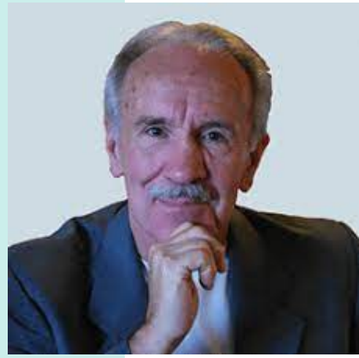
The Structure of Magic

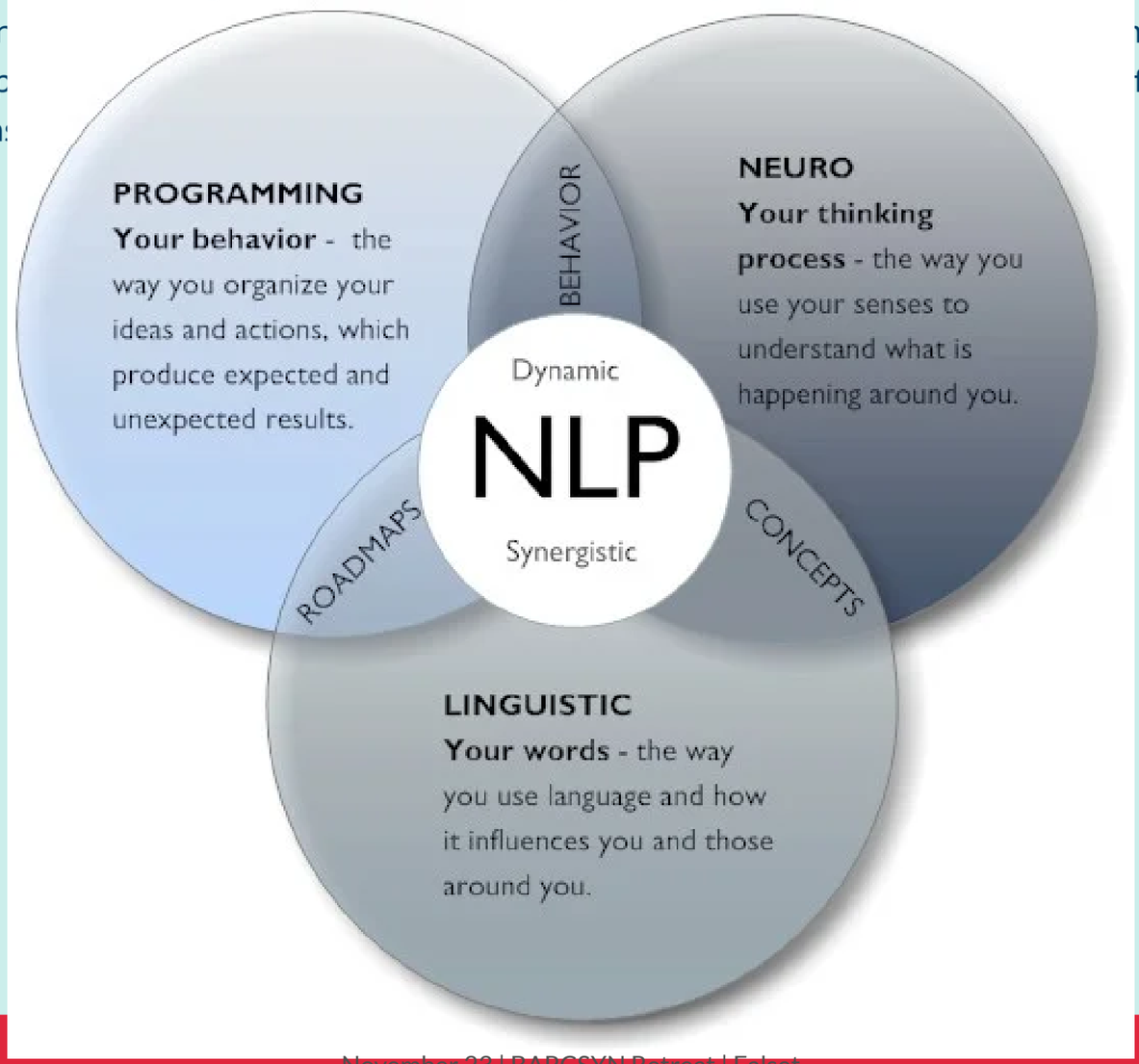
NLP in a Nutshell

NLP

Neuro-Linguistic Programming In A Nutshell

NLP was developed by Richard Bandler and John Grinder, who believed that the thoughts and behaviors of successful people could be taught to others. Neuro-linguistic programming is a means of changing the thoughts or behaviors of an ir ming (NLP) is based on the idea that p ften constrained by certain limitation





Natural Language Processing

Natural language processing (**NLP**) is an interdisciplinary subfield of computer science and linguistics. It is primarily concerned with giving computers the ability to support and manipulate human language. It involves processing natural language datasets, such as text corpora or speech corpora, using either rule-based or probabilistic (i.e. statistical and, most recently, neural network-based) machine learning approaches. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

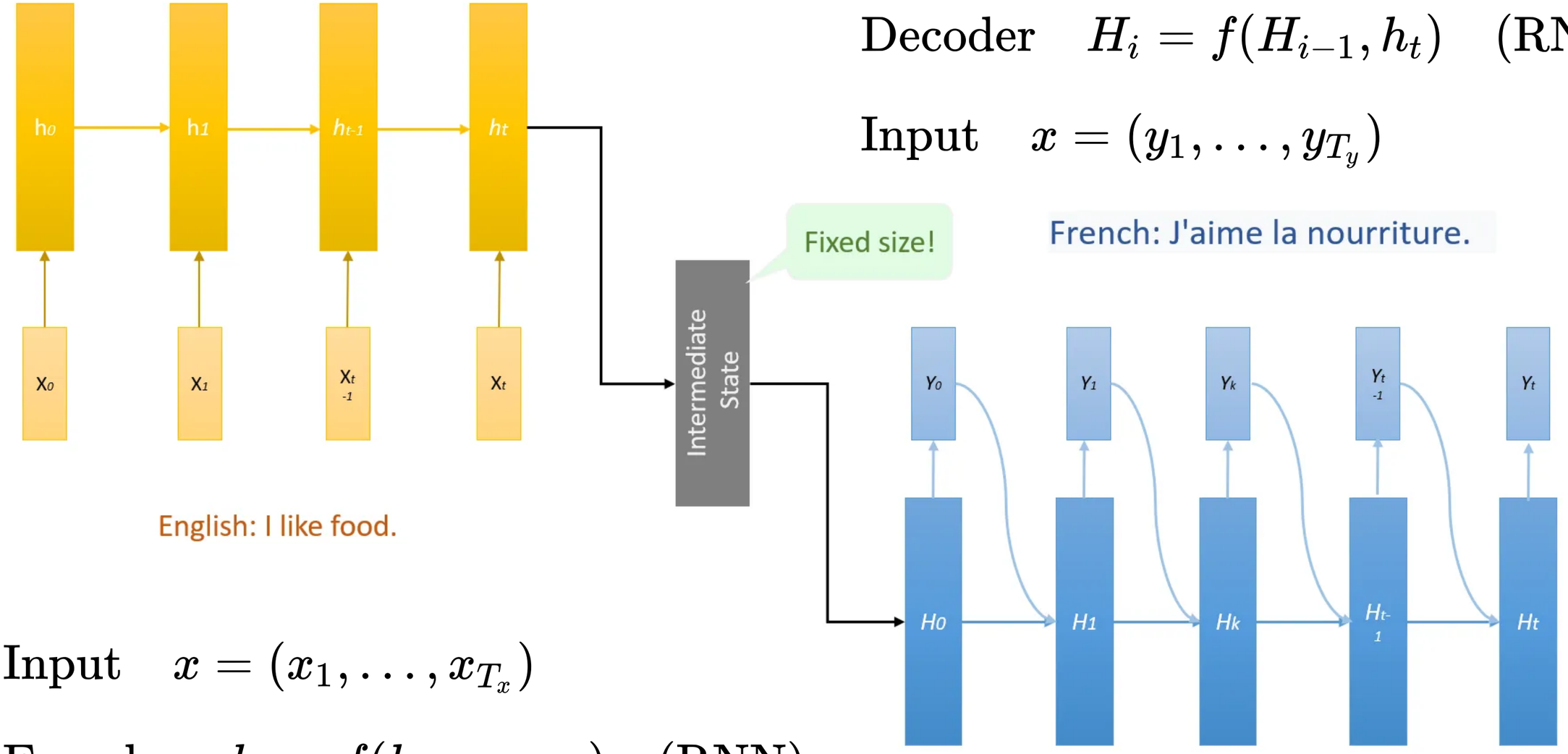
The Translation Problem

- Let’s go over a task — popularly solved in many NLP models — translation:
  - A word to word translation doesn’t work in most cases
  - as most languages don’t share a common sentence structure.

```
English => French
red => rouge
dress => robe
"red dress" => "robe rouge"
# Notice how red is before dress in English but rouge is after robe in French.
```

- The way NLP models go about is by capturing all the information in the input sentence in an intermediate state.

Capturing Relationships between objects



$$\text{Decoder} \quad H_i = f(H_{i-1}, h_t) \quad (\text{RNN})$$

$$\text{Input} \quad x = (y_1, \ldots, y_{T_y})$$

$$\text{Input} \quad x = (x_1, \ldots, x_{T_x})$$

$$\text{Encoder} \quad h_i = f(h_{i-1}, x_{i-1}) \quad (\text{RNN})$$

Long Input Sentences

> It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way — in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

A Tale of Two Cities, Charles Dickens.

Long Input Sentences

- Now could you memorize, not even translate, this sentence after going over it once?
  - Not so easy, is it?
- With very long sentences as input, the intermediate state fails and is not sufficient to capture all the information.

Modeling Attention

An encoder reads an input sentence $x = (x_1, \ldots, x_{T_x})$ into a vector c.

The most common approach is to use an RNN such that

$$h_t = f\left(x_{t-1}, h_{t-1}\right)$$

and

$$c = q\left(\{h_1, \ldots, h_{T_x}\}\right)$$

Modeling Attention

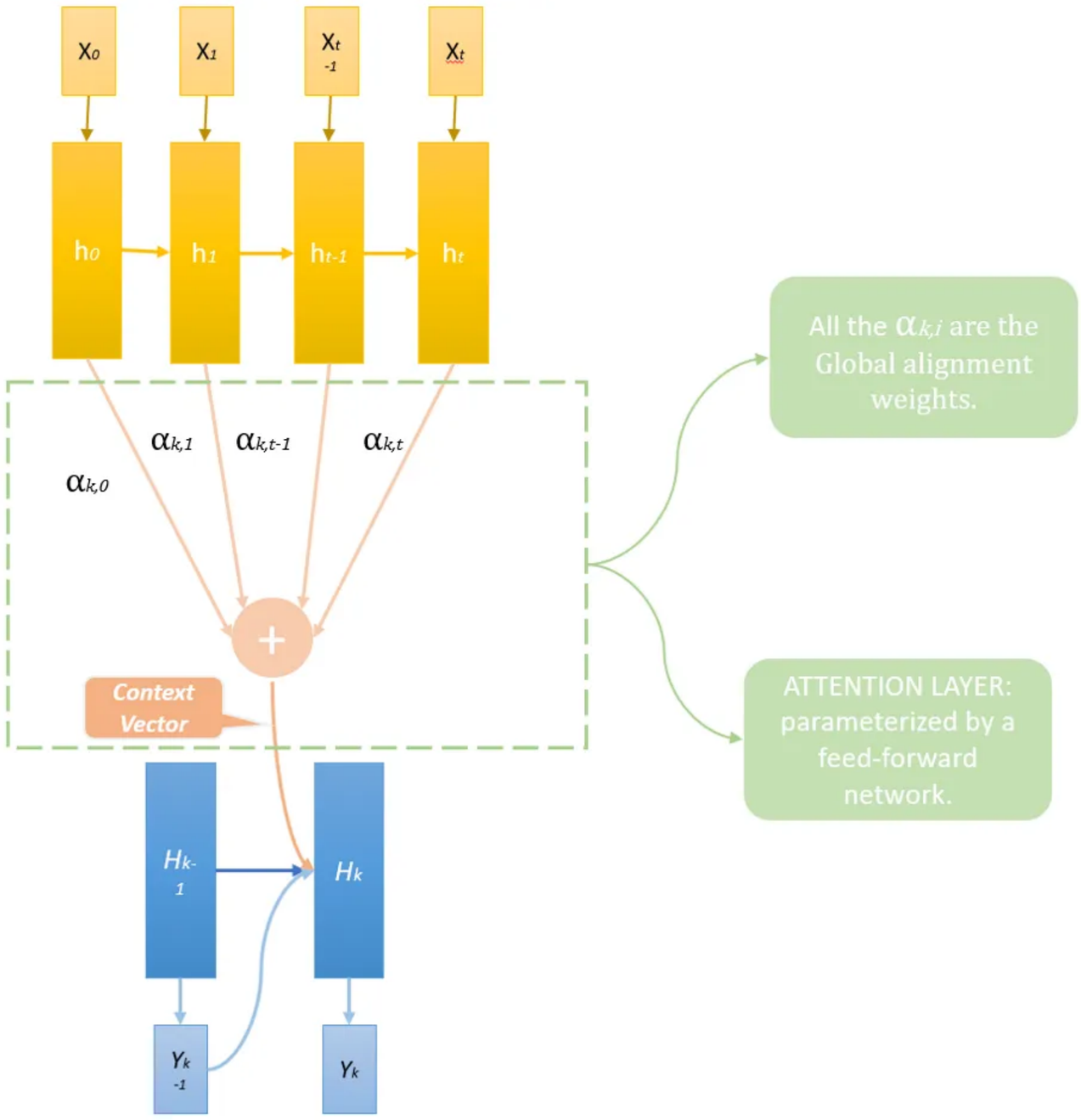
The decoder is then trained to predict the next word y_t, given a context vector c and all the previously predicted words, $\{y_1, \ldots, y_{t-1}\}$.

This is equivalent to defining a probability over the translation, y, by decomposing the joint probability into ordered conditionals:

$$p(y) = \Pi_{t=1}^{T} p(y_t | y_1, \ldots, y_{t-1}, c)$$

where each conditional probability is modeled as

$$p(y_t | y_1, \ldots, y_{t-1}, c) = g(y_{t-1}, H_t, c)$$



Input $x = (x_1, \ldots, x_{T_x})$

Encoder $h_i = f(h_{i-1}, x_{i-1})$ (RN

Context $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$

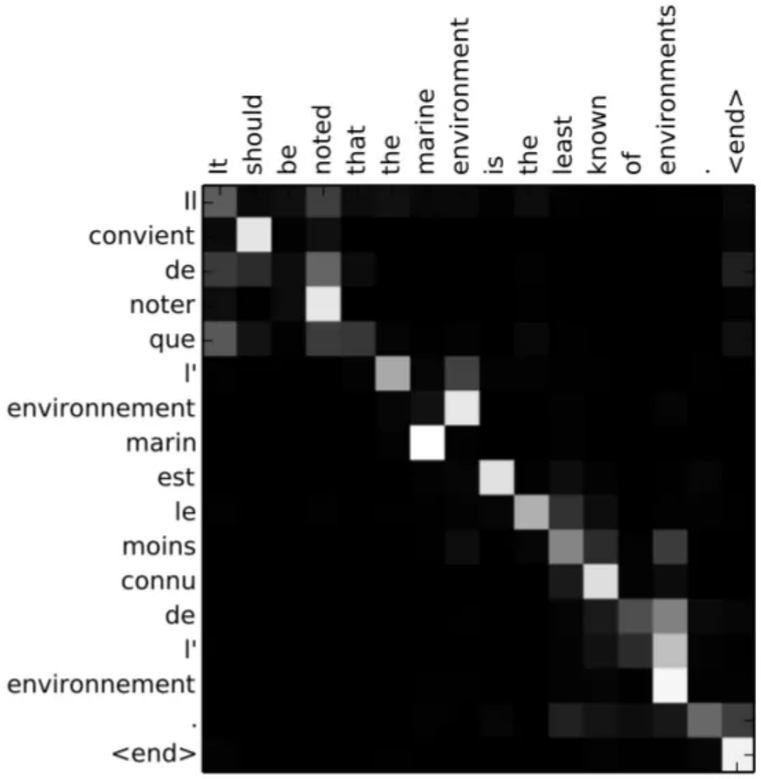
Alignement $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j=1}^{T_x} \exp(e_{ik})}$

Energy $e_{ij} = a(H_{i-1}, h_j)$ (FFN

Decoder $H_i = f(H_{i-1}, y_{i-1}, c_i)$(R

Output $y = (y_1, \ldots, y_{T_y})$

Global Alignement



The global alignment weights tell us which annotations to focus on for the next output.

- While predicting the next step, weights are high only for a few words at a time.
- Attention can return to an input word — look at the word “que” in output and how a part of its attention is on the first word “It”.
- Attention models outperform conventional encoder–decoder model significantly.
- The model correctly align each target word with the relevant words, or their annotations, in the source sentence as it generated a correct translation.

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

What is a Transformer?

- The Transformer in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease.
- It relies entirely on **self-attention** to compute representations of its input and output **WITHOUT** using sequence-aligned RNNs or convolution.
- Attention allowed us to focus on parts of our input sequence while we predicted our output sequence.
- Self attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

Self Attention

Self attention helps us create similar connections but within the same sentence. Look at the following example:

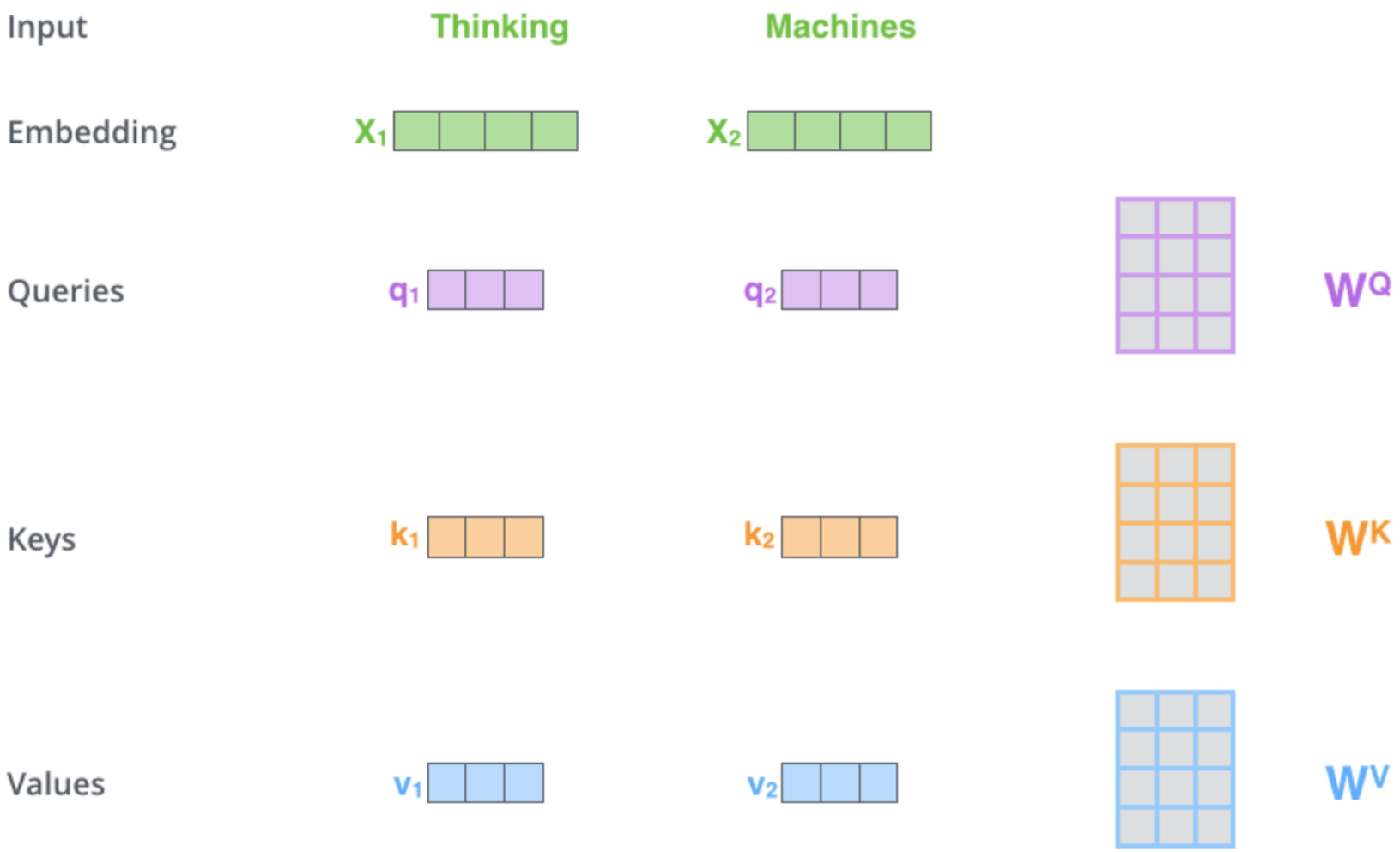
```
1 “I poured water from the bottle into the cup until it was full.”
2 it => cup
3 “I poured water from the bottle into the cup until it was empty.”
4 it=> bottle
```

By changing one word “full” — > “empty” the reference object for “it” changed. If we are translating such a sentence, we will want to know to what the word “it” refers to.

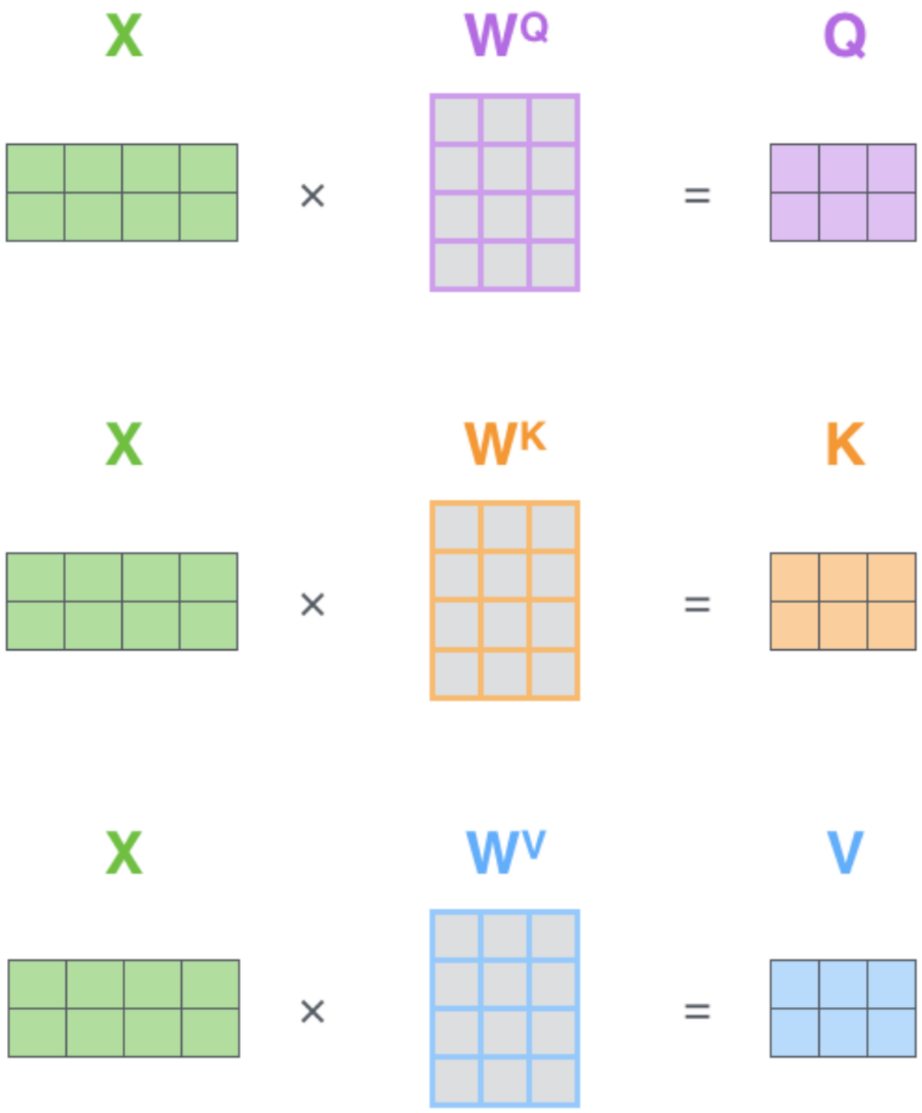
Self Attention

- The three kinds of Attention possible in a model:
  - Encoder-Decoder Attention: Attention between the input sequence and the output sequence.
  - Self attention in the input sequence: Attends to all the words in the input sequence.
  - Self attention in the output sequence: Self attention is limited to the words that occur before a given word. This prevents any information leaks during the training of the model.

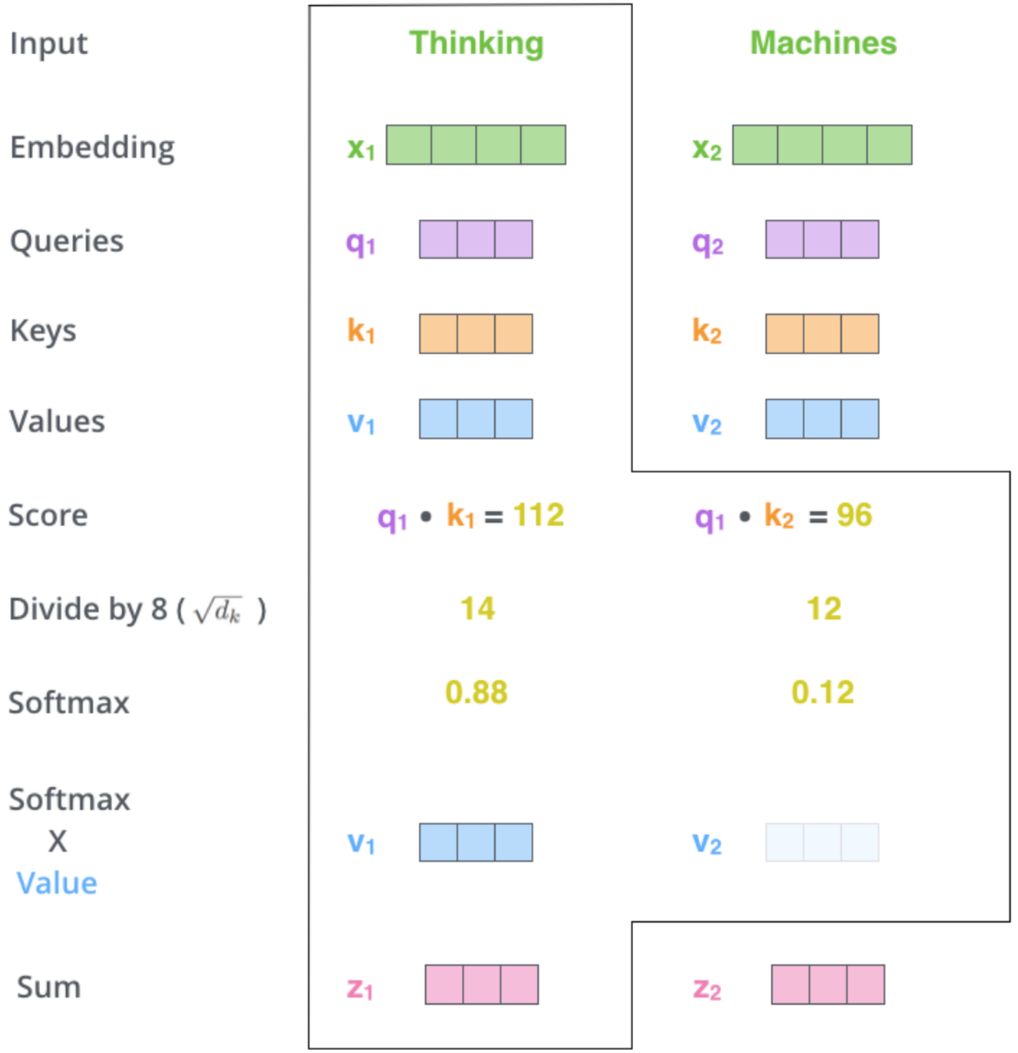
Keys, Values and Queries



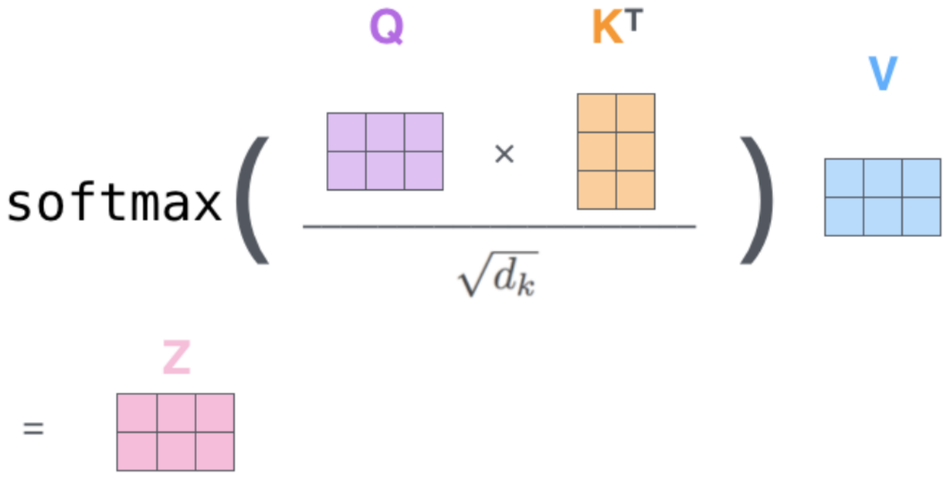
Keys, Values and Queries



Calculating Self Attention



Self Attention



$$Step\ 1 = q_i \cdot k_j \qquad for\ all\ 0 \leqq j \leqq n$$

$$Step\ 2 = \frac{q_i \cdot k_j}{\sqrt{dim(k_j)}} \qquad for\ all\ 0 \leqq j \leqq n$$

$$Step\ 3 = softmax\left(\frac{q_i \cdot k_j}{\sqrt{dim(k_j)}}\right) \qquad for\ all\ 0 \leqq j \leqq n$$

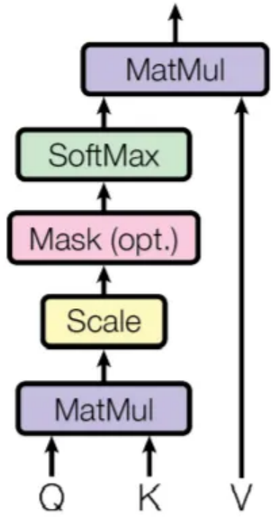
$$Step\ 4 = \left\{softmax\left(\frac{q_i \cdot k_j}{\sqrt{dim(k_j)}}\right) \cdot v_i\right\} \qquad for\ all\ 0 \leqq j \leqq n$$

$$Finally : z_i = \sum_{j=0}^{n}\left(softmax\left(\frac{q_i \cdot k_j}{\sqrt{dim(k_j)}}\right) \cdot v_i\right)$$

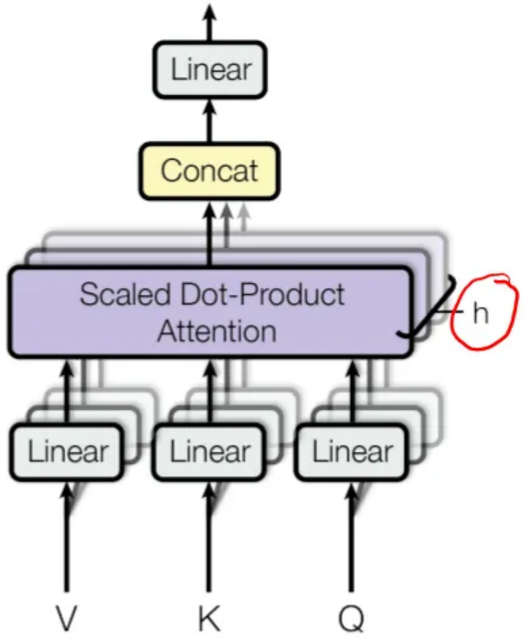
Multihead Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention



Multi-Head Attention



NOTICE!

Multihead Attention

1) This is our input sentence*

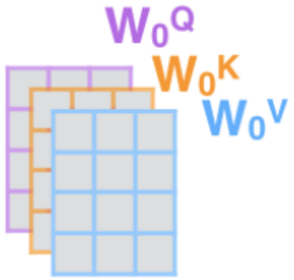
2) We embed each word*

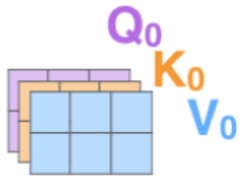
3) Split into 8 heads. We multiply X or R with weight matrices

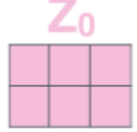
4) Calculate attention using the resulting Q/K/V matrices

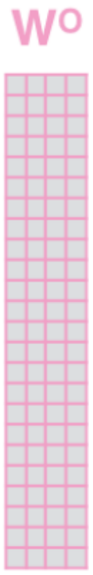
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

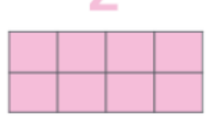
Thinking Machines

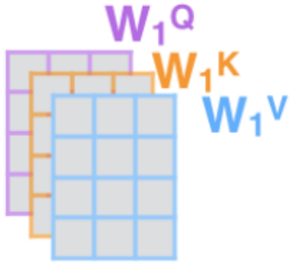


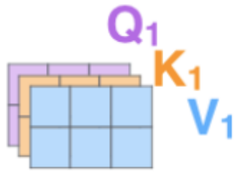


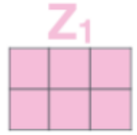




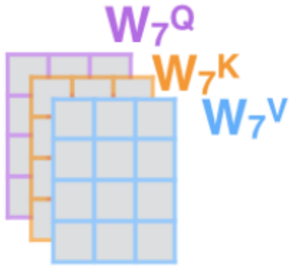


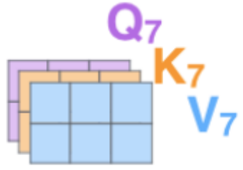


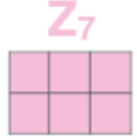




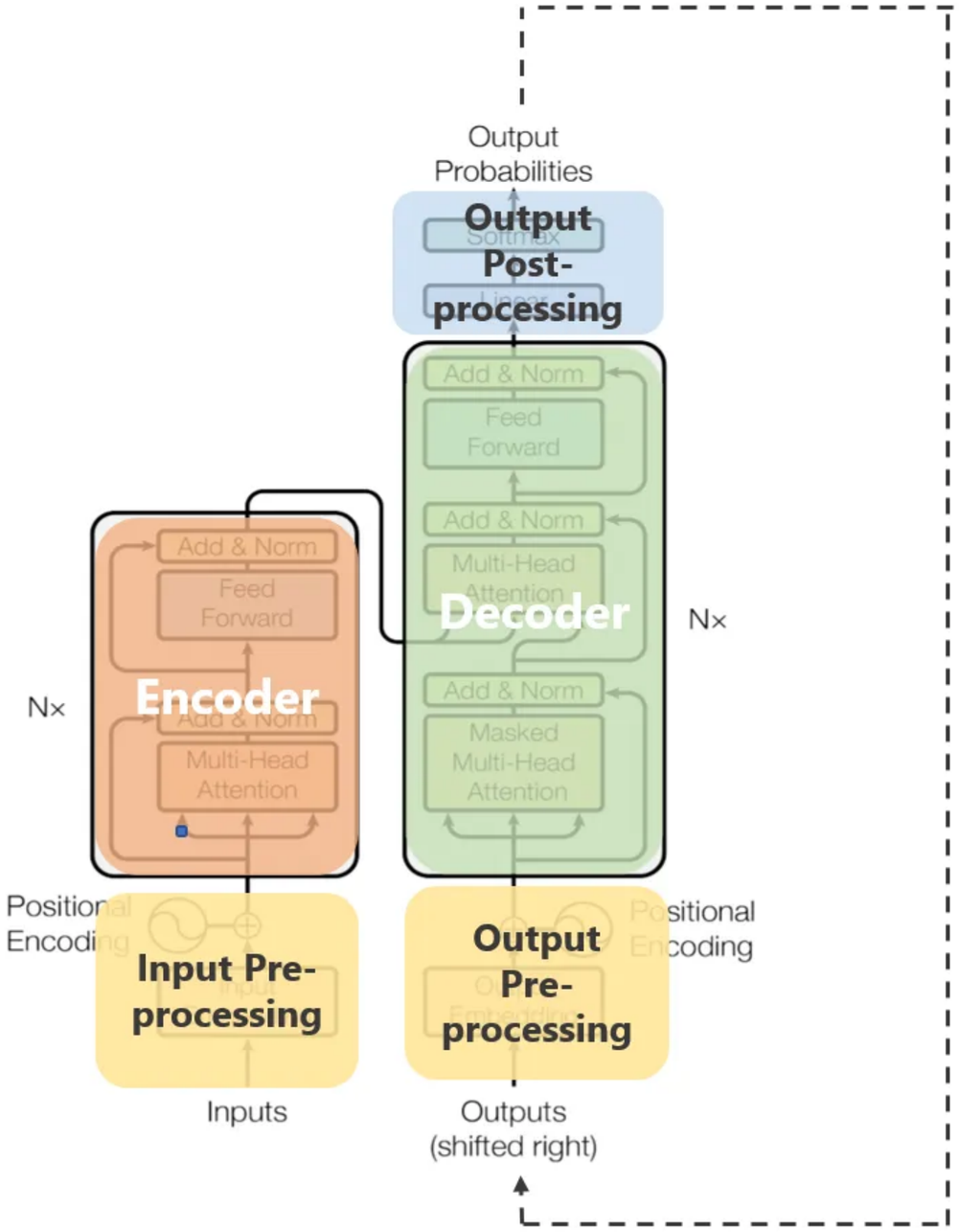
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



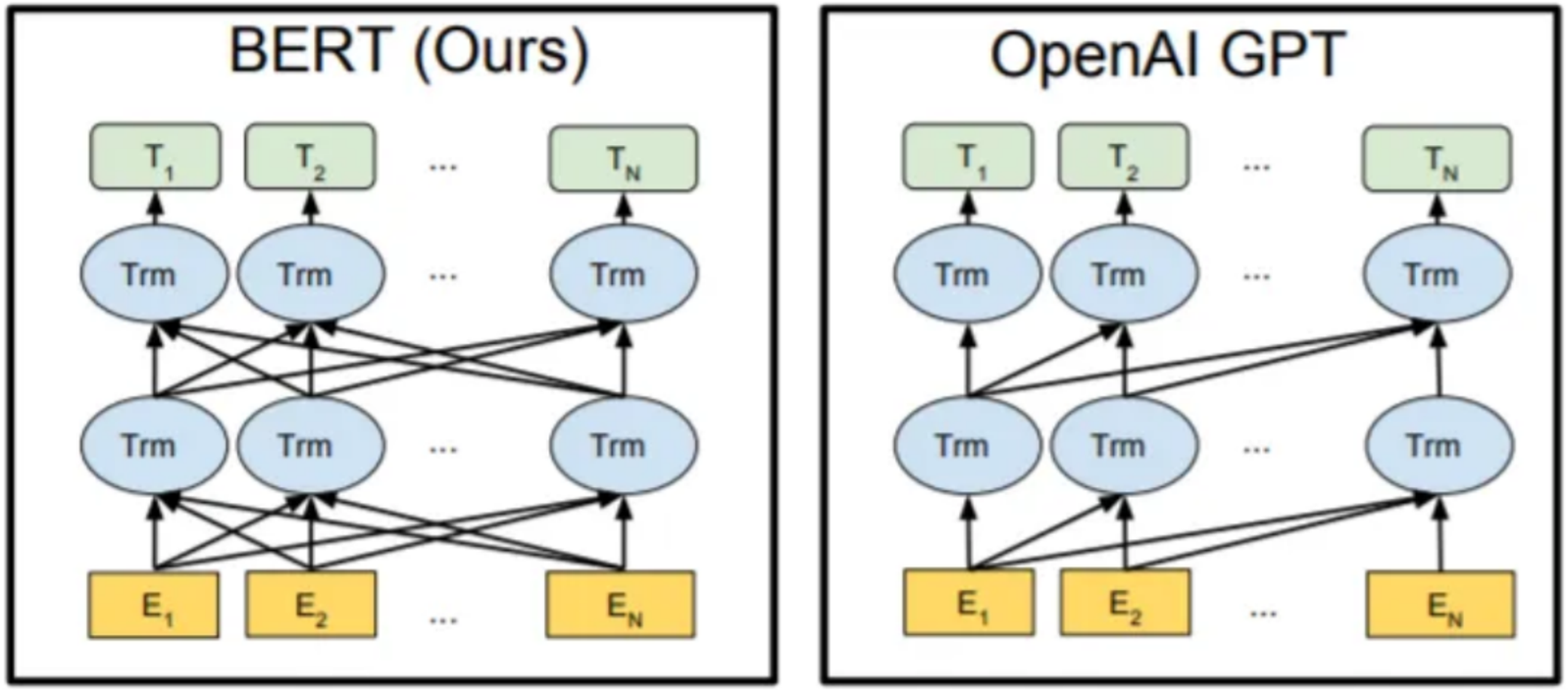




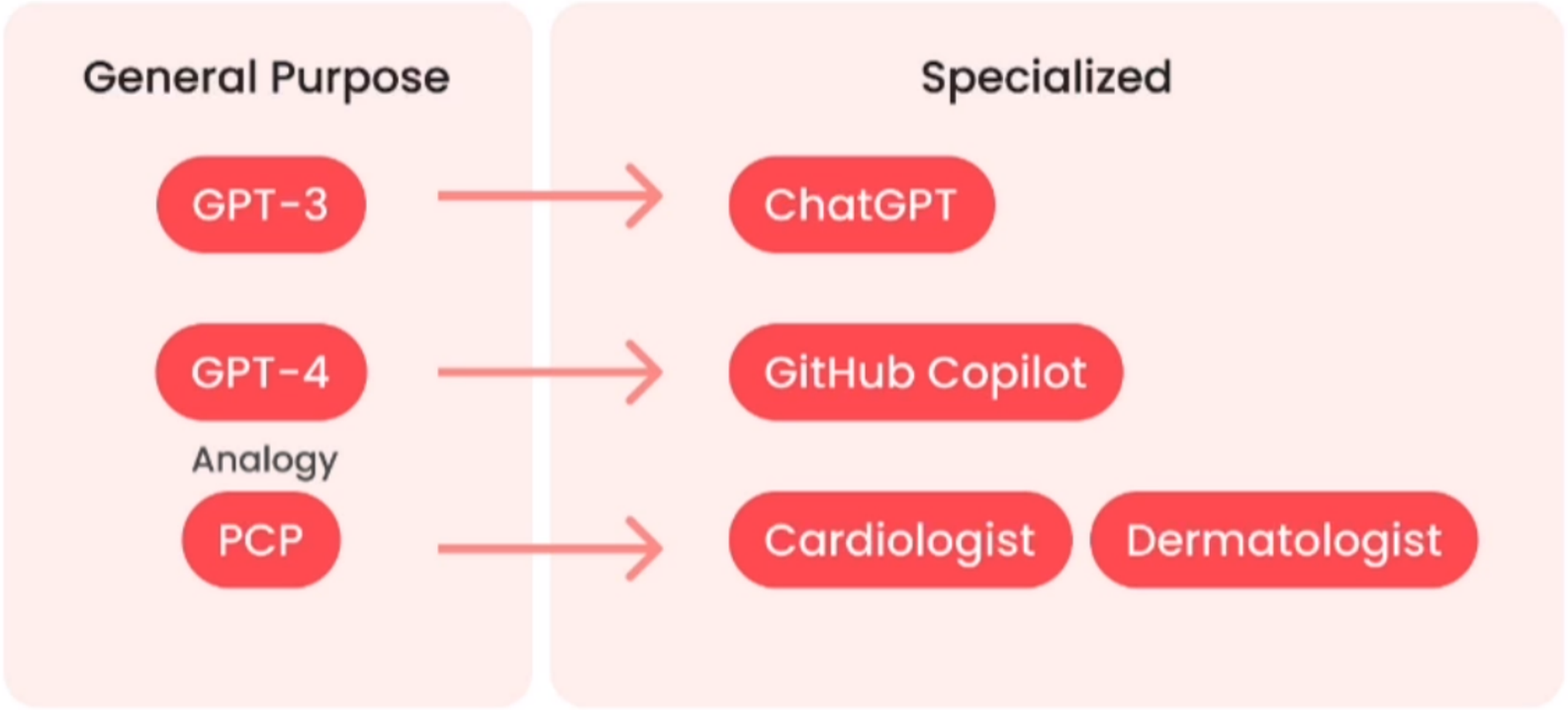
The Full Model



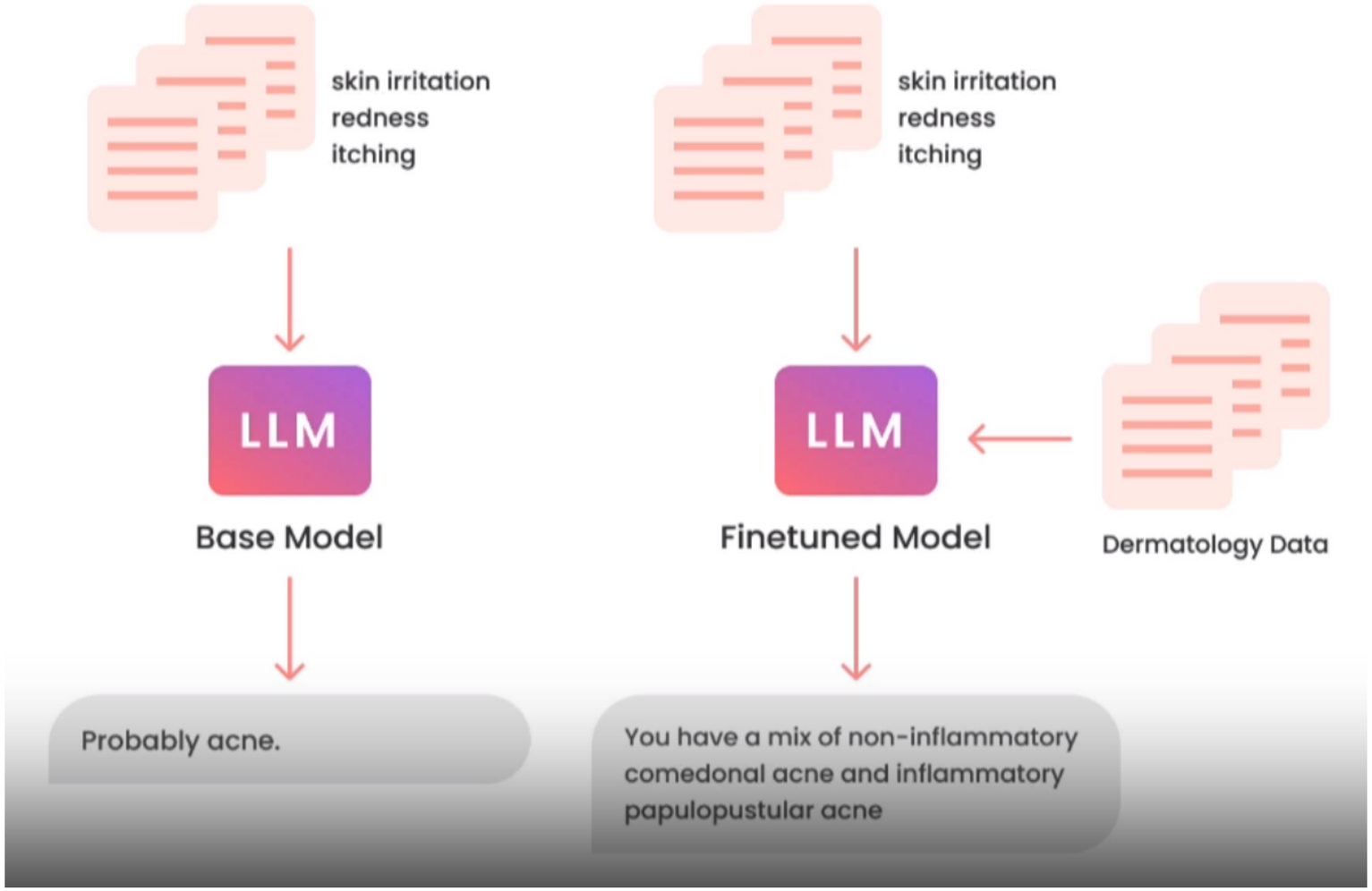
Some Current LLMs



What is Fine Tuning?



What is Fine Tuning?



Why Fine Tuning?

Prompt Engineering vs. Finetuning

	Prompting	Finetuning
Pros	• No data to get started • Smaller upfront cost • No technical knowledge needed • Connect data through retrieval (RAG)	• Nearly unlimited data fits • Learn new information • Correct incorrect information • Less cost afterwards if smaller model • Use RAG too
Cons	• Much less data fits • Forgets data • Hallucinations • RAG misses, or gets incorrect data	• More high-quality data • Upfront compute cost • Needs some technical knowledge, esp. data
	Generic, side projects, prototypes	Domain-specific, enterprise, production usage, ...privacy!

Examples

- SciGPT
- Hugging Face Hub
- HF Autotrain
- Galactica
- Instruct GPT
- SciBert

Thank You