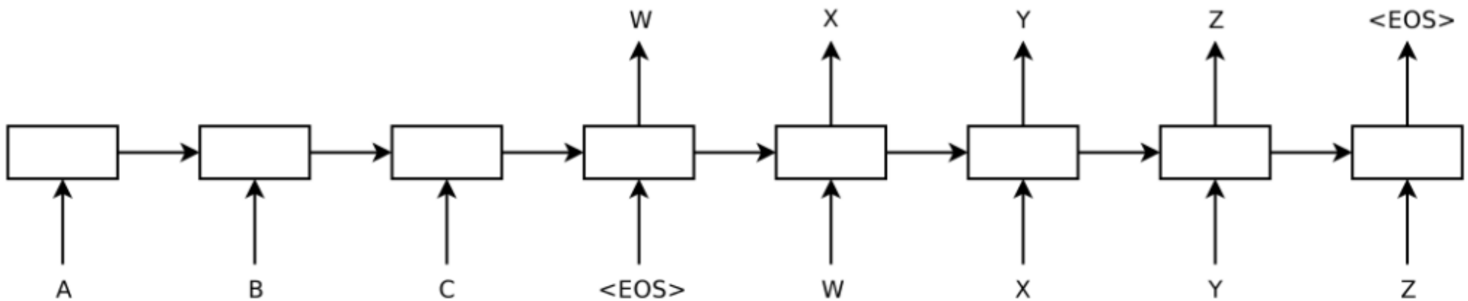
Attention model

Hung-Hsuan Chen

Overview

- It is still challenging for an RNN (and its variants) to generate long sequences when the decoder can only access the final state from the encoder
- Attention model improves the performance on long sequences

Encoder-decoder example (machine translation)

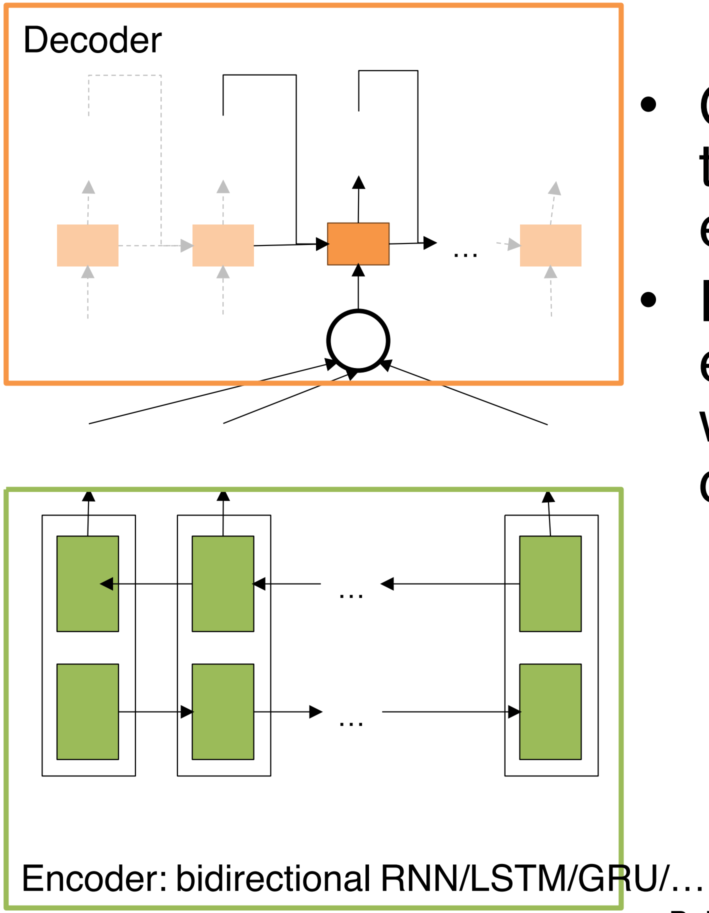


- All input information are stored in one hidden vector, which is used by decoder to generate output sequence
- If input is really long, one hidden vector could be problematic to store the initial inputs
- Let's allow the decoder refer to input sequence

Attention-based machine translation

- Each output word comes from one or multiple input words
- Build a model that learns to attend to only the relevant input words when producing a output word

Encoder-decoder model + attention



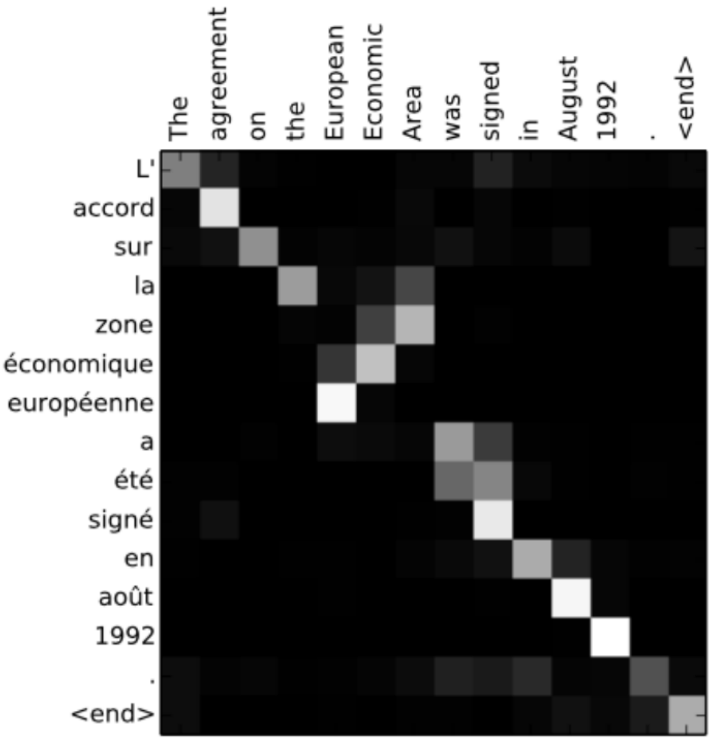
- Original encoder-decoder model: the final hidden state of the encoder is sent to the decoder
- Encoder-decoder + attention: at each time step, compute the weighted sum of each hidden state of the encoder
  - The weights s are computed by various ways
  - If length of input is , length of output is , in total we need to compute weights

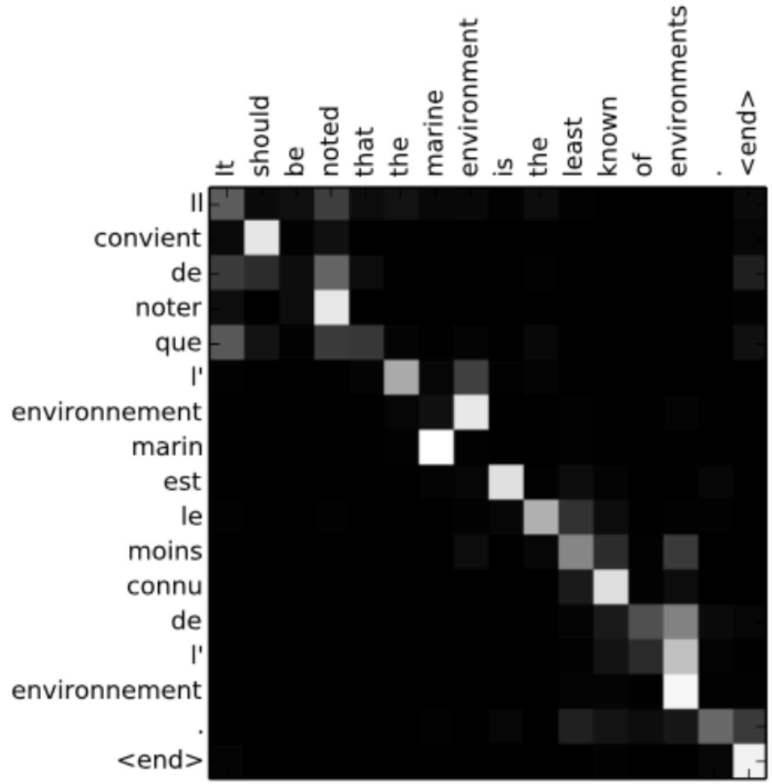
Bahdanau et al., Neural machine translation by jointly learning to align and translate. ICLR, 2015 (15000+ citations ~2020)

Various ways to compute the weights

- where could be
  - Inner product between hidden state of decoder and output of encoder
    - E.g.,
  - Linear transformation and inner product
    - E.g., , is learned jointly
  - A small NN
    - E.g., , and are learned jointly
- Attention function depends on the annotation vector, rather than the position in the sentence

Visualization of attention map for machine translation



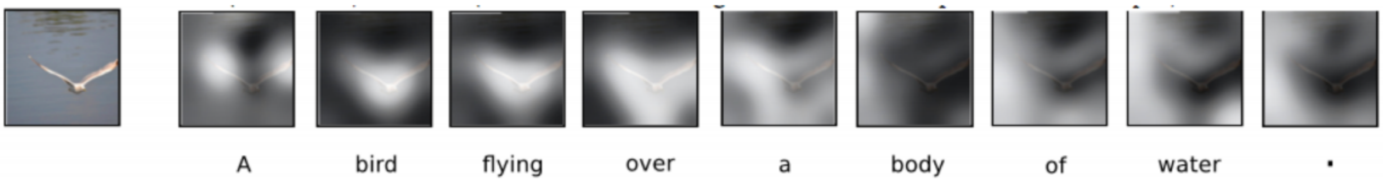


Caption generation by attention model

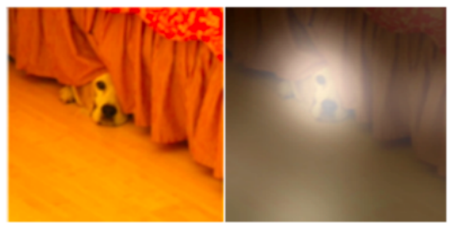
- Task: take an image as input, generate the caption as output
- Encoder: CovNet
- Decoder: attention-based RNN

Xu et al. Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention. ICML, 2015. (6000+ citations ~2020)

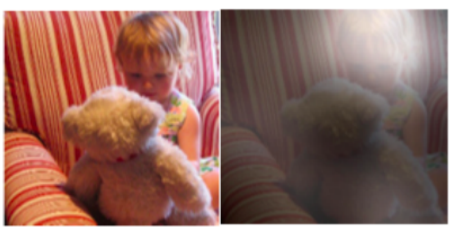
Visualization of attention map for caption generation




A woman is throwing a frisbee in a park.


A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.


A little girl sitting on a bed with a teddy bear.


A group of people sitting on a boat in the water.


A giraffe standing in a forest with trees in the background.

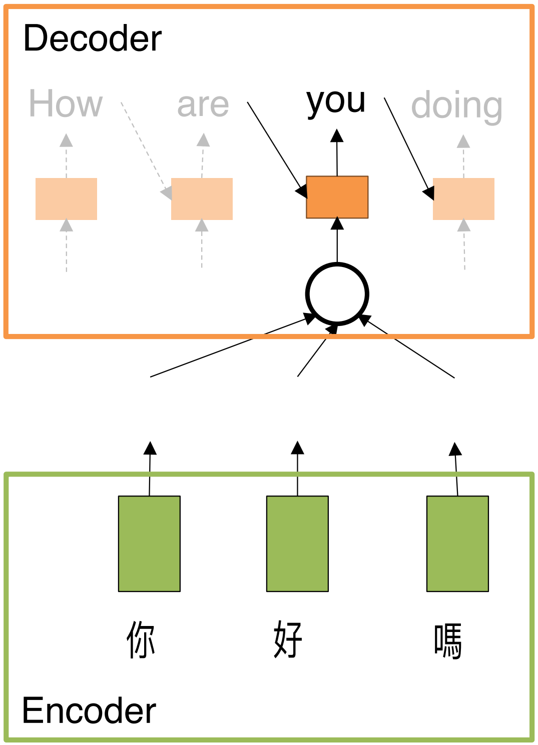
Main computational cost of RNN + Attention

- It is hard to parallelize RNN

Transformer model

- "Attention is all you need" by Google researchers
  - NIPS 2017
  - 16000+ citations ~2020
- Used in machine translation task
- No recurrent layers and no convolution layers
- Use attention to learn the importance of long term and short term vocabularies

Proposed model v1

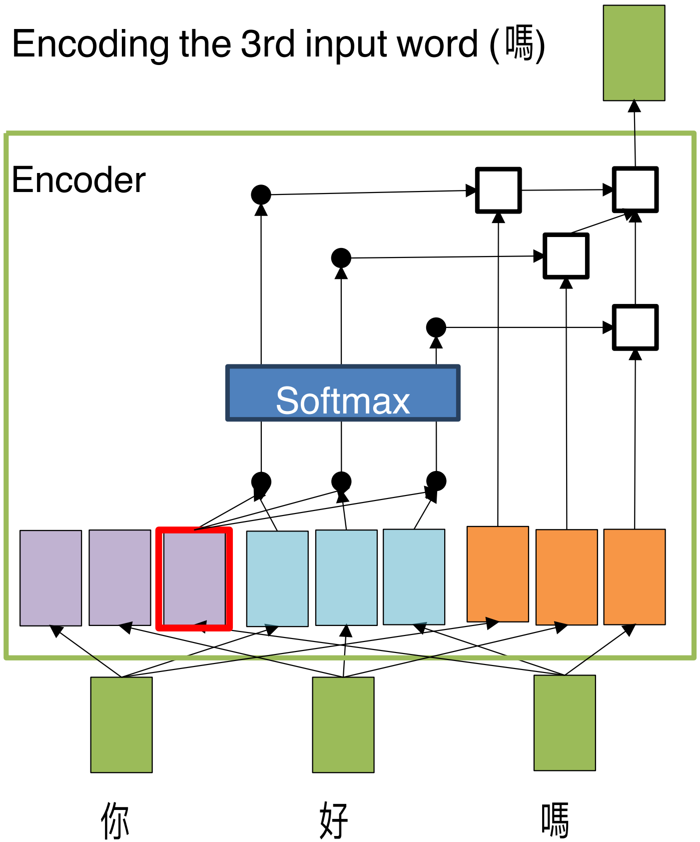


- Remove all the connections between hidden states
- The decoder takes two types of inputs to generate one word at a time
  1. A weighted sum of the encoder outputs
     - This resembles a 1D CNN
  2. The outputted word of the decoder in the last time step
- Problems
  3. The same token should be encoded differently based on context
  4. Decoder should consider not only the last outputted word but also the earlier outputted words
  5. How to decide s?
  6. Sequential information in the source language is missing

Dealing with problem 1 and problem 3

- P1: The same token should be encoded differently based on context
- Possible solution 1: use RNN as the encoder, so encoding a word depends on both the word and the hidden state
  - However, we don’t want to use RNN, because RNN is hard to scale
- Possible solution 2: apply “attention”, so all words in the input sentence are considered to generate each encoded token
  - This also solves P3: how to decide s?

Self-attention at encoder



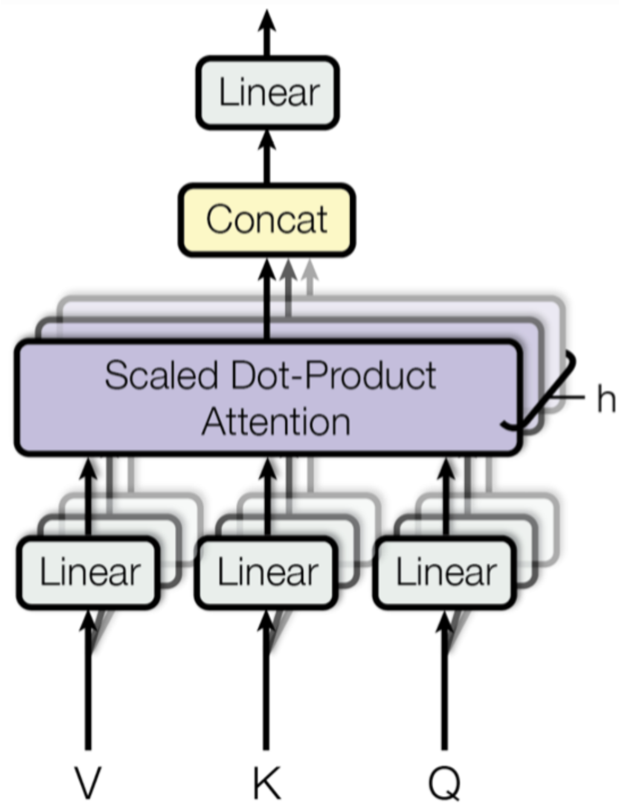
1. Compute queries, keys and values for every input token

2. Compute (unnormalized) attention score , is the dimension of and
   – Scaling factor scales down the similarity scores to avoid saturating the softmax function, which would lead to tiny gradients

3. Softmax each

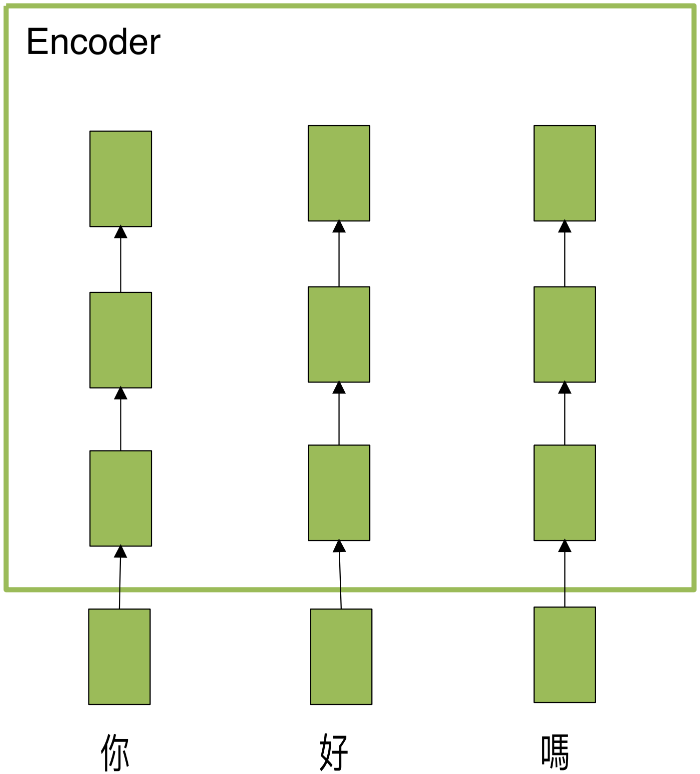
4. Compute weighted sum of s

Multi-head self-attention at encoder



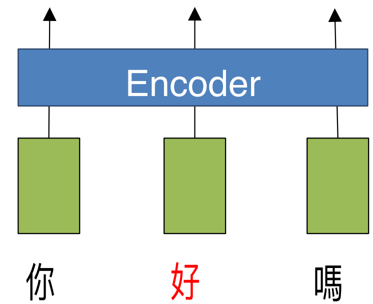
- Multi-head attention computes attention multiple () times in parallel
- The independent attention outputs are concatenated and linearly transformed into the expected dimensions

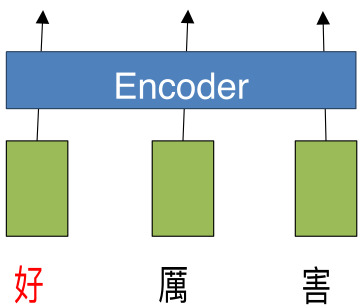
Multi-layer self-attention



- Each input token is "encoded" as one vector
- We may stack many layers

Output of self-attention encoder



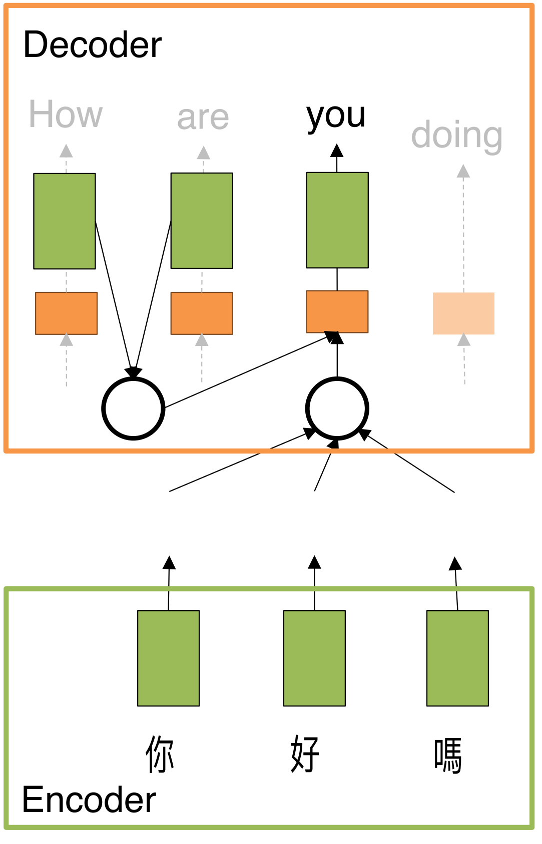


- The same token (e.g., “好”) may be encoded differently based on the contexts
- 你好嗎? fine
- 好厲害? very

Dealing with problem 2

- P2: Decoder should consider not only the last outputted word but also the earlier outputted words
- Masked self-attention
  - Apply attention to the already outputted words
    - At inference time, decoder can only access to the words it already output, not future words, so future words are “masked” during training
  - The machine generates the new token partially based on the weighted sum of the outputted tokens
  - To mask out future key/value pairs, simply add a very large negative value to the corresponding similarity scores before computing the softmax

Encoder-decoder model with attention (no RNN)

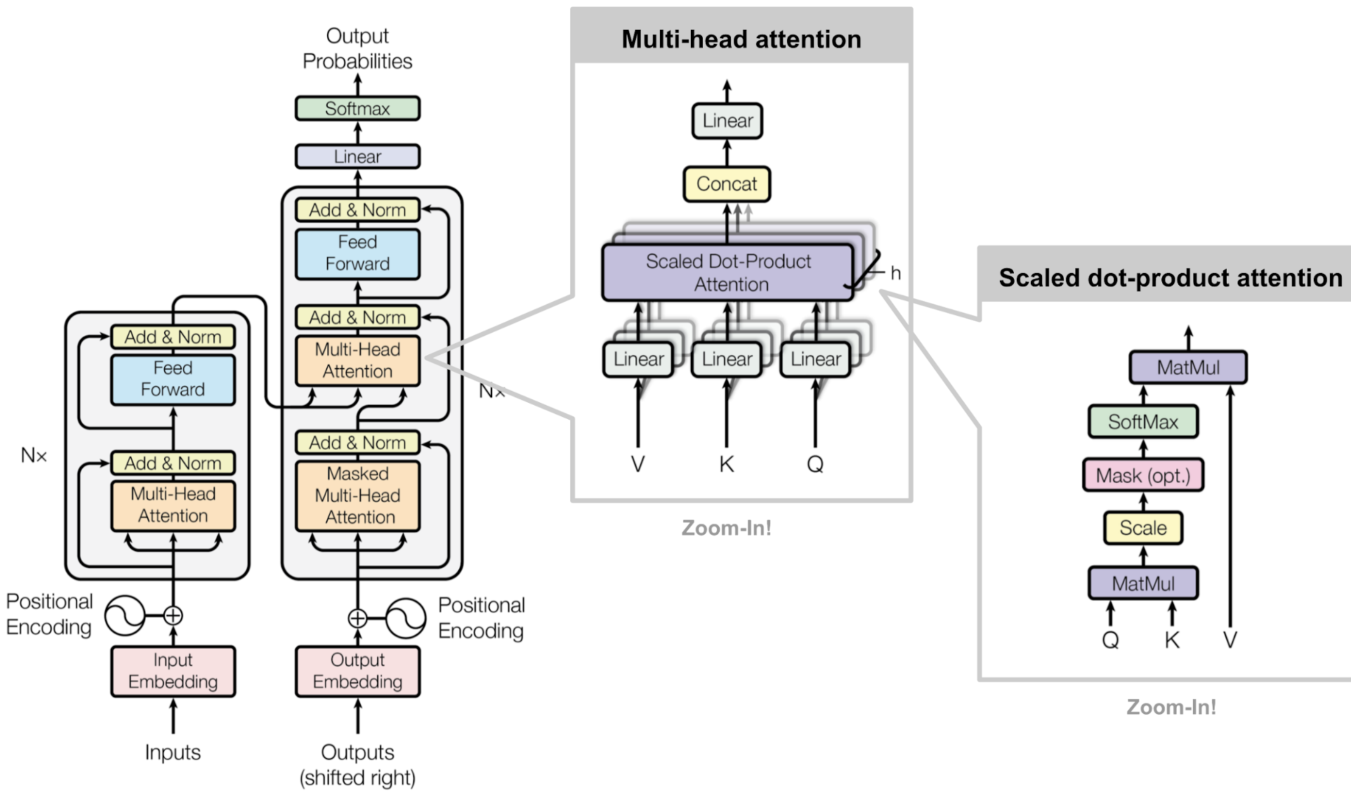


- The decoder takes two types of inputs to generate one word at a time
  1. A weighted sum of the encoder outputs
  2. A weighted sum of the already-generated decoder outputs so far

Solving problem 4

- P4: Sequential information is missing
- Positional embedding (PE)
  - The th positional embedding is added to the word embedding of the th word in the sentence
  - Let denote the th component of the embedding for the word located at the th position of a sentence

Put everything together



Summary

- RNN encodes entire input sequence into one fixed-length vector
- Attention mechanism allows network to refer the entire input sequence
- The attention model motivates many following works
  – ELMo, ULMFiT, GPT, BERT, ALBERT, RoBERTa, Transformer-XL