All you need is Attention

Evolution of Machine Translation Models

- Natural Language Processing - Machine Translation Problem
- Seq2seq (Ilya Sutskever et al., 2014) & Attention Mechanism (Bahdanau et al., 2014)
- Transformer model (Vaswani et al., 2017): Significant training speed improvement (over RNN, CNN), clear performance improvement (BLEU score machine translation)

Andrew Dahlstrom
M.Sc. Candidate
Data Science & AI
San Francisco State University
5/12/2024

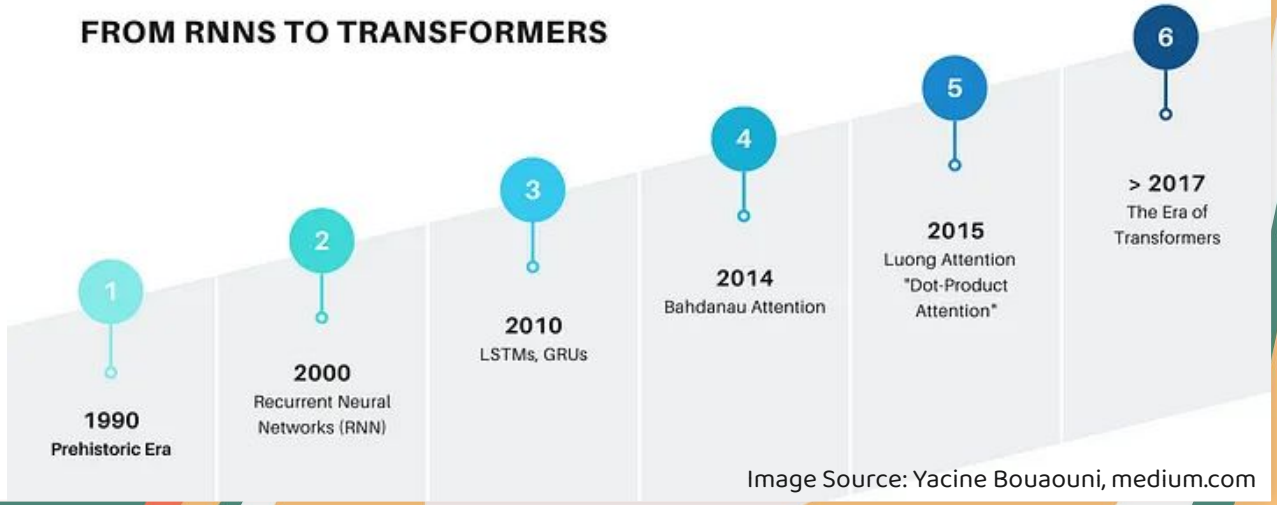
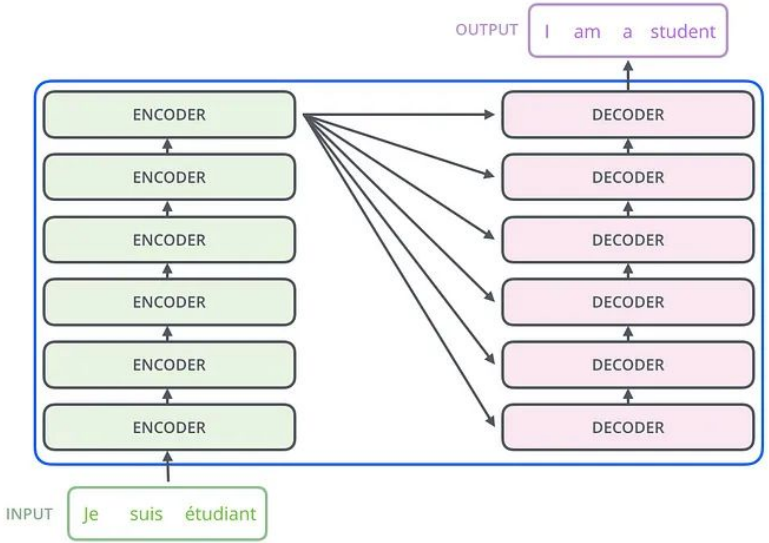


Image Source: Yacine Bouaouni, medium.com

Encoder - Decoder Architecture

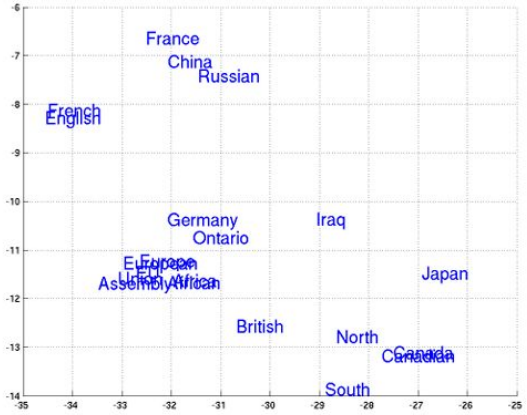
- Problem: DNNs struggled with variable length inputs, RNNs struggled with long-range dependencies
- Sutskever et al. 2014, proposed an encoder-decoder architecture using deep (4) Long Short-Term Memory layers
- Deep LSTM handled long-range dependency problem (gates control flow of info and layers abstract info)
- LSTMs also handles variable length input, compressed by encoder into a fixed-size context vector
- Decoder uses context vector to generate variable length output



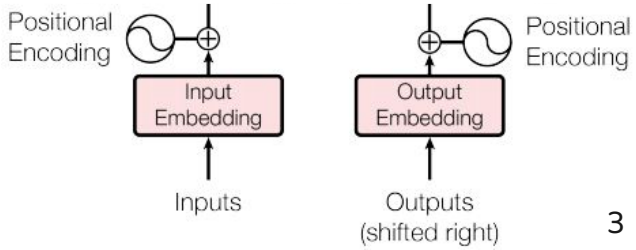
Encoder-Decoder Flow. Source: http://jalammar.github.io/illustrated-transformer/

Word Embeddings & Positional Encoding

- Word embeddings transform sparse one-hot encoded representations of words into dense vectors in high-dimensional continuous space (pre-trained embeddings using Word2Vec and GloVe)
- Cho et al. 2014, proposed method within an encoder-decoder framework to learn embeddings from scratch that capture both semantic and syntactic info directly from training (not pre-trained)
- Vaswani et al. 2017, proposed a positional encoding approach that allowed for a parallel (instead of sequential) processing while still retaining info about word order
- This approach uses sinusoidal functions of different frequencies to encode the positions, which are then element-wise added to the embedding vectors



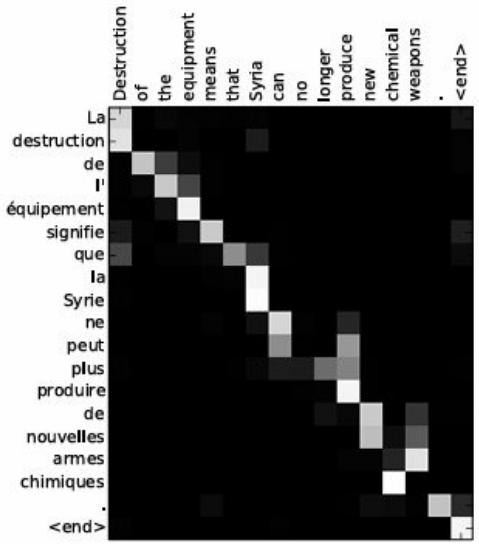
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Self-Attention Mechanism

- Bahdanau et al. 2015 proposed an attention mechanism which calculates weights focusing on different parts of the input for each step of the output instead of using a single fixed-length context vector to encode entire input
- Vaswani et al. later proposed instead of computing attention based on sequential hidden states, self-attention recalculates attention for every element in the input sequence simultaneously, allow for parallelization
- Self-Attention is represented in the formula below. Query, Key and Value are the input embedding transformed by three distinct linear transformations
- Product of Q.K' is attention score matrix defining the strength of the inter-sequence word dependencies

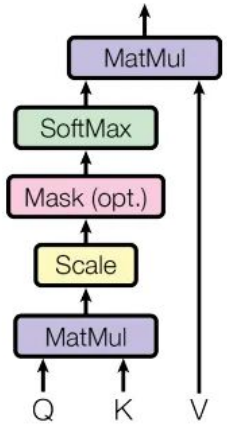
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Single-Head Attention

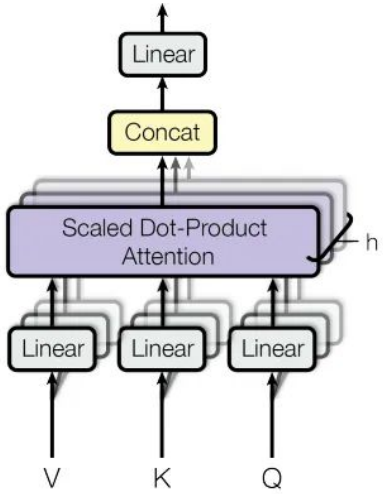
- Luong et al. 2015, proposed an attention mechanism with a global part attending to all input words and local part focusing on a subset of input words, sequential process
- Vaswani et al. parallelizable process uses a d_model = 512 dimension word embeddings
- Each word in the input sequence is represented as Query, Key and Value vectors computed through 3 distinct linear transformations using trainable weights Wq, Wk and Wv
- For a given input sequence of length n,
  X is the [n, 512] matrix of word embeddings + their positional encoding
- Q = X * Wq
  K = X * Wk
  V = X * Wv
- Q.K' (attention scores) are normalized then dot product with Value matrix
  to get a weighted some of values

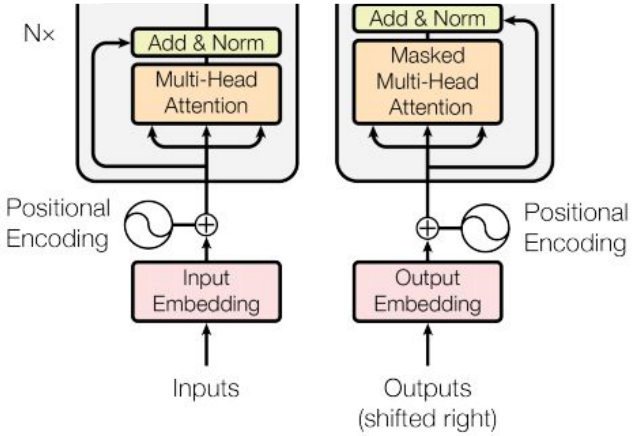
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

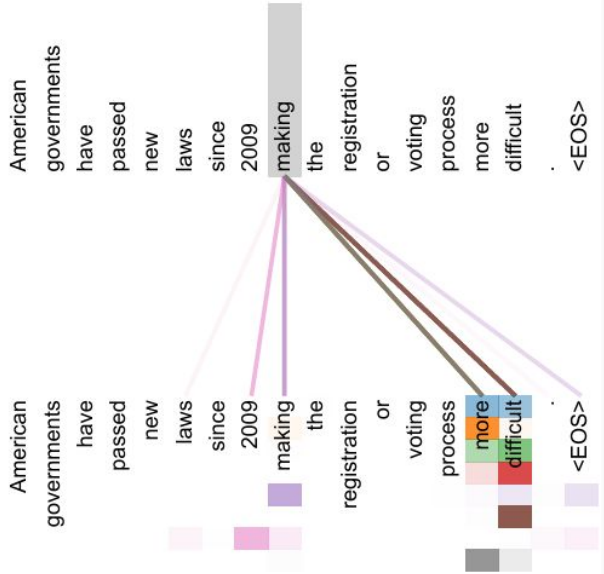


Multi-Head Attention

- Single head Attention(Q,K,V) weighs the relevance of different words in a sequence with respect to each other and the query
- Multi head attention runs multiple sets of attention computations in parallel focusing on different patterns of the input sequence
- Instead of three matrices (Q, K, V), for h = # heads, it becomes (Q1, K1, V1) ... (Qk, Kk, Vk) sets of matrices

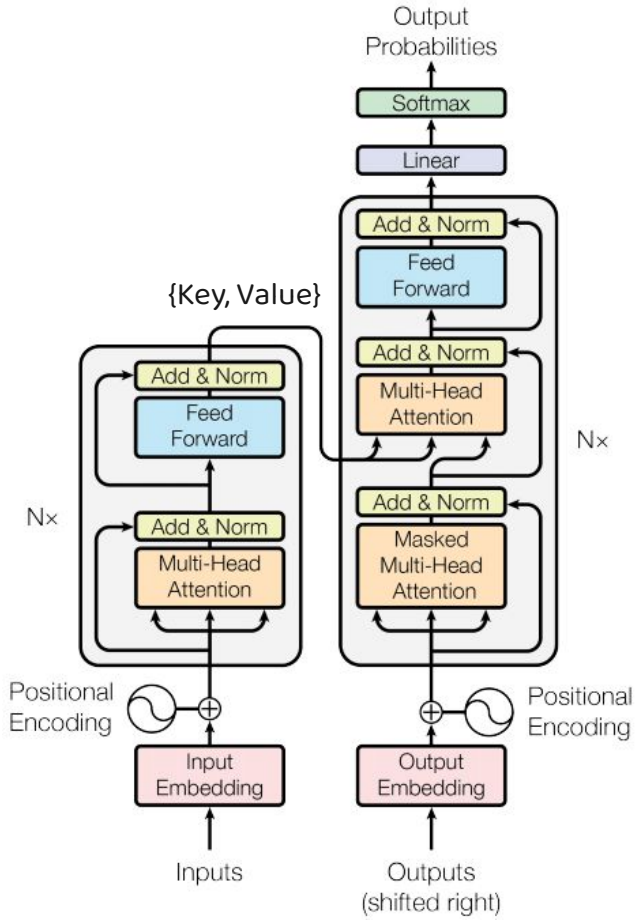






Encoder-Decoder in Transformers

- Next layer position-wise feed forward network, linear transformation with a ReLU activation function
- Encoder output becomes decoder cross-attention mechanism (attend to the encoder's contextual representations)
- Encoder output provides {Key, Value} dictionary for Decoder previous layer Query
- Every position in decoder attends to all positions in the input sequence, similar concept to Bahdanau et al. sequence-to-sequence
- Decoder masked attention layer prevents decoder from attending to future words during training (prevent copying vs predicting)



Attention is all you need

- Transformer model output is transformed representations of the input sequence which can be tailored to specific tasks: classification, translation, or generation
- Xu et al. 2015, applied attention to image processing, focusing on different parts of image to generate text description
- Computer Vision: image captioning, visual question answering, and object detection
- Future: Autonomous Vehicles, Robotics and Healthcare

Ashish Vaswani et al.

Xu et al. 2015

MIT ViT Autonomous Driving