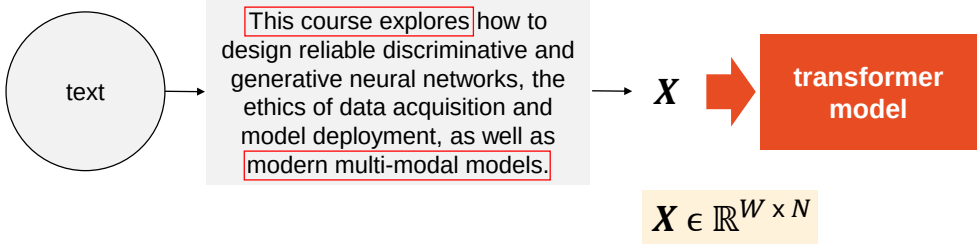
EE-559
Deep Learning

What's on today?

- Natural Language Processing: how to analyse / synthesize language
- Tokens: on breaking text into small units
- Self-attention: how a model can focus on relevant parts of the input
- Transformer: how to use attention in a neural network
- Encoder model: how to process text to create a useful representation
- Decoder model: how to generate text
- Encoder-decoder model: how to map a sequence to a sequence
- Exercises: exploring attention and transformers

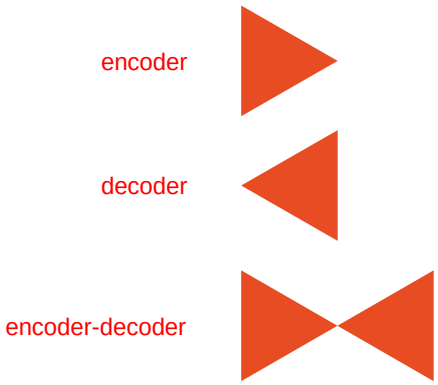
Natural Language Processing

Natural Language Processing (NLP)



Concepts:
Analysis and synthesis of speech and language, syntax, pronouns, connections between words

Transformer models



Generates a representation from the text embedding
e.g. for text or word classification

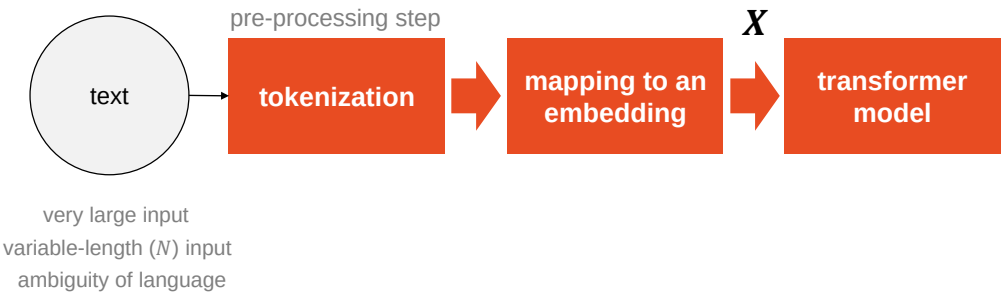
Predicts the next token to continue the input text
e.g. autoregressive language model

Converts a text string into another
e.g. sequence-to-sequence tasks

Concepts:
Sentiment analysis, name entity recognition, tokens, generative pre-trained transformers, language translation

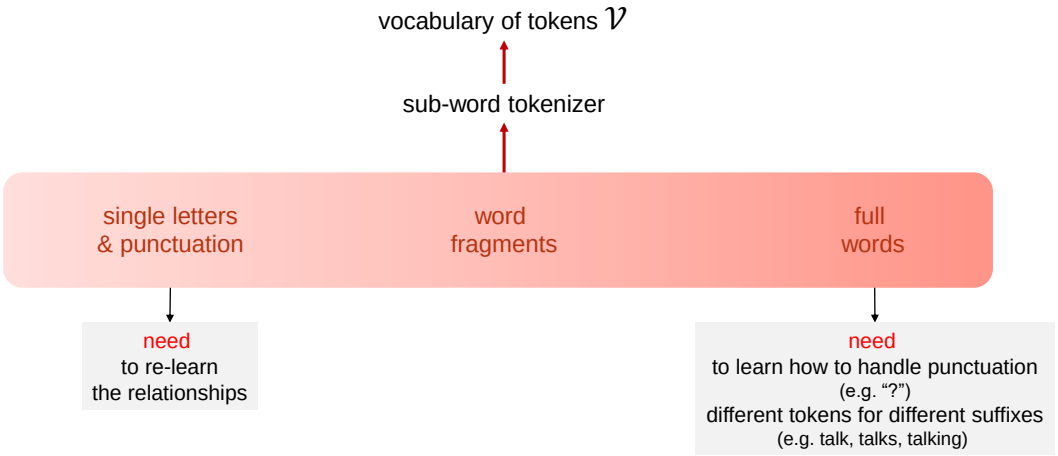
Tokens

Tokenization of text



Concepts:
Words and word fragments, sub-word tokenizer, token embedding

Tokenization: choices



Concepts:
Use frequency of commonly occurring sub-strings to merge them (byte pair encoding)

Tokenizers

word and punctuation tokens

Whitespace

separates words

Punctuation-based

separates words and punctuation

sub-word tokens

Byte Pair Encoding

each character as a word → repeatedly merges the **most frequent pairs of characters**

WordPiece

considers prefixes and suffixes (**sub-word units**) → better for rare and unknown words

Concepts: Counting the number of words in a text, identifying frequently occurring character combinations in training text, a tokenizer does not work equally well for all languages

More on units of text

Between words and characters:
A Brief History of Open-Vocabulary Modeling and Tokenization in NLP

Sabrina J. Mielke [1,2] Zaid Alyafeai [3] Elizabeth Salesky [1]
Colin Raffel [2] Manan Dey [4] Matthias Gallé [5] Arun Raja [6]
Chenglei Si [7] Wilson Y. Lee [8] Benoît Sagot [9*] Samson Tan [10*]
BigScience Workshop Tokenization Working Group

[1]Johns Hopkins University [2]HuggingFace [3]King Fahd University of Petroleum and Minerals [4]SAP
[5]Naver Labs Europe [6]Institute for Infocomm Research, A*STAR Singapore [7]University of Maryland
[8]BigScience Workshop [9]Inria Paris [10]Salesforce Research Asia & National University of Singapore
sjm@sjmielke.com

Abstract

What are the units of text that we want to model? From bytes to multi-word expressions, text can be analyzed and generated at many granularities. Until recently, most natural language processing (NLP) models operated over words, treating those as discrete and atomic tokens, but starting with byte-pair encoding (BPE), subword-based approaches have become dominant in many areas, enabling small vocabularies while still allowing for fast inference. Is the end of the road character-level model or byte-level processing? In this survey, we connect several lines of work from the pre-neural and neural era, by showing how hybrid approaches

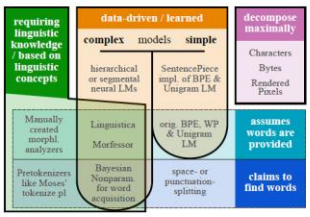


Figure 1: A taxonomy of segmentation and tokenization algorithms and research directions

arXiv:2112.10508

Mapping to an embedding

input embeddings

$\boldsymbol{X} \in \mathbb{R}^{W \times N}$

$\boldsymbol{X} = \mathbf{E}\,\mathbf{T}$

W size of the embedding (e.g. 1024)

N length of the input (variable)

$|\mathcal{V}|$ size of the vocabulary (e.g. 30,000 or 100,000)

$\mathbf{E} \in \mathbb{R}^{W \times |\mathcal{V}|}$

learned matrix with the embedding of the tokens

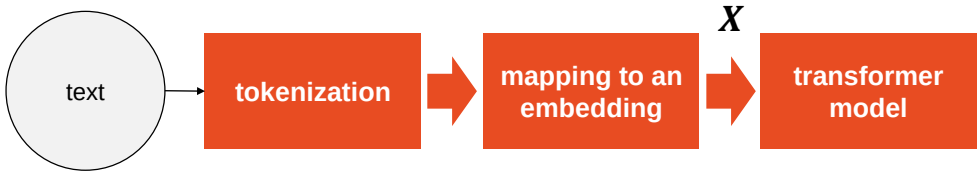
$\mathbf{T} \in \mathbb{R}^{|\mathcal{V}| \times N}$

matrix encoding N input tokens

columns are $|\mathcal{V}| \times 1$ one-hot vectors

Concepts:
One-hot vector: the only non-zero element is the entry corresponding to the token, **T** is sparse

Input embeddings



$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m, \ldots, \boldsymbol{x}_N]$$

$$\boldsymbol{x}_m \in \mathbb{R}^W$$

Concepts:
Input of length N, embedding of size W, embedding of the token ($\boldsymbol{x}_m$)

Self-attention

Self-attention

$$s_n[x_1, x_2, \ldots, x_N] = \sum_{m=1}^{N} a[x_m, x_n](\theta_0^v + \theta_1^v x_m)$$

self-attention (connection between word representations)

attention (scalar weight)

value (linear transformation of each x_m)

$$= \sum_{m=1}^{N} a[x_m, x_n] v_m$$

$a[., x_n] \geq 0$

$\sum_{n=1}^{N} a[., x_n] = 1$

$X \in \mathbb{R}^{W \times N}$

$x_m \in \mathbb{R}^W$

$$v_m = \theta_0^v + \theta_1^v x_m$$

biases weights

$\theta_0^v \in \mathbb{R}^W$

$\theta_1^v \in \mathbb{R}^{W \times W}$

Concepts:
Connections between the inputs, sparse attention coefficients, value (v_m)

Attention

$\boldsymbol{q}_n = \boldsymbol{\Theta}_0^q + \boldsymbol{\Theta}_1^q \boldsymbol{x}_n$ query

$\boldsymbol{k}_m = \boldsymbol{\Theta}_0^k + \boldsymbol{\Theta}_1^k \boldsymbol{x}_m$ key

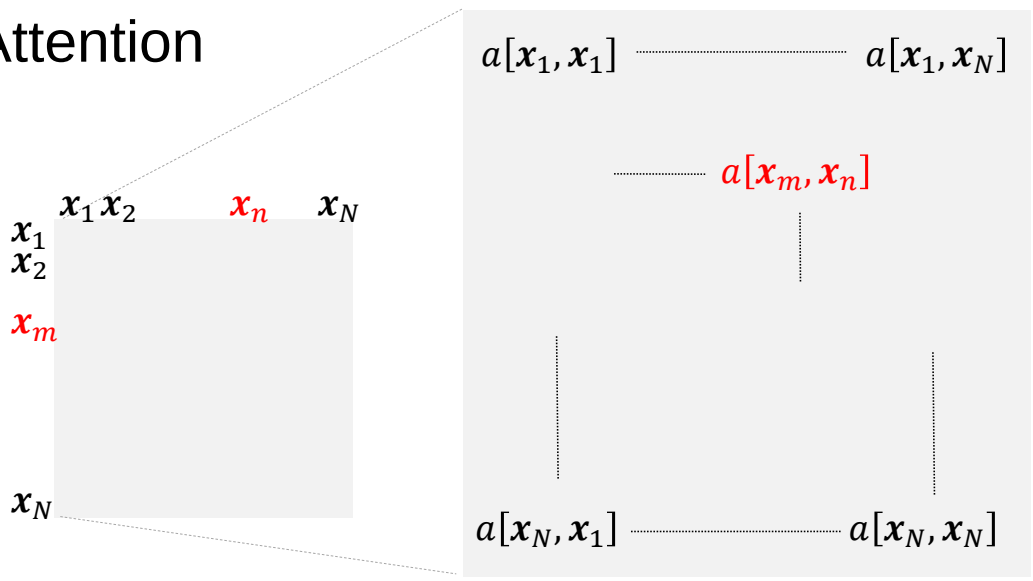
$(\boldsymbol{k}_m)^{\mathrm{T}} \boldsymbol{q}_n$ measure of similarity (dot product)

$$a[\boldsymbol{x}_m, \boldsymbol{x}_n] = \mathrm{softmax}_m\left((\boldsymbol{k}_*)^{\mathrm{T}} \boldsymbol{q}_n\right) = \frac{e^{(\boldsymbol{k}_m)^{\mathrm{T}} \boldsymbol{q}_n}}{\sum_{m'=1}^{N} e^{(\boldsymbol{k}_{m'})^{\mathrm{T}} \boldsymbol{q}_n}}$$

non-linear due to the dot product and the softmax

Concepts: Query and key from information retrieval, dot-product self-attention, measure of similarity between the input $\boldsymbol{x}_n$ and all the other inputs

Attention



Concept:
Modeling data with *long-range* dependencies

Self-attention

$$\boldsymbol{x}_1 \qquad \boldsymbol{s}_1[\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_N] = \sum_{m=1}^{N} a[\boldsymbol{x}_m, \boldsymbol{x}_1]\boldsymbol{v}_m$$

$$\boldsymbol{x}_n \qquad \boldsymbol{s}_n[\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_N] = \sum_{m=1}^{N} a[\boldsymbol{x}_m, \boldsymbol{x}_n]\boldsymbol{v}_m$$

scales linearly with N

quadratic dependence on N

$$\boldsymbol{x}_N \qquad \boldsymbol{s}_N[\boldsymbol{x}_1, \boldsymbol{x}_2, .., \boldsymbol{x}_N] = \sum_{m=1}^{N} a[\boldsymbol{x}_m, \boldsymbol{x}_N]\boldsymbol{v}_m$$

Concepts: Value is a linear function of the input, same weights and biases applied to each input, attention weights are *sparse* and are a *non-linear* function of the input, parallelization

$\boldsymbol{X} \in \mathbb{R}^{W \times N}$

$\boldsymbol{x}_m \in \mathbb{R}^{W}$

Self-attention: matrix form

$\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_0^v, \boldsymbol{\Theta}_1^v, \boldsymbol{\Theta}_0^q, \boldsymbol{\Theta}_1^q, \boldsymbol{\Theta}_0^k, \boldsymbol{\Theta}_1^k\}$ shared set of parameters across inputs

$$\boldsymbol{V}(\boldsymbol{X}) = \boldsymbol{\Theta}_0^v \mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}_1^v \boldsymbol{X}$$

$$\boldsymbol{Q}(\boldsymbol{X}) = \boldsymbol{\Theta}_0^q \mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}_1^q \boldsymbol{X}$$

$\mathbf{1}_N$: N-dimensional vector containing only 1s

$$\boldsymbol{K}(\boldsymbol{X}) = \boldsymbol{\Theta}_0^k \mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}_1^k \boldsymbol{X}$$

$$\boldsymbol{S}(\boldsymbol{X}) = \boldsymbol{V}(\boldsymbol{X})\mathrm{softmax}\left(\left(\boldsymbol{K}(\boldsymbol{X})\right)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{X})\right)$$

$$= \boldsymbol{V}\,\mathrm{softmax}\left((\boldsymbol{K})^{\mathrm{T}}\boldsymbol{Q}\right)$$

scaled dot-product self-attention

$$\boldsymbol{S}(\boldsymbol{X}) = \boldsymbol{V}\,\mathrm{softmax}\left(\frac{(\boldsymbol{K})^{T}\boldsymbol{Q}}{\sqrt{W_q}}\right)$$

dimension of queries (and keys)

Concepts:
Scaled dot-product self-attention: facilitates training, avoides domination of the largest value

Multi-head self-attention

$$\boldsymbol{V}_h(\boldsymbol{X}) = \boldsymbol{\Theta}^v_{0h}\mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}^v_{1h}\boldsymbol{X}$$

$$\boldsymbol{Q}_h(\boldsymbol{X}) = \boldsymbol{\Theta}^q_{0h}\mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}^q_{1h}\boldsymbol{X}$$

$$\boldsymbol{K}_h(\boldsymbol{X}) = \boldsymbol{\Theta}^k_{0h}\mathbf{1}_N^{\mathrm{T}} + \boldsymbol{\Theta}^k_{1h}\boldsymbol{X}$$

$$\boldsymbol{\Theta}_h = \left\{\boldsymbol{\Theta}^v_{0h}, \boldsymbol{\Theta}^v_{1h}, \boldsymbol{\Theta}^q_{0h}, \boldsymbol{\Theta}^q_{1h}, \boldsymbol{\Theta}^k_{0h}, \boldsymbol{\Theta}^k_{1h}\right\}$$

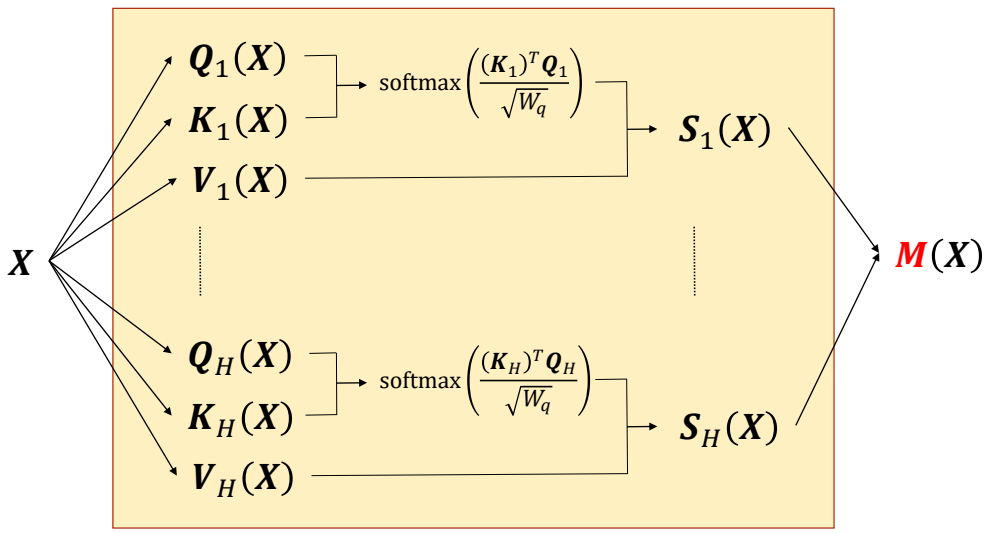
$$\boldsymbol{S}_h(\boldsymbol{X}) = \boldsymbol{V}_h \,\mathrm{softmax}\left(\frac{(\boldsymbol{K_h})^T\boldsymbol{Q_h}}{\sqrt{W_q}}\right)$$

$$\boldsymbol{M}(\boldsymbol{X}) = \boldsymbol{\Theta_M}\left(\left(\boldsymbol{S}_1(\boldsymbol{X})\right)^{\mathrm{T}}, \left(\boldsymbol{S}_2(\boldsymbol{X})\right)^{\mathrm{T}}, \ldots, \left(\boldsymbol{S}_H(\boldsymbol{X})\right)^{\mathrm{T}}\right)^{\mathrm{T}}$$

$\boldsymbol{\Theta_M}$ linear transformation

Concepts:
Self-attention mechanisms (heads) applied in parallel, outputs are vertically concatenated

Multi-head self-attention



Transformer

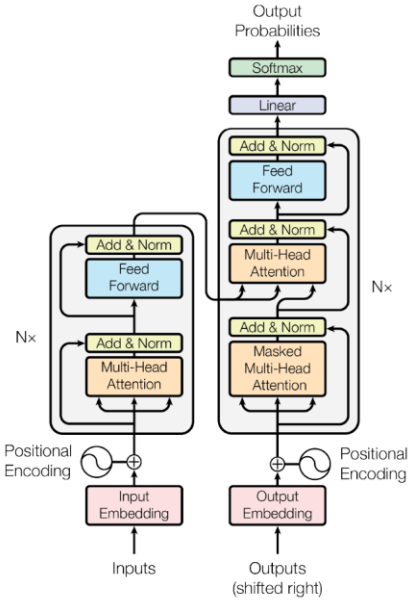
Attention Is All You Need

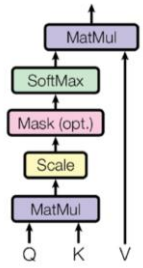
Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

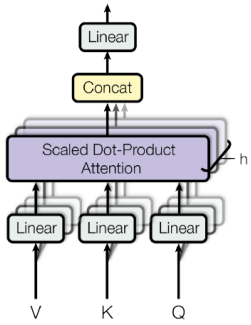
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

arXiv:1706.03762





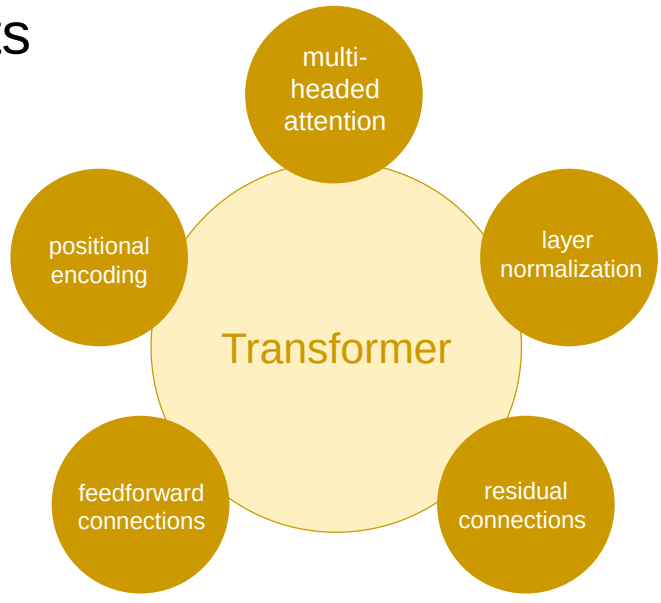
scaled dot-product attention



multi-head attention

arXiv:1706.03762

Key components



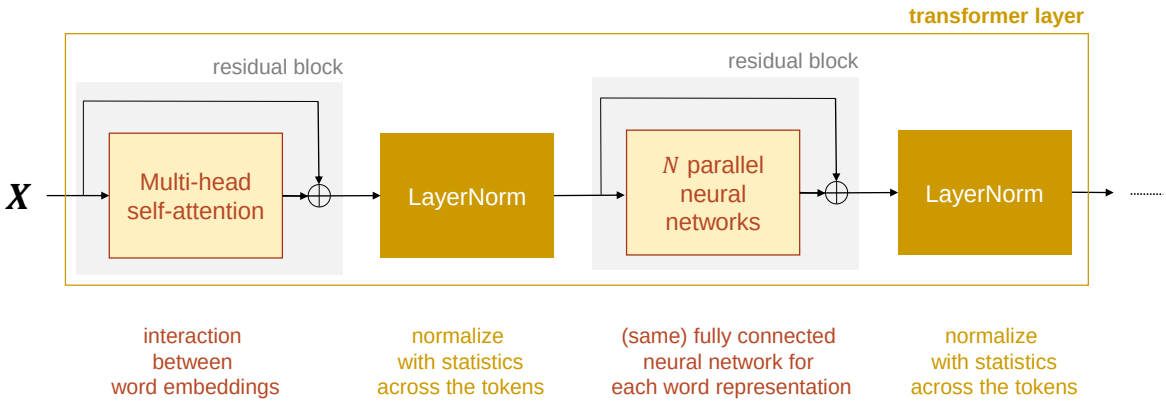
Positional encoding

$X \rightarrow X + \Pi$

Π
- each column of Π is unique (information about the *position* in the input sequence)
- added to the input only or at every layer
- learned or "hand-crafted"

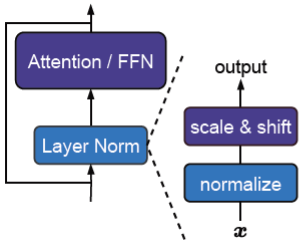
Concept:
Self-attention disregards the order of the tokens in a sentence

Transformer layer

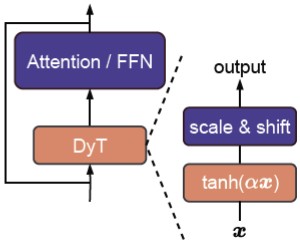


Concepts:
Output has the same size as the input (W x N), residual block: output added to the original input

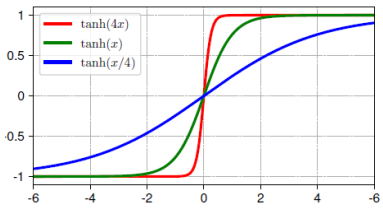
Layer normalization or Dynamic Tanh?



original transformer block with layer normalization



block with Dynamic Tanh (DyT) layer

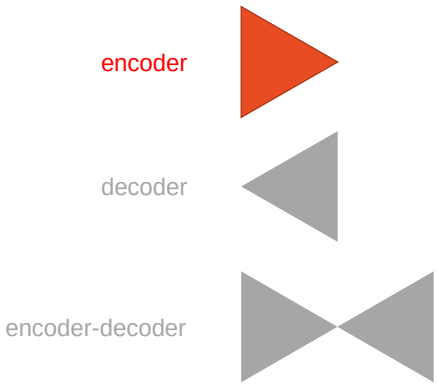


arXiv:2503.10622

Concept:
Transformers without normalization

Encoder model

Transformer models



Generates a representation from the text embedding
e.g. for text or word classification

Predicts the next token to continue the input text
e.g. autoregressive language model

Converts a text string into another
e.g. sequence-to-sequence tasks

Encoder model example: BERT

$P = 340{,}000{,}000$ parameters

$W = 1{,}024$ size of the embedding

$|\mathcal{V}| = 30{,}000$ size of the vocabulary (number of tokens)

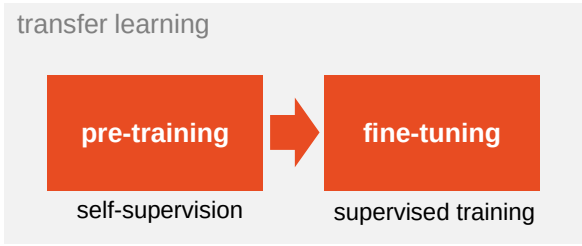
24 transformers (transformer layers)

$H = 16$ heads for the self-attention of each transformer

$W_v = W_q = W_k = 64$ dimension of the value, query, key

Concept: Bidirectional Encoder Representations from Transformers (BERT):
jointly conditioning on both left and right context (in all layers)

Training BERT

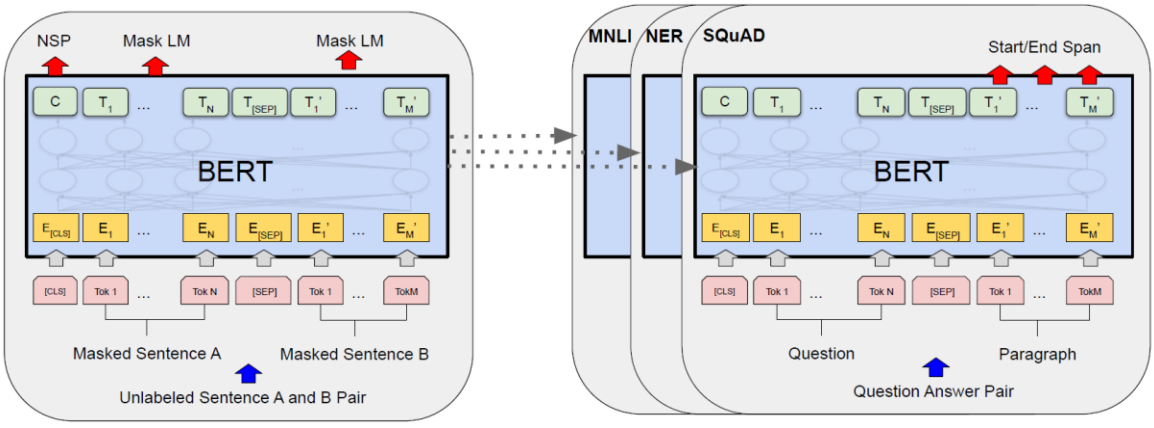


to learn general information about the statistics of language (on a large corpus of text)

adapted to solve a **task** using a smaller body of supervised training data

Concepts: Missing word prediction, understanding syntax, (statistical) understanding about the world, fine-tune with extra layer after the transformer network

BERT: pre-training and fine-tuning

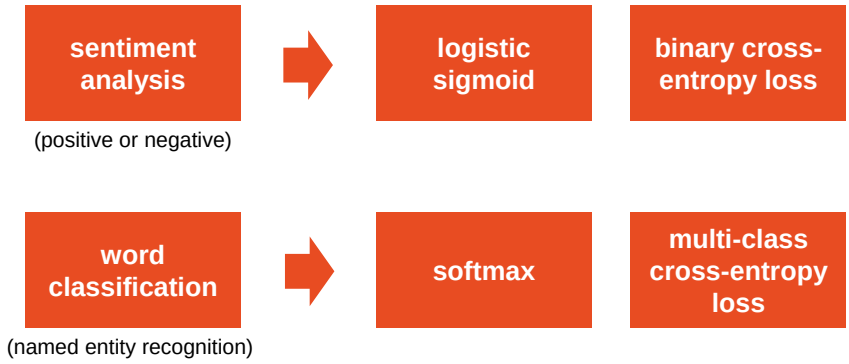


pre-training

fine-tuning
(all parameters)

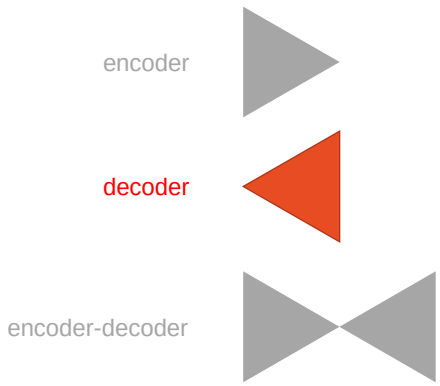
arXiv:1810.04805

BERT: tasks



Decoder model

Transformer models



Generates a representation from the text embedding
e.g. for text or word classification

Predicts the next token to continue the input text
e.g. autoregressive language model

Converts a text string into another
e.g. sequence-to-sequence tasks

Decoder model example: GPT3

$P = 175{,}000{,}000{,}000$ parameters

$W = 12{,}288$ size of the embedding

$|\mathcal{V}| = 50{,}257$ size of the vocabulary (number of tokens)

96 transformers

$H = 96$ heads for the self-attention of each transformer

$W_v = W_q = W_k = 128$ dimension of the value, query, key

Context:
GPT4 has got $P = 1{,}800{,}000{,}000{,}000$ parameters

GPT3: next token prediction

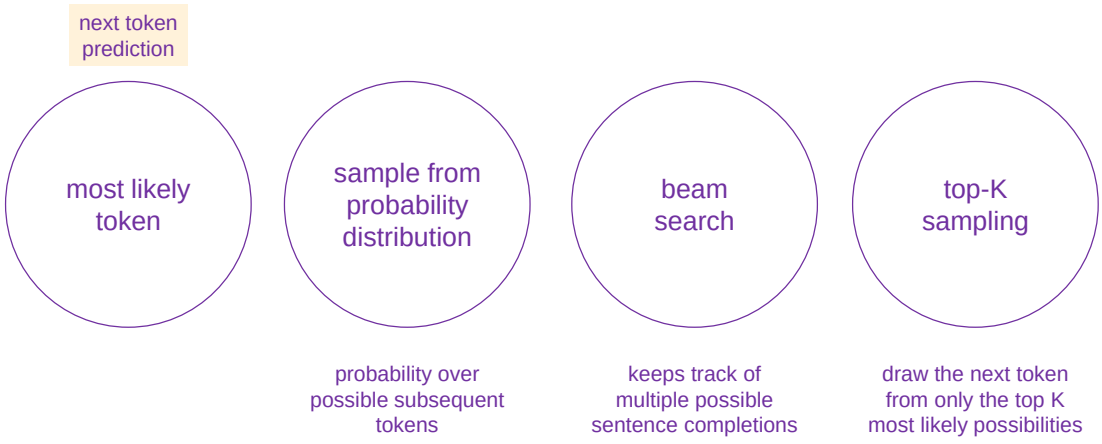
Each token t_n only interacts with previous tokens (masked self-attention)

$$P(\underbrace{t_1, t_2, \dots, t_{N-1}, t_N}_{\text{sentence}}) = P(t_1) \prod_{n=2}^{N} P(t_n | t_1, t_2, \dots, t_{n-1})$$

New extended sequence: fed back to the encoder for the next token prediction

Concepts: Autoregressive language model, predict the next token w/o access to the future, connection between maximizing the log probability of the tokens (loss function) and the next token prediction task

GPT3: generative model



slido

To understand transformers, was today's lecture more useful than prompting a large language model?

ⓘ Start presenting to display the poll results on this slide.

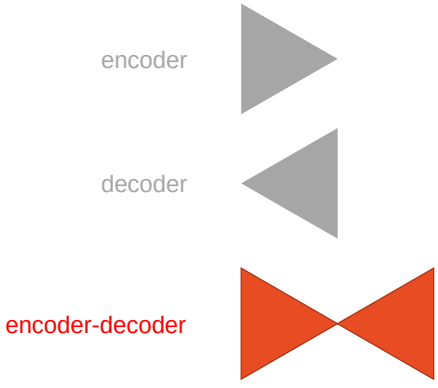
slido

In terms of learning about transformers, how did today's lecture compare to using a large language model?

ⓘ Start presenting to display the poll results on this slide.

Encoder-decoder model

Transformer models



Generates a representation from the text embedding
e.g. for text or word classification

Predicts the next token to continue the input text
e.g. autoregressive language model

Converts a text string into another
e.g. sequence-to-sequence tasks

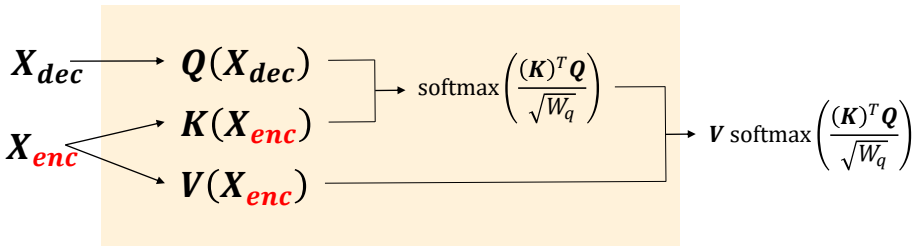
Example: machine translation

decoder layers also attend to the output of the **encoder**

each output token is *conditioned* on the previous output tokens and the source sentence

Concepts: Sequence-to-sequence task, information about the source language (encoder) and the target language statistics (decoder)

Encoder-decoder attention: cross attention



cross-attention

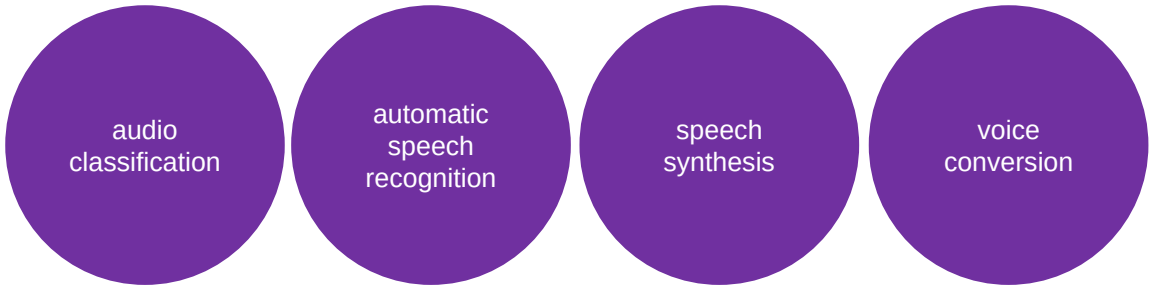
Concepts: Cross-attention (encoder-decoder attention), queries computed from the decoder embeddings, keys and values computed from the encoder embeddings

Transformers

Concepts: Transformer as *self-supervised* learning algorithm, sequences of *bytes*, trained to maximize the likelihood (mode covering), can be applied to any data type

Transformers for visual tasks

Transformers for audio tasks



Transformers for audio-visual tasks

Exercises

Today's exercises

Practice
You will become familiar with fine-tuning a **transformer** for binary text classification

Marked *(last one for Submission 1!)*
Includes **positional encoding**, scaled dot-product **attention** and multi-head **attention**

slido

What questions do you have regarding the mini-project?

ⓘ Start presenting to display the poll results on this slide.

What did we learn today?

- Natural Language Processing
- Tokens
- Self-attention
- Transformer
- Encoder model
- Decoder model
- Encoder-decoder model
- Exercises

EE-559
Deep Learning

andrea.cavallaro@epfl.ch