Deep Learning for Natural Language Processing
The Transformer model

UNIVERSITY OF GOTHENBURG

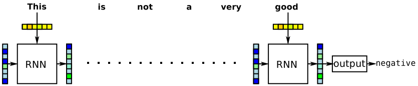
CHALMERS

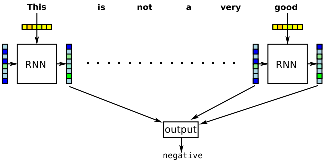
WASP | WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Richard Johansson

richard.johansson@gu.se

drawbacks of recurrent models

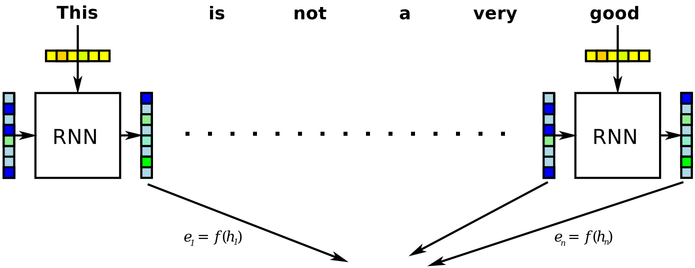




- even with GRUs and LSTMs, it is difficult for RNNs to preserve information over long distances
- we introduced **attention** as a way to deal with this problem
- can we skip the RNN and **just use attention**?

attention models: recap

- first, compute an "energy" e_i for each state $\boldsymbol{h}_i$



- for the attention weights, we apply the softmax:

$$\alpha_i = \frac{\exp e_i}{\sum_{j=1}^{n} \exp e_j}$$

- finally, the "summary" is computed as a weighted sum

$$s = \sum_{i=1}^{n} \alpha_i \boldsymbol{h}_i$$

the Transformer

- the Transformer (Vaswani et al., 2017) is an architecture that uses attention for information flow
  “Attention is all you need”
- it was originally designed for machine translation and has two parts:
  - an **encoder** that “summarizes” an input sentence
  - a **decoder** (a conditional LM) that generates an output, based on the input
- let’s consider the encoder

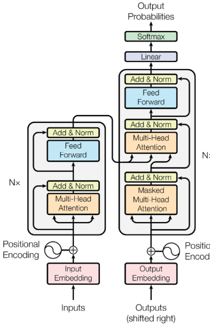


illustration of a Transformer block

Jane lives in Paris . She is a dentist .

illustration of a Transformer block

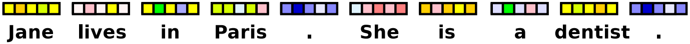


illustration of a Transformer block

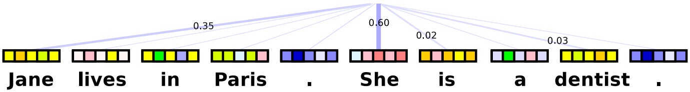


illustration of a Transformer block

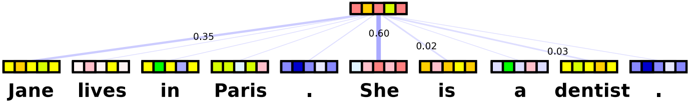


illustration of a Transformer block

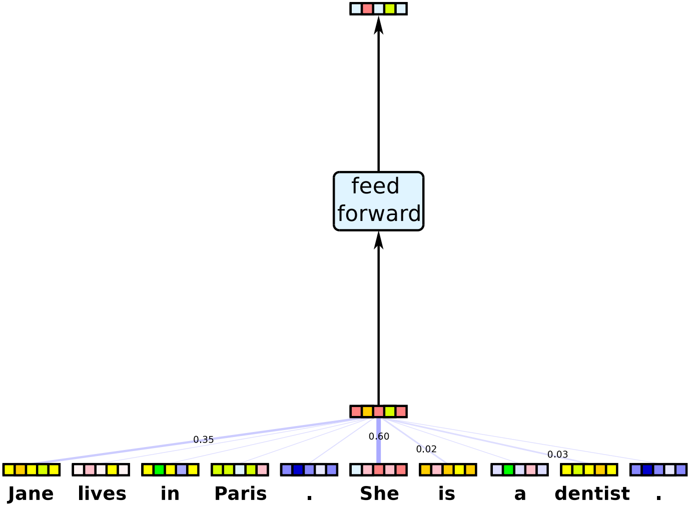
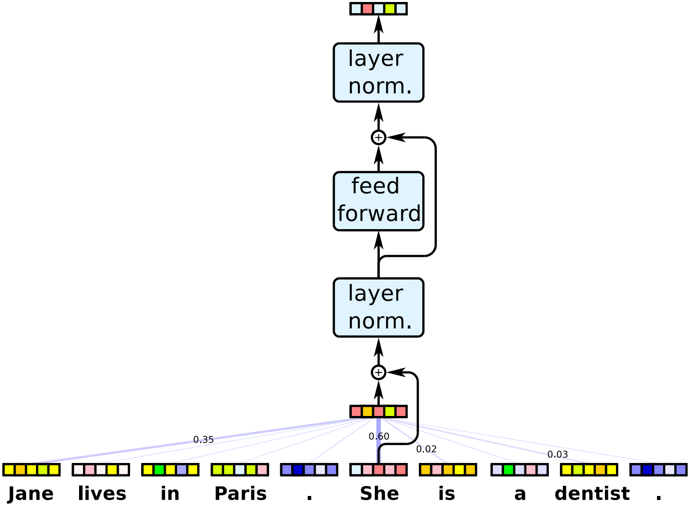


illustration of a Transformer block

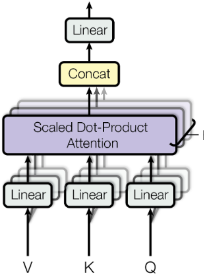


multi-head attention

- in each layer, the Transformer applies several attention models ("heads") in parallel
- intuitively, the heads are "looking" for different types of information
- each attention head computes a **scaled dot product attention**:

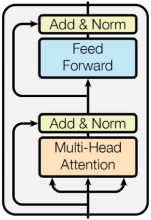
$$e_i = \frac{1}{\sqrt{d}} \boldsymbol{q}_i \cdot \boldsymbol{k}_j$$
$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{e})$$

where $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$ are linear transformations of the input at positions i and j

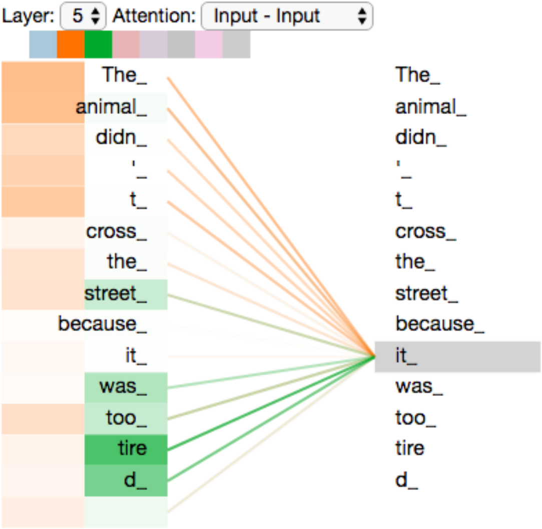


a layer in the Transformer encoder

- after each application of multi-head attention, a 2-layer feedforward model (with ReLU activation) is applied
- residual connections (“shortcuts”) and layer normalization (Ba et al., 2016) added for robustness and to facilitate training
- the Transformer encoder consists of a stack of this type of block



what do the attention heads look at?



- see (Vig, 2019)

pros and cons

+ short path length for information flow
– quadratic complexity

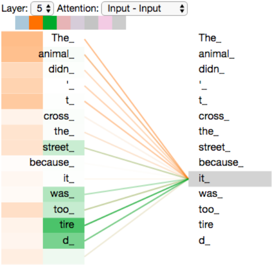
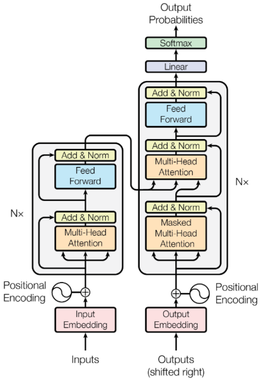


Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

the road ahead

- the full Transformer is an effective model for machine translation
- we'll return to it when we discuss **encoder–decoder** architectures
- for now, let's use it simply as a pre-trained representation



reading

The Illustrated Transformer

references

J. Ba, J. Kiros, and G. Hinton. 2016. Layer normalization. arXiv:1607.06450.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. Attention is all you need. In *NIPS 30*.

J. Vig. 2019. Visualizing attention in transformer-based language representation models. arXiv:1904.02679.