Transformer Models

Sargur N. Srihari
srihari@buffalo.edu

This is part of lecture slides on Deep Learning:
http://www.cedar.buffalo.edu/~srihari/CSE676

Topics in NLP

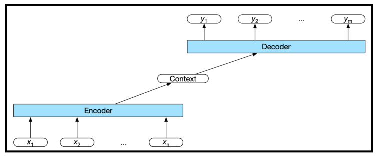
1. N-gram Models
2. Neural Language Models
3. High-Dimensional Outputs
4. Combining Neural Language Models with n-grams
5. Neural Machine Translation
6. Attention Models
7. Transformer Models
8. Historical Perspective

Topics in Transformer Models

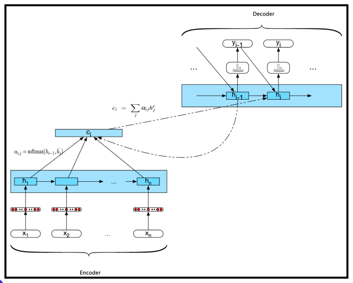
- Types of Attention Models
- The Transformer Model
- Full Transformer with Attention
- Multi-head Attention
- Improvements of Transformer
  - Generative Pre-training (GPT)
  - Bidirectional Encoder Representations from Transformers (BERT)

Types of Attention Models

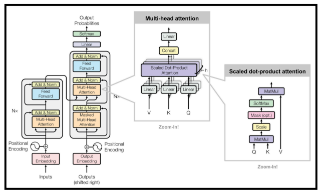
1. Basic encoder-decoder RNN model



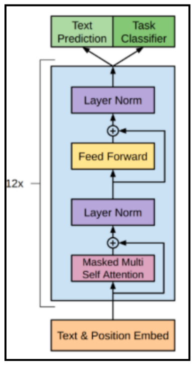
2. Encoder-decoder RNN with Attention



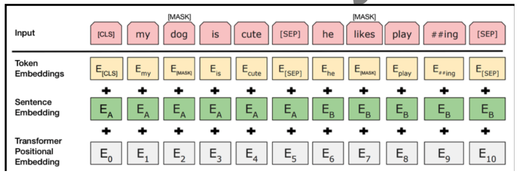
3. Transformer model with no recurrence



4. Generative pre-training of Attention model



5. BERT with no decoder, using bidirectionality

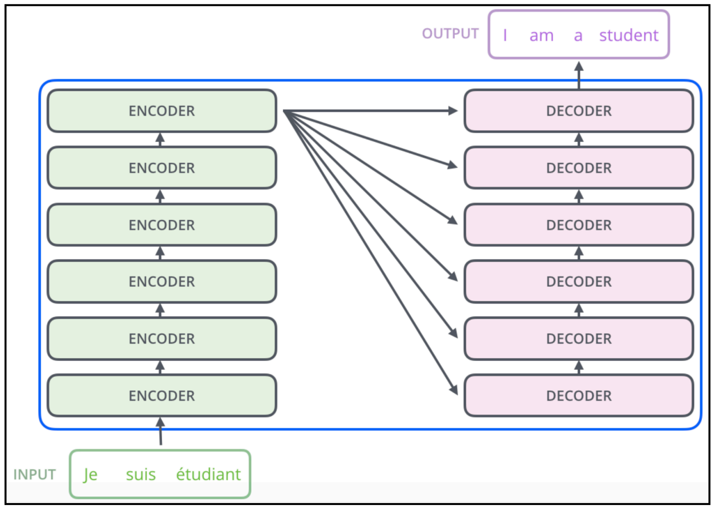


Transformer has no recurrence

- Transformer is an attention mechanism to learn contextual relations between words
  - It includes two mechanisms
    1. An Encoder that reads text input
    2. A Decoder that produces a prediction for the task
- Model has no recurrence
  - Self-attention to represent input/output without RNN
- Allows more parallelism
  - High translation quality: state-of-the-art
    - Training 12 hours on eight P100 GPUs

The Transformer Model

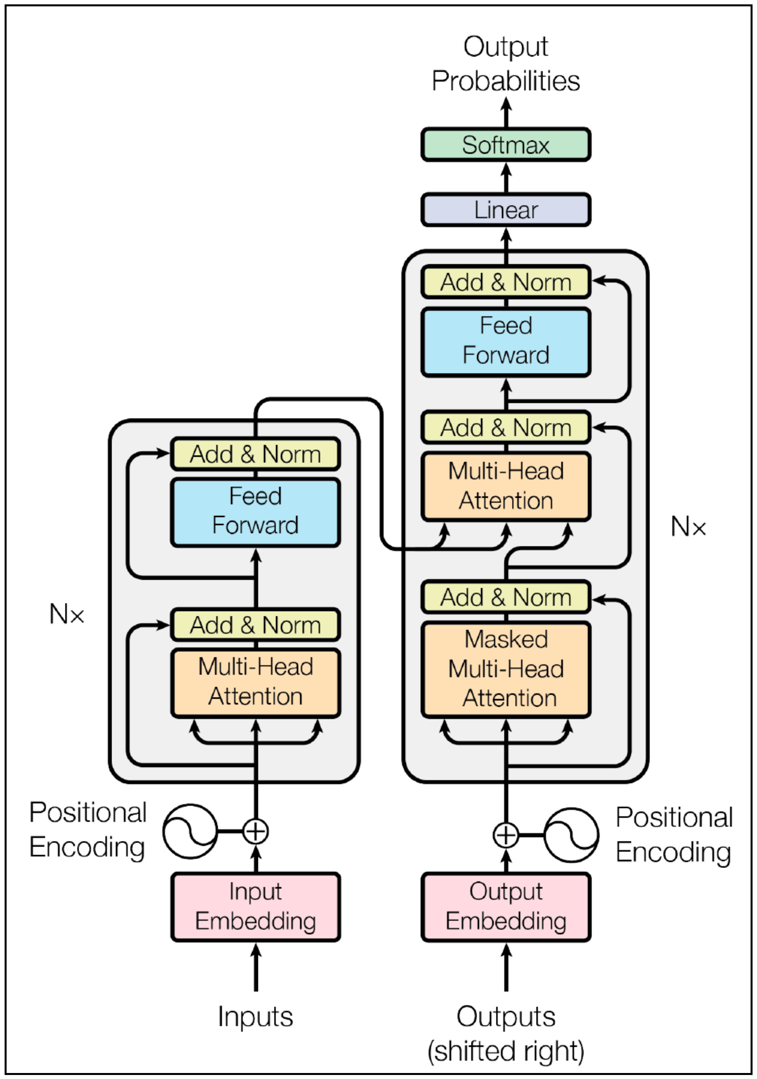
- Has an encoder-decoder architecture
  - Encoder maps input $(x_1,.., x_n)$ to representations $\mathbf{z} = (z_1,.., z_n)$
  - Given $\mathbf{z}$, decoder generates $(y_1,.., y_m)$ element-wise
- At each step model is auto-regressive
  - i.e., consumes previously generated symbols as additional input when generating the next



A stack of encoders and a stack of decoders are used

Encoder and Decoder use Attention

Encoder



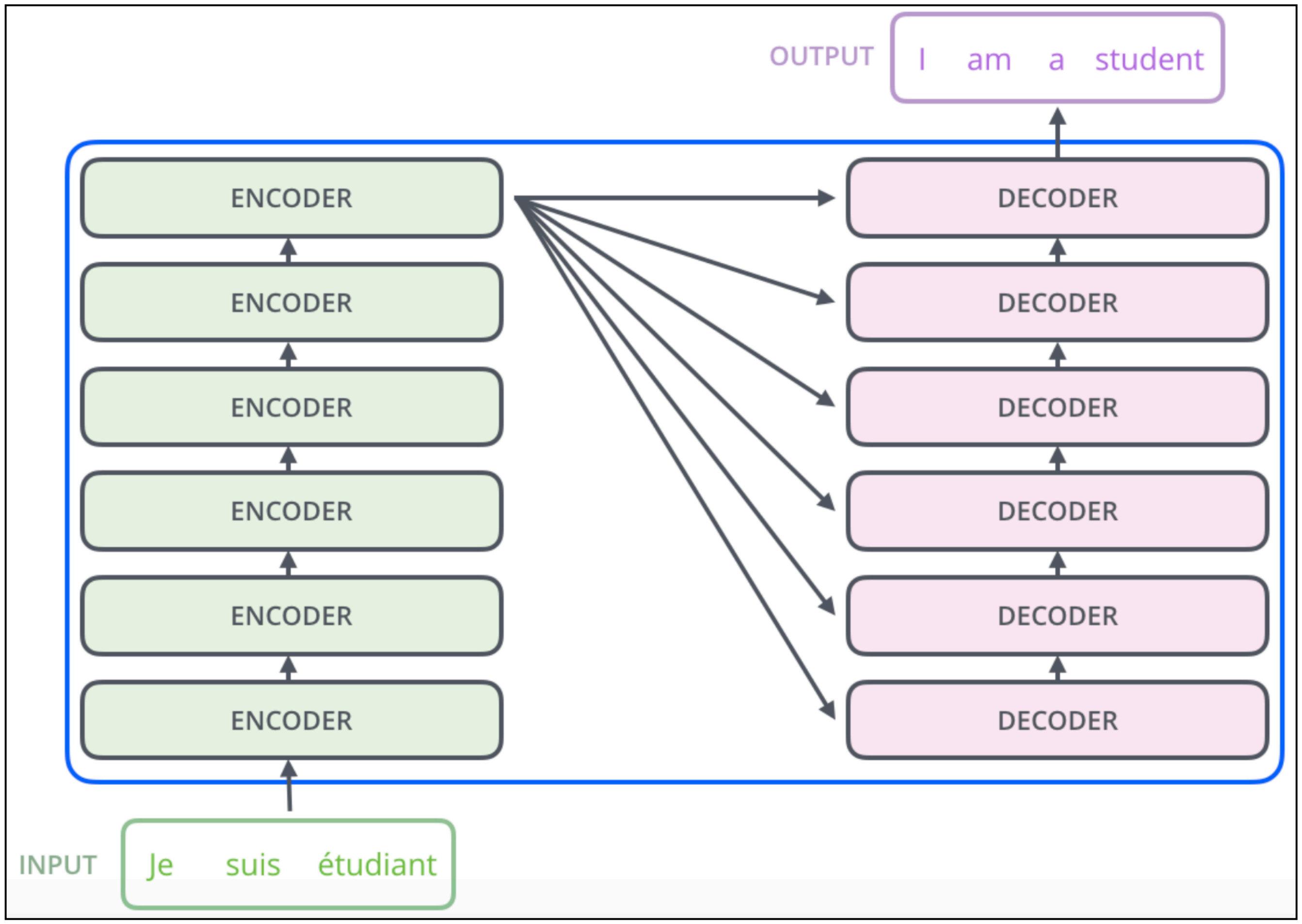
Decoder

Encoder & decoder are:
1. stacked (See next slide)
2. self-attention
3. point-wise
4. fully connected layers

Autoregression
Consume previously generated output

Source: Vaswani, Ashish, et al. "Attention is all you need." Advances in Neural Information Processing Systems. 2017.

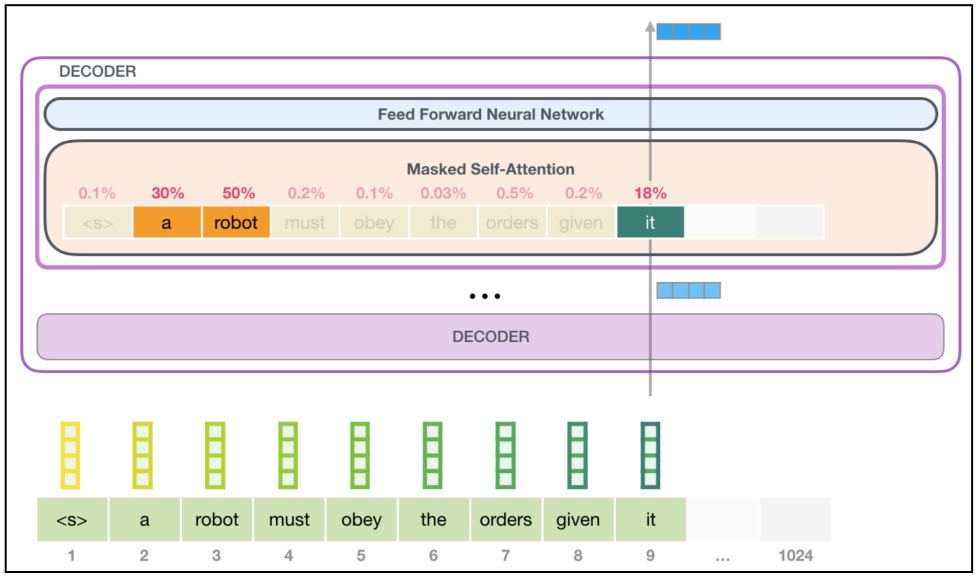
Stacked Encoder, Decoder

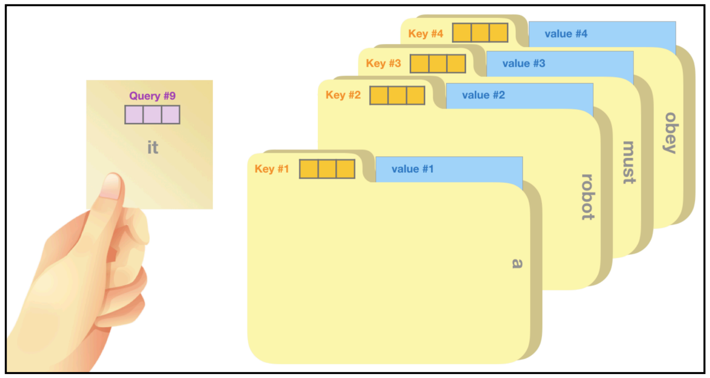


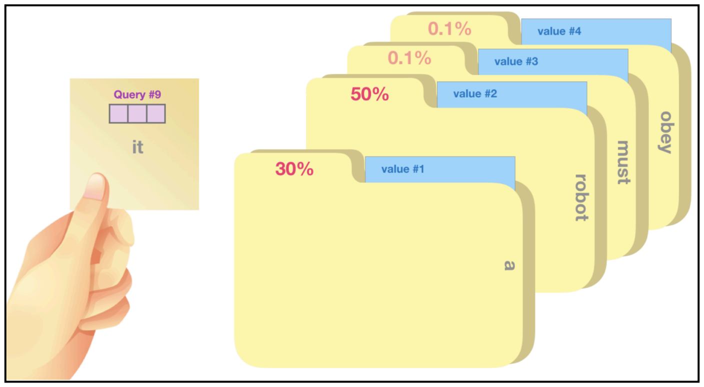
Self-Attention

- Input sentence to translate:
    - Second Law of Robotics: A robot must obey the orders given **it** by human beings except where **such orders** would conflict with the **First Law**.
- Highlighted words refer to other words
    - **it** refers to the robot, **such orders** refers to the earlier part of the law, namely "the orders given it by human beings", **The First Law** refers to the entire First Law
- This is what self-attention does
    - It bakes-in the model's understanding of relevant and associated words that explain the context of a certain word before processing that word
    - By assigning scores to how relevant each word in the segment is, and adding up their vector representation

Self-Attention Process

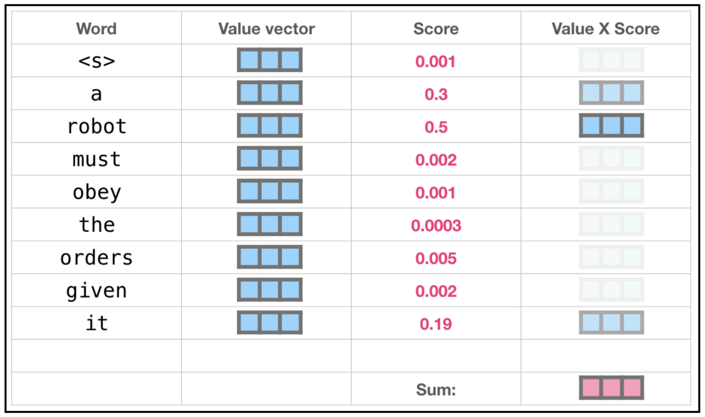

DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1% 30% 50% 0.2% 0.1% 0.03% 0.5% 0.2% 18%
<s> a robot must obey the orders given it
...
DECODER
<s> a robot must obey the orders given it
1 2 3 4 5 6 7 8 9 ... 1024





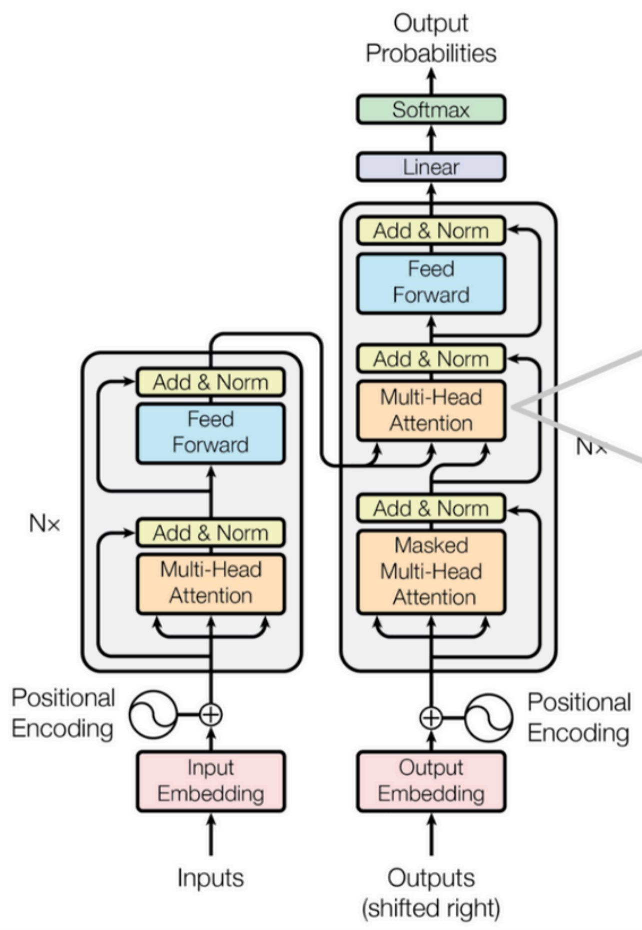
This self-attention layer is paying attention to "a robot" when it processes the word "it". The vector it will pass to its neural network is a sum of the vectors for each of the three words multiplied by their scores.

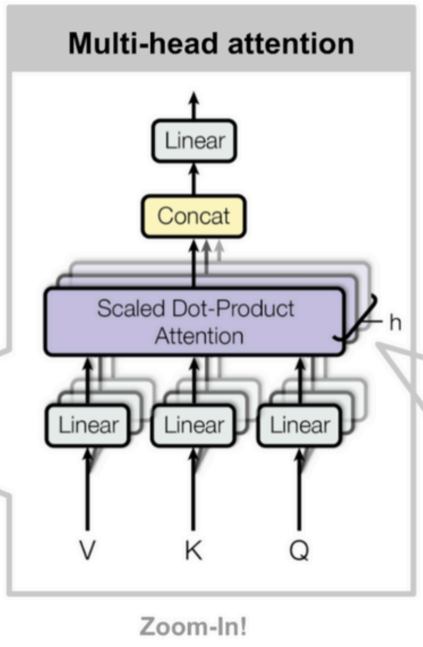
Self-attention is processed along the path of each token in the segment.
The significant components are three vectors: Query, Key and Value



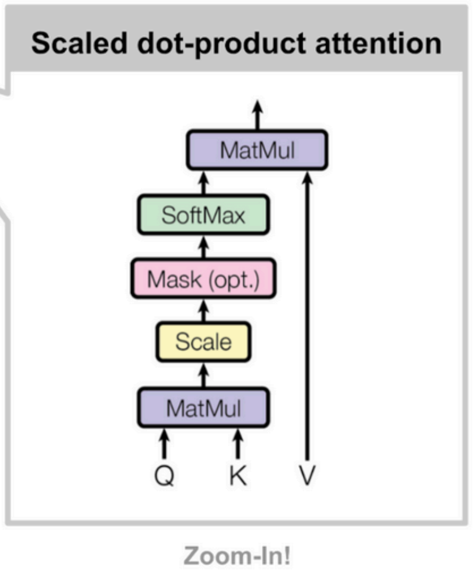
Word	Value vector	Score	Value X Score
<s>	🟦🟦🟦	0.001	⬜⬜⬜
a	🟦🟦🟦	0.3	🟦🟦🟦
robot	🟦🟦🟦	0.5	🟦🟦🟦
must	🟦🟦🟦	0.002	⬜⬜⬜
obey	🟦🟦🟦	0.001	⬜⬜⬜
the	🟦🟦🟦	0.0003	⬜⬜⬜
orders	🟦🟦🟦	0.005	⬜⬜⬜
given	🟦🟦🟦	0.002	⬜⬜⬜
it	🟦🟦🟦	0.19	🟦🟦🟦
		Sum:	🟥🟥🟥

Full Transformer (with Attention)





Several attention layers run in parallel



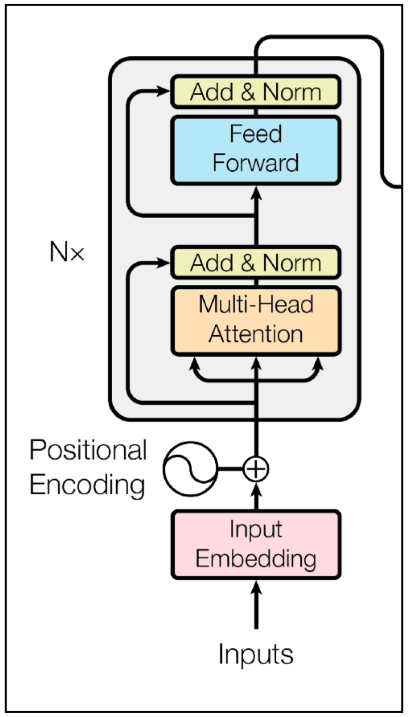
MultiHead (Q, K, V) = Concat(head$_1$,.., head$_h$)
where head$_i$ = Attention$(QW_i^Q, KW_i^K; VW_i^V)$

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

Transformer Encoder Detail

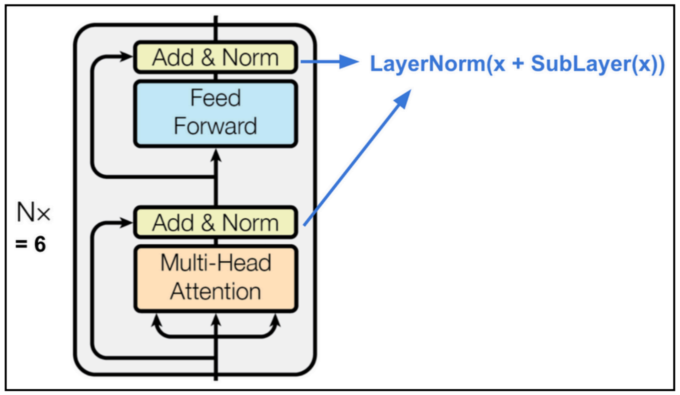
- A stack of $\mathrm{Nx} = 6$ identical layers
- Each layer has two sub-layers:
  1. A multi-head self-attention mechanism
  2. A position-wise fully connected feed-forward network

     A residual connection around each of the two sub-layers, followed by layer normalization
  - Layer normalization: normalization statistics from the summed inputs to the neurons within a hidden layer



Output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself.

Residual connections, sub-layers, embedding layers produce outputs of dimension $d_{\mathrm{model}} = 512$

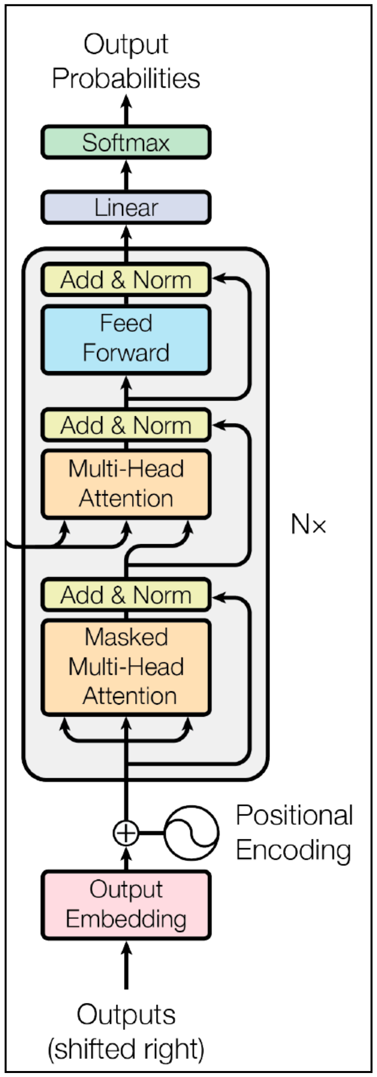


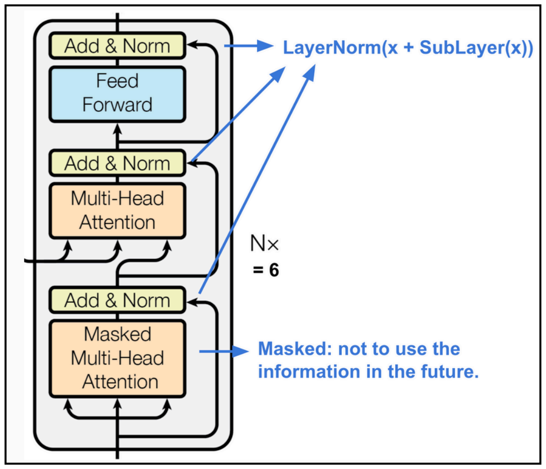
Transformer Decoder Detail

Decoder is also a stack of $\mathrm{Nx} = 6$ identical layers.

Has a third sub-layer which performs
multi-head attention over the output of the encoder stack.
Residual connections around each of
the two sub-layers, followed by layer normalization

Self-attention sub-layer in the decoder stack is modified
to prevent positions from attending to subsequent positions.
This masking, and by off-setting output embeddings by one position,
predictions for at i depend only on known outputs at positions $< i$




Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
LayerNorm(x + SubLayer(x))
Nx = 6
Masked: not to use the information in the future.

Query, Key and Value in Attention

- To compute next word in translation, attention mechanism creates a vector using the source sentence and what has been generated so far

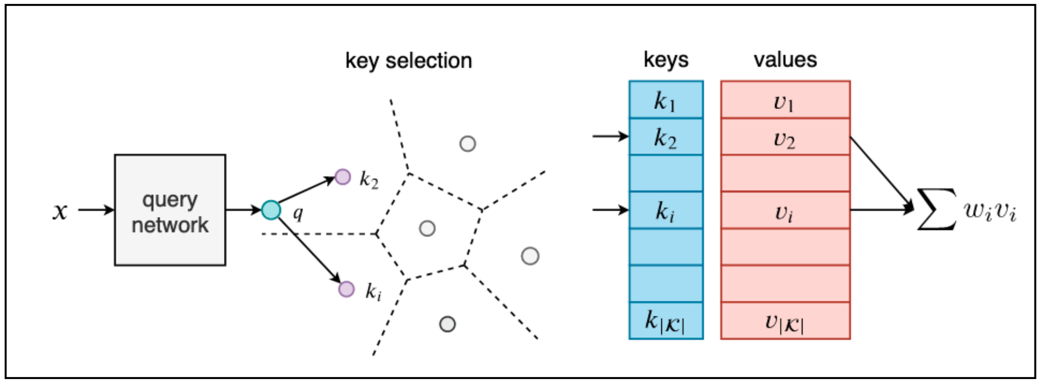
Input sentence	elle	alla	à	la	plage
Key	subject	verb	filler	filler	location
Value	*she*	*to go, past tense*	-	-	*beach*

Output sentence	she	went	to	the	?????
Query	subject	verb	filler	filler	location

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

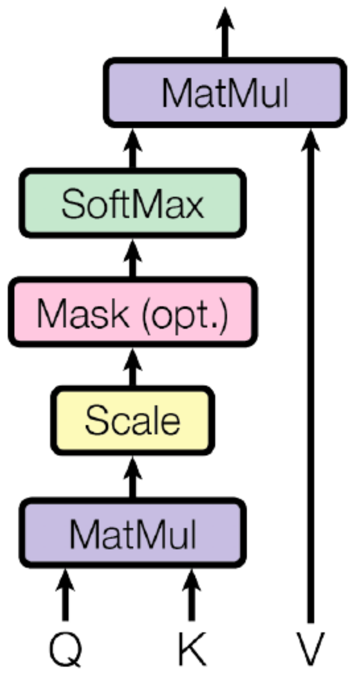
Query, Key and Value in QA

- Key-Value Memory Networks for Question Answering
  - Example
    - Blade Runner directed-by Ridley Scott has the form *subject-relation-object*
    - *Key* is composed of *subject* (Blade Runner) and *relation* (directed by)
    - *Value* is the *object* (Ridley Scott)
    - *Query* is Who directed Blade Runner?



Source: https://arxiv.org/pdf/1907.05242v1.pdf

Detail of Scaled dot product attention



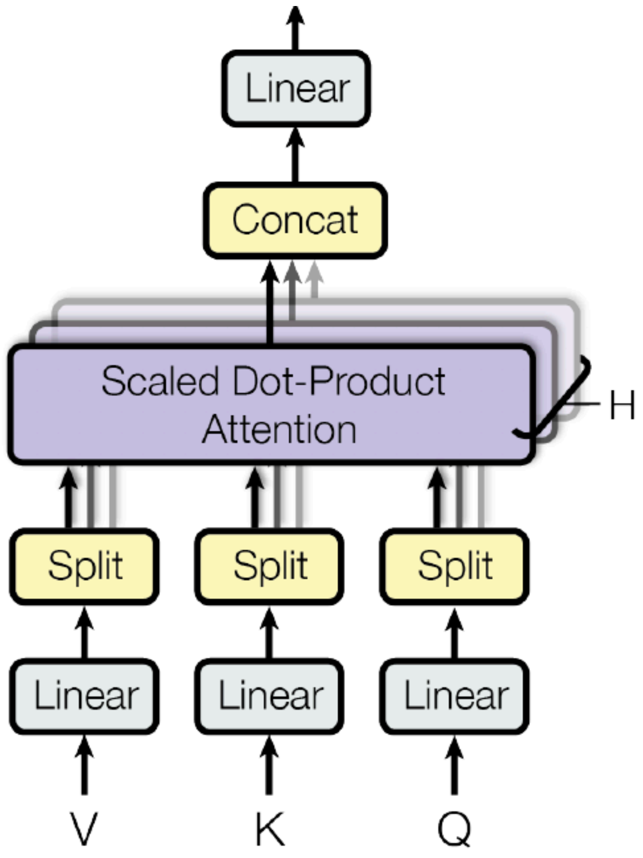
The input consists of queries and keys of dimension d_k, and values of dimension d_k. We compute the dot products of the query with all keys, divide each by $\sqrt{}\ d_k$, and apply a softmax function to obtain the weights on the values.

We compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as:

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

Detail of Multi-head attention

- Several attention layers run in parallel



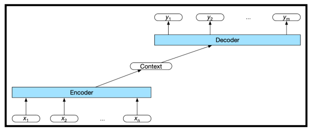
Outputs are concatenated and linearly transformed into the expected dimensions
Allows model to jointly attend to information from different representation **subspaces** at different positions

$$MultiHead(Q, K, V) = Concat(\mathrm{head}_1, .., \mathrm{head}_h)$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K; VW_i^V)$$

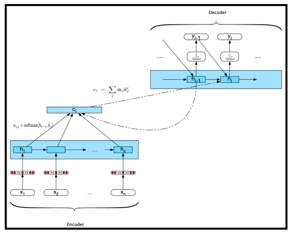
$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k} \text{ and } W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

Types of Attention Models

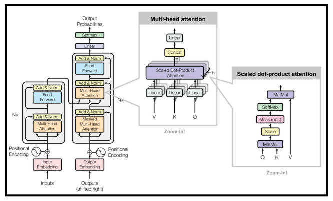
1. Basic encoder-decoder RNN model



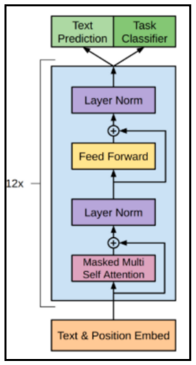
2. Encoder-decoder RNN with Attention



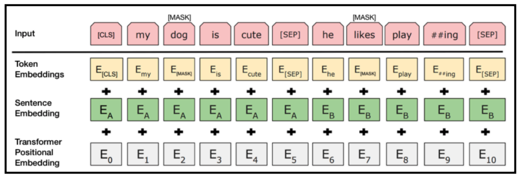
3. Transformer model with no recurrence



4. Generative pre-training of Attention model



5. BERT with no decoder, using bidirectionality

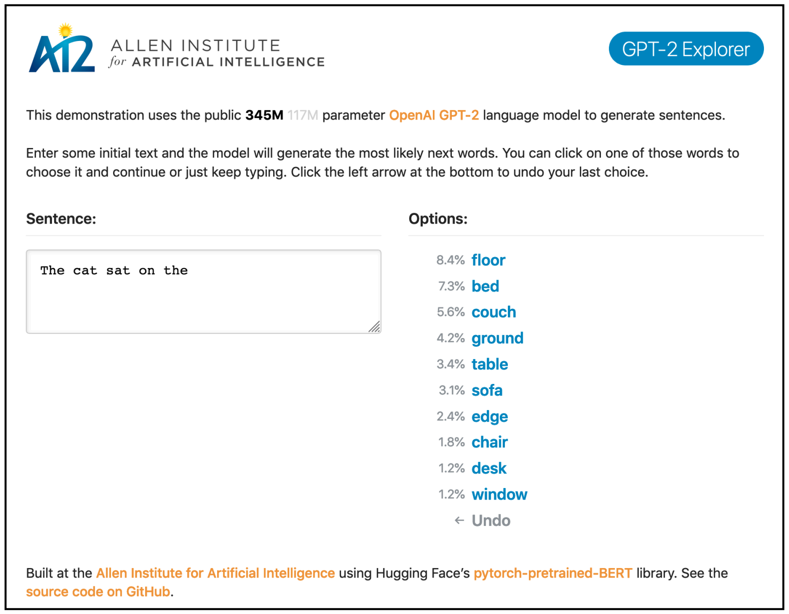


Improvements of Transformer

1. Open AI's *Generative Pre-Training (GPT)*
   - Combines supervised and unsupervised learning to improve word vectors
     - Performs Generative Pre-Training
   - State-of-the art for following tasks:
     - Textual entailment, semantic similarity, reading comprehension, common sense reasoning, linguistic acceptability, multi-task benchmark
2. Google's *BERT*
   - *Bidirectional Encoder Representations from Transformers*
   - BERT is an improvement of GPT
     - Key technical innovation is bidirectional training of Transformer

GPT-2 generates human-like output

- Trained to predict next word (40GB of internet text)



This demonstration uses the public **345M** 117M parameter OpenAI GPT-2 language model to generate sentences.

Enter some initial text and the model will generate the most likely next words. You can click on one of those words to choose it and continue or just keep typing. Click the left arrow at the bottom to undo your last choice.

Sentence:

```
The cat sat on the
```

Options:

8.4%	floor
7.3%	bed
5.6%	couch
4.2%	ground
3.4%	table
3.1%	sofa
2.4%	edge
1.8%	chair
1.2%	desk
1.2%	window
←	Undo

Built at the Allen Institute for Artificial Intelligence using Hugging Face's pytorch-pretrained-BERT library. See the source code on GitHub.

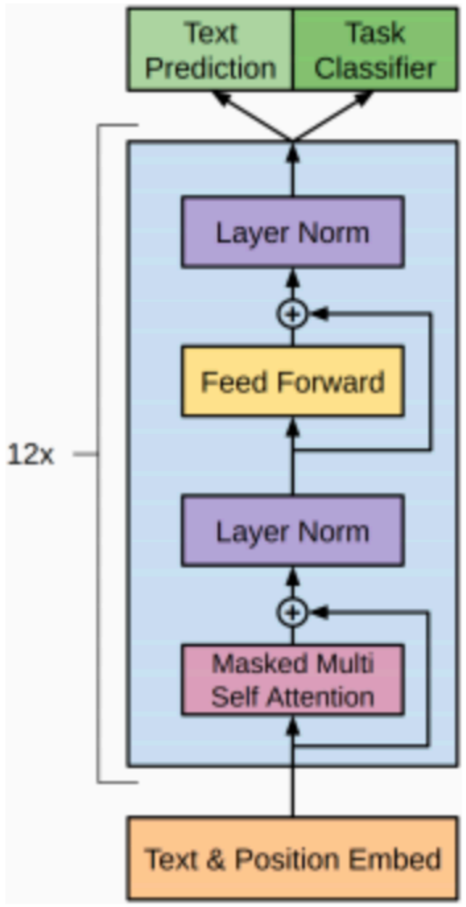
- While trained to predict the next word it surprisingly learned basic competence in tasks like MT and QA.
  - That's without ever being told that it would be evaluated on those tasks.

GPT-2 generated synthetic text

- **System Prompt (human-written)**
- *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
- **Model Completion (machine-written, 10 tries)**
- The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.
- Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.
- Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.
- Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.
- Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

Architecture of GPT

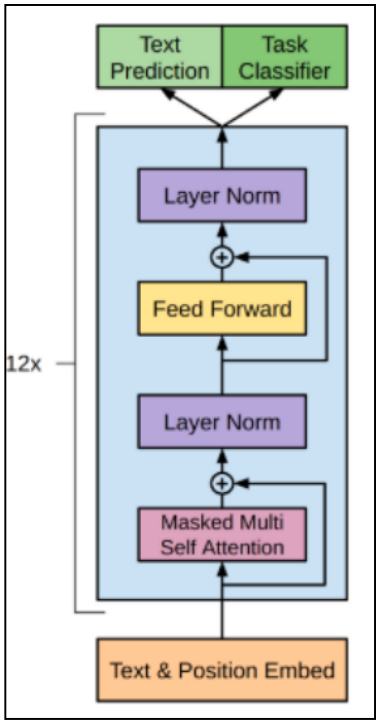
1. Text and position will be transformed to a vectors
2. Pass to multi-head self-attention
3. Combining result from step 1 and step 2 and performing a normalization
4. Pass to a fully-connected feed-forward network
5. Combining result from step 3 and 4 and performing a normalization
6. Finally, combing multi-head (total 12 self-attention block) to together for computing vectors.



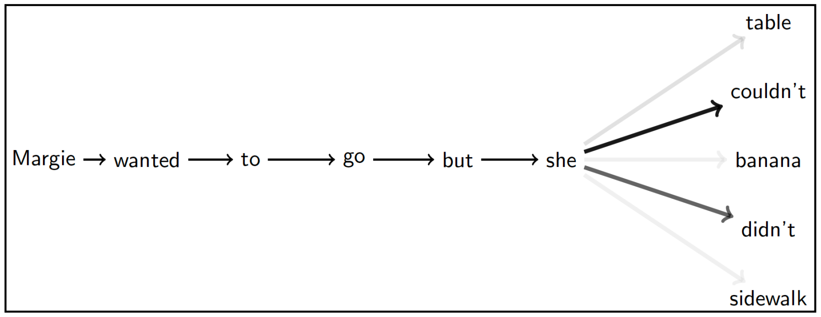
Generative Pre-training (GPT)

- OpenAI multilayer decoder

  W_e is the token embedding matrix
  W_p is the position embedding matrix
  $h_0 = UW_e + W_p$
  h_i = transformer block $(h_{i-1})\ \forall i \in [1, n]$



- Unsupervised pretraining task
  - Language Modeling



Language Modeling Loss

$$P(u) = \text{softmax}(h_n W_e^T)$$
$$L_1(U) = \Sigma_i \log P(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Classification Loss

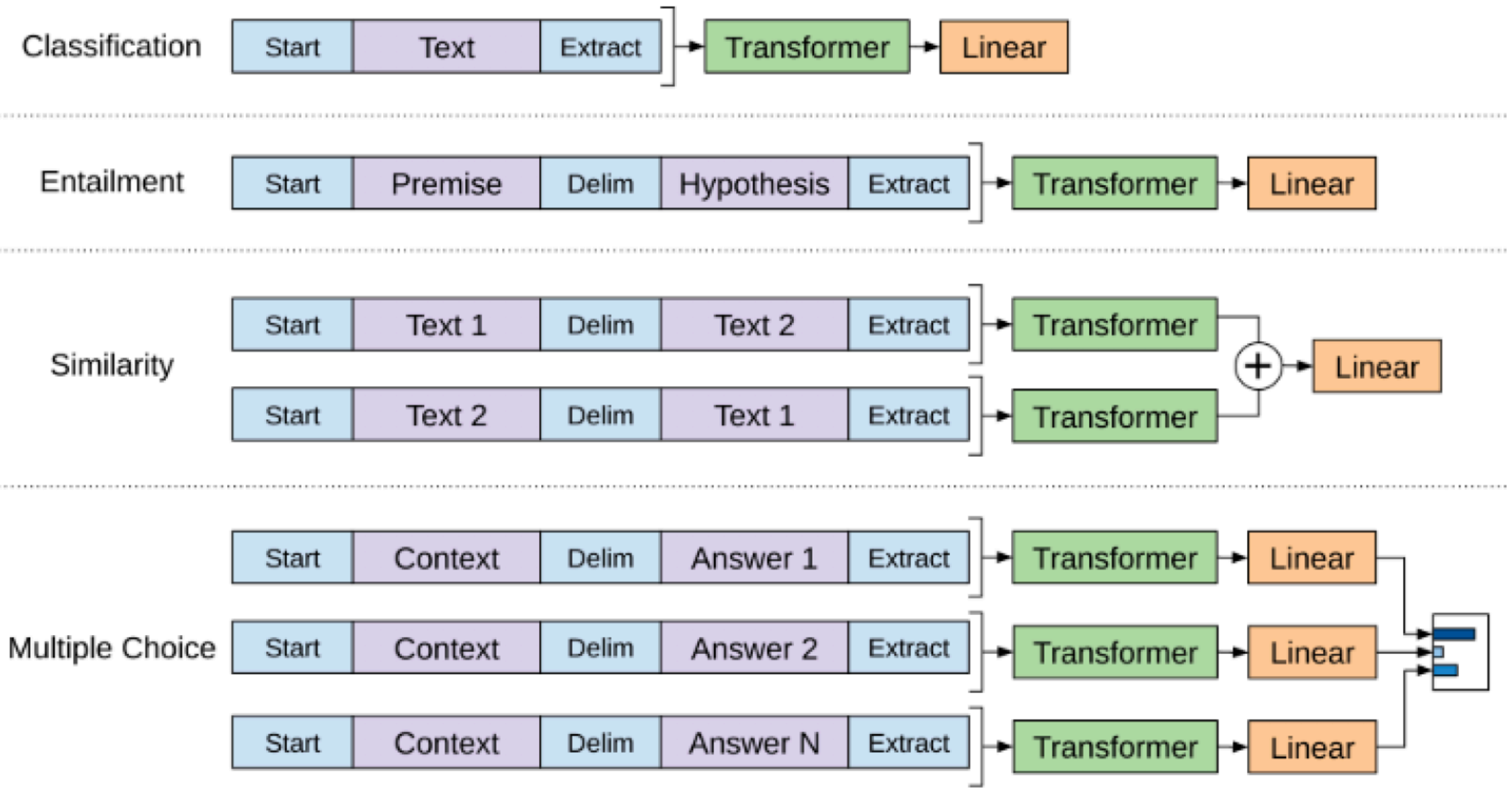
$$P(y \mid x^1, \ldots, x^m) = \text{softmax}(h_n^m W_y)$$
$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y \mid x^1, \ldots, x^m)$$

Final Loss

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

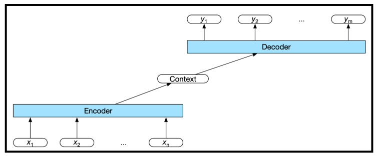
- Supervised fine-tuning
  - Multitask learning

Input Formatting for GPT

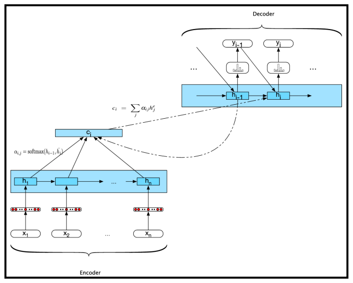


Types of Attention Models

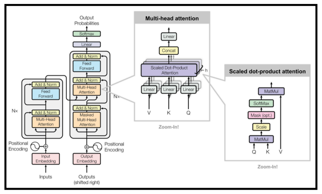
1. Basic encoder-decoder RNN model



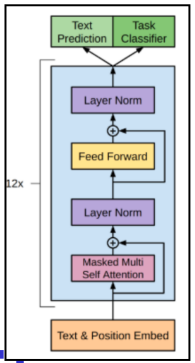
2. Encoder-decoder with Attention


Decoder
y_{t-1}
y_t
$c_t = \sum_j \alpha_{tj} h_j$
$\alpha_{tj} = \text{softmax}(h_{t-1}, h_j)$
c_t
h_1
h_2
h_n
x_1
x_2
...
x_n
Encoder

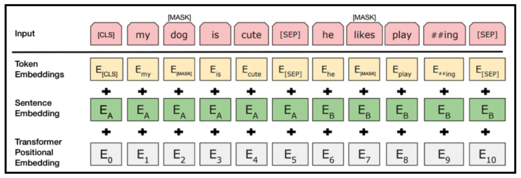
3. Transformer model with no recurrence



4. Generative pre-training of Attention model



5. BERT with no decoder, using bidirectionality

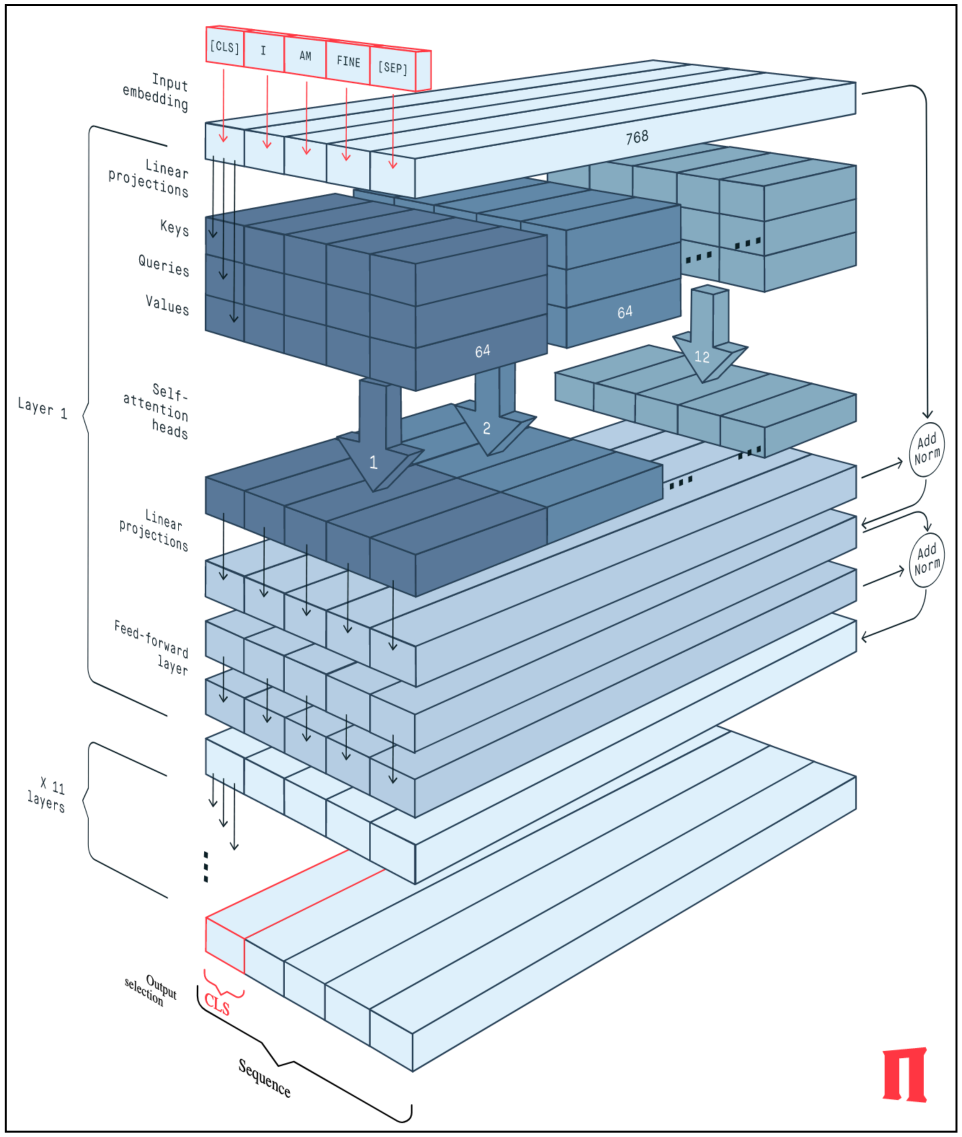


Google's BERT

(Bidirectional Encoder Representations from Transformers)

1. Uses Transformer attention mechanism to learn contextual relations between words
   - Transformer includes two mechanisms
     1. An Encoder that reads text input
     2. A Decoder that produces a prediction for the task
   - Since BERT's goal is to generate a language model, only the encoder mechanism is necessary
2. Uses Bidirectional training
   - Deeper sense of context/flow than single-direction
     - Masked LM (MLM) allows bidirectional training

Structure of BERT



The BERT Encoder block implements the base version of the BERT network

It is composed of 12 successive transformer layers, each having 12 attention heads

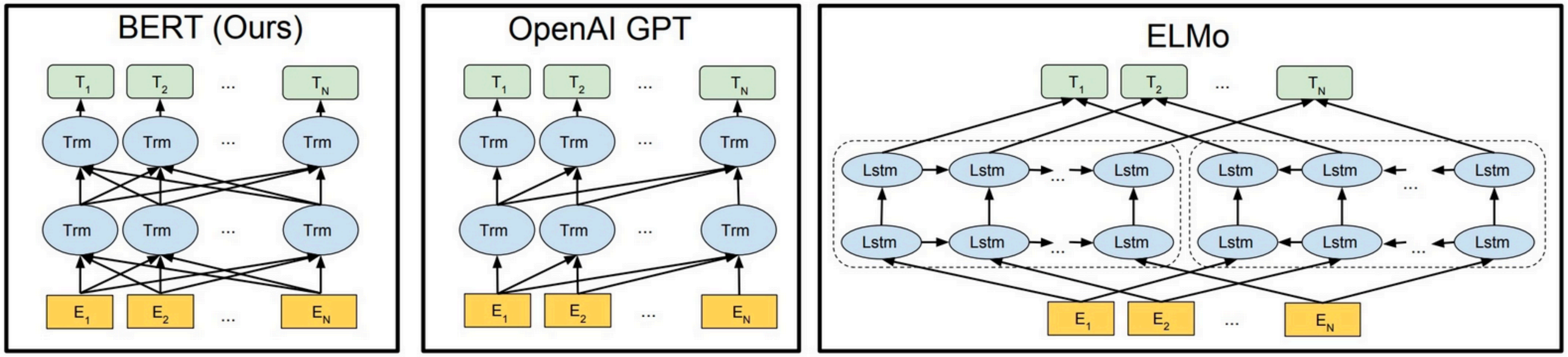
The total number of parameters is 110 million.

https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/bert-encoder?fbclid=IwAR2rt4ObbHCMvuGsfrNNOnotfW_Dfc-5y3QorqW1nQUbw4XzuwjjCbnQDy0

Comparison of BERT and GPT

- BERT is an improvement on GPT
- The main differences are:
  - Bidirectional training
  - Different pre-training tasks (masked language model and next sentence prediction)
  - Trained on a much bigger corpus (BookCorpus (800M words) + Wikipedia (2500M words))
  - 3 x as many parameters for the large version
  - Pre-trained model for 102 languages

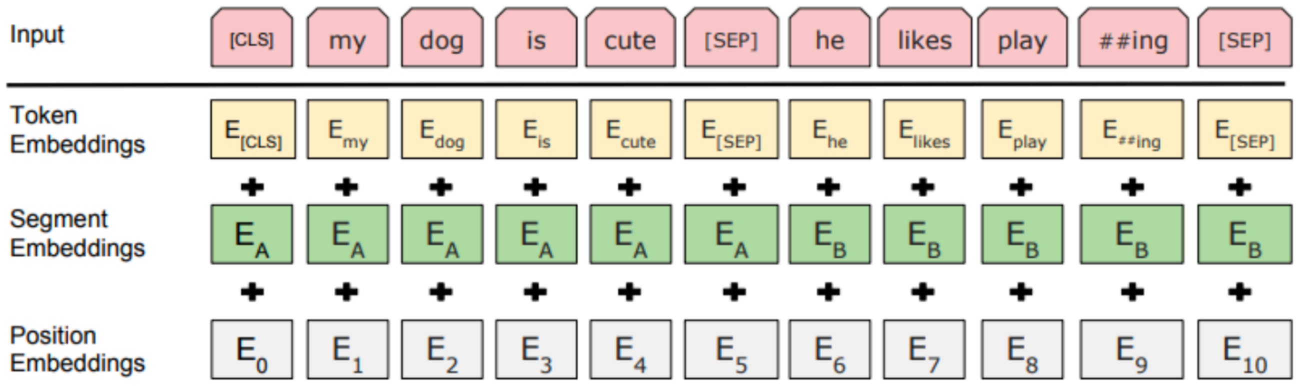
Bidirectionality



Arrows indicate information flow from one layer to next
Green boxes at top indicate final contextualized representation of each input word
Evident from the above image:

- BERT is bi-directional
- GPT is unidirectional (information flows from left-to-right)
- ELMO is shallowly bidirectional

Text Pre-processing


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Position Embeddings:
BERT learns and uses positional embeddings to express the position of words in a sentence. These are added to overcome the limitation of Transformer which, unlike an RNN, is not able to capture "sequence" or "order" information

Segment Embeddings:
BERT can also take sentence pairs as inputs for tasks (Question-Answering). That's why it learns a unique embedding for the first and the second sentences to help the model distinguish between them. In the above example, all the tokens marked as EA belong to sentence A (and similarly for EB)

Token Embeddings:
These are the embeddings learned for the specific token from the WordPiece token vocabulary

Pre-training BERT

- BERT is pre-trained on two NLP tasks:
  1. Masked Language Modeling(For Bidrectionality)
  2. Next Sentence Prediction

Masked Language Model

- Before feeding word sequences 15% of words in each sequence replaced with a [MASK] token
- Model attempts to predict original value of masked words, based on context provided by non-masked words in sequence
- Prediction of the output words requires
  - Adding a classifier layer on top of encoder output
  - Multiplying output vectors by the embedding matrix, transforming them into the vocabulary dimension.
  - Calculating probability of each word in the vocabulary with softmax.

Word prediction using masking

Word prediction using context from only one side

Alaska
Alaska is
Alaska is about
Alaska is about twelve
Alaska is about twelve times
Alaska is about twelve times larger
Alaska is about twelve times larger than
Alaska is about twelve times larger than New
Alaska is about twelve times larger than New York

Left-to-right prediction

York
New York
than New York
larger than New York
times larger than New York
twelve times larger than New York
about twelve times larger than New York
is about twelve times larger than New York
Alaska is about twelve times larger than New York

Right-to-left prediction

Word prediction using context from both sides (e.g. BERT)

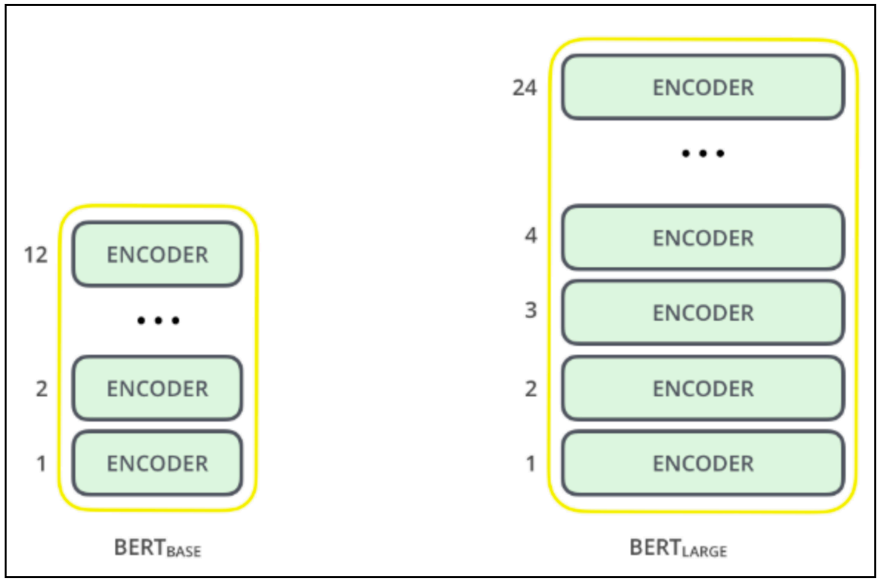
Alaska is about twelve times larger than New York
Alaska is about twelve times larger than New York
Alaska is about twelve times larger than New York
Alaska is about twelve times larger than New York

Next Sentence Prediction

- A text dataset of 100,000 sentences has 50,000 training examples or pairs of sentences as training data.
  - For 50% of pairs, the second sentence would be the next sentence to the first sentence
  - For the remaining 50% of pairs, second sentence would be a random sentence from corpus
    - The labels for the first case would be ***'IsNext'*** and ***'NotNext'*** for the second case

BERT Architecture

- BERT architecture builds on top of Transformer
- There are two variants:
  - BERT Base:
    - 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
  - BERT Large:
    - 24 layers (transformer blocks), 16 attention heads and, 340 million parameters



Summary and Conclusion

- Attention has revolutionised NLP models
  - Now a standard fixture for NLP
  - Because it enables model to “remember” all words in the input and focus on specific words when formulating a response
- Early models: seq2seq models with RNNs
- Recent: Self-Attention and Transformer models
- Google’s BERT, OpenAI’s GPT and XLNet are more popular today
  - Based on self-attention and Transformer

References

1. Encoder-decoder RNN
   - Jurafsky and Martin, *Speech and Language Processing*, 2020
2. Transformer Attention Models
   - A. Vaswani, et.al., *All you need is Attention*, NIPS 2017, https://arxiv.org/abs/1706.03762
3. Generative Pre-training
   - G. Chatel Disaitek, *Generative Pretraining of Transformer Networks* https://nlpparis.files.wordpress.com/2019/01/gchatel_pretrained_transformer.pdf
4. GPT
   - J. Alammar, The Illustrated GPT-2 (Visualizing Transformer Language Models) https://jalammar.github.io/illustrated-gpt2/
5. BERT
   - M. Rizvi, Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework

 https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-. framework/