# What is a Language Model?

- **Definition**: A language model is a machine learning model that predicts the likelihood of a sequence of words.
- **Primary Task**: Given some input text, the model **predicts the next word** in the sequence.
- **Simple Example**: "The cat sat on the _____."

# How Language Models Work

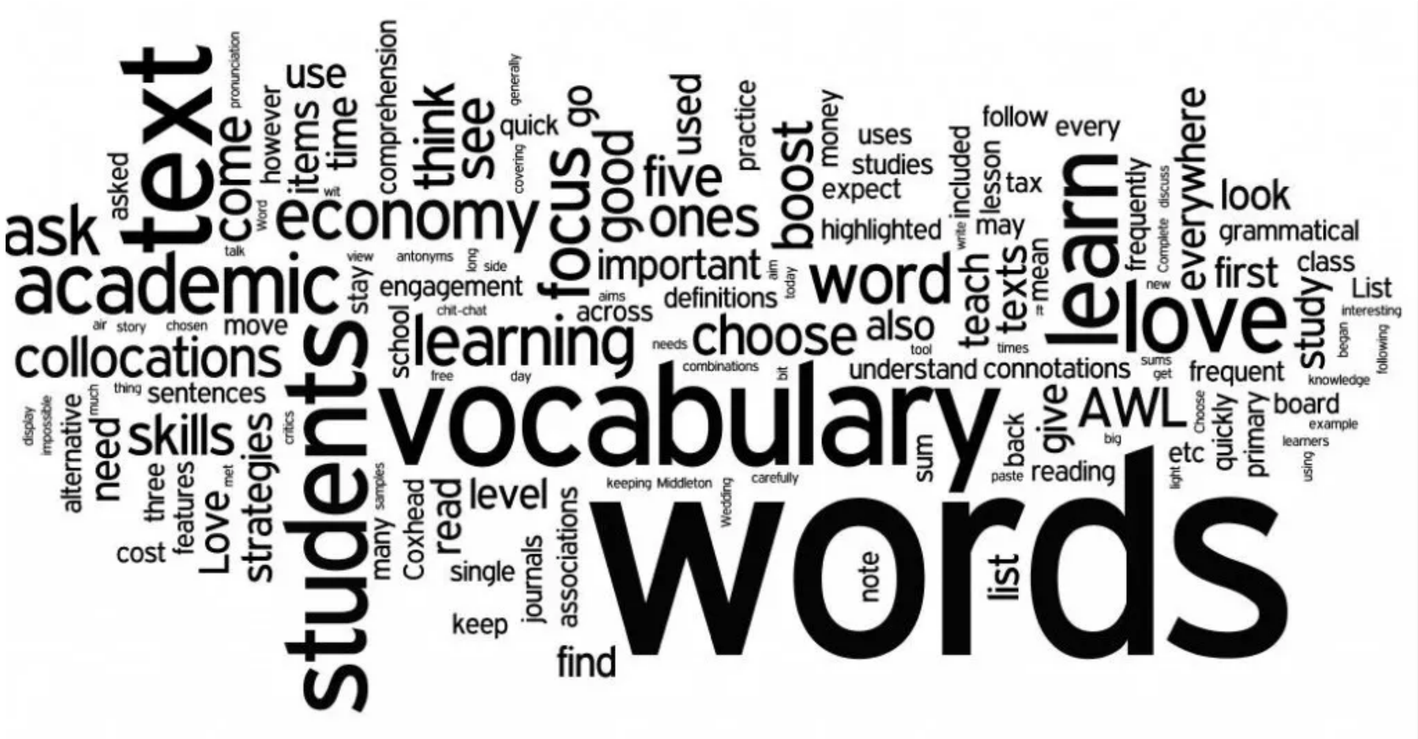
- **Input**: A sequence of words or tokens.
- **Output**: A probability distribution over the possible next words.
- Example: "The cat sat on the [mat, chair, floor]" with probabilities.

# Training a Language Model

- **Data**: Large text corpora are used to train models.
- **Objective**: Minimize the difference between predicted and actual words.
- **Loss Function**: Cross-entropy loss measures prediction accuracy.
- Example: "The cat sat on the [mat, chair, floor]" with probabilities.

# Tokenization

**Tokenization** is the process of splitting a large chunk of text into smaller, manageable pieces called tokens.



# Tokenization

- **Word-level**: Splits text at the word boundary. Simple but can lead to issues like out-of-vocabulary (OOV) words.
- **Subword-level**: Breaks down words into smaller parts, used in Byte-Pair Encoding (BPE) and SentencePiece to manage OOV problems.
- **Character-level**: Splits text into individual characters. Less common but useful in specific cases.

# Word Character Subword tokenization!

How can we combine the high coverage of character-level representation with the efficiency of word-level representation?

**Subword tokenization! (e.g., Byte-Pair Encoding)**

- Start with character-level representations
- Build up representations from there

Original BPE Paper (Sennrich et al., 2016)

https://arxiv.org/abs/1508.07909

# Byte-pair encoding - Properties

- Efficient to run (greedy vs. global optimization)
- Lossless compression
- Potentially some shared representations - e.g., the token “hug” could be used both in “hug” and “hugging”

# Byte-pair encoding - Usage

- Basically state of the art in tokenization
- Used in all modern left-to-right large language models (LLMs), including ChatGPT

| Model/Tokenizer | Vocabulary Size |
|---|---|
| GPT-3.5/GPT-4/ChatGPT | 100k |
| GPT-2/GPT-3 | 50k |
| Llama2 | 32k |
| Falcon | 65k |

# Byte-pair encoding - ChatGPT Example

## Moby Dick as tokenized by ChatGPT

Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people’s hats off—then, I account it high time tozz get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

TEXT TOKEN IDS

Tokens 239

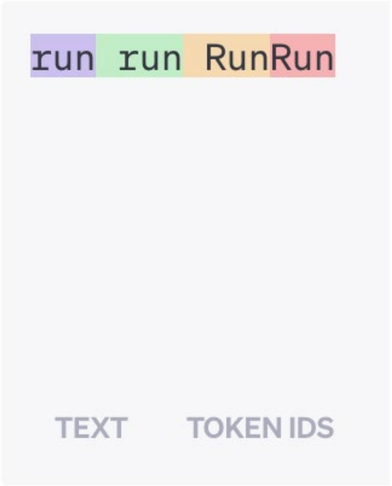
Characters 1109

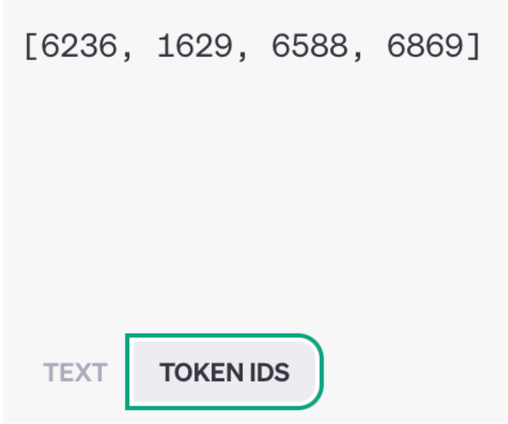
[7368, 757, 57704, 1764, 301, 13, 4427, 1667, 4227, 2345, 37593, 4059, 1268, 1317, 24559, 2345, 69666, 2697, 477, 912, 3300, 304, 856, 53101, 11, 323, 4400, 4040, 311, 2802, 757, 389, 31284, 11, 358, 3463, 358, 1053, 30503, 922, 264, 2697, 323, 1518, 279, 30125, 727, 961, 315, 279, 1917, 13, 1102, 374, 264, 1648, 358, 617, 315, 10043, 1022, 279, 87450, 268, 323, 58499, 279, 35855, 13, 43633, 358, 1505, 7182, 7982, 44517, 922, 279, 11013, 26, 15716, 433, 374, 264, 41369, 11, 1377, 73825, 6841, 304, 856, 13836, 26, 15716, 358, 1505, 7182, 4457, 3935, 6751, 7251, 985, 1603, 78766, 83273, 11, 323, 12967, 709, 279, 14981, 315, 1475, 32079, 358, 3449, 26, 323, 5423, 15716, 856, 6409, 981, 636, 1778, 459, 8582, 1450, 315, 757, 11, 430, 433, 7612, 264, 3831, 16033, 17966, 311, 5471, 757, 505, 36192, 36567, 1139, 279, 8761, 11, 323, 1749, 2740, 50244, 1274, 753, 45526, 1022, 2345, 3473, 11, 358, 2759, 433, 1579, 892, 311, 10616, 636, 311, 9581, 439, 5246, 439, 358, 649, 13, 1115, 374, 856, 28779, 369, 40536, 323, 5041, 13, 3161, 264, 41903, 67784, 356, 4428, 3872, 5678, 5304, 813, 20827, 26, 358, 30666, 1935, 311, 279, 8448, 13, 2684, 374, 4400, 15206, 304, 420, 13, 1442, 814, 719, 7020, 433, 11, 4661, 682, 3026, 304, 872, 8547, 11, 1063, 892, 477, 1023, 11, 87785, 1890, 16024, 7119, 279, 18435, 449, 757, 13]

TEXT TOKEN IDS

# Weird properties of tokenizers

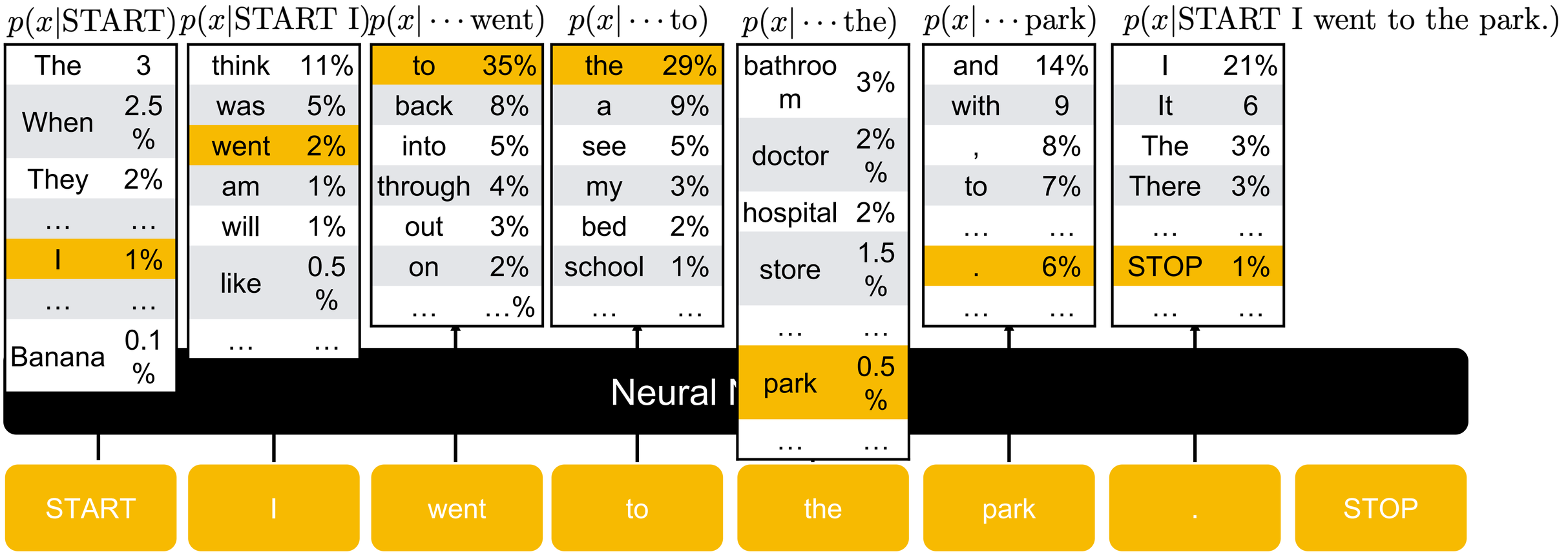
- Token != word
- Spaces are part of token
  - “run” is a different token than “ run”
- Not invariant to case changes
  - “Run” is a different token than “run”



# Inputs/Outputs

- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)



# One-hot encoding

- In order to feed in the tokens to a machine learning algorithm, we need to input them as standard **features**
- One approach: **One-hot encoding**
  - Recall from linear algebra:
  - One-hot encoding:

$$\text{Standard basis of } \mathbb{R}^n : e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ldots, e_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$\text{If } |\mathcal{V}| = n\text{:}$$

$$\text{features}(v_i) = e_i \in \mathbb{R}^n$$

  - Sparse vector representation

# One-hot encoding

**Pros:**

- Sparse representation
- No learning necessary

**Cons:**

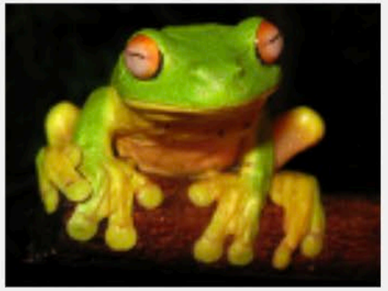
- Feature space must be same size as vocabulary
- No way to encode shared representations of words

# Embeddings

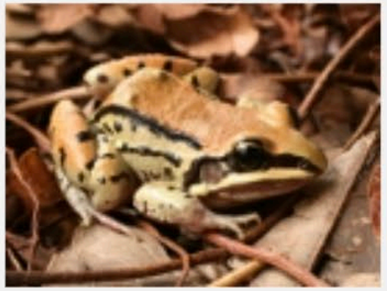
- Word2Vec (Mikolov, 2013), GloVe (Pennington, 2014) trained embeddings on word-level tokens
- Nearest neighbors embeddings are semantically similar

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

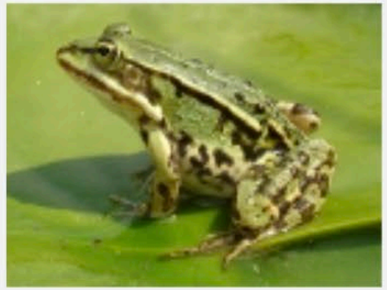
0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus



3. litoria



4. leptodactylidae



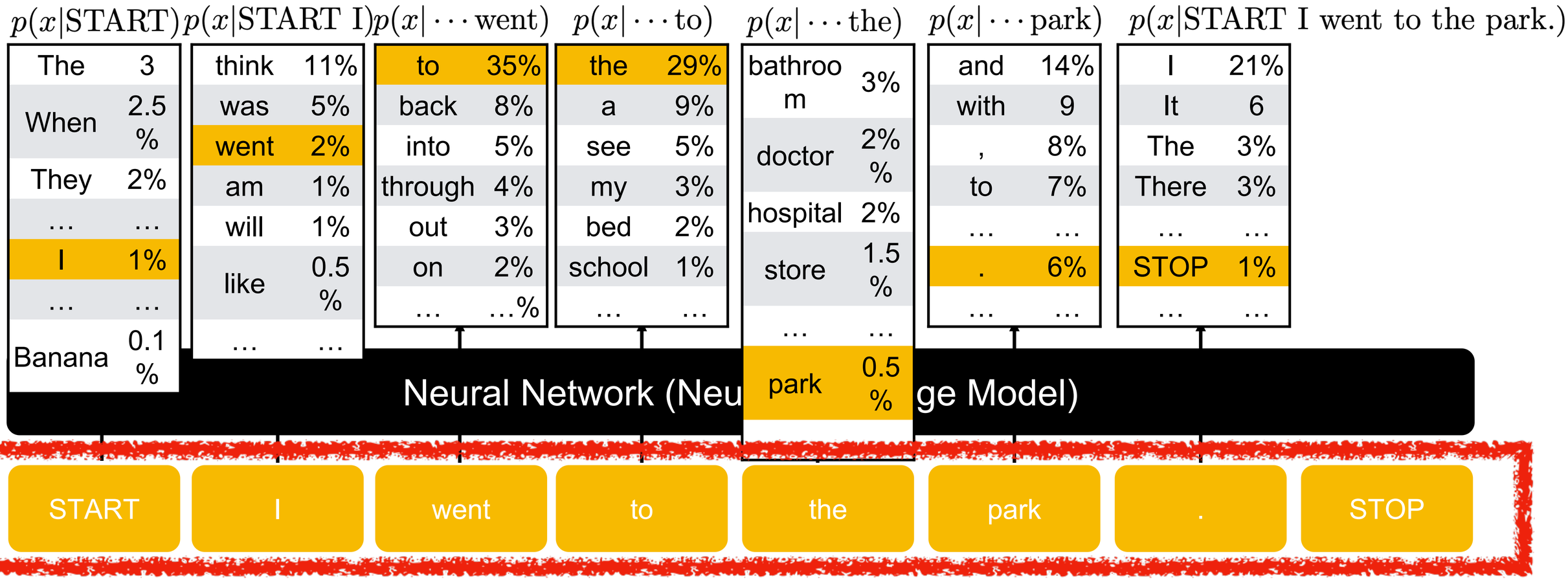
5. rana

7. eleutherodactylus

https://nlp.stanford.edu/projects/glove/

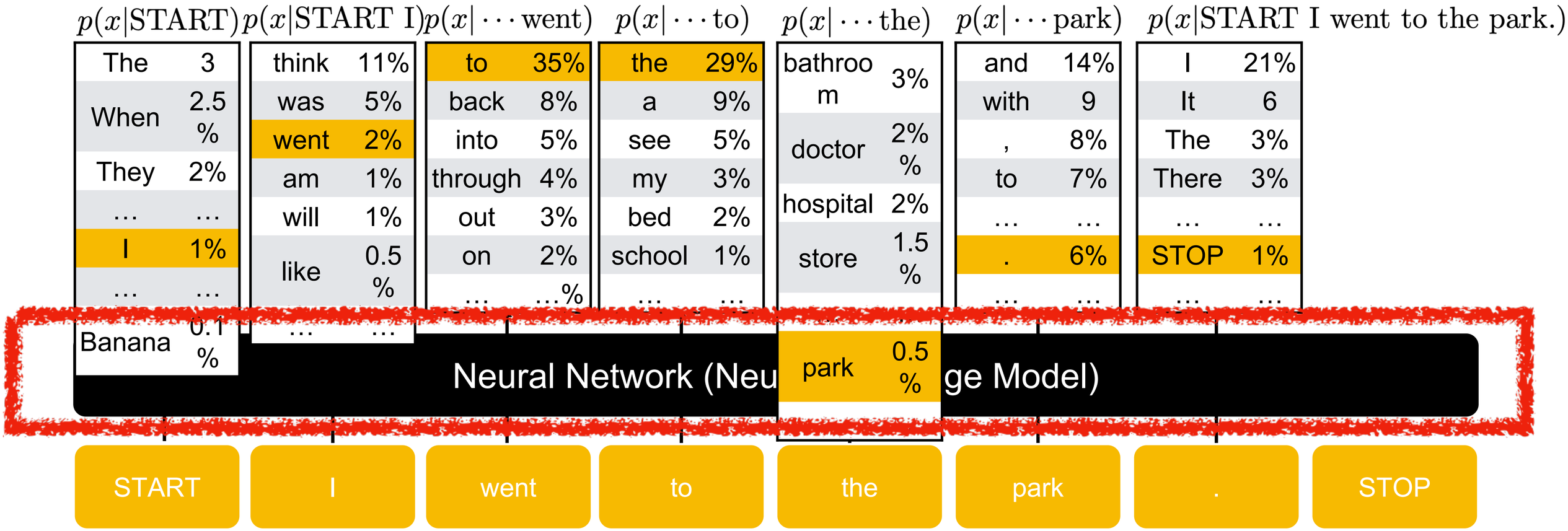
# Inputs/Outputs

- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)

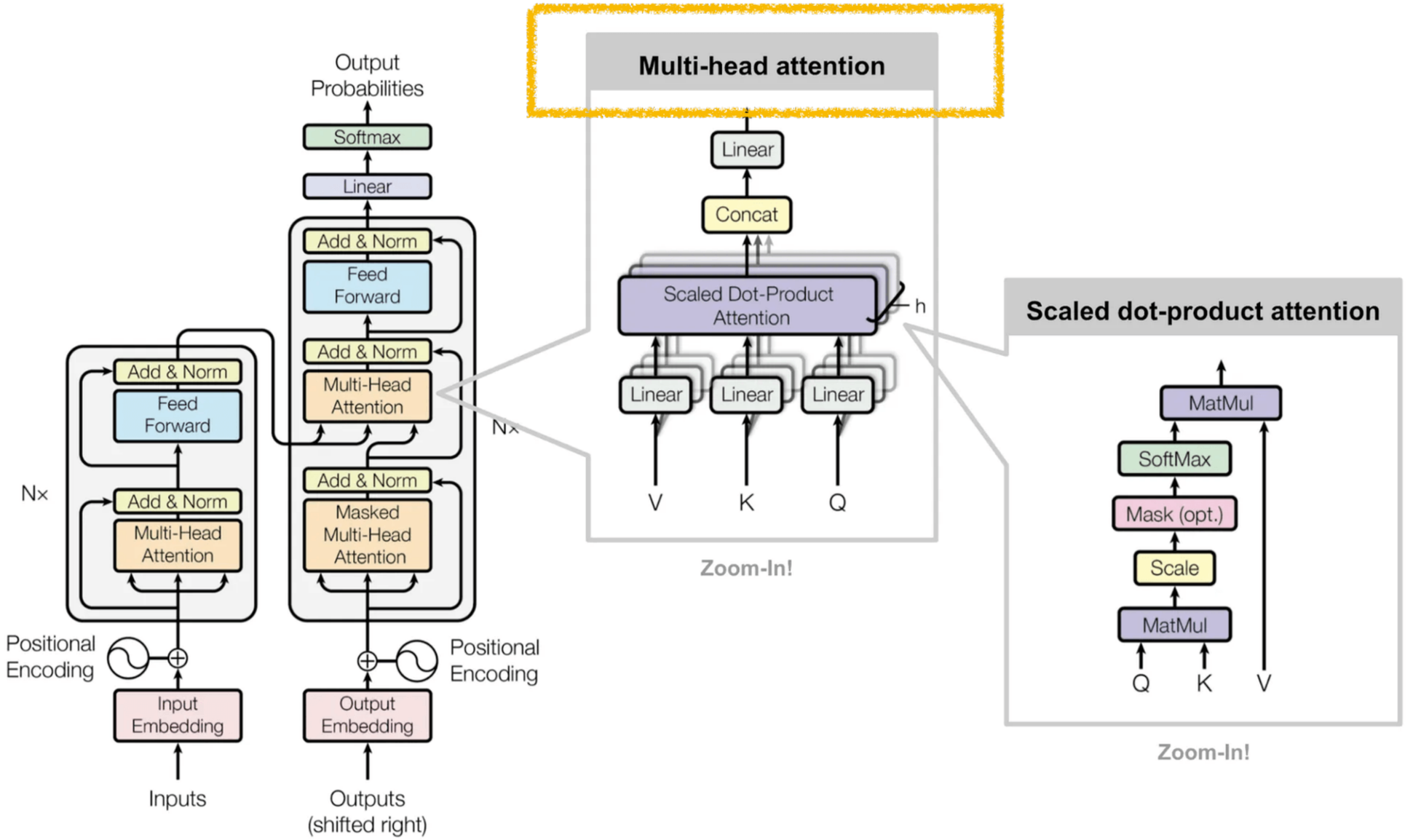


# Inputs/Outputs

- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)



# Transformer



# Evolution of Sequence Models

Deep Learning Review

LSTM

RNN

Transformer

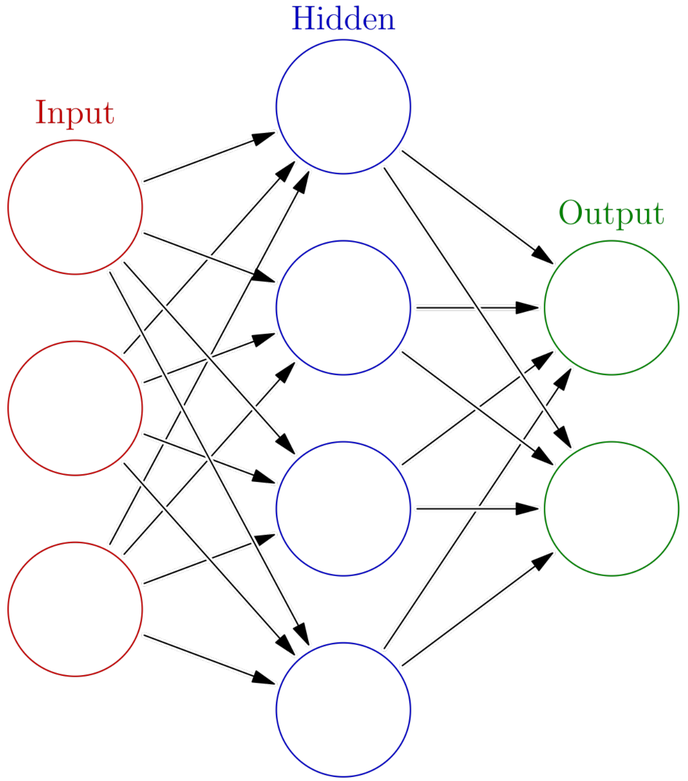
# Deep Learning Review

# Neural Networks

**Goal:** Approximate some function

**Essential elements:**

- Input: Vector , Output:
- Hidden representation layers
- Non-linear, differentiable (almost everywhere) activation function (applied element-wise)
- Weights and bias




$$\hat{y} = W_2\sigma(W_1 x + b_1) + b_2$$

$$\text{where } x \in \mathbb{R}^3, \hat{y} \in \mathbb{R}^2, W_1 \in \mathbb{R}^{4\times 3},$$

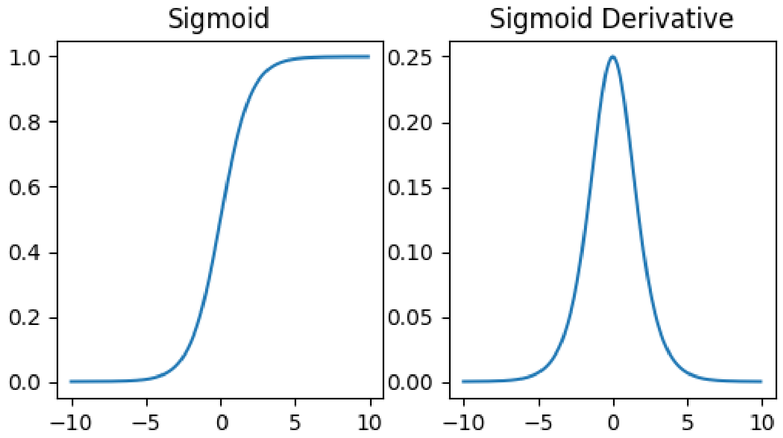
$$W_2 \in \mathbb{R}^{2\times 4}, b_1 \in \mathbb{R}^4, \text{ and } b_2 \in \mathbb{R}^2$$

https://en.wikipedia.org/wiki/Artificial_neural_network

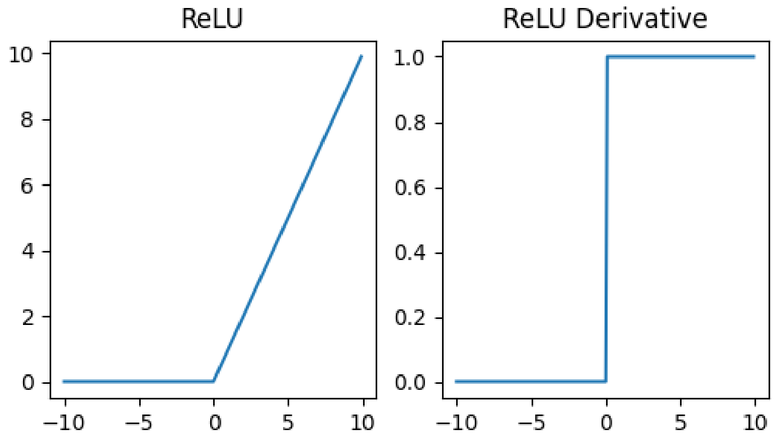
# Common activation functions

## Sigmoid

$$\frac{1}{1+e^{-x}}$$



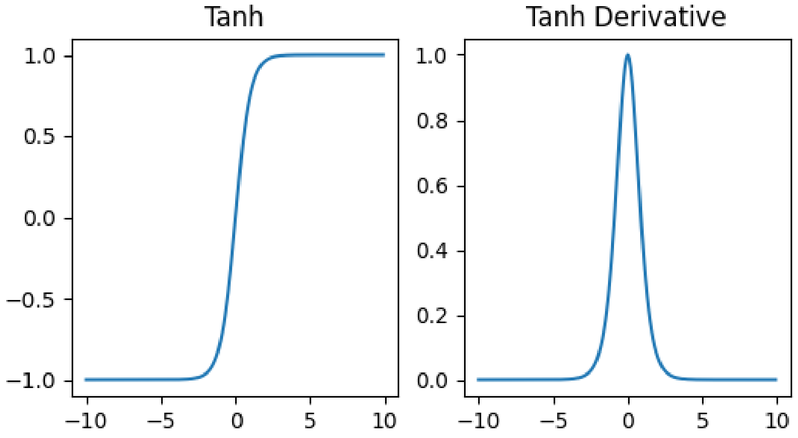

## ReLU



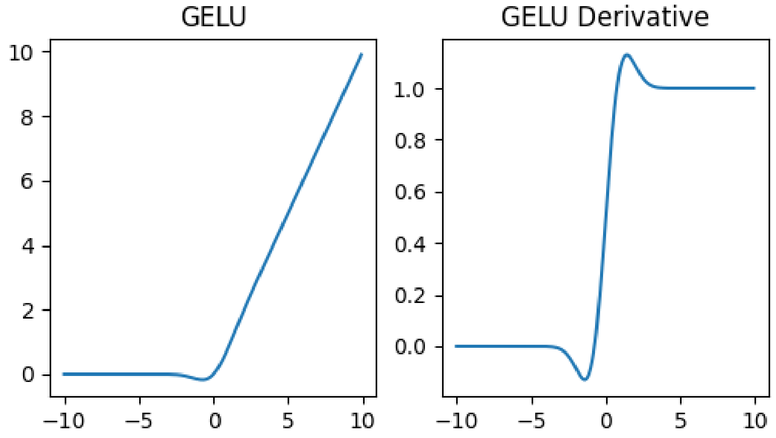

$$\max(0, x)$$

## Tanh

$$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$




## GeLU




$$\frac{1}{2}x\left(1+\operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

# Softmax outputs a valid probability distribution

$$\text{softmax}(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{d} \exp(y_j)}$$

**Proof that softmax is a valid probability distribution:**

1.Non-negativity: $\forall i,\ \text{softmax}(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{d} \exp(y_j)} \geq 0$ since $\exp(y_i) \geq 0$ for all $i$.

2.Normalization: $\sum_{i=1}^{d} \text{softmax}(y)_i = \sum_{i=1}^{d} \frac{\exp(y_i)}{\sum_{j=1}^{d} \exp(y_j)} = \frac{\sum_{i=1}^{d} \exp(y_i)}{\sum_{j=1}^{d} \exp(y_j)} = 1.$

# Outline

Deep Learning Review

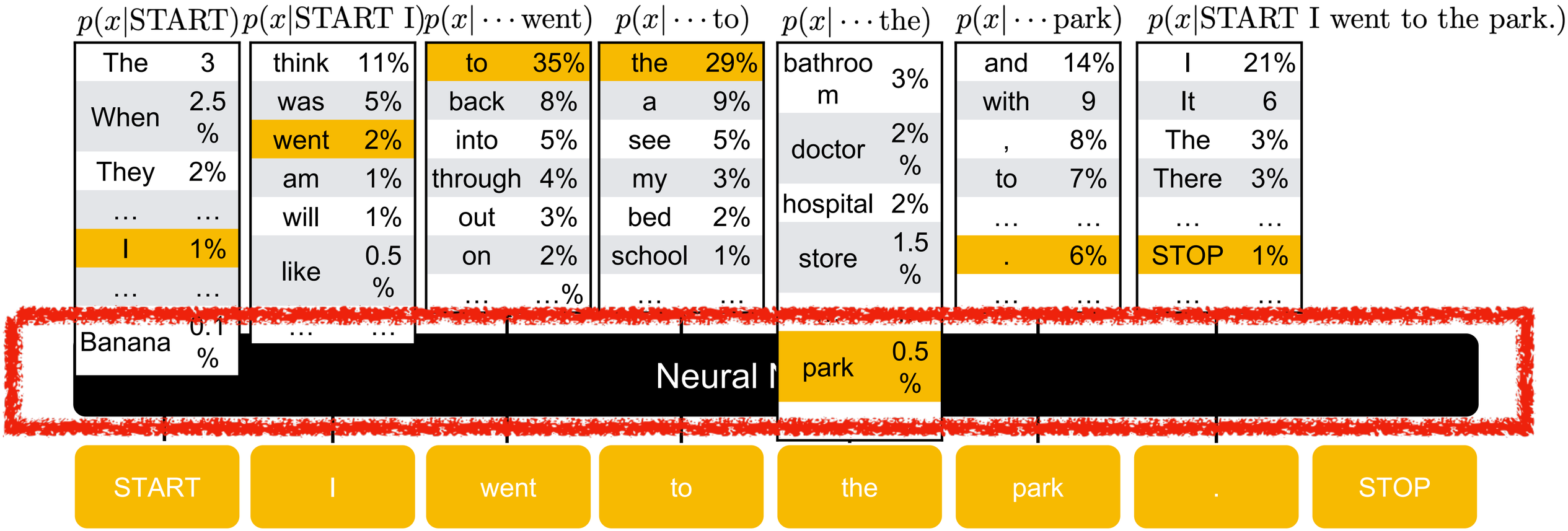
LSTM

RNN

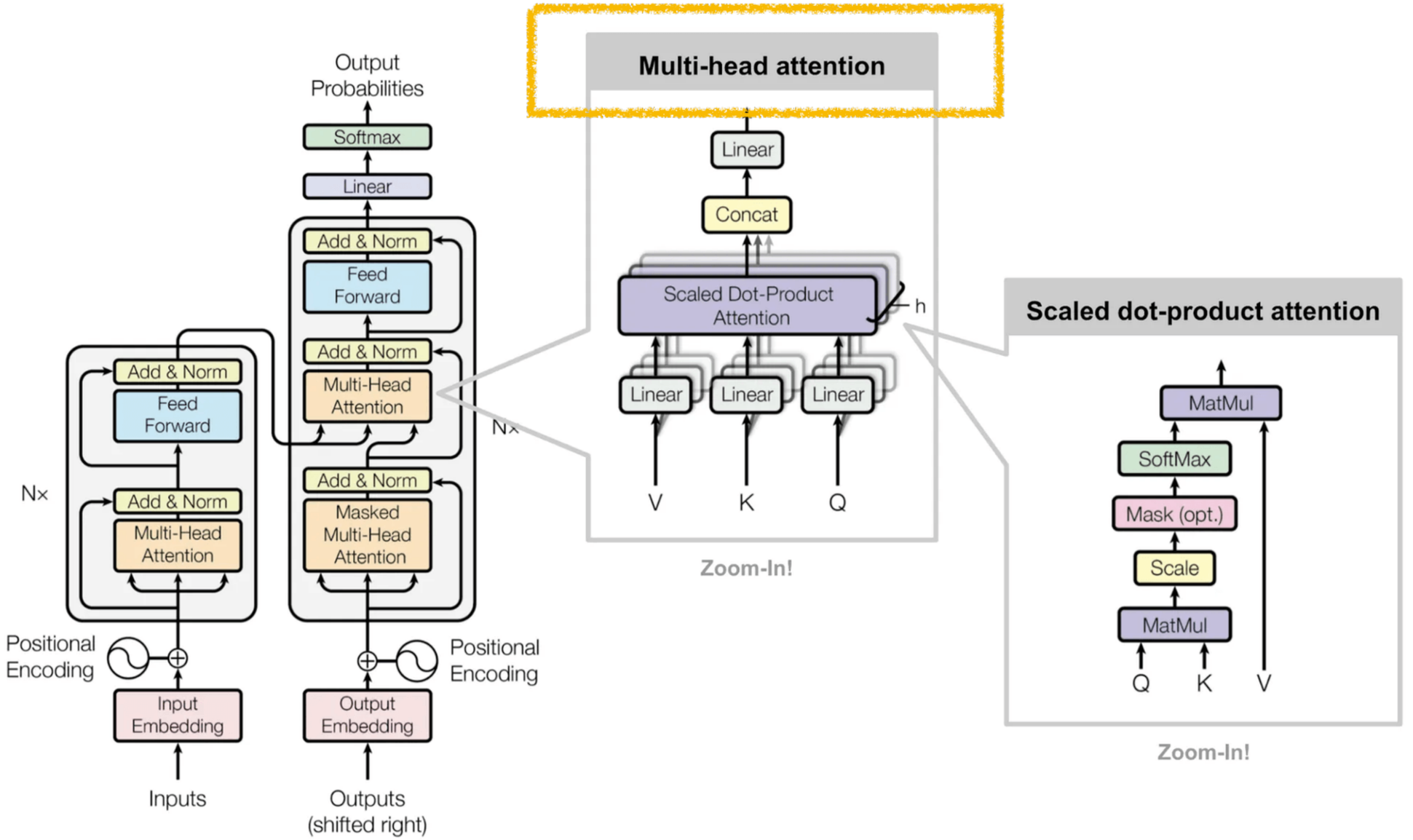
Transformer

# Inputs/Outputs

- **Input:** sequences of words (or tokens)
- **Output:** probability distribution over the next word (token)



# Transformer



# Attention Is All You Need (NeurIPS 2017)

## 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

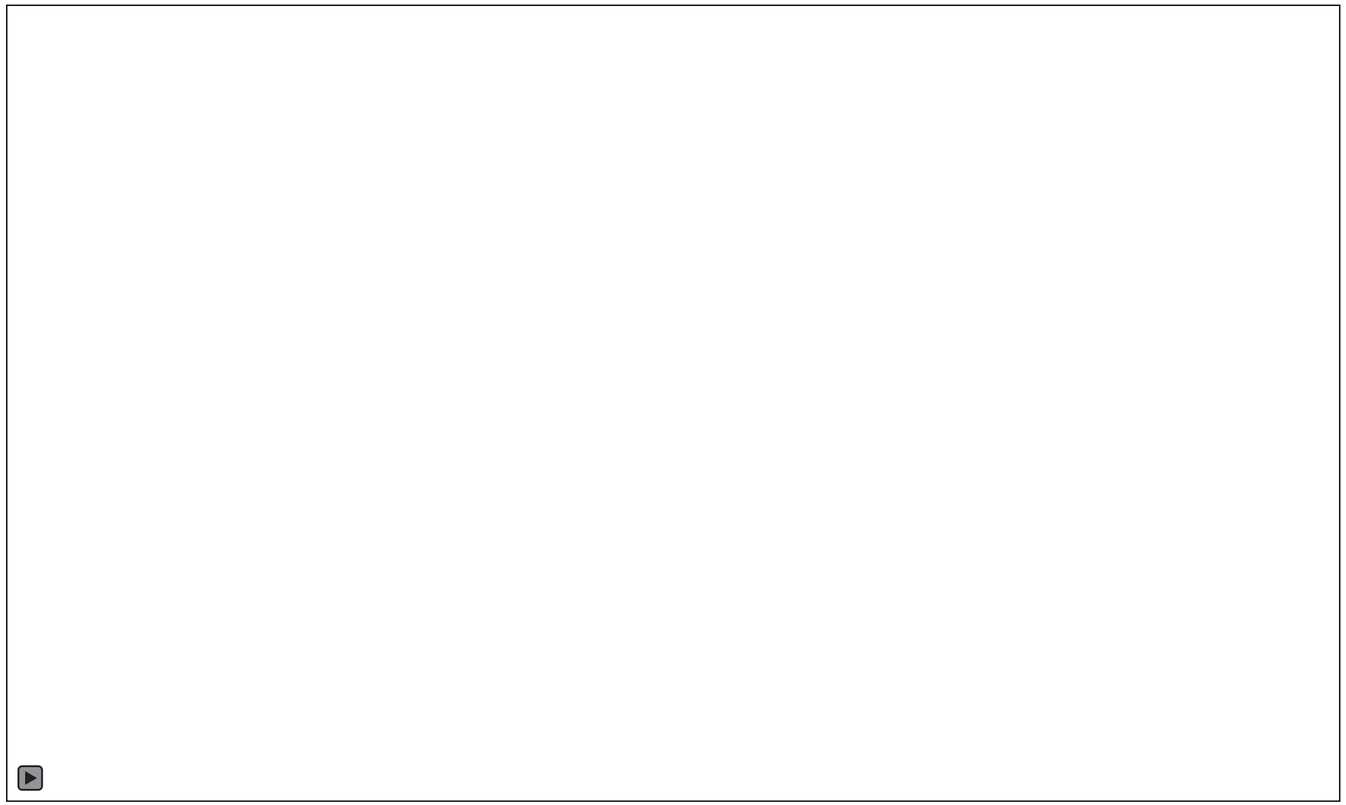
**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com

# The New De Facto Method: Attention

Instead of deciding the next token solely based on the previously seen tokens, **each token will "look at" all input tokens at the same to decide which ones are most important** to decide the next token.



**In practice, the actions of all tokens are done in parallel!**

# Self-Attention: Summary

Let $w_{1:n}$ be a sequence of words in vocabulary $V$, like *Steve Jobs founded Apple.*

For each $w_i$, let $a_i = Ew_i$, where $E \in \mathbb{R}^{d \times |V|}$ is an embedding matrix.

1. Transform each word embedding with weight matrices $W_Q, W_K, W_V$ , each in $\mathbb{R}^{d \times d}$

$$q_i = W_Q a_i \text{ (queries)} \qquad k_i = W_K a_i \text{ (keys)} \qquad v_i = W_V a_i \text{ (values)}$$

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

$$\alpha_{i,j} = k_j q_i \qquad \alpha_{i,j}{}' = \frac{e^{\alpha_{i,j}}}{\sum_j e^{\alpha_{i,j}}}$$

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

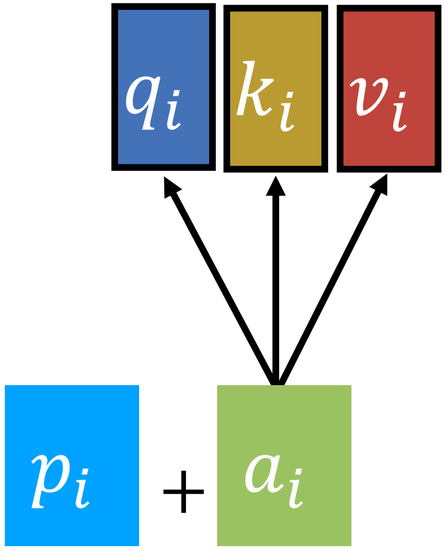
$$b_i = \sum_j \alpha_{i,j}{}' v_j$$

# No Sequence Order → Position Embedding

- All tokens in an input sequence are **simultaneously** fed into self-attention blocks. Thus, there's no difference between tokens at different positions.
  - **We lose the position info!**

- **How do we bring the position info back, just like in RNNs?**
  - **Representing each sequence index as a vector:** $p_i \in \mathbb{R}^d$, for $i \in \{1, \ldots, n\}$

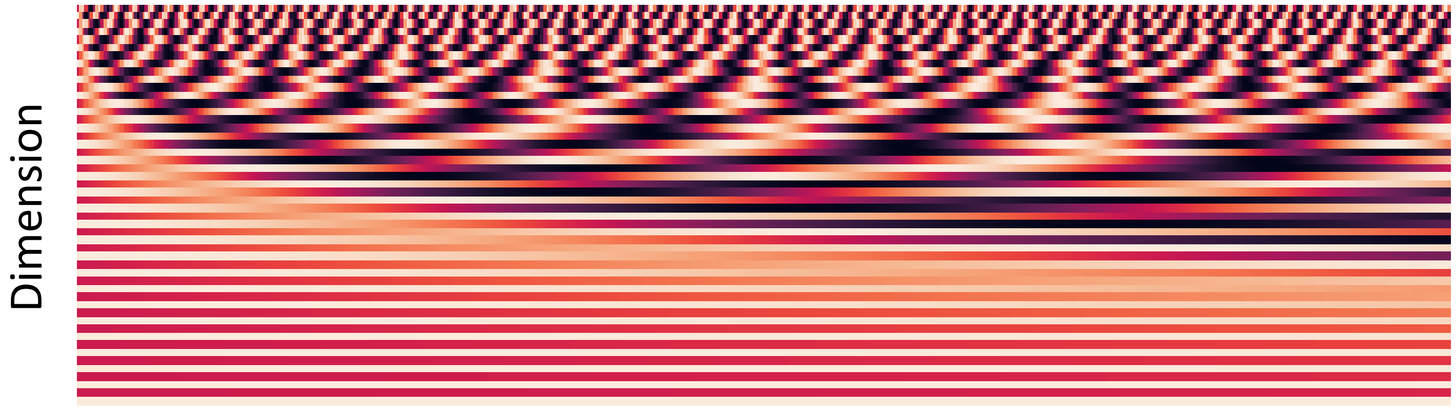
- **How to incorporate the position info into the self-attention blocks?**
  - **Just add the $p_i$ to the input:** $\hat{a}_i = a_i + p_i$
  - where $a_i$ is the embedding of the word at index $i$.
  - In deep self-attention networks, we do this at the **first layer**.
  - We can also concatenate $a_i$ and $p_i$, but more commonly we add them.



# Position Representation Vectors via Sinusoids

**Sinusoidal Position Representations (from the original Transformer paper):** concatenate sinusoidal functions of varying periods.

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




https://timodenk.com/blog/linear-relationships-in-the-transformers-positional-encoding/

- Periodicity indicates that maybe "absolute position" isn't as important
- Maybe can extrapolate to longer sequences as periods restart!

- Not learnable; also the extrapolation doesn't really work!

# Learnable Position Representation Vectors

**Learned absolute position representations:** $p_i$ contains learnable parameters.

- Learn a matrix $p \in \mathbb{R}^{d \times n}$, and let each $p_i$ be a column of that matrix
- Most systems use this method.

- **Flexibility:** each position gets to be learned to fit the data

- Cannot extrapolate to indices outside $1, \ldots, n$.

**Sometimes people try more flexible representations of position:**

- Relative linear position attention [Shaw et al., 2018]
- Dependency syntax-based position [Wang et al., 2019]

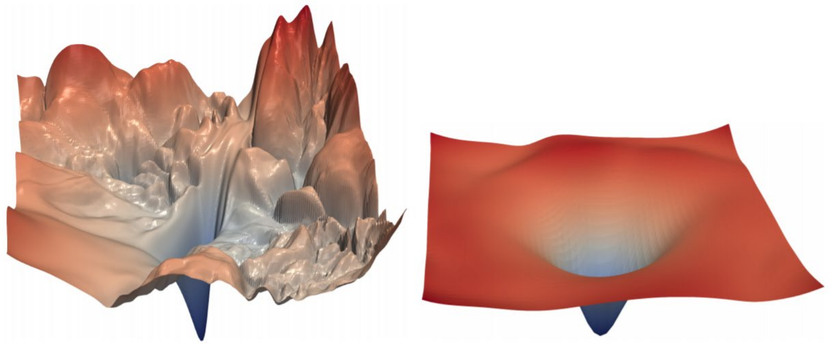
# Residual Connections

[He et al., 2016]

- Residual connections are a trick to help models train better.
  - Instead of $X^{(i)} = Layer(X^{(i-1)})$ (where $i$ represents the layer)


  - We let $X^{(i)} = X^{(i-1)} + Layer(X^{(i-1)})$ (so we only have to learn "the residual" from the previous layer)


  - Gradient is great through the residual connection; it's 1!
  - Bias towards the identity function!



**[no residuals]** **[residuals]**

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

# Layer Normalization [Ba et al., 2016]

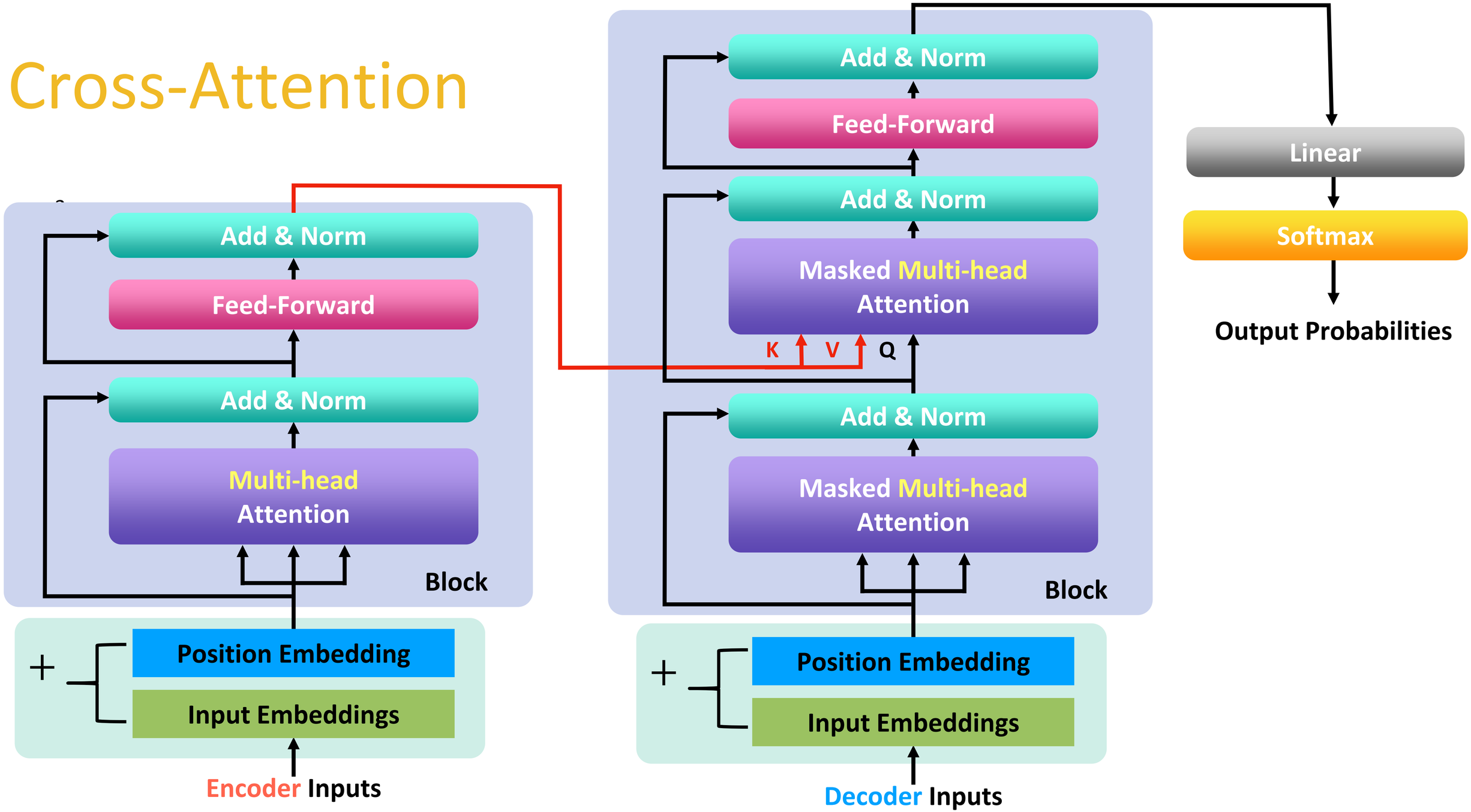
35

- Layer normalization is a trick to help models train faster.
- **Idea:** cut down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation within each layer.
  - LayerNorm's success may be due to its normalizing gradients [Xu et al., 2019]
- Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.
- Let $\mu = \sum_{j=1}^{d} x_j$; this is the mean; $\mu \in \mathbb{R}$.
- Let $\sigma = \sqrt{\frac{1}{d} \sum_{j=1}^{d} (x_j - \mu)^2}$; this is the standard deviation; $\sigma \in \mathbb{R}$.
- Let $\gamma \in \mathbb{R}^d$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)
- Then layer normalization computes:
  - $\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$

Normalize by scalar mean and variance

Modulate by learned element-wise gain and bias

# Cross-Attention



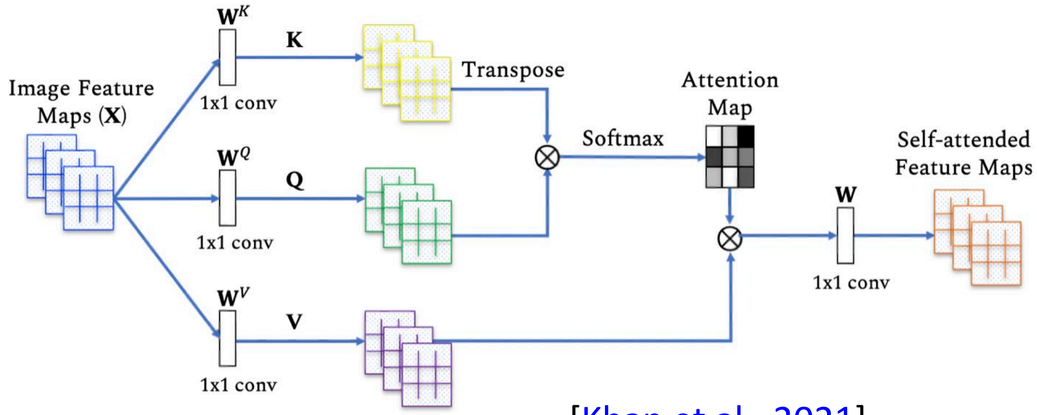
# Cross-Attention Details

- 3/7 **Self-attention:** queries, keys, and values come from the same source.
- **Cross-Attention:** *keys* and *values* are from **Encoder** (like a memory); *queries* are from **Decoder**.
- Let $h_1, \ldots, h_n$ be output vectors from the Transformer **encoder**, $h_i \in \mathbb{R}^d$.
- Let $z_1, \ldots, z_n$ be input vectors from the Transformer **decoder**, $z_i \in \mathbb{R}^d$.
- **Keys** and **values** from the **encoder**:
  - $k_i = W_K h_i$
  - $v_i = W_V h_i$
- **Queries** are drawn from the **decoder**:
  - $q_i = W_Q z_i$

# The Revolutionary Impact of Transformers

- **Almost all current-day leading language models** use Transformer building blocks.
  - E.g., GPT1/2/3/4, T5, Llama 1/2, BERT, … almost anything we can name
  - Transformer-based models dominate nearly all NLP leaderboards.

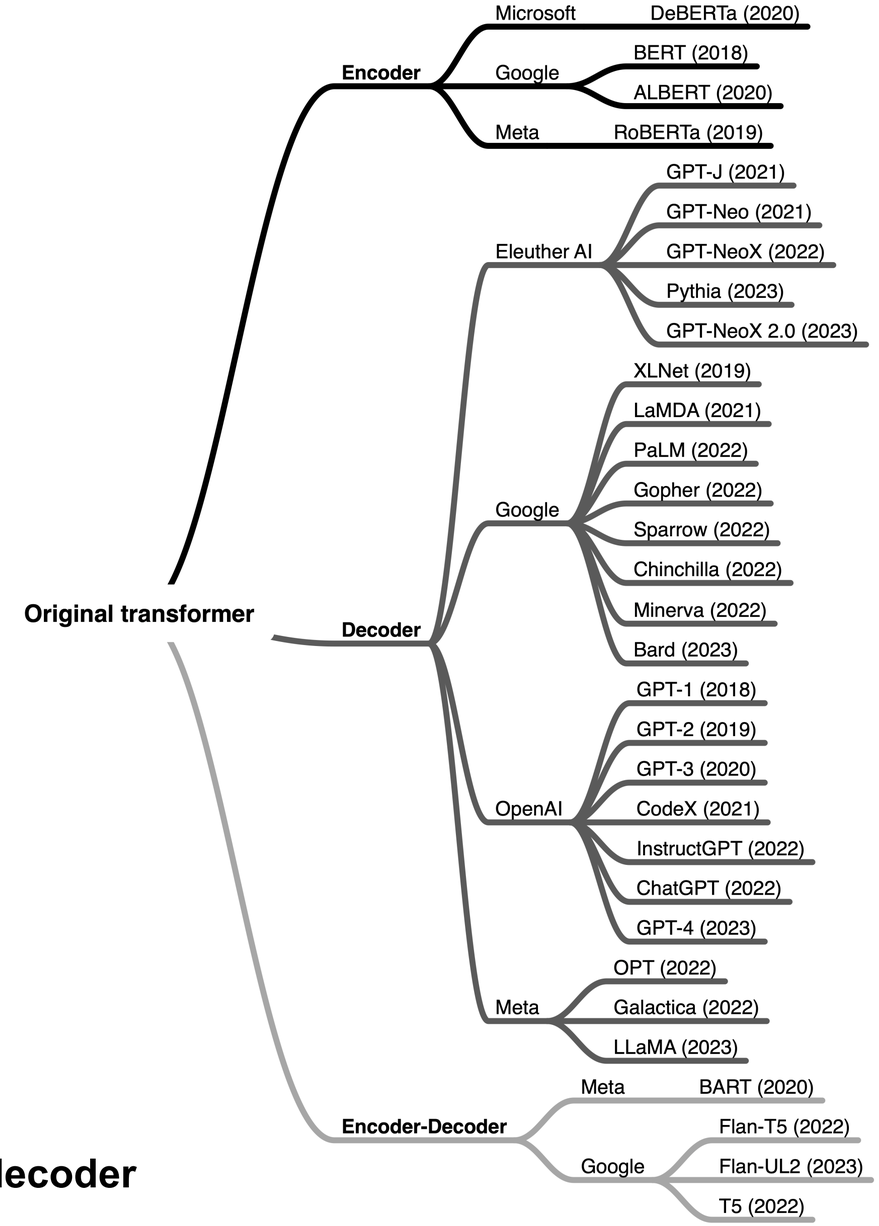
- Since Transformer has been popularized in language applications, computer vision also adapted Transformers, e.g., **Vision Transformers**.




[Khan et al., 2021]

**What's next after Transformers?**

# Transformer types and examples




**https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder**

# Lots of Information in Raw Texts

The dish was a symphony of flavors, with each bite delivering a harmonious blend of sweet and savory notes that left my taste buds in a state of culinary _____. euphoria

The dish fell short of expectations, as the flavors lacked depth and the texture was disappointingly bland, leaving me with a sense of culinary _____ letdown

Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____ disappointing

Despite a promising premise, the movie failed to live up to its potential, as the plot felt disjointed, the characters lacked depth, and the pacing left me disengaged, resulting in a rather _____ cinematic experier amazing

# Lots of Information in Raw Texts

**Verb** I went to Hawaii for snorkeling, hiking, and whale _____ watching

**Preposition** I walked across the street, checking for traffic _____ over y shoulders.

**Commonsense** I use ____ knife __ and fork to eat steak.

**Time** Ruth Bader Ginsburg was born in _____ 1933

**Location** University of Washington is located at _____ Seattle Washington.

**Math** I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, _____. 34

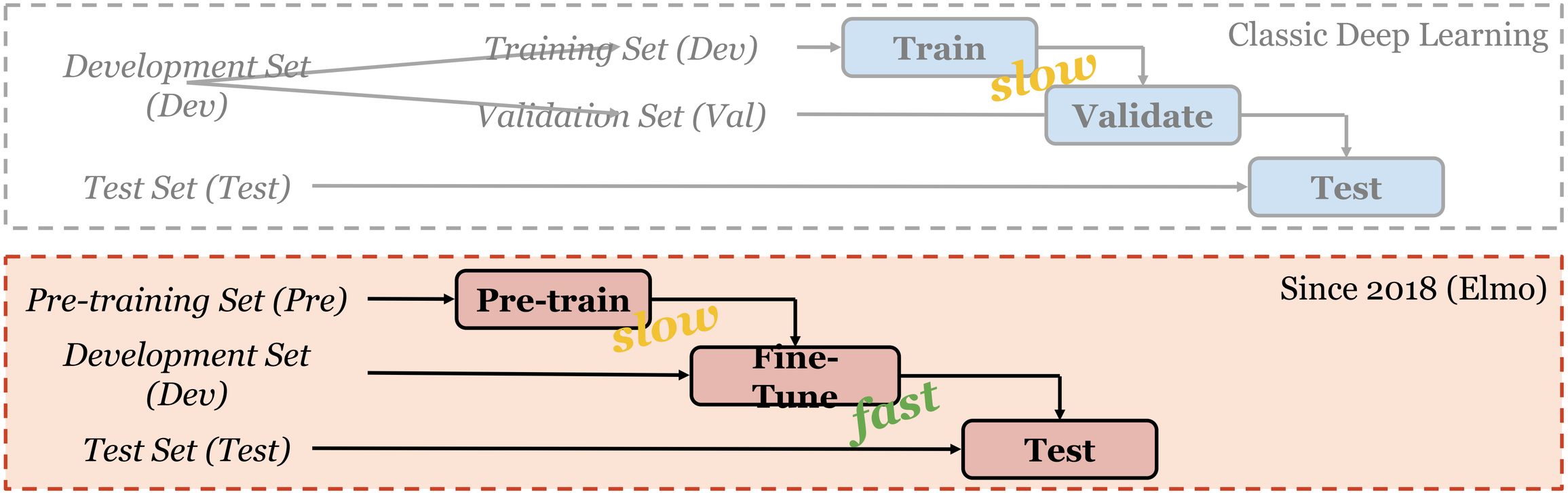
**Chemistry** Sugar is composed of carbon, hydrogen, and _____ oxygen

…

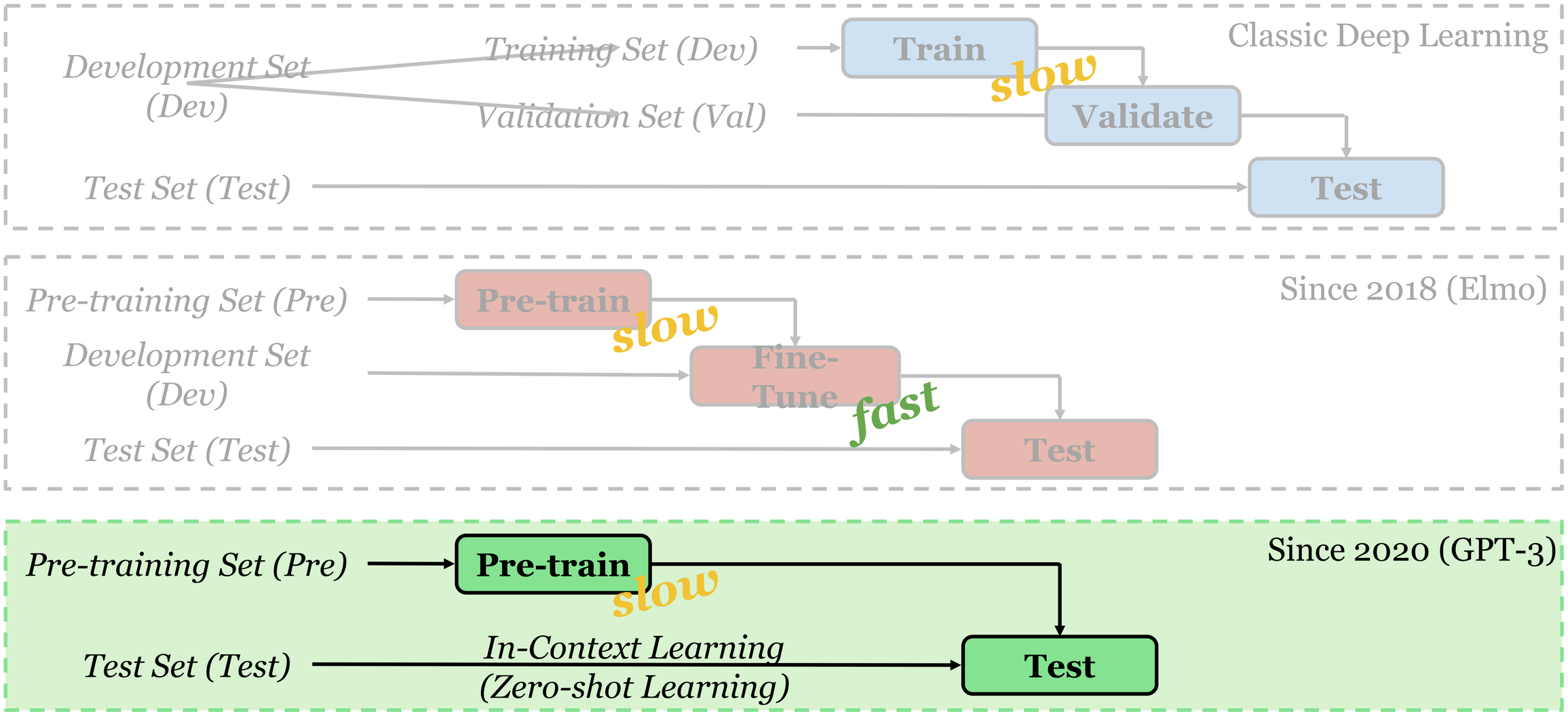
# How to Harvest Underlying Patterns, Structures, and Semantic Knowledge from Raw Texts?

**Pre-training! (aka self-supervised learning)**

# Overview: The Paradigm Shift

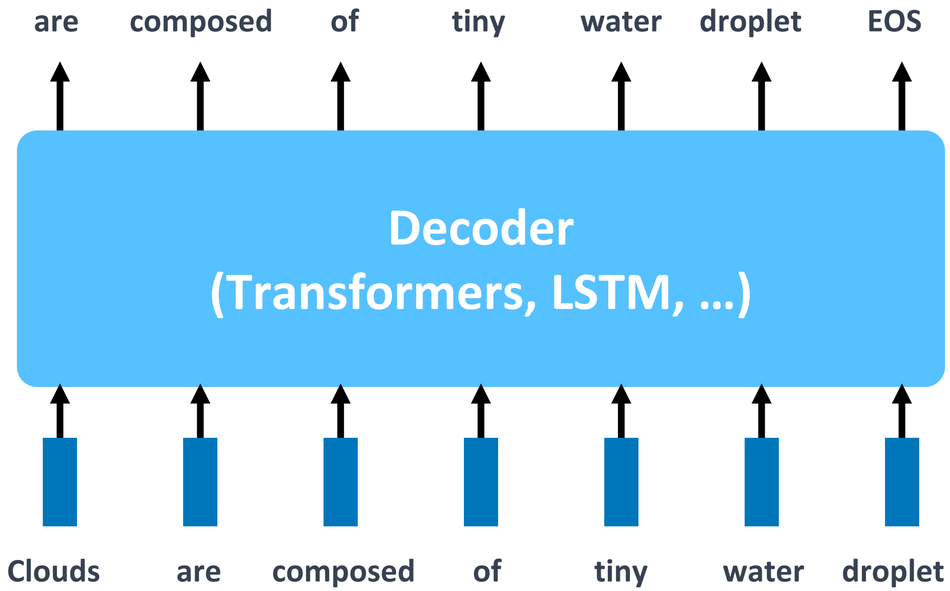


# Overview: The Paradigm Shift



# Self-supervised Pre-training for Learning Underlying Patterns, Structures, and Semantic Knowledge

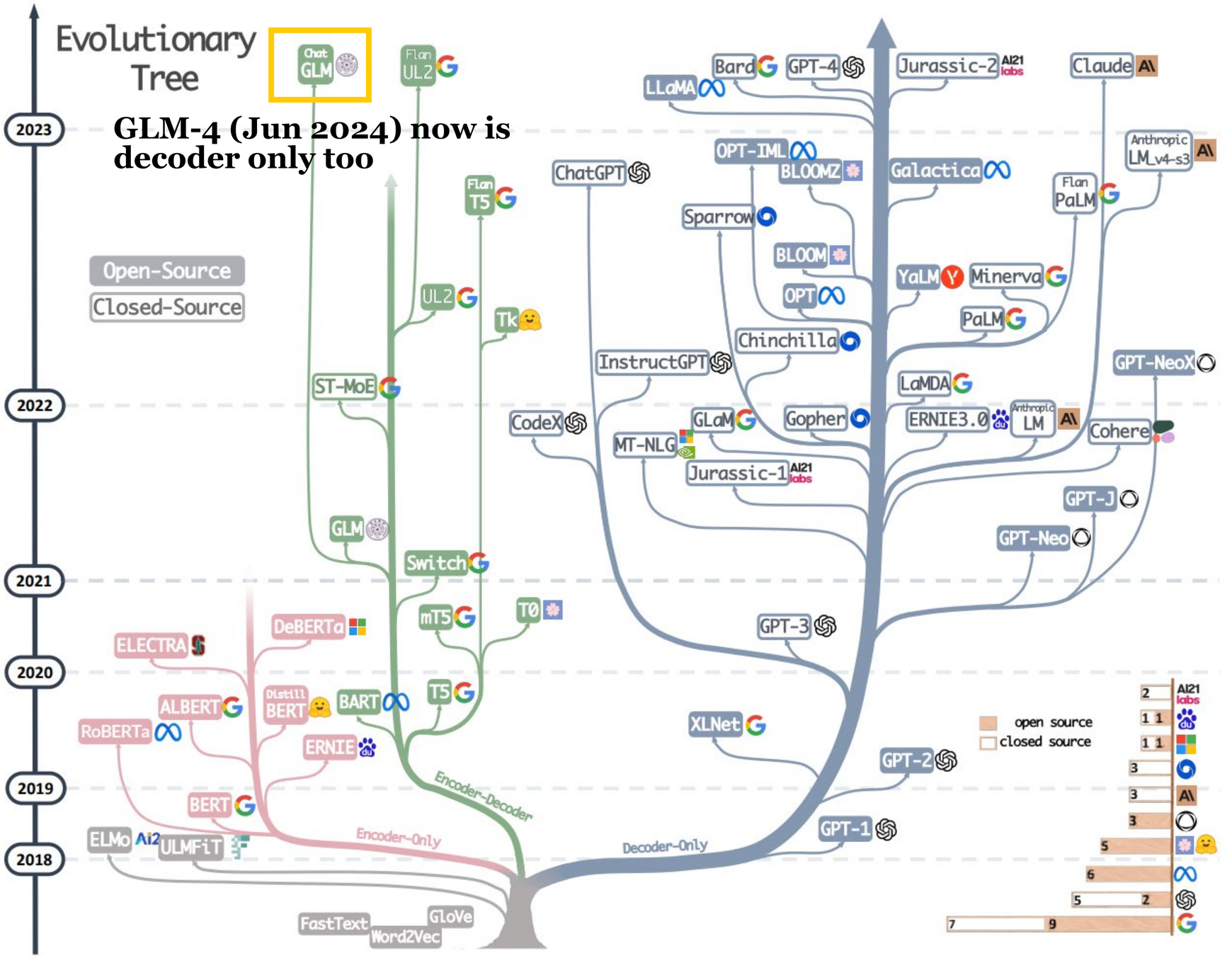
- Pre-training through **language modeling** [Dai and Le, 2015]
  - Model $P_\theta(w_t|w_{1:t-1})$, the probability distribution of the next word given previous contexts.
  - **There's lots of (English) data for this!** E.g., books, websites.
  - **Self-supervised** training of a neural network to perform the language modeling task with massive raw text data.
  - Save the network parameters to reuse later.



# Pretraining

- Evolution tree
  - Until Apr 2023

Yang et al. "Harnessing the power of llms in practice: A survey on chatgpt and beyond." ACM Transactions on Knowledge Discovery from Data 18, no. 6 (2024): 1-32.



# Pre-training Data

- **Ideally, we want high-quality data for pre-training.**
  - High-quality: natural, clean, informative, and diverse;
  - Books, Wikipedia, news, scientific papers;
  - High quality data eventually runs out.
- **In practice, internet is the most viable option for data.**
  - In the digital era, the web is the go-to place for general domain human knowledge;
  - It is massive and unlikely to grow slower than computing resources*
  - Publicly available*

  * Are these claim still true these days? Questionable.

# Pre-training Data

- **Web data is plentiful, but can be challenging to work with.**
  - Data is noisy, dirty, and biased
    - Recency bias / Demographic biases /Language biases
  - Web is much more dynamic than static HTML pages
    - CSS, JavaScript, interactivity, etc.
    - Responsive design
    - Many HTML pages involves 20+ secondary URLs, iframes, etc.

# Pre-training Data

- **Web data is plentiful, but can be challenging to work with.**
  - What counts as content?
    - Ads, recommendation, navigation, etc.
    - Multimedia: images, videos, tables, etc.
    - HTML? What if you want to train a code language model?
    - Spam?
  - **Copyright and usage constraints** can get extremely complicated
  - Data is contaminated with auto-generated text
    - Not just from LLM usage, but also tons of templated text.
    - Training on synthetic data can lead to language model collapse (Seddik et al 2024).

Seddik et al. "How bad is training on synthetic data? a statistical analysis of language model collapse." arXiv preprint arXiv:2404.05090 (2024).

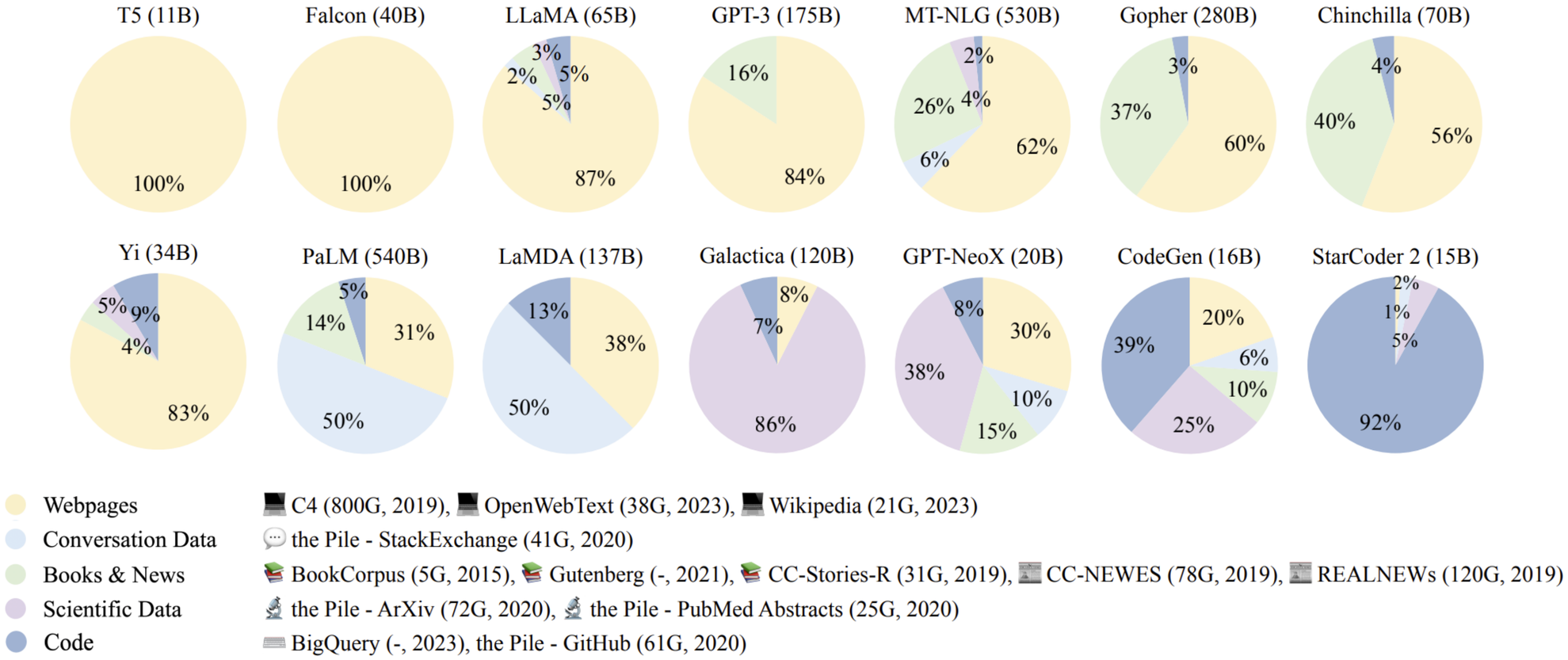
# Pre-training Data

- **The diversity of pre-training data matters**
  - Meta Llama-3 team (2024) performed extensive experiments to evaluate the best ways of mixing data from different sources in our final pretraining dataset.
  - Llama-3 final data mix:
    - 50% of tokens corresponding to **general** knowledge;
    - 25% of mathematical and **reasoning** tokens;
    - 17% **code** tokens;
    - 8% multilingual tokens.

Dubey, Abhimanyu, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur et al. "The llama 3 herd of models." arXiv preprint arXiv:2407.21783 (2024).

# Pre-training Data

- The diversity of pre-training data matters

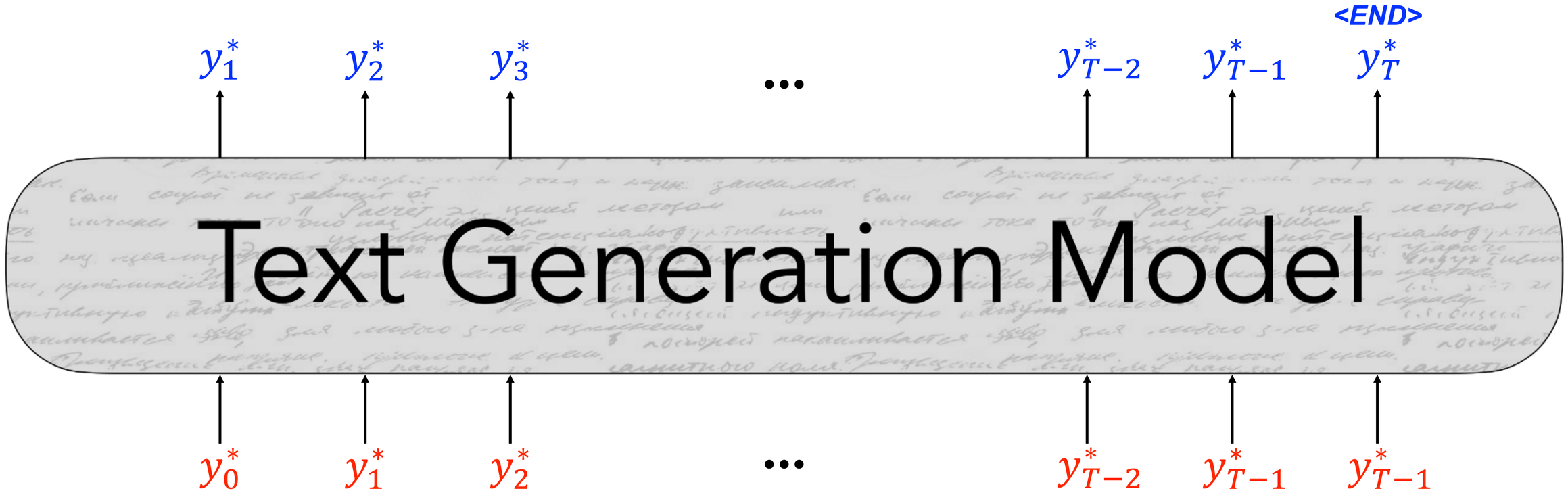



Zhao et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 (2023).

# Maximum Likelihood Pre-Training

- Trained to generate the next word $y_t^*$ given a set of preceding words $\{y^*\}_{<t}$

$$L = -\sum_{t=1}^{T} \log P(y_t^* | \{y^*\}_{<t})$$



# Pre-training Paradigms/Architectures

**Encoder**

- E.g., BERT, RoBERTa, DeBERTa, …
- **Autoencoder** model
- **Masked** language modeling

**Encoder-Decoder**

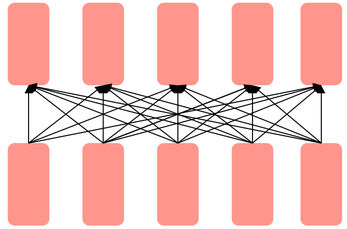
- E.g., T5, BART, …
- **seq2seq** model

**Decoder**

- E.g., GPT, GPT2, GPT3, …
- **Autoregressive** model
- **Left-to-right** language modeling

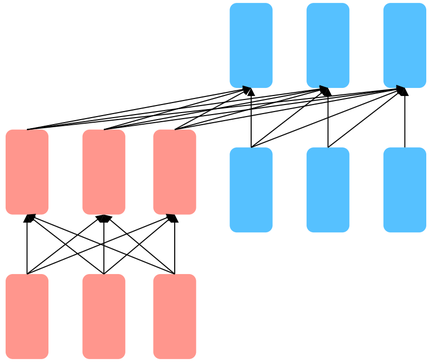
# Pre-training Paradigms/Architectures

**Encoder**



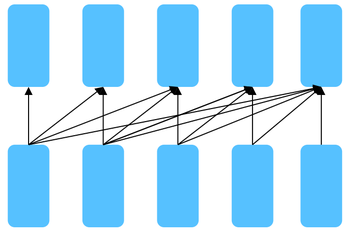
- Bidirectional; can condition on the future context

**Encoder-Decoder**



- Map two sequences of different length together

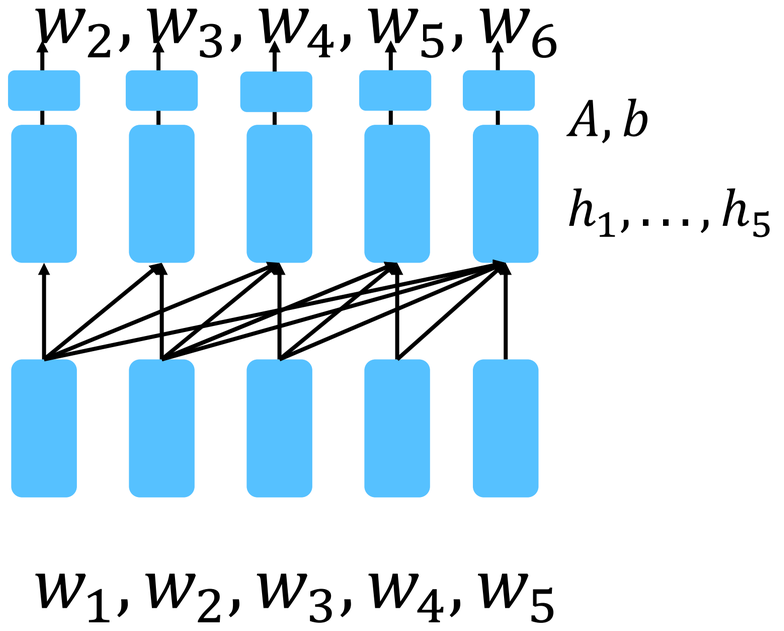
**Decoder**



- Language modeling; can only condition on the past context

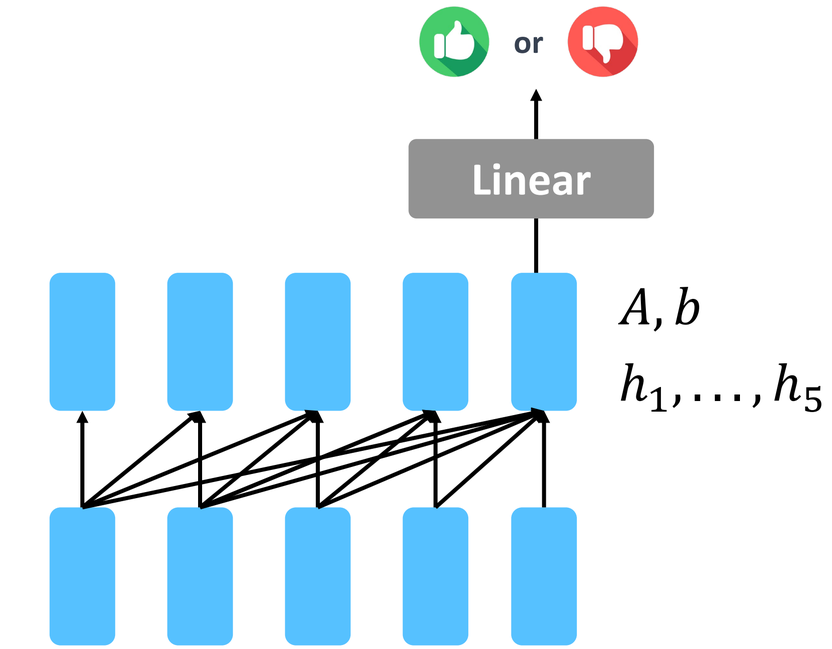
# Decoder: Training Objective

- Many most famous generative LLMs are **decoder-only**
  - e.g., GPT1/2/3/4, Llama1/2
- **Language modeling!** Natural to be used for **open-text generation**
- **Conditional LM:** $p(w_t|w_1, \ldots, w_{t-1}, x)$
  - Conditioned on a source context $x$ to generate from left-to-right
- Can be fine-tuned for **natural language generation (NLG)** tasks, e.g., dialogue, summarization.



# Decoder: Training Objective

- Customizing the pre-trained model for downstream tasks:
  - Add a **linear layer** on top of the last hidden layer to make it a classifier!
  - During fine-tuning, trained the randomly **initialized linear layer**, along with **all parameters** in the neural net.

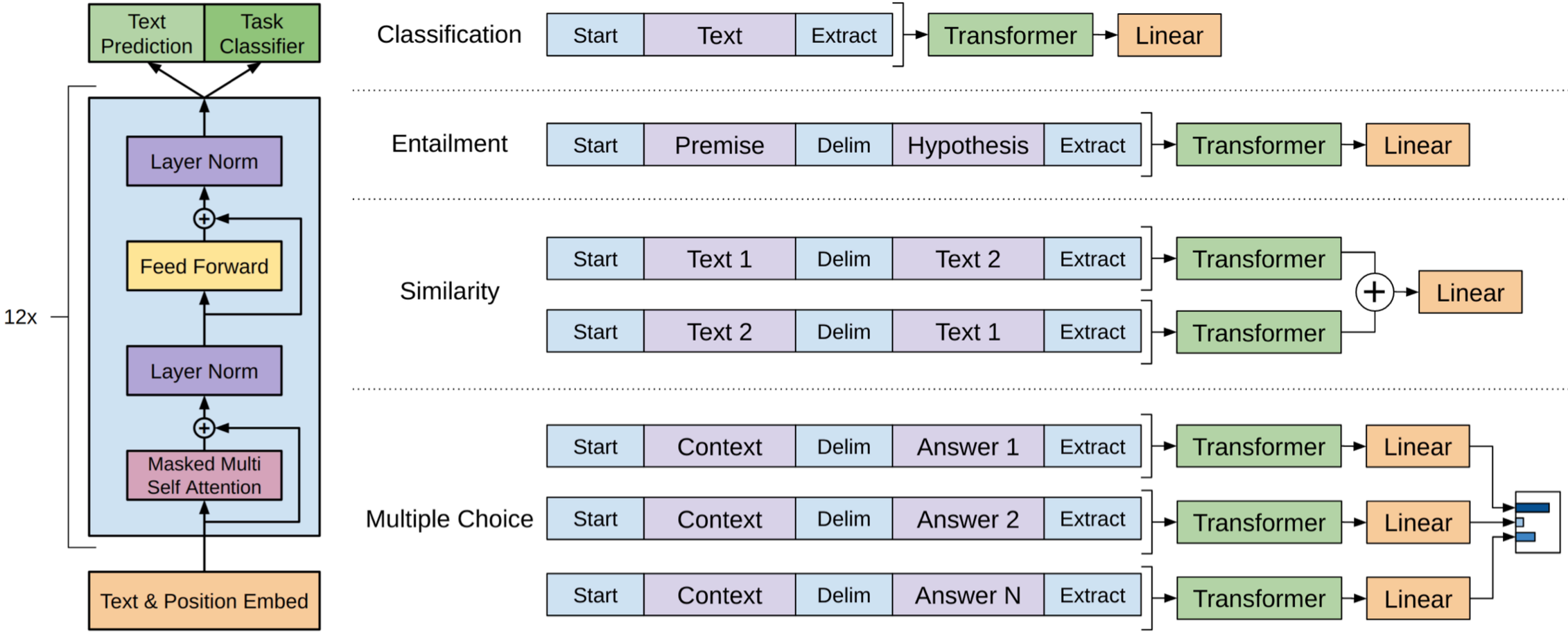



Is Santa Claus real figure?

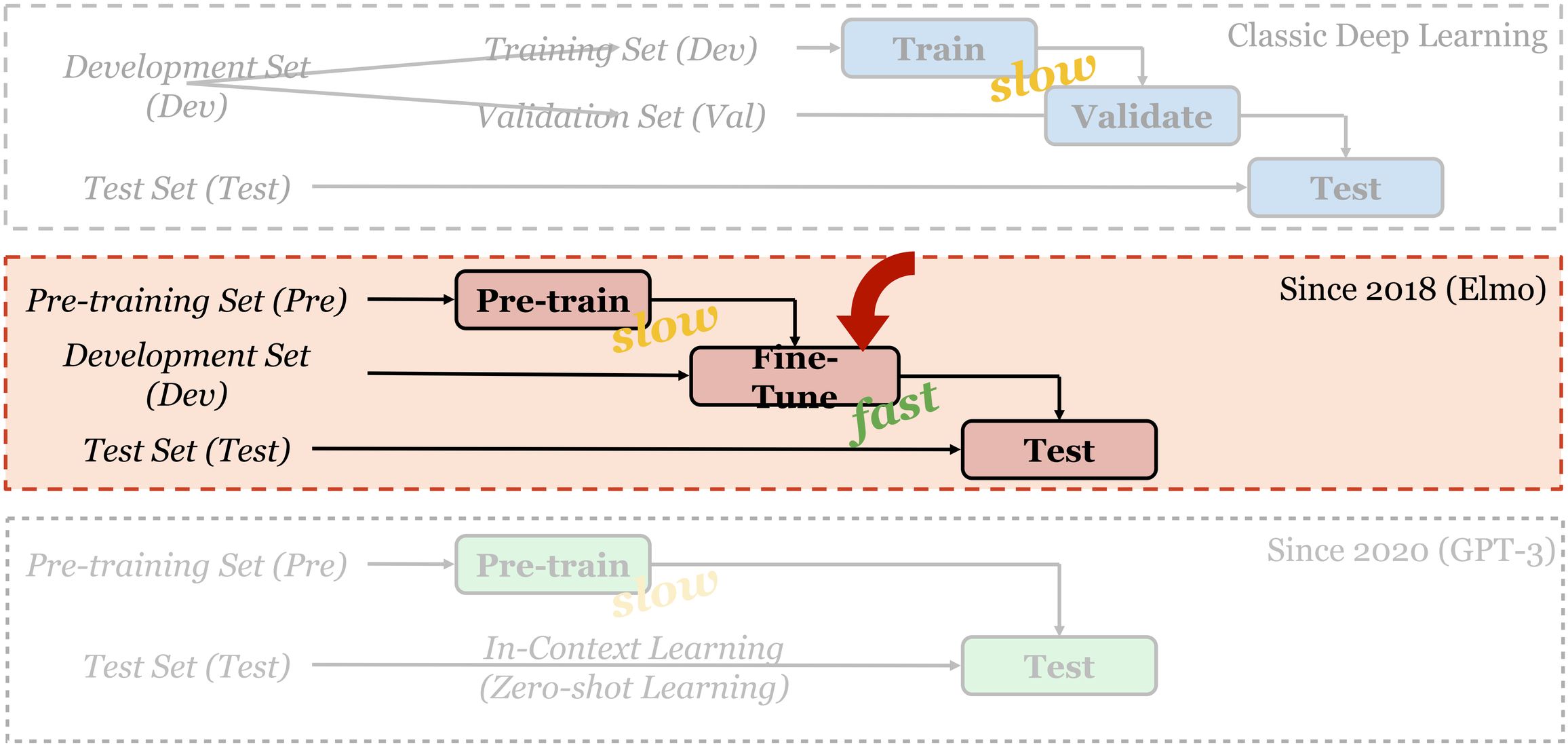
# Decoder: GPT

**G**enerative **P**re-trained **T**ransformer

[Radford et al., 2018]

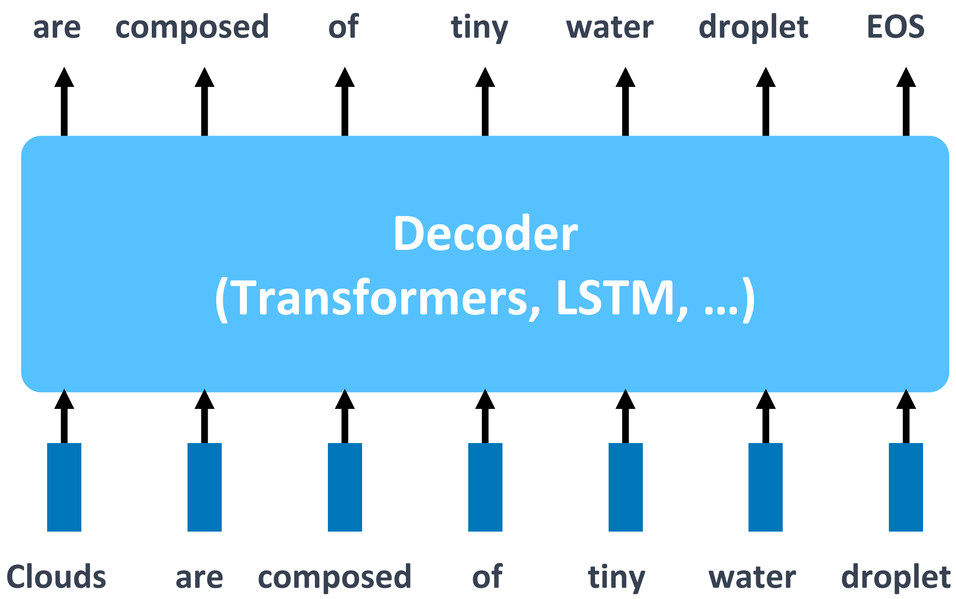


# Overview: The Paradigm Shift



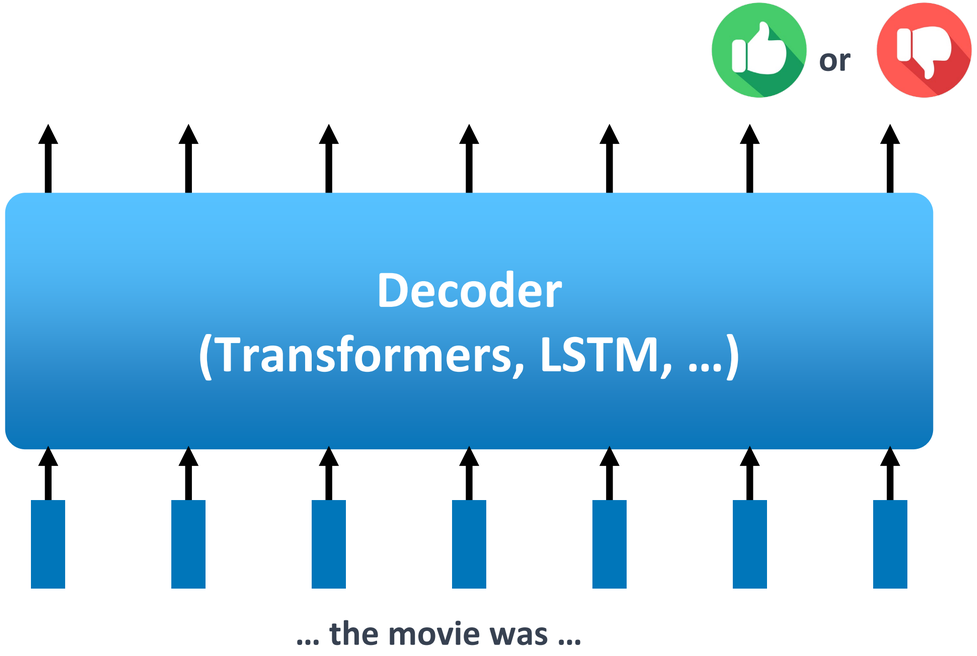
# Supervised Fine-tuning for Specific Tasks

**Step 1:**
**Pre-training**




Abundant data; learn general language

**Step 2:**
**Fine-tuning**




Limited data; adapt to the task

# Advantages of Pre-training & Fine-tuning

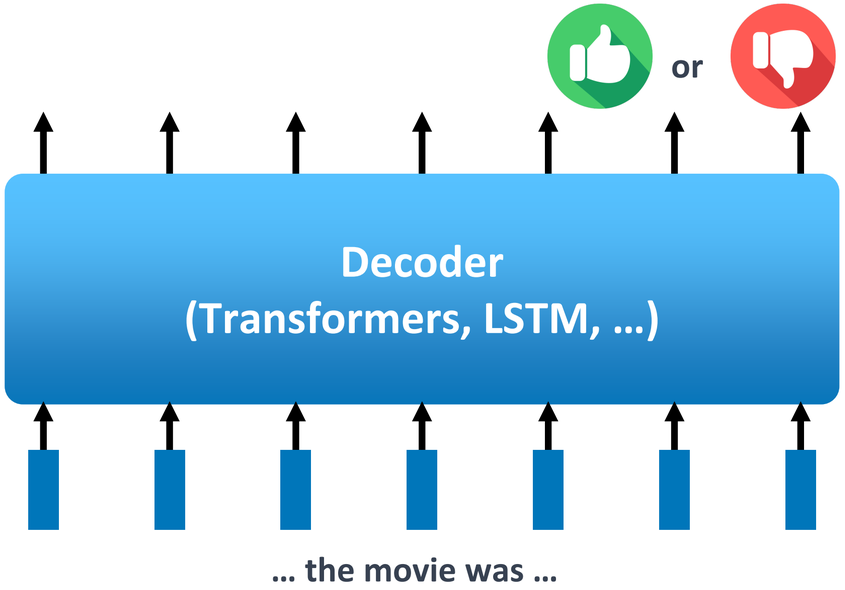
- **Leveraging rich underlying information** from abundant raw texts.
- **Reducing the reliance of task-specific labeled data** that is difficult or costly to obtain.
- **Initializing model parameters** for more **generalizable** NLP applications.
- **Saving training cost** by providing a reusable model checkpoints.
- **Providing robust representation** of language contexts.

# Caveat: Catastrophic Forgetting

- **Sequentially** pre-train then fine-tune may result in **catastrophic forgetting**, meaning that **while adapting to the new fine-tuning task, the model may lose previously learned information.**
- However, as modern language models are becoming larger in size and are pre-trained on massive raw text, they do encode tremendous amount of valuable information. **Thus, it's generally still more helpful to leverage information learned from the pre-training stage, than training on a task completely from scratch.**

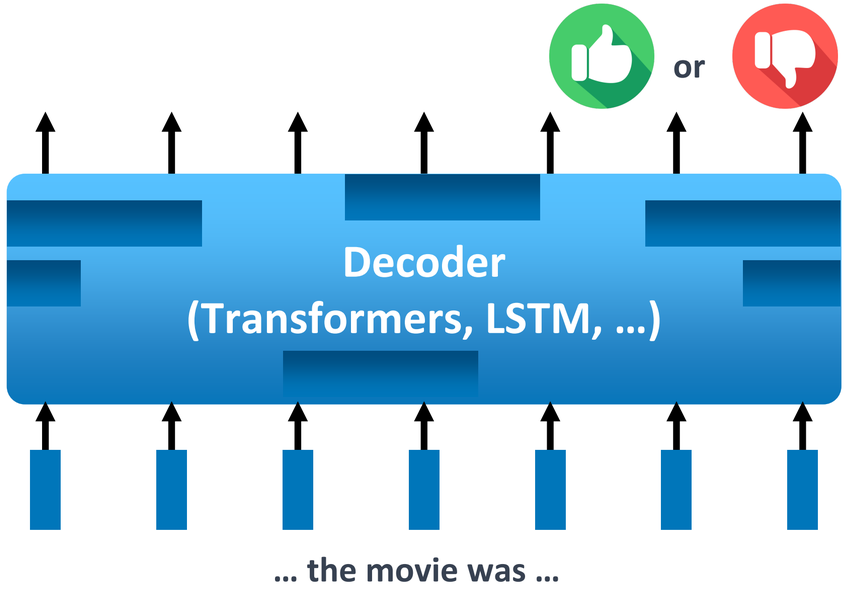
# Parameter-Efficient Fine-tuning

Instead of updating all parameters in the massive neural network (up to many billions of parameters), **can we make fine-tuning more efficient?**




**Full Fine-tuning**

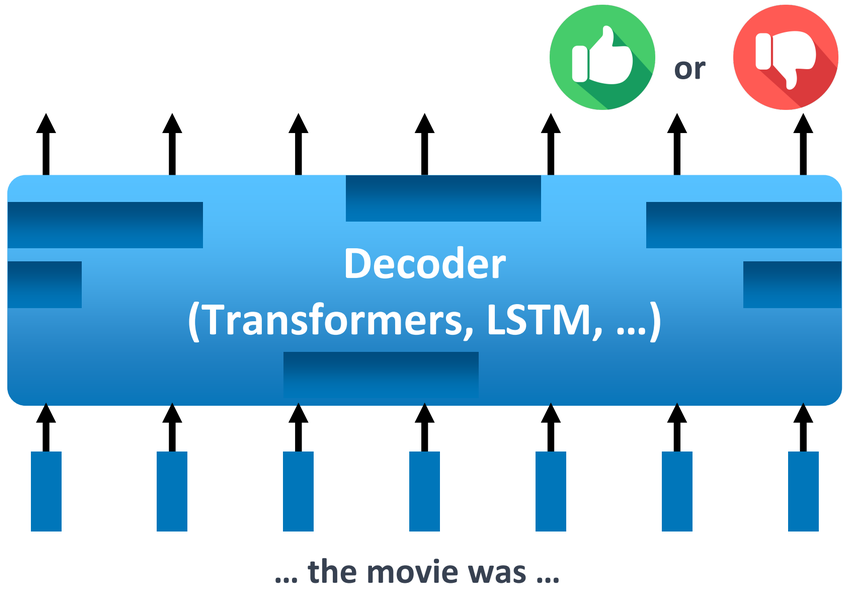
Updating all parameters




**Parameter-Efficient Fine-tuning**

Updating a few existing or new parameters

# Parameter-Efficient Fine-tuning



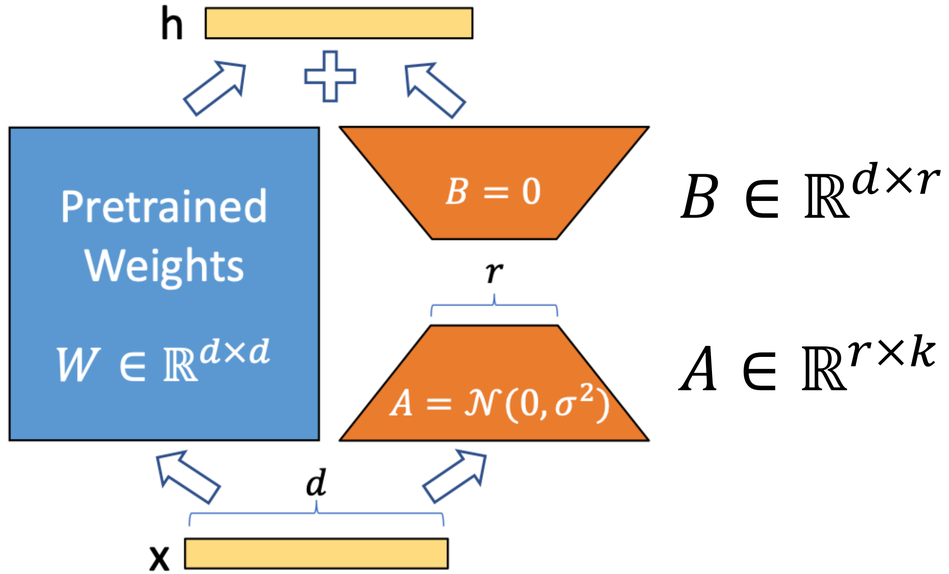

**Parameter-Efficient Fine-tuning**

Updating a few existing or new parameters

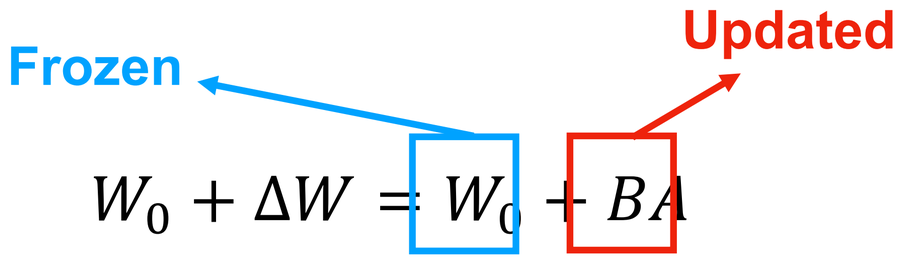
- **More efficient at fine-tuning & inference time**
- **Less overfitting** by keeping the majority of parameters learned during pre-training

# Low-Rank Adaptation (LoRA)

- **Main Idea:** learn a low-rank "diff" between the pre-trained and fine-tuned weight matrices.
- ~10,000x less fine-tuned parameters, ~3x GPU memory requirement.
- **On-par** or **better** than fine-tuning all model parameters in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3.
- **Easier** to learn than prefix-tuning.




where rank $r \ll min(d, k)$




[Hu et al., 2021]

# QLoRA: Efficient Finetuning of Quantized LLMs

**Tim Dettmers*** **Artidoro Pagnoni*** **Ari Holtzman**

**Luke Zettlemoyer**

University of Washington
{dettmers,artidoro,ahai,lsz}@cs.washington.edu

## Abstract

We present QLoRA, an efficient finetuning approach that reduces memory usage enough to finetune a 65B parameter model on a single 48GB GPU while preserving full 16-bit finetuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters (LoRA). Our best model family, which we name **Guanaco**, outperforms all previous openly released models on the Vicuna benchmark, reaching 99.3% of the performance level of ChatGPT while only requiring 24 hours of finetuning on a single GPU. QLoRA introduces a number of innovations to save memory without sacrificing performance: (a) 4-bit NormalFloat (NF4), a new data type that is information theoretically optimal for normally distributed weights (b) Double Quantization to reduce the average memory footprint by quantizing the quantization constants, and (c) Paged Optimizers to manage memory spikes. We use QLoRA to finetune more than 1,000 models, providing a detailed analysis of instruction following and chatbot performance across 8 instruction datasets, multiple model types (LLaMA, T5), and model scales that would be infeasible to run with regular finetuning (e.g. 33B and 65B parameter models). Our results show that QLoRA finetuning on a small high-quality dataset leads to state-of-the-art results, even when using smaller models than the previous SoTA. We provide a detailed analysis of chatbot performance based on both human and GPT-4 evaluations showing that GPT-4 evaluations are a cheap and reasonable alternative to human evaluation. Furthermore, we find that current chatbot benchmarks are not trustworthy to accurately evaluate the performance levels of chatbots. A lemon-picked analysis demonstrates where **Guanaco** fails compared to ChatGPT. We release all of our models and code, including CUDA kernels for 4-bit training.[2]