Deep Learning

Lecture 9 – Deep time series models 2

UPPSALA
UNIVERSITET

Carl Andersson
Department of Information Technology
Uppsala University

carl.andersson@it.uu.se

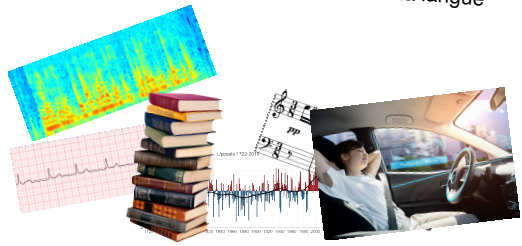
Recap

Summary lecture 8 (I/IV): The sequential problem

A sequential model can be used on:

- Sequential input, $x_{1:N}$
- Sequential output, $y_{1:T}$
- In any combination

Language translation $\iff$ La traduction de la langue



We divided the problems into 3 categories

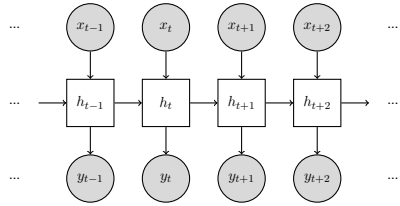
- Classification & regression
- System modeling
- Generative (which can be conditioned)

Summary lecture 8 (II/IV): Recurrent neural network

A basic RNN is the Elman network,

$$h_t = f(h_{t-1}, x_t)$$
$$y_t = g(h_t)$$

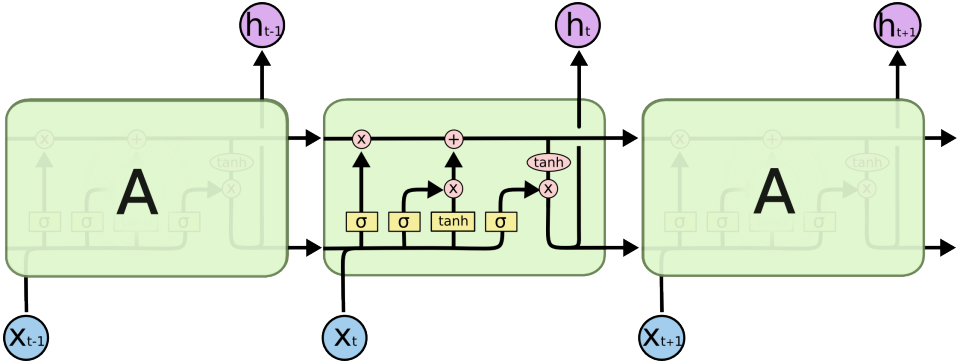
where both f and g are neural networks.



$$f(h_{t-1}, x_t) = \sigma(W_h h_{t-1} + W_x x_t + b_h)$$
$$g(h_t) = \sigma(W_y h_t + b_y)$$

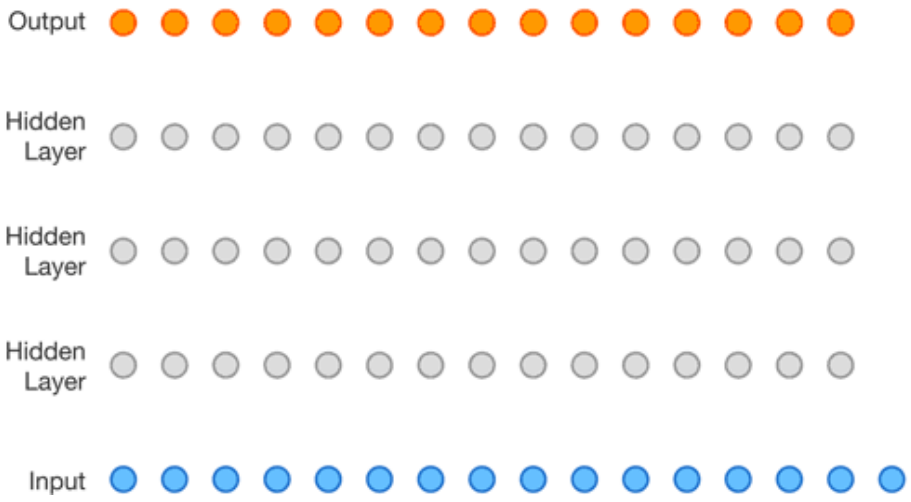
Summary lecture 8 (III/IV): Long-Short Term Memory

- 2 different states c_t for long memory and h_t for the last output
- Designed to propagate gradients better, make them more useful for tasks that require longer memory



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Summary lecture 8 (IV/IV): Temporal convolutional network



A. van der Oord et al, **Wavnet: A Generative model for audio**, *arxiv: 1609.03499*

UPPSALA UNIVERSITET

Revisit batch normalization

- Batch normalization
  - Performs poorly for sequential data
- Weight normalization
  - Unstable for really deep networks
- Layer normalization
  - Efficient for sequential data

https://towardsdatascience.com/different-normalization-layers-in-deep-learning-1a7214ff71d6

Transformer

Transformer

Problem formulation

- Typically NLP problem
- Generative (GPT2/GPT3) / conditional generative (translation)
- Classification

https://openai.com/blog/better-language-models/

Running example: Translation

- Translate from English to German
- Word level
- Generative, conditioned on sequence
- This example highlights all the different usages of transformers
  - Extract features from sequences
  - Generate sequences
  - Merge sequences

Inputs: The animal didn't cross the street because it was too tired

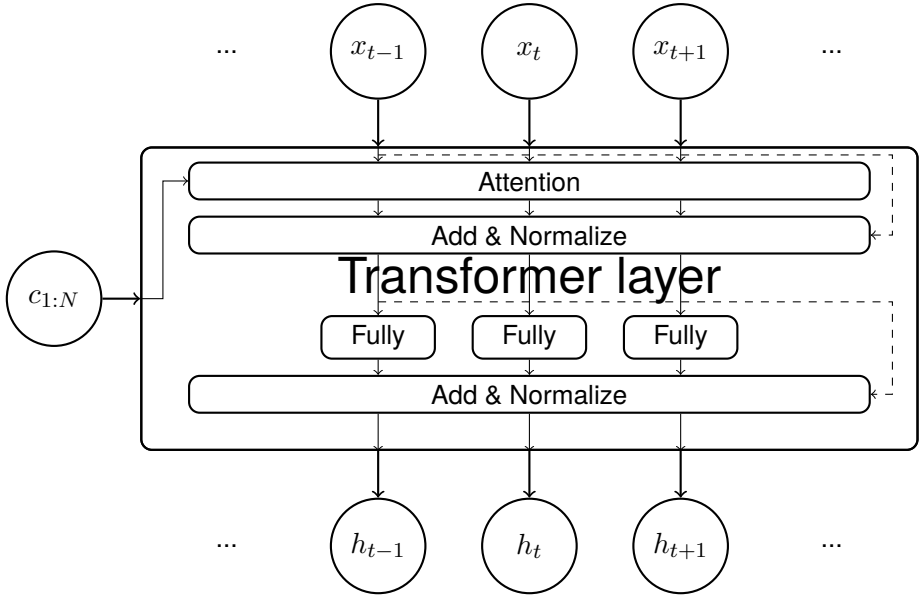
Outputs: Das Tier überquerte die Straße nicht, weil es zu müde war.

http://jalammar.github.io/illustrated-transformer/

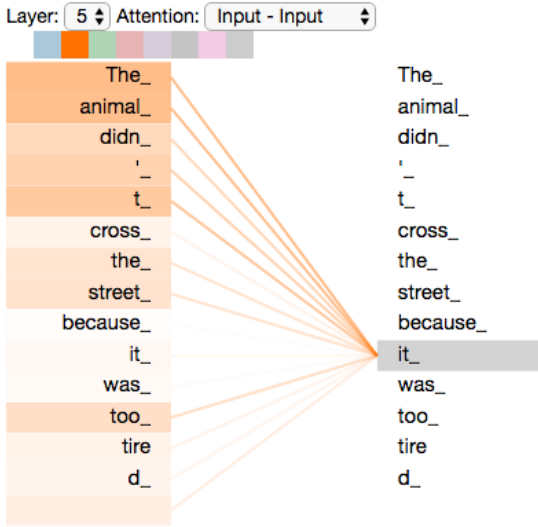
Intuition behind Transformers

- Transformers use **residual connections** and **attention**
- Can naturally be **conditioned** on an other sequence
- We can **interpret** some of hidden units in the network
- Achieves **long memory** without vaninshing gradients

Attention: Gates & residual connections



Attention: Interpretable



Attention: Long memory

- Attends to all data simultaneously without using a state
- Still have relative few parameters

Transformer block

Three kinds of building blocks

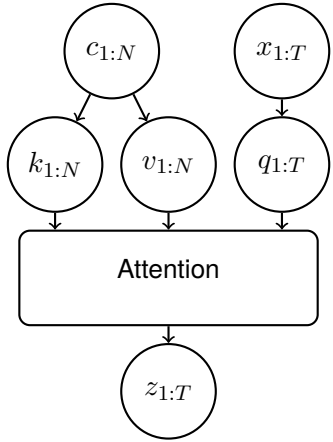
- Self-attention, fully connected
- Self-attention, autoregressive
- Auxiliary attention, conditional

Attention layer

The core concepts of attention are

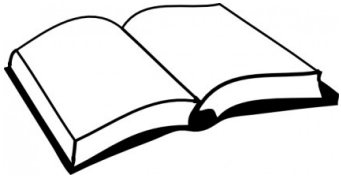
- Keys, $k_{1:N}$
- Values, $v_{1:N}$
- Queries, $q_{1:T}$

$$k_i = W_K c_i$$
$$v_i = W_V c_i$$
$$q_t = W_Q x_t$$



Attention layer: Intuition

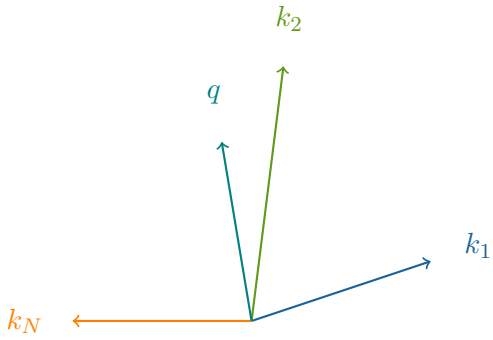
Keys and values act as a dictionary



Queries extracts information from the dictionary

Attention layer

Single query



$$(k_1, v_1)$$
$$(k_2, v_2)$$
$$\vdots$$
$$(k_N, v_N)$$

Match calculated as the dot product

$$m_i = k_i \cdot q$$

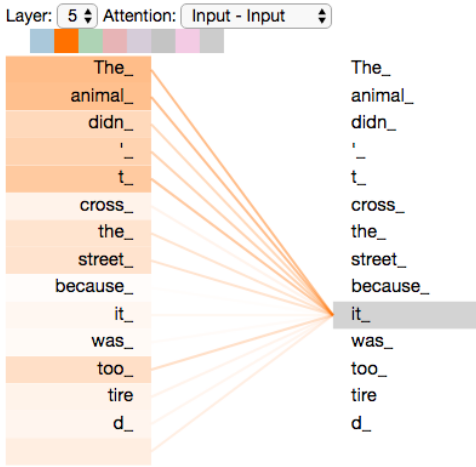
Attention layer

Single query

Normalized with softmax

$$n_i = \frac{\exp(m_i)}{\sum_k \exp(m_k)}$$

$$z = \sum_i \frac{\exp(m_i) v_i}{\sum_k \exp(m_k)}$$

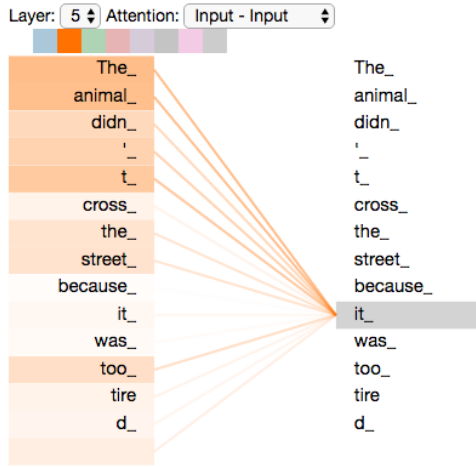


Attention layer

Multiple queries

$$K = (k_1, k_2, \ldots, k_N)$$
$$V = (v_1, v_2, \ldots, v_N)$$
$$Q = (q_1, q_2, \ldots, q_T)$$

$$Z = \text{softmax}(QK^{\mathsf{T}})V$$

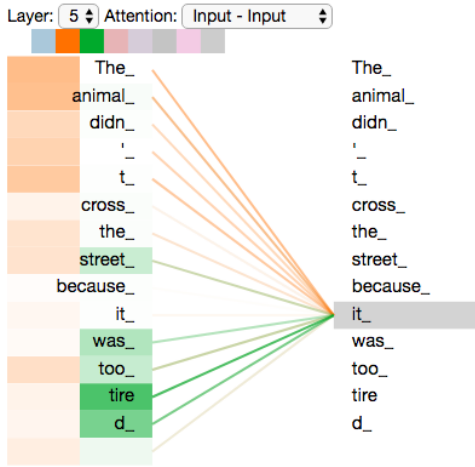


Multi-head attention

$$k_i^h = W_K^h c_i$$
$$v_i^h = W_V^h c_i$$
$$q_t^h = W_Q^h x_t$$

$$K^h = (k_1^h, k_2^h, \ldots, k_N^h)$$
$$V^h = (v_1^h, v_2^h, \ldots, v_N^h)$$
$$Q^h = (q_1^h, q_2^h, \ldots, q_T^h)$$

$$Z^h = \text{softmax}(Q^h (K^h)^\mathsf{T}) V^h$$

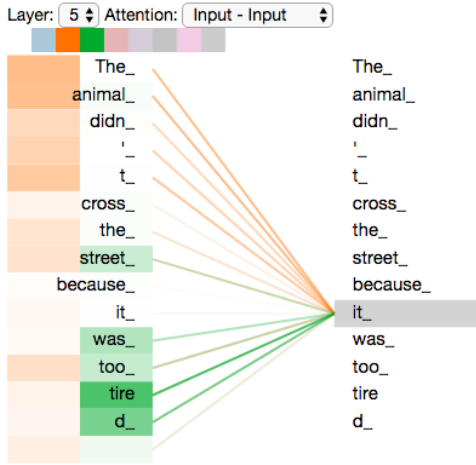


Self-attention, Fully connected

The conditioning sequence is the same as input sequence

Usage:

- Classification
- Feature extraction



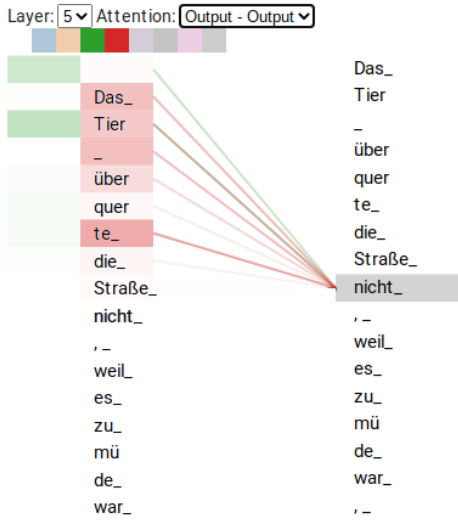
Self-attention, Autoregressive

The conditioning sequence is the same as input sequence but can not gather information from the future

Usage:

- Generation

Implemented by forcing the upper off-diagonal triangular part of $Q^h(K^h)^\mathsf{T}$ to $-\infty$

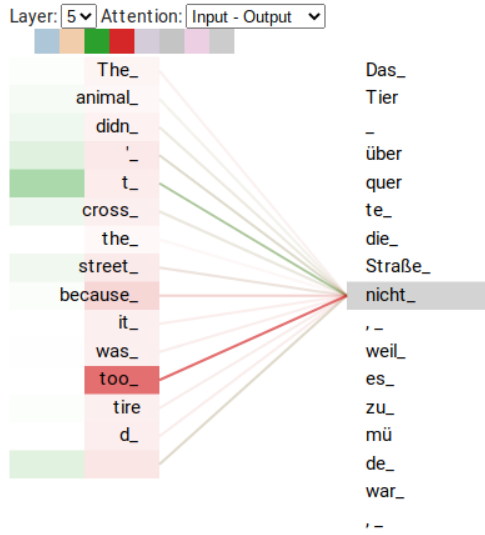


Auxiliary attention, conditional

The conditioning sequence and the input sequence are different

Usage:

- Conditioning



Word embedding

One-hot encoding

- The inputs and outputs are represented as one-hot encoded words
- Large corpus leads to very large vectors
- animal = $(0, 0, 0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{R}^{10000}$

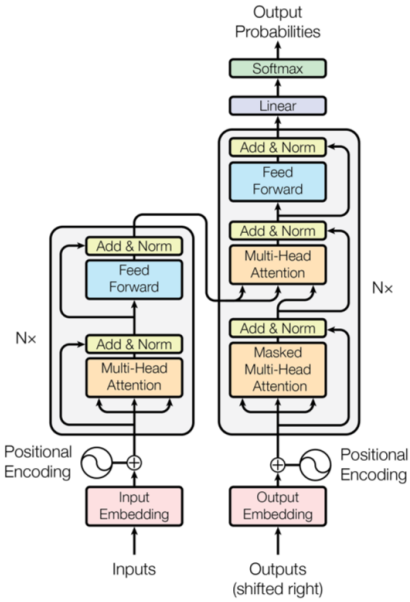
Word embedding

- Each word is represented as a vector
- Similar words will be represented as similar vectors
- animal = $(1.4, -1, 0.2, -1.1, \ldots, -0.4) \in \mathbb{R}^{256}$

Position embedding

- Attention is not dependent on the sequential order!
- Concatenate the word embeddings with positional embedding, $P(t)$
- Vector of trigonometric functions of t with different frequencies
- Typically the same size as the word embedding, i.e. $\mathbb{R}^{256}$

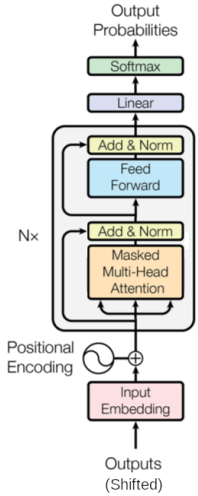
Example: Attention is all you need



- Left (Encoder): Self-attention, fully connected
- Right (Decoder): Self-attention, autoregressive (masked) & Auxiliary attention

`https://arxiv.org/abs/1706.03762`

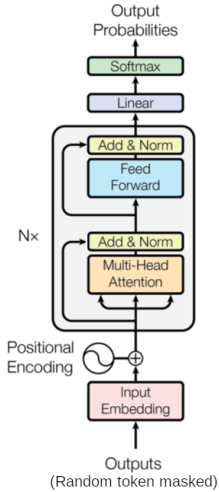
Example: GPT 1/2/3



- Self-attention, autoregressive
- 40000+ different words/tokens
- 12-96 transformer layers
- 125 million - 175 billion parameters
- Dataset is 300 billion words/tokens long

https://machinelearningknowledge.ai/openai-gpt-3-demos-to-convince-you-that-ai-threat-is-real-or-is-it/

Example: BERT



- Self-attention, fully connected
- Pretrained with self-supervised training
- Can be used for transfer learning for many language based problems
- Classification
  – Take the first encoding of the first token and use that in a prediction model

https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Conclusion

- Very versatile and performing set of model
- Very computationally heavy to train (use pretrained models)
- Primarily used for NLP problems