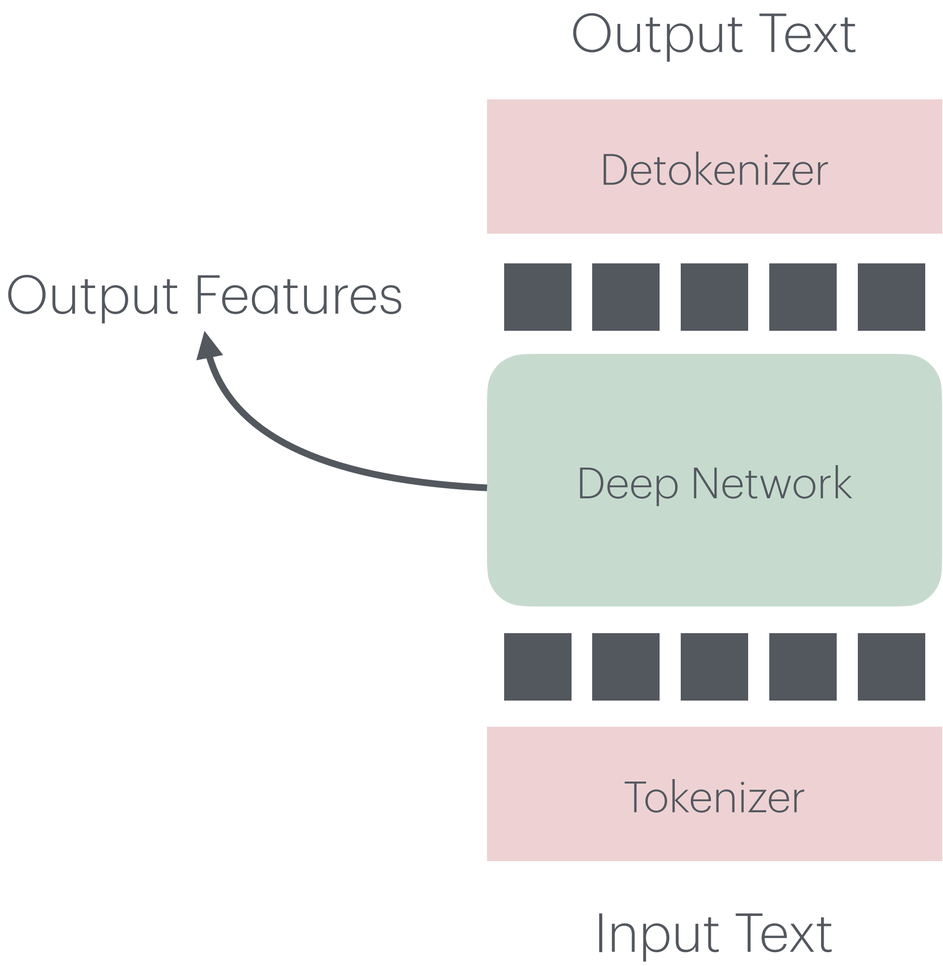
Architectures

Philipp Krähenbühl, UT Austin

LLM - Basics

- Tokenizer
- Deep Network
  - Encoder-Decoder (original transformer)
  - Encoder-only
  - Decoder-only
  - Sequence Models

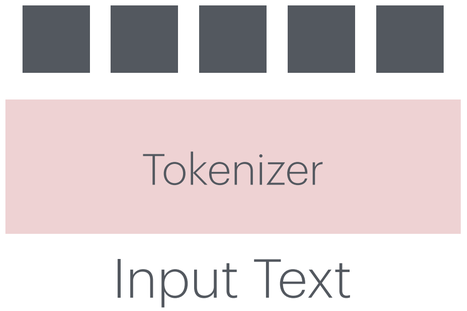


Tokenization

- Convert text into inputs / outputs of network
- What inputs a network understand?
  - Continuous vectors
- What outputs a network understand?
  - Continuous vectors
  - Distributions over categories

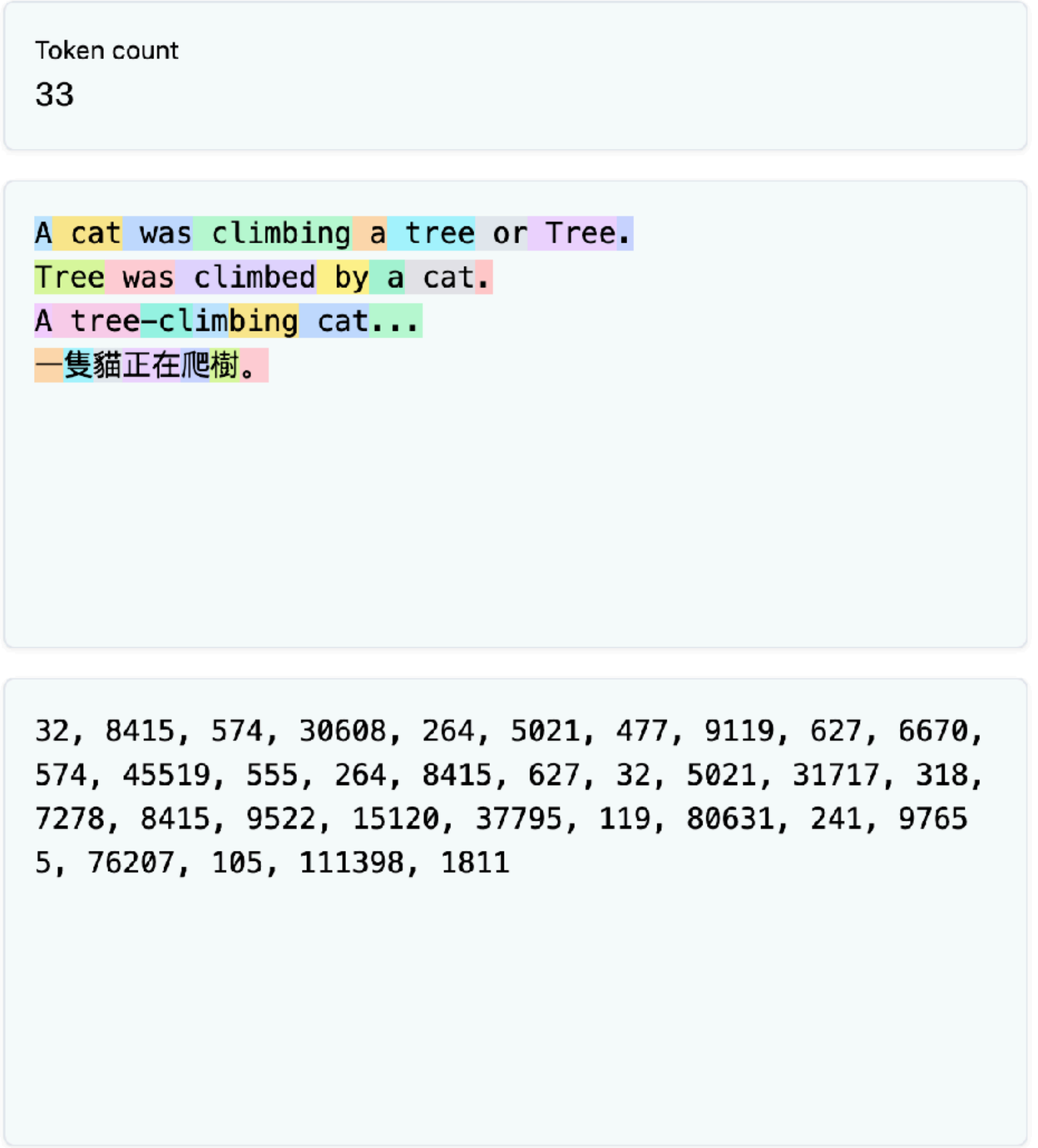
What is text?

- A sequence of characters
  $c_1 \ldots c_N \in \{0 \ldots 255\}$
  - English language
    - 1 symbol = 1 character
  - Other languages (UTF-8)
    - 1 symbol = 1-4 characters



Tokenization

- Demo https://tiktokenizer.vercel.app/?model=meta-llama%2FMeta-Llama-3-70B



Tokenization

Character-level tokenizer V0

- Convert input characters $c_1 \ldots c_N$ to
  - integers $\text{ord}(c_i)$
- Bad idea
  - Integers do not have an order
  - Math on ordinal values does not make sense

    $\text{ord}('a') + \text{ord}('b') \neq \text{ord}('c')$

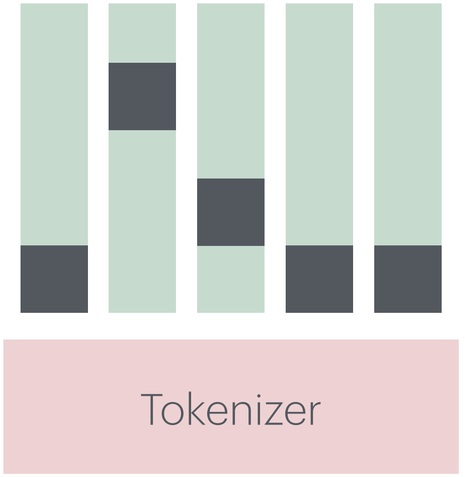
Tokenization

Character-level tokenizer V0.1

- Convert input characters $c_1 \ldots c_N$ to
  - Use one-hot encodings

$$\begin{bmatrix} \text{ord}(c_i) = 1 \\ \text{ord}(c_i) = 2 \\ \text{ord}(c_i) = 3 \\ \cdots \end{bmatrix}$$

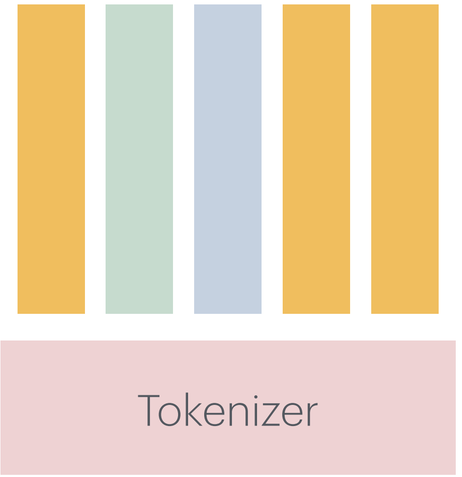
- Improvement
  - Combine with first layer in LLM



Tokenization

Character-level tokenizer

- Convert input characters $c_1 \ldots c_N$ to
  - Learned character-level embeddings $e_1, \ldots, e_{256}$
  - Tokenize $e_{\text{ord}(c_i)}$

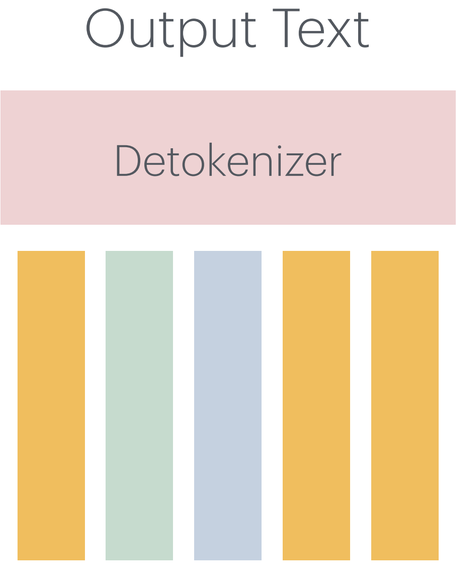


`torch.nn.Embedding`

Detokenization

Character-level tokenizer

- Convert network outputs to
  - Characters $c_1 \ldots c_N$
- Predict distribution over characters
  - Sample and convert to UTF-8
- Any issues?
  - Not every sequence of characters is valid UTF-8

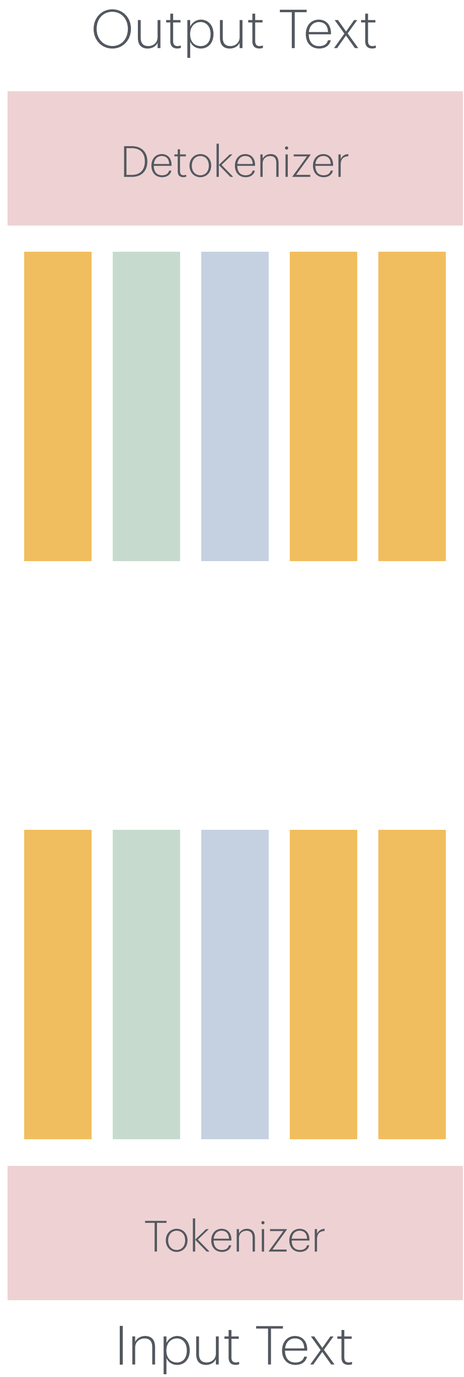


`torch.nn.Softmax`
`torch.distributions.Categorical`

Tokenization

Character-level tokenizer

- Discussion
  - Symmetric
    - Optionally share input embeddings + output classifier weights
  - Few tokens
  - Very long sequences



Tokenization

Byte-pair encoding

- Learned Text Compression Algorithm
  - Training set D
- Vocabulary V (string → int)
- Merge most frequent consecutive tokens
- Trained on large text corpus
  - Vocabulary size 40k - 256k

```
def BPE(D: bytes, N: int):
  # Initialize single charater vocab
  V = {chr(i): i for i in range(256)}

  # Tokenize
  d = [V[c] for c in D]

  while len(V) < N:
    # Find most frequent pair of tokens
    cnt = Counter([(d[i], d[I+1])
                   for i in range(len(d)-1)])
    a,b = cnt.most_common(1)[0][0]

    # Add a new token
    V[(a,b)] = len(V)

    # Update the dataset
    d_update = []
    for x in d:
      if len(d_update) and d_update[-1] == a\
         and x == b:
        d_update[-1] = V[(a,b)]
      else:
        d_update.append(x)
    D = d_update
```

Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.

Tokenization

Byte-pair encoding training

- In practice
  - **Split** words / numbers / special strings **with regex** in entire dataset
  - Attach white space to next word / token
- Run BPE over entire dataset
  - Converts dataset into sequence of tokens

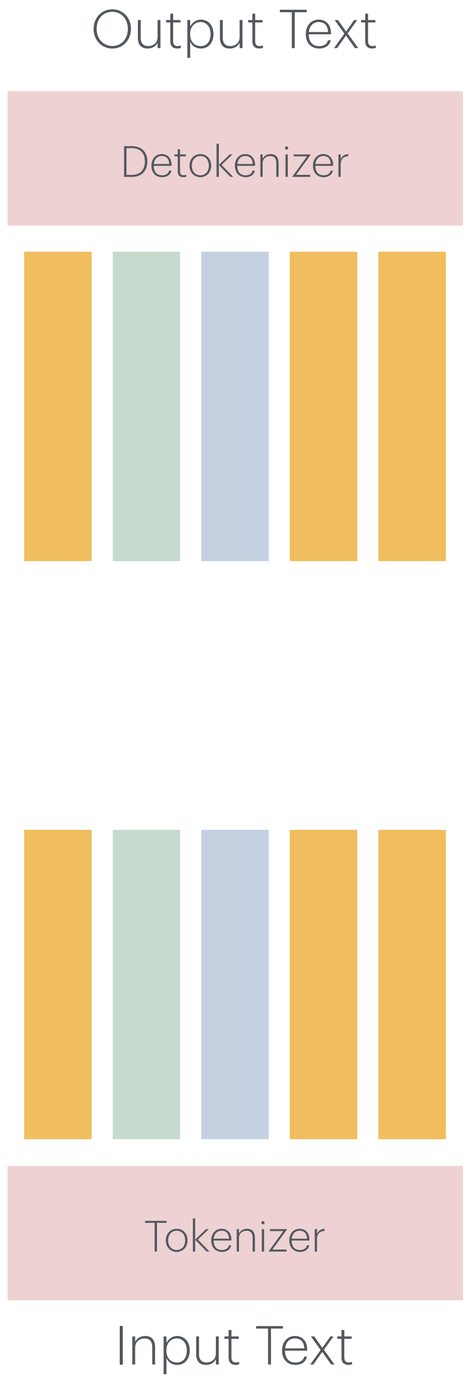
The cat in the hat sat in a chair

The
·Cat
·in
·the
·Hat
·sat
·in
·a
·chair

Tokenization

Byte-pair encoding

- Tokenization
  - Merge elements of vocabulary one by one
- Detokenization
  - Replace token with vocabulary text
- Embed and predict compressed tokens



Tokenization

Byte-pair encoding

- Demo https://tiktokenizer.vercel.app/?model=meta-llama%2FMeta-Llama-3-70B

Token count
33

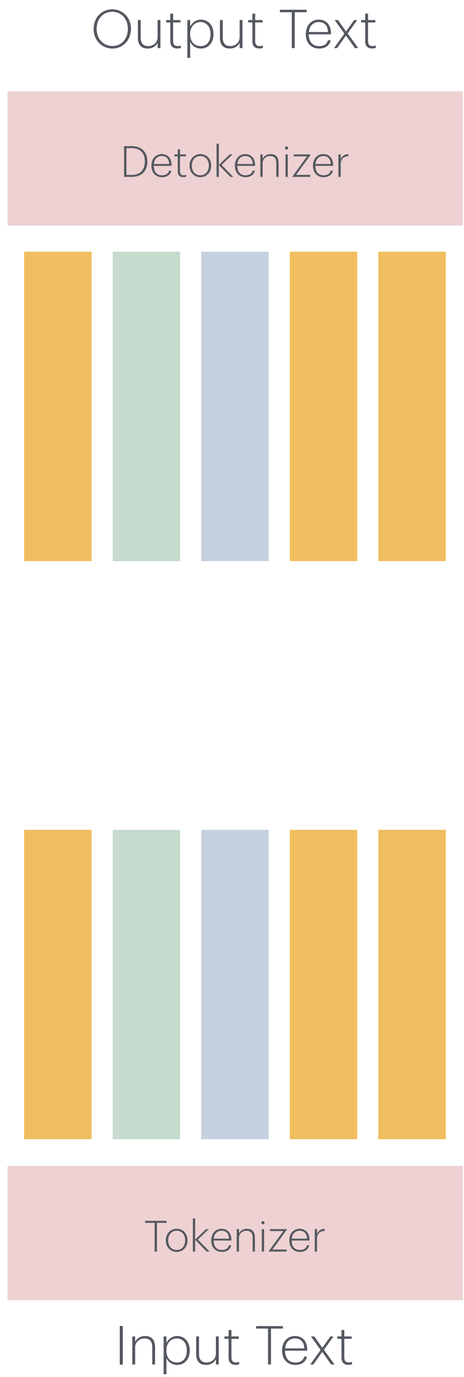
A cat was climbing a tree or Tree.
Tree was climbed by a cat.
A tree-climbing cat...
一隻貓正在爬樹。

32, 8415, 574, 30608, 264, 5021, 477, 9119, 627, 6670, 574, 45519, 555, 264, 8415, 627, 32, 5021, 31717, 318, 7278, 8415, 9522, 15120, 37795, 119, 80631, 241, 9765 5, 76207, 105, 111398, 1811

Tokenization

Byte-pair encoding

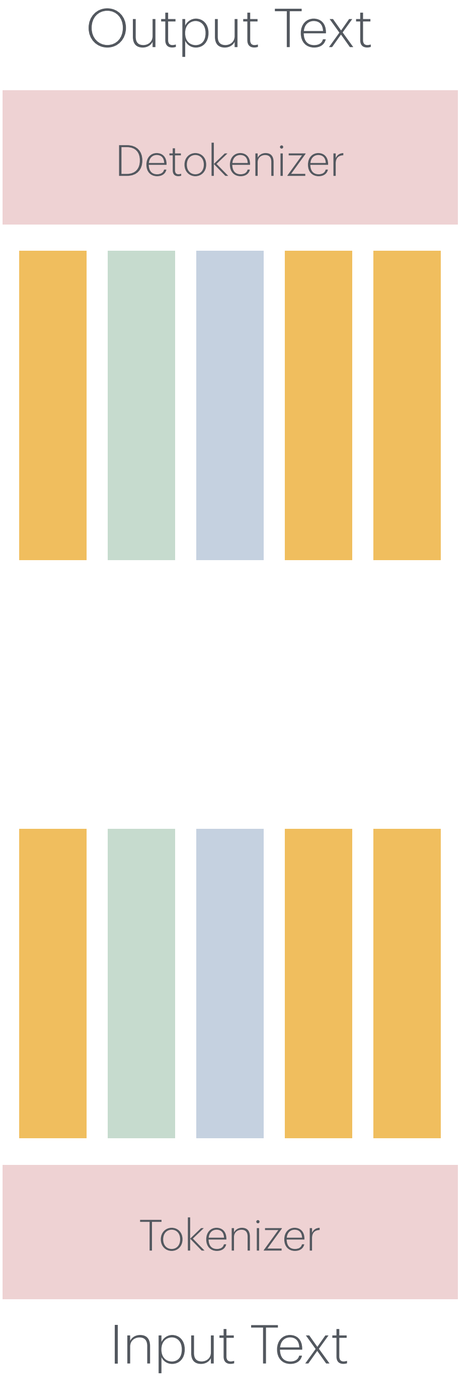
- Special tokens
  - Beginning of Sequence [BOS]
  - End of Sequence [EOS]
  - Classification [CLS]
  - Separator [SEP]
  - ...



Tokenization

Byte-pair encoding

- Alternative: WordPiece
- Discussion
  - Many tokens (large vocabulary)
  - Shorter sequences
- One of the main sources of issues for LLMs



Tokenization - Issues

- Demo https://tiktokenizer.vercel.app/?model=meta-llama%2FMeta-Llama-3-70B

Tokenization - Issues

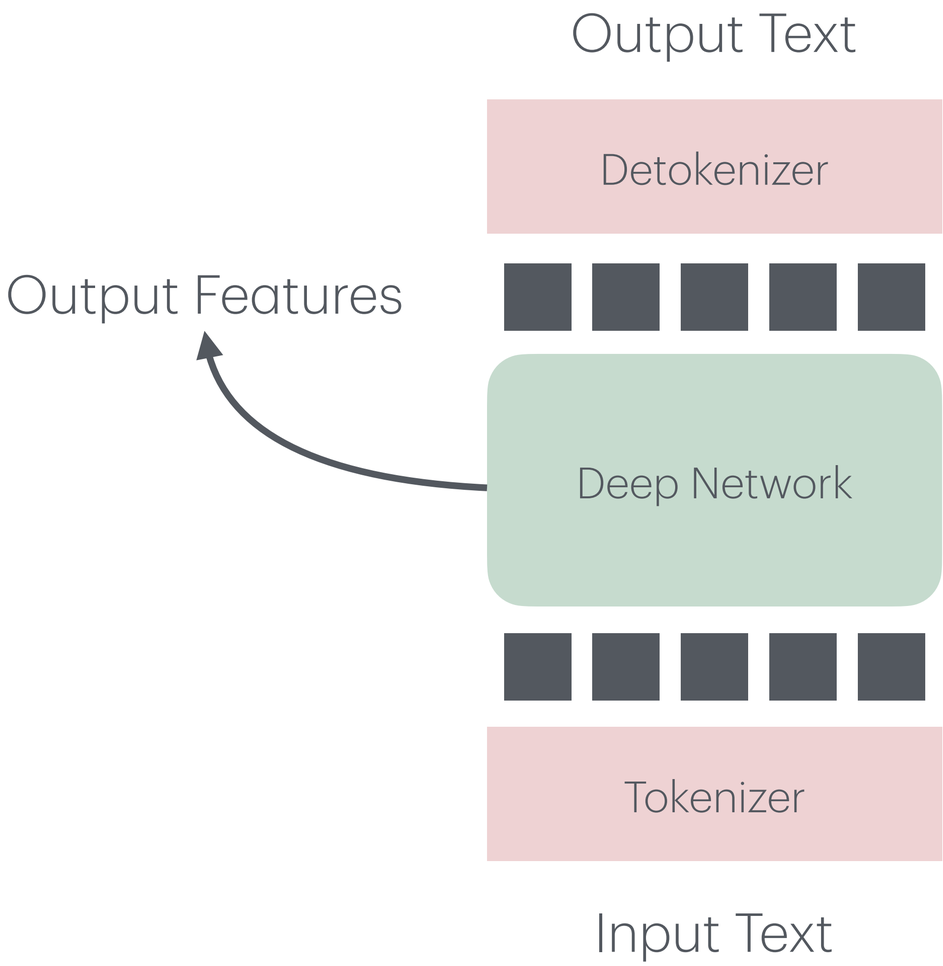
- Not a one-to-one mapping
  - Many token-streams detokenize to same text
- Counting letters / word manipulations / string processing are hard
- LLMs are bad at math
- Potential exploits around tokens with few training examples

Tokenization - Issues

- Demo
  ollama run llama3.1

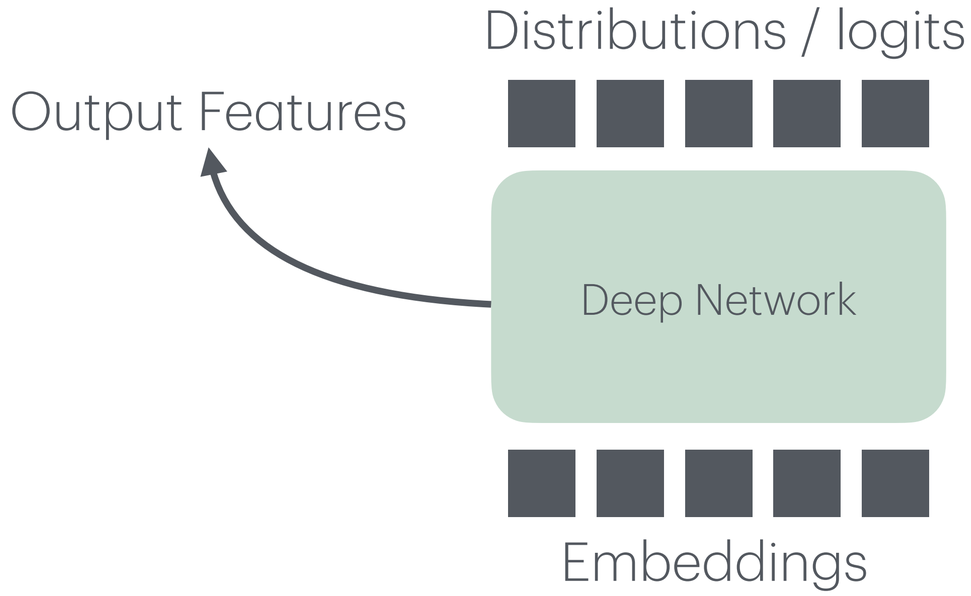
LLM - Basics

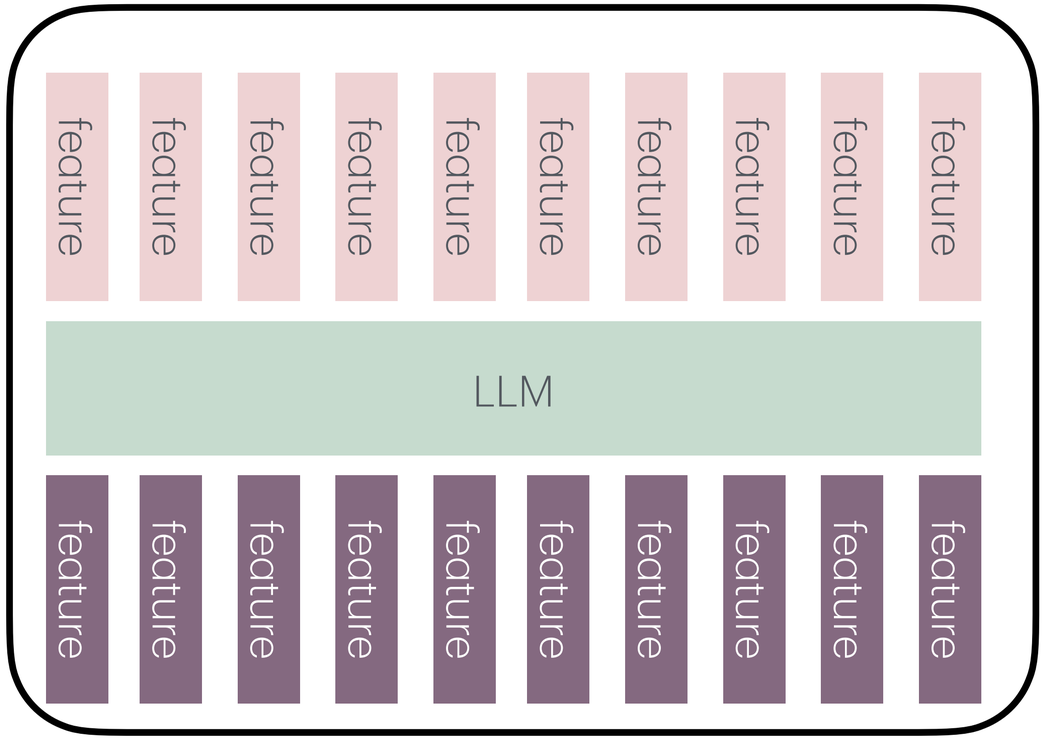
- Tokenizer
- Deep Network
  - Encoder-Decoder (original transformer)
  - Encoder-only
  - Decoder-only
  - Sequence Models



LLM - Architectures

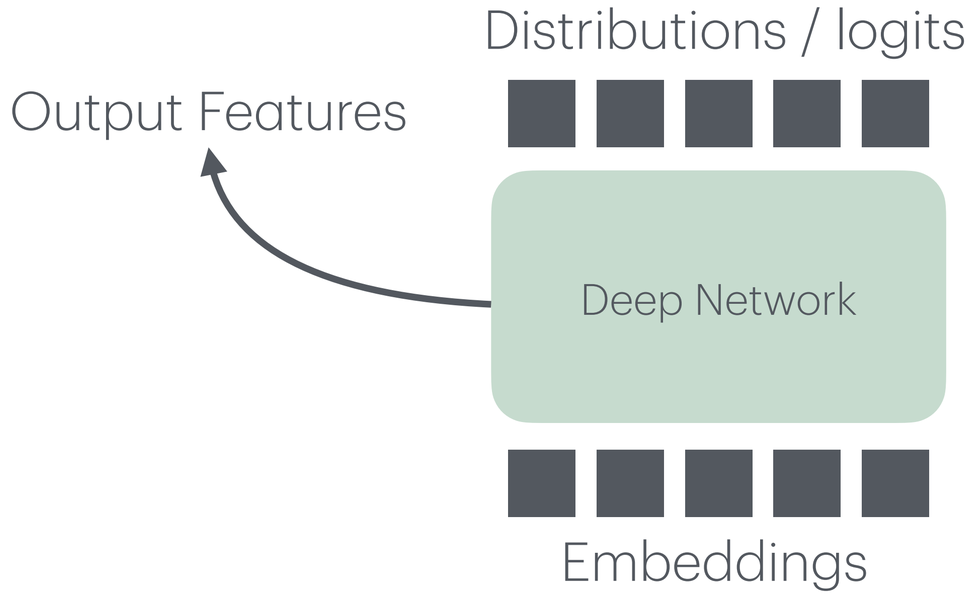
- **Encoder-Decoder (original transformer)**
- Encoder-only
- Decoder-only
- Sequence Models

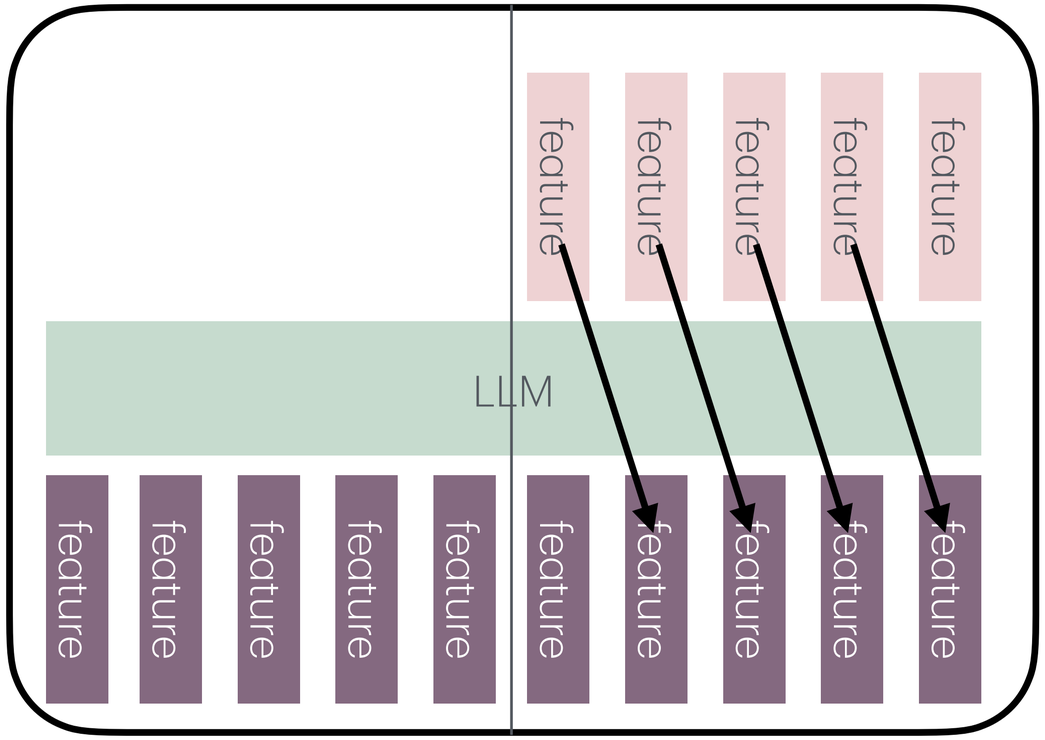




LLM - Architectures

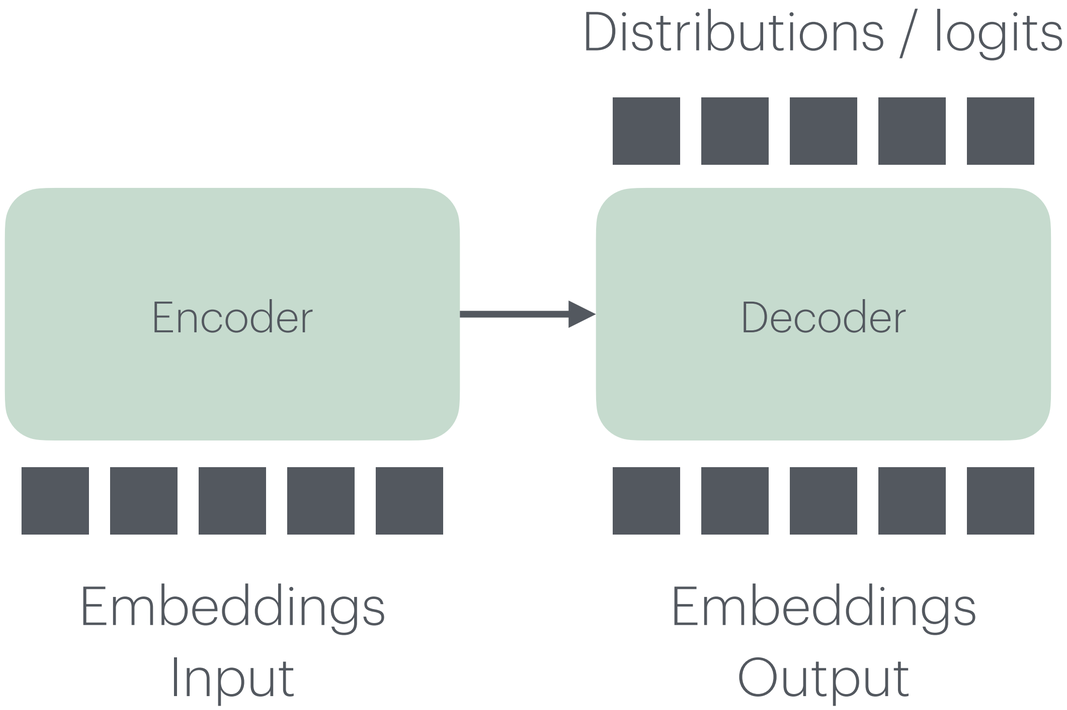
- **Encoder-Decoder (original transformer)**
- Encoder-only
- Decoder-only
- Sequence Models





The transformer

- Input:
  - Text: Question / prompt
- Output:
  - Autoregressive probability over text
  - Token-by-token

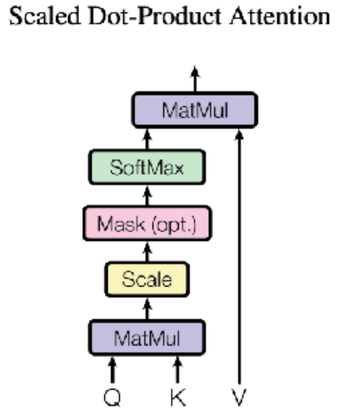


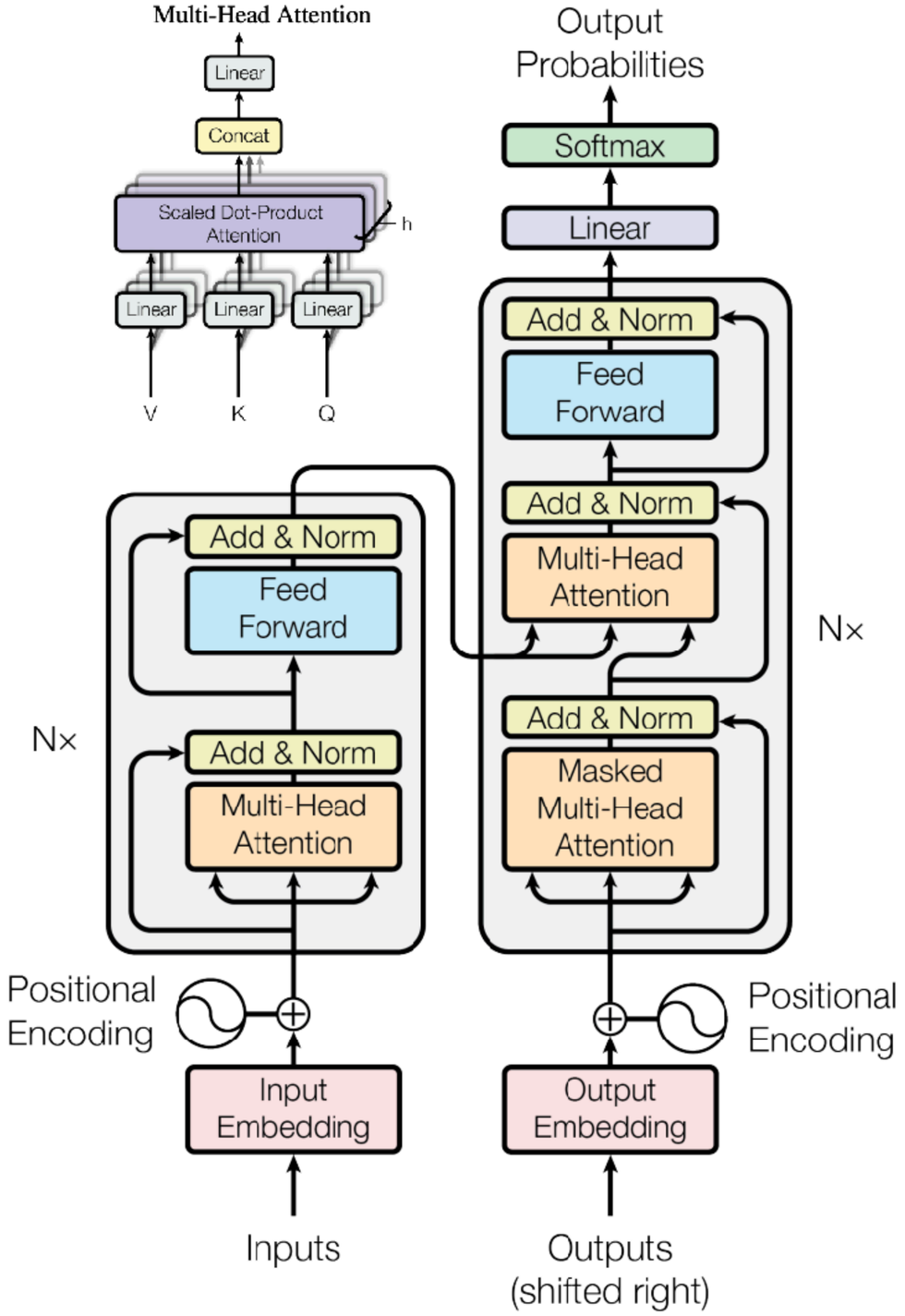
Attention Is All You Need. Vaswani et al. 2017.

The transformer

Architecture

- Encoder
  - Self-attention
  - MLP
- Decoder
  - Causal self-attention
  - MLP



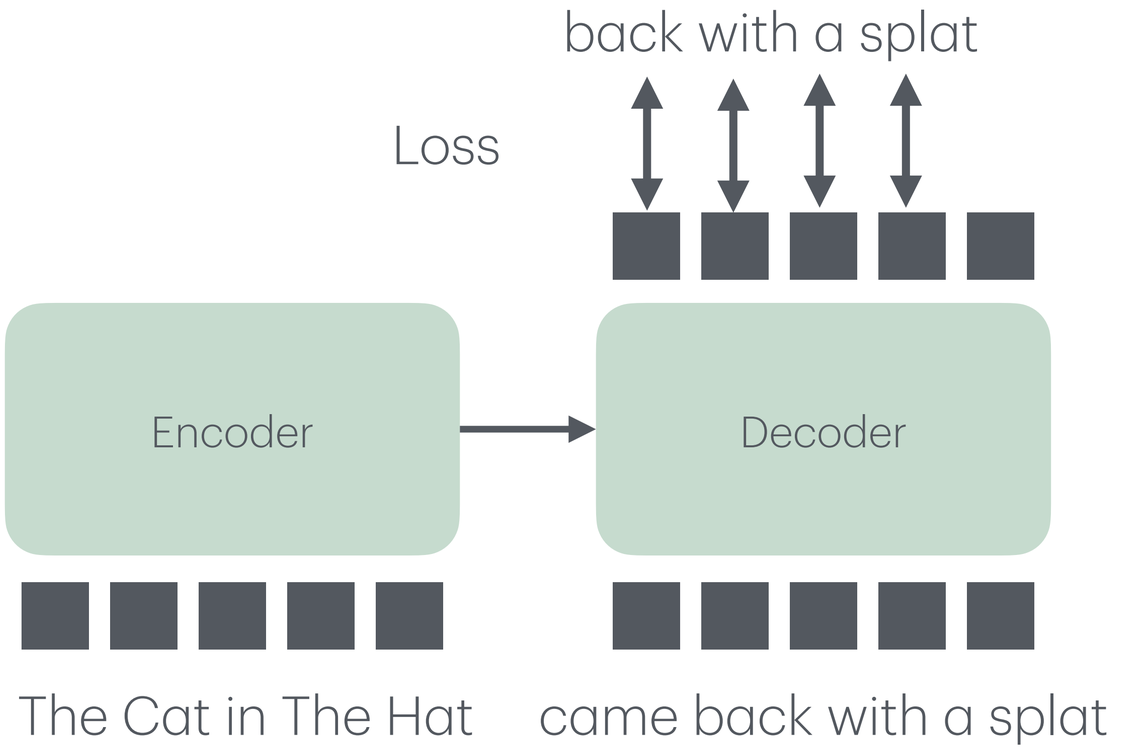


Attention Is All You Need. Vaswani et al. 2017.

The transformer

Pre-training

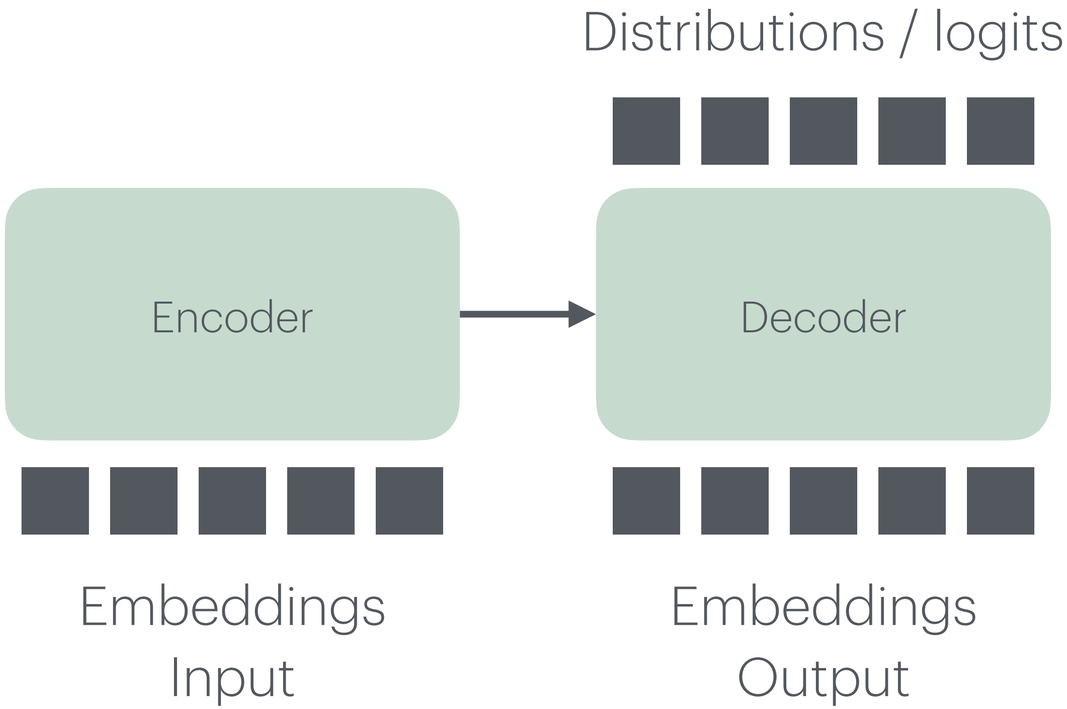
- Encoder
  - Training set text input
  - No loss
- Decoder
  - Training set text input
  - Loss: CrossEntropy shifted training text



Attention Is All You Need. Vaswani et al. 2017.

Encoder-Decoder models

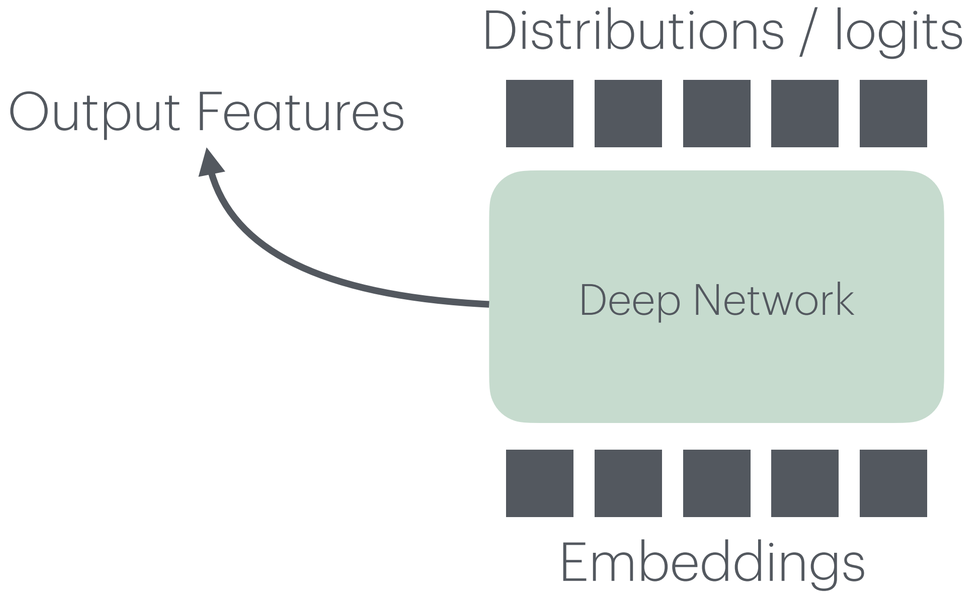
- Original transformer
- Not used much anymore
  - Now decoder-only models more popular

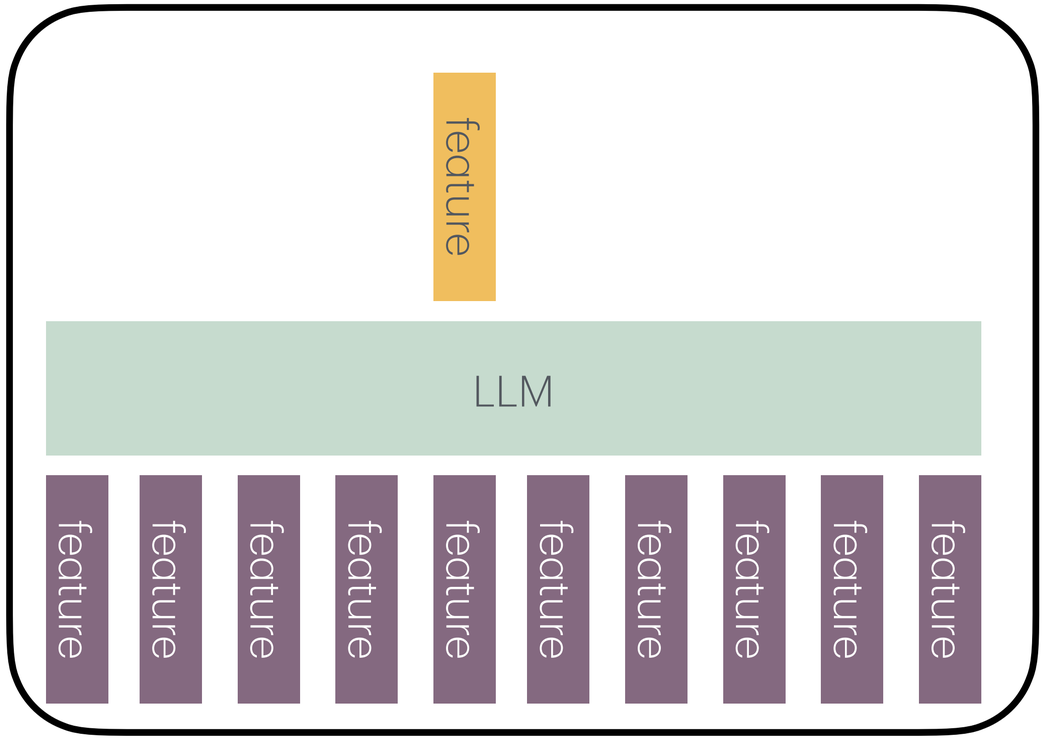


Attention Is All You Need. Vaswani et al. 2017.

LLM - Architectures

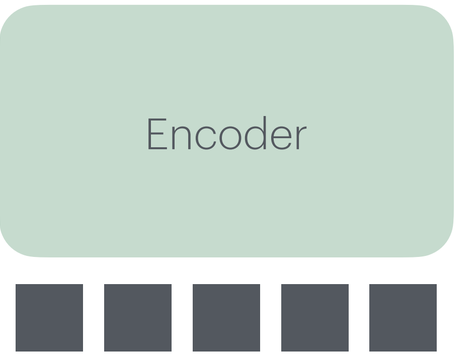
- Encoder-Decoder (original transformer)
- **Encoder-only**
- Decoder-only
- Sequence Models





Encoder-only

- Input:
  - Text: Question / prompt
- Output:
  - A (sequence of) embeddings

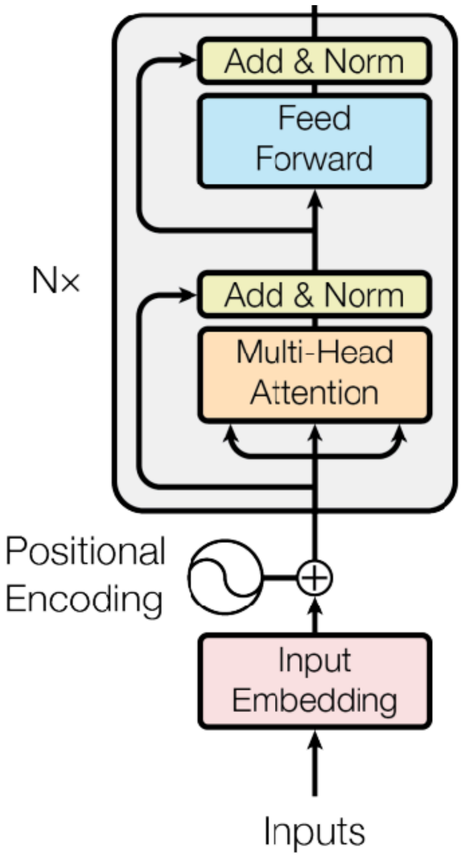


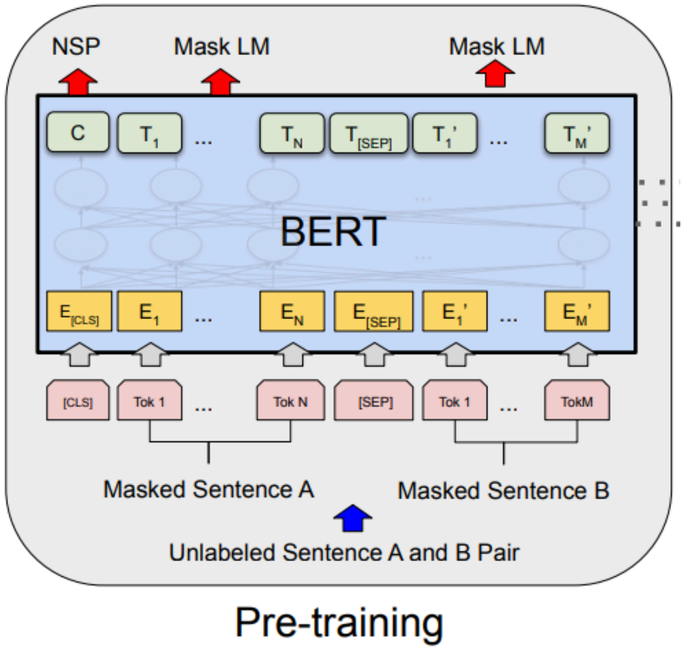
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. 2018.

Encoder-only

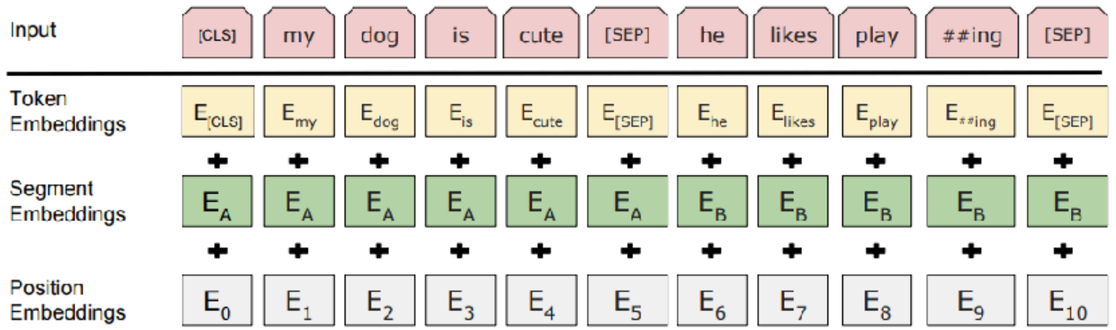
Architecture

- Transformer encoder (i.e. BERT)
  - Bidirectional attention
- Multiple sizes (BERT)
  - L : #layers 12, 24
  - H: Hidden size 768, 1024
  - A: # Attention heads 12, 16





Pre-training

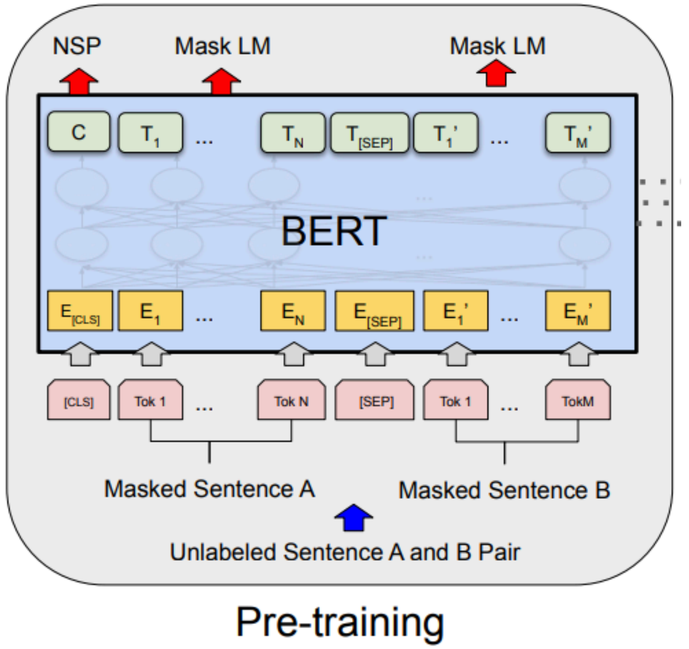


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. 2018.

Encoder-only

Pre-training

- Masked language modeling
  - Mask out tokens in input
  - Reconstruct masked out tokens
- Next sentence prediction
  - Binary tasks
  - Uses [CLS] token

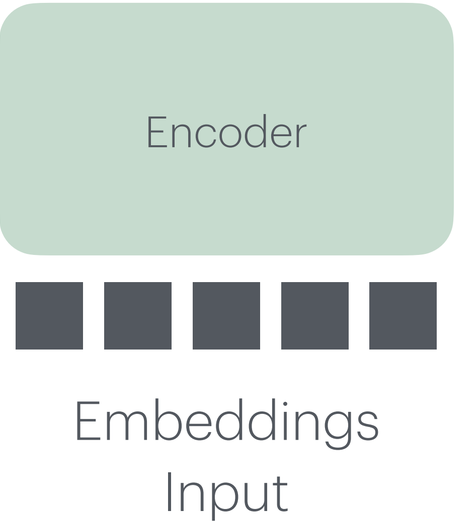


Pre-training

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. 2018.

Encoder-only model

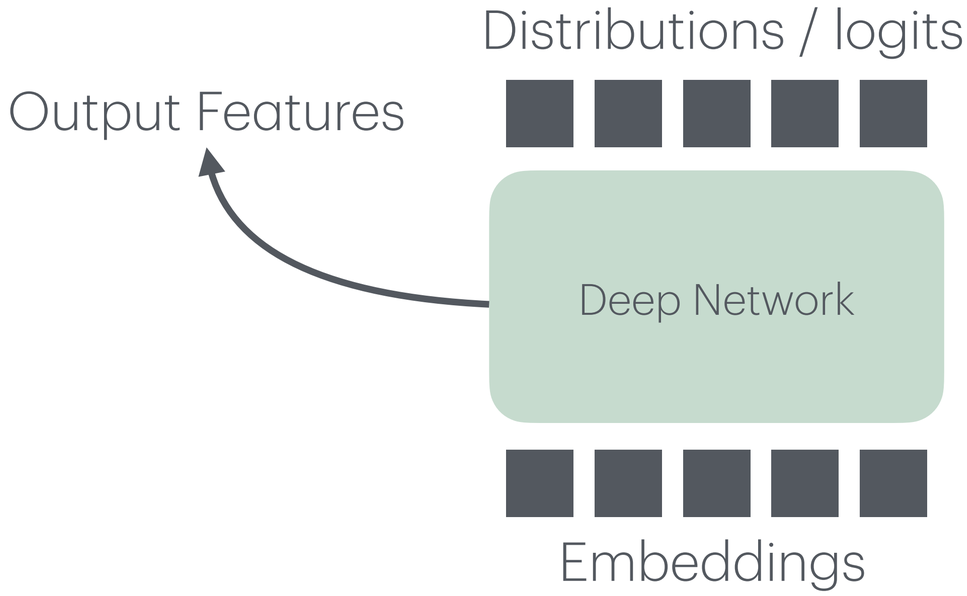
- Very popular 5 years ago
- Good at embedding / classifying text
- Used less today
  - Not good at generation
  - Scale worse with data

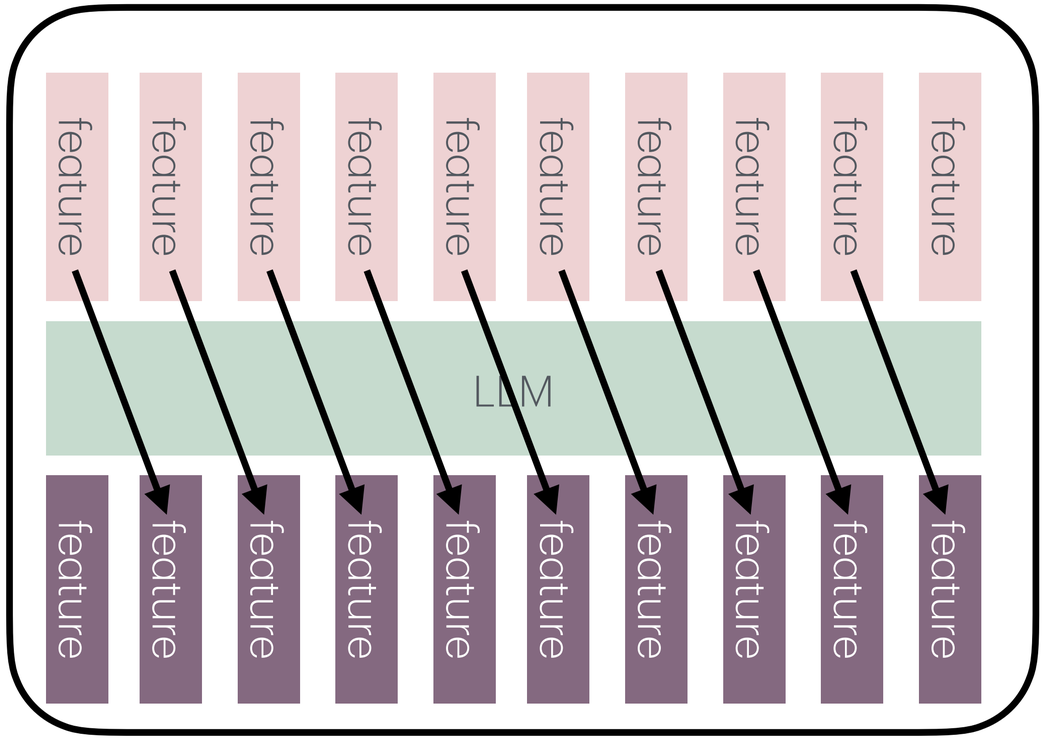


Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. Allen-Zhu and Li. 2023.

LLM - Architectures

- Encoder-Decoder (original transformer)
- Encoder-only
- **Decoder-only**
- Sequence Models

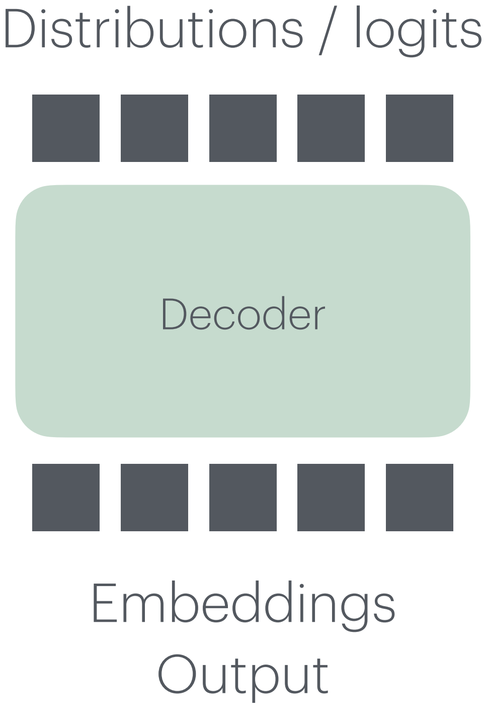




Decoder-only models

GPT and friends

- Input:
  - Text
- Output:
  - Autoregressive probability over text
  - Token-by-token

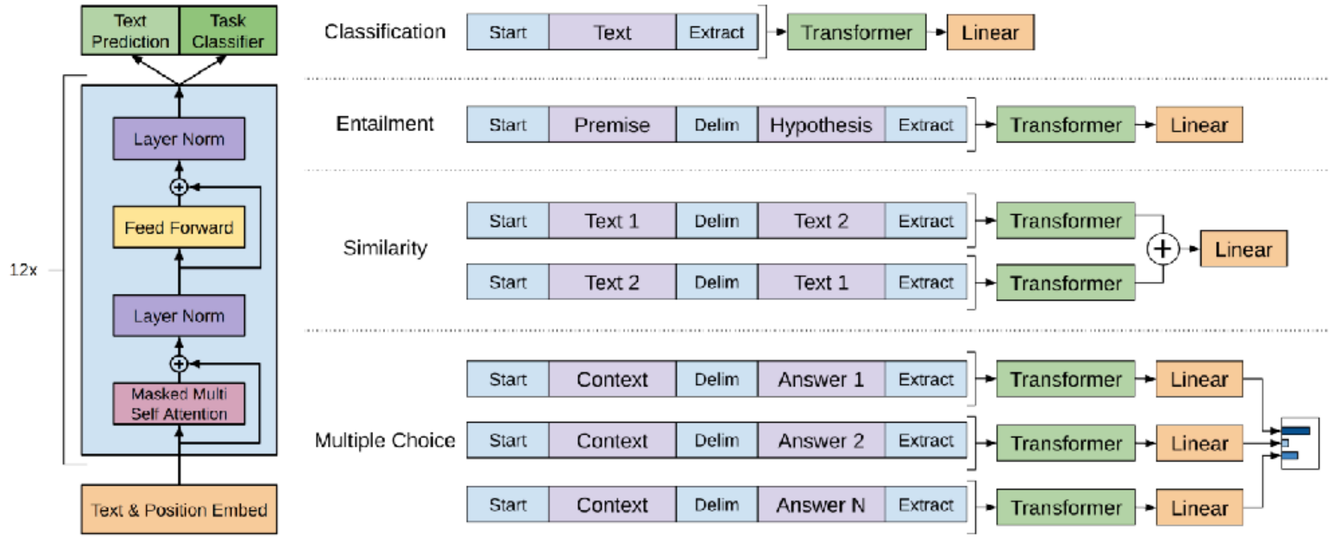


Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.

Decoder-only models

Architecture

- Decoder-only (GPT-1)
  - Causal self-attention
  - MLP
  - Special [extract] token for classification tasks

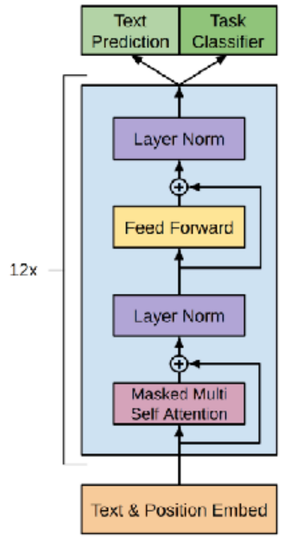


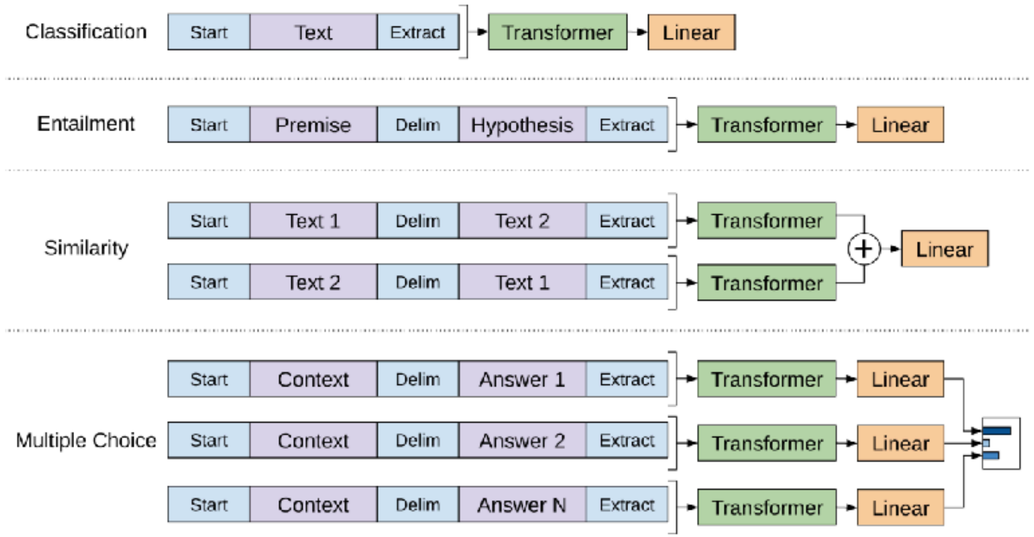
Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.

Decoder-only models

Architecture

- GPT-1: 12 layers, 768 hidden size, MLP 3072 dim, 12 attention heads; **0.1B** params
- GPT-2: 48 layers, 1600 hidden size, larger vocab (50k); **1.5B** params
- GPT-3: 96 layers, 12288 hidden size, 96 attention heads; **175.0B** params
- GPT-4: ??





Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.

Language Models are Unsupervised Multitask Learners. Radford et al. 2019.

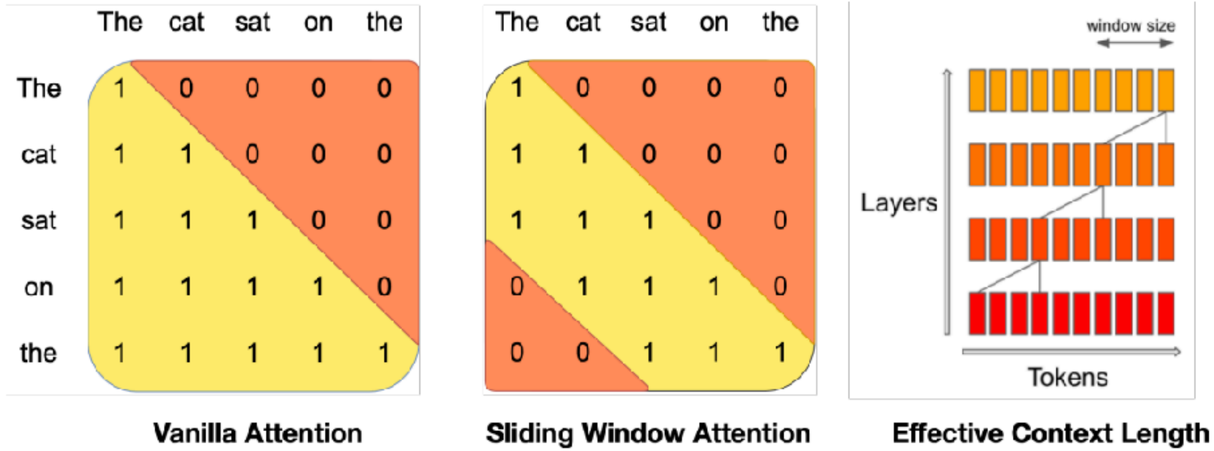
Language Models are Few-Shot Learners. Brown et al. 2020.

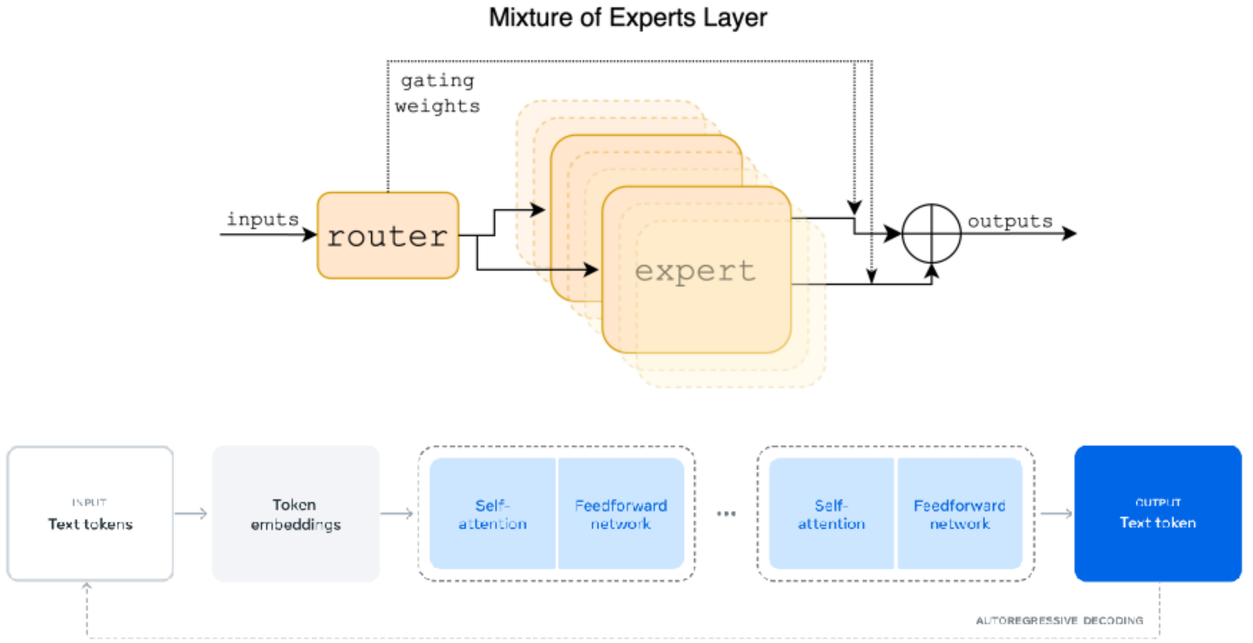
Decoder-only models

Architecture

- Mistral 7b: 32 layers, 4096 hidden size, MLP 14336 dim, 32 attention heads; window attention, Group Query Attention
- Mixtral 8x7b: Mistral 7b + Mixture of experts per MLP layer
- Llama 3.x:

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		





Mistral 7B. Jiang et al. 2023.

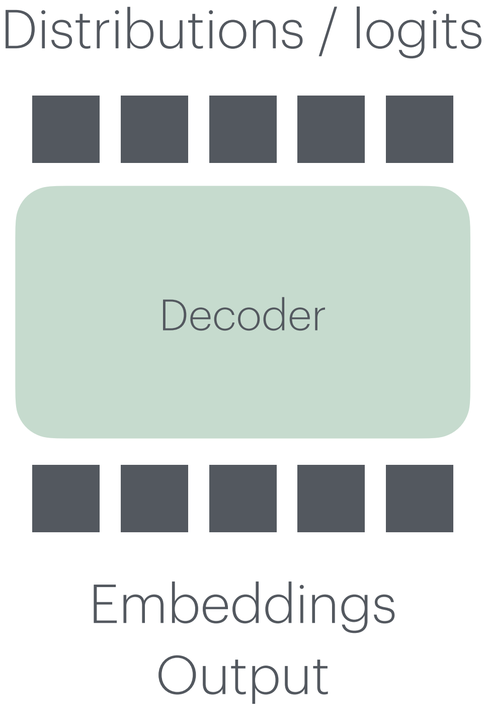
Mixtral of Experts. Jiang et al. 2023.

Llama 3 Herd of Models. Dubey et al. 2024.

Decoder-only models

Architecture

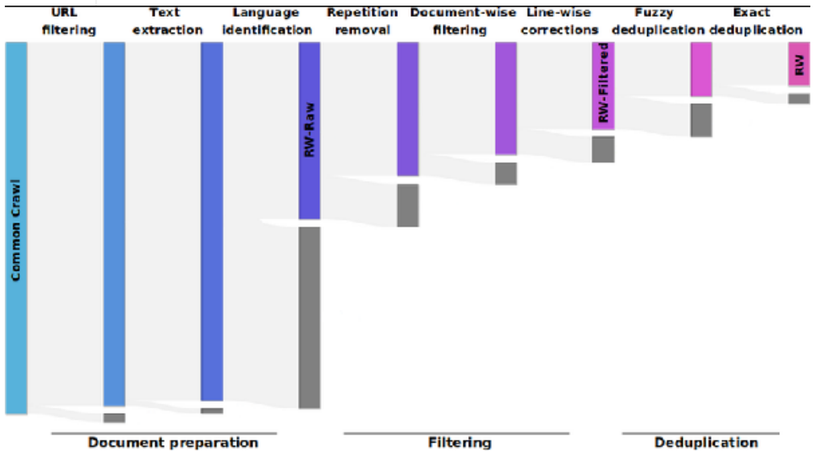
- A big transformer
  - Input: Sequence
  - Output: Probability over next token

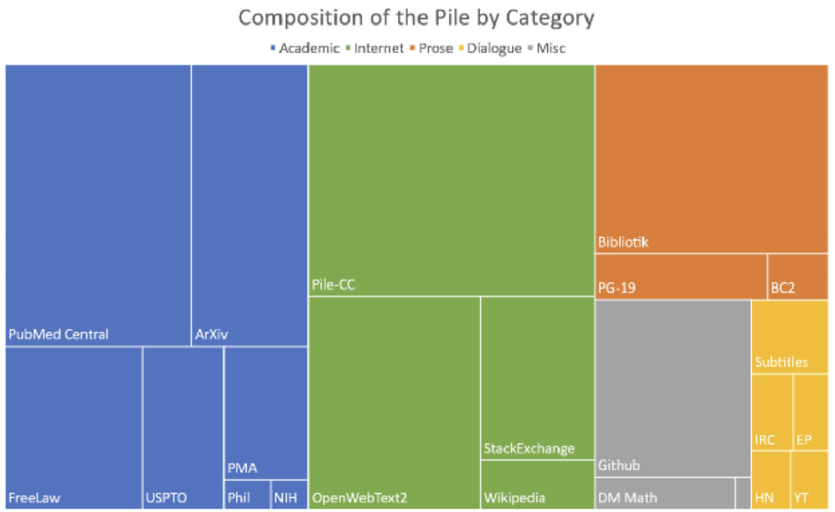


Decoder-only models

Pre-training

- Next token prediction
  - On all written and digitized text available
    - Filtered and cleaned up
    - Industry: Lots of mined (and now licensed) internal data





Common Crawl. Common Crawl Foundation. 2008-present.

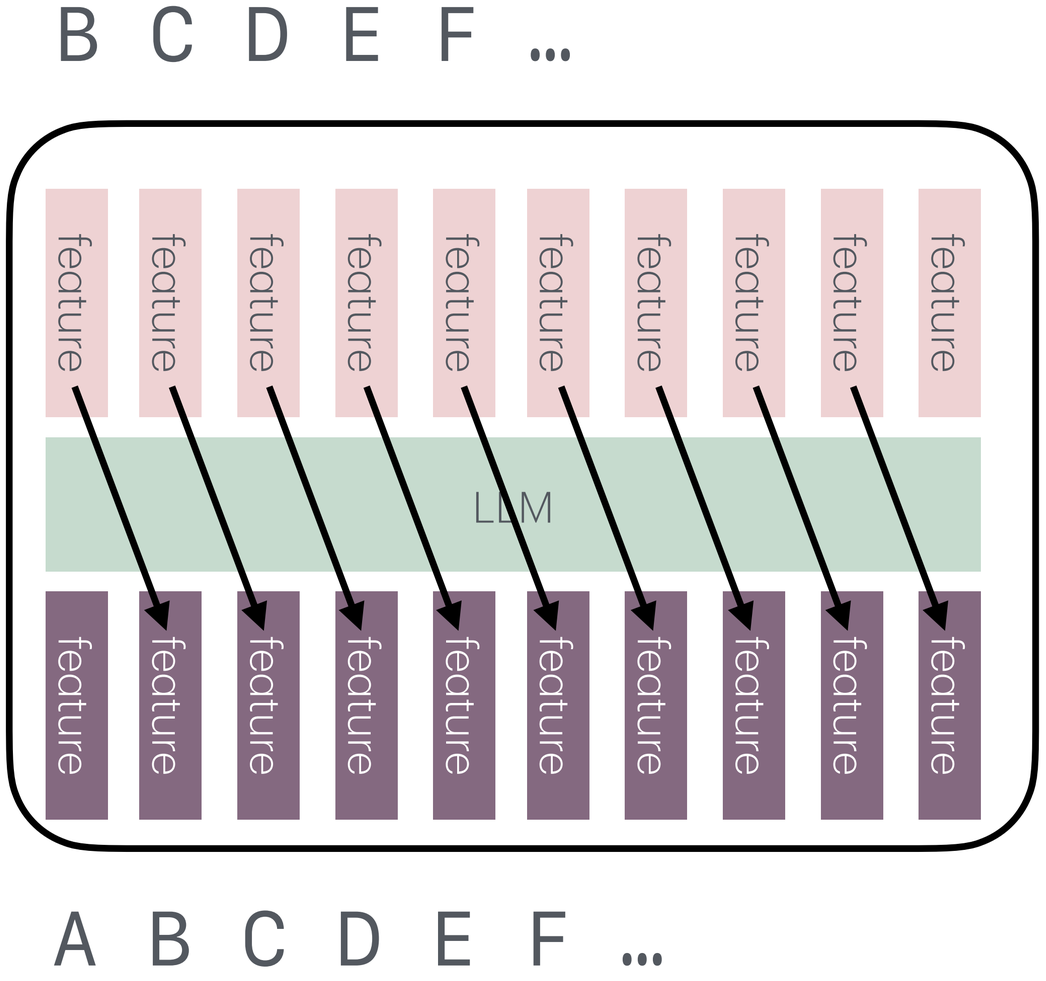
The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data. Penedo et al. 2023.

The Pile: An 800GB Dataset of Diverse Text for Language Modeling. Gao et al. 2020.

Decoder-only models

Pre-training

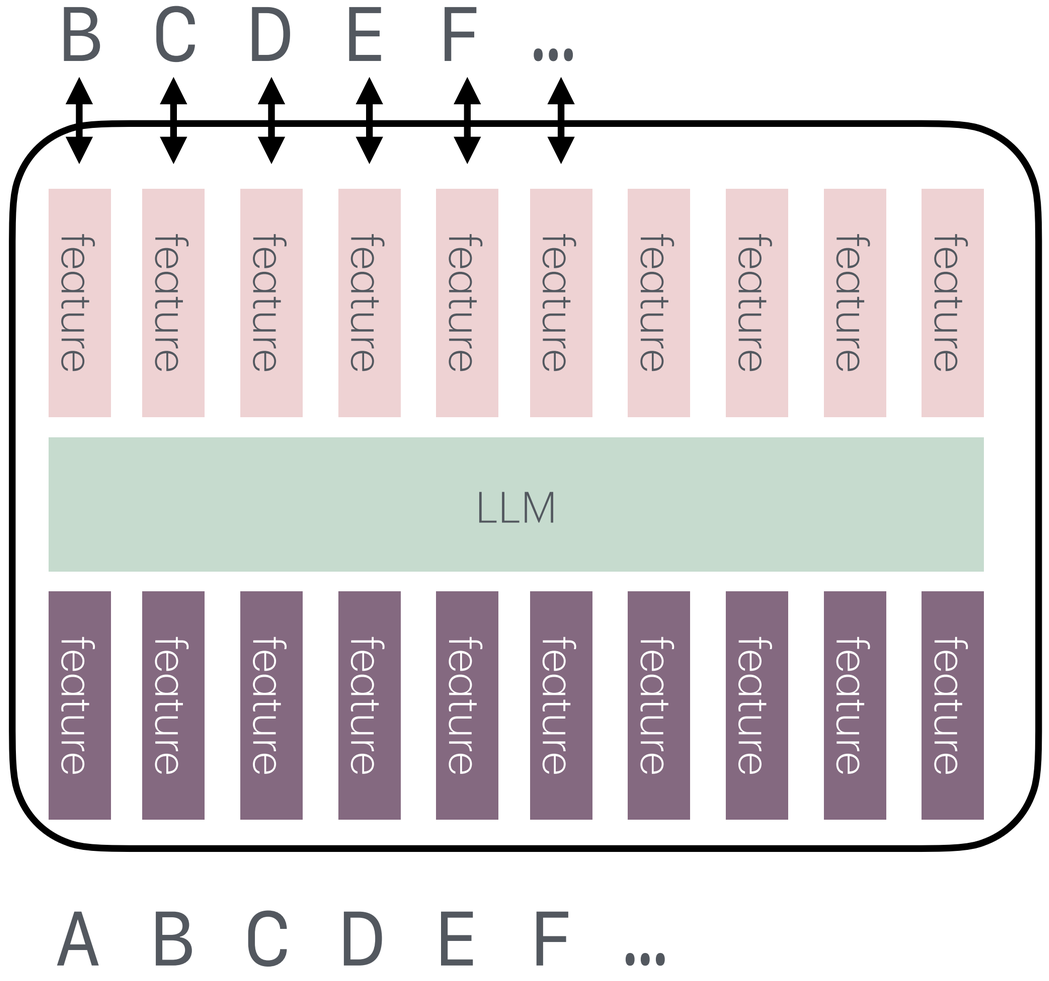
- Teacher forcing
  - Feed in ground truth tokens
  - Supervised shifted ground truth



Decoder-only models

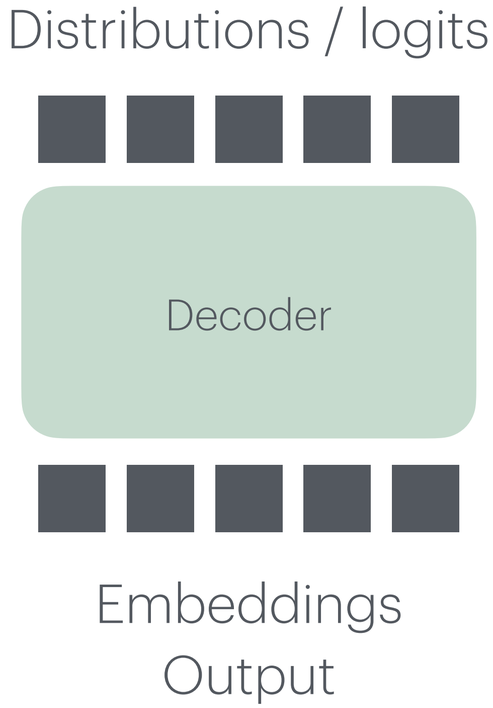
Pre-training

- Teacher forcing
  - Feed in ground truth tokens
  - Supervised shifted ground truth



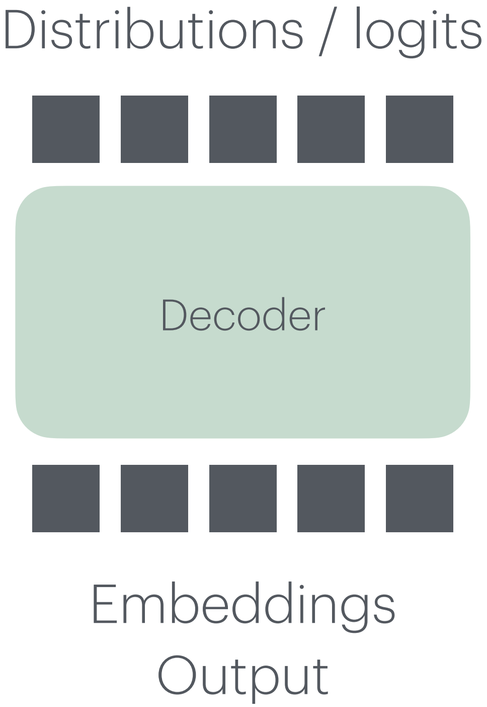
Decoder-only models

- Demo
  ollama run llama3.1:8b-text-q4_0



Decoder-only model

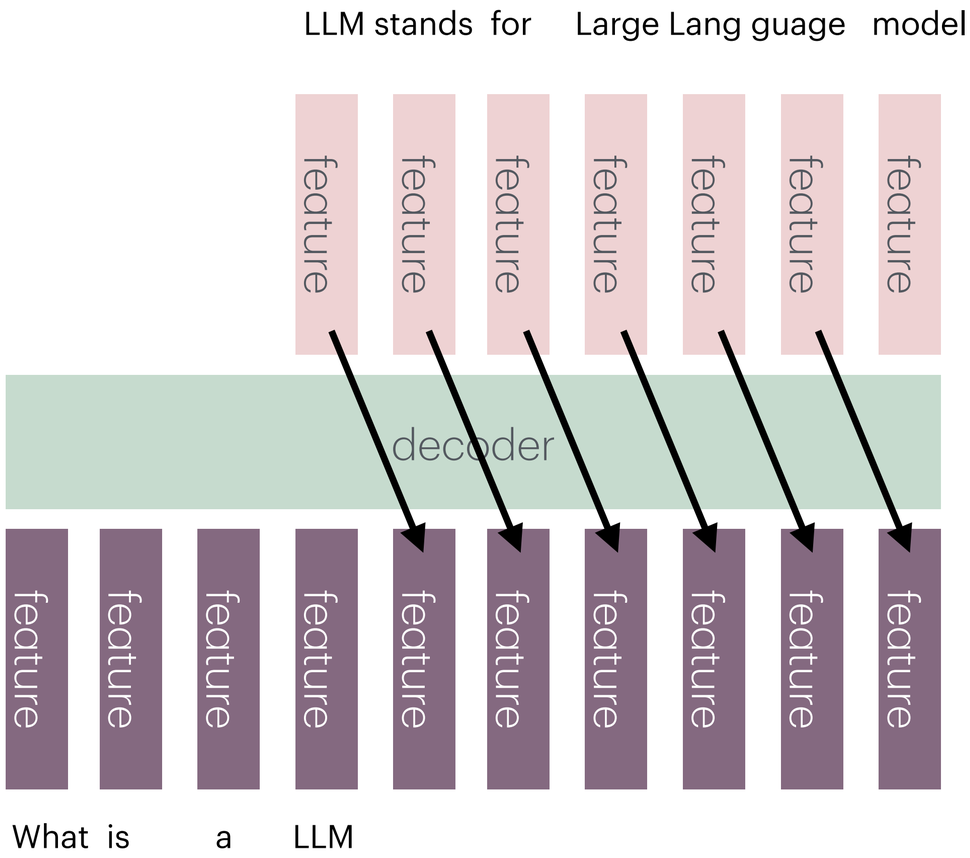
- Decoder-only models = LLM nowadays
- Best generation models
- Good embeddings
- Able to memorize training data well
  - Up to 2 bits per parameter [1]



[1] Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. Allen-Zhu and Li. 2023.

Decoder-only model

- Decoder-only models can mimic encoder-decoder models

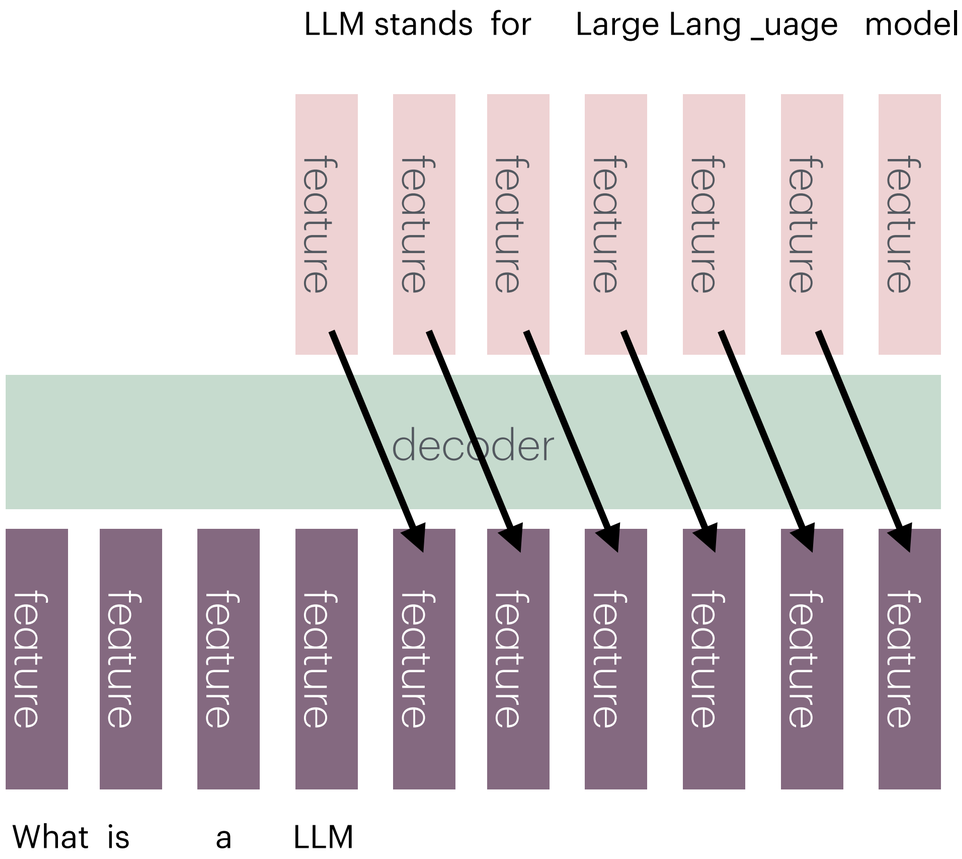


[1] Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. Allen-Zhu and Li. 2023.

Decoder-only model

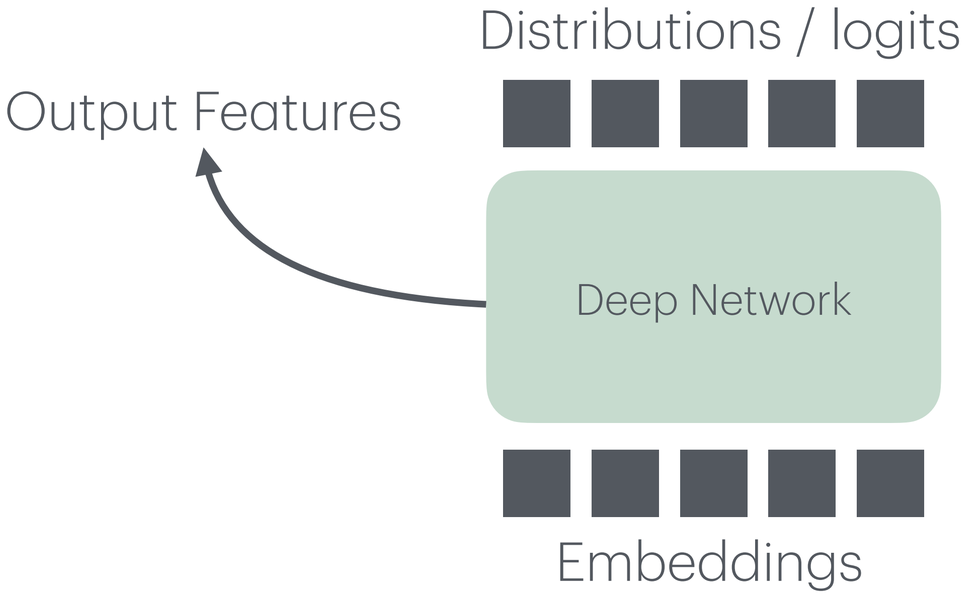
Where do models store data?

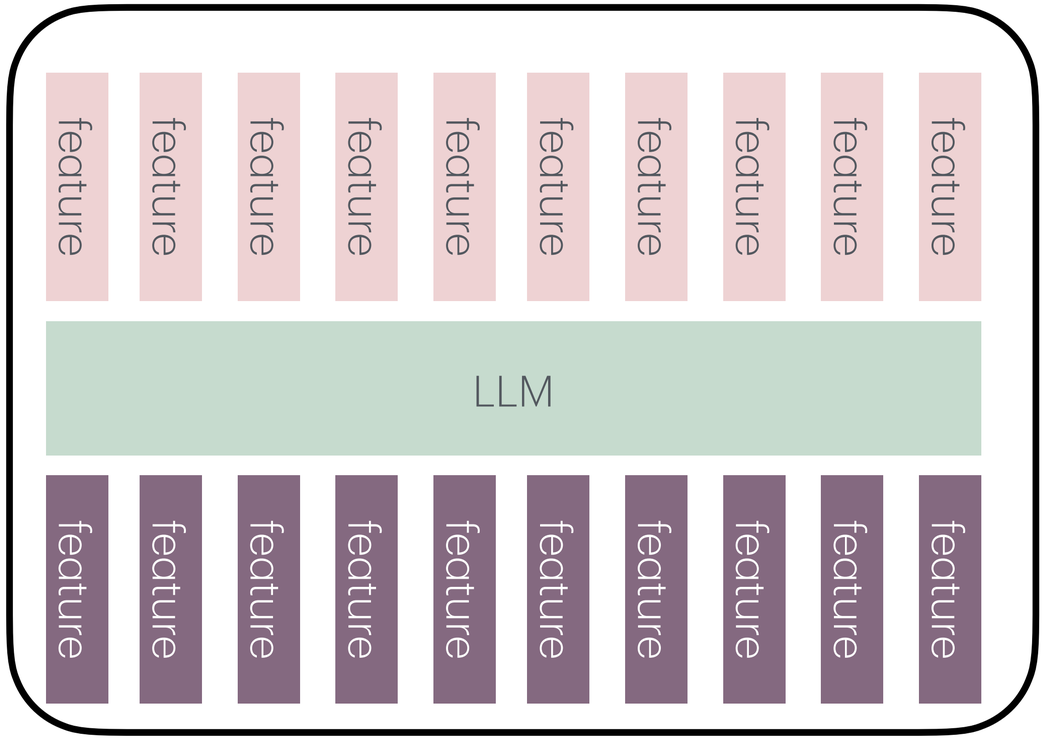
- Option A
  - In their context / prompt
- Option B
  - In their weights



LLM - Architectures

- Encoder-Decoder (original transformer)
- Encoder-only
- Decoder-only
- **Sequence Models**

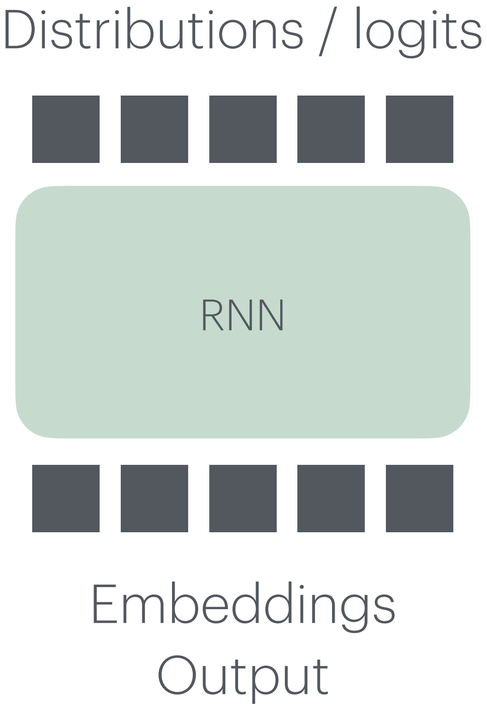




Sequence Models

Mamba and friends

- Input:
  - Text
- Output:
  - Autoregressive probability over text
  - Token-by-token

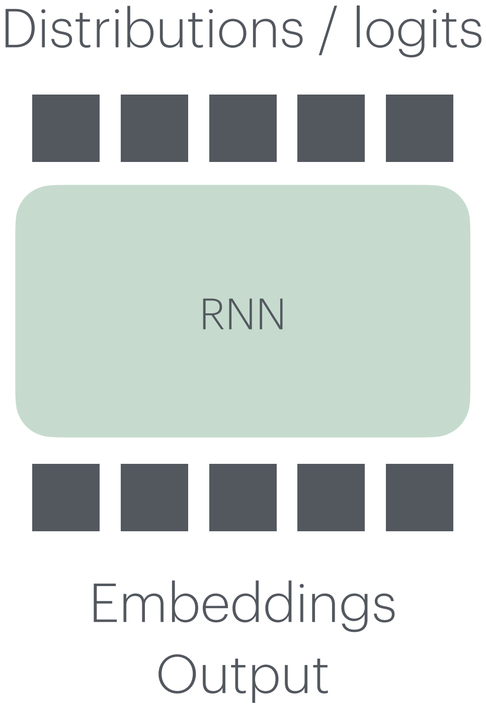


Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.

Sequence Models

Architecture

- State-space model layer:
  - Input: u_t
  - Hidden state: $x_t = Ax_{t-1} + Bu_t$
  - Output: $y_t = Cx_t$



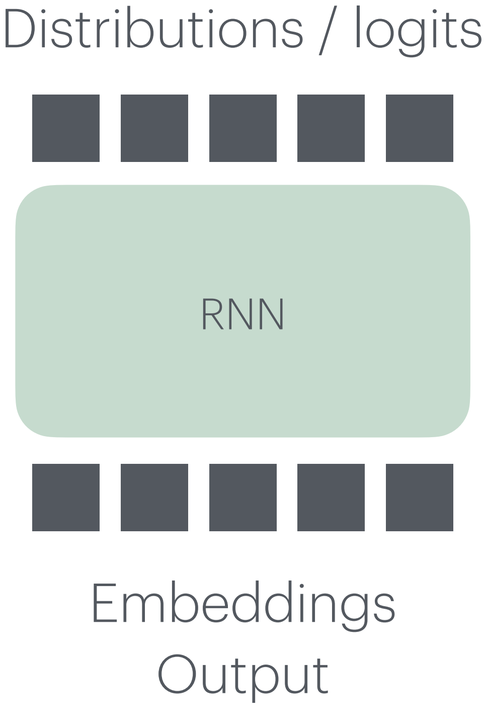
Mamba: Linear-Time Sequence Modeling with Selective State Spaces. Gu et al. 2023.

Efficiently Modeling Long Sequences with Structured State Spaces. Gu et al. 2021.

Sequence Models

Training

- State-space model layer
  - Unroll
  - Similar to convolution / transformer with more structured attention matrix
  - Same training as decoder-only model

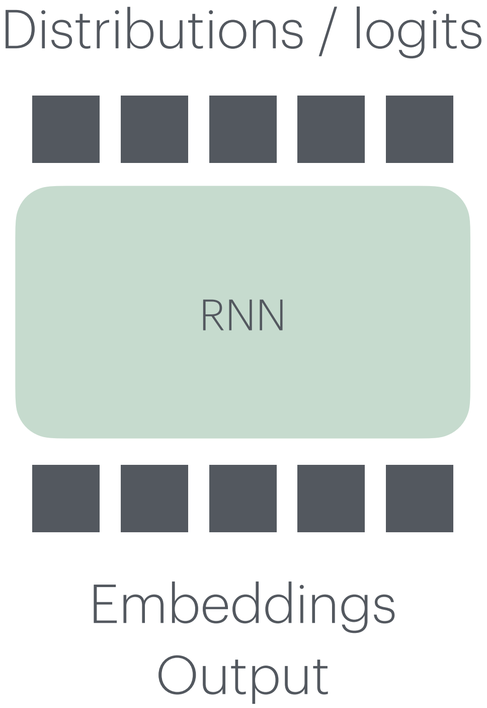


Mamba: Linear-Time Sequence Modeling with Selective State Spaces. Gu et al. 2023.
Efficiently Modeling Long Sequences with Structured State Spaces. Gu et al. 2021.

Sequence Models

Discussion

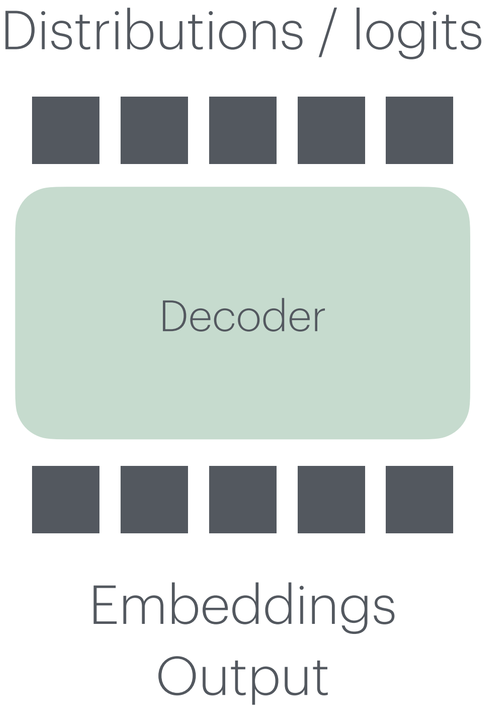
- Potentially faster inference than transformer
- Similar training
- Fixed bottleneck
- Not yet as same performance



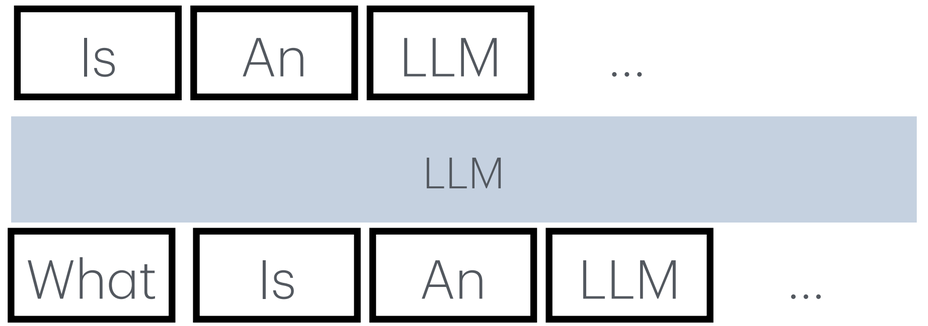
Mamba: Linear-Time Sequence Modeling with Selective State Spaces. Gu et al. 2023.

Efficiently Modeling Long Sequences with Structured State Spaces. Gu et al. 2021.

Architecture



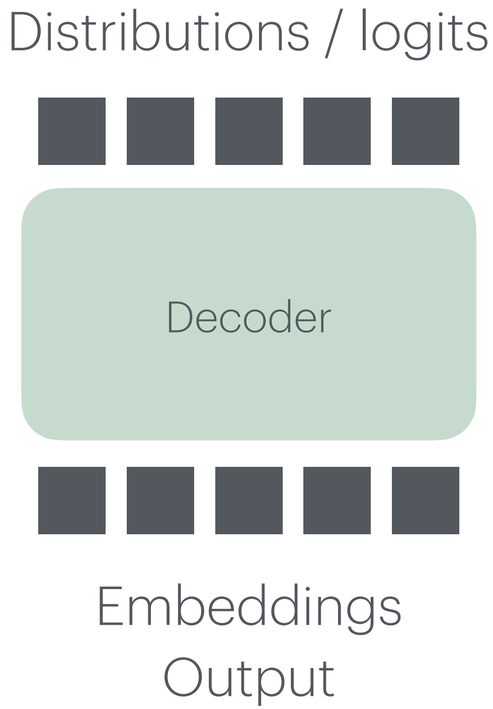
- LLMs are decoder only nowadays
  - Modeling auto-regressive distribution over tokens
  - $P(\mathbf{t}) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_1, t_2)P(t_4 \mid t_1 \ldots t_3) \ldots$
- Able to memorize training data well
  - Up to 2 bits per parameter [1]
- Very good generative models



[1] Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. Allen-Zhu and Li. 2023.

What's next?

- Generation: How do we sample from decoder-only models?
- Instruction tuning: How do we convert language models into chat models?
- Preference tuning: How to tell an what "good" and safe answers are?
- Tasks and datasets: How to evaluate performance of LLMs



References

- [1] Improving Language Understanding by Generative Pre-Training. Radford et al. 2018.
- [2] Attention Is All You Need. Vaswani et al. 2017.
- [3] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. 2018.
- [4] Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. Allen-Zhu and Li. 2023.
- [5] Language Models are Unsupervised Multitask Learners. Radford et al. 2019.
- [6] Language Models are Few-Shot Learners. Brown et al. 2020.
- [7] Mistral 7B. Jiang et al. 2023.
- [8] Mixtral of Experts. Jiang et al. 2023.
- [9] Llama 3 Herd of Models. Dubey et al. 2024.
- [10] Common Crawl. Common Crawl Foundation. 2008-present.
- [11] The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data. Penedo et al. 2023.
- [12] The Pile: An 800GB Dataset of Diverse Text for Language Modeling. Gao et al. 2020.
- [13] Mamba: Linear-Time Sequence Modeling with Selective State Spaces. Gu et al. 2023.
- [14] Efficiently Modeling Long Sequences with Structured State Spaces. Gu et al. 2021.