Transformers and Algorithmic Skepticism: Rethinking Embedding Spaces in Machine Learning

Spring 2025

Dr. Jeff M. Byers

for

Laboratory for Computational Physics/Physics of Data

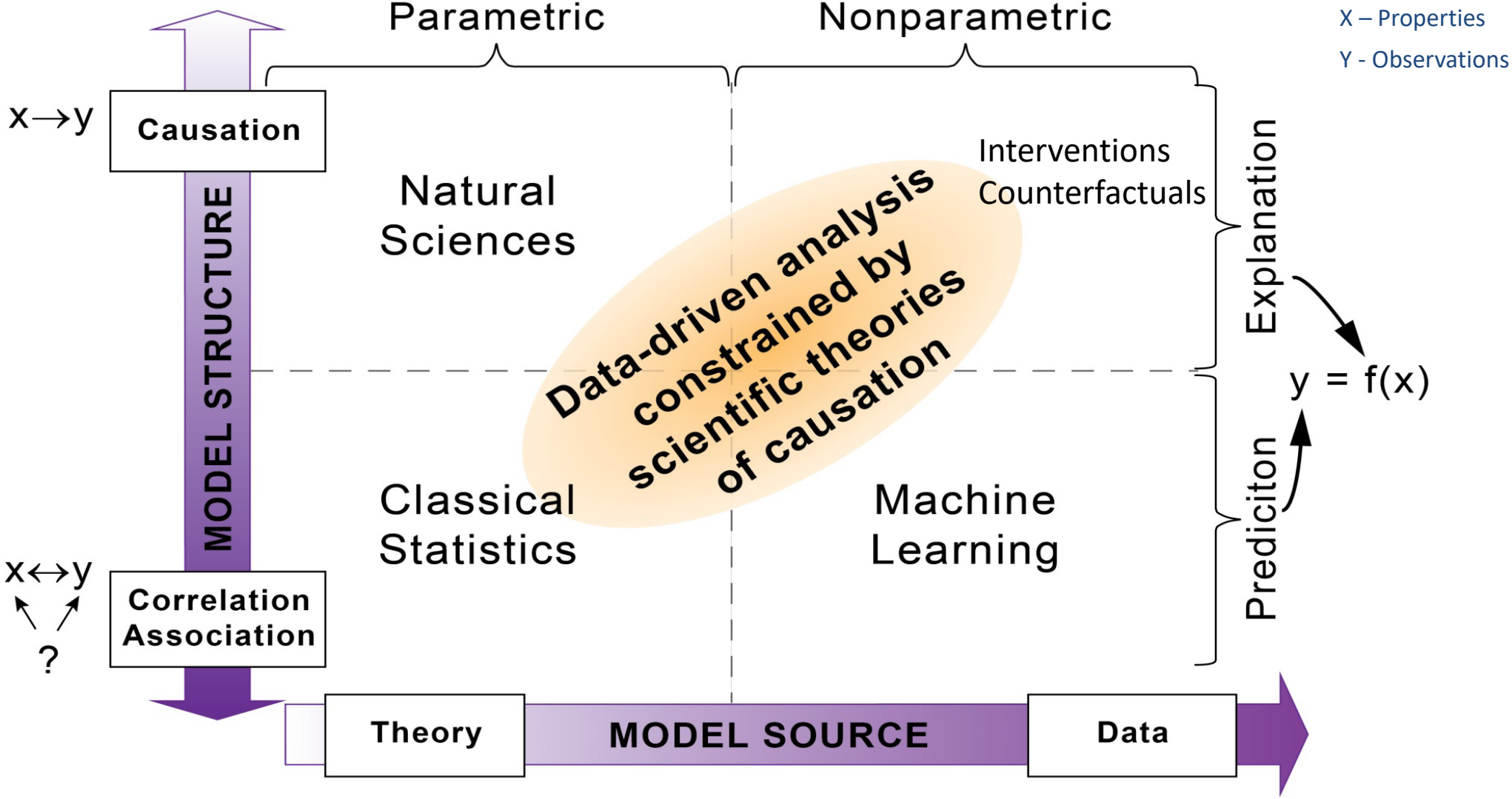
Wednesday April 23

Title & Abstract

Title: *Transformers and Algorithmic Skepticism: Rethinking Embedding Spaces in Machine Learning*

Abstract: Machine learning is being transformed. Self-Supervised Learning and the Transformer architecture is revolutionizing our everyday lives and the analysis tools of science and engineering. But what is the underlying reason for this breakthrough? In part, it is increased data and computational resources, but the Transformer is also a new algorithmic approach. The key lies in first understanding what machine learning is as fundamentally distinct from other forms of data analysis because it exchanges probabilistic relationships for geometric distances in an embedding space. The Transformer takes a more skeptical attitude towards this embedding space than other algorithms and leverages the data and GPU architecture to treat the embedding as a contextual representation for each part of the data. This talk discusses the algorithmic skepticism of the Transformer towards treating everything as a vector by providing a description of the Attention mechanism from a geometric perspective.

Types of Models



Machine Learning vs. Artificial Intelligence

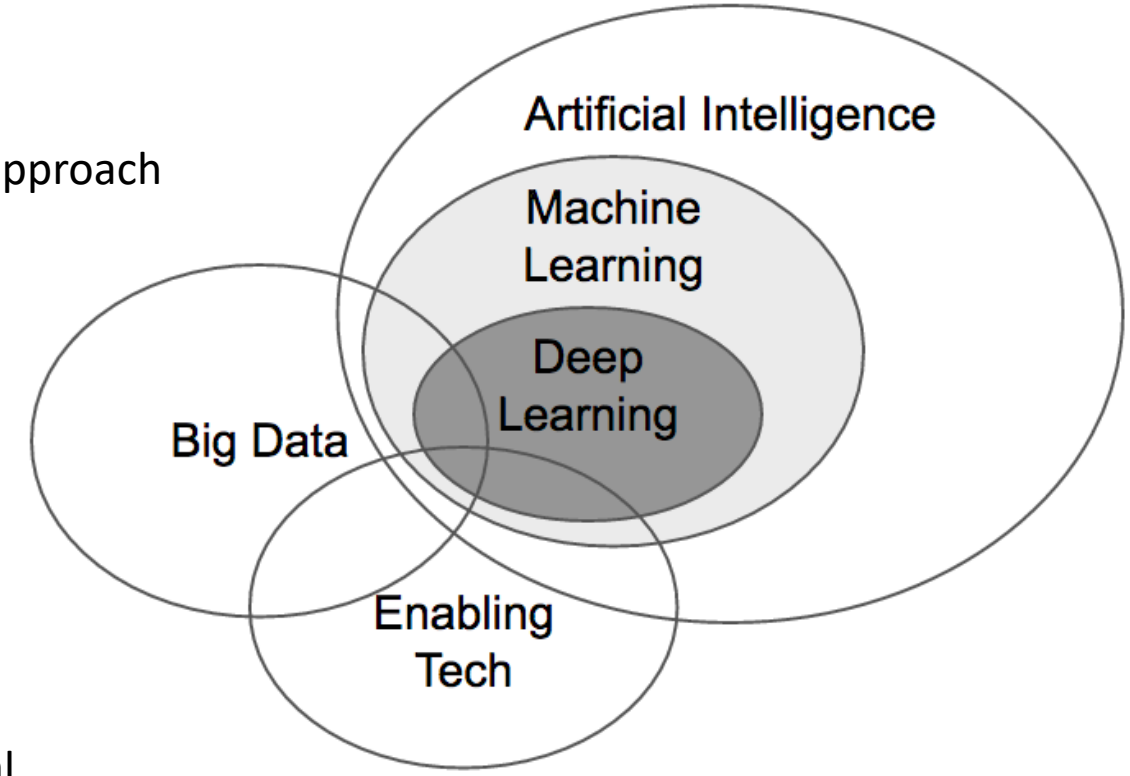
Connectionist vs. Symbolic (GOFAI)

Artificial Intelligence:

- Inefficient training procedures
- Concept/Language-driven → Top-down approach
- Database implementations (e.g., CYC)
- Elements of Human Commonsense
- Can represent causal knowledge.

Machine Learning:

- Highly-efficient training procedures
- Data-driven → Bottom-up approach
- Mindless implementation
- Devoid of commonsense
- Lack of causal connections, only statistical.



Computer Scientists – Data agnostic, Blind to Risk in Decision-Making
Statisticians – Resistant to probability as reasoning under uncertainty
Physicists – Mind Projection Fallacy (Probability is out in the WORLD)

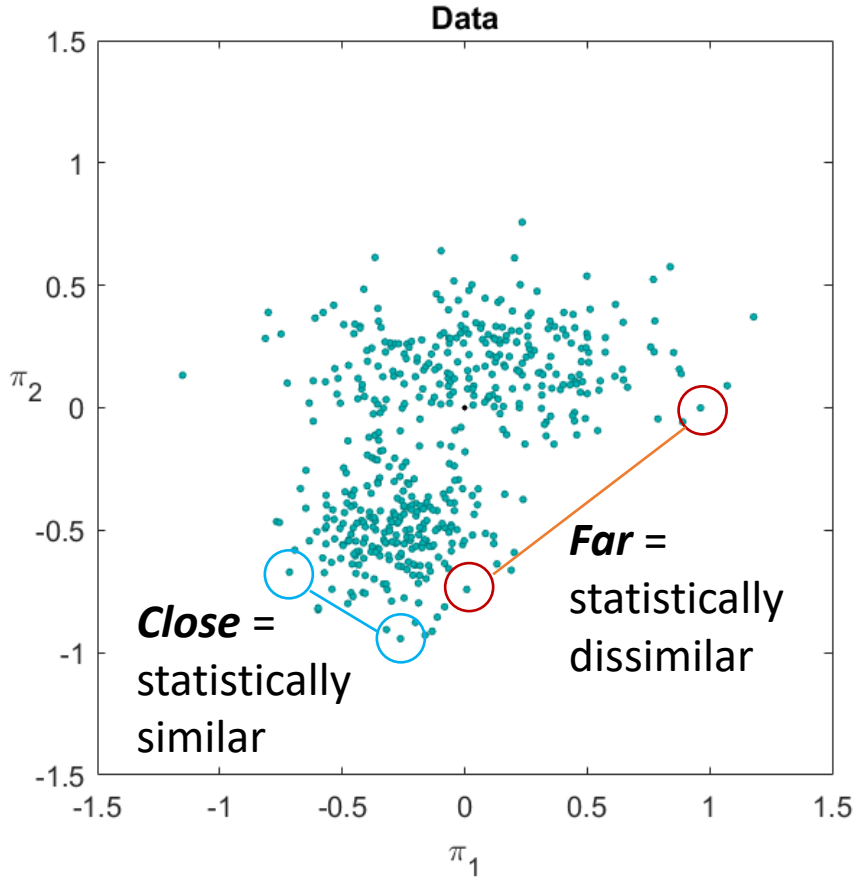
What is Machine Learning?

Create/Learn an Embedding or Feature Vector space that encodes information about *statistical similarity as distances*.

$$-\log p_{\text{similarity}}\left(y, y'\right) \approx \frac{1}{2}\left\|\mathbf{y}-\mathbf{y}'\right\|_2^2$$

How much DATA is enough?

If distant data points are mapped nearby in the model space, then there has to be significant inferential evidence (more data), and likewise if local data points are mapped to large separations by the model.

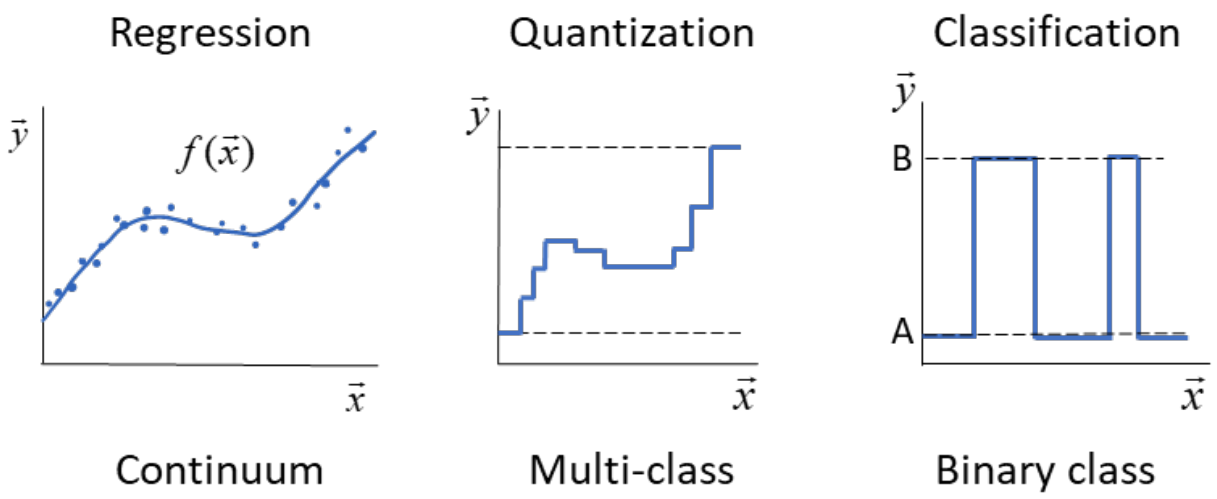


Machine Learning

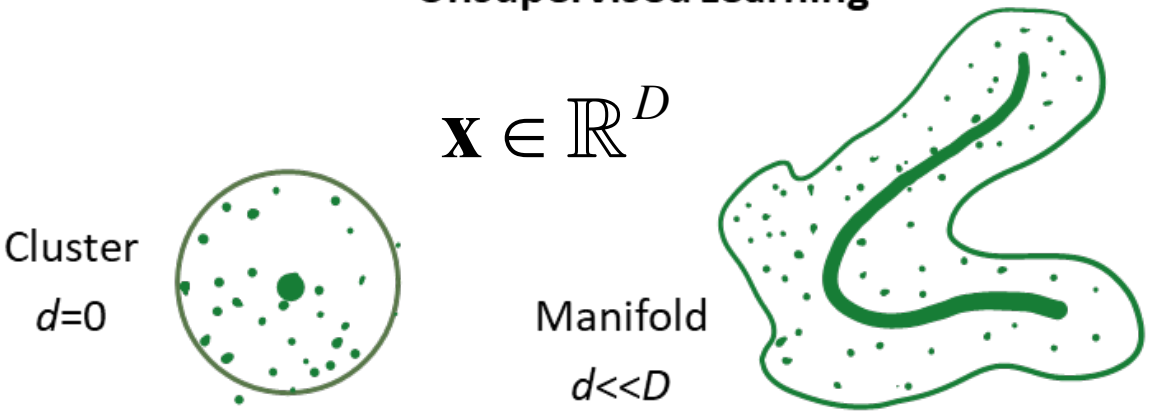
Relevance Measure → "Labels"

Similarity Measure → Distances

Supervised Learning



Unsupervised Learning

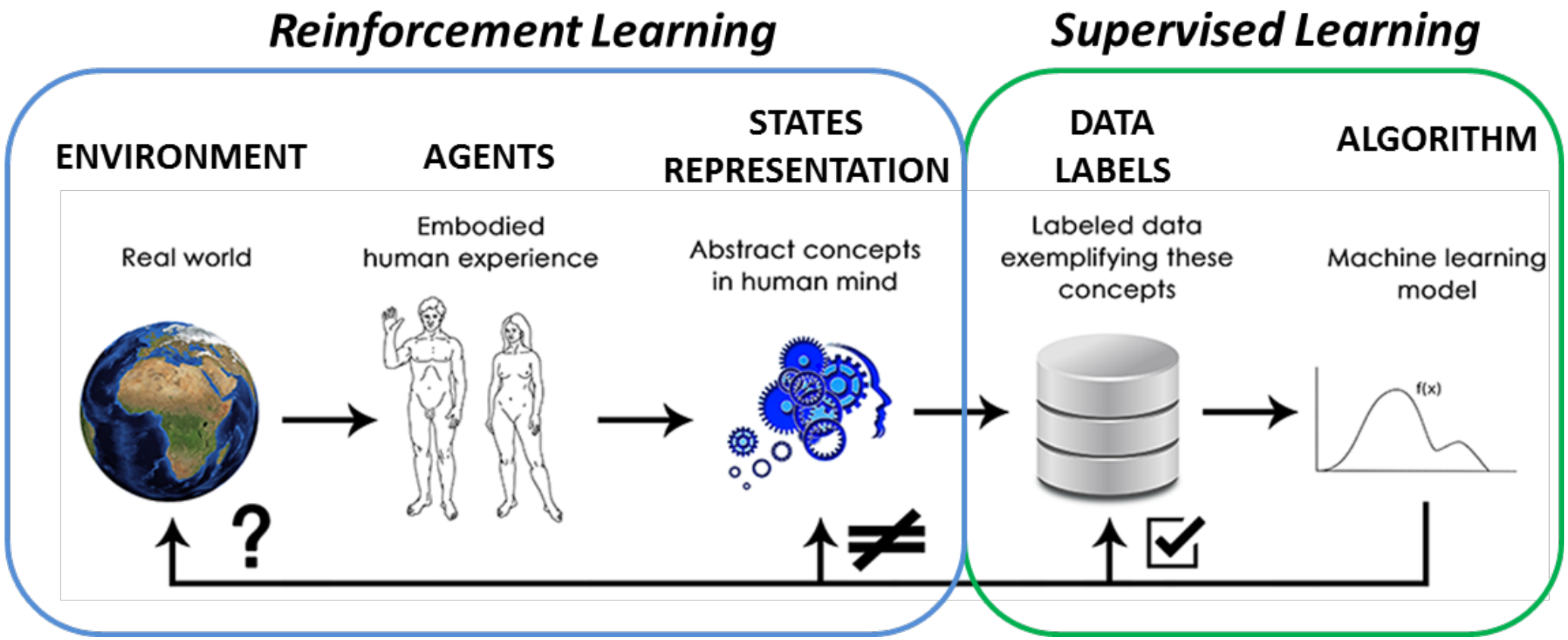


$$f : \mathbb{R}^D \to L \qquad y = f(\mathbf{x})$$

$$p : \mathbb{R}^D \to \mathbb{R}_+ \qquad \mathbf{x} \sim p(\mathbf{x})$$

Can AI Emerge from Supervised Learning?

Building a mirror ***NOT*** a mind

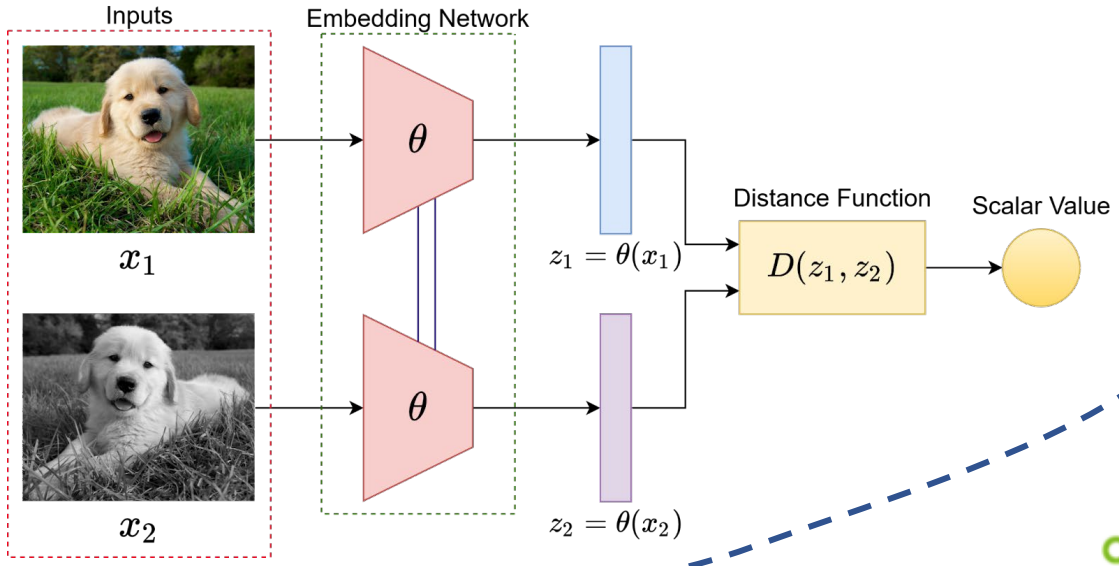


The Emergence of Self-Supervised Learning

Fundamentally, it is about the Learned MODEL predicting Synthetic or Missing DATA.

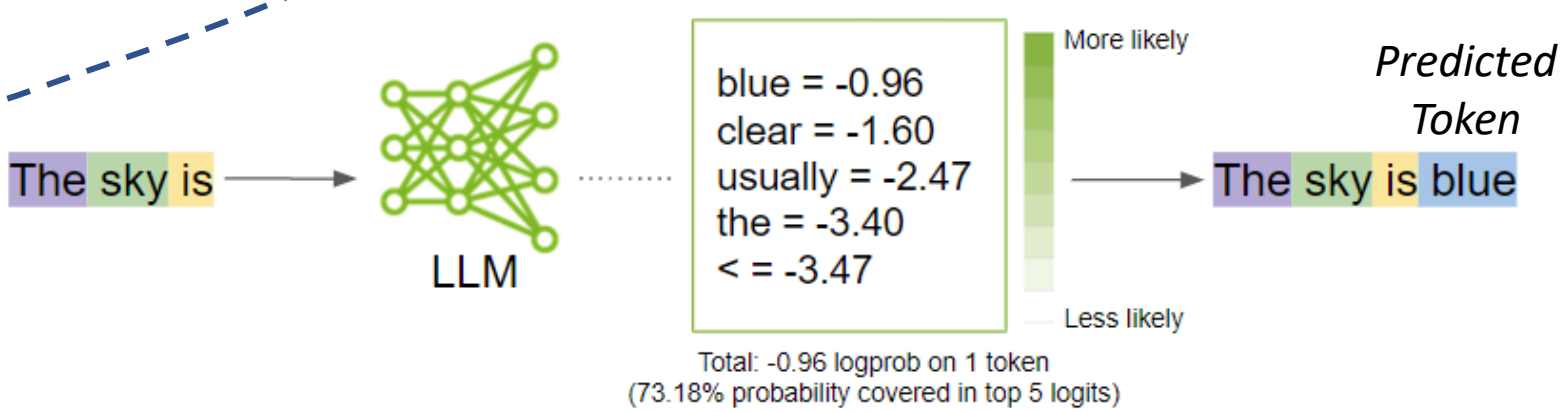
1 Data Augmentation

Apply symmetry operations on DATA that seem irrelevant to the class labels.

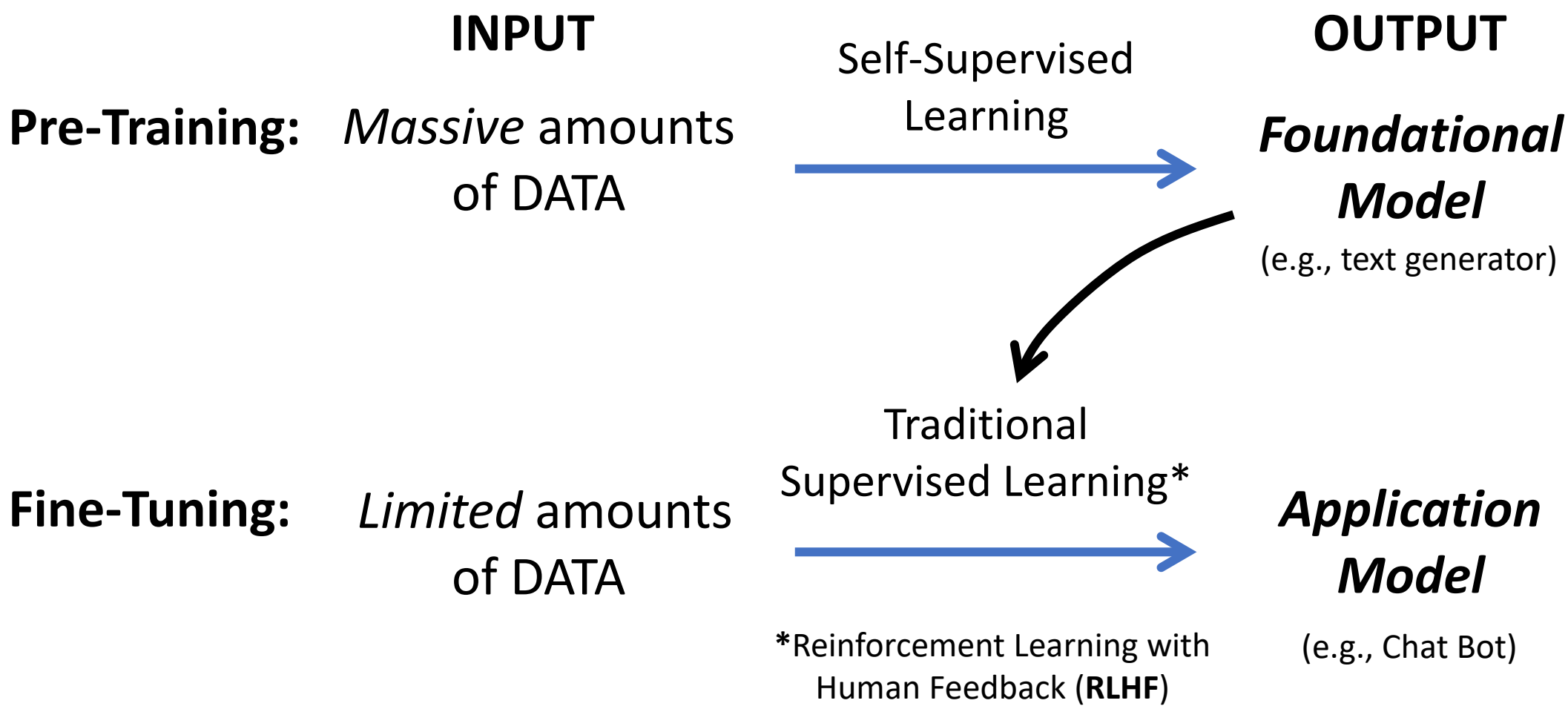


2 Missing data prediction

Mask parts of the data and train to predict.

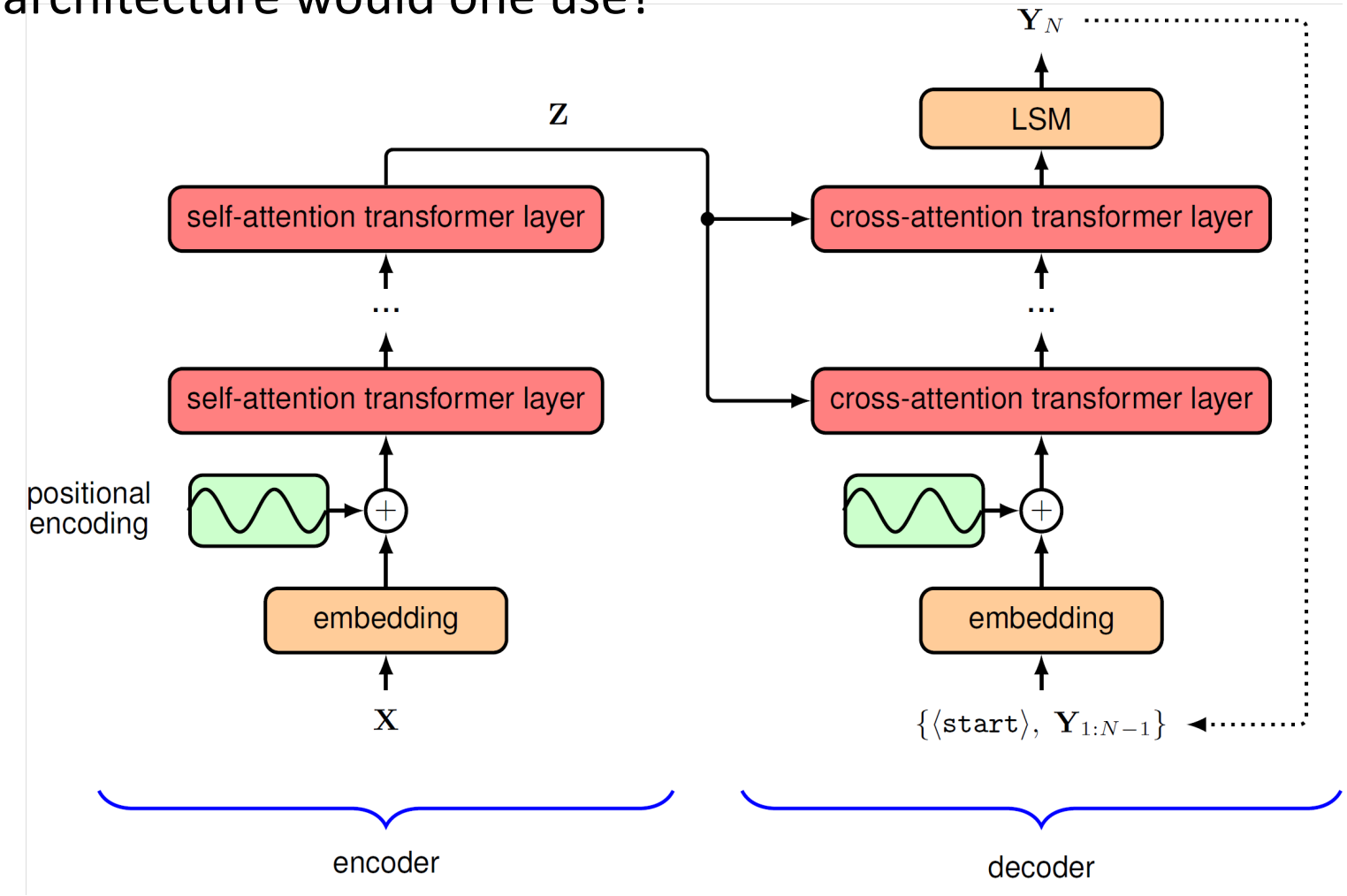


The *New* Machine Learning via *Self-Supervision*

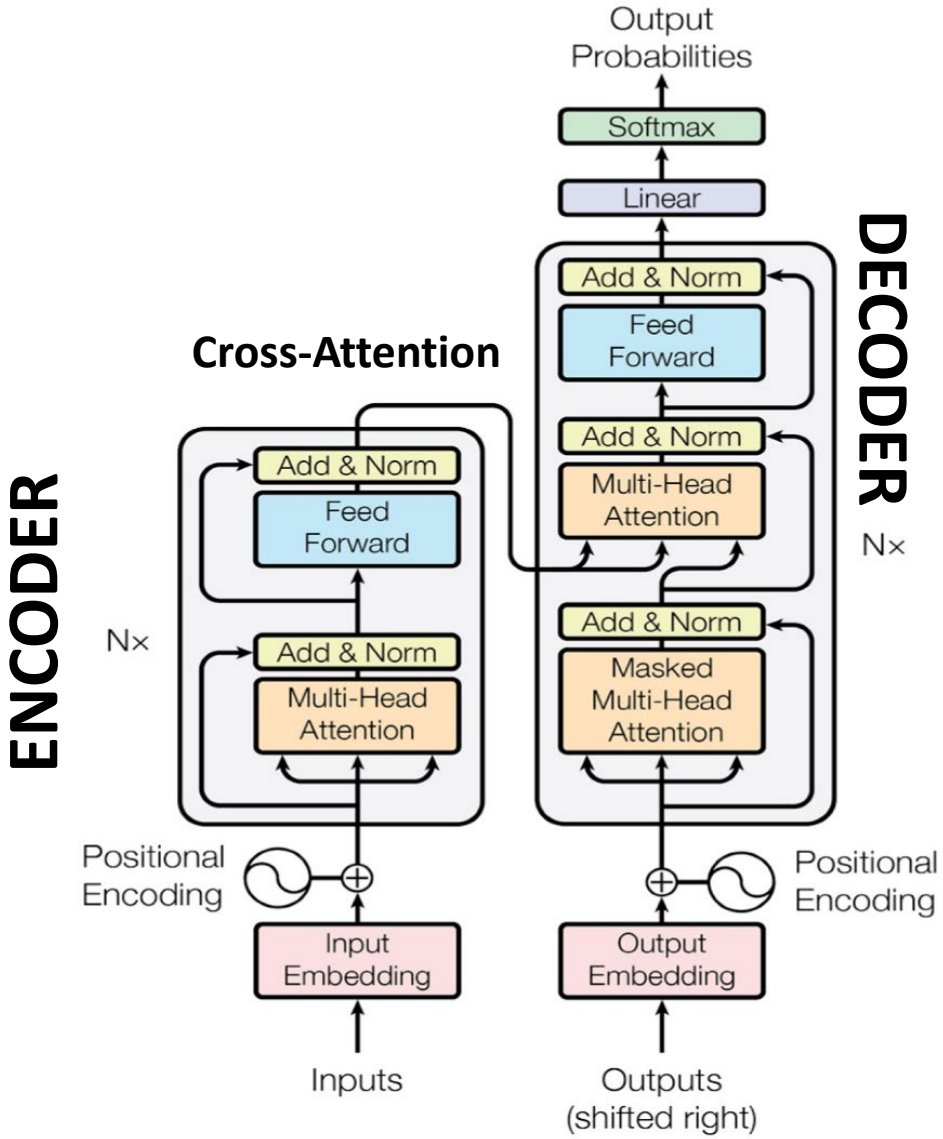


Encoder-Decoder and Cross-Attention

So what architecture would one use?

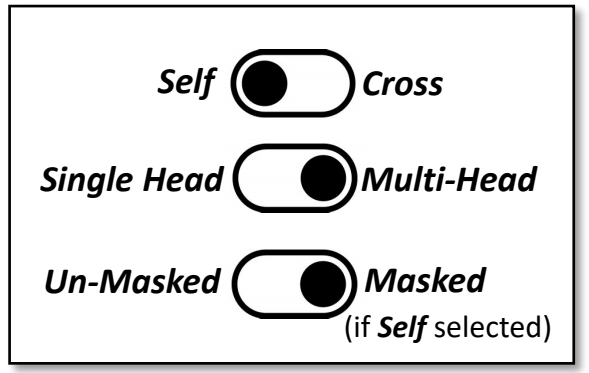


Transformer: "Attention is All You Need"



GPT = Generative Pre-trained Transformer
(DECODER only architecture)

Attention →

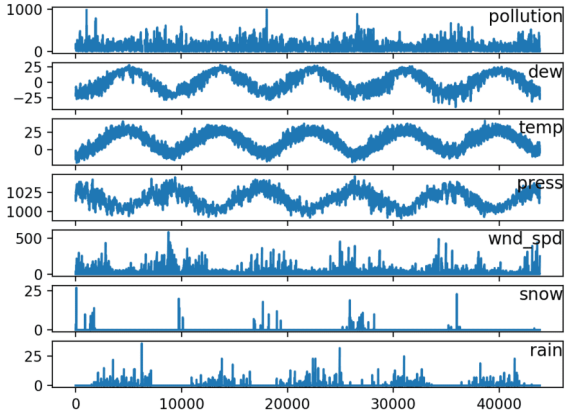


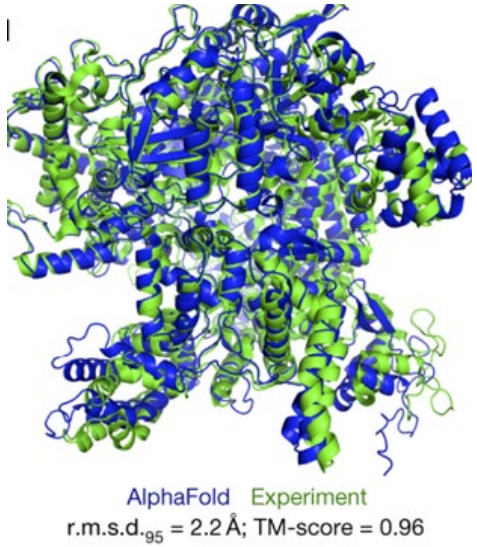
Transformers in Generative Models

- **GPT-3 and GPT-4 (OpenAI)** - These models can generate human-like text, translate languages, summarize text, and answer questions, showcasing the ability of Transformers to handle a wide range of natural language processing tasks.
- **BERT (Bidirectional Encoder Representations from Transformers)** - Developed by Google, BERT is used for understanding the context of words in a sentence by looking at both the left and right sides of a word simultaneously. It's widely used in search engine optimization, sentiment analysis, and more.
- **Audio Processing** - Speech recognition (*Wav2Vec*, *Whisper*), music generation (*Jukebox*), speech synthesis (*Tacotron*), audio classification.
- **Image Processing** - Image classification (*ViT*), object detection and segmentation (*DETR*), image generation (*DALL-E*), image super-resolution.
- **Multi-Modal Analysis and Generative Tasks** - Multi-modal Transformers (*MMT*), CLIP, video analysis, integration with GANs, speech-to-text and text-to-speech systems.

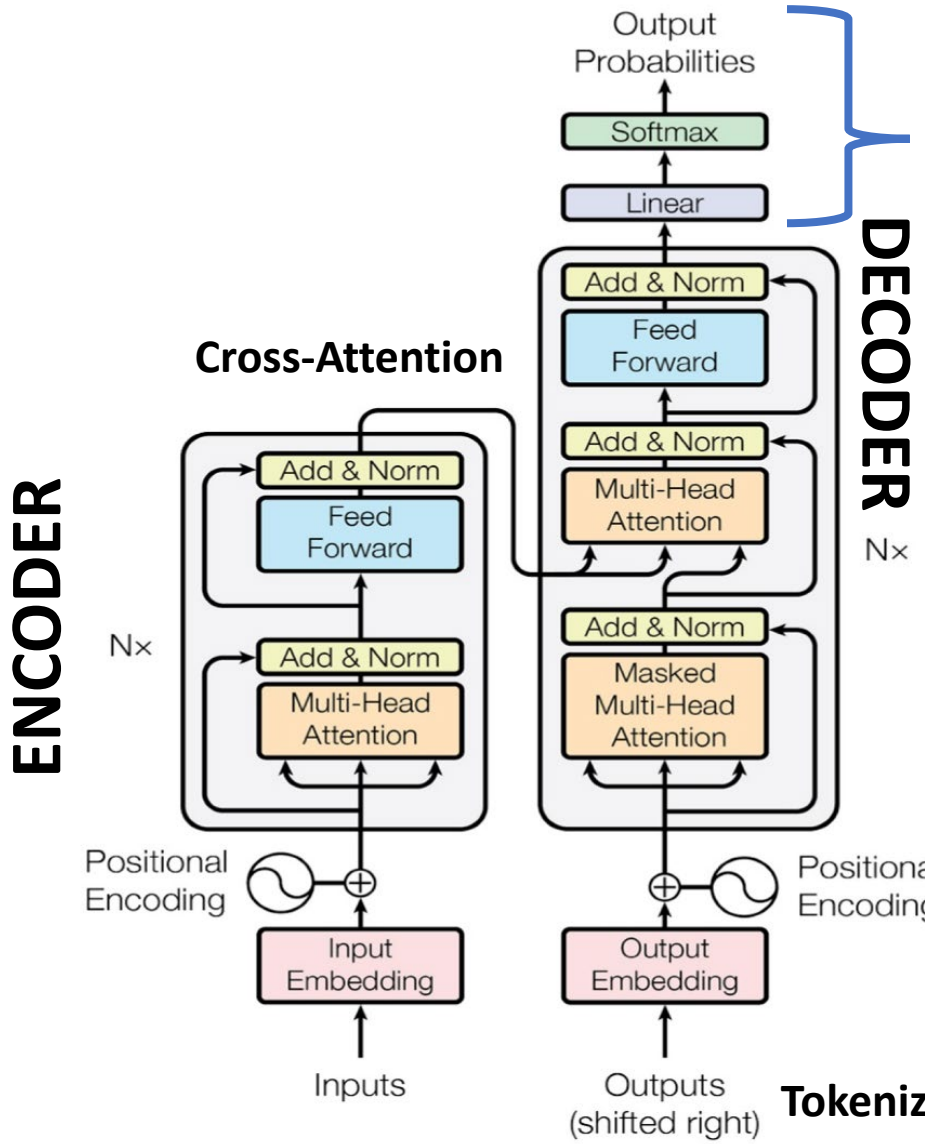
Transformers in Scientific Applications

- **Time Series Modeling** - Forecasting financial markets, energy demand prediction, weather forecasting, anomaly detection in industrial systems, healthcare time series analysis.
- **Dynamical Systems** - Modeling physical systems, control systems, climate dynamics, biological systems, economic and social systems.
- **Astronomy** - In astronomy, Transformers are used for tasks such as classifying celestial objects, analyzing light spectra from stars and galaxies, and identifying exoplanets from telescope data.
- **Molecule Generation** - Transformers are used to design novel molecules for drug discovery. They can generate chemical compounds with desired properties by understanding the complex relationships between molecular structures.
- **Protein Folding** - AlphaFold uses Transformer architecture to predict protein structures with remarkable accuracy. This breakthrough has significant implications for biology and medicine, such as drug discovery and understanding diseases.
- **Genomic Sequence Analysis** - Transformers help in the analysis of genomic sequences, identifying patterns and anomalies that are crucial for understanding genetic diseases and developing gene therapies.





“Attention is All You Need” … and few other tricks

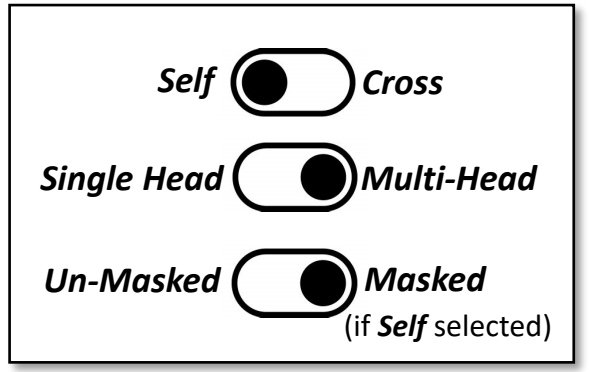


Inverse Embedding back to Tokens, “W^{-1}”

Add & Normalization

Feed-Forward Neural Network

Attention →



Positional Encoding

Token Embedding, W

Tokenized Sequence

Encoding Layer: Word Encoding

Word Encoding: Each word/token maps to a vector in a D-dimensional space using a previously learned dictionary.

Semantic dependency

$$P_{similarity}(a_i, a_j)$$

a_1= **Girl**
a_2= **Game**
a_3= **Queen**

Embedding

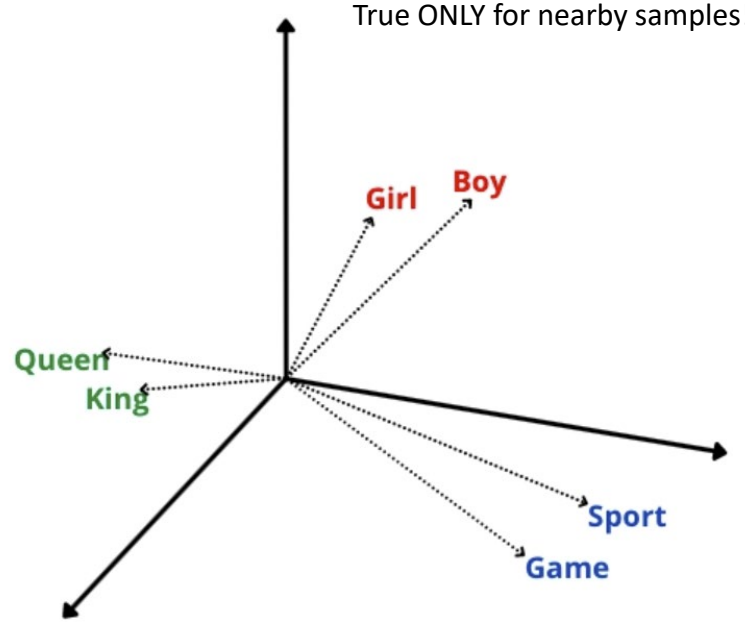
$$\mathbf{x}_k \in \mathbb{R}^D$$

$\mathbf{x}_1 = f(a_1) \rightarrow$ 1.2 -0.1 4.3 3.2
$\mathbf{x}_2 = f(a_2) \rightarrow$ 0.4 2.5 -0.9 0.5
$\mathbf{x}_3 = f(a_3) \rightarrow$ 2.1 0.3 0.1 0.4

Word embedding space

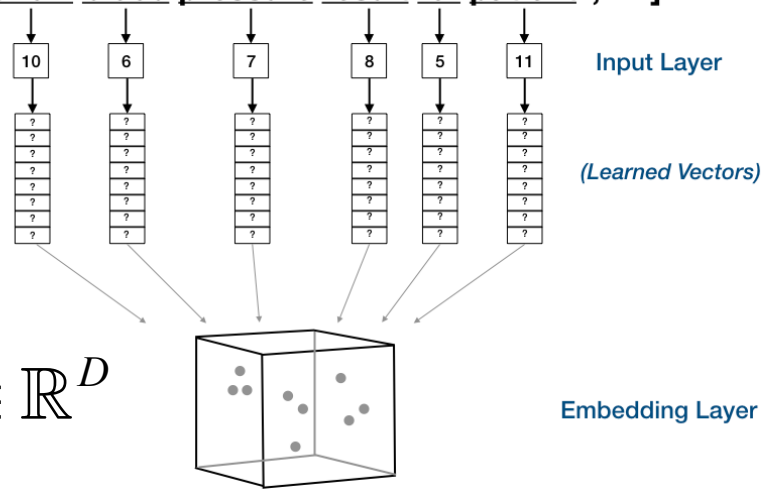
$$\left|\mathbf{x}_i - \mathbf{x}_j\right|^2 \approx -\log P_{similarity}(a_i, a_j)$$

True ONLY for nearby samples!



Encoding a sentence:

["I want to search for blood pressure result history", "Show blood pressure result for patient", ...]

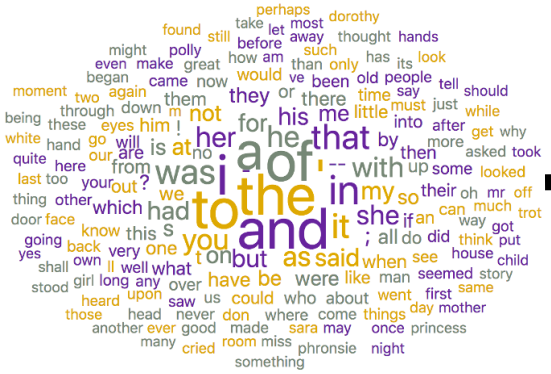


$$\mathbf{x}_k \in \mathbb{R}^D$$

i	1
want	2
to	3
search	4
for	5
blood	6
pressure	7
result	8
history	9
show	10
patient	11
...	
LAST	20

Fine-Tuning a Pre-Trained Large Language Model

General language context

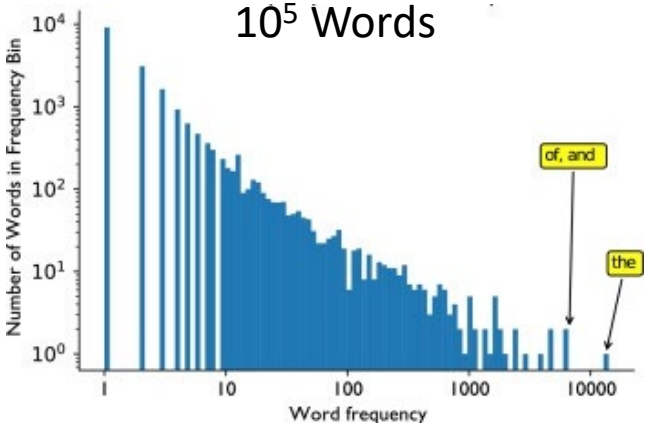


Special Interest context

Training:

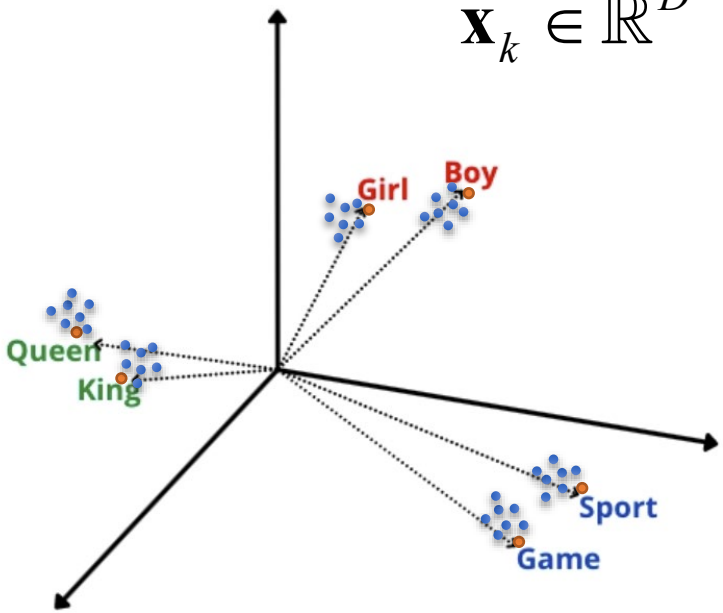
Frequency Distribution of Words

10^5 Words



Word + Position embedding space

$$\mathbf{x}_k \in \mathbb{R}^D$$



Embedding space: Form words to vectors

Input sequence: It is hard to understand … Transformers.

Tokenized sequence: $T_{seq} = \boxed{t_1} \; \boxed{t_2} \; \cdots \; \boxed{t_N}$

N = # of Tokens in buffer

K = Vocabulary Size of Tokens

D = dimensionality of embedding

Embedded sequence:

$$\mathbf{X}_{seq} = \overbrace{\mathbf{T}}^{N\times K} \cdot \overbrace{\mathbf{E}}^{K\times D} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ - & \mathbf{x}_2 & - \\ & \vdots & \\ - & \mathbf{x}_N & - \end{pmatrix} \in \mathbb{R}^{N\times D}$$

One-Hot Vector Encoding of Tokens

Embedding matrix

Positional Encoding

This is the step in the Transformer where Position in Sequence is used.

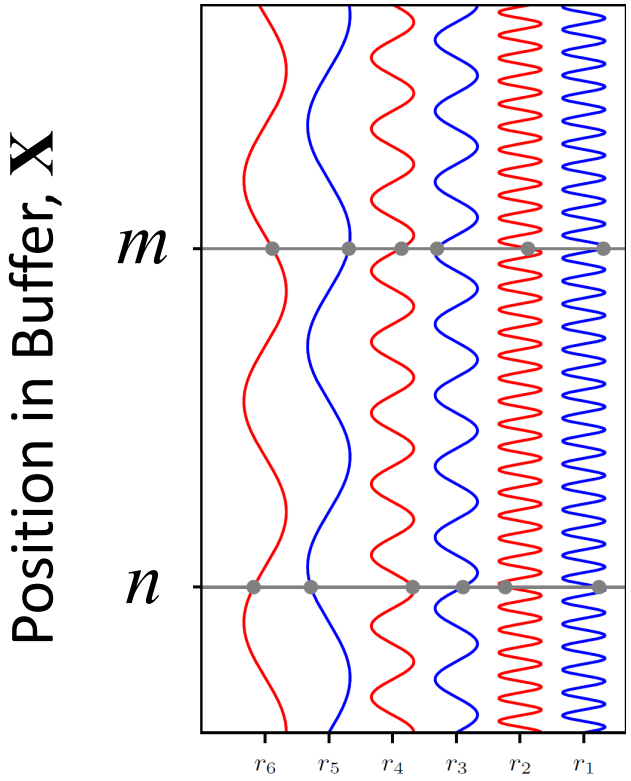
$$\text{PosEnc}(n, 2k) = \sin\left(n / L^{2k/D}\right)$$

$$\text{PosEnc}(n, 2k+1) = \cos\left(n / L^{2k/D}\right)$$

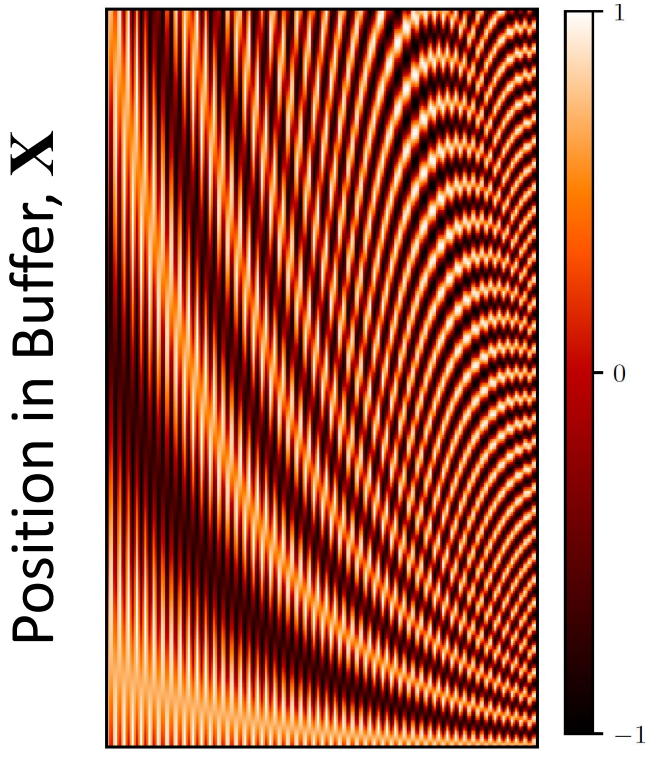
$$k = 0, \ldots, \tfrac{1}{2}D$$

N=200, D=100, L=30,

$N \times D$

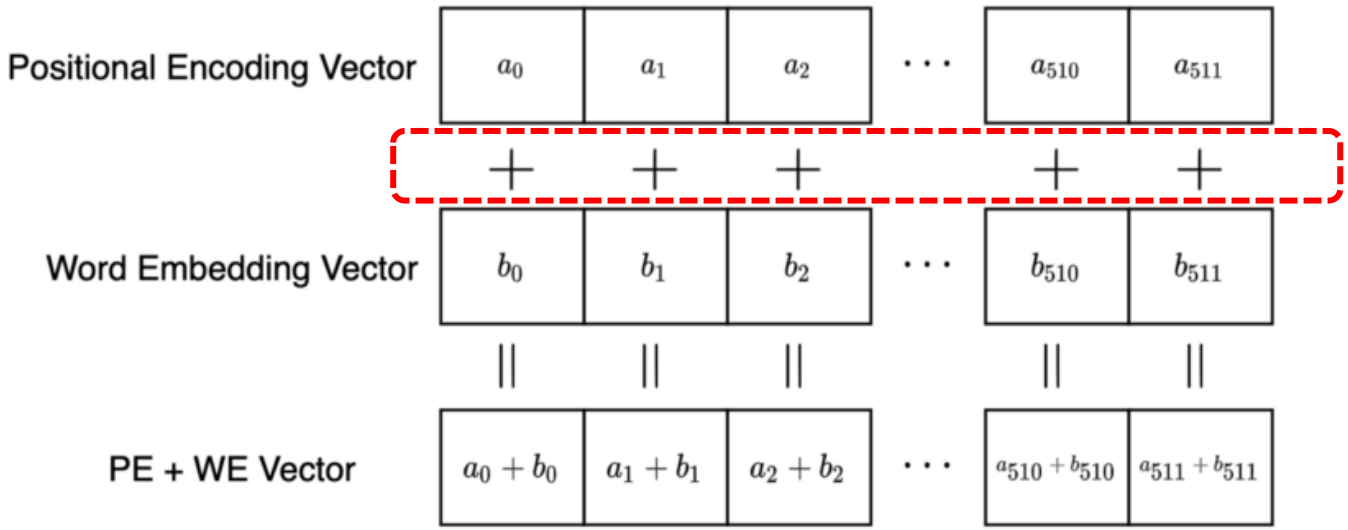


$$\mathbf{P} = \sum_{i=1}^{D} r^i \hat{\mathbf{e}}_i =$$



Encoding Layer: The *Miracula Summa*

The dimensionality of the word encoding is the same as the positional encoding!



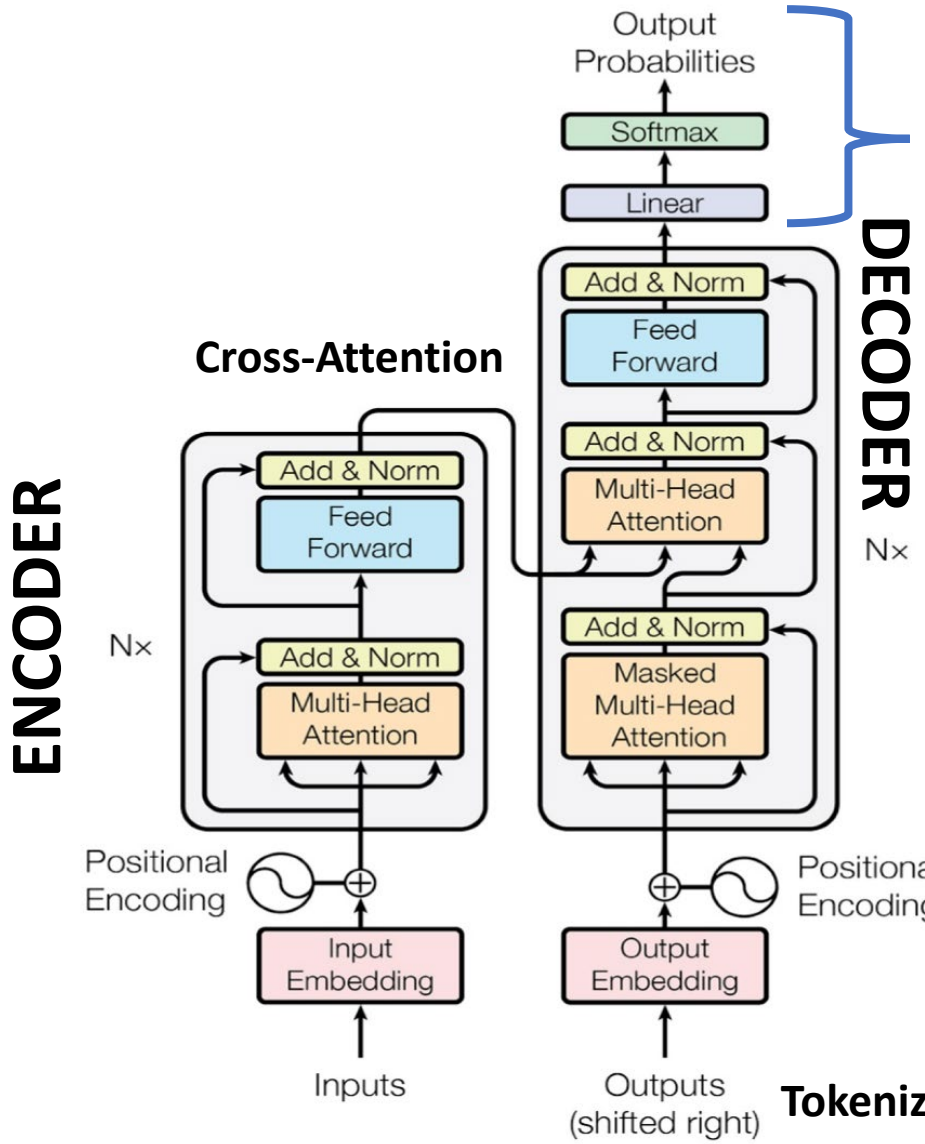
$$D_{\text{word}} = D_{\text{position}}$$

ADD THEM TOGETHER!

One-Hot Coding of Buffer | Embedding Map | Position Encoding Matrix

$$\underset{N\times D}{\mathbf{X}_{\text{initial}}} = \underset{N\times K}{\mathbf{T}_{\text{initial}}} \cdot \underset{K\times D}{\mathbf{E}} + \underset{N\times D}{\mathbf{P}}$$

“Attention is All You Need” … and few other tricks

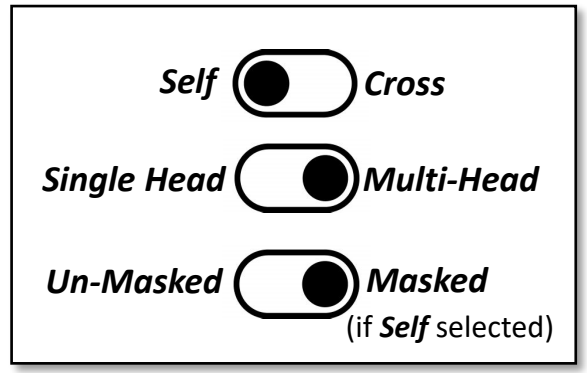


Inverse Embedding back to Tokens, “$\mathbf{E}^{-1}$”

Add & Normalization

Feed-Forward Neural Network

Attention →



Positional Encoding

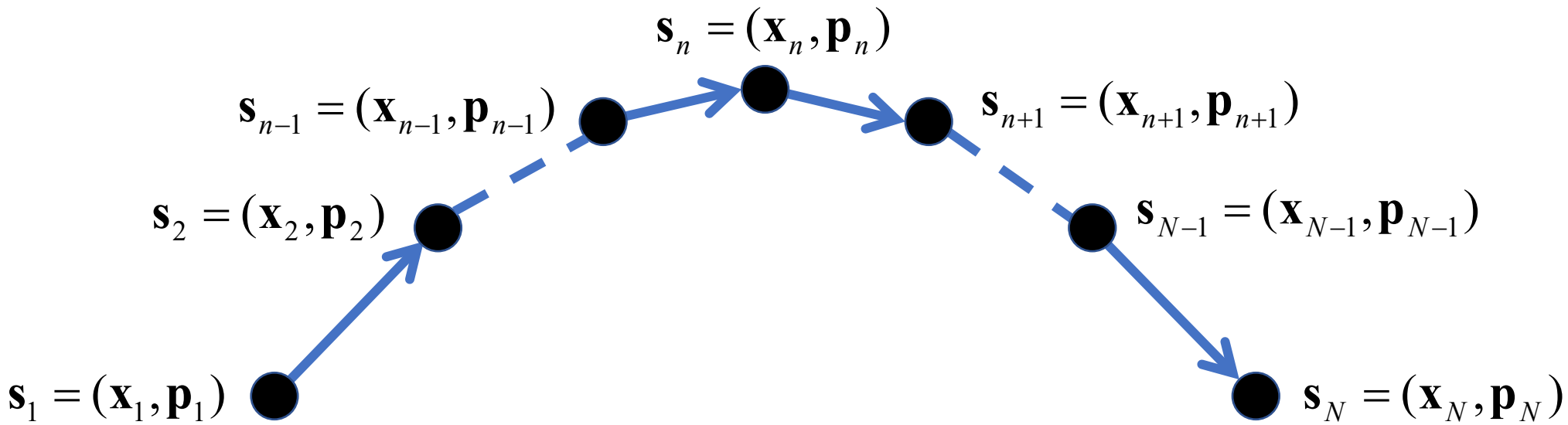
Token Embedding, $\mathbf{E}$

$$\mathbf{X}_{\text{initial}} = \mathbf{T}_{\text{initial}} \cdot \mathbf{E} + \mathbf{P}$$

Tokenized Sequence

Trusting the representation/embedding

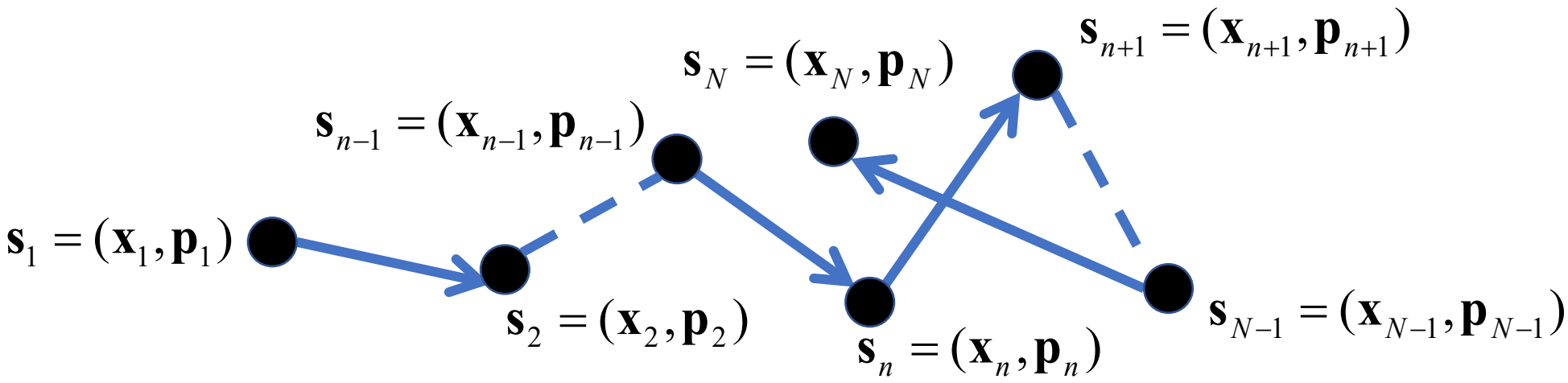
Newtonian Mechanics: How much do you trust the *hidden state* momentum, $\mathbf{p}$?



Predicting the next position: $\mathbf{x}_n = \mathbf{x}_{n-1} + \frac{\mathbf{p}_{n-1}}{m} \cdot \Delta t$

Trusting the representation/embedding

Newtonian Mechanics: How much do you trust the hidden state momentum, $\mathbf{p}$?



Predicting the next position: $\mathbf{x}_n = \mathbf{x}_{n-1} + \frac{\mathbf{p}_{n-1}}{m} \cdot \Delta t$

Not a good hidden state in the presence of random forces …

Types of Sequences Models

Bag-of-Words: $p(\mathbf{x}_1,\ldots,\mathbf{x}_N)=\prod_{n=1}^{N} p(\mathbf{x}_n)$

Discards the sequence information!

Auto-Regressive Model (*n*-gram):

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_N)=\prod_{n=1}^{N} p(\mathbf{x}_n \mid \mathbf{x}_1,\ldots,\mathbf{x}_{n-1})$$

Stores sequential information

Recurrent Neural Networks:

HIDDEN State

$$p(\mathbf{x}_1,\ldots,\mathbf{x}_N)=\prod_{n=1}^{N} p\left(\mathbf{x}_n \mid \mathbf{h}_{n-1}(\mathbf{x}_1,\ldots,\mathbf{x}_{n-1})\right)$$

Attempts to compress the sequence into a STATE variable

If the buffer is *not* small *and* the sequential order *really* mattered, then a Transformer requires an *impossible* amount of data to learn a generalization for next token prediction.
The Transformer's Attention Mechanism incrementally transforms, using residual connections, the embedding space using a contextual 2-token correlation Bag-of-Words model.

Transformer: Stochastic Dynamical System

$$p(\text{next state} \mid \text{previous states}) \to p_\theta(x_t \mid x_{t-1}, \ldots, x_{t-\tau})$$

$$\to -\log p_\theta(x_t \mid x_{t-1}, \ldots, x_{t-\tau})$$

$\tau =$ buffer size

$$\text{Training data} = \{x_1, \ldots, x_T\}$$

Minimize the Entropy

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \log p_\theta(\underbrace{x_t}_{\text{output state}} \mid \underbrace{x_{t-1}, \ldots, x_{t-\tau}}_{\text{input trajectory}})$$

$$\approx -\sum_{t=1}^{T} \sum_{v \in \text{tokens}} \underbrace{\mathbf{y}_t(v)}_{\text{ground truth}} \log \underbrace{\hat{\mathbf{y}}_t(v)}_{\text{prediction}}$$

$$\left(N_{\text{tokens}}\right)^{\text{buffer size}}$$

Size of dictionary: $N_{\text{tokens}} = 10^5$ tokens

Size of buffer: $\tau = 10^4$ tokens

Size of training data: $T = 10^{13}$ tokens

Cross-entropy
(Self-Supervised Learning)

Missing data (ground truth): $\mathbf{y}_t(v) = \delta_{v, v_{true}(x_t)}$

Predictive probability: $\hat{\mathbf{y}}_t(v) \approx p_\theta(x_t \mid x_{t-1}, \ldots, x_1)$

Sequential Models: Parameters & computational complexity

DEFINITIONS

K = # of Observed States (size of vocabulary)
N = # of previous states (size of buffer)
D = # of embedding dimensions

D_{in} = input dimension of RNN
d_H = dimension of hidden layer of the NN
D_{out} = output dimension of RNN

Model	# of parameters	Computational complexity (inference)	Key operations
Independent, $\text{AR}(0)$ *Bag-of-Words*	$K \times D$	$O\left(N \times D\right)$	Embedding Look-ups
Auto-regressive, $\text{AR}(N)$ N-gram	K^N	$O(K^N)$	Look-Up Table ops on conditionals
Recurrent Neural Network (RNN)	$D_{\text{in}} d_{\text{H}} + d_{\text{H}}^2 + d_{\text{H}} D_{\text{out}}$	$O(N D_{\text{in}} d_{\text{H}} + N d_{\text{H}}^2)$	Sequential Hidden state updates
$D_{in} \approx D_{out} \approx D$	$2Dd_{\text{H}} + d_{\text{H}}^2$	$O(NDd_{\text{H}} + Nd_{\text{H}}^2)$	***Poor parallelism!***
Transformer (Decoder) Large Language Model	$4LD^2 + 2LD \cdot d_H$	$O(N^2 D + NDd_H)$	Self-attention calculations and Feed-Forward NN ops
Typically, $d_{\text{H}} \sim 4D$	$12LD^2$	$O(\underline{N^2 D} + ND^2)$ **ATTN** **FFNN**	***Great parallelism!***

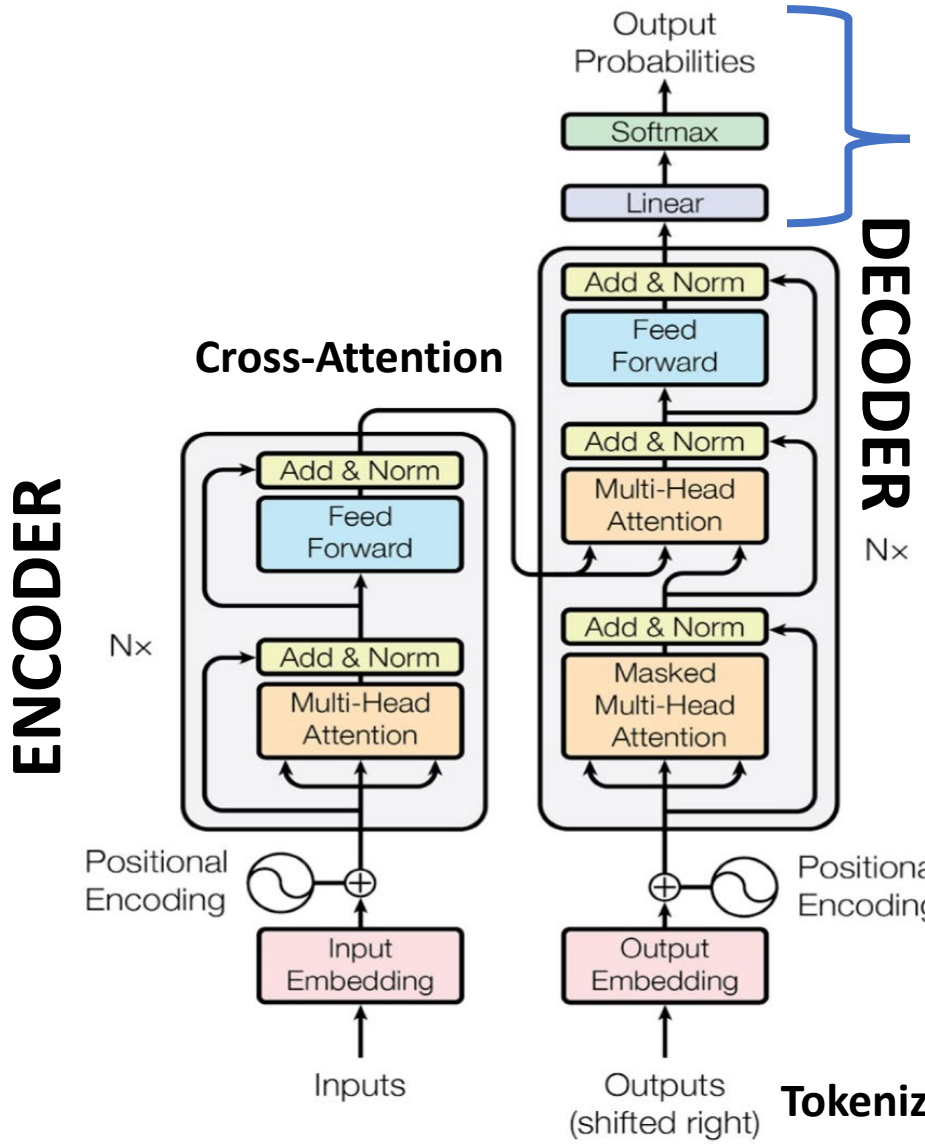
Size of Transformer Models

Embedding dimension

# of hidden units in Feed-Forward Neural Net (FFNN)

Model	D x 4 →	d_H
Transformer Base	512	2048
Transformer Large	1024	4096
BERT Base	768	3072
BERT Large	1024	4096
GPT-1 Base	768	3072
GPT-2/3 Varied	Varied	Varied

“Attention is All You Need” … and few other tricks

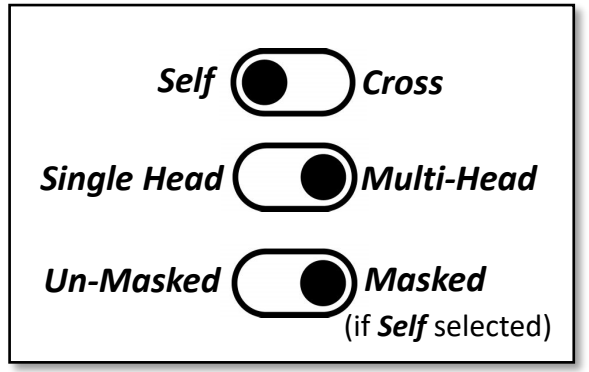


Inverse Embedding back to Tokens, “W^{-1}”

Add & Normalization

Feed-Forward Neural Network

Attention →



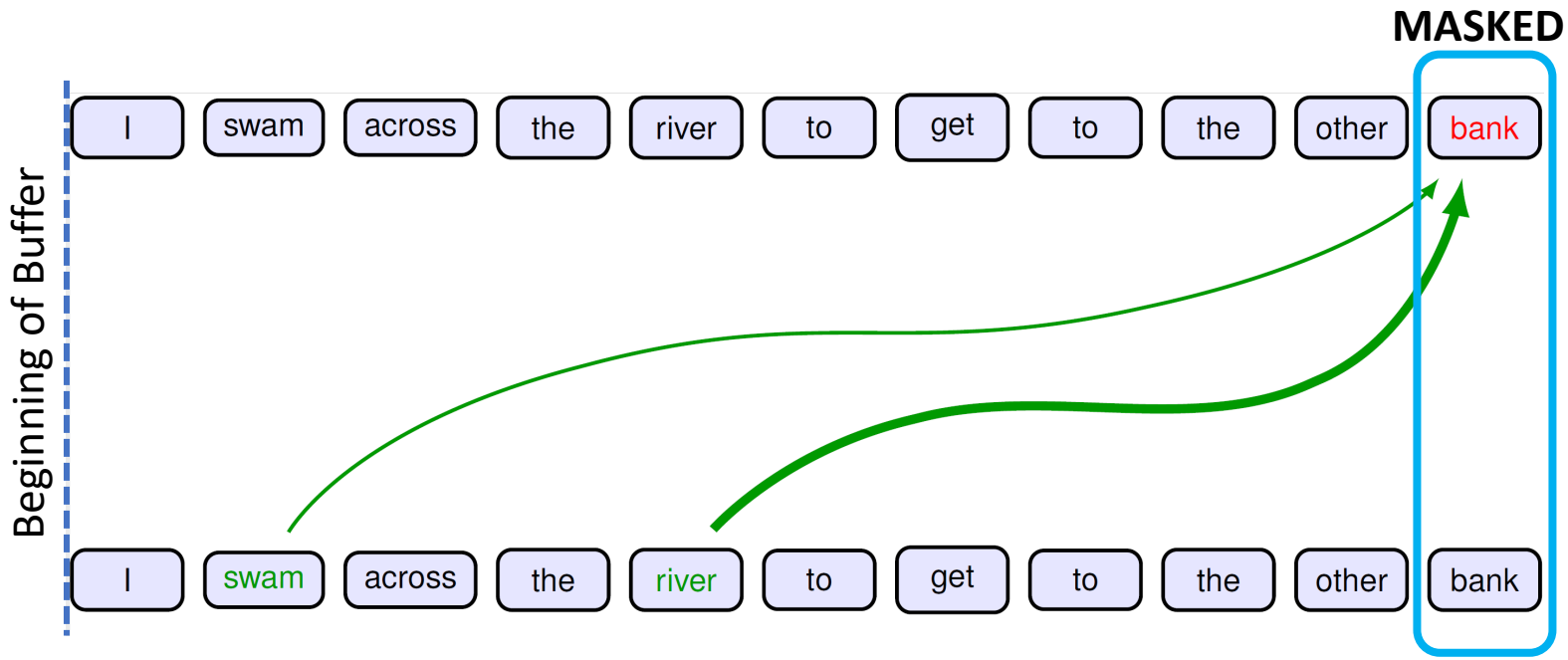
Positional Encoding

Token Embedding, W

Tokenized Sequence

Why Transformers? An interacting batch …

Unlike a *typical* neural network, analyzing sequential data requires more than just batch processing on the buffer as if these were individual pieces of data.



What is critical is how to pay attention to the correct parts of the sequence to enable a correct prediction of the next output.

Attention: Transforming the Embedding Geometry

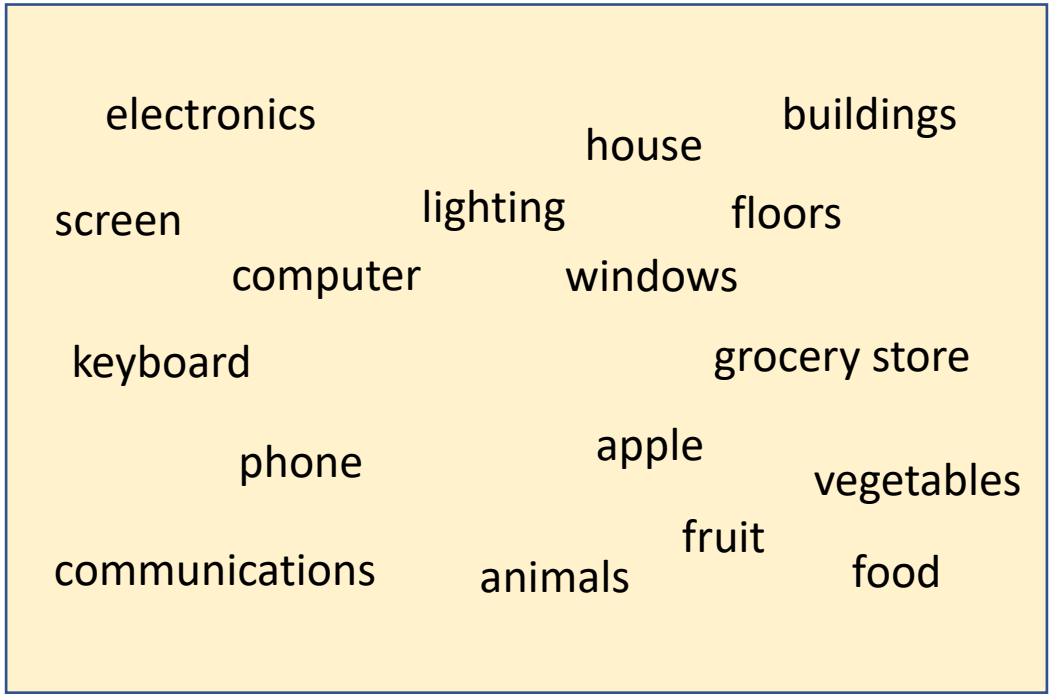
“I phoned about the apples in the window.”

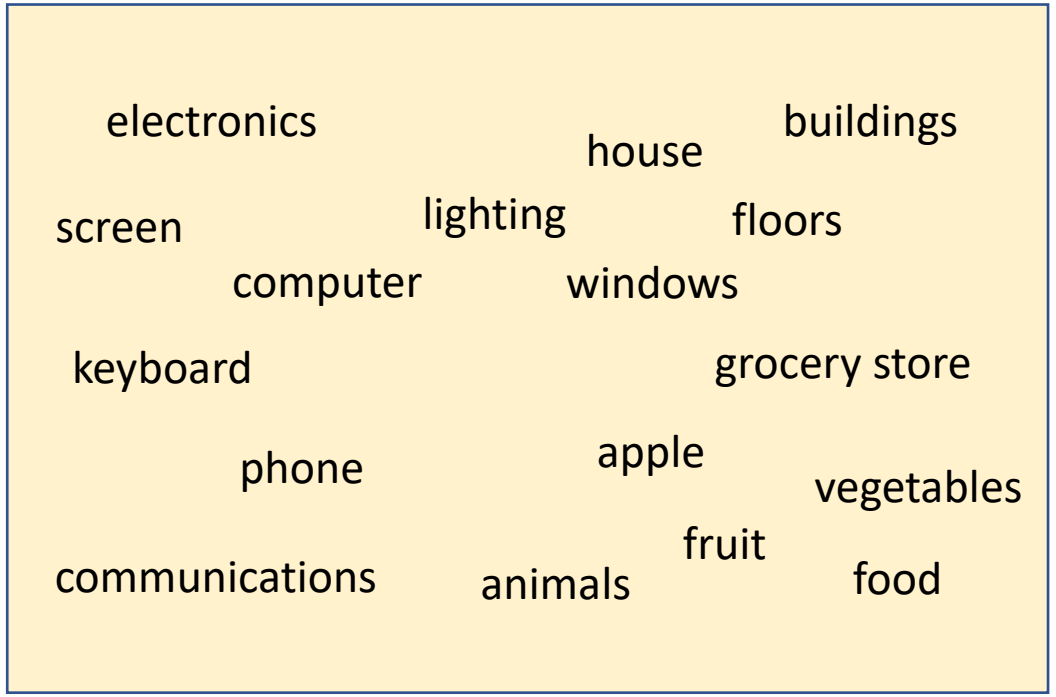
“The Apple phone can work with Windows.”

Attention: Transforming the Embedding Geometry

"I phoned about the apples in the window."

"The Apple phone can work with Windows."



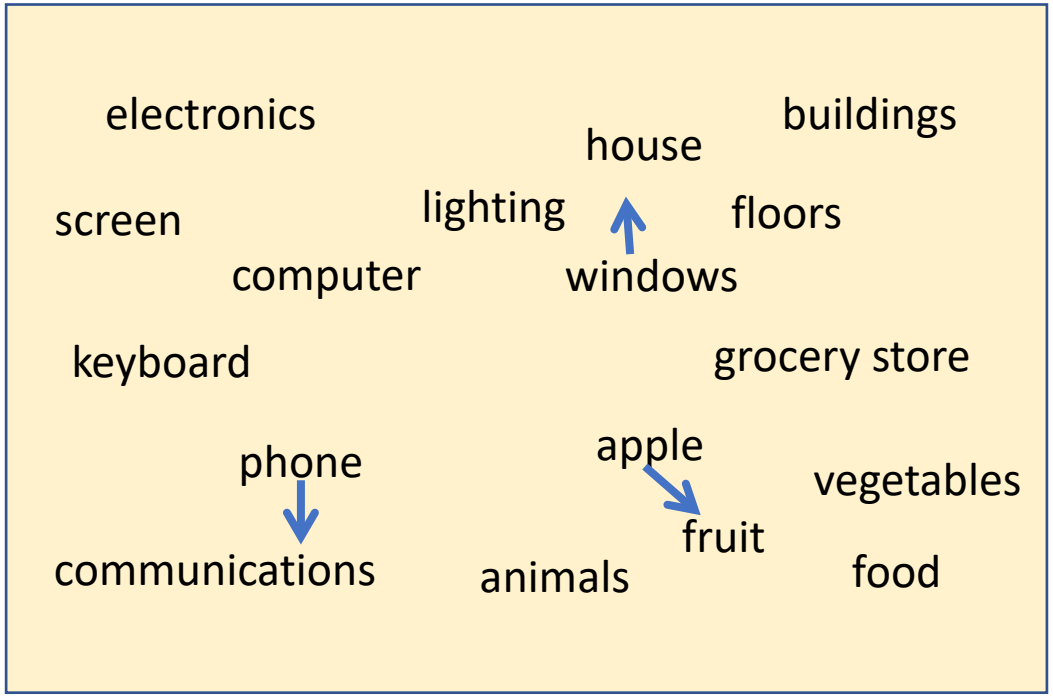


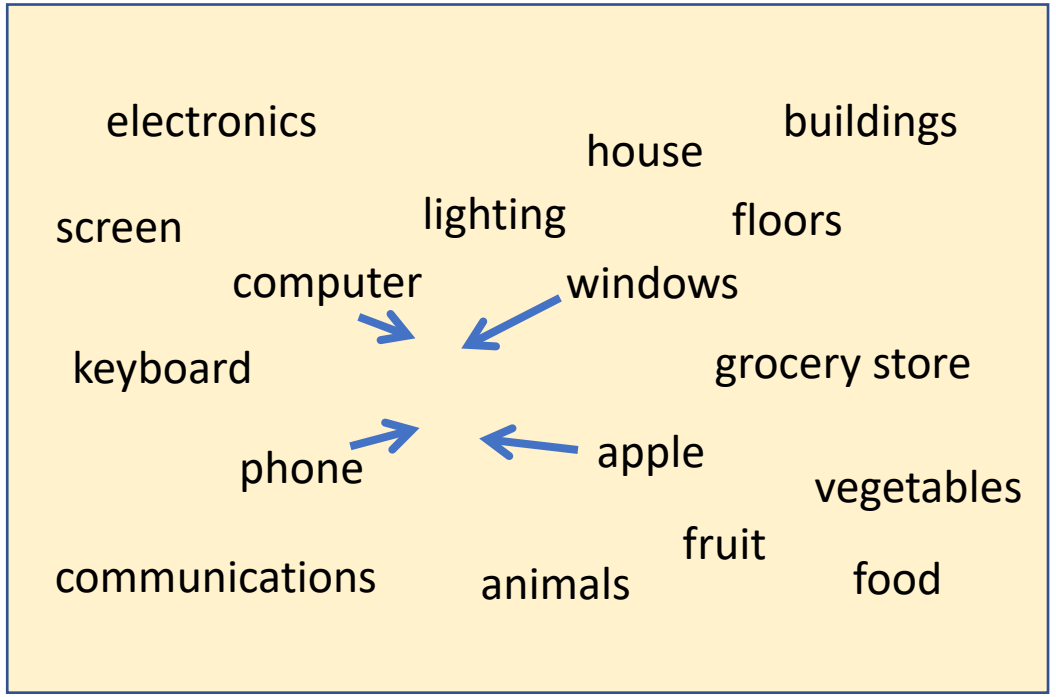
Initial Embedding of the Tokens

Attention: Transforming the Embedding Geometry

"I phoned about the apples in the window."

"The Apple phone can work with Windows."





Transformed Embedding of the Tokens

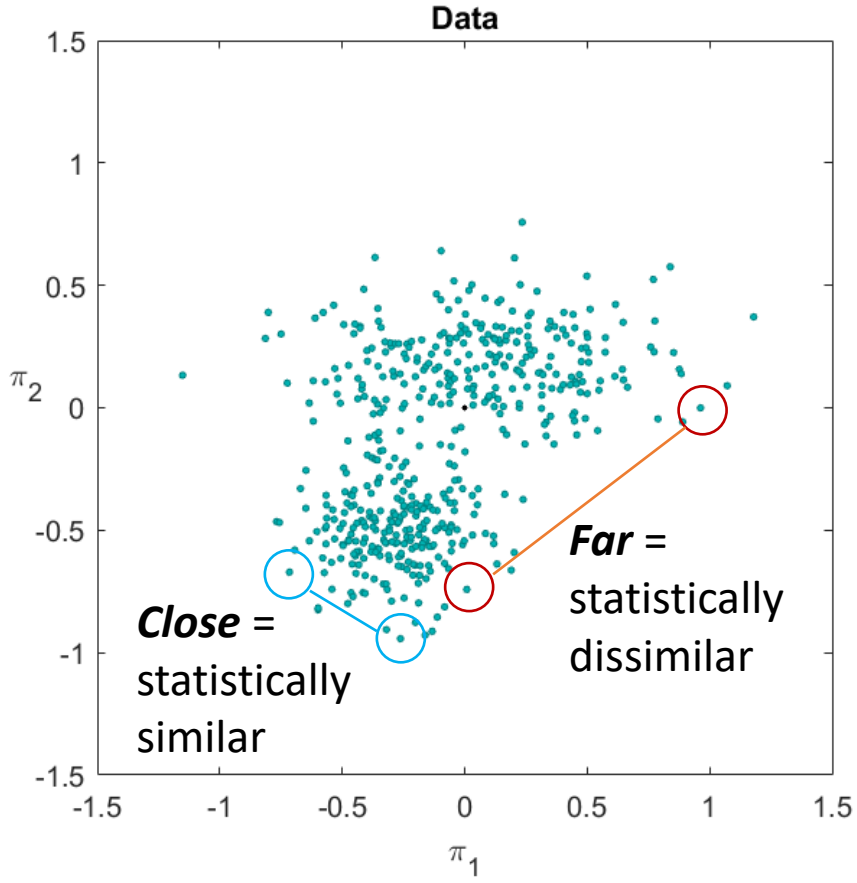
ML: Probability of Similarity becomes Distances

Create/Learn an Embedding or Feature Vector space that encodes information about *statistical similarity as distances*.

$$-\log p_{\text{similarity}}\left(y, y'\right) \approx \frac{1}{2}\left\|\mathbf{y}-\mathbf{y}'\right\|_2^2$$

How much DATA is enough?

If distant data points are mapped nearby in the model space, then there has to be significant inferential evidence (more data), and likewise if local data points are mapped to large separations by the model.



Attention: Query, Keys, Values …

The *Masked Self-Attention* mechanism is how a database engineer would build an *Auto-Regressive* signal processing algorithm!

Database Engineering Perspective:

A *query* is used to retrieve information (*values*) based on features (*keys*). This relies on efficient indexing and retrieval mechanisms to access relevant information quickly and accurately.

Signal Processing Perspective:

Traditional *autoregressive* (AR) models in signal processing predict future signals based on a weighted set of past observations.

Machine Learning/Attention Mechanism:

- The attention mechanism uses *queries*, *keys*, and *values* to dynamically weigh and retrieve information from different parts of the input sequence to generate each part of the output sequence.
- This process resembles database retrieval (*queries* and *keys*) but is applied to the sequential prediction task typical of signal processing.

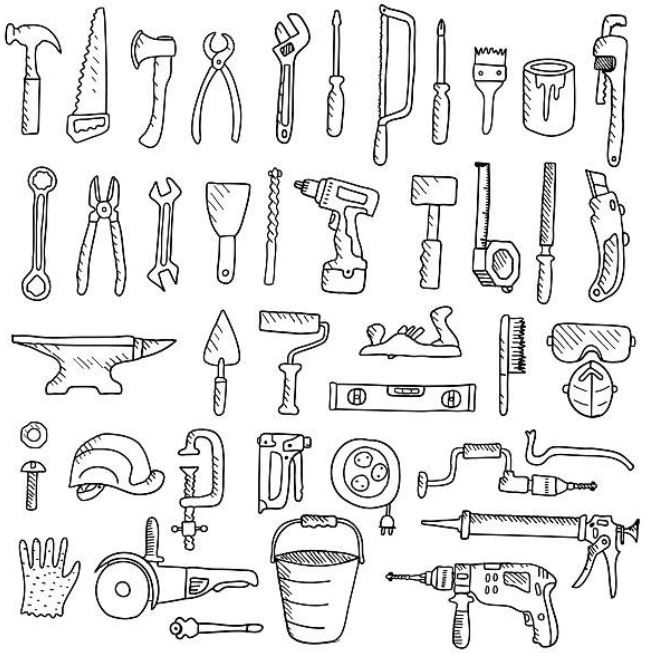
Example of Query, Keys, & Values : Using Tools

QUERY: What tools should I use to fix a loose board?

QUERY and VALUES meet among the KEYS to discover the relevance of particular VALUES to the QUERY.

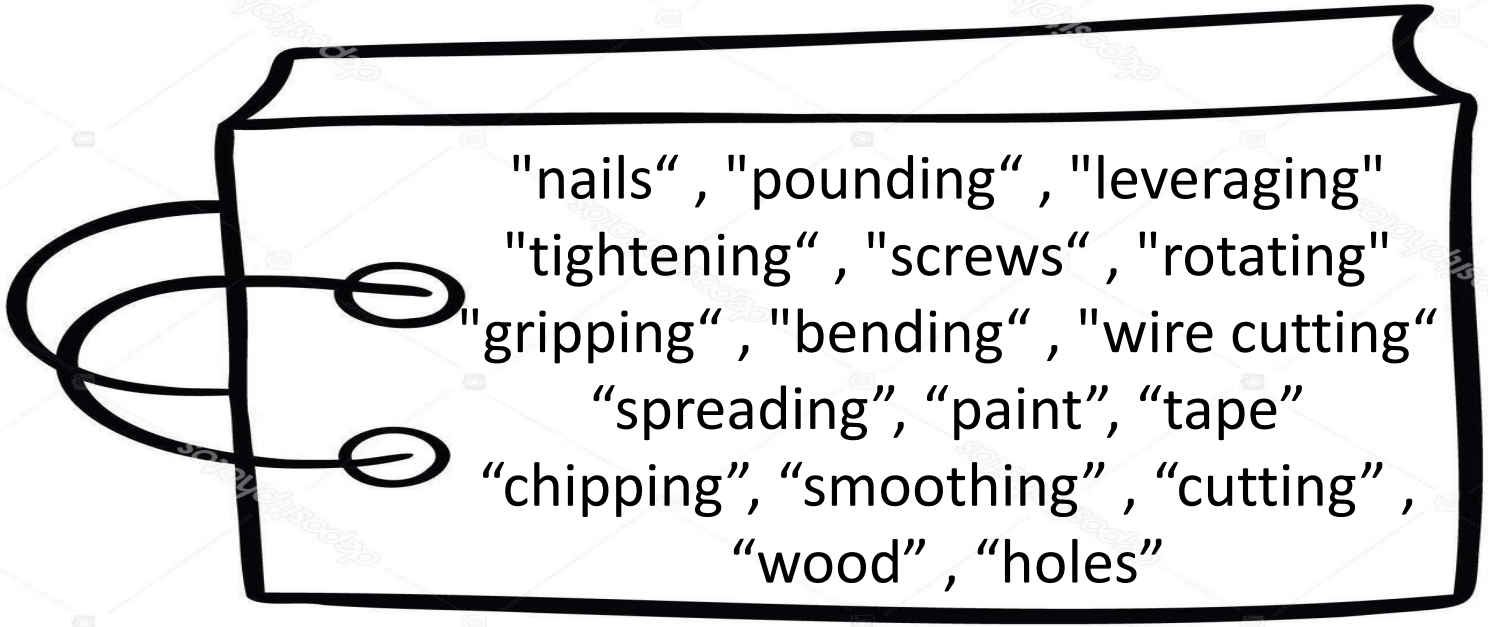
VALUES

Objects or Processes



KEYS

Properties of Objects or Processes



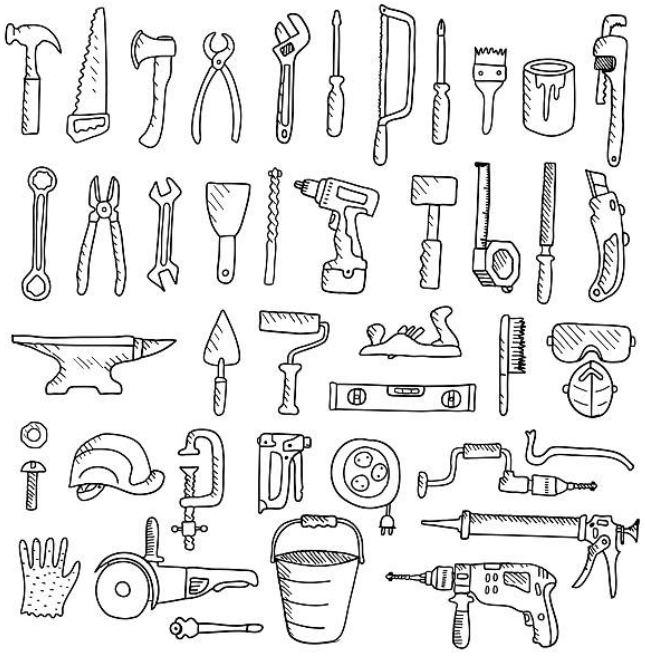
Example of Query, Keys, & Values : Using Tools

QUERY: What tools should I use to fix a loose board?

$$\mathbf{QK}^T = \{\ \text{"pounding"}\ ,\ \text{"leveling"}\ ,\ \text{"wood"}\ \}$$

VALUES

Objects or Processes



KEYS

Properties of Objects or Processes

Hammer:	"nails“ , "pounding“ , "leveraging"
Screwdriver:	"tightening“ , "screws“ , "rotating"
Pliers:	"gripping“ , "bending“ , "wire cutting“
Paintbrush:	“spreading”, “paint”, “tape”
Chisel:	“chipping”, “smoothing” , “wood”
Saw:	“cutting” , “wood” , …
Drill:	“holes” , …

Attention Mechanism and Query, Key & Value

The purpose of the *Attention* mechanism is to connect a **Query** to relevant **Values** by using the structural organization imposed by the **Keys**.

- The **Keys** are a basis onto which a **Query** is linearly projected.
- **Values** are the data instances organized by the **Keys**.

Query **Key** **Value**

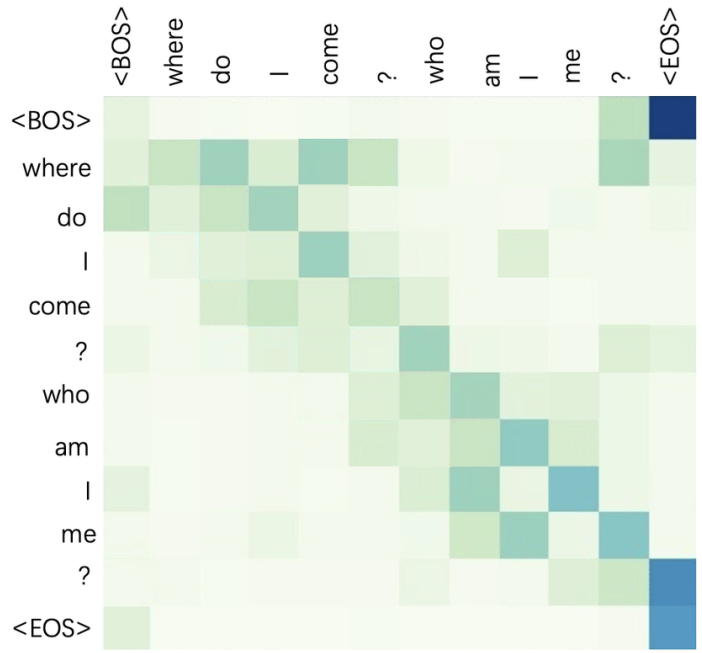
$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V$$

$N \times N$
NOT Symmetric

$$\mathbf{Q}\mathbf{K}^T \xrightarrow{\textit{Softmax}\ \text{normalization}} Z^{-1} \cdot \exp\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D}}\right)$$

Self-Attention is like a Batch interacting with itself!

Attention



Attention Mechanism and Query, Key & Value

The purpose of the *Attention* mechanism is to connect a **Query** to relevant **Values** by using the structural organization imposed by the **Keys**.

- The **Keys** are a basis onto which a **Query** is linearly projected.
- **Values** are the data instances organized by the **Keys**.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$$

$$\mathbf{K} = \mathbf{X}\mathbf{W}_K$$

$$\mathbf{V} = \mathbf{X}\mathbf{W}_V$$

$\mathbf{A}$, *Attention Score* matrix

$$\underset{N_{seq1} \times d_V}{\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})} = Z^{-1} \cdot \exp\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_K}}\right) \cdot \mathbf{V}$$

Softmax normalization along the rows of $\mathbf{A}$ matrix

$$\left.\begin{aligned} \dim \mathbf{Q} &= N_{seq1} \times d_K \\ \dim \mathbf{K} &= N_{seq2} \times d_K \end{aligned}\right\} \rightarrow \dim \mathbf{A} = N_{seq1} \times N_{seq2}$$

$$\dim \mathbf{V} = N_{seq2} \times d_V$$

$$d_K = \frac{D, \text{ embedding dim}}{n_H, \ \#\text{ of heads}}$$

Causal (Masked) Self-Attention

This is this the OTHER step in the Transformer where Position in Sequence is used.

Single-Head Attention: $\dim \mathbf{Q} = N \times D$

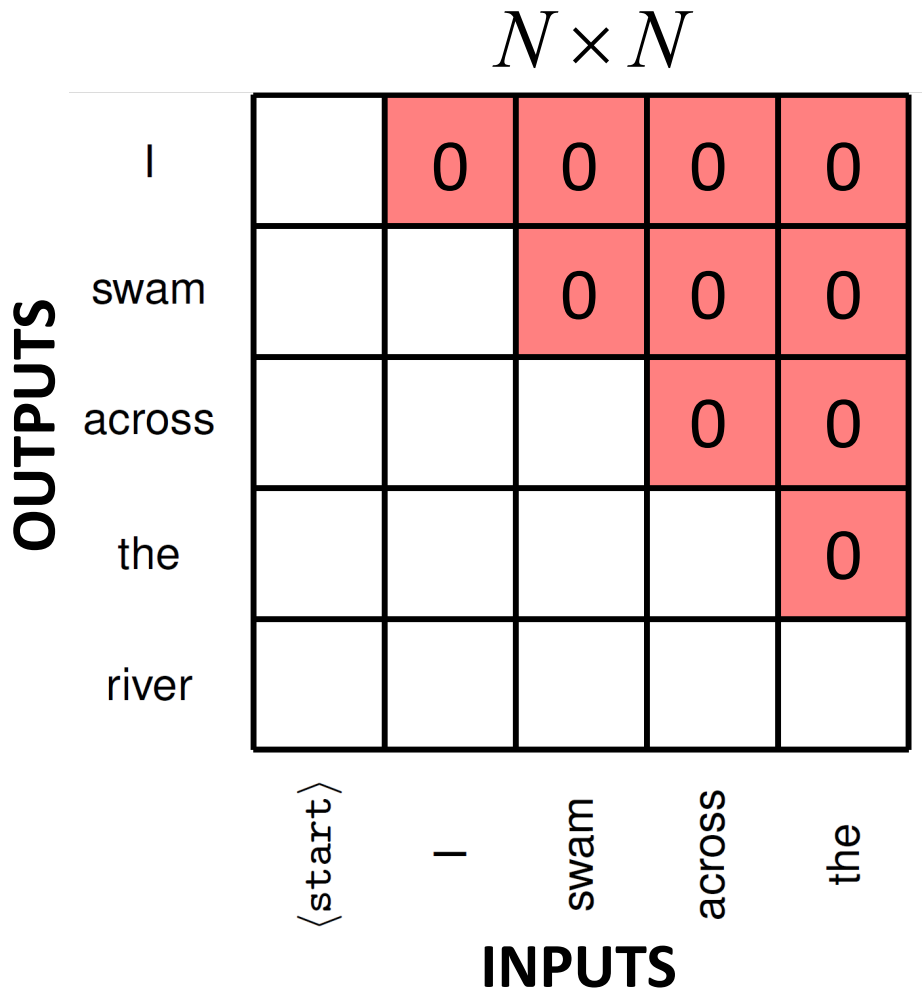
$\dim \mathbf{K} = N \times D$

Attention Score matrix:

$$\mathbf{A}(\mathbf{Q},\mathbf{K}) = Z^{-1} \cdot \exp\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D}}\right) =$$

Compute the Attention:

$$\underbrace{\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V})}_{N\times D} = \underbrace{\mathbf{A}(\mathbf{Q},\mathbf{K})}_{N\times N} \cdot \underbrace{\mathbf{V}}_{N\times D}$$



Multi-Head Attention: Dividing up the Embedding space by dimension

$d_h = D / \#\ \text{of heads}$

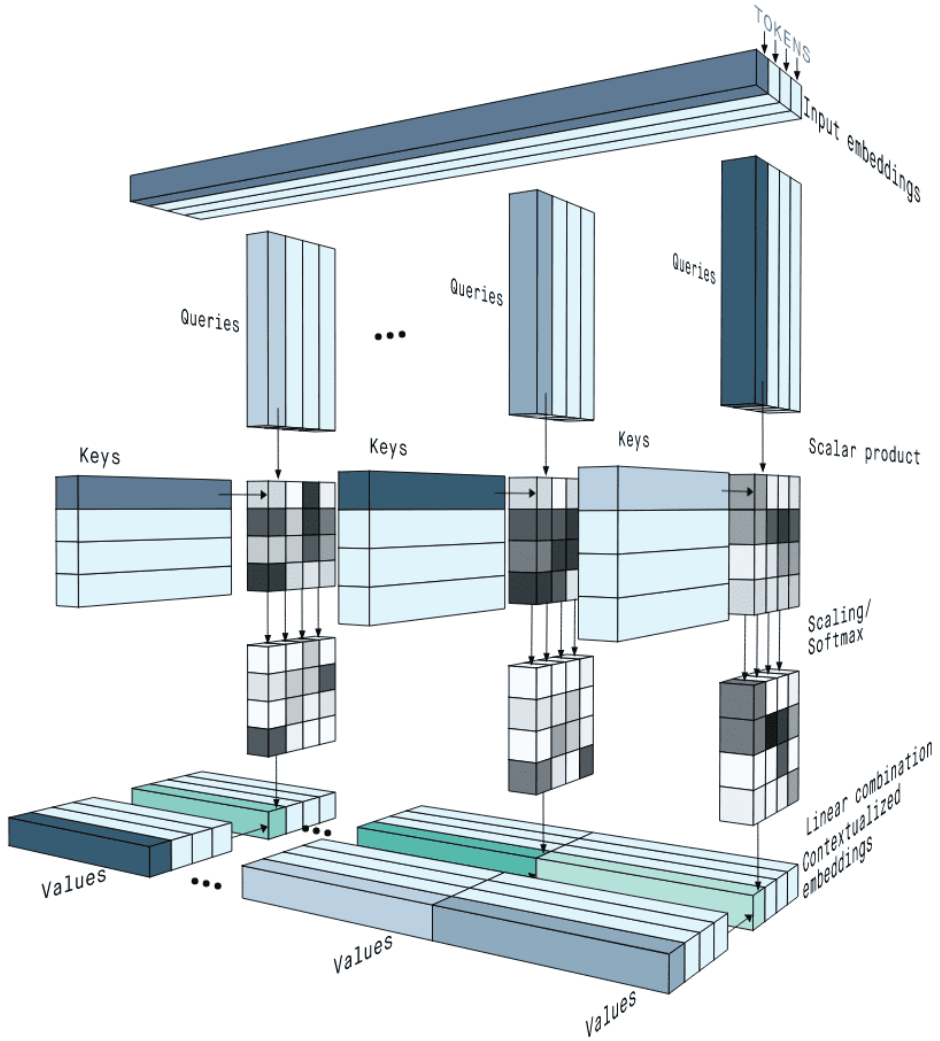
$N \times D$

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_N & - \end{pmatrix}$$

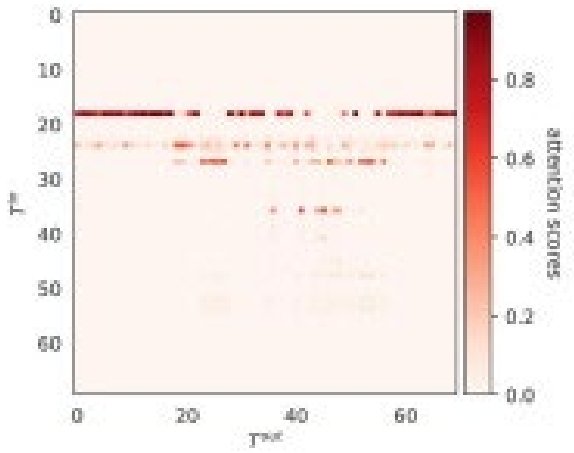
$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_{1,1:d_h} & - \\ & \vdots & \\ - & \mathbf{x}_{N,1:d_h} & - \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_{1,d_h:(2d_h-1)} & - \\ & \vdots & \\ - & \mathbf{x}_{N,d_h:(2d_h-1)} & - \end{pmatrix}$$

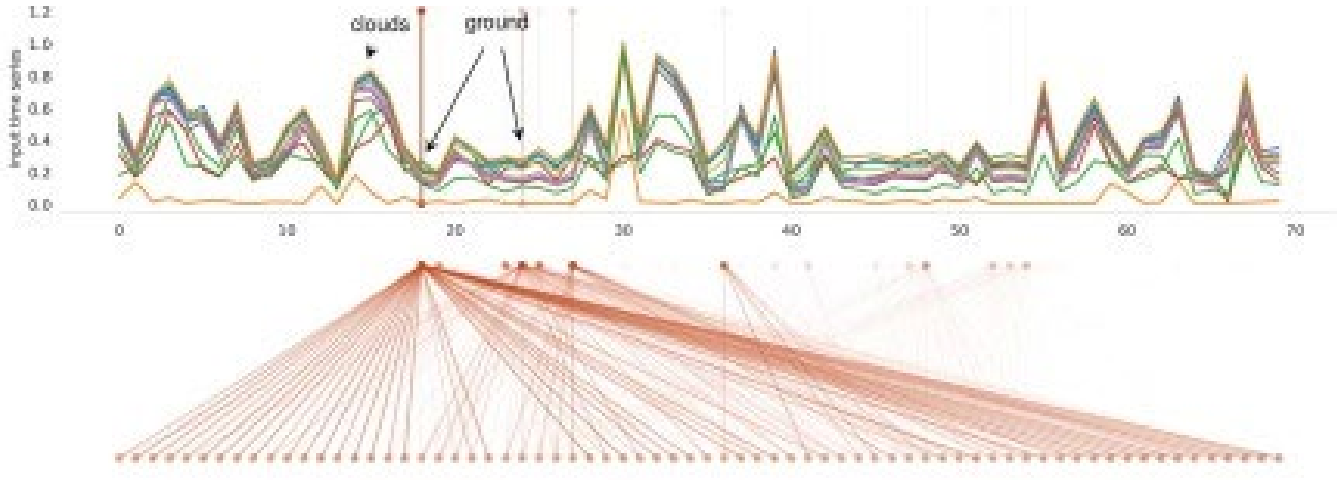
$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_{1,*:D} & - \\ & \vdots & \\ - & \mathbf{x}_{N,*:D} & - \end{pmatrix}$$



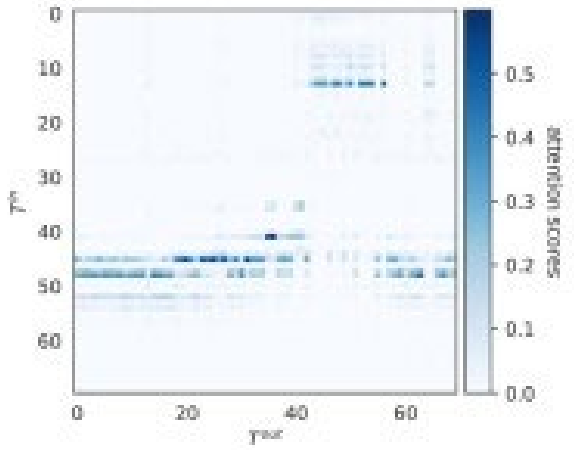
What does Multi-Head Attention do?



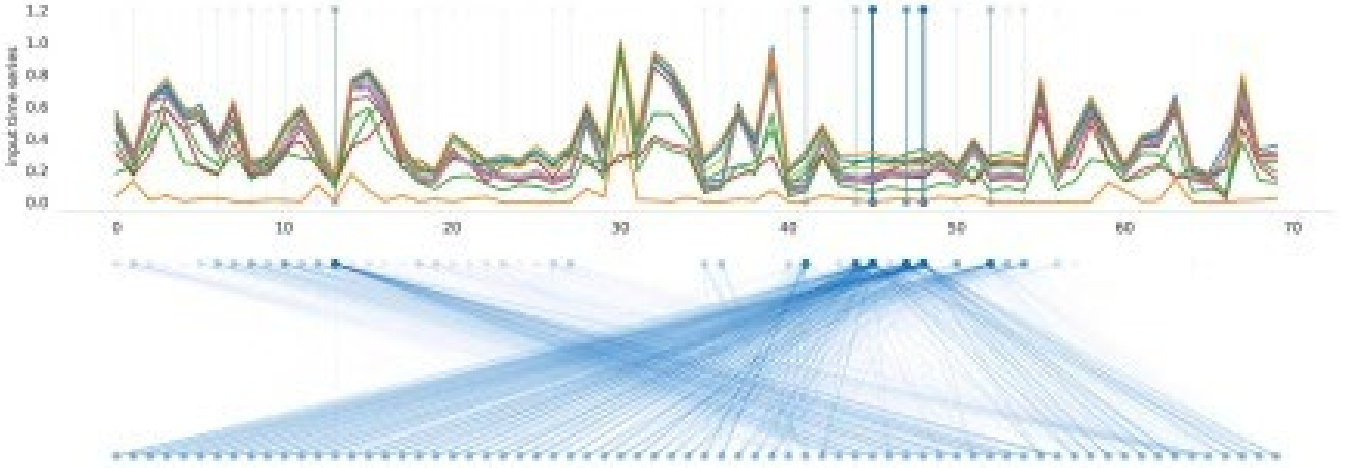
(a) Self-Attention Matrix of Head 1



(b) Head 1 as bipartite graph visualization in context of the input time series

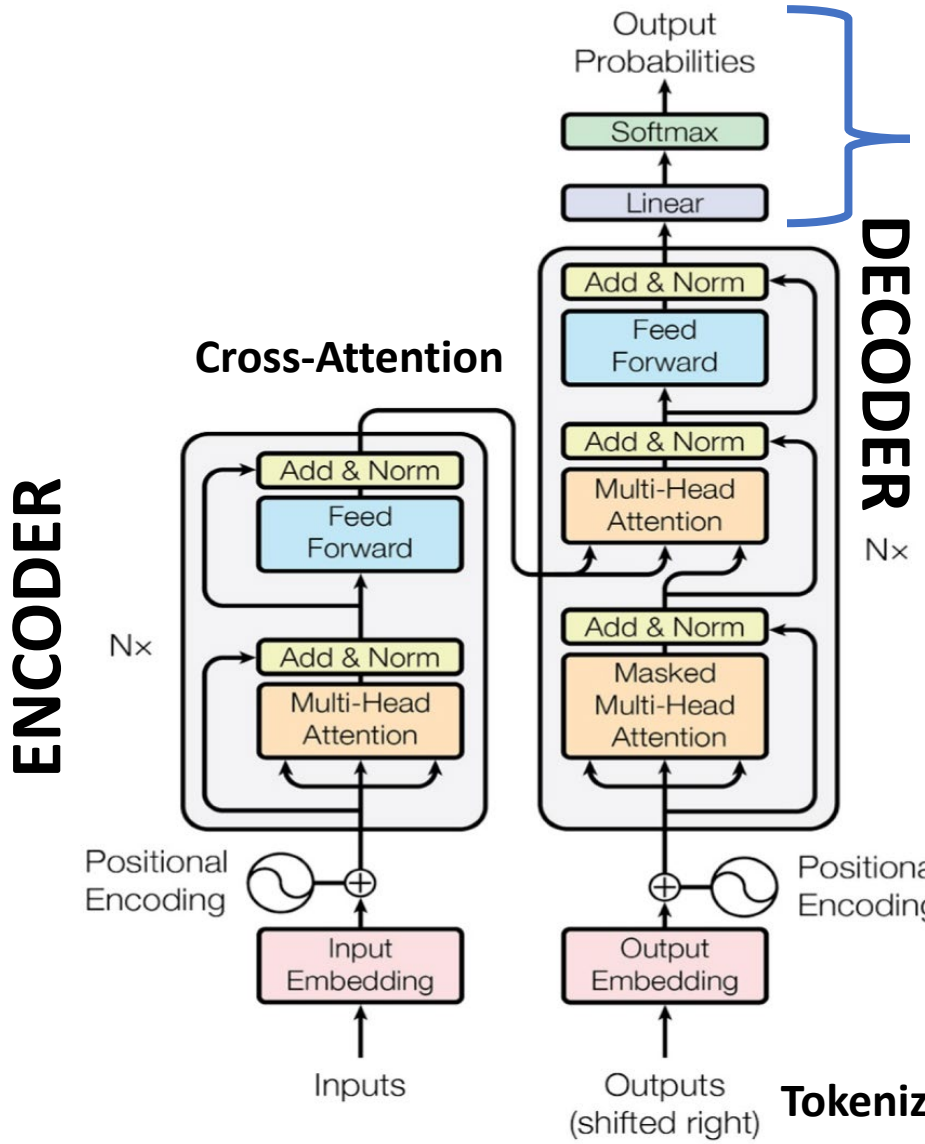


(c) Self-Attention Matrix of Head 2



(d) Head 2 as bipartite graph visualization in context of the input time series

“Attention is All You Need” … and few other tricks

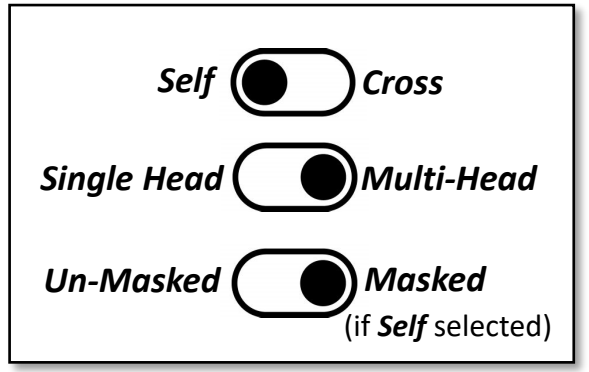


Inverse Embedding back to Tokens, “W^{-1}”

Add & Normalization

Feed-Forward Neural Network

Attention →



Positional Encoding

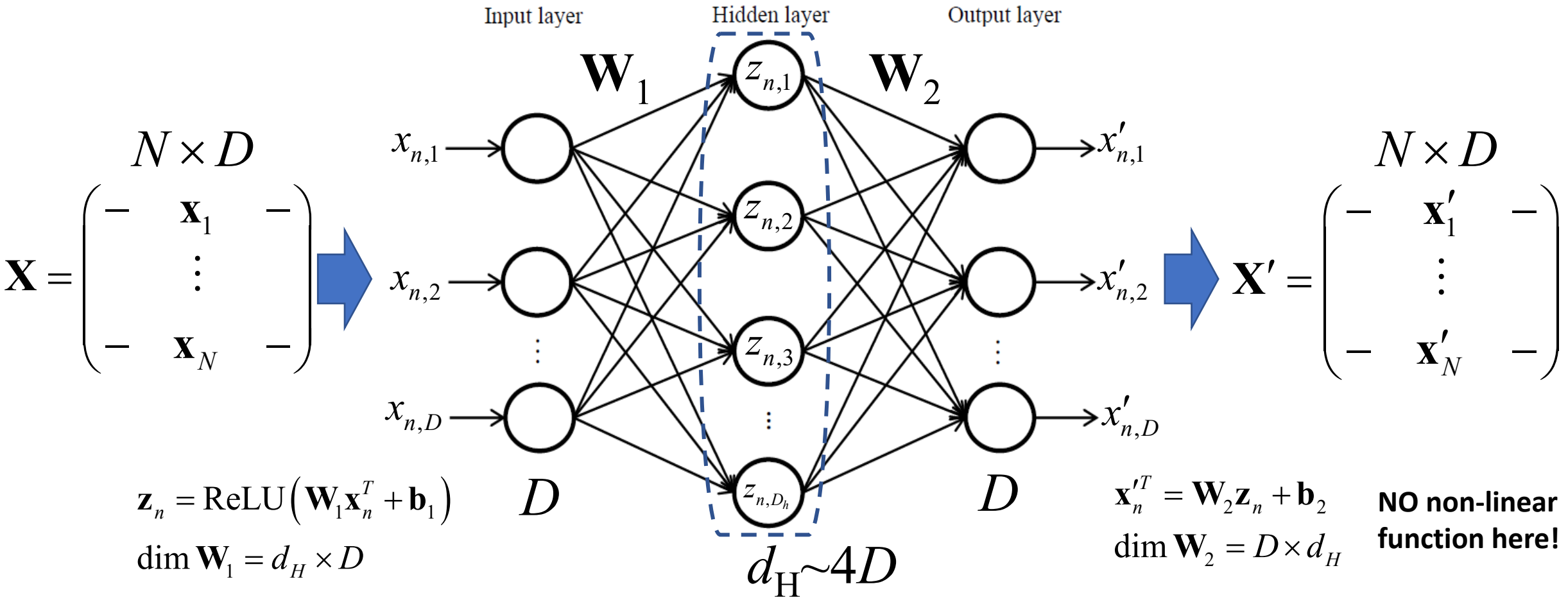
Token Embedding, W

Tokenized Sequence

Two-Layer Fully Connected Feed Forward NN

The *FFNN* acts independently on each vector, $\mathbf{x}_n$, of the buffer, $\mathbf{X}$, as if it's a batch of size N!

Learnable parameters: $(D_{\mathrm{h}}+1)\times D + D_{\mathrm{h}}\times(D+1) \approx 8D^2$



$$\mathbf{z}_n = \mathrm{ReLU}\left(\mathbf{W}_1\mathbf{x}_n^T + \mathbf{b}_1\right)$$

$$\dim \mathbf{W}_1 = d_H \times D$$

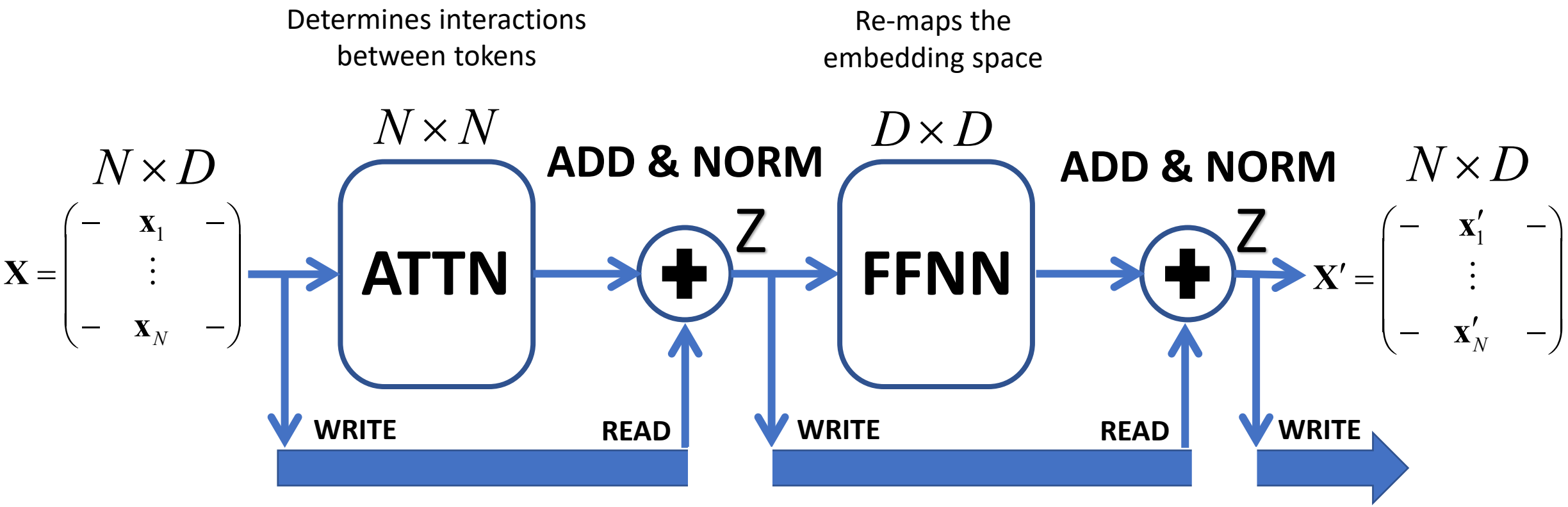
$$\mathbf{x}_n'^T = \mathbf{W}_2\mathbf{z}_n + \mathbf{b}_2$$

$$\dim \mathbf{W}_2 = D \times d_H$$

NO non-linear function here!

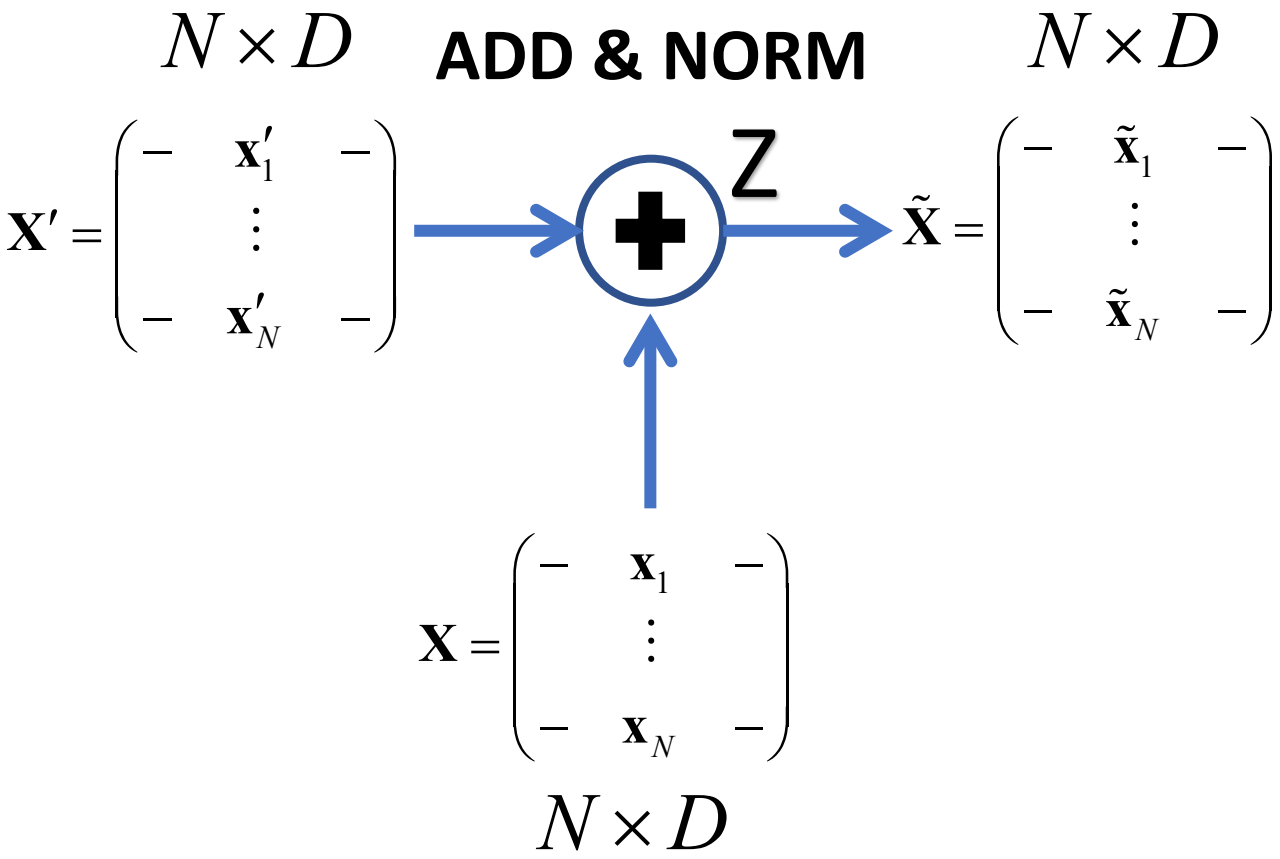
Residual Connections and Layer Normalization

We tend to think of *residual connections* as "going around sub-modules" in the stack. Instead, consider the sub-modules as performing "read/write" to a data stream.



Residual Connections

Residual Connections and Layer Normalization



1 Token vector (row) normalization.

$$m_n = D^{-1} \cdot \sum_{d=1}^{D} (x_{nd} + x'_{nd})$$

$$s_n^2 = D^{-1} \cdot \sum_{d=1}^{D} (x_{nd} + x'_{nd} - m_n)^2$$

$$\tilde{\mathbf{x}}_n^* = \frac{\mathbf{x}_n + \mathbf{x}'_n - m_n \cdot \mathbf{I}_{1\times D}}{\sqrt{s_n^2 + \varepsilon}}$$

---

2 Feature (column) normalization.

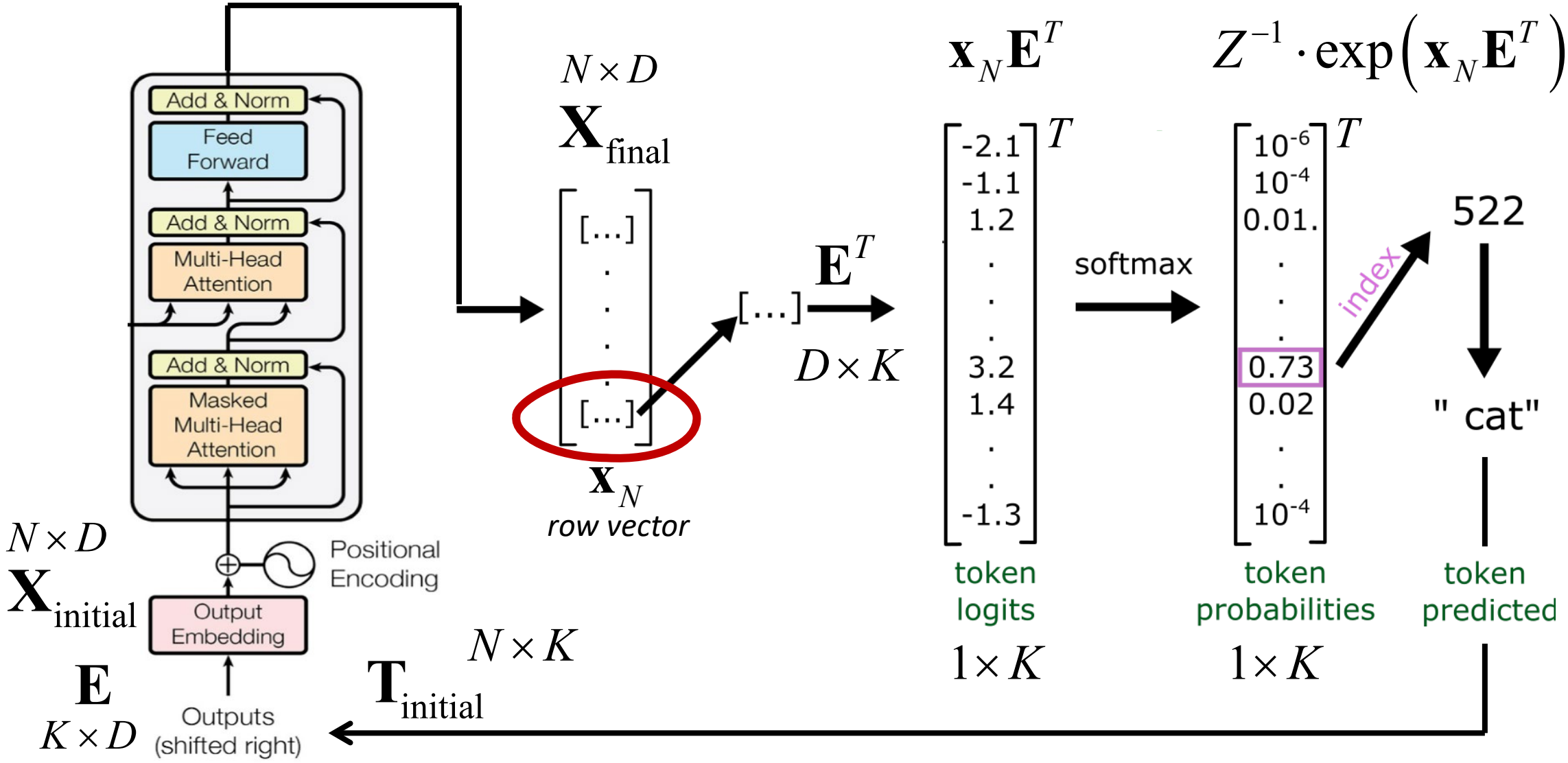
$$\tilde{x}_{nd} = \gamma_d \tilde{x}^*_{nd} + \beta_d$$

$$\tilde{\mathbf{x}}_n = \boldsymbol{\gamma} \odot \tilde{\mathbf{x}}^*_n + \boldsymbol{\beta} \text{ where } \boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{1\times D}$$

$2D$ learnable parameters per norm

NOTE: In general, transformers do not use Batch Normalization.

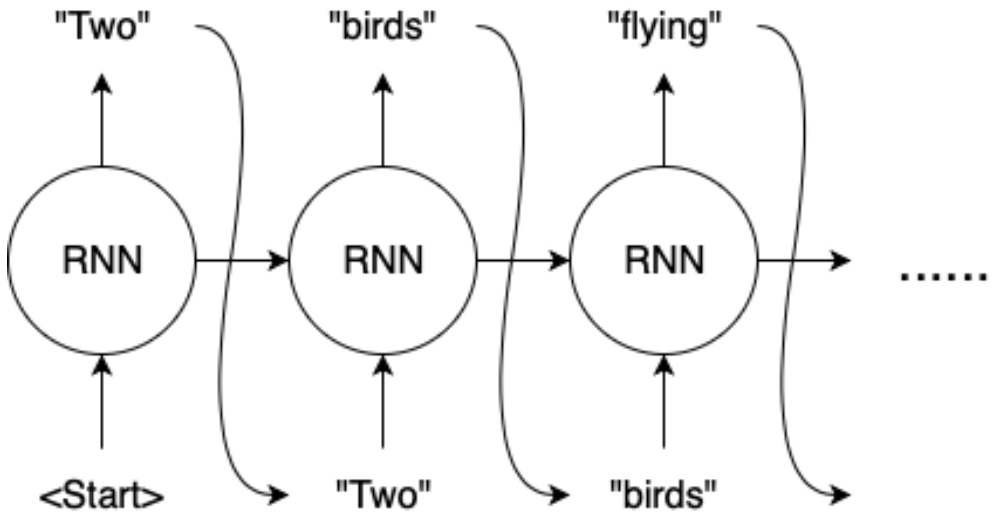
Context Aggregation to the Last Token in the Buffer



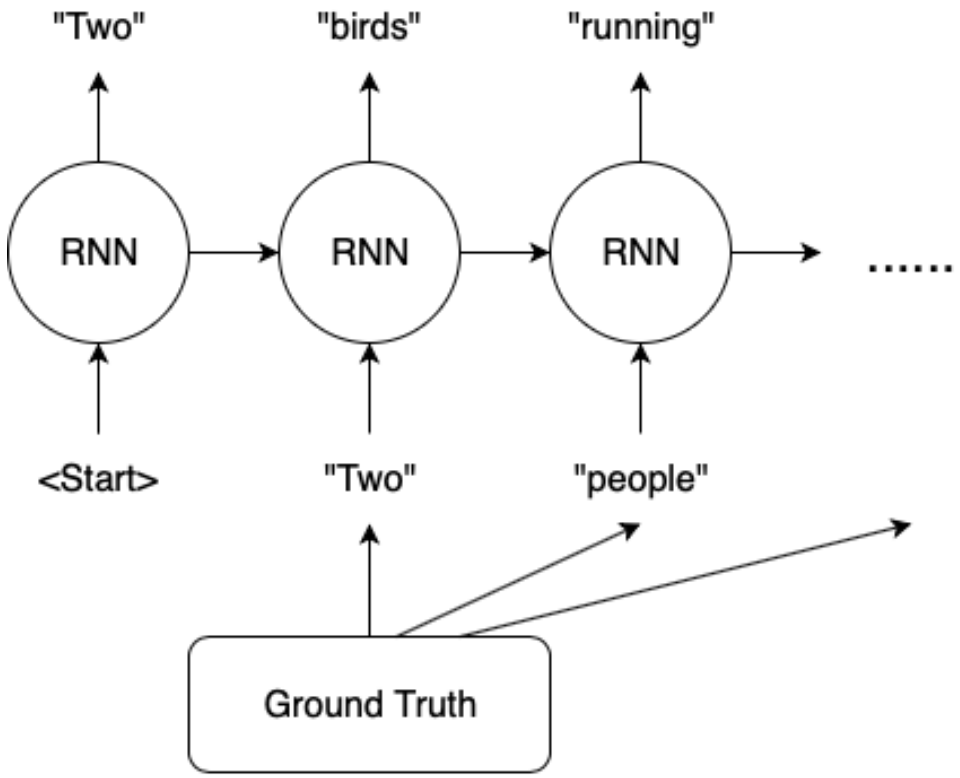
Context Aggregation to the Last Token in the Buffer

Teacher Forcing inserts correct answer at each step for sequence continuation

Without Teacher Forcing



With Teacher Forcing



Why Transformers?

1. Transformers can respond to contextual information in sequential data more robustly via the Attention mechanism.

2. Transformers use an initial vector space embedding like other ML algorithms but treat that embedding with an *algorithmic skepticism* by always returning to the original data space to restart computations for predicting the next step in the sequence.