STATS 507
Data Analysis in Python

Week13-2: Deep Sequential Modeling (LSTM, Attention)

Dr. Xian Zhang

Adapted from slides by Ava Amini

MIT Introduction to Deep Learning

Solution for in-class practice

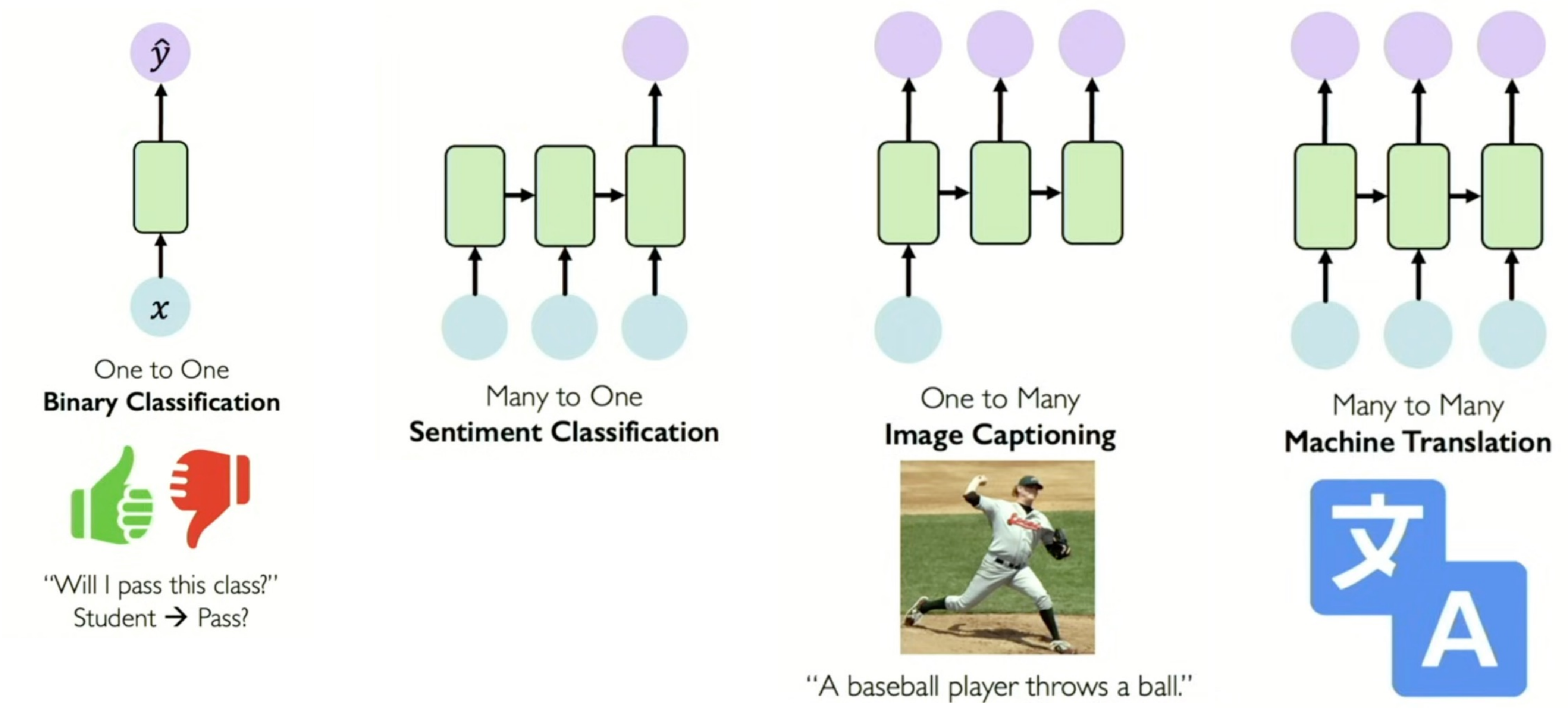
```python
def train_model(model, train_loader, num_epochs, learning_rate=0.001):
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

    for epoch in range(num_epochs):
        model.train()
        total_loss = 0

        for batch_idx, (inputs, targets) in enumerate(train_loader):
            # Initialize hidden state
            hidden = model.init_hidden(inputs.size(0))

            # Forward pass
            outputs, hidden = model(inputs, hidden)
            loss = criterion(outputs[:, -1, :], targets)

            # Backward pass and optimize
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            total_loss += loss.item()

        avg_loss = total_loss / len(train_loader)
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {avg_loss:.4f}')
```

Training loop for each epoch

Gradient descent with mini-batches

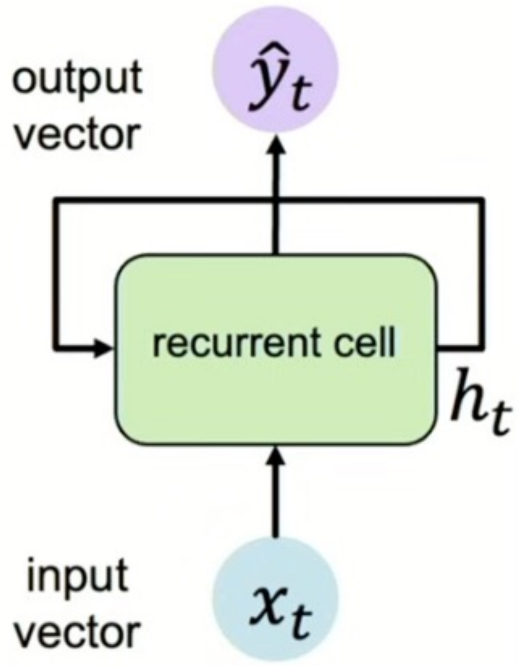
Update both output and hidden state

Recap: Sequence modeling



Slide Credit: Ava Amini

Recap: RNN



Update hidden state (cell state)

past memory

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t)$$

Hidden State Activation function Previous hidden state Input vector

$$\hat{y}_t = W_{hy}^T h_t$$

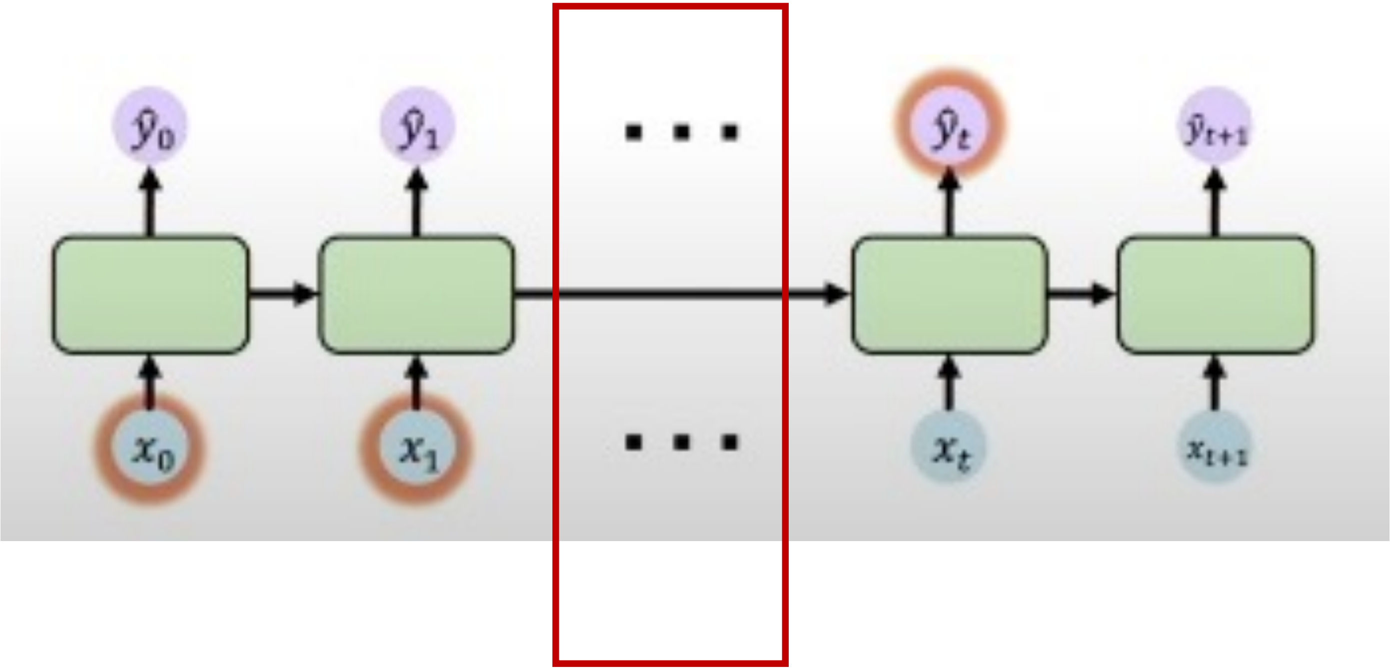
Note: in RNN, we use the SAME function and set of parameters at every time step.

Slide Credit: Ava Amini

Recap: motivation of LSTM

Why is vanishing gradient is a problem?

Vanishing gradient is a problem because if we multiply many small numbers together, we would have smaller and smaller gradient, and the bias parameters are only there to capture short-term dependencies.



Gates can optionally let information through the cell

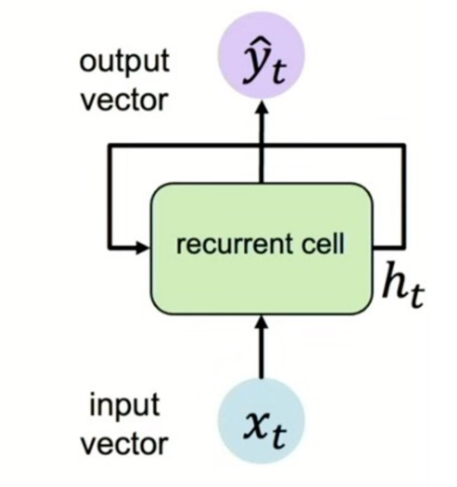
Slide Credit: Ava Amini

1. More on LSTM

2. Attention and Transformers

LSTM

Vanilla RNN



Previous hidden state

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)$$

Input vector

LSTM

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Hadamard elementwise product operator

At each time step: we introduce 4 **gates** and a new **state** -> cell state

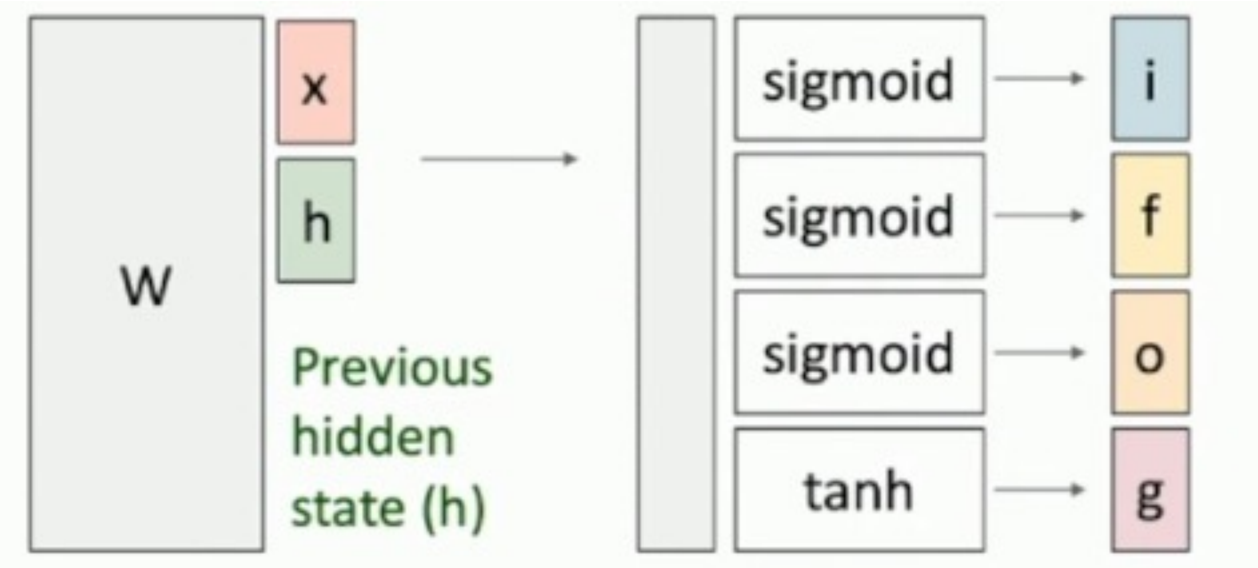
Cell state: $c_t \in \mathbb{R}^H$
Hidden state: $h_t \in \mathbb{R}^H$

Slide Credit: Justin Johnson

Unpack Gates in LSTM

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell

At each time step: we introduce 4 **gates** and a new **state** -> cell state



$$i_t = \sigma(x_t W_{xi} + h_{t-1} W_{hi}) + b_i$$

$$f_t = \sigma(x_t W_{xf} + h_{t-1} W_{hf}) + b_f$$

[0,1]

$$o_t = \sigma(x_t W_{xo} + h_{t-1} W_{ho}) + b_o$$

$$g_t = tanh(x_t W_{xg} + h_{t-1} W_{hg}) + b_g$$

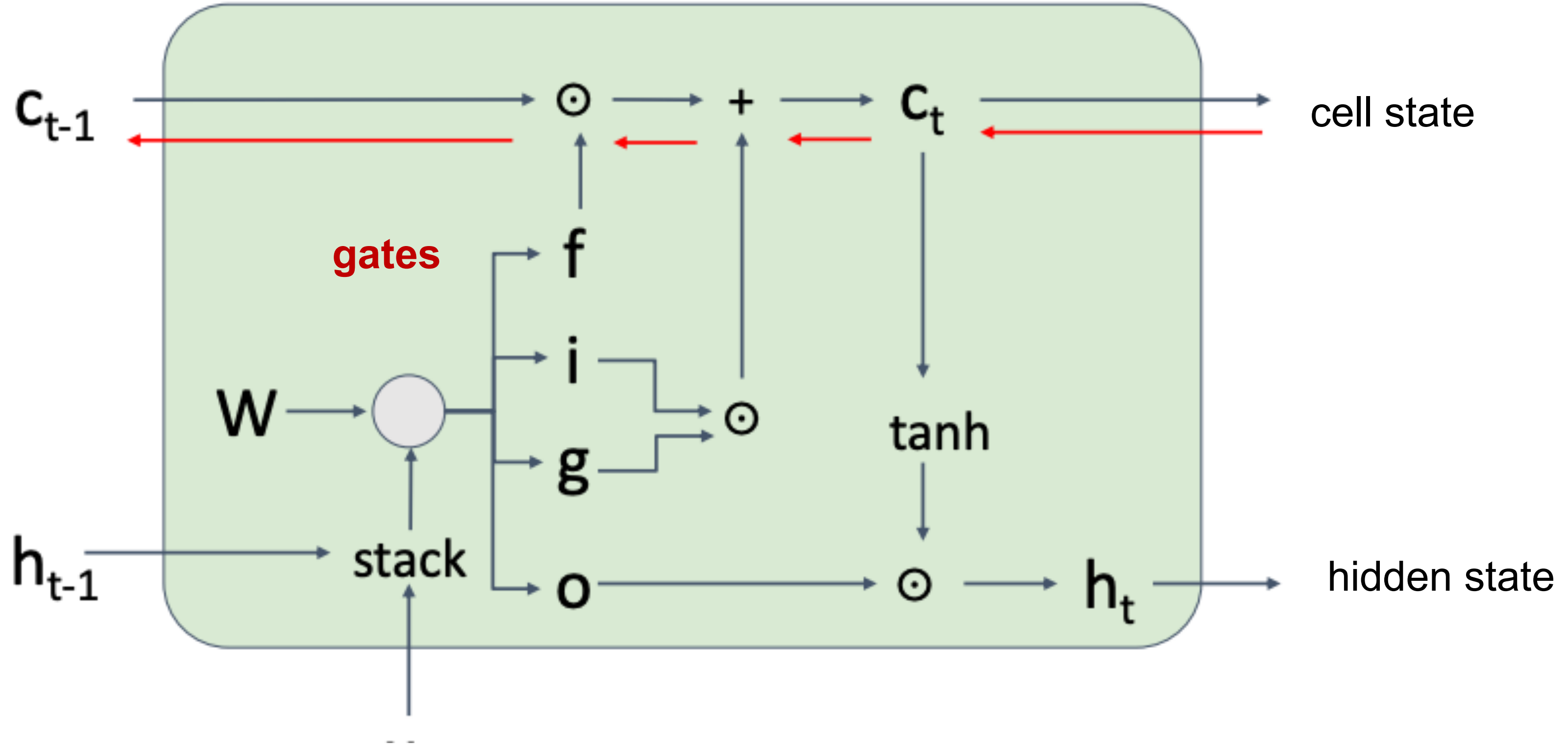
[-1,1]

Learned weights!

Slide Credit: Justin Johnson

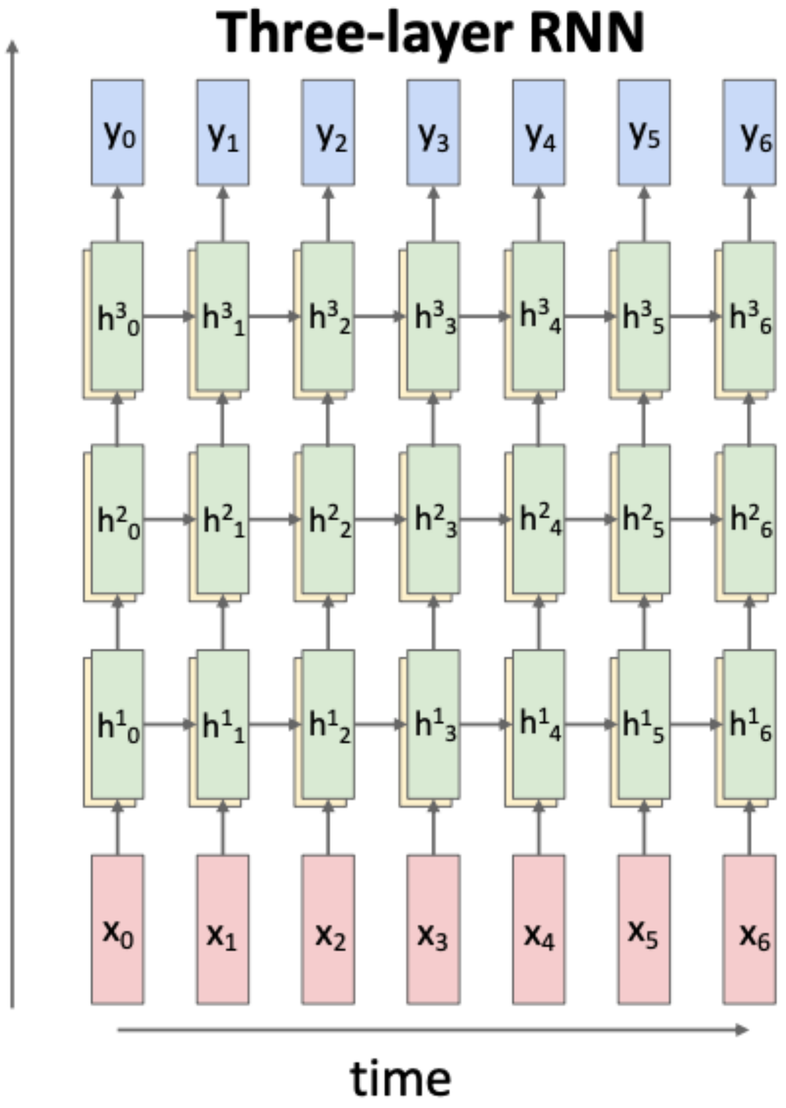
LSTM

At each time step: we introduce 4 **gates** and a new **state** -> cell state



Slide Credit: Justin Johnson

Stacking RNN and LSTM



Mutilayer RNNs

$$\boxed{h_t^\ell = \tanh\left(W\begin{pmatrix}h_{t-1}^\ell\\ h_t^{\ell-1}\end{pmatrix} + b_h^\ell\right)}$$

LSTM:

$$\begin{pmatrix}i_t^\ell\\ f_t^\ell\\ o_t^\ell\\ g_t^\ell\end{pmatrix} = \begin{pmatrix}\sigma\\ \sigma\\ \sigma\\ \tanh\end{pmatrix}\left(W\begin{pmatrix}h_{t-1}^\ell\\ h_t^{\ell-1}\end{pmatrix} + b_h^\ell\right)$$

$$c_t^\ell = f_t^\ell \odot c_{t-1}^\ell + i_t^\ell \odot g_t^\ell$$

$$h_t^\ell = o_t^\ell \odot \tanh(c_t^\ell)$$

Slide Credit: Justin Johnson

Key Concepts

RNN

- Can model the **sequential** data
- Backward flow of gradients in RNN can explode or vanish
    - Exploding is controlled with gradient clipping
    - Vanishing is controlled with additive interaction (gates cells)

LSTM

- Maintain a **cell state**
- Use gates to control the flow of information
    - Input gate stores relevant info
    - Forget gate gets ride of irrelevant info
    - Selectively update cell state
    - Output fate returns a filtered version of the cell state

1. LSTM

2. Attention and Transformers

Attention Is All You Need

ChatGPT BERT

Transformer

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

Abstract

Core idea:

- **Attending** to the most important parts of an input
- Extract the features with high attention

A very approximate timeline

- 1990 Static Word Embeddings
- 2003 Neural Language Model
- 2008 Multi-Task Learning
- 2015 Attention
- 2017 Transformer
- 2018 Contextual Word Embeddings and Pretraining
- 2019 Prompting

Attention as a mechanism

A mechanism for helping compute the **embedding** for a token by selectively attending to and integrating information from surrounding tokens (at the previous layer).

More formally: a method for doing a weighted sum of vectors.

Input: Given a sequence of token embeddings:

$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \mathbf{x}_4 \quad \mathbf{x}_5 \quad \mathbf{x}_6 \quad \mathbf{x}_7$

Output: $\mathbf{a}_i$ = a weighted sum of $\mathbf{x}_1$ through $\mathbf{x}_7$
Weighted by their similarity to $\mathbf{x}_i$

$$\mathbf{a}_i = \boxed{\sum_{j \le i} \alpha_{ij}}\mathbf{x}_j$$

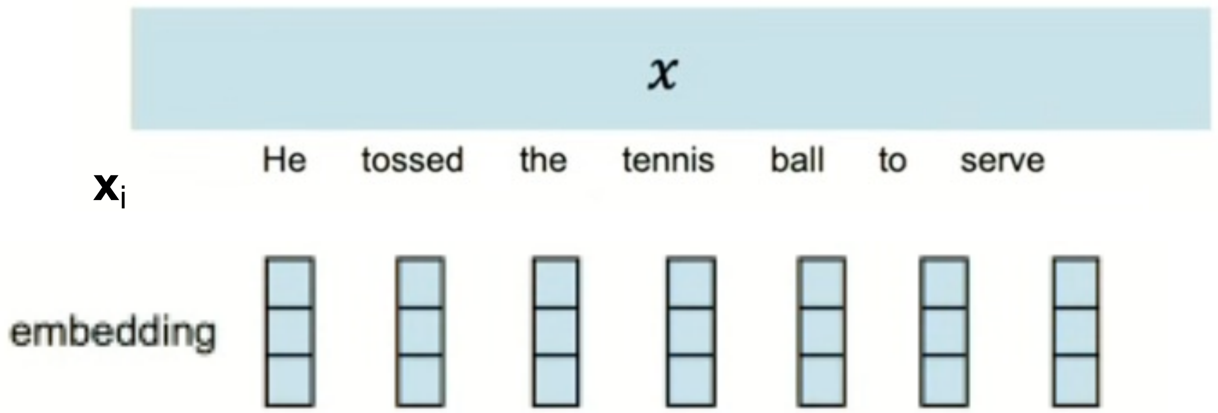
Attention introduces 3 separate rols

- High-level idea: instead of using input vectors directly, we'll represent 3 separate roles each vector x_i plays:

- **query:** As the current element being compared to the preceding inputs.
- **key:** as a preceding input that is being compared to the current element to determine a similarity
- **value:** a value of a preceding element that gets weighted and summed

Attention for sequence modeling

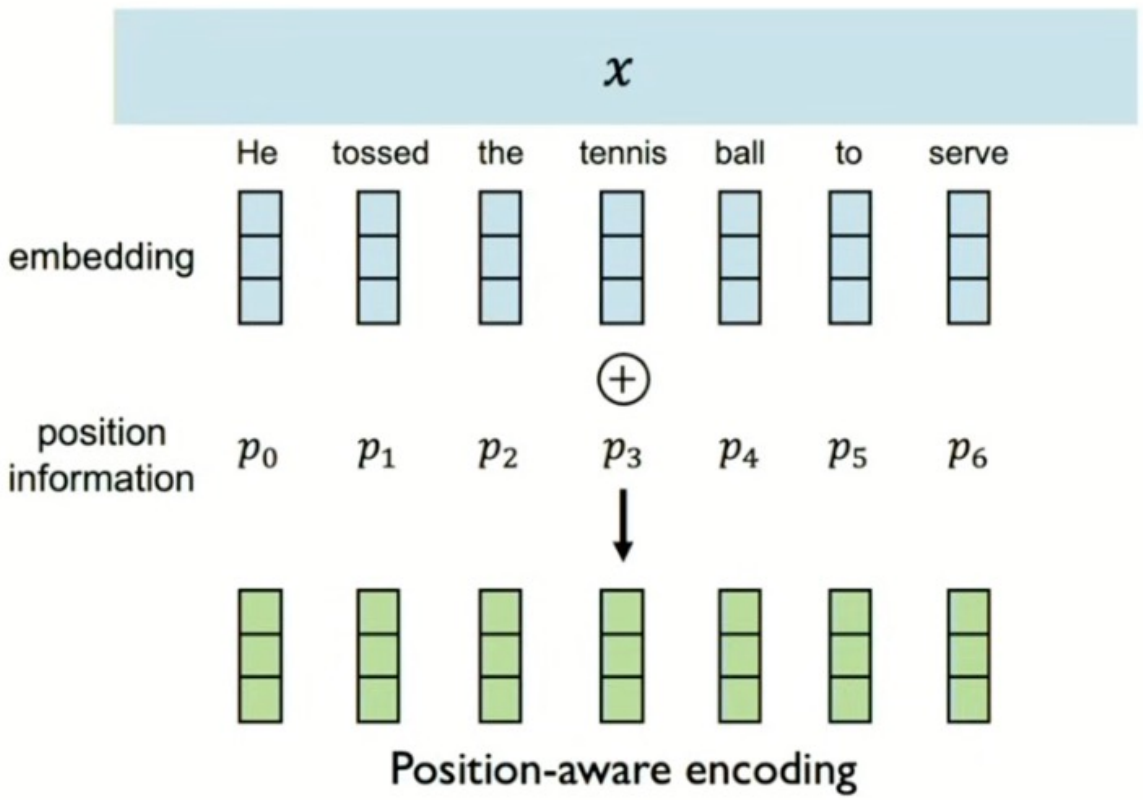
Goal: identify and attend to most important features in input



- Data is fed in all at once, not going to handle this timestep by timestep.
- What information is lost?

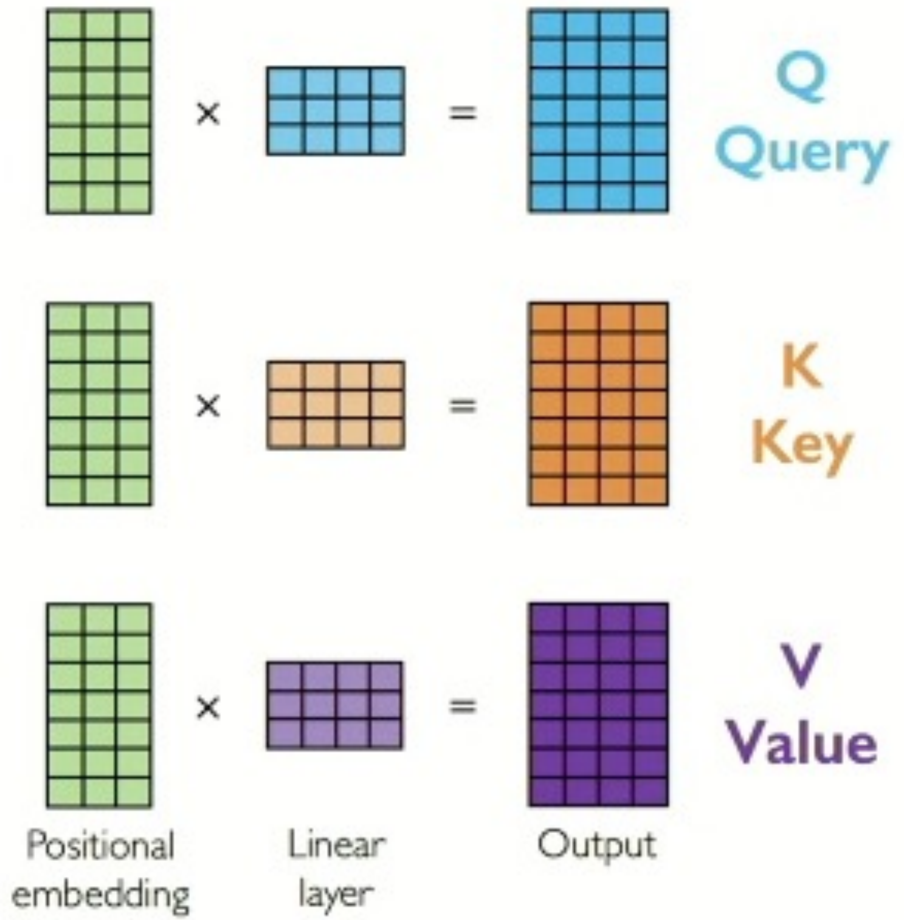
Positional embedding

Data is fed in all at once, we need to encode position information



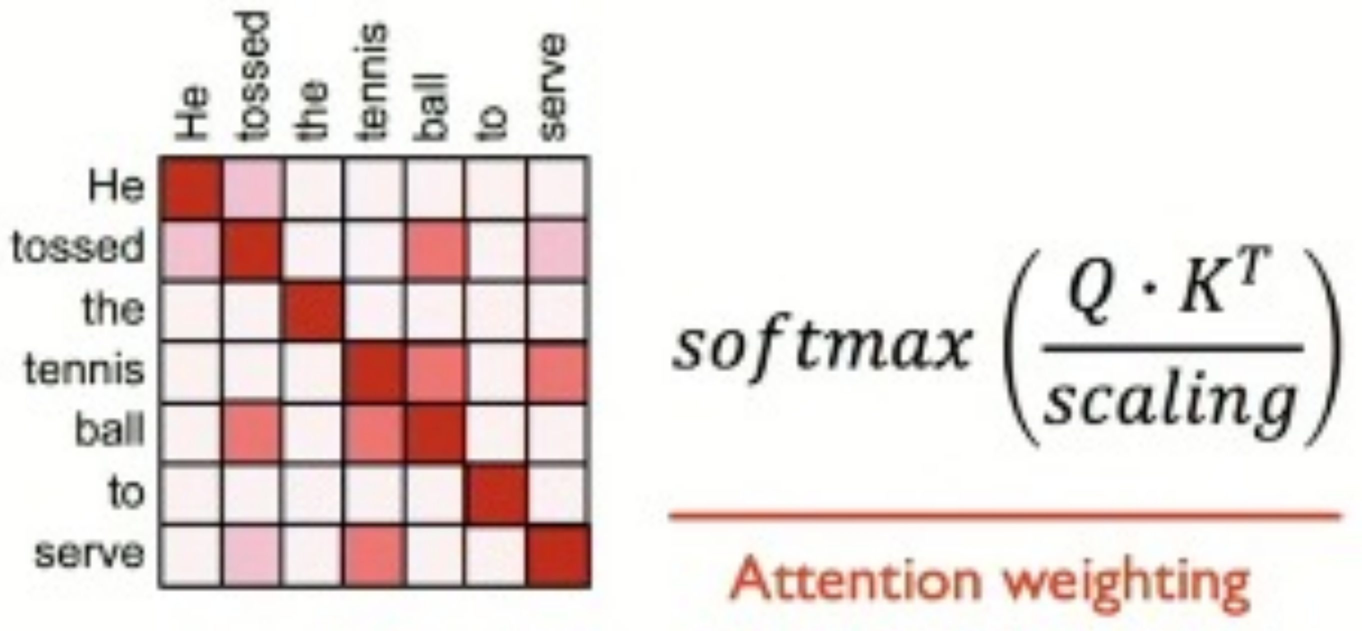
Extract query, key, value

Extract the three introduced components *Q*, *K*, *V*



Compute attention weights

Extract the three introduced components *Q, K, V*

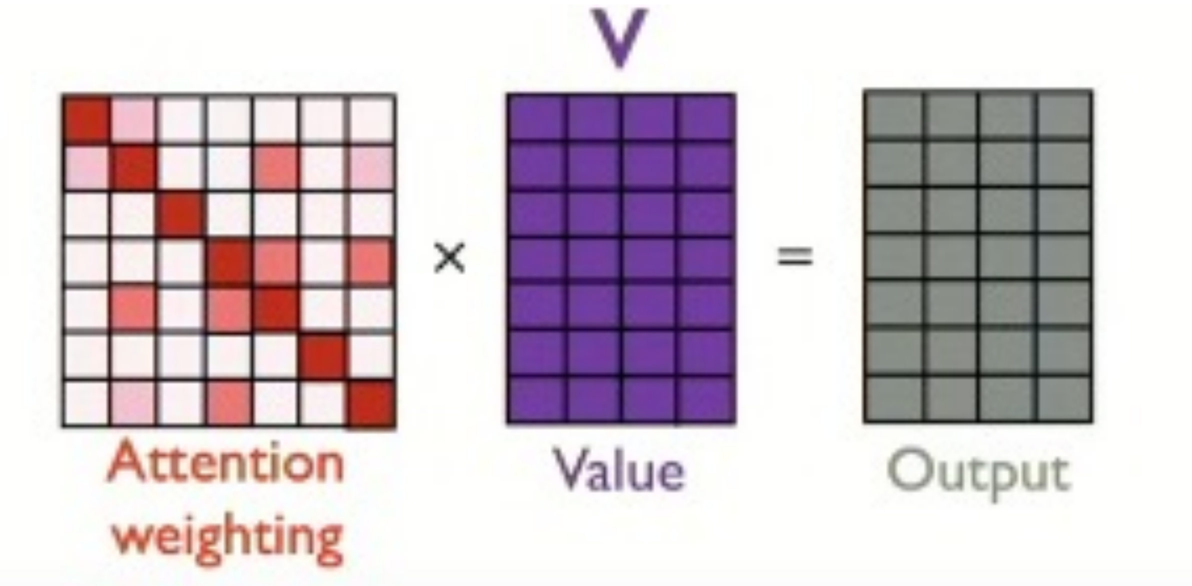


Extract features using self attention

Input: Given a sequence of token embeddings:

$\mathbf{x}_1 \quad \mathbf{x}_2 \quad \mathbf{x}_3 \quad \mathbf{x}_4 \quad \mathbf{x}_5 \quad \mathbf{x}_6 \quad \mathbf{x}_7$

Output: $\mathbf{a}_i$ = a weighted sum of $\mathbf{x}_1$ through $\mathbf{x}_7$
Weighted by their similarity to $\mathbf{x}_i$

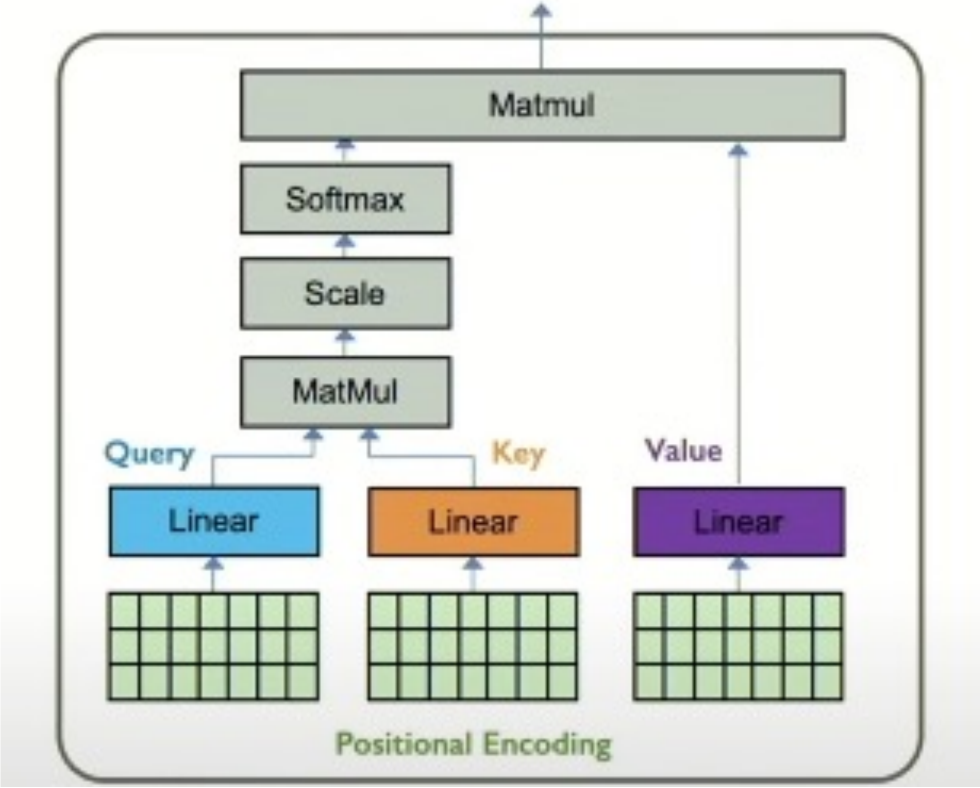


$$\mathbf{a}_i = \sum_{j \leq i} \alpha_{ij} \mathbf{x}_j$$

Learning attention with neural networks

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V$$



in-class practice

Attention from scratch

Actual Attention: multi-head

- Instead of one attention head, we'll have lots of them!
- Intuition: each head might be attending to the context for different purposes
    - Different linguistic relationships or patterns in the context

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

From attention to transformer model

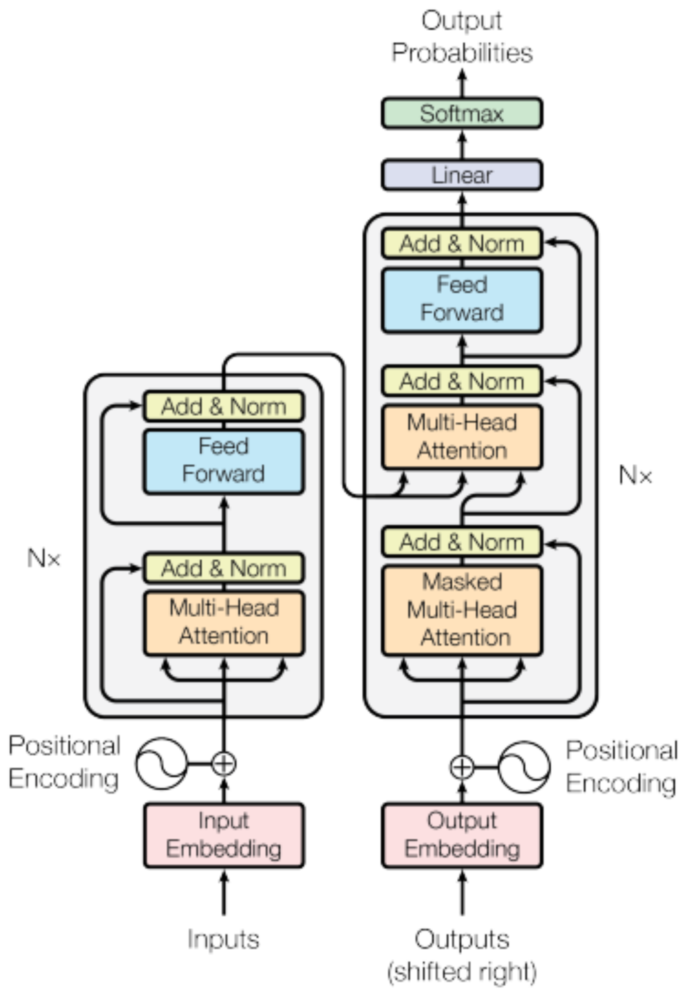


Figure 1: The Transformer - model architecture.

Summary

- RNN are well suited for sequence modeling tasks.
- Model sequence via a recurrence relation
- LSTM can resolve the gradient vanishing issue
- Attention mechanism can model sequences without recurrence
- Attention is the basis for transformer and many large language models

FAQ about final project

- About dataset size
- About model complexity
- About the performance of the model

Other things

HW8 out.

Final project guideline out (**start early**)

Coming next:

CNN