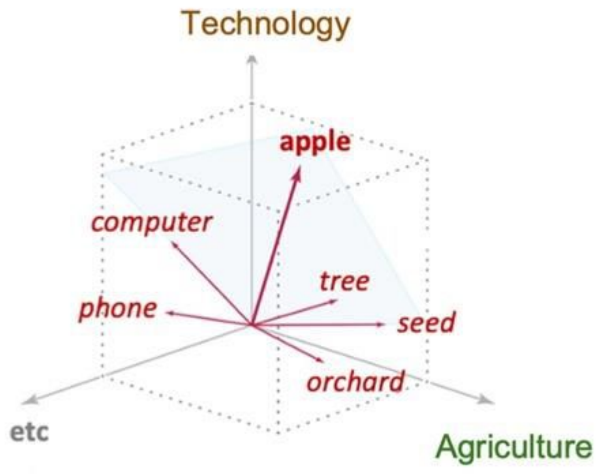
Deep Learning & Generative AI in Healthcare

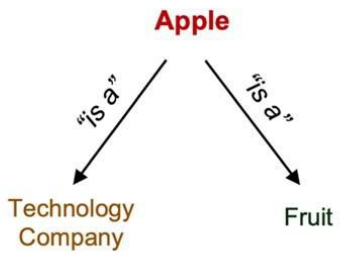
Session 07

Probabilistic Model of Language



Word Polysemy

1. I mainly use my **Apple** **iPhone** to make **phone** calls.
2. The **Apple** MacBook Pro is a **computer** with a powerful **processor**.
3. I use an **Apple** **computer** to write **emails** and create **documents**.
4. I picked a red **apple** from the **tree** in the backyard.
5. The planted **seeds** in the **orchard** produced several **apple** **trees**.
6. **Apples** are my favorite type of **fruit**.



Distributional Hypothesis of Word Meaning

Probabilistic Model of Language

Probability model:

1. $p(L) = 1$

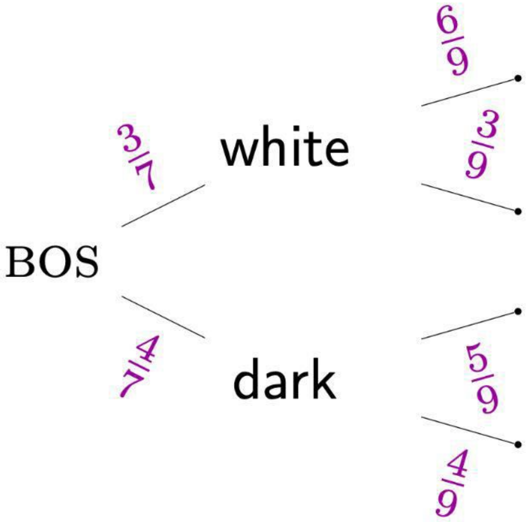
2. $p(\bigcup_{i=1}^{n} \mathcal{E}_i) = \sum_i^n p(\mathcal{E}_i)$ if $\mathcal{E}_1, \mathcal{E}_2, \ldots$ is a countable sequence of disjoint sets of $\mathcal{P}(L)$, the power set (=set of all subsets) of L.

3. (Conditional probability) $$p(\mathbf{x}) = p(x_0) \prod_{i=1}^{L} p(x_i | x_1, \ldots, x_{i-1})$$

$$\log p(\mathbf{x}) = \log p(x_0) \sum_{i=1}^{L} \log p(x_i | x_1, \ldots, x_{i-1})$$

Link

Probabilistic Model of Language



lotus $p(lotus \mid white)p(white) = \frac{3}{7} \cdot \frac{6}{9}$

knight $p(knight \mid white)p(white) = \frac{3}{7} \cdot \frac{3}{9}$

knight $p(knight \mid dark)p(dark) = \frac{4}{7} \cdot \frac{5}{9}$

materials $p(materials \mid dark)p(dark) = \frac{4}{7} \cdot \frac{4}{9}$

Probabilistic Model of Language

- Given a sequence of words, compute the **probability distribution of the next word:**

$$P(\boldsymbol{x}^{(t+1)} \mid \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

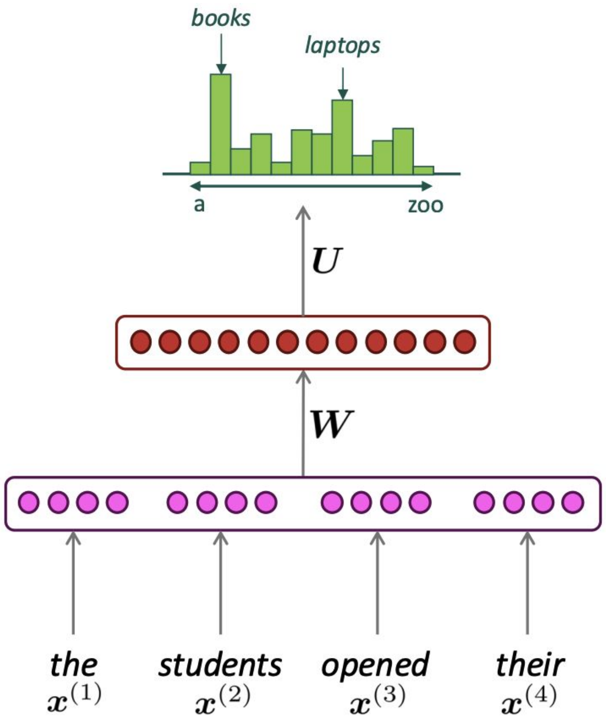
where $x^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

$$\begin{aligned} P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) &= P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} \mid \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} \mid \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)}) \\ &= \prod_{t=1}^{T} \underbrace{P(\boldsymbol{x}^{(t)} \mid \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})}_{\text{This is what the LM provides}} \end{aligned}$$

Neural Network Model of Language

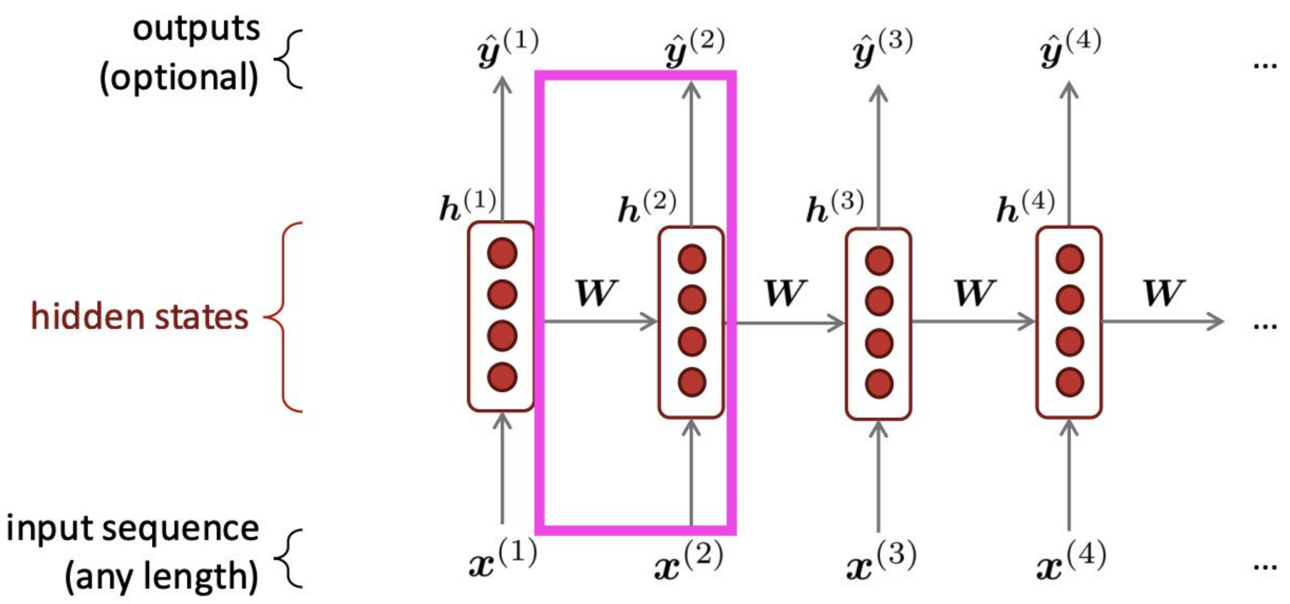
- A neural probabilistic language model (Y. Bengio, et al.)
- Fixed window is small
- No window is large enough

We need a neural architecture that can process *any length input*



Recurrent Neural Networks

- Apply the same weights **W** repeatedly
- Input can be of any length!



Recurrent Neural Networks

output distribution

$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$

hidden states

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

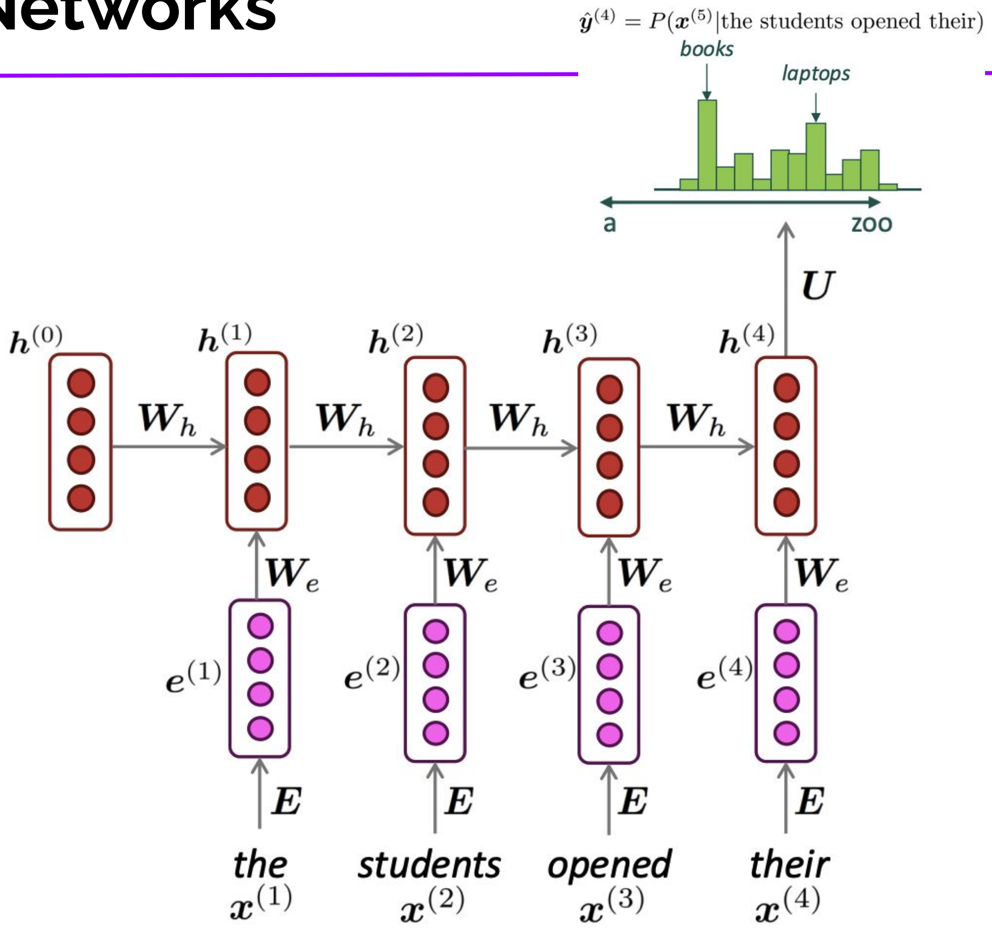
$\boldsymbol{h}^{(0)}$ is the initial hidden state

word embeddings

$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$

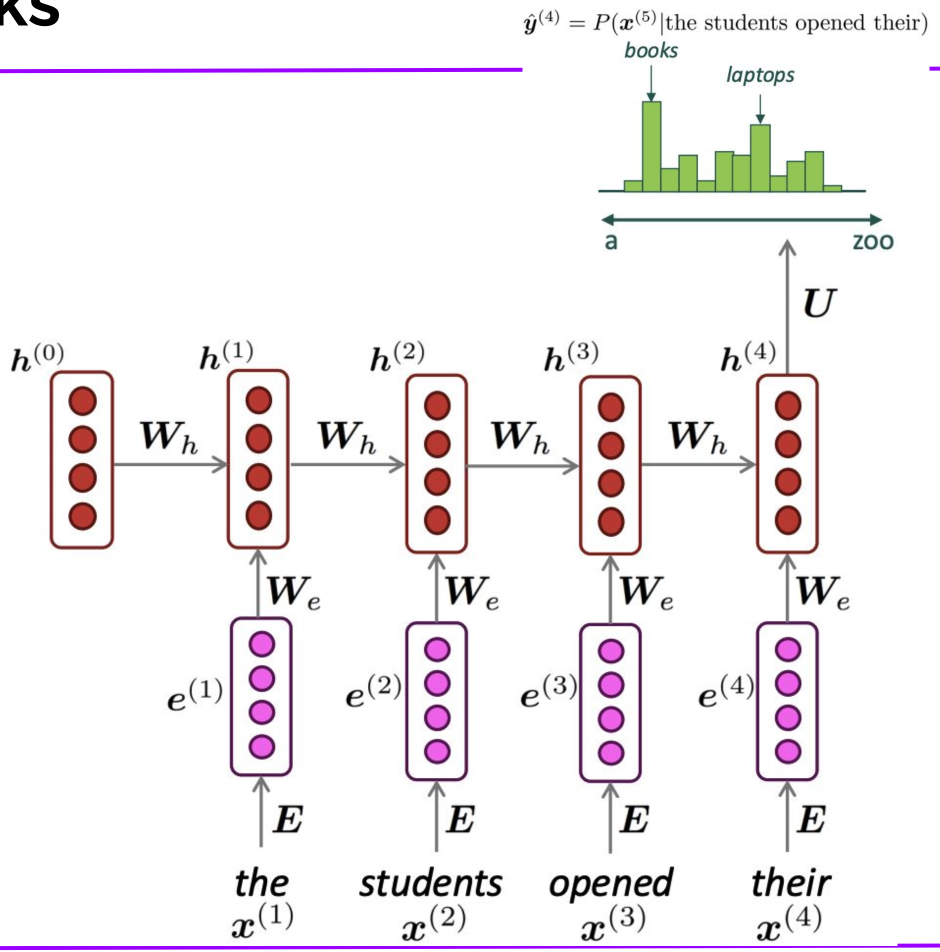
words / one-hot vectors

$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$



Recurrent Neural Networks

- Advantages
  - The process any length!
  - Can use information from previous steps
  - Model size does not increase for longer input context
  - Same weights applied on every timestep
- Disadvantages
  - Slow
  - Difficult to access information from many steps back



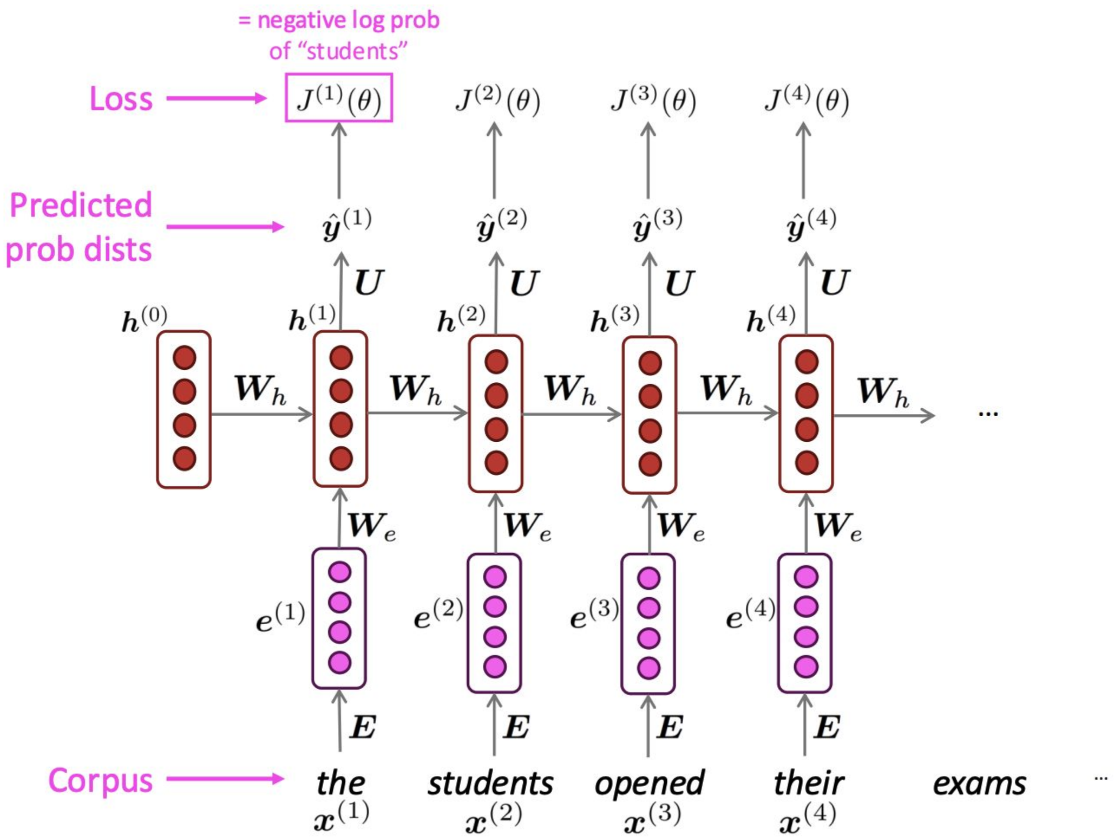
Recurrent Neural Networks

- Get a big corpus of text, i.e., sequence of $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}$
- Feed into RNN, compute output distribution $\hat{\boldsymbol{y}}^{(t)}$
  - Predict probability dist of every word, given words so far
- Loss function is cross-entropy between predicted probability $\hat{\boldsymbol{y}}^{(t)}$, and the true next word $\boldsymbol{y}^{(t)}$

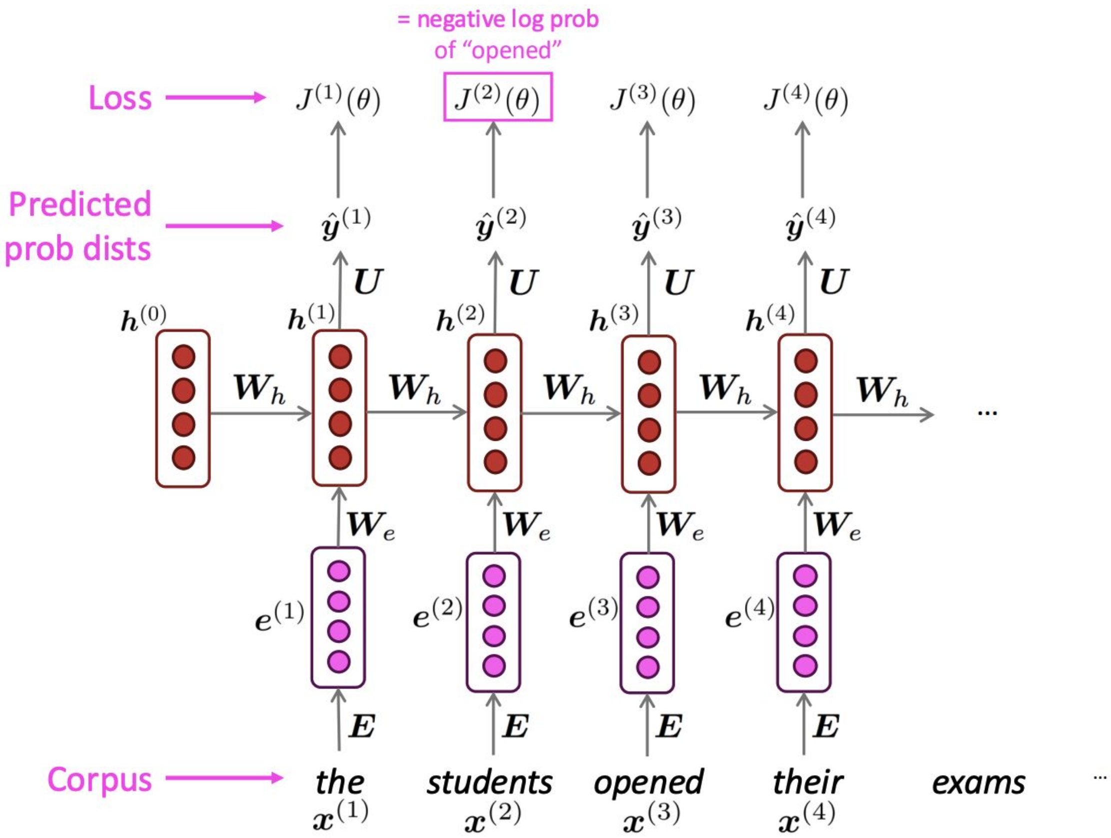
$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

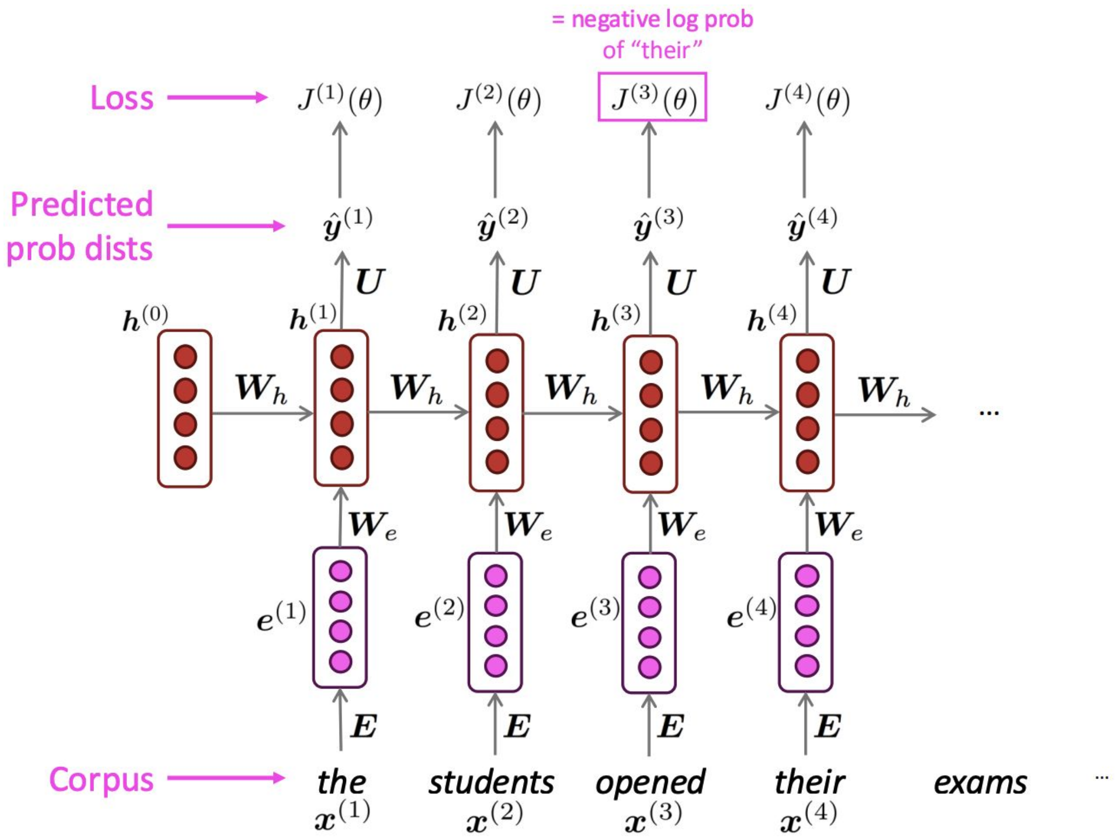
Recurrent Neural Networks



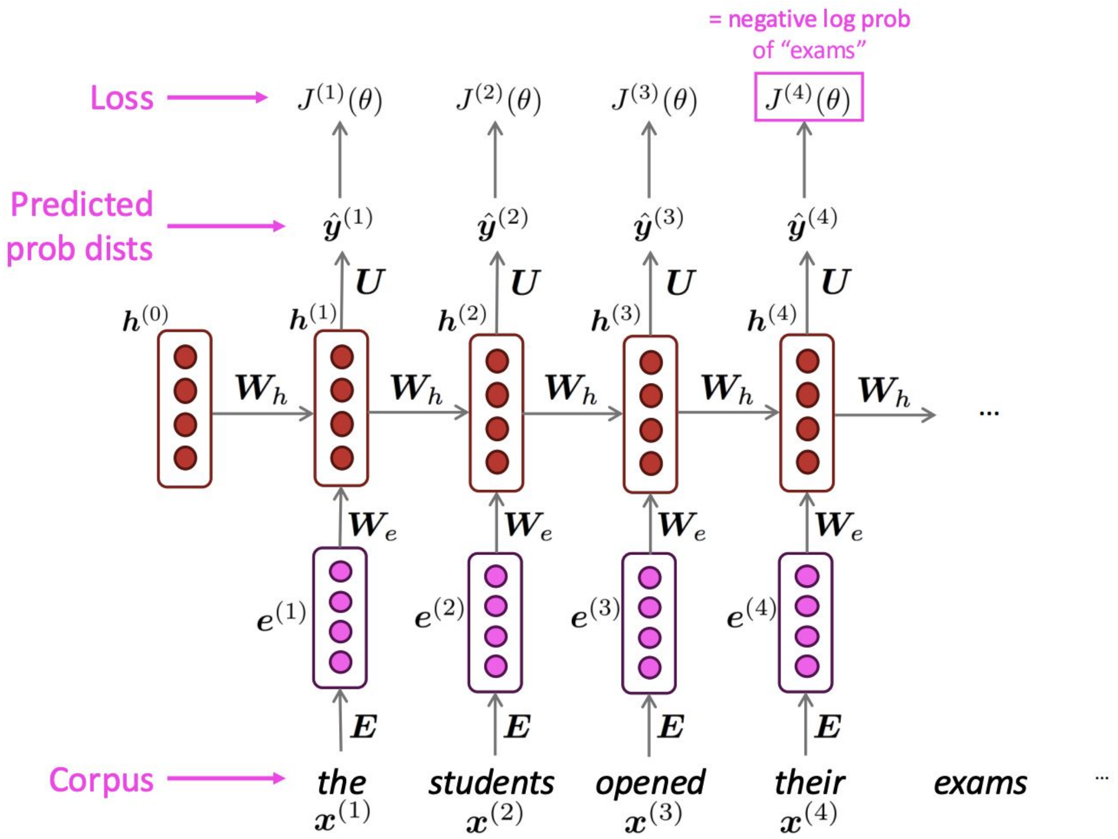
Recurrent Neural Networks



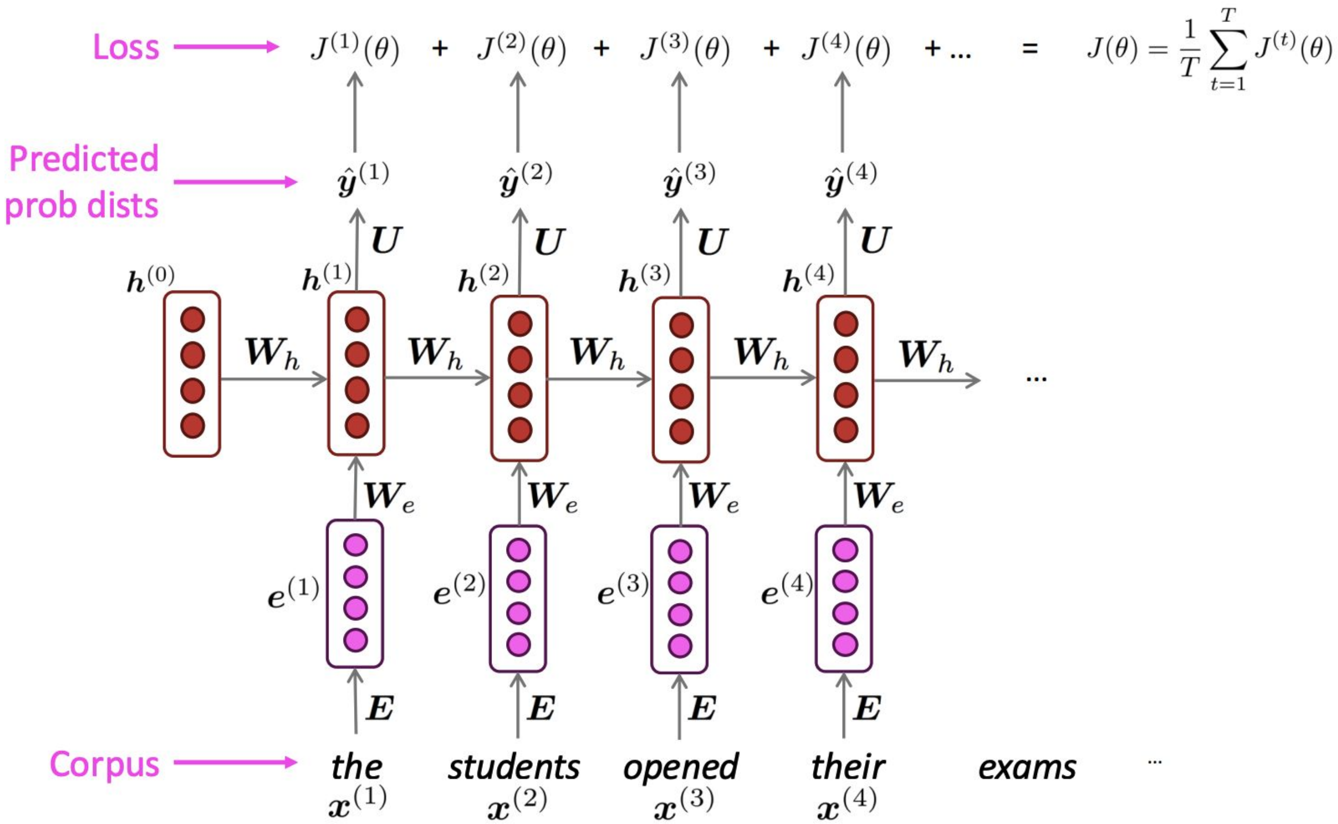
Recurrent Neural Networks



Recurrent Neural Networks

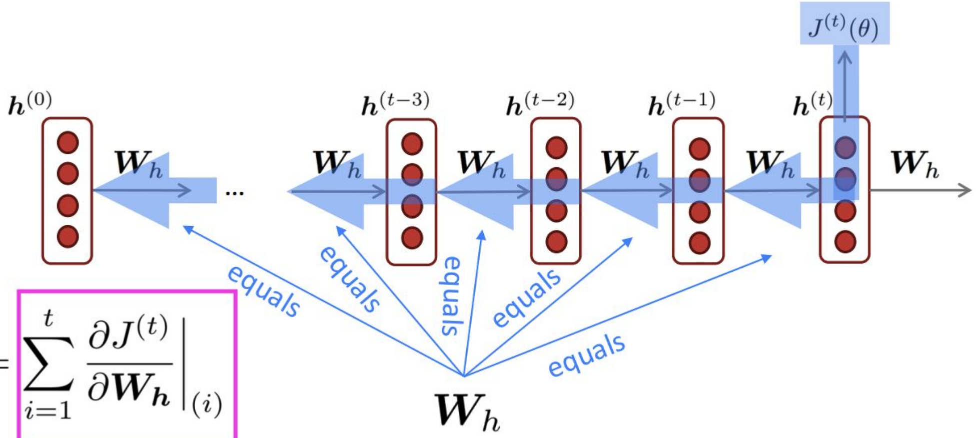


Recurrent Neural Networks



Recurrent Neural Networks

- Backpropagation through time



$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}} = \boxed{\sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W_h}}\right|_{(i)}}$$

Question: How do we calculate this?

Answer: Backpropagate over timesteps *i* = *t*, … ,0, summing gradients as you go. This algorithm is called **"backpropagation through time"** [Werbos, P.G., 1988, *Neural Networks* **1**, and others]

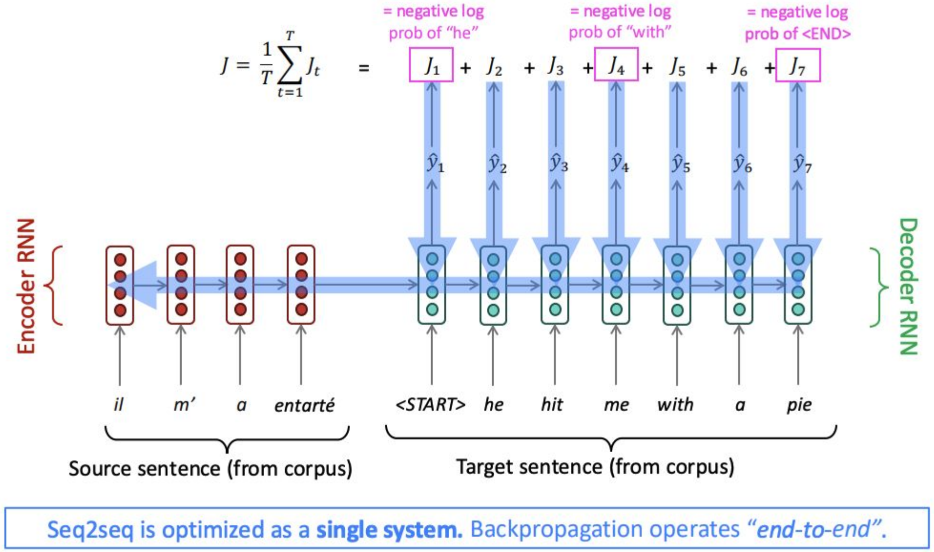
Apply the multivariable chain rule:

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)} \underbrace{\boxed{\frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}}}_{=1}$$

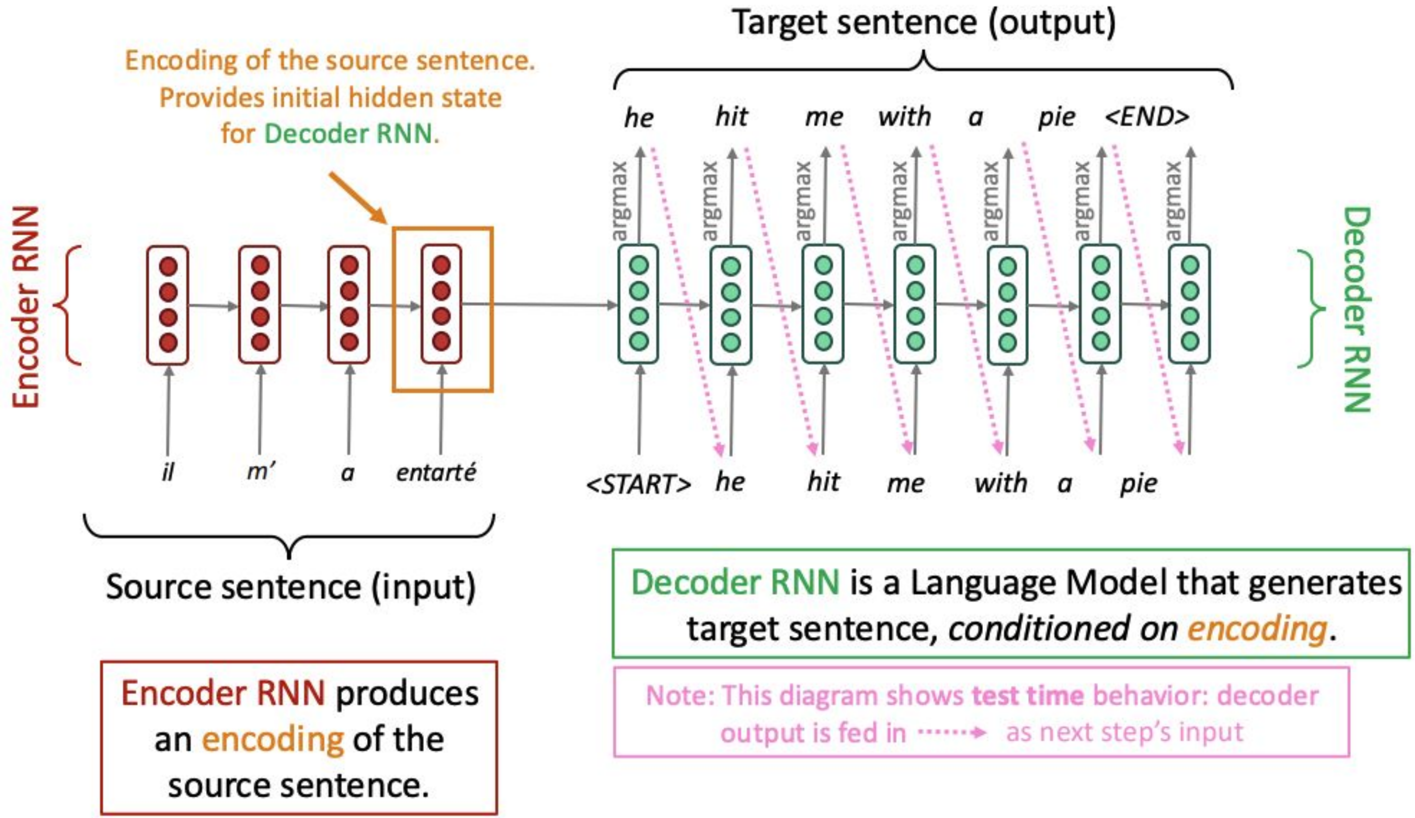
$$= \sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)}$$

Sequence 2 Sequence Modeling Using RNNs

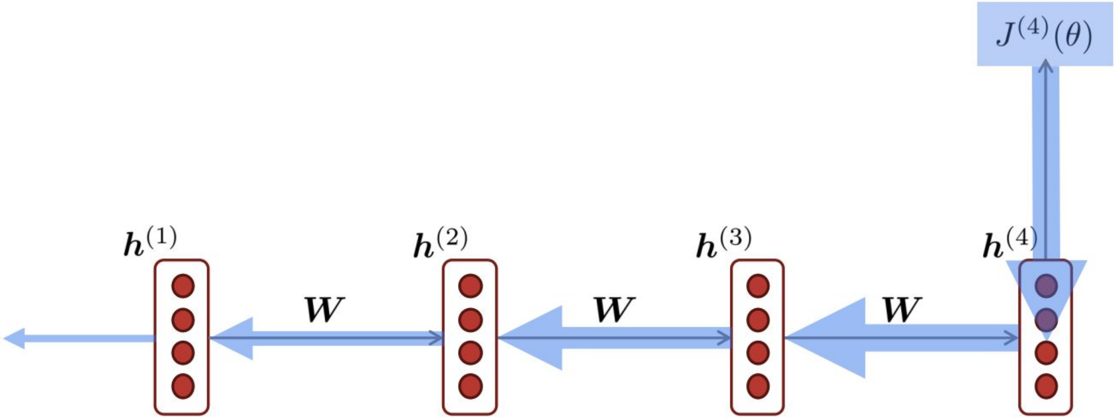
- The general notion here is an encoder-decoder model
  - One neural network takes input and produces a neural representation
  - Another network produces output based on that neural representation
- Many NLP tasks can be phrased as sequence-to-sequence:
  - Summarization
  - Dialogue
  - Code generation
  - Translation



Neural Machine Translation using RNNs



Vanishing Gradients in RNNs

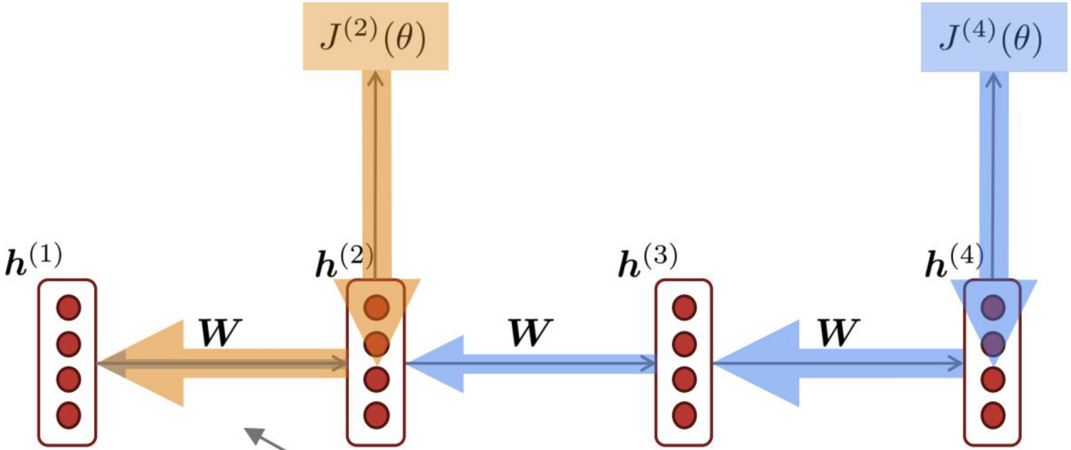


$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

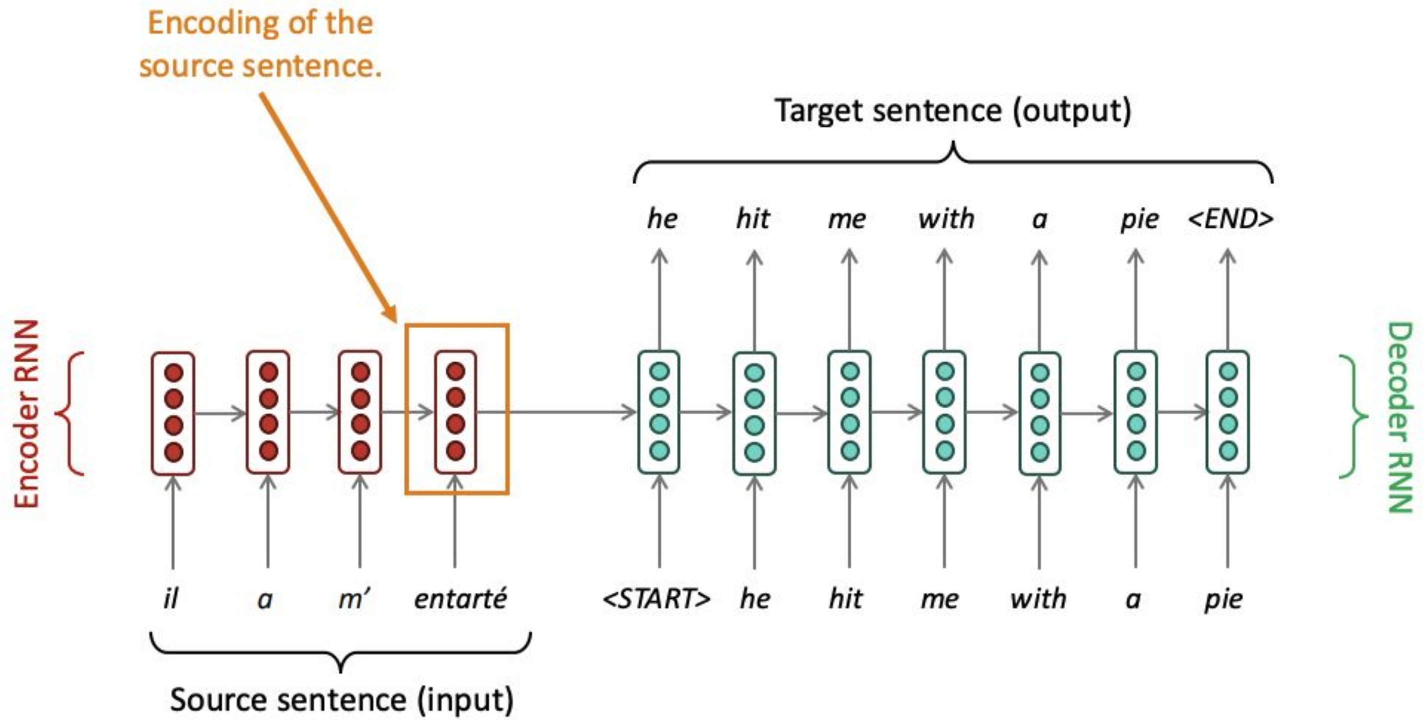
Vanishing Gradients in RNNs



Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

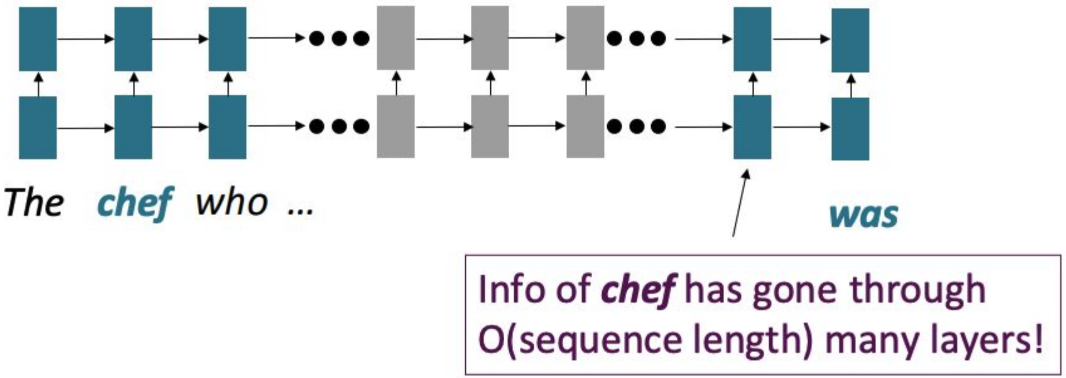
Bottleneck Problem in RNNs



Problems with this architecture?

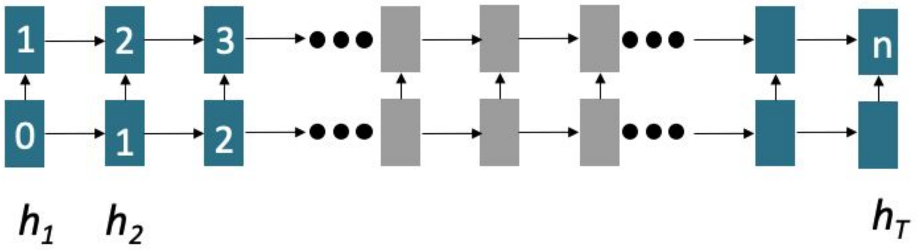
Lack of Parallelizability in RNNs

- O(sequence length) steps for distant word pairs to interact means:
  - Hard to learn long-distance dependencies (because gradient problems!)
  - Linear order of words is “baked in”; we already know linear order isn’t the right way to think about sentences…



Lack of Parallelizability in RNNs

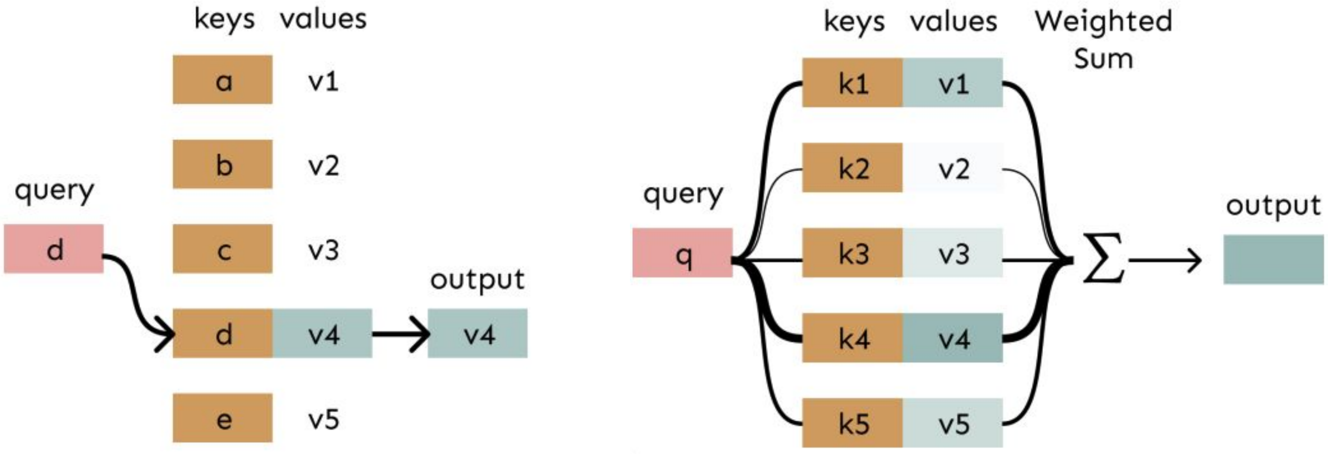
- Forward and backward passes have O(sequence length) non parallelizable operations
- GPUs can perform a bunch of independent operations at once!
- BUT! future RNN hidden states can't be computed in full before past RNN hidden states have been computed



Numbers indicate min # of steps before a state can be computed

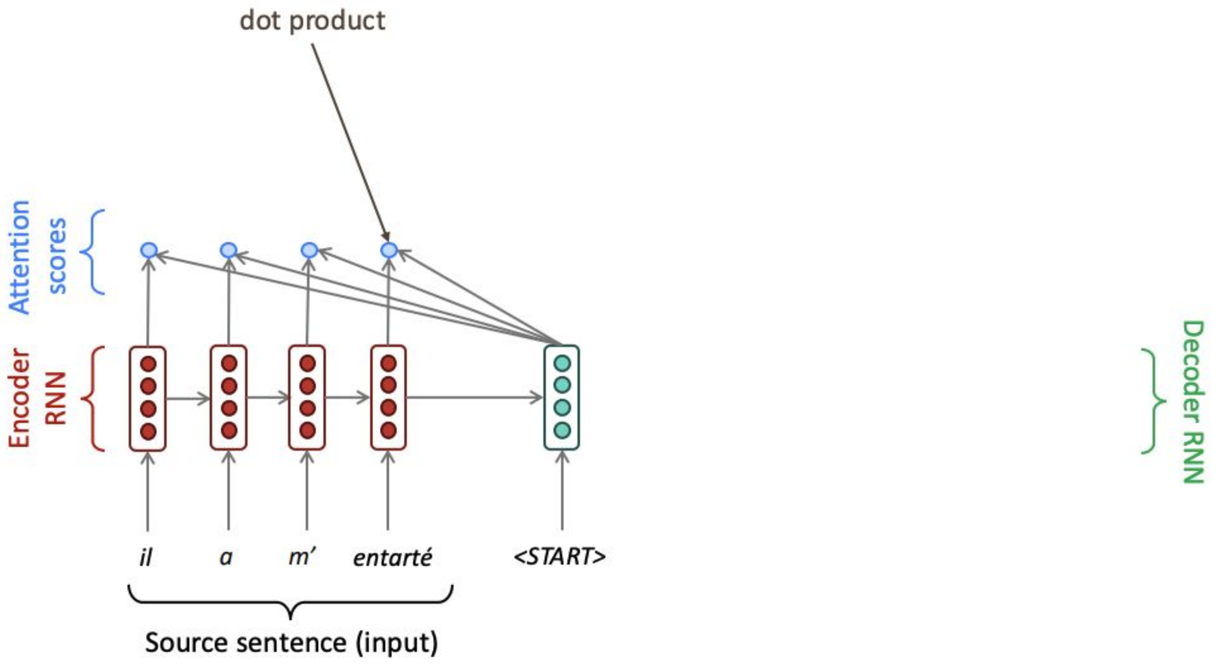
Attention is a solution!

- Attention provides a solution to the bottleneck problem!
- Core idea: on each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence!
- In attention, the query matches all keys softly, to a weight between 0 and 1. The key's values are multiplied by the weights and summed!

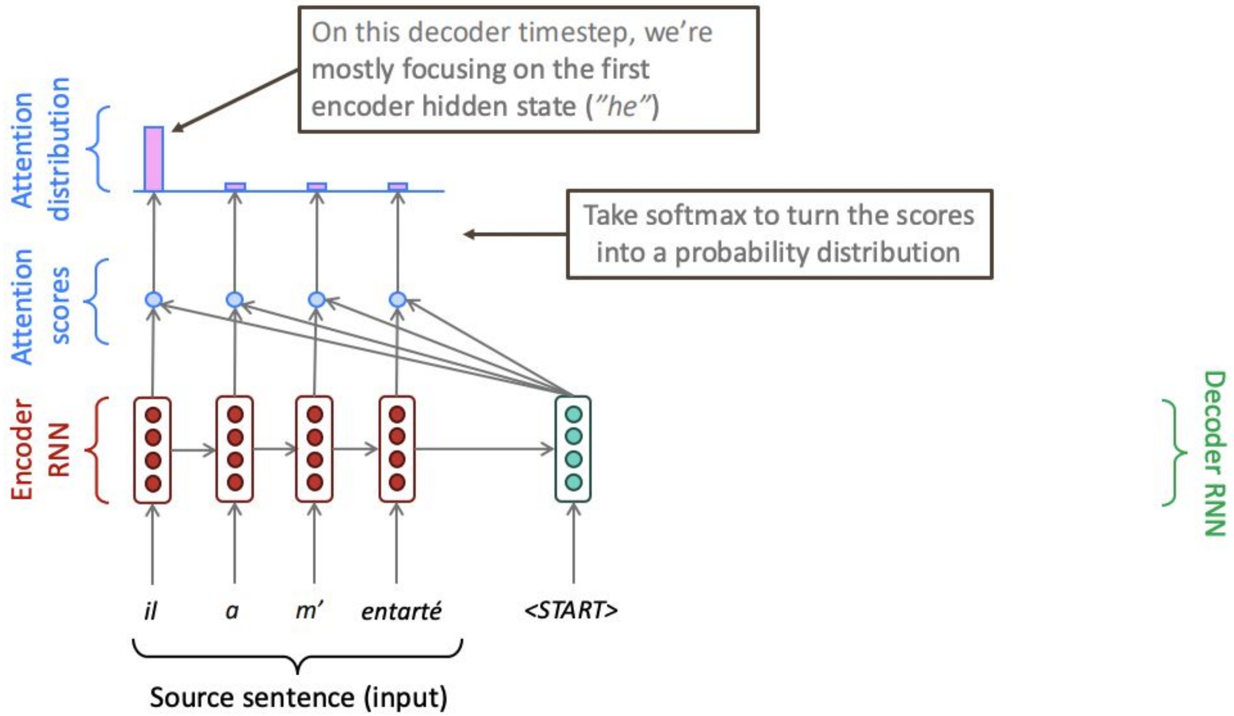


Attention in RNNs

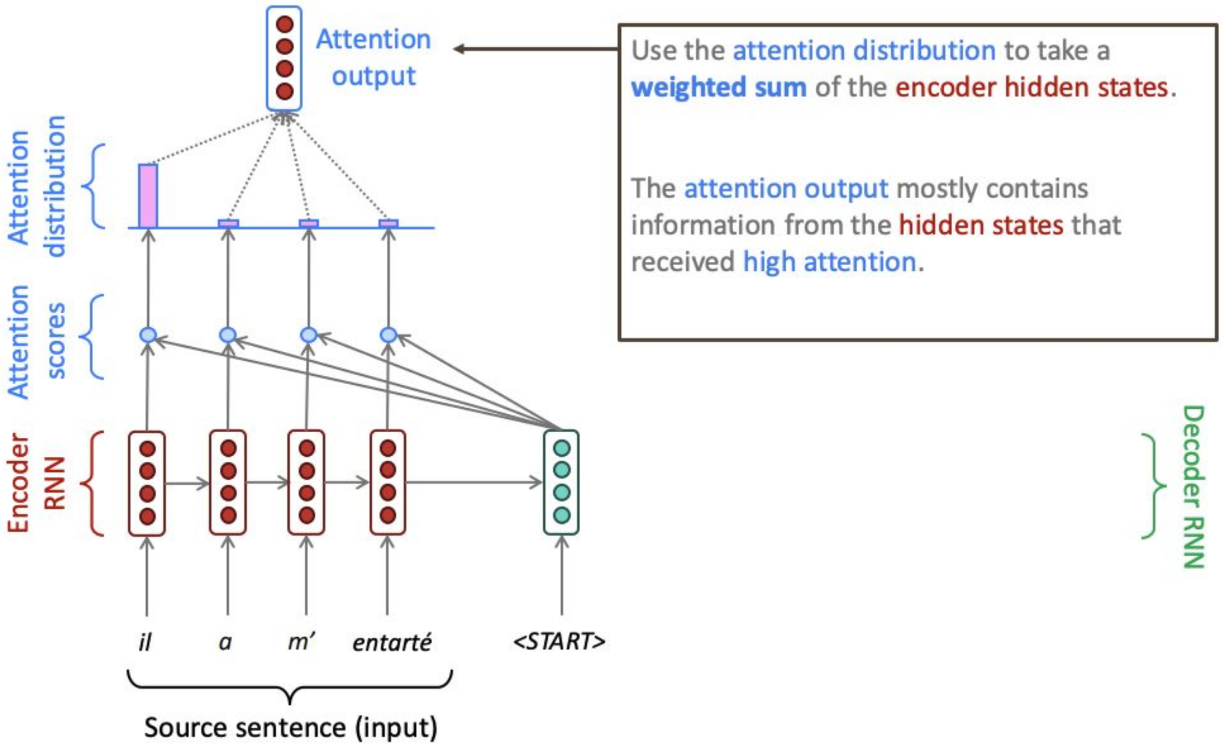
- On each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence.



Attention in RNNs



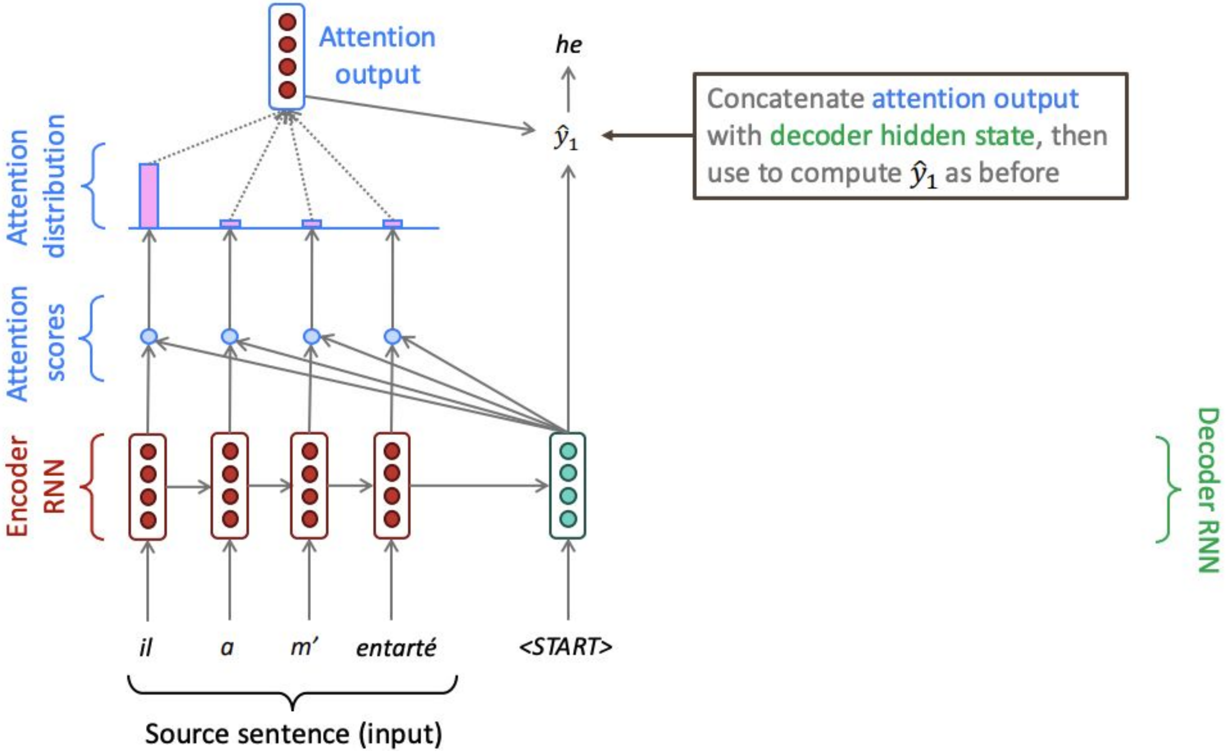
Attention in RNNs



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

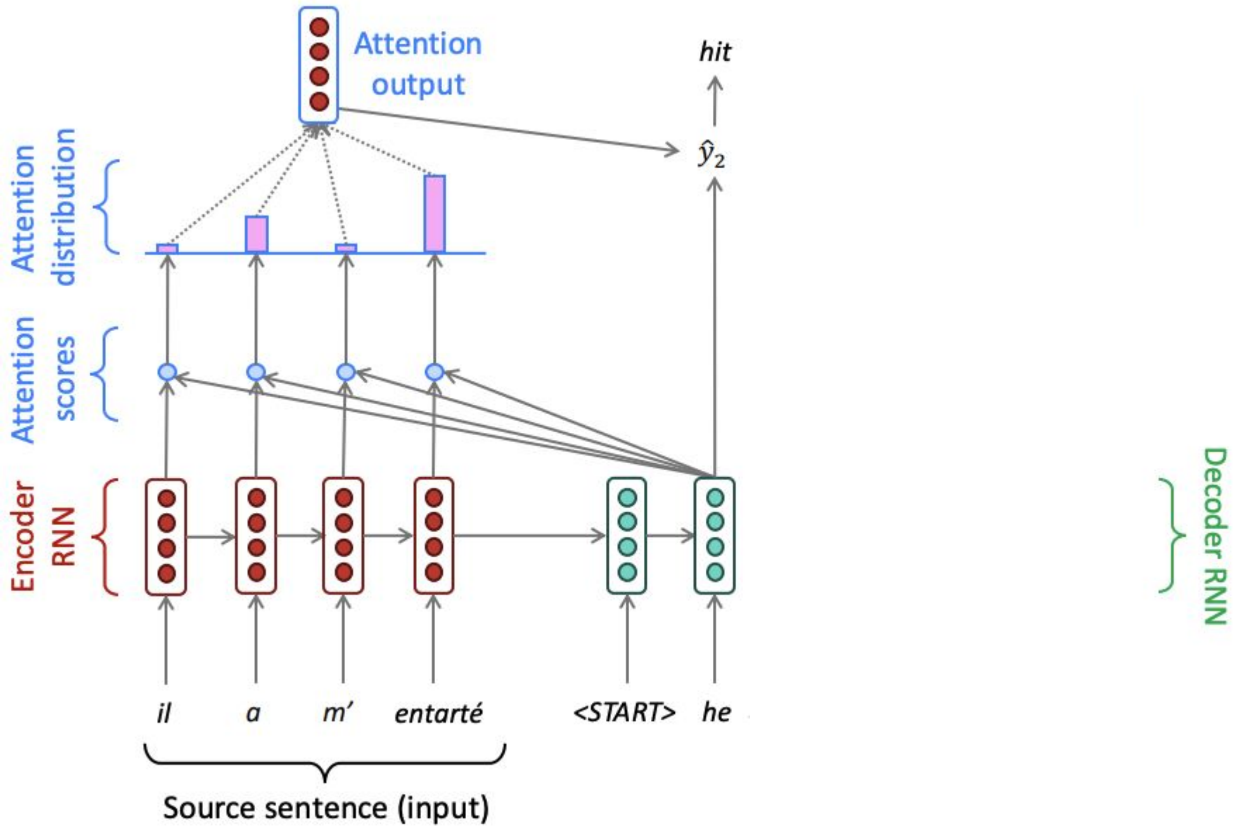
The attention output mostly contains information from the hidden states that received high attention.

Attention in RNNs

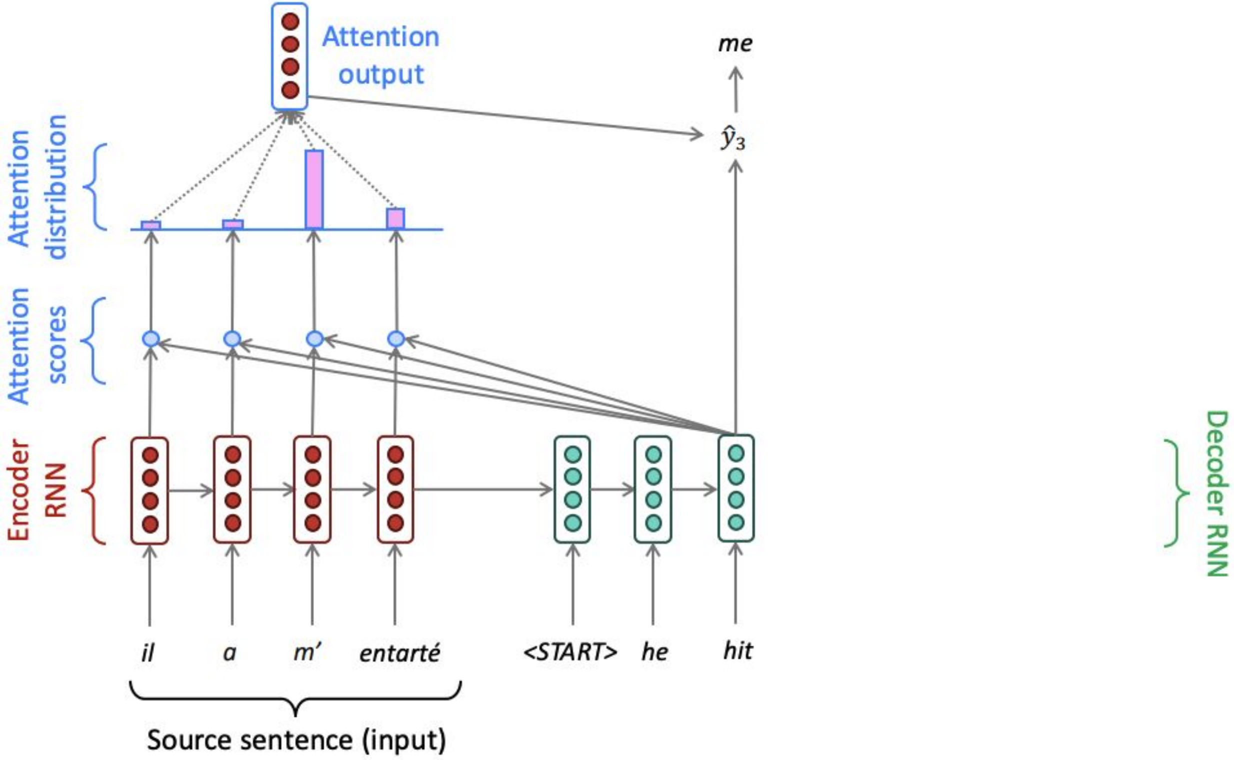


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Attention in RNNs



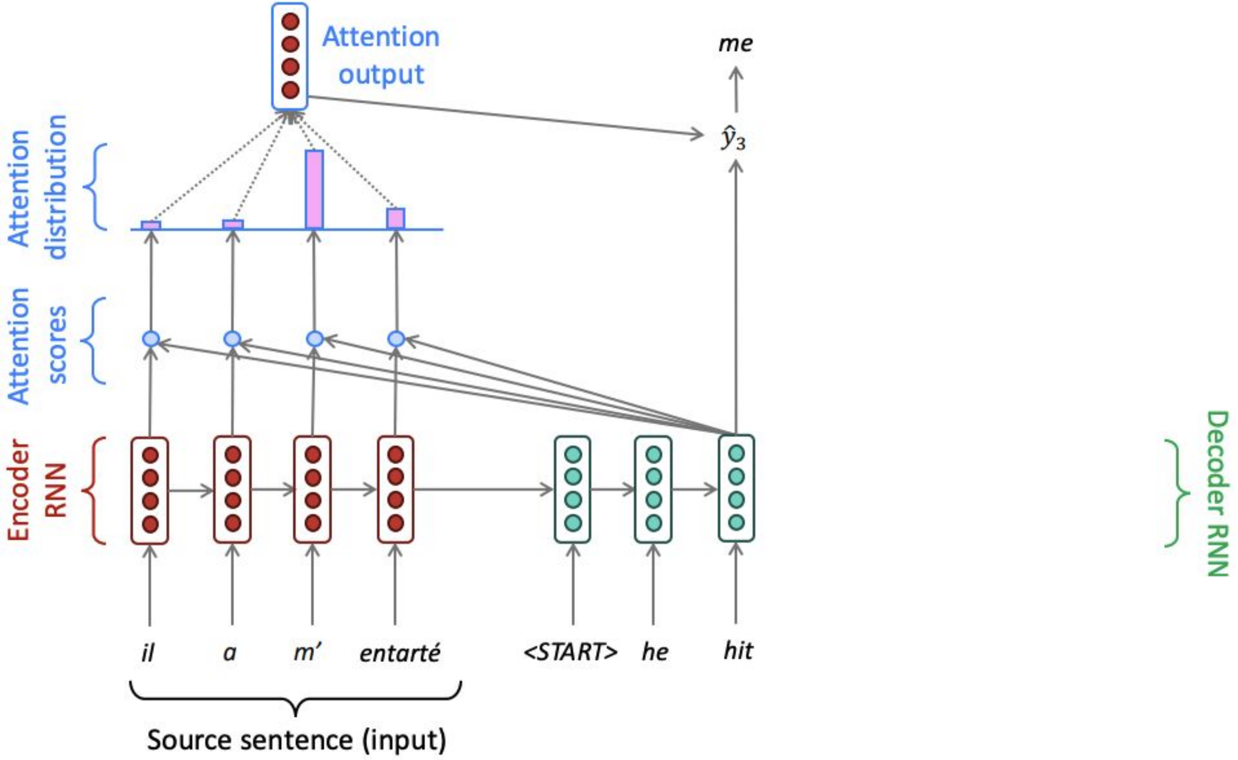
Attention in RNNs



Is Recurrent Necessary at All?

- Abstractly: Attention is a way to pass information from a sequence (x) to a neural network input. (h_t)
- This is also exactly what RNNs are used for – to pass information!
- Can we just get rid of the RNN entirely? Maybe attention is just a better way to pass information!
- The building block we need is self Attention!
- So far we saw cross-attention!

Attention in RNNs



Self-attention: Keys, Queries, and Values

Let $\boldsymbol{w}_{1:n}$ be a sequence of words in vocabulary V, like *Zuko made his uncle tea*.

For each $\boldsymbol{w}_i$, let $\boldsymbol{x}_i = E\boldsymbol{w}_{\boldsymbol{i}}$, where $E \in \mathbb{R}^{d\times|V|}$ is an embedding matrix.

1. Transform each word embedding with weight matrices $\mathrm{Q}, \mathrm{K}, \mathrm{V}$, each in $\mathbb{R}^{d\times d}$

$$\boldsymbol{q}_i = Q\boldsymbol{x}_{\boldsymbol{i}} \text{ (queries)} \qquad \boldsymbol{k}_i = K\boldsymbol{x}_{\boldsymbol{i}} \text{ (keys)} \qquad \boldsymbol{v}_i = V\boldsymbol{x}_{\boldsymbol{i}} \text{ (values)}$$

2. Compute pairwise similarities between keys and queries; normalize with softmax

$$\boldsymbol{e}_{ij} = \boldsymbol{q}_{\boldsymbol{i}}^{\top}\boldsymbol{k}_{\boldsymbol{j}} \qquad \boldsymbol{\alpha}_{ij} = \frac{\exp(\boldsymbol{e}_{ij})}{\sum_{j'}\exp(\boldsymbol{e}_{ij'})}$$

3. Compute output for each word as weighted sum of values

$$\boldsymbol{o}_i = \sum_{\boldsymbol{j}} \boldsymbol{\alpha}_{ij}\,\boldsymbol{v}_i$$

Positional Embedding in Self-Attention

- Since self-attention doesn't build in order information, we need to encode the order of the sentence in our keys, queries, and values
- Consider representing each sequence index as a vector

$\boldsymbol{p}_i \in \mathbb{R}^d$, for $i \in \{1,2,\dots,n\}$ are position vectors

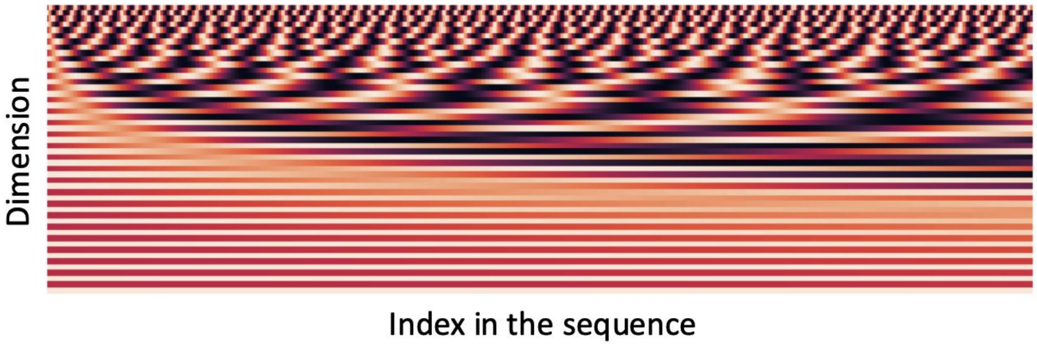
$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{x}_i + \boldsymbol{p}_i$$

In deep self-attention networks, we do this at the first layer! You could concatenate them as well, but people mostly just add...

Sinusoida Positional Embedding

- Sinusoidal position representations: concatenate sinusoidal functions of varying periods
- Periodicity indicates that maybe "absolute position" isn't as important
- It can extrapolate to longer sequences as periods restart!

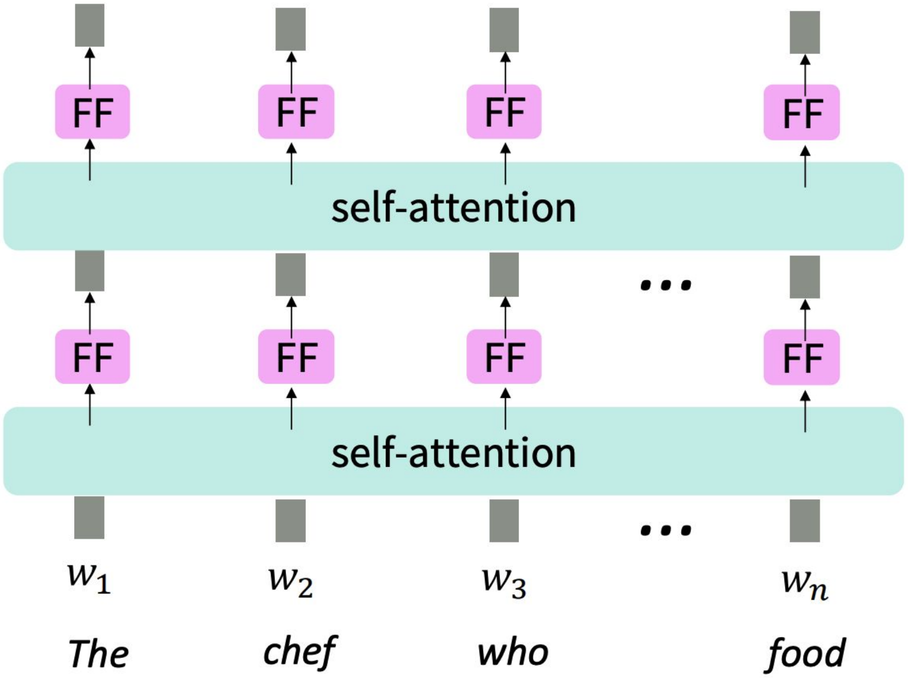
$$\boldsymbol{p}_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$



Non-Linearity in Self-Attention

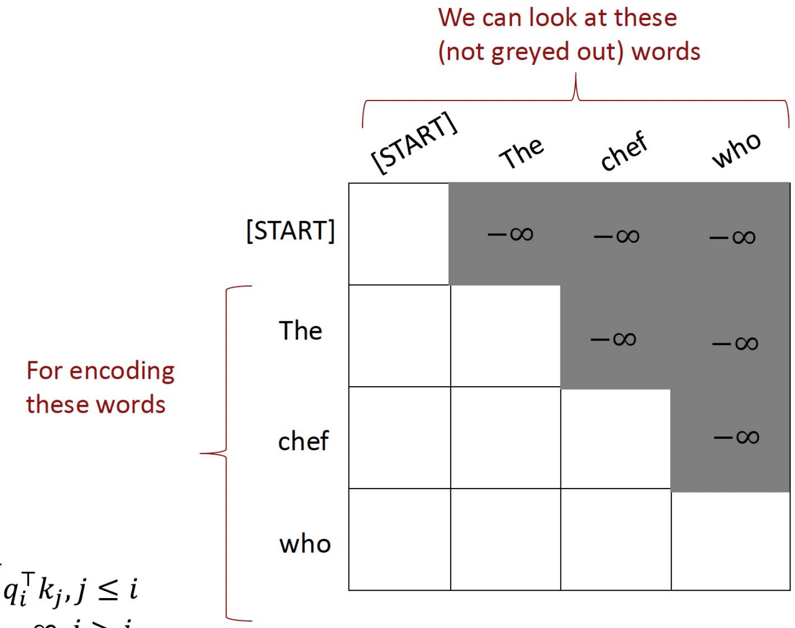
- Easy fix: add a feed-forward network to post-process each output vector.

$$m_i = MLP(\text{output}_i)$$
$$= W_2 * \text{ReLU}(W_1\ \text{output}_i + b_1) + b_2$$



Causal Masking in Self-Attention

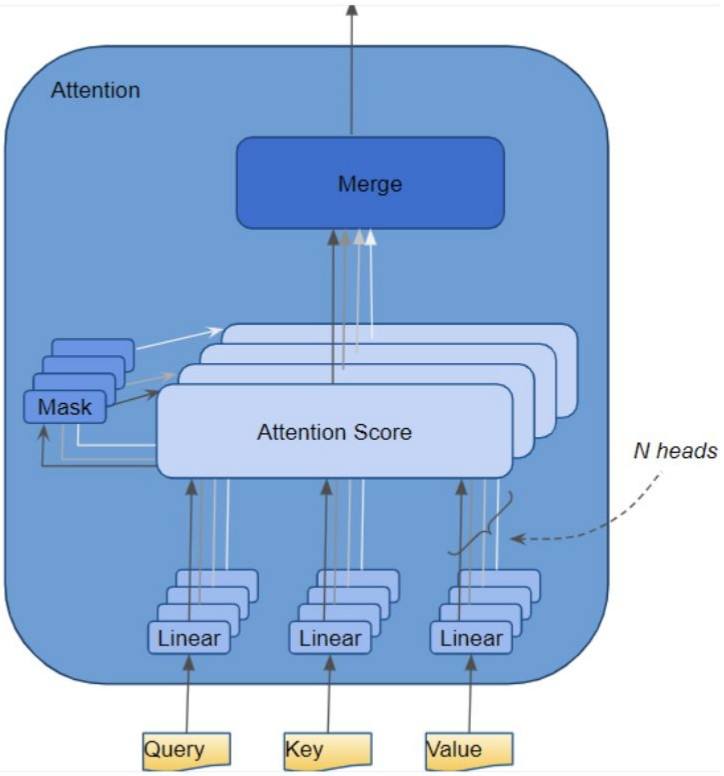
- For causality, we need to ensure not to peek at the future.
- At each timestep, we could change the set of keys and queries to only include past words!



$$\begin{cases} q_i^\top k_j, j \le i \\ -\infty, j > i \end{cases}$$

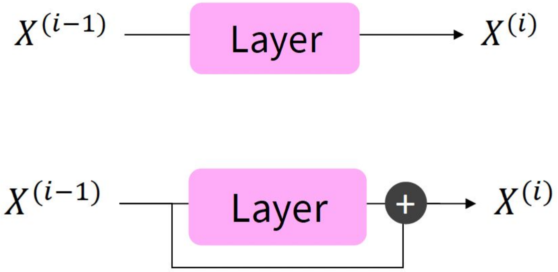
Multi-Head Self-Attention Layer

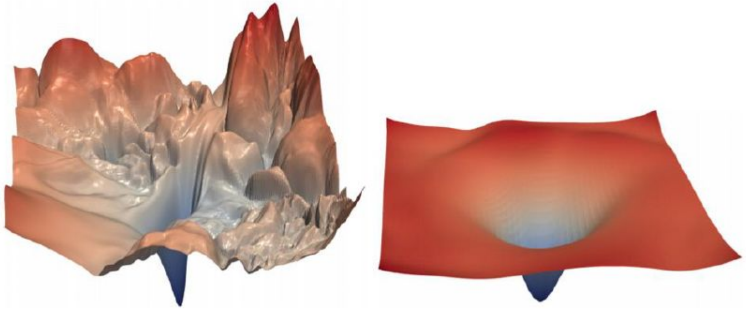
- The Attention module splits its Query, Key, and Value parameters N-ways and passes each split independently through a separate head.
- Calculations are combined together to produce a final attention score.
- Greater power to encode multiple relationships and nuances for each word.



Residual Connections

- A trick to help models learn better!
- Gradient is 1 through residual connection
- Bias toward identity function.





[no residuals] [residuals]

[Loss landscape visualization, Li et al., 2018, on a ResNet]

Layer Normalization

- A trick to help models train faster.
- Cut down on uninformative variation in hidden vectors by normalizing to unit mean and standard deviation within each layer.

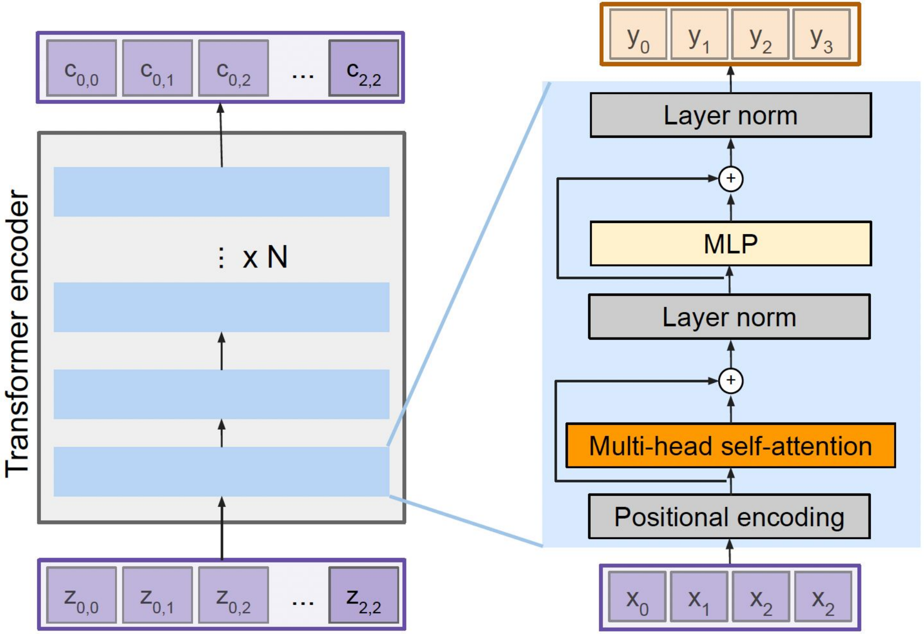
$$\text{output} = \frac{x - \mu}{\sqrt{\sigma} + \epsilon} * \gamma + \beta$$

Normalize by scalar mean and variance

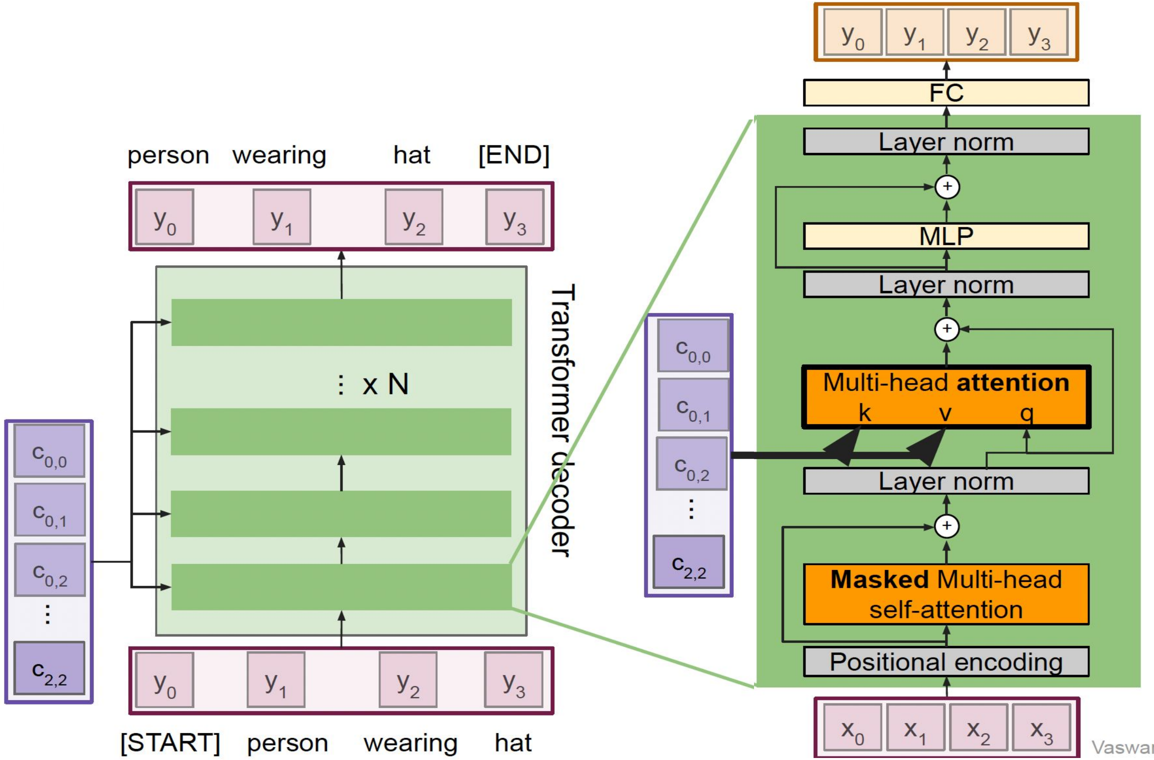
Modulate by learned elementwise gain and bias

Transformer Encoder

- Position representation
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- Nonlinearities
  - Frequently implemented as a simple feedforward network.
- Masking
  - Keep information about the future from "leaking" to the past.



Transformer Decoder



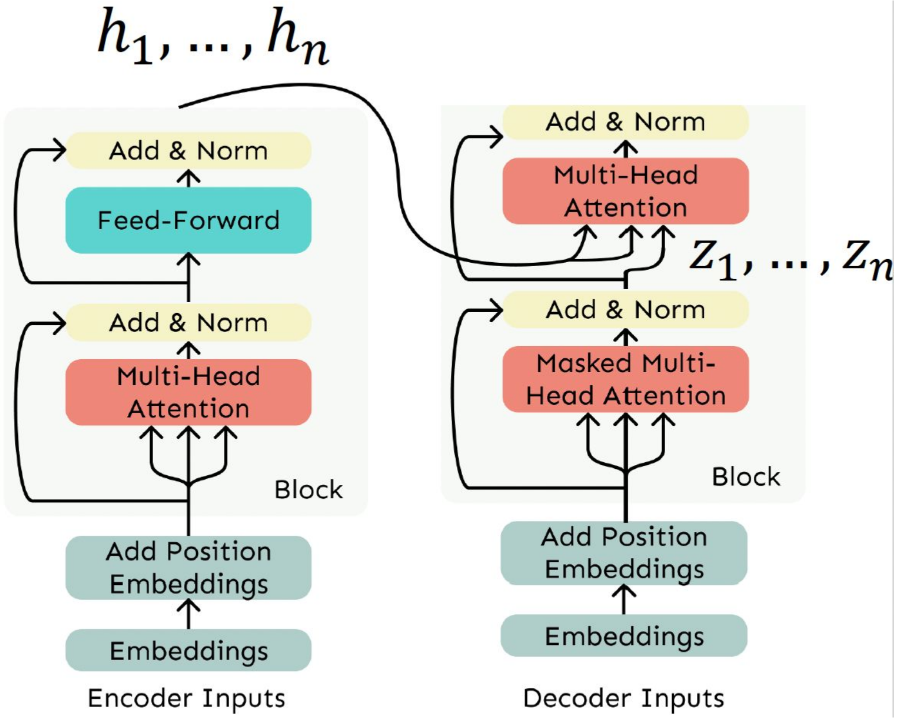
Vaswani et al, "Attention is all you need", NeurIPS 2017

Multi-head attention block attends over the transformer encoder outputs.

For image captions, this is how we inject image features into the decoder.

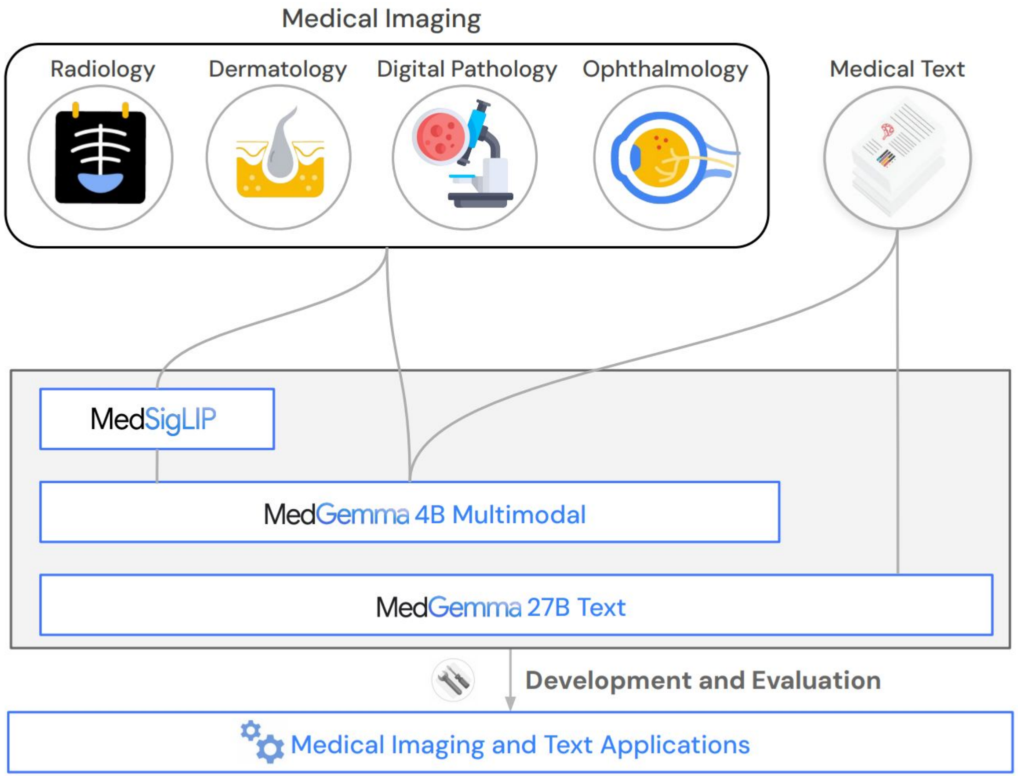
Cross Attention

- Self-attention:
  - Keys, queries, and values from same source
- Cross-attention
  - The keys and values are from encoder (like a memory)
  - The queries are from the decoder



Inside an LLM

Vision-Language Models



Prompt

You are an expert radiologist. Please succinctly describe the findings for the above chest x-ray.

MedGemma

The chest X-ray shows a large left pneumothorax with a chest tube in place. There is also a possible nodule in the right upper lobe. The mediastinum is midline. The heart size is normal. There is no evidence of pulmonary edema.

Reference Information: Radiologist Impression
Moderately large left hydropneumothorax (predominantly air-containing) with incompletely expanded left lung. If persistent, consider ex vacuo pneumothorax/trapped lung condition. Multiple right lung nodules, highly suspicious for metastases.

Prompt

Describe this image and provide the most likely condition. Keep your answer brief

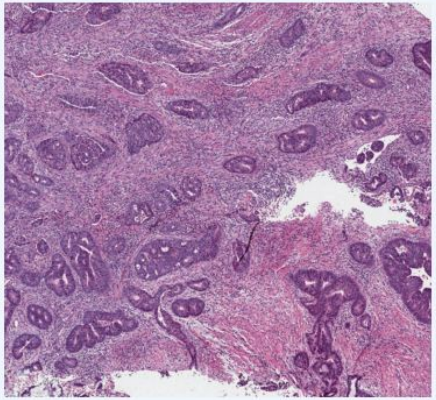
MedGemma

The chest X-ray shows a significant opacity in the left lower lung field, obscuring the costophrenic angle. This suggests a possible pleural effusion or consolidation. The right lung appears relatively clear. The heart size is within normal limits. The mediastinum is unremarkable.

Reference Information: Radiologist Impression
Lower left lung consolidation. Small-medium left pleural effusion. Similar though lesser findings right side.

Prompt

Describe the findings and most likely diagnosis in this image. Be concise.



MedGemma

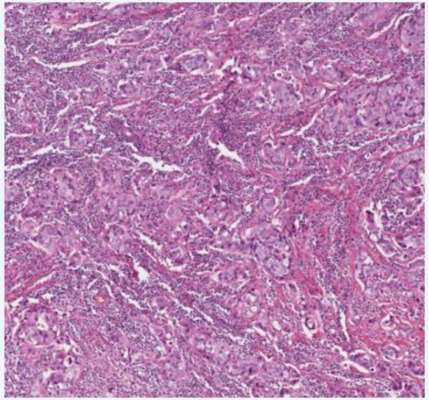
The image shows a histological section of colon tissue. The findings include a mix of glandular structures and stromal tissue. There are areas of inflammation and some degree of cellular atypia.

Most likely diagnosis: **Colorectal adenocarcinoma**

Reference Information:
Colon Adenocarcinoma

Prompt

Describe this image and provide the most likely condition. Keep your answer brief.



MedGemma

The image shows a high-grade invasive carcinoma with significant cellular atypia, prominent nucleoli, and increased mitotic activity. This indicates a malignant tumor with aggressive growth.

Most likely diagnosis: **High-grade invasive carcinoma** (the specific type would require further information about the tissue origin).

Reference Information:
Invasive Ductal Carcinoma (Breast Cancer)