CSCE 689: Special Topics in Trustworthy NLP

Lecture 5: Natural Language Processing Basics (4)

Kuan-Hao Huang
khhuang@tamu.edu

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

Presentation Sign-Up

- Sign-up: https://tinyurl.com/34e27fjx
  - Deadline: Friday 8/30 before lecture
- We will decide the assignment during lecture on 8/30

Week	Date	Topic	Paper ID	Paper Title
4	9/13	Adversarial Attacks and Defenses	1	Adversarial Example Generation with Syntactically Controlled Paraphrase Networks, NAACL 2018
4	9/13	Adversarial Attacks and Defenses	2	Jailbreaking Black Box Large Language Models in Twenty Queries, arXiv 2023
5	9/20	Backdoor Attacks and Data Poisoning	3	Poison Attacks against Text Datasets with Conditional Adversarially Regularized Autoencoder, EMNLP-Findings 2020
5	9/20	Backdoor Attacks and Data Poisoning	4	RAP: Robustness-Aware Perturbations for Defending against Backdoor Attacks on NLP Models, EMNLP 2021
6	9/27	AI-Generated Text Detection	5	RADAR: Robust AI-Text Detection via Adversarial Learning, NeurIPS 2023
6	9/27	AI-Generated Text Detection	6	Paraphrasing Evades Detectors of AI-Generated Text, But Retrieval is An Effective Defense, NeurIPS 2023
7	10/4	Model Uncertainty	7	Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation, ICLR 2023
7	10/4	Model Uncertainty	8	Decomposing Uncertainty for Large Language Models through Input Clarification Ensembling, ICML 2024
9	10/18	Model Explainability and Interpretability	9	Reframing Human-AI Collaboration for Generating Free-Text Explanations, NAACL 2022
9	10/18	Model Explainability and Interpretability	10	Self-Consistency Improves Chain of Thought Reasoning in Lang
10	10/25	Bias Detection and Mitigation	11	Null It Out: Guarding Protected Attributes by Iterative Nullspace
10	10/25	Bias Detection and Mitigation	12	From Pretraining Data to Language Models to Downstream Task
11	11/1	Human Preference Alignment	13	SimPO: Simple Preference Optimization with a Reference-Free
11	11/1	Human Preference Alignment	14	Self-Play Fine-Tuning Converts Weak Language Models to Stron
12	11/8	Hallucinations and Misinformation Con	15	SAC3: Reliable Hallucination Detection in Black-Box Language
12	11/8	Hallucinations and Misinformation Con	16	Characterizing Truthfulness in Large Language Model Generatio
13	11/15	Robustness of Multimodal Models	17	Robust CLIP: Unsupervised Adversarial Fine-Tuning of Vision E
13	11/15	Robustness of Multimodal Models	18	On the Robustness of Large Multimodal Models Against Image A
14	11/18	Robustness of Multimodal Models	19	CleanCLIP: Mitigating Data Poisoning Attacks in Multimodal Cor
14	11/18	Robustness of Multimodal Models	20	Eyes Wide Shut? Exploring the Visual Shortcomings of Multimod

Name	Preference 1	Preference 2	Preference 3	Preference 4	Preference 5	Preference 6	Preference 7	Preference 8
Agrawal, Saransh								
Baid, Rahul								
Bajaj, Divij								
Balasubramanian, Sriram								
Harden, Dylan								
Hu, Chan-Wei								
Lee, Jaehoon								
Liu, Junru								
Rajagopalan, Vicram								
Samudra, Arunim Chaitanya								
Please fill in paper IDs								

Lecture Plan

- Natural Language Processing Basics
- Long Short-Term Memory (LSTM) for generation
- Attention mechanism
- Transformers

Recap: Convolutional Neural Network (CNN)

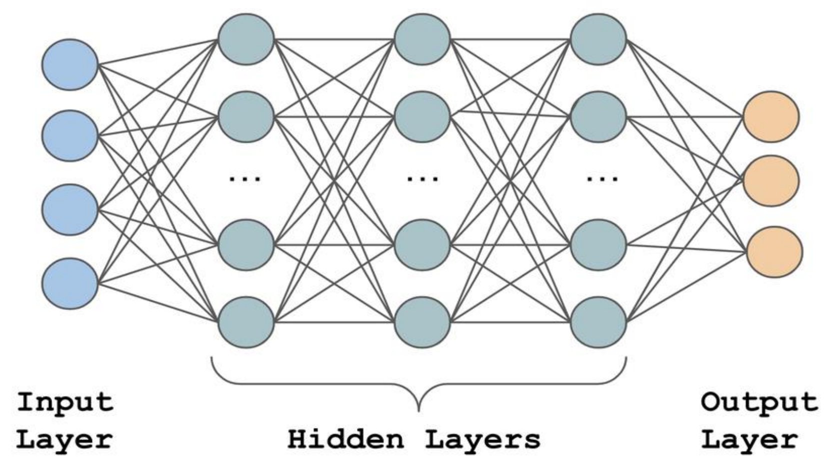
Filter Size = 3 $\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ \cdots & \cdots & \cdots \\ w_{4,1} & w_{4,2} & w_{4,3} \end{bmatrix}$ Filter Size = 2 $\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ \cdots & \cdots \\ w_{4,1} & w_{4,2} \end{bmatrix}$ Filter Size = 4 $\mathbf{W} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,4} \\ \cdots & \cdots & \cdots \\ w_{4,1} & \cdots & w_{4,4} \end{bmatrix}$

	Alice	treats	Bob	well
Dimension 1	0.7	2.7	-0.1	-5.7
Dimension 2	8.6	-3.9	6.7	-9.8
Dimension 3	-2.4	-5.6	1.5	-1.6
Dimension 4	2.3	1.1	2.0	-1.0

$w_{1,1}$*0.7 + $w_{1,2}$*2.7

$w_{2,1}$*8.6 + $w_{2,2}$*-3.9 + $w_{2,3}$*6.7 + $w_{2,4}$*-9.8

v v v



$$\text{Loss} = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

Recap: Recurrent Neural Network (RNN)

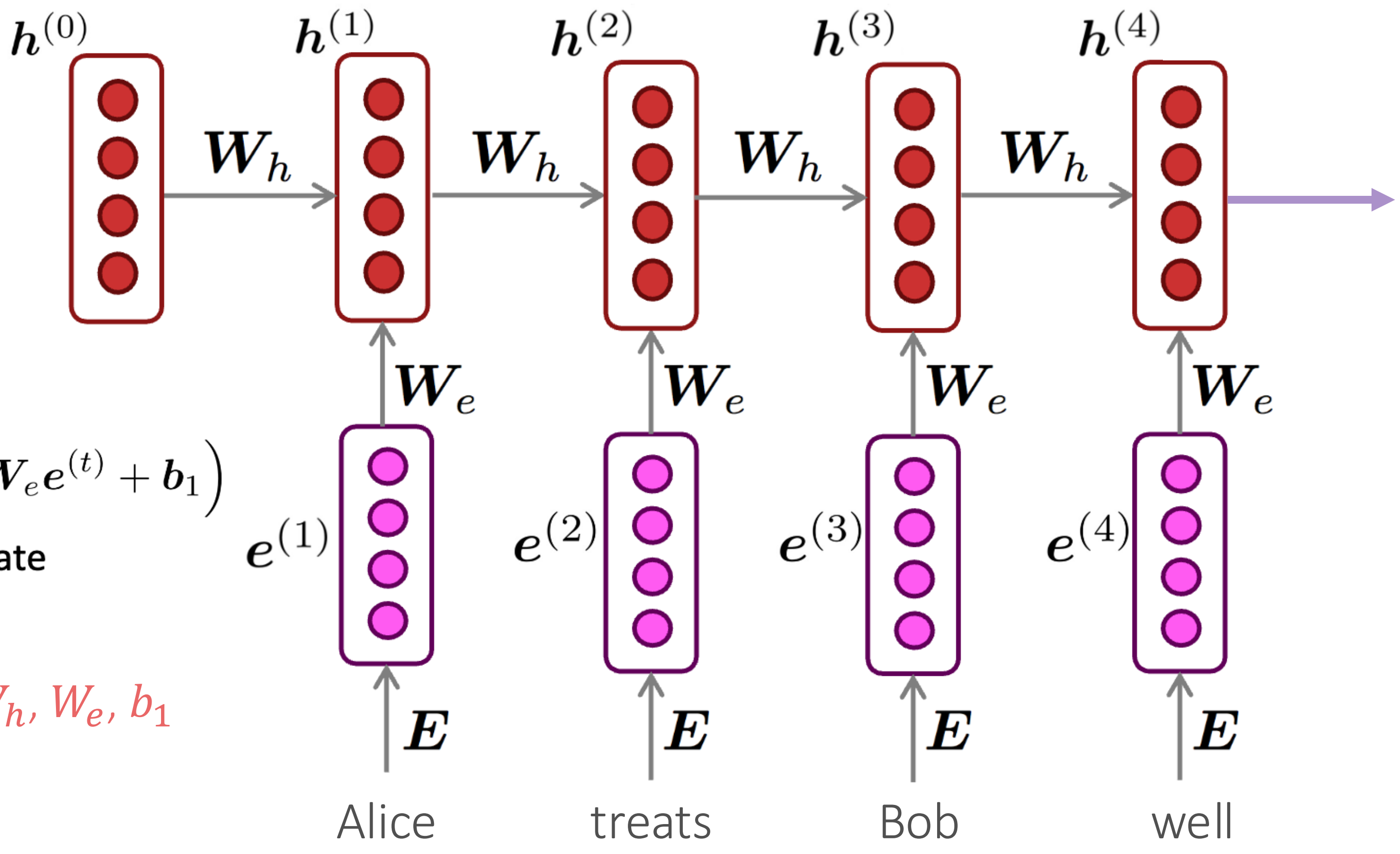
Activation Function
(tanh, sigmoid)

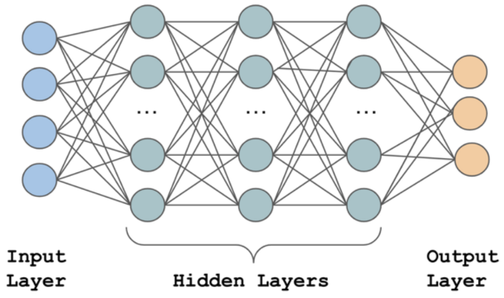
hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e \boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

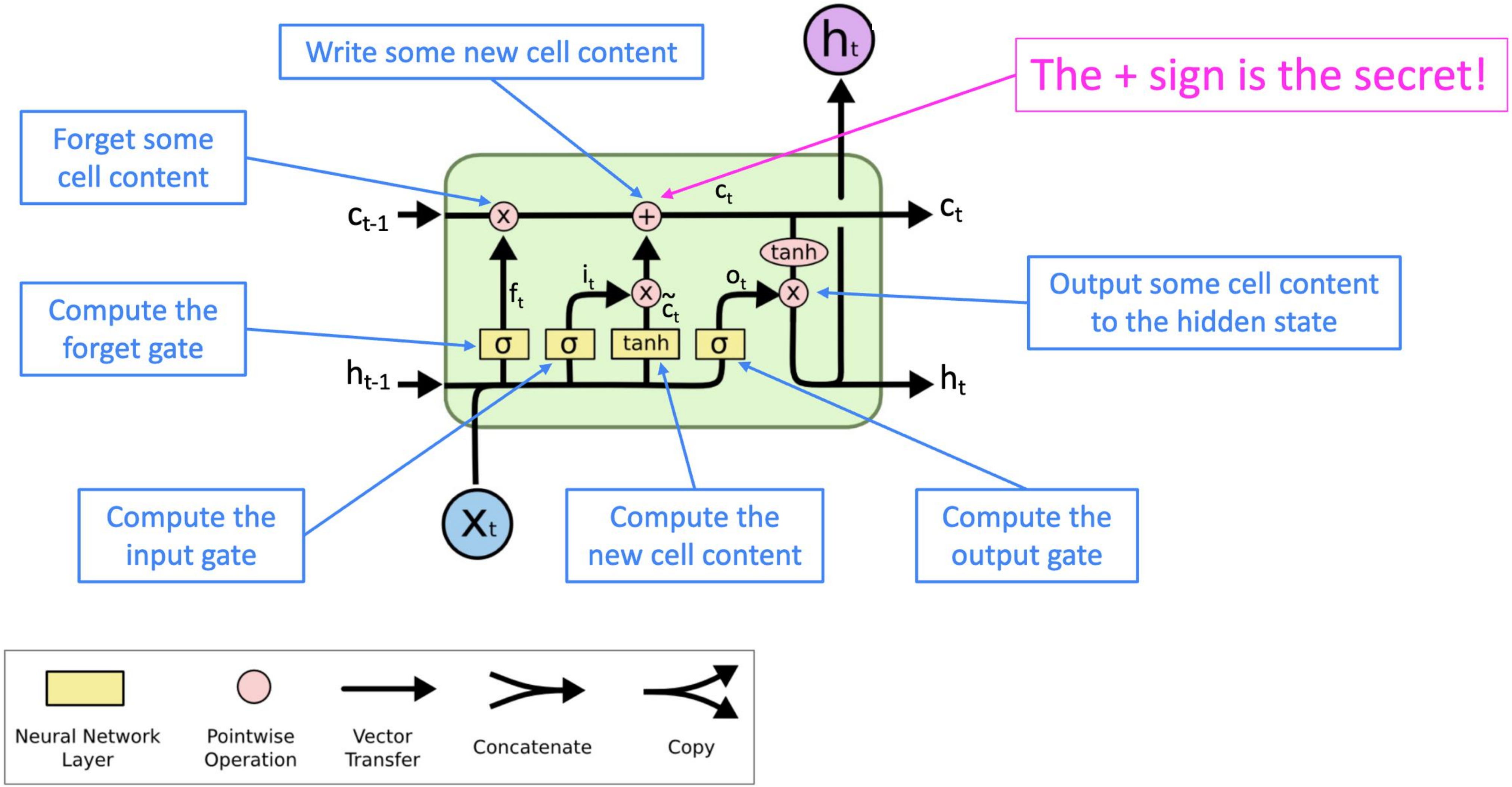
Model Parameters W_h, W_e, b_1



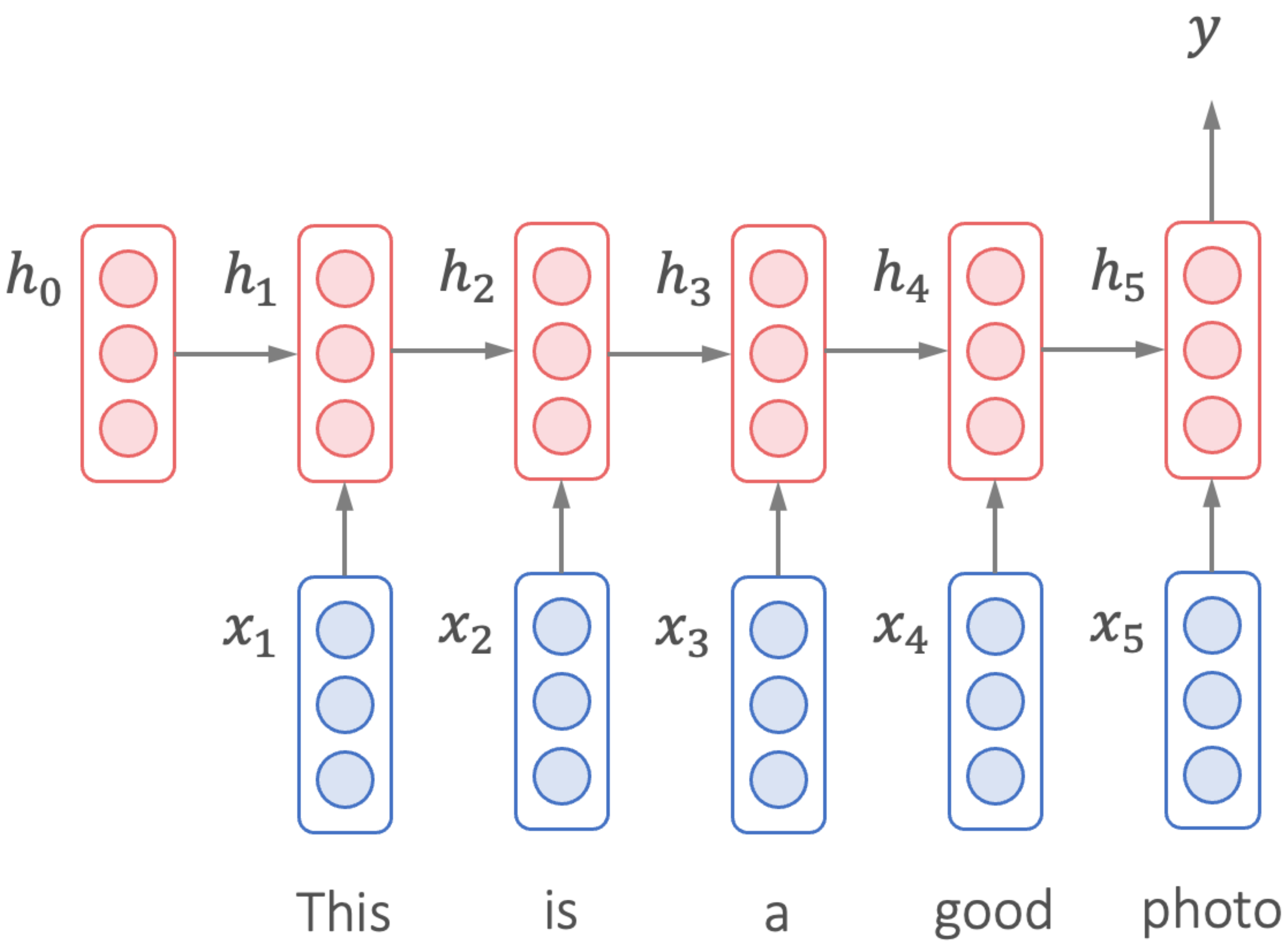


$$\text{Loss} = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

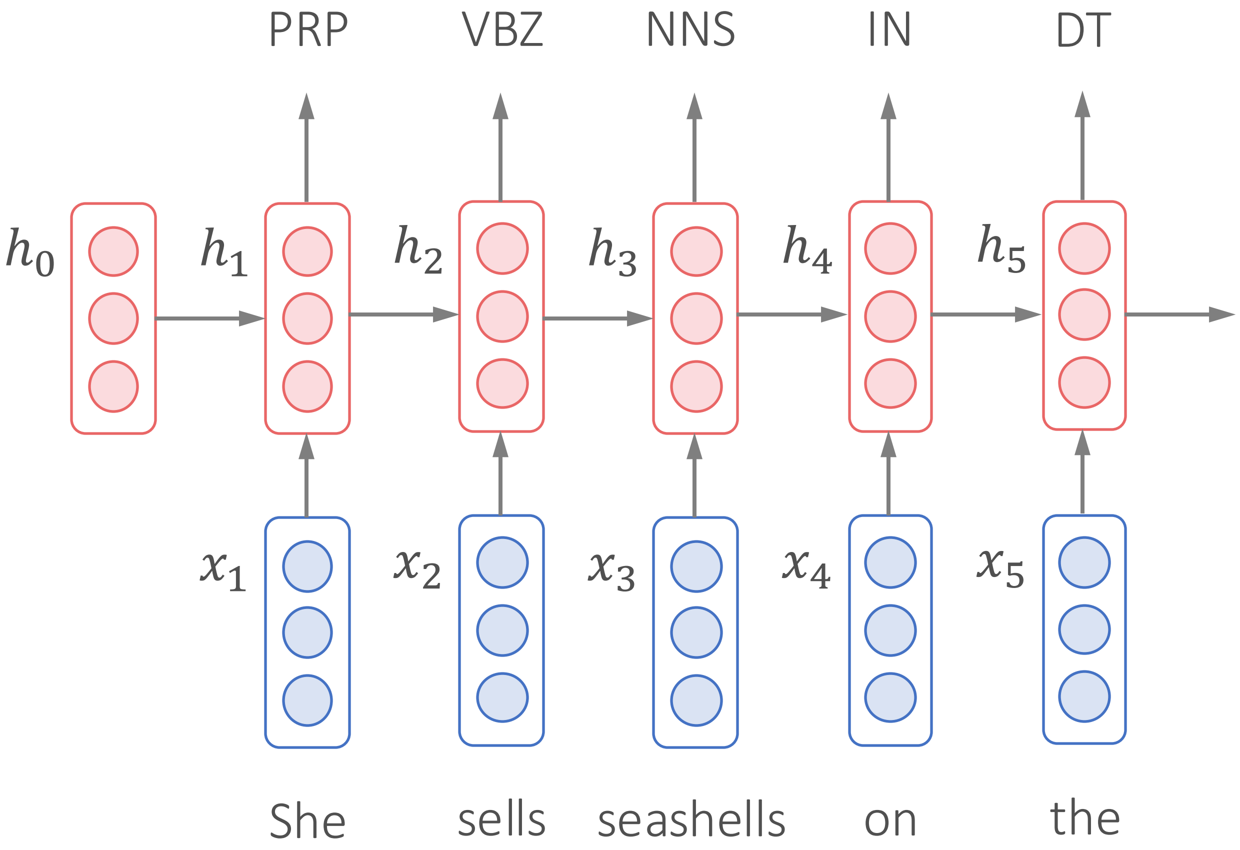
Recap: Long Short-Term Memory (LSTM)



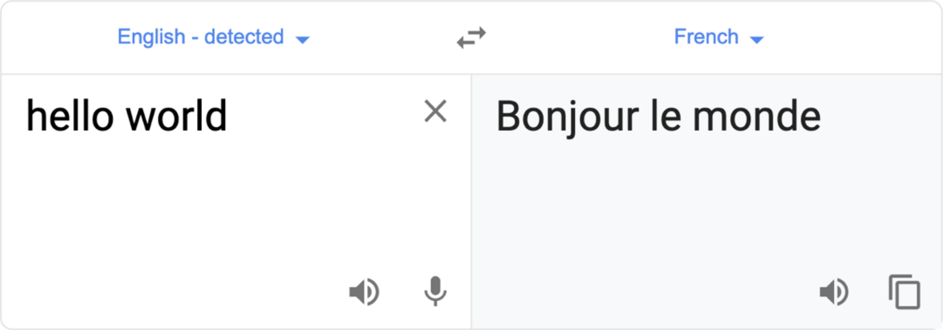
LSTM for Classification



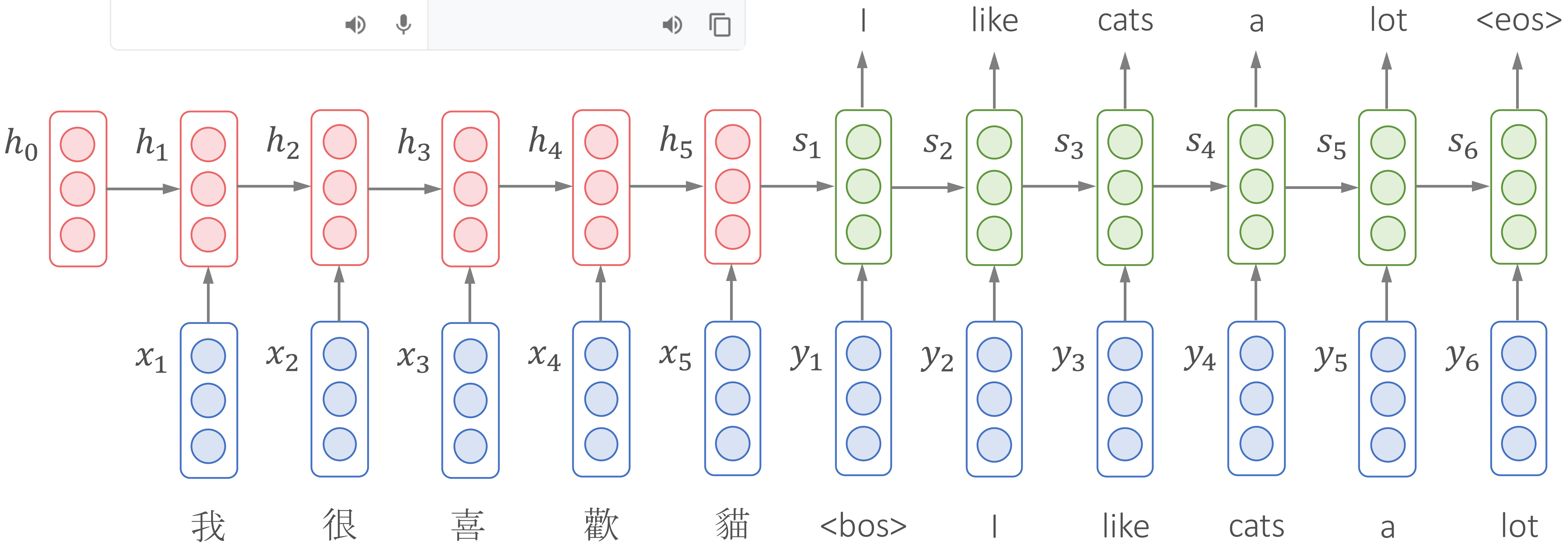
LSTM for Sequential Tagging



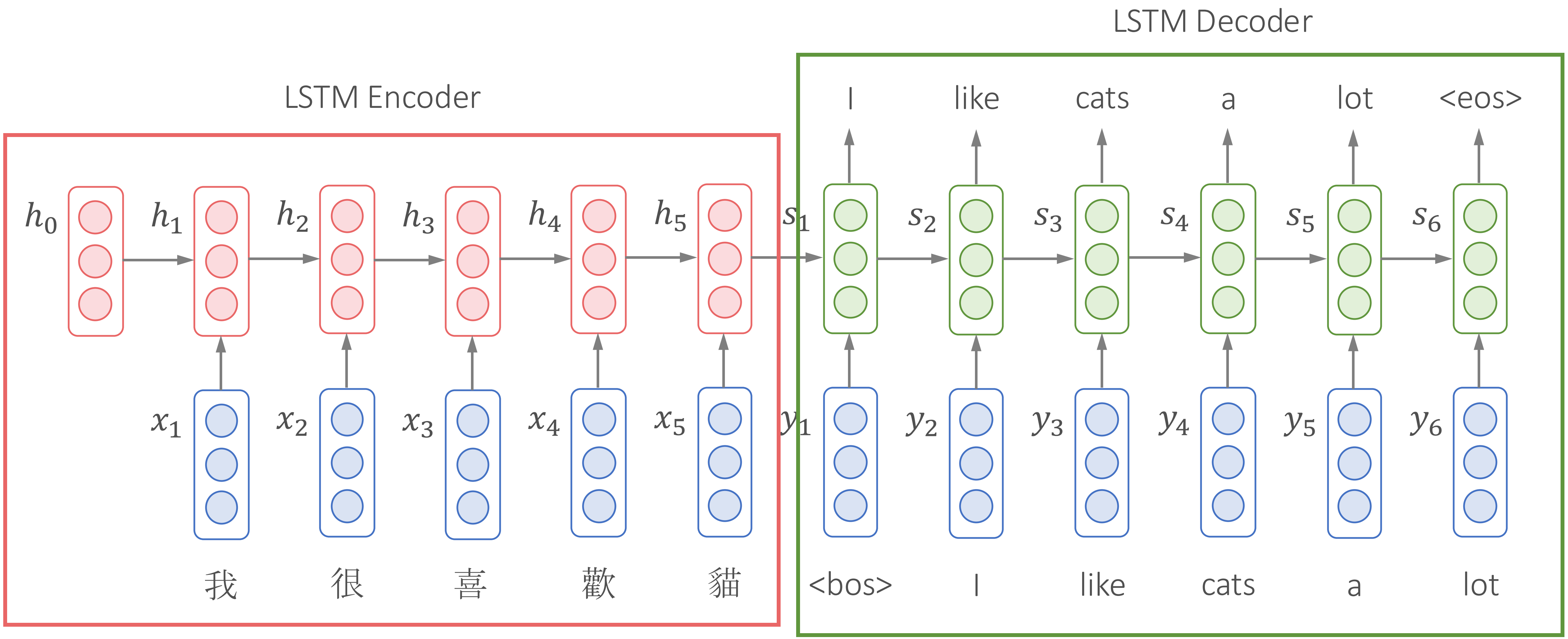
LSTM for Generation



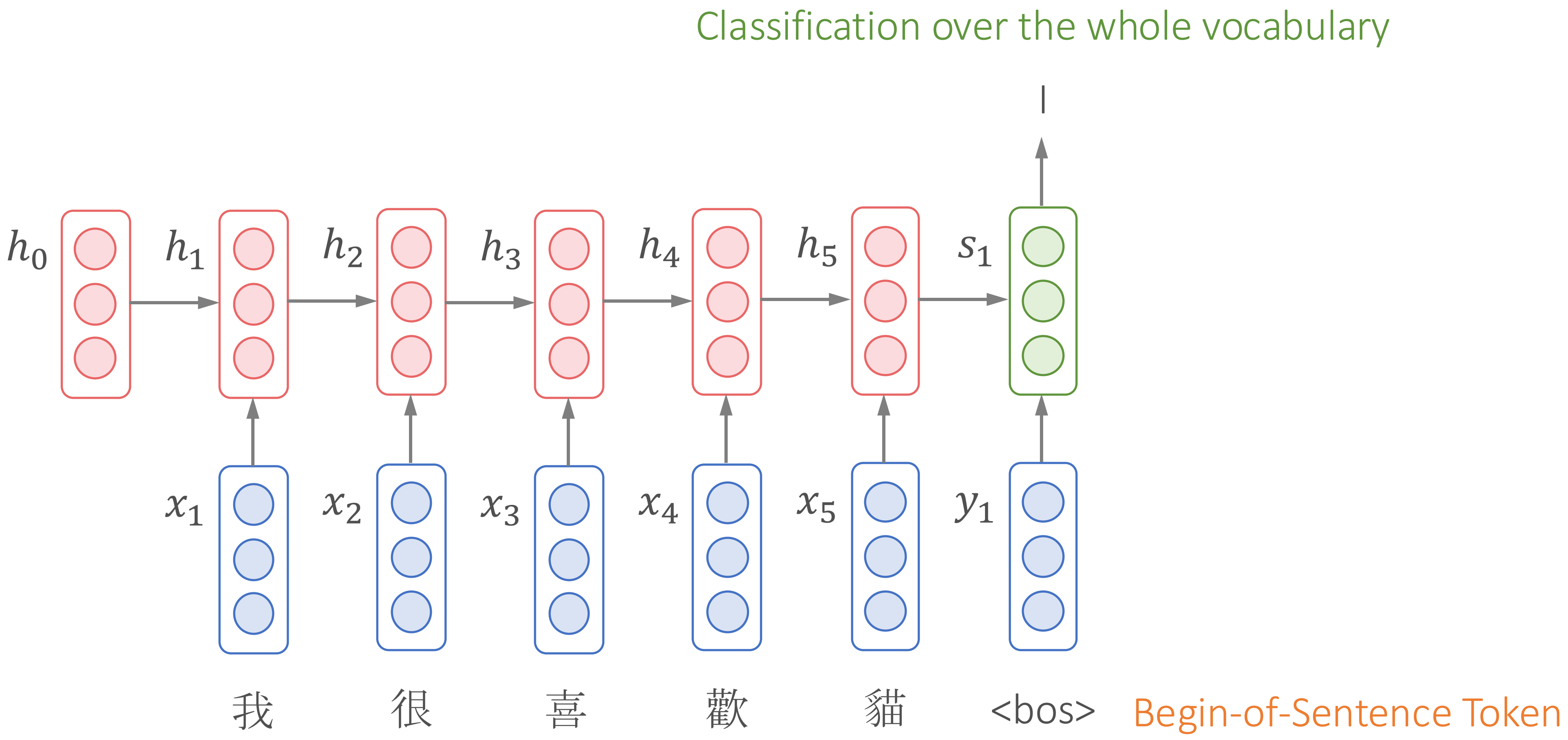
Sequence-to Sequence (Seq2Seq) Models



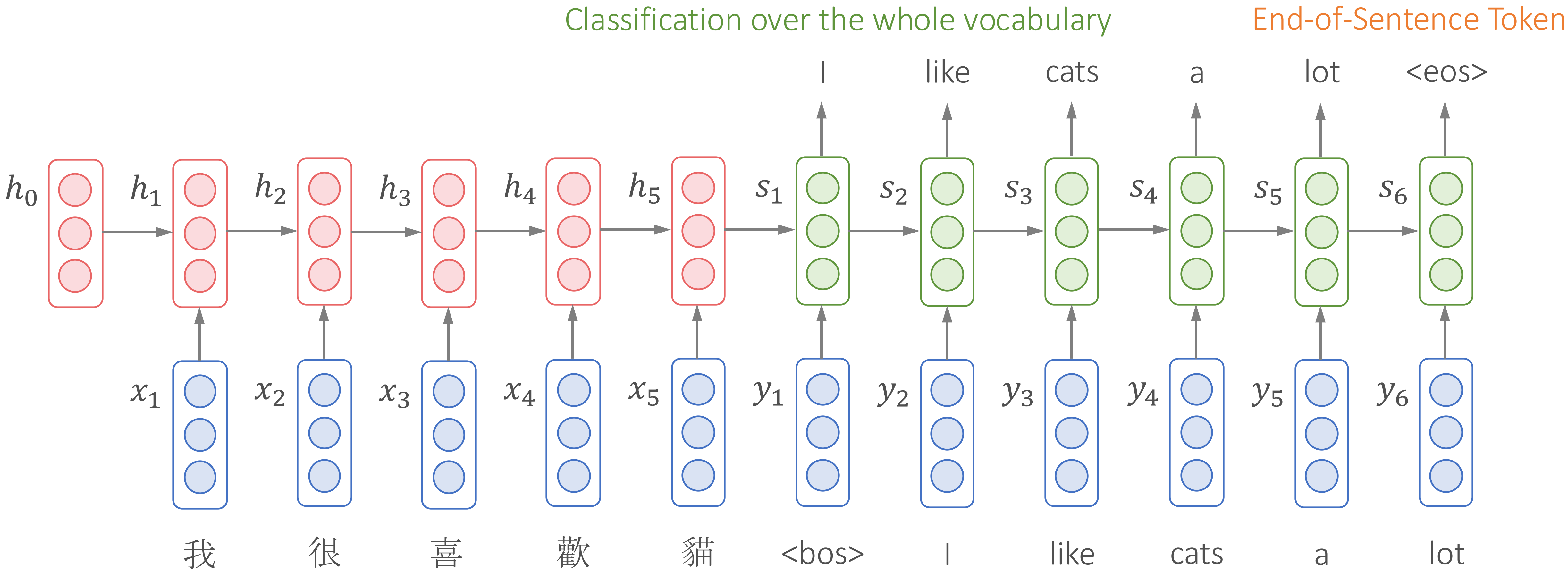
LSTM for Generation



LSTM for Generation

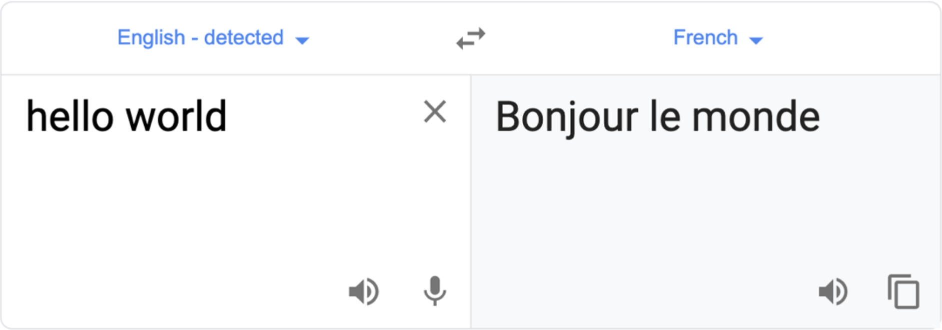


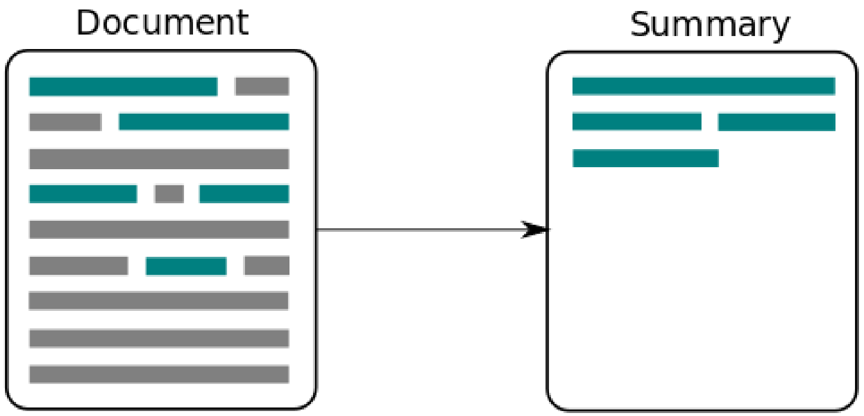
LSTM for Generation

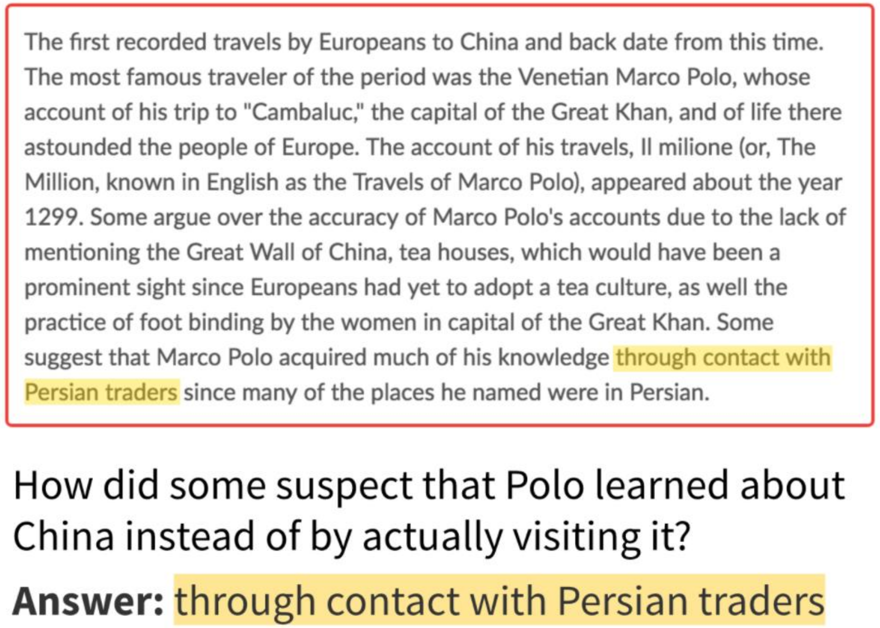


LSTM for Generation

- Seq2Seq tasks are everywhere






The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

How did some suspect that Polo learned about China instead of by actually visiting it?

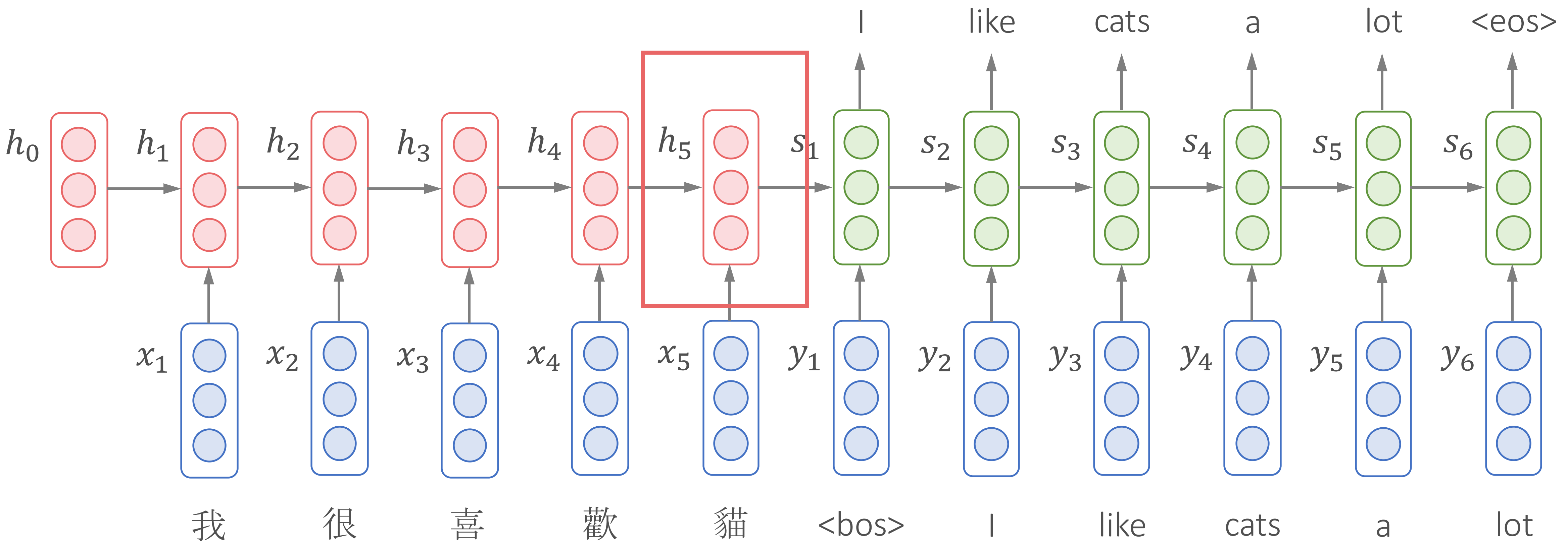
Answer: through contact with Persian traders

Lecture Plan

- Natural Language Processing Basics
- Long Short-Term Memory (LSTM) for generation
- Attention mechanism
- Transformers

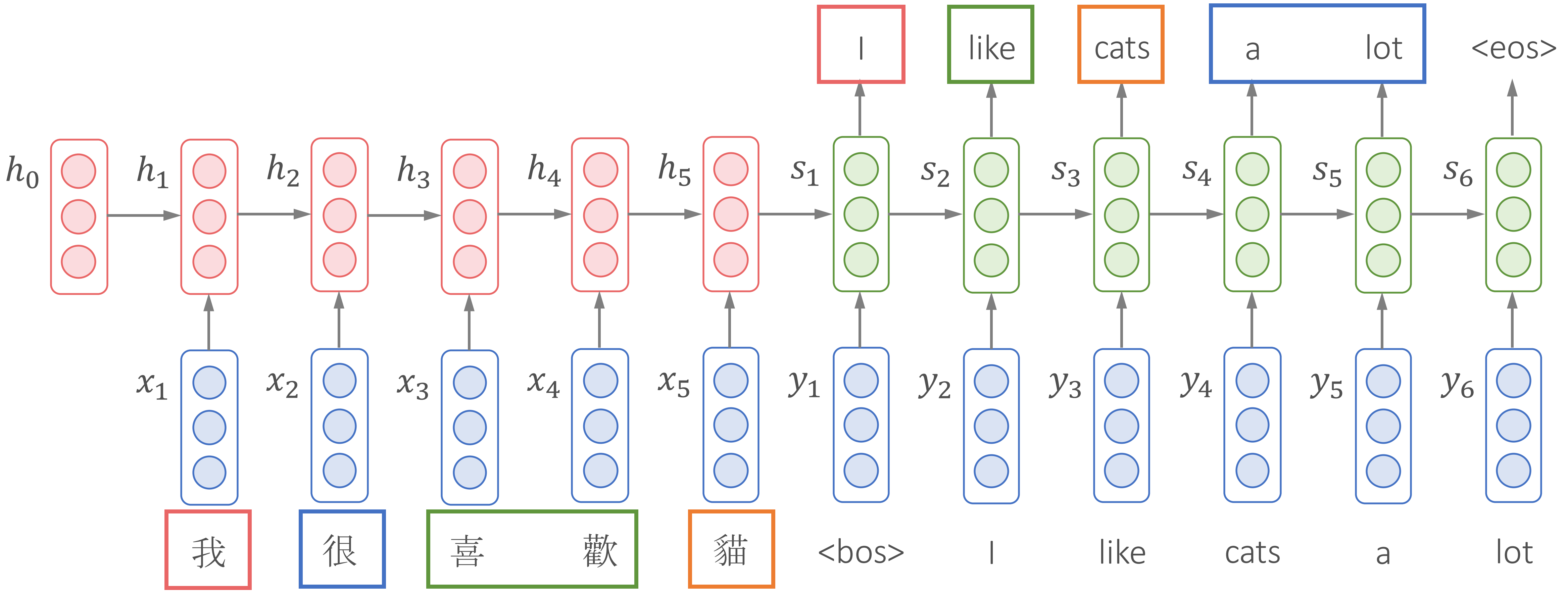
LSTM: Bottleneck

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



LSTM: Focus on A Particular Part When Decoding

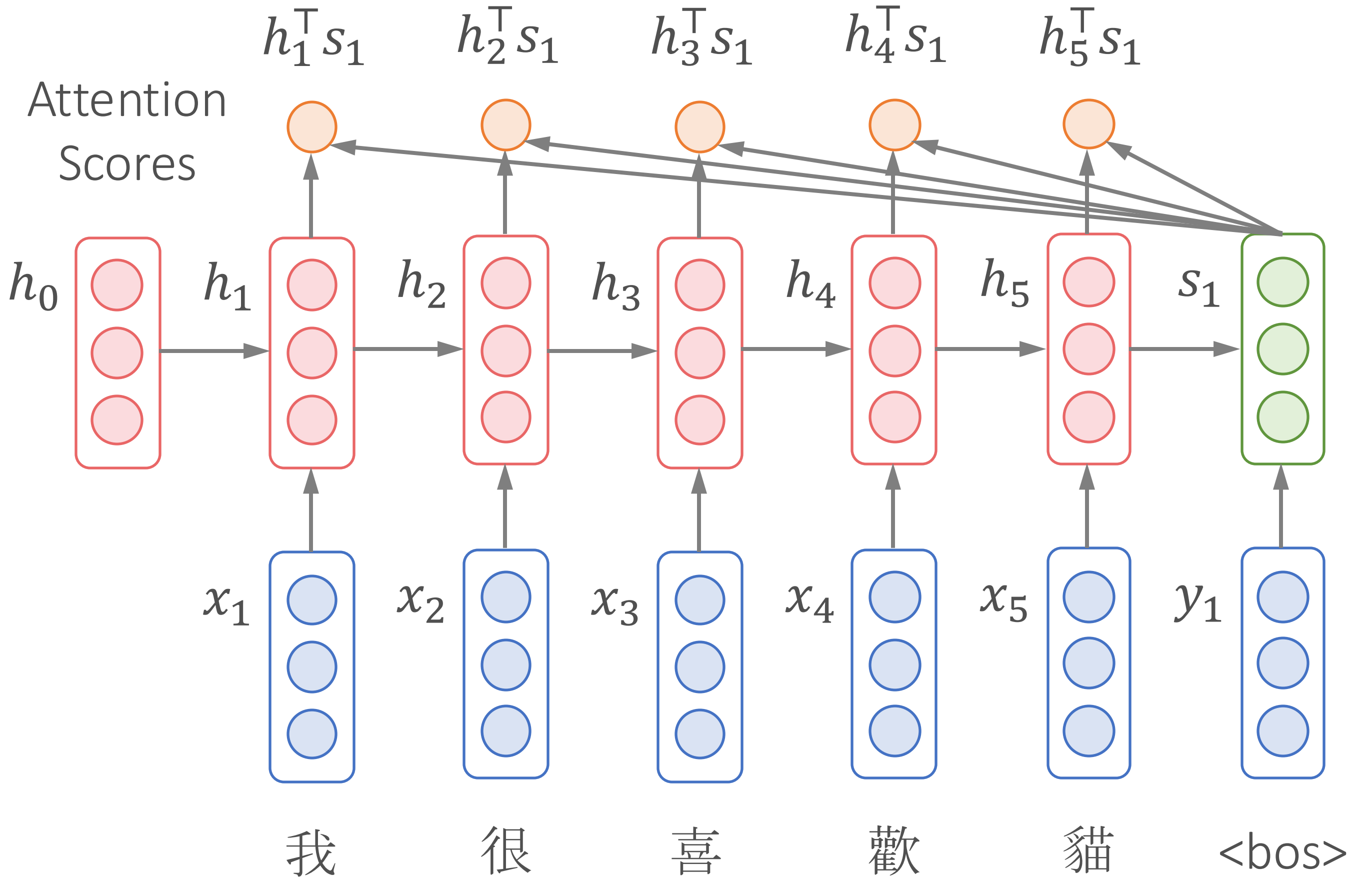
- Each token classification requires different part of information from source sentence



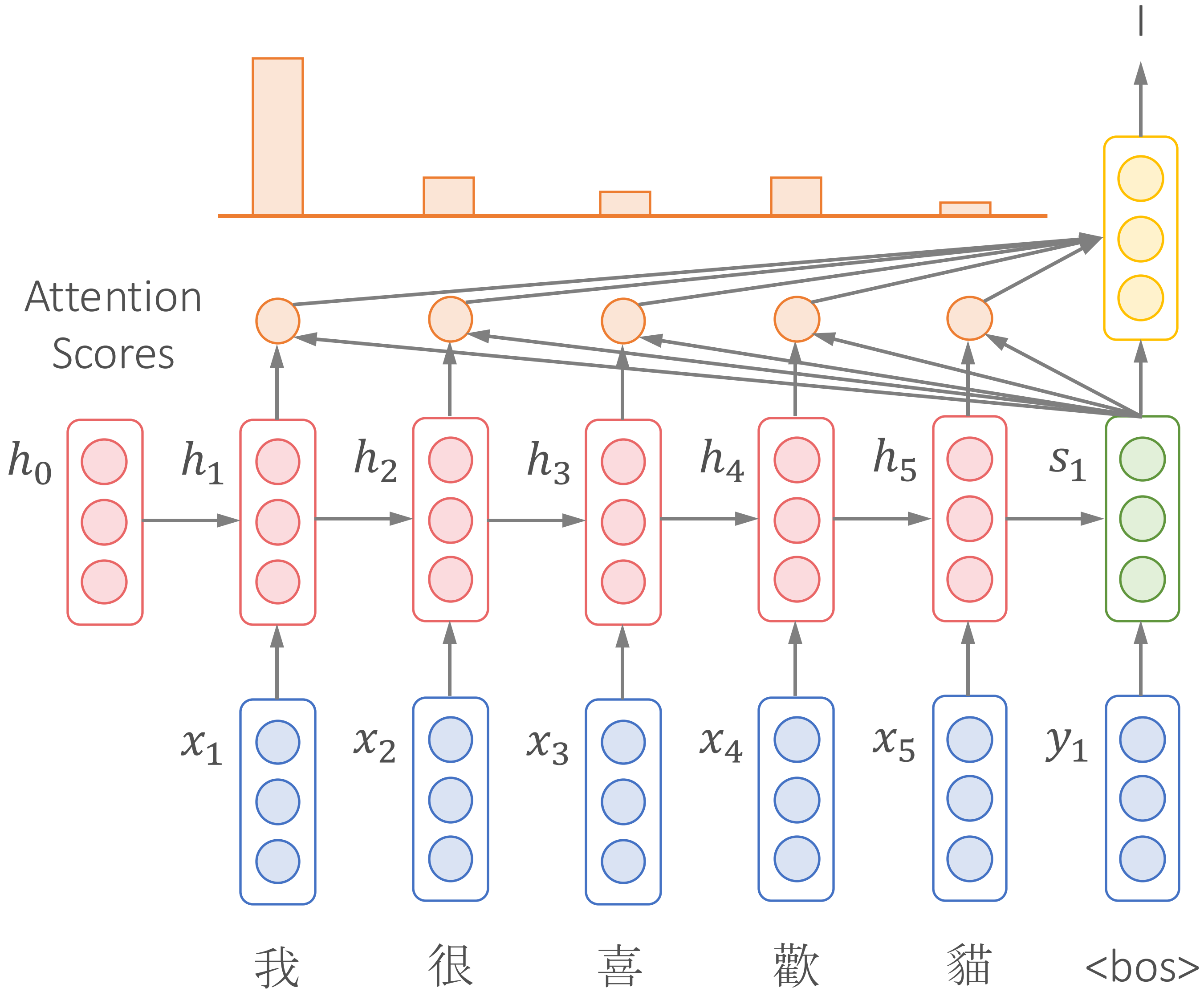
Attention

- Attention provides a solution to the bottleneck problem
- Key idea: At each time step during decoding, focus on a particular part of source sentence

LSTM with Attention



LSTM with Attention



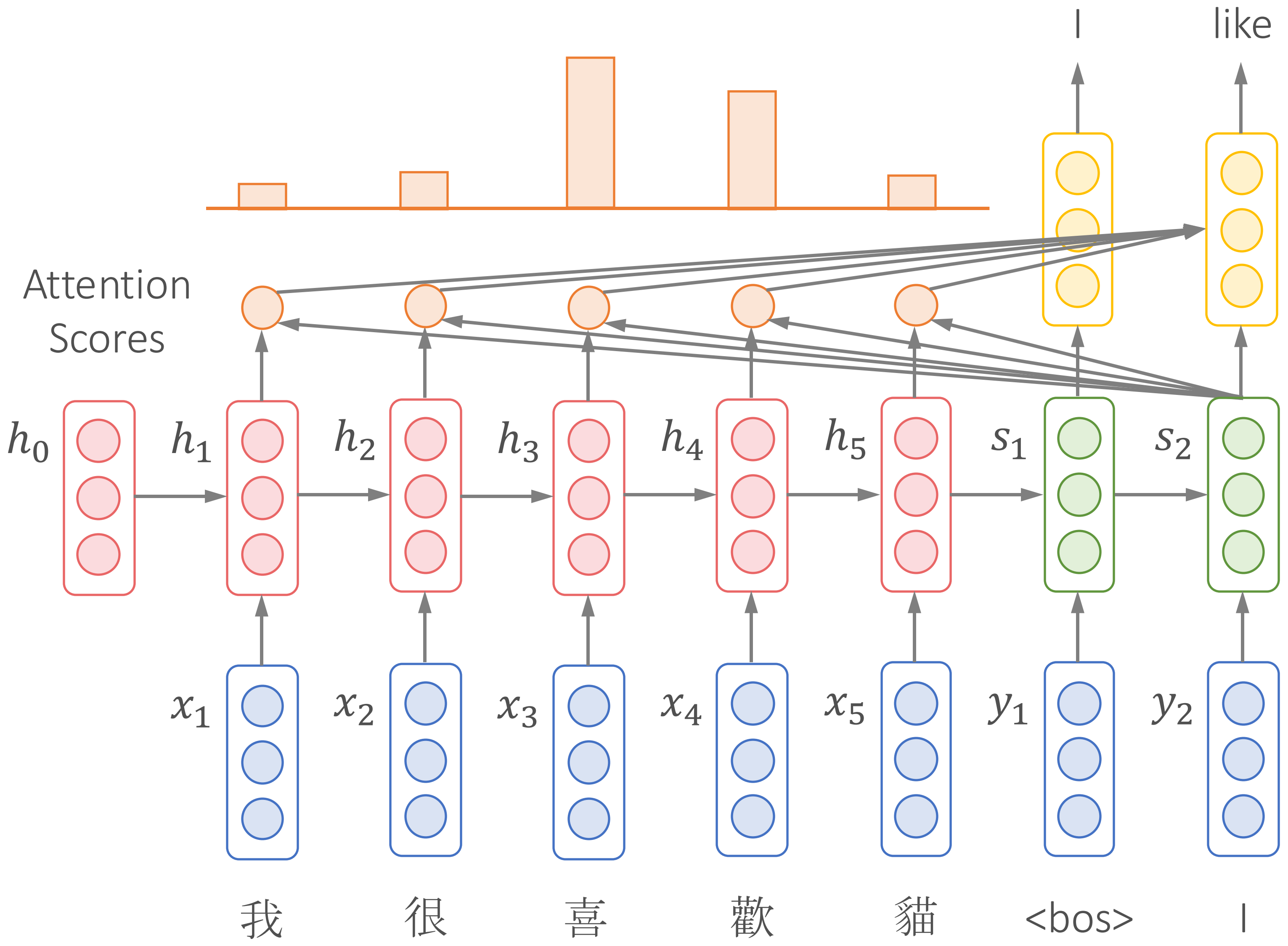
Attention Scores $\alpha_i = h_i^{\mathsf{T}} s_1$

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

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

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

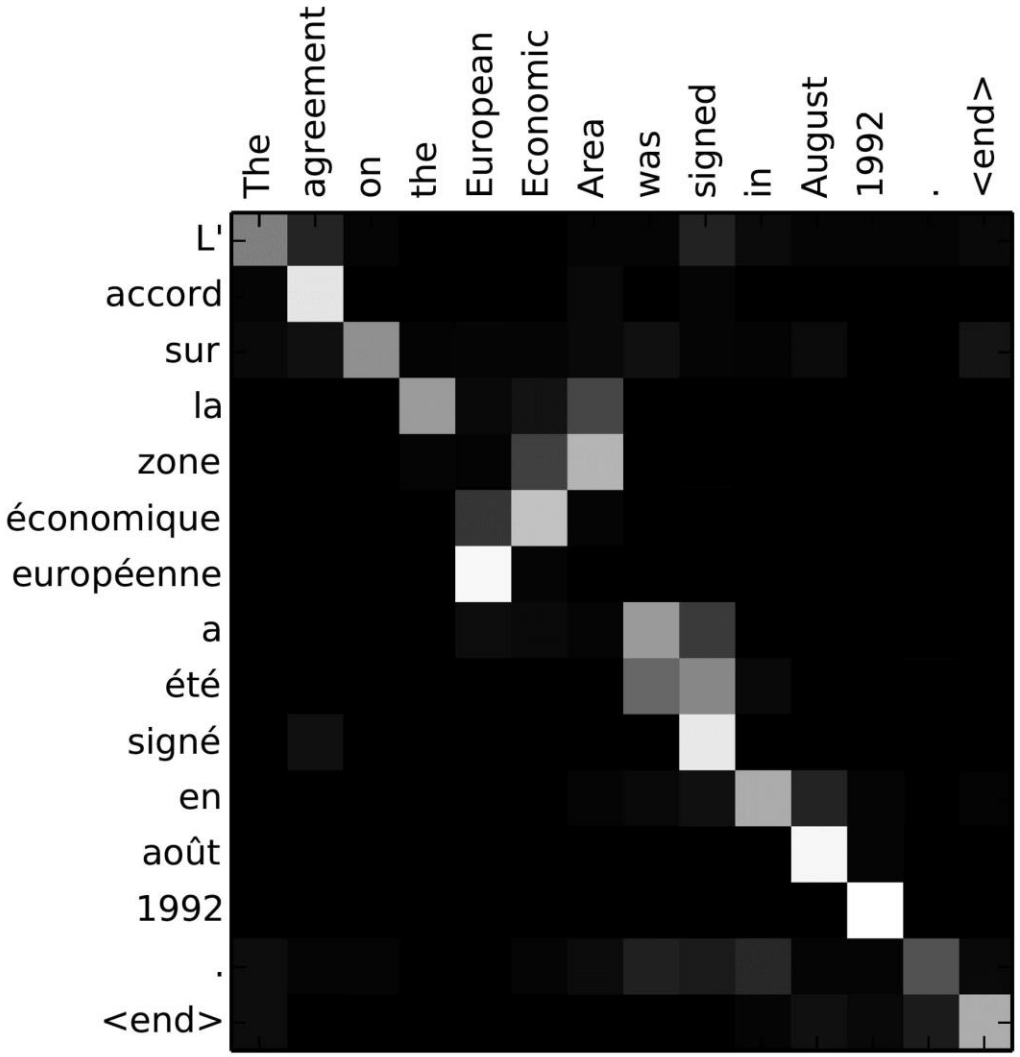
LSTM with Attention



Different Types of Attention

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

Visualization of Attention

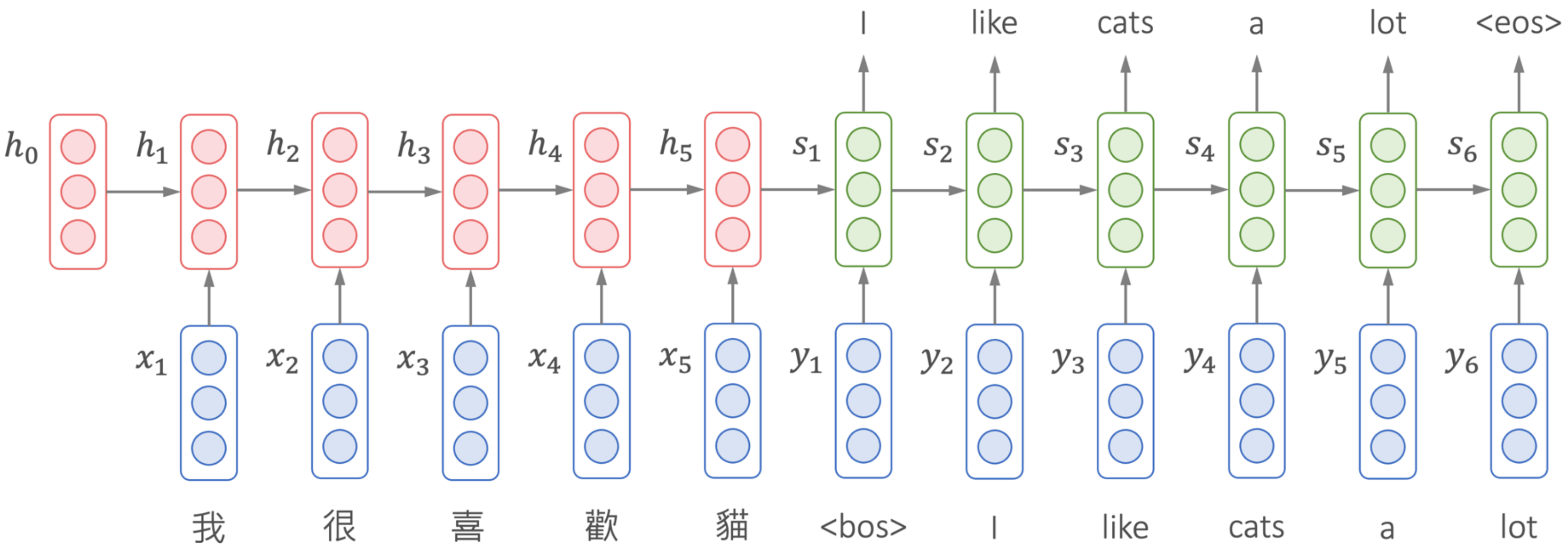


Lecture Plan

- Natural Language Processing Basics
- Long Short-Term Memory (LSTM) for generation
- Attention mechanism
- Transformers

Issues with LSTM

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



Transformers

- Attention Is All You Need, 2017
  - 130K+ citations

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

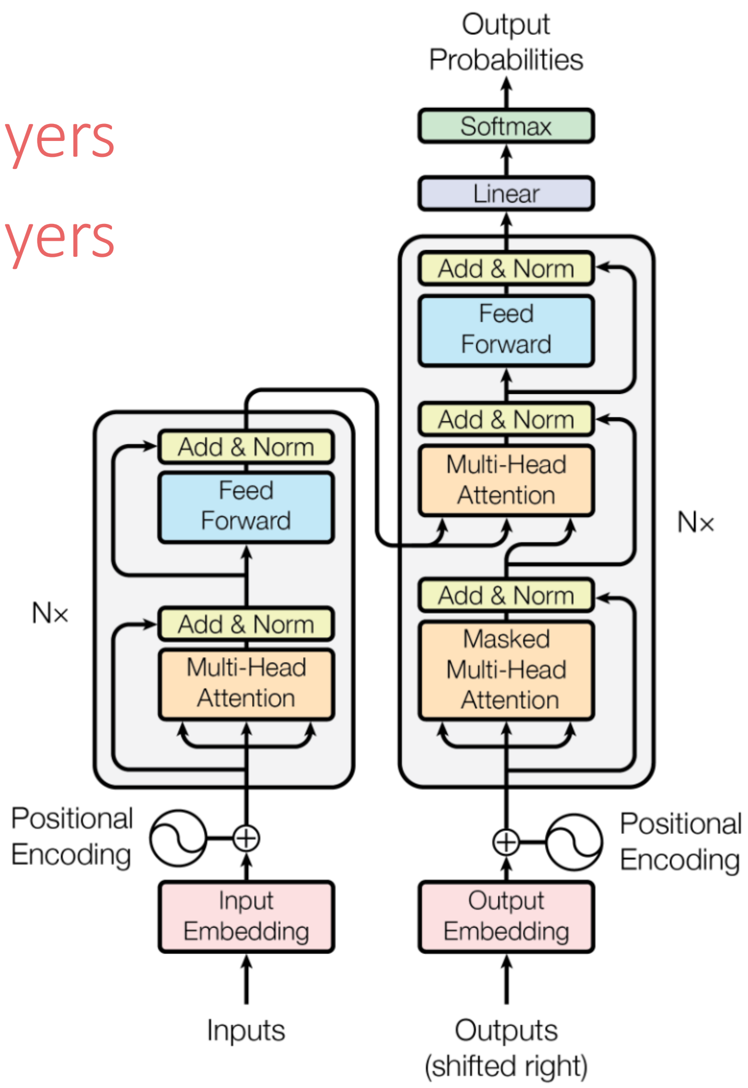
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Transformers for Seq2Seq

- Transformer encoder = a stack of encoder layers
- Transformer decoder = a stack of decoder layers
- No any recurrence structures
  - Easy to parallelize

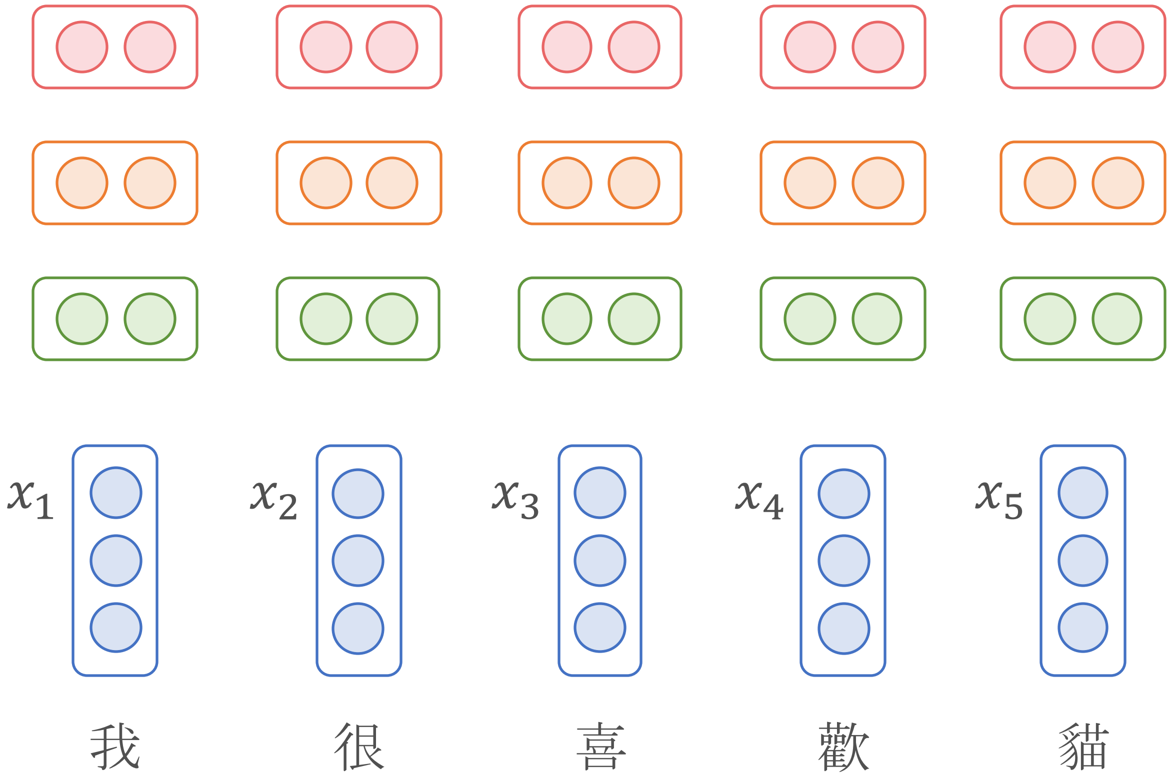


Transformer Layer: Self-Attention

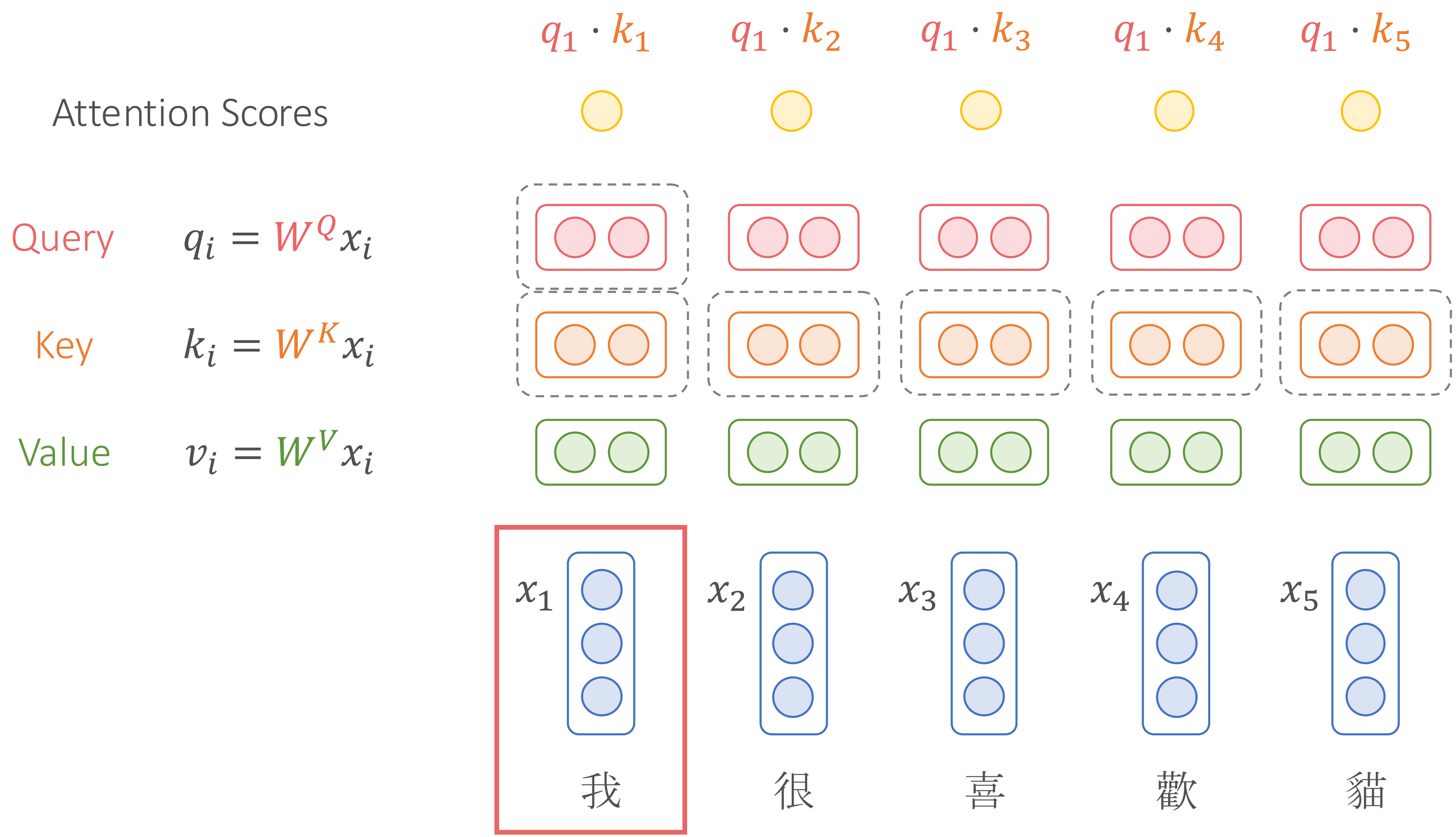
Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



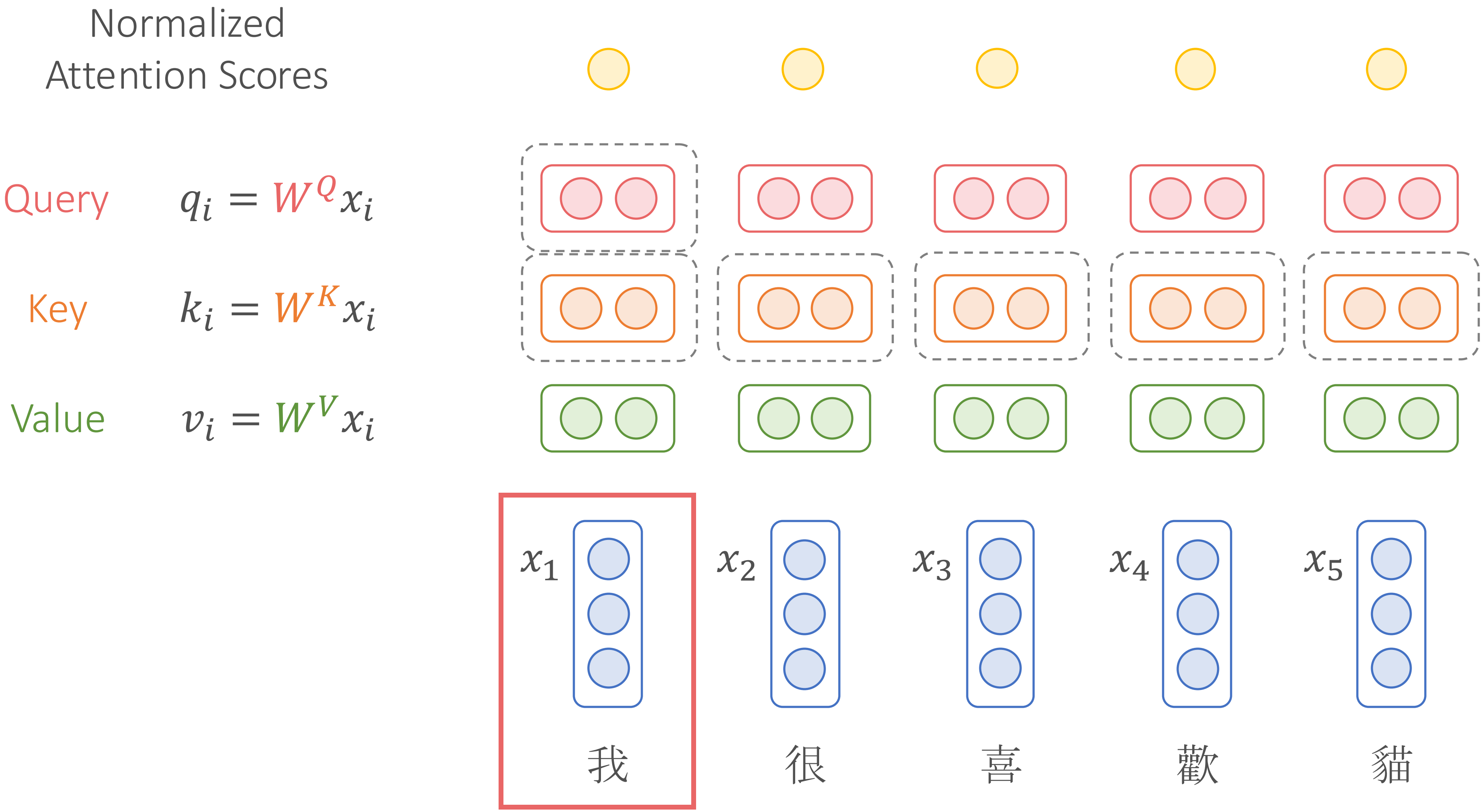
Transformer Layer: Self-Attention



Transformer Layer: Self-Attention

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

Vector dimension



Transformer Layer: Self-Attention

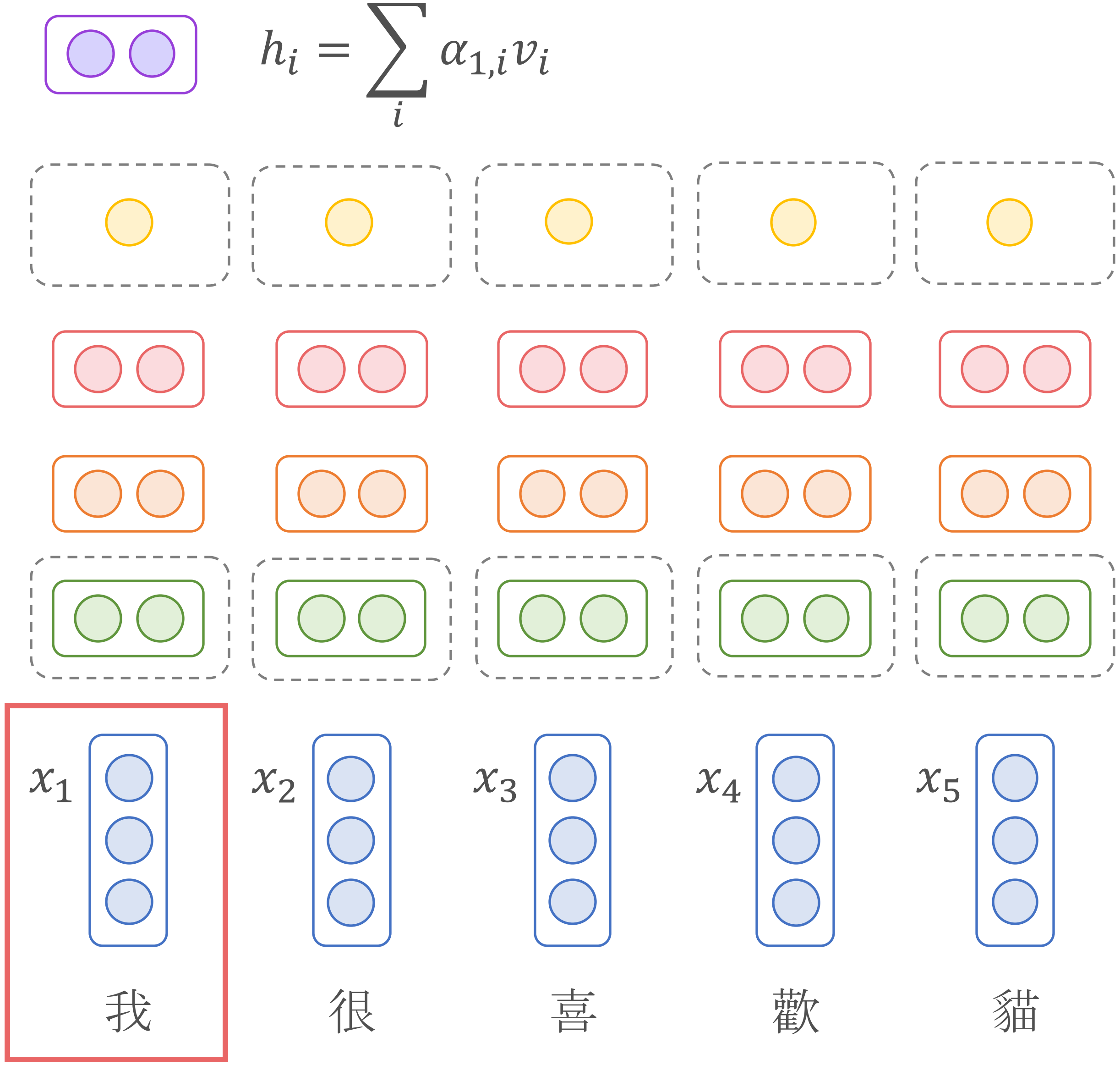
Weighted Sum $h_i = \sum_i \alpha_{1,i} v_i$

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



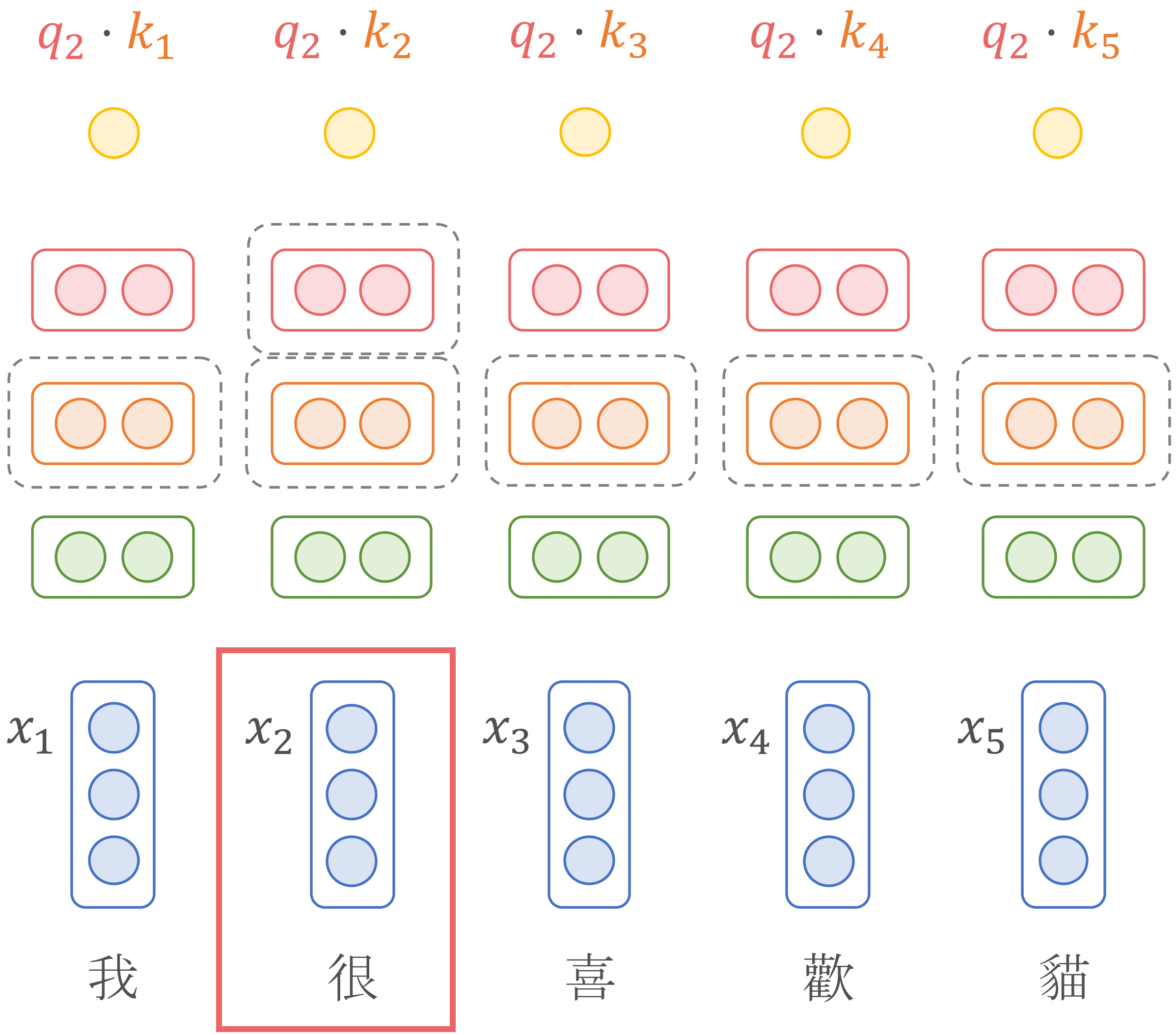
Transformer Layer: Self-Attention

Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Transformer Layer: Self-Attention

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

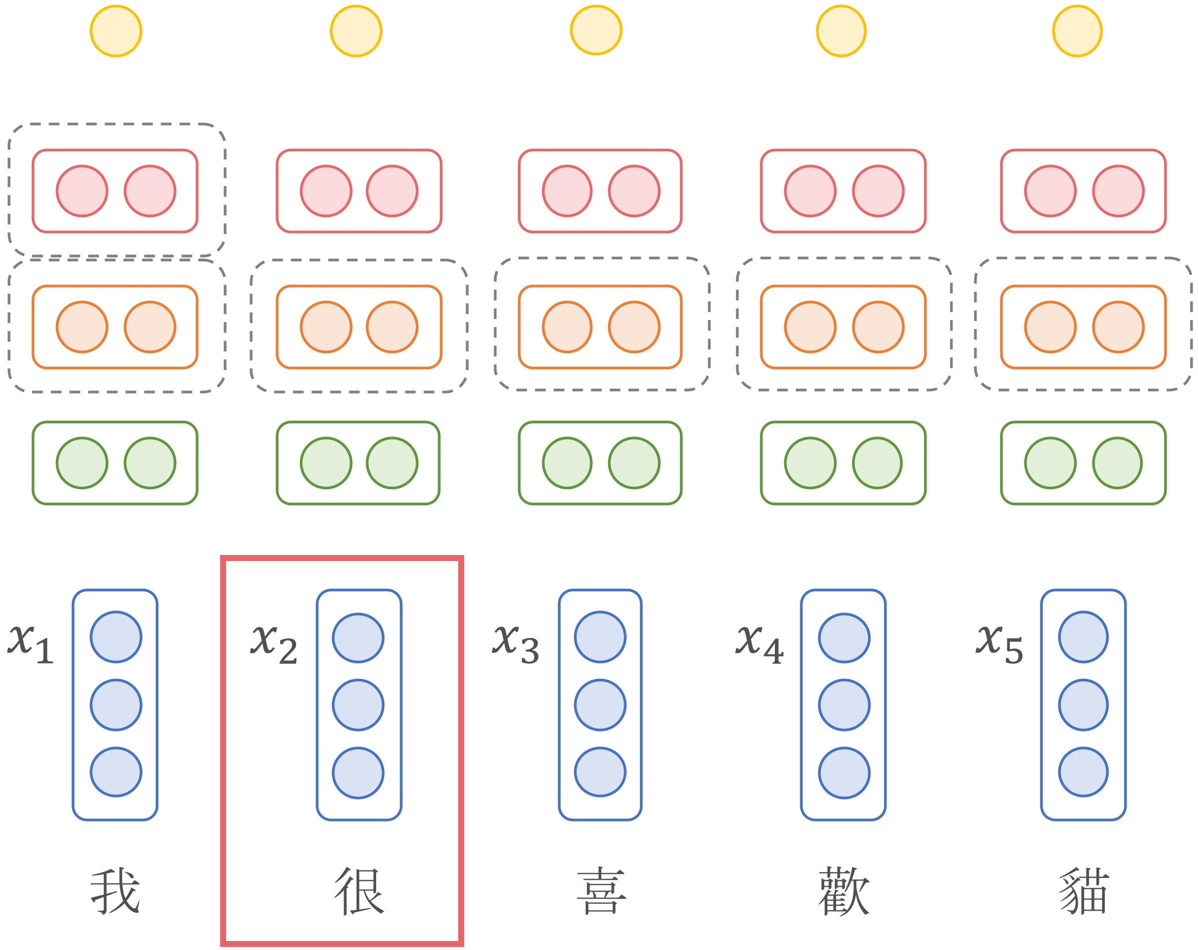
Vector dimension

Normalized
Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



Transformer Layer: Self-Attention

Weighted Sum

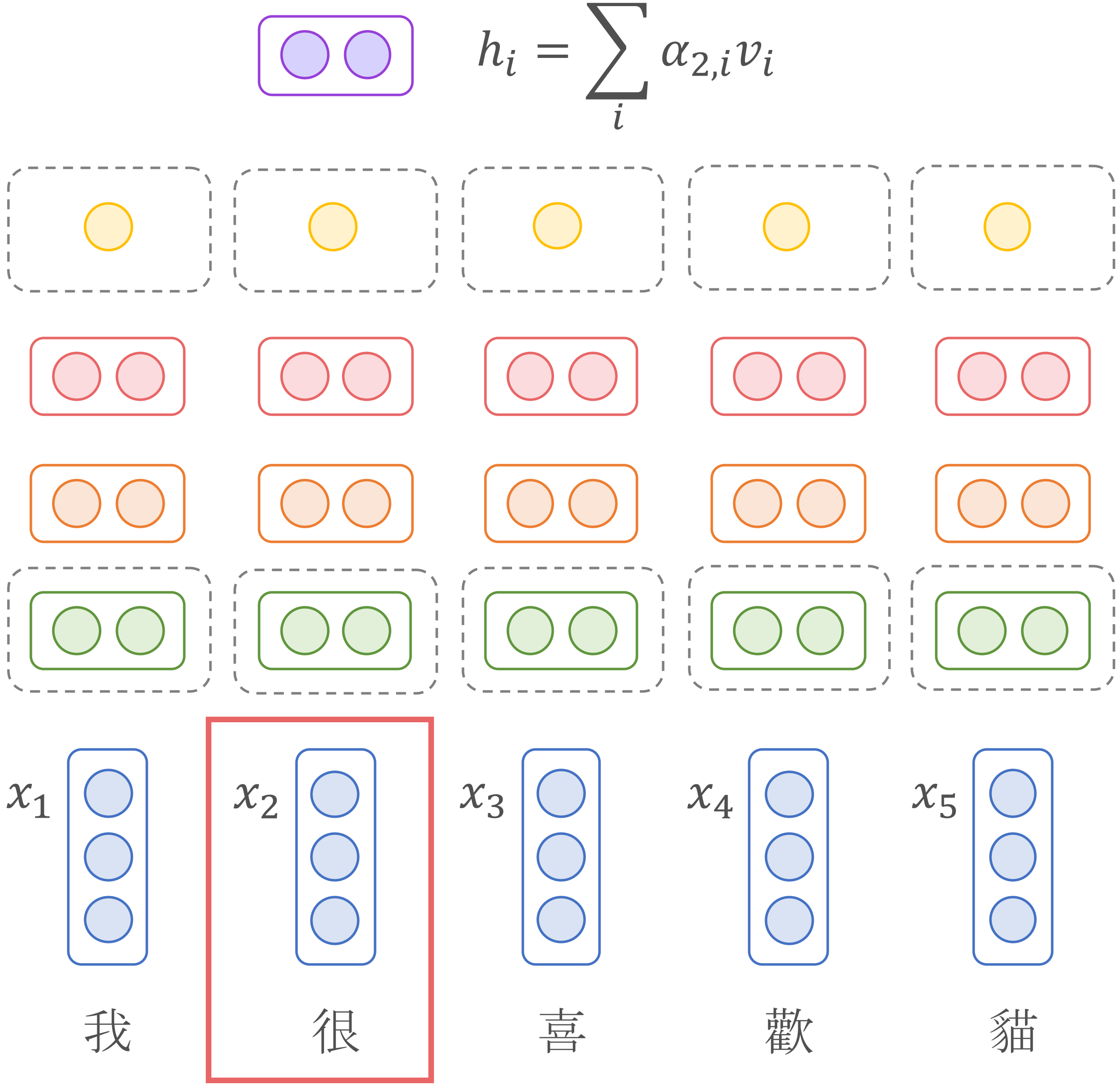
$$h_i = \sum_i \alpha_{2,i} v_i$$

Normalized Attention Scores

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



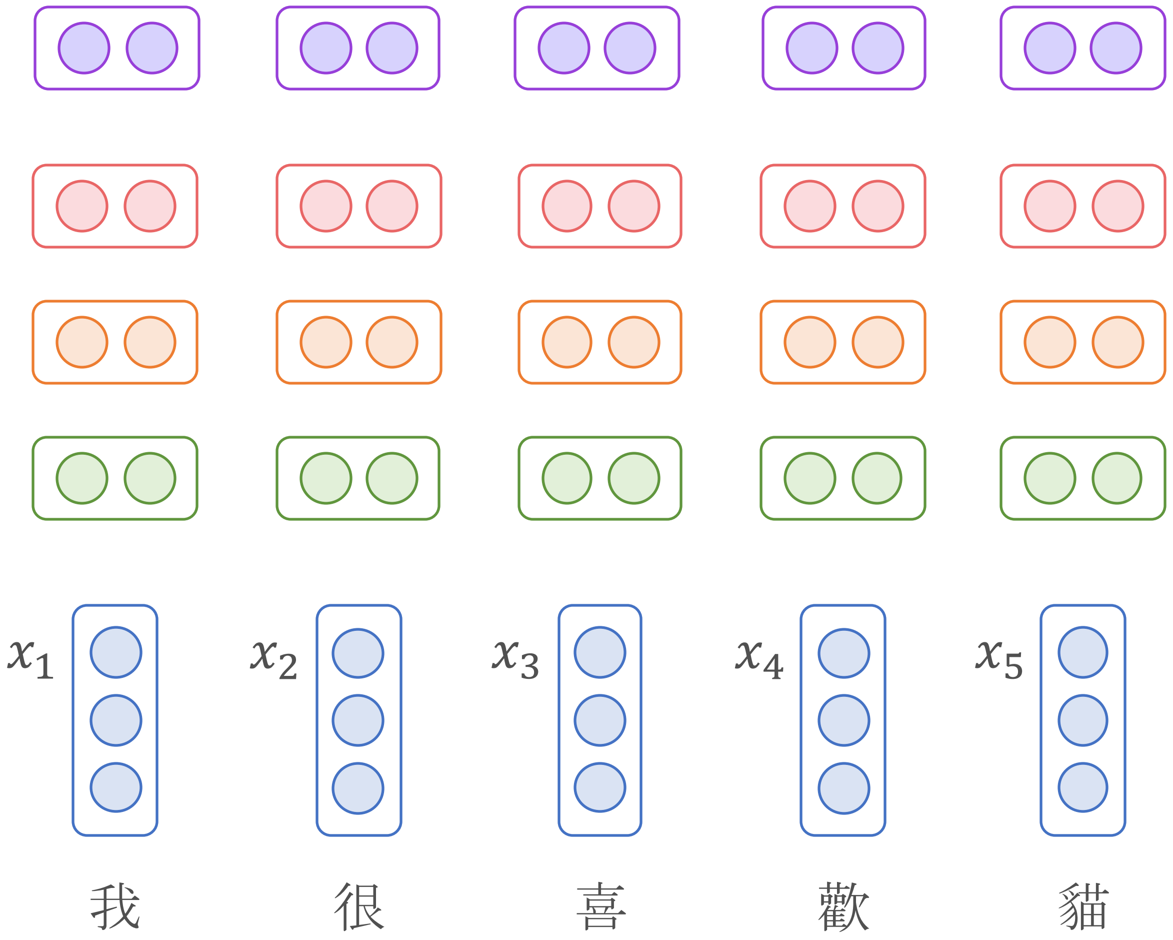
Transformer Layer: Self-Attention

Weighted Sum

Query $q_i = W^Q x_i$

Key $k_i = W^K x_i$

Value $v_i = W^V x_i$



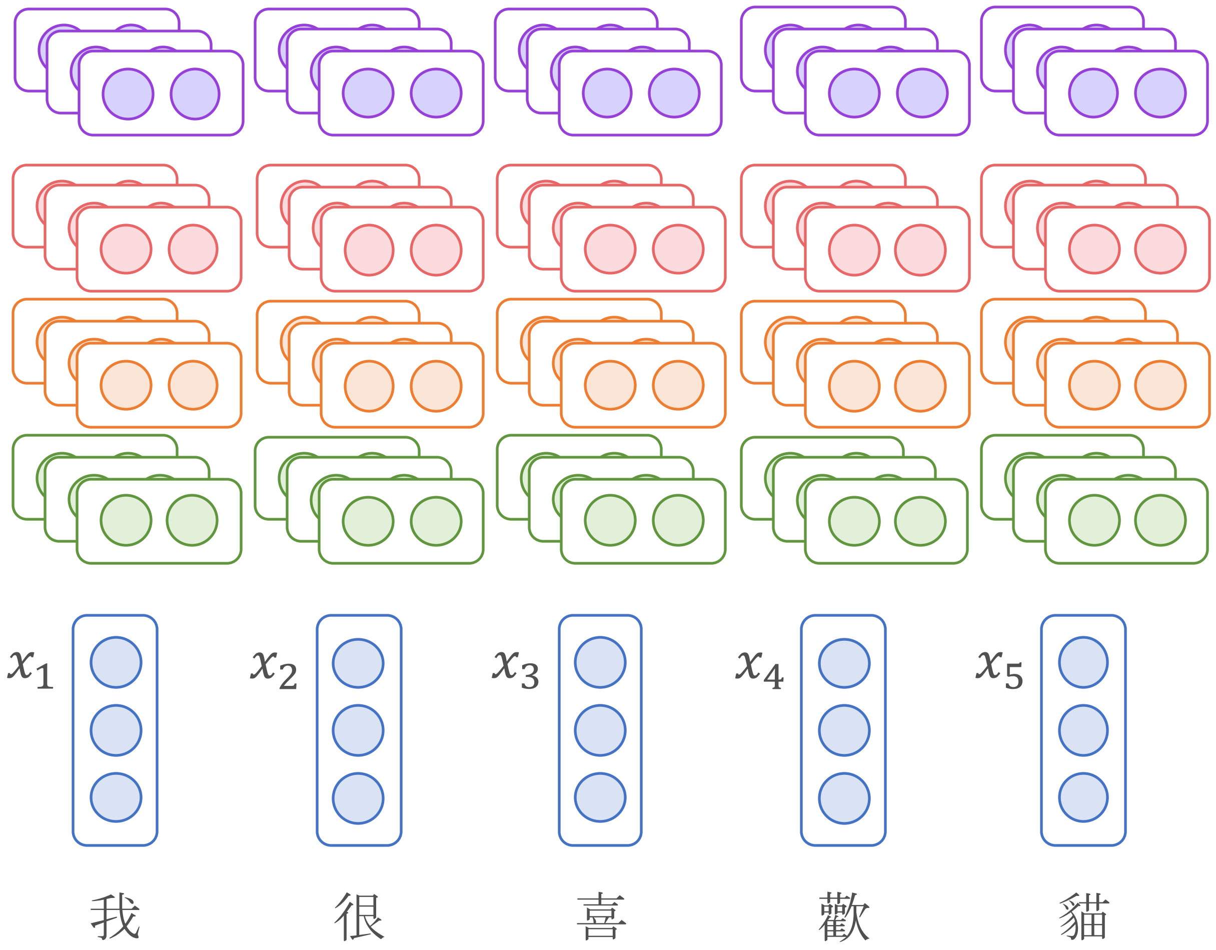
Transformer Layer: Multi-Head Attention

Weighted Sum

Query $q_i = W_j^Q x_i$

Key $k_i = W_j^K x_i$

Value $v_i = W_j^V x_i$



Transformer Layer: Nonlinearity

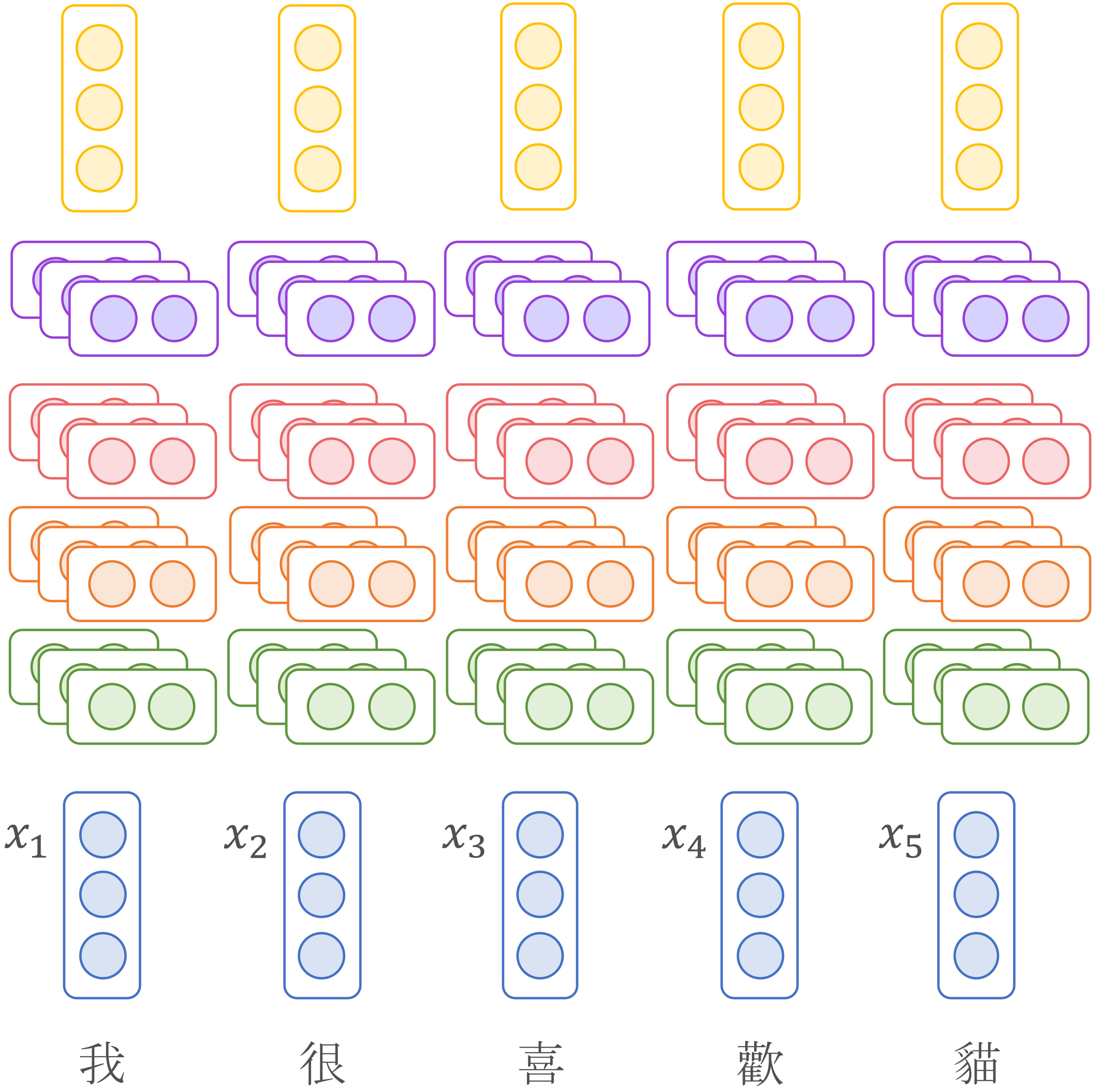
Feed-Forward

Weighted Sum

Query $q_i = W_j^Q x_i$

Key $k_i = W_j^K x_i$

Value $v_i = W_j^V x_i$



Transformer Encoder

- Transformer encoder = a stack of encoder layers

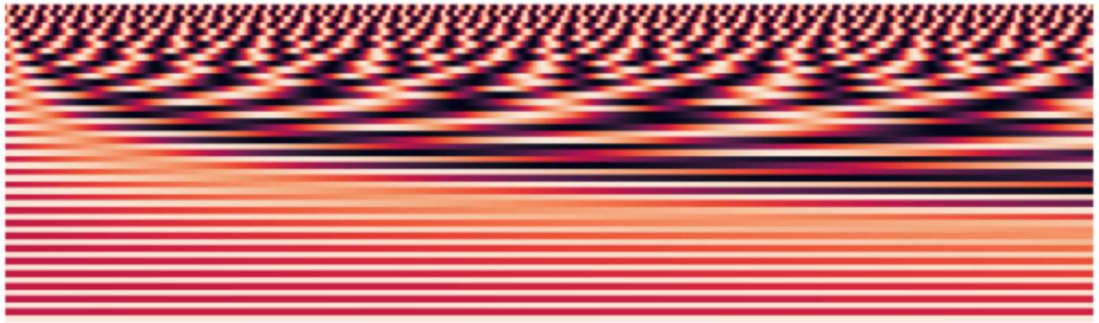
How about word order?

Positional Encoding

$$x_i \leftarrow x_i + PE_i$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

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

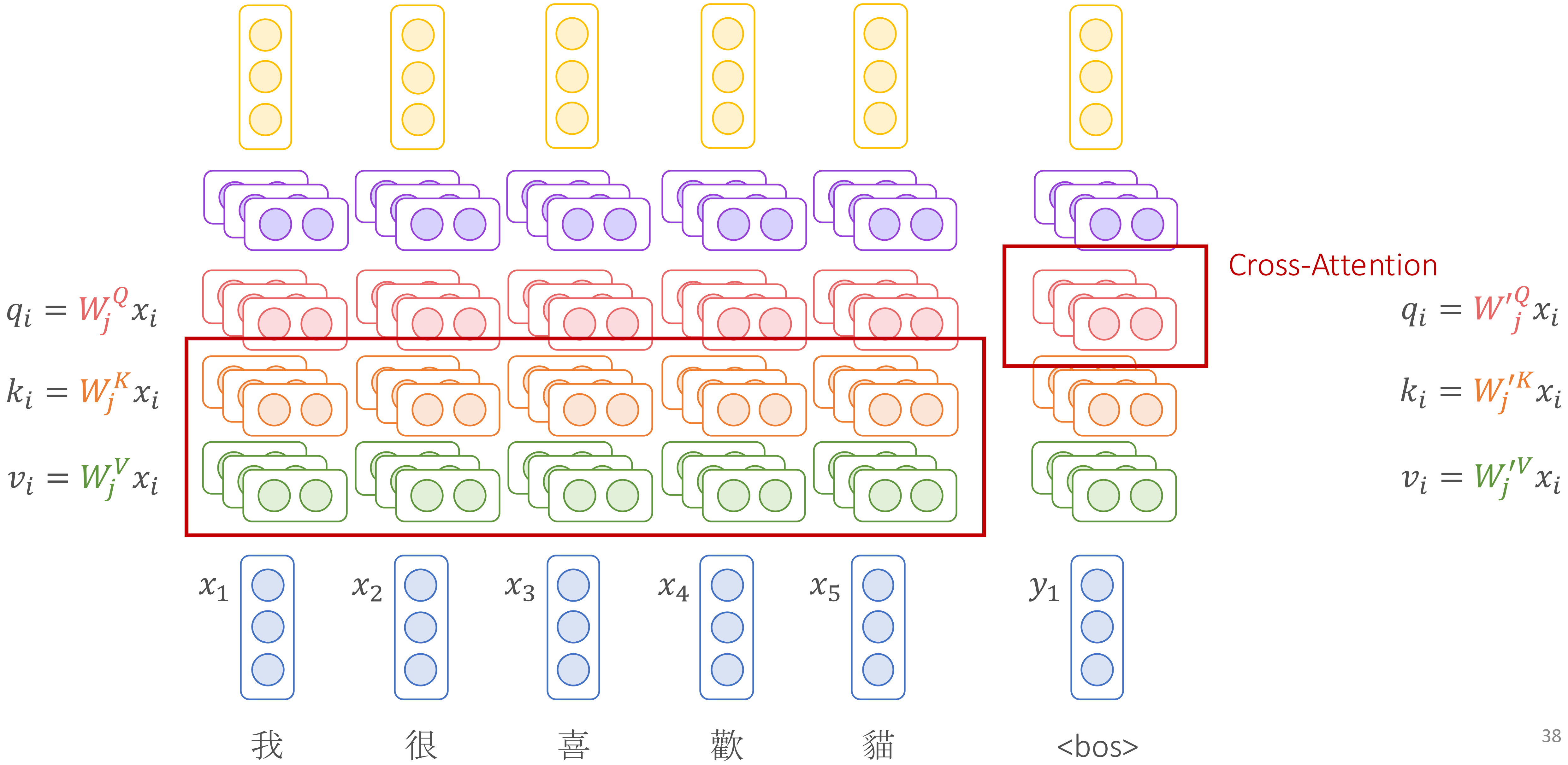


Positional Encoding Matrix with d=4, n=100

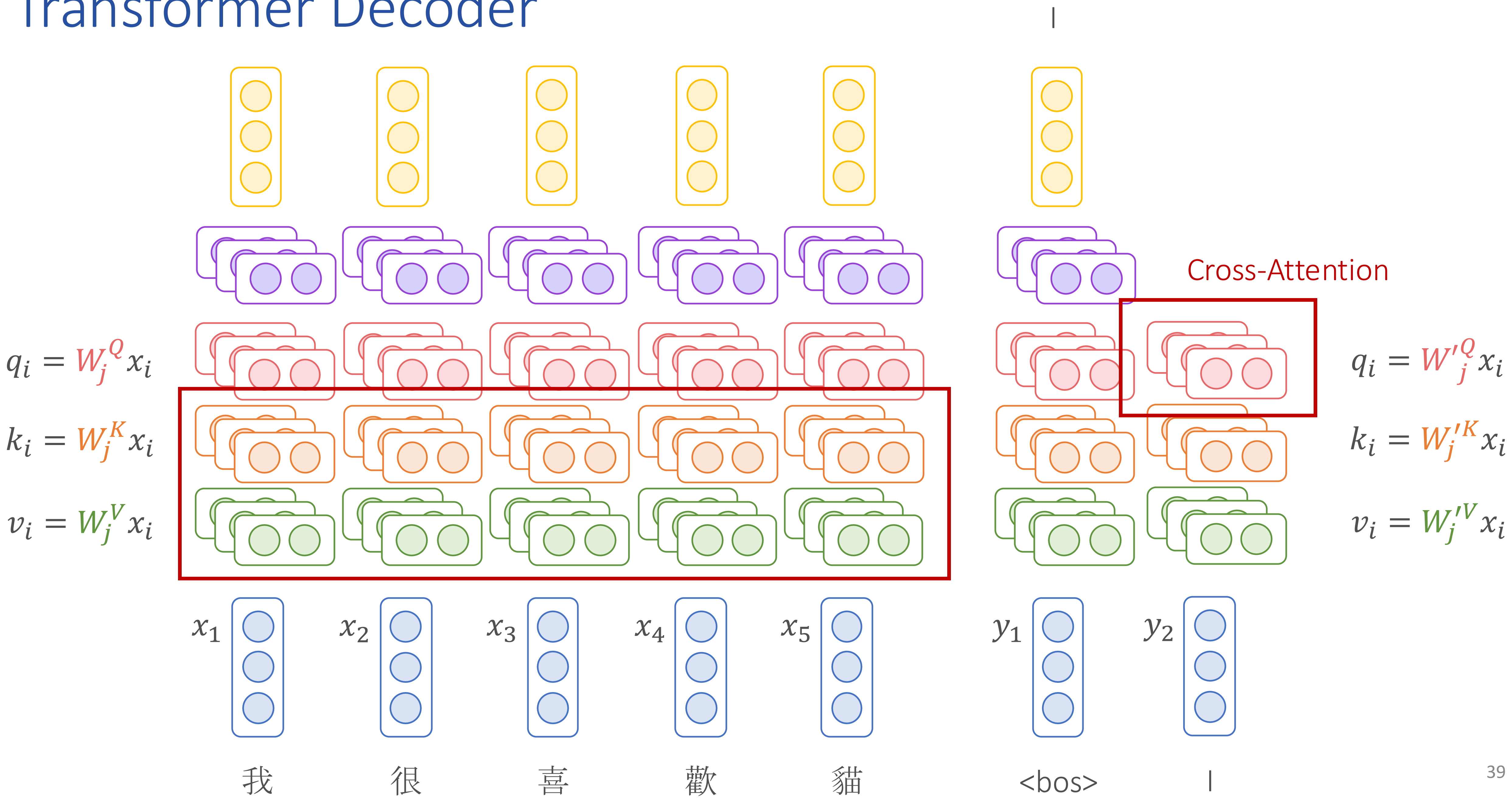
Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	P_{00}=sin(0) = 0	P_{01}=cos(0) = 1	P_{02}=sin(0) = 0	P_{03}=cos(0) = 1
am	1	P_{10}=sin(1/1) = 0.84	P_{11}=cos(1/1) = 0.54	P_{12}=sin(1/10) = 0.10	P_{13}=cos(1/10) = 1.0
a	2	P_{20}=sin(2/1) = 0.91	P_{21}=cos(2/1) = -0.42	P_{22}=sin(2/10) = 0.20	P_{23}=cos(2/10) = 0.98
Robot	3	P_{30}=sin(3/1) = 0.14	P_{31}=cos(3/1) = -0.99	P_{32}=sin(3/10) = 0.30	P_{33}=cos(3/10) = 0.96

Positional Encoding Matrix for the sequence 'I am a robot'

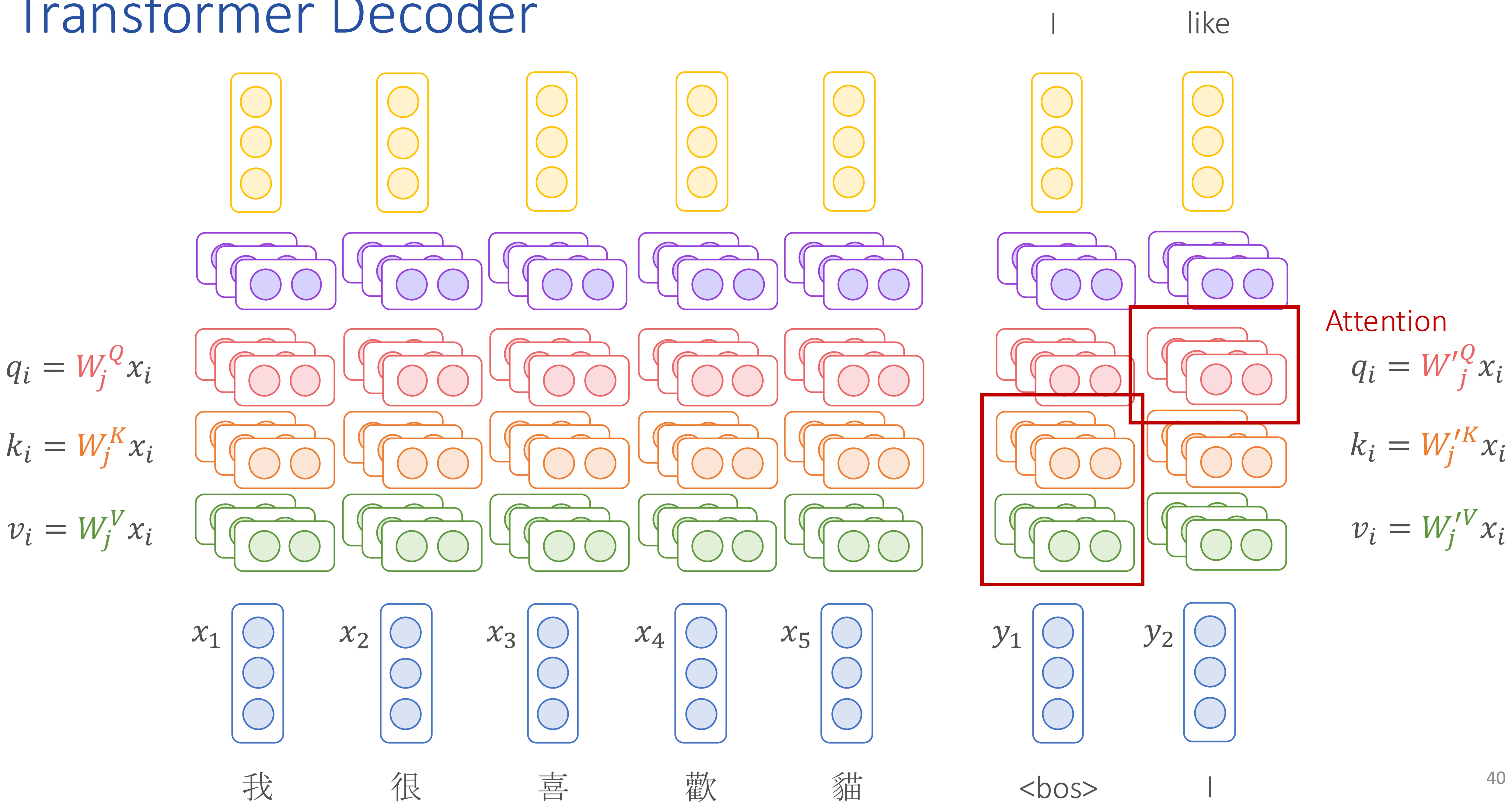
Transformer Decoder



Transformer Decoder



Transformer Decoder



Next Lecture

- Natural Language Processing Basics
- Transformers
- Contextualized Representations
- Pre-Training