Chapter NLP:IV

IV. Text Models

- Language Modeling
- Large Language Models
- Text Generation

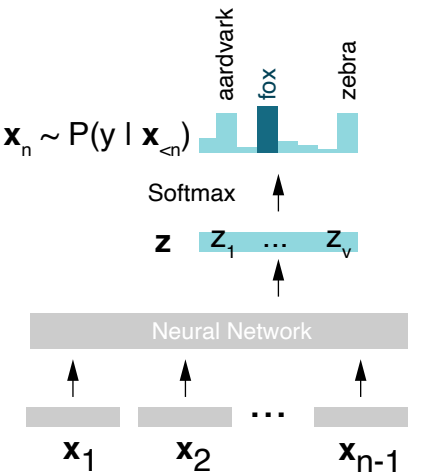
Large Language Models

Neural Language Models [ML:IV-125] [ML:IX-29]

Any neural network architecture is able to learn a language modeling task with the (standard) cross-entropy loss.

- Input are the previous tokens $\mathbf{x}_{<n}$.
- Output is a layer $\mathbf{z}$ of the vocabulary length $|V|$.
- The Softmax function σ determines $P(x_n \mid \mathbf{x}_{<n})$.

$$\sigma_\Delta(\mathbf{z}) = \frac{e^{z_1}}{\sum_{i=1}^{v} e^{z_i}}$$



- Training: The prediction error is the cross-entropy between P and the true token x_n.
  Given enough cases with identical $\mathbf{x}_{<n}$ but different x_n, the model learns the distribution of x_n.

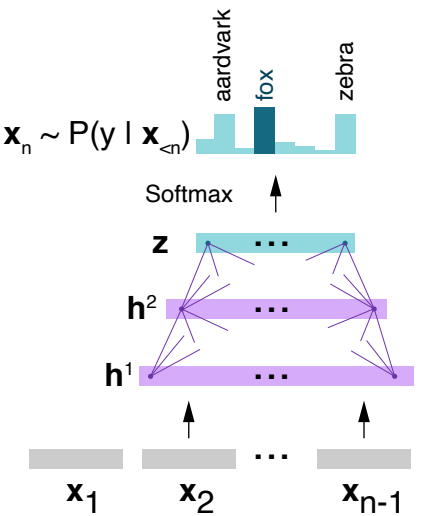
Large Language Models

Neural Language Models [ML:IV-125] [ML:IX-29]

Any neural network architecture is able to learn a language modeling task with the (standard) cross-entropy loss.

Feed-forward Neural Networks

- Due to the fixed input size, chose the latest k tokens for prediction.
- Train and test the model by sliding over the sequence.
- Decode $P(x_1, \ldots, x_n)$ via chain rule (as with n-gram models.)



Problems:

- Fixed sequence length.
- Poor long-distance performance.

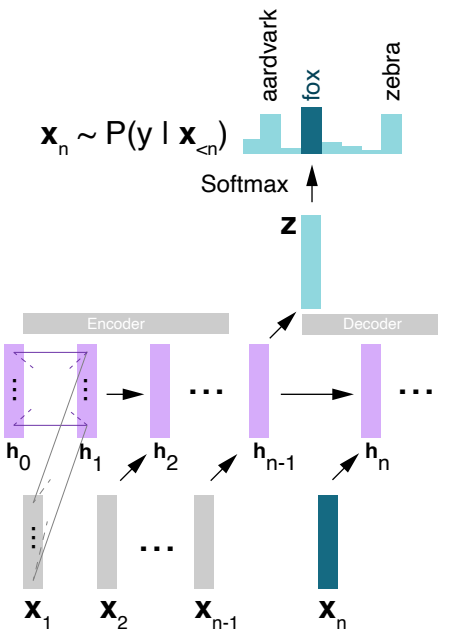
Large Language Models

Neural Language Models [ML:IV-125] [ML:IX-29]

Any neural network architecture is able to learn a language modeling task with the (standard) cross-entropy loss.

Recurrent Neural Network

- Aggregate (encode) the sequence $x_1, \ldots, x_{n-1}$ iteratively into a hidden layer h.
- At each step t, the hidden layer $\mathbf{h}_t$ depends on the previous hidden layer $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$
- At step $n-1$, decode the output $\mathbf{z}$ from the hidden layer $\mathbf{h}_{n-1}$.



Problems:

- Struggles with long-term dependencies (memory is limited to 1 hidden layer).
- Hard to train in parallel.

Remarks:

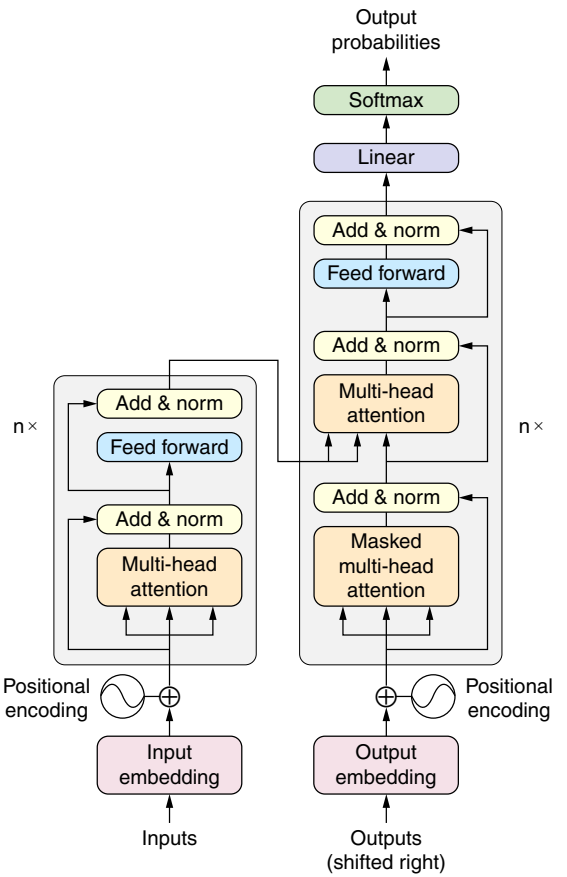
- RNN Notation: t incidates the timestep. The weight matrices w^h for encoding and w^o are the same at every t. The training process tries to fit these two weight matrices.
- Convolutional neural networks also work. However, many stacks are neccessary for long contexts and many stacks require residual and highway connections to train the CNN effectively. All of this makes the training very expensive and the performance is usually not worth it.
- Due to the frequent multiplication of $\mathbf{h}$ and $\mathbf{w^h}$, RNNs face the *vanishing gradient problem* when training: the training signal becomes very small quickly and only signals at the end of the sequence have a meaningful impact.
- Long Short-Term Memory Networks (LSTMs) and Gated Recurrent Units (GRUs) can lessen the vanishing gradient by providing relevance, update, and forget gates in each cell. This greatly improves language modeling performance.

Large Language Models

Transformer Architecture Overview [Vaswani, 2017][ML:IX]

The transformer is a neural network architecture that forms the basis of all current large language models.

- Developed as an encoder-decoder model for sequence-to-sequence tasks like machine translation.
- Long-term dependencies are modelled between words using attention.
- Parallelization: Attention uses no recurrent connections. It can be trained more efficiently at scale.

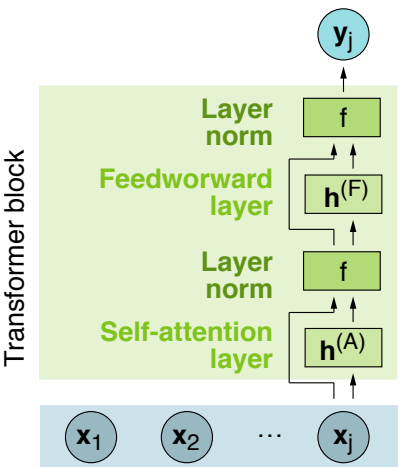


Large Language Models

Transformer Architecture Overview

A transformer models consist of at least one block per output token. Blocks can be stacked in height to add parameters.

- A block has one output token vector $\mathbf{y}_j$ and several input token vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots$.
- Self-attention layers $\mathbf{h}^{(A)}$:
  - Maps a sequence $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to $\mathbf{y}_1, \ldots, \mathbf{y}_n$
  - Contextual embeddings: Model each $\mathbf{x}_j$ based on the context of the other inputs. Learn which inputs are relevant to which others.



- Layer norm: Regularize $\mathbf{h}$ via mean and standard deviation to get a convex function for efficient gradient computation. [ML:III-65]
- Feed forward layer $\mathbf{h}^{(F)}$: Map layer to block input length for block stacking.
- Residual connections: Pass the block input on to deeper layers.

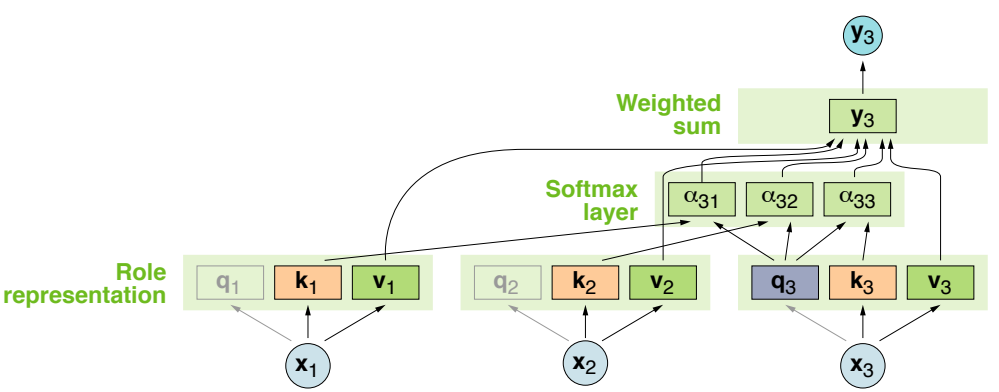
Large Language Models

Transformer Architecture Overview

Self-attention scores the relevance of all context tokens $\mathbf{x}_k$ for the focus token $\mathbf{x}_j$ fo the block. The scores are used to weight the influence of $\mathbf{x}_k$ for the output $\mathbf{y}_j$ of $\mathbf{x}_j$.

- Value (v_j) weights the influence of $\mathbf{x}_j$ without context.
- The α terms are the relative influence of each context on the focus tokens.
  - Query (q_j) weights the focus token $\mathbf{x}_j$.
  - Key (k_j) weights the respective context token $\mathbf{x}_k$.

Self-attention computation per head (focus x_3)

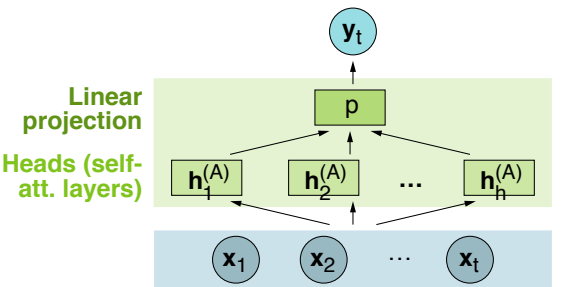


Large Language Models

Transformer Architecture Overview

Multi-headed self-attention:

- Words may relate in multiple ways simultaneously.
- Each transformer block has multiple parallel attention *heads*.
- Each head learns different weights k, q, and v.



The output $\mathbf{y}_t$ of a block is the **contextualized embedding** of the focus token $\mathbf{x}_t$.

- It resembles a word embedding, but contains information determined by the context words and regarding the task the model is trained for.
- Contextualized embedding are strong text representations.

Remarks:

- The inputs $\mathbf{x}_i$ consist of the token embedding and a *positional embedding* which encodes the position of the token in the input sequence.
- Attention originates as an extension to LSTMs where it was meant to inform the decoder about the importance of the input tokens towards the current decoding focus.
- We cover the block-internal attention (self-attention) with one input and one output sequence. The transformer decoder additionally uses cross-attention, which takes two input sequences.
- Attention makes for the bulk of the memory use of a transformer and thus limits the model size and the context length. Long-context models like the *Longformer* limit the number of tokens that are attended to.

Large Language Models

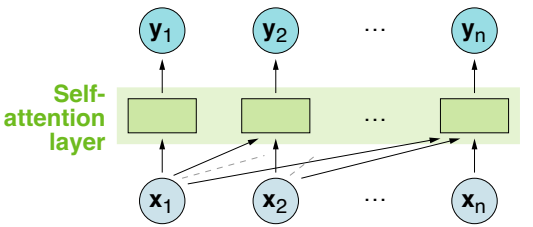
Transformer Language Models: Types

There are three archetypes of transformer language models:

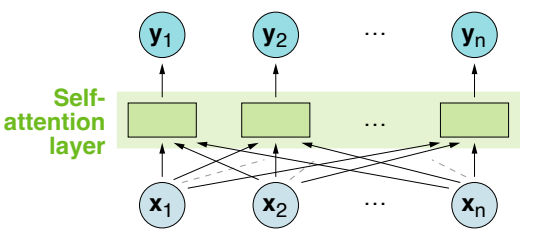
1. Autoregressive Language Models
   The input tokens (context) of each block are all left-side tokens. This equals the language modeling task $P(\mathbf{x}_k \mid \mathbf{x}_{<k})$.

2. Bidirectional Language Models
   The input tokens (context) of each block are all left and right-side tokens. This allows strong attention scores and leads to strong contextualized embeddings. Usefull for classification and tagging tasks.

Left-side self-attention



Full self-attention



3. Encoder-decoder Language Models
   Combines both architectures. The encoder applies full attention, the decoder right-side attention. The decoder has an additional cross-attention layer to that takes the output of the encoder and the output of the decoder self-attention to combine both sides. For conditional language modeling.

Large Language Models

Pre-training and Fine-tuning

Problem: Training a large transformer model requires a lot of data.

- BERT (2019) - 110M parameters and ca. 3 billion words training text.
  The English Wikipedia
- Current generative LLMs have 7–100 billion parameters.
- There are no big-enough task-specific corpora.

Large transformer models are first pre-trained and then fine-tuned for specific tasks.

- **Pre-training:** Train a general transformer model via self-supervision (i.e. language modeling) where the class is derived from the data.
- **Fine-tuning:** Train a pre-trained model on limited data for a downstream task like classification, translation, ...
- Both pre-training and fine-tuning tasks depend on the model architecture.

Large Language Models

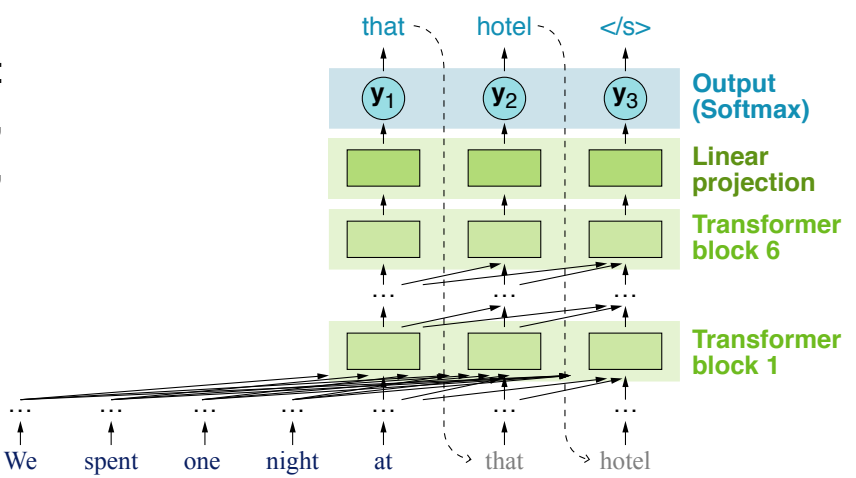
Autoregressive Language Models [ML:IX] [NLP:IV-XXX]

Autoregressive language models use left-side self-attention to contextualize all prior tokens $\mathbf{x}_{<k}$.

- **Pre-training:** Predict the next word k using teacher forcing: k is always taken from the data, independently from the model output at $k-1$.
- Used for text generation by using the output at $k-1$ as the input for k.
- **Pre-training Data:** Often web text. FineWeb with 15T tokens from the CommonCrawl

Examples:

- All current generative LLMs: GPT, Llama, Mistral, Falcon, Gemini, Gemma, Claude, Qwen, Command-R, ...
- Parameters (Mistral-7B):
  - Vocabulary: 32,000
  - Blocks per token: 32
  - Attention Heads: 32
  - Embeddings: 4,096



Large Language Models

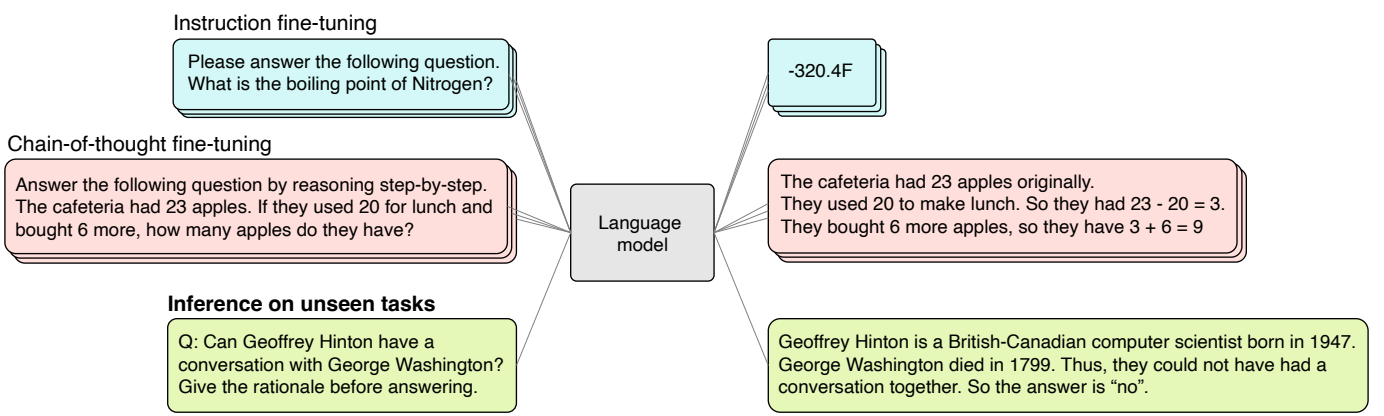
LLM Fine-tuning: Instructions [Chung, 2022] [Wang, 2022]

Autoregressive LLMs are fine-tuned for multi-task solving via instruction tuning.

- Training data consist of instruction, data, and answer.

```
Answer the following question.  When did Beyonce start becoming popular?  In the late 1990s.
```

- Instruction and data are passed as the prefix. The model is trained to generate the answer.
- Instruction data is synthesized from many task specific datasets using templates. Translation, Question Answering, Sentiment Analysis, ...
- The FLAN dataset contains ca. 1,800 tasks and 1.4B tokens.



Large Language Models

Bidirectional Language Models [Devlin, 2019]

Bidirectional transformer language models use full self-attention to contextualize all tokens in an input sequence. Fixed input/output size

- **Pre-training:** Masked language modeling and next sentence prediction on pairs of sentences. BERT. Other models (RoBERTa, Electra) are pretrained differently.
- Masked language modeling: Randomly replace 30% of words with a [MASK] token. Train the model to predict the masked word.
- Next sentence prediction: Predict if the second sentence (after the [SEP] token) follows the first one in the source documents. ca. 50% of sentence pairs
- First token x_i is always [CLS]. The output y_i is used for classification.

Unsupervised input	The first series became the most-watched series in Northern Ireland since modern records. The series was renewed shortly after the pilot episode aired.
Masked input	[CLS] The first [MASK] became the [MASK]-watched [MASK] in Northern [MASK] since modern records. [SEP] The series was [MASK] shortly after the pilot [MASK] aired. [PAD] [PAD] . . .

Large Language Models

Bidirectional Language Models [Devlin, 2019]

Bidirectional transformer language models use full self-attention to contextualize all tokens in an input sequence. Fixed input/output size

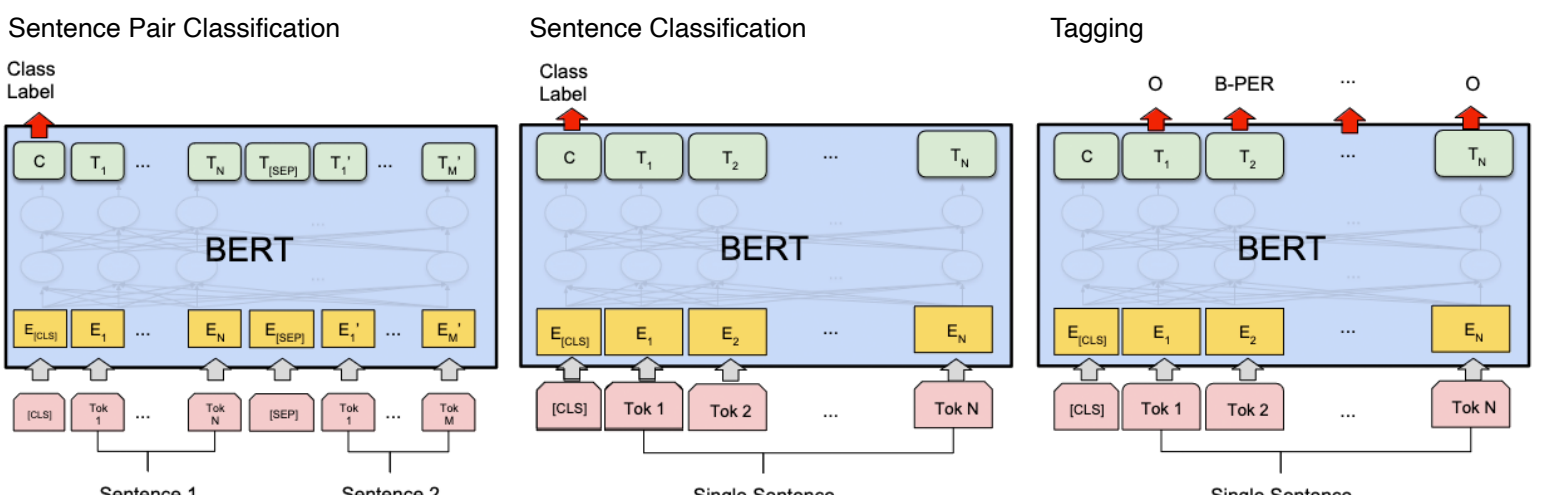
- **Pre-training:** Masked language modeling and next sentence prediction on pairs of sentences. BERT. Other models (RoBERTa, Electra) are pretrained differently.
- Masked language modeling: Randomly replace 30% of words with a [MASK] token. Train the model to predict the masked word.
- Next sentence prediction: Predict if the second sentence (after the [SEP] token) follows the first one in the source documents. ca. 50% of sentence pairs
- First token x_i is always [CLS]. The output y_i is used for classification.
- **Pre-training Data:** High quality text (i.e. Wikipedia). Many genre and language specific versions. CamemBERT, TweetBERT
- **Parameters (BERT base):** Vocabulary: 30,522, Blocks per token: 12, Attention Heads: 12, Embedding Size: 768, Input length: 512

Large Language Models

BERT Fine-tuning [Devlin, 2019]

Bidirectional LLMs can be fine-tuned on various tasks by adding and training a classification layer on top of the [CLS] output or the contextualized embeddings.

- **Sequence Classification** Sentiment Analysis, Toxicity, ...
- **Sentence Pair Classification:** Ranking (`[CLS] Query [SEP] Document`) Argumentation Pro/Con (`[CLS] Topic [SEP] Argument`)
- **Tagging** Part-of-Speech, Named Entities, ...
- **Question Answering** `[CLS] Question [SEP] Reference Document`



Large Language Models

Encoder-Decoder Language Models [Lewis, 2019][Raffel, 2019]

Encoder-decoder LLMs combine a bidirectional transformer to produce a strong encoding of the input with a flexible autoregressive transformer for decoding.

- Works well for generation tasks with a strong condition and little creativity.
- **Pre-training:** Corrupt the encoder input and re-generate the uncorrupted input (de-noising).
- Corrupt by masking, shuffling, or removing tokens or sub-sequences.
- **Fine-tune:** Summarization, translation, conclusion generation, de-biasing, . . .
- Examples: BART, T5, . . .