ARTIFICIAL INTELLIGENCE

Transformers Explained Vi (Part 2): How it works, ste step

A Gentle Guide to the Transformer under the hood, a operation.

Ketan Doshi
Jan 2, 2021 12 min read

INTUITIVE TRANSFORMERS SERIES NLP

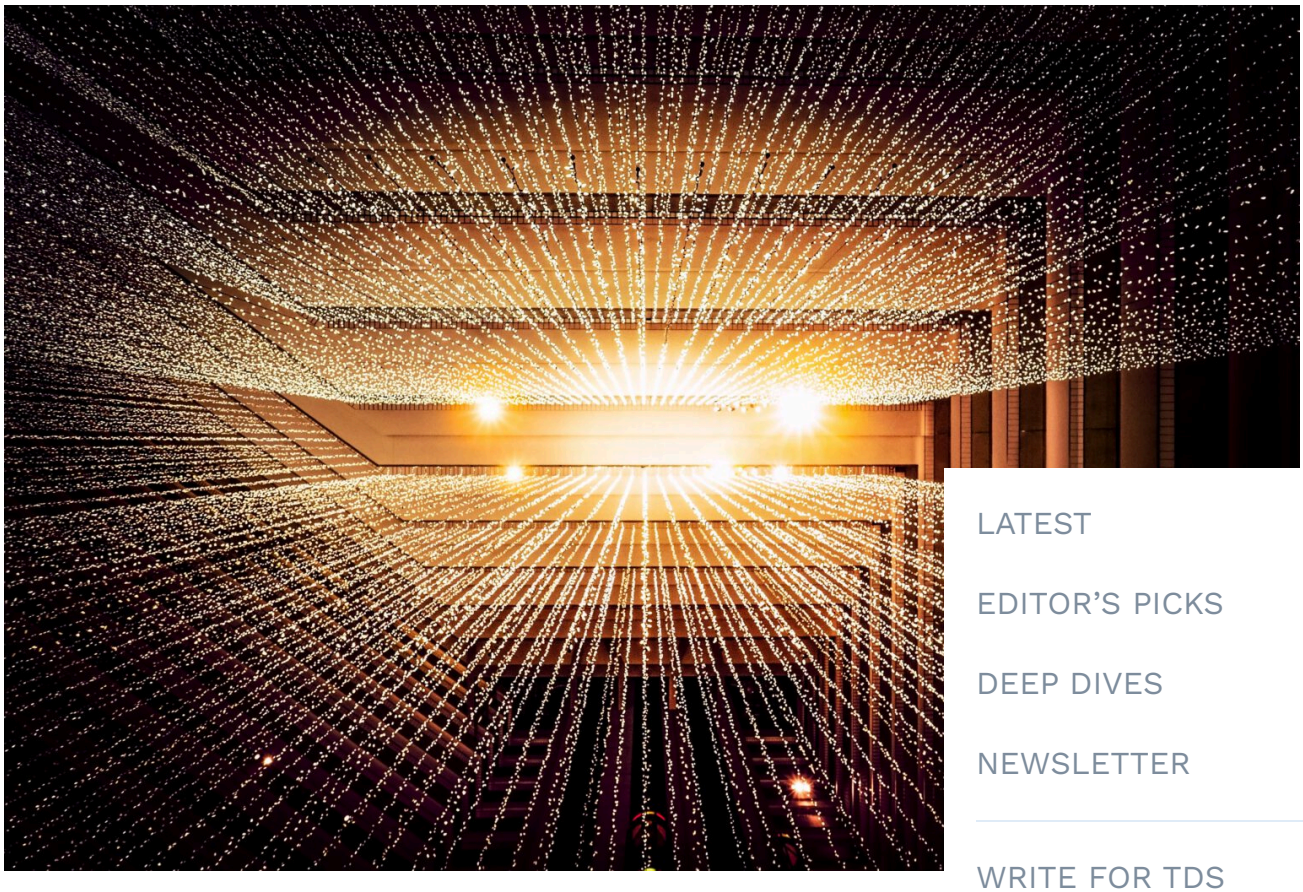


Photo by Joshua Sortino on Unsplash

This is the second article in my series on Transf
first article, we learned about the functionality
how they are used, their high-level architecture,
advantages.

In this article, we can now look under the hood
how they work in detail. We'll see how data flow
system with their actual matrix representations
understand the computations performed at eac

Here's a quick summary of the previous and foll
the series. My goal throughout will be to unders
something works but why it works that way.

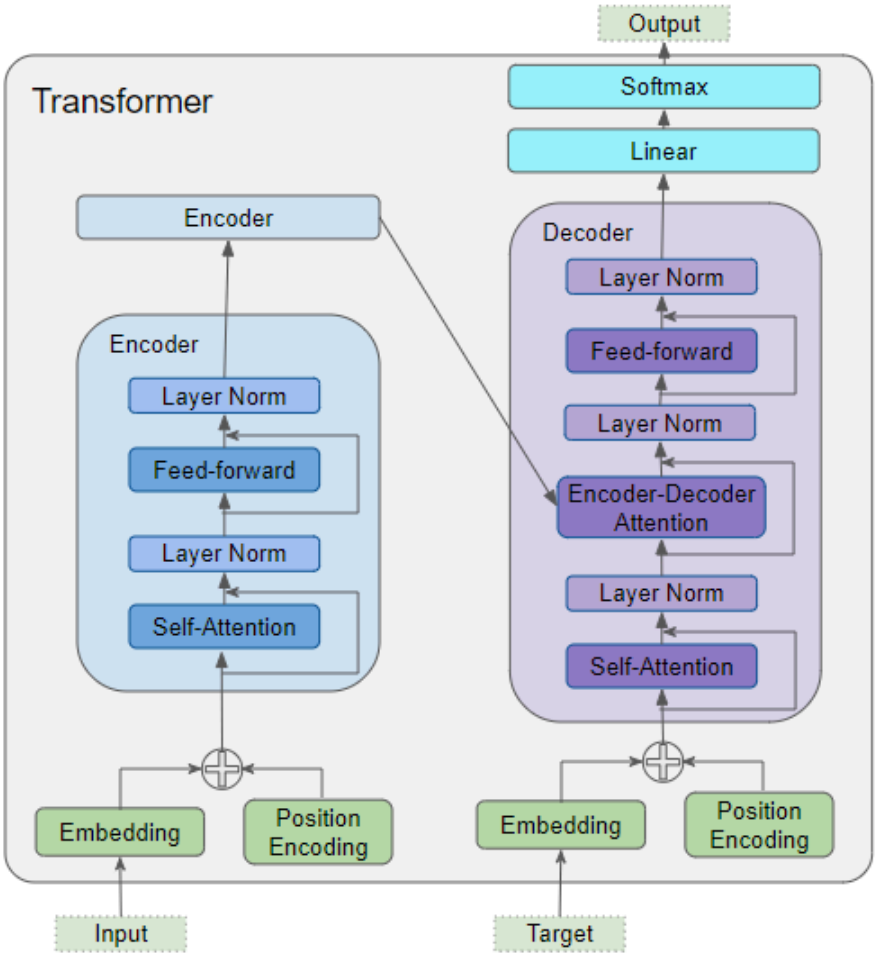
1. Overview of functionality **** *(How Transformers are used, and why they are better than RNNs. Components of the architecture, and behavior during Training and Inference*)
2. **How it works – this article** *(Internal operation end-to-end. How data flows and what computations are performed, including matrix representations)*
3. Multi-head Attention *(Inner workings of the Attention module throughout the Transformer)*
4. Why Attention Boosts Performance *(Not just does but why it works so well. How does Atte relationships between words in a sentence)*

And if you're interested in NLP applications in ge some other articles you might like.

1. Beam Search *(Algorithm commonly used by and NLP applications to enhance predictions*
2. Bleu Score (*Bleu Score and Word Error Rate metrics for NLP models*)

Architecture Overview

As we saw in Part 1, the main components of th are:



(Image by Author)

Data inputs for both the Encoder and Decoder, v

- Embedding layer
- Position Encoding layer

The Encoder stack contains a number of Encode
contains:

- Multi-Head Attention layer
- Feed-forward layer

The Decoder stack contains a number of Decod
contains:

- Two Multi-Head Attention layers
- Feed-forward layer

Output (top right) – generates the final output, and contains:

- Linear layer
- Softmax layer.

To understand what each component does, let's
working of the Transformer while we are trainin
translation problem. We'll use one sample of ou
which consists of an input sequence ('You are w
English) and a target sequence ('De nada' in Spa

Embedding and Position Encoding

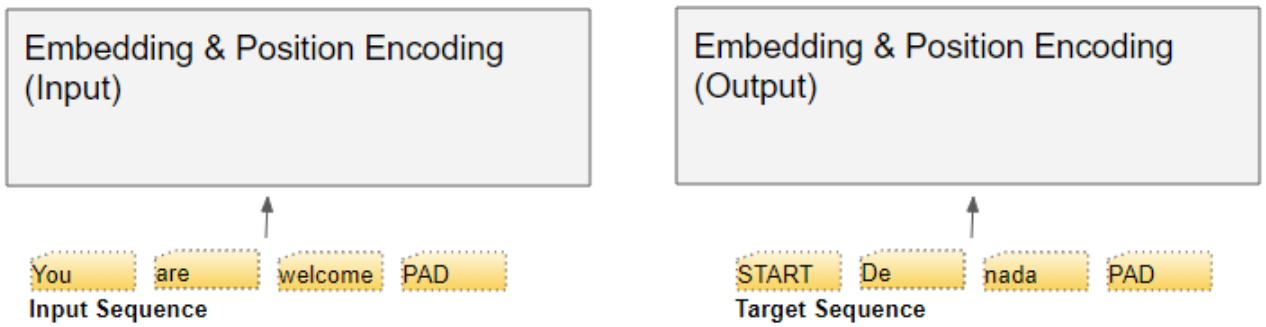
Like any NLP model, the Transformer needs two
each word – the meaning of the word and its po
sequence.

- The Embedding layer encodes the meaning
- The Position Encoding layer represents the
  word.

The Transformer combines these two encodings

Embedding

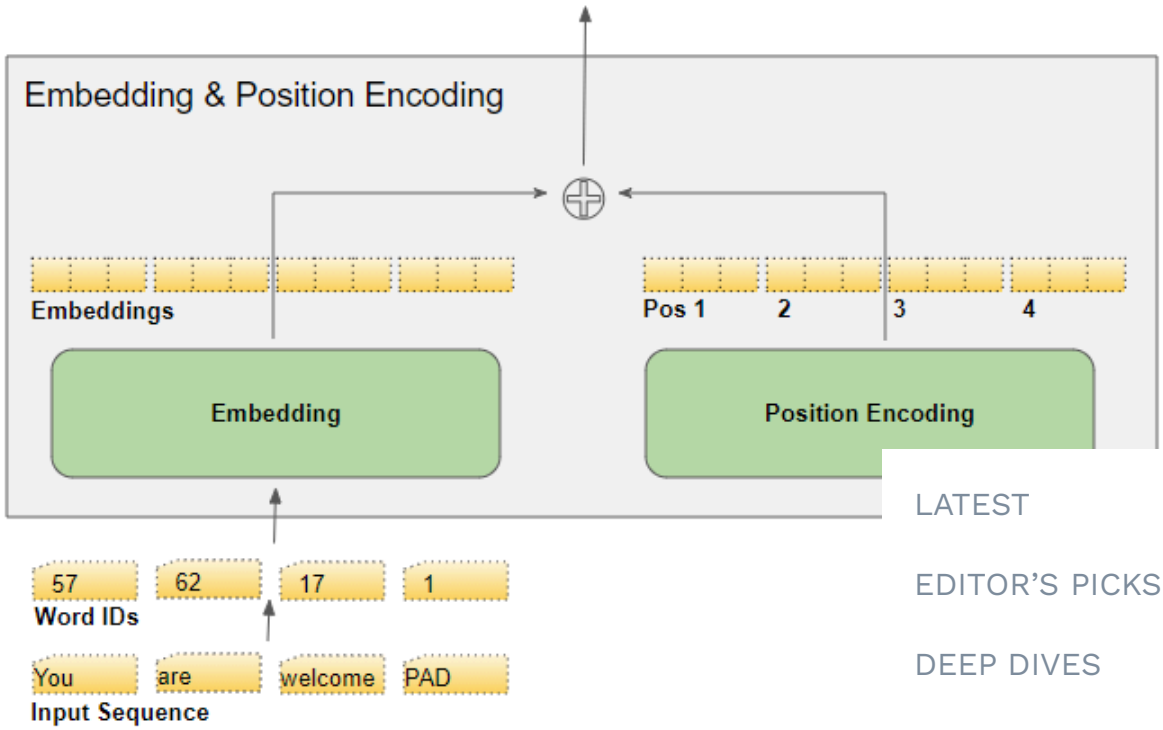
The Transformer has two Embedding layers. The
is fed to the first Embedding layer, known as the
Embedding.



(Image by Author)

The target sequence is fed to the second Embe
shifting the targets right by one position and ins
token in the first position. Note that, during Infe
no target sequence and we feed the output seq
second layer in a loop, as we learned in Part 1. T
called the Output Embedding.

The text sequence is mapped to numeric word I
vocabulary. The embedding layer then maps eac
an embedding vector, which is a richer represen
meaning of that word.



(Image by Author)

Position Encoding

Since an RNN implements a loop where each wo
sequentially, it implicitly knows the position of e

However, Transformers don't use RNNs and all w
sequence are input in parallel. This is its major a
the RNN architecture, but it means that the pos
is lost, and has to be added back in separately.

Just like the two Embedding layers, there are tw
Encoding layers. The Position Encoding is compu
independently of the input sequence. These are
depend only on the max length of the sequence

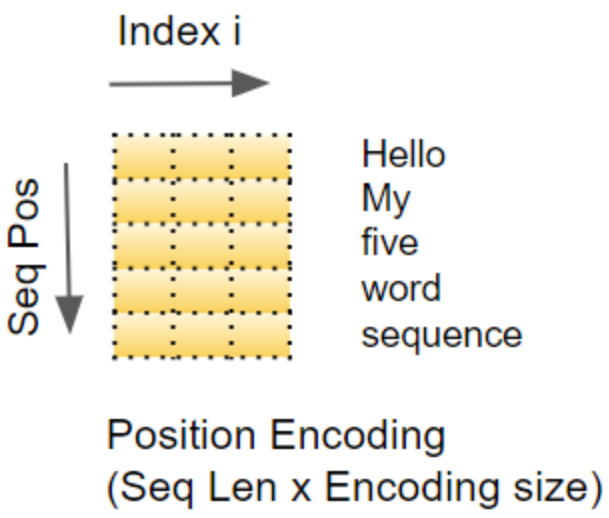
- the first item is a constant code that indica
position

- the second item is a constant code that indicates the second position,
- and so on.

These constants are computed using the formula below, where

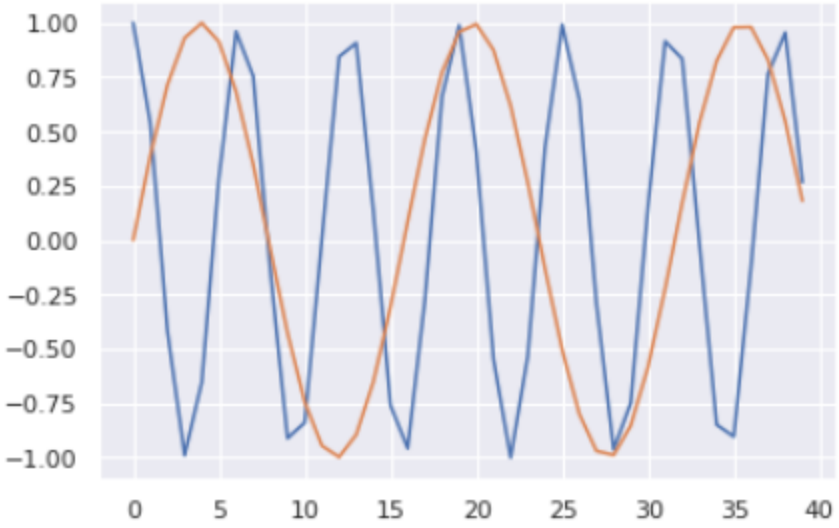
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

- *pos* is the position of the word in the seque
- _d*model* is the length of the encoding vecto embedding vector) and
- *i* is the index value into this vector.



(Image by Author)

In other words, it interleaves a sine curve and a sine values for all even indexes and cos values f indexes. As an example, if we encode a sequenc can see below the encoding values for a few (wo encoding_index) combinations.

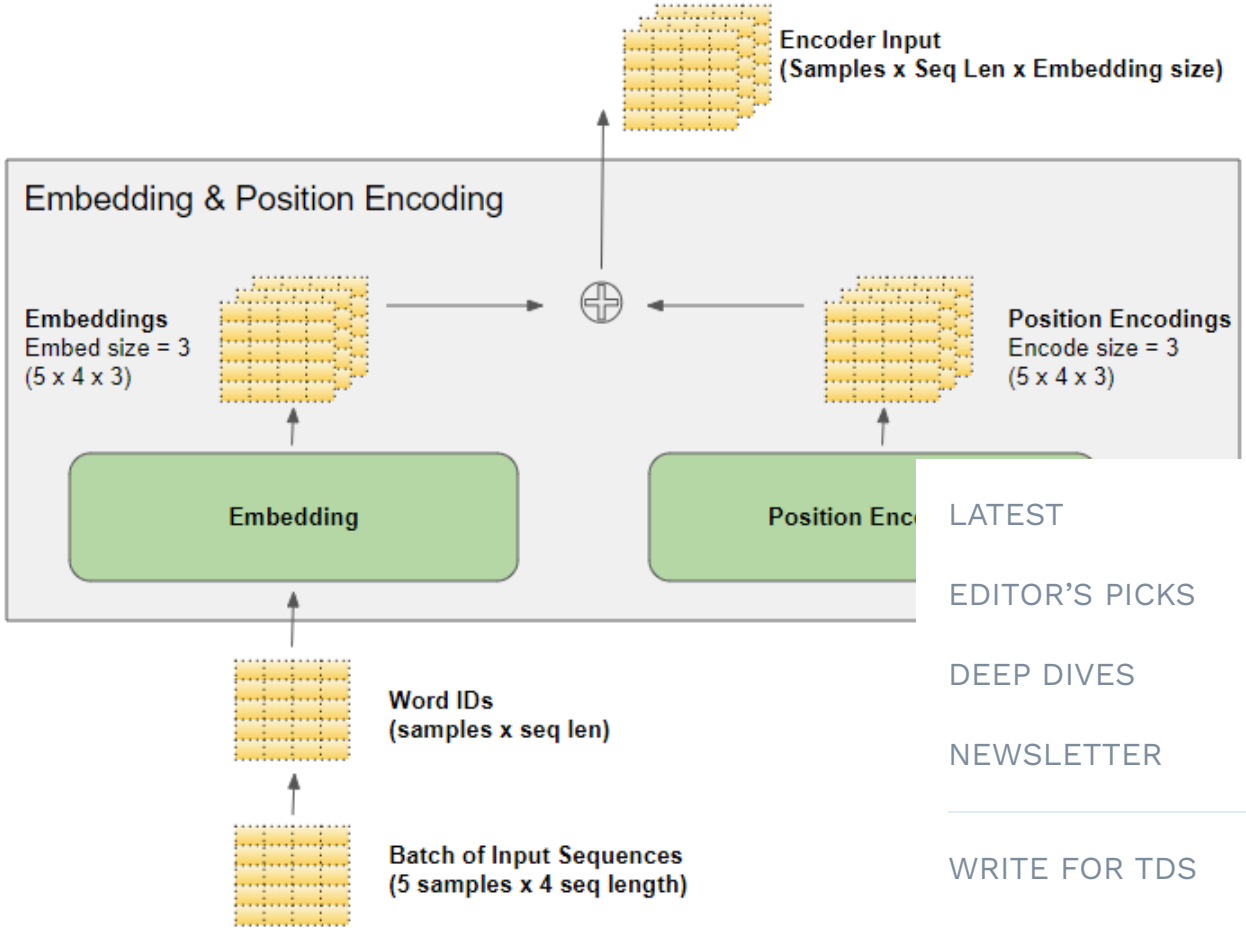


(Image by Author)

The blue curve shows the encoding of the 0th ir
word-positions and the orange curve shows the
1st index for all 40 word-positions. There will be
for the remaining index values.

Matrix Dimensions

As we know, deep learning models process a ba
samples at a time. The Embedding and Position
operate on matrices representing a batch of sec
The Embedding takes a (samples, sequence leng
matrix of word IDs. It encodes each word ID into
whose length is the embedding size, resulting ir
sequence length, embedding size) shaped outpu
Position Encoding uses an encoding size that is
embedding size. So it produces a similarly shape
can be added to the embedding matrix.



(Image by Author)

The (samples, sequence length, embedding size) by the Embedding and Position Encoding layers through the Transformer, as the data flows throu and Decoder Stacks until it is reshaped by the f layers.

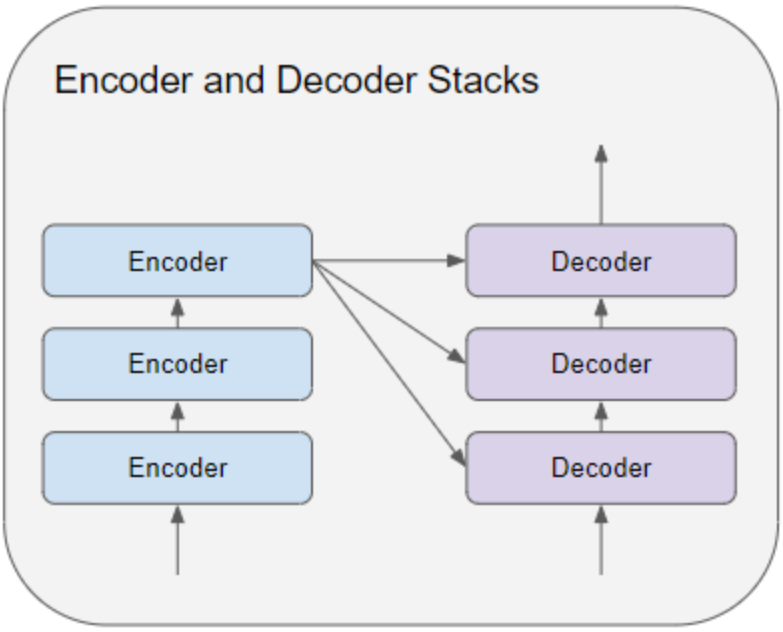
This gives a sense of the 3D matrix dimensions i Transformer. However, to simplify the visualizati we will drop the first dimension (for the sample representation for a single sample.

(Image by Author)

The Input Embedding sends its outputs into the
Similarly, the Output Embedding feeds into the

Encoder

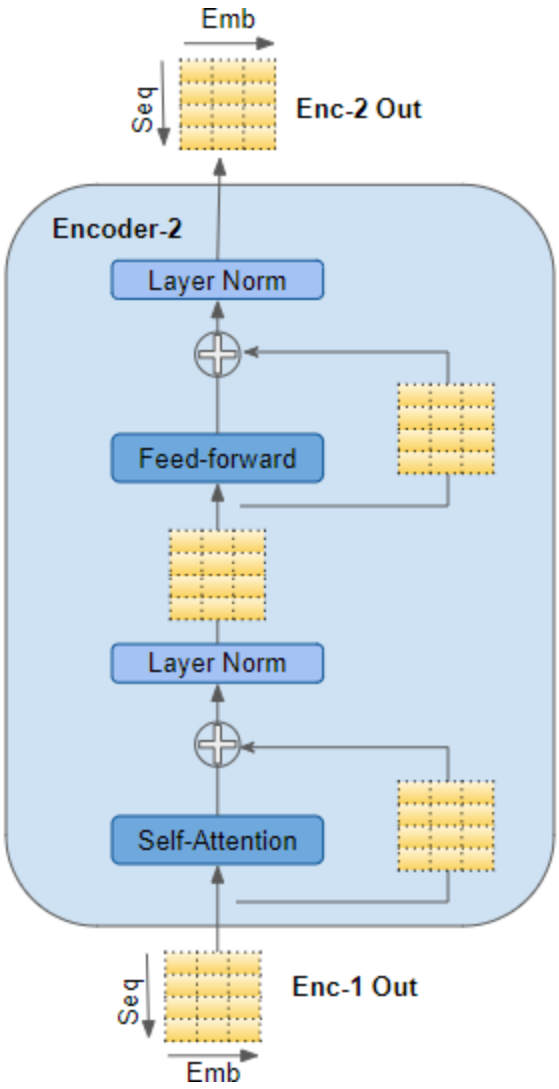
The Encoder and Decoder Stacks consists of se
Encoders and Decoders respectively, connected



(Image by Author)

The first Encoder in the stack receives its input from the Embedding and Position Encoding. The other Encoders in the stack receive their input from the previous Encoder.

The Encoder passes its input into a Multi-head Self-attention layer. The Self-attention output is passed into a Feed-forward layer, which then sends its output upwards to tl



(Image by Author)

Both the Self-attention and Feed-forward sub-l
residual skip-connection around them, followed

Normalization.

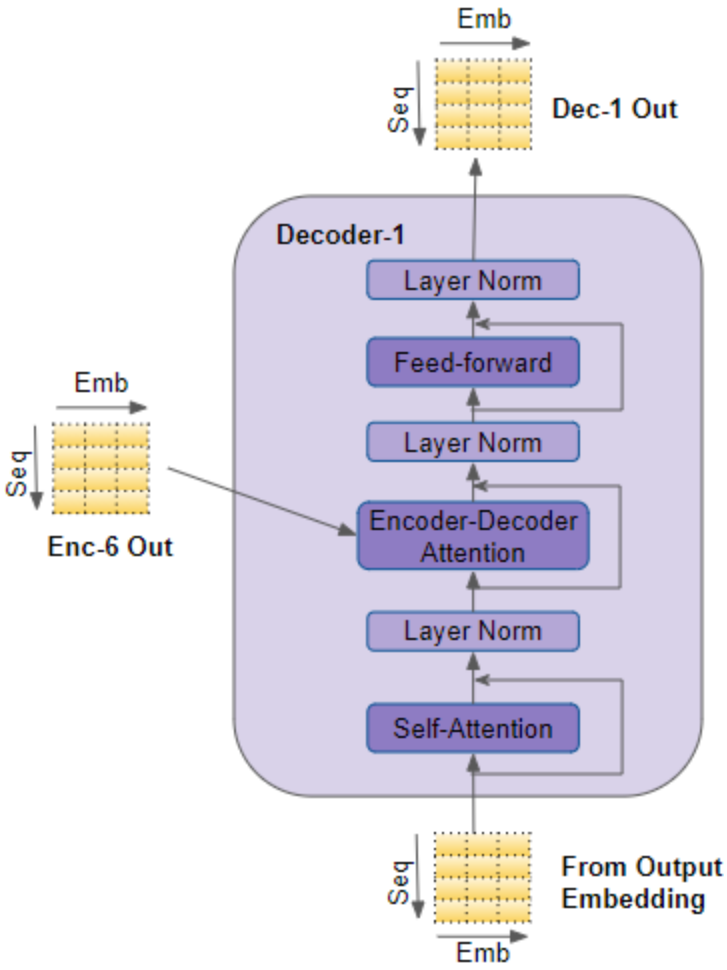
The output of the last Encoder is fed into each Decoder in the Decoder Stack as explained below.

Decoder

The Decoder's structure is very similar to the Encoder's but with a couple of differences.

Like the Encoder, the first Decoder in the stack from the Output Embedding and Position Encod Decoders in the stack receive their input from t Decoder.

The Decoder passes its input into a Multi-head layer. This operates in a slightly different way th Encoder. It is only allowed to attend to earlier p sequence. This is done by masking future positi talk about shortly.



(Image by Author)

Unlike the Encoder, the Decoder has a second M
attention layer, known as the Encoder-Decoder
The Encoder-Decoder attention layer works like
except that it combines two sources of inputs –
attention layer below it as well as the output of
stack.

The Self-attention output is passed into a Feed-
which then sends its output upwards to the nex

Each of these sub-layers, Self-attention, Encode
attention, and Feed-forward, have a residual ski
around them, followed by a Layer-Normalization

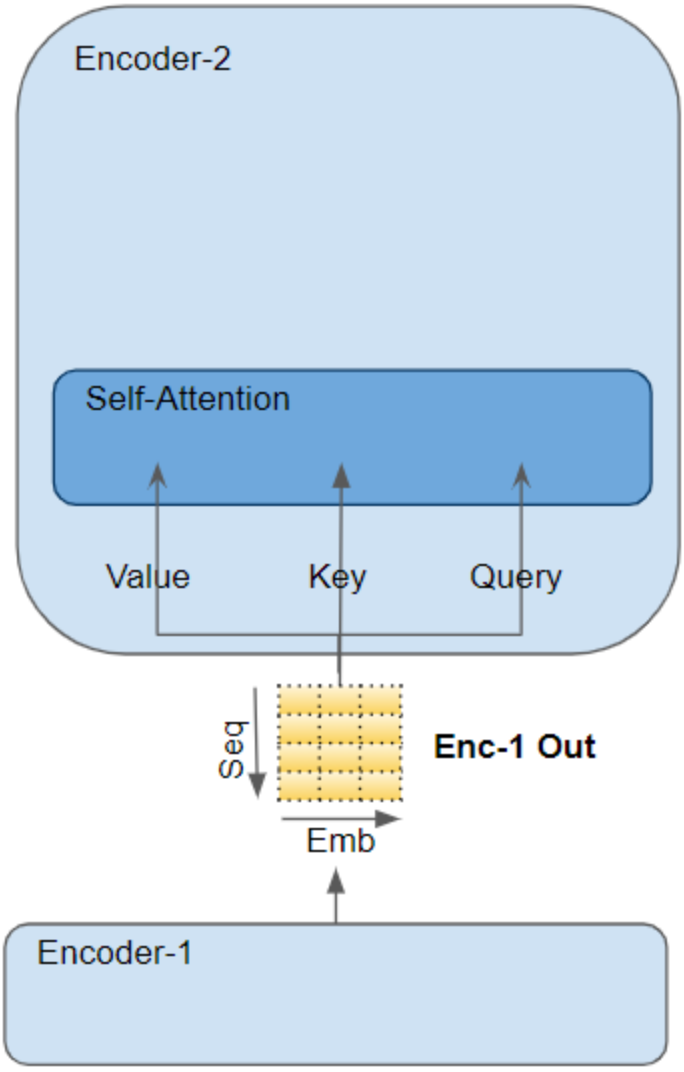
Attention

In Part 1, we talked about why Attention is so important while processing sequences. In the Transformer, Attention is used in three places:

- Self-attention in the Encoder – the input sequence pays attention to itself
- Self-attention in the Decoder – the target se attention to itself
- Encoder-Decoder-attention in the Decoder - sequence pays attention to the input seque

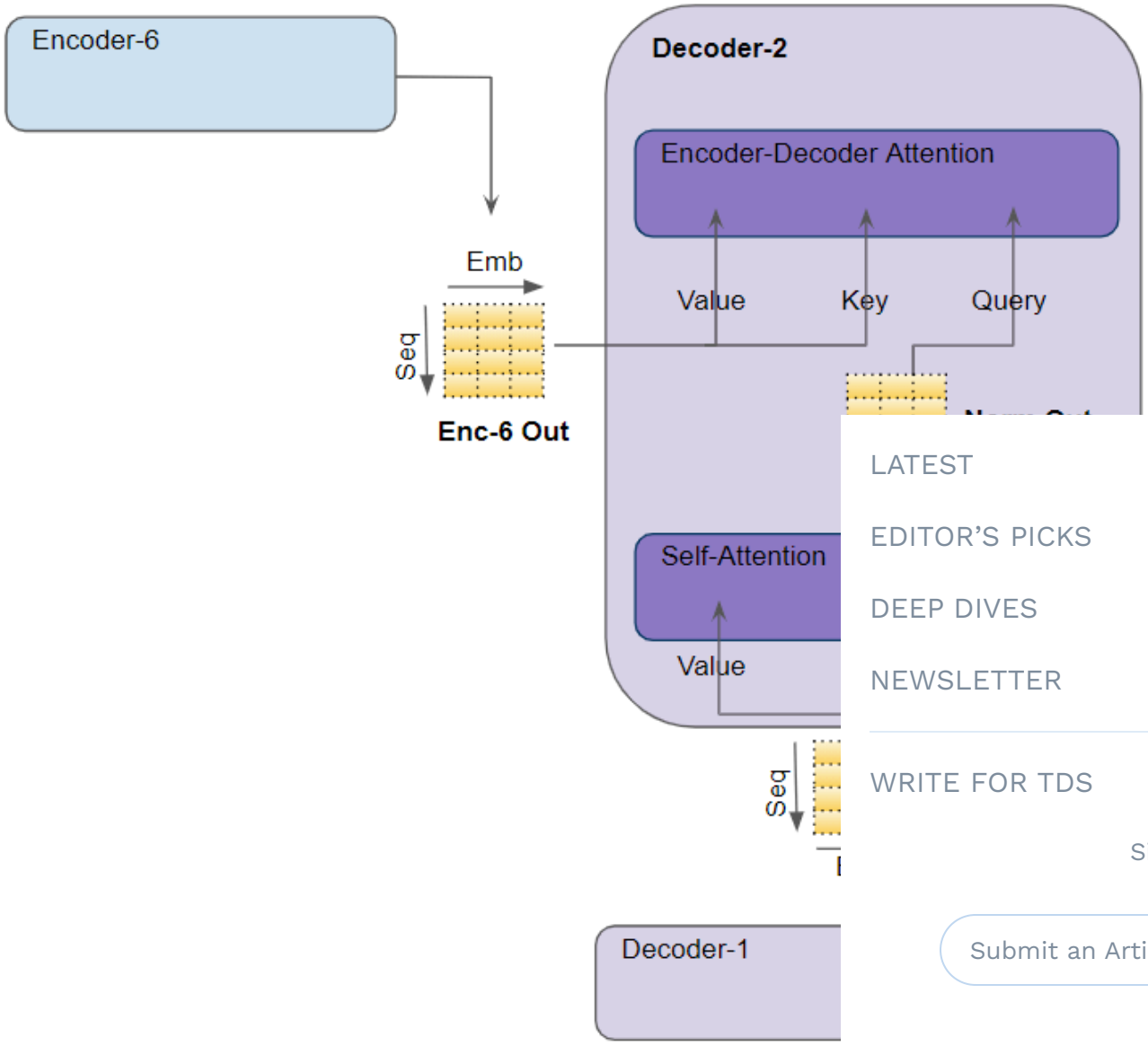
The Attention layer takes its input in the form o parameters, known as the Query, Key, and Value

- In the Encoder's Self-attention, the Encoder to all three parameters, Query, Key, and Valu



(Image by Author)

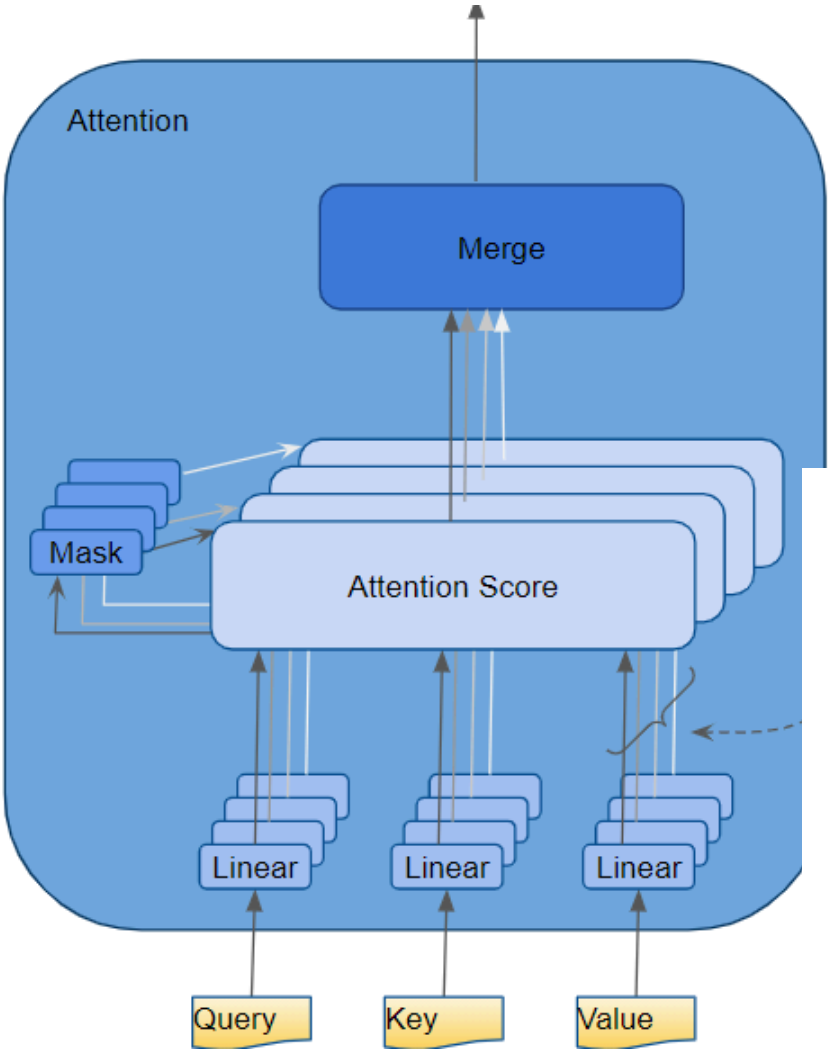
- In the Decoder's Self-attention, the Decoder to all three parameters, Query, Key, and Valu
- In the Decoder's Encoder-Decoder attention the final Encoder in the stack is passed to t parameters. The output of the Self-attentior Norm) module below it is passed to the Que



(Image by Author)

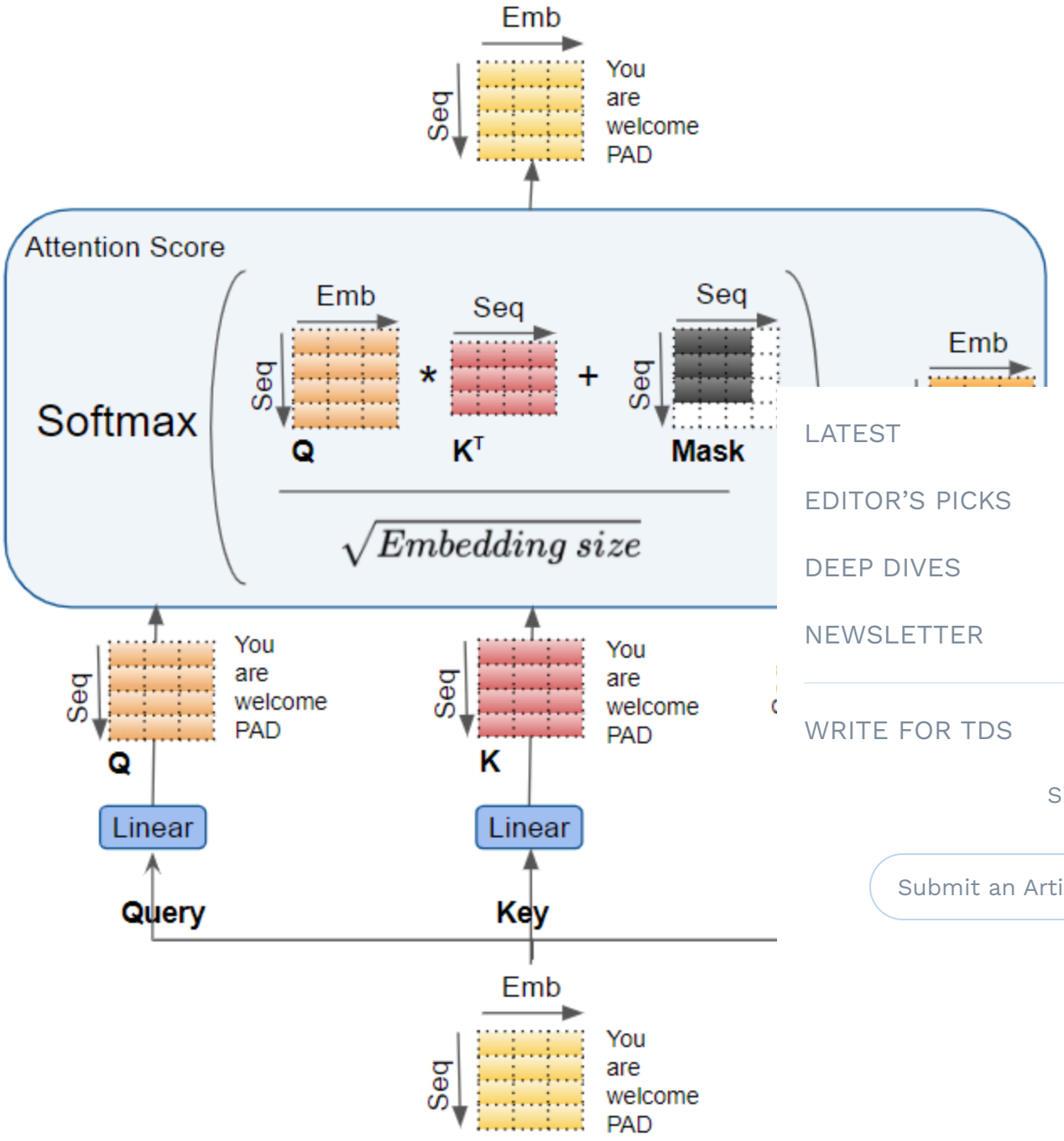
Multi-head Attention

The Transformer calls each Attention processor and repeats it several times in parallel. This is k head attention. It gives its Attention greater pow discrimination, by combining several similar Atte calculations.



(Image by Author)

The Query, Key, and Value are each passed throu
Linear layers, each with their own weights, prod
results called Q, K, and V respectively. These are
together using the Attention formula as shown b
the Attention Score.



(Image by Author)

The important thing to realize here is that the Q
carry an encoded representation of each word i
The Attention calculations then combine each w
other word in the sequence, so that the Attentic
a score for each word in the sequence.

When discussing the Decoder a little while back, we briefly mentioned masking. The Mask is also shown in the Attention diagrams above. Let's see how it works.

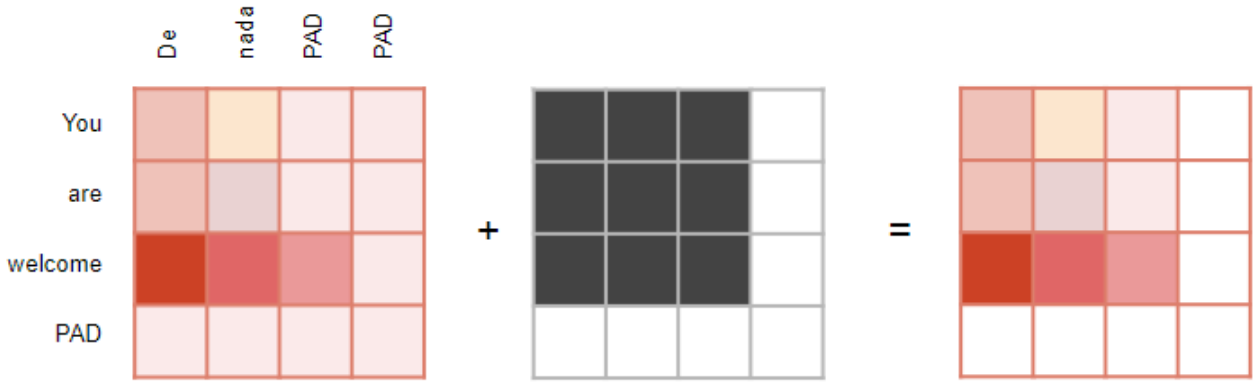
Attention Masks

While computing the Attention Score, the Attention module implements a masking step. Masking serves two

In the Encoder Self-attention and in the Encode attention: masking serves to zero attention outp is padding in the input sentences, to ensure tha contribute to the self-attention. (Note: since inp could be of different lengths they are extended tokens like in most NLP applications so that fixe can be input to the Transformer.)

(Image by Author)

Similarly for the Encoder-Decoder attention.



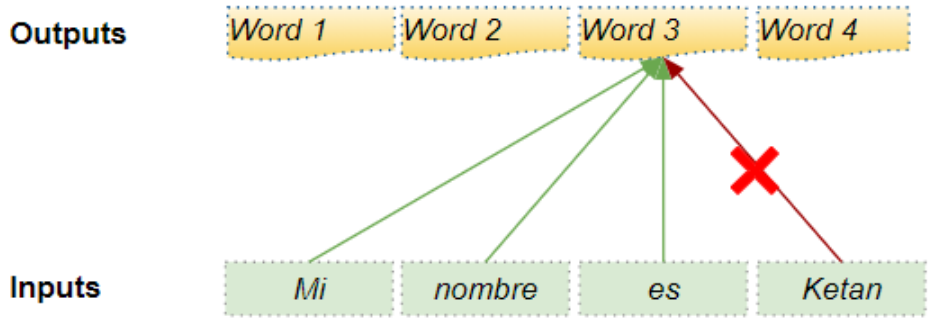
Encoder-Decoder Attention Scores

(Image by Author)

In the Decoder Self-attention: masking serves t decoder from 'peeking' ahead at the rest of the when predicting the next word.

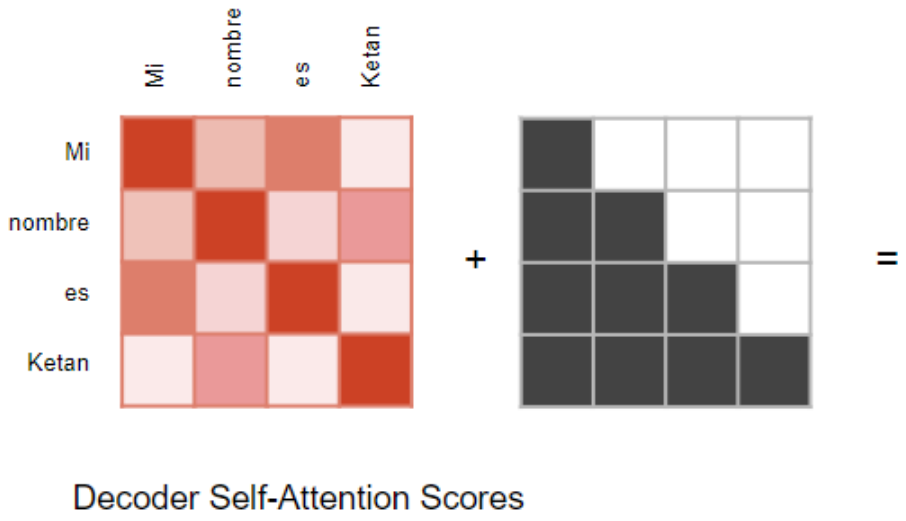
The Decoder processes words in the source seq them to predict the words in the destination se training, this is done via Teacher Forcing, where target sequence is fed as Decoder inputs. There predicting a word at a certain position, the Decc to it the target words preceding that word as we words following that word. This allows the Deco using target words from future 'time steps'.

For instance, when predicting '*Word 3*', the Decc only to the first 3 input words from the target b word '*Ketan*'.



(Image by Author)

Therefore, the Decoder masks out input words t
in the sequence.



(Image by Author)

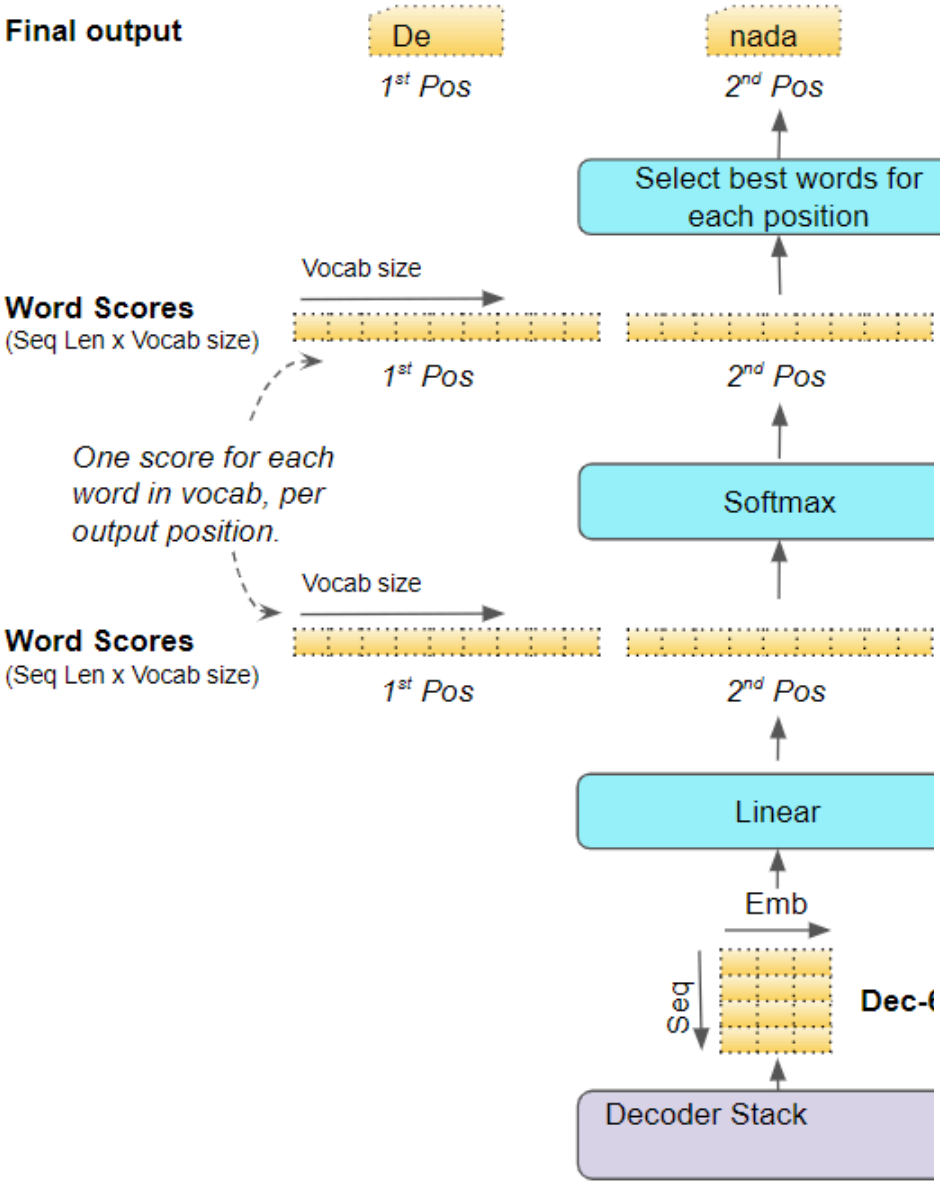
When calculating the Attention Score (refer to tl
showing the calculations) masking is applied to
just before the Softmax. The masked out eleme
squares) are set to negative infinity, so that Soft
values to zero.

Generate Output

The last Decoder in the stack passes its output
component which converts it into the final outp

The Linear layer projects the Decoder vector into Word Scores, with a score value for each unique word in the target vocabulary, at each position in the sentence. For instance, if our final output sentence has 7 words and the target Spanish vocabulary has 10000 unique words, we generate 10000 score values for each of those 7 words. The score values indicate the likelihood of occurrence for each word in the vocabulary in that position of the sentence.

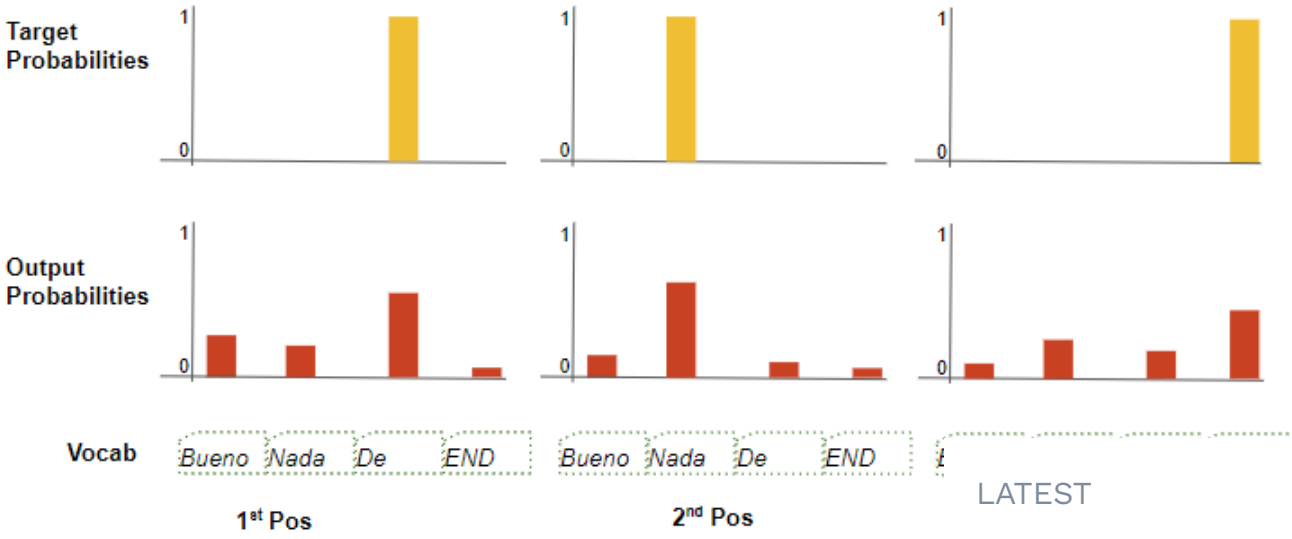
The Softmax layer then turns those scores into
(which add up to 1.0). In each position, we find t
word with the highest probability, and then map
corresponding word in the vocabulary. Those wo
output sequence of the Transformer.



(Image by Author)

Training and Loss Function

During training, we use a loss function such as c
to compare the generated output probability dis
target sequence. The probability distribution giv
of each word occurring in that position.



(Image by Author)

Let's assume our target vocabulary contains just
goal is to produce a probability distribution that
expected target sequence "De nada END".

This means that the probability distribution for
position should have a probability of 1 for "De" v
for all other words in the vocabulary being 0. Sir
"END" should have a probability of 1 for the sec
word-positions respectively.

As usual, the loss is used to compute gradients
Transformer via backpropagation.

Conclusion

Hopefully, this gives you a feel for what goes on
Transformer during Training. As we discussed in
article, it runs in a loop during Inference but mc
processing remains the same.

The Multi-head Attention module is what gives the Transformer its power. In the next article, we will continue our journey and go one step deeper to really understand the details of how Attention is computed.

And finally, if you liked this article, you might also enjoy my other series on Audio Deep Learning, Geolocation Machine Learning, and Image Caption architectures.

Audio Deep Learning Made Simple (Part 1): St Techniques

Leveraging Geolocation Data for Machine Lea Techniques

Image Captions with Deep Learning: State-of Architectures

Let's keep learning!

WRITTEN BY

Ketan Doshi

See all from Ketan Doshi

Artificial Intelligence | Data Science | Deep Learn

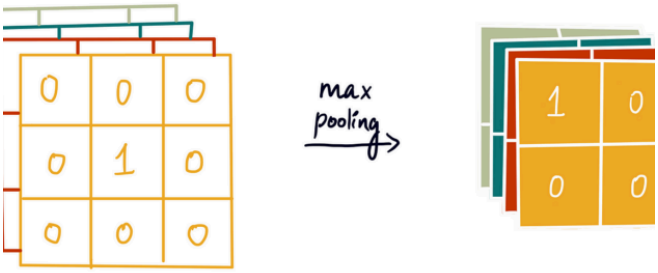
Machine Learning NLP

Share This Article

Towards Data Science is a community publication. Submit your insights to reach our global audience and earn through the TDS Author Payment Progra

Write for TDS

Related Articles



ARTIFICIAL INTELLIGENCE

Implementing Convolutional Neural Networks in TensorFlow

Step-by-step code guide to building a Convolutional Neural Network

Shreya Rao

August 20, 2024 6 min read

ARTIFICIAL INTELLI

What Do Larg "Understand"

A deep dive on th understanding ar LLMs

Tarik Dzekman

August 21, 2024 3

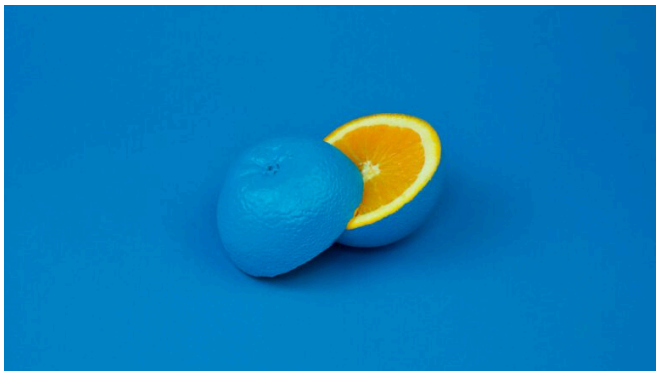
ARTIFICIAL INTELLIGENCE

How to Forecast Hierarchical Time Series

A beginner's guide to forecast reconciliation

Dr. Robert Kübler

August 20, 2024 13 min read



DATA SCIENCE

Hands-on Tim… Detection usi… with Python

Here's how to use… detect signals wi… lines of…

Piero Paialunga

August 21, 2024 1…

MACHINE LEARNING

3 AI Use Cases (That Are Not a Chatbot)

Feature engineering, structuring unstructured data, and lead scoring

Shaw Talebi

August 21, 2024 7 min read

DATA SCIENCE

Solving a Con… Scheduling Pr… Quantum Ann…

Solving the resou… scheduling proble… Wave's hybrid co… model (CQM)

Luis Fernando PÉRE…

August 20, 2024 …

DATA SCIENCE

Back To Basics, Part Uno: Linear Regression and Cost Function

An illustrated guide on essential machine learning concepts

Shreya Rao

February 3, 2023 6 min read