Recurrent Models and Attention

ECE 57000 — October 28, 2026

David I. Inouye

Shared recurrent computation learns from every prediction loss

- Shift targets by one token; split whole emails before forming training examples.
- Reuse one transition across positions; BPTT accumulates gradients through every use.
- GRU gates mix retained state with candidate content; longer sequences still require more steps.

Today: finish the RNN limitations, then replace the state chain with attention.

Recurrent memory creates information and training bottlenecks

Limitation	Why it matters
Fixed-width memory	Earlier information must survive repeated state updates; a final-state summary must represent the whole sequence.
Long gradient paths	Distant credit passes through many Jacobians, complicating optimization.
Sequential computation	State $\boldsymbol{h}_\ell$ must wait for $\boldsymbol{h}_{\ell-1}$, limiting training parallelism across positions.

Next: retain all position representations and combine them directly.

Attention and Position

David I. Inouye

Dynamic Content · Last Updated: October 26, 2026

Self-attention combines representations from the same sequence

Arc 6: Sequence-model architectures → Attention and position

1. Write the complete **self-attention equation**.
2. Unpack its **input-dependent soft lookup**.
3. Extend to **cross-attention**, then restrict access for our next-token task.

Position information comes next.

Self-attention is one input-dependent matrix operation

Input $X \in \mathbb{R}^{L \times d}$: **one row per position**. Biases omitted.

Projection	Learned matrix	Result shape
$Q = XW_Q$	$W_Q \in \mathbb{R}^{d \times d_k}$	$L \times d_k$
$K = XW_K$	$W_K \in \mathbb{R}^{d \times d_k}$	$L \times d_k$
$V = XW_V$	$W_V \in \mathbb{R}^{d \times d_v}$	$L \times d_v$

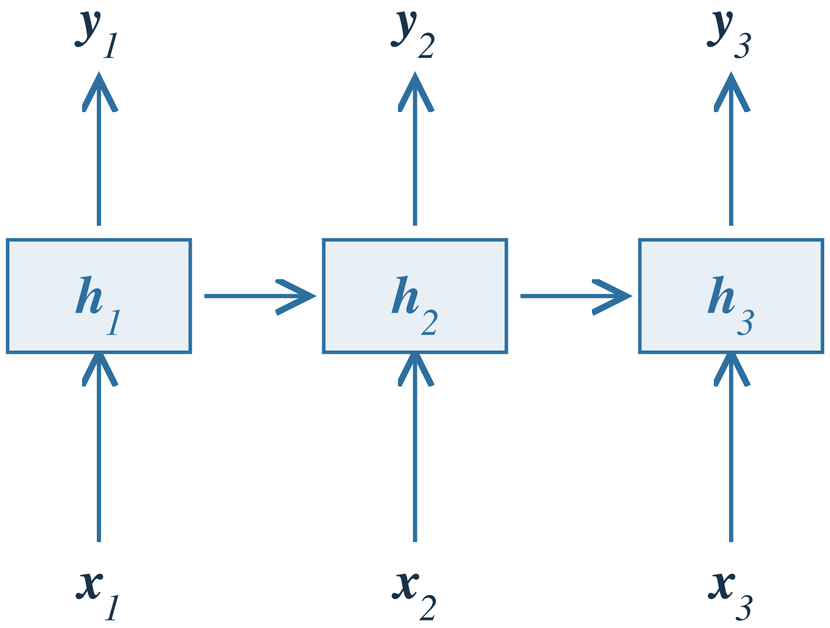
$$A = \mathrm{softmax}_{\mathrm{row}}\left(\frac{QK^{\mathsf{T}}}{\sqrt{d_k}}\right) \in \mathbb{R}^{L \times L}, \qquad Y = AV \in \mathbb{R}^{L \times d_v}$$

Attention matrix A: input-dependent weights, one row per output position.

$$\mathrm{SA}(X) = \mathrm{softmax}_{\mathrm{row}}\left(\frac{(XW_Q)(XW_K)^{\mathsf{T}}}{\sqrt{d_k}}\right)(XW_V)$$

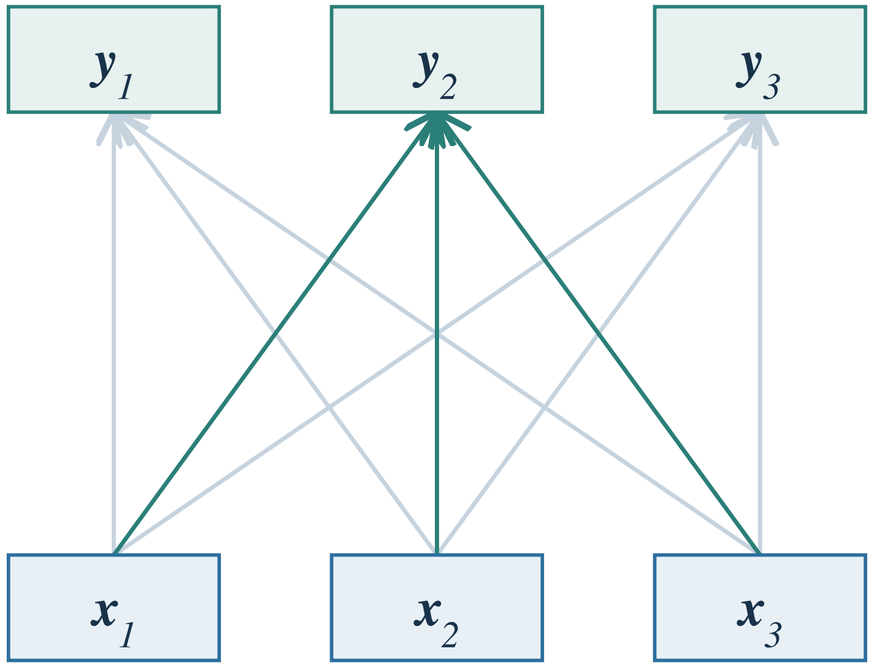
Self-attention combines inputs directly, with input-dependent weights

RNN: context follows a state chain



Later states wait for earlier states

Self-attention: direct input mixing



Each output has input-dependent weights

Each y_ℓ combines information from **all rows of** X.

The learned projection matrices stay fixed; the combination weights are **input-dependent**.

Self-attention removes the sequential chain within a layer

RNN	Self-attention
$\boldsymbol{h}_1 \to \boldsymbol{h}_2 \to \cdots \to \boldsymbol{h}_L$	$X \to \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$ together
Distant information travels through successive updates	Each output directly combines all input positions
Position ℓ waits for the preceding state	Outputs share the same available input collection

Shorter paths for learning; parallel operations across positions during training.

Attention builds an input-dependent weighted combination

For each input row, form a value: $\boldsymbol{v}_j = f_v(\boldsymbol{x}_j) = \boldsymbol{x}_j W_V$.

$$\boldsymbol{y}_\ell = \sum_{j=1}^{L} a_{\ell j} \boldsymbol{v}_j, \qquad a_{\ell j} \geq 0, \quad \sum_{j=1}^{L} a_{\ell j} = 1$$

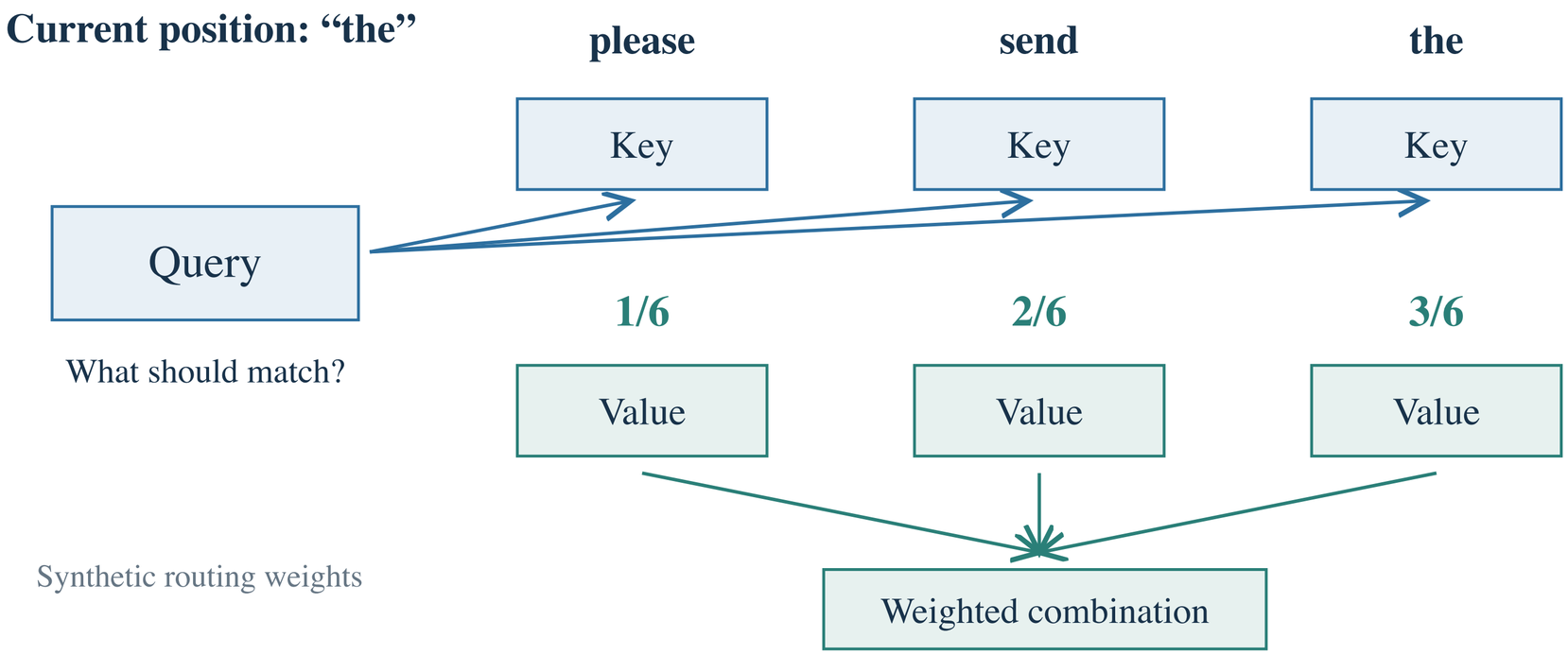
- The score for pair (ℓ, j) is a function of **both inputs**: $s_{\ell j} = f_s(\boldsymbol{x}_\ell, \boldsymbol{x}_j)$.
- $a_{\ell j} = \mathrm{softmax}_j(s_{\ell 1}, \ldots, s_{\ell L})$: normalize against **all candidate inputs**.

How do queries and keys turn these input pairs into useful weights?

Attention is a soft lookup in a key–value table

Dictionary lookup: an exact key selects one stored value.

Attention: a query scores every key, then blends the paired values.



Query: features to match. **Key:** matching features. **Value:** features to combine.

Softmax turns matching scores into relative combination weights

$$s_{\ell j} = \frac{\boldsymbol{q}_\ell \boldsymbol{k}_j^\top}{\sqrt{d_k}}, \qquad a_{\ell j} = \frac{\exp(s_{\ell j} - m_\ell)}{\sum_{k=1}^{L} \exp(s_{\ell k} - m_\ell)}, \qquad m_\ell = \max_k s_{\ell k}$$

- A larger dot product gives a larger weight **relative to the other scores**.
- Subtracting the row maximum preserves the weights and avoids large exponentials.

These weights are **not probabilities that a token is true or important**.

Next: calculate one complete weighted combination.

One weighted combination uses values, not token IDs

Synthetic one-feature query $q = [1]$; the query is at `the`.

Source token	Key	Score	Softmax weight	Value v_j
`please`	$[0]$	0	$1/6$	$[1, 0]$
`send`	$[\log 2]$	$\log 2$	$2/6$	$[0, 2]$
`the`	$[\log 3]$	$\log 3$	$3/6$	$[2, 2]$

$$y = \tfrac{1}{6}[1, 0] + \tfrac{2}{6}[0, 2] + \tfrac{3}{6}[2, 2] = [7/6, 5/3]$$

This is a context feature vector. A later vocabulary head predicts the next token.

Matrix multiplication computes all output states together

Stack token representations as **rows** of $X \in \mathbb{R}^{L \times d}$.

$$S = QK^\top / \sqrt{d_k}, \qquad A = \mathrm{softmax}_{\mathrm{row}}(S), \qquad Y = AV$$

Object	Shape	Meaning
Q, K	$L \times d_k$	Query/key features per token
$QK^\top$	$L \times L$	Query rows × key columns
A	$L \times L$	One normalized weight row per query
$Y = AV$	$L \times d_v$	Combined value features per token

One row of A gives the input-dependent weights for one output row of Y.

Divide by $\sqrt{d_k}$ to keep score fluctuations comparable

Assume independent query/key entries with **mean** 0 **and variance** 1.

$$z = \boldsymbol{q}\boldsymbol{k}^\top = \sum_{i=1}^{d_k} q_i k_i, \qquad \mathbb{E}[z] = 0, \qquad \mathrm{Var}(z) = d_k$$

Positive and negative terms partly cancel. **Variances add; standard deviations grow as $\sqrt{d_k}$.**

Dividing by $\sqrt{d_k}$ keeps the score standard deviation at 1**, for any width**.

Example, $d_k = 20$**:** two keys with a score gap of $\sqrt{20} \approx 4.47$.

Scores	Softmax weights
$[0, \sqrt{20}]$	$[1.1\%, 98.9\%]$
Divide by $\sqrt{20}$: $[0, 1]$	$[26.9\%, 73.1\%]$

Avoid near-one-hot attention caused merely by width: saturated softmax has small gradients.

Cross-attention gets queries and context from different sequences

Query-side inputs $X \in \mathbb{R}^{L_q \times d_x}$; context inputs $C \in \mathbb{R}^{L_c \times d_c}$.

$$Q = XW_Q, \qquad K = CW_K, \qquad V = CW_V$$

$$\mathrm{CA}(X, C) = \mathrm{softmax}_{\mathrm{row}}\left(\frac{(XW_Q)(CW_K)^\mathsf{T}}{\sqrt{d_k}}\right)(CW_V)$$

- Example: **question tokens** query representations of a **passage**.
- Weight matrix: $L_q \times L_c$. Output: one row per query, $L_q \times d_v$.

Self-attention is the case $C = X$. The same soft-lookup mechanism applies.

Next-token prediction determines which context is available

At position ℓ, we predict $w_{\ell+1}$.

> **Which input tokens may this output attend to?**
>
> **A. Past and current tokens: $w_0, \ldots, w_\ell$**
>
> **B. Later tokens: $w_{\ell+1}, w_{\ell+2}, \ldots$**
>
> **C. Both earlier and later tokens**

A. Later inputs include the answer we are trying to predict.

We need an **autoregressive (causal) mask** to enforce that restriction.

A causal mask removes future positions before softmax

Key position / input token available to read

Query position	<BOS>	please	send	the	report
<BOS>	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
please	0	0	$-\infty$	$-\infty$	$-\infty$
send	0	0	0	$-\infty$	$-\infty$
the	0	0	0	0	$-\infty$
report	0	0	0	0	0

Add mask to scores BEFORE row-wise softmax

$$A = \mathrm{softmax}_{\mathrm{row}}(S + M), \qquad Y = AV$$

Position tells the model where a matching token occurred

Arc 6: Attention and position

1. Combine values using query–key matches.
2. Restrict future-token access for next-token prediction.
3. **Locate tokens with position information.**

Then combine information in several learned subspaces.

Self-attention without positions is permutation equivariant

For any **permutation matrix** P, with **no mask or positional encoding**:

$$\mathrm{SA}(PX) = P\,\mathrm{SA}(X)$$

The outputs are the same, just reordered.

Input rows	Output rows
Item A · Item B · Item C	$\boldsymbol{y}_A \cdot \boldsymbol{y}_B \cdot \boldsymbol{y}_C$
Item C · Item A · Item B	$\boldsymbol{y}_C \cdot \boldsymbol{y}_A \cdot \boldsymbol{y}_B$

Useful for an unordered set: changing the storage order should not change what each item means.

A sequence needs order beyond an unordered collection

Same words, different order	Requested action
Send the **report before** the slides.	Report first, slides second.
Send the **slides before** the report.	Slides first, report second.

Focus on **report**: its relationship to **before** has changed.

Without a mask or positions, self-attention gives **report the same output vector in both sentences**—just in a different row.

We want token representations to reflect order-dependent relationships.