Transformer and BERT from scratch

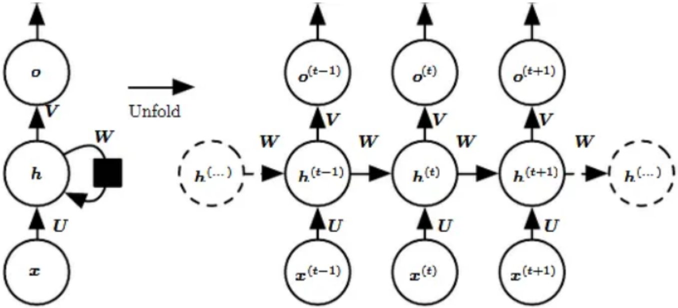
CS3012 Natural Language Processing

Muthu Palaniappan M

Shiv Nadar University Chennai

June 29, 2024

Recurrent Neural Networks



Recurrent Neural Networks

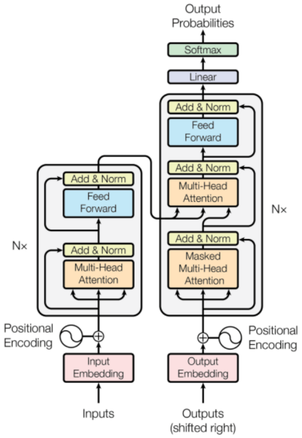
$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}$$

Problems with Recurrent Neural Networks

Problems

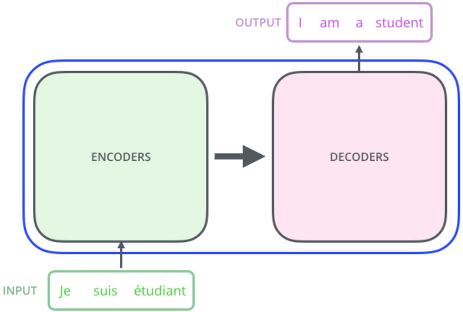
- Slow computation for long sequences
- Vanishing and exploding gradients
- Difficulty in accessing information from long time ago

Transformers



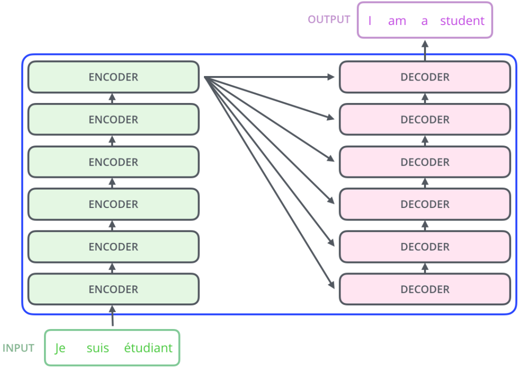
High Level Look

Next Level Look



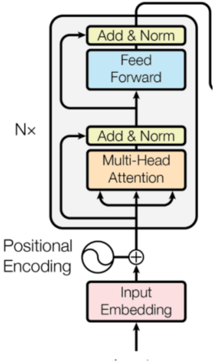
Next Level Look

We have 6 Encoders and Decoders in the stack PS: *6 - Some trail and Error methods*



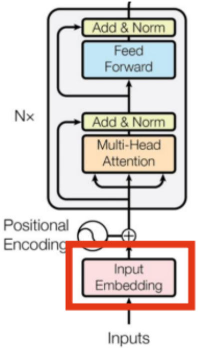
what is Encoder?

Takes the input sequence and generates a contextualized representation for each word/token.



Encoder - Input Embeddings

The abstraction that is common to all the encoders is that they receive a list of vectors each of the size 512



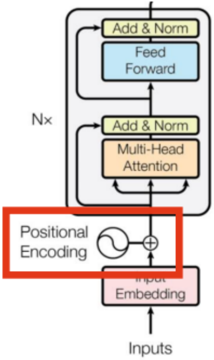
Encoder - Input Embeddings

Original sentence (tokens)	YOUR	CAT	IS	A	LOVELY	CAT
Input IDs (position in the vocabulary)	105	6587	5475	3578	65	6587
Embedding (vector of size 512)	952.207	171.411	621.659	776.562	6422.693	171.411
	5450.840	3276.350	1304.051	5567.288	6315.080	3276.350
	1853.448	9192.819	0.565	58.942	9358.778	9192.819
	...	...	...	...	...	...
	1.658	3633.421	7679.805	2716.194	2141.081	3633.421
	2671.529	8390.473	4506.025	5119.949	735.147	8390.473

Encoder - Input Embeddings

- Neither Word2Vec nor GloVe is used as Transformers are a newer class of algorithms.
- Word2Vec and GloVe are based on static word embeddings while Transformers are based on dynamic word embeddings.
- The embeddings are trained from scratch.

Encoder - Positional Encoding



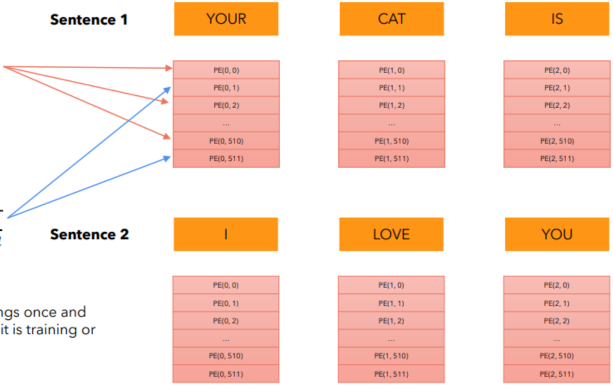
Encoder - Positional Encoding

- We want each word to carry some information about its position in the sentence.
- We want the model to treat words that appear close to each other.
- The Transformer model doesn't inherently learn the order of words.

Encoder - Positional Encoding

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i+1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

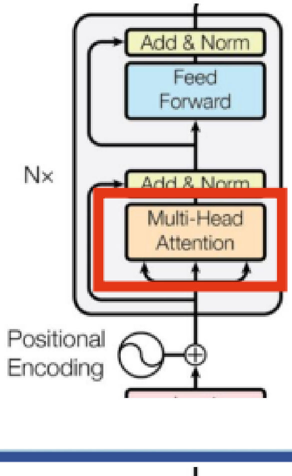


We only need to compute the positional encodings once and then reuse them for every sentence, no matter if it is training or inference.

Encoder - Positional Encoding

Original sentence	YOUR	CAT	IS	A	LOVELY	CAT
Embedding (vector of size 512)	952.207	171.411	621.659	776.562	6422.693	171.411
	5450.840	3276.350	1304.051	5567.288	6315.080	3276.350
	1853.448	9192.819	0.565	58.942	9358.778	9192.819
	...	...	...	...	...	...
	1.658	3633.421	7679.805	2716.194	2141.081	3633.421
	2671.529	8390.473	4506.025	5119.949	735.147	8390.473
	+	+	+	+	+	+
Position Embedding (vector of size 512). Only computed once and reused for every sentence during training and inference.	...	1664.068	...	...	...	1281.458
	...	8080.133	...	...	...	7902.890
	...	2620.399	...	...	...	912.970
	...	...	...	...	...	3821.102
	...	9386.405	...	...	...	1659.217
	...	3120.159	...	...	...	7018.620
	=	=	=	=	=	=
Encoder Input (vector of size 512)	...	1835.479	...	...	...	1452.869
	...	11356.483	...	...	...	11179.24
	...	11813.218	...	...	...	10105.789
	...	...	...	...	...	...
	...	13019.826	...	...	...	5292.638
	...	11510.632	...	...	...	15409.093

Encoder - Attention



What is Self-Attention?

- **Self-Attention allows the model to relate words to each other.**
- In this simple case we consider the sequence length seq $=$ 6 and dmodel $=$ dk$=$ 512.
- The matrices **Q, K and V are just the input sentence.**

How to compute Self-Attention?

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

How to compute Self-Attention?

$$\text{softmax}\left(\frac{\underset{(6,\,512)}{Q} \times \underset{(512,\,6)}{K^T}}{\sqrt{512}}\right) =$$

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

How to compute Self-Attention?

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

Self Attention

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

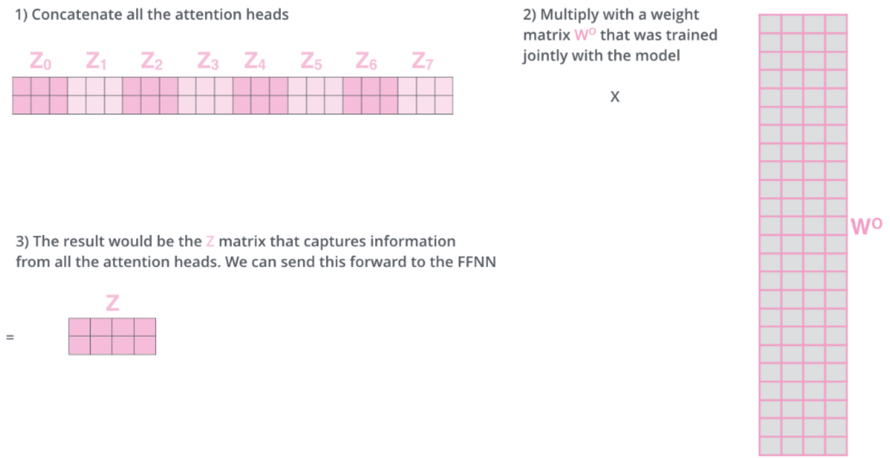
(6, 6)

X **V** (6, 512) = **Attention** (6, 512)

Multi Attention

- The self-attention mechanism is typically employed in a multi-head fashion, where multiple attention heads operate in parallel.
- Each attention head learns different attention patterns, allowing the model to capture different aspects of the input sequence.
- The outputs of multiple attention heads are concatenated and linearly transformed to produce the final attention representation.

Multi Attention



What is Query,Key and Value?

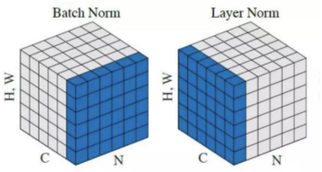
Query = 'Love'

KEY	VALUE
Terror	Surya
Romantic	Lohith
Horror	M*thu

Layer Normalization?

Batch of 3 items

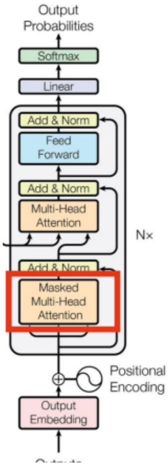
ITEM 1	ITEM 2	ITEM 3
50.147	1242.223	9.370
3314.825	688.123	4606.674
...	...	...
...	...	...
8463.361	434.944	944.705
8.021	149.442	21189.444
μ_1	μ_2	μ_3
σ_1^2	σ_2^2	σ_3^2



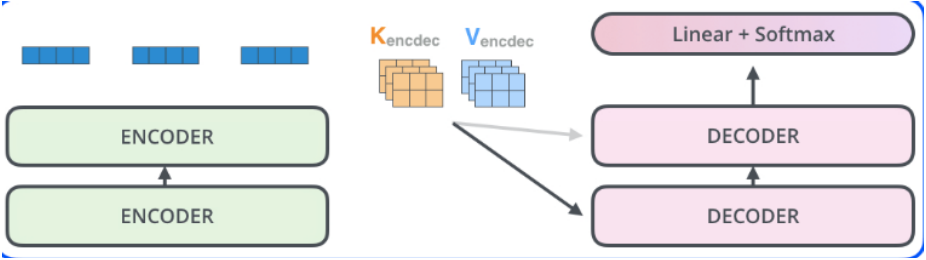
$$\hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

We also introduce two parameters, usually called **gamma** (multiplicative) and **beta** (additive) that introduce some fluctuations in the data, because maybe having all values between 0 and 1 may be too restrictive for the network. The network will learn to tune these two parameters to introduce fluctuations when necessary.

Decoder


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Positional
Encoding
Output
Embedding

What goes to Decoder?



Masked Multi-Head Attention

Our goal is to make the model causal: it means the output at a certain position can only depend on the words on the previous positions. The model **must not** be able to see future words.

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	~~0.119~~	~~0.134~~	~~0.148~~	~~0.179~~	~~0.152~~
CAT	0.124	0.278	~~0.201~~	~~0.128~~	~~0.154~~	~~0.115~~
IS	0.147	0.132	0.262	~~0.097~~	~~0.218~~	~~0.145~~
A	0.210	0.128	0.206	0.212	~~0.119~~	~~0.125~~
LOVELY	0.146	0.158	0.152	0.143	0.227	~~0.174~~
CAT	0.195	0.114	0.203	0.103	0.157	0.229

The Final Linear and Softmax Layer

- The decoder stack outputs a vector of floats. How do we turn that into a word? That's the job of the final Linear layer which is followed by a Softmax Layer.
- The Linear layer is a simple fully connected neural network that projects the vector produced by the stack of decoders, into a much, much larger vector called a **logits** vector.
- The softmax layer then turns those scores into probabilities (all positive, all add up to 1.0). The cell with the highest probability is chosen, and the word associated with it is produced as the output for this time step.

Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers

- BERT's architecture is made up of layers of encoders of the Transformer model.
- BERT's key technical innovation is applying the bidirectional training of Transformer.
- **BASE**
  - 12 encoder layers
  - The size of the hidden size of the feedforward layer is 3072
  - 12 attention heads
- **LARGE**
  - 24 encoder layers
  - The size of the hidden size of the feedforward layer is 4096
  - 16 attention heads

Why do we need bi-directional - Left Context?

Fan 1: Are you ready for tonight's match?
Fan 2: Absolutely! It's RCB vs CSK, always an electrifying contest.
Fan 1: I hope RCB batting lineup fires today. We need those big runs!
Fan 2: Definitely. And CSK bowlers need to contain the opposition early on.

Why do we need bi-directional - Right Context?

Imagine there's a kid who just boke his mom's favorite necklace. The kid doesn't want to tell the truth to his mom, so he decides to make up a lie.

So, instead of saying directly: "Your favorite necklace has broken"

The kid may say: "Mom, I just saw the cat playing in your room and your favorite necklace has broken."

Combination of Left and Right Context

- As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional.
- This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

Combination of Left and Right Context

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

(10, 10)

BERT pre-training

The primary objective of BERT pre-training is to learn general-purpose language representations that **capture rich semantic and syntactic information from large text corpora**

BERT uses two training strategies

- Masked Language Model (MLM)
- Next Sentence Prediction (NSP)

MLM - Training

- Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a *[MASK]* token.
- The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.
- The model must predict the right word given the left and right context.

MLM - Training

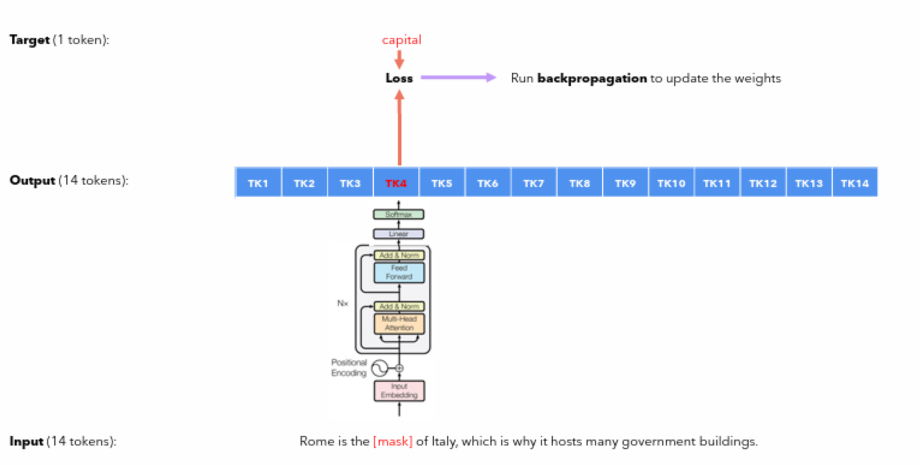
Rome is the capital of Italy, which is why it hosts many government buildings.

Randomly select one or more tokens and replace them with the special token **[MASK]**

Rome is the [MASK] of Italy, which is why it hosts many government buildings.

capital

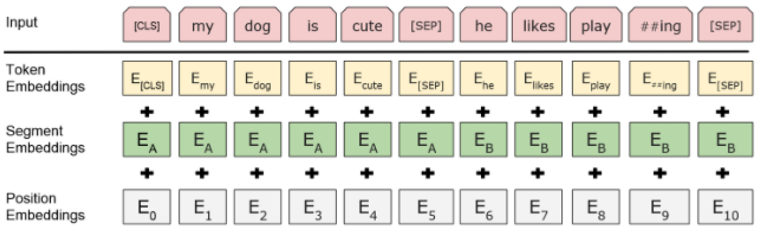
MLM - Training



NSP - Next Sentence Prediction

- The model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.
- During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence

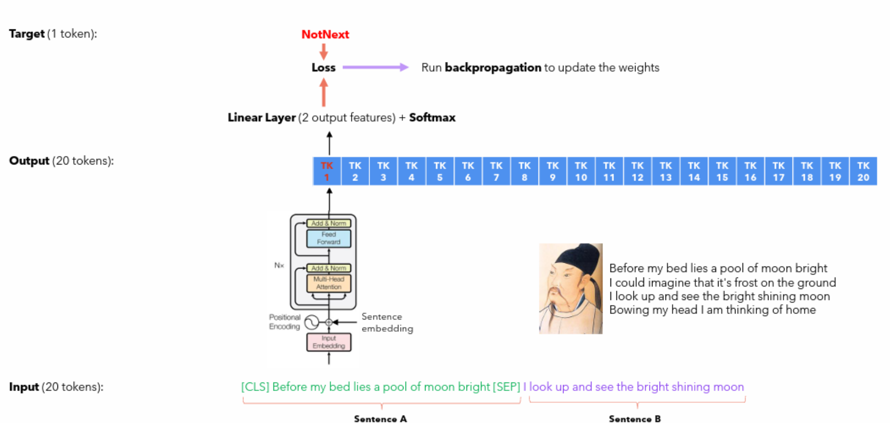
Embedding in BERT: Segment



Segment Embedding

- A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
- A sentence embedding indicating Sentence A or Sentence B is added to each token.
- A positional embedding is added to each token to indicate its position in the sequence.

NSP: Training



Why [CLS] token?

- The [CLS] token always interacts with all the other tokens, as we do not use any mask.
- So, we can consider the [CLS] token as a token that "captures" the information from all the other tokens.

	[CLS]	Before	my	bed	lies	a	pool	of	moon	bright
[CLS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

Fine-Tuning BERT