Seq2Seq and Attention

Tudor Andrei Dumitrascu Student Nr. 2682776

May 07, 2021

Disclaimer

Both papers are focused on machine translation, but they could be used for other tasks that deal with sequential data.

BLEU Score [1]

- Bilingual evaluation understudy.
- Used for machine translation evaluation.
- There are multiple correct translations for a sentence
- If it's pretty close to the given reference sentence, that were created by a human.
- The higher, the better.

[1] Kishore Papineni et. al., BLEU, 2001

Sequence to sequence (2014)

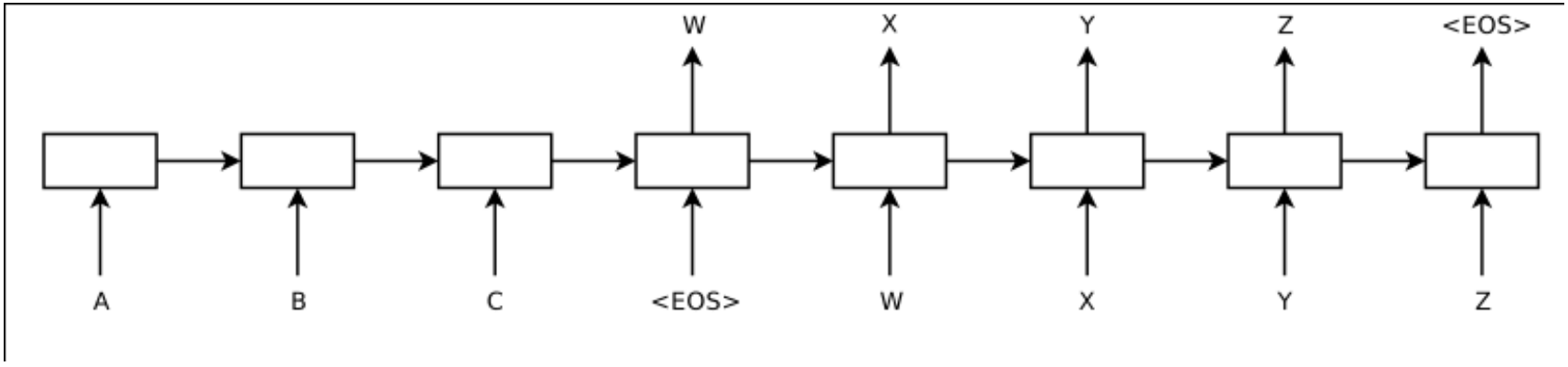

Figure 1: Model Architecture

- Encoder (RNN) that reads the input seq
- Decoder (RNN) that generates the output
- Beam Search decoding, takes the best N outputs, with the highest probabilities.

Architecture (cont.)

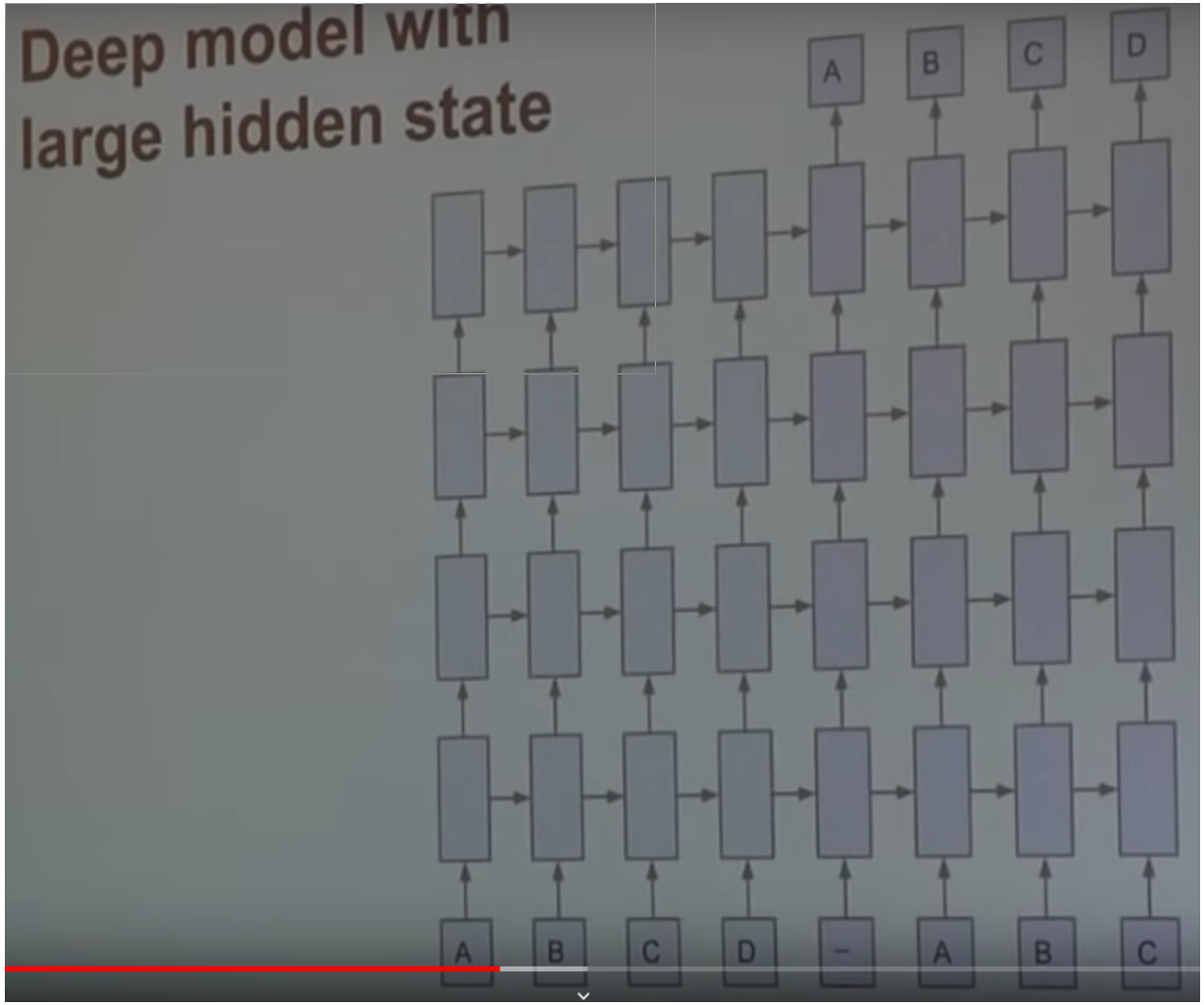

Figure 2: Detailed Architecture[2]

[2](https://www.youtube.com/watch?v=-uyXE7dY5H0)

Model

- multi layer LSTM:
  - LSTM solve the issue of vanishing gradients;
- maps the input seq to a vector of fixed dimension
  - the hidden state of the last LSTM of the encoder acts as an embedding vector;
- reversing the word order of the inputs -> better performance.

Implementation

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

Figure 3: Training objective, S = Source sentence, T = translation

$$\hat{T} = \arg\max_{T} p(T|S)$$

Figure 4: Output

Training

- Easy to train
- 160,000 Input Vocabulary
- 8 GPU machine
- 10 days of training
- WM14 English-French dataset
  - 36M sentence pairs

Results

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	~45

Figure 5: Result on the test set

Long sentences

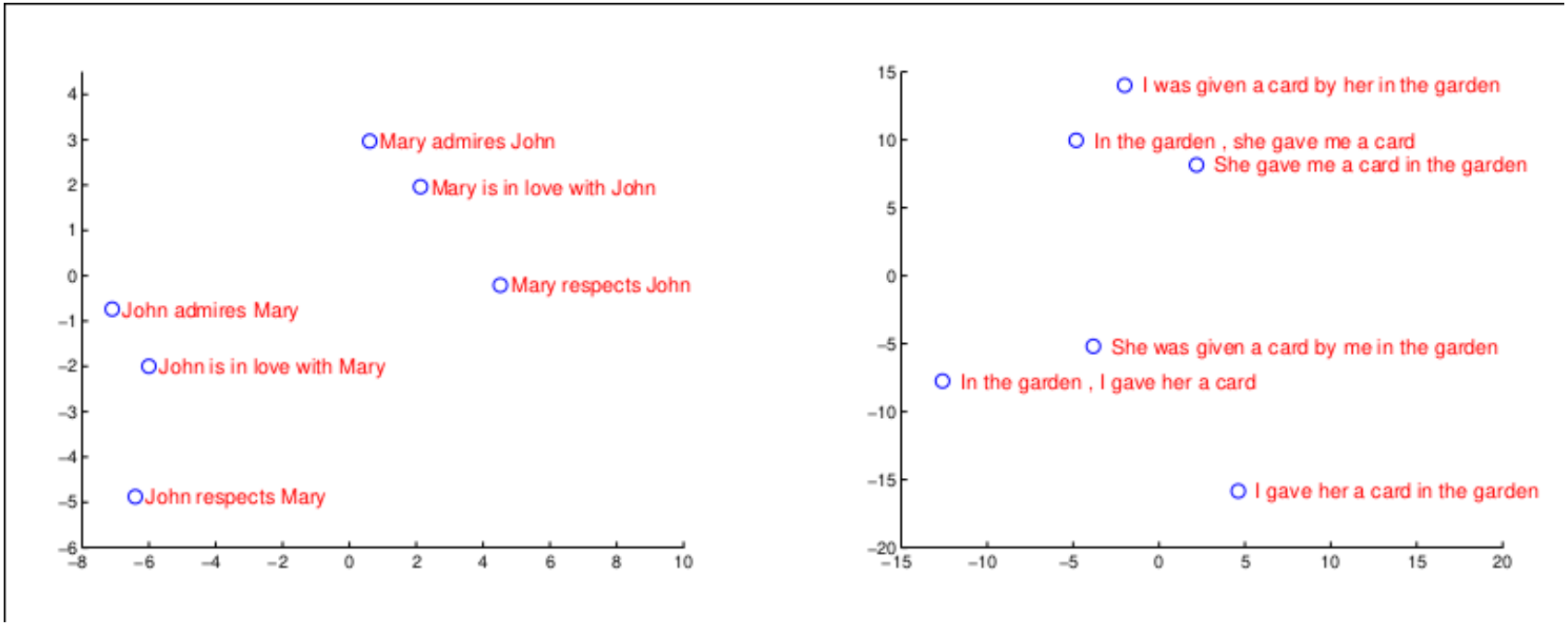

Figure 6: Long sentences

Long sentences

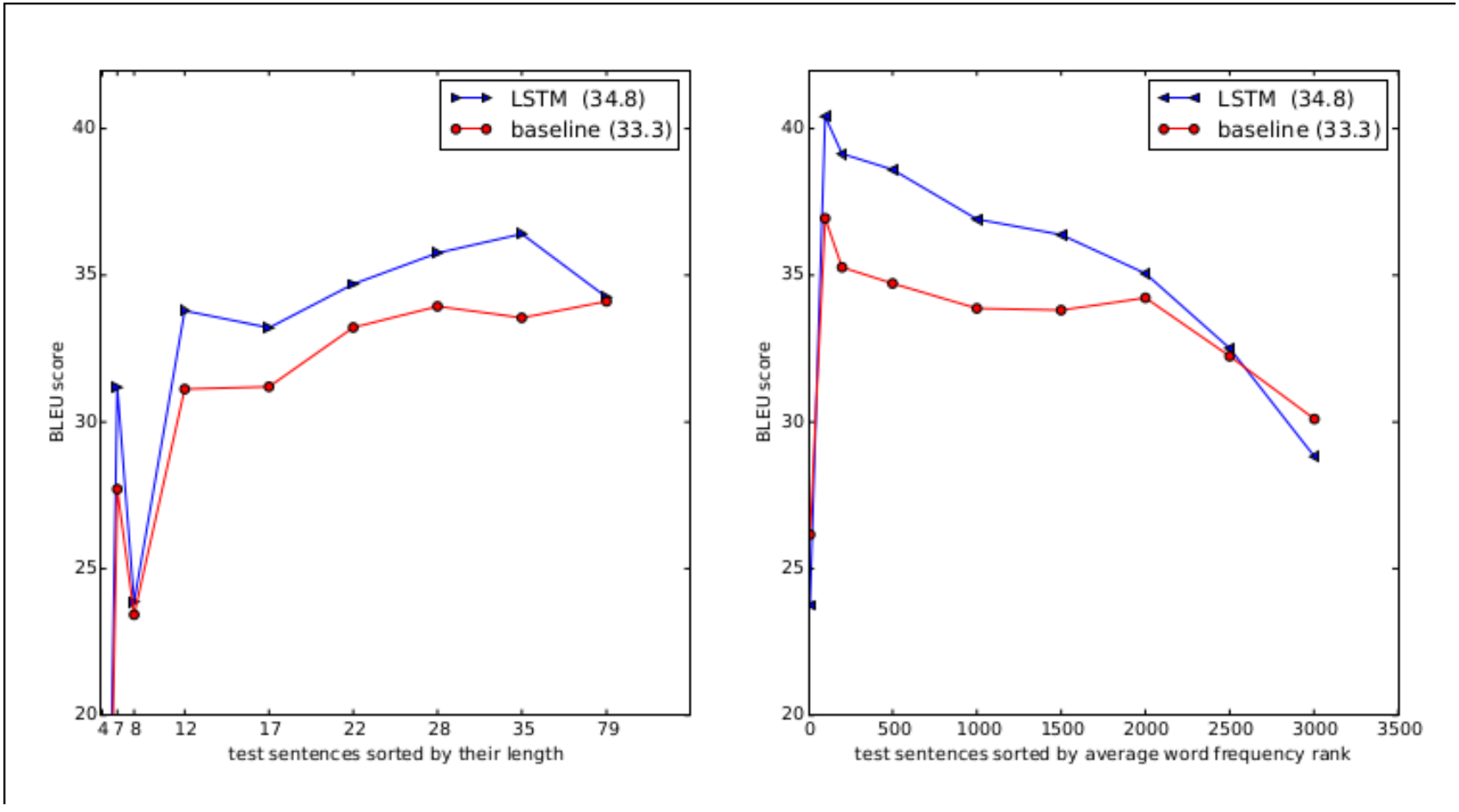

Figure 7: Performance over sentence length

Conclusion

- Simple approach to solve a complex problem.
- Better than the Statistical Models.
- The model does well on long sentence, but the higher the freq of rare words, the worse it performs.

Attention and Transformers (2017)

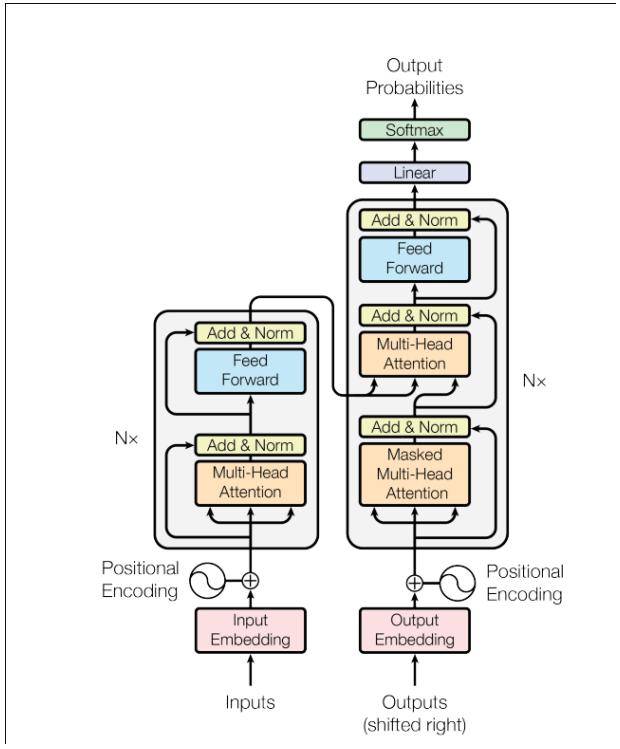

Figure 8: Transformer

Encoding Layer + Positional Encoding

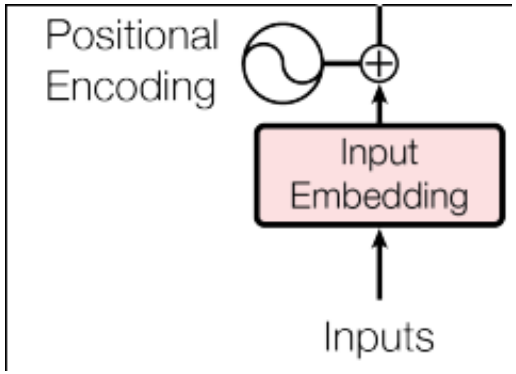

Figure 9: embeddings

- Transform strings to numbers that the algorithm can read
- Because the whole sentence is fed to the network, the order of words is lost, therefore some of the meaning is lost.

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

Figure 10: Positional Embeddings

Multi-Head attention layer

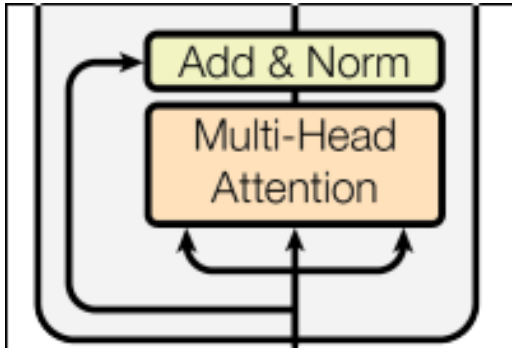

Figure 11: Multi-Head Attention layer

- Uses multiple attention layers;
- Attention: a vector for each word showing which word from sentence is relevant.

Scaled Dot-product Attention

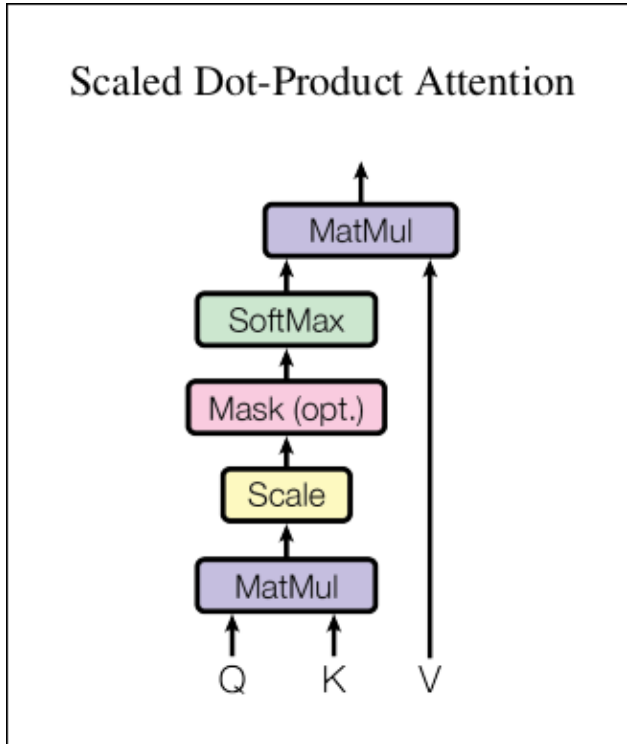

Figure 12: Scaled Dot Product

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

Figure 13: Attention Formula

Scaled Dot-product Attention

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

Figure 14: Attention Formula

- Q = the word that we want to calculate the attention for
- K = keys of the previously generated words
- V = values of the previously generated words
- $1/\sqrt{(d_k)}$ = scaling factor. If d_k is large, the dot product has larger magnitude (Q * K).

Multi-Head attention layer (overview)

- Multi head attention: used to reduce the attention on the same word.

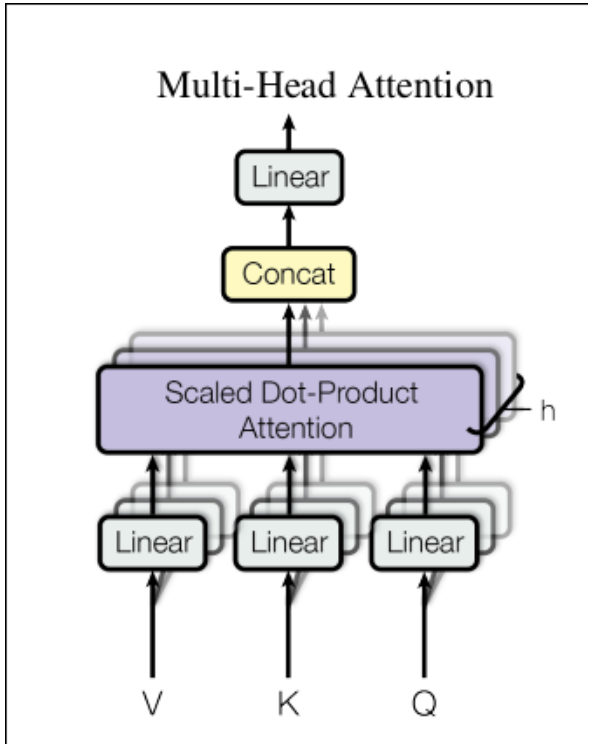

Figure 15: Multi head attention layer

Feed Forward network + Add & Norm

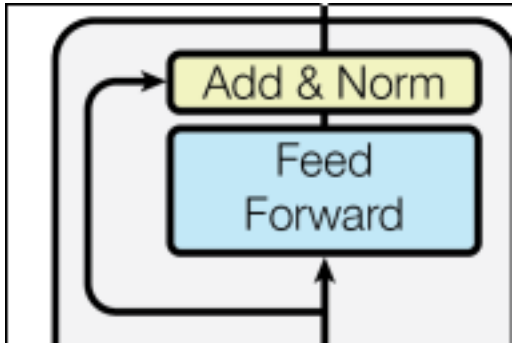

Figure 16: Feed Forward network

- FFN is applied to each attention vector, used for changing the shape of the attention vector.
- Layer normalization.

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Figure 17: Feed Forward Networks

Output embedding

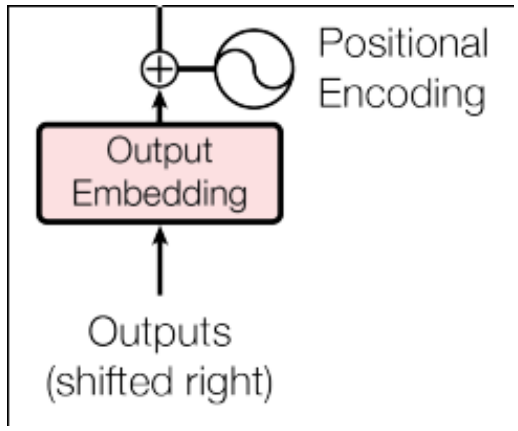

Figure 18: Output Embedding

- Used for faster convergence, at training time.
- Same as input embedding. For MT, the embedding is done in the target language.

Masked multi head attention

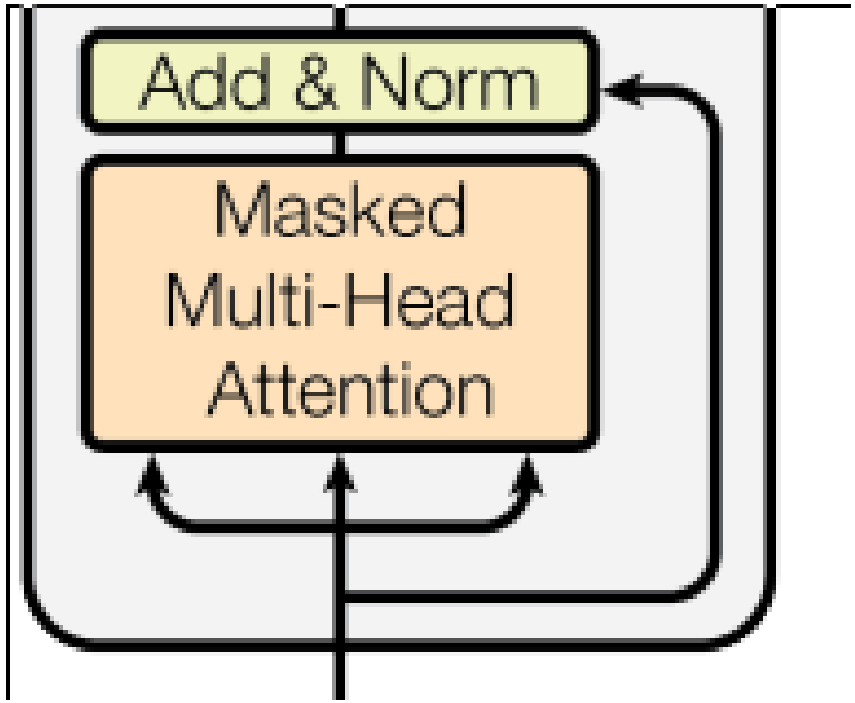

Figure 19: Masked Multi-Head Attention Layer

- So that the model doesn't "cheat", and look ahead in the future words, the words, that haven't been yet "seen" are masked out [3]

[3] The attention of the 1st word is 1, the attention of the 2nd word is among the 1st and itself, the attention of the 3rd word is among the 1st, 2nd and itself etc.

Encoder Decoder Attention (the middle block in the Decoder)

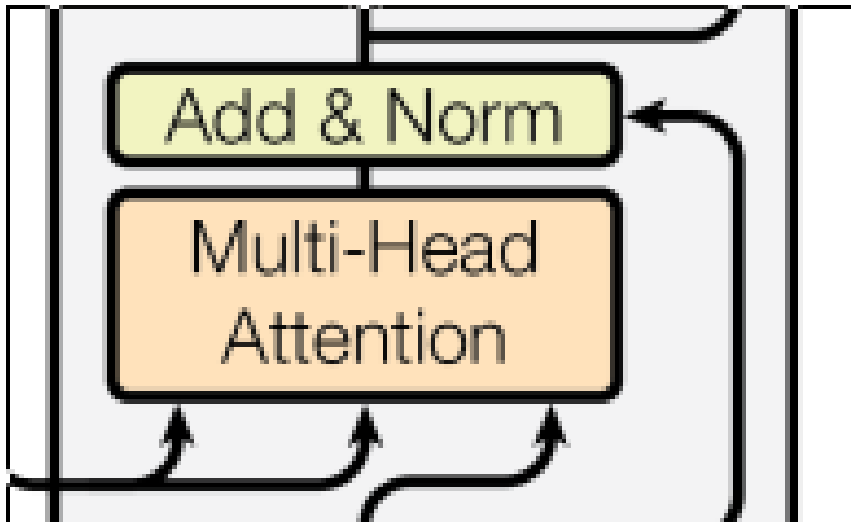

Figure 20: Encoder/Decoder Attention

- Combines both languages, and also has the attention for each word.
- This is where the translation takes place

Linear & Softmax

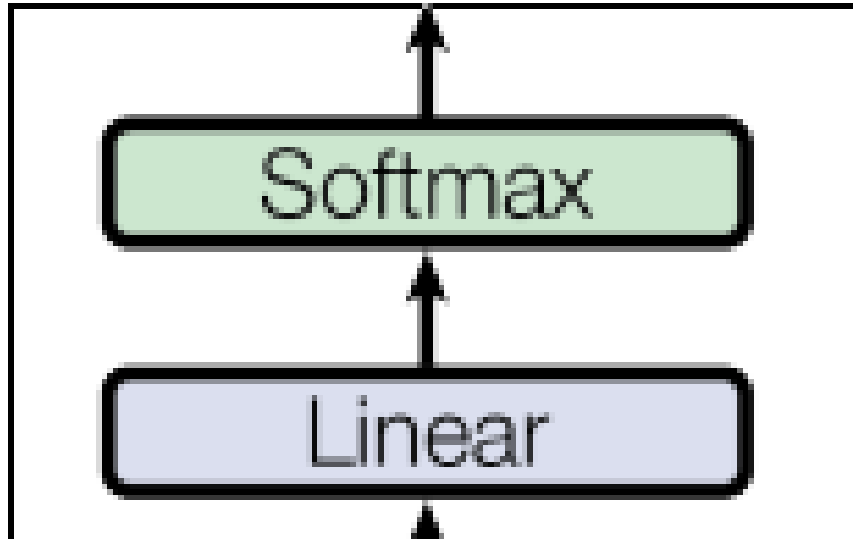

Figure 21: Output layers

- Feed Forward Layer expands the dimension of the attention vector to the length of the target language vocabulary.
- Probability of the word, over the entire target language vocabulary.

Training

- WMT14 English-German dataset
  - 4.5M sentence pairs
- WM14 English-French dataset
  - 36M sentence pairs
- The big models were trained for 3.5 days
- Diminishing Learning rate

$$lrate = d_{model}^{-0.5} \cdot (step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

Results

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Figure 22: results_transformer

- EN-DE - SOTA 28.4 BLEU Score
- EN-FR - 41.0 High Score for low computational cost

Advantages

- Faster than RNN or LSTM
- Faster to train than recurrent layer or convolution layers