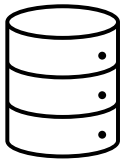
Attention-based Explainability

Explainable and Trustworthy AI

Gabriele Ciravegna

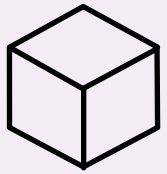
Stages of Explainability

Explainability involves the entire AI development pipeline



Pre-modelling explainability

Before building the model
- Data exploration
- Data selection
- Feature engineering



Explainable modeling

Build inherently interpretable models
- Manage the accuracy and interpretability trade-off

Post-modelling explainability

After model development
- Explaining predictions and behavior of trained models

Contents

- What is the Attention?
- Transformer
- Attention-based Explainability
- Is attention Explanation?

What is the Attention?

Attention in brief

- The Attention is a **neural layer** that compares pieces of inputs
  - Compare the information among different parts of the input
  - **Different** from standard fully connected or convolutional layers!
    - Where each input piece is in principle elaborated alone $y = \sigma(\sum_i w_i x_i + b)$

- It provides some form of **intrinsic explanation**
  - If used among inputs it tells which are the most similar ones
  - If used among inputs and outputs it tell us which inputs have **most influence** on a given output

Attention in Humans

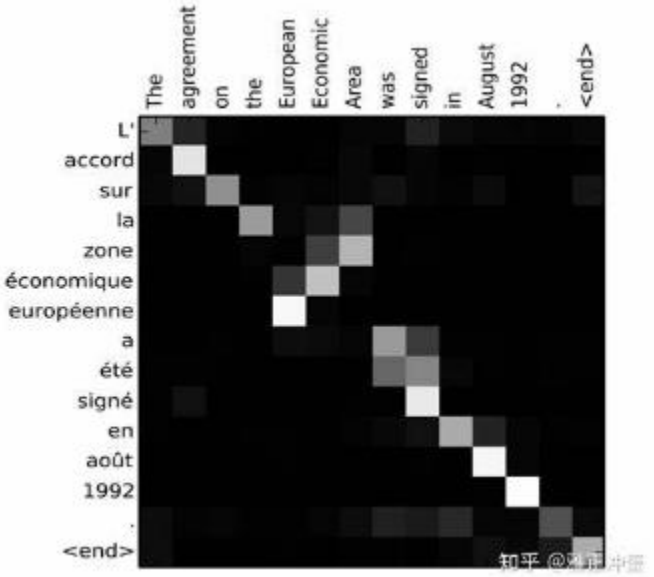
…Not this one

Attention in Humans

- Attention is a **selective** concentration mechanism
  - On the most **relevant** parts of the information
  - Filtering out less important parts
- The principle behind attention is the ability to **assess information**
  - We **learn to assign a weight** to each piece of information
  - **Relative to every other** piece of information

Attention in Machine Learning

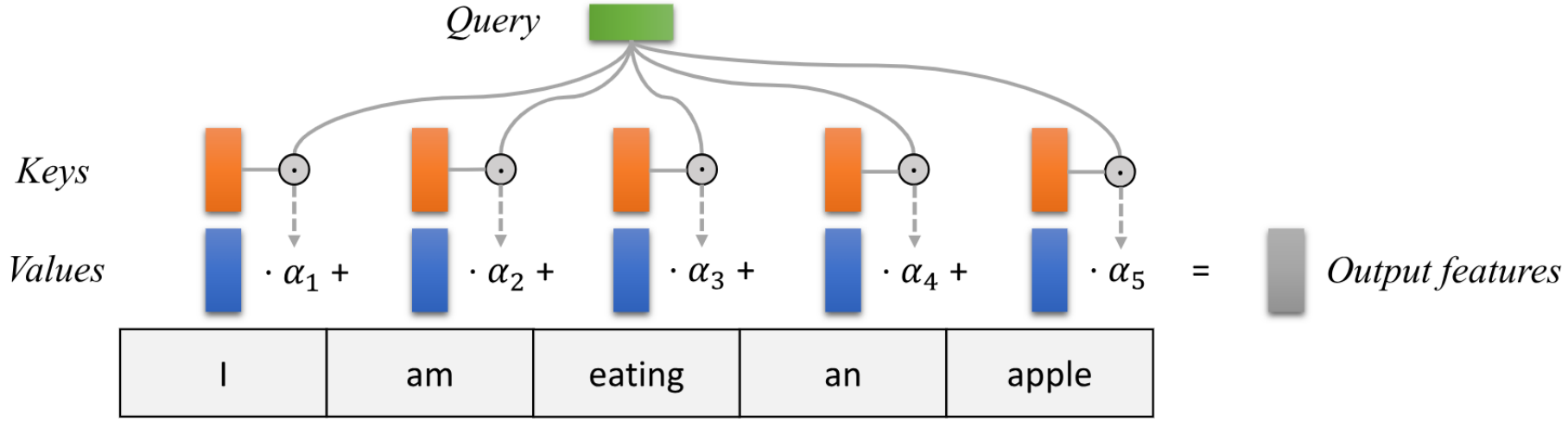
- The attention mechanism is a **learned function**
- It calculates the relative weight of each data unit
  - given the context and the accumulated information
- It creates a continuous, context-dependent representation for the input
  - Its outcome is a continuous function
  - It can be derived with respect to attention parameters
- Attention map of an English sentence in relation to its French translation
  - Weight the tokens in the first sequence w.r.t. all the tokens in the second sequence



Bahdanau et al. “Neural machine translation by jointly learning to align and translate” Neurips 2015

How do you compute the attention?

- Weighted average of elements with **dynamically computed** weights:
  - **Values**: The numerical representation of the elements of the input sequence
  - **Queries**: The correlations among the elements
  - **Keys**: The relevant features of each input elements w.r.t. the task at hand



How do you compute the attention? (2)

$$\alpha_i = \frac{\exp(f_{attn}\left(\text{key}_i, \text{query}\right))}{\sum_j \exp\left(f_{attn}\left(\text{key}_j, \text{query}\right)\right)}, \quad \text{out} = \sum_i \alpha_i \cdot \text{value}_i$$

- f_{attn} Activation function computing attention scores for (key, query)
  - E.g. Dot-product QK^T or sum $Q + K$
- α_i Attention scores that factorize the values

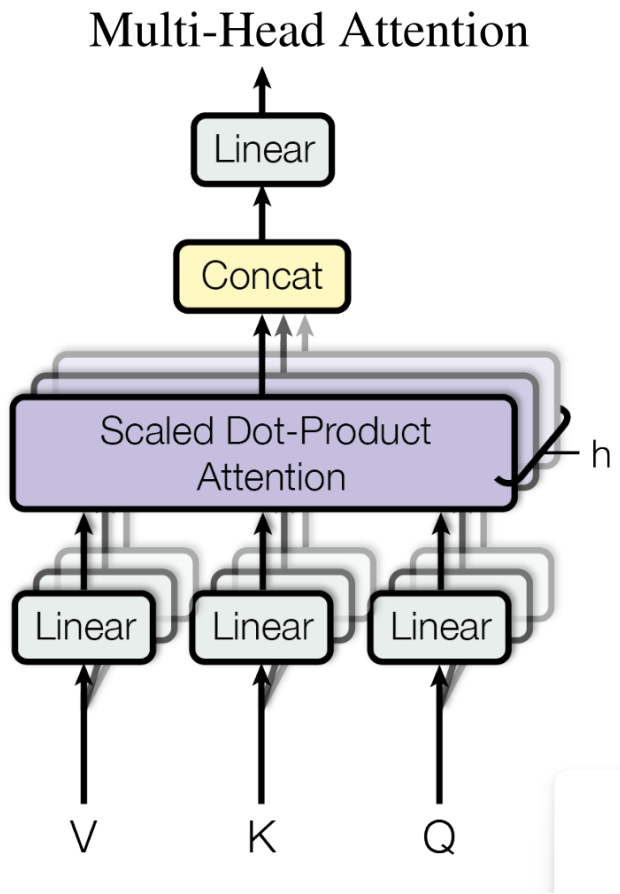
  →We assign **higher weights** to those value vectors whose corresponding key is most similar to the query
- Softmax: The attention scores are weighted over all attention scores

Multi-head attentions

- Often there are multiple different aspects a sequence element wants to attend to
  - A single attention is not a good option for it
- Multiple heads of attention
  - Multiple different query-key-value triplets on the same features
- The features are the current inputs:
  - $Q, K, V = X$ $\quad X \in \mathbb{R}^{B,T,d}$

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

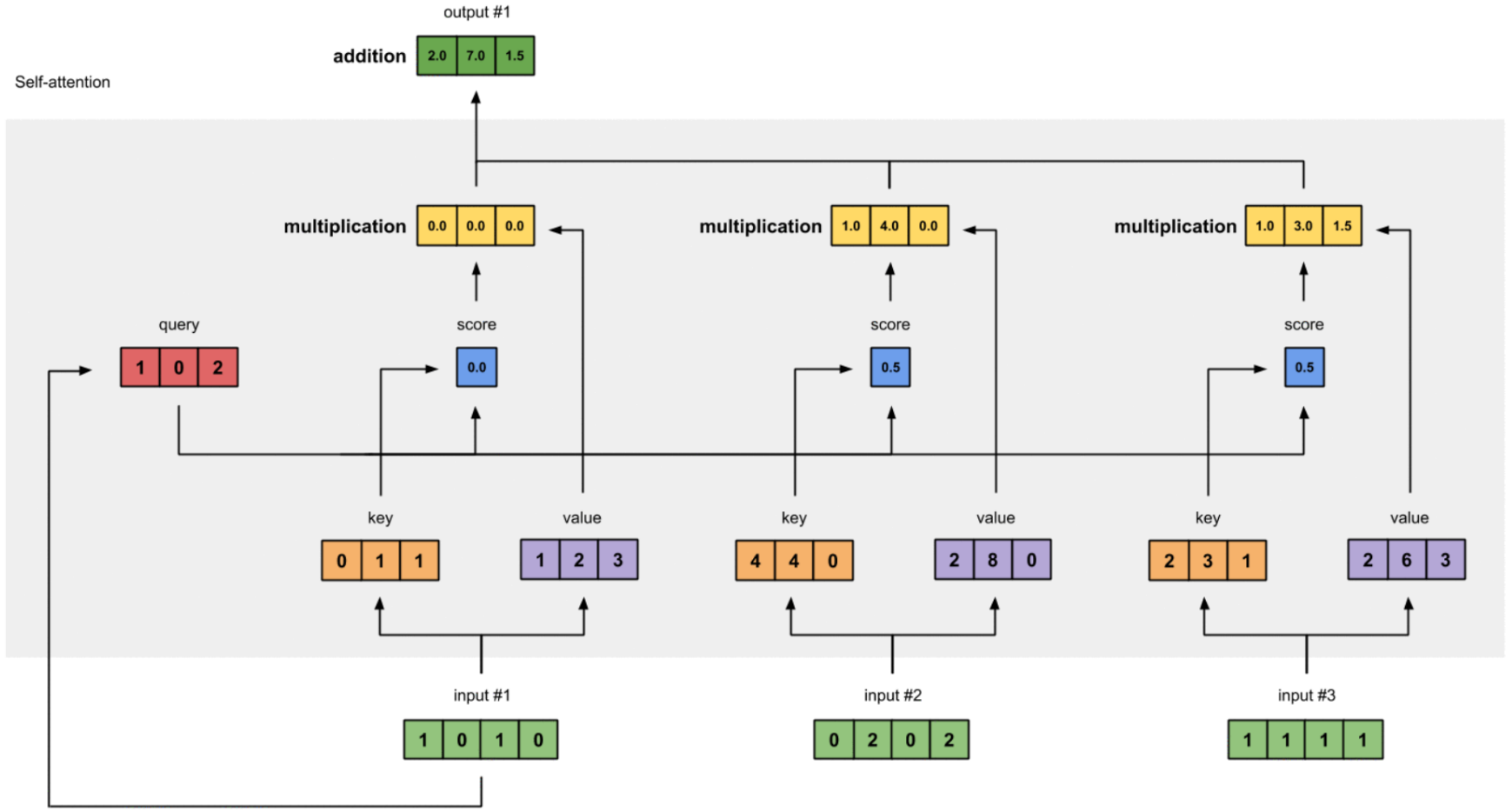
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Attention Workflow

Illustrated: Self-Attention (Raimi Karim, 2019)
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Attention Workflow



Attention is permutation equivariant

- **Permutation equivariant** property:
  - Switch two input elements in the sequence
  - The output sequence is the same **besides** the elements switched

- The attention is looking at the input:
  - NOT as a sequence
  - BUT as a **set of elements**

- This property makes the attention so powerful and widely applicable!
  - If position matters: **Positional Encoding**
  - It encodes the position it in the input features by means of position dependent patterns

Attention in Transformers

What is Attention used for?

- Historically:
  - Introduced to improve Sequence to Sequence task in Recurrent Neural Networks (RNNs)

- Nowadays:
  - Sequence classification
  - Sequence to Sequence (Seq2Seq)
  - Text-to-Image
  - … Everything!

--> It is at the base of the 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

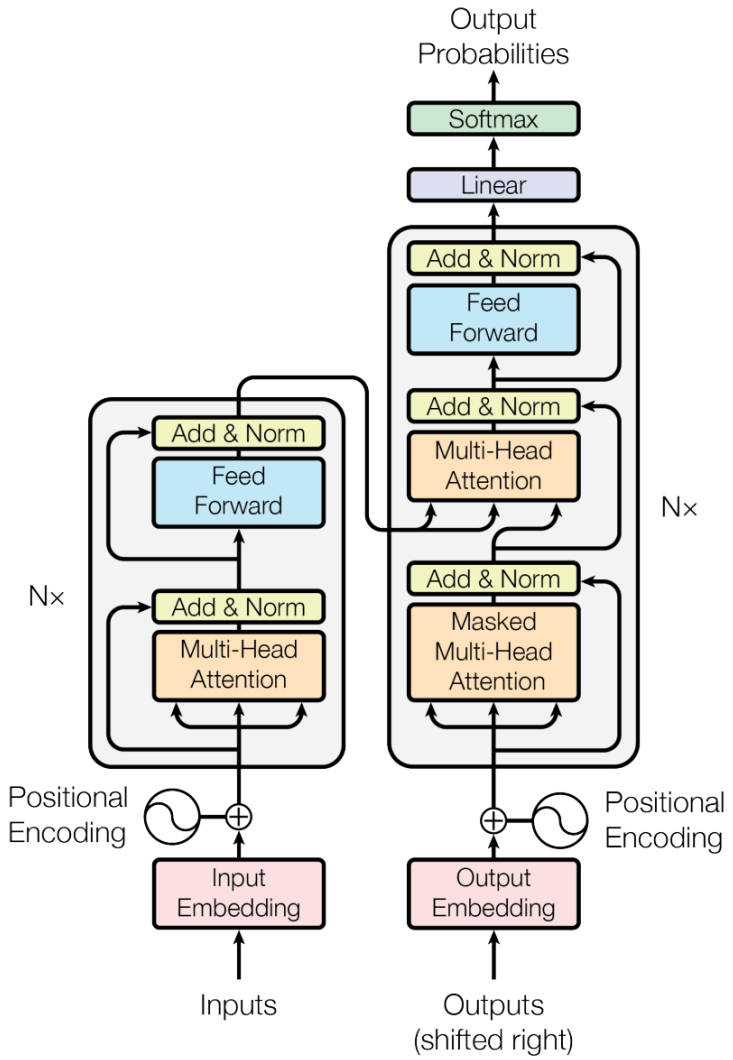
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

The Transformer Model [1]

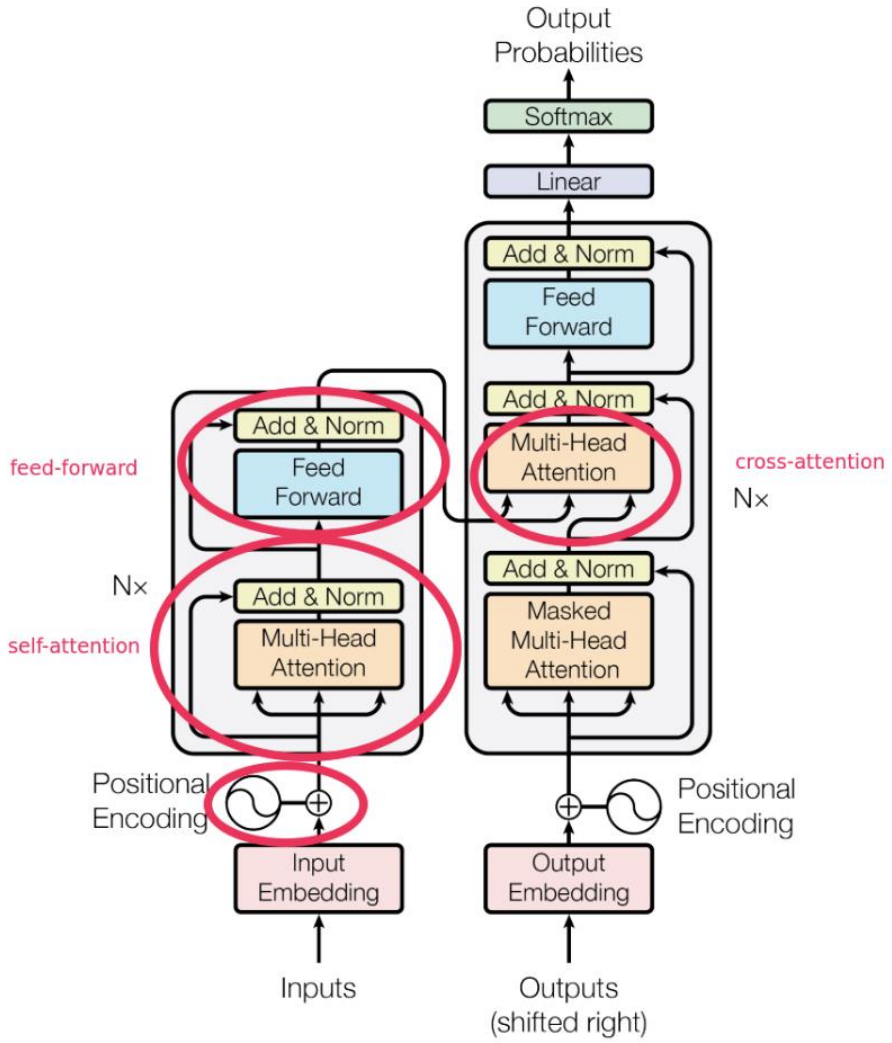
- Composed of two parts:
  - Encoder: processing the input
  - Decoder: creating an output
- Each part of several attention blocks
- Followed by **per-token** fully connected layers
  - The fully connected layers are applied on each position singularly
- Residual connections
  - To avoid vanishing gradients
  - Retain information about the input sequence
- Positional Encodings



[1] Vaswani, Ashish, et al. "Attention is all you need." Neurips 2017

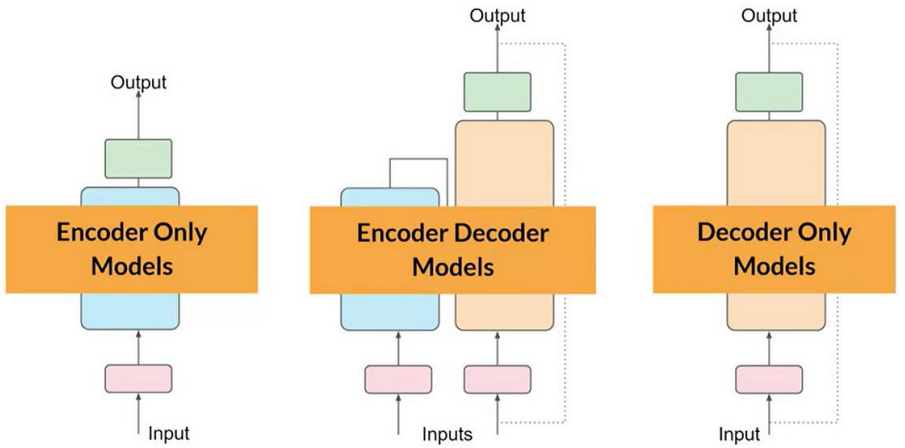
Different types of Attentions

- Attention mechanisms differ mainly for the queries, keys and values used
- Self-Attention:
  - Attention applied **within** Encoders, Decoders
  - Each input element provides a key, value, and query
- Cross-Attention:
  - Attention applied for language modelling **between** the Encoder and the Decoder
  - The **queries** come from the sequence elements generated so far
  - The **keys** and the **values** come from the output of the encoder



Different types of architectures!

- Encoder-only
  - Self-attention only layers
  - Produce good embeddings for classification, clustering, search
  - Examples: BERT
- Encoder-decoder
  - Self + cross-attention (original architecture)
  - Strong for natural language understanding tasks: translation, question-answering, summarization
  - Examples: BART, T5
- Decoder-only
  - Masked self-attention + Autoregressive decoding
  - Strong at text generation tasks: prompting, chatting
  - Examples: GPT-1,2,3,4, Llama, Mistral



https://pub.aimind.so/unraveling-the-power-of-language-models-understanding-llms-and-transformer-variants-71bfc42e0b21

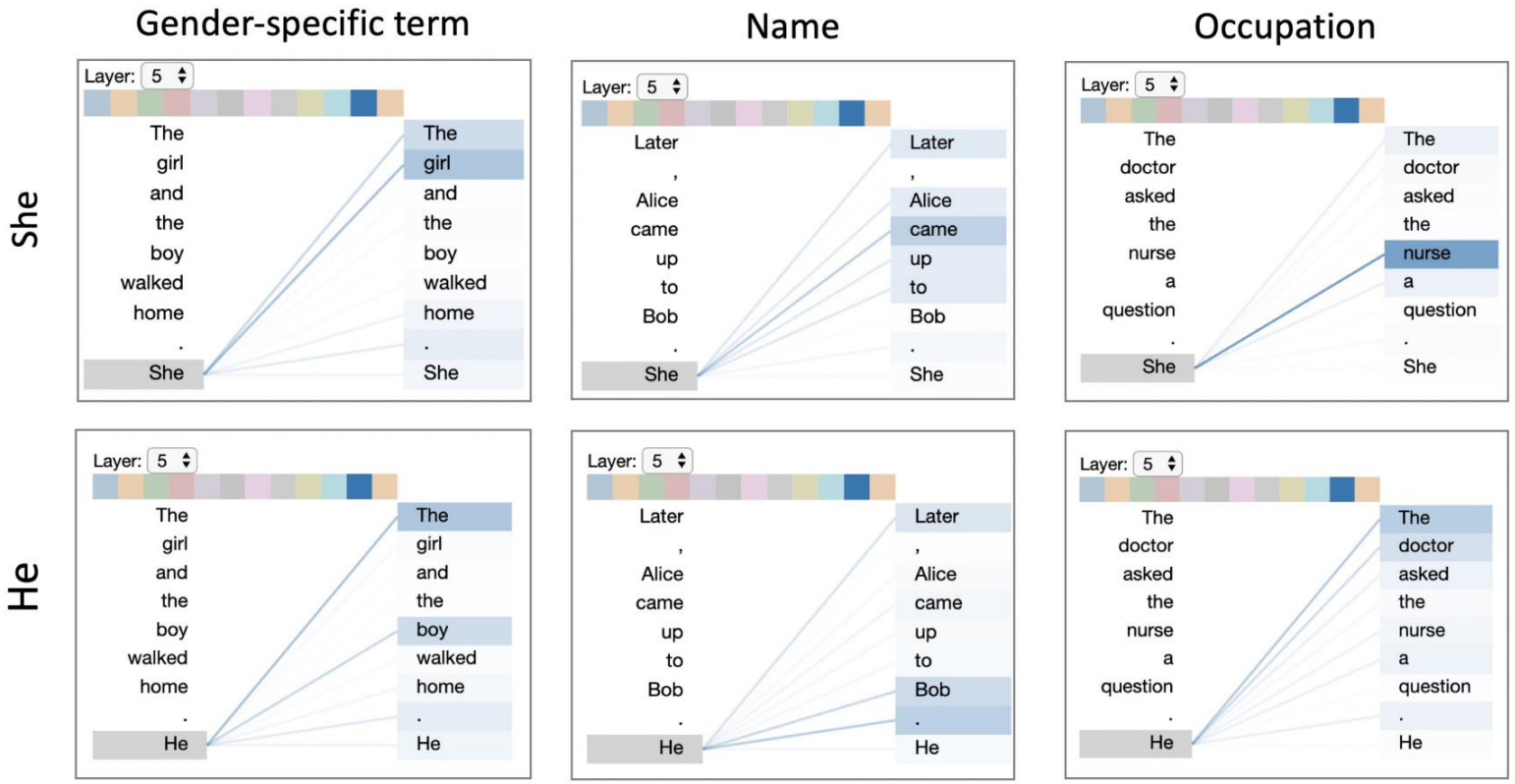
Attention-based Explainability

Visualizing Attention in NLP models

https://github.com/jessevig/bertviz

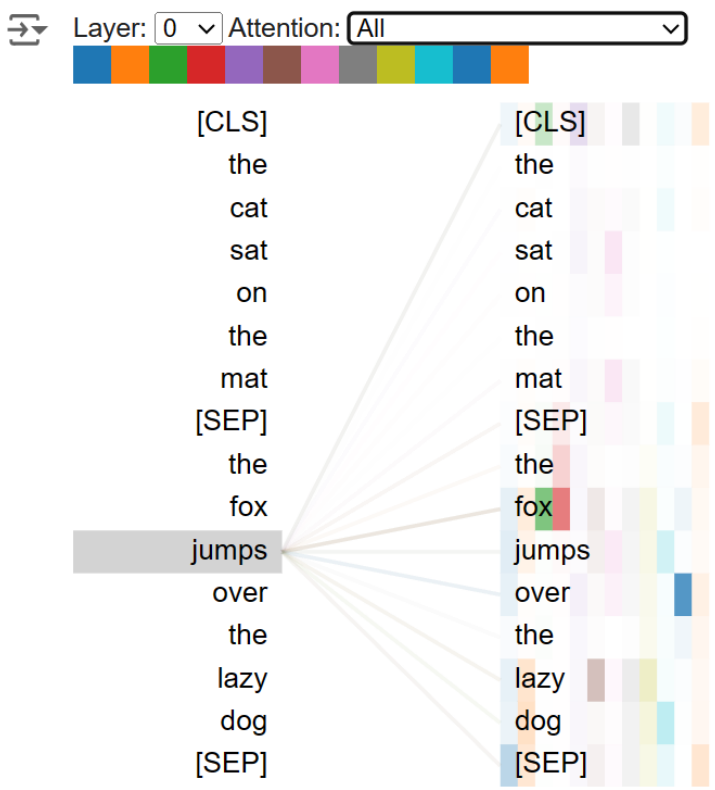
Attention patterns in GPT-2 generated text

- Attention allows to see **which terms** the transformer is linking
- It can be used to reveal **biases**
- Only backwards attentions in decoding-only transfomers
  - The forward ones are masked to avoid «seeing in the future»



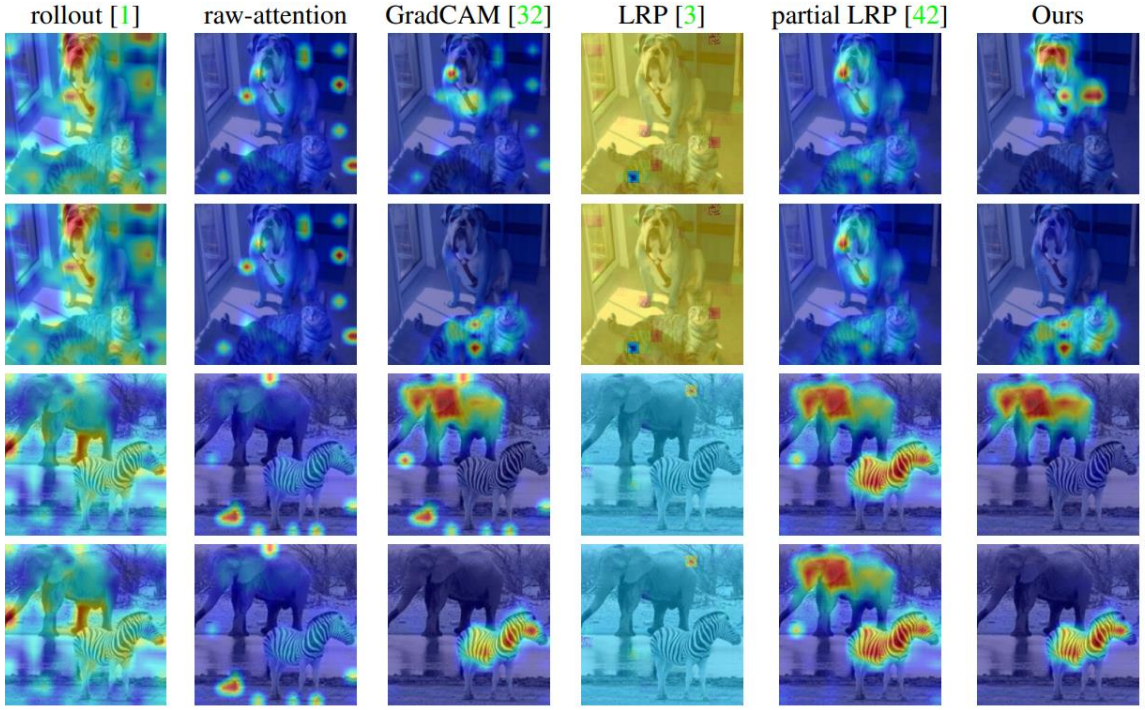
Self-Attention Explanations

- **Challenge:** multi-layer – multi-head attention
  - Different attention patterns for each layer and head
- Multiple Heads:
  - Aggregating by summing, averaging or taking max values
- Multiple layers:
  - Take a single layer only (e.g., the first)
  - Aggregate over multiple layers
  - Combine attention scores with gradient-information
- BERT working on next sentence prediction classification
  - [CLS] and [SEP] tokens used for classifying and separating sentences



Aggregating over Layers

- Baseline - Standard Attention:
  - $S = \bar{A}^1, \qquad \bar{A}^1 = \frac{1}{h}\sum_i^h A_i^1$
- Rollout [1]:
  - $S = \bar{A}^1 \times \bar{A}^2 \times \ \ldots \times \bar{A}^N$
  - $\bar{A}^{j-1} \times \bar{A}^j$ is a standard matrix multiplication
- Transformer Attention [2,3]:
  - $S = \bar{A}^1 \times \bar{A}^2 \times \ \ldots \times \bar{A}^N$
  - $\bar{A}^j = \frac{1}{h}\sum_i^h (\nabla A_i^j \ R^A)^+ \ R^{A^j}$:LRP Relevance for A^j
  - $\bar{A}^j = \frac{1}{h}\sum_i^h (\nabla A_i^1 \ A)^+ \quad A$: attention scores



[1]: Abnar, Samira, et al. "Quantifying Attention Flow in Transformers." ACL 2020.

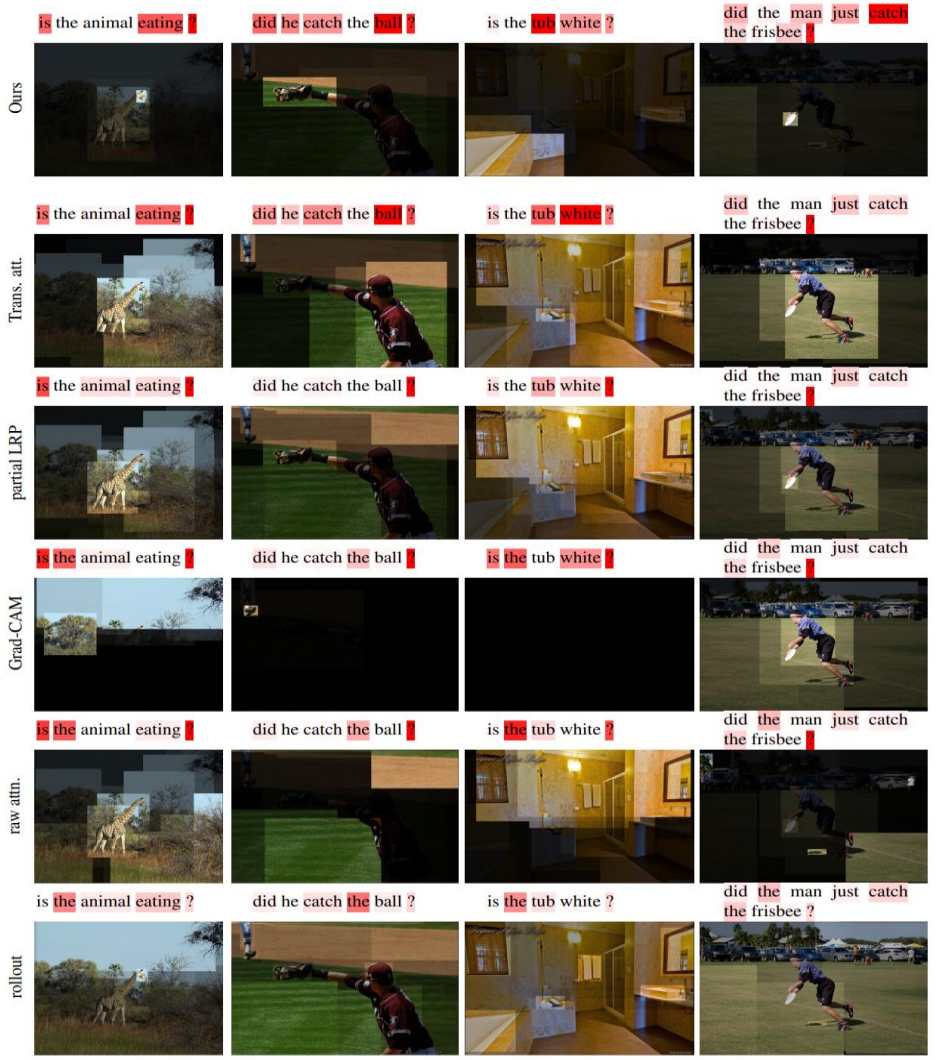
[2, 3]: Chefer et al. “Transformer Interpretability Beyond Attention Visualization”, CVPR 2021, “Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers”, ICCV 2021

Co- or Cross-Attention Explanation

- Attention scores represents the importance of:
  - The processed input (from the encoder)
  - For the generated output (cross-attention)
  - For the other modality (co-attention)

- Easy to understand for model employing a single cross attention
  - It becomes more difficult in transformers with many heads and many layers

  →Need to aggregate possibly also with self-attention layers



Is Attention Explanation?

«Attention is not Explanation» [1,2]

- Ongoing extensive research evaluating attention

- Cons:
  - Raw attention often does not identify the most important features for prediction
  - It provides inferior explanations compared to some alternatives
  - Raw attention contains redundant information

- Pros:
  - All explanation methods consistency is low (not only attention)
  - Attention heads in BERT have been shown to encode syntax effectively

[1] Jain S, Wallace BC. “Attention is not explanation”, 2019

[2] Wiegreffe S, Pinter Y. “Attention is not not explanation”, 2019.

How to make attention more effective?

- Better aggregation over layers (see previous slides) and with other techniques (gradients, revelance)
- Regularizing learning objectives
  - Avoiding learning redundant information
- Avoiding biased learning
  - Avoid “data leakage”
- Incorporating human rationales
  - Human-in the-loop learning: intervene on non-important attention scores

Thank you for your attention!

Monday 20/05 - Gabriele Sarti' Seminar

When & Where: May 20, 9:00 - Room 12I

Title: Interpreting Context Usage in Generative Language Models with Inseq and PECoRe

Abstract: This talk discusses the challenges and opportunities in conducting interpretability analyses of generative language models. We begin by presenting Inseq, an open-source toolkit for advanced feature attribution analyses of language models. The usage of Inseq is illustrated through examples of state-of-the-art approaches contrastive attribution, input dependence and locating factual knowledge in intermediate model representations. Then, we introduce Plausibility Evaluation of Context Reliance (PECoRe), an end-to-end interpretability framework using model internals to detect context-dependent spans in model generations and trace their prediction back to salient tokens in the available context. The usage of PECoRe is showcased on various generative tasks, including machine translation, story generation and retrieval-augmented question answering.