上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 21: Attention

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

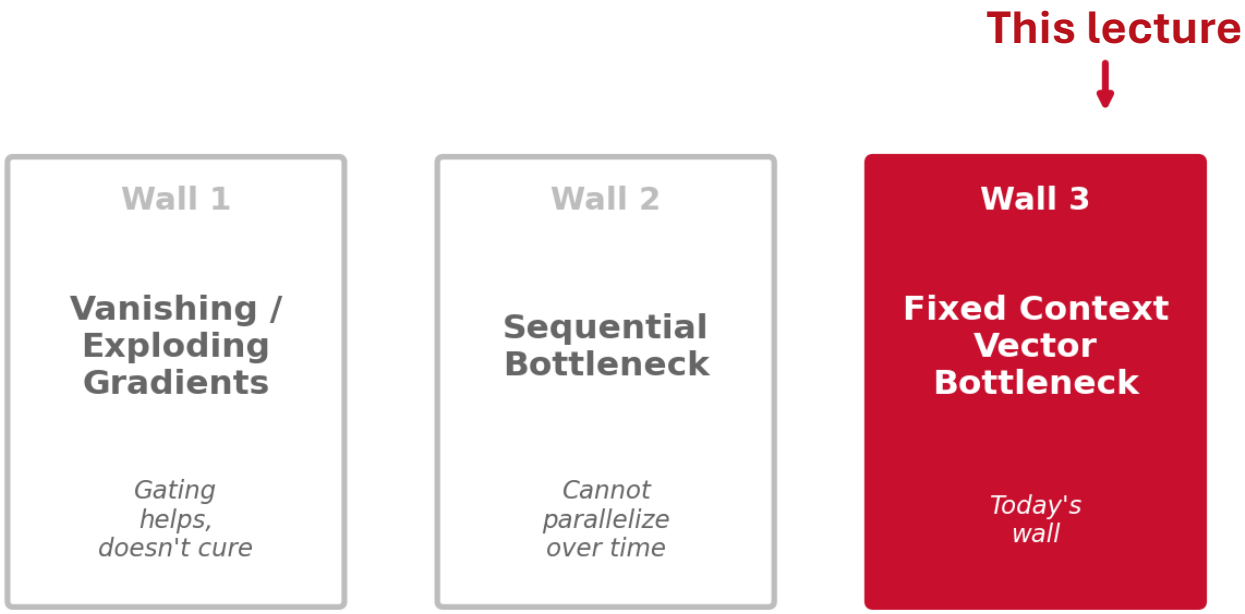
https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

Where We Stopped Last Lecture

L22 ended with the three walls of RNNs:

1. Vanishing / exploding gradients (gating helps, doesn't cure)
2. Sequential bottleneck — can't parallelize over time
3. Fixed context-vector bottleneck



Objectives

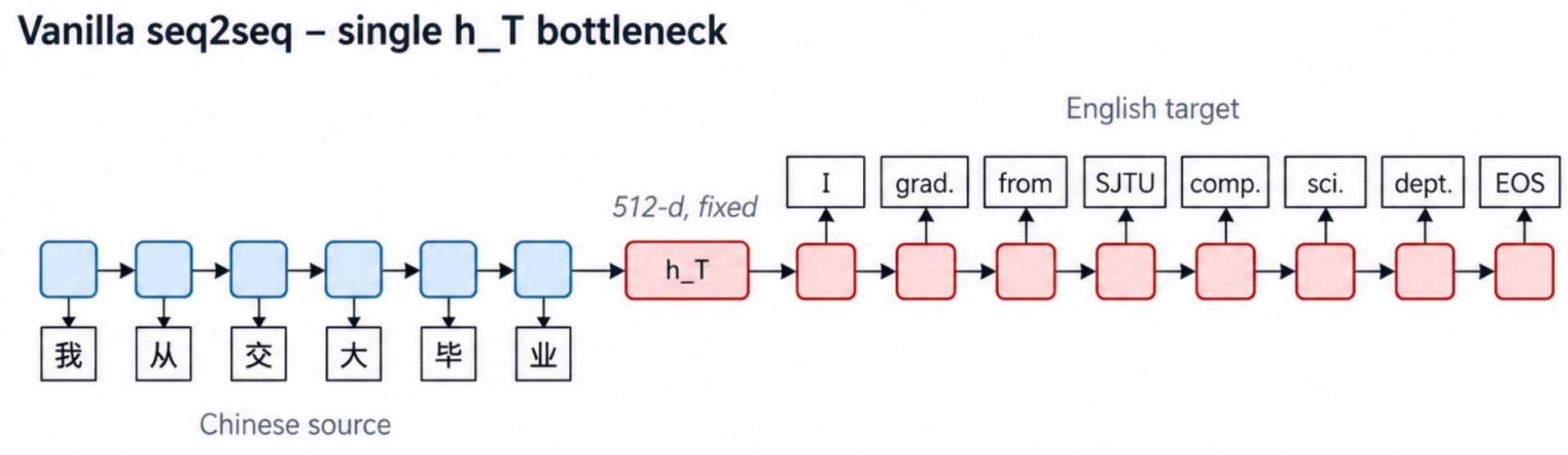
After this lecture you will be able to:

- Diagnose the encoder bottleneck quantitatively (information-theoretic argument)
- Derive the attention mechanism from two independent starting points
- Distinguish Bahdanau (additive) and Luong (multiplicative) variants and pick between them
- Predict where attention alignments should peak in a translation task
- Foresee why attention will replace recurrence entirely (next lecture)

1. The Encoder Bottleneck

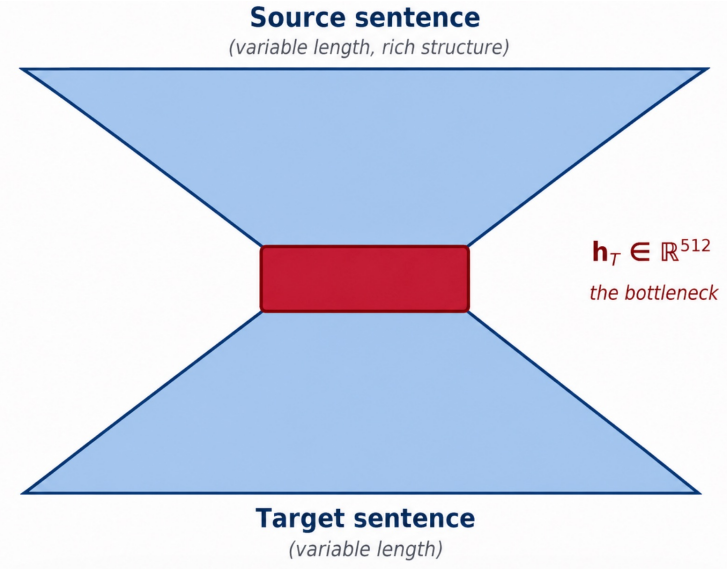
Recap: Seq2seq with RNNs (Sutskever et al., 2014)

- Encoder: $x_1, \ldots, x_T \rightarrow h_1, \ldots, h_T$, final state h_T is the "thought vector"
- Decoder: initialized from h_T; generates $y_1, \ldots, y_{T'}$ autoregressively
- Trained end-to-end with cross-entropy loss
- First neural system to beat statistical MT on WMT'14 En→Fr



The Bottleneck

- Decoder's only window into the source: $h_T \in \mathbb{R}^d$ (typically d = 512 or 1024)
- Source sentence has variable length T and arbitrary content



How many bits of information can a 512-dim float carry?

How many bits is a sentence?

- Quick estimate: 512 × 32 bits ≈ 16 kb capacity; 30 English words at ~10 bits/word ≈ 300 bits — fine.
- For 50+ word sentences with rare entities, numbers, syntax — the vector becomes lossy.

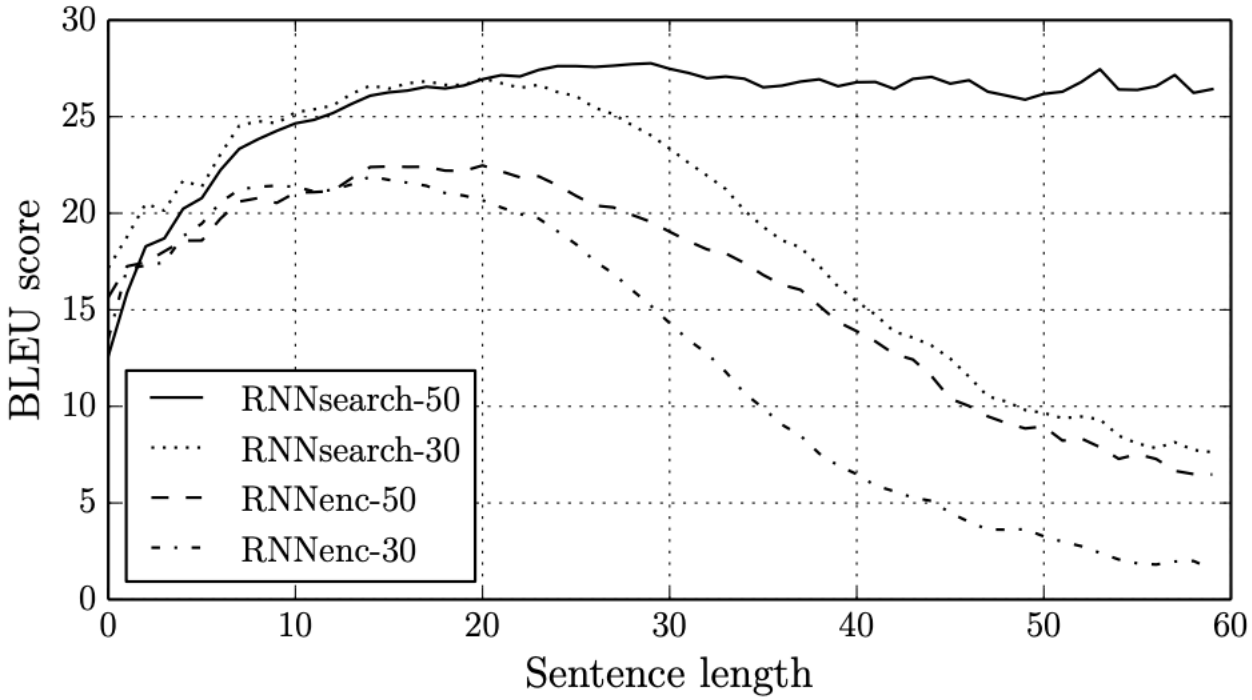
BLEU Collapses on Long Sentences

- Cho et al. 2014 and Bahdanau et al. 2014 both report the same curve.
- Vanilla seq2seq BLEU degrades sharply for sentences > 30 tokens.

This is NOT an optimization failure.

- Bigger model, more data, more training do not fix it.

It is a representational failure.



Bahdanau et al, 2014. RNNsearch.

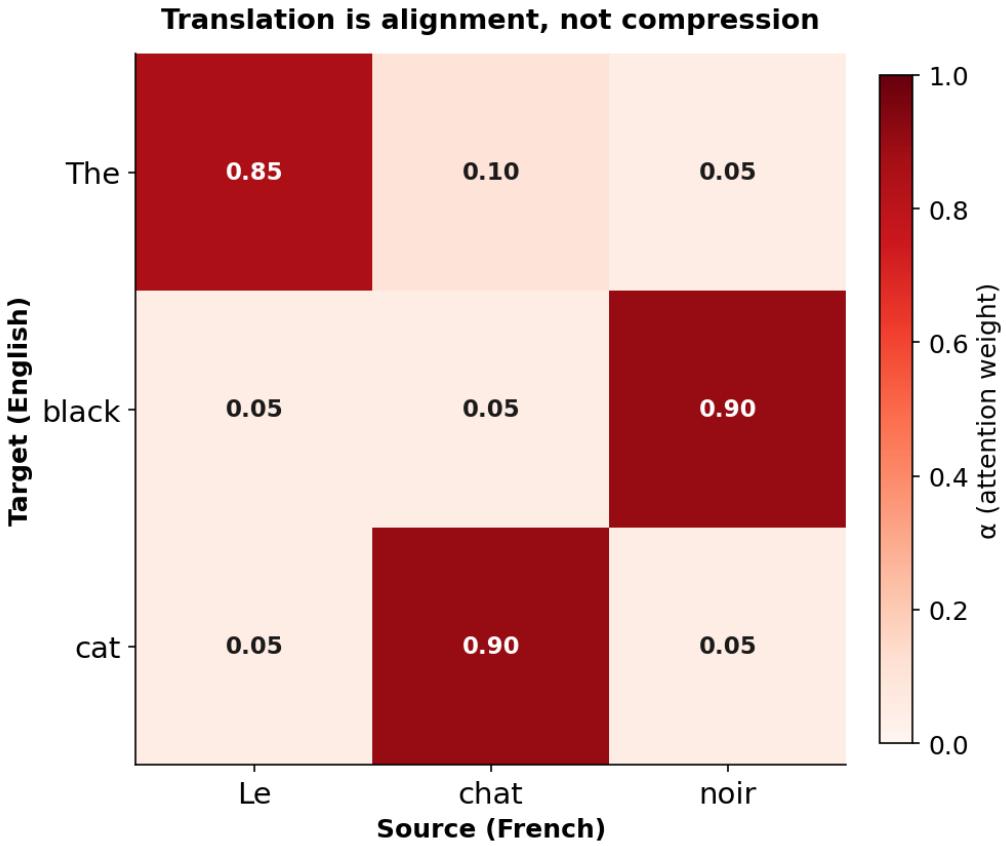
What Does the Decoder Actually Need?

- At decoder step t, generating y_t (e.g., the 5th English word):
- The decoder doesn't need the whole source uniformly compressed.
- It needs the relevant parts of the source for this specific output position.

Examples:

- Generating "cat" → look at "chat"
- Generating "black" → look at "noir"

This is soft alignment.



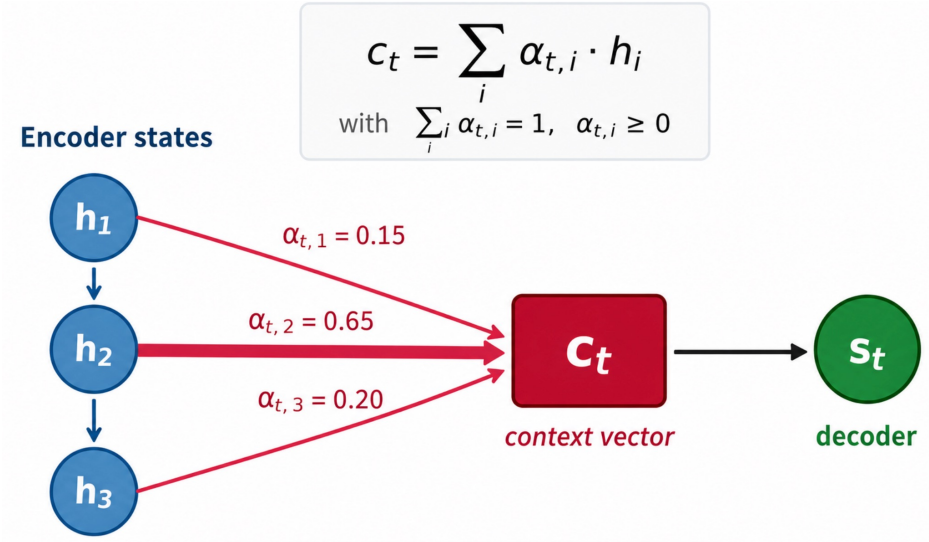
2. Deriving the Attention Mechanism

Derivation A: Information-Theoretic

- Goal: provide the decoder access to ALL encoder states h_1...h_T, not just h_T.
- Naive idea: concatenate them → input dim grows with T → breaks parameter sharing.
- Better idea: at each decoder step, compute a weighted average:

$$c_t = \sum_i \alpha_{ti}\, h_i, \quad \text{with} \sum_i \alpha_{ti} = 1,\ \alpha_{ti} \geq 0$$

- Capacity is now O(T), not O(1). The bottleneck is gone — if we can learn the α_{ti}.



Where Do the Weights Come From?

The α weights must depend on both:

- the current decoder state s_{t-1} (what we're trying to generate)
- each encoder state h_i (what's available at source position i)

So: $\alpha_{ti} = \text{softmax}_i\left(\text{score}(s_{t-1}, h_i)\right)$

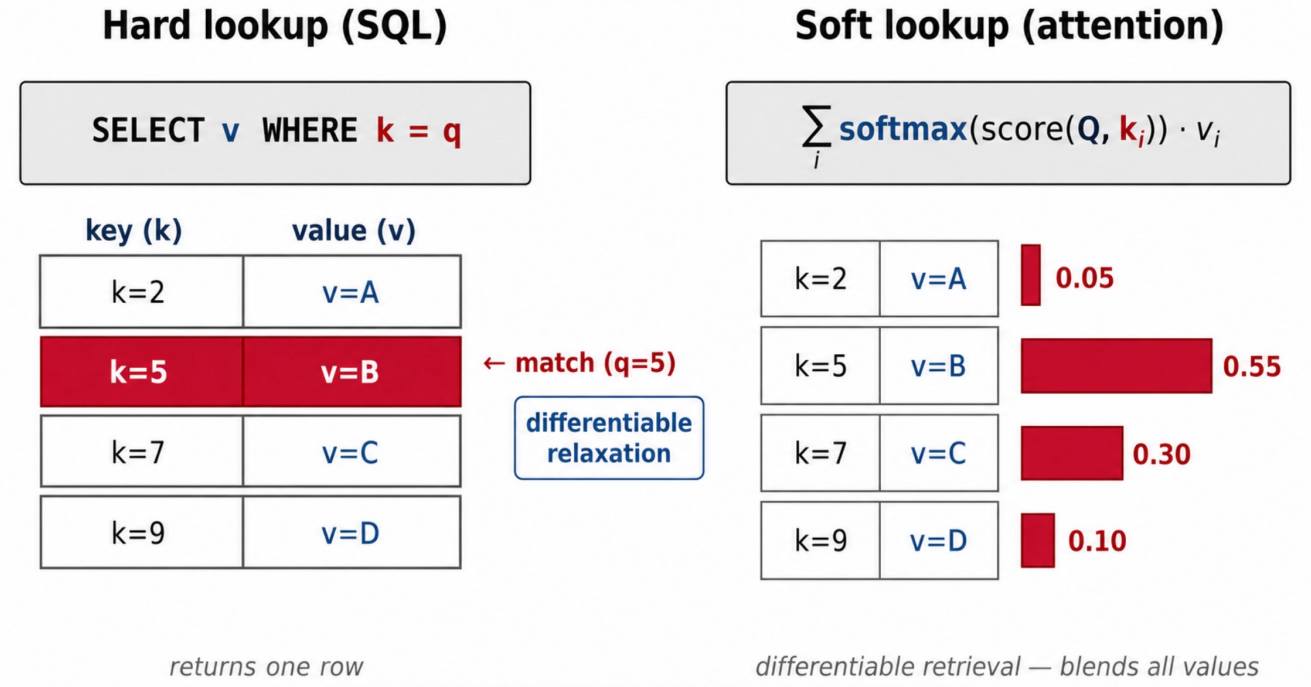
- Question: what is the simplest differentiable function that takes two vectors and returns a scalar?
- Answer: an inner product. Or a small MLP. We will see both.

Derivation B: The Database / QKV Framing

Re-cast the same idea as a soft database lookup:

- Query Q — "what am I looking for?" (decoder state s_{t-1})
- Keys $K = \{k_1, \ldots, k_T\}$: "what each source position offers" (encoder states)
- Values $V = \{v_1, \ldots, v_T\}$: "what to return if matched" (also encoder states)

Output = $\sum_i \text{softmax}\big(\text{score}(Q, k_i)\big)_i \, v_i$



The Attention Equation

the score function is the design choice — everything else is fixed

$$c_t = \sum_i \alpha_{t,i} \cdot v_i$$

$$\alpha_{t,i} = \frac{\exp(\text{score}(q_t, k_i))}{\sum_j \exp(\text{score}(q_t, k_j))}$$

q_t	← decoder query (what I want)
k_i	← encoder key (what's offered)
v_i	← encoder value (what to return)
score(·,·)	← design choice — Bahdanau and Luong differ here
softmax	← turns scores into a probability distribution

Quick Check

Given an attention layer with queries $q_t \in \mathbb{R}^{64}$ and 30 keys $k_i \in \mathbb{R}^{64}$:

- (a) What is the shape of α_t?
- (b) What does $\sum_i \alpha_{ti}$ equal?

Answer:

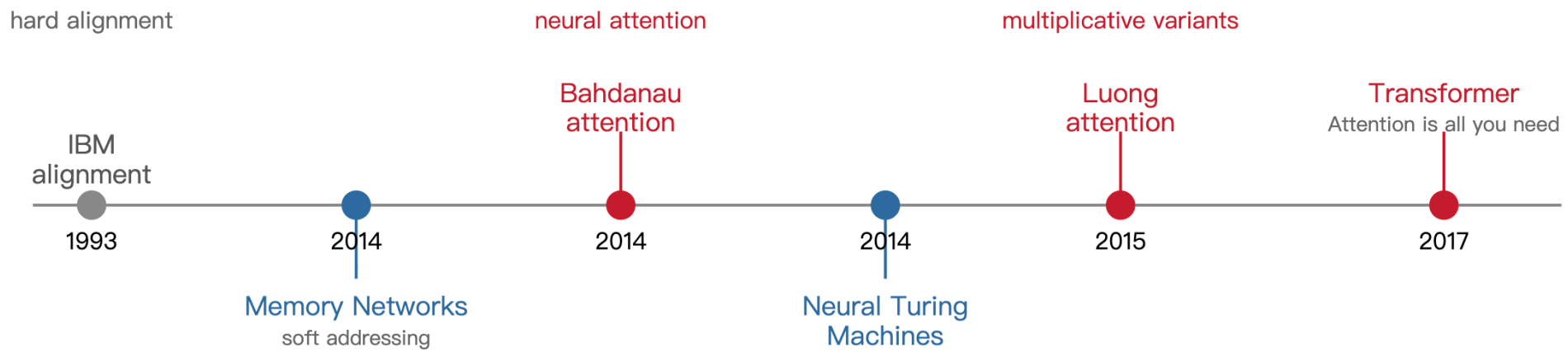
- (a) [30] (one weight per source position)
- (b) 1 (probability distribution over source positions)

Pre-history: Where Did This Idea Come From?

Attention didn't appear from nowhere in 2014. Three threads converged:

- Memory Networks (Weston, Chopra, Bordes — 2014): explicit memory + soft addressing for QA
- Neural Turing Machines (Graves, Wayne, Danihelka — 2014): differentiable read/write heads with content-based addressing — direct ancestor of softmax(QK^T)
- Statistical-MT alignment (IBM models 1–5, since 1993): hard alignments via EM; attention is the soft, learned successor

Bahdanau's contribution was the form and the task (NMT), not the idea of soft retrieval.

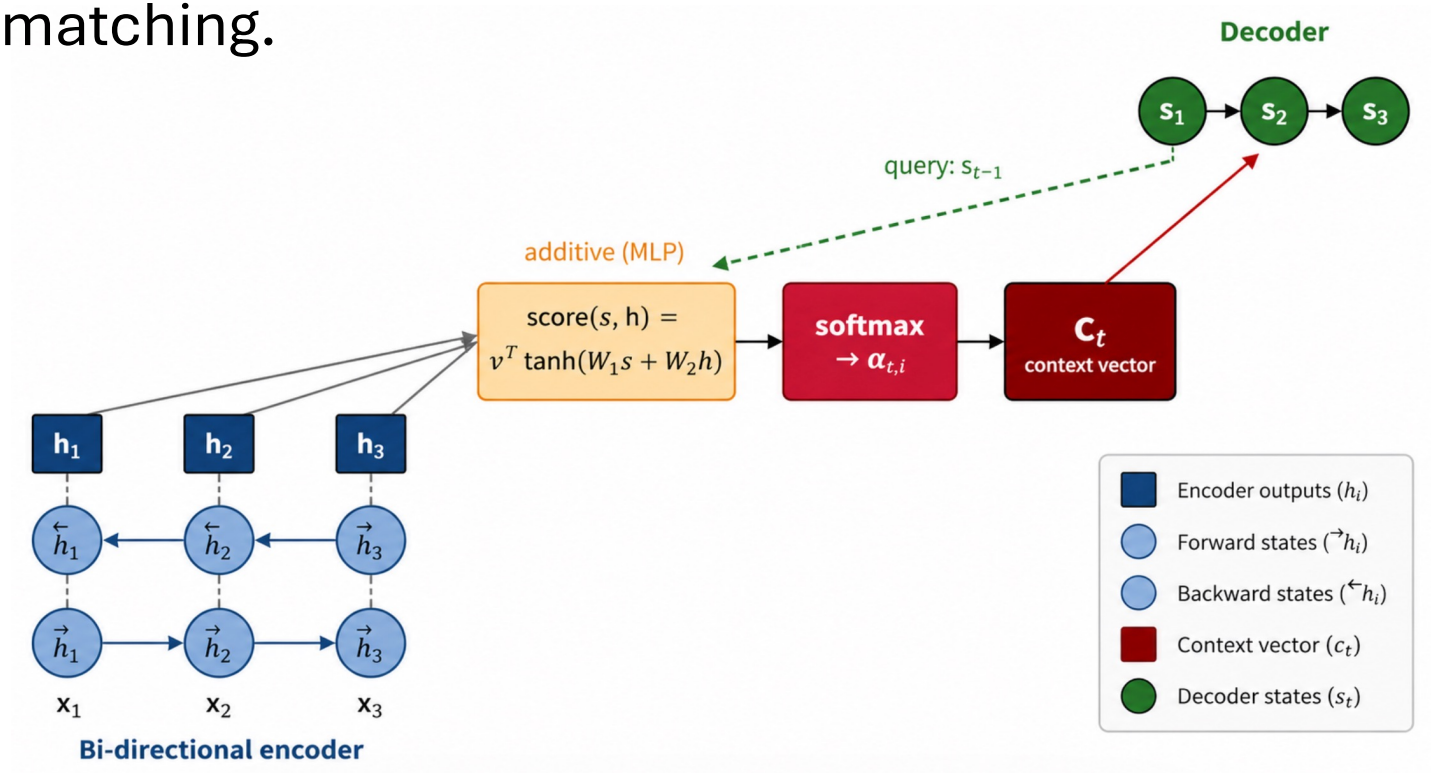


3. Bahdanau & Luong: The Two Designs That Worked

Bahdanau Attention (2014) — Additive

Bahdanau, Cho, Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015 (arXiv Sept 2014).

- Score function: $\text{score}(s, h) = v^T \tanh(W_1 s + W_2 h)$ — a tiny MLP
- Why additive? Handles different dim sizes for s and h naturally; expressive without dim matching.

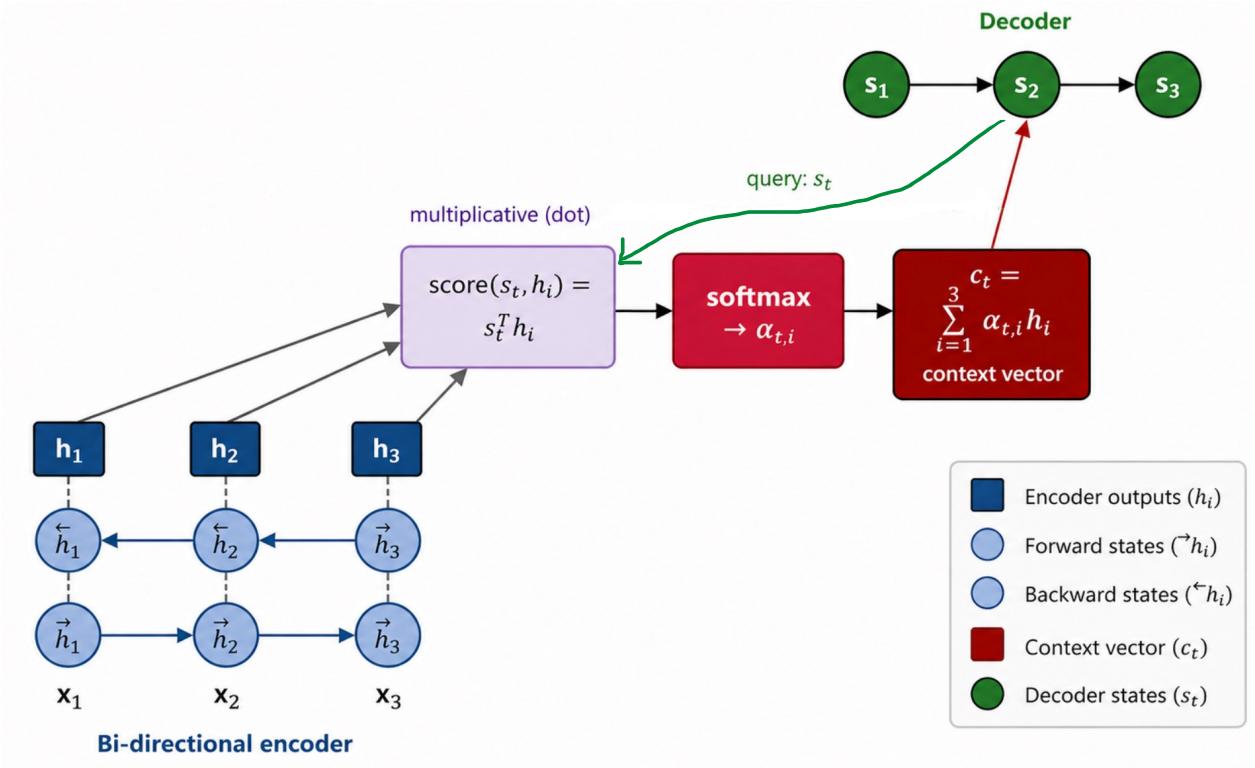


Luong Attention (2015) — Multiplicative

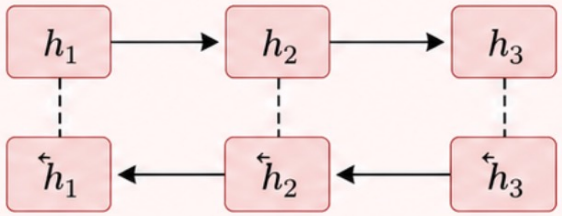
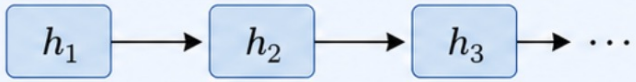
Luong et al. Effective Approaches to Attention-based Neural Machine Translation. EMNLP 2015.

Three score variants:

- dot: score(s, h) = $s^T h$ — cheapest, requires same dim
- general: score(s, h) = $s^T W h$ — adds one trainable matrix; dims can differ
- concat: score(s, h) = $v^T \tanh(W [s; h])$ — essentially Bahdanau



Bahdanau vs. Luong

Aspect	Bahdanau Attention (Additive)	Luong Attention (Multiplicative)
Score function	**Additive (MLP)** $\text{score}(s_t, h_i) = v^T \tanh(W_1 h_i + W_2 s_t)$	**Dot / General / Concat** • **Dot:** $s_t^T h_i$ • **General:** $s_t^T W h_i$ • **Concat:** $v^T \tanh(W[s_t; h_i])$
Encoder	**Bidirectional RNN** 	**Uni- or Bi-directional RNN**  (uni-directional) or (bi-directional)
Query	s_{t-1} (previous decoder state)	s_t (current decoder state)
Compute (extra cost)	$O(T \cdot d^2)$ extra parameters	• **Luong-general:** $O(d^2)$ • **Luong-dot:** $O(0)$ (no extra parameters)
Numerical stability	**tanh saturation possible** (can cause vanishing gradients)	**Luong-dot grows with $\sqrt{d}$** (may cause large scores for large d)

Why Multiplicative Wins on Modern Hardware

- Dot product = single matmul → maximally parallel on GPUs.
- Additive = elementwise tanh → memory-bound, not compute-bound.
- For the Transformer (next lecture) with hundreds of attention heads in parallel, only multiplicative is viable.

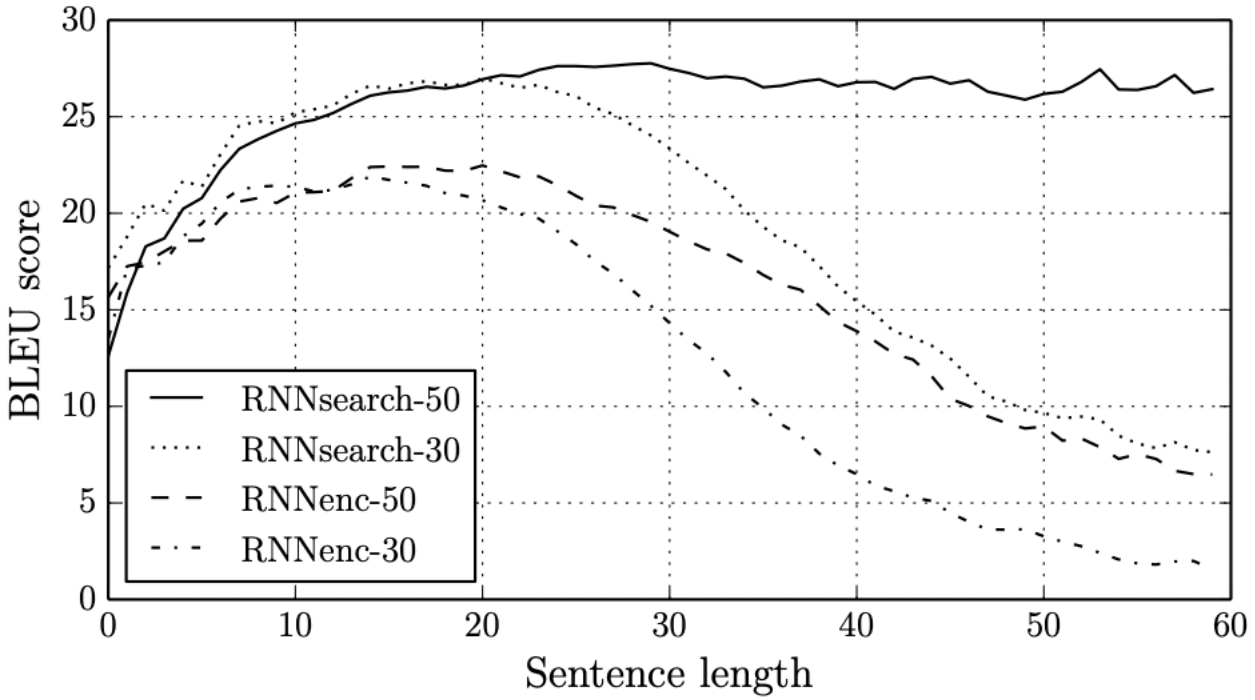
4. Benefits of Attention

Benefit #1: Long-Sentence BLEU Recovers

- The BLEU-vs-length curve from earlier — now with the gap closed.

- The bottleneck was real.
- The fix is real.

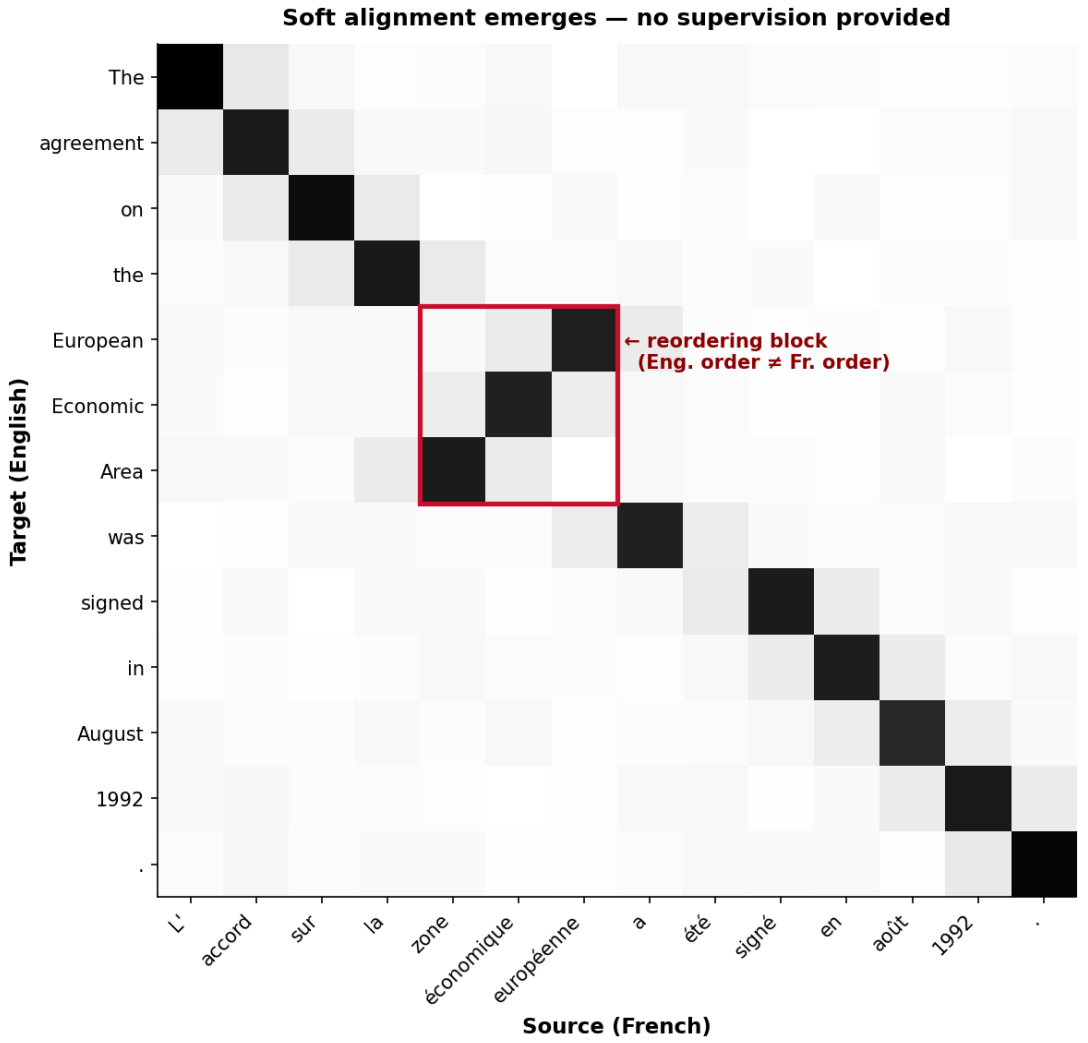
Improvements: 5 to 10 BLEU points on long sentences.



Bahdanau et al, 2014. RNNsearch.

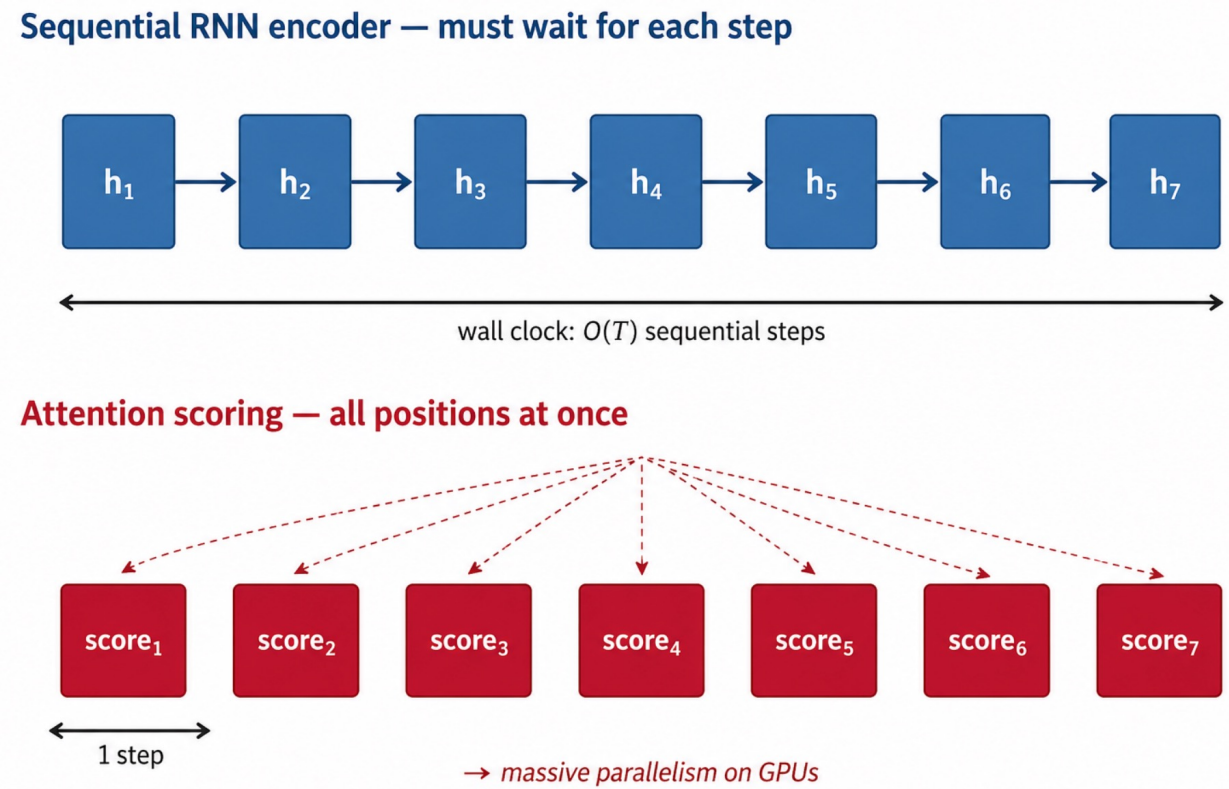
Benefit #2: Interpretability

- The α matrix is a soft alignment between source and target tokens.
- Bahdanau et al. showed alignment matrices that visibly capture French ↔️ English word reorderings.
- Implication: attention is a debugging gift. You can see what the model is doing.



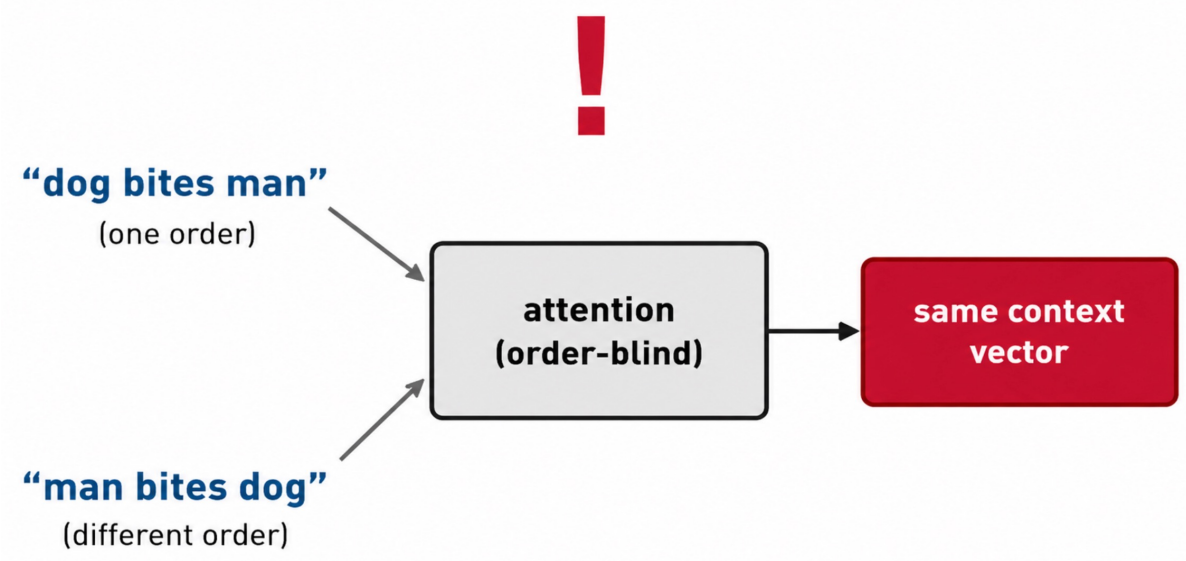
Benefit #3: Scoring Is Parallelizable

- All scores score(q_t, k_i) for i = 1…T can be computed in parallel — one matmul.
- The decoder is still sequential (q_t depends on previous outputs).
- But the scoring over the source has no sequential dependency.



The Catch: Attention Is Permutation-Equivariant

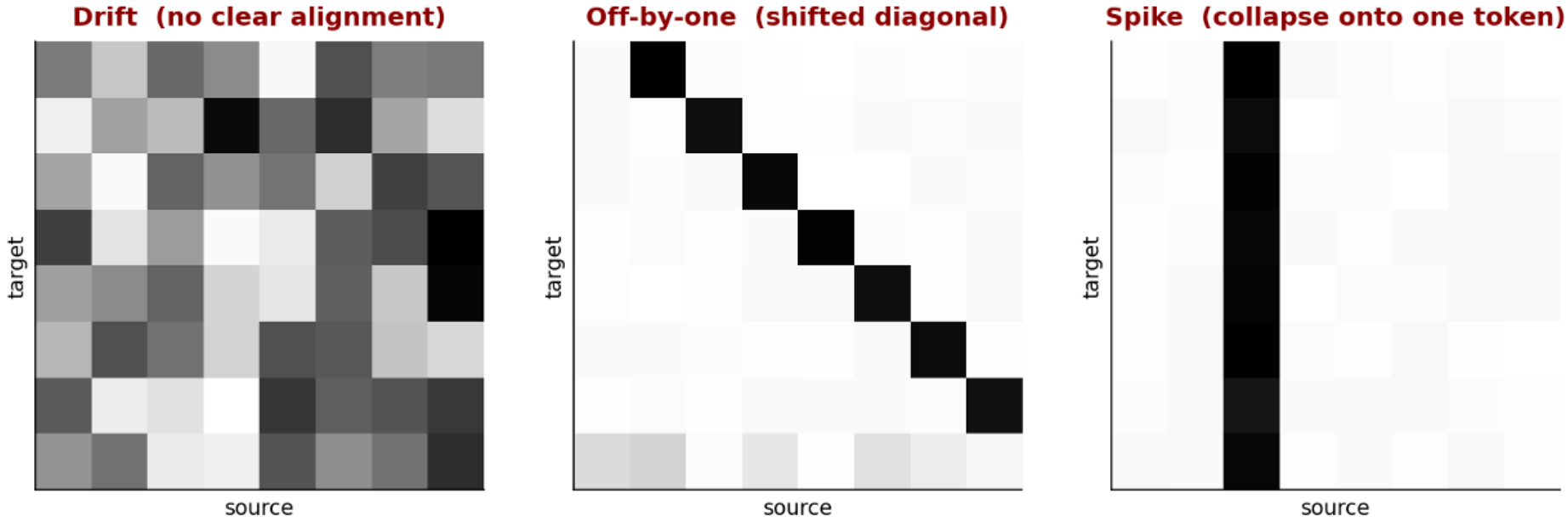
- attention(Q, π(K), π(V)) = π(attention(Q, K, V)) for any permutation π.
- We sum over the position axis → order is lost.
- In current seq2seq, the encoder RNN injects order for free (recurrence ⇒ position-aware h_i).



Same context, opposite meaning — attention forgets order

Failure Modes

- Wrong tokens: attention can attend to the wrong place — visible in the alignment matrix.
- Attention drift: long generation → output ignores source after many steps. Motivates coverage (Tu et al. 2016).
- Bias amplification: softmax pushes weight to dominant token, can ignore long tail of relevant context.



5. If Attention Is This Good — Why Keep Recurrence?

The Inevitable Question

Recurrence costs us:

- sequential computation (slow training)
- vanishing gradients across long ranges
- information bottleneck (h_T)

Attention gives us:

- direct access to every position
- parallelizable scoring
- interpretable alignments

"Attention is all you need." — Vaswani et al., 2017

Summary

- What attention is: a learned, differentiable, soft retrieval — softmax(score(Q, K)) · V
- Why we needed it: the encoder bottleneck was a representational, not optimization, failure
- Two variants: Bahdanau (additive, expressive) and Luong (multiplicative, fast) — multiplicative + scaling won
- What it bought: long-sentence BLEU, free alignments, parallelizable scoring

- What‘s next (L22): drop recurrence entirely → self-attention → **Transformer**