|  |  |
| --- | --- |
| Transformers | Introduction to Large Language |
| and Large | Models |
| Language |  |
| Models |  |



Language models

* Remember the simple n-gram language model
  + Assigns probabilities to sequences of words
  + Generate text by sampling possible next words
  + Is trained on counts computed from lots of text
* Large language models are similar and different:
* Assigns probabilities to sequences of words
* Generate text by sampling possible next words
* **Are trained by learning to guess the next word**

Neural Large Language Models (LLMs)

* Self-supervised learners
  + Take a text, remove a word
  + Use your neural model to guess what the word was
  + If the model is wrong, use stochastic gradient descent to make the model guess better next time
* Advantages (?):
* All we need is a **lot of** text (GPT3: 500 billion tokens)
* (And a **lot of** compute)

LLMs are built out of transformers

Transformer: a specific kind of network architecture, like a fancier feedforward network, but based on attention



Attention Is All You Need



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ashish Vaswani⇤ | Noam Shazeer⇤ | Niki Parmar⇤ | | Jakob Uszkoreit⇤ |
| Google Brain | Google Brain | Google Research | | Google Research |
| avaswani@google.com | noam@google.com nikip@google.com | | | usz@google.com |
| Llion Jones⇤ | Aidan N. Gomez⇤ † |  | Łukasz Kaiser⇤ | |
| Google Research | University of Toronto |  | Google Brain | |
| llion@google.com | aidan@cs.toronto.edu | | lukaszkaiser@google.com | |

Illia Polosukhin⇤ ‡

illia.polosukhin@gmail.com

A very approximate timeline

1990 Static Word Embeddings

2003 Neural Language Model

2008 Multi-Task Learning

2015 Attention

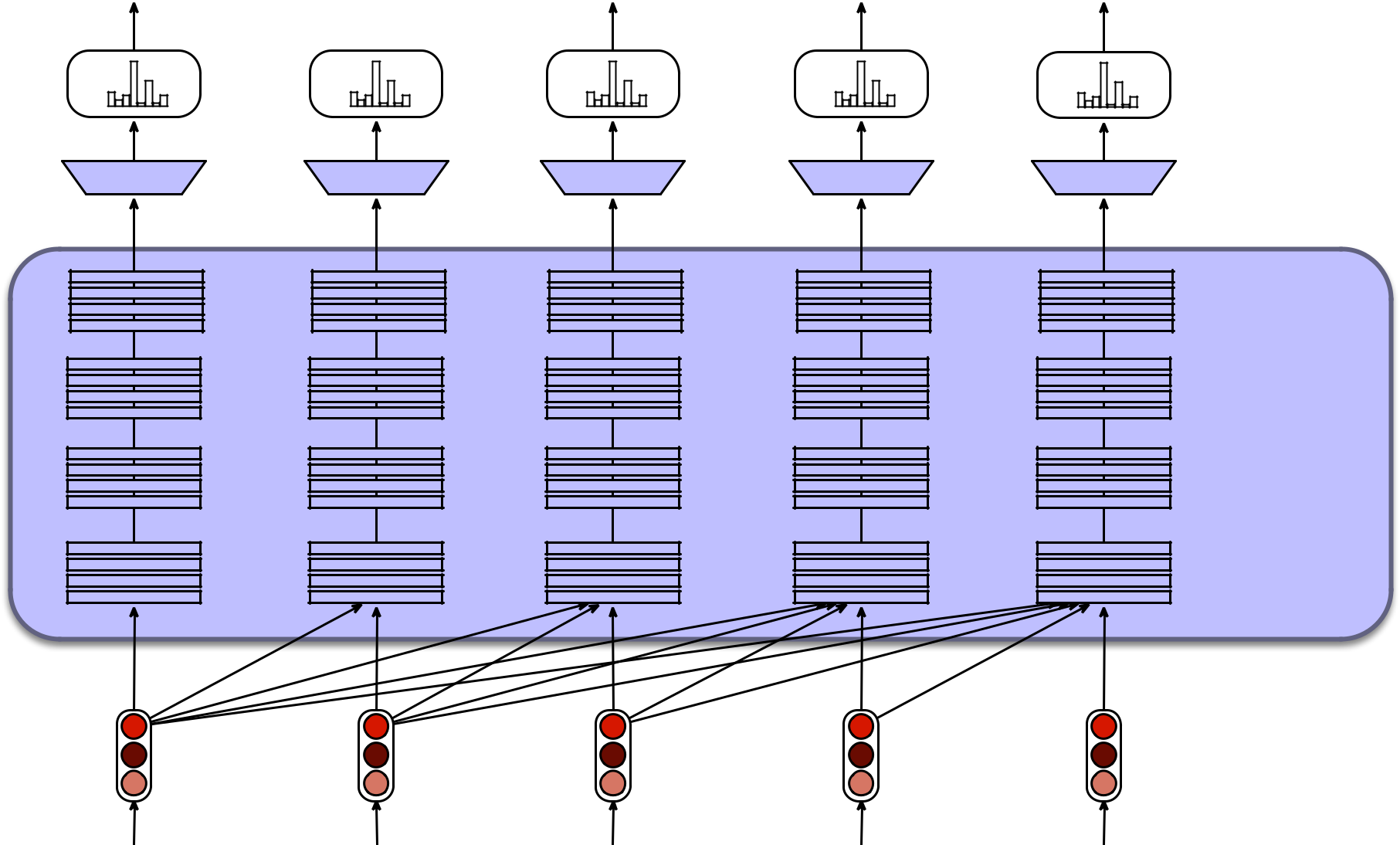
2017 Transformer

2018 Contextual Word Embeddings and Pretraining

2019 Prompting

A picture of a transformer language model

Output long and thanks for



Softmax over

Vocabulary

Linear Layer

Layers of

Transformer

Blocks

Input

Embeddings

all

…

…

…

So long and thanks for

|  |  |
| --- | --- |
| Transformers | Introduction to Large Language |
| and Large | Models |
| Language |  |
| Models |  |



Transformers Attention

and Large

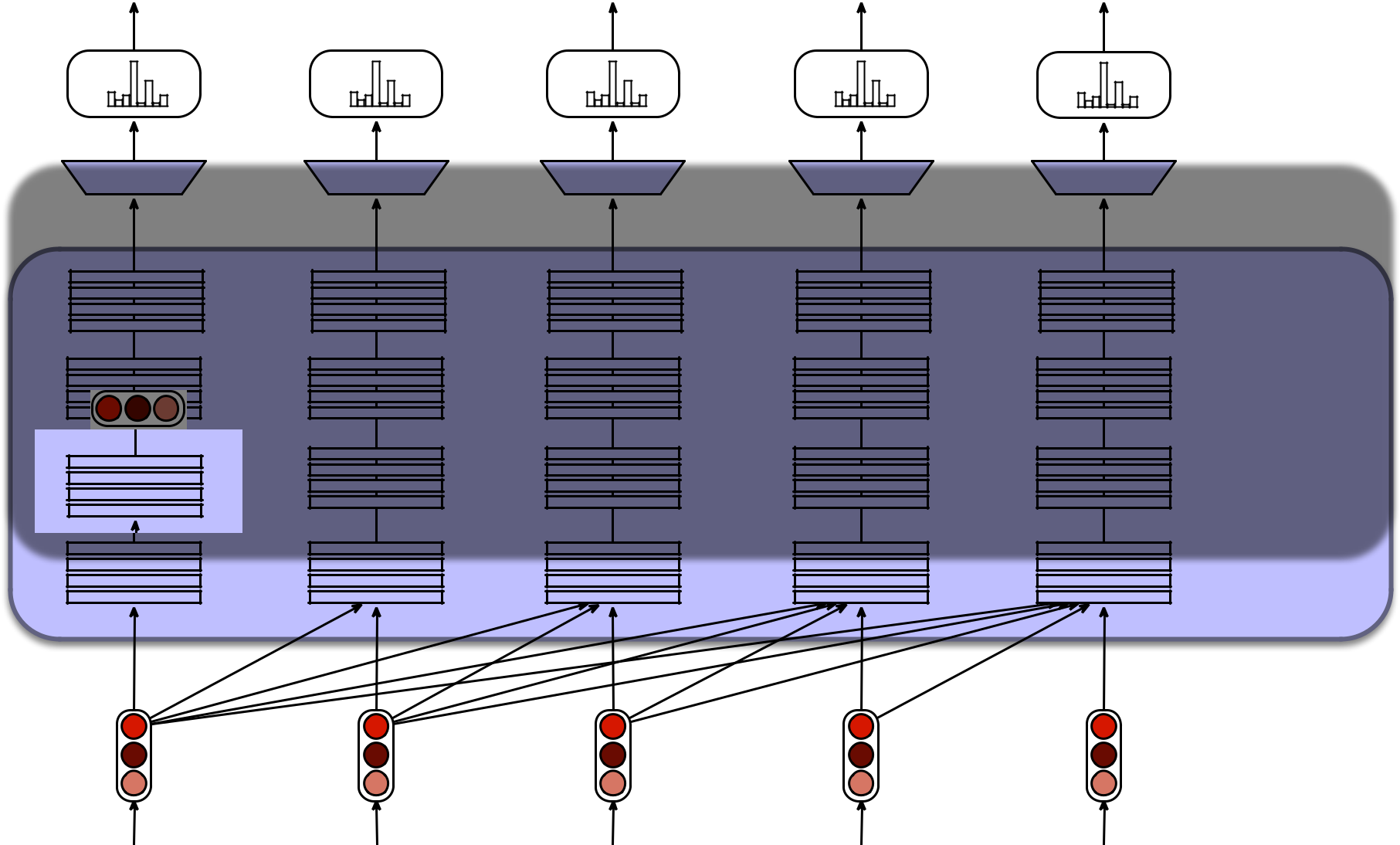
Language

Models

Instead of starting with the big picture

Let's consider the embeddings for an individual word from a particular layer

Output long and thanks for



Softmax over

Vocabulary

Linear Layer

Layers of

Transformer

Blocks

Input

Embeddings

all

…

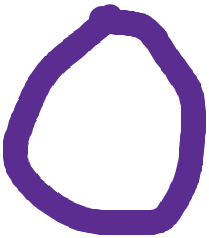
…

…

So long and thanks for

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 street because it was too tired

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

Contextual Embeddings

* Intuition: a representation of meaning of a word should be different in different contexts!
* **Contextual Embedding**: each word has a different vector that expresses different meanings depending on the surrounding words
* How to compute contextual embeddings?
  + **Attention**

Contextual Embeddings

The chicken didn't cross the street because it

What should be the properties of "it"?

The chicken didn't cross the street because it was too **tired** The chicken didn't cross the street because it was too **trafficy**

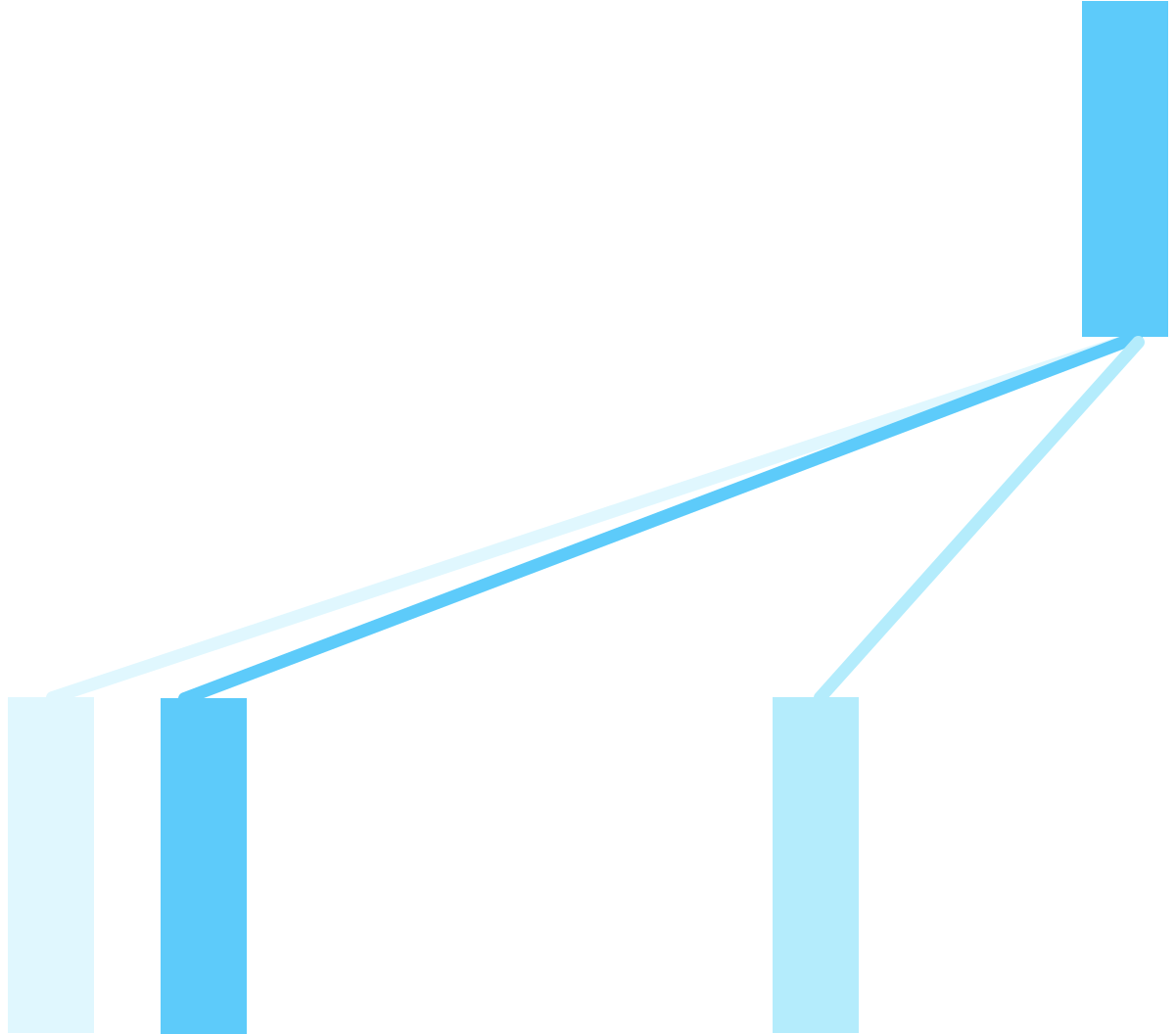
At this point in the sentence, it's probably referring to either the chicken or the street

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

Intuition of attention:



Layer 6

|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

|  |
| --- |
| didn’t |

|  |
| --- |
| animal |

|  |
| --- |
| The |

self-attention distribution

Layer 5

|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

|  |
| --- |
| didn’t |

|  |
| --- |
| animal |

|  |
| --- |
| The |

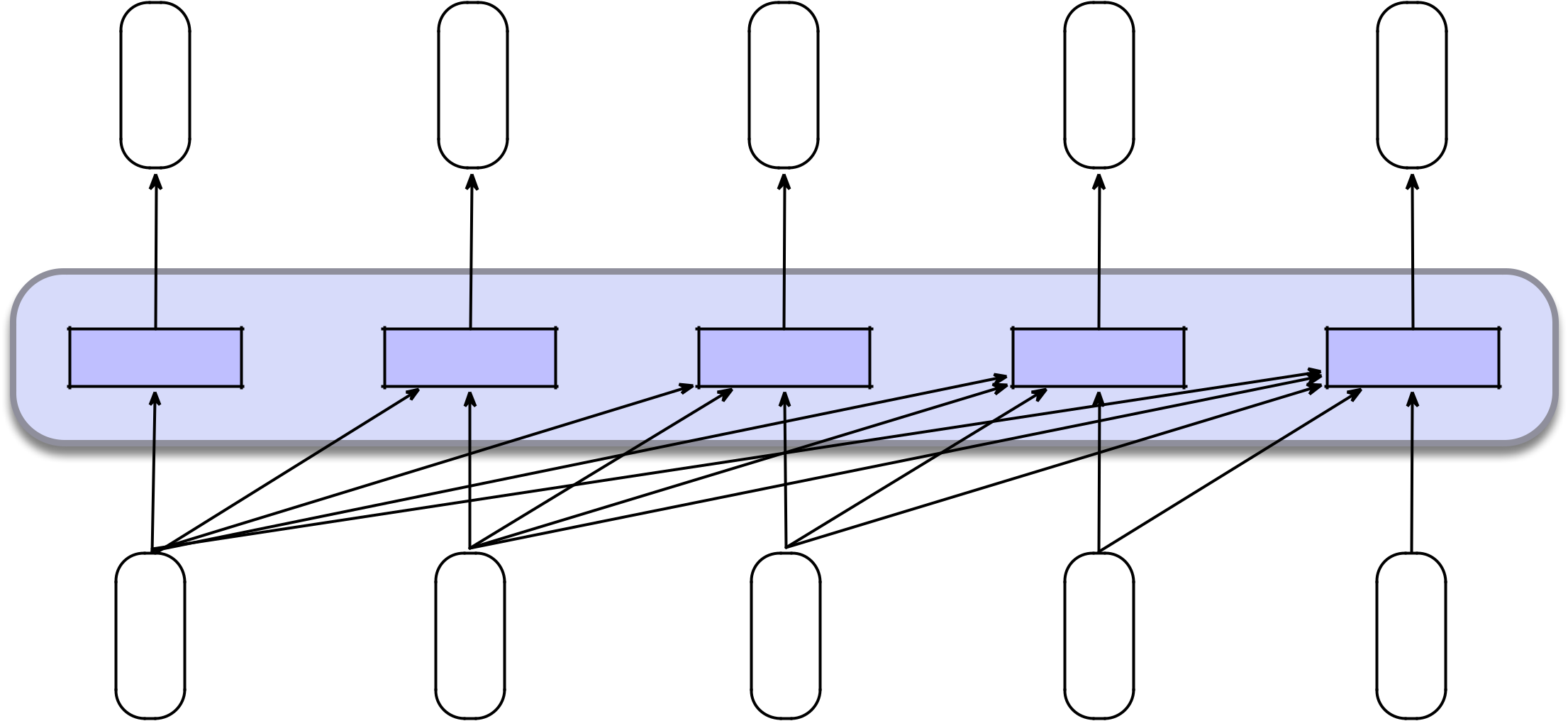
Attention definition

A mechanism for helping compute the embedding for a token by selectively attending to and integrating information from surrounding tokens (at the previous layer).

More formally: a method for doing a weighted sum of vectors.

Attention is left-to-right

**a**1 **a**2 **a**3 **a**4 **a**5



Self-Attention Layer

**x**1 **x**2 **x**3 **x**4 **x**5

Simplified version of attention: a sum of prior words weighted by their similarity with the current word

Given a sequence of token embeddings:

**x**1 **x**2 **x**3 **x**4 **x**5 **x**i

Produce: **a**i = a weighted sum of **x**1 through **x**5 Weighted by their similarity to **x**i

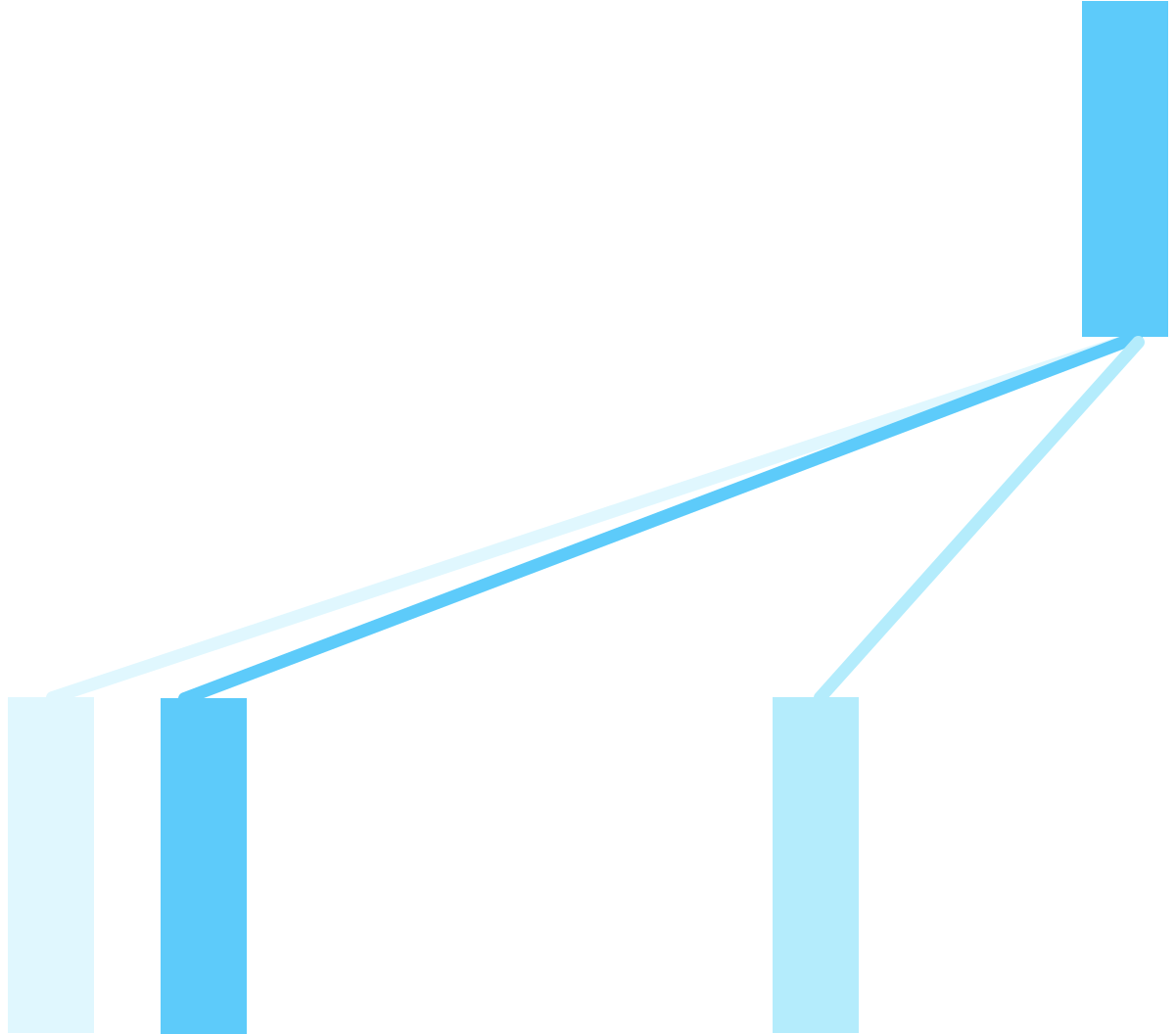
score(xi, x j ) = xi · x j

ai j = softmax(score(xi, x j)) 8 j i

X

ai = ai j xj

Intuition of attention:



Layer 6

|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

|  |
| --- |
| didn’t |

|  |
| --- |
| animal |

|  |
| --- |
| The |

self-attention distribution

Layer 5

|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

|  |
| --- |
| didn’t |

|  |
| --- |
| animal |

|  |
| --- |
| The |

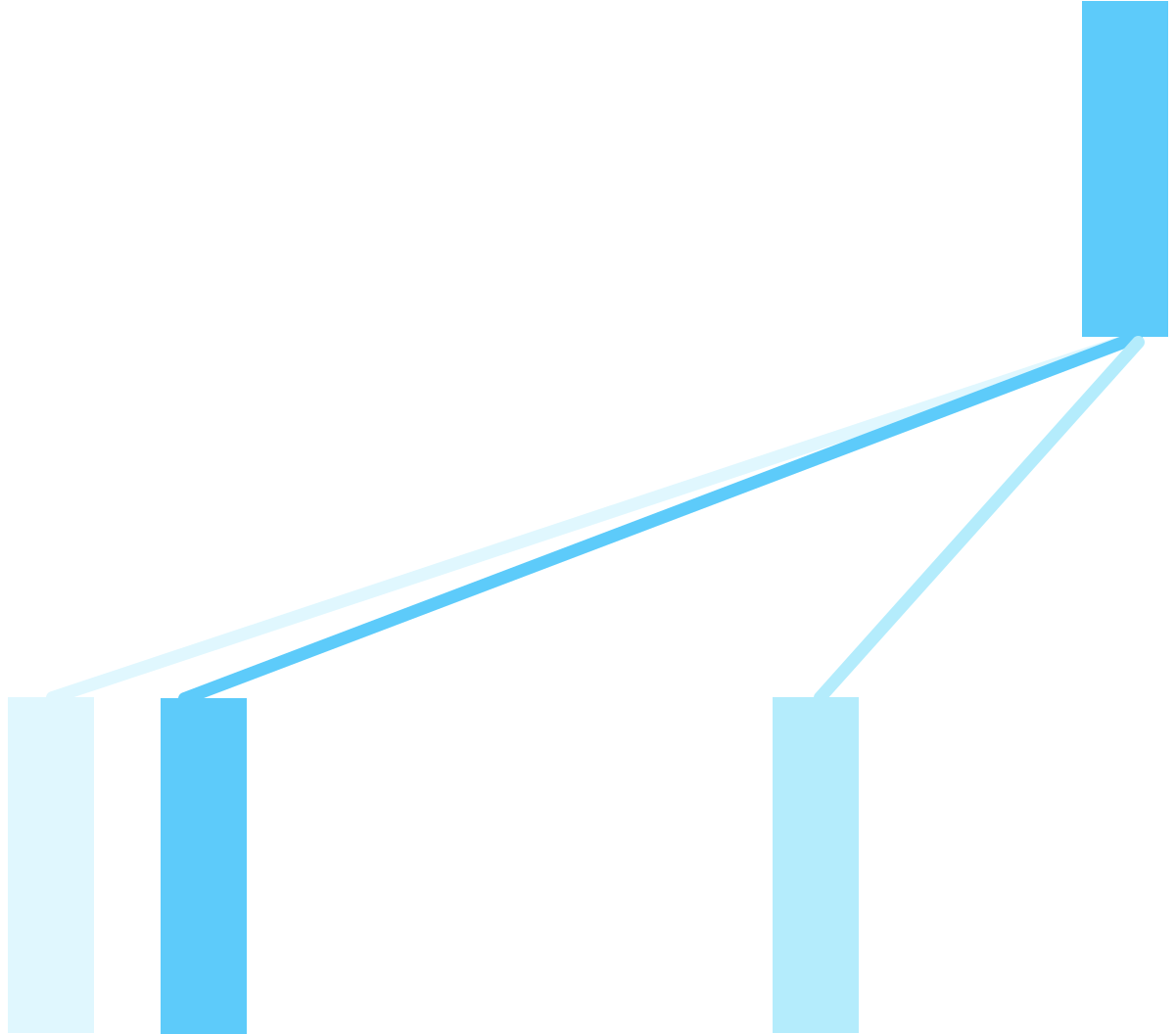
**x1 x2 x3 x4 x5 x6 x7 xi**

Intuition of attention:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer 6 | The | animal | didn’t |  |
|  |  |  |  |

self-attention distribution

**query**



|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

Layer 5

|  |
| --- |
| Theanimaldidn’tcrossthestreetbecauseitwastootired |

**x1 x2 x3 x4 x5 x6 x7 xi**

**values**

Attention

Actually it's slightly more complicated, but I won't get into that

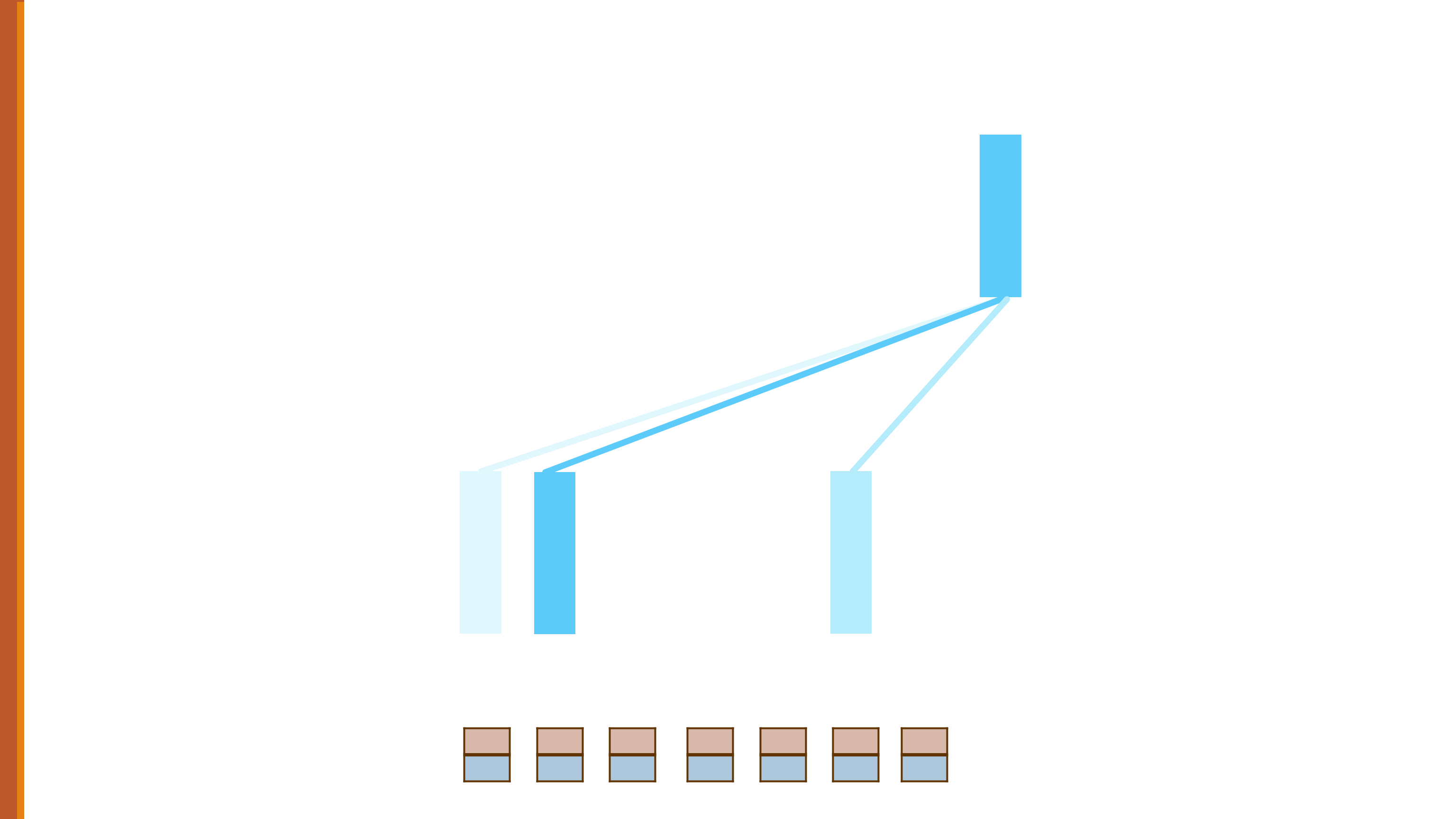
High-level idea: instead of just having a query and a set of values, each embedding actually also has a key

Intuition of attention:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer 6 | The | animal | didn’t |  |
|  |  |  |  |

self-attention distribution

**query**



|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

Layer 5

|  |
| --- |
| tired |

|  |
| --- |
| too |

|  |
| --- |
| was |

|  |
| --- |
| it |

|  |
| --- |
| because |

|  |
| --- |
| street |

|  |
| --- |
| the |

|  |
| --- |
| cross |

|  |
| --- |
| didn’t |

|  |
| --- |
| animal |

|  |
| --- |
| The |

**keys**

**values**

**x1**

k

v

**x2**

k

v

|  |  |
| --- | --- |
| **x3** | **x4** |
| k | k |
| v | v |

**x5**

k

v

**x6**

k

v

**x7 xi**

k

v

Summary

Attention is a method for enriching the representation of a token by incorporating contextual information

The result: the embedding for each word will be different in different contexts!

Contextual embeddings: a representation of word meaning in its context.

Transformers Attention

and Large

Language

Models

Transformers and Large Language Models

The rest of the transformer applied to language modeling

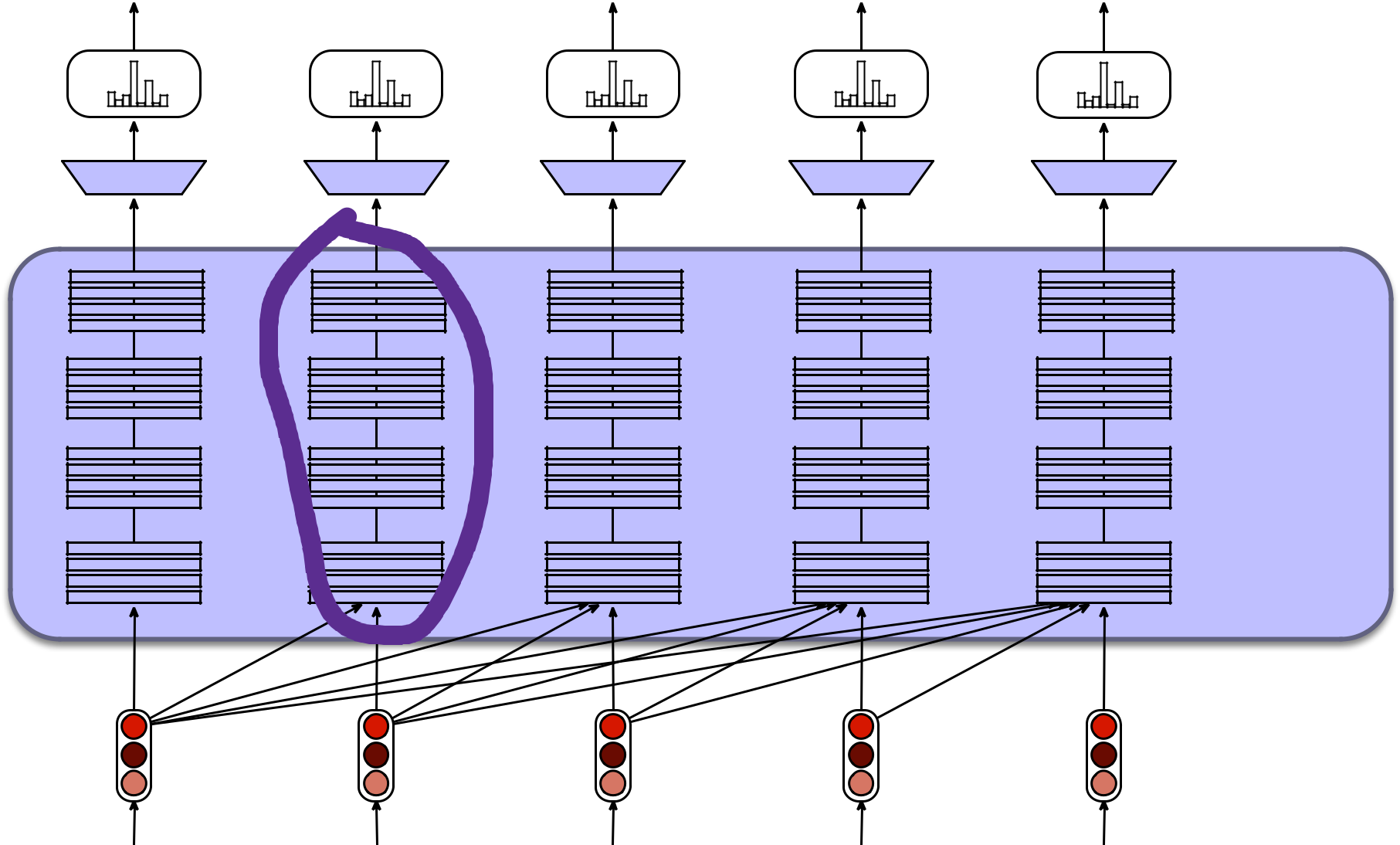
The transformer

Attention is just part of computing embeddings in a transformer.

Let's see more of the mechanism

Reminder: transformer language model

Output long and thanks for



Softmax over

Vocabulary

Linear Layer

Layers of

Transformer

Blocks

Input

Embeddings

all

…

…

…

So long and thanks for

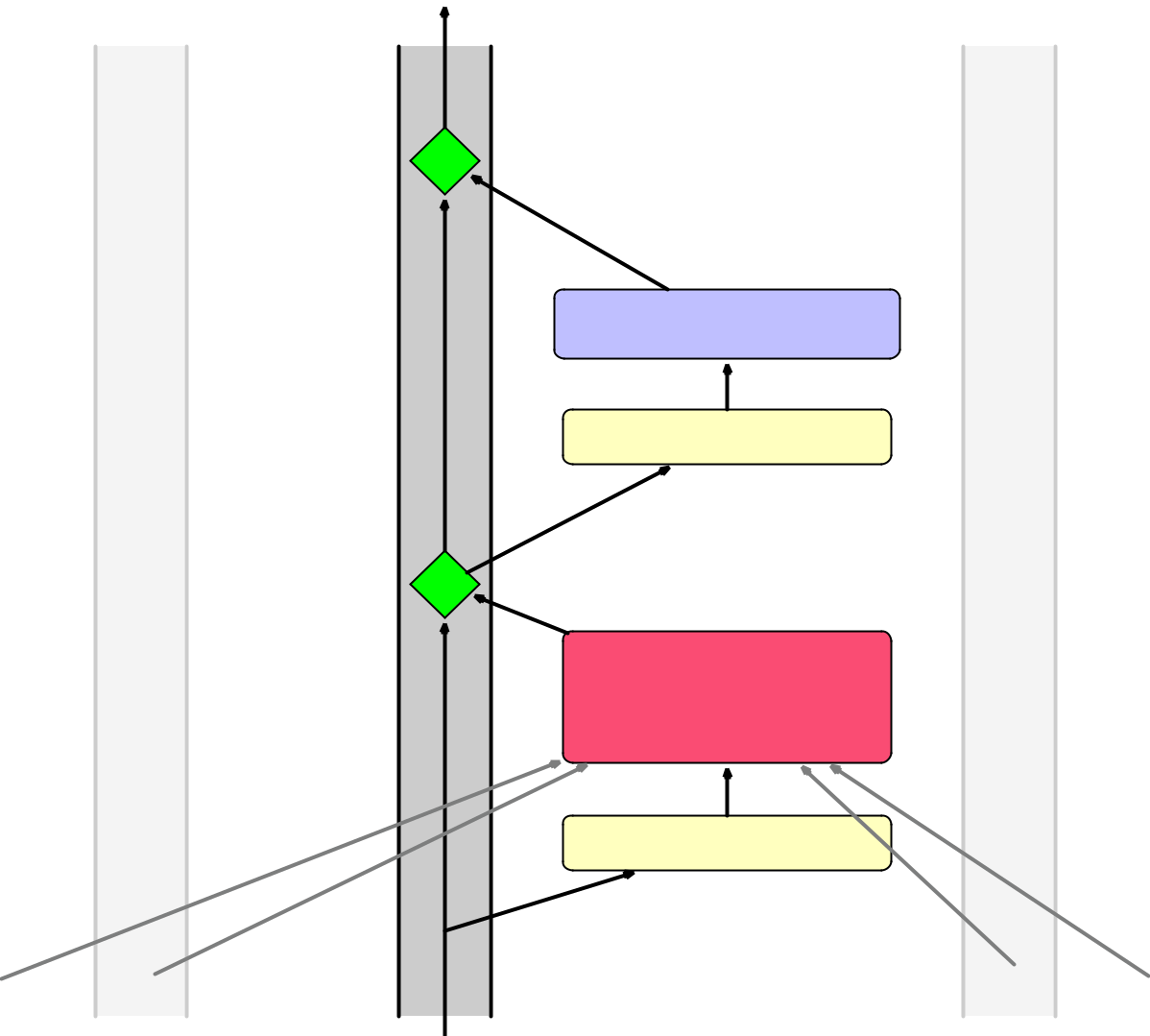


The residual stream: each token gets passed up and modified

*hi-1*

…

*hi* *hi+1*



+

Feedforward

Layer Norm

…

+

MultiHead

Attention

Layer Norm

*xi-1* *xi* *xi+1*

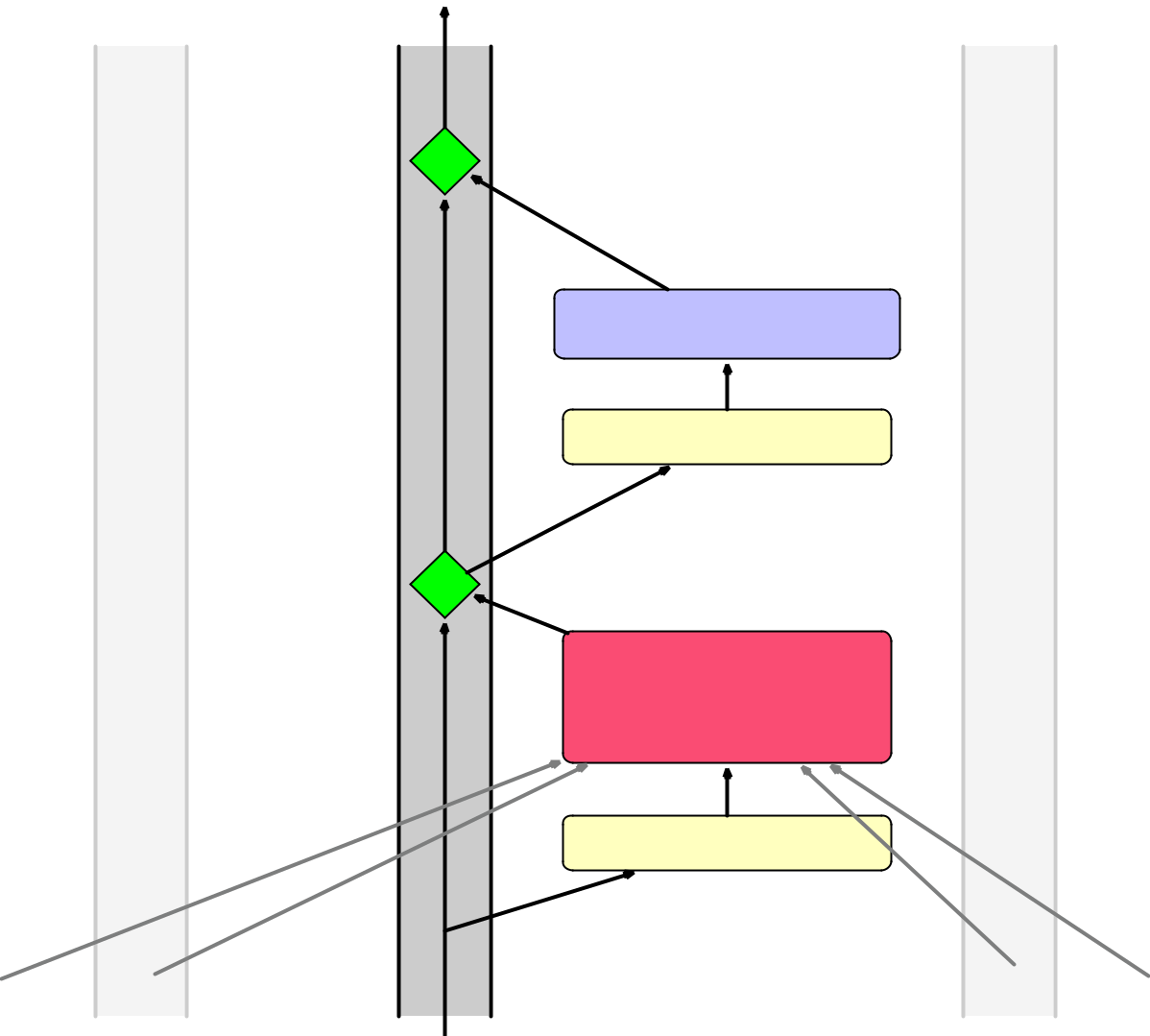


We'll need nonlinearities, so a feedforward layer

*hi-1*

…

*hi* *hi+1*



+

Feedforward

Layer Norm

…

+

MultiHead

Attention

Layer Norm

*xi-1* *xi* *xi+1*

A transformer is a stack of these blocks

Block 2

Block 1

*hi-1*

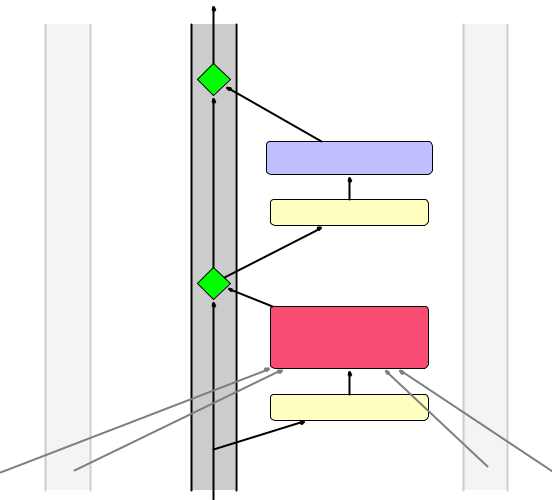
…

*xi-1*

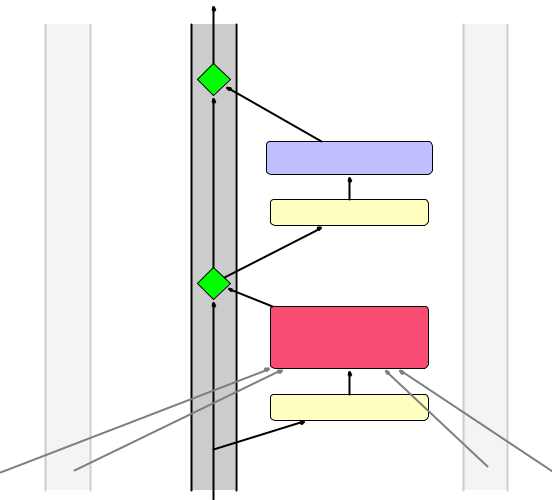
*hi-1*

…

*hi* *hi+1*



|  |  |  |
| --- | --- | --- |
| + |  |  |
|  | Feedforward |  |
|  | Layer Norm |  |
| + | … |  |
|  |  |
|  | MultiHead |  |
|  | Attention |  |
|  | Layer Norm |  |
| *xi* | *xi+1* |  |
| *hi* | *hi+1* |  |



+

Feedforward

Layer Norm

…

+

MultiHead

Attention

Layer Norm

*xi-1* *xi* *xi+1*

Inputs

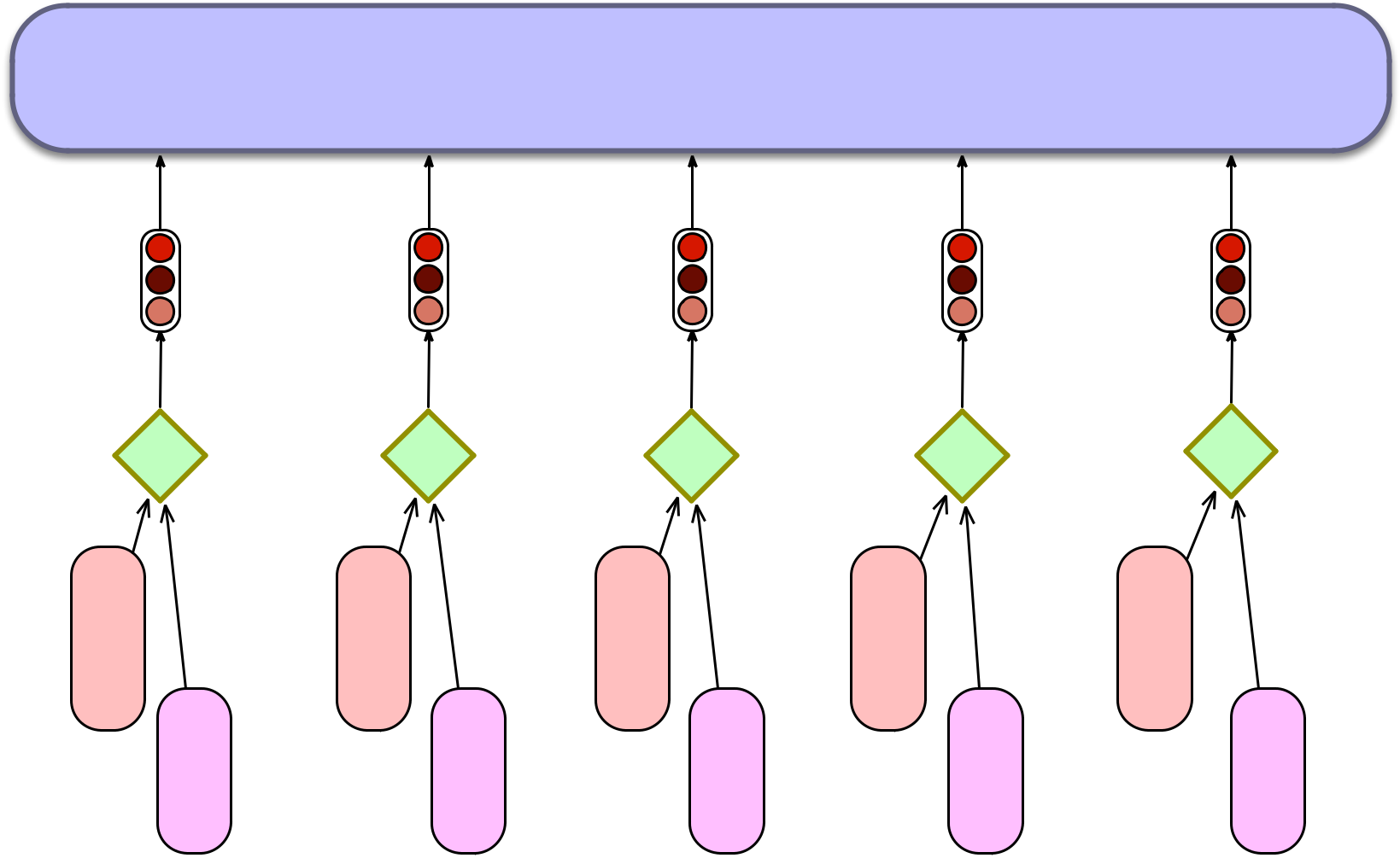
X = Composite

Embeddings

(word + position)

|  |  |  |
| --- | --- | --- |
|  | + |  |
| Word | Janet |  |
| Embeddings |  |
| Position | 1 |  |
| Embeddings |  |  |

Transformer Block



|  |
| --- |
| + 5 |

|  |
| --- |
| bill |

|  |
| --- |
| + 4 |

|  |
| --- |
| the |

|  |
| --- |
| + 3 |

|  |
| --- |
| back |

|  |
| --- |
| + 2 |

|  |
| --- |
| will |

Janet will back the bill

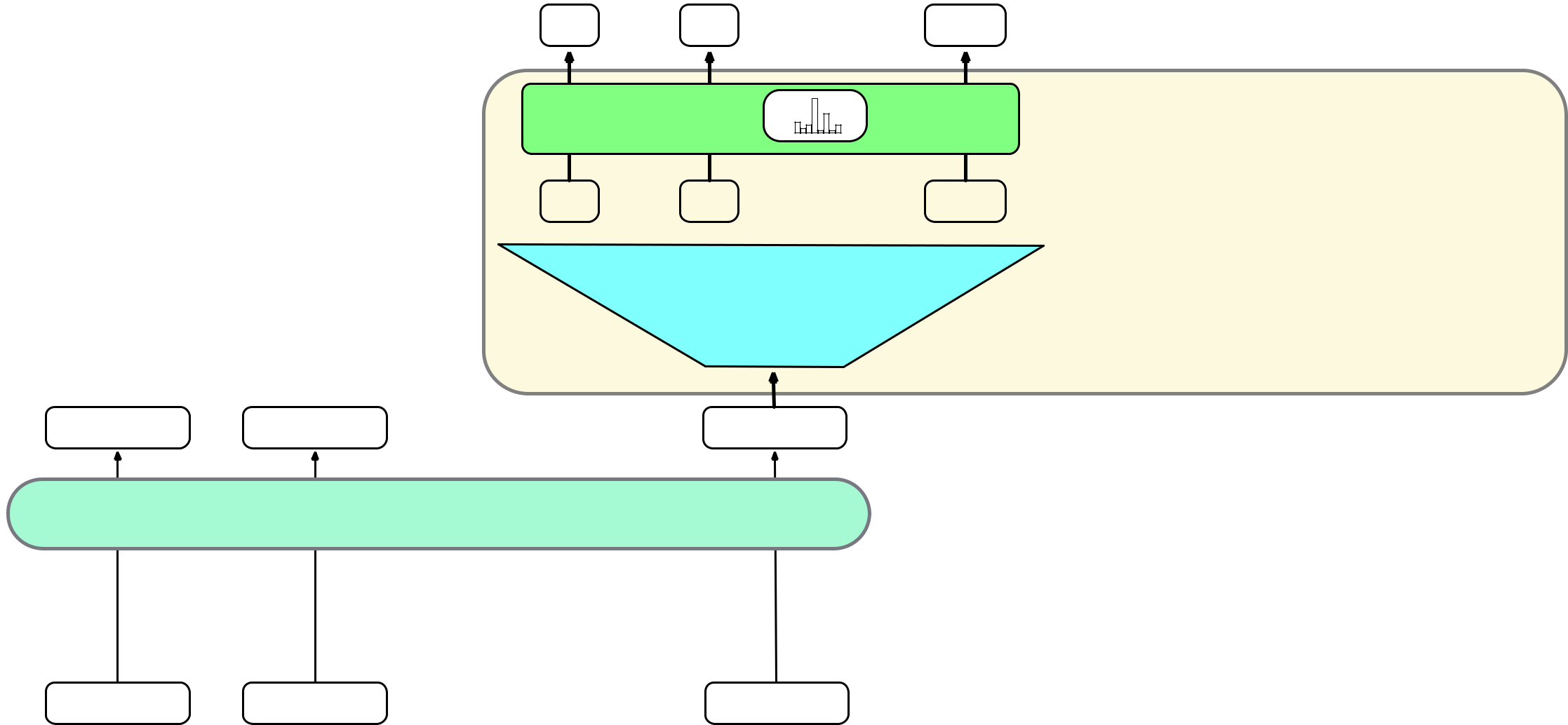
Language modeling head

**Language Model Head**

takes hLN and outputs a

distribution over vocabulary V

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| y1 | y2 | … | y|V| | Word probabilities | 1 x |V| |  |
|  |  | … |  | Softmax over vocabulary V | |  |
| u1 | u2 | u|V| | Logits 1 x |V| |  |  |
|  | Unembedding | |  | Unembedding layer | d x |V| |  |
|  | layer = **ET** | |  |  |
|  |  |  |  |  |



**hL1** **hL2** **hLN** 1 x d

Layer L

Transformer

Block

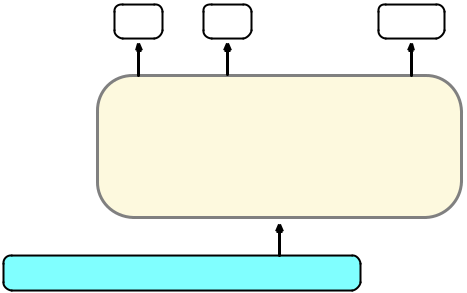
…

w1 w2 wN



The final transformer model

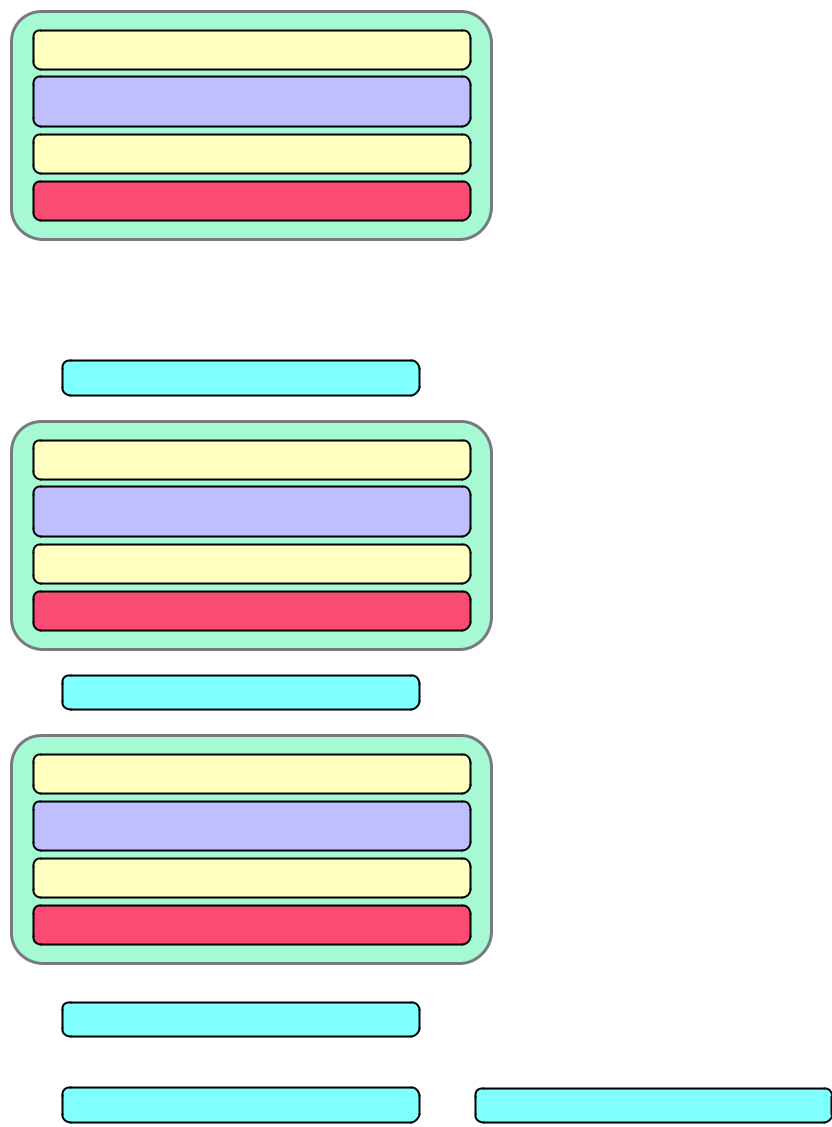
|  |  |  |  |
| --- | --- | --- | --- |
| Sample token to generate |  | wN+1 |  |
| at position N+1 |  |  |  |
| Token probabilities | y1 | y2 | … y|V| |



Language Model

Head

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **h1 h2** | **…** | **hN** |  |
|  | Layer Norm | | |  |
| Layer L | Feedforward | | |  |
|  | Layer Norm | | |  |
|  | Multihead Self-Attention | | |  |
|  |  | … |  |  |
|  | **h1 h2** | **…** | **hN** |  |
|  | Layer Norm | | |  |
| Layer 2 | Feedforward | | |  |
|  | Layer Norm | | |  |
|  | Multihead Self-Attention | | |  |
|  | **h1 h2** | **…** | **hN** |  |
|  | Layer Norm | | |  |
| Layer 1 | Feedforward | | |  |
|  | Layer Norm | | |  |
|  | Multihead Self-Attention | | |  |
| X | **x1 x2** | **…** | **xN** |  |
| Add token + position embeddings | **E**[w1] **E**[w2] … **E[wN]** + | | | **P1 P2…PN** |



Input tokens w1 w2 … wN

Transformers and Large Language Models

The rest of the transformer applied to language modeling

Pretraining (and how to train

Large transformers for language

Language modeling)

Models

Pretraining

The big idea that underlies all the amazing performance of language models

First **pretrain** a transformer model on enormous amounts of text

Then **apply** it to new tasks.

Intuition of language model training

We just train them to predict the next word!

1. Take a corpus of text
2. At each time step *t*
   1. ask the model to predict the next word
   2. train the model using gradient descent to minimize the error in this prediction

Intuition of language model training: loss

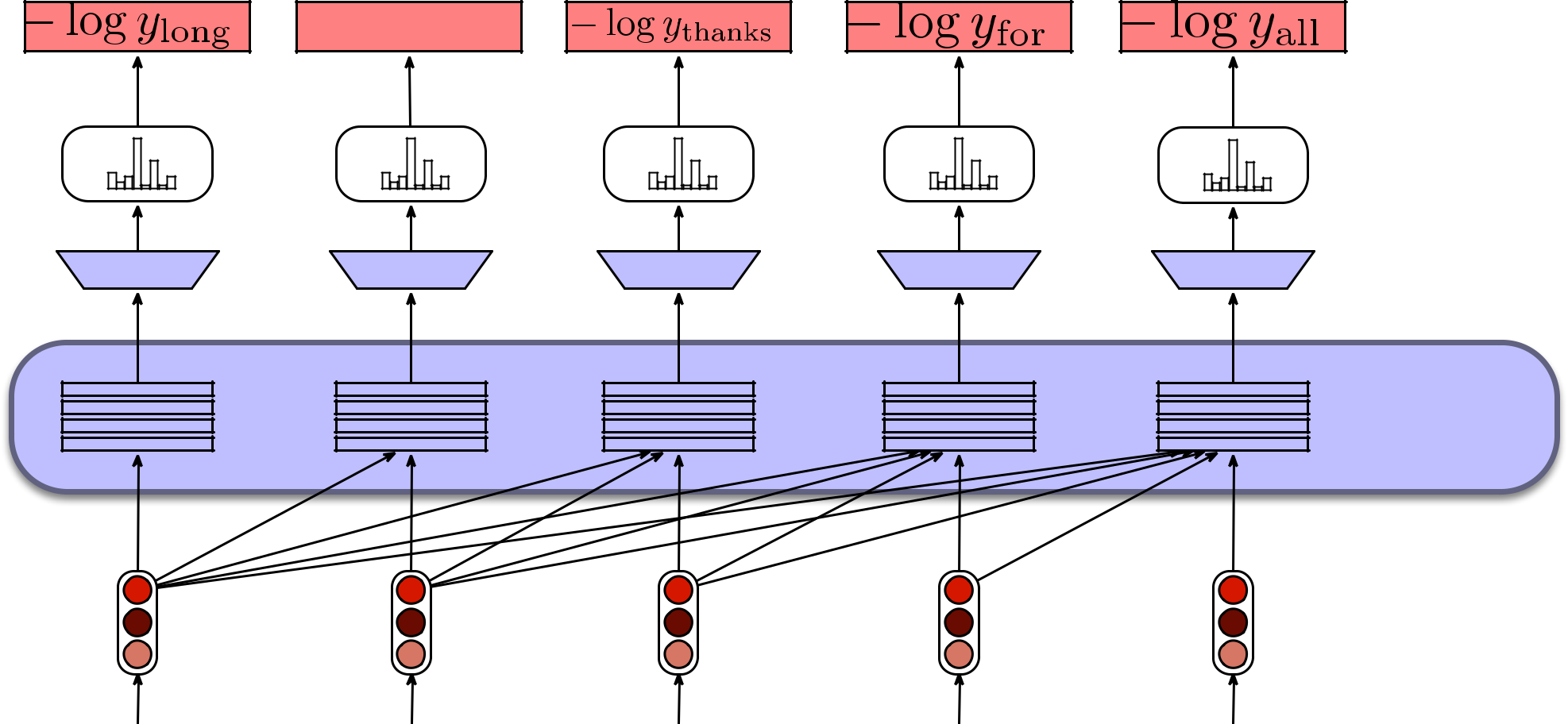
* Same loss function: **cross-entropy loss**
  + We want the model to assign a high probability to true word *w*
  + = want loss to be high if the model assigns too low a probability to w
* CE Loss: The negative log probability that the model assigns to the true next word w
* If the model assigns too low a probability to w
* We move the model weights in the direction that assigns a higher probability to w

Training a transformer language model

Next word long and thanks for all



Loss



Softmax over

Vocabulary

Linear Layer

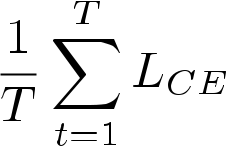
Transformer

Block

Input

Embeddings

* =



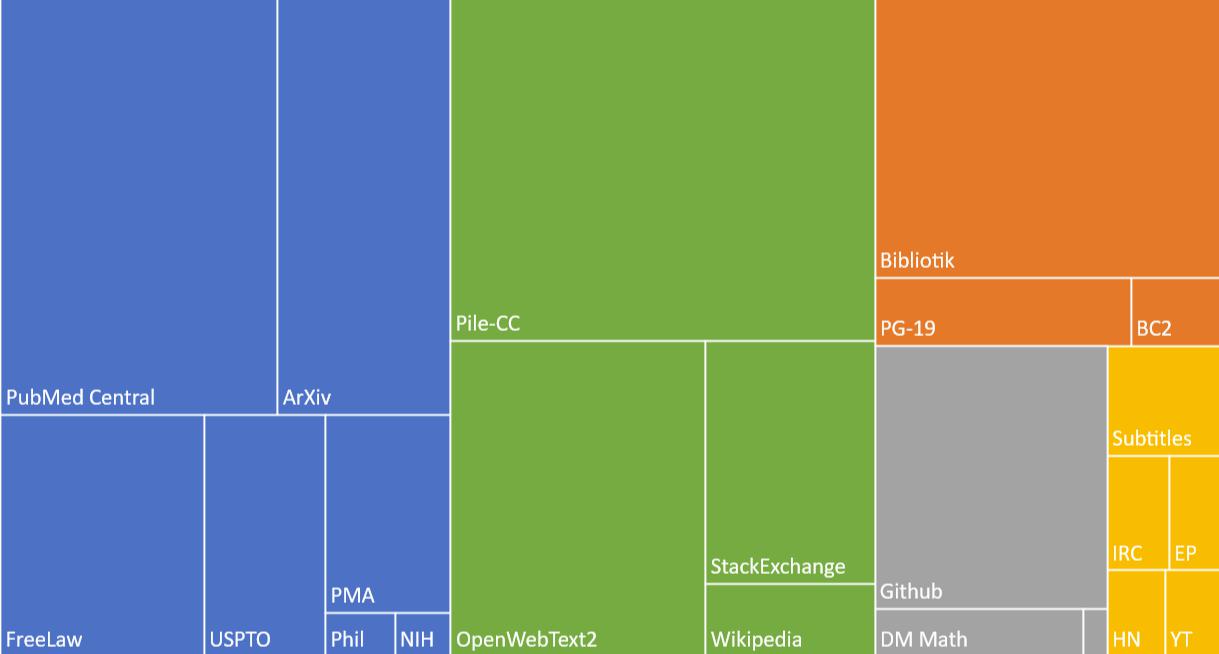
…

…

So long and thanks for

Pretraining Data: mostly from the web (Common Crawl)

The Pile:



What does a model learn from pretraining?

* There are canines everywhere! One dog in the front room, and two dogs
* It wasn't just big it was enormous
* The author of "A Room of One's Own" is Virginia Woolf
* The doctor told me that he
* The square root of 4 is 2



Big idea

Text contains enormous amounts of knowledge

Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much

Pretraining (and how to train

Large transformers for language

Language modeling)

Models

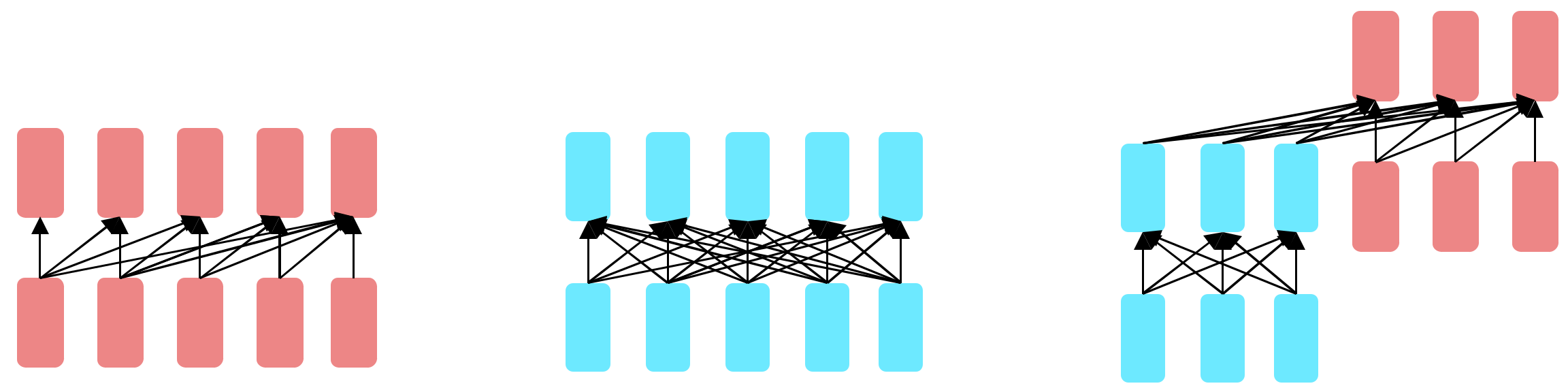
|  |  |  |
| --- | --- | --- |
| Large | Large Language Models: |  |
| Applying pretrained models to |  |
| Language | new tasks |  |
| Models |  |  |



Big idea

Many tasks can be turned into tasks of predicting words!

Three architectures for large language models



**Decoders**

**Encoders**

**Encoder-decoders**

GPT, Claude,

BERT family,

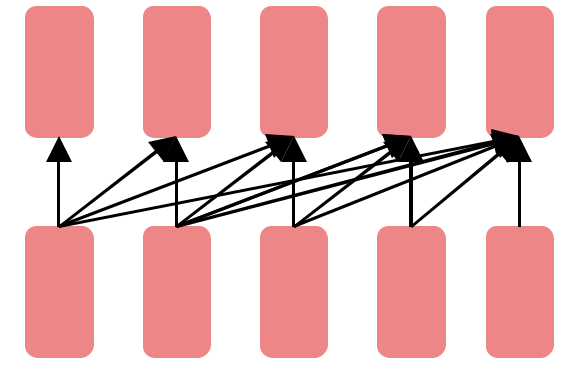
Flan-T5, Whisper

Llama 2

HuBERT

Mixtral

Decoders

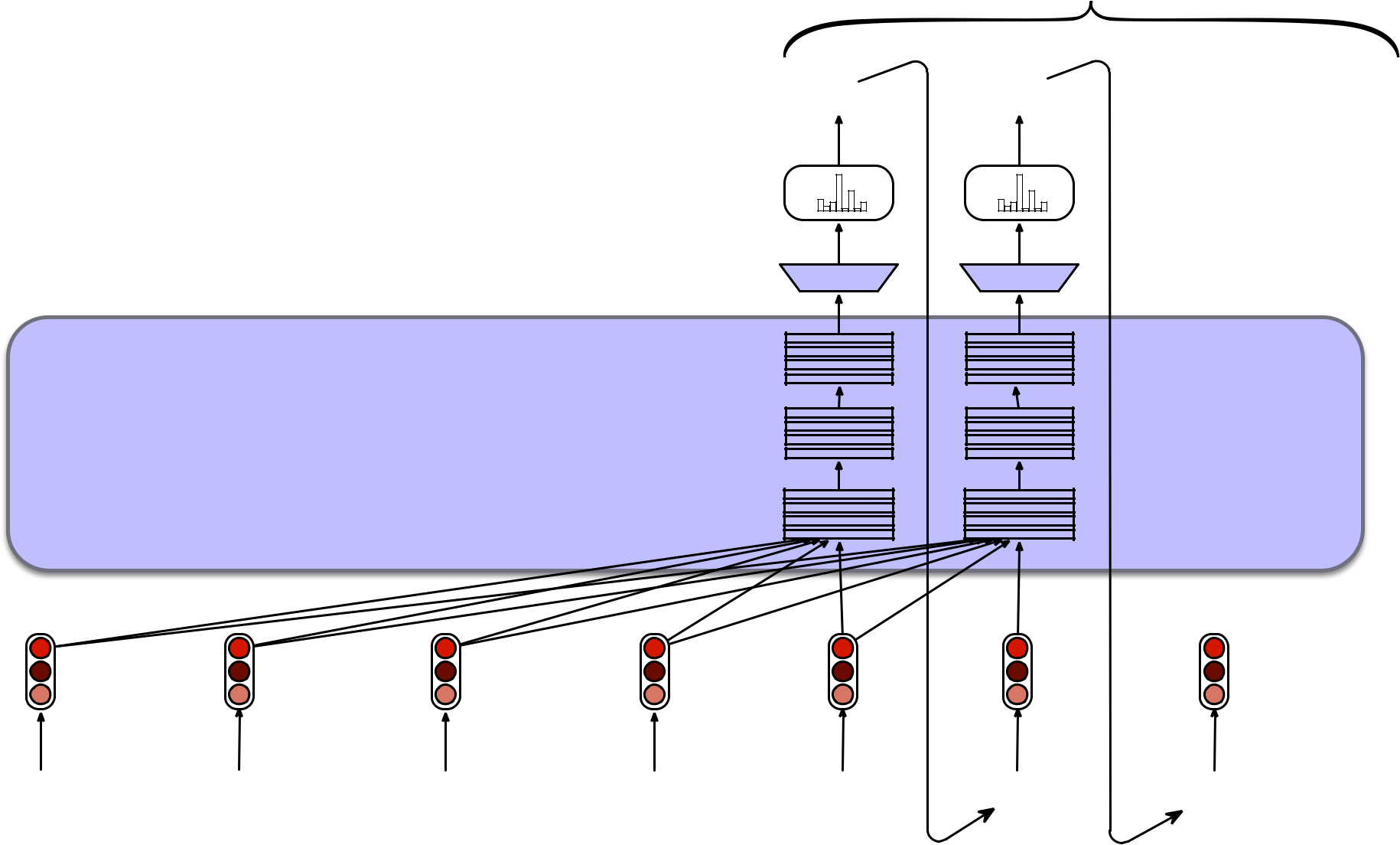


Also called:

* Causal LLMs
* Autoregressive LLMs
* Left-to-right LLMs
* Predict words left to right

Conditional Generation: Generating text conditioned on previous text!

Completion Text



all the

Sample from Softmax

linear layer

Transformer

Blocks

…

Input

Embeddings

So long and thanks for all the



Prefix Text

Framing lots of tasks as conditional generation

Sentiment analysis: “I like Jackie Chan”

1. We give the language model this string:

The sentiment of the sentence "I like Jackie Chan" is:

1. And see what word it thinks comes next:

P(positive|The sentiment of the sentence “I like Jackie Chan” is:)

P(negative|The sentiment of the sentence “I like Jackie Chan” is:)

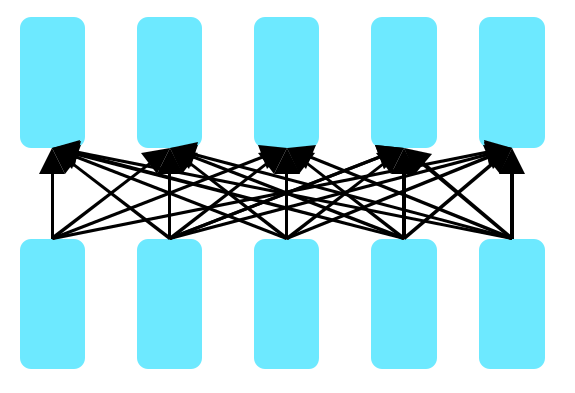
Framing lots of tasks as conditional generation

QA: “Who wrote The Origin of Species”

1. We give the language model this string:
   1. Who wrote the book ‘‘The Origin of Species"? A:
2. And see what word it thinks comes next:

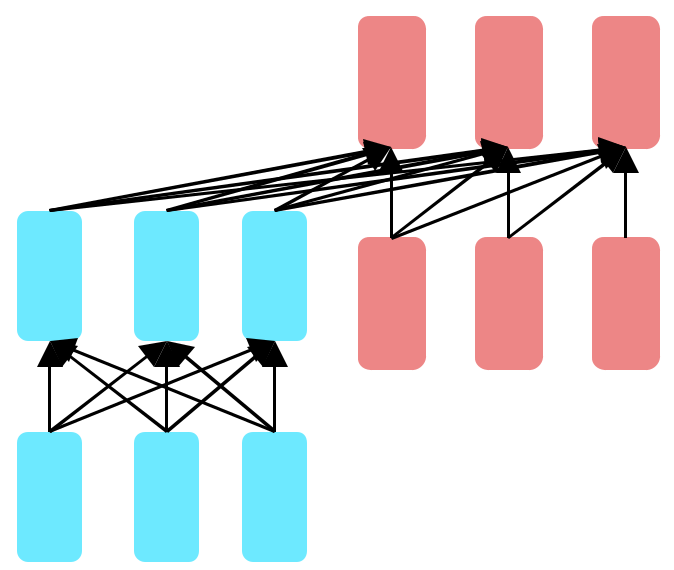
P(w|Q: Who wrote the book “The Origin of Species”? A:)

Encoders



Many varieties!

* Popular: Masked Language Models (MLMs)
* BERT family
* Trained by predicting words from surrounding words on both sides
* Are usually **fine-tuned** (trained on supervised data) for classification tasks.

Encoder-Decoders

* Trained to map from one sequence to another
* Very popular for:
  + machine translation (map from one language to another)
  + speech recognition (map from acoustics to words)

Many more things I didn't talk about

Instruction Fine-tuning

Preference Alignment

Prompt Engineering

Where to learn these? CS224N!

|  |  |  |
| --- | --- | --- |
| Large | Large Language Models: |  |
| Applying pretrained models to |  |
| Language | new tasks |  |
| Models |  |  |



Harms of Large Language

Large Models

Language

Models

Hallucination



Current research direction to address hallucination

Retrieval-Augmented Generation (RAG)

Use information retrieval to **retrieve** some passages from a high-quality source

Then use a language model to **generate** an answer from those passages

Copyright



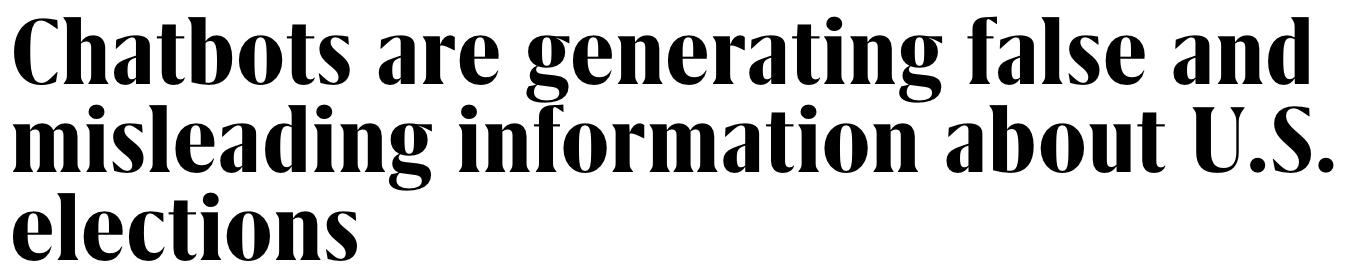
Privacy



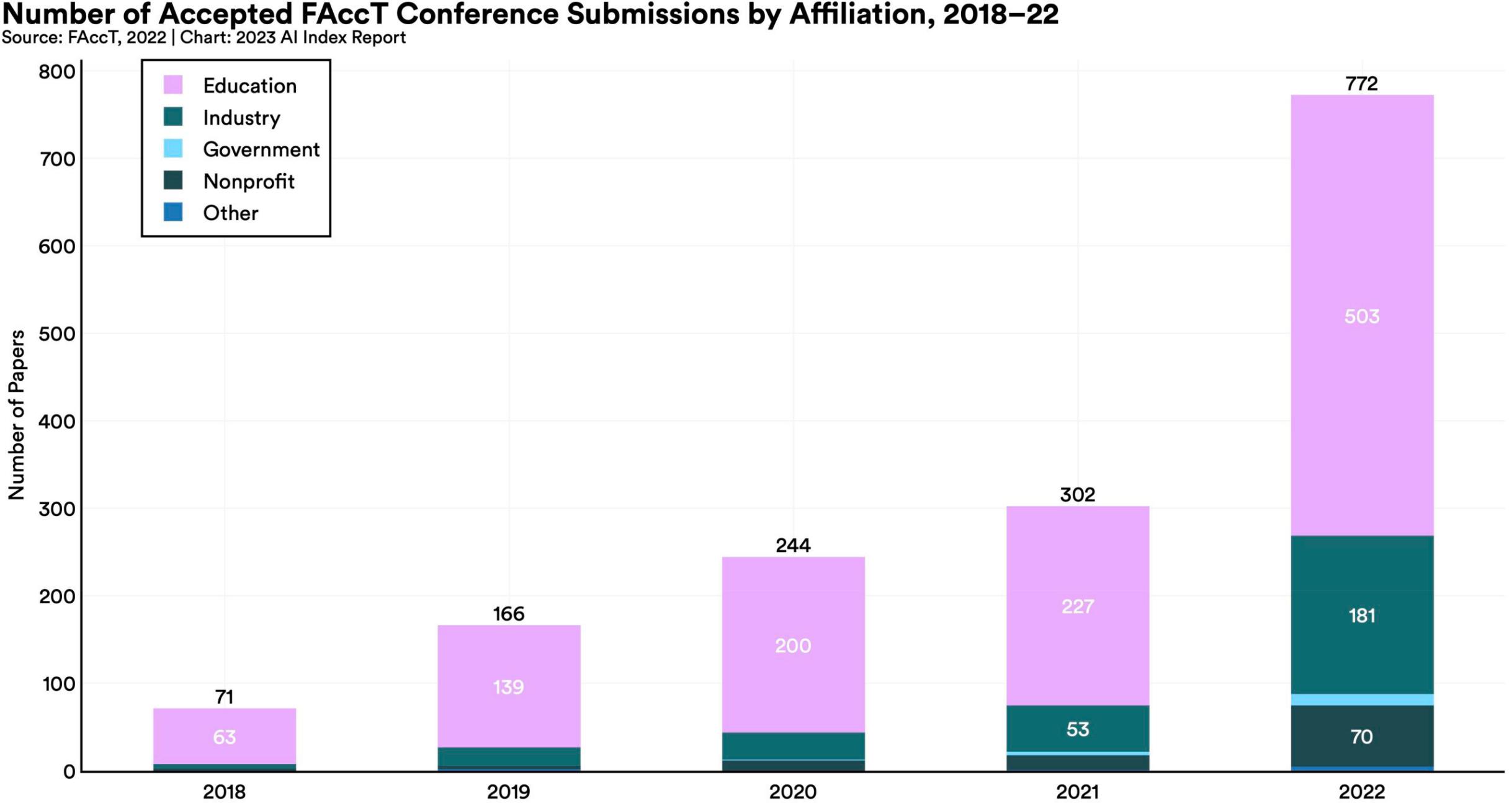
Toxicity and Abuse



Misinformation



Vast growth in interest in Ethics of LLMs and AI



HAI: AI Index 2023

Harms of Large Language

Large Models

Language

Models



Large

Language

Models

Our last class together!

Learning goals

Write regular expressions for text tasks

Apply the edit distance algorithm

Build a supervised classifier

Build a search engine

Work with neural word embeddings Train a neural network

Build a recommendation engine Build a chatbot

Prompt a large language model

What's next? Spring 2024 NLP adjacent courses

**CS224N: Natural Language Processing with Deep Learning (Chris Manning)**

Algorithmic internals: transformers, GPT, parsing, machine translation and other applications. More of the gory details! More math, more machine learning

**CS224C: NLP for Computational Social Science (Diyi Yang)**

…machine learning and theories from social science to study human behaviors and important societal questions at scale. NLP, social networks, causal inference, application to social topics like hate speech, misinformation, and social movements.

**CS 224S: Spoken Language Processing: (Andrew Maas)**

Introduction to spoken language technology with an emphasis on dialogue and conversational systems.

**CS 336: Language Modeling from Scratch (Tatsu Hashimoto and Percy Liang)**65

every aspect of language model creation, including data collection and cleansing for pre-training, transformer model construction, model training, and evaluation before deployment. Application required.

**CS 246: Mining Massive Data Sets (Jure Leskovec)**

Next year courses!

**CS 224V: Conversational Virtual Assistants with Deep Learning (Monica Lam)**

Topics include: (1) growing LLMs' knowledge, (2) stopping LLMs from hallucination

1. experimentation and evaluation of conversational assistants based on LLMs, (5) controlling LLMs to achieve tasks, (6) persuasive LLMs, (7) multilingual assistants, and
2. combining voice and graphical interfaces.

**CS329X Human Centered NLP) (Diyi Yang)**

human-centered design thinking in NLP, human-in-the-loop algorithms, fairness, and accessibility.

**CS329R Race and NLP (Dan Jurafsky and Jennifer Eberhardt)**

Integrate methods from natural language processing with social psychological66

perspectives on race to build practical systems that address significant societal issues

Fun courses outside of CS

**Spring:**

**Linguist 173:** Invented Languages

**Linguist 134A**: The Structure of Discourse

**Linguist 156**: Language, Gender, and Sexuality

**COMM 154:** The Politics of Algorithms

**Next year:**

**Linguistics 150**: Language and Society 67

**Linguistics 130a**: Introduction to semantics & pragmatics