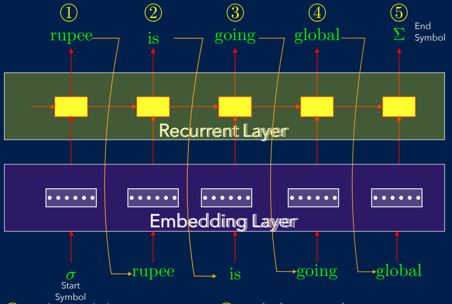
Large Language Models

Landscape, Scaling and Responsible Use

Ramaseshan Ramachandran

RECURRENT TEXT GENERATOR/TRANSDUCER MODEL



1 $p(rupee|\sigma)$

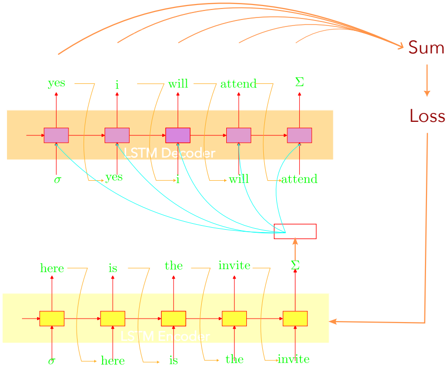
2 $p(is|\sigma, rupee)$

3 $p(going|\sigma, rupee, is)$

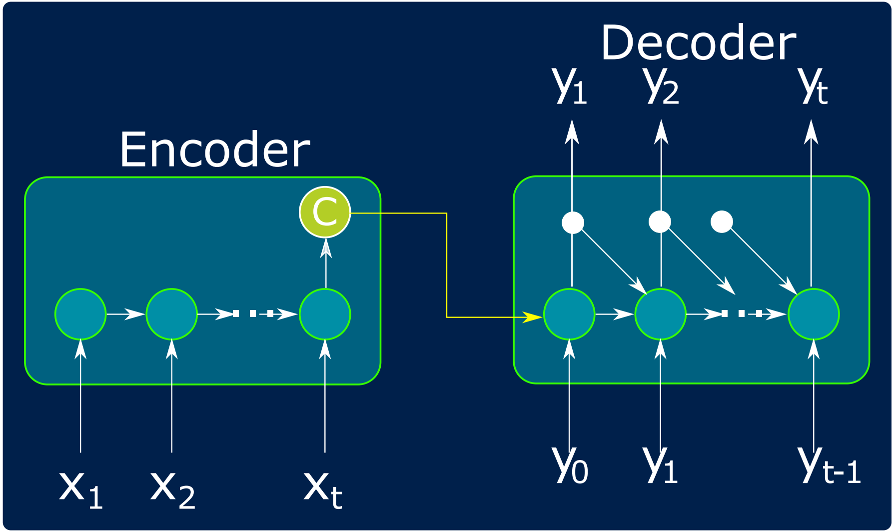
4 $p(global|\sigma, rupee, is, going)$

5 $p(\Sigma|\sigma, rupee, is, going, global)$

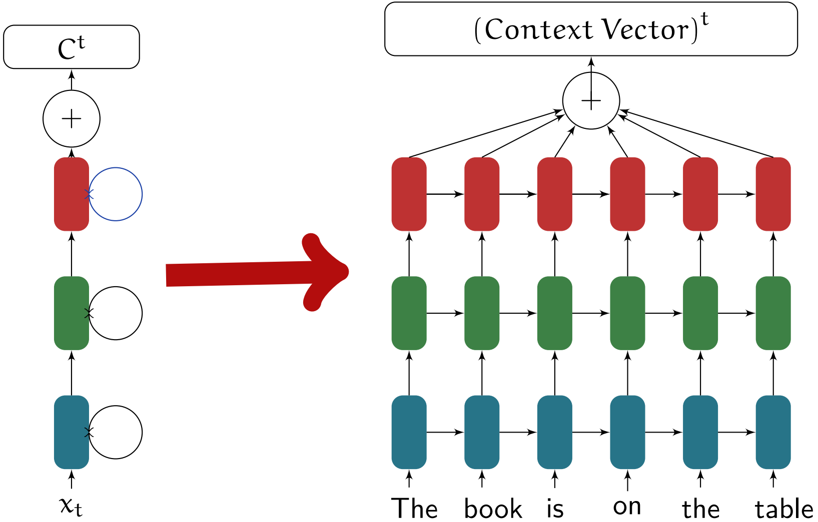
SEQUENCE TO SEQUENCE GENERATOR



RNN-BASED ENCODER-DECODER - SEQ2SEQ TRANSLATION



SEQ2SEQ ENCODER



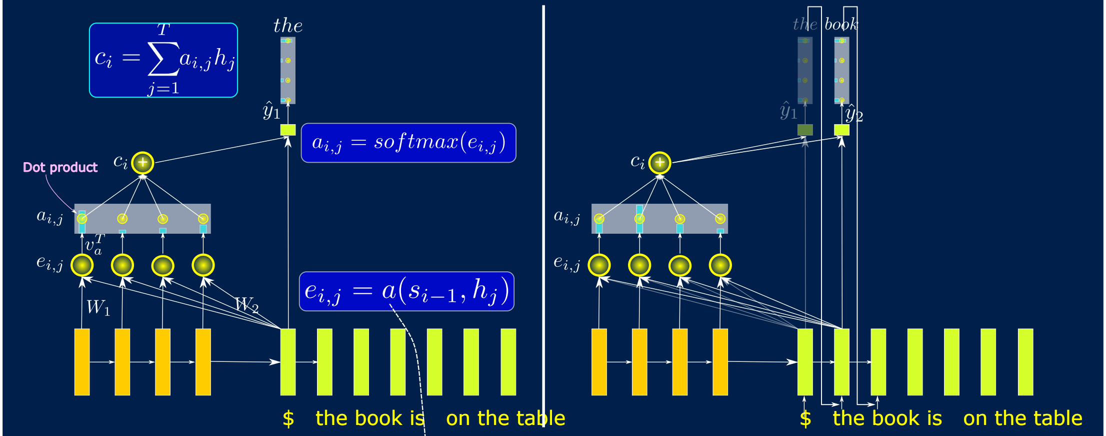
BAHDANAU ET AL. NMT MODEL

Input is a sequence of words $x_1, \ldots, x_n$
Target sentence is again a sequence of words $y_1, \ldots, y_m$

Encoder
Let $(h_1, \ldots, h_n)$ be the hidden vectors representing the input sentence. These vectors are the output of a bi-LSTM/bi-GRU for instance, and capture contextual representation of each word in the sentence

Decoder
The hidden states s_i of the decoder are computed using a recursive formula of the form $s_i = f(s_{i-1}, y_{i-1}, c_i)$, where s_{i-1} is the previous hidden vector, y_{i-1} is the generated word at the previous step, and c_i is a context vector that capture the context from the original sentence that is relevant to the time step i of the decoder.

ATTENTION



$$e_{i,j} = v_a^T tanh(W_1 h_j + W_2 s_{i-1})$$

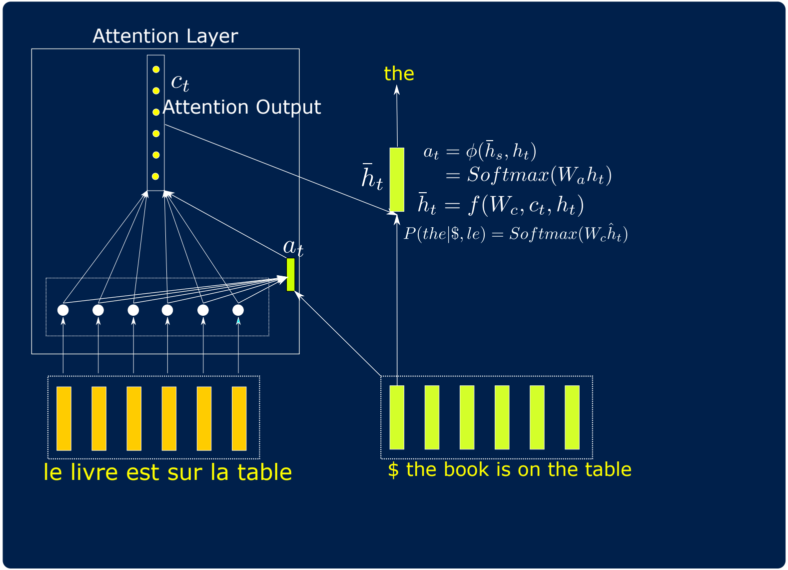
$e_{i,j}$ – attention score

$a_{i,j}$ – attention distribution

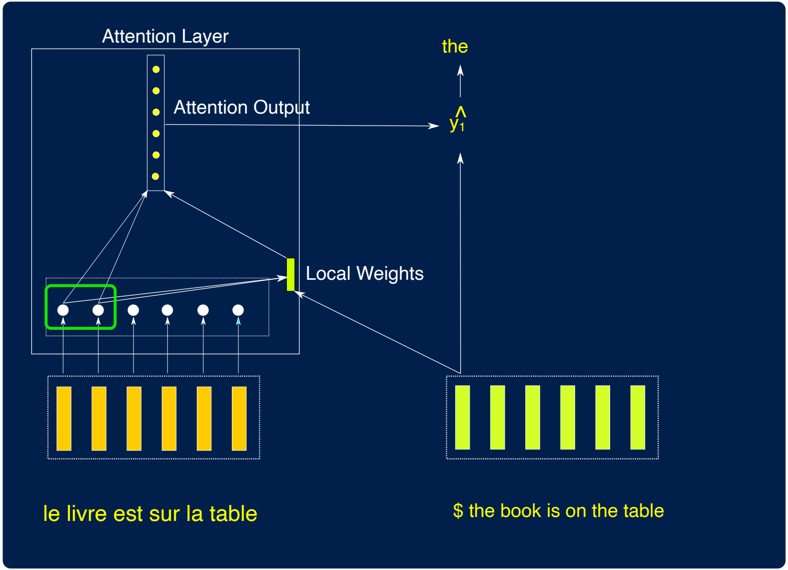
c_i – attention output

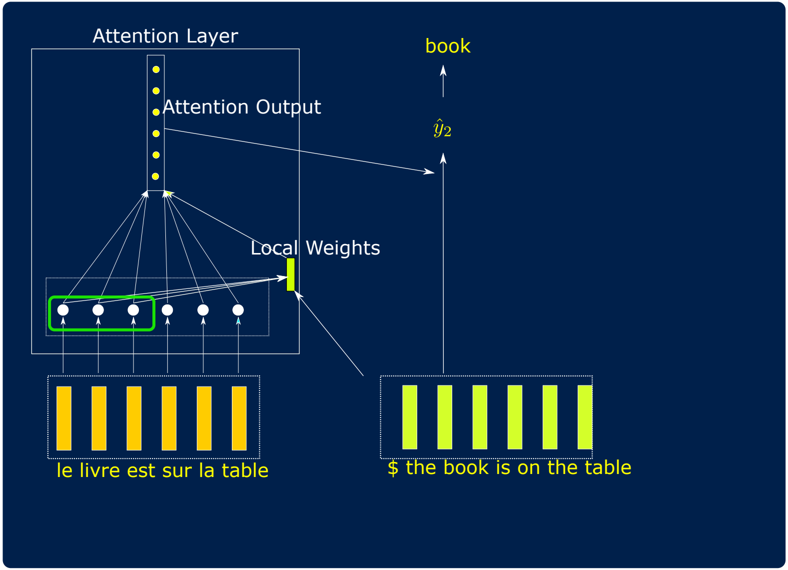
a – Attention scoring Function

TRANSLATION WITH GLOBAL ATTENTION

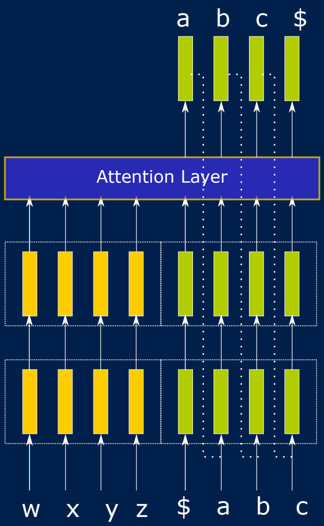


TRANSLATION WITH LOCAL ATTENTION





Attention Layer
book
Attention Output
$\hat{y}_2$
Local Weights
le livre est sur la table
$ the book is on the table



Source: Minh-Thang Luong et al, Effective Approaches to AIttention-based Neural Machine Translation

TRAINING WITH CONTEXT

- The next token in the sequence gets the largest probability mass
- If there are t words in the sequence $(w_1, w_2, w_3, \ldots, w_t)$, the transducer will have $t+1$ input and output neurons

$$\begin{aligned} p(w_{j+1}|w_1, w_2 \ldots, w_j) &= f(\text{RNN}(\theta, w_1, w_2, \ldots, w_j)), \text{ where } j \in [1, t] \\ &= f(\text{RNN}(\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_j)) \end{aligned}$$

  and $\hat{w}_j = p(w_j|\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{j-1})$
- It is possible to create the next token using a conditional context, such as a topic or class of the current content, active or passive voice, etc.
- Example sentence generated without a context
  - *determine the time it takes a piece of glass to hit the ground?* ***A car drives straight off the edge of a cliff***
- Example sentence with context
  - *determine the time it takes a piece of glass to hit the ground,* ***if a ball is dropped from a height of 45m***.

A TYPICAL SETUP

Sentence pairs	3-5M
English words	110M
French words	116M
Vocabulary	≈50K (Source and Target)
Word Embedding size	1000
Hidden layer	1000 LSTM cells
Stacked Hidden Layer	4-8
Learning Rate	Initially as high as 1 and exponential reduction
Training	
Mini batch Gradient Descend size	128
Training Time	1 GPU - about 7-10 days
Evaluation	Bleu - scores ranging from 27-32

SEQ2SEQ-BASED APPLICATION

- **Next word/sentence prediction**

  Train pairs of consecutive sentences to generate a contexnt sensitive sentences in a sequence
- **Question-Anwering**

  Given a question (including word problems), find context sensitive answers
- **Auto-encoding**

  Building sentence vectors for identifying similar sentences - extending the distributional hypothsis of words into sentences
- **Machine Translation**

  Given pairs of language texts for training, trsnslate an unknown sentence usig the trained model
- **Precise writing**

  Given a long paragraph, compress into one two sentences
- **Title generation**

  Given abstract and title pairs, train and generate title for any new abstract
- **Chatbots**
- **Discourse Analysis**

  Analysis of written, spoken, sign language

ADVANTAGES OF ATTENTION

- Ability to focus on significant part of the sentence
- Ability to peek into source sentence
- Reduces the problem of vanishing gradient
- Alignments are found automatically - no need to train
- Improves NMT performance for alignment

SENSE DISAMBIGUATION OF THE WORD BANK

Synset('bank.n.01')	sloping land (especially the slope beside a body of water)
Synset('depository-financial-institution.n.01')	a financial institution that accepts deposits and channels the money into lending activities
Synset('bank.n.03')	a long ridge or pile
Synset('bank.n.10')	a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)
Synset('bank.v.02')	enclose with a bank
Synset('bank.v.03')	do business with a bank or keep an account at a bank
Synset('bank.v.04')	act as the banker in a game or in gambling
Synset('bank.v.05')	be in the banking business
Synset('deposit.v.02')	put into a bank account
Synset('trust.v.01')	have confidence or faith in

SENSE DISAMBIGUATION - THE WORD PROGRAM

Synset('plan.n.01')	a series of steps to be carried out or goals to be accomplished
Synset('program.n.02')	a system of projects or services intended to meet a public need
Synset('broadcast.n.02')	a radio or television show
Synset('platform.n.02')	a document stating the aims and principles of a political party
Synset('program.n.05')	an announcement of the events that will occur as part of a theatrical or sporting event
Synset('course_of_study.n.01')	an integrated course of academic studies
Synset('program.n.07')	(computer science) a sequence of instructions that a computer can interpret and execute
Synset('program.n.08')	a performance (or series of performances) at a public presentation
Synset('program.v.01')	arrange a program of or for
Synset('program.v.02')	write a computer program

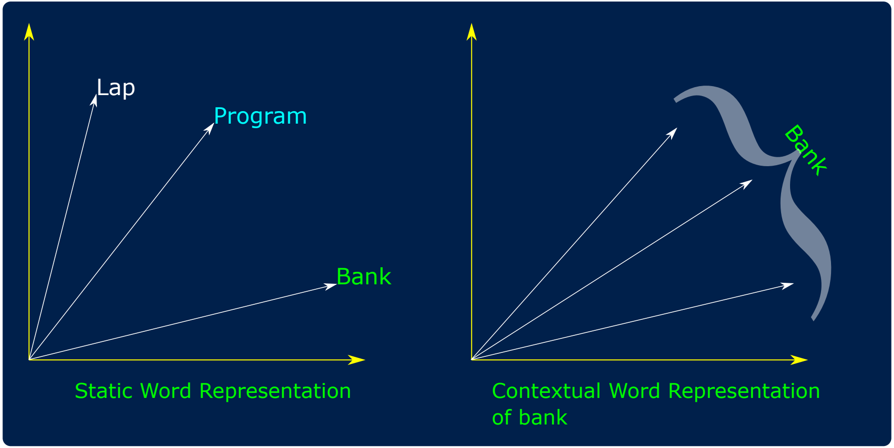
STATIC WORD REPRESENTATION

- Static representation of words
- Irrespective of the context, a polysemous word has the same vector representation
- Insensitive to the context in which they appear
- The word representation of a polysemous word is a biased representation due to certain context/pattern that appear more than any other context in the given corpus
- **It represents syntactic and semantic representations**
- **It does not represent the same word appearing in different linguistic contexts**

LATENT RELATIONSHIP

- The pairs of words that co-occur in similar patterns tend to have semantic relations. Using this property every word in the vocabulary is encoded as dense vectors using vector space semantic models.
- The individual dimensions of the dense vectors are no longer interpretable; a dimension's value merely reflects the strength of the relationship the word has with the rest of the vocabulary
- Probabilistic language models use only nearby contexts; they do not capture relationships across the surrounding sentences of a corpus
- Is it possible to compute the contextual relationship among words, sentences, and paragraphs?

CONTEXTUAL WORD REPRESENTATION



CONTEXTUAL WORD REPRESENTATION

Boys are playing cricket near the Indian **bank**

A fundamental aircraft motion is a **banking** turn

Boys are playing cricket on Cauvery river **bank**

What is your **program** today?

Have you completed the **programming** assignment today?

A cultural **program** was telecast on TV

- The hidden states of an RNN (or the layers of a transformer) hold context-specific representations of the words
- Why not use them to generate contextual word vectors?
- Can we use them for downstream NLP tasks? (This is exactly what ELMo, BERT, and their successors do)
- Each token receives a representation that depends on the whole sentence: $w_i = f(w_1, w_2, w_3, \ldots, w_n)$, where n is the number of tokens in the sentence

ELMO

1. ELMo produces *contextual* word vectors: a word's vector depends on the whole sentence it appears in [1], so $w_i = f(w_1, w_2, w_3, \ldots, w_T)$

$$\text{FORWARD}: \quad p(w_1, w_2, \ldots, w_T) = \prod_k^T p(w_k | w_1, w_2, \ldots\ldots, w_{t-1})$$

$$\text{BACKWARD}: \quad p(w_1, w_2, \ldots, w_T) = \prod_k^T p(w_k | w_{k+1}, w_{k+2}, \ldots\ldots, w_T)$$

2. If there are L layers in both forward and backward LSTMs, then the after L layers, $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are computed
3. $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are scaled (may use a layer norm)
4. $\overrightarrow{h}_{k,L}$ and $\overleftarrow{h}_{k,L}$ are concatenated for downstream applications
5. For each token, ELMo takes a task-specific weighted (linear) combination of its representations across the L layers
6. ELMo uses a character-based input layer, so there is no need for an **unknown-token** representation
7. Contrast: ELMo uses stacked biLSTMs; BERT and later models replaced these

WHAT IS LAYER NORMALIZATION?

- Layer Normalization (LayerNorm) is a technique used to normalize the inputs across the features for each training example.
- It helps in stabilizing the learning process in deep neural networks.

Given a vector $x = [x_1, x_2, \ldots, x_d]$, the layer norm is given by

$$\mathrm{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sigma} + \beta \tag{1}$$

where μ is the mean of x, σ is the standard deviation of x and γ and β are learnable scale and shift parameters, respectively

STEPS OF LAYER NORMALIZATION

1. Compute the mean μ of the input vector x. $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$
2. Compute the variance σ^2 of the input vector x. $\sigma^2 = \frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2$
3. Normalize the input by subtracting the mean and dividing by the square root of variance. $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$ where ϵ is a small constant for numerical stability.
4. Scale and shift the normalized values. $y_i = \gamma\hat{x}_i + \beta$

LAYERNORM VS BATCHNORM

- **Batch normalization** normalizes each feature across the *batch* dimension. It depends on batch statistics and struggles with variable-length sequences and small batches.
- **Layer normalization** normalizes across the *features* of a single example. It is independent of batch size, which suits sequence models.
- This is why transformers use LayerNorm (or RMSNorm) rather than BatchNorm.

LARGE LANGUAGE MODELS

- LLMs are typically trained on massive datasets of text and code
- Large language models transform natural language processing
- Capable of generating human-like text
- Learn patterns, context, and semantic relationships in data
- GPT (OpenAI), Claude (Anthropic), Gemini (Google), and Llama (Meta) are current state-of-the-art LLM families
- Revolutionizing various applications across industries

TYPICAL APPLICATIONS OF LLMS

- Natural language understanding
- Natural language generation
- Machine translation
- Question answering
- Text summarization
- code generation

LANGUAGE UNDERSTANDING

- Comprehends and interprets text with high accuracy
- Captures meaning, context, and nuances in language
- Understands complex queries and generates relevant responses
- Enables chatbots and virtual assistants to communicate effectively
- Enhances machine translation and language comprehension tasks

CREATIVE CONTENT GENERATION

- Generates coherent and contextually appropriate text.
- Creates engaging and original stories, articles, and scripts.
- Assists with content creation for marketing and advertising.
- Provides innovative solutions for writing, gaming, and entertainment.
- Expands creative possibilities for artists and content creators.

KNOWLEDGE EXPANSION

- Accesses vast amounts of information from the Internet.
- Processes and summarizes complex texts and articles.
- Helps in research, data analysis, and decision-making.
- Provides insights and answers to specific questions.
- Enables continuous learning and knowledge sharing

REAL WORLD APPLICATIONS

- Improves customer service through chatbots and support systems.
- Personalizes user experiences in e-commerce and recommendation systems.
- Enhances medical diagnostics and healthcare decision-making.
- Powers virtual assistants and voice-activated technologies.
- Transforms education and online learning platforms.

HOW DO LARGE LANGUAGE MODELS WORK?

- Uses transformer architecture
- Learns long-range dependencies between words
- Generalize the statistical regularities of human language

WHO ARE MY NEIGHBOURS?

Traditional domain in a windowed (ramp or otherwise) approach includes frequency counts (co-occurrence) and correlation measures[2]

- Frequency count - cooccurence
- Correlation

SVD Domain

- The transformation of the incidence matrix into SVD domain leads to define the relationship between two words based on the magnitude of the diagonal matrix Σ

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^T$$

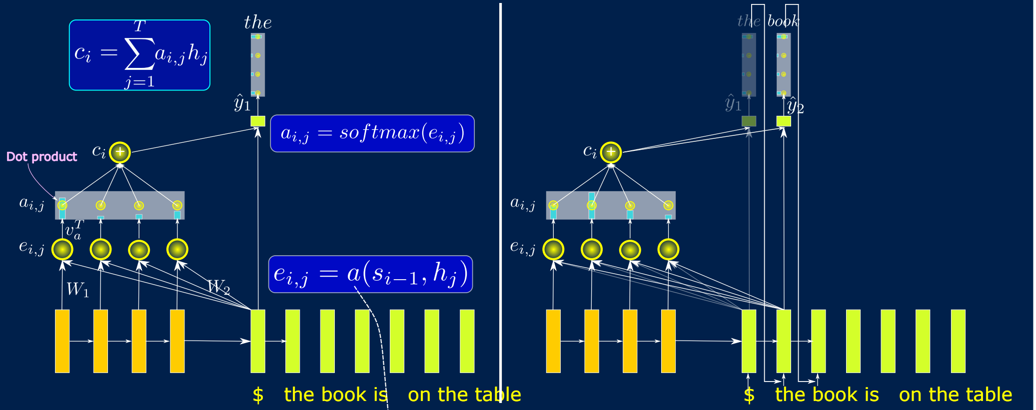
  σ_j capture the strength of latent semantic dimensions
  The low-rank approximation captures as much information/energy as possible with respect to the word relationships

GloVe Domain

- GloVe is a global log-bilinear regression model capturing global corpus statistics. Ratio of the co-occuring word probabilities are transformed into word embeddings[3]

Useful in

- Auto-generation of sentences
- Auto-completion of sentences
- Creating contextualized word-vectors
- Machine Translation



$e_{i,j}$ – attention score

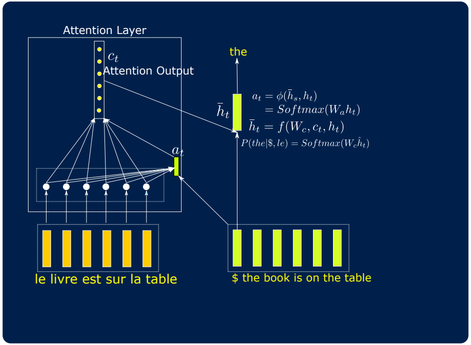
$a_{i,j}$ – attention distribution

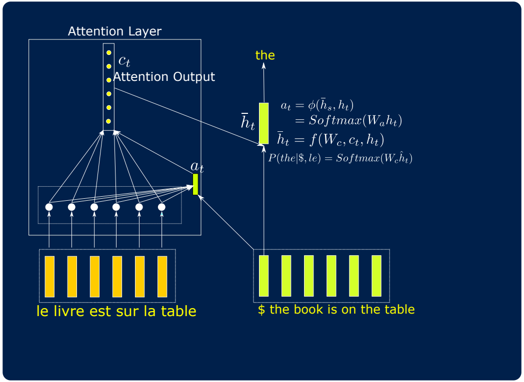
c_i – attention output

a – Attention scoring Function

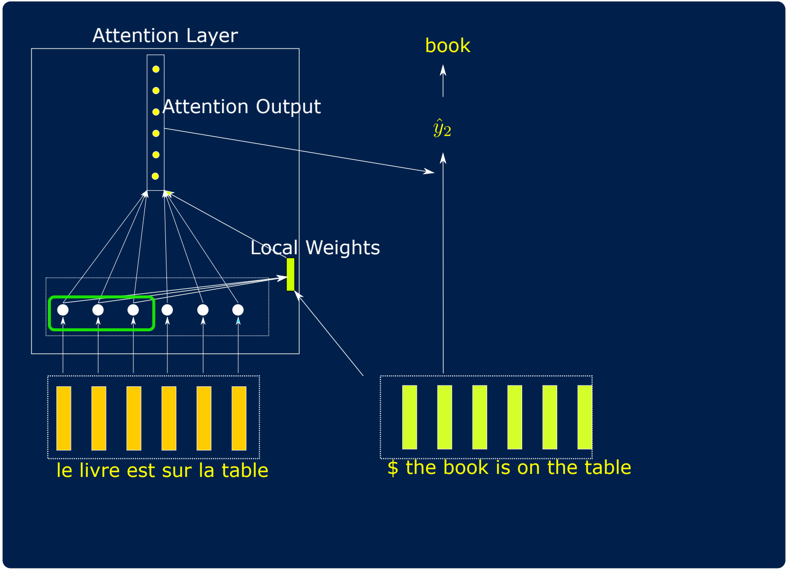
$$e_{i,j} = v_a^T tanh(W_1 h_j + W_2 s_{i-1})$$

ATTENTION: WHERE DID IT BEGIN? [4] II





ATTENTION: WHERE DID IT BEGIN? [4] III



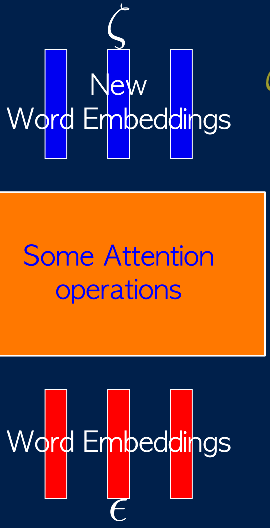
DO I KNOW MORE ABOUT MYSELF AND MY NEIGHBORS?

- How much influence should I receive from my neighboring words??
- How can I effectively capture information from my neighbors?

Ideally word embeddings aim to capture meaningful semantic relations beyond raw counts or co-occurrence frequencies

ANY CHANGE REQUIRED?

- Is it possible to encode the time series data differently?
- How can we manage the complexities of recurrent models (serial processing, vanishing/exploding gradient)?
- Is it possible to encode each word vector differently depending on its context?



ζ will be the weighted combinations of ϵ

Given a set of vector values, and a vector query, attention is a technique to compute a weighted sum of the values, dependent on the query

QUERY-KEY INTERACTION

- Computes relevance scores for each query token by considering all keys from input tokens
- Evaluates its relationship with every other token in the sequence, including itself
- Enables a comprehensive assessment of contextual dependencies

MECHANISM

The process begins with the query matrix Q for a specific token being dot-producted with the entire key matrix K^T, where K encompasses representations from all tokens in the context, resulting in a score for each possible pair.
These scores are then scaled by $\sqrt{d_k}$ and passed through a softmax to produce attention weights, ensuring that the query's focus is distributed across all relevant keys without omission.
This full consideration promotes parallelism and captures global interactions, as opposed to local windows in other architectures, making it efficient for sequences of varying lengths.

MECHANISM OF CONSIDERATION (EQUATIONS)

The scaled dot-product computation:

$$s_{ij} = \frac{q_i \cdot k_j^T}{\sqrt{d_k}} \tag{2}$$

Attention weights via softmax:

$$\alpha = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{3}$$

IMPLICATIONS FOR ATTENTION COMPUTATION

- Generates a weighted sum of value vectors based on the relevance of tokens in the context.
- Token Consideration: No token is excluded initially, allowing the model to prioritize relationships across the entire input.
- Dynamically prioritizes distant or subtle relationships, such as coreferences or syntactic dependencies.

In practice, for a sequence of length n, this leads to an $O(n^2)$ complexity per attention head, but optimizations like masking can limit it in decoder contexts without altering the core all-keys consideration principle.
The final output is:

$$\text{Output} = \alpha V \tag{4}$$

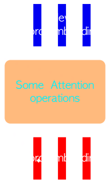
THE CAUSAL (DECODER) MASK

A language model must not see the future: when predicting token i, only positions $j \leq i$ may contribute. This is enforced *before* the softmax:

$$S_{ij} = \frac{(QK^{\top})_{ij}}{\sqrt{d_k}}, \qquad S_{ij} \leftarrow -\infty \text{ for all } j > i \tag{5}$$

- $e^{-\infty} = 0$, so masked positions get **exactly zero** attention weight and each row still sums to 1 over the surviving entries
- Row 0 can attend only to itself, so its output is exactly V_0
- The *last* row is unchanged by the mask: it could already see everything
- Encoder (BERT-style) attention omits the mask; decoder (GPT-style) attention always applies it. This single matrix of $-\infty$s is the difference between "understanding" and "generating"

Masking, then softmax, never the other way round. Renormalisation is what redistributes the blocked probability mass over the visible tokens.



$$A : \epsilon_i \rightarrow \zeta_i$$

W^q $q_i = W^q x_i$

W^k $k_i = W^k x_i$

W^v $v_i = W^v x_i$

$$s_j = q_i k_j$$

Self-attention score for the j[th] word

$$\zeta_j = \sum_i \alpha_i v_i$$

Attention for ϵ_1

1. x_1 has three associative vectors - q_1, k_1, v_1

2. $s_1 = q_1.k_1$
$s_2 = q_1.k_2$
$s_3 = q_1.k_3$

s_i : measures the similarity between q_i and k_i

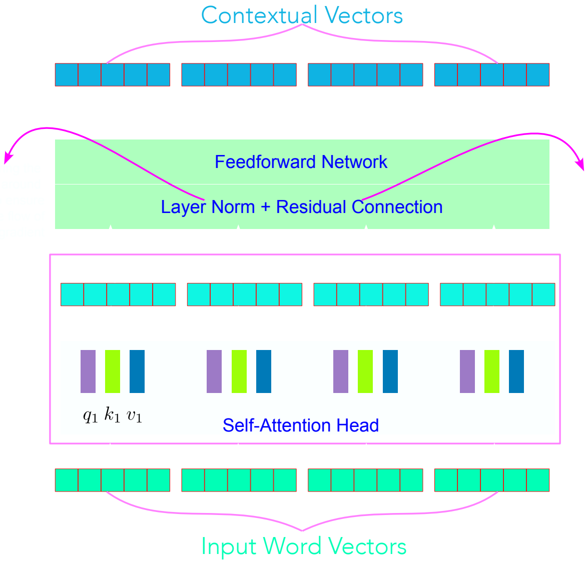
3. $\alpha_1 = \sigma\left(\frac{s_1}{\sqrt{d}}\right)$
$\alpha_2 = \sigma\left(\frac{s_2}{\sqrt{d}}\right)$
$\alpha_3 = \sigma\left(\frac{s_3}{\sqrt{d}}\right)$

$v_1 = W^v x_1$
$v_2 = W^v x_2$
$v_3 = W^v x_3$

4. $\zeta_1 = \alpha_1.v_1 + \alpha_2.v_2 + \alpha_3.v_3$

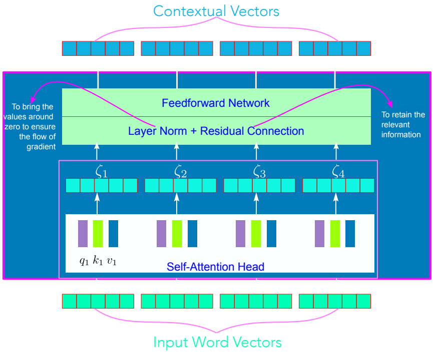
Mapping a query and key-value pair to an output vector

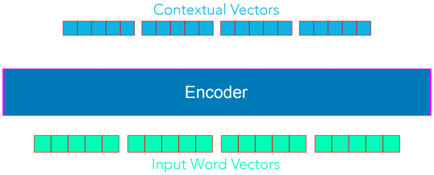
SINGLE TRANSFORMER LAYER

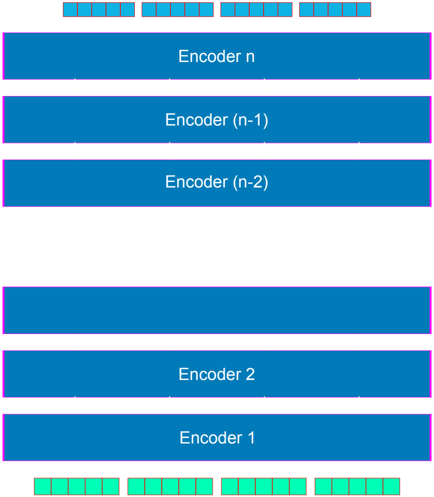


Reference: Vaswani et al [5]

TRANSFORMER ENCODER







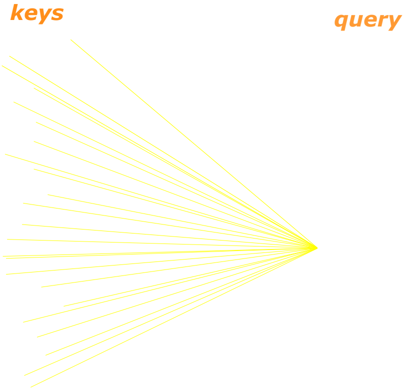
SELF-ATTENTION

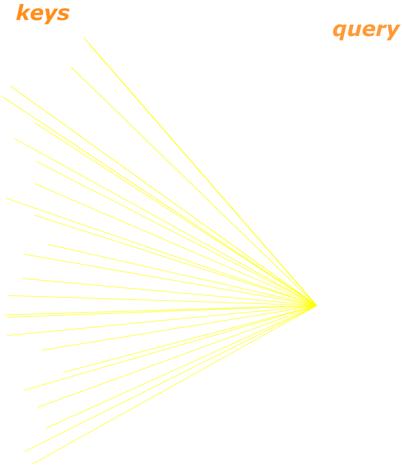
Peltophorum pterocarpum is a species of **Peltophorum**, native to tropical southeastern Asia and **it** is a popular ornamental tree grown around the world

What does it refer to?

Self-attention allows us to associate *it* with Peltophorum pterocarpum

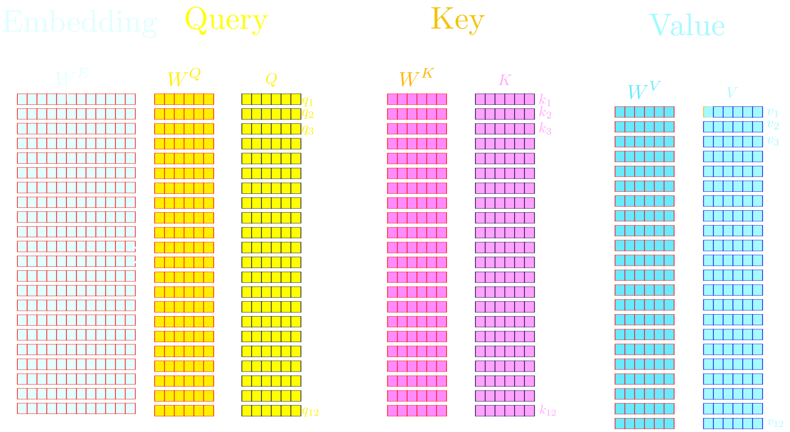
SELF-ATTENTION - WORD LEVEL



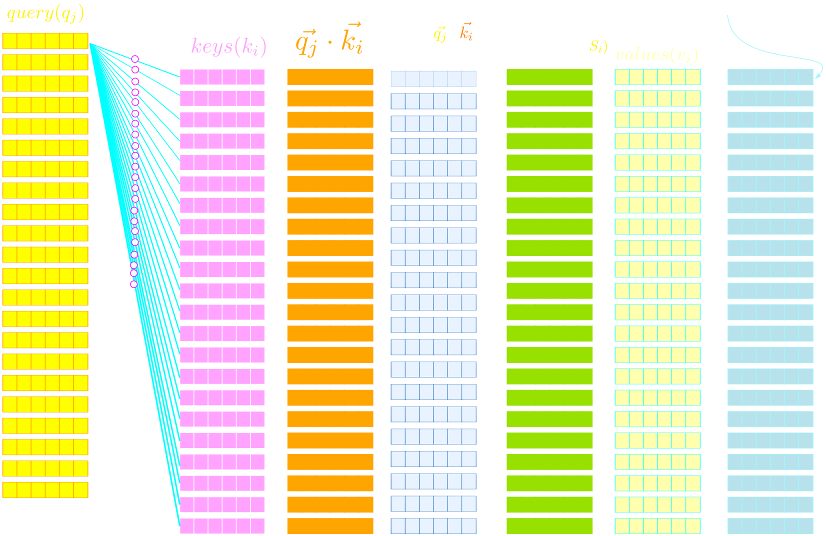


Self-attention allows each word to align itself to other words using their positions and looks for clues for a better contextual encoding

CREATING QUERY, KEY AND VALUES



COMPUTING ATTENTION(Q,K,V)

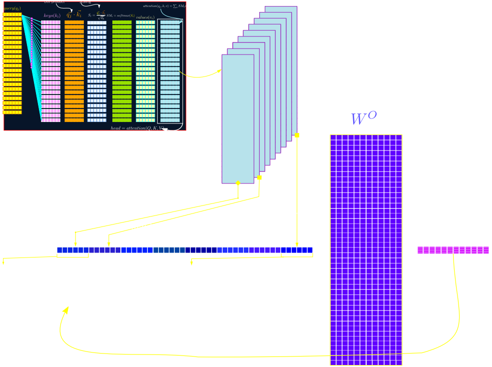


MULTI-HEAD ATTENTION

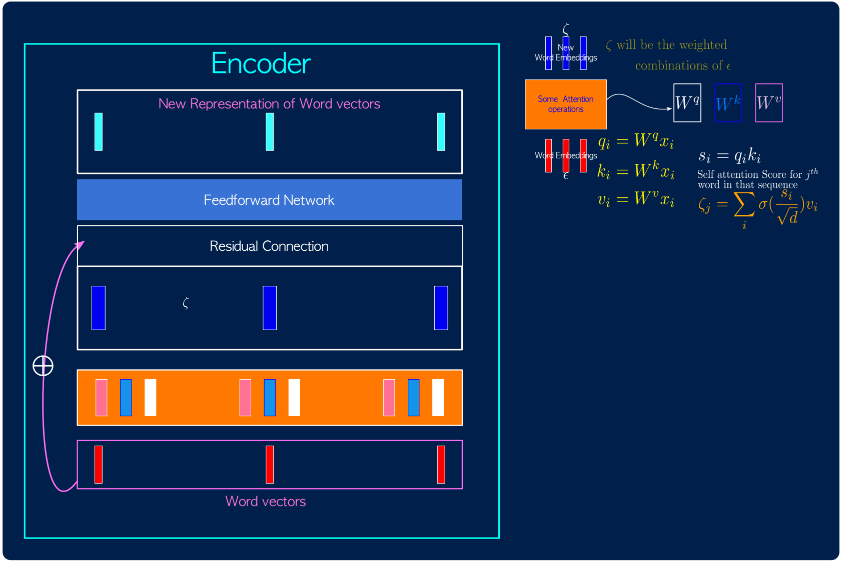
The summed attention score,

$$a(q, k_i, v_i) = \sum_i \left(\frac{e^{\frac{q \cdot k_i}{\sqrt{d_k}}}}{\sum_j e^{\frac{q \cdot k_j}{\sqrt{d_k}}}} \right) v_i$$

has information about all the word, but may have more information about words that have similarity scores higher than others in that context.



SELF-ATTENTION



ATTENTION COMPUTATION: A SIMPLIFIED EXAMPLE I

Let's compute attention using a simplified example with word embeddings. We'll use a small set of embeddings for demonstration purposes.
Assume we have the following word embeddings for simplicity:

- **Query (q)**: Embedding for "who"
- **Keys (k)**: Embeddings for "is", "the", "best"
- **Values (v)**: Same as Keys for simplicity, but in practice, they could be different.

SAMPLE SET OF WORDS

Let's say each embedding is a 3-dimensional vector:

$$
\begin{aligned}
&q : [0.1, 0.2, 0.3] \\
&k_1 \ (\text{is}) : [0.4, 0.5, 0.6] \\
&k_2 \ (\text{the}) : [0.2, 0.1, 0.3] \\
&k_3 \ (\text{best}) : [0.6, 0.7, 0.8] \\
&v_1 \ (\text{is}) : [0.4, 0.5, 0.6] \\
&v_2 \ (\text{the}) : [0.2, 0.1, 0.3] \\
&v_3 \ (\text{best}) : [0.6, 0.7, 0.8]
\end{aligned}
$$

Compute similarity scores - Dot Product

$$q \cdot k_1 = 0.1 \cdot 0.4 + 0.2 \cdot 0.5 + 0.3 \cdot 0.6 = 0.04 + 0.1 + 0.18 = 0.32$$
$$q \cdot k_2 = 0.1 \cdot 0.2 + 0.2 \cdot 0.1 + 0.3 \cdot 0.3 = 0.02 + 0.02 + 0.09 = 0.13$$
$$q \cdot k_3 = 0.1 \cdot 0.6 + 0.2 \cdot 0.7 + 0.3 \cdot 0.8 = 0.06 + 0.14 + 0.24 = 0.44$$

Apply Softmax Normalization

- **Exponentiation** $e^{0.32/\sqrt{3}} \approx 1.20, \quad e^{0.13/\sqrt{3}} \approx 1.08, \quad e^{0.44/\sqrt{3}} \approx 1.29$
- **Sum of exponents:** $1.20 + 1.08 + 1.29 = 3.57$
- **Attention weights:** $\alpha_1 = \frac{1.20}{3.57} \approx 0.34, \quad \alpha_2 = \frac{1.08}{3.57} \approx 0.30, \quad \alpha_3 = \frac{1.29}{3.57} \approx 0.36$

STEP 2: WEIGHTED SUM OF VALUE VECTORS ζ

$$
\begin{aligned}
\text{Attention}(q, k, v) &= \alpha_1 v_1 + \alpha_2 v_2 + \alpha_3 v_3 \\
&= 0.34 \cdot [0.4, 0.5, 0.6] + 0.3 \cdot [0.2, 0.1, 0.3] + 0.36 \cdot [0.6, 0.7, 0.8] \\
&= [0.41, 0.45, 0.58]
\end{aligned}
$$

The attention output for the query "who" given the keys and values "is", "the", "best" is approximately:

$$\zeta = [0.41, 0.45, 0.58]$$

This vector represents a weighted combination of the input values, where the weights are determined by how relevant each key is to the query, as computed through the attention mechanism.

Note: This is a highly simplified example. In real-world scenarios, embeddings are much higher dimensional, and the attention mechanism is applied across all positions in the sequence, often in parallel for efficiency.

PYTHON CODE TO COMPUTE ζ

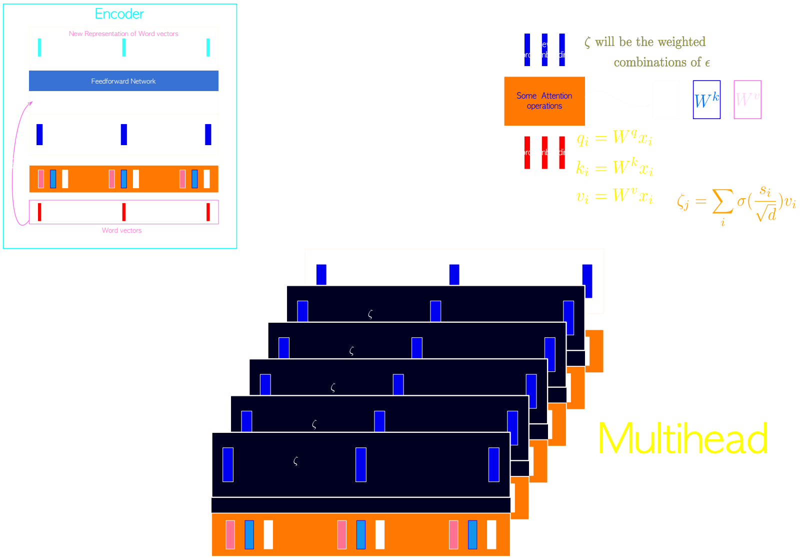
```
import numpy as np

def compute_attention(q, keys, values):
d = q.shape[0]
# Compute scaled dot products
dot_products = np.dot(keys, q) / np.sqrt(d)

exp_scores = np.exp(dot_products - np.max(dot_products))
attention_weights = exp_scores / np.sum(exp_scores)

zeta = np.sum(attention_weights[:, None] * values, axis=0)
return attention_weights, zeta
```

MULTIHEAD ATTENTION



POSITIONAL ENCODING

- RNN models encode the time signal in a sequential manner
- Positional encoding is important for contextual learning
- Transformers use a separate positional vector which is added to the embedding
- Positional encoding should be unique, not just a scalar
- Positional values should be bounded
- Positional values between any two positions should be consistent
- Sentence length should not be a constraint
- Deterministic

POSITIONAL ENCODING

$$\vec{pe}_t^{\,i} = \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k+1 \end{cases}$$

where $\omega_k = \dfrac{1}{10000^{(2k/d)}}$.

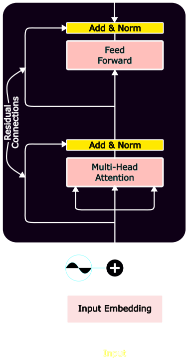
This positional vector will be a pair of $\sin$ and $\cos$ values and i is the index of the dimension in the word vector

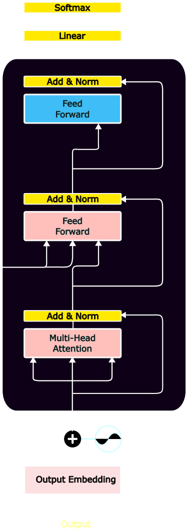
1. Good reference for Transformer - The Illustrated Transformer, Jay Alammar
2. The Annotated Transformer - Austin Huang et al

POSITIONAL ENCODING - PYTHON CODE

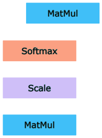
```python
def positional_encoding(position, d_model):
    angle_rates = 1 / np.power(10000, \
    (2 * (np.arange(d_model) // 2)) / d_model)
    position = position.reshape(-1, 1)
    angle_rates = angle_rates.reshape(1, -1)
    angle_rates = position * angle_rates
    angle_rates[:, 0::2] = np.sin(angle_rates[:, 0::2])
    angle_rates[:, 1::2] = np.cos(angle_rates[:, 1::2])
    return angle_rates
```

TRANSFORMER ARCHITECTURE

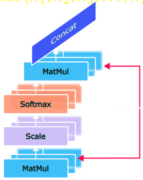




$$Attention = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



$$MultiHead(Q, K, V) =$$
$$Concat\,(h_1, h_2, h_3, \ldots, h_N)\,w^O$$



NUMBER OF PARAMETERS IN A TRANSFORMER

Embedding Layer - $N_{emb} = V \times D$
where N_{emb} is the number of parameters in the embedding layer, V is the vocabulary size and D is the embedding dimension

Multi-Head Attention - $N_{attn} = 3 \times D \times d_k \times h + h \times d_v \times d_k + 2 \times h \times d_k$ (biases)
where N_{attn} is the number of parameters in multi-head attention, D is the embedding dimension, d_k is the dimension of the key vector, h is the number of heads and d_v is the dimension of the value vector

Feed-Forward Network - $N_{ffn} = 2 \times D \times H + H + H \times D$
where N_{ffn} is the number of parameters in the feed-forward network, D is the embedding dimension and H is the hidden dimension

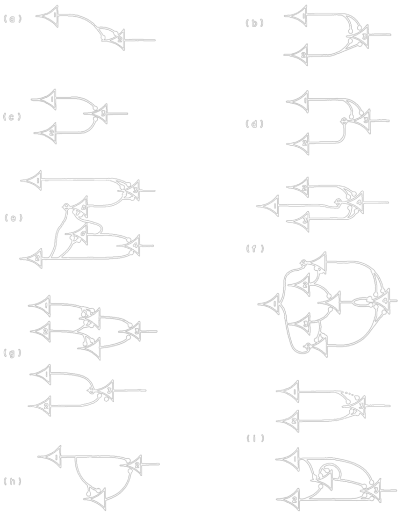
Layer Norm - $N_{ln} = 2 \times D$
where N_{ln} is the number of parameters in the layer normalization, D is the embedding dimension

Total Number of Parameters per Encoder/Decoder Layer - $N_{layer} = N_{attn} + N_{ffn} + N_{ln}$
Total Number of Parameters in the Transformer - $N_{total} = N_{emb} + N_{layer} \times L$
where L is the number of layers

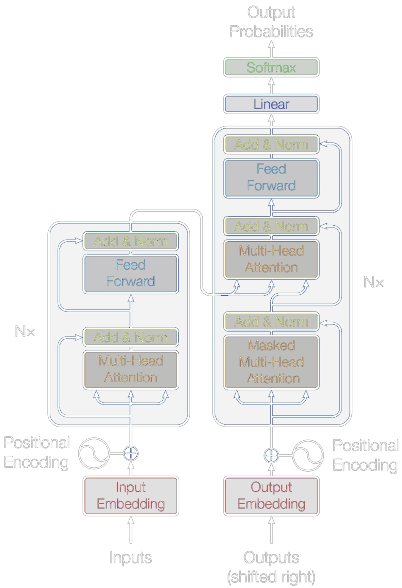
CHARACTERISTICS OF DIFFERENT LANGUAGE MODELS

Dependency Type	Dependency Structure	Models	
Independent Prediction	Does not depend on any token	Unigram	$P(X) = \prod_{i=1}^{\lvert V \rvert} P(x_i)$
Markov (Left2Right)	Depends on a fixed number of previous tokens	N-gram	$P(X) = \prod_{i=1}^{\lvert V \rvert} P(x_i \mid x_{i-n+1}, \ldots, x_{i-1})$
Unidirectional autoregressive	Depends on the previous tokens	RNN,GPT	$P(X) = \prod_{i=1}^{\lvert X \rvert} P(x_i \mid x_1, \ldots, x_{i-1})$
Bidirectional context	Left2Right /right2left Autoregressive	BERT	$P(X) = \prod_{i=1}^{\lvert X \rvert} P(x_i \mid x_{\neq i})$

MCCULLOCH-PITTS VS TRANSFORMER



1943 -> 2017



QUESTION: WHAT DO TRANSFORMER HEADS SPECIALIZE AT?

- All attention heads receive the same input embeddings.
- Heads start with random weights.
- How do they specialize in attending to different aspects of input?

RANDOM INITIALIZATION OF WEIGHTS

- Each head has unique weight matrices (W_{Q}, W_{K}, W_{V}, W_{O}).
- Random initialization leads to diverse transformations of input.
- Ensures heads start with distinct attention patterns.

SPECIALIZATION THROUGH TRAINING

- Heads specialize via backpropagation during training.
- Loss function guides heads to focus on:
  - Syntactic relationships (e.g., dependencies between words).
  - Semantic relationships (e.g., context or meaning).
  - Positional or structural patterns.

MULTI-HEAD ATTENTION MECHANISM

- Each head computes its own attention scores and representation.
- Representations are concatenated and linearly transformed.
- Enables model to aggregate diverse perspectives, such as:
  - Word-level dependencies.
  - Sentence-level context.
  - Positional relevance.

LAYER STACKING

- Outputs from one layer (all heads) serve as inputs to the next.
- Subsequent layers refine or build on earlier attention patterns.
- Hierarchical structure deepens specialization.

IMPLICIT COMPETITION

- Heads compete to minimize loss:
  - Redundant attention patterns provide diminishing returns.
  - Specialization maximizes each head's contribution.
- Encourages efficient use of capacity.

EXPERIMENTAL FINDINGS

- Certain heads consistently specialize across tasks and datasets:
  - Positional encoding.
  - Syntax parsing.
  - Semantic roles.
- Some heads act as:
  - **Generalists:** Broad attention across sequences.
  - **Specialists:** Focused attention on specific patterns.

SUMMARY

- Specialization arises from:
  - Random initialization of weights.
  - Differentiated learning through backpropagation.
  - Layer interactions and multi-head architecture.
- Leads to a division of labor among attention heads.
- Key to the power and flexibility of transformer models.

PURPOSE OF RESIDUAL CONNECTIONS

- **Mitigate Vanishing Gradients:**
  - Ensure gradients flow through deeper networks.
- **Enable Deeper Architectures:**
  - Support the use of multiple layers without degradation.
- **Easier Optimization:**
  - Simplify the training of complex models.

MECHANISM OF RESIDUAL CONNECTIONS

- **Post-Attention:**

$$\text{Output} = \text{LayerNorm}(\text{Input} + \text{SelfAttention}(\text{Input}))$$

- **Post-Feed-Forward:**

$$\text{Output} = \text{LayerNorm}(\text{PreviousOutput} + \text{FeedForward}(\text{PreviousOutput}))$$

BENEFITS OF RESIDUAL CONNECTIONS

- **Stability in Training:**
  - Prevents exploding or vanishing gradients in deep models.
- **Feature Preservation and Enhancement:**
  - Maintains input features while enhancing them through transformation.
- **Increased Model Expressiveness:**
  - Allows the model to learn complex representations.

WORD EMBEDDING

1. Word embedding is fundamental to Deep Learning of NLP
2. Contextual word embedding is now used in most of the downstream applications

CLOZE TEST

According to a report in yesterday's newspaper (1)——— police dog was taken to Raj Bhavan (2)——— Monday. This was to trace the (3)——— of the "very important horse" which (4)——— reported missing on Sunday. The dog picked (5)——— the scent on some traces of (6)——— and ran a few yards before losing the (7)———. The police have launched a vigorous (8)——— into the whole affair. They have (9)——— the services of a forensic expert, (10)——— fingerprint expert and a photographer. (11)——— are now fourteen horses at Raj Bhavan (12)——— are kept in a large shed near the gate.

1	once	a	new
2	at	next	on
3	police	killers	dogs
...	...	...	...
11	There	We	So
12	who	were	which

The purpose of the Cloze test is to measure the reading comprehension of a student with respect to grammar, usage, vocabulary, and contextual understanding.

MASKED LM[6]

Objectives

- Predict the masked word using the context
- Combine left and right contexts to predict the masked word
- Use sentence pairs to predict the next sentence
- Use the trained model for token-level and sentence-level tasks
- Mask some of the tokens from the input
- Predict the original token using its context
- Use Bidirectional deep transformer model fuse the left and right context and develop a contextual representation

HOW DO WE MASK?

- 15% of the words are randomly masked
- Perform the following operation on the the i^{th} token
    1. Replace with a $<$MASK$>$ 80% of the time
    2. Replace with a random token 10% of the time
    3. Retain the original 10% of the time

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT) - ARCHITECTURE

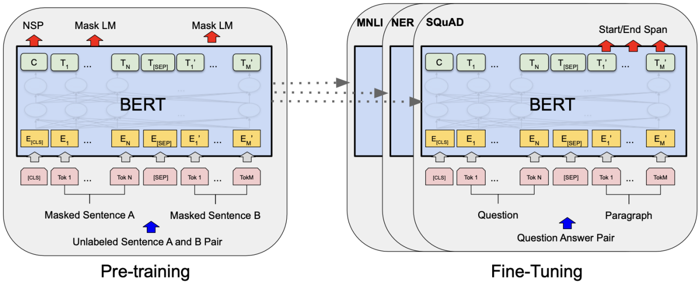


Figure: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architec- tures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).Next Sentence Prediction (NSP) And Perturbation is a method used to assess or enhance the performance of BERT-like models by altering or analyzing their training dynamics, specifically in tasks involving next sentence prediction.

DOES BERT PREDICT THE "NEXT WORD"?

- **Unlike GPT-style models, BERT does not predict the next word in the sequence**
- Instead, BERT predicts the **original word at the masked position**, leveraging:
  - **Bidirectional context** from both preceding and following tokens.
- **Example:**
  - Input: `The [MASK] sat on the mat.`
  - BERT considers both sides of `[MASK]`:
    - Context: "The" and "sat on the mat."
    - Prediction: "cat."
- **Why is this effective?**
  - Bidirectional prediction enables **contextual understanding**.
  - Excels in tasks like **question answering** and **natural language inference**.

GENERATIVE PRE-TRAINED TRANSFORMER - GPT

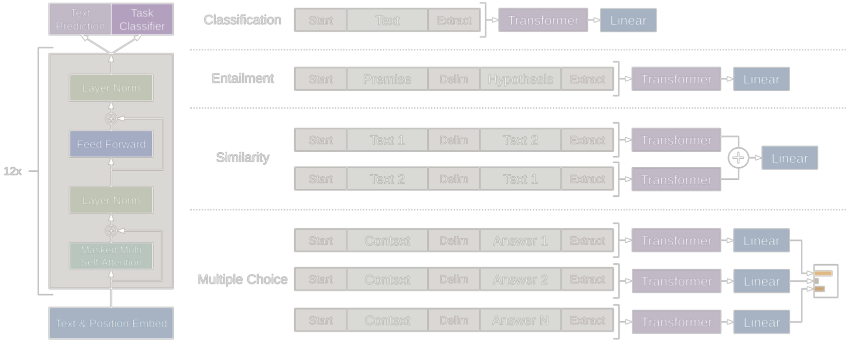
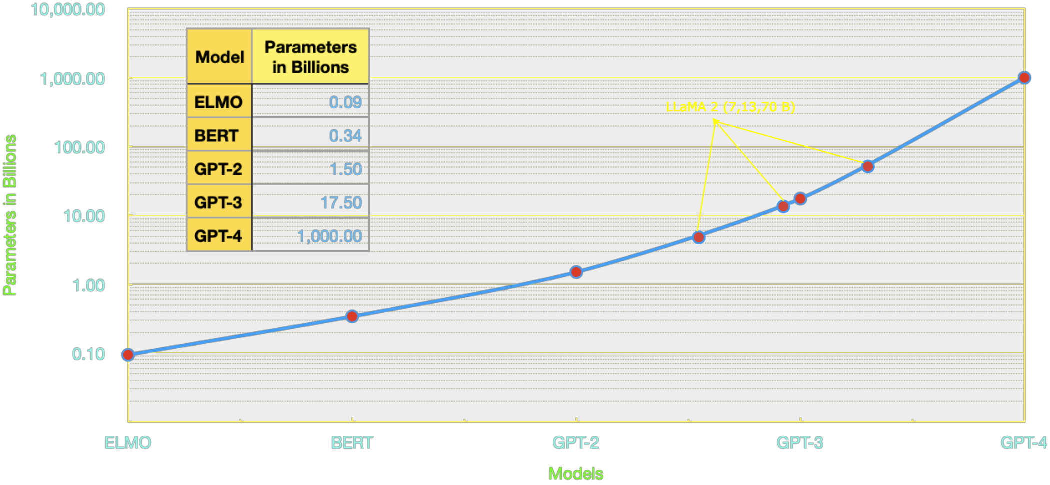


Figure: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer[7]

PARAMETERS OF LARGE LANGUAGE MODEL



LLAMA 2 OVERVIEW

- Open-source foundation model: Llama 2[8]
- Fine-tuned chat models for conversational AI
- Improves upon Llama 1 with new architectures and training methods
- Released by Meta AI
- Goals: Advance conversational AI, promote open research

KEY CONTRIBUTIONS

- Architecture: decoder-only Transformer with pre-normalization (RMSNorm), SwiGLU activations, and rotary positional embeddings (RoPE)
- Enhanced training: supervised fine-tuning followed by RLHF
- Large-scale pre-training: 2 trillion tokens; models of 7B, 13B, and 70B parameters; 4K-token context
- Grouped-query attention (in the 70B model) for faster inference
- Open-weights release: model weights and usage license (later Llama 3 models scaled to 15T training tokens and longer contexts)

TRAINING OF LLAMA 2-CHAT

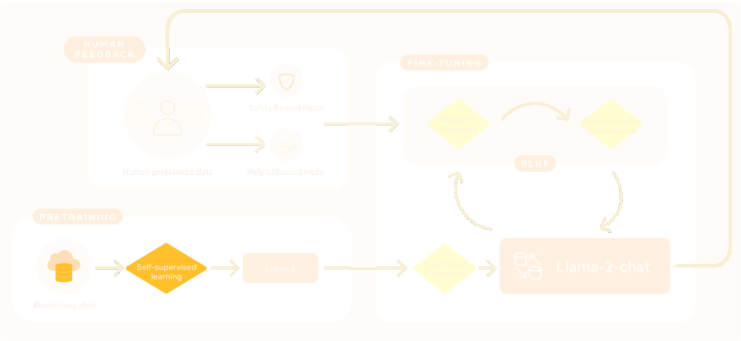


Figure: Thisprocessbeginswiththepretraining of Llama 2 usingpublicly availableonlinesources. Followingthis,wecreateaninitialversionof Llama 2-Chat throughtheapplication of supervised fine-tuning. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback (RLHF) methodologies, specifically through rejection sampling and Proximal Policy Optimization(PPO).ThroughouttheRLHFstage,theaccumulationof iterativerewardmodelingdata in parallel with model enhancements is crucial toensure thereward modelsremain within distribution[8].

IMPACT AND RESULTS

- State-of-the-art performance on conversational benchmarks
- Improved conversational flow and coherence
- Enhanced ability to understand context and nuance
- Demonstrates potential for real-world applications
- Opens doors for future research in conversational AI

SAMPLE TEXT GENERATION - MINIGPT

```
const LEARNING_RATE: f64 = 0.0003;
const BLOCK_SIZE: i64 = 128;
const BATCH_SIZE: i64 = 64;
const EPOCHS: i64 = 100;
const SAMPLING_LEN: i64 = 4096;
```

```
Config {
       vocab_size: 130,
       n_embd: 128,
       n_head: 8,
       n_layer: 8,
       block_size: 128,
       attn_pdrop: 0.1,
       resid_pdrop: 0.1,
       embd_pdrop: 0.1,
       };
```

Generated Text

```
for the gbr genome determines a comprehensive comparison from existing quality was
commonly important to a lung w fusion if they can be a compared x-rayed, with the
blood scenario of the lesion of t-cell support probability.in this later, the next
for the first room of responses to materials and infection has classified a
retrospective response of younger genes [138] .
```

LLAMA 2 OVERVIEW

- Open-source foundation model: Llama 2
- Fine-tuned chat models for conversational AI
- Improves upon Llama 1 with new architectures and training methods
- Released by Meta AI
- Goals: Advance conversational AI, promote open research

KEY CONTRIBUTIONS

- Architecture: decoder-only Transformer with RMSNorm pre-normalization, SwiGLU activations, and rotary positional embeddings (RoPE)
- Enhanced training: supervised fine-tuning followed by RLHF
- Large-scale pre-training: 2 trillion tokens; models of 7B, 13B, and 70B parameters; 4K-token context
- Grouped-query attention (in the 70B model) for faster inference
- Open-weights release: model weights and usage license

TRAINING OF LLAMA 2-CHAT

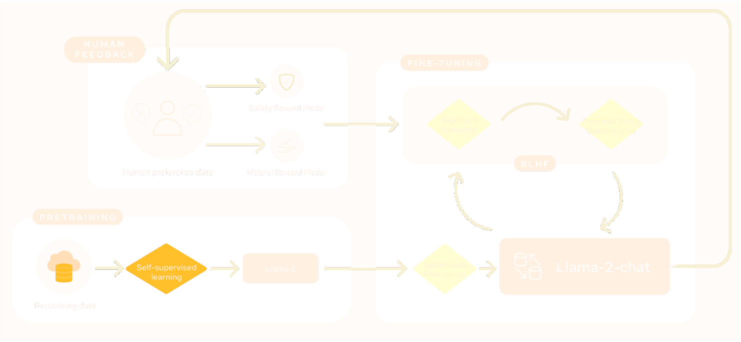


Figure: This process begins with the pretraining of Llama 2 using publicly available online sources. Following this,we create an initial version of Llama 2-Chat through the application of supervised fine-tuning. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback (RLHF) methodologies, specifically through rejection sampling and Proximal Policy Optimization(PPO).Throughout the RLHF stage,the accumulation of iterative reward modeling data in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

[8]

IMPACT AND RESULTS

- State-of-the-art performance on conversational benchmarks
- Improved conversational flow and coherence
- Enhanced ability to understand context and nuance
- Demonstrates potential for real-world applications
- Opens doors for future research in conversational AI

LLAMA 2

Llama-2 [8] was pretrained on publicly available online data sources. The fine-tuned model, Llama-2-chat, leverages publicly available instruction datasets and over 1 million human annotations.

- LLaMA stands for Large Language Model Meta AI.
- Llama 2 pretrained models are trained on 2 trillion tokens

MODEL DETAILS[9]

	Training Data	Parameters	Content Length	GQA	Tokens	Learning Rate
Llama 2	Publicly available online data	7B	4k	NA	2.0T	3.0×10^4
Llama 2	"	13B	4k	NA	2.0T	3.0×10^4
Llama 2	"	70B	4k	YES	2.0T	1.5×10^4

CHALLENGES

- Computationally expensive to train and deploy
  - Facebook's 65B LLaMA model - Trained for 21 days on 2048 Nvidia A100 GPUs, at a cost of ≈ 4 million USD
  - Google's 540B PaLM was trained on 6144 v4 TPUs for 1200hrs, at a cost of $\approx \$27$ million USD
  - Cost of training ChatGPT 4 is estimated to be over $100 million USD
    - Costs include data preparation (usually around $10M), cost of computing power and cost of engineering staff & administration expenses)
    - Power used to train ChatGPT 4 is $\approxeq 1.287$ gWH. Average Indian house-hold consumtion per year is 1200 kWH.
- Can be biased, reflecting the biases of the data they are trained on
- May generate harmful or offensive content.

FAIRNESS AND INCLUSION

Everyone should be treated fairly when using our products and they should work equally well for all people.

ROBUSTNESS AND SAFETY

- Should meet high performance standards
- Should be robust and safe, with rigorous testing and validation to avoid unintended negative consequences

TRANSPARENCY AND ACCOUNTABILITY

- Developers and users of the system should be responsible for its actions
- Rules and regulations governing the use of AI systems should be transparent,
- Mechanisms should be in place to investigate and punish misuse of the system

EXPLAINABILITY

- Should enable users to understand the underlying processes to build trust
- System should explain its decisions in a clear and understandable way
- Open for scrutiny

Thank you

Linkedin:Ramaseshan
Email:Ramaseshan Ramachandran

REFERENCES I

[1] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[2] Douglas LT Rohde, Laura M Gonnerman, and David C Plaut. "An improved model of semantic similarity based on lexical co-occurrence". In: *Communications of the ACM* 8.627-633 (2006), p. 116.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". English (US). In: *arXiv* (2014).

REFERENCES II

[5] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[6] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[7] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[8] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[9] Joshua Ainslie et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. arXiv: 2305.13245 [cs.CL]. URL: https://arxiv.org/abs/2305.13245.