Natural Language Processing with Deep Learning

Seq2Seq, Attention and Transformer

Myungjun Kim

Seoul National University

February 18, 2022

Contents

Machine Translation

- The core idea of **statistical machine translation** (SMT) is to learn a probabilistic models from data.

$$\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y)$$

  We want to find best target language sentence y, given source language sentence x.

- **Neural machine translation** (NMT) is a way to do machine translation with a *single end-to-end* neural network.

Advantages of NMT

Compared to SMT, NMT has many advantages:

- Better performance
  - More fluent
  - Better use of context
  - Better use of phrase similarities
- A single neural network to be optimized end-to-end
  - No subcomponents to be individually optimized
- Requires much less human engineering effort
  - No feature engineering
  - Same method for all language pairs

BLEU (Bilingual Evaluation Understudy) Score

BLEU compares the machine-written translation to human-written translation, and computes a similarity score based on *n-gram precision* and a penalty for too-short system translations.

$$\text{BLEU} = \beta \prod_{n=1}^{N} p_n^{w_n}$$

where $\beta = e^{\min(0, 1 - \frac{\text{length of ref.}}{\text{length of MT}})}$, $w_n = 1/N$ and

$$p_n = \frac{\#(\text{matched n-grams})}{\#(\text{n-grams in candidate translation})}.$$

BLEU (Bilingual Evaluation Understudy) Score

빛을 쐬는 사람은 완벽한 어둠에서 잠든 사람과 비교할 때 우울증이 심해질 가능성이 훨씬 높았다.

VS

빛을 쐬는 노인은 완벽한 어두운곳에서 잠든 사람과 비교할 때 강박증이 심해질 기회가 훨씬 높았다.

- 1-gram precision $= \frac{10}{14}$
- 2-gram precision $= \frac{5}{13}$
- 3-gram precision $= \frac{2}{12}$
- 4-gram precision $= \frac{1}{11}$

$$\prod_{n=1}^{N} p_n^{w_n} = \left(\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11}\right)^{1/4}$$

with $w_n = 1/N = 1/4$.

Contents

Sequence-to-Sequence Learning

- **Sequence-to-sequence** models [1], [2] are deep learning models that have achieved a lot of success in downstream tasks.
  - Machine Translation
  - Text Summarization
  - Image Captioning
- A sequence-to-sequence model is a model that takes a sequence of items and outputs another sequence of items.

Sequence-to-Sequence Learning

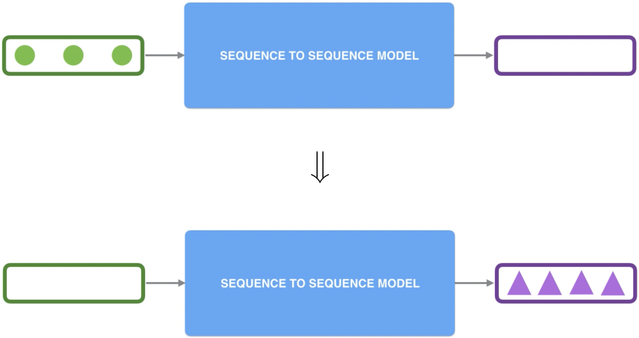


Figure 1: A trained sequence-to-sequence takes a sequence of items (words, letters, features of an images etc.) and outputs another sequence of items. [3]

RNN Encoder-Decoder

- A Seq2Seq model consists of an **encoder** and a **decoder**.
- The encoder processes each item in the input sequence, and compiles the information into a **context** vector.
- After processing the entire input sequence, the encoder sends the context over to the decoder, which begins producing the output sequence item by item.

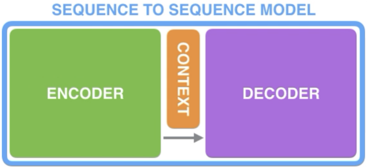


Figure 2: An encoder and a decoder of a Seq2Seq model.

RNN Encoder-Decoder

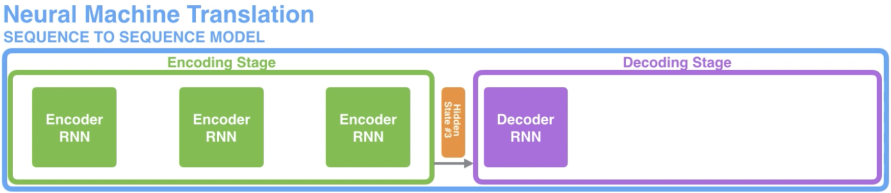


Figure 3: An unrolled view of a RNN encoder-decoder.

Contents

Bottleneck Problem

- What if an input sentence becomes too long?
- The context vector turned out to be a **bottleneck** for these types of models, which makes it challenging for the models to deal with long sentences.

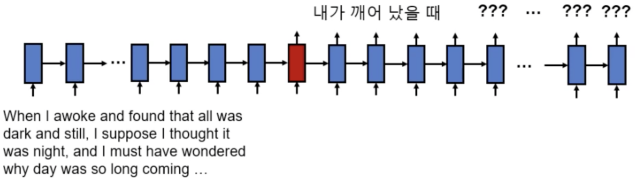


Figure 4: It is hard to encode a long input sentence into a fixed-length context vector.

Bottleneck Problem

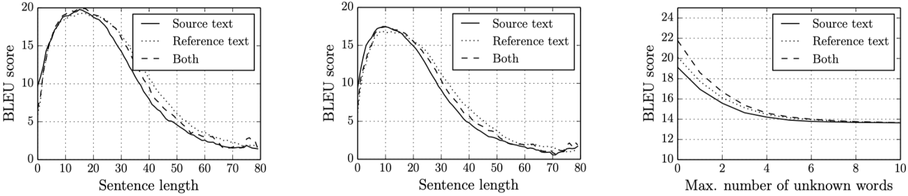


Figure 5: The BLEU scores achieved by several methods. The neural machince translation system underperforms with long sentences. [4]

Seq2Seq with Attention

- **Attention** [5], [6] highly improved the quality of machine translation systems.
- Rather than using fixed context vector, we can use encoder's each state with current state to generate *dynamic* context vector.

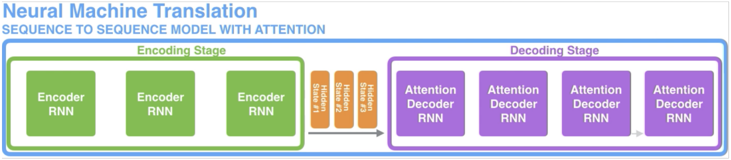
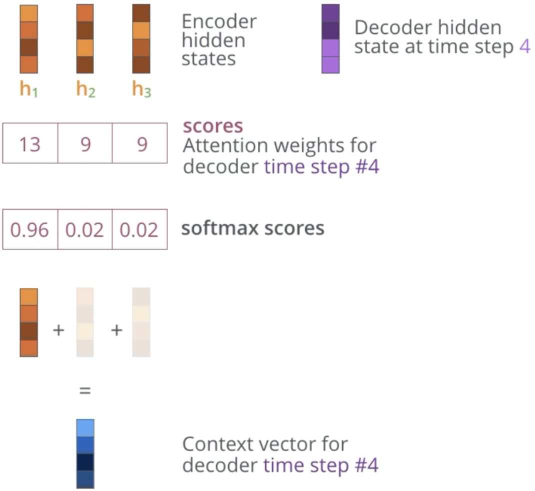


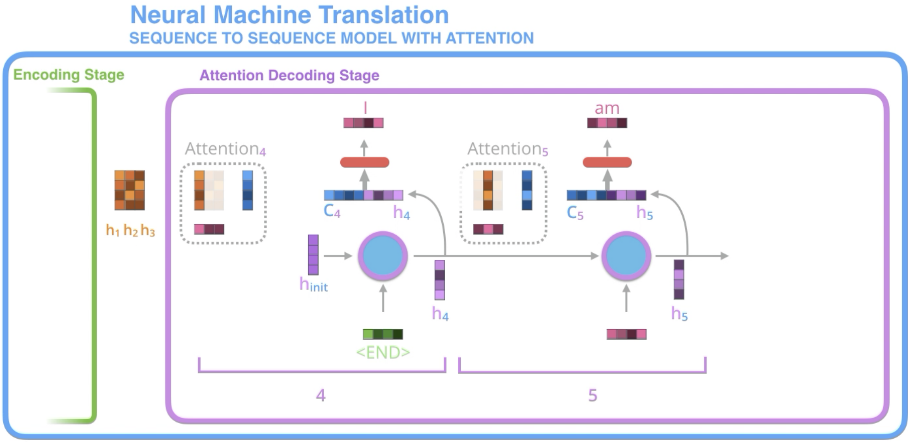
Figure 6: Attention model encodes information into sequence of vectors not in a single context vector, and chooses a subset of these vectors adativley while decoding the translation.

Seq2Seq with Attention

1. Prepare inputs
2. Score each hidden state
3. Softmax the scores
4. Multiply each vector by its softmaxed score
5. Sum up the weighted vectors



Seq2Seq with Attention



Seq2Seq with Attention

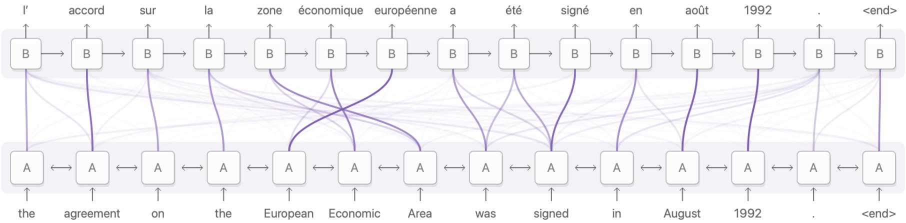


Figure 7: Attention allows the model to focus on the relevant parts of the input sequence as needed. [7]

Contents

Transformer

- The **Transformer** was proposed in the paper *Attention is All You Need*. [8]
- It uses attention to boost the speed with which these models can be trained and easy to *parallerize*.
- Inside the Transformer, there are an encoding component, a decoding compoenent and connections between them.
- Recent state-of-the-arts models (e.g., GPT-3, BERT) are based on Transformer architecture.

Transformer

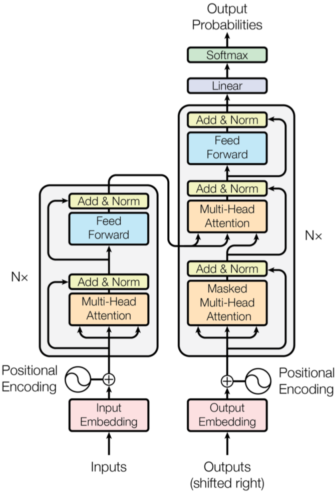


Figure 8: The Transformer - model architecture.

Positional Encoding

- The Transformer contains no recurrence and no convolution.
- In order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence.
- We add **positional encodings** to the input embeddings at the bottoms of the encoder and decoder stacks.

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

Positional Encoding

Two properties that a good positional encoding scheme should have

- The norm of encoding vector is the same for all positions.
- The further the two positions, the larger the distance.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	0.000	1.275	2.167	2.823	3.361	3.508	3.392	3.440	3.417	3.266
X2	1.275	0.000	1.104	2.195	3.135	3.511	3.452	3.442	3.387	3.308
X3	2.167	1.104	0.000	1.296	2.468	3.067	3.256	3.464	3.498	3.371
X4	2.823	2.195	1.296	0.000	1.275	2.110	2.746	3.399	3.624	3.399
X5	3.361	3.135	2.468	1.275	0.000	1.057	2.176	3.242	3.659	3.434
X6	3.508	3.511	3.067	2.110	1.057	0.000	1.333	2.601	3.169	3.118
X7	3.392	3.452	3.256	2.746	2.176	1.333	0.000	1.338	2.063	2.429
X8	3.440	3.442	3.464	3.399	3.242	2.601	1.338	0.000	0.912	1.891
X9	3.417	3.387	3.498	3.624	3.659	3.169	2.063	0.912	0.000	1.277
X10	3.266	3.308	3.371	3.399	3.434	3.118	2.429	1.891	1.277	0.000

Figure 9: A simple example of the positional encoding with $n = 10, d_{model} = 10$.

Transformer - Encoder

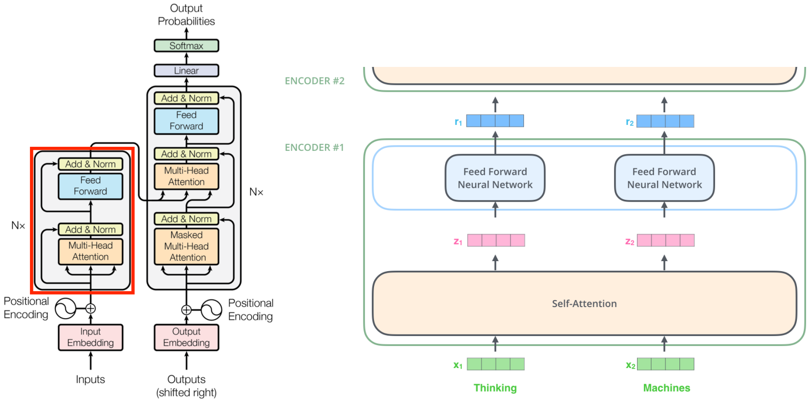


Figure 10: The encoder structure of Transformer. The encoding component is a stack of encoders. [9]

Self-Attention

- An attention function can be described as mapping a query and a set of key-value pairs to an output.
- The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by the query with the corresponding key.
- Self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding.
- Self attention is the method the Transformer uses to bake the *understanding* of other relevant words into the one which currently processed.

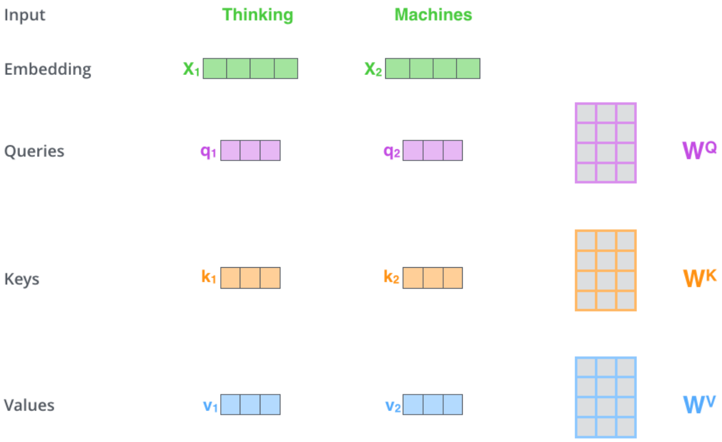
Self-Attention

"The animal didn't cross the street because it was too tired."

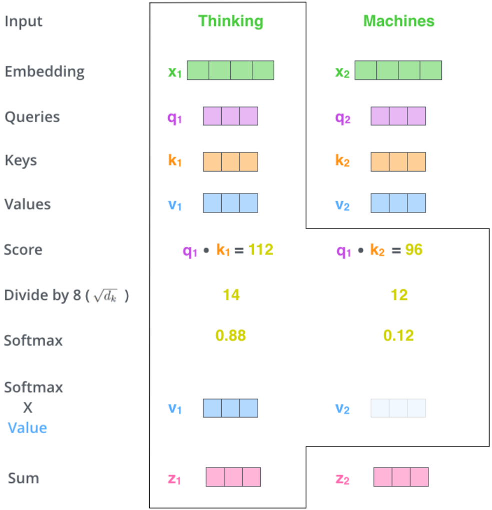
- **Query** is a representation of the current word used to score against all the other words.
- **Keys** are like labels for all the words in the segment.
- **Values** are actual word representations, once we have scored how relevant each word is, these are the values we add up to represent the current word.

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

Self-Attention



Self-Attention



Self-Attention

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

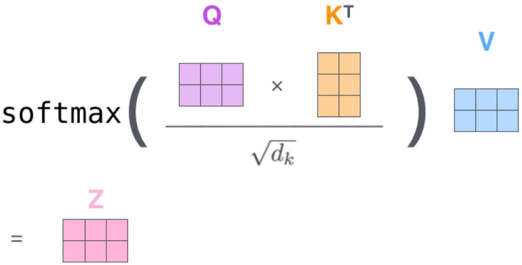


Figure 11: The self-attention calculation in matrix form.

Self-Attention

1) This is our input sentence*

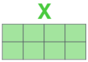
2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

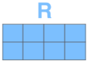
4) Calculate attention using the resulting Q/K/V matrices

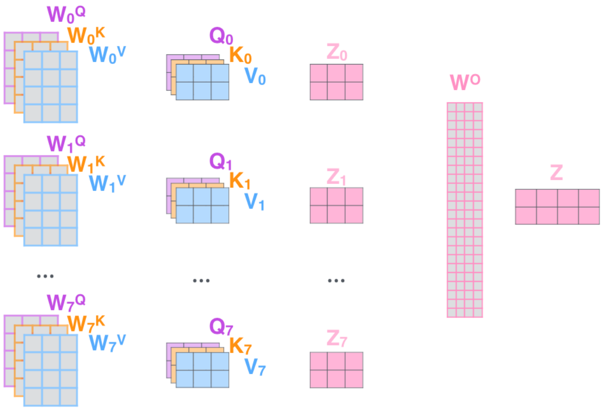
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

Thinking Machines



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one





Transformer - Decoder

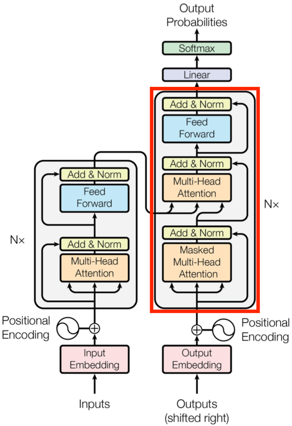


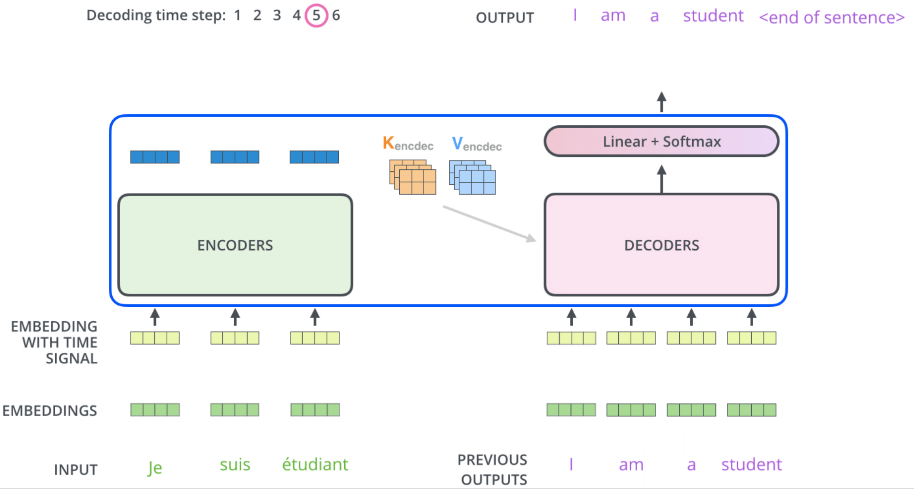
Figure 12: The decoder structure of Transformer. The decoding component is also a stack of decoders.

Attention in the Transformer

The Transformer uses multi-head attention in three different ways:

- Self-Attention
  - The encoder contains self-attention layesrs. Values and queries come from the same place, the output of the previous layer in the encoder.
- Masked Self-Attention
  - Self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to annd including that position.
- Encoder-Decoder Attention
  - The queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sentence.

Transformer - Decoder



References

[1] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[2] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[3] Visualizing a neural machine translation model (mechanics of seq2seq models with attention). `https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/`.

[4] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[6] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models, 2016.

[7] Attention and augmented recurrent neural networks. `https://distill.pub/2016/augmented-rnns/`.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[9] The illustrated transformer. `https://jalammar.github.io/illustrated-transformer/`.