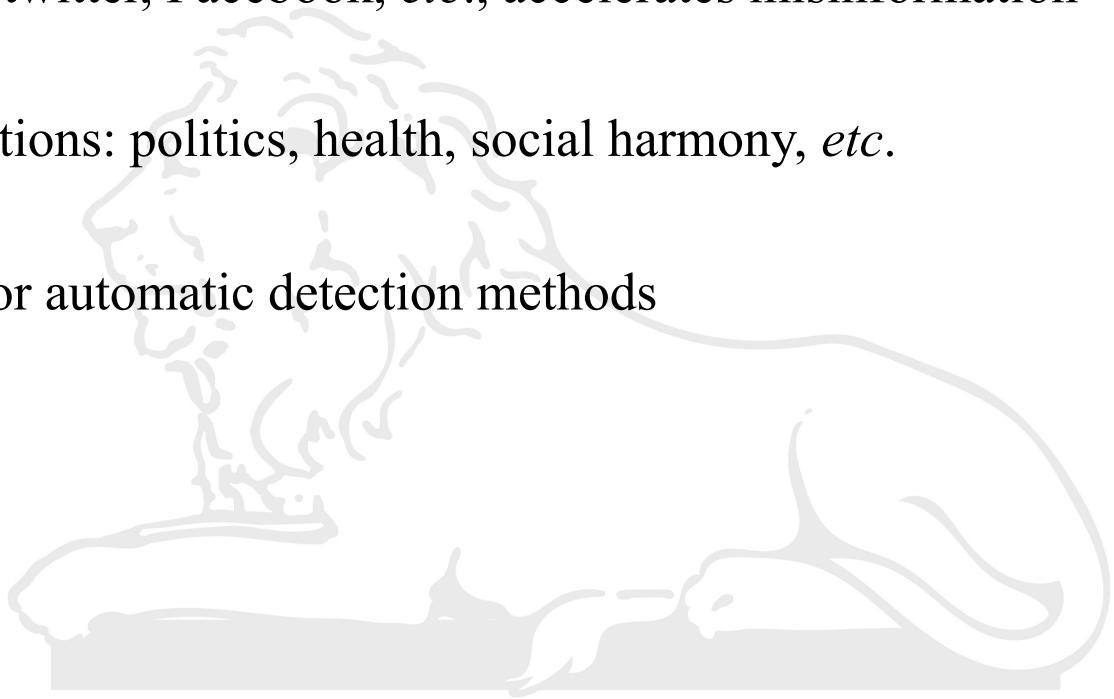
Misinformation and Transformers as Languages Models

Prof. Raksha Sharma
Associate Professor

Department of Computer Science and Engineering
IIT Roorkee

Misinformation/Rumour/Fake-News

- Fake news = False or misleading information spread online
- Social media- twitter, Facebook, *etc*., accelerates misinformation
- Major implications: politics, health, social harmony, *etc*.
- Urgent need for automatic detection methods



Misinformation/Rumour/Fake-News: Challenges

- *Huge volume and speed of information spread*
- *Multilingual and multi-modal data*
- *Sophisticated fake content (deepfakes, memes)*
- *User bias and echo chambers*
- *Balancing free speech vs. regulation*

Misinformation/Rumour/Fake-News: Techniques

- Content-based detection (text classification, NLP, ML, DL)
- Context-based detection (user behavior, network analysis)
- Fact-checking and knowledge graphs
- Hybrid approaches combining multiple signals

Misinformation and Domains

2016 U.S. Presidential Election:
Pope Francis endorses Donald Trump //completely false, but shared millions of times.

Indian Elections (2019 & 2024):
Fake videos and morphed images circulated to mislead voters, such as altered speeches or false claims about political leaders.

COVID-19 Pandemic:
Drinking hot water or alcohol cures COVID
//https://www.bbc.com/future/article/20200403-coronavirus-will-hot-drinks-protect-you-from-covid-19
or
cow-excrement (ie both cow urine and cow dung) as a remedy for Covid-19
//https://ijme.in/articles/narrative-based-misinformation-in-india-about-protection-against-covid-19-not-just-another-moo-point/?galley=print

Polio & Measles Vaccines:
Misinformation campaigns on WhatsApp in Nigeria and Pakistan falsely claimed vaccines cause infertility. //https://www.nature.com/articles/s41598-025-92731-0

Misinformation and Domains

Mob Violence in India (2018):

Five men have been lynched by a mob in India's western state of Maharashtra allegedly over rumours of child abduction spreading over WhatsApp.

//https://www.bbc.com/news/world-asia-india-44678674

Fake videos and conspiracies fuel falsehoods about Los Angeles protests (June 2025)

https://www.cbsnews.com/news/fake-videos-conspiracies-falsehoods-los-angeles-protests/

Russia–Ukraine War (2022):

Fake images/videos of bombings or troop movements spread rapidly, sometimes generated or altered by AI/deepfakes.

Entertainment:

Amitabh Bachchan hospitalized in critical condition

https://www.thip.media/health-news-fact-check/fact-check-is-amitabh-bachchan-in-icu-with-critical-liver-damage/117377/

Misinformation and Domains: Operation Sindoor

Mob Violence in India (2018):

Pakistan's defence minister stating that Indian soldiers were captured during the strikes

https://www.hindustantimes.com/india-news/how-india-is-fighting-pakistan-s-disinformation-campaign-101746644575505.html

False claims surfaced online alleging severe damage to the Indian Armed Forces, the capture of military installations, the downing of fighter jets, and attacks on prominent religious sites.

https://www.newindianexpress.com/nation/2025/May/10/indias-fcu-battles-pakistans-digital-propaganda-with-swift-rebuttals-following-operation-sindoor

Data Sources in Social Media Monitoring

User-generated content: Posts, tweets, comments, reviews.

Engagement metrics: Likes, shares, followers, watch time.

Network data: Social graphs, followers-following patterns, communities.

Multimedia content: Videos, memes, images.

Location and temporal data: Check-ins, timestamps, trending patterns.

Fake news Detection

- Using any ML algorithm like Sentiment Analysis Settings (Logistic Regression with TFIDF)
- Using Bi-directional Encoders
- Using Decoders/Large Language Models
- Combination of BERT and Decoders

Transformers: *Attention all you need*

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Transformers generating word Meanings/Representations

Goal: Derive a perfect representation of words from context words as per distributional hypothesis.

- Not all context words have equal linguistic relationship with each other.
- A word having rich linguistic relationship may be far away pervades language.

"*We remember those incidents more for which we pay more attention*"

~

"*The true art of memory is the art of attention*" Samuel Johnson, Idler #74, September 1759

Problem with Word2Vec/GloVe

They are static! The embedding for a word doesn't reflect how its meaning changes in context.

The chicken didn't cross the road because it was too tired

What is the meaning represented in the static embedding for “it” without seeing context words?

Intuition: a representation of meaning of a word should be different in different contexts!

Contextual Embedding:

- Each word has a different vector that expresses different meanings depending on the surrounding words.
  - **Each word w will be represented by a different vector each time it appears in different context, hence two different embeddings for ‘river bank’ and ‘financial bank’**

Transformers generating word Meanings/Representations

Meaning/representation of ‘it’ will define ‘it’ refers to what-

The chicken didn’t cross the road because it was too tired.
The chicken didn’t cross the road because it was too wide.

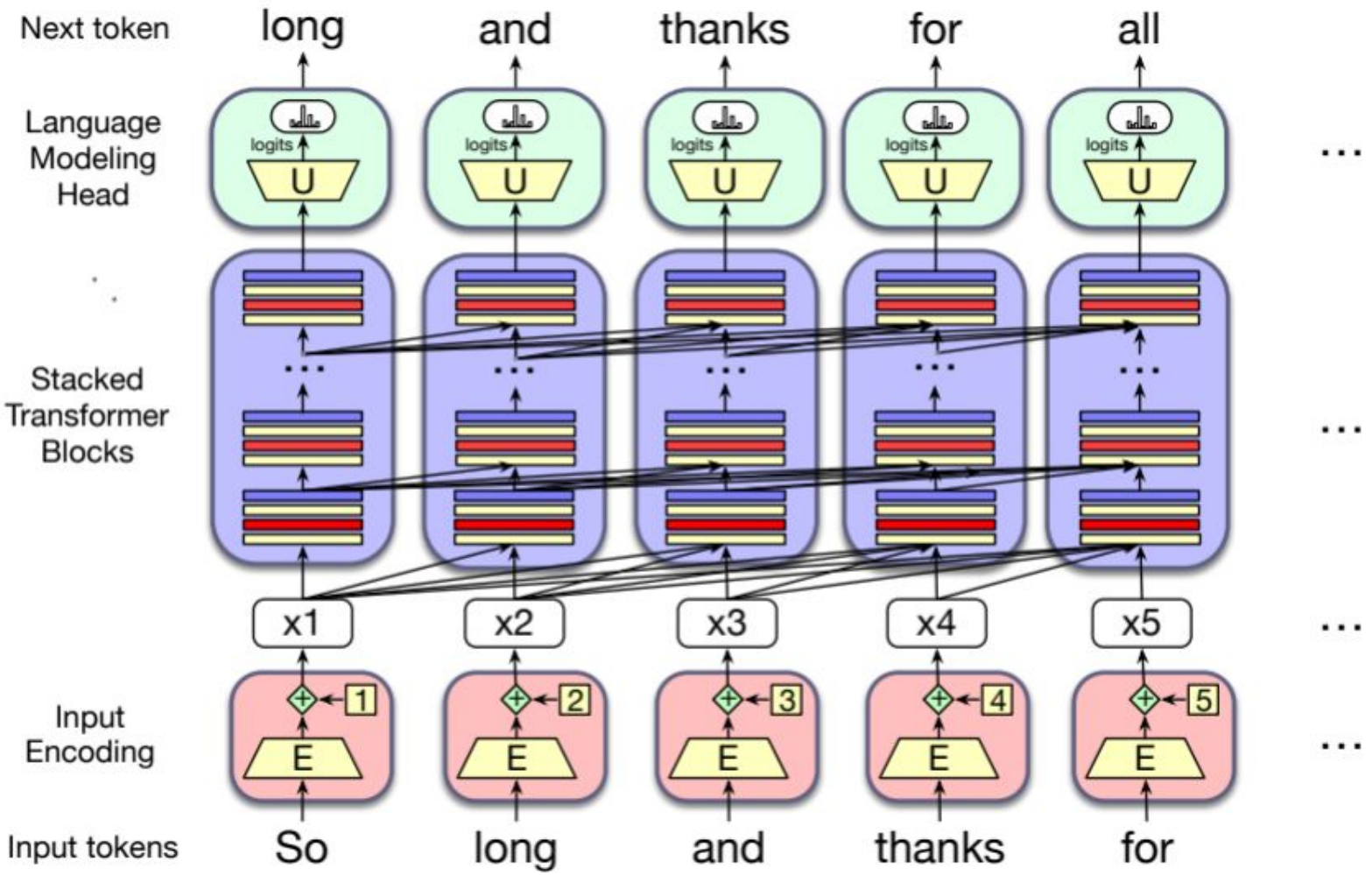
Deriving the precise linguistic properties of ‘are’ as plural and ‘bank’as river bank -

The keys to the cabinet are on the table.
I walked along the pond, and noticed one of the trees along the bank .

Inference: The contextual words that help to derive the meaning of words can be far away in the sentence and paragraph.

Transformers provide a solution by giving attention to a long context window.

Transformer as Language Models



The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.

Input to Transformer (Same for MLM)

- The input to transformer is a series of subword tokens, computed by 3 popular tokenization algorithms -
  - Byte Pair Encoding, (BPE)
  - WordPiece algorithm and the
  - SentencePiece Unigram LM
- Word embeddings for all input tokens are intialized using E embedding matrix and combined with positional embeddings to form the input to transformer

Transformers generating word Meaning: Contextual Embeddings

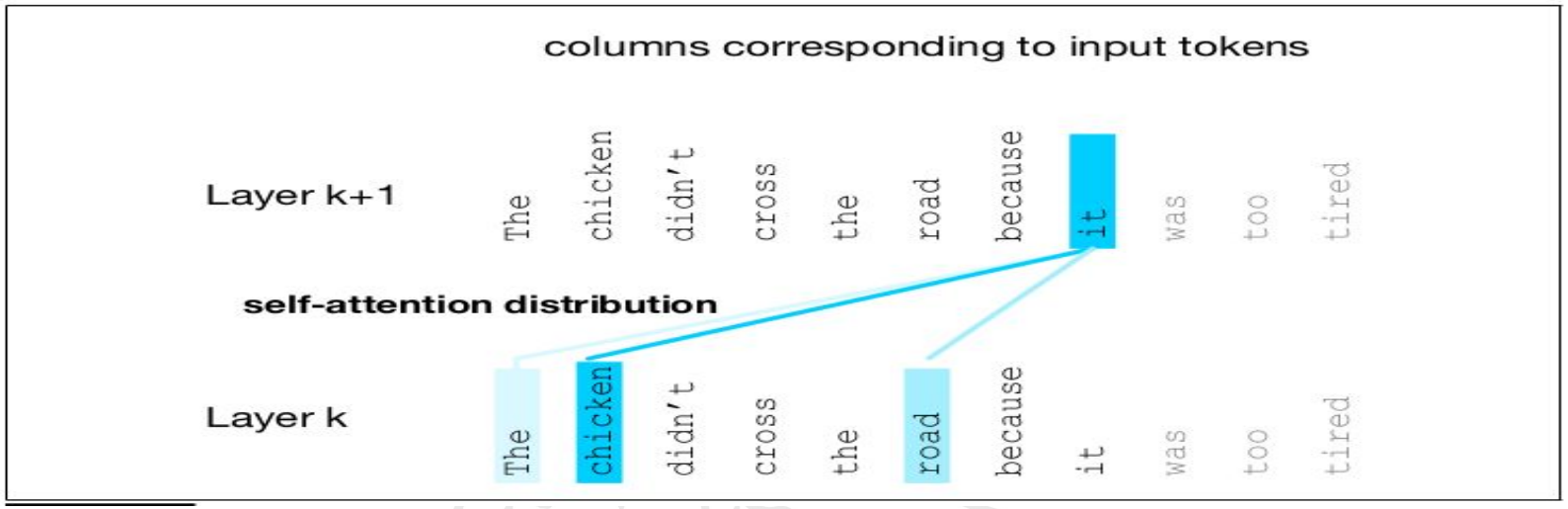
Contextual Embeddings: Transformer learns contextual embeddings by integrating the representations of contextual words.

Very informative embeddings: It gets Richer and richer contextualized representation layer by layer in a transformer.

Binding of the network:

- Compute representation of token *i* by combining information about *i* from previous layer with information about the neighbouring tokens from the previous transformer blocks belonging to i-1, i-2… upto context window size.
- The input for layer k+1 comes from layer k for all neighbouring words and current word. -> Using attention mechanism

Transformers generating word Meanings/Representations



Shades of Blue->attention distribution

Representation of 'it' depends heavily on representation of 'chicken' and 'road'.

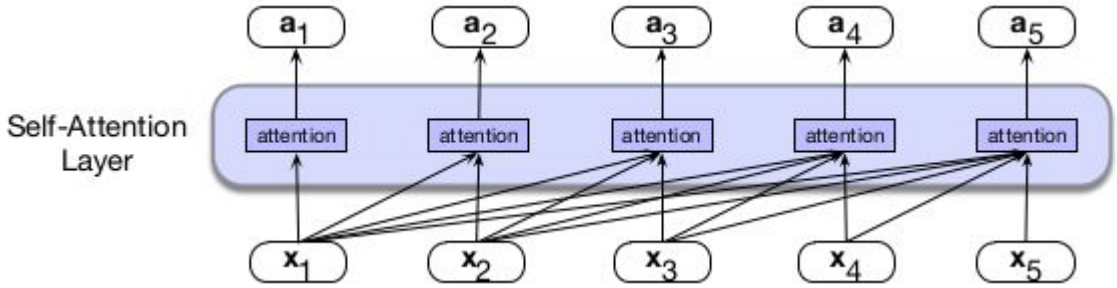
Representation of 'it' to draw on the representation of these two earlier words.

Transformer: Attention Computation at attention layer

Attention distribution: Compute vector representations for a token at layer k+1 of transformer by selectively attending to and integrating information from prior tokens at layer k.

- Left to right (causal or autoregressive) language modeling.

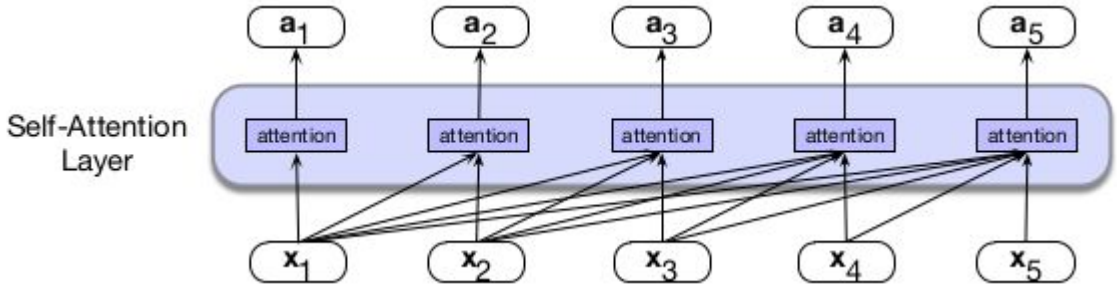
Input: xi corresponding to the input token at position i, and prior tokens in a context window x1 ..xi−1. //no token after i
Output: Produces an output a_i.



- Attention computation happens in parallel at each token position i.
- Hence, self-attention layer maps input sequences (x1 , ..., xn) to output sequences of the same length (a1 , ..., an), attention distribution.

Transformer: Attention vector a_3

- We compute a3 by computing three scores: x3 · x1 , x3 · x2 and x3 · x3 , normalizing them by a softmax, and using the resulting probabilities as weights indicating each of their proportional relevance to the current position i.
- Softmax weight will likely be highest for xi , since xi is very similar to itself, resulting in a high dot product.
- Use these weights as the α values to compute the weighted sum that is our a3 .



Transformer: Attention vector a_i

$\mathbf{a}_i$ (attention output) at token position i// a vector of size x_i:

weighted sum of all the embeddings x_j, for all j<=i. Weighted by their similarity to x_i using α_ij as weights

$$\textit{Simplified version:} \quad \mathbf{a}_i = \sum_{j \le i} \alpha_{ij} \mathbf{x}_j$$

α_ij are weights - It controls how much neighbour x_j contribute to a_i.

Given a sequence of token embeddings:

x1 x2 x3 x4 x5 x6 x7 x_i

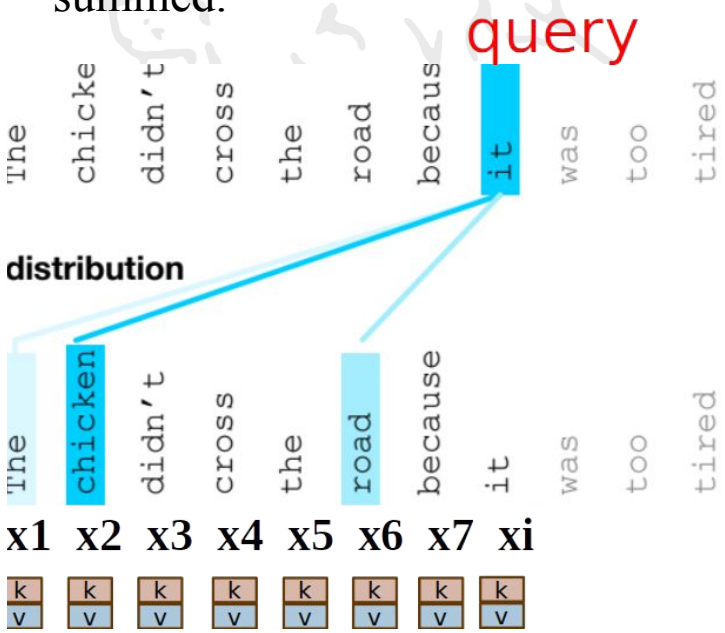
Produce: a_i= a weighted sum of x1 through x7, and xi

$$\textit{Simplified Version:} \quad \text{score}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \; \forall j \le i$$

An Actual Attention Head: Slightly more complicated

- Each input vector xi plays 3 different roles, hence transformer captures three different representation for each role.
  - query (q_i): As the current element being compared to the preceding inputs.
  - key (k_i): as a preceding input that is being compared to the current element to determine a similarity.
  - value (v_i): a value of a preceding element that gets weighted and summed.



An Actual Attention Head: Slightly more complicated

- Depicting computation of Single self attention output vector a_i from single input vector x_i.
- Transformer introduce weight matrices to project each input vector x_i into a representation of its role as query, key, value:
- query: W_Q
- key: W_K
- value: W_V

$$\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{V}}$$

// Normalize the dot product by a factor related to the size (d_k) of the query or key embeddings.

$$\text{score}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}}$$

$$\alpha_{ij} = \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \ \forall j \le i$$

$$\mathbf{head}_i = \sum_{j \le i} \alpha_{ij}\mathbf{v}_j$$

// Weighted sum over the value vectors v of size 1*d_v

$$\mathbf{a}_i = \mathbf{head}_i\mathbf{W}^{\mathbf{O}}$$

// Multiplication of W^o to get output of $head_i$ back as [1*d] dimensions. W^o is of [d_v * d].

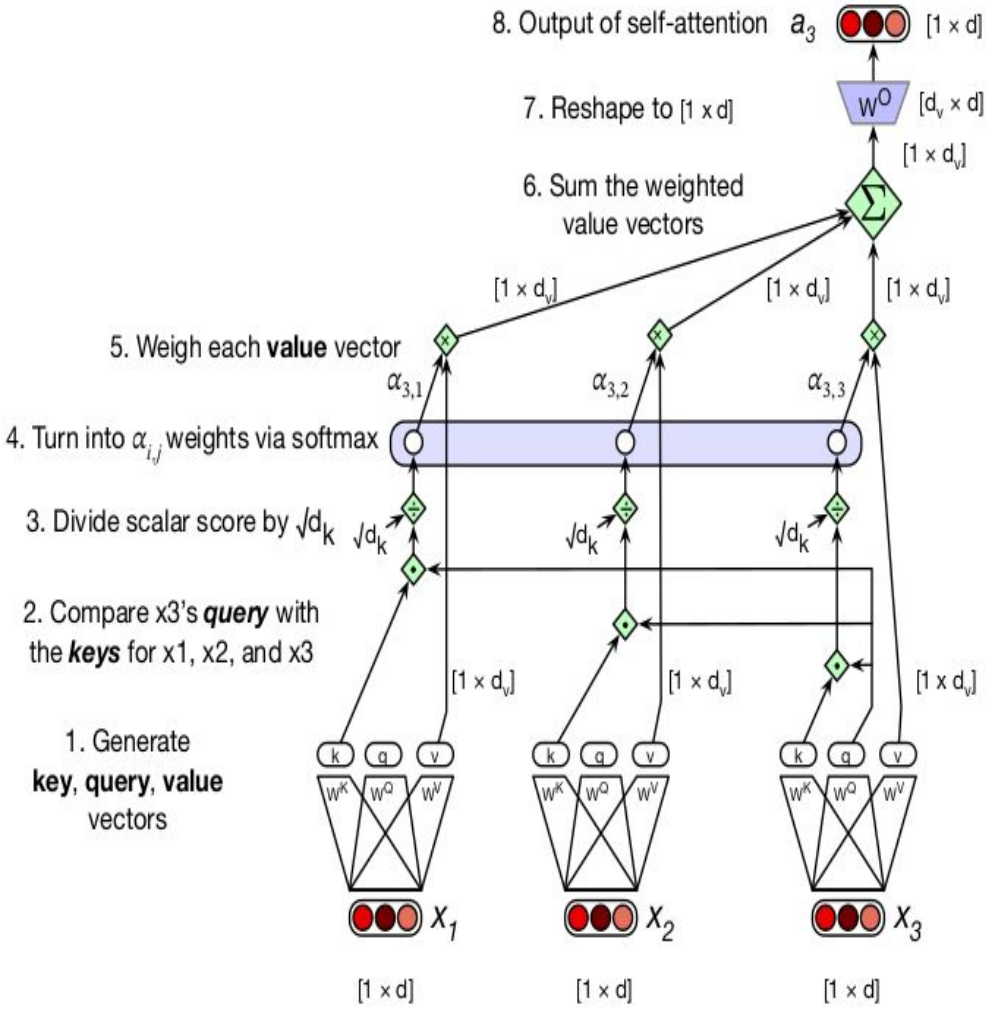
As per Vaswani et al., 2017
*x_i is of dimension 1*d = 1*512.*
*d_k or d_v = 1*64, dimensions of Query and Key vectors.*

d_q will be of dimension d_k always as they participate in dot product computation.

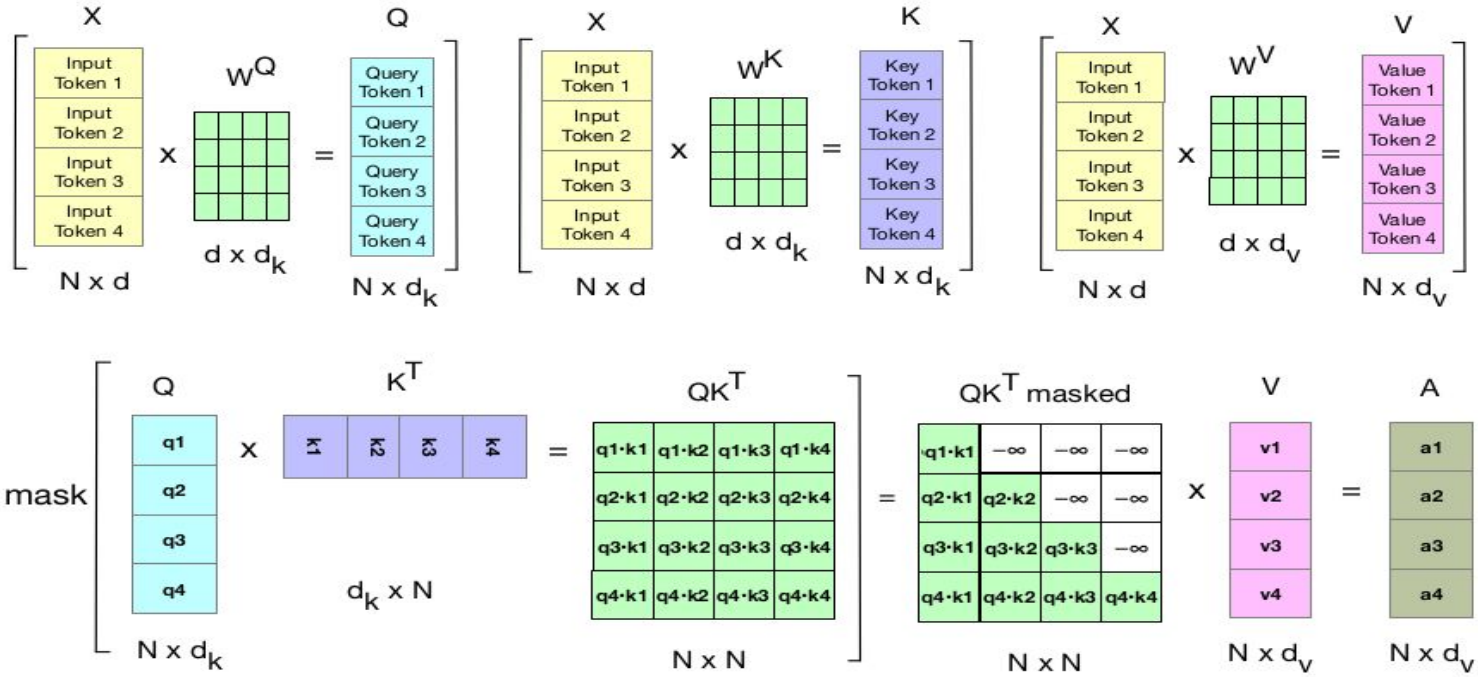
Transformer: Actual Attention Score for a_3

- [1*d] is the model dimensionality, Transformer blocks' input and output dimensionality.

- The transform matrix W_Q has shape $[d \times d_k]$, W_K is $[d \times d_k]$, and W_V is $[d \times d_v]$ and W_O is of shape $[d_v \times d]$.



Parallelizing computation of Attention using a single matrix X

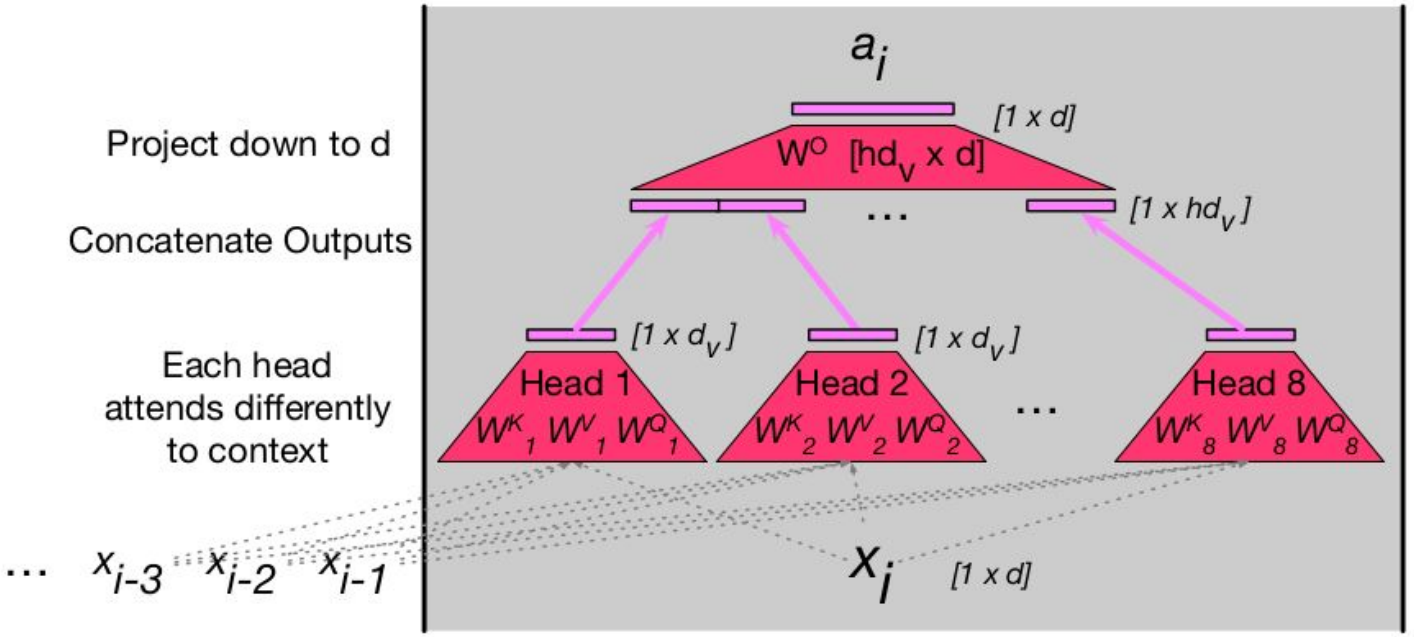


- Pack input embedding for the N tokens (context length) into a single matrix X of size [N*d]. // N is from 1K to 32K to 128K or million of tokens for LLMs.

- The first row shows computation of Q, K and V matrices.

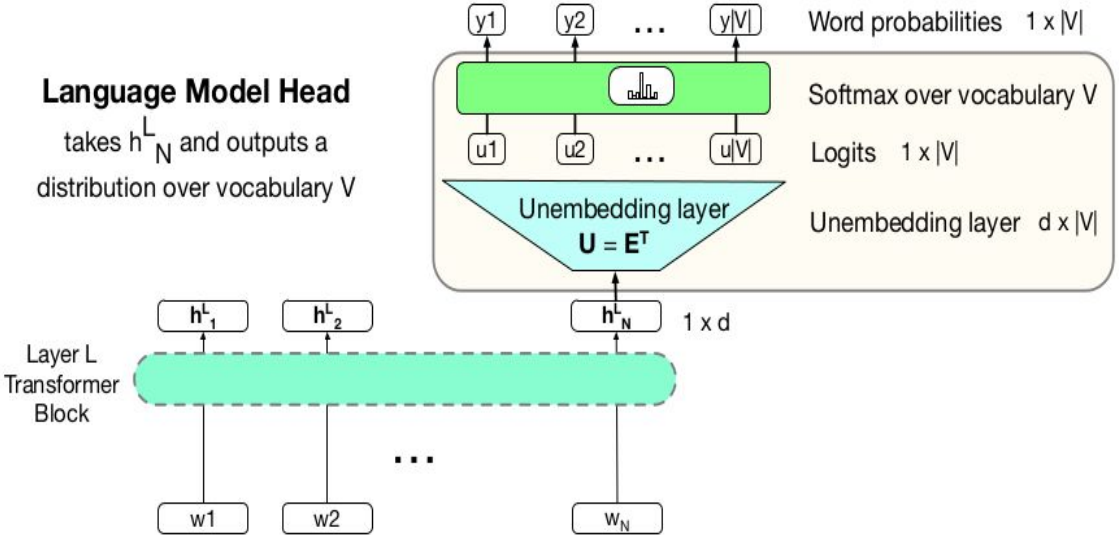
- The second row shows computation of QK^T , (Masking of QK^T, then Normalization by square-root of d_k and softmax over all values of QK^T
- Multiply resultant QK^T by V to obtain attention vector.

Multi-head Attention



- In Multi-head attention, we have A separate attention heads, working parallely at the same depth.
- Each head i in a self-attention layer has its own set of key, query and value matrices:
- W^{Ki} , W^{Qi} and W^{Vi} .
- Out of each head is concatenated. The final a_i will be of model dimension that is [1*d]

Transformer's Language Modeling Head



Objective: Language modeling head takes the output of the final transformer layer from the last token N and use it to predict the upcoming word at position N + 1.

Unembedding Layer (E^T): The embedding matrix E at the input stage (of shape [|V | × d]) is used to map from a one-hot vector over the vocabulary (of shape [1 × |V |]) to an embedding (of shape [1 × d]). The same E (the transpose of the embedding matrix (of shape [d × |V |])) is used to map back from an embedding (shape [1 × d]) to a vector over the vocabulary (shape [1×|V |]) in the Language modeling head.

Logits (u): is a linear layer produces a logit vector u of size [1*v], containing a single score for each or the |V| possible words.

Softmax(u): Turns the vector u into probabilities y over the vocabulary.

Transformer as Decoder: Transformer for word prediction when input is w_i

L layers or L transformer blocks: Each block has 4 processing layers

-

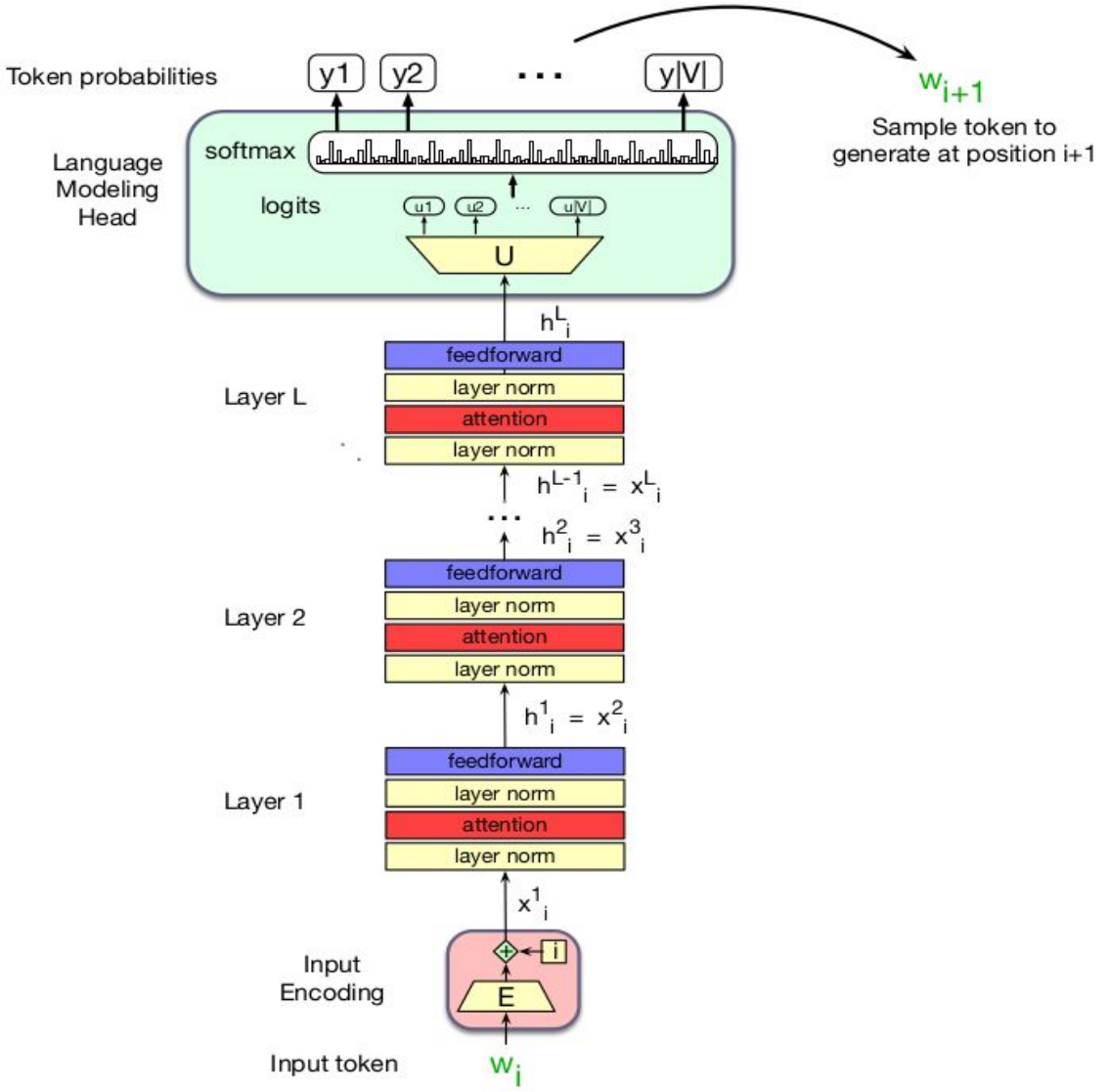
Layer Norm: To bring vector in trainable range by Gradient descent. Can use Mean normalization for this.

Attention Layer: To compute attention representation of token i.

Layer Norm

Feedforward: Two layers (one hidden and one output) fully connected network.

$$
\begin{aligned}
\mathbf{T^1} &= \text{LayerNorm}(\mathbf{X}) \\
\mathbf{T^2} &= \text{MultiHeadAttention}(\mathbf{T^1}) \\
\mathbf{T^3} &= \mathbf{T^2} + \mathbf{X} \\
\mathbf{T^4} &= \text{LayerNorm}(\mathbf{T^3}) \\
\mathbf{T^5} &= \text{FFN}(\mathbf{T^4}) \\
\mathbf{H} &= \mathbf{T^5} + \mathbf{T^3}
\end{aligned}
$$



Masked Language Models: Bi-directional Encoders (BERT) by Devlin et. al., 2019

- Causal LMs are meant to improve next word prediction, hence embeddings are learnt based on left context only.

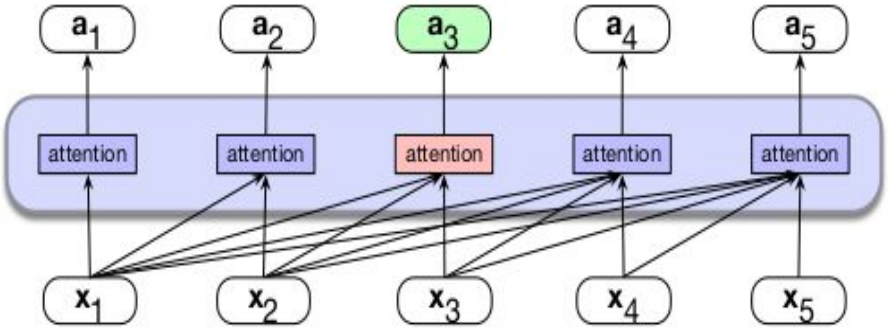
- In reality, context on both sides define the linguistic properties of the words, and may produce more informative embeddings.

- Bi-directional encoders is focused on learning more informative context vectors of words, very suitable for NER, POS Tagging, Parsing.

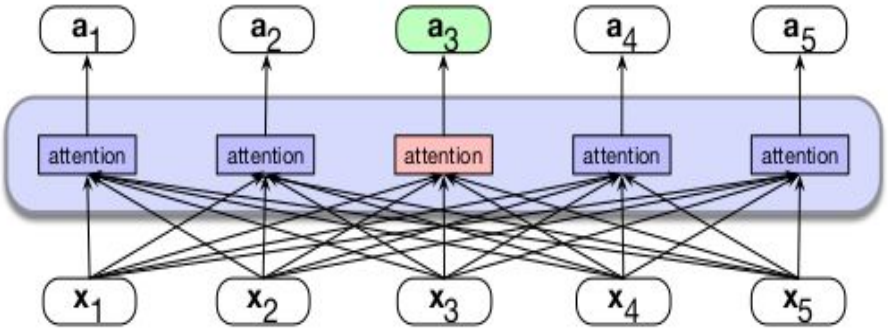
- BERT is trained via masked language modeling - we ask model to predict a masked word in between given the both the left and right context.

- Are known as Encoder only model.

Masked Language Models: Self Attention Layer



a) A causal self-attention layer



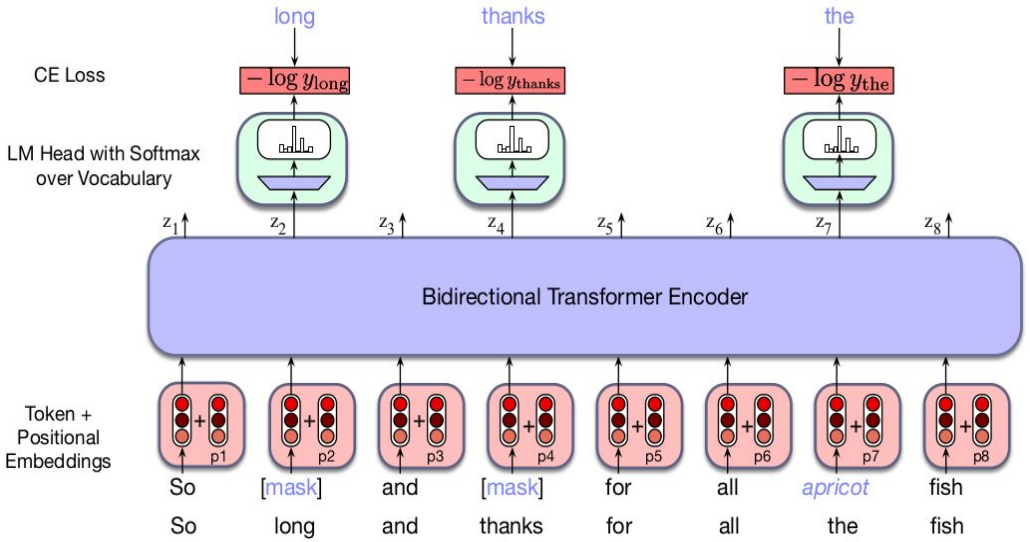
b) A bidirectional self-attention layer

Masked Language Models (MLM):Training

- The input to the MLM is the same as Causal Transformer, however 15% tokens are masked (manipulated) in the input sequence for prediction.
- MLM's Training objective: Predict the masked token and the average cross-entropy loss over all masked tokens from the basis for the weight updates by gradient descent. prediction derives the training of all the parameters of the model.

$$L_{MLM} = -\frac{1}{|M|}\sum_{i \in M} \log P(x_i|\mathbf{h}_i^L)$$

//Average cross entropy loss over all masked tokens.
// h^L computation is exactly same as causal transformer

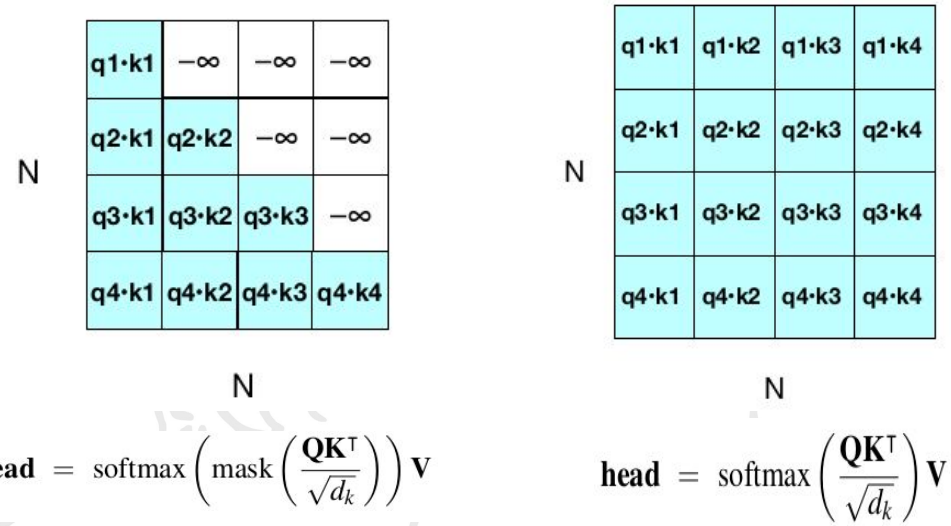


- *Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to*

these three items are used as the training loss. The other 5 tokens don't play a role in training loss.

Masked Language Models: Self Attention Layer

- Computation of attention head is identical to causal Transformer's attention layer, except we remove the attention masking step from the computation of attention vector.



Masked QKT of Causal transformer and Unmasked QK^T of Bi-directional Encoder.

BERT's Configuration (By Devlin et al., 2019)

- An English-only subword vocabulary consisting of 30,000 tokens generated using the WordPiece algorithm (Schuster and Nakajima, 2012).
- Input context window N=512 tokens, and model dimensionality d=768
- So X, the input to the model, is of shape [N × d] = [512 × 768].
- L=12 layers of transformer blocks, each with A=12 (bidirectional) multihead attention layers.
- The resulting model has about 100M parameters.

Large Multilingual XLM-RoBERTa's Configuration (trained on 100 languages)

- A multilingual subword vocabulary with 250,000 tokens generated using the SentencePiece Unigram LM algorithm (Kudo and Richardson, 2018).
- Input context window N=512 tokens, and model dimensionality d=1024, hence X, the input to the model, is of shape [N × d] = [512 × 1024].
- L=24 layers of transformer blocks, with A=16 multihead attention layers each
- The resulting model has about 550M parameters.

Note that 550M parameters is relatively small as large language models go (Llama 3 has 405B parameters, so is 3 orders of magnitude bigger).

Fine-tuning of pre-trained models

- Take a pre-trained transformer network (will give information rich contextual word representations), add a neural net classifier (head) after the top layer of the network, train the entire network on some additional labeled data to perform some downstream task like NER and language inference.

- This instance of fine-tuning is a transfer learning task.

For SA using BERT

https://www.kaggle.com/code/kritanjalijain/movie-review-sentiment-analysis-eda-bert