Figure 1: “The document ended with a cartoony image of six Transformers in mountainous terrain, zapping lasers at one another.”

“I never really understood the name. It sounds cool though.”

- Aidan Gomez

Overview: Next $\sim$3 Weeks

Assumed: ML/DL background

1. Semantic embeddings
2. Attention mechanism
3. GPT architecture (Simplified)
4. GPT architecture (More Details)
5. How LLMs are grown (pretraining/SFT/RLHF)
6. Other salient topics (e.g. inference-scaling)

After that: will rotate around as paper-of-the-week.

2020: An Alternate Timeline

Figure 2: From Voyager 3 (under the water-ammonia ocean)

Bioluminescent Neptunian Orbs: A Timeline

Behavioral characteristics:

- Glowing displays form patterns
- Imitative
- Reflecting all modes and forms!

2020-2021: First contact with alien intelligence! Astrobiologists launch international Orb Genome Project (OGP) to unravel the mysteries.
2022-2023: Most scientists now astrobiologists, join OGP or related projects. "Inner thoughts" of orbs begin decoding, mapped to luminous outputs via certain rules.
2024-: Further progress! "Xenoinformatics" matures as a field integrating classical information theory with orb neuroscience.

Bioluminescent Neptunian Orbs: A (Different) Timeline

Behavioral characteristics:

- Glowing displays form patterns
- Imitative
- Reflecting all modes and forms!

2020-2021: Orbs deemed similar to Earth-based parrots. Nothing to see here.
2022-2023: Orbs can be domesticated for workplace tasks! Let's farm these at scale for big $$$.
2024-: New products: NotebookOrb, OrbCopilot, multimodal orbs, orb agents! Bigger orbs get bigger $$$! How do they work again?

Outline

Word Embeddings

A word has many dimensions of meaning.[citation needed]

Why not represent these as literal dimensions in a vector space?

A *word embedding* is a map $E : \texttt{words} \to \mathbb{R}^d$ that captures the meaning of different words.

"Many dimensions of meaning": Need a very high dimensional space for this to be possible at all (the *embedding dimension* of a model).

- Original transformer: $d_{model} = 512$
- GPT-3: $d_{model} = 12288$

Key Property 1: Measuring Similarity

We have 2 words such that $v_1 = E(\texttt{word}_1)$ and $v_2 = E(\texttt{word}_2)$.
How to measure "similarity" of meaning?
Idea 1: Find their angle θ between to see how "aligned" they are.
Idea 2: Compute their dot product $v_1 \cdot v_2$ to see how much their components agree.
Once we normalize by magnitude $||v_1||\ ||v_2||$, these ideas agree:

$$\cos(\theta) = \frac{v_1 \cdot v_2}{||v_1||\ ||v_2||}$$

Property (Similarity Measurement)

Cosine similarity $\cos(\theta)$ gives a good measure of "similar meanings" between words.

- $\cos(0) = 1$: perfect alignment
- $\cos(90) = 0$: orthogonal/unrelated
- $\cos(180) = -1$: polar opposites

Key Property 2: Arithmetic of Meaning

If some analogy like $\texttt{uncle} : \texttt{man} \approx \texttt{woman} : \texttt{aunt}$ holds, we can express it arithmetically in the embedding space:

$$E(\texttt{uncle}) - E(\texttt{man}) + E(\texttt{woman}) \approx E(\texttt{aunt})$$

Can do further manipulations:

$$\overrightarrow{plur} := E(\texttt{cats}) - E(\texttt{cat})$$

$$\overrightarrow{plur} \cdot E(\texttt{three}) > \overrightarrow{plur} \cdot E(\texttt{two}) > \overrightarrow{plur} \cdot E(\texttt{one})$$

Key Property 2: Arithmetic of Meaning

If some analogy like $\texttt{uncle} : \texttt{man} \approx \texttt{woman} : \texttt{aunt}$ holds, we can express it arithmetically in the embedding space:

$$E(\texttt{uncle}) - E(\texttt{man}) + E(\texttt{woman}) \approx E(\texttt{aunt})$$

Can do further manipulations:

$$\begin{aligned}
\overrightarrow{plur} &:= E(\texttt{cats}) - E(\texttt{cat}) \\
\overrightarrow{plur} \cdot E(\texttt{three}) &> \overrightarrow{plur} \cdot E(\texttt{two}) > \overrightarrow{plur} \cdot E(\texttt{one})
\end{aligned}$$

Property (Arithmetic of Meaning)

The relative meanings of words can be decomposed and recombined in nontrivial equations/inequalities.

Handling Ambiguity

Also, given a polysemous word like `tie` which has multiple meanings $\texttt{tie}_1, \texttt{tie}_2, \texttt{tie}_3$, then $\exists \alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}$ such that

$$E(\texttt{tie}) \approx \alpha_1 \cdot E(\texttt{tie}_1) + \alpha_2 \cdot E(\texttt{tie}_2) + \alpha_3 \cdot E(\texttt{tie}_3)$$

Method:

> *Take two random words w_1, w_2. Combine them into an artificial polysemous word w_{new} by replacing every occurrence of w_1 or w_2 in the corpus by w_{new}. Next, compute an embedding for w_{new} using the same embedding method while deleting embeddings for w_1, w_2 but preserving the embeddings for all other words. Compare the embedding $v_{w_{new}}$ to linear combinations of v_{w_1} and v_{w_2}.*

Word Embeddings, Constructively

Firth's principle (1957):

You shall know a word by the company it keeps.

Naïve implementation on a corpus $\mathcal{C}$:

1. Randomly initialize word embeddings.
2. Trawl the corpus, updating the embedding $E(w)$ of some w by looking at the window of $W = 10$ words surrounding it.
    1. Let $\vec{\mu} :=$ the average embedding of those.
    2. Set $E(w) \mathrel{+}= \eta * \vec{\mu}$ for some choice of "learning rate" η.
3. (Can modify this by changing W, or by weighting closer words higher with some simple decay function).

Word Embeddings, Constructively

Firth's principle (1957):

You shall know a word by the company it keeps.

Naïve implementation on a corpus $\mathcal{C}$:

1. Randomly initialize word embeddings.
2. Trawl the corpus, updating the embedding $E(w)$ of some w by looking at the window of $W = 10$ words surrounding it.
   1. Let $\vec{\mu} :=$ the average embedding of those.
   2. Set $E(w) \mathrel{+}= \eta * \vec{\mu}$ for some choice of "learning rate" η.
3. (Can modify this by changing W, or by weighting closer words higher with some simple decay function).

This will (mostly) have Property 1 but not much more. Word embeddings in the 1960s looked roughly like this.

Slightly better idea: TF-IDF (1975)

Much better idea: deep learning on a specific NLP task

Thought Vectors

Suppose we have an embedding that captures the meaning of words really well.

It does this for *a lot* of words.

There's a universe of words out there, with very refined meanings that capture very nuanced shades of connotation.

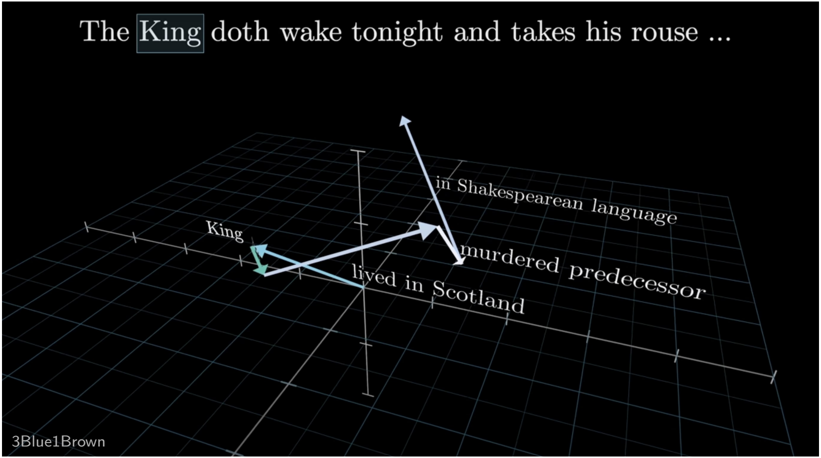
If we're doing this, we're already representing the relations between fairly complicated thoughts.

In principle, this *embedding space* would be rich enough to accommodate phrases, whole sentences, or even multiple sentences. Hence the (mostly synonymous) ideas of:

- *Sentence embeddings*
- *Thought vectors* (Hinton)

Sentence Embedding

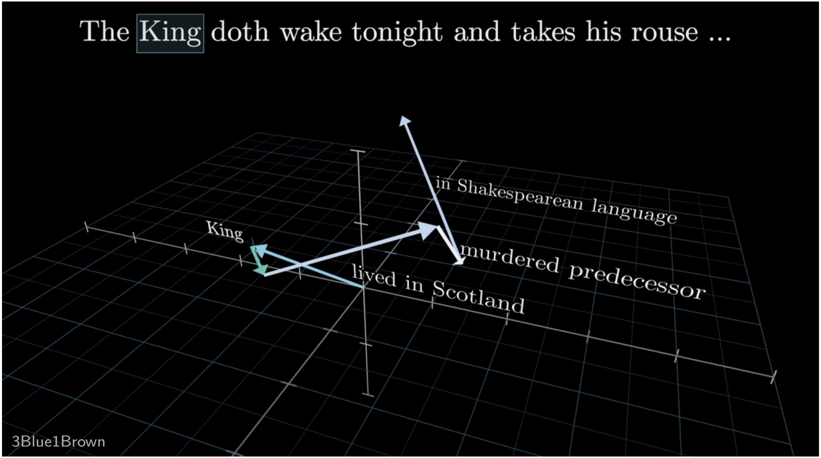
A word's meaning depends on the *context* of words around it.



We must somehow update its embedding to reflect these changes of meaning.

Sentence Embedding

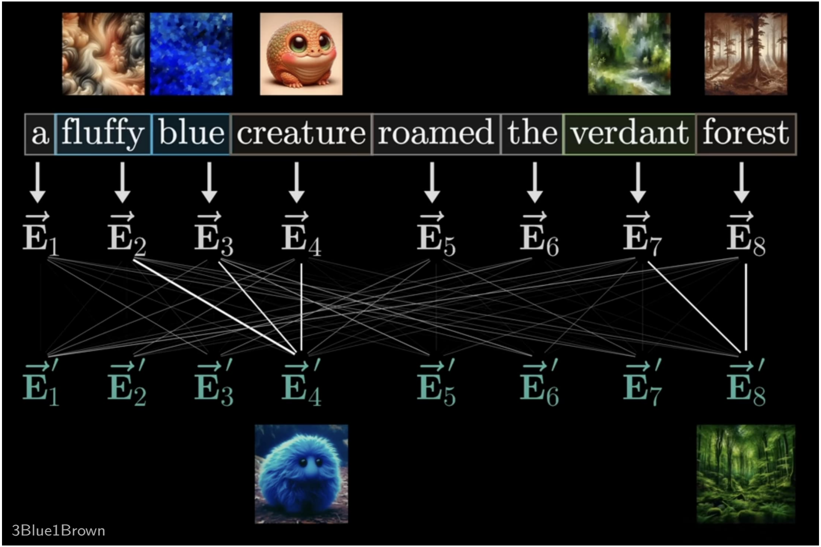
A transformer builds up the *cumulative meaning* of the text its seen so far, reading left-to-right.[1]

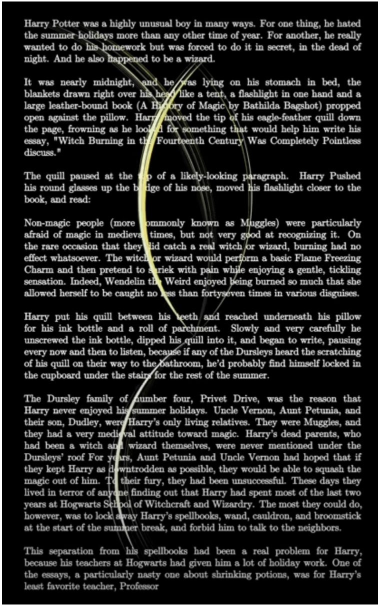


[1] Computationally, it doesn't work this way: transformers process text in an atemporal, parallel manner. But these are mathematically equivalent.

Sentence Embedding

Our general goal here:





Harry Potter was a highly unusual boy in many ways. For one thing, he hated the summer holidays more than any other time of year. For another, he really wanted to do his homework but was forced to do it in secret, in the dead of night. And he also happened to be a wizard.

It was nearly midnight, and he was lying on his stomach in bed, the blankets drawn right over his head like a tent, a flashlight in one hand and a large leather-bound book (A History of Magic by Bathilda Bagshot) propped open against the pillow. Harry moved the tip of his eagle-feather quill down the page, frowning as he looked for something that would help him write his essay, "Witch Burning in the Fourteenth Century Was Completely Pointless discuss."

The quill paused at the top of a likely-looking paragraph. Harry Pushed his round glasses up the bridge of his nose, moved his flashlight closer to the book, and read:

Non-magic people (more commonly known as Muggles) were particularly afraid of magic in medieval times, but not very good at recognizing it. On the rare occasion that they did catch a real witch or wizard, burning had no effect whatsoever. The witch or wizard would perform a basic Flame Freezing Charm and then pretend to shriek with pain while enjoying a gentle, tickling sensation. Indeed, Wendelin the Weird enjoyed being burned so much that she allowed herself to be caught no less than forty-seven times in various disguises.

Harry put his quill between his teeth and reached underneath his pillow for his ink bottle and a roll of parchment. Slowly and very carefully he unscrewed the ink bottle, dipped his quill into it, and began to write, pausing every now and then to listen, because if any of the Dursleys heard the scratching of his quill on their way to the bathroom, he'd probably find himself locked in the cupboard under the stairs for the rest of the summer.

The Dursley family of number four, Privet Drive, was the reason that Harry never enjoyed his summer holidays. Uncle Vernon, Aunt Petunia, and their son, Dudley, were Harry's only living relatives. They were Muggles, and they had a very medieval attitude toward magic. Harry's dead parents, who had been a witch and wizard themselves, were never mentioned under the Dursleys' roof For years, Aunt Petunia and Uncle Vernon had hoped that if they kept Harry as downtrodden as possible, they would be able to squash the magic out of him. To their fury, they had been unsuccessful. These days they lived in terror of anyone finding out that Harry had spent most of the last two years at Hogwarts School of Witchcraft and Wizardry. The most they could do, however, was to lock away Harry's spellbooks, wand, cauldron, and broomstick at the start of the summer break, and forbid him to talk to the neighbors.

This separation from his spellbooks had been a real problem for Harry, because his teachers at Hogwarts had given him a lot of holiday work. One of the essays, a particularly nasty one about shrinking potions, was for Harry's least favorite teacher, Professor

This meaning can depend on context from away.

Attention is a mechanism for tracking these long-range dependencies.

Outline

Caveats

We're now ready to understand the attention mechanism. Some technicalities I will postpone until later:

- Word/token distinction
- Positional encodings
- Masked attention

I'm going to describe *decoder-only self-attention*. This is different from the original encoder-decoder transformer, but it is actually simpler and more common nowadays.

Attention: The Basic Idea

Attention lets each word vector directly interact with every previous word vector, in 2 steps:

1. Compute *relevance scores* between the current word and all previous words.
2. Updates the current word's representation based on the meaning (i.e. *value*) of the relevant previous words.

Relevance Scores

When we read a word, there are many different ways surrounding words can be relevant to it, e.g.

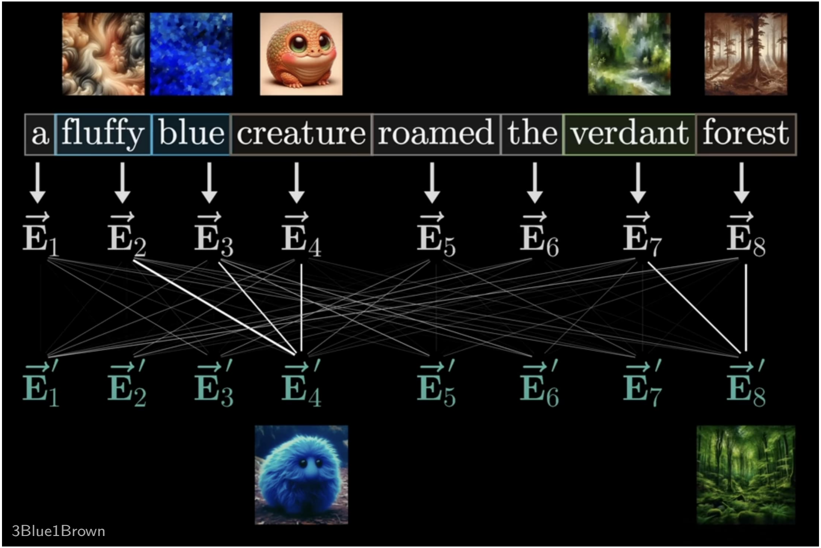
- Adjectives modifying nouns: "The fluffy blue creature."
- Names/pronouns influencing gender information much later: "Alice and Bob were [...]. He [...]."
- Polysemy resolution: "Cross the [river/street] to reach the bank."
- Anaphora resolution: "The law will never be perfect, but its application should be just."
- ... and many much more subtle examples.

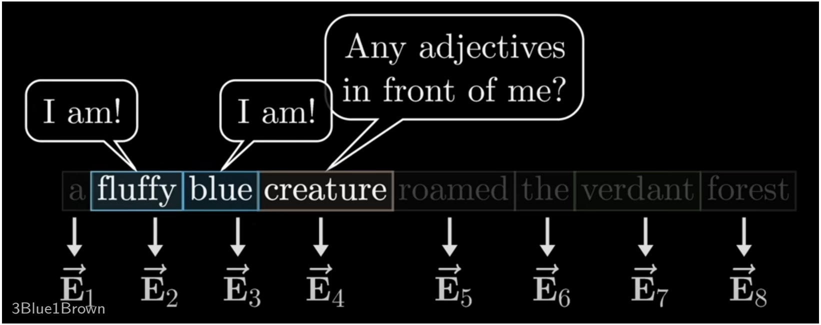
It is useful[2] to think of a single attention head as focusing on one of these possible ways context can inform a word's meaning.

We want to have "bank" *attend* to "[river/street]", but not to the less relevant words.

[2] But not true; see *polysemanticity/superposition*.

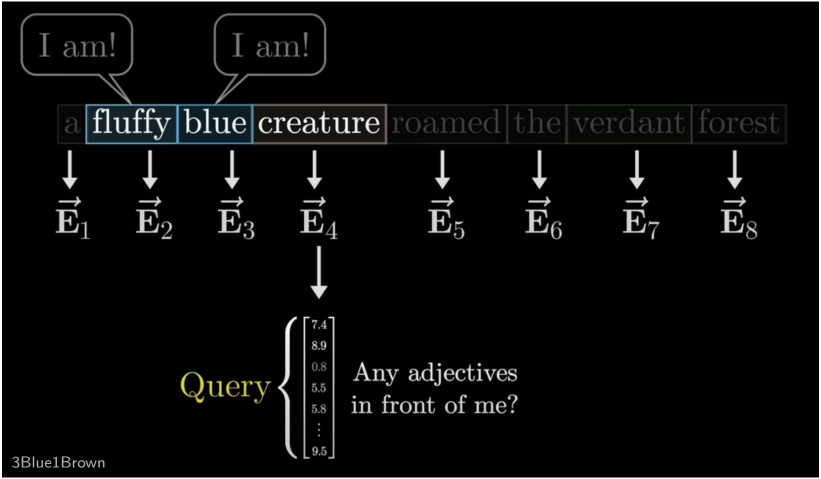
Recap: Our general goal





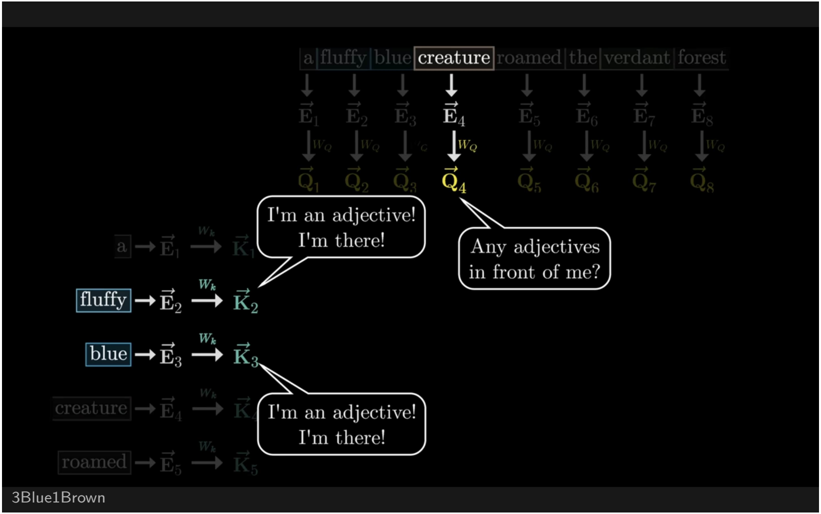
Each word gets to ask certain questions (*queries*) about previous words.

Previous words can potentially respond to these questions (*keys*).



A query is represented as a vector $\vec{Q}_j$.

Queries x Keys



A key is represented as a vector $\vec{K}_i$.

Queries x Keys

We want to check if the key $\vec{K}_i$ "answers" (is *compatible* with) the query $\vec{Q}_j$.

We can check if the vector $\vec{K}_i$ is aligned with the query $\vec{Q}_j$.

- Take the dot product $\vec{K}_i \cdot \vec{Q}_j$.

$\vec{K}_i \cdot \vec{Q}_j$ is the *relevance score*.

- High relevance: word j will *attend* to word i.


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$
blue $\rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$
creature $\rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$
the $\rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$
forest $\rightarrow \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$
$\vec{K}_1 \cdot \vec{Q}_1$ $\vec{K}_1 \cdot \vec{Q}_2$ $\vec{K}_1 \cdot \vec{Q}_3$ $\vec{K}_1 \cdot \vec{Q}_4$ $\vec{K}_1 \cdot \vec{Q}_5$ $\vec{K}_1 \cdot \vec{Q}_6$ $\vec{K}_1 \cdot \vec{Q}_7$ $\vec{K}_1 \cdot \vec{Q}_8$
$\vec{K}_2 \cdot \vec{Q}_1$ $\vec{K}_2 \cdot \vec{Q}_2$ $\vec{K}_2 \cdot \vec{Q}_3$ $\vec{K}_2 \cdot \vec{Q}_4$ $\vec{K}_2 \cdot \vec{Q}_5$ $\vec{K}_2 \cdot \vec{Q}_6$ $\vec{K}_2 \cdot \vec{Q}_7$ $\vec{K}_2 \cdot \vec{Q}_8$
$\vec{K}_3 \cdot \vec{Q}_1$ $\vec{K}_3 \cdot \vec{Q}_2$ $\vec{K}_3 \cdot \vec{Q}_3$ $\vec{K}_3 \cdot \vec{Q}_4$ $\vec{K}_3 \cdot \vec{Q}_5$ $\vec{K}_3 \cdot \vec{Q}_6$ $\vec{K}_3 \cdot \vec{Q}_7$ $\vec{K}_3 \cdot \vec{Q}_8$
$\vec{K}_4 \cdot \vec{Q}_1$ $\vec{K}_4 \cdot \vec{Q}_2$ $\vec{K}_4 \cdot \vec{Q}_3$ $\vec{K}_4 \cdot \vec{Q}_4$ $\vec{K}_4 \cdot \vec{Q}_5$ $\vec{K}_4 \cdot \vec{Q}_6$ $\vec{K}_4 \cdot \vec{Q}_7$ $\vec{K}_4 \cdot \vec{Q}_8$
$\vec{K}_5 \cdot \vec{Q}_1$ $\vec{K}_5 \cdot \vec{Q}_2$ $\vec{K}_5 \cdot \vec{Q}_3$ $\vec{K}_5 \cdot \vec{Q}_4$ $\vec{K}_5 \cdot \vec{Q}_5$ $\vec{K}_5 \cdot \vec{Q}_6$ $\vec{K}_5 \cdot \vec{Q}_7$ $\vec{K}_5 \cdot \vec{Q}_8$
$\vec{K}_6 \cdot \vec{Q}_1$ $\vec{K}_6 \cdot \vec{Q}_2$ $\vec{K}_6 \cdot \vec{Q}_3$ $\vec{K}_6 \cdot \vec{Q}_4$ $\vec{K}_6 \cdot \vec{Q}_5$ $\vec{K}_6 \cdot \vec{Q}_6$ $\vec{K}_6 \cdot \vec{Q}_7$ $\vec{K}_6 \cdot \vec{Q}_8$
$\vec{K}_7 \cdot \vec{Q}_1$ $\vec{K}_7 \cdot \vec{Q}_2$ $\vec{K}_7 \cdot \vec{Q}_3$ $\vec{K}_7 \cdot \vec{Q}_4$ $\vec{K}_7 \cdot \vec{Q}_5$ $\vec{K}_7 \cdot \vec{Q}_6$ $\vec{K}_7 \cdot \vec{Q}_7$ $\vec{K}_7 \cdot \vec{Q}_8$
$\vec{K}_8 \cdot \vec{Q}_1$ $\vec{K}_8 \cdot \vec{Q}_2$ $\vec{K}_8 \cdot \vec{Q}_3$ $\vec{K}_8 \cdot \vec{Q}_4$ $\vec{K}_8 \cdot \vec{Q}_5$ $\vec{K}_8 \cdot \vec{Q}_6$ $\vec{K}_8 \cdot \vec{Q}_7$ $\vec{K}_8 \cdot \vec{Q}_8$
3Blue1Brown

The *attention pattern*: "creature" is paying attention to "fluffy" and "blue"; "forest" attends to "verdant".

Relevance Scores

Upshot: $\vec{K}_i \cdot \vec{Q}_j$ measures the relevance of word i to word j.

We pack all these vector-vector products into a single matrix-matrix[3] product $QK^\top$.

Therefore $QK^\top$, the *attention pattern*, contains all the pairwise relevance scores.

We then scale/normalize this to obtain:

$$\text{Relevance}(Q, K) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$$

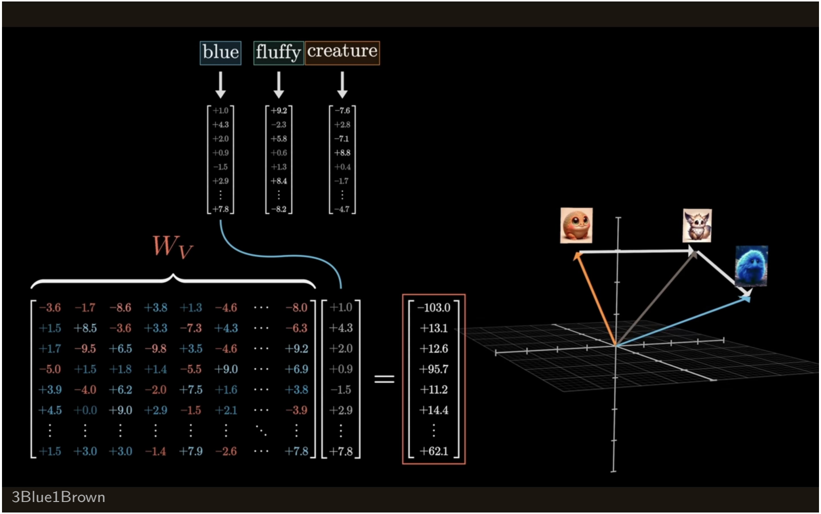
called *scaled dot product attention*.[4]

[3] $K = [\vec{K}_1|\vec{K}_2|\vec{K}_3|...]^\top$ is called the *key matrix* and Q is the *query matrix*

[4] d_K is the size of the key/query vectors: $d_K = 64$ in OG transformer, 128 in GPT-3.

Value Matrix

We now have the (scaled) relevance scores. We need to use these to update the meaning of word j:



Value Matrix

A word's *value vector* $\vec{V}_i$ essentially says:

> *If I am relevant to another word's meaning, how should its meaning be updated?*

For example, the value vector $\vec{V}_2$ for "fluffy":

> *If a noun is fluffy, how should that change its contextual meaning?*

But we already solved the "if" problem! So we can just compute:

$$\text{Relevance}(\vec{Q}_j, \vec{K}_i) \cdot \vec{V}_i$$

to figure out how word i should update the meaning of word j.

Attention

As before, we can combine the value vectors into a single *value matrix* V. Then the "meaning update" equation is:

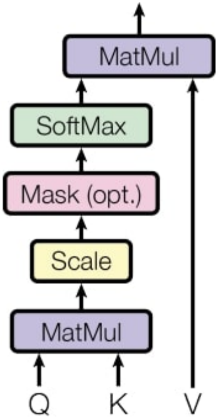
$$\begin{aligned} \text{Attention}(Q, K, V) &:= \text{Relevance}(Q, K) \cdot \text{Value} \\ &:= \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \end{aligned}$$

More details: the actual weights in the attention head are the matrices W_Q, W_K, W_V.

These interact with the activations $E = [\vec{E}_1 | \vec{E}_2 | \vec{E}_3 | \ldots]$ as follows:

$$\begin{aligned} Q &:= W_Q \cdot E \\ K &:= W_K \cdot E \\ V &:= W_V \cdot E \end{aligned}$$

Scaled Dot-Product Attention



An *attention head* is the primitive unit that does this calculation.

Outline

Single-Head Attention

(1) tells us how much we should *update* the meaning of words.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

If $\vec{E}_j$ is the embedding of word j, then we revise its meaning to $\vec{E}_j'$ based on the rule:

$$\vec{E}_j' := \vec{E}_j + \Delta\vec{E}_j$$

where $\Delta\vec{E}_j$ is what we get out of equation (1).

$\vec{E}_j$ gets passed along via a *residual connection*, so that it can be recombined with $\Delta\vec{E}_j$.

Multi-Head Attention

Recall the many ways a word's context can inform its meaning:

- Adjectives modifying nouns: "The fluffy blue <u>creature</u>."
- Names/pronouns influencing gender information much later: "Alice and Bob were [...]. <u>He</u> [...]."
- Polysemy resolution: "Cross the [river/street] to get to the <u>bank</u>."
- Anaphora resolution: "The law will never be perfect, but <u>its</u> application should be just."

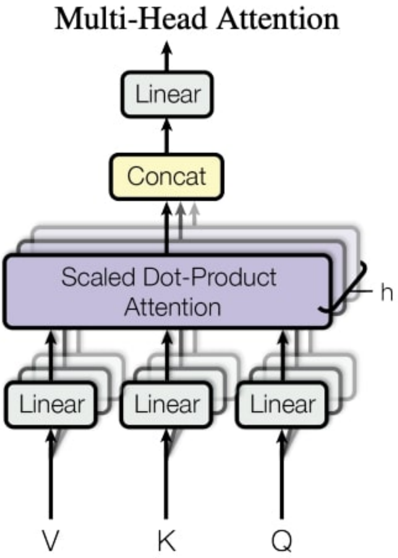
  ⋮

Convenient lie: A single attention head computes the meaning updates from a single contextual clue C.

Many contextual clues $C_1, C_2, C_3, \ldots$ can be computed independently from each other.

Why not compute them *in parallel*?

Multi-Head Attention



Stack Moar Layerz

But not all contextual clues are unrelated to each other.

- You have to gather basic data before asking more abstract questions.

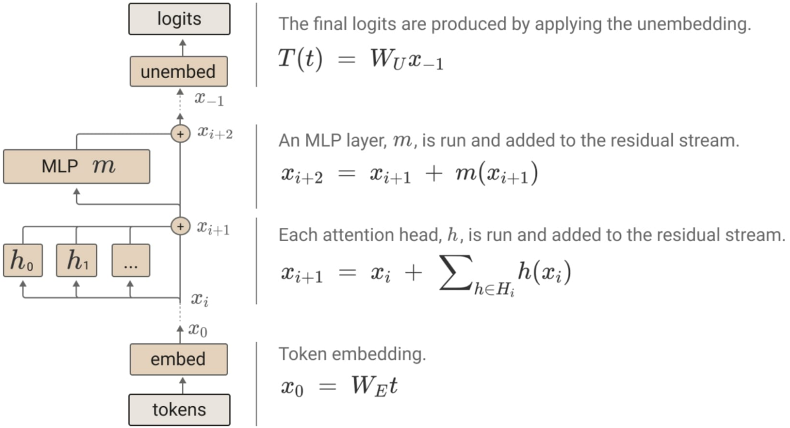
Parallel attention heads form an *attention layer*. We stack a bunch of these end-to-end:

- OG transformer: $h = 8$ parallel heads/layer, $N = 6$ layers
- GPT-3: $h = 96$ parallel heads/layer, $N = 96$ layers

There are also MLP layers alternating between the attention layers.

- Attention layer + MLP layer = "Attention block"

The Residual Stream



As a word goes through different attention heads, multiple updates get applied to it. By the end we have its full contextual meaning.

Outline

Prologue

2012: Deep learning boom begins with AlexNet. The most dramatic early achievements were in computer vision with convolutional neural networks (CNNs).

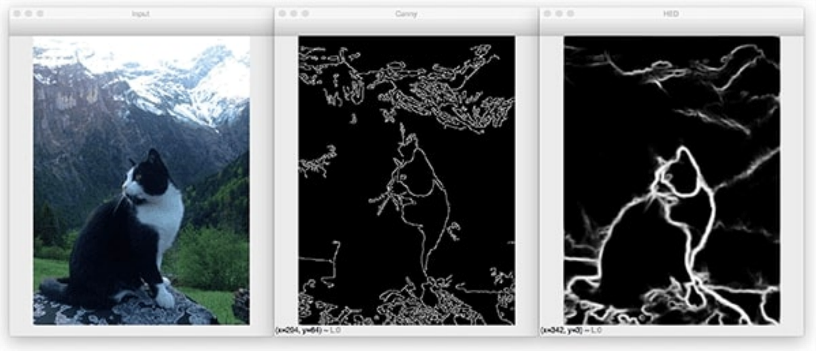
In contrast, it took a few years for DL to achieve supremacy at natural language processing (NLP).

This was eventually done with recurrent neural networks (RNNs, which are designed for processing *sequential* inputs). Some innovations were still required:

1. Word2vec (2013) arithmetic on word embeddings
2. Encoder-Decoder/seq2seq architectures ($\sim$2014)
3. Attention (2014): *A mechanism for tracking long-range dependencies in text*

CNNs vs. RNNs

Image classification is an easier problem than NLP:



To check whether a pixel is part of an edge, it suffices to look at nearby pixels.

In contrast, the *contextual* meaning of a given word/phrase can heavily depend on words several pages earlier.

Timeline

2014: Attention mechanism introduced (in RNNs).

2017: Entirely new NN architecture introduced which centralizes attention to the exclusion of almost everything else (such as the sequentiality of RNNs). Called the *transformer*.

2018: GPT-1 and BERT. Almost no one uses RNNs anymore.

2019: GPT-2 begins dominance of decoder-only generative transformers.

Andrej Karpathy @karpathy 8 Dec 2021

The ongoing consolidation in AI is incredible. Thread: ➡️ When I started ~decade ago vision, speech, natural language, reinforcement learning, etc. were completely separate; You couldn't read papers across areas - the approaches were completely different, often not even ML based.

Dec 8, 2021 · 12:03 AM UTC

336 1,657 250 8,106

Andrej Karpathy @karpathy 8 Dec 2021

In 2010s all of these areas started to transition 1) to machine learning and specifically 2) neural nets. The architectures were diverse but at least the papers started to read more similar, all of them utilizing large datasets and optimizing neural nets.

2 38 1 890

Andrej Karpathy @karpathy 8 Dec 2021

But as of approx. last two years, even the neural net architectures across all areas are starting to look identical - a Transformer (definable in ~200 lines of PyTorch github.com/karpathy/minGPT/b…), with very minor differences. Either as a strong baseline or (often) state of the art.

6 88 13 1,284