Summary of 'Attention is all you need' paper

Abstract:

Sequence Transduction models are the models which convert input sequence to output sequence (input and output sequence could be text, speech audio,images). They are used in language translation, speech processing and computer vision.

These sequence models are either recurrent neural networks or convolutional neural networks which include an encoder and a decoder. Other best performing models included an encoder and decoder connected through an attention mechanism. This paper presents Transformer, a new network architecture which relies on attention mechanism without depending on recurrence and convolutional properties

BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of machine translation by comparing the output of a machine translation system to one or more reference translations.

This paper performed machine translation experiments using the Transformer model and used BLEU score for evaluating performance. It achieved the following

1. 28.4 BLEU score for WMT 2014 English-German Translation task. This performance surpassed existing benchmarks including ensemble of 2 models by 2 BLEU score
2. 41.8 BLEU score for WMT 2014 English-French Translation task. This is a new state of art performance with a single model which was achieved by training the model for 3.5 days on 8 GPUs
3. Training costs for this model are a fraction of training costs incurred for previous models
4. It could generalize well for other tasks. It could perform 'English grammar understanding given a sentence' task well with limited data or more data

1 Introduction:

Recurrent models such as RNNs,LSTMs and GRUs are considered as the state of art approaches in sequence modeling(e.g. time series analysis) ,language translation and language modeling(predict the likelihood of a sequence of words, which is crucial for applications like speech recognition and text generation) tasks. Many efforts were made to push the boundaries of encoder, decoder architectures and recurrent language models

Recurrent models process input and output sequences one element (or symbol) at a time, sequentially.
They generate hidden states,each one denoted by 'ht' that aligns with the position of input symbol xt . Each hidden state ht is a function of previous hidden state ht-1 and current input xt

This inherent usage of serialization in recurrent models prevents from using the parallelization within training examples. Parallelization is highly useful for long sequences because serialization needs whole sequences of data samples to be used in batches which is difficult due to memory constraints

In recurrent models, computational efficiency could be improved by factorization tricks(decomposing bigger calculations into smaller ones) and conditional computation(it means selectively activating certain parts of the model based on the input or context). The latter improved model performance as well. The constraint of sequential computation(not allowing parallelization) still remains.

Attention mechanisms which were integrated into many sequence modeling and transduction(translation and language modeling) models highly helped them to achieve greater performance in various tasks modeling various dependencies.Traditional models often struggle to connect distant elements in a sequence (e.g., linking the subject of a sentence to a verb that appears much later). Attention allows the model to consider any part of the input, regardless of how far apart the relevant parts are. Except for a few cases, attention mechanisms are commonly used with recurrent models.

Transformer model does not use recurrence but it completely depends on the attention mechanism to draw global dependencies between input and output.

Transformer achieved state of the art performance (allowing high parallelization) with a training duration of just 12 hours using 8 P100 GPUs

2 Background

Models such as Extended Neural GPU, ByteNet and ConvS2S, all of which use convolutional neural networks as basic building blocks, computing hidden representations in parallel for all input and output positions. These reduce sequential computation

In these models, the number of operations to relate two arbitrary input and output positions is proportional to the distance between such positions. This proportion is linear in Conv2S models and logarithmic in ByteNet model. Such an equation makes it more difficult to relate input and output positions that are far away. In Transformer model, this is reduced to constant number of operations, which causes reduction in effectiveness in text generation with clarity .But, this problem is surmounted using multi headed attention

Self attention also called intra attention is a mechanism
which relates different positions of a sequence to compute representation of an input sequence. It has been used with success in tasks such as reading comprehension, abstractive summarization(it means generating concise summary of given text) ,task independent sentence representation and textual entailment

Textual entailment is the processing of inferring something related to the given input text .E.g:
Input : John likes to do diving and swimming
Inference : John likes water sports

There are two recurrence mechanisms 1.Sequential recurrence and 2. Recurrent attention mechanism . The latter proved to be better performing in language modeling tasks and simple language question answering tasks

Transformer is the first transduction model that completely relies on self attention mechanism to compute input and output representations without depending on sequence aligned RNNs or convolution

3. Model architecture

The best performing sequence transduction models have encoder and decoder mechanism
Encoder maps input sequence containing symbol representations (x1,x2…xn) to continuous representations z=(z1, z2…zn) . For the given input zn , the decoder then generates an output sequence of symbols (y1,y2 … ym) , one symbol at one time .
At each step , the model is auto regressive which means that the symbols generated in previous output are used as additional input for generating current output

Transformer model contains stacked self attention and point wise fully connected layers for both encoder and decoder

3.1 Encoder and decoder stacks

Encoder: Encoder consists of six identical layers . Each layer again consists of 2 sub layers. Among these 2 sub layers , one is a multi head self attention mechanism and the second one is a simple position wise fully connected feed forward network.

Around each of these 2 sub layers, there is a residual connection and a layer normalization layer which means that output of each layer is LayerNorm(x + Sublayer(x)), Sublayer(x) denotes function implemented by Sublayer itself.

To make these residual connections possible, all sublayers including embedding layer produce outputs of equal dimension which is 512

Decoder: Decoder also contains six identical layers. But , each layer in the decoder consists of 3 sublayers instead of 2 sublayers. The 3rd sub layer performs multi head attention on the output coming from the encoder stack .
This 3rd sublayer is in between first and second sublayers witin the layer

Like in encoder, decoder too has residual connections and layer normalization layer . Output of each layer is again LayerNorm(x + Sublayer(x)).

The multi head self attention mechanism in the first sublayer is modified to prevent positions from attending to subsequent positions (it's crucial that when generating a position (like the next word in a sentence), the model can only attend to the current and previous positions. This is done to ensure that the model does not "cheat" by looking at future tokens, which is important for autoregressive tasks like language generation)

This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.
This means that for each token position i in the output, the decoder actually receives the token from position i−1 as input.
Example: Translating "I love cats" to "J'aime les chats"
Input to Decoder at Each Step:

- Step 1: <start>
- Step 2: <start>, "J'"
- Step 3: <start>, "J'", "aime"
- Step 4: <start>, "J'", "aime", "les"
- Step 5: <start>, "J'", "aime", "les", "chats

Tokens from ground truth data are fed into decoder. Tokens in the future are masked to attention
When input "I" is used as input to encoder , <start> token is used as input to decoder
The masked multi head attention layer attends to <start > hiding all subsequent tokens "J'aime les chats"

Summary of Encoder and decoder

Encoder:

- Processes each input sequence independently.
- Uses padding and masks when necessary during batch processing.
- Outputs a set of context-aware vectors regardless of the input length.

Decoder:

- Starts with a special start token and generates tokens one by one.
- Uses masked self-attention to focus only on previous tokens.
- Ends generation when a specific token (like <eos>) is produced, accommodating different output lengths dynamically

3.2 Attention

Attention is a mapping of the query 'Q' and a set of key 'K', value 'V' pairs to an output where Q,K,V and output values are vectors.
Output is computed as weighted sum of values , weight assigned to each value is computed by a compatibility function of query with its corresponding key .
The two types of attention mechanism are:

3.2.1 Scaled Dot product attention

a. The input consists of queries and keys of dimension d_k, and values of dimension d_v.
b. Compute dot products of the query with all keys and divide result by $\sqrt{dk}$

c. Apply Softmax function for result in step b to obtain weight values
d. In practice, attention function on a set of queries is computed simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V .The matrix of outputs is

Attention(Q, K, V) = softmax((QK^T)/$\sqrt{dk}$)V (Note: QK^T is matrix multiplication)

e. Dot product attention and additive attention are most commonly used attention mechanisms
The former is similar to scaled dot product attention except that scaling factor 1/($\sqrt{dk}$) is used . Additive attention computes the compatibility function using a feed-forward network with a single hidden layer.

Dot product and additive attention have the same theoretical complexity. As matrix multiplication is used to implement dot product attention , it is much faster and space efficient than the latter in practice

For small values of d_k , dot product attention performs same as additive attention and it lags behind additive attention if is not scaled for larger values of d_k

For large values of dk, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients . To counteract this effect, we scale the dot products by 1/($\sqrt{dk}$) .

3.2.2 Multi head attention

a. Instead of performing a single attention function with dmodel-dimensional keys, values and queries, it is found useful to project the queries, keys and values h times with different, learned linear projections to dk, dk and dv dimensions, respectively
b. Attention function is performed in parallel on each of the projected versions of queries ,key and values giving d_v dimensional output values . These are concatenated and projected again resulting in final values
c. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head,this cannot be achieved

MultiHead(Q, K, V) = Concat(head1, ..., headh)W^O
where $head_i$ = Attention(QW^Q_i , KW^K_i , VW^V_i)
where the projections are parameter matrices $W^Q_i \in R^{dmodel \times dq}$, $W^K_i \in R^{dmodel \times dk}$,
$W^V_i \in R^{dmodel \times dv}$ and $W^O \in R^{hdv \times dmodel}$.

H = 8 (8 heads) dv = dk = dq =dmodel/h = 512/8 =64

Consider there are n(n =10) tokens each token when converted into embedding , its size is 512, dmodel =512

Queries (Q): Shape would be (n,dmodel)=(10,512)
QW^Q dimension is (10,512)x(512,64) = (10,64)

Keys (K): Shape would be (m,dmodel)=(10,512)
KW^K_i dimension is (10,512)x(512,64) = (10,64)

Values (V): Shape would be (m,dmodel)=(10,512)
VW^V_i dimension is (10,512)x(512,64) = (10,64)

Each attention head would process its Q, K, and V with dimensions (n,dk)=(10,64)
attention =softmax((QK^T)/$\sqrt{dk}$)V
Dimension of attention output is is ((10,64)x(64,10)) x(10,64) = (10,64)
These outputs are concatenated from 8 heads. (8 heads produce outputs in parallel)

So the result dimension is (10,512) which when multiplied with W^o is of dimension (8*64, 512)

(10,512) x (512,512) gives final output of size (10,512)

d. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality

3.2.3 Application of attention in our Model

Multihead attention is used by the transformer in three ways.

a. In encoder- decoder attention layer(the middle sublayer in the decode layer) , queries come from previous decoder layer and memory keys , values come from the output of encoder layer
This allows every position in the decoder to attend all other positions in the input sequence. This effect mimics typical encoder and decoder attention mechanism in sequence to sequence models in papers referred by current work
b. Encoder contains self attention sublayers . In self attention, all keys, queries and values come from the same place which is the output of the previous layer of encoder. Each position in encoder can attend to all positions in previous layer of the encoder
c. Self attention sublayers in decoder (first sublayer in decoder) allows decoder to attend to all positions in decoder up to and including that position . Masking prevents the future information flowing towards leftward thus preserving auto regressive property. This masking is done in the scaled dot product attention where positions that correspond to illegal connections by setting their values to negative infinity before passing input to softmax

3.3. Position-wise Feed-Forward Networks

Both encoder and decoder layers contain a fully connected feed forward network sublayer that is applied to each position separately and identically.

This consists of two linear transformations with a ReLU activation in between
$FFN(x) = \max(0, xW_1+b_1)W_2 + b_2$
This transformation is the same across different positions but different parameters are used from layer to layer.
Another way of describing this is as two convolutions with kernel size 1

For one output neuron
Input $X=[x_1,x_2,x_3,x_4 \ldots.x_{512}]$, $W1=[w_1,w_{,2},w_{,3},w_{,4} \ldots w_{,2048}]$

Xdim = (1,512) W1dim = (1,2048)

Output1 $=w_1*x_1 + w_{,1}*x_2 +w_1* x_3+w_1 *x_4 \ldots.+w_{512)}*x_{512}+b_{x(1)}$ This is similar to 1D convolution
(For a kernel size of 1, it means that each input feature is multiplied by the same weight independently)

There will be such 2048 outputs and hence middle layer consists of 2048 values

For the last layer input , $Y= [y_1,y_2,y_3 \ldots.y_{2048}]$, $W2=[w_1,w_2,w_3,w_4 \ldots w_{512}]$.
Ydim = (1,2048) , W2dim = (1,512)

Output1 $= w_1*y1_1 + w_2*y_2 +w_3* y_3+w_4 *y_4 \ldots.w_{2048}*y_{2048} +b_{y(1)}$ This is similar to 1D convolution
(For a kernel size of 1, it means that each input feature is multiplied by the same weight independently.)

There will be such 512 outputs and hence output consists of 512 values

3.4 Embeddings and Softmax

Like other transduction models, learned embeddings are used to convert the input tokens and output tokens to vectors of dimension $d_{model} = 512$.
Linear transformation and softmax function are used to convert decoder output into predicted next token probabilities.
In the transformer model (in current work), a common weight matrix is shared between 2 embedding layers and pre softmax linear transformation layer(this is present in the output of the decoder) .Of the 2 embedding layers one is present in the encoder and other is present in decoder .
In embedding layers, weights are multiplied by $\sqrt{dmodel}$

Comparison of self attention mechanism with the other sequential mechanisms.

Layer Type	Complexity per Layer	Sequential Maximum Path Length Operations	Self-Attention
Self attention	$O(n^2 d)$	O(1)	O(1)
Recurrent	$O(n d^2)$	O(n)	O(n)
Convolutional	$O(k.n d^2)$	O(1)	$O(\log_k(n))$
Self attention(restricted)	O(r.n.d)	O(1)	O(n/r)

3.5. Positional Encoding

Transformer model does not contain either convolution or recurrence. Hence, in order the model can know the order of sequence, information about absolute or relative positions of tokens must be injected. To achieve this , positional encodings are added before passing input to the embedding layers of both encoder and decoder stacks . Positional encodings and embeddings have the same dimensions (which is d_{model} and its value is 512) so that they can be summed.

There are multiple choices for positional encodings and in the current work , sine and cosine functions of different frequencies are used

$P E_{(pos,2i)} = \sin(pos/10000^{2i/dmodel})$
$P E_{(pos,2i+1)} = \cos(pos/10000^{2i/dmodel})$

In above equations :
a.pos = position of the token
b.i = dimension / index in the positional encodings vector , vector size is d_{model}
c.Each dimension corresponds to sinusoid and the wavelengths of all these sinusoids are in geometric progression
d.The reason for choosing sinusoidal functions for positional encoding is that the model will be easily learning to attend(learn the relationships)by relative positions because for any fixed offset k,
PEpos+k can be represented as a linear function of PEpos
e. It was also experimented using learned positional embeddings and found that the performance was nearly identical to that when positional encoding is used.
f. Positional encoding is preferred because it allows the model to extrapolate for longer sequence lengths that are not seen during training

4. Why Self attention
Comparison of various aspects of self attention layers with recurrent and convolutional layers that are commonly to used to map variable length sequence of symbol representations (x1,x2…xn) to another sequence of equal length (z1,z2…zn)is explained in this section . Such layers are used as a hidden layer in a typical transduction encoder or decoder . There are three things that are wanted which motivate use of self attention.

1. Total computational complexity of the layer
2. Amount of computation that can be parallelized (measured by the minimum number of sequential operations required)
3. Path length between long range dependencies in the network

   Learning long range dependencies is a key challenge in many sequence transduction tasks .
   The shorter the lengths have to be traveled by the forward and backward signals between any combination of positions in input and output sequences ,the easier it is to learn long range dependencies.
   Hence maximum path length between any two input and output positions in networks composed of different layers is considered

Self attention layer connects(attends) all positions with a constant number of sequential operations whereas the recurrent layer requires O(n) sequential operations.
In terms of computational complexity,self attention layers are faster than recurrent layers when the sequence length is smaller than the representational dimensionality which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece and byte-pair representations.

Word piece representation example:
happiness : hap, pin,ess

BPE encoding:

Imagine a corpus is made of 3 words bun fun den and these 3 words are repeated many times in text

Let the frequency count be {{“bun”,2} ,{“fun”,3} ,{”den ,4}}.
Base vocabulary is made of characters that build these words : {‘b’,’d’ ’f’,’e’ ’u’,’n’}

Let count map is split as { {‘b’,’u’,’n’ ,2}, {‘f’,’u’,’n’,3}, {‘d’,’e’,’n’,4}}
Count which pair exist most “un” is there for 5 times, need to pair and add it in vocabulary

Updated vocabulary : {“b”,”d”, “f”,”e”, “u”,”n”,”un”}

{ {“b”,”un” ,2}, {“f”,”un”,3}, {‘d’,’e’,’n’,4}}

Until desired size vocabulary is reached this encoding is continued

To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to O(n/r). This is planned in future work

If the input size is n and kernel size is k,
a.Then it requires O(n/k) convolutions to connect all inputs to outputs . This is contiguous convolution
b. Complexity is $O(\log_k(n))$ for dilated convolutions which increases longest path between any 2 positions in the network
c. Convolutional layers are generally more expensive than recurrent layers, by a factor of k
d. Separable convolutions , however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$.
When k = n, the complexity of a separable convolution is equal to $O(n^2d + n.d^2)$. This is same computational complexity of a self-attention layer and a point-wise feed-forward layer combined

Self attention helps to achieve more interpretable models . It is found that individual attention heads clearly learn syntactic relationships (like subject-verb agreements), while another might capture semantic roles (like entities or actions)

5 Training

5.1 Training Data and Batching

a. For English to German translation task, WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs was used for training
BPE was used for representation of input, which resulted in creating a shared source-target vocabulary of 37,000 tokens
b. For English to French translation task, ,WMT 2014 English-French dataset consisting of 36M sentences was used for training
Word piece was used to represent input which resulted in creation of 32,000 word piece vocabulary
c. Sentences were batched together by approximate sequence length
d. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens

5.2 Hardware and Schedule

a. Machine with 8 NVIDIA P 100 GPUs is used to train the model
b. For base models, using the hyperparameters described in the paper, each training step took about 0.4 seconds.They are trained for 12 hrs i.e. 100,000 steps

c. For bigger models, step time was 1.0 seconds . Models are trained for 300,000 steps in 3.5 days

5.3 Optimizer

a. Adam optimizer with the parameters with $\beta1 = 0.9$, $\beta2 = 0.98$ and $\epsilon = 10^{-9}$ was used
b. lrate = $d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$, using this formula ,learning rate was increased for the first warmup_steps training steps (warmup_steps = 4000 was used)
c. For the next training steps, the learning rate was decreased proportional to the inverse square root of step number.

5.4 Regularization

Residual Dropout

a. Dropout is applied to output of each sub layer before it is passed as input to the same sub layer and normalized
b. Dropout is also applied to sums of positional encodings and embeddings in both the encoder and decoder stacks .For the base model, a dropout rate Pdrop =0.1 used

Label smoothing :

Label smoothing of $\epsilon_{ls} = 0.1$ is used to make the model a little ensure about the next token predicted (Perplexity: It means the model's uncertainty in predicting the next word in a sequence) .It improves the model prediction accuracy and BLEU score.

For e.g. if there are one of the 25 tokens to be predicted, ground truth probability is 1-0.1 =0.9 and other class probabilities shall be 0.1/25 = 0.004

6 Results

6.1 Machine Translation

a. In 2014 German to English Translation task, big Transformer model outperforms all previous models including ensemble models by a BLEU score of 2, establishing a new start of the art BLEU score 28.4
It took 3.5 days of training the model on 8 P100 GPUs
The base model too surpasses all the previously published models and ensembles at a fraction of training cost of any of the competitive models

b. In 2014 English to French Translation task, big Transformer model achieves a BLEU score of 41.0, outperforming all the previously published single models at 1/4th training cost of the previously trained state of the art model.The big Transformer model trained for this task used a dropout rate $P_{dropout} = 0.1$ instead of 0.3

c. Checkpoints: During training, a model's parameters (weights and biases) are saved at regular intervals, called checkpoints. This allows you to resume training from a certain point or to evaluate the model's performance at different stages of training.

For the base models, the final model was created by averaging the parameters of the last 5 checkpoints saved every 10 minutes. This means the model takes the weights from those 5 most recent checkpoints and computes their average to create a single, final model. This can help stabilize predictions by mitigating the noise from individual training epochs

For the big models, the last 20 checkpoints were averaged.This likely provides even more stability and potentially better performance by considering a broader range of the model's training history.

d. Beam Search: This is an algorithm that explores multiple possible output sequences simultaneously. Instead of choosing the single best option at each step (like greedy decoding), beam search keeps track of the top k (the beam size) hypotheses at each step and expands them.

Beam search of beam size =4 was used in this paper

Length Penalty ($\alpha = 0.6$): This is a technique used to adjust the scoring of sequences based on their length. A length penalty discourages the model from generating excessively short or long sequences. The value $\alpha=0.6$ suggests that the algorithm penalizes longer sequences more than shorter ones, promoting a balance between fluency and conciseness.

These hyperparameters were chosen after experimentation on the development set.

e. Setting the maximum output length to "input length + 50" means that for any given input sequence, the model is allowed to generate up to 50 additional tokens beyond the length of the input. For example, if the input is 100 tokens long, the maximum output can be 150 tokens.However, if the model can complete its response earlier (for example, when it detects that the answer is fully formed), it will stop generating instead of reaching the maximum limit

f . The current work compares translation quality and training costs to other model architectures from the other research works. The number of floating point operations used to train a model is estimated by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU(refers to the measurement of how many calculations a GPU can perform using single-precision floating-point numbers over a sustained period)

6.2 Model variations

To evaluate different components of the Transformer model, the base model is varied in different ways measuring the change in performance in English to German translation task on the newest development set, newstest 2013. Beam search was used but not model check point averaging.

Experiments were made to vary the number of attention heads and the attention key, value dimensions, keeping the amount of computation constant.

Following observations were made:

a. Single head attention is worst than best setting by 0.9 , quality also drops off with too many heads

b. Reducing the attention key size d_k reduces the model performance

c. The "compatibility function" is what the model uses to determine how well different elements relate to each other. In the case of traditional attention mechanisms, this is often calculated using the dot product of the query and key vectors

The observation implies that the dot product may be too simplistic to capture complex relationships in the data. A more advanced method could improve the model's ability to determine relevance and improve overall performance

d. Bigger models are better, and dropout is very helpful in avoiding overfitting

e. Replacing the sinusoidal positional encoding with learned positional embeddings produced nearly identical results to the base model

6.3 English Constituency Parsing

English constituency parsing is a process in natural language processing (NLP) that involves analyzing the grammatical structure of a sentence to identify its constituents—like phrases and words—and how they relate to each other

To check if the transformer model can generalize well, the authors of work have made experiments using the english constituency parsing. In this task, the output can be longer than input.

RNN models could not achieve the start of the results in small data regimes

A 4-layer transformer with dmodel = 1024 on the Wall Street Journal (WSJ) portion of the Penn Treebank, about 40K training sentences

(When training language models, especially in a semi-supervised setting, the quality of the training data is crucial. "High-confidence" data likely means that the sentences or annotations within these datasets are:

a. Well-formed: Sentences that follow grammatical rules and structures.

b. Accurately labeled: If the dataset includes annotations (like part-of-speech tags or syntactic structures), these annotations have been verified to be correct.
c. Less noisy: The data has been filtered to exclude errors, ambiguities, or low-quality examples that might confuse the mode)

Also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora(not labeled) from with approximately 17M sentences.

Also, used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting

Semi supervised training means both labeled and not labeled data is used in training

Only a small number of experiments were performed to select the dropout, both attention and residual, learning rates and beam size on the Section 22 of WSJ development set, all other parameters remained unchanged from the English-to-German base translation model

During inference , maximum output length was increased to input length + 300

A beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting

Authors have proved that despite the lack of task specific training ,model performs well better than all previously reported models except the recurrent neural network Grammar

In contrast to RNN sequence-to-sequence models , the Transformer outperforms the BerkeleyParser even when training only on the WSJ training set of 40K sentences

7 Conclusion

In this work, the authors presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers

On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, a new state of the art is achieved. In the former task the best transformer model outperforms even all previously reported ensembles

The authors plan
a. To use the Transformer model for other tasks and extend it to other input and output modalities that are not just text
b. To investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goal.

References:
[1] Attention is all you need https://arxiv.org/pdf/1706.03762

[2]Byte pair encoding
https://huggingface.co/learn/nlp-course/en/chapter6/5