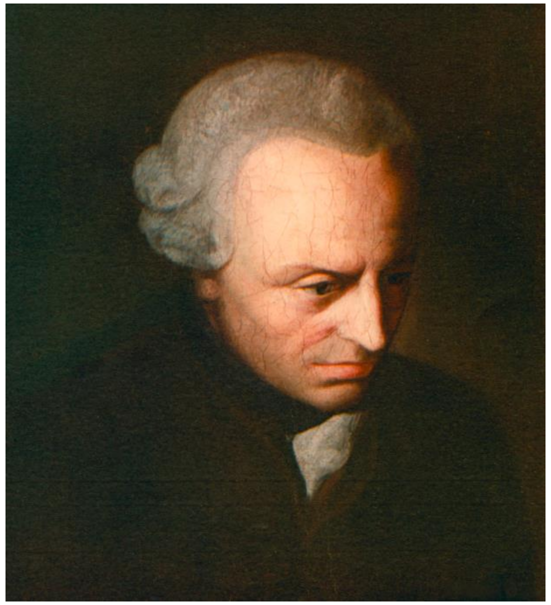
Attention and Transformers

Oualid Missaoui

Agenda

Philosophy

You can agree with Kant or disagree with him, but you can't get past him.



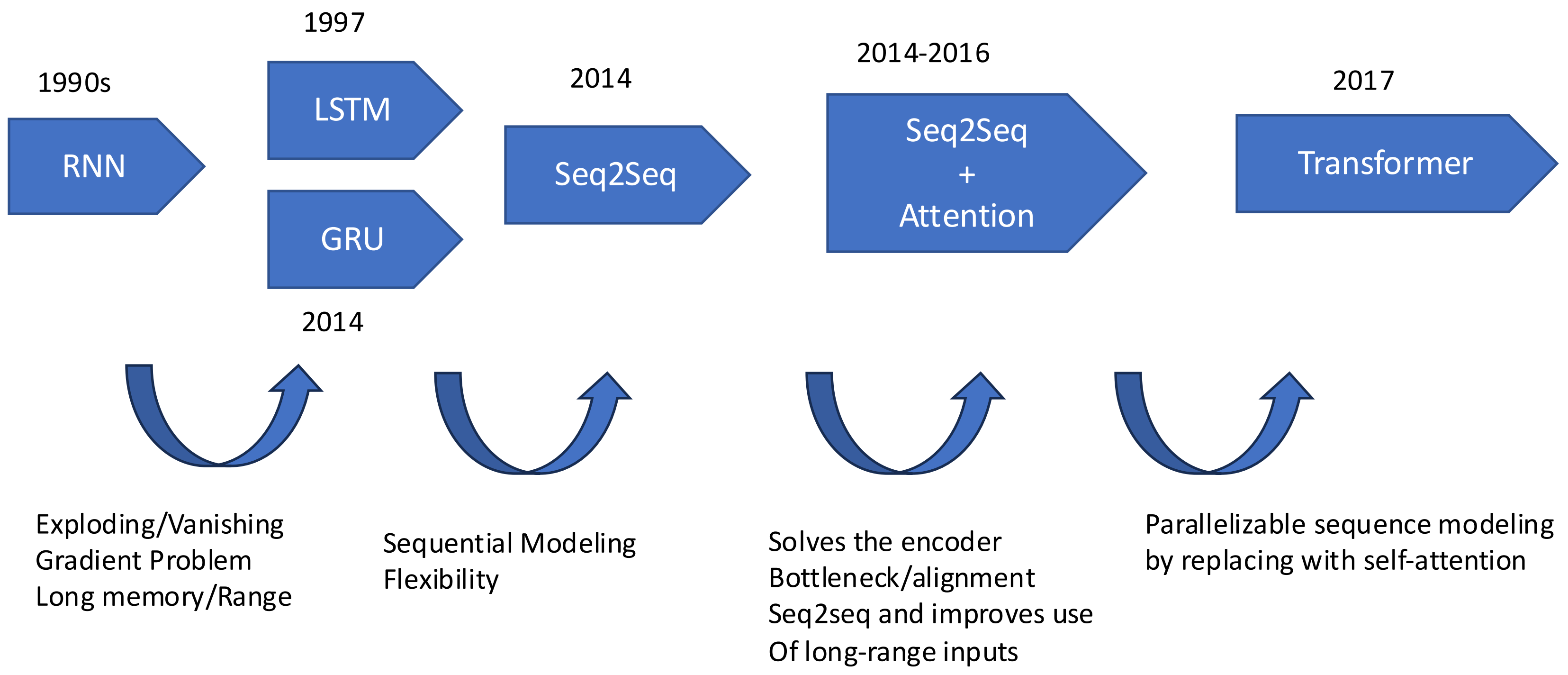
Immanuel Kant

AI

Use Transformers or don't; no well-rounded modern AI scientist gets to sidestep them.

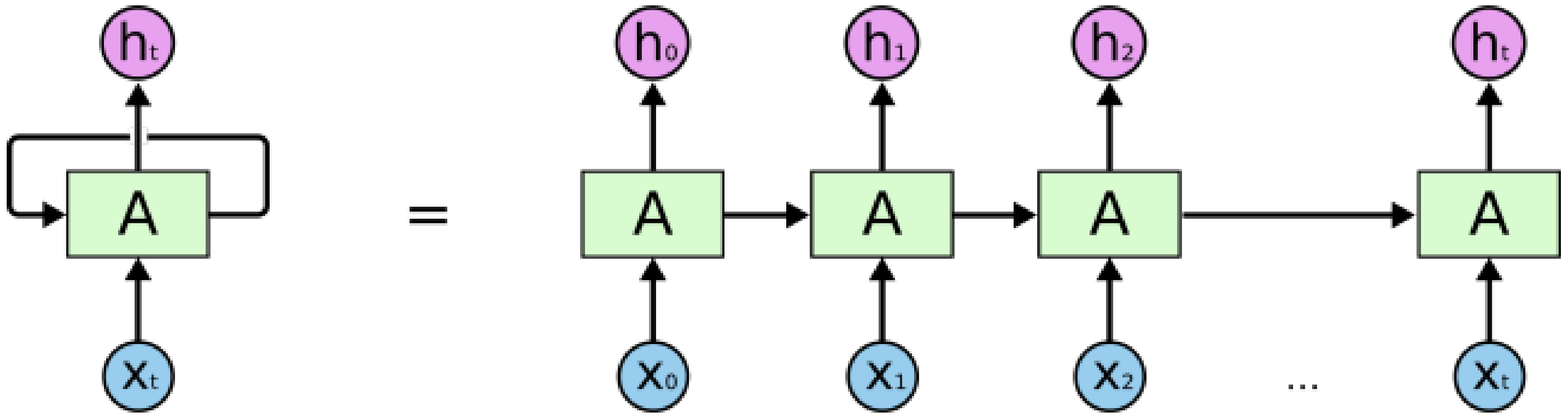
Ashish Vaswani

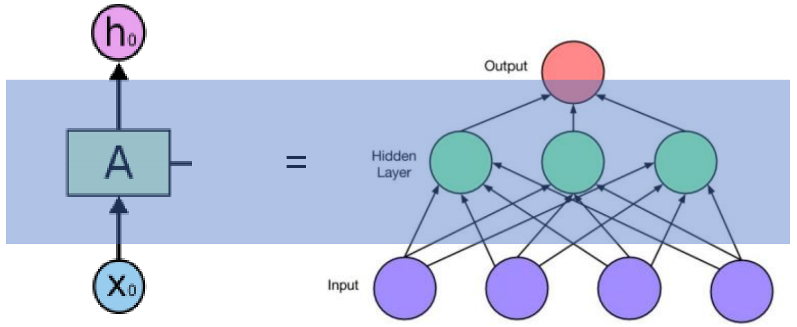
Historical development



Recurrent Neural Networks (RNN)

- History:
  - John Hopfield (1982), Neural Networks and Physical Systems with Emergent Collective Computational Abilities
  - David Rumelhart, G. Hinton and R. J. Wiliams (1986), Learning Internal Representations by Error Propagation





$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d(\sigma(x))}{dx} = \sigma(x)(1-\sigma(x))$$

LSTM vs GRU

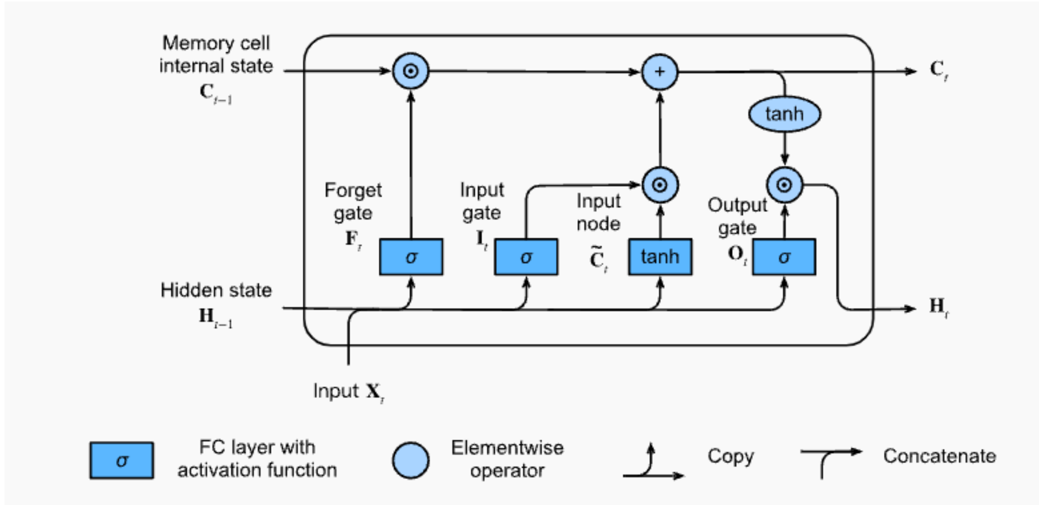


Fig. 10.1.4 Computing the hidden state in an LSTM model.

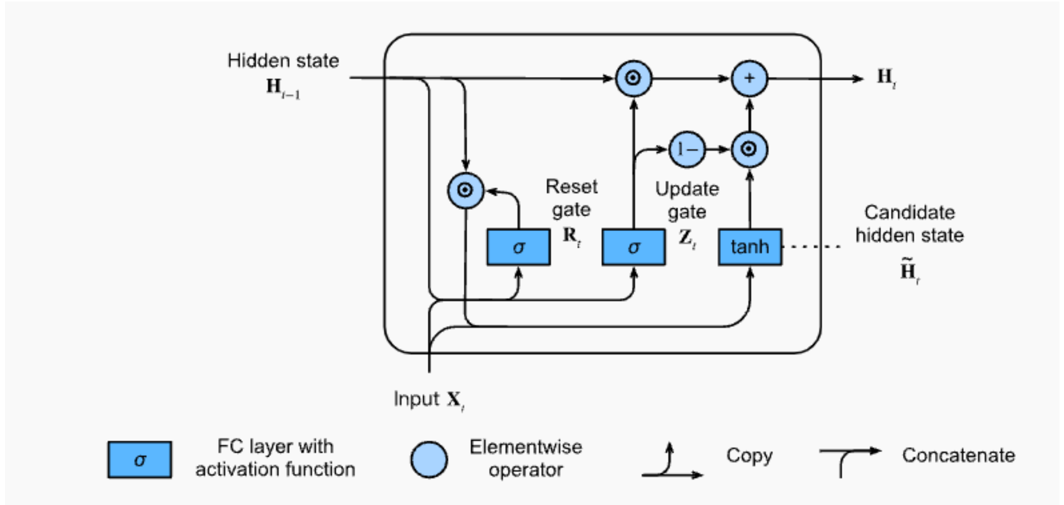
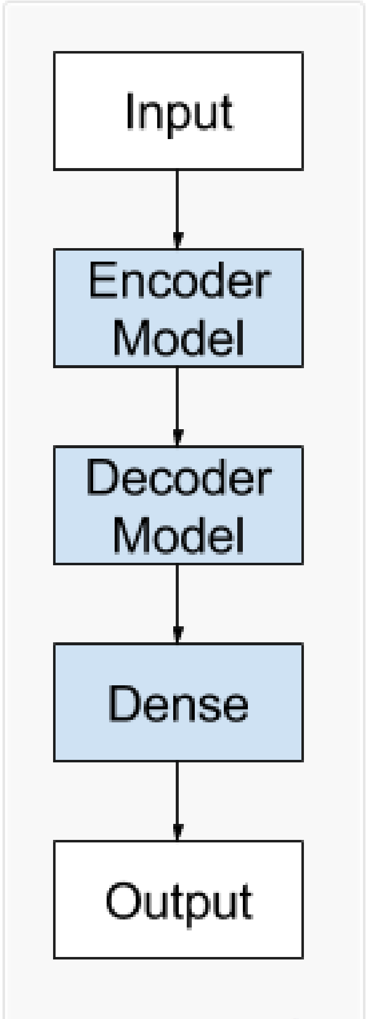
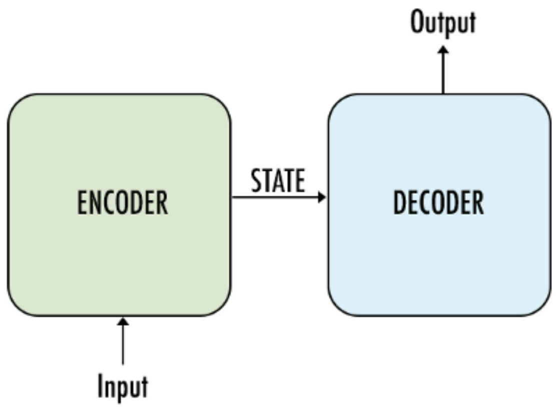


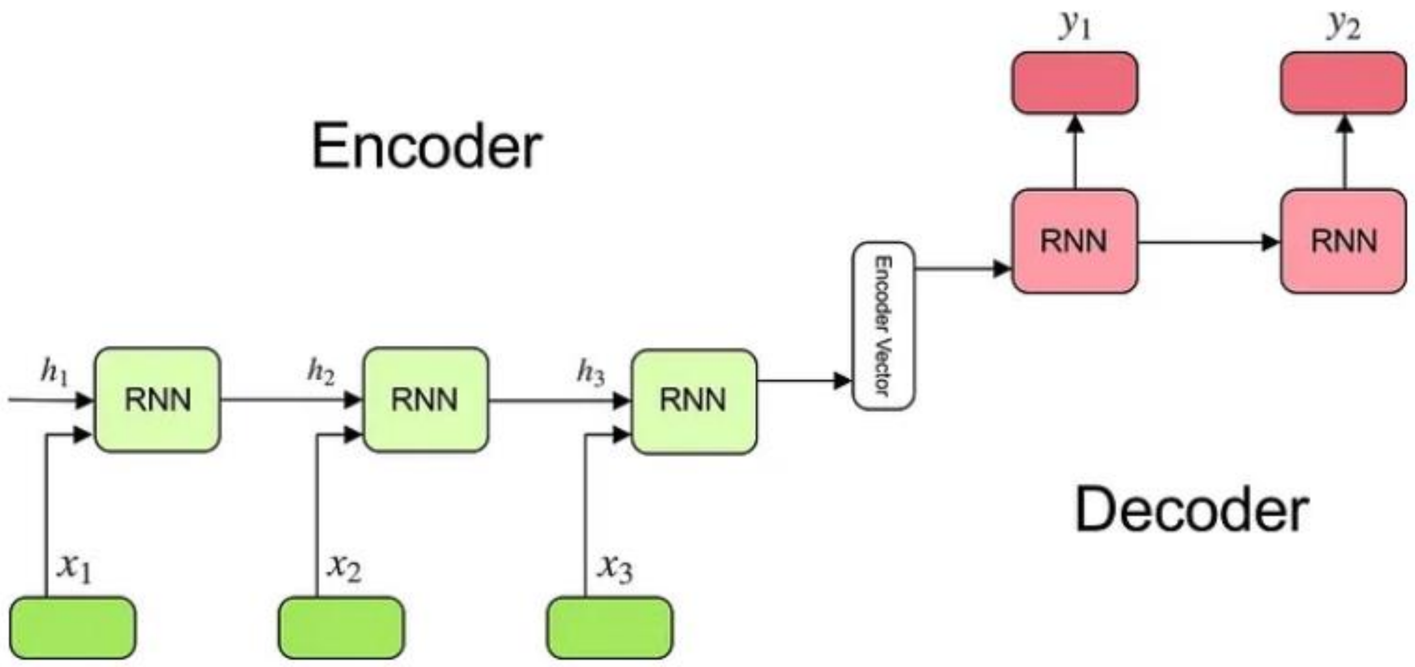
Fig. 10.2.3 Computing the hidden state in a GRU model.

Seq2Seq Architecture



Encoder-Decoder LSTM Model Architecture





Encoding an input sentence into a fixed-size vector for the decoder

- A limitation of the architecture is that it encodes the input sequence to a fixed length internal representation.
- This imposes limits on the length of input sequences that can be reasonably learned and results in worse performance for very long input sequences.

The need for attention

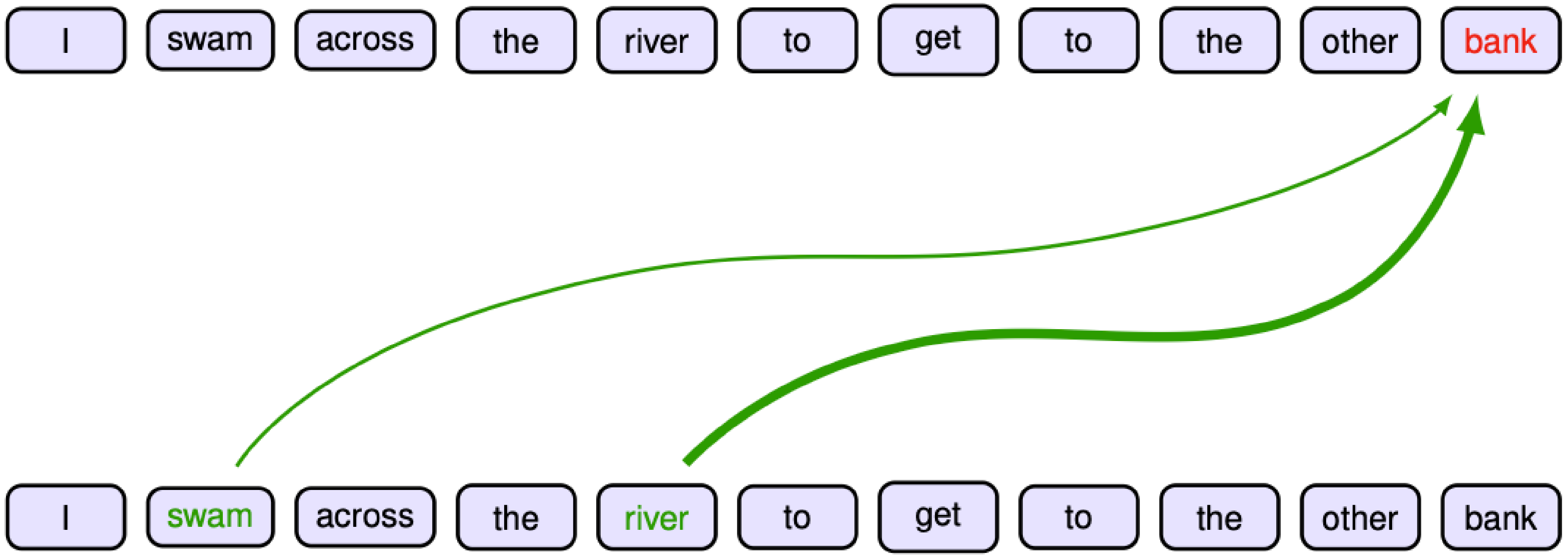


Figure 12.1 Schematic illustration of attention in which the interpretation of the word ‘bank’ is influenced by the words ‘river’ and ‘swam’, with the thickness of each line being indicative of the strength of its influence.

Seq2Seq with Attention (Bahdanau)

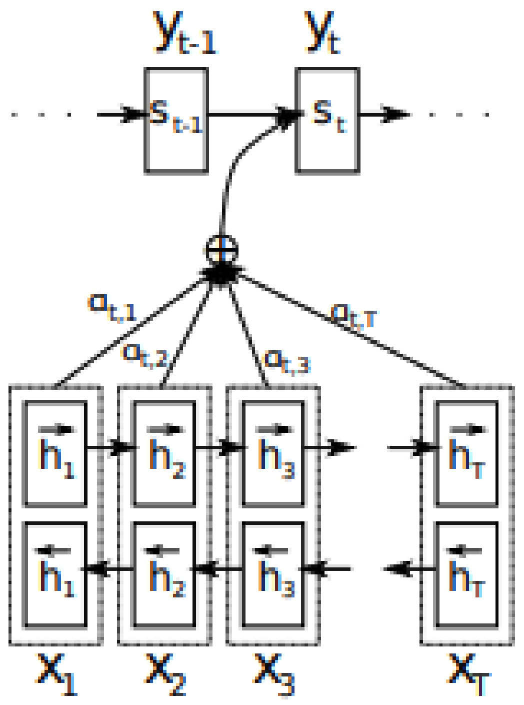
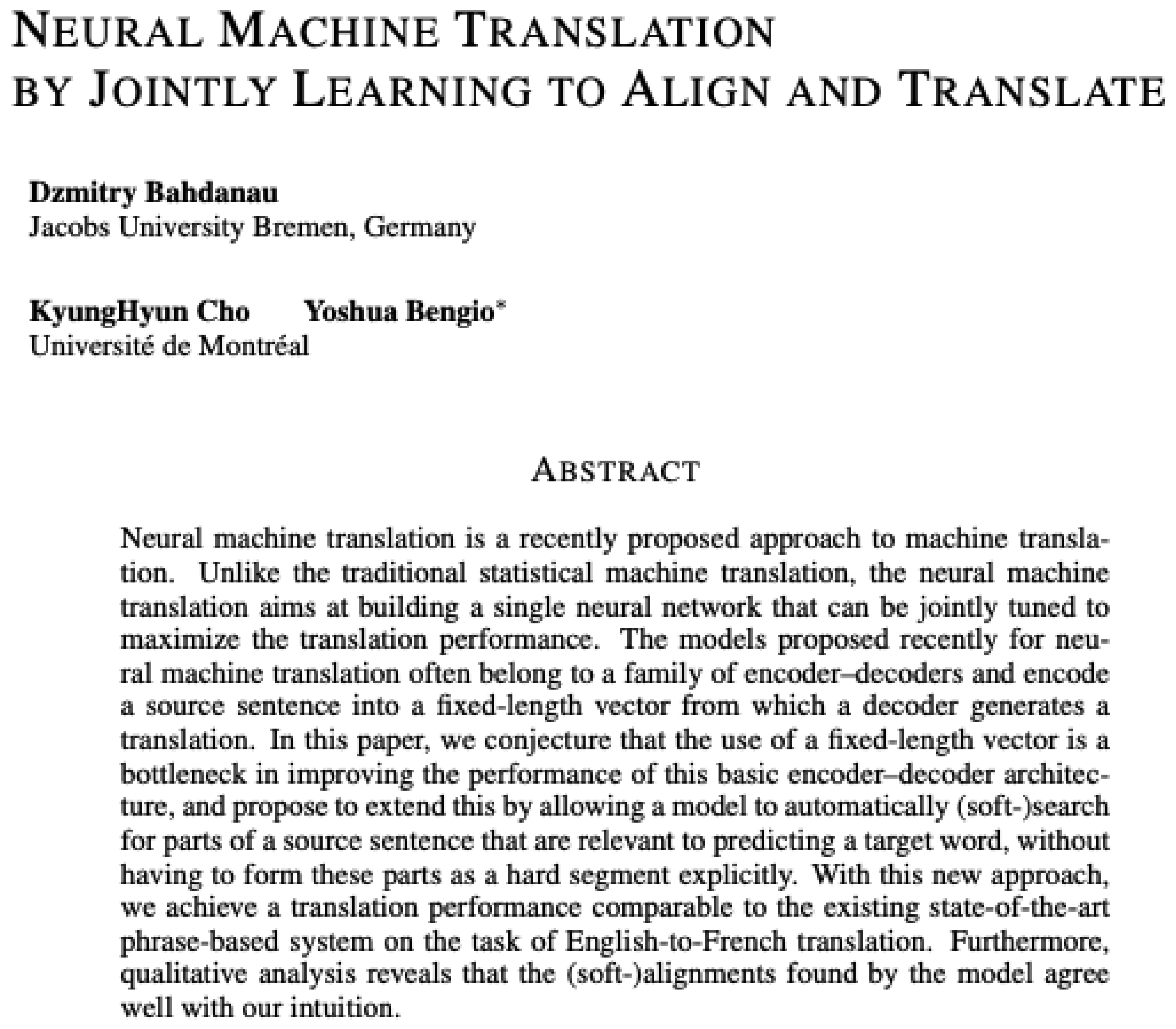


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio**[*]
Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder–decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder–decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly. With this new approach, we achieve a translation performance comparable to the existing state-of-the-art phrase-based system on the task of English-to-French translation. Furthermore, qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

Dzmitry Bahdanau et al, Neural Machine Translation by jointly learning to align and translate

Seq2Seq: fixed vs dynamic context (c)

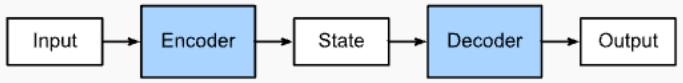


Fig. 10.6.1 The encoder-decoder architecture.

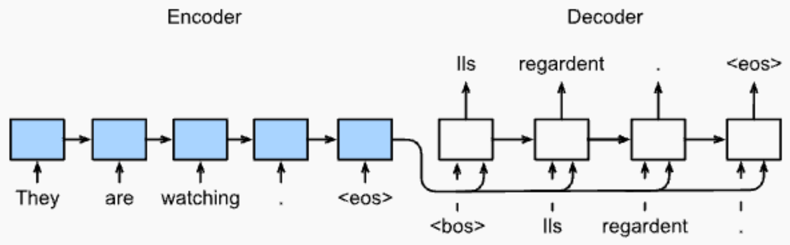


Fig. 10.7.1 Sequence to sequence learning with an RNN encoder and an RNN decoder.

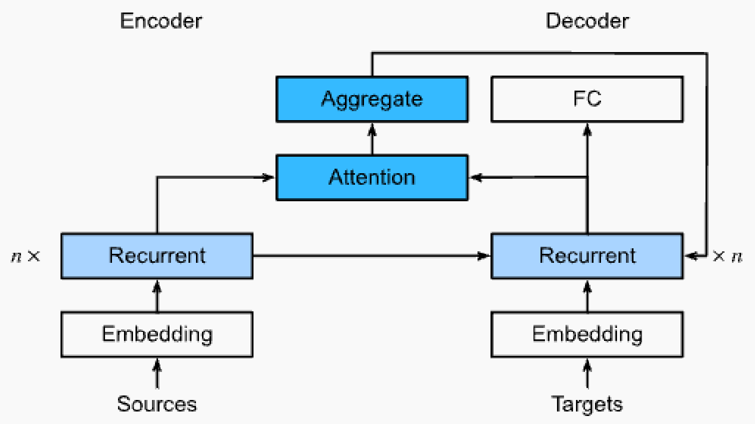


Fig. 11.4.2 Layers in an RNN encoder-decoder model with the Bahdanau attention mechanism.

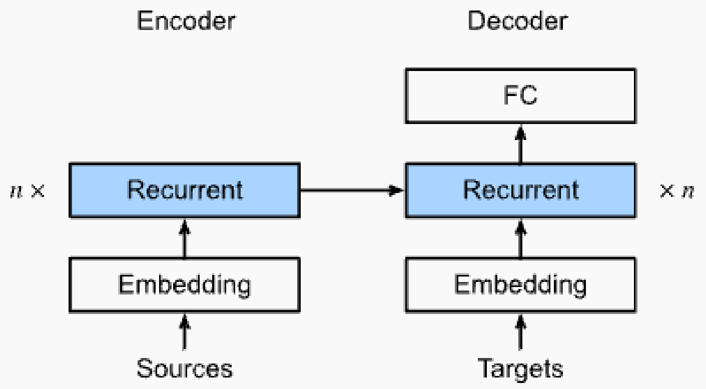


Fig. 10.7.2 Layers in an RNN encoder-decoder model.

$$\mathbf{s}_{t'} = g(y_{t'-1}, \mathbf{c}, \mathbf{s}_{t'-1}).$$

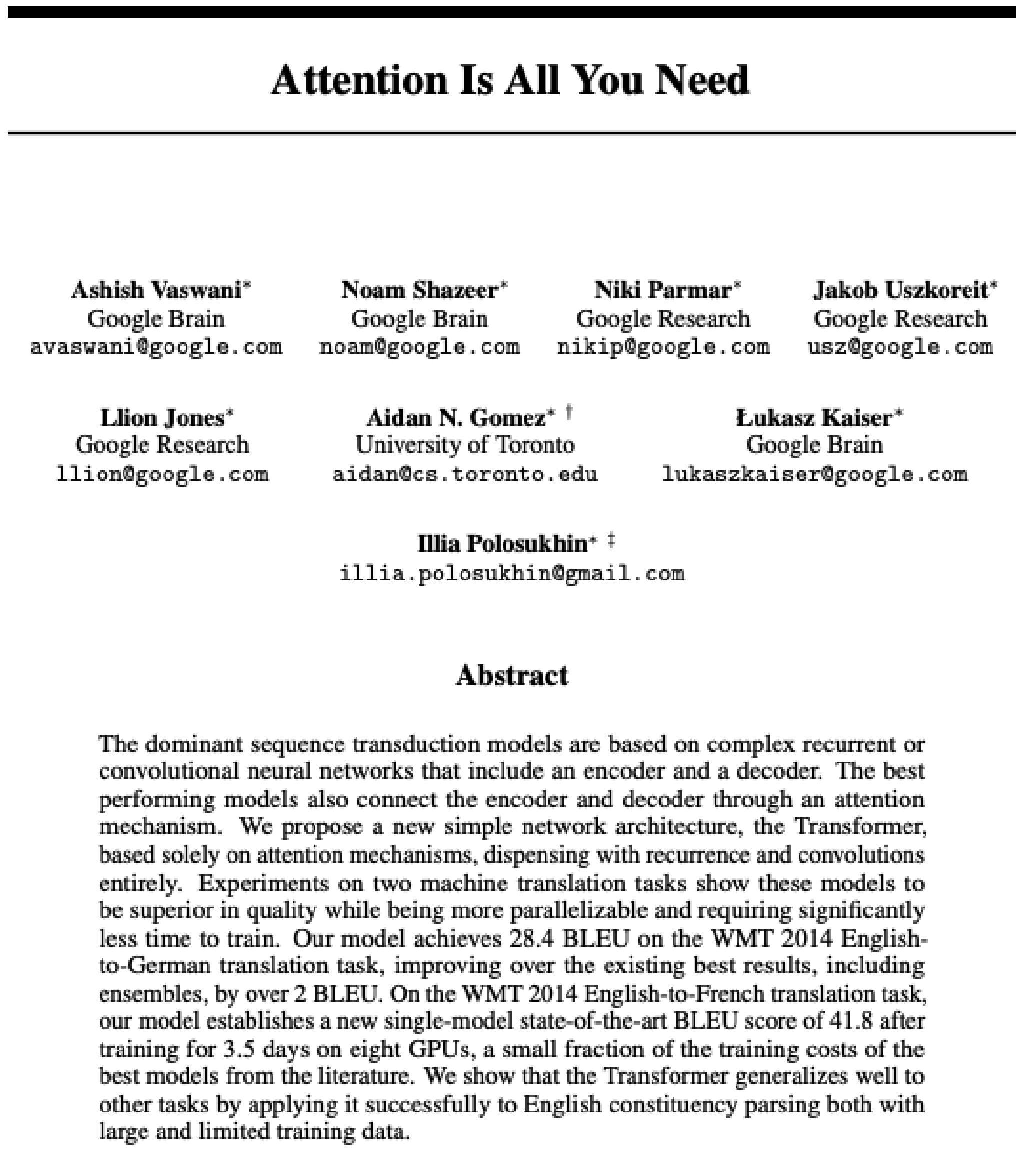
The key idea is that instead of keeping the state , i.e. the context variable c summarizing the source sentence as fixed, we dynamically update it, as a function of both:

1- the original text (encoder hidden states)

2- the text that was already generated (decoder hidden states)

$$\mathbf{c}_{t'} = \sum_{t=1}^{T} \alpha(\mathbf{s}_{t'-1}, \mathbf{h}_t)\mathbf{h}_t.$$

Transformers



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

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

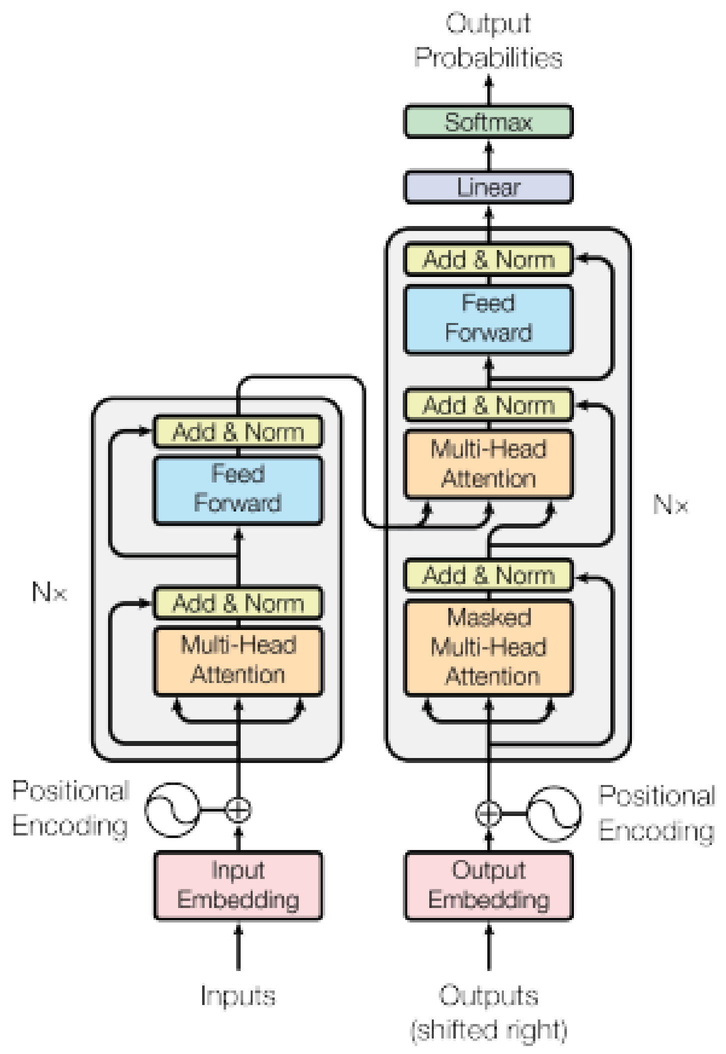


Figure 1: The Transformer - model architecture.

Ashish Vaswani et al (2017), [Attention is all you Need]

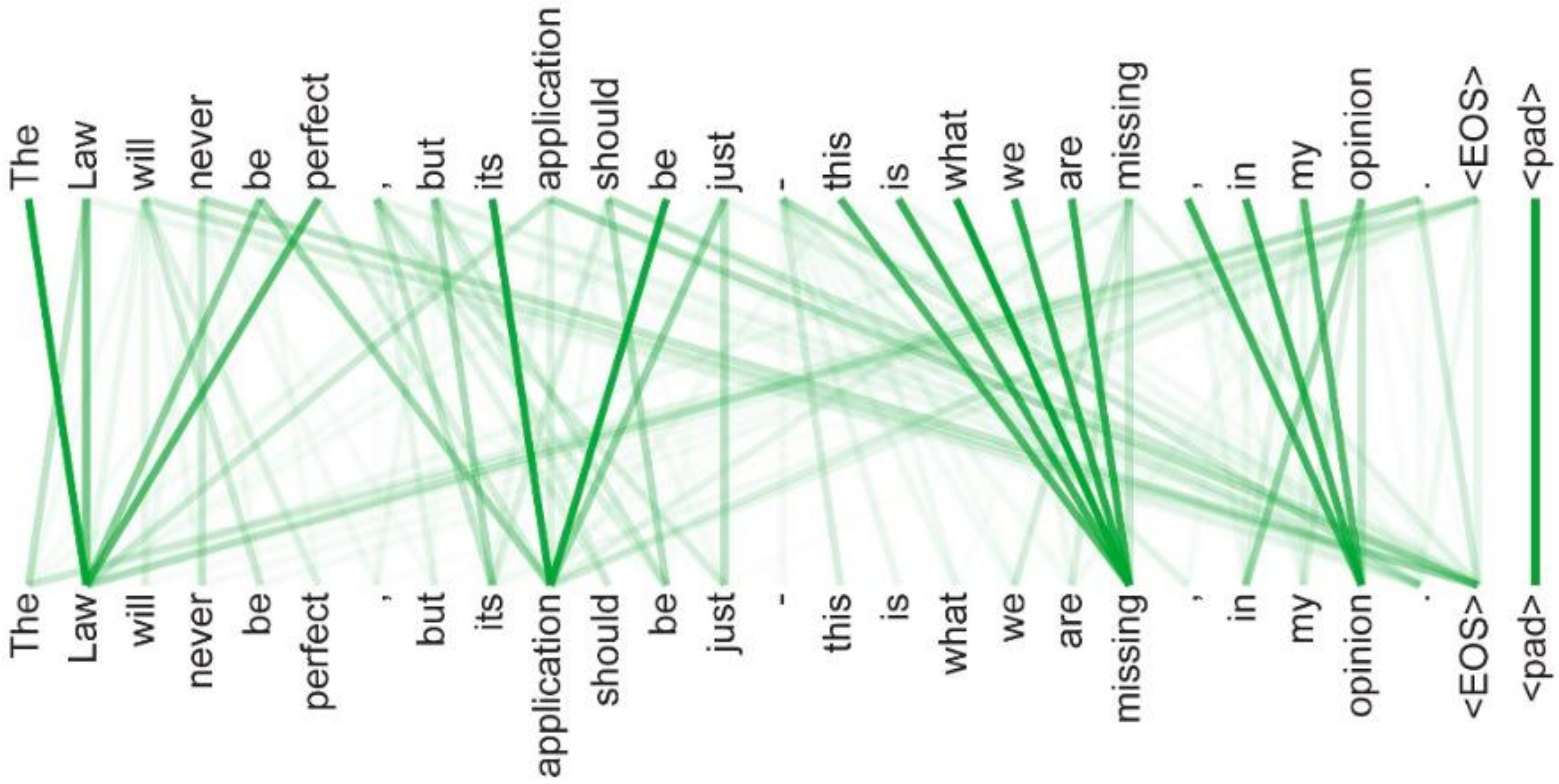
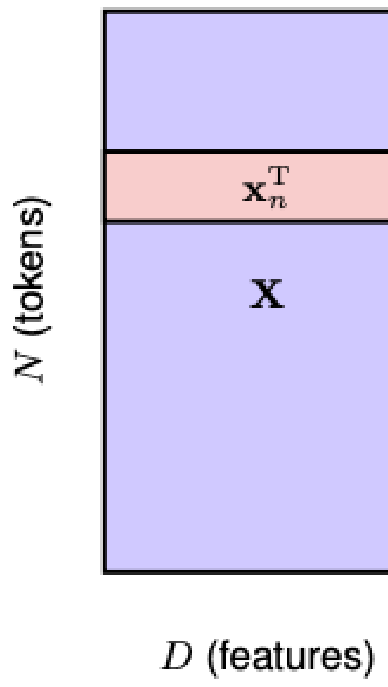
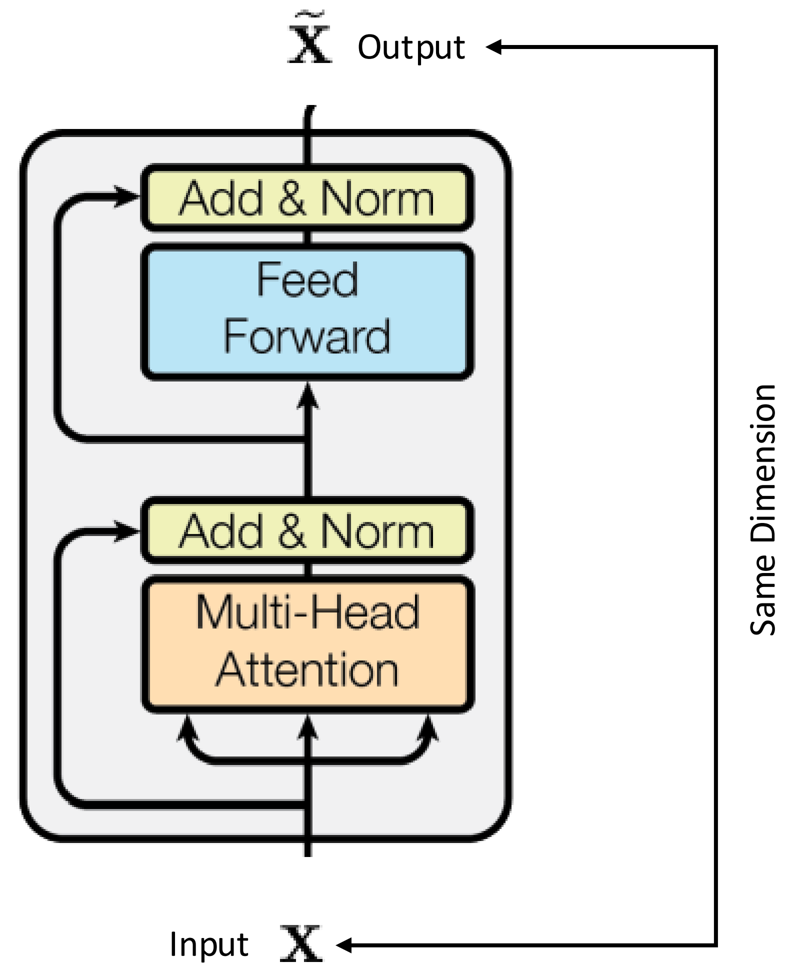


Figure 12.2 An example of learned attention weights. [From Vaswani *et al.* (2017) with permission.]

Transformer Layer



The structure of the data matrix $\mathbf{X}$, of dimension $N \times D$, in which row n represents the transposed data vector $\mathbf{x}_n^{\mathrm{T}}$.



$$\tilde{\mathbf{X}} = \text{TransformerLayer}\left[\mathbf{X}\right].$$

Attention

$$\begin{tikzcd}\mathbf{x}_1, \ldots, \mathbf{x}_N \arrow[r] & \boxed{\mathbf{y}_n = \sum_{m=1}^{N} a_{nm} \mathbf{x}_m} \arrow[r] & \mathbf{y}_1, \ldots, \mathbf{y}_N\end{tikzcd}$$

attention weights

$$a_{nm} \geqslant 0$$

$$\sum_{m=1}^{N} a_{nm} = 1$$

Self-Attention

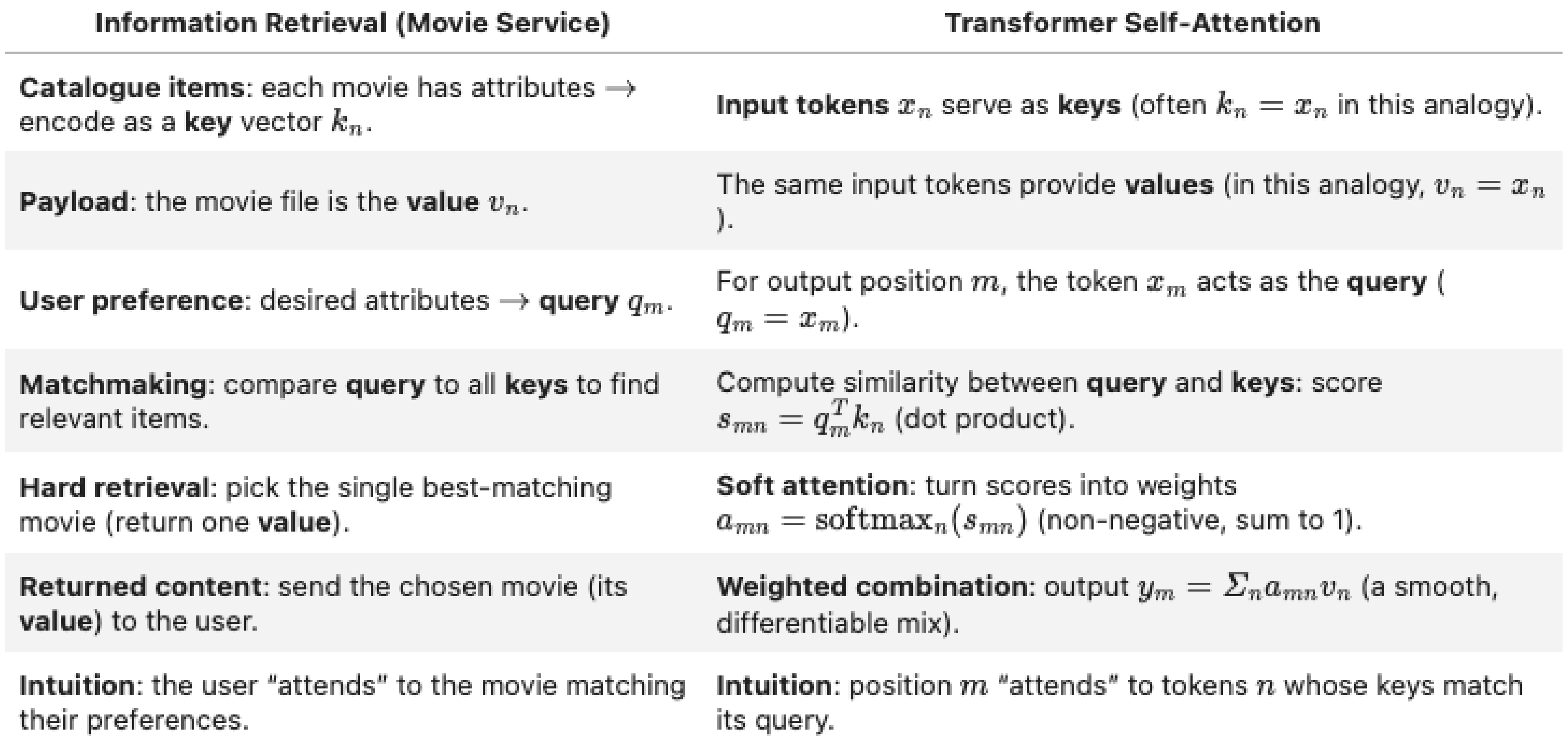
$$\mathbf{x}_1, \ldots, \mathbf{x}_N \longrightarrow \boxed{\mathbf{y}_n = \sum_{m=1}^{N} a_{nm} \mathbf{x}_m} \longrightarrow \mathbf{y}_1, \ldots, \mathbf{y}_N$$

attention weights

$$a_{nm} = \frac{\exp(\mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m)}{\sum_{m'=1}^{N} \exp(\mathbf{x}_n^{\mathrm{T}} \mathbf{x}_{m'})}$$

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{X}^{\mathrm{T}}\right] \mathbf{X}$$

Self Attention as Information Retrieval



Information Retrieval (Movie Service)	Transformer Self-Attention
Catalogue items: each movie has attributes $\rightarrow$ encode as a **key** vector k_n.	**Input tokens** x_n serve as **keys** (often $k_n = x_n$ in this analogy).
Payload: the movie file is the **value** v_n.	The same input tokens provide **values** (in this analogy, $v_n = x_n$).
User preference: desired attributes $\rightarrow$ **query** q_m.	For output position m, the token x_m acts as the **query** ($q_m = x_m$).
Matchmaking: compare **query** to all **keys** to find relevant items.	Compute similarity between **query** and **keys**: score $s_{mn} = q_m^T k_n$ (dot product).
Hard retrieval: pick the single best-matching movie (return one **value**).	**Soft attention**: turn scores into weights $a_{mn} = \mathrm{softmax}_n(s_{mn})$ (non-negative, sum to 1).
Returned content: send the chosen movie (its **value**) to the user.	**Weighted combination**: output $y_m = \Sigma_n a_{mn} v_n$ (a smooth, differentiable mix).
Intuition: the user "attends" to the movie matching their preferences.	**Intuition**: position m "attends" to tokens n whose keys match its query.

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right]\mathbf{V}$$

$$\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$$

Learning the Attention (1)

$$\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{X}$$

$$\downarrow$$

$$\mathbf{XU}$$

Learnable weight parameters

$\mathbf{U}$ is a $D \times D$

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{X}\mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\right]\mathbf{X}\mathbf{U}$$

Unfortunately symmetric, we want asymmetry support

Learning the Attention (2)

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(\mathrm{q})}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(\mathrm{k})}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(\mathrm{v})}$$

Learnable parameters

$$\mathbf{Y} = \mathrm{Softmax}\left[\mathbf{Q}\mathbf{K}^{\mathrm{T}}\right]\mathbf{V}$$

$\mathbf{W}^{(\mathrm{k})}$ is a $D \times D_{\mathrm{k}}$

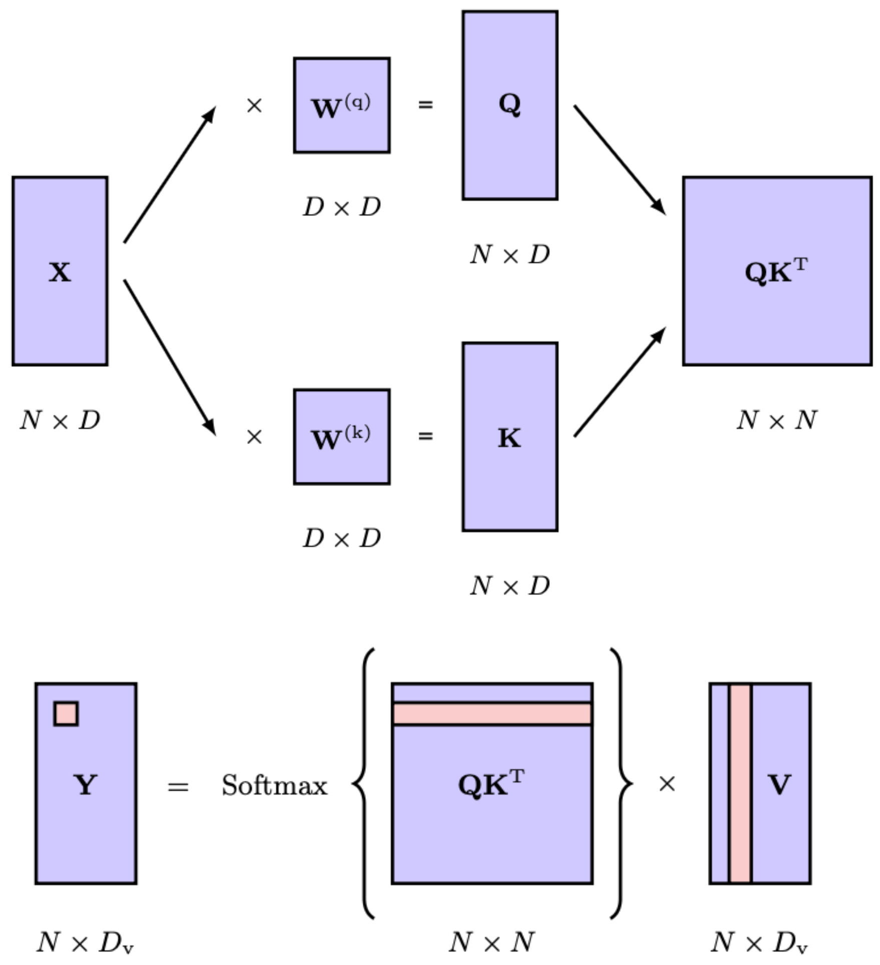
$\mathbf{W}^{(\mathrm{q})}$ is a $D \times D_{\mathrm{k}}$

$\mathbf{W}^{(\mathrm{v})}$ is a $D \times D_{\mathrm{v}}$

$\mathbf{Q}\mathbf{K}^{\mathrm{T}}$ is a $N \times N$

$\mathbf{Y}$ is a $N \times D_{\mathrm{v}}$

For convenience, we can set $D_{\mathrm{k}} = D$ and $D_{\mathrm{v}} = D$



Scaled Self-attention

- **Problem (unscaled):** Softmax saturates when logits are large $\rightarrow$ tiny gradients.
  In attention, logits are $q_m^\top k_n$. If q, k have i.i.d. entries with mean 0 and var 1, then
  $\mathbb{E}[q^\top k] = 0,\ \mathrm{Var}(q^\top k) = D_k \rightarrow$ magnitude grows with $\sqrt{D_k}$.

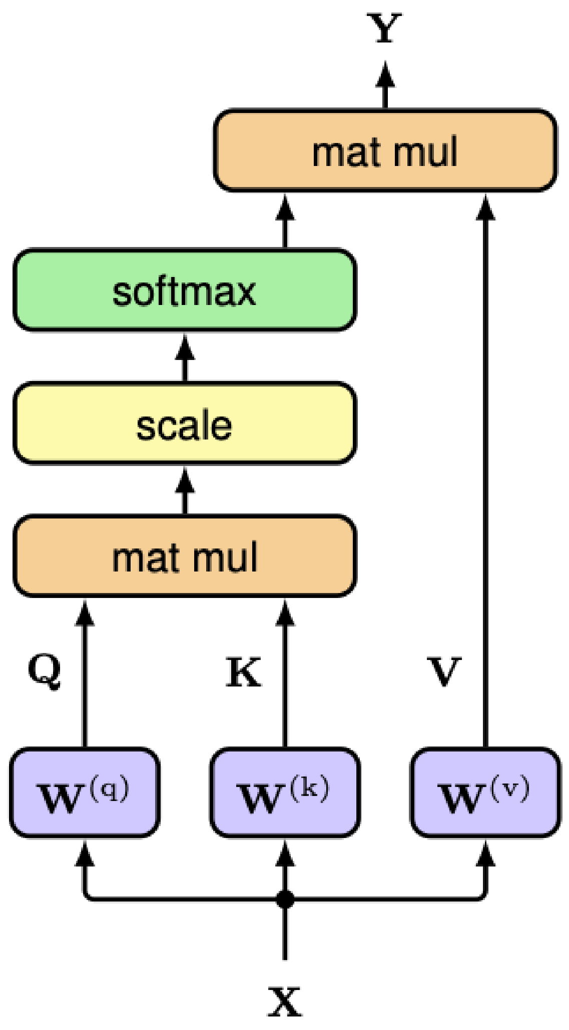
- **Consequence:** As key/query dim D_k increases, logits spread widens $\Rightarrow$ softmax becomes overly peaky, gradients vanish.

- **Fix (scaling):** Normalize by the std. dev.:

$$\mathbf{Y} = \mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V}$$

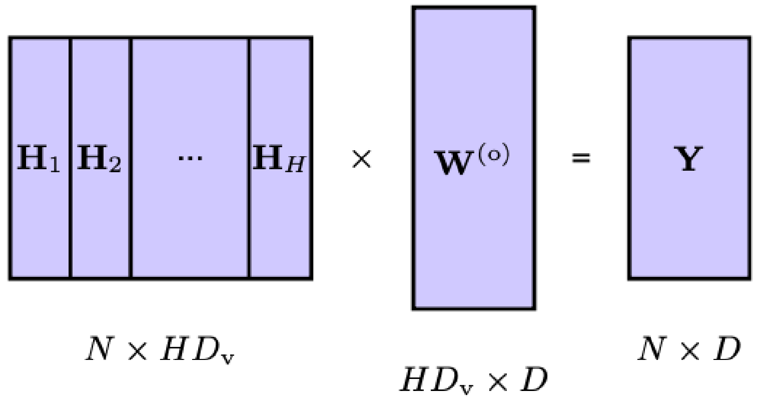
  (equivalently, softmax temperature $T = \sqrt{D_k}$).

- **Effect/intuition:** Keeps logits in a responsive range, stabilizes gradients, and makes behavior more dimension-agnostic.
  Scaling preserves ranking; it just reduces over-confidence.



Multi-Head attention

- A single attention head can blur/average multiple concurrent patterns.
- In language, different patterns matter at the same time (e.g., tense vs. vocabulary).
- To capture these simultaneously, we need multiple distinct attention patterns.



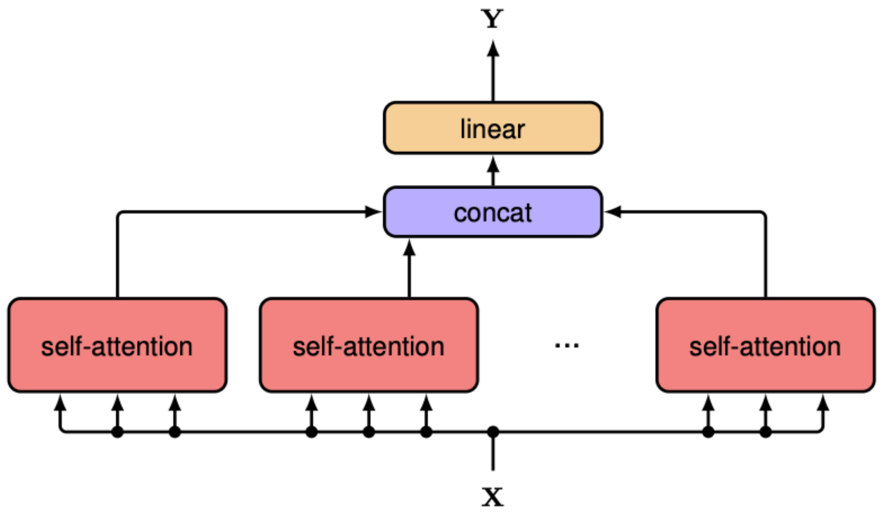
$$\mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{q})}$$

$$\mathbf{H}_h = \mathrm{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h) \qquad \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{k})}$$

$$h = 1, \ldots, H \qquad \mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{v})}$$

$\mathbf{H}_h$ is a $N \times D_\mathrm{v}$

$$\boxed{\underbrace{\mathbf{Y}(\mathbf{X})}_{N \times D} = \mathrm{Concat}\,\underbrace{[\mathbf{H}_1, \ldots, \mathbf{H}_H]}_{N \times HD_\mathrm{v}}\,\underbrace{\mathbf{W}^{(\mathrm{o})}}_{HD_\mathrm{v} \times D}}$$



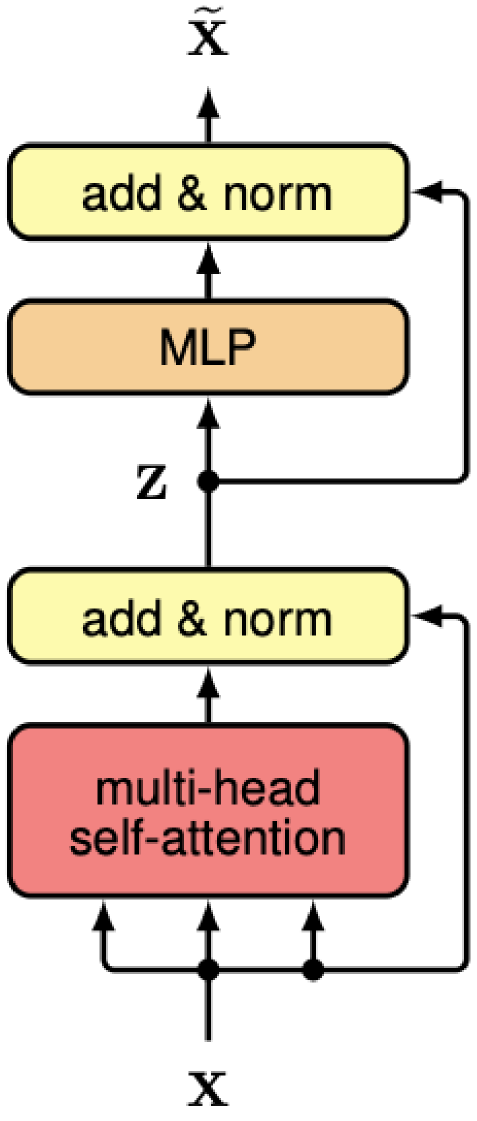
The transformer Layers

Residual

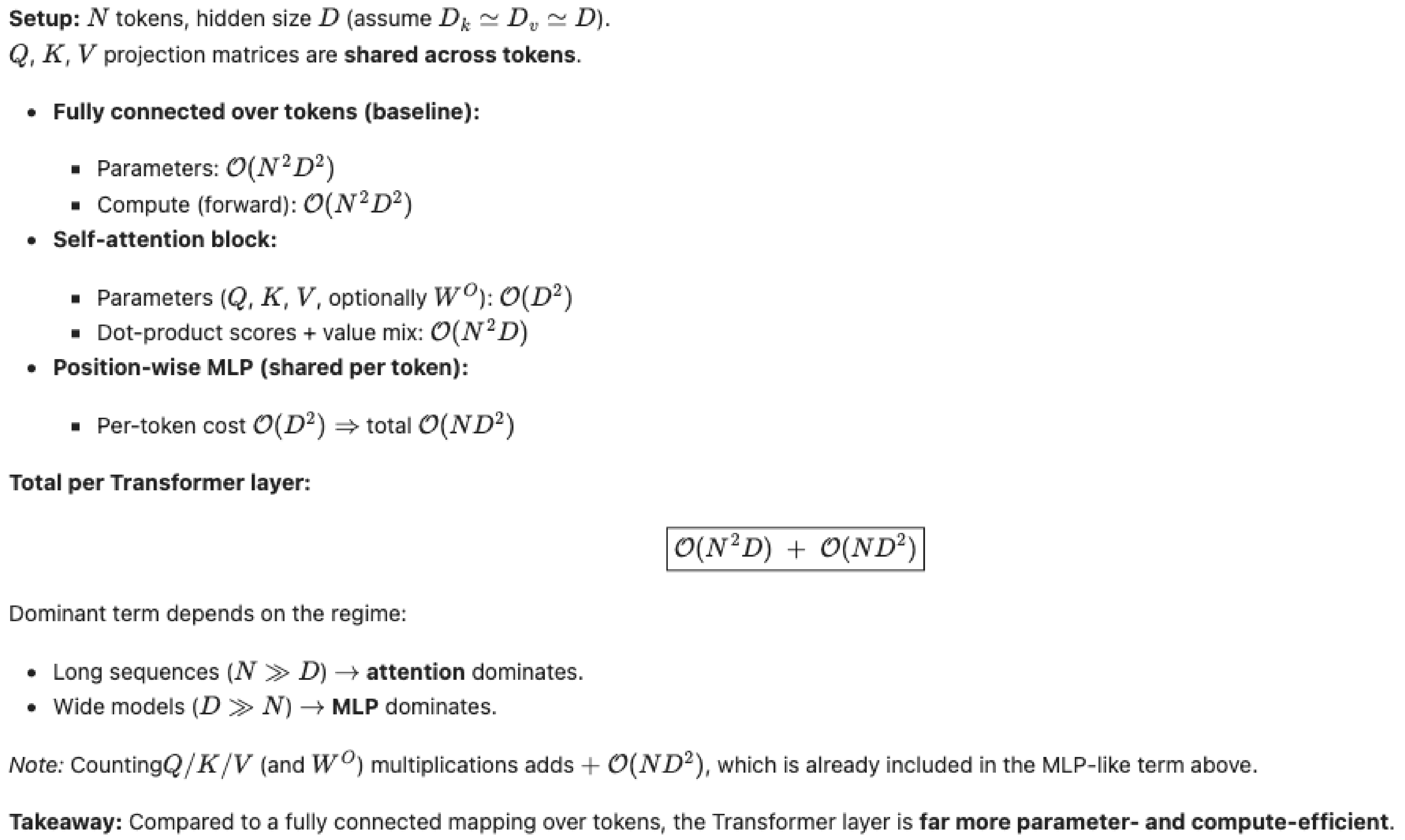
$N \times D$

$$\begin{tikzcd} \mathbf{X} \arrow[d] \\ \mathbf{Y}(\mathbf{X}) \arrow[d] \\ \mathbf{Z} = \mathrm{LayerNorm}\left[\mathbf{Y}(\mathbf{X}) + \mathbf{X}\right] \arrow[d] \\ \widetilde{\mathbf{X}} = \mathrm{LayerNorm}\left[\mathrm{MLP}\left[\mathbf{Z}\right] + \mathbf{Z}\right] \end{tikzcd}$$

LayerNorm (layer normalization) rescales each example (or token) independently by normalizing across its feature dimension.



Computational Complexity



Setup: N tokens, hidden size D (assume $D_k \simeq D_v \simeq D$).
Q, K, V projection matrices are **shared across tokens**.

- **Fully connected over tokens (baseline):**
  - Parameters: $\mathcal{O}(N^2D^2)$
  - Compute (forward): $\mathcal{O}(N^2D^2)$
- **Self-attention block:**
  - Parameters (Q, K, V, optionally W^O): $\mathcal{O}(D^2)$
  - Dot-product scores + value mix: $\mathcal{O}(N^2D)$
- **Position-wise MLP (shared per token):**
  - Per-token cost $\mathcal{O}(D^2) \Rightarrow$ total $\mathcal{O}(ND^2)$

Total per Transformer layer:

$$\boxed{\mathcal{O}(N^2D) \; + \; \mathcal{O}(ND^2)}$$

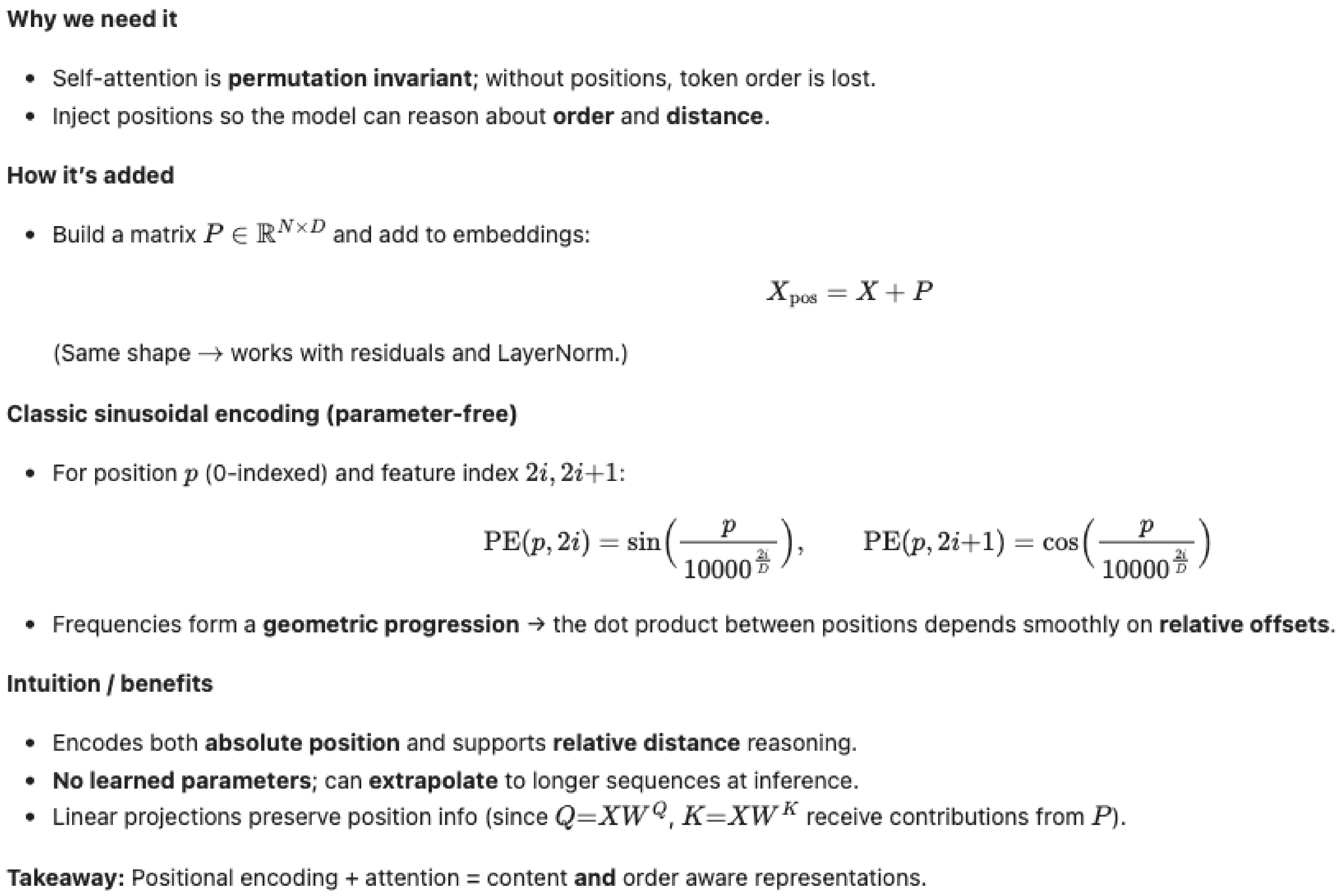
Dominant term depends on the regime:

- Long sequences ($N \gg D$) $\rightarrow$ **attention** dominates.
- Wide models ($D \gg N$) $\rightarrow$ **MLP** dominates.

Note: Counting$Q/K/V$ (and W^O) multiplications adds $+\,\mathcal{O}(ND^2)$, which is already included in the MLP-like term above.

Takeaway: Compared to a fully connected mapping over tokens, the Transformer layer is **far more parameter- and compute-efficient**.

Positional Encoding



Why we need it

- Self-attention is **permutation invariant**; without positions, token order is lost.
- Inject positions so the model can reason about **order** and **distance**.

How it's added

- Build a matrix $P \in \mathbb{R}^{N \times D}$ and add to embeddings:

$$X_{\text{pos}} = X + P$$

(Same shape $\rightarrow$ works with residuals and LayerNorm.)

Classic sinusoidal encoding (parameter-free)

- For position p (0-indexed) and feature index $2i, 2i{+}1$:

$$\mathrm{PE}(p, 2i) = \sin\left(\frac{p}{10000^{\frac{2i}{D}}}\right), \qquad \mathrm{PE}(p, 2i{+}1) = \cos\left(\frac{p}{10000^{\frac{2i}{D}}}\right)$$

- Frequencies form a **geometric progression** → the dot product between positions depends smoothly on **relative offsets**.

Intuition / benefits

- Encodes both **absolute position** and supports **relative distance** reasoning.
- **No learned parameters**; can **extrapolate** to longer sequences at inference.
- Linear projections preserve position info (since $Q{=}XW^Q$, $K{=}XW^K$ receive contributions from P).

Takeaway: Positional encoding + attention = content **and** order aware representations.

The transformer architecture

- Embedding to encode text
- Positional encoding to preserve the sequence order
- Multi-head self-attention for
  - Parallel computation
  - Multi-view learning
- Add & Normalize
  - Residual connection followed by layer normalization (not batchnorm)
- Position-wise feed-forward network
  - Pool/aggregate the "learned" information for each position
- Masking: each position in the decoder as allowed to only attend to all positions in the decoder up to the position

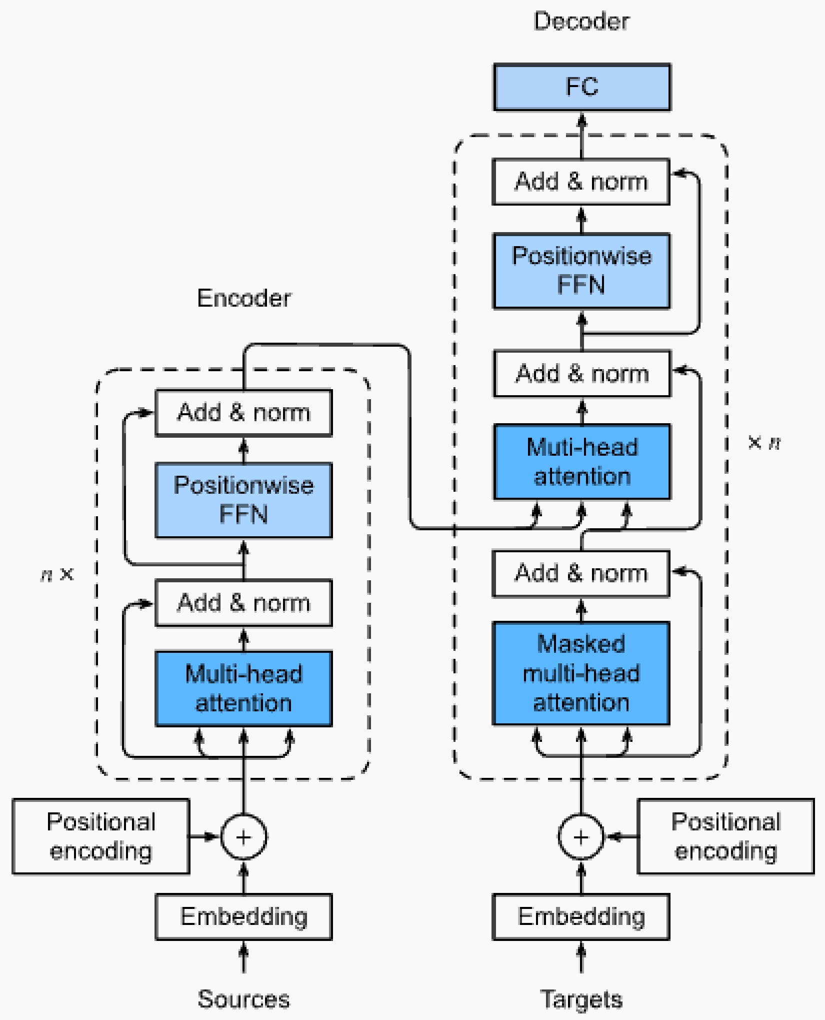


Fig. 11.7.1 The Transformer architecture.

Genealogy of Transformer Based LLMs

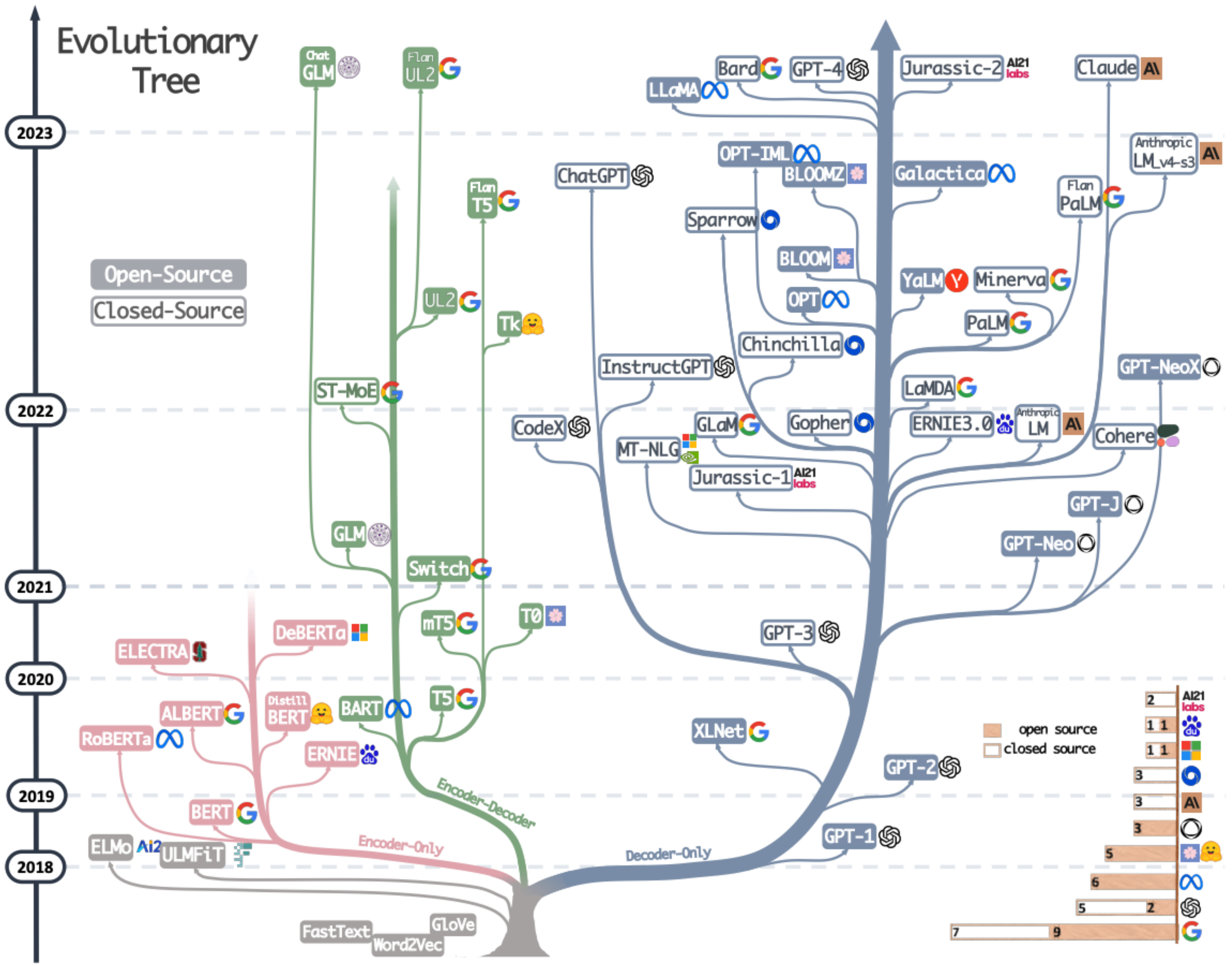


Fig. 1. The evolutionary tree of modern LLMs traces the development of language models in recent years and highlights some of the most well-known models. Models on the same branch have closer relationships. Transformer-based models are shown in non-grey colors: decoder-only models in the blue branch, encoder-only models in the pink branch, and encoder-decoder models in the green branch. The vertical position of the models on the timeline represents their release dates. Open-source models are represented by solid squares, while closed-source models are represented by hollow ones. The stacked bar plot in the bottom right corner shows the number of models from various companies and institutions.

source

References

- Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014
- Sequence to Sequence Learning with Neural Networks, 2014
- Neural Machine Translation by jointly learning to align and translate, 2014
- Attention is All you need, 2017
- Uday Kamath et al, Transformers for Machine Learning: a deep dive
- Charu C. Aggarwal, Neural Networks and Deep Learning: A textbook,