CSC413 Neural Networks and Deep Learning

Lecture 9: Attention Mechanism

Lecture 9

March 12/14, 2024

Section 1

Recurrent Neural Networks with Attention

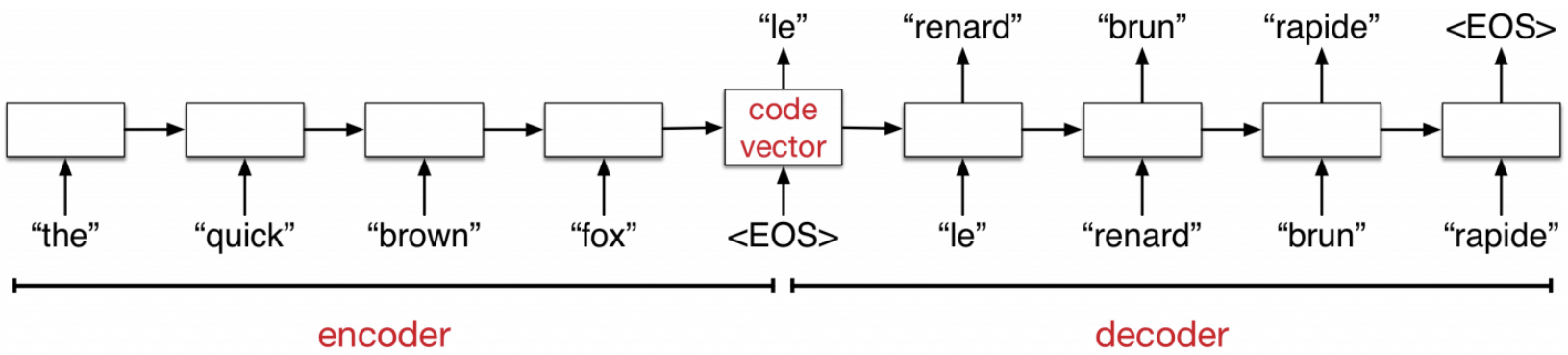
Recurrent Neural Networks

In lecture 8, we showed a **discriminative RNN** that makes a *prediction* based on a sequence (sequence as an input).

In a future tutorial, we will build a **generator RNN** to generate sequences (sequence as an output)

Sequence-to-sequence tasks

Another common example of a sequence-to-sequence task (seq2seq) is **machine translation**.



The network first reads and memorizes the sentences.

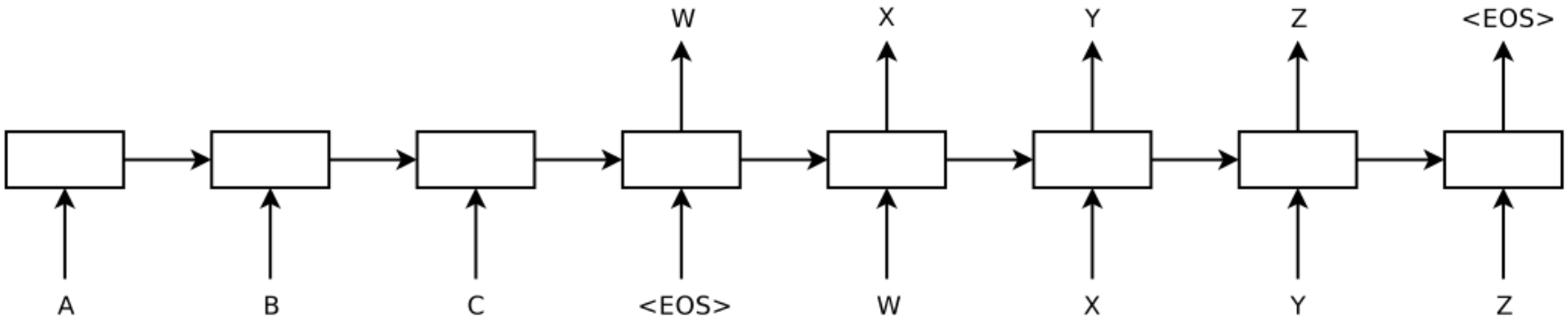
When it sees the "end token", it starts outputting the translation.

The "encoder" and "decoder" are two different networks with different weights.

How Seq2Seq works

The **encoder network** reads an input sentence and stores all the information in its hidden units.

The **decoder network** then generates the output sentence one word at a time.



But some sentences can be really long. Can we really store all the information in a vector of hidden units?

Human translators **refer back to the input**.

Attention-Based Machine Translation

We'll look at the translation model from the classic paper:

> *Bahdanau et al., Neural machine translation by jointly learning to align and translate. ICLR, 2015.*

Basic idea: each **output word** comes from **one input word**, or a handful of input words. Maybe we can learn to attend to only the relevant ones as we produce the output.

We'll use the opportunity to look at architectural changes we can make to RNN models to make it even more performant.

Encoder & Decoder Architectures

The encoder computes an **annotation** (hidden state) of each word in the input.

- The encoder is a **bidirectional RNN**

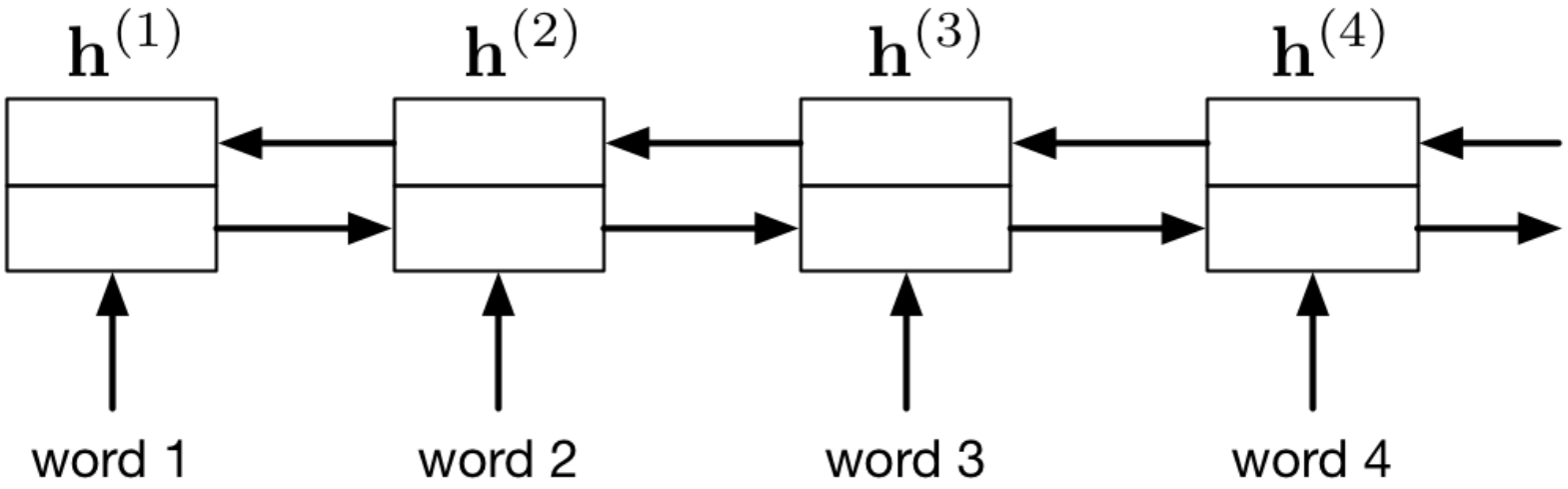
The decoder network is also an RNN, and makes predictions one word at a time.

- The decoder uses an **attention** mechanism (RNN with attention)

Encoder: Bidirectional RNN

The encoder is a **bidirectional RNN**. We have two RNNs: one that runs forward and one that runs backwards. These RNNs can be LSTMs or GRUs.

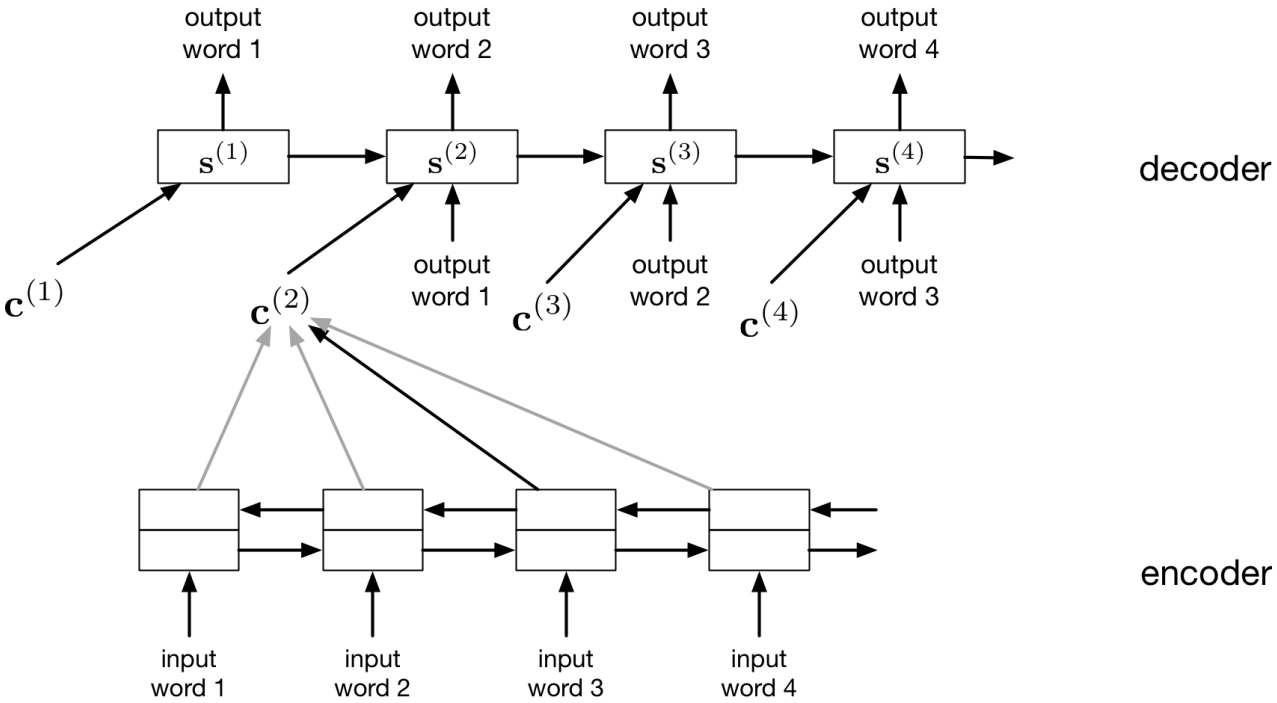
The annotation of a word is the concatenation of the forward and backward hidden vectors.



Decoder: RNN with attention

The decoder network is also an RNN, and makes predictions one word at a time.

The difference is that it also derives a **context vector** $\mathbf{c}^{(t)}$ at each time step, computed by **attending to the inputs**



Intuition behind "attending to the input"

> *"My language model tells me the next word should be an adjective. Find me an adjective in the input"*

We would like to *refer back* to one (or a few) of the input words to help with the translation task (e.g. find the adjective)

If you were programming a translator, you might

1. find an input word that is most likely an adjective (the attention function)
2. look up the input word that is the adjective (the weighted average)
3. translate the input word, e.g. using the input word, and a dictionary (the projection MLP)

An attentional decoder is like a *continuous* form of these last three steps.

The math behind "attending to the input"

The context vector is computed as a **weighted average** of the encoder's annotations:

$$\mathbf{c}^{(i)} = \sum_j \alpha_{ij} \mathbf{h}^{(j)}$$

The attention weights are computed as a softmax, where the input depends on the *annotation* $\mathbf{h}^{(j)}$ and the *decoder states* $\mathbf{s}^{(t)}$:

$$e_{ij} = a(\mathbf{s}^{(i-1)}, \mathbf{h}^{(j)})$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} exp(e_{ij'})}$$

The attention function depends on the **annotation vector**, rather than the **position in the sentence**. It is a form of content-based addressing.

- "My language model tells me the next word should be an adjective. Find me an adjective in the input"

Example: how to obtain a context vector?

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{c}^{(t)} = \begin{bmatrix} ? & ? & ? \end{bmatrix}^T$$

Example: average pooling is context independent

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\begin{aligned} \mathbf{c}^{(t)} &= \mathit{average}(\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) \\ &= \begin{bmatrix} 2 & 0.6 & 1 \end{bmatrix}^T \end{aligned}$$

Example: Attention

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$
$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$
$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{s}^{(t-1)} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$$

$$\alpha_t = \text{softmax}\left(\begin{bmatrix} f(\mathbf{s}^{(t-1)}, \mathbf{h}^{(1)}) \\ f(\mathbf{s}^{(t-1)}, \mathbf{h}^{(2)}) \\ f(\mathbf{s}^{(t-1)}, \mathbf{h}^{(3)}) \end{bmatrix}\right) = \begin{bmatrix} \alpha_{t1} \\ \alpha_{t2} \\ \alpha_{t3} \end{bmatrix}$$

Example: Dot-Product Attention

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$
$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$
$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{s}^{(t-1)} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$$
$$\alpha_t = \text{softmax}\left(\begin{bmatrix} \mathbf{s}^{(t-1)} \cdot \mathbf{h}^{(1)} \\ \mathbf{s}^{(t-1)} \cdot \mathbf{h}^{(2)} \\ \mathbf{s}^{(t-1)} \cdot \mathbf{h}^{(3)} \end{bmatrix} \right) = \begin{bmatrix} \alpha_{t1} \\ \alpha_{t2} \\ \alpha_{t3} \end{bmatrix}$$

Example: Dot-Product Attention

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{s}^{(t-1)} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$$

$$\alpha_t = \text{softmax}\left(\begin{bmatrix} 1 & 0 & 5 \\ 3 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right)$$

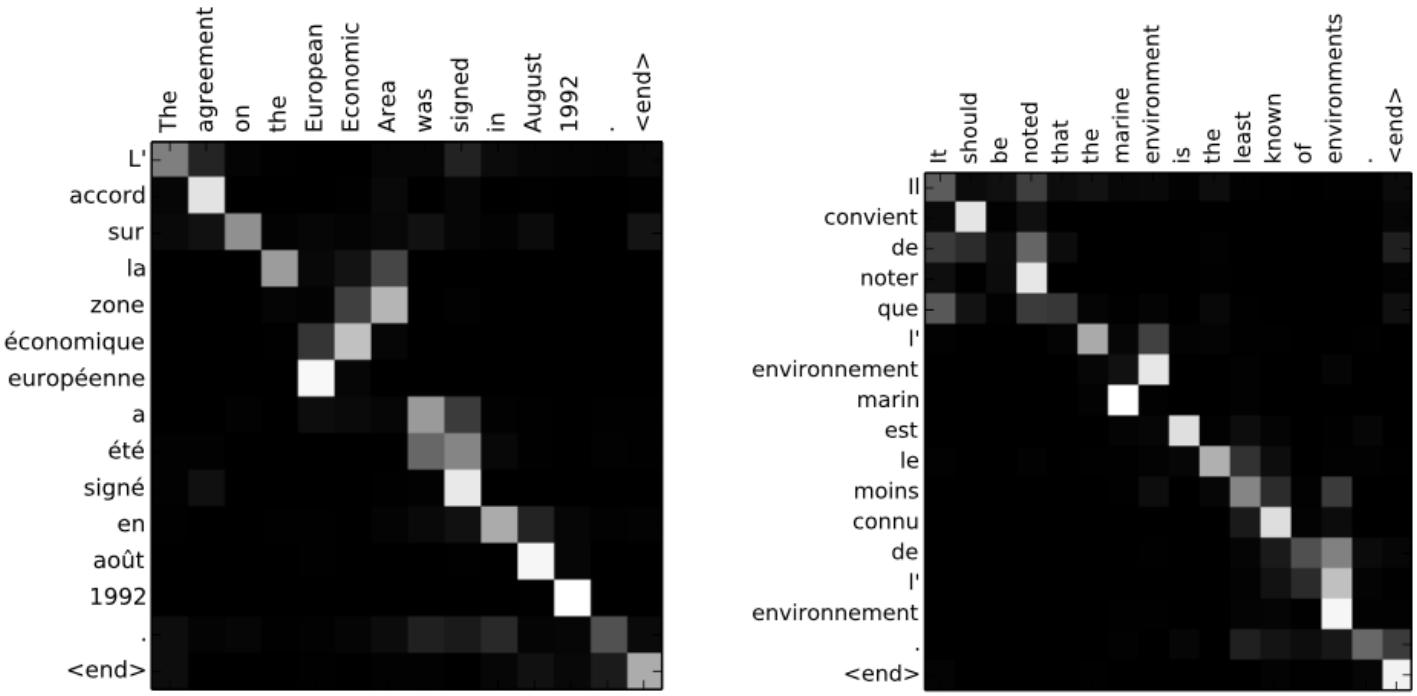
$$\mathbf{c}^{(t)} = \alpha_{t1}\mathbf{h}^{(1)} + \alpha_{t2}\mathbf{h}^{(2)} + \alpha_{t2}\mathbf{h}^{(3)}$$

Attention Demo Video

https://play.library.utoronto.ca/watch/9ed8b3c497f82b510e9ecf441c5eef4f

Visualization of Attention

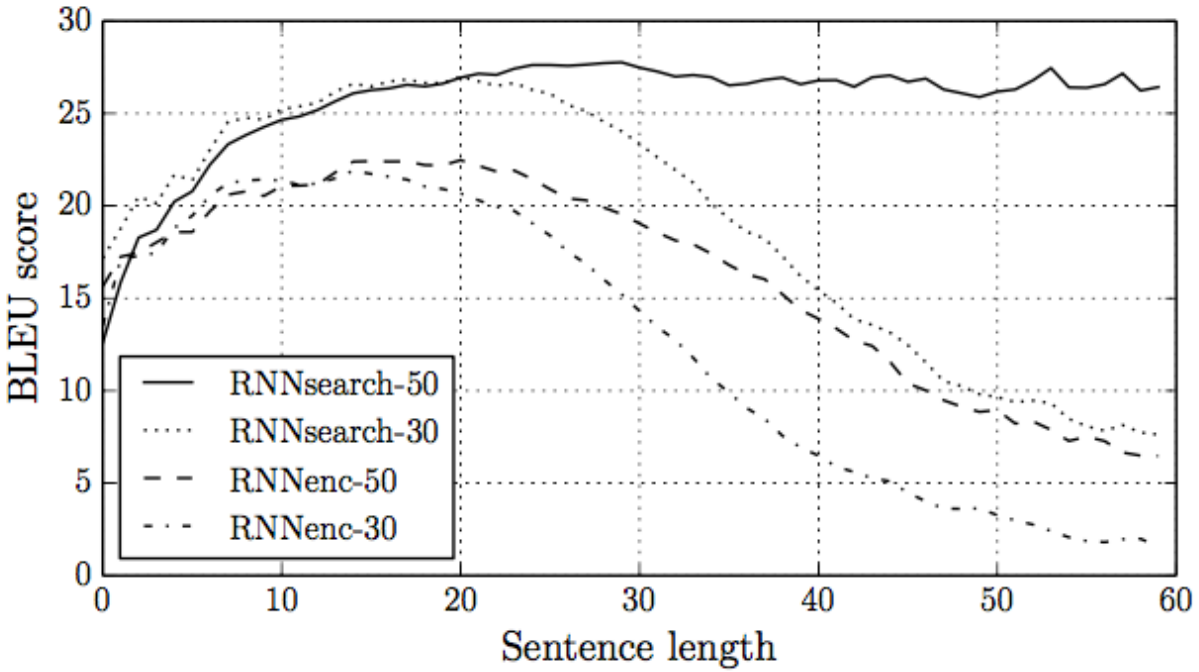
Visualization of the attention map (the α_{ij}s at each time step)



Nothing forces the model to go (roughly) linearly through the input sentences, but somehow it learns to do it!

Attention Performance

The attention-based translation model does much better than the encoder/decoder model on long sentences.



Attention-Based Caption Generation

Caption Generation Task:

- Input: Image
- Output: Caption (sequence of words or characters)

Attention can also be used to understand images.

- We humans can't process a whole visual scene at once.
- The fovea of the eye gives us high-acuity vision in only a tiny region of our field of view.
- Instead, we must integrate information from a series of glimpses.

The next few slides are based on this paper from the UofT machine learning group:

Xu et al. Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention. ICML, 2015.

Attention for Caption Generation

The caption generation task: take an image as input, and produce a sentence describing the image.

- **Encoder**: a classification conv net like VGG. This computes a bunch of feature maps over the image.
- **Decoder**: an attention-based RNN, analogous to the decoder in the translation model
  - In each time step, the decoder computes an attention map over the entire image, effectively deciding which regions to focus on.
  - It receives a context vector, which is the weighted average of the conv net features.

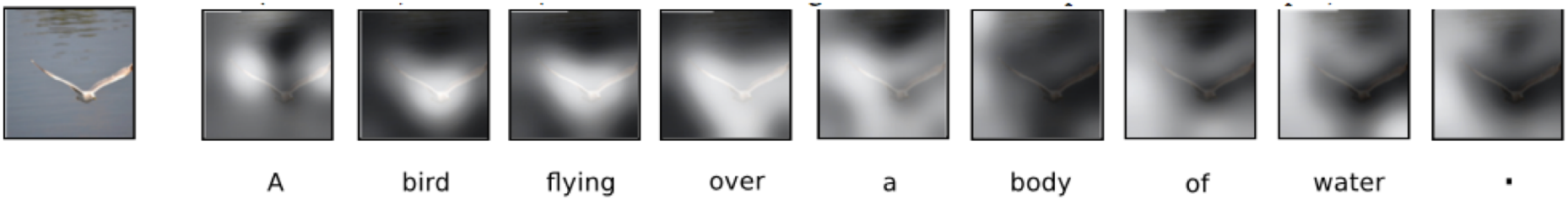
Similar math as before: difference is that j is a pixel location

$$e_{ij} = a(\mathbf{s}^{(i-1)}, \mathbf{h}^{(j)})$$

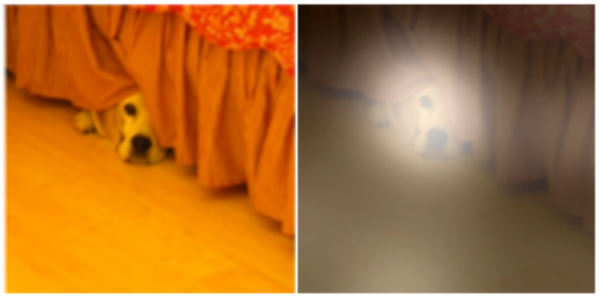
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} exp(e_{ij'})}$$

What Attention tells us

This lets us understand where the network is looking as it generates a sentence.




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.

What Attention tells us about mistakes

This can also help us understand the network's mistakes.

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

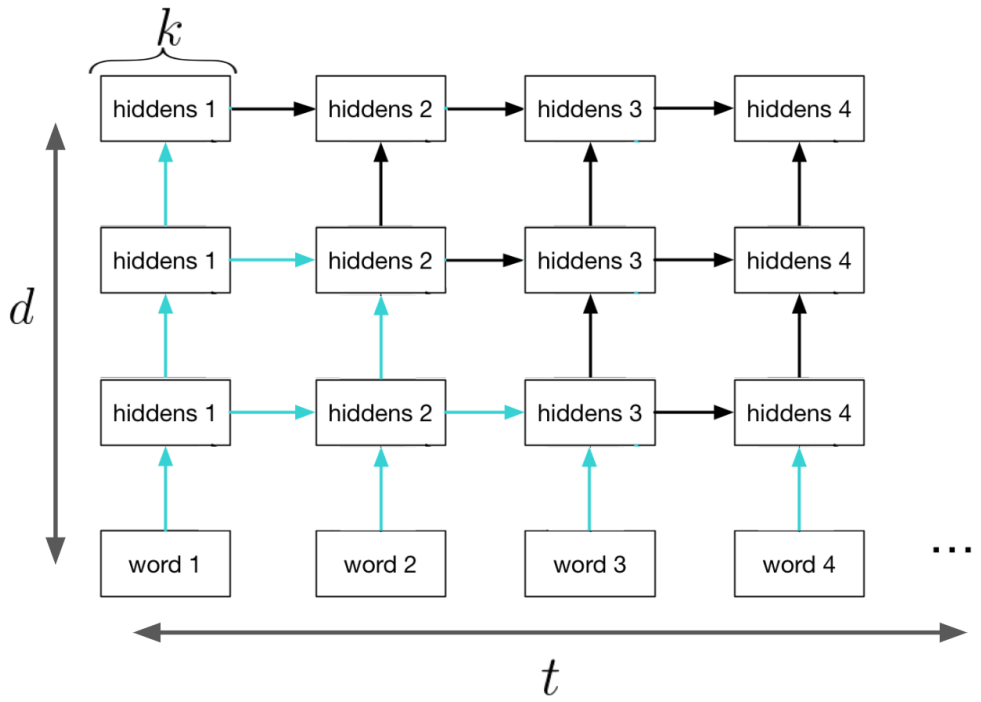
A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

Multi-Layer RNNs

Finally, to get more capacity/performance out of RNNs, you can **stack** multiple RNN's together!

The hidden state of your first RNN becomes the input to your second layer RNN.



RNN Disadvantage

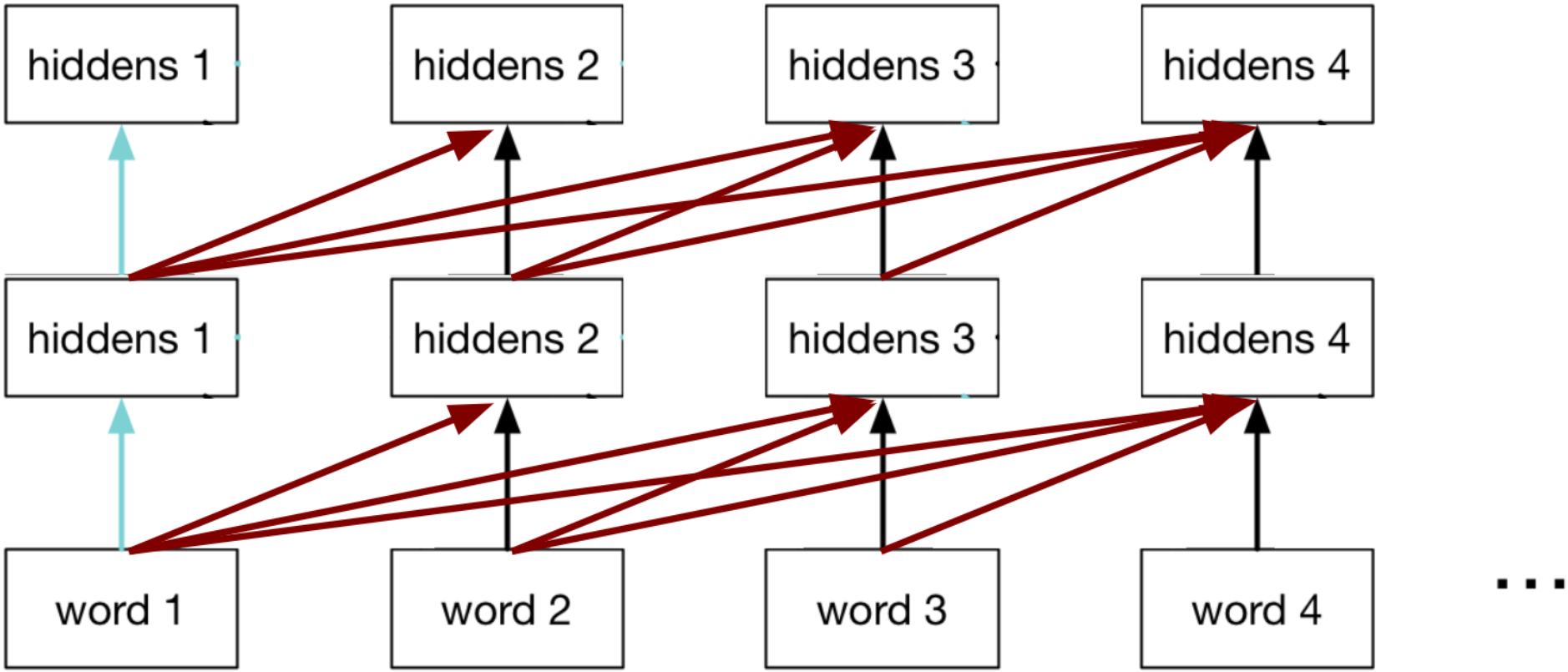
One disadvantage of RNNS (and especially multi-layer RNNs) is that they require a long time to train, and are more difficult to parallelize. (Need the previous hidden state $h^{(t)}$ to be able to compute $h^{(t+1)}$)

Section 2

Transformer

Transformer

Idea: Do away with recurrent networks altogether; instead exclusively use attention to obtain the history at the hidden layers



Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.

Vaswani, Ashish, et al. Attention is all you need.

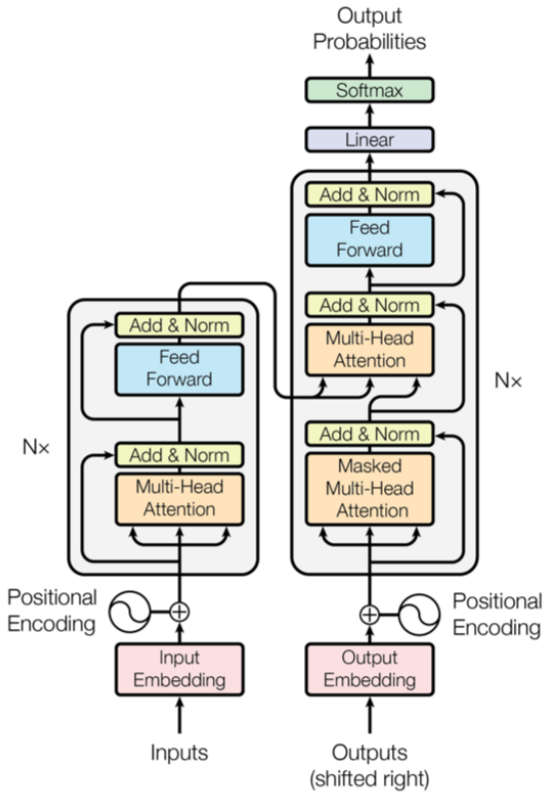


Figure 1: The Transformer - model architecture.

Figure 1: typical encoder-decoder blocks of a transformer

Transformer has a encoder-decoder architecture similar to the previous sequence-to-sequence RNN models, except all the recurrent connections are replaced by the attention modules.

Attention Mapping

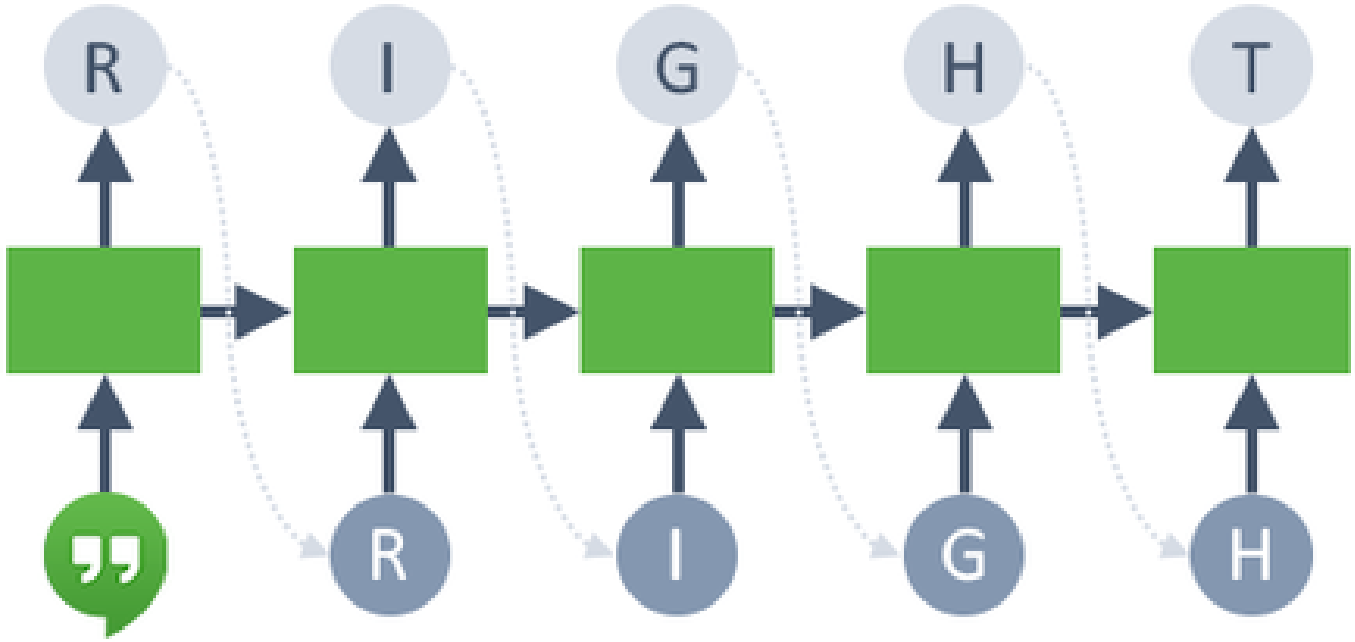
In general, attention mapping can be described as a function of a query and a set of key-value pairs. Transformer uses a "scaled dot-product attention" to obtain the context vector:

$$\mathbf{c}^{(t)} = \text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right) V$$

This is very similar to the attention mechanism we saw earlier, but we scale the pre-softmax values (the logits) down by the square root of the key dimension d_K.

Attention Mapping in the decoder

When training the decoder (e.g. to generate a sequence), we **mask out** the desired output so that we preserve the autoregressive property.



Recall: Dot-Product Attention

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{s}^{(t-1)} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$$

$$\alpha_t = \text{softmax}\left(\begin{bmatrix} 1 & 0 & 5 \\ 3 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right)$$

$$\mathbf{c}^{(t)} = \alpha_{t1}\mathbf{h}^{(1)} + \alpha_{t2}\mathbf{h}^{(2)} + \alpha_{t2}\mathbf{h}^{(3)}$$

Scaled Dot-Product Attention

$$\mathbf{h}^{(1)} = \begin{bmatrix} 1 & 3 & 9 \end{bmatrix}^T$$

$$\mathbf{h}^{(2)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$\mathbf{h}^{(3)} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}^T$$

$$\mathbf{s}^{(t-1)} = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$$

$$\alpha_t = \text{softmax}\left(\frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 0 & 5 \\ 3 & 0 & -1 \\ 0 & 1 & 2 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right)$$

$$\mathbf{c}^{(t)} = \alpha_{t1}\mathbf{h}^{(1)} + \alpha_{t2}\mathbf{h}^{(2)} + \alpha_{t2}\mathbf{h}^{(3)}$$

Q: Which values represent the Q, K, and V?

Attending to the input (encoder)

Transformer models attend to both the encoder annotations and its previous hidden layers.

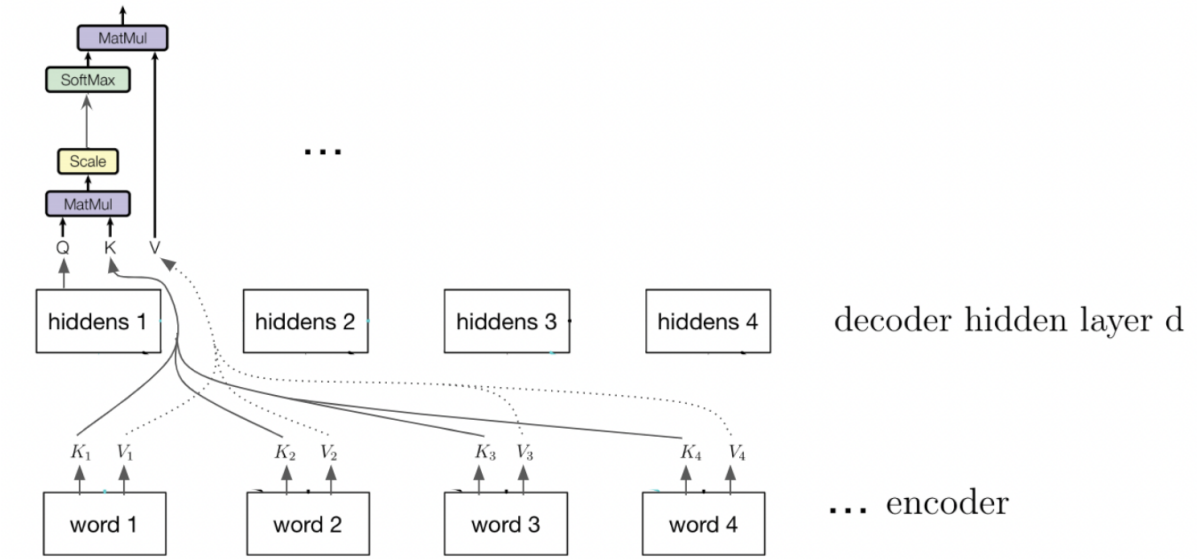


Figure 2: how the encoder contributes K and V

When attending to the encoder annotations, the model computes the key-value pairs using linearly transformation of the encoder outputs.

Self-attention

Transformer models also use "self-attention" on its previous hidden layers. When applying attention to the previous hidden layers, the causal structure is preserved:

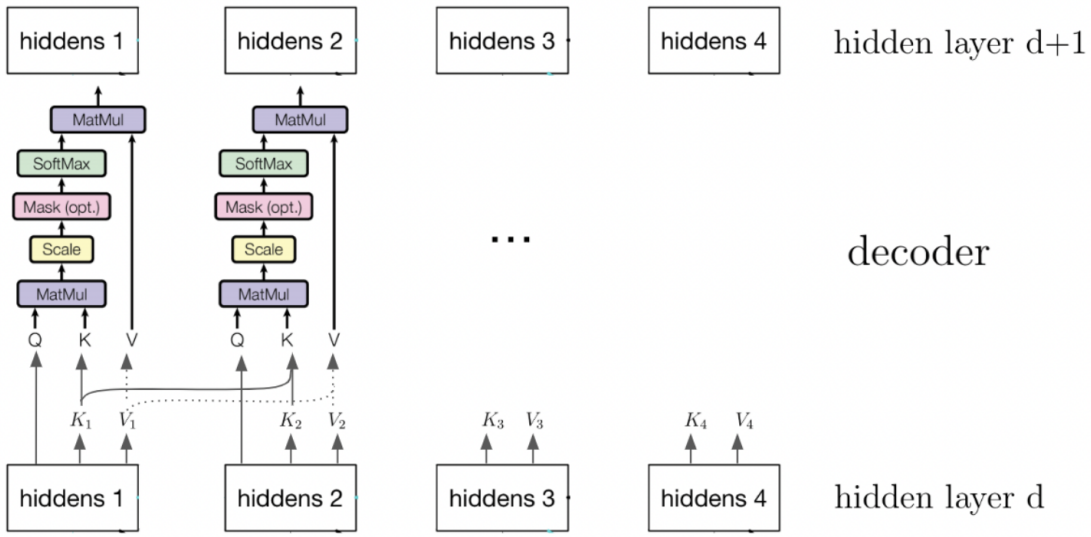


Figure 3: feeding to deeper layers and future hidden states

Multi-headed Attention

The Scaled Dot-Product Attention attends to one or few entries in the input key-value pairs.

But humans can attend to many things simultaneously

Idea: apply scaled dot-product attention *multiple times* on the linearly transformed inputs:

$$\mathbf{c}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{MultiHead}(Q, K, V) = \mathit{concat}(\mathbf{c_1}, \ldots, \mathbf{c}_h)W^O$$

Input Sequence Order

$$\mathbf{c}_i = \text{attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{MultiHead}(Q, K, V) = \mathit{concat}(\mathbf{c_1}, \ldots, \mathbf{c}_h)W^O$$

Unlike RNNs and CNN encoders, the attention encoder output do *not* depend on the order of the inputs. Can you see why?

However, the order of the sequence convey important information for the machine translation task, language modeling, and other tasks.

Positional Encoding

Idea: Add positional information of each input token in the sequence into the input embedding vectors.

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

The final input embeddings are the concatenation of the learnable embeddings and the positional encoding.

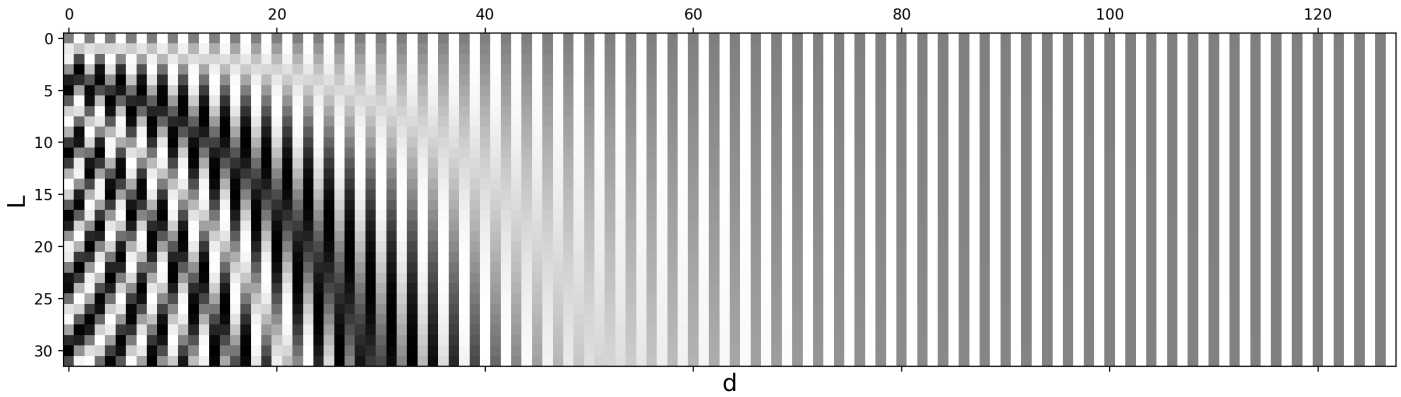


Figure 4: sinusoidal position encodings visualized

Transformer Machine Translation

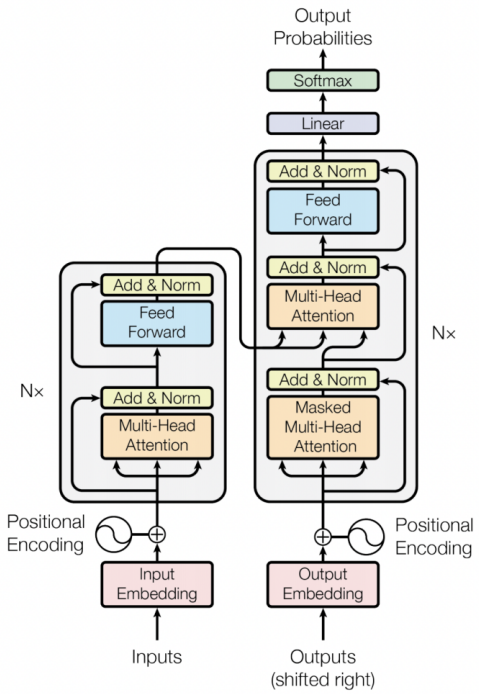


Figure 5: typical encoder-decoder blocks of a transformer

- Encoder-Decoder architecture like RNN, but use attention modules instead of recurrent modules.
- Use N stacked self-attention layers.
- Skip-connections help preserve the positional and identity information from the input sequences.

Visualizing Attention

Self-attention layer learns that "it" could refer to different entities in different contexts.

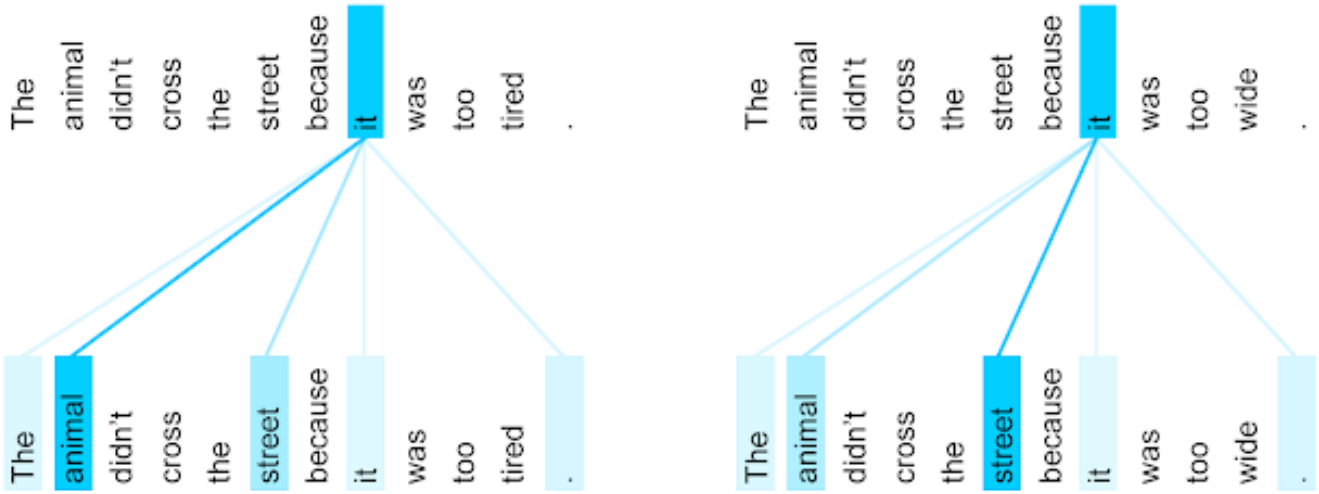


Figure 6: visualized attention on a sentence

See ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Backprop

During backprop, in the standard encoder-decoder RNN, the maximum path length across time is the number of time steps.

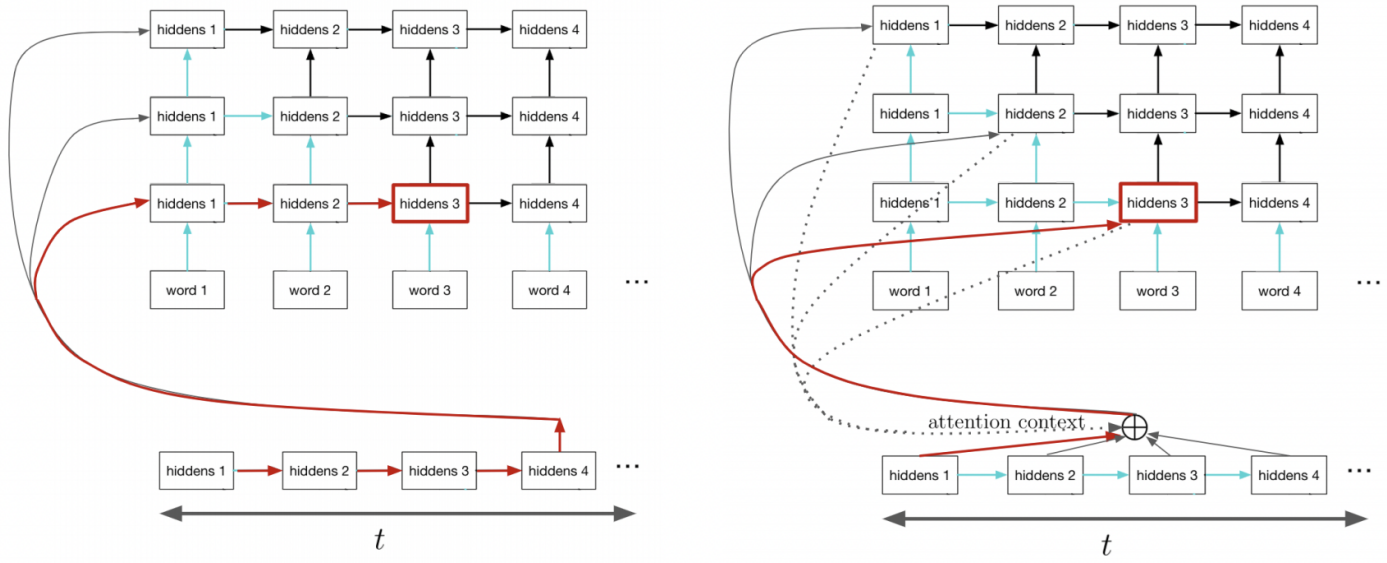


Figure 7: backprop paths in RNNs vs transformers

Attention-based RNNs have a constant path length between the encoder inputs and the decoder hidden states. (Learning becomes easier. Why?)

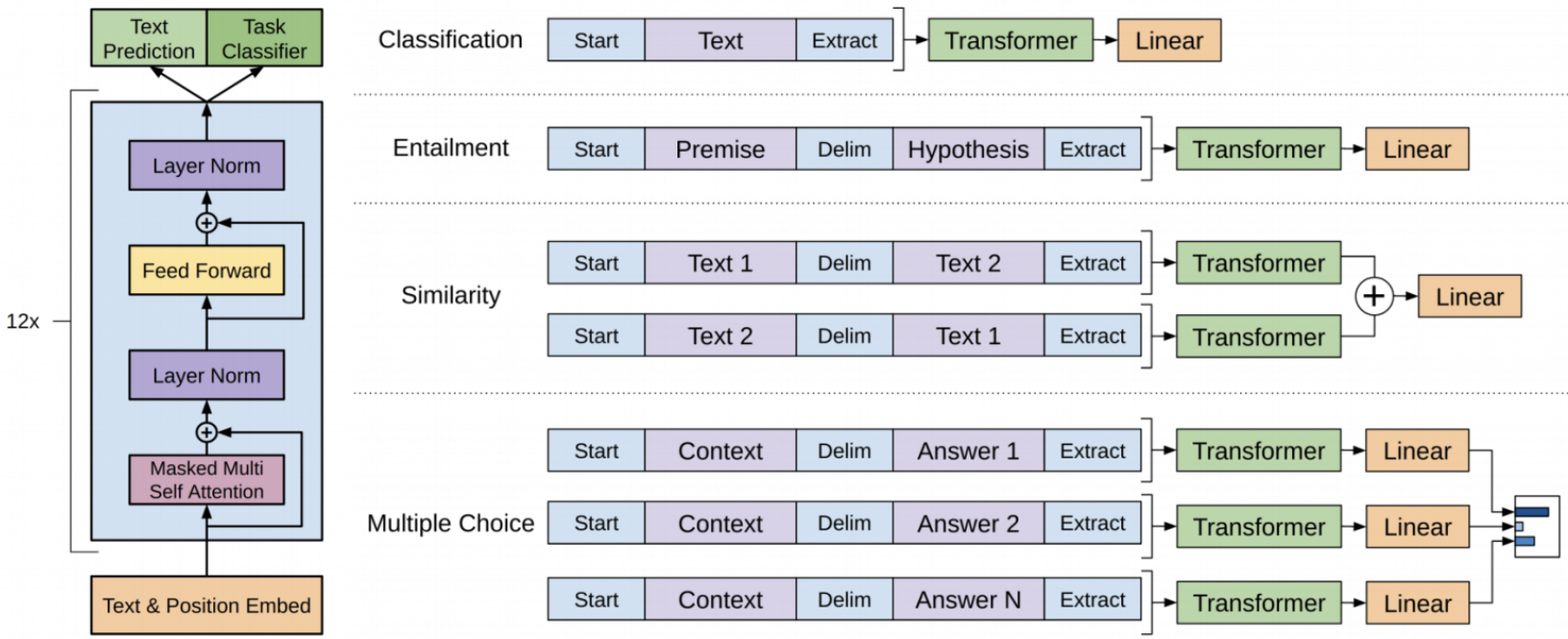
Transformer Tutorial

www.youtube.com/watch?v=XSSTuhyAmnI&ab_channel=AriSeff

Transformer Language Pre-Training

Similar to pre-training computer vision models on ImageNet, we can pre-train a language model for NLP tasks.

The pre-trained model is then fine-tuned on textual entailment, question answering, semantic similarity assessment, and document classification



Radford, Alec, et al. "Improving Language Understanding by Generative Pre-Training," 2018

GPT-3: Generating an Article

Title: United Methodists Agree to Historic Split

Subtitle: Those who oppose gay marriage will form their own denomination

Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.

The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban the

Brown et al. 2020, Language Models are Few-Shot Learners

GPT-3: Correcting English Grammar

Poor English input: The patient was died.

Good English output: The patient died.

Poor English input: We think that Leslie likes ourselves.

Good English output: We think that Leslie likes us.

Poor English input: Janet broke Bill on the finger.

Good English output: Janet broke Bill's finger.

Poor English input: Mary arranged for, in St. Louis, John to rent a house cheap.

Good English output: Mary arranged for John to rent a house in St. Louis.

Poor English input: Today I have went to the store to to buys some many bottle of water.

Good English output: Today I went to the store to buy some bottles of water.

Poor English input: I have tried to hit ball with bat, but my swing is has miss.

Good English output: I tried to hit the ball with the bat, but my swing missed.

Brown et al. 2020, Language Models are Few-Shot Learners

Large Language Models (LLMs)

Many Transformer-based models have been developed since 2017.

- Generative Pre-Trained Transformer (GPT-1/2/3/4)
  - Decoder-only 12-layer/12-headed Transformer.
  - Inspired GPT-Neo, GPT-J, GPT-NeoX, ChatGPT.
- Bidirectional Encoder Representations from Transformers (BERT)
  - Encoder-only 12-layer/12-headed Transformer.
  - Inspired ALBERT, RoBERTa, SBERT, DeBERTa, etc.
- And many, many more: en.wikipedia.org/wiki/Large_language_model

Many benchmarks have been developed such as GLUE and SQuAD.

Big players in the LLM space include OpenAI, Cohere, Google, Meta, Microsoft, Amazon, EleutherAI, Hugging Face.

ChatGPT

What is ChatGPT? We'll let it speak for itself:

> *I am ChatGPT, a large language model developed by OpenAI. I use machine learning algorithms to generate responses to questions and statements posed to me by users. I am designed to understand and generate natural language responses in a variety of domains and topics, from general knowledge to specific technical fields. My purpose is to assist users in generating accurate and informative responses to their queries and to provide helpful insights and suggestions.*

It's based on OpenAI's GPT-3, which itself is based on the **transformer** architecture.