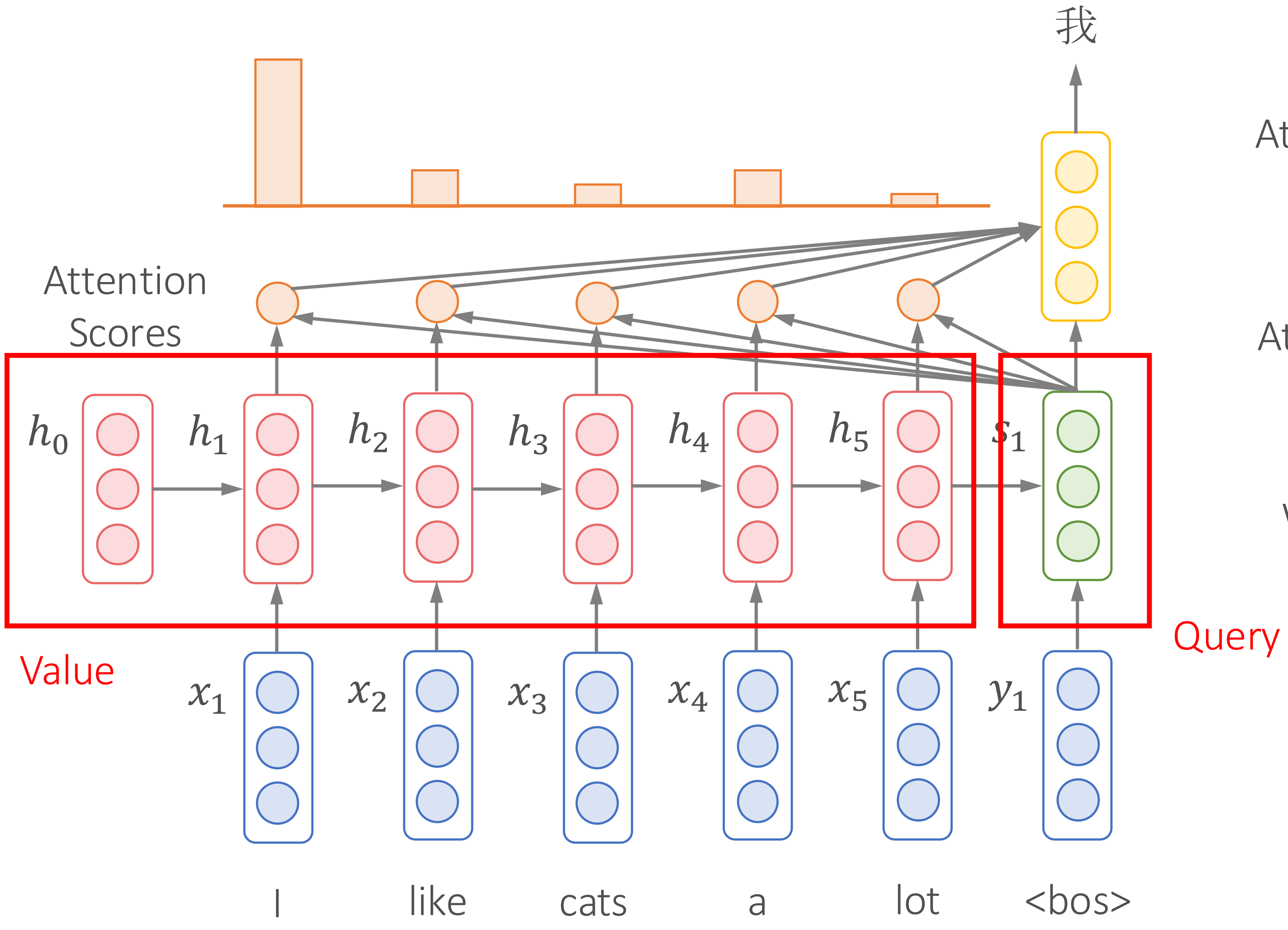
Attention Scores $\alpha_i = h_i^\top s_1$

Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $a = \sum_i \hat{\alpha}_i h_i$

Attention Output $\tanh(\mathbf{W}[a; s_1])$

Look Back at RNN with Attention



Inner Product of Query and Values

Attention Scores $\boxed{\alpha_i = h_i^{\mathsf{T}} s_1}$

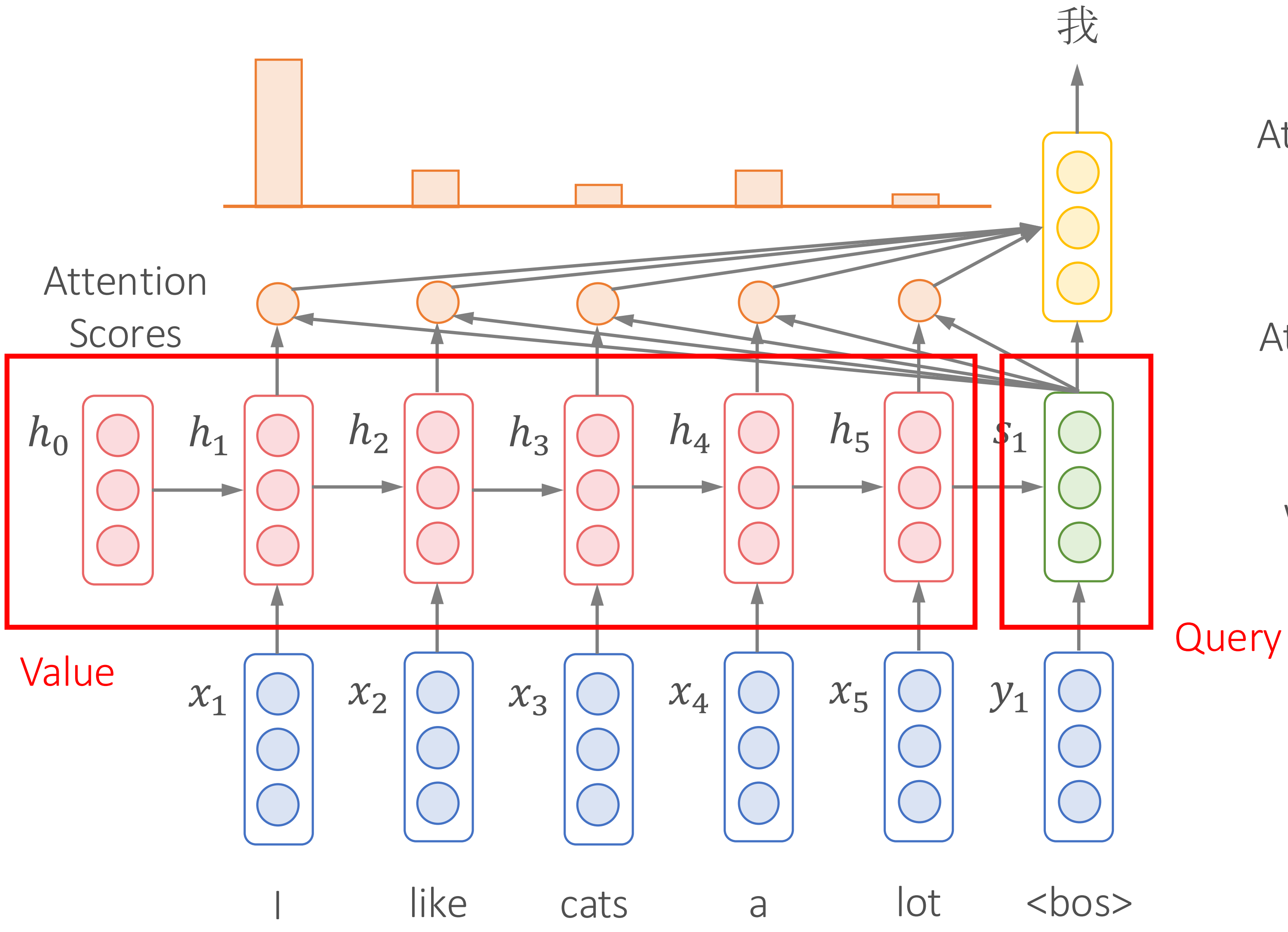
Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $\boxed{a = \sum_i \hat{\alpha}_i h_i}$

Weighted Sum of Values

Attention Output $\tanh(\mathbf{W}[a; s_1])$

Look Back at RNN with Attention – General Version



Inner Product of Query and Keys

Attention Scores $\boxed{\alpha_i = h'^{\mathsf{T}}_i s_1}$

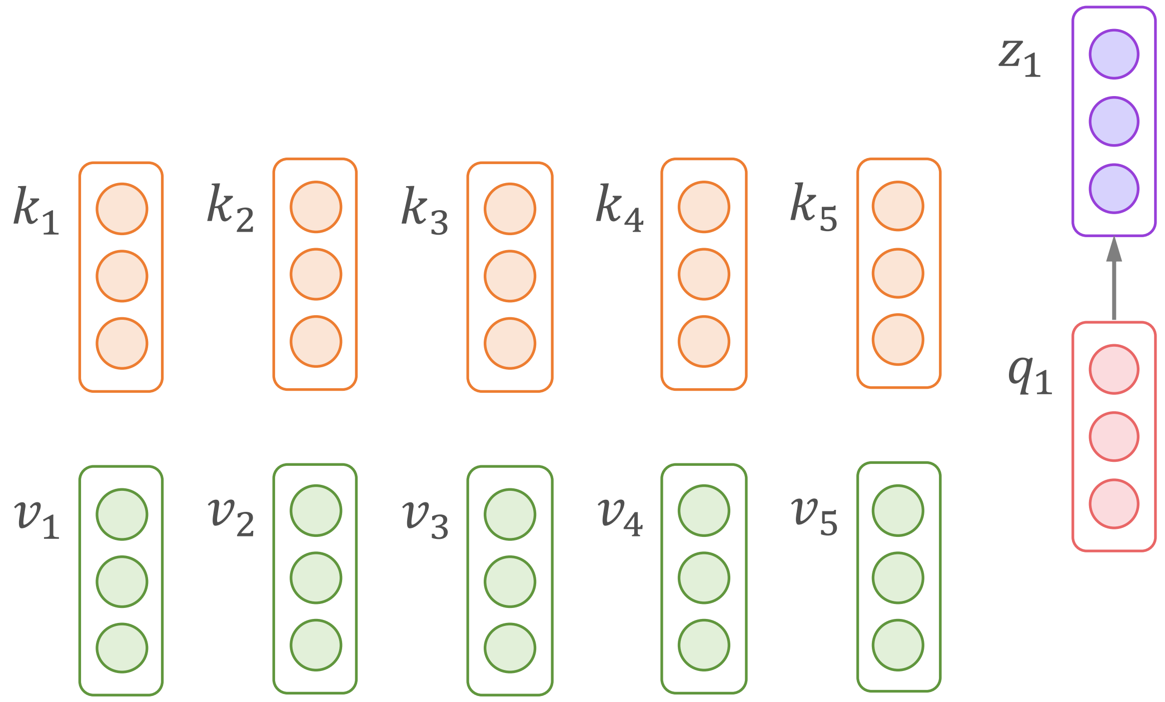
Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $\boxed{a = \sum_i \hat{\alpha}_i h_i}$

Weighted Sum of Values

Attention Output $\tanh(\mathbf{W}[a; s_1])$

Attention – General Version



Attention Scores $\alpha_i = k_i^\top q_1$

Normalized Attention Scores $\hat{\alpha}_i = \mathrm{softmax}(\alpha_i)$

Weighted Sum $z_1 = \sum_i \hat{\alpha}_i v_i$

From Attention to Self-Attention

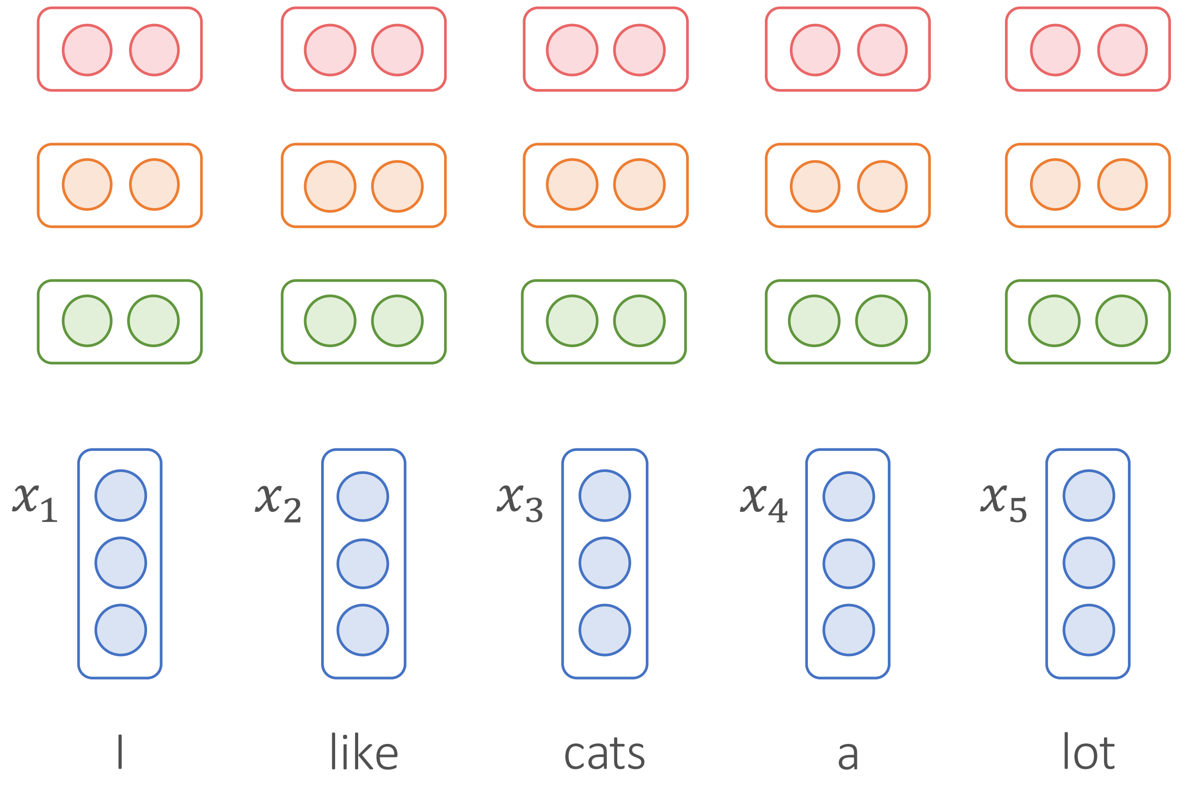
- Self-attention = attention from the sequence to itself
  - The queries, keys and values come from the same source
- Any word can be a query
- Any word can be a key
- Any word can be a value

Self-Attention

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



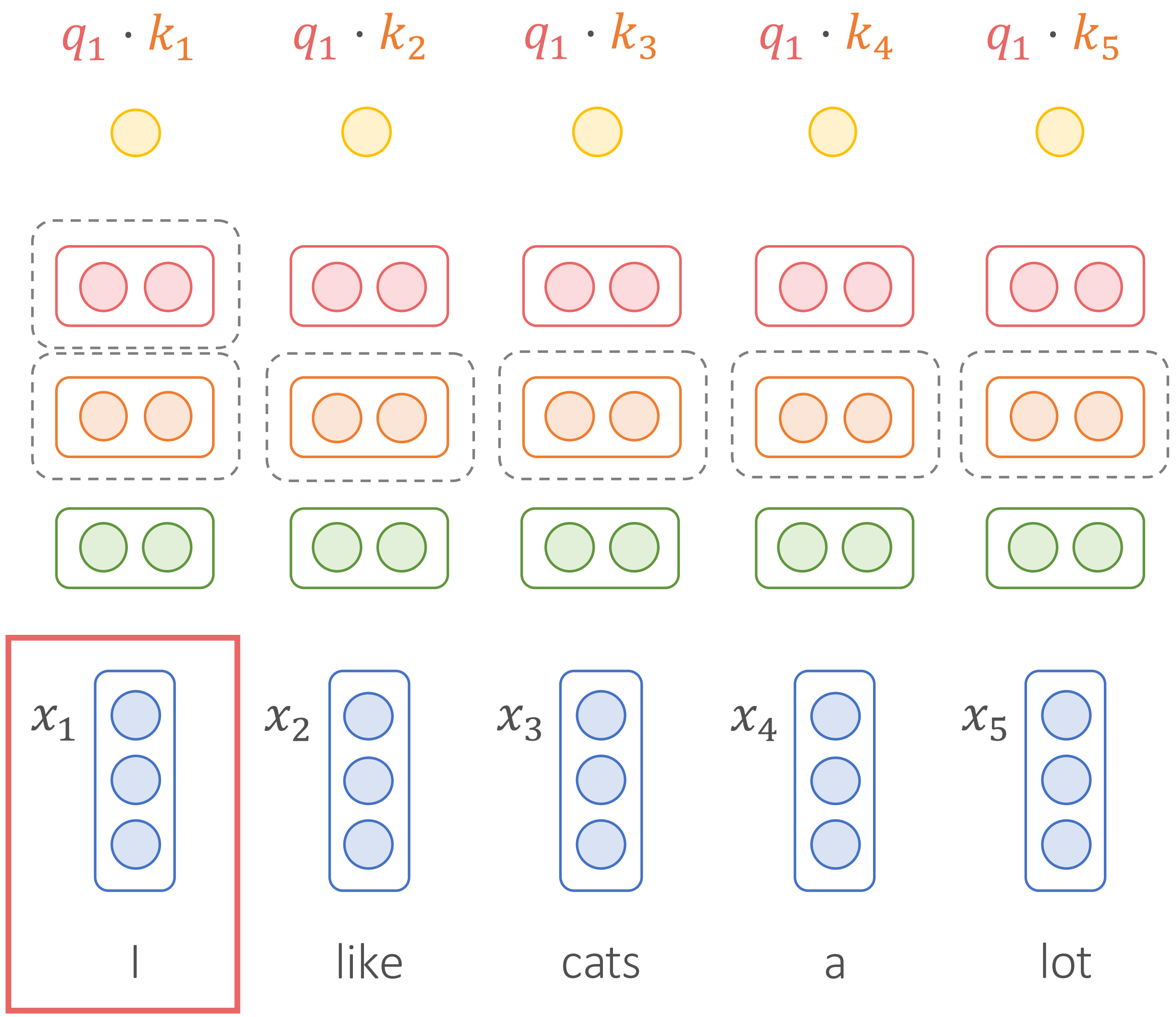
Self-Attention

Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

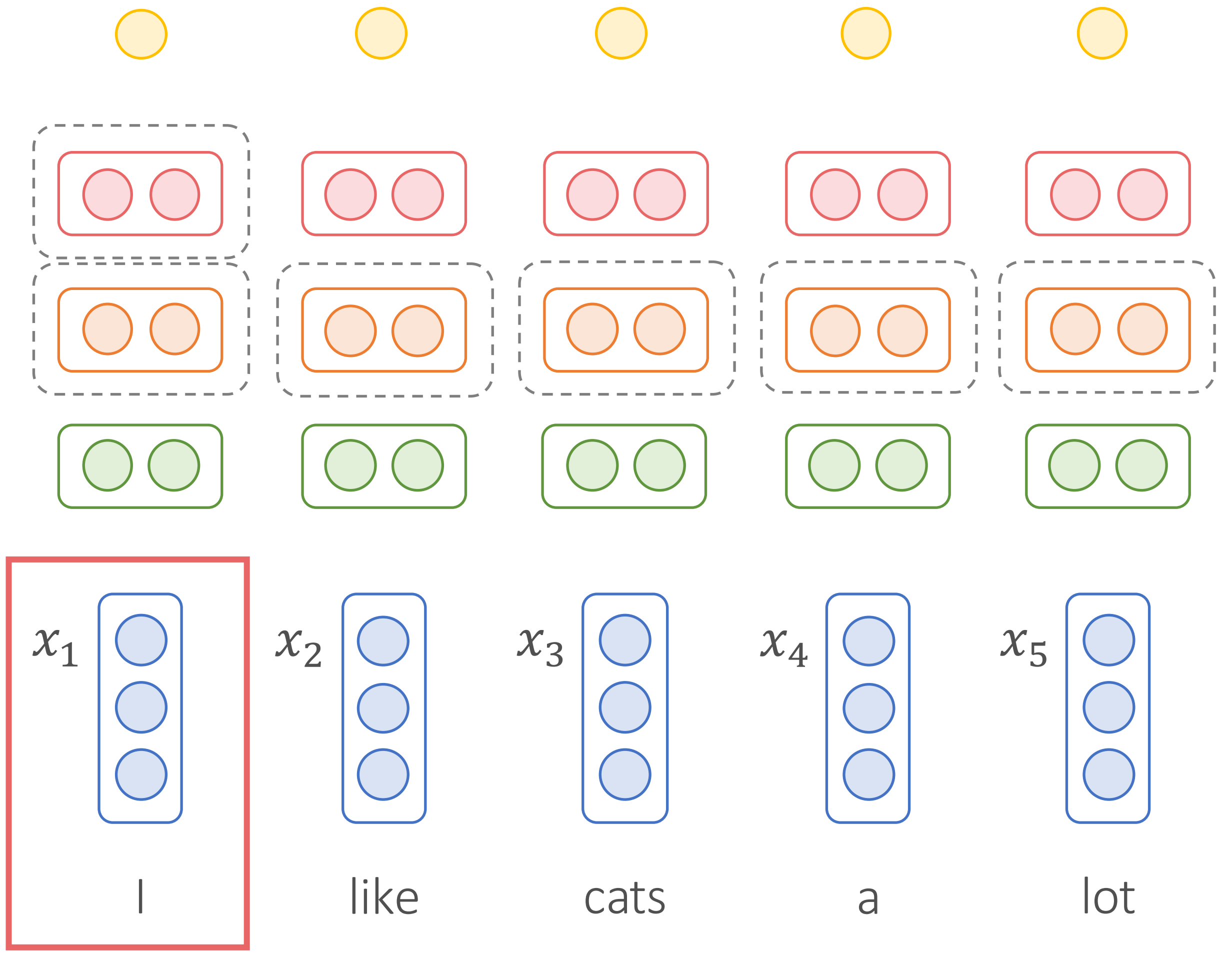
$$\alpha_{1,i} = \text{softmax}\left(\frac{q_1 \cdot k_i}{\sqrt{d}}\right)$$ Vector dimension

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

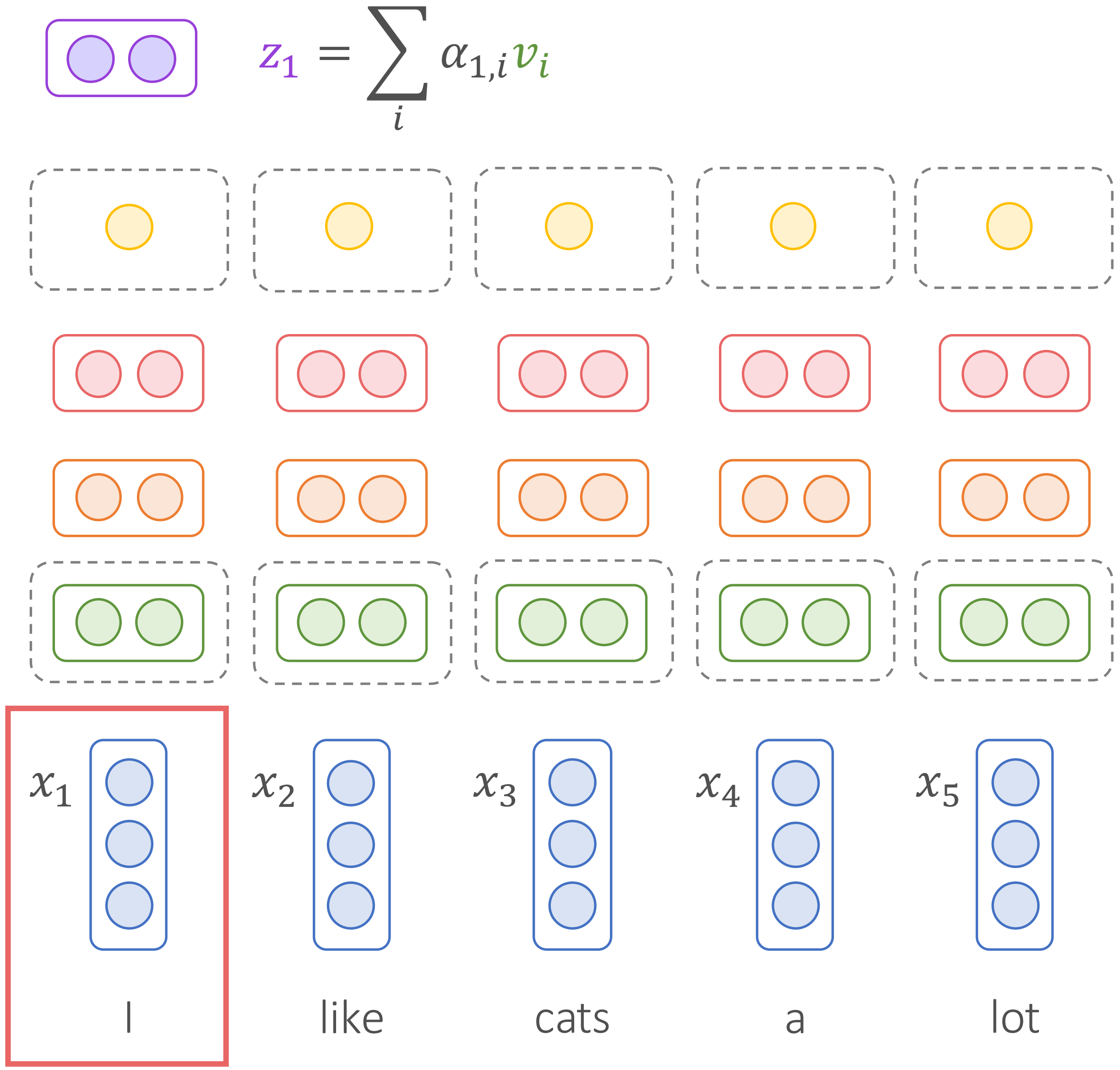
Weighted Sum $z_1 = \sum_i \alpha_{1,i} v_i$

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



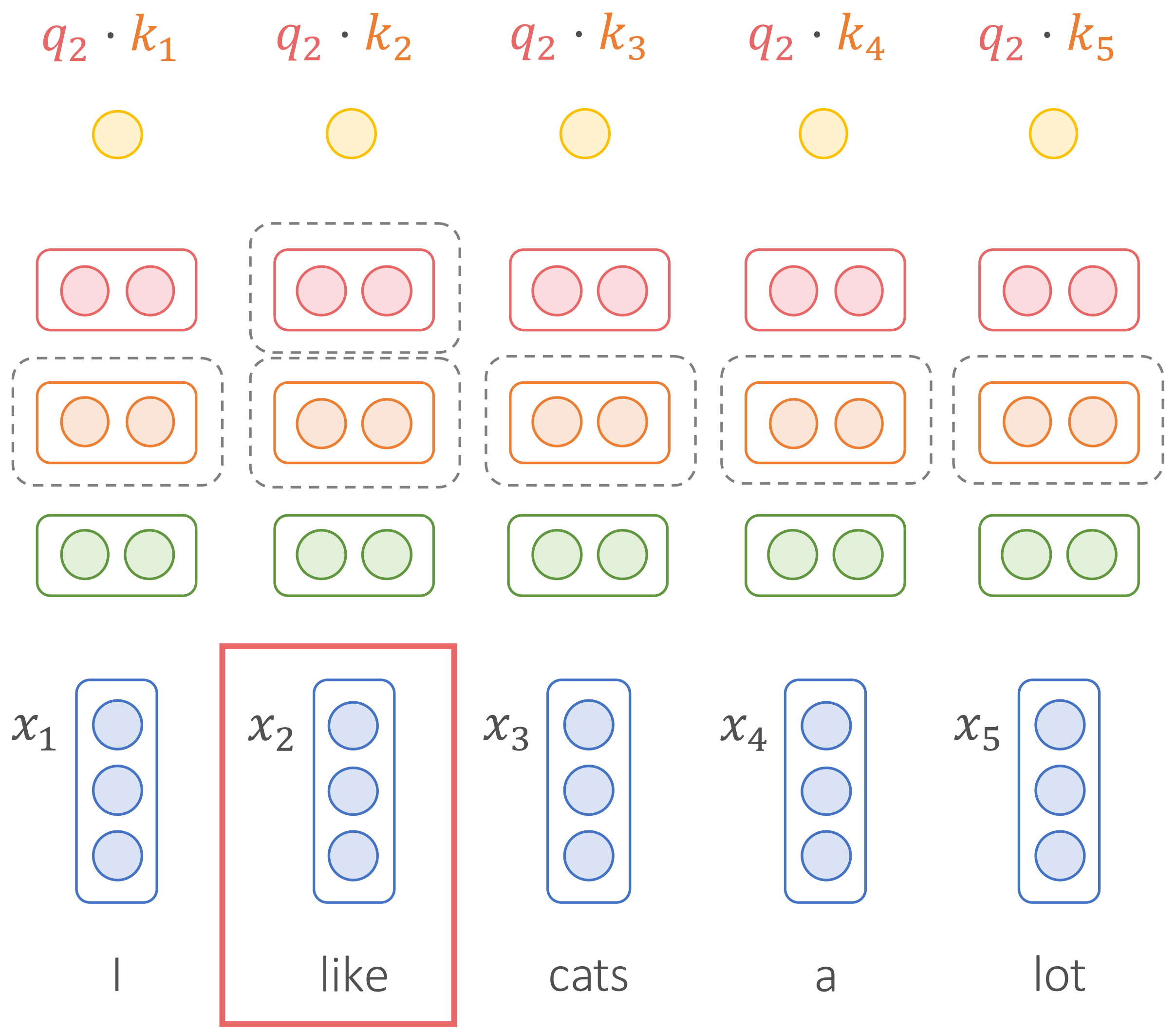
Self-Attention

Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

$$\alpha_{2,i} = \text{softmax}\left(\frac{q_2 \cdot k_i}{\sqrt{d}}\right)$$

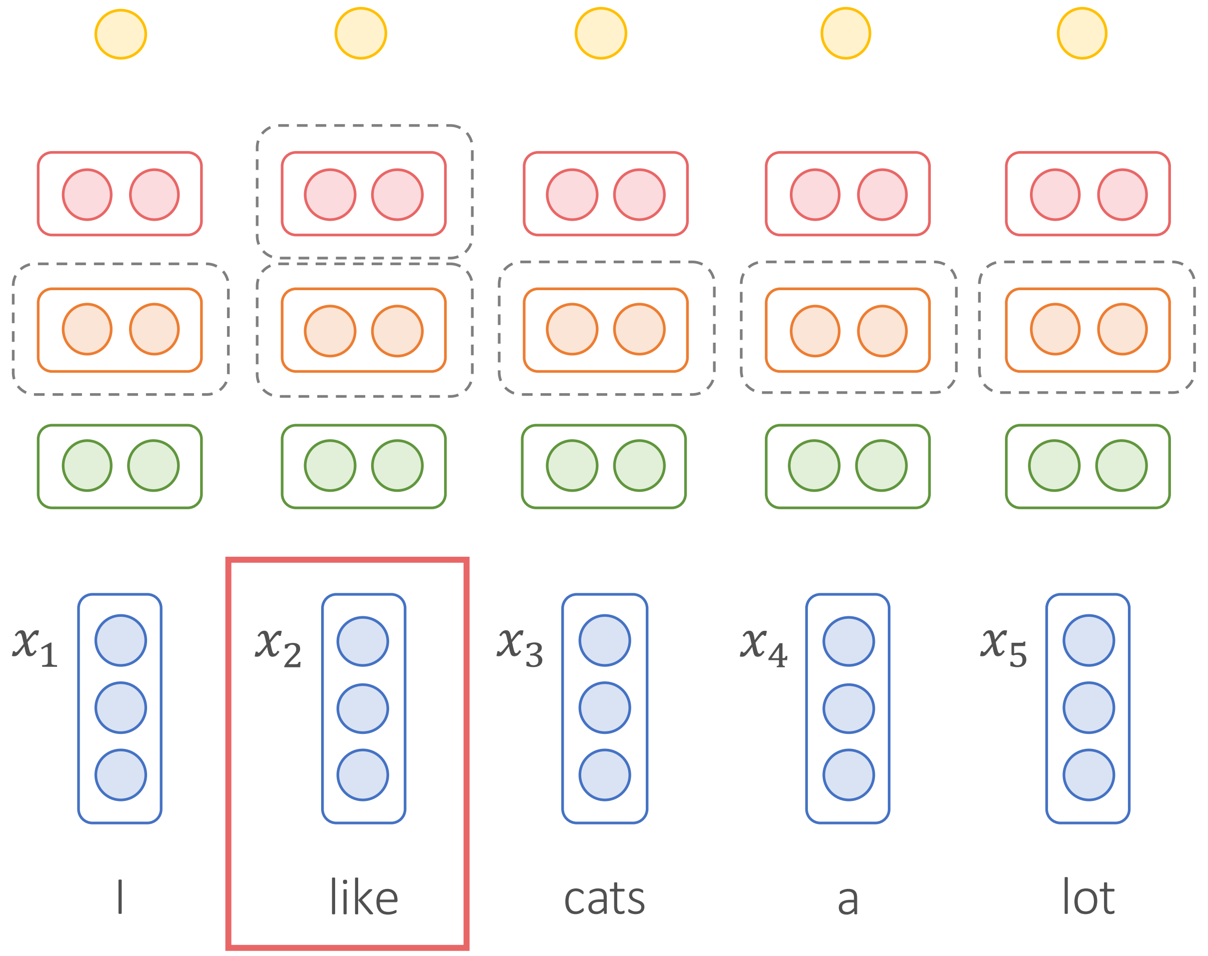
Vector dimension

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention

Weighted Sum

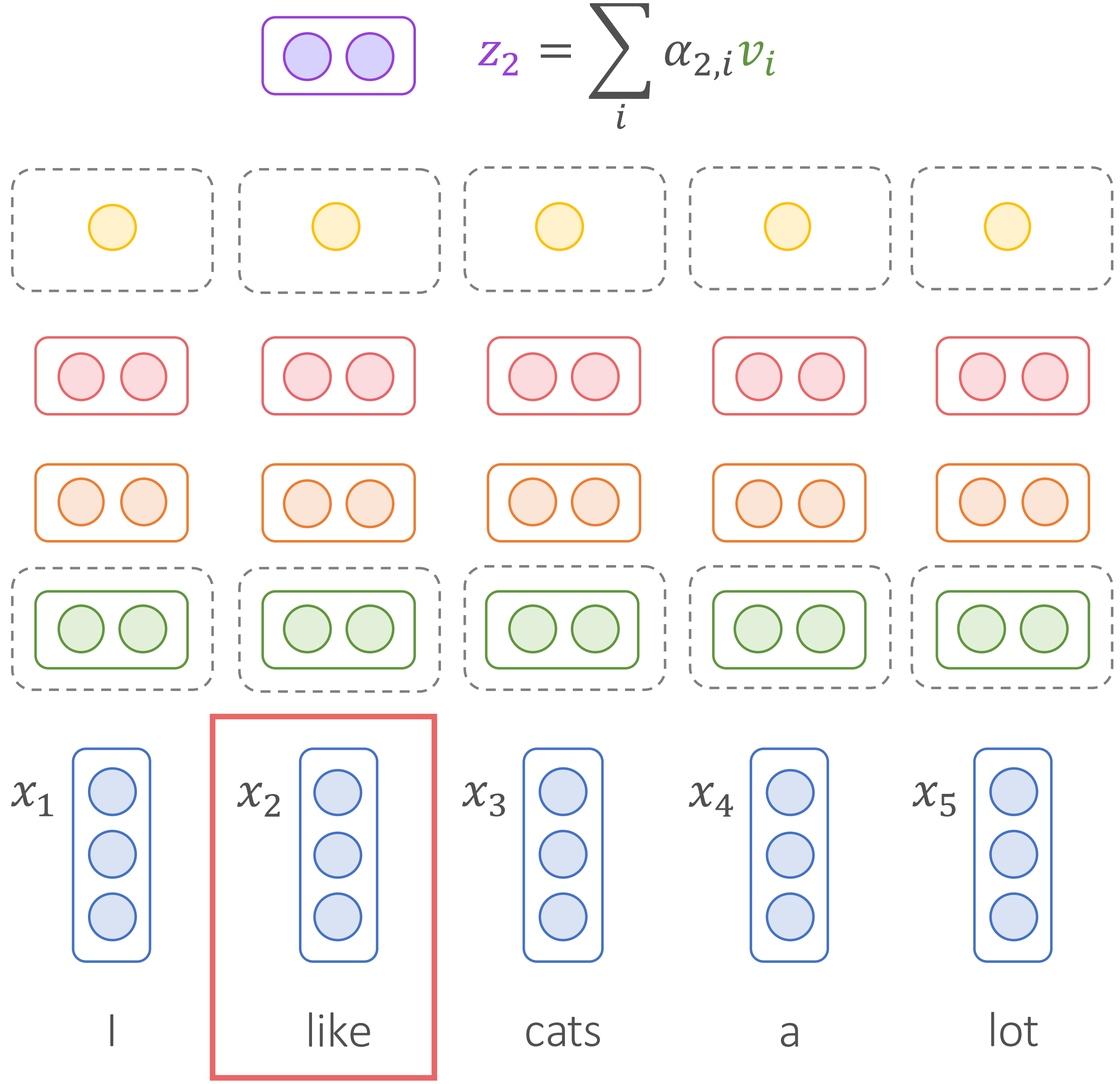
Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$

$$z_2 = \sum_i \alpha_{2,i} v_i$$



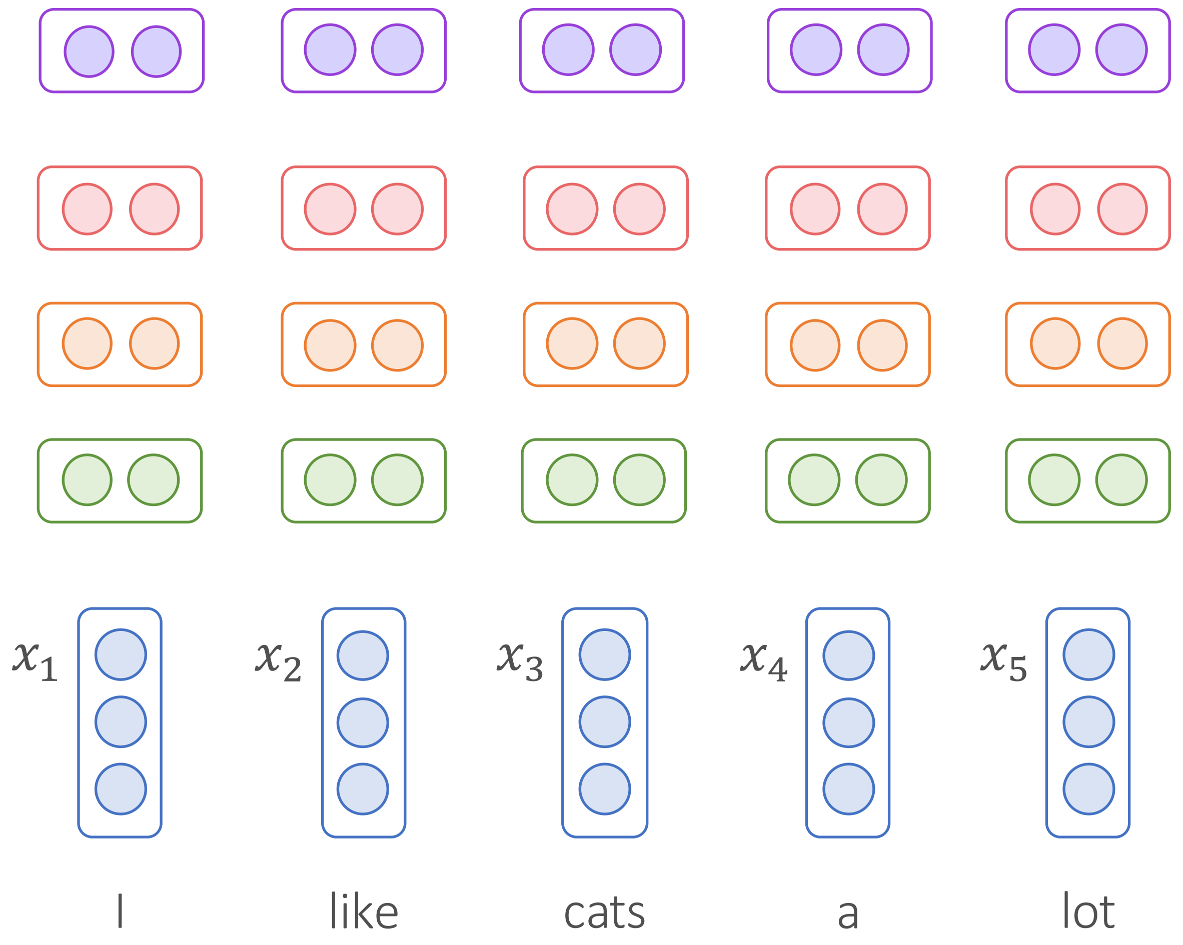
Self-Attention

Self-Attention Output

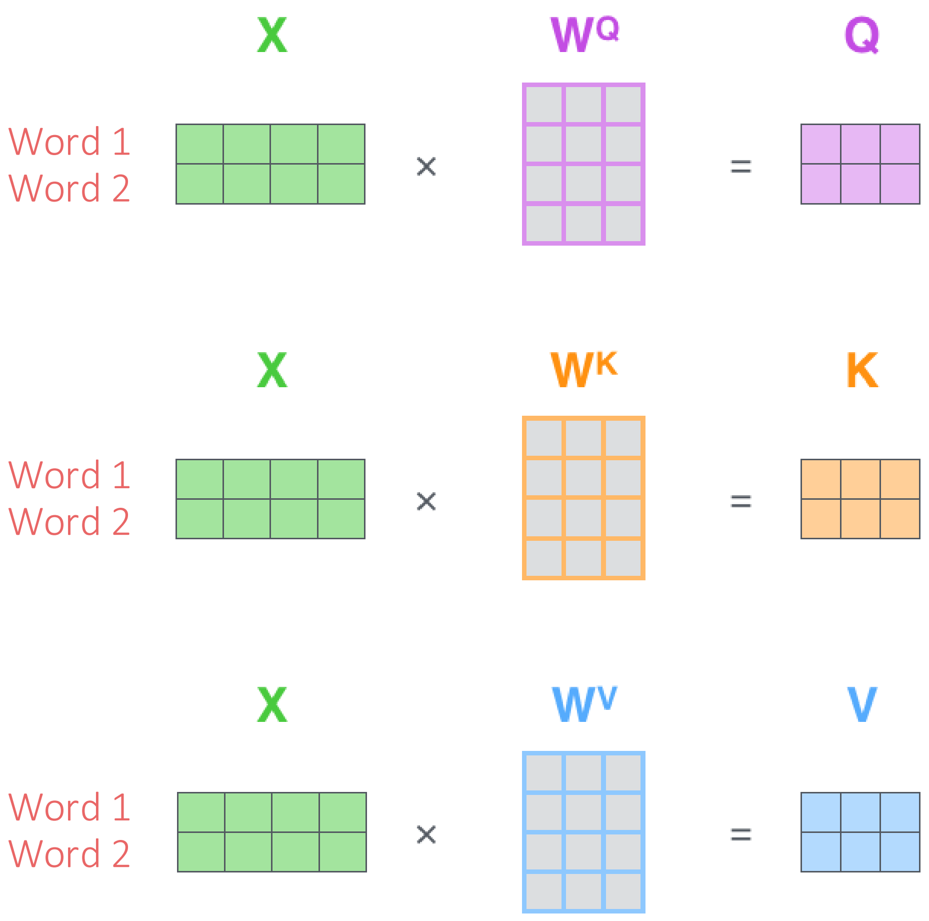
Query $q_i = W^Q x_i$

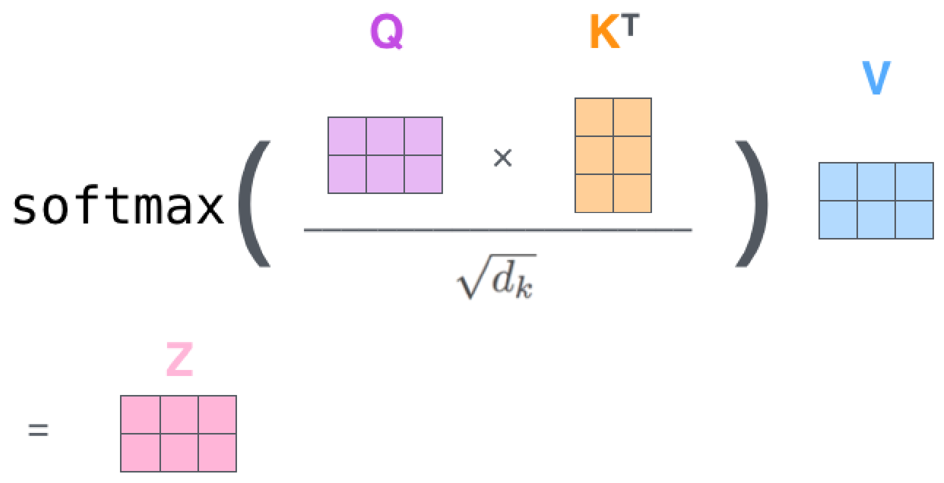
Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Self-Attention – Matrix Form





$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

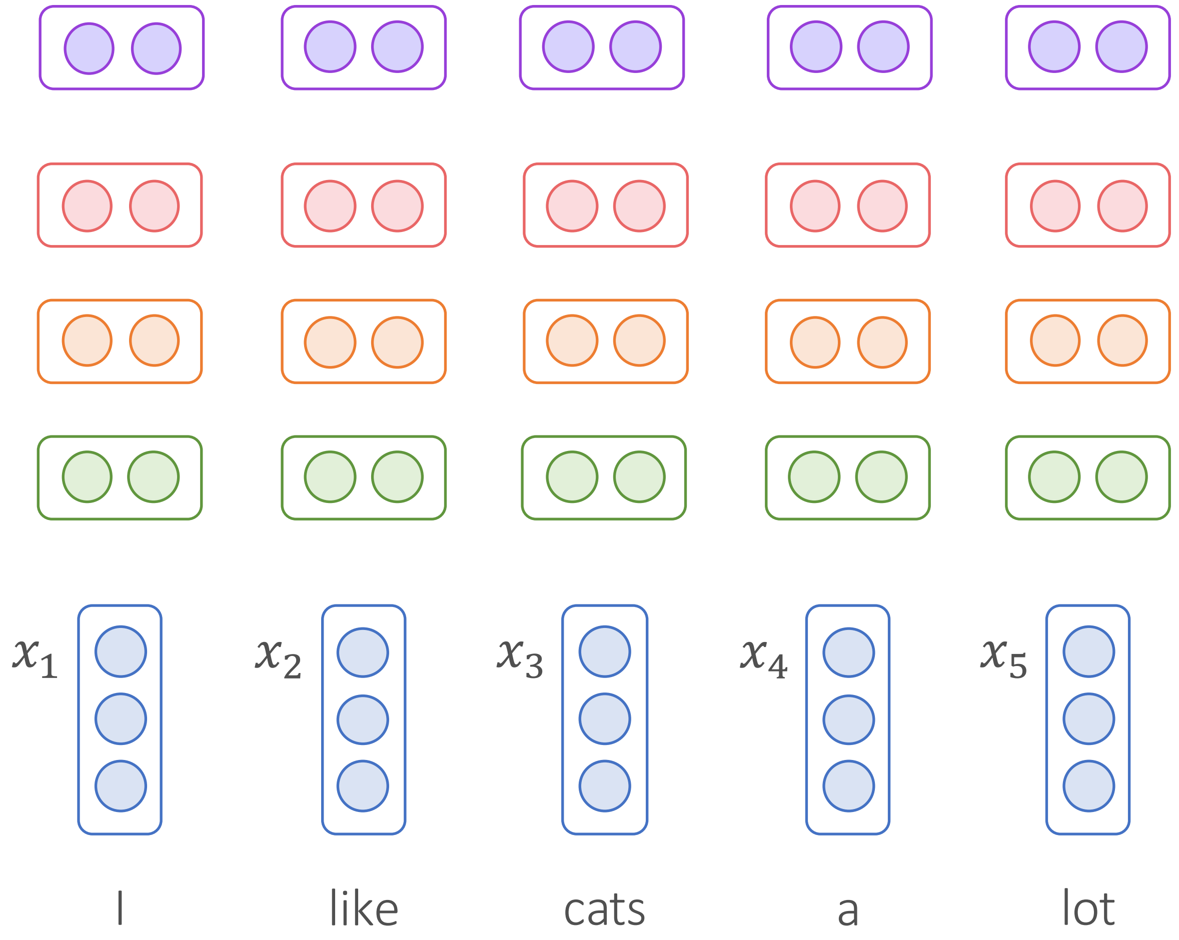
Single-Head Attention

Self-Attention Output

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Multi-Head Attention

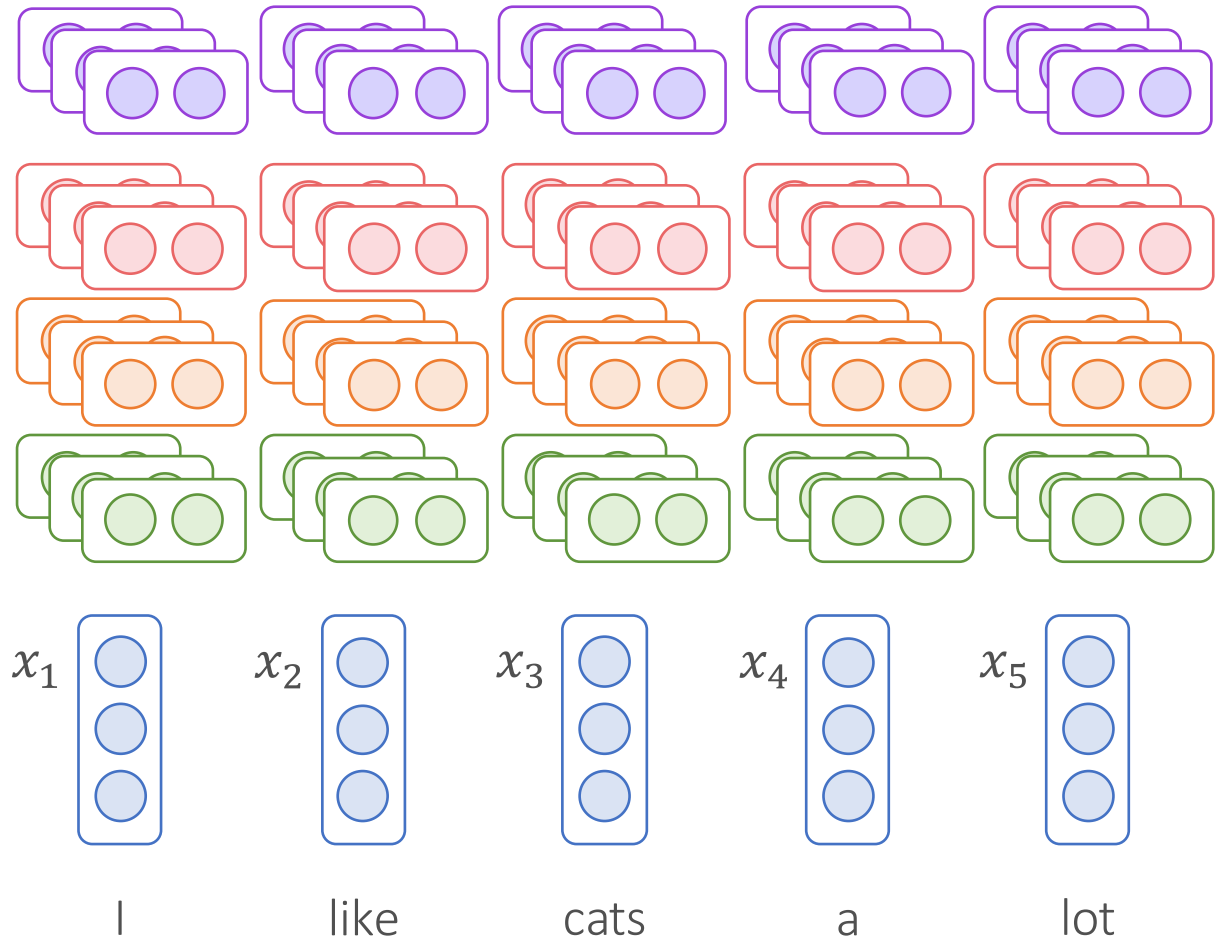
Each attention head focuses on different parts of understanding!

Multi-Attention Output

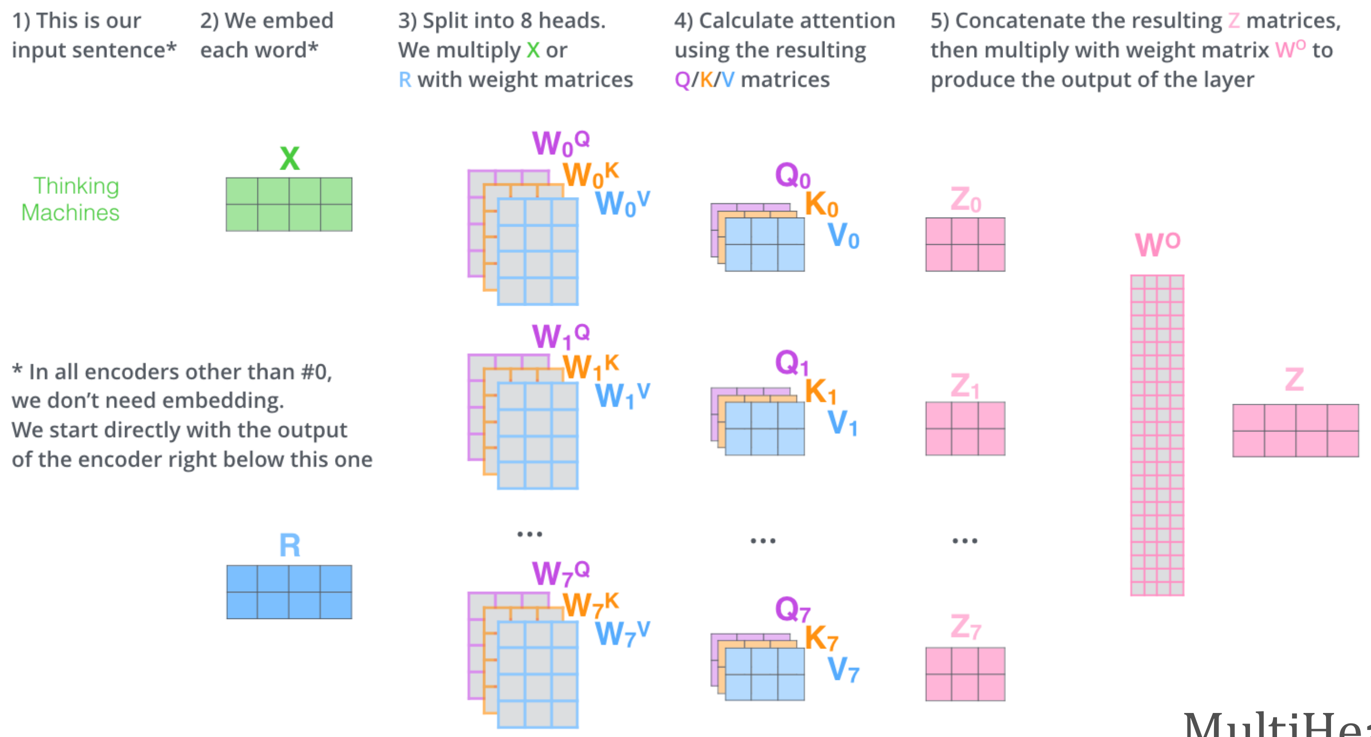
Query $q_i = W_j^Q x_i$

Key $k_i = W_j^K x_i$

Value $v_i = W_j^V x_i$



Multi-Head Attention – Matrix Form



1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

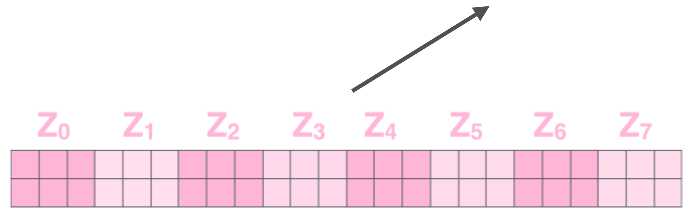
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

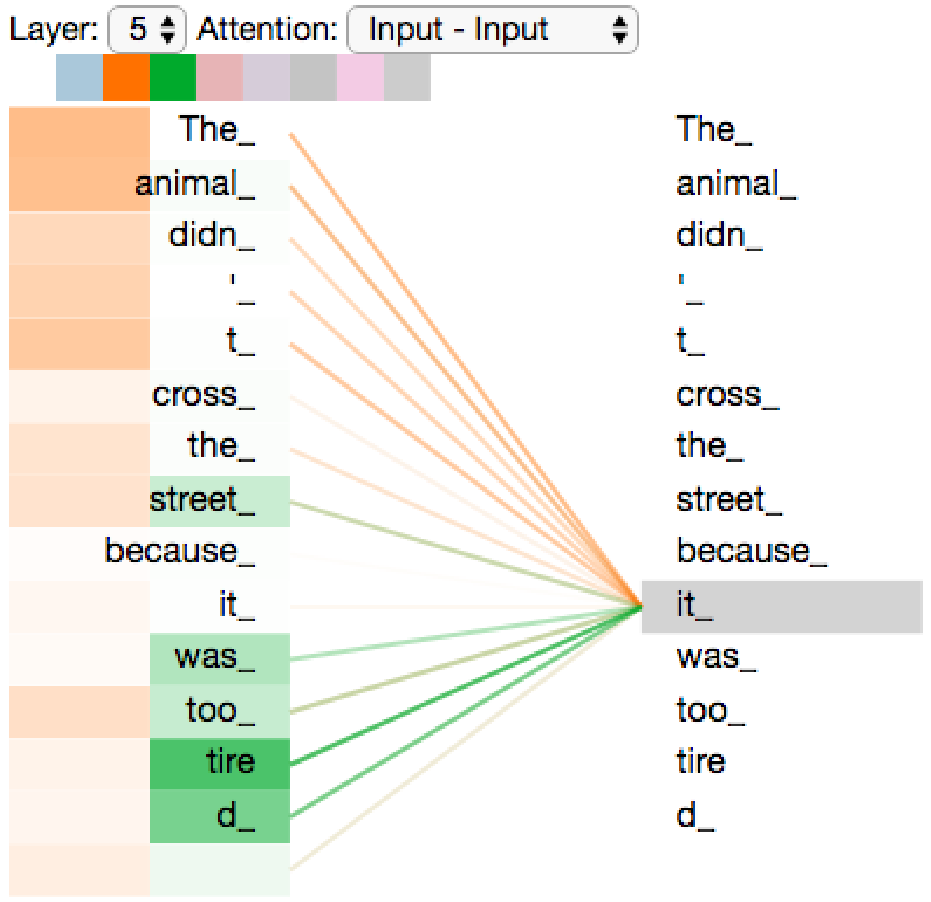
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

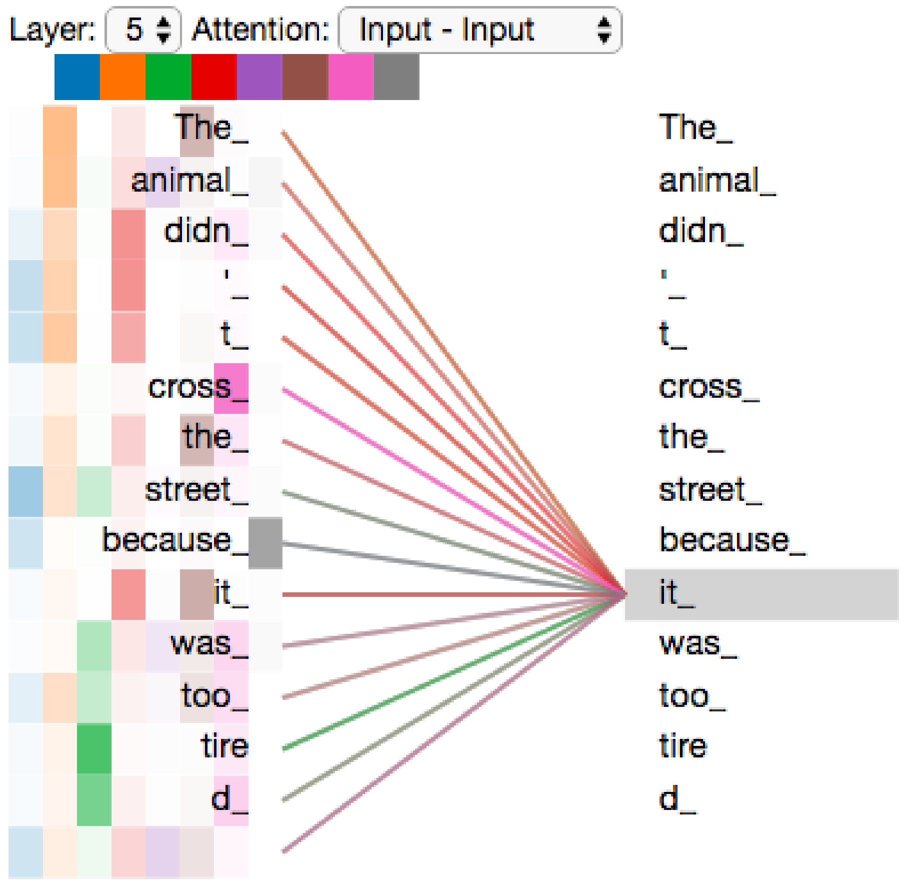
$$\text{head}_i = \text{Attention}\left(XW_i^Q, XW_i^K, XW_i^V\right)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

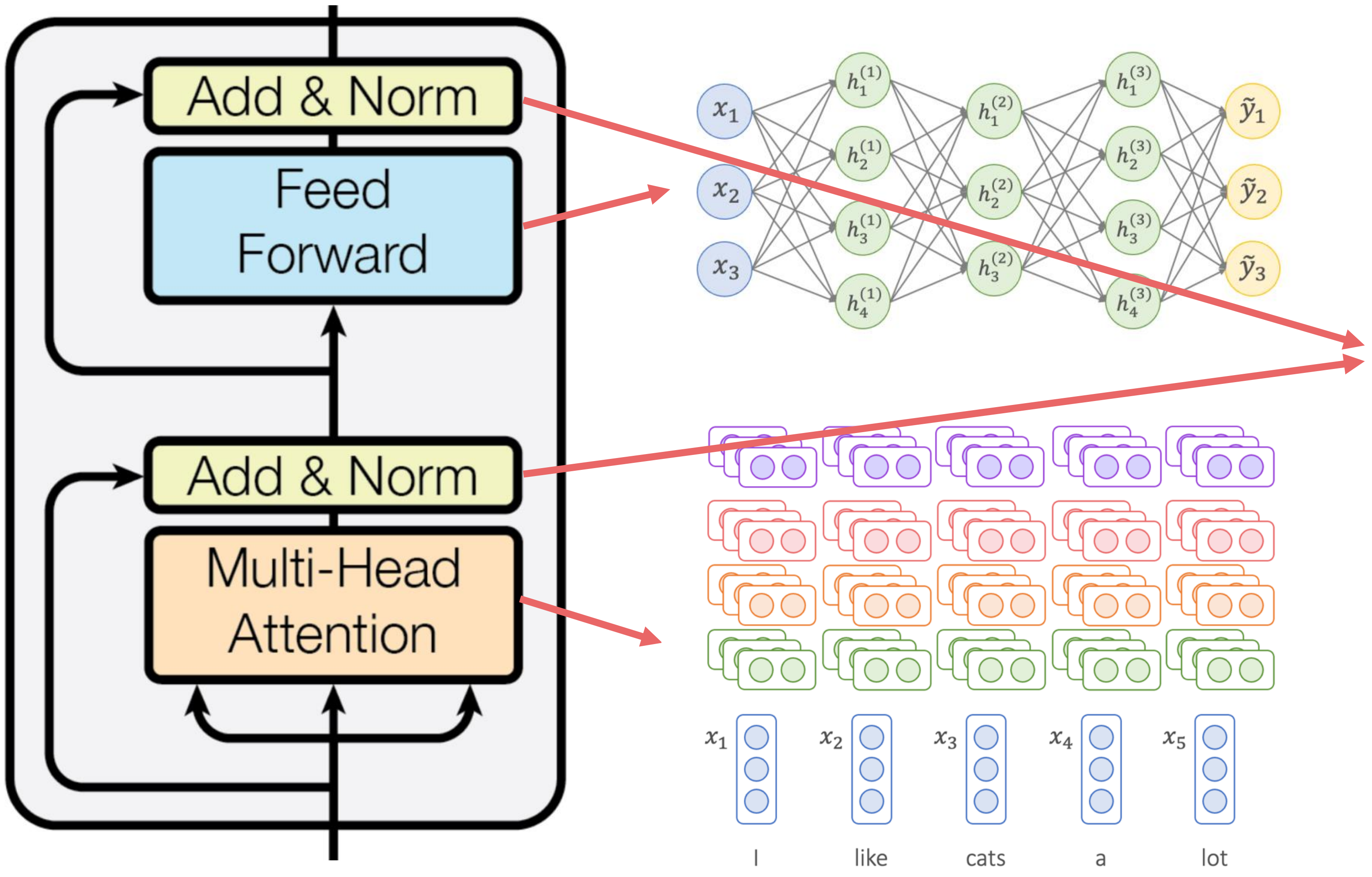


What Does Multi-Head Attention Learn?





Transformer Layer

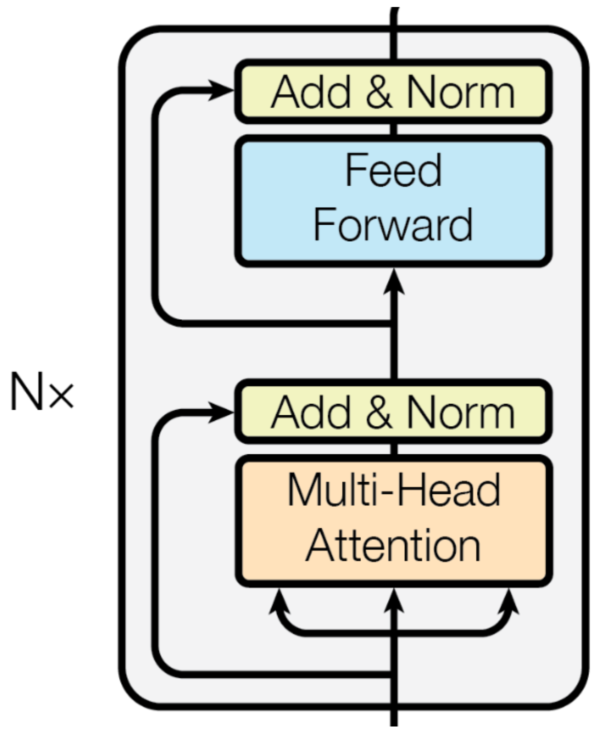

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
x_1 x_2 x_3
$h_1^{(1)}$ $h_2^{(1)}$ $h_3^{(1)}$ $h_4^{(1)}$
$h_1^{(2)}$ $h_2^{(2)}$ $h_3^{(2)}$
$h_1^{(3)}$ $h_2^{(3)}$ $h_3^{(3)}$ $h_4^{(3)}$
$\tilde{y}_1$ $\tilde{y}_2$ $\tilde{y}_3$
x_1 x_2 x_3 x_4 x_5
I
like
cats
a
lot

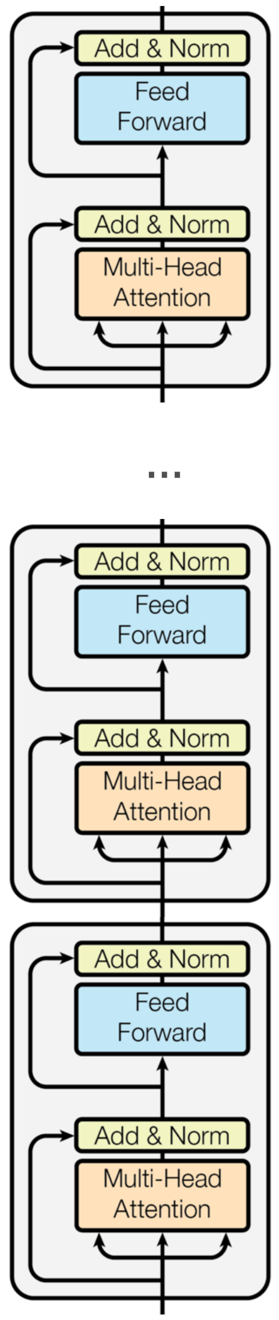
$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

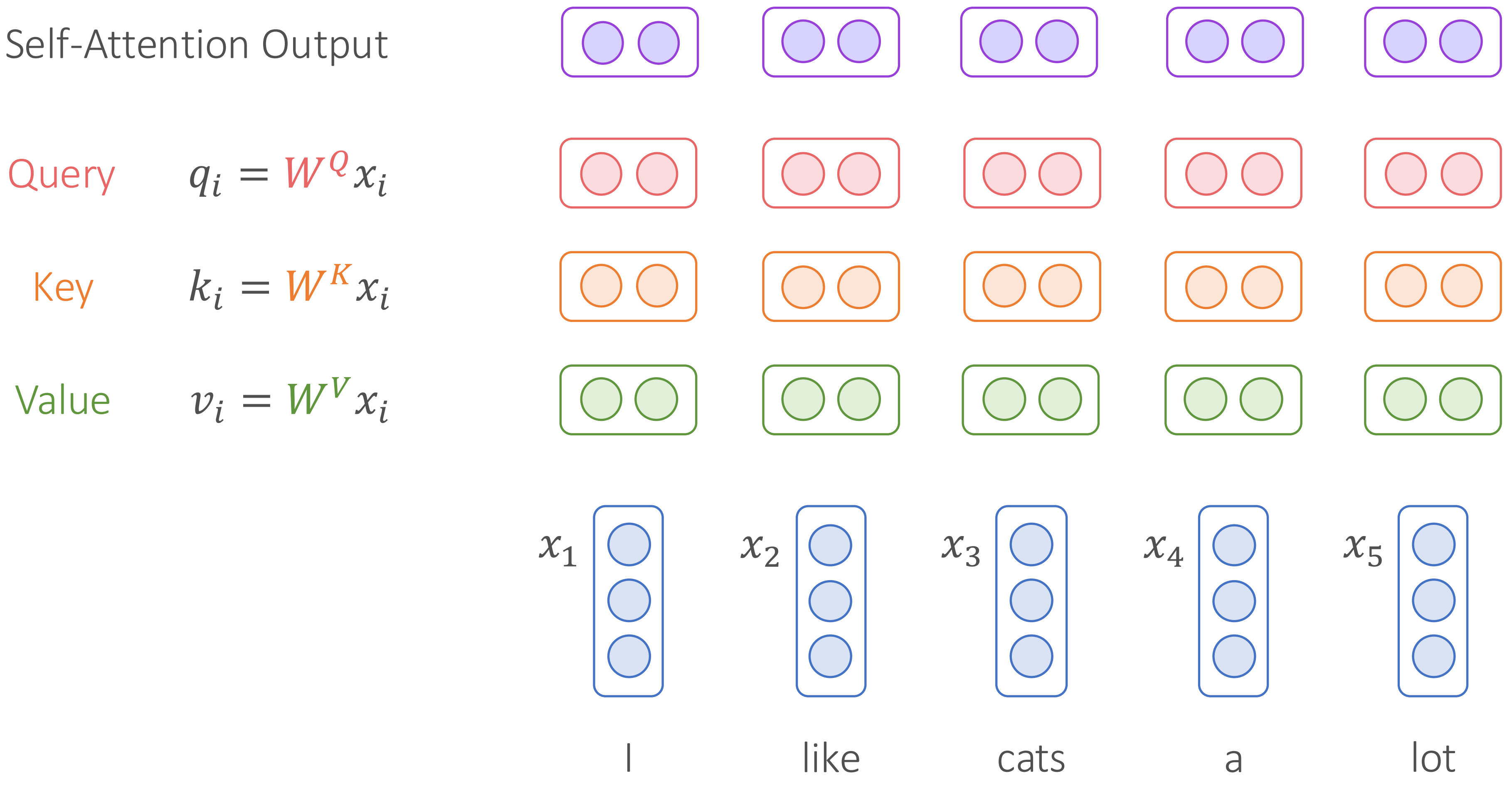
Residual connection (He et al., 2016)
Layer normalization (Ba et al., 2016)

Transformer Encoder





How About Word Order?



How About Word Order?

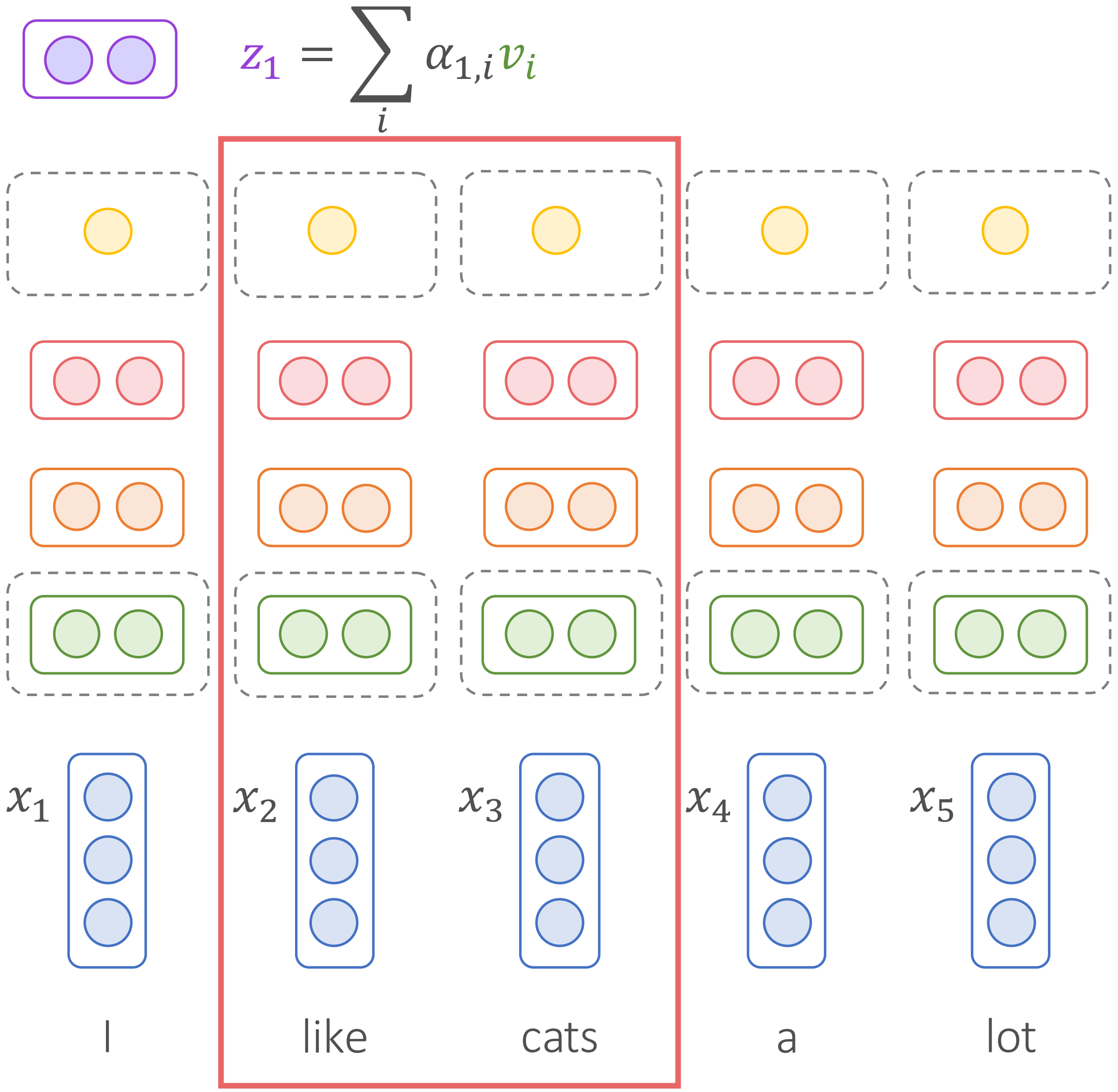
Weighted Sum

Normalized Attention Scores

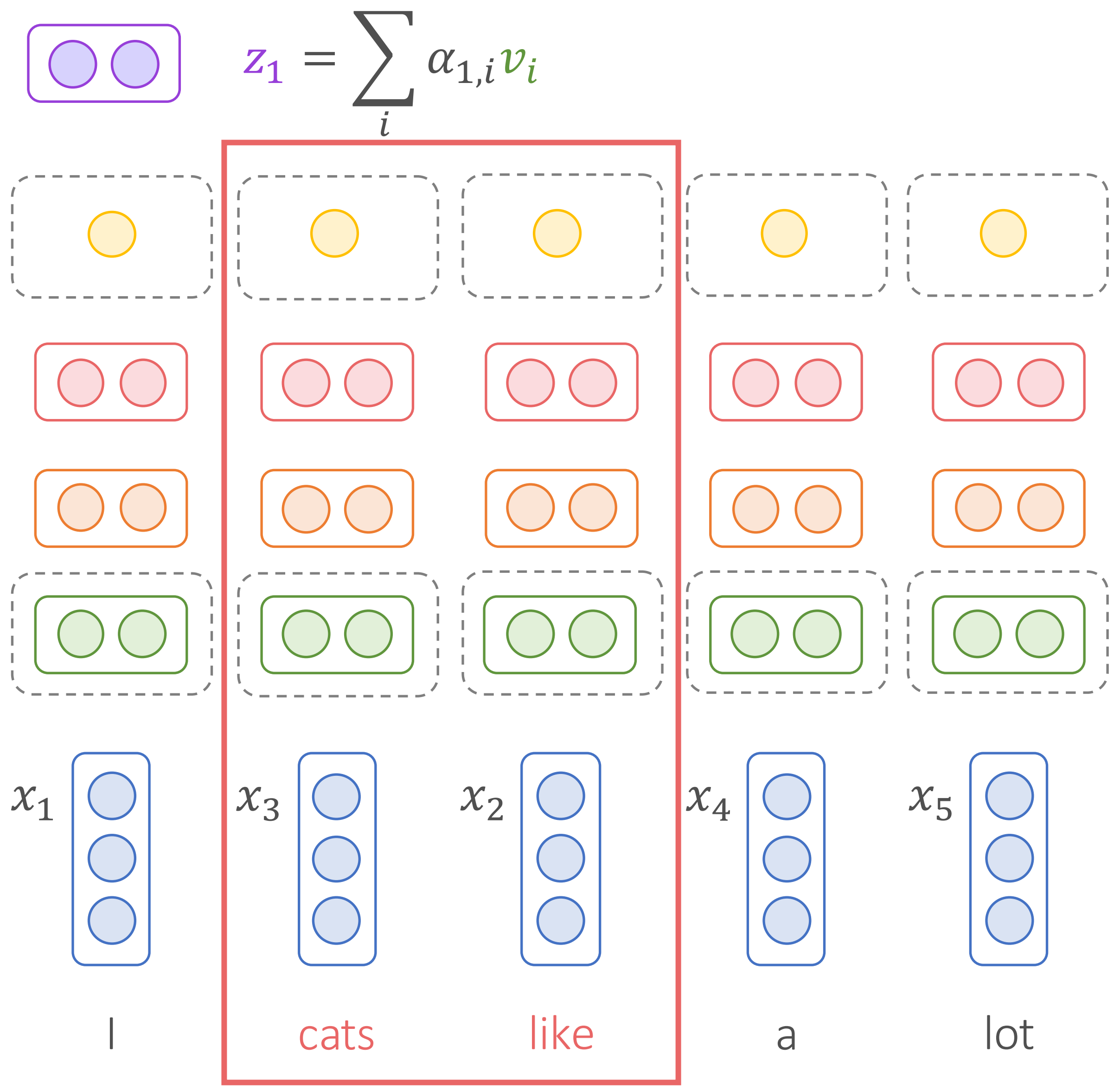
Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



How About Word Order?



Weighted Sum

Normalized Attention Scores

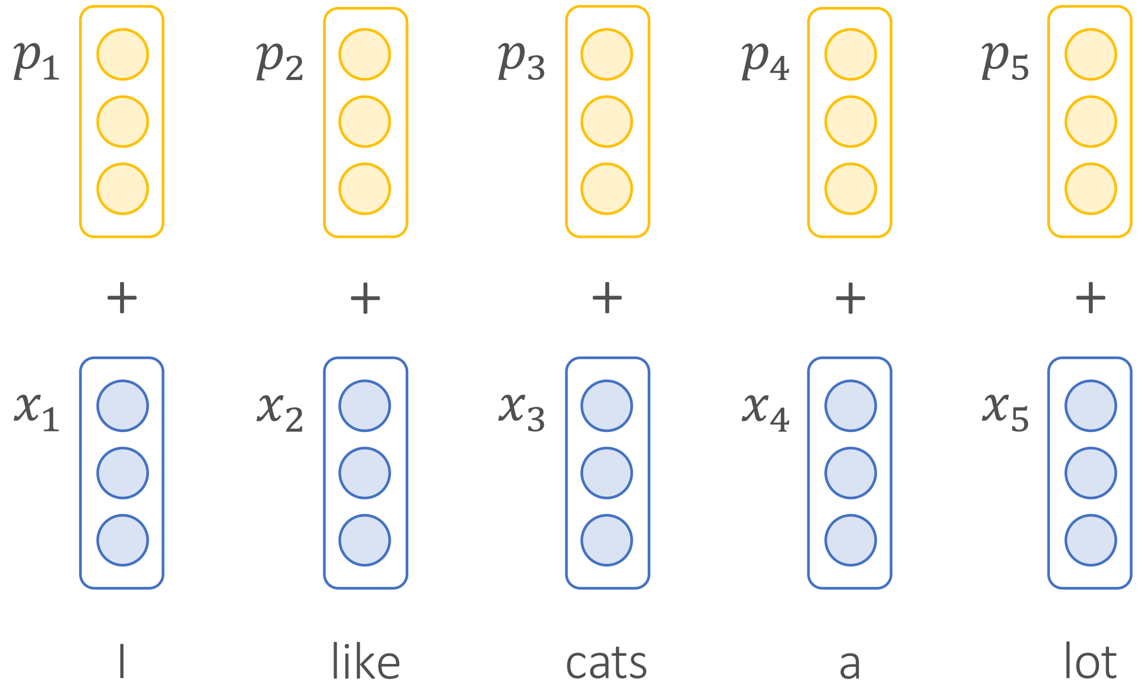
Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$

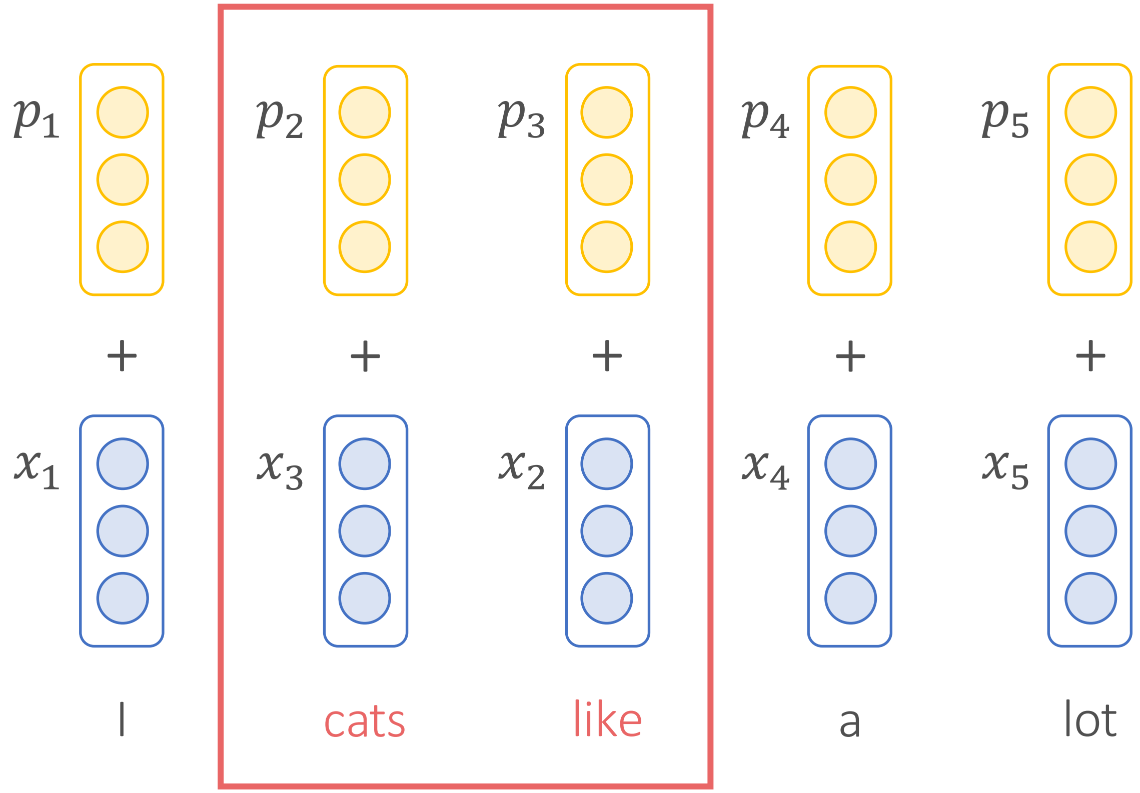
Solution: Positional Encoding

$$x_i \leftarrow x_i + PE_i$$



Solution: Positional Encoding

$$x_i \leftarrow x_i + PE_i$$



Solution: Positional Encoding

- Unique encoding for each position
- Closer positions should have more similar encodings
- Distance between neighboring positions should be the same

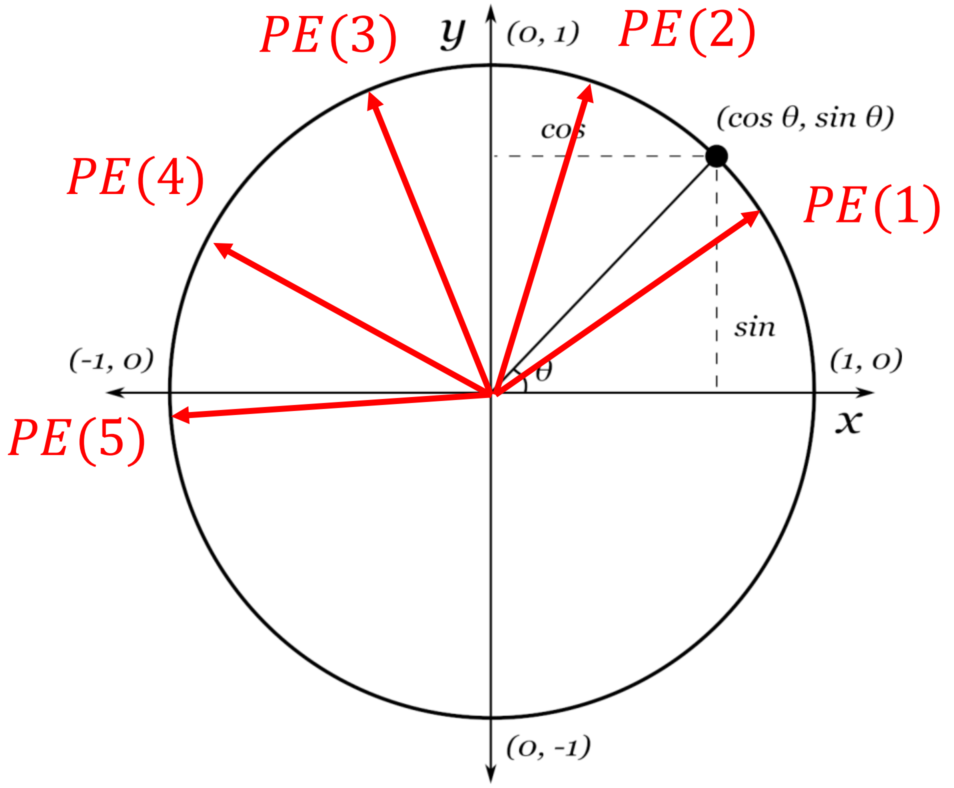
Sinusoidal Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Why this?

Sinusoidal Positional Encoding: Intuition

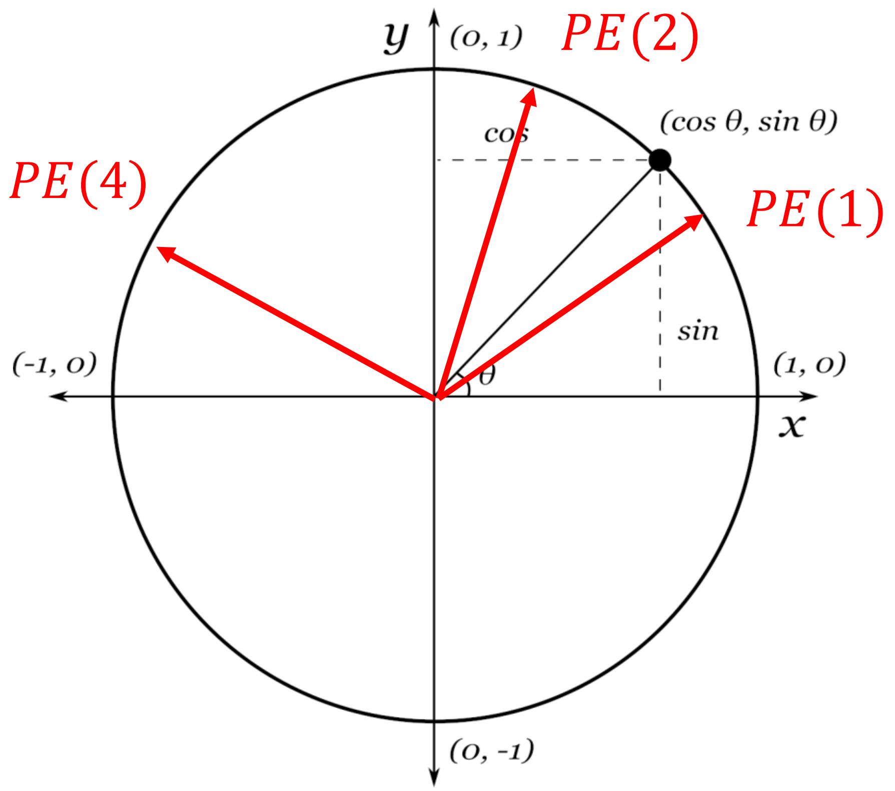
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



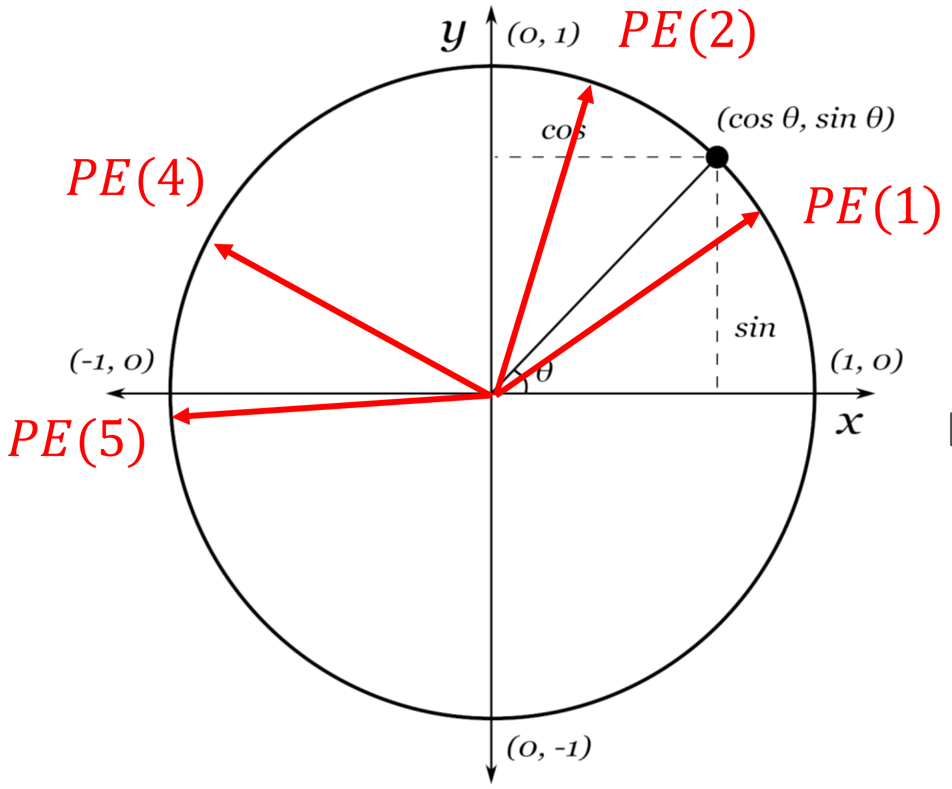
$$Cosine(PE(1), PE(2)) > Cosine(PE(1), PE(4))$$

Closer positions should have more similar encodings

https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



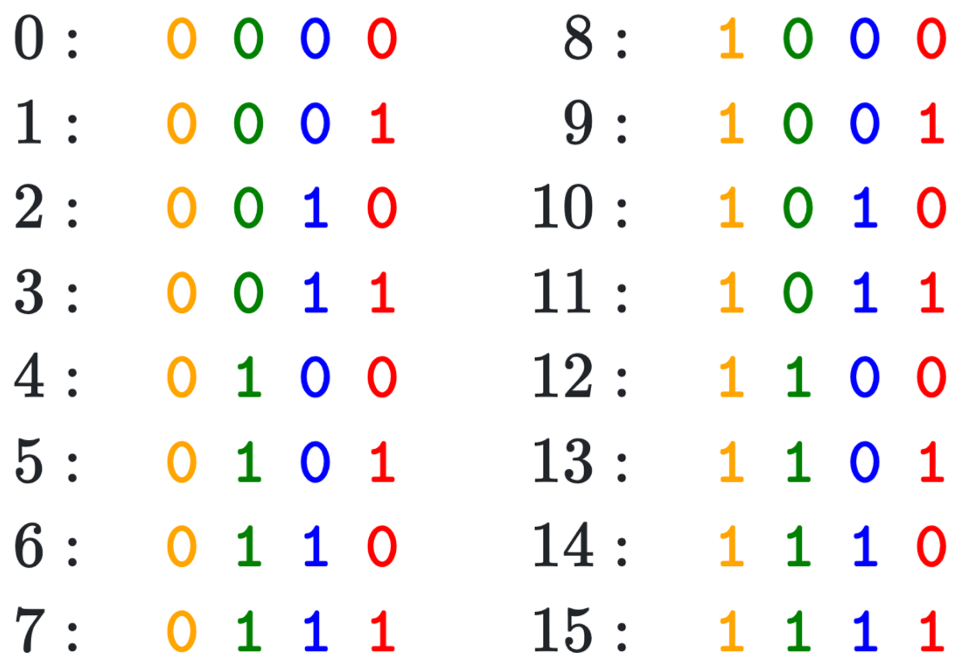
$$Cosine(PE(1), PE(2)) = Cosine(PE(4), PE(5))$$

Distance between neighboring positions should be the same

https://www.inchcalculator.com/unit-circle-calculator/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let’s consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

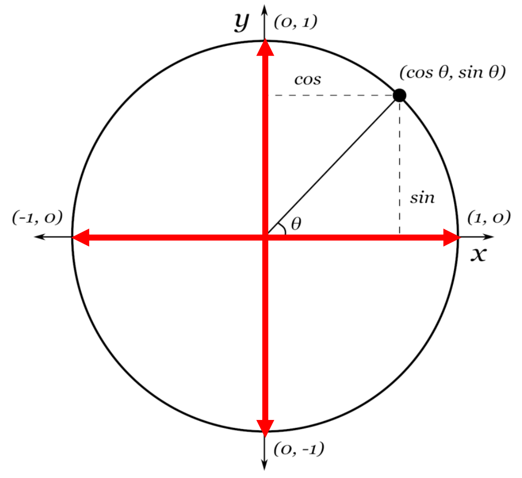


0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1



Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let’s consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

High frequency rotation

Low frequency rotation

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding: Intuition

- How to expand to high-dimension?
- Let's consider binary positional encoding first
- How to use 4 bits to represent position 0~15?

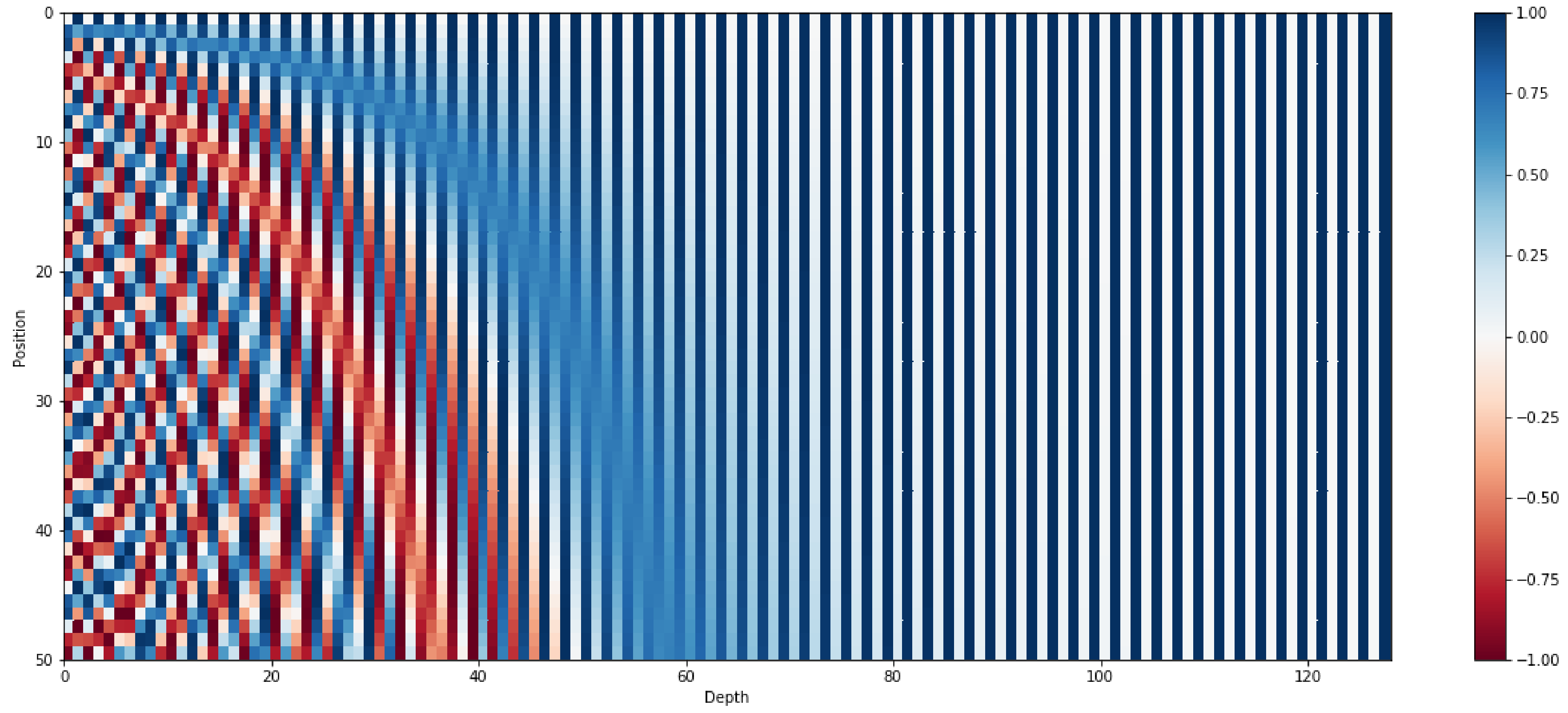
0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Soft version of alternating bits

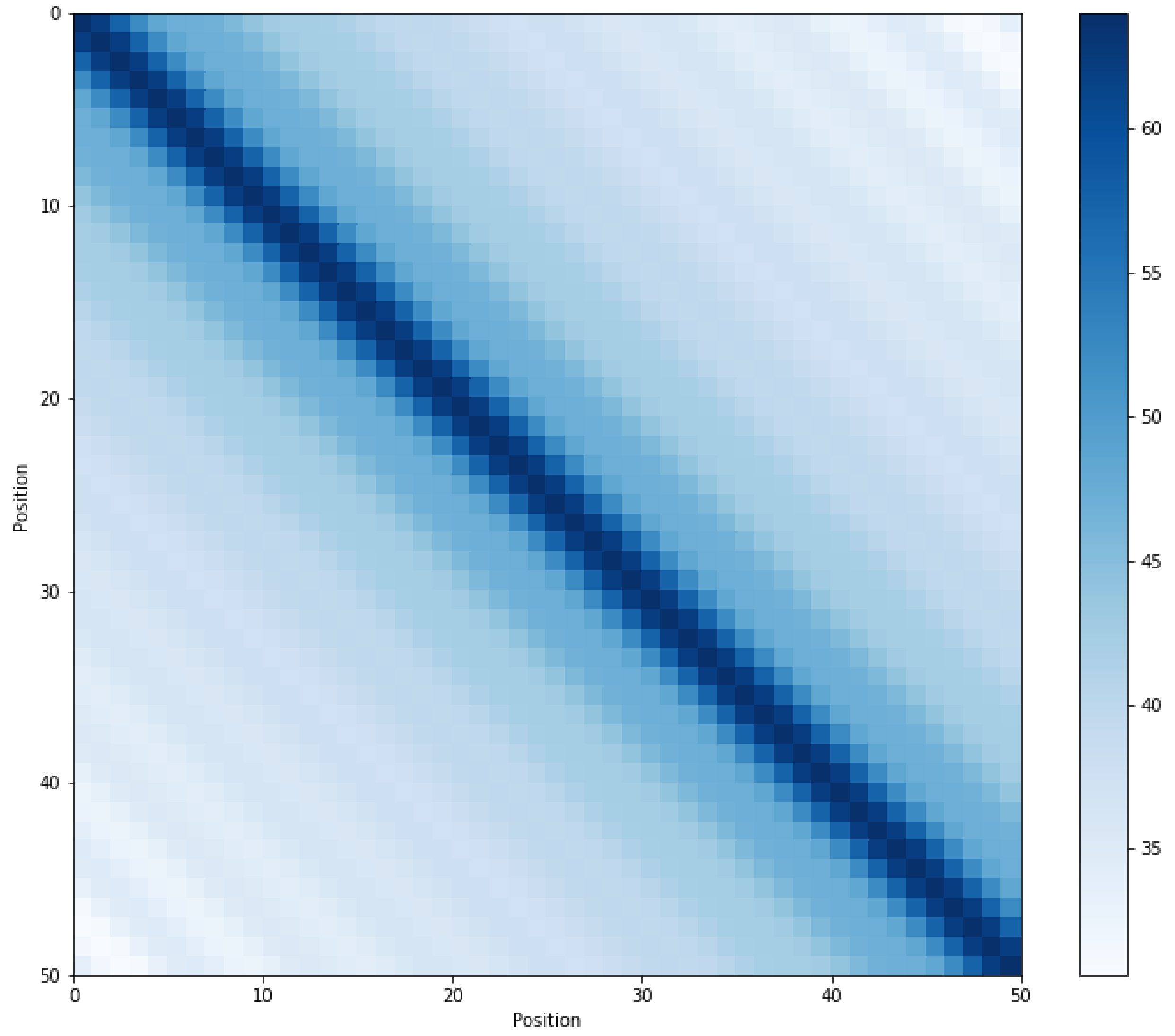
Sinusoidal Positional Encoding

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



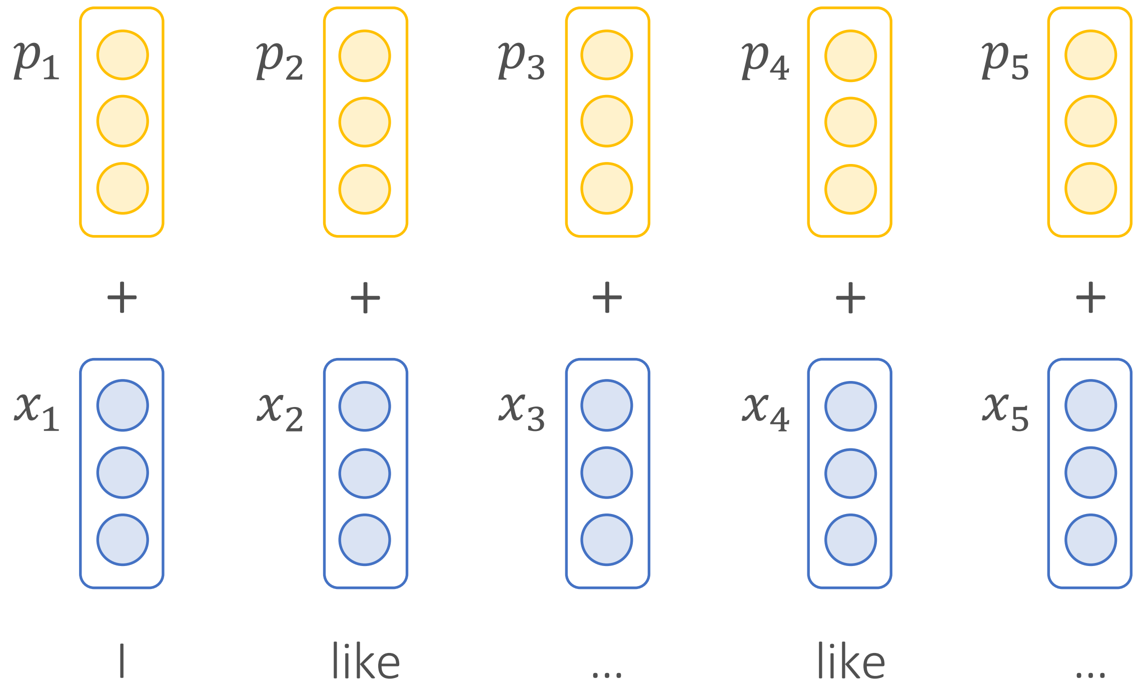
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Sinusoidal Positional Encoding



https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Encoding



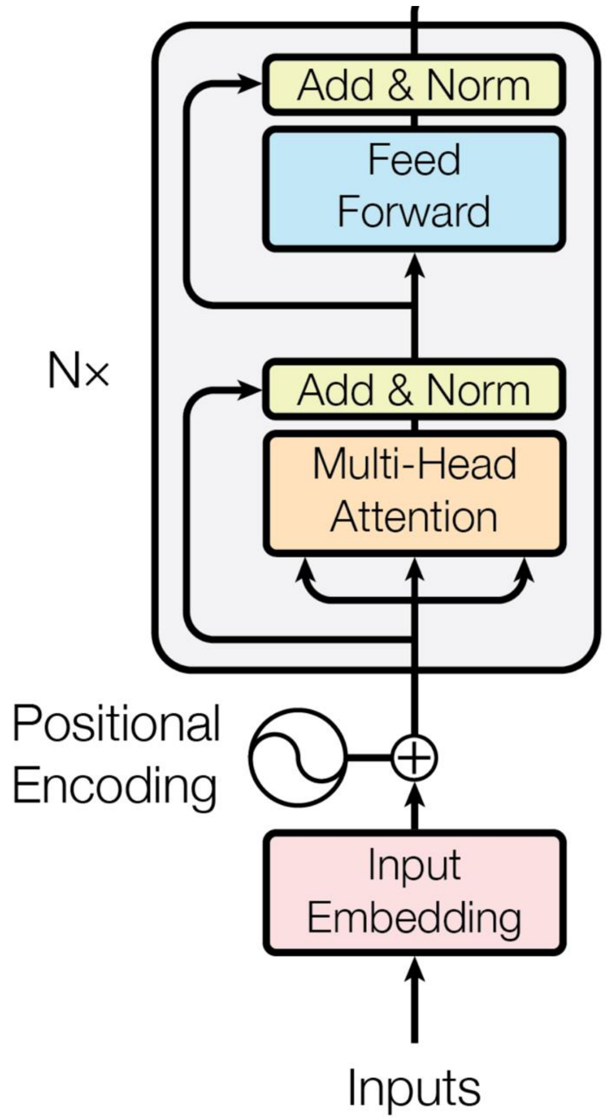
$$\big(E(\mathrm{I}) + PE(1)\big)\big(E(\mathrm{like}) + PE(2)\big) = E(\mathrm{I})E(\mathrm{like}) + \boxed{E(\mathrm{I})PE(2)} + PE(1)E(\mathrm{like}) + \boxed{PE(1)PE(2)}$$

$$\big(E(\mathrm{I}) + PE(1)\big)\big(E(\mathrm{like}) + PE(4)\big) = E(\mathrm{I})E(\mathrm{like}) + \boxed{E(\mathrm{I})PE(4)} + PE(1)E(\mathrm{like}) + \boxed{PE(1)PE(4)}$$

In expectation, they are the same

Position difference

Transformer with Positional Encoding


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs