Transformers

CS60010

Transformers

Attention Is All You Need

(2017)

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

Great Results with Transformers: Rise of LLMs

Today, Transformer-based models dominate LMSYS Chatbot Arena Leaderboard!

Rank	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	GPT-4-Turbo-2024-04-09	1258	+4/-4	26444	OpenAI	Proprietary	2023/12
1	GPT-4-1106-preview	1253	+3/-3	68353	OpenAI	Proprietary	2023/4
1	Claude 3 Opus	1251	+3/-3	71500	Anthropic	Proprietary	2023/8
2	Gemini 1.5 Pro API-0409-Preview	1249	+4/-5	22211	Google	Proprietary	2023/11
3	GPT-4-0125-preview	1248	+2/-3	58959	OpenAI	Proprietary	2023/12
6	Meta Llama 3 70b Instruct	1213	+4/-6	15809	Meta	Llama 3 Community	2023/12
6	Bard (Gemini Pro)	1208	+7/-6	12435	Google	Proprietary	Online
7	Claude 3 Sonnet	1201	+4/-2	73414	Anthropic	Proprietary	2023/8

Gemini / Bard (Google)

ChatGPT / GPT-4 (OpenAI)

Claude 3 (Anthropic)

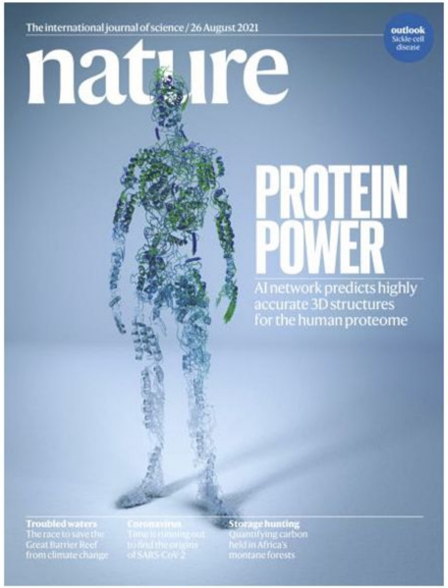
Llama 3 (Meta)

[Chiang et al., 2024]

https://web.stanford.edu/class/cs224n/

Transformers have shown promise outside NLP

Protein Folding



[Jumper et al. 2021] aka AlphaFold2!

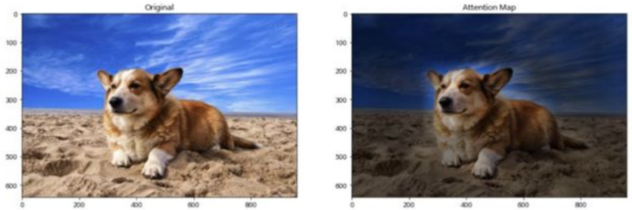
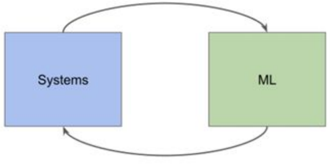
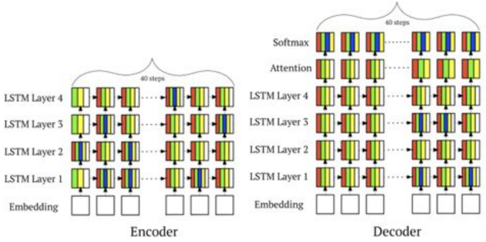


Image Classification

[Dosovitskiy et al. 2020]: Vision Transformer (ViT) outperforms ResNet-based baselines with substantially less compute.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k





ML for Systems

[Zhou et al. 2020]: A Transformer-based compiler model (GO-one) speeds up a Transformer model!

Model (#devices)	**GO-one** (s)	HP (s)	METIS (s)	HDP (s)	Run time speed up over HP / HDP	Search speed up over HDP
2-layer RNNLM (2)	0.173	0.192	0.355	0.191	9.9% / 9.4%	2.95x
4-layer RNNLM (4)	0.210	0.239	0.503	0.251	13.8% / 16.3%	1.76x
8-layer RNNLM (8)	0.320	0.332	OOM	0.764	3.8% / 58.1%	27.8x
2-layer GNMT (2)	0.301	0.384	0.344	0.327	27.6% / 14.3%	30x
4-layer GNMT (4)	0.350	0.469	0.466	0.432	34% / 23.4%	58.8x
8-layer GNMT (8)	0.440	0.562	OOM	0.693	21.7% / 36.5%	7.35x
2-layer Transformer-XL (2)	0.223	0.268	0.37	0.262	20.1% / 17.4%	40x
4-layer Transformer-XL (4)	0.230	0.27	OOM	0.259	17.4% / 12.6%	26.7x
8-layer Transformer-XL (8)	0.350	0.46	OOM	0.425	23.9% / 16.7%	16.7x
Inception (2) b32	0.229	0.312	OOM	0.301	26.6% / 23.9%	13.5x
Inception (2) b64	0.423	0.731	OOM	0.498	42.1% / 29.3%	21.0x
AmoebaNet (4)	0.394	0.44	0.426	0.418	26.1% / 6.1%	58.8x
2-stack 18-layer WaveNet (2)	0.317	0.376	OOM	0.354	18.6% / 11.7%	6.67x
4-stack 36-layer WaveNet (4)	0.659	0.988	OOM	0.721	50% / 9.4%	20x
GEOMEAN	-	-	-	-	**20.5% / 18.2%**	**15x**

https://web.stanford.edu/class/cs224n/

Recap: Attention is a general technique

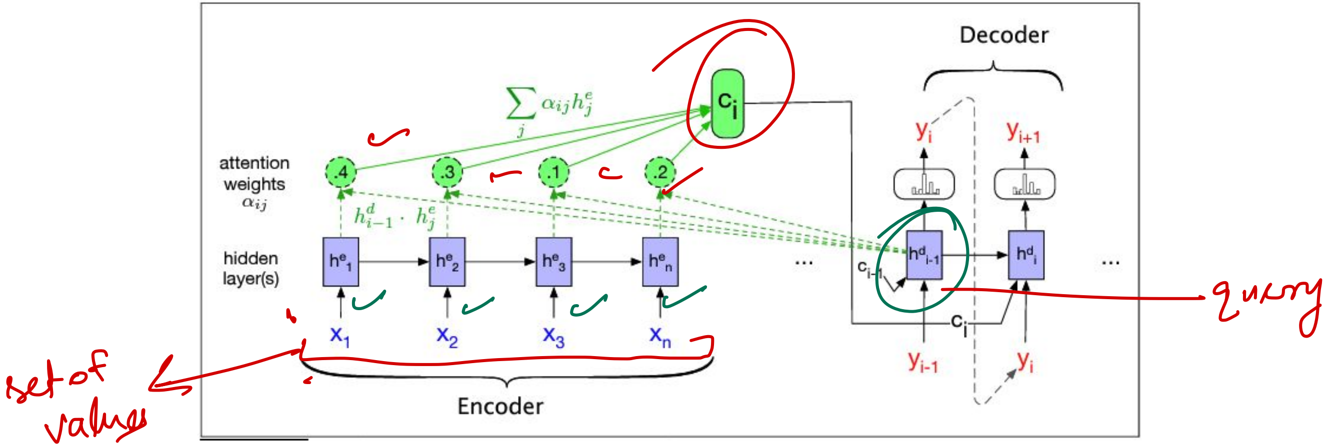
We can use attention in many architectures not just seq2seq and many tasks (not just MT)

More general definition of attention:

➔ Given a set of vector **values**, and a vector **query**, **attention** is a technique to compute a weighted sum of the values, dependent on the query

We say that the **query attends to the values**

In the seq2seq+attention model, each decoder hidden state (query) *attends to* all the encoder hidden state (values)



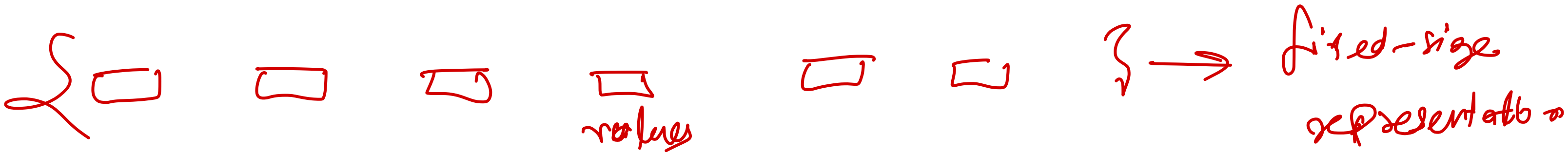
Attention is a general deep learning technique

More general definition of attention:

➔ Given a set of vector **values**, and a vector **query**, **<u>attention</u>** is a technique to compute a weighted sum of the values, dependent on the query

Intuition:

- The weighted sum is a **selective summary** of the information contain in the values, where the query determines which values to focus on
- Attention is a way to obtain a **fixed-size representation of an arbitrary set of representations**, *dependent on some other representation*

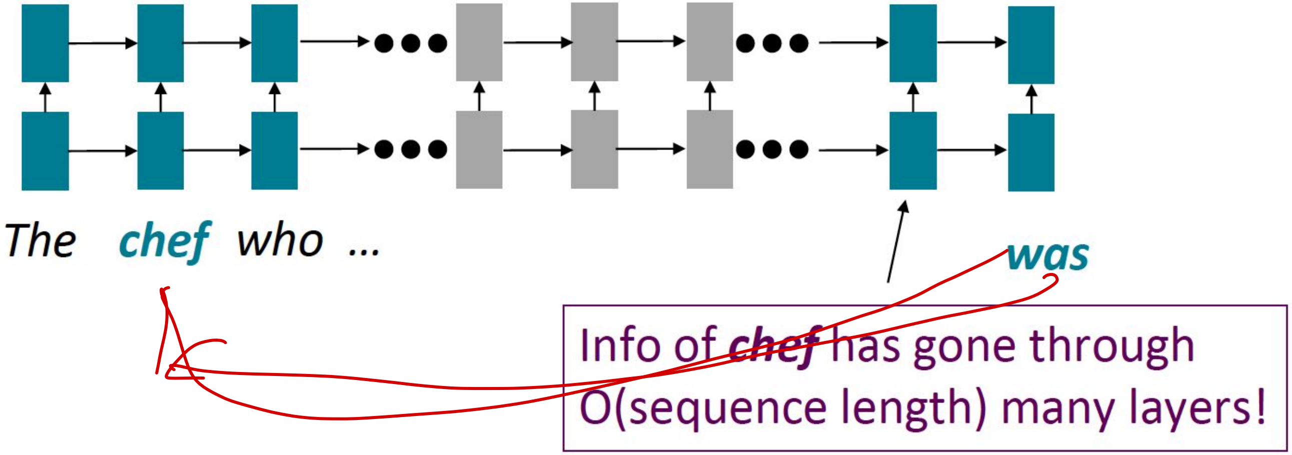


https://utah-cs6340-nlp.notion.site/Natural-Language-Processing-bd1a2ca290fc44f69556908ad8d25c70

Issues with recurrent models

O(sequence length) steps for distant word pairs

- Hard to learn long-distance dependencies (gradient problems!)
- Linear order of words is “baked in”; not the right way to think about sentences

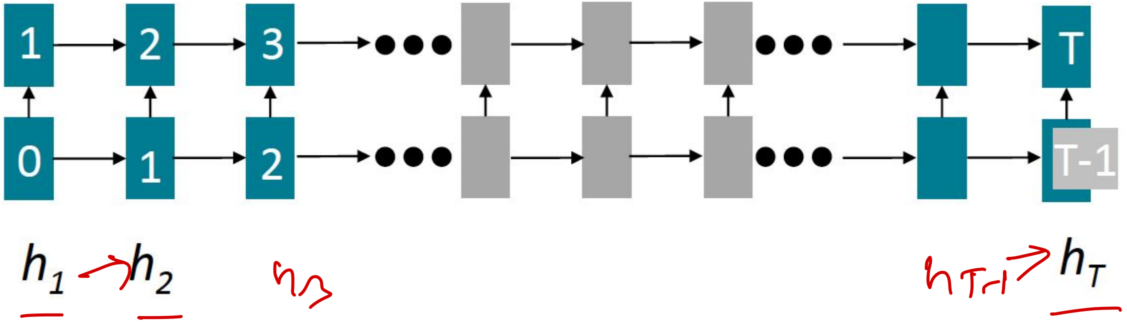


https://web.stanford.edu/class/cs224n/

Issues with recurrent models

Lack of parellilizability

- Future RNN hidden states can't be computed in full before past RNN hidden states have been computed



Inhibits training on very large datasets!

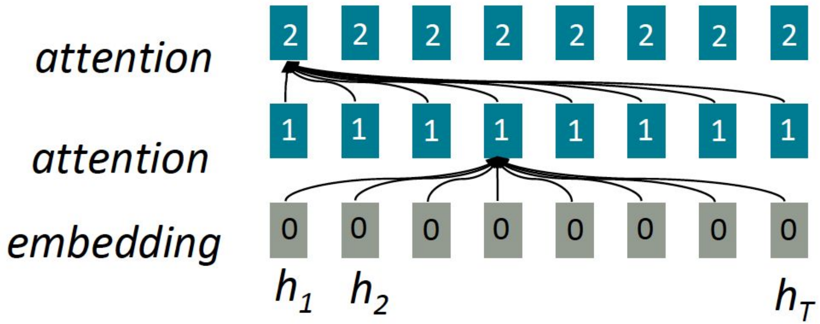
Numbers indicate min # of steps before a state can be computed

https://web.stanford.edu/class/cs224n/

If not recurrence, then what?

Attention

- Given a word as query, attention can be used to access and incorporate information from a set of values (other words)
- Can we do this within a single sentence?
- All words can interact with each other and computation can be done in parallel!!



All words attend to all words in previous layer; most arrows here are omitted

https://web.stanford.edu/class/cs224n/

Transformer Encoder-Decoder

- Transformer is introduced as an **encoder-decoder architecture**; later we will see **encoder-only** & **decoder-only** transformers
- **Encoder** produces a sophisticated representation of the source sequence that the decoder will use to condition its generation process
- **Decoder** generates one token at the time to produce a target sequence; it produces representations that combine the history and a new token

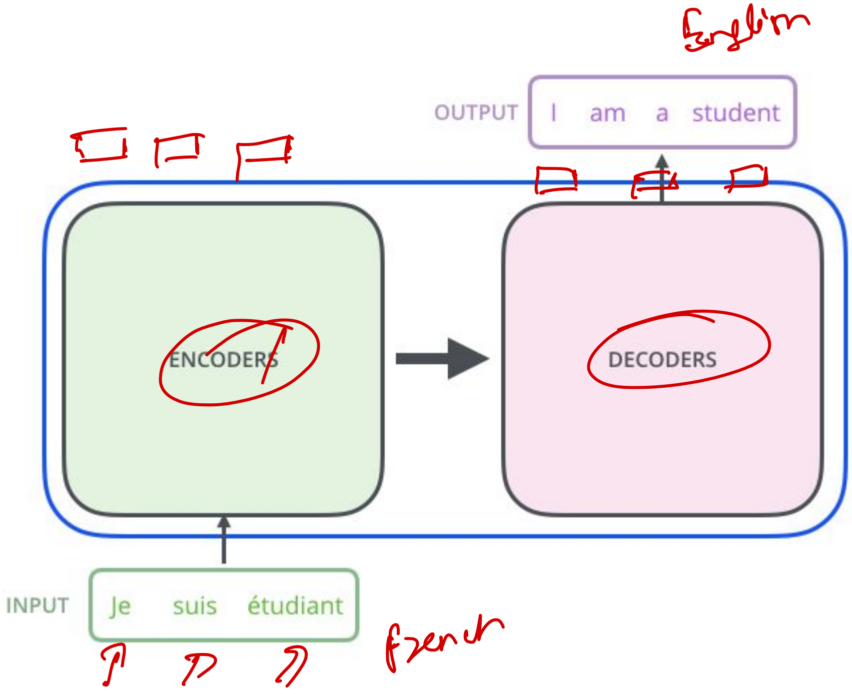
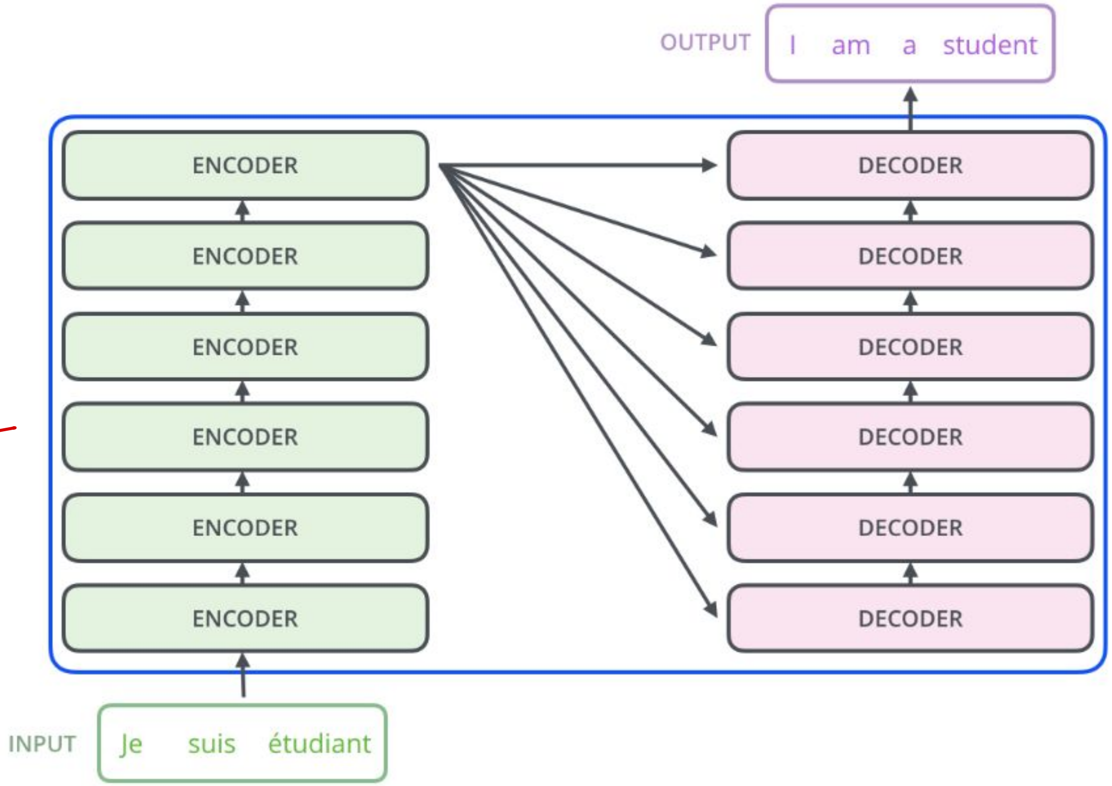


Figure: Jay Alammar

https://utah-cs6340-nlp.notion.site/Natural-Language-Processing-bd1a2ca290fc44f69556908ad8d25c70

A stack of encoder blocks



A stack of decoder blocks

Figure: [Jay Alammar]

Each encoder block consists of **self-attention & FFNN**

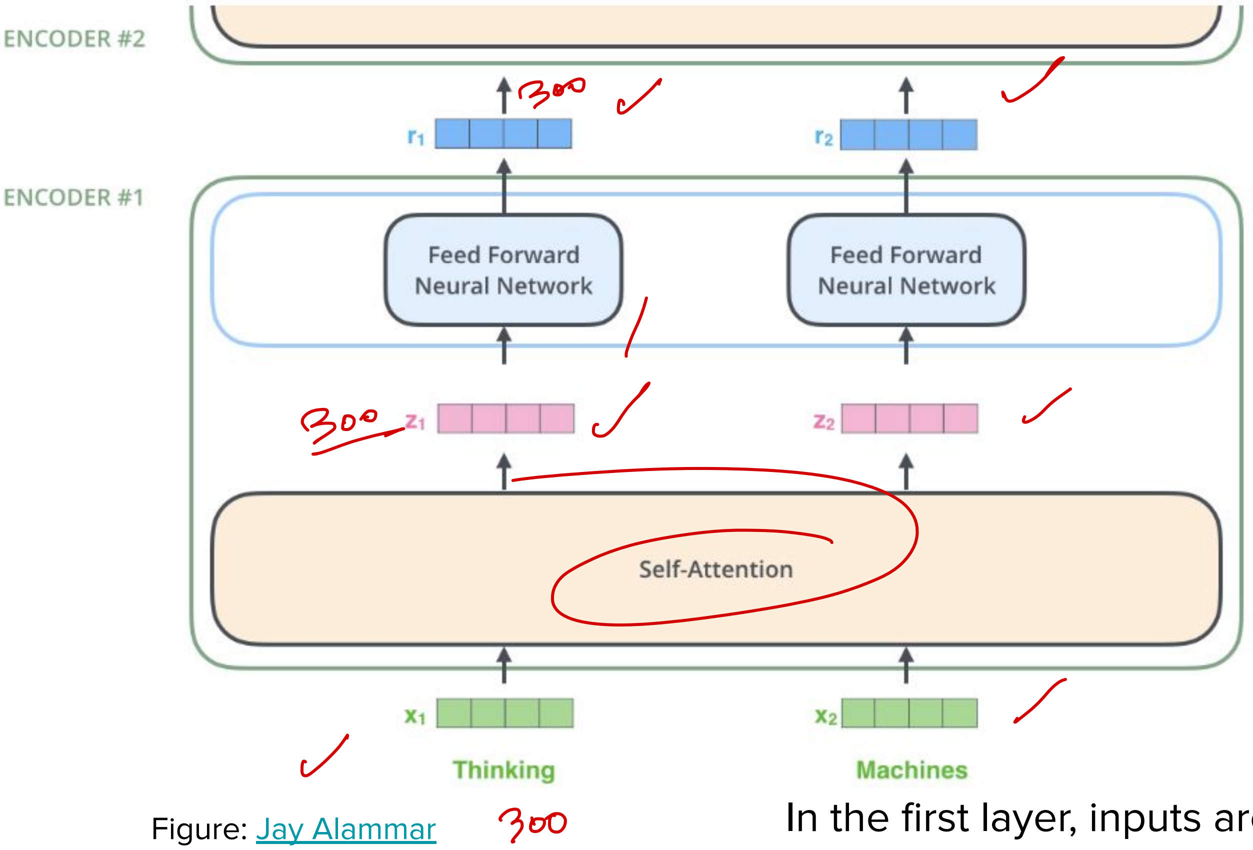


Figure: [Jay Alammar]()

Deeper layers get outputs of the previous layers as inputs

Input to each encoder block has the same size as original token embeddings

In the first layer, inputs are static token emeddings

Intuition for Self-Attention

Problem with static embeddings (word2vec)

They are static! The embedding for a word doesn't reflect how its meaning changes in context.

```
The chicken didn't cross the road because it was too tired
```

What is the meaning represented in the static embedding for "it"?

https://web.stanford.edu/~jurafsky/slp3/

Intuition for attention

`The chicken didn't cross the road because it`

Contextual representation

What should be the properties of "it"?

`The chicken didn't cross the road because it was too` **`tired`**

`The chicken didn't cross the road because it was too` **`wide`**

At this point in the sentence, it's probably referring to either the animal or the street

https://web.stanford.edu/~jurafsky/slp3/

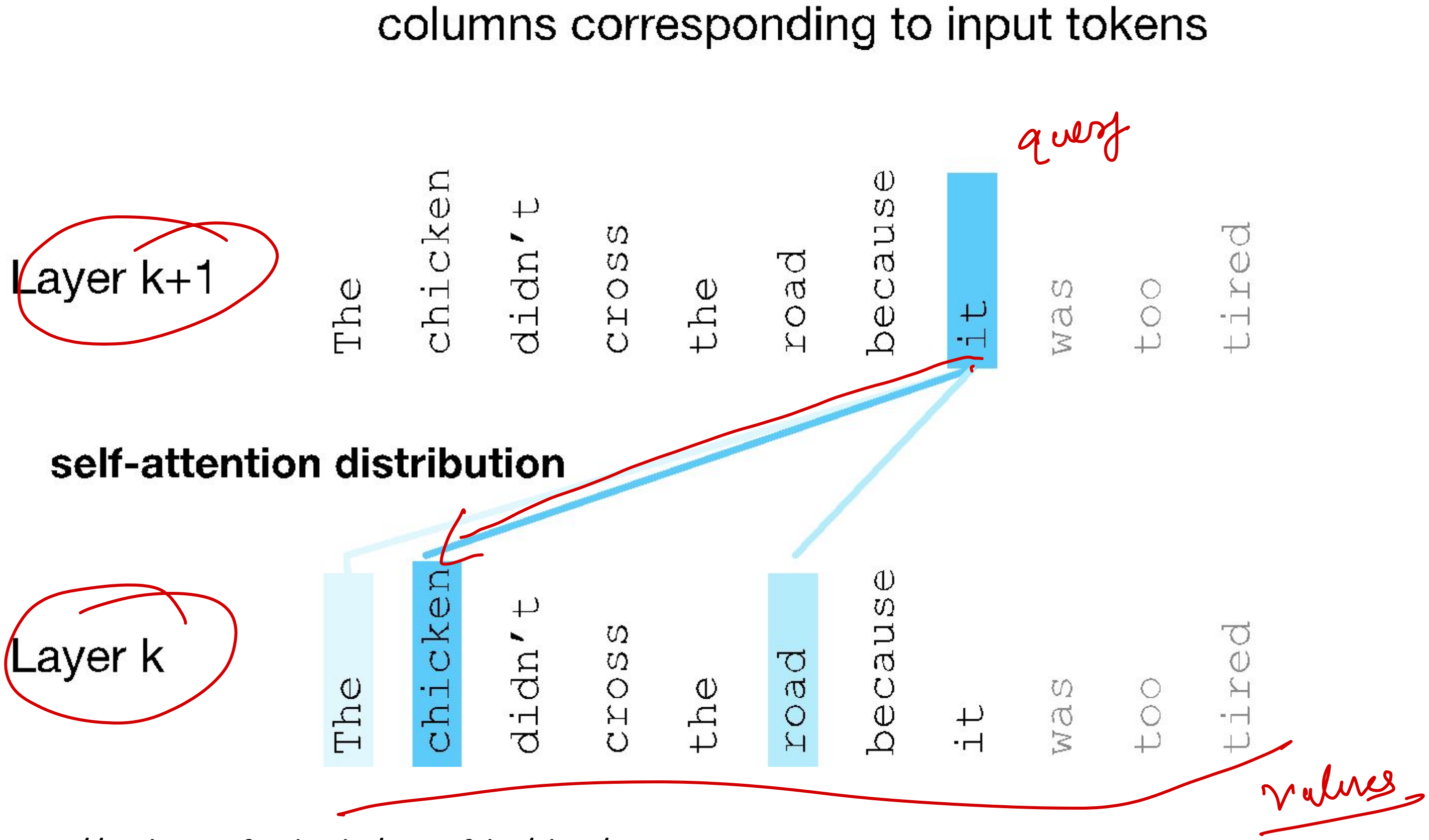
Intuition of attention

Build up the contextual embedding from a word by selectively integrating information from all the neighboring words

We say that a word "attends to" some neighboring words more than others

https://web.stanford.edu/~jurafsky/slp3/

Intuition of attention



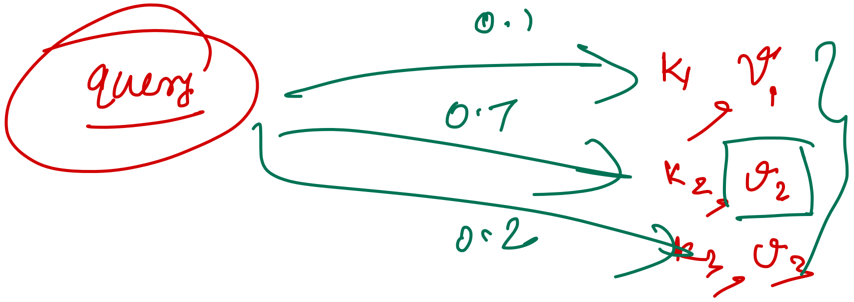
https://web.stanford.edu/~jurafsky/slp3/

Intuition for Self-attention

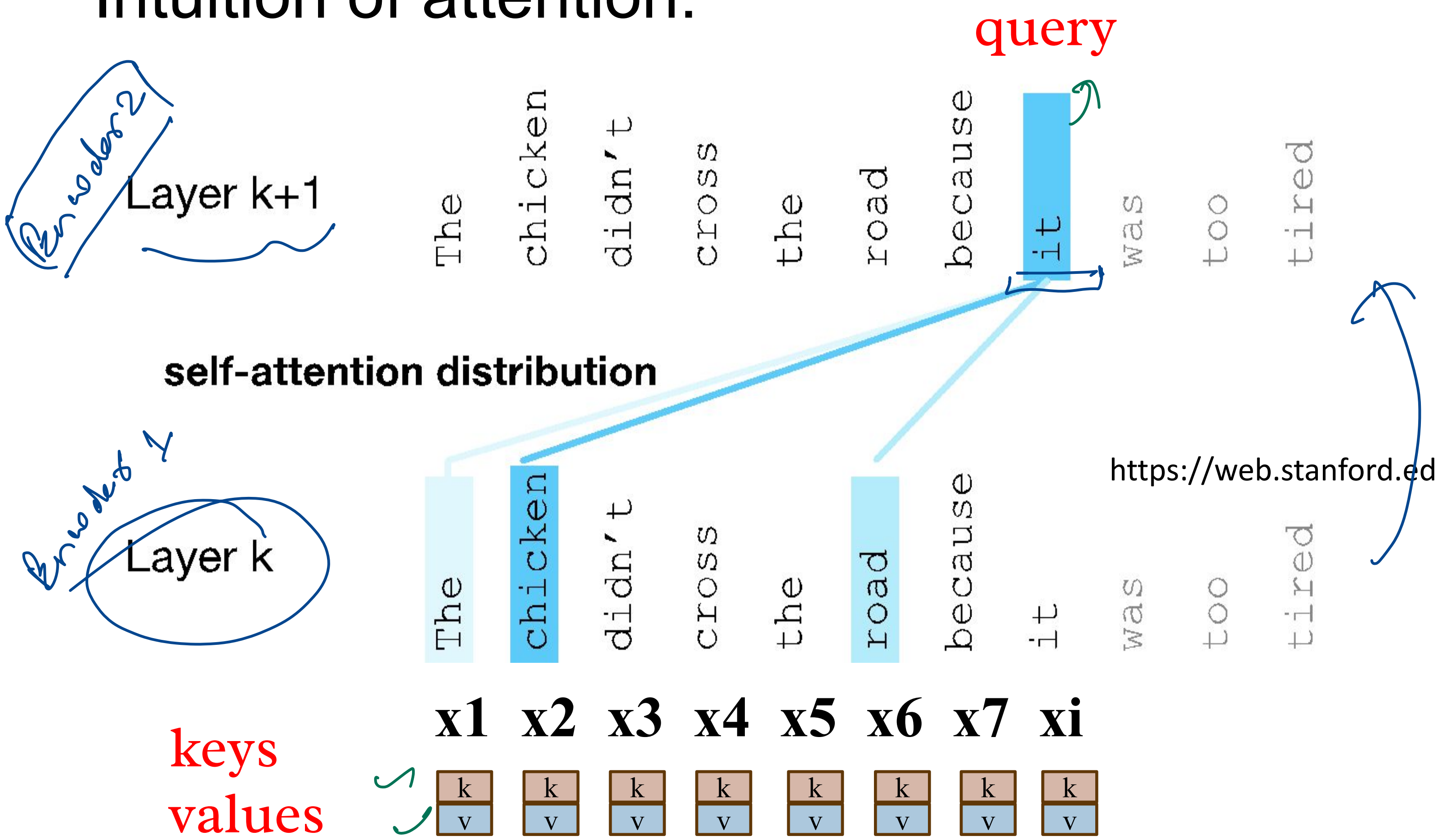
Attention is based on key/value/query concept -- analogous to retrieval systems.

When you search for videos on Youtube

- The search engine will map your query (text in the search bar) against a set of keys (video title, description, etc.) associated with candidate videos in their database
- It will then present you the best matched videos (values).



Intuition of attention:



https://web.stanford.edu/~jurafsky/slp3/

An Actual Attention Head: slightly more complicated

We'll use matrices to project each vector $\mathbf{x}_i$ into a representation of its role as query, key, value:

- **query**: $\mathbf{W}^Q$
- **key**: $\mathbf{W}^K$
- **value**: $\mathbf{W}^V$

3 learnable parameter (matrices)

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^Q; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^K; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^V$$

https://web.stanford.edu/~jurafsky/slp3/

An Actual Attention Head: slightly more complicated

Given these 3 representation of $\boldsymbol{x}_i$

To compute similarity of current element $\boldsymbol{x}_i$ with some prior element $\boldsymbol{x}_j$

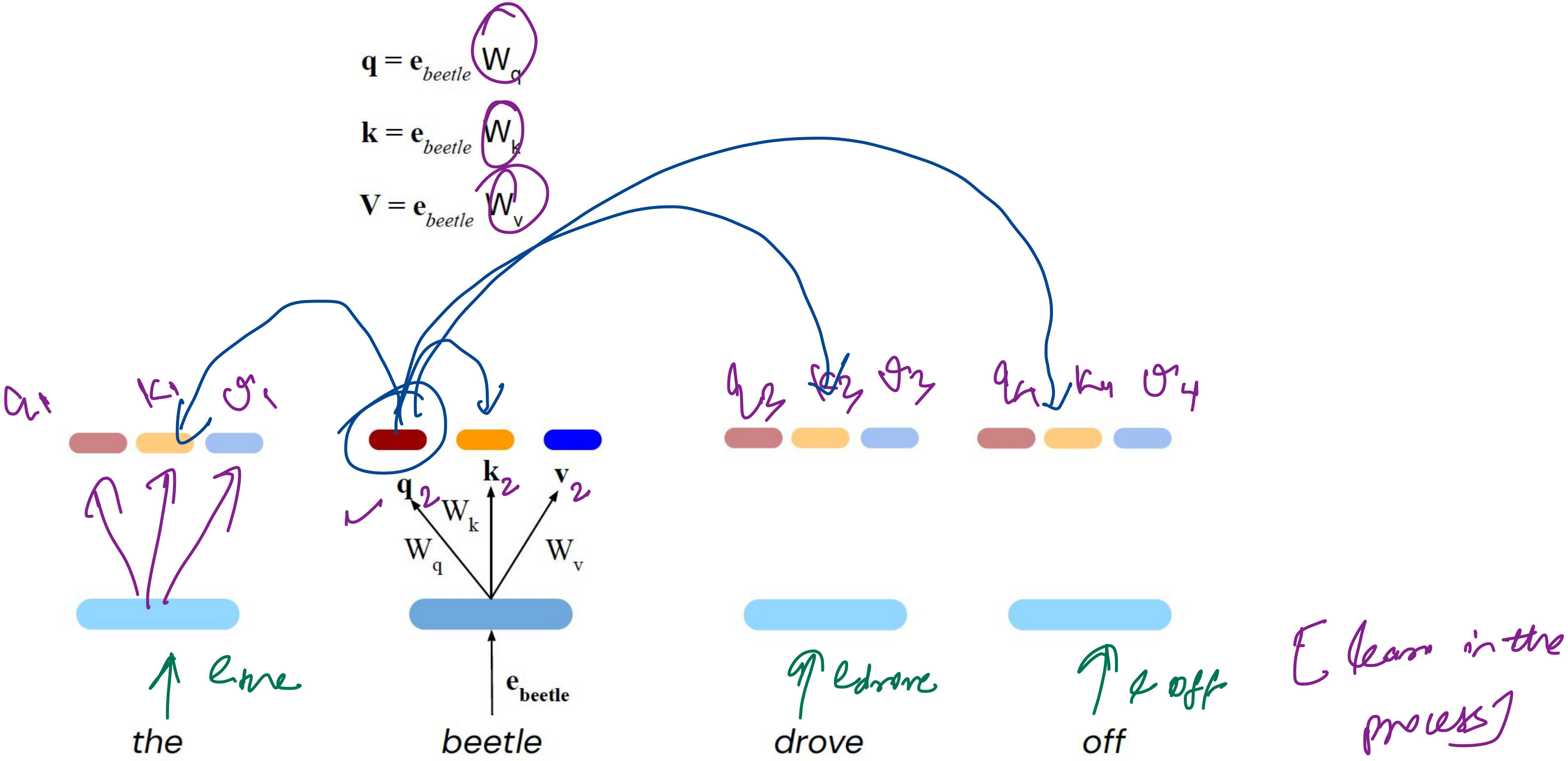
We'll use dot product between $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$.

And instead of summing up $\boldsymbol{x}_j$, we'll sum up $\boldsymbol{v}_j$

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}}$$

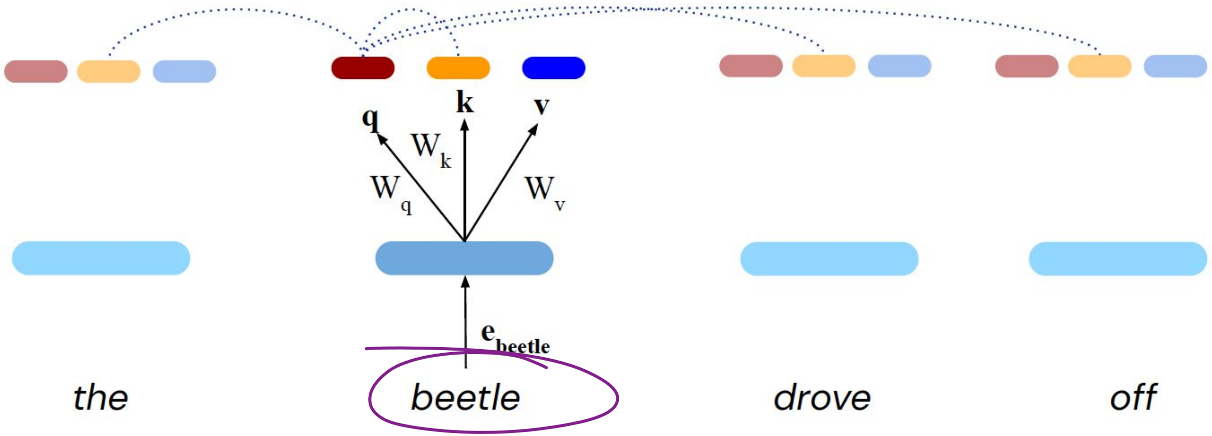
https://web.stanford.edu/~jurafsky/slp3/

Transformers: Self-attention over input



Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

Self-attention over input embeddings

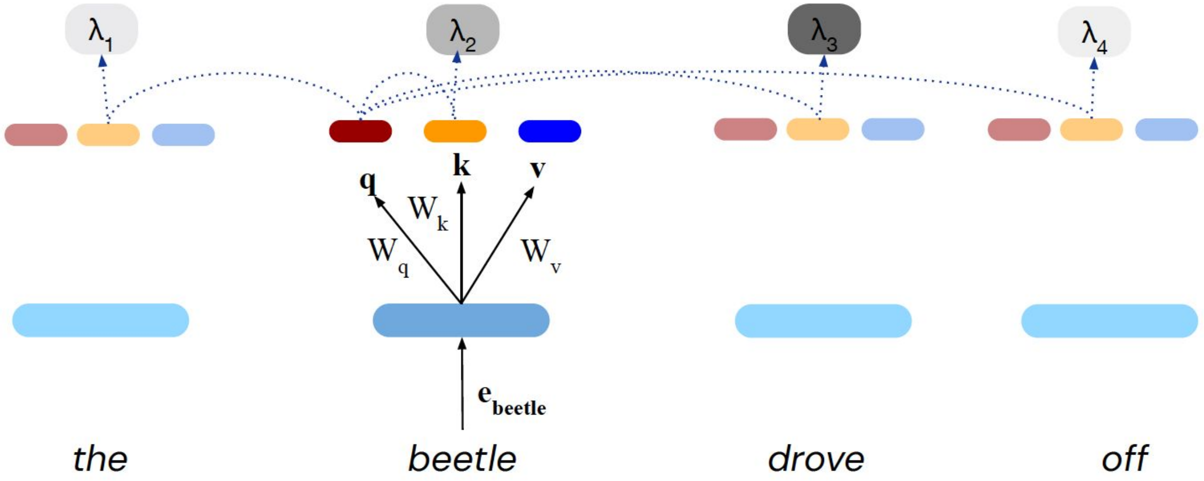


Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

Self-attention over input embeddings

$$\lambda_i = \frac{e^{\mathbf{q}\cdot\mathbf{k}_i}}{\sum_{i=1}^{4} e^{\mathbf{q}\cdot\mathbf{k}_i}}$$

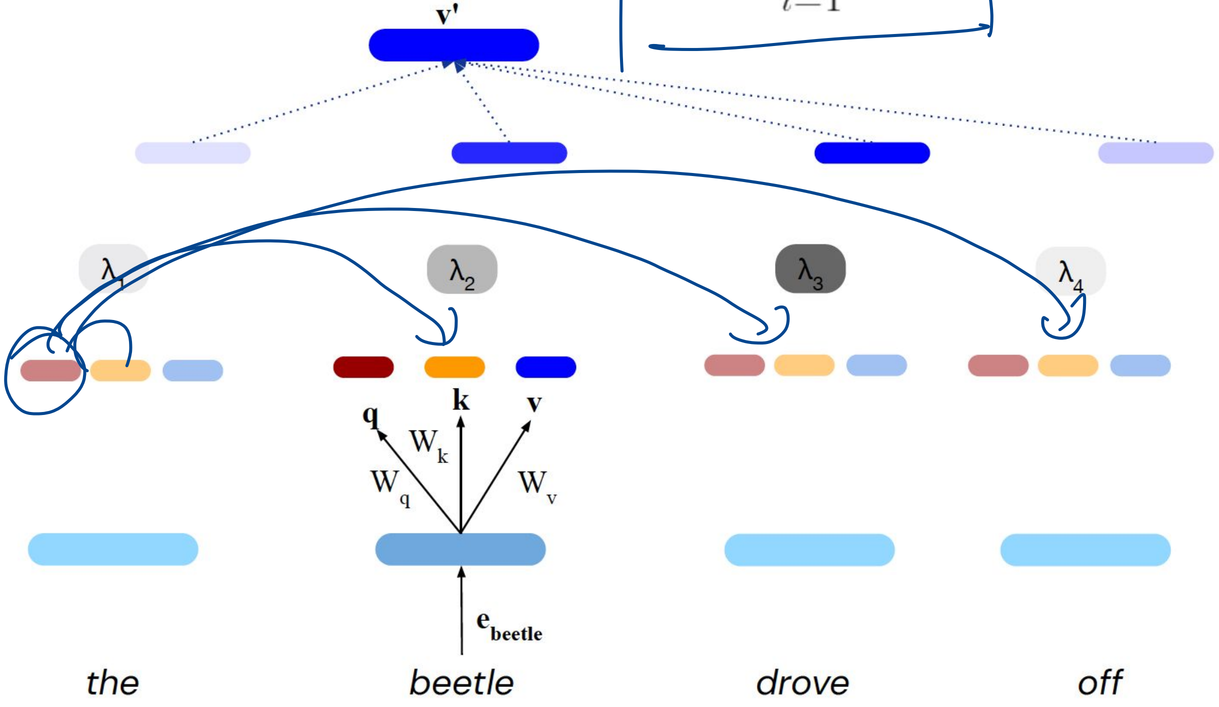
$\boxed{\sum \lambda_i v_i}$



Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

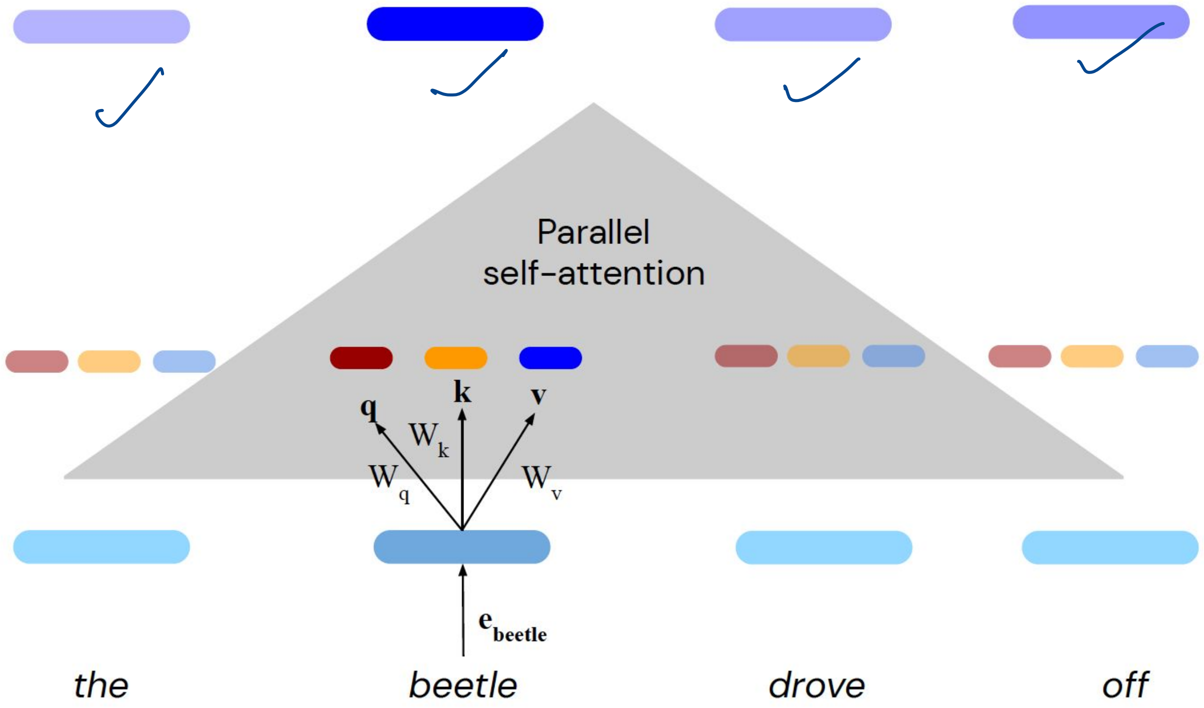
Self-attention over input embeddings

$$\mathbf{v}' = \sum_{i=1}^{4} \lambda_i \mathbf{v}_i$$



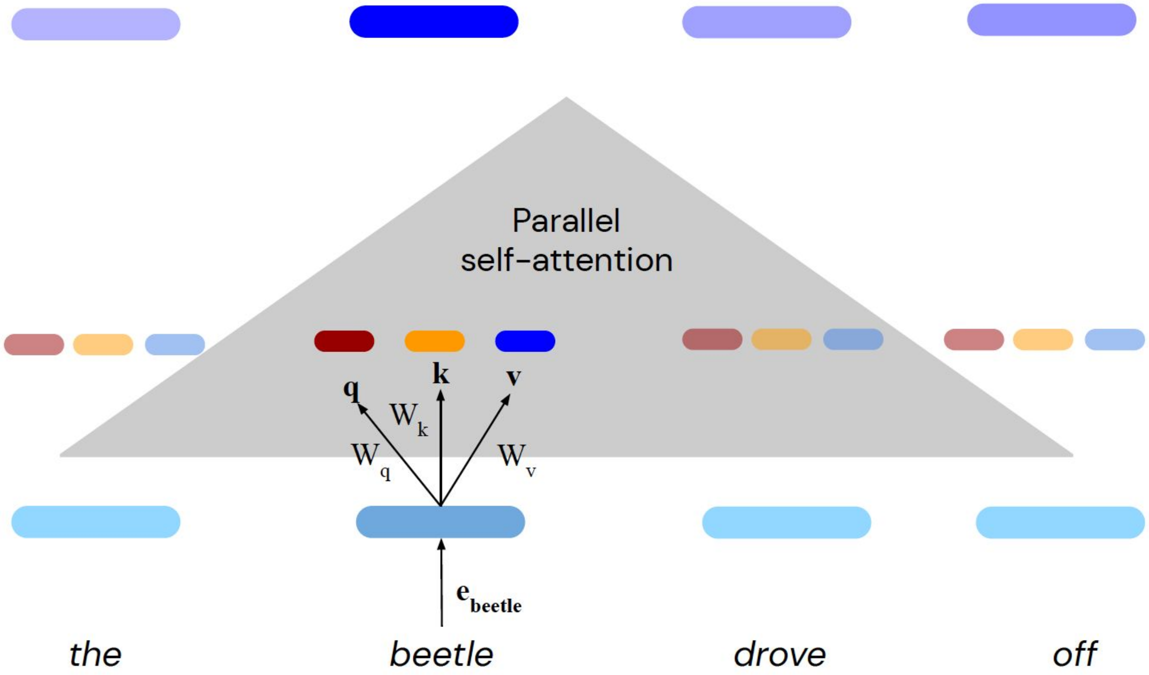
Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

Self-attention over all words (in parallel)



Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

Self-attention over all words (in parallel)



Source: *https://deepmind.com/learning-resources/deep-learning-lecture-series-2020*

Self-attention: In equations

hardmaru
@hardmaru · Follow

The most important formula in deep learning after 2018

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d, $X \in \mathbf{R}^{n \times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ to extract feature representations Q, K, and V, referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q, K, V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \tag{1}$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \tag{2}$$

where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

7:38 AM · Feb 10, 2021

3.1K Reply Copy link

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j \mathbf{W}^{\mathbf{V}}$$

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j))$$

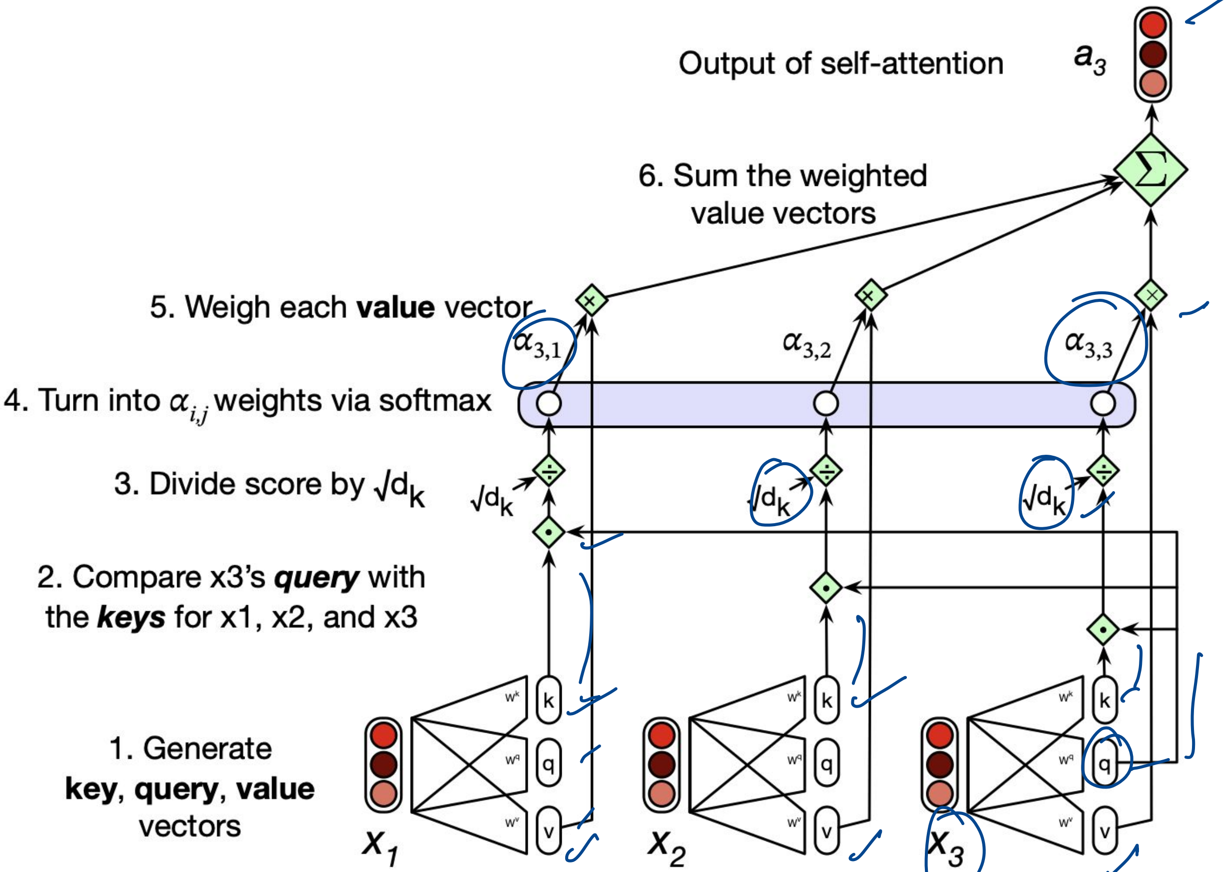
$$\mathbf{a}_i = \sum \alpha_{ij} \mathbf{v}_j$$

Scaled dot-product: *more on this later*

Source: *https://theaisummer.com/self-attention/* *https://web.stanford.edu/~jurafsky/slp3/*

Calculating the self-attention output

weighted average of values



https://web.stanford.edu/~jurafsky/slp3/

Try this problem

Suppose, you give the following input to your transformer encoder: {flying, arrows} The input embeddings for these two words are **[0,1,1,1,1,0] and [1,1,0,-1,-1,1]**, respectively. Suppose you are trying to represent the first word ‘flying’ with the help of self-attention in the first encoder. For the first attention head, the query, key and value matrices just take the 2 dimensions from the input each. Thus, the first 2 dimensions define the query vector, and so on. What will be the self-attention output for the word ‘flying’ corresponding to this attention head. You are using the scaled dot vector.