

Probabilistic generative models

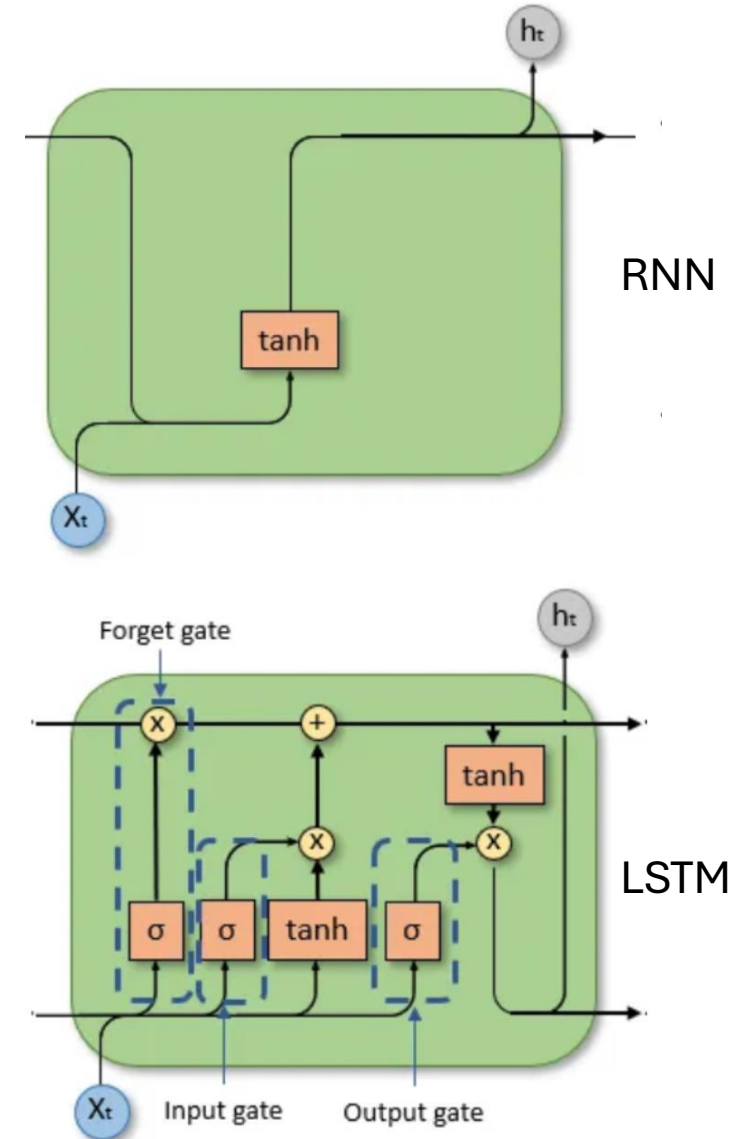
Text generation

Introduction to Transformers and Pre-training

<https://tinyurl.com/2wamt7p7>

Introduction to Transformers - Motivation and History

- **Problems with RNNs and LSTMs:**
 - **RNNs (Recurrent Neural Networks)** and **LSTMs (Long Short-Term Memory Networks)** were designed to process sequential data but come with significant limitations.
 - **Long-Range Dependency:**
 - RNNs and LSTMs struggle to capture long-range dependencies in sequences due to gradient degradation.
 - **Parallelization:**
 - RNNs and LSTMs process data sequentially, making parallelization difficult and prolonging training times.



Introduction to Transformers - Motivation and History

- Publication of "**Attention is All You Need**" (Vaswani et al., 2017):
 - This paper introduced the Transformer architecture, eliminating recursion by relying solely on the attention mechanism.
 - The phrase "**Attention is All You Need**" summarizes the idea that the attention mechanism alone, without RNNs or CNNs, is sufficient to capture sequential dependencies in data.

The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

ASHISH, Vaswani. Attention is all you need. *Advances in neural information processing systems*, 2017, vol. 30, p. 1.

Introduction to Transformers - Motivation and History

- **Revolution in Natural Language Processing (NLP) and Beyond:**

- Transformers have revolutionized **Natural Language Processing (NLP)** by setting new standards for accuracy and efficiency across various tasks (e.g., machine translation, text summarization, text generation).

- **Expanded Applications:**

- **Computer Vision:** Use of Transformers for image processing and generation (e.g., Vision Transformers or ViTs).
- **Bioinformatics:** Application in predicting protein structures, analyzing DNA sequences, etc.

Introduction to Transformers - Overall Architecture of a Transformer

Modular Structure:

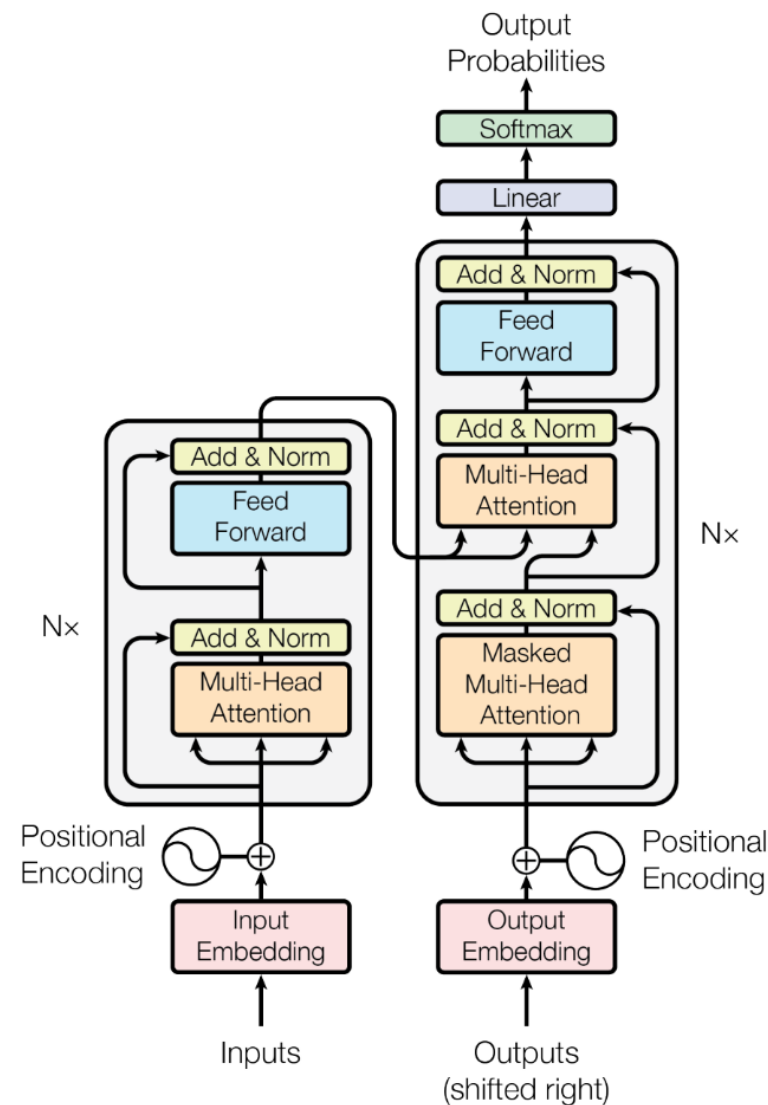
- A Transformer consists of two main blocks: the **encoder** and the **decoder**.

Encoder:

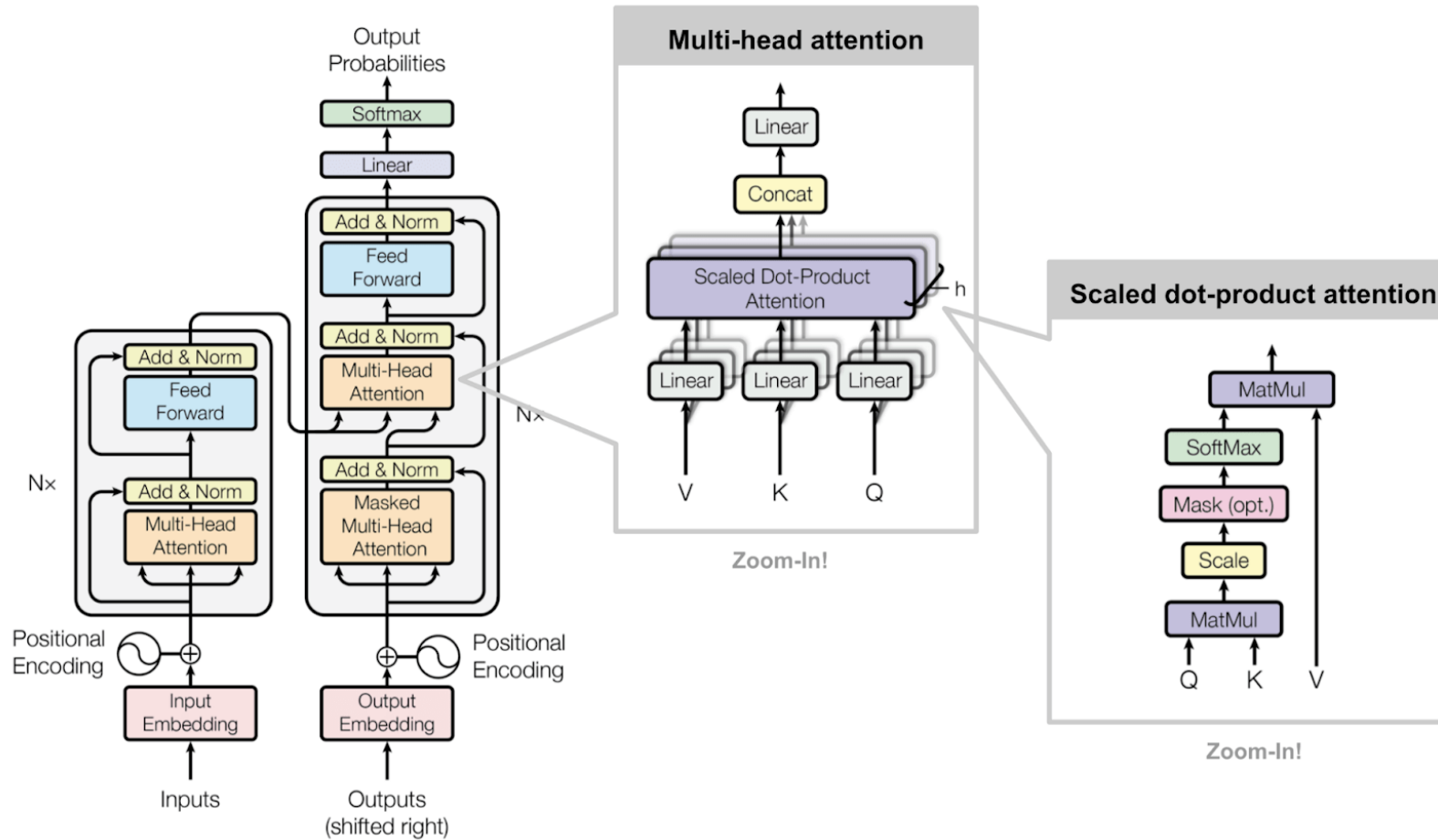
- Composed of identical layers, each containing two main sub-layers.
- Each layer contains two main sub-layers:
 - Multi-Head Self-Attention Mechanism:** Allows the model to focus its attention on different parts of the input.
 - Fully Connected Feed-Forward Network:** Two layers of feed-forward networks applied independently at each sequence position.
- Each sub-layer is followed by batch normalization and a skip connection

Decoder:

- Similar to the encoder but with an additional sub-layer for attention between the encoder output and the decoder input.
- Like the encoder, but includes an additional attention sub-layer.
- This sub-layer applies attention to the encoder outputs.
- The decoder thus has three sub-layers:
 - Self-Attention:** Applied to the decoder's own inputs.
 - Encoder-Decoder Attention:** Applied between the encoder's outputs and the decoder's inputs.
 - Fully Connected Feed-Forward Network.**



Introduction to Transformers - Overall Architecture of a Transformer



ASHISH, Vaswani. Attention is all you need. *Advances in neural information processing systems*, 2017, vol. 30, p. 1.

Introduction to Transformers - Attention Mechanism

Transformers revolutionize the way sequences are modeled by using attention mechanisms instead of traditional sequential models. Understanding their motivation, overall architecture, and internal attention mechanisms is crucial to fully leverage their power in NLP applications and beyond.

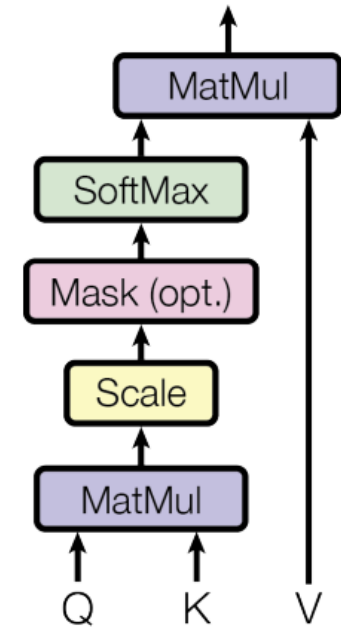
- Scaled Dot-Product Attention
- Similarity Calculation:
 - The attention mechanism calculates the similarity between elements in a sequence using **query** Q , **key** K , and **value** V matrices.

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

where d_k is the dimension of the key vectors.

- Self-Attention:
 - **Application to Same Input Data:** In self-attention, the query, key, and value matrices are all linear projections of the same input data.
 - **Allows each sequence position to attend to every other position.**

Scaled Dot-Product Attention



Introduction to Transformers - Attention Mechanism

- **Multi-Head Attention:**

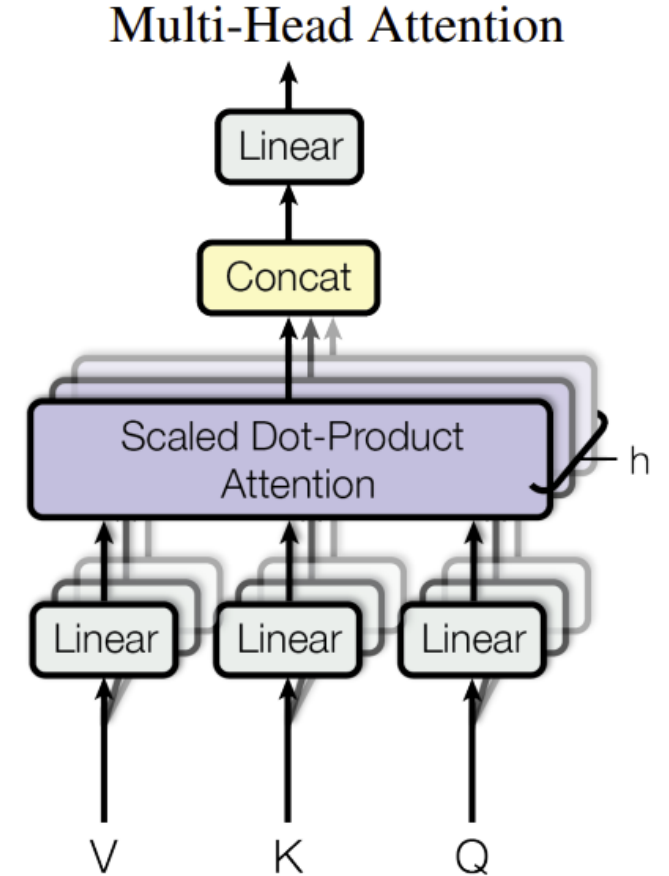
- **Parallel Processing of Different (Q, K, V) Pairs:** Divide the Q, K, and V matrices into multiple subspaces, apply the self-attention mechanism to each subspace, and recombine the results.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where each head is calculated as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

W_i^Q, W_i^K and W_i^V are the attention weights for the i -th head. W^O is the weight matrix for recombining the results of the different heads.



Introduction to Transformers - Attention Mechanism

Simplified Code Example for Attention Mechanism in PyTorch:

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class ScaledDotProductAttention(nn.Module):
    def __init__(self):
        super(ScaledDotProductAttention, self).__init__()
    def forward(self, Q, K, V, mask=None):
        d_k = Q.size(-1)
        scores = torch.matmul(Q, K.transpose(-2, -1)) / torch.sqrt(torch.tensor(d_k, dtype=torch.float32))
        if mask is not None:
            scores = scores.masked_fill(mask == 0, -1e9)
        attention = F.softmax(scores, dim=-1)
        output = torch.matmul(attention, V)
        return output, attention
```

```
class MultiHeadAttention(nn.Module):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.d_model = d_model
        self.num_heads = num_heads
        assert d_model % num_heads == 0

        self.depth = d_model // num_heads
        self.WQ = nn.Linear(d_model, d_model)
        self.WK = nn.Linear(d_model, d_model)
        self.WV = nn.Linear(d_model, d_model)
        self.fc = nn.Linear(d_model, d_model)

    def split_heads(self, x, batch_size):
        x = x.view(batch_size, -1, self.num_heads, self.depth)
        return x.transpose(1, 2)

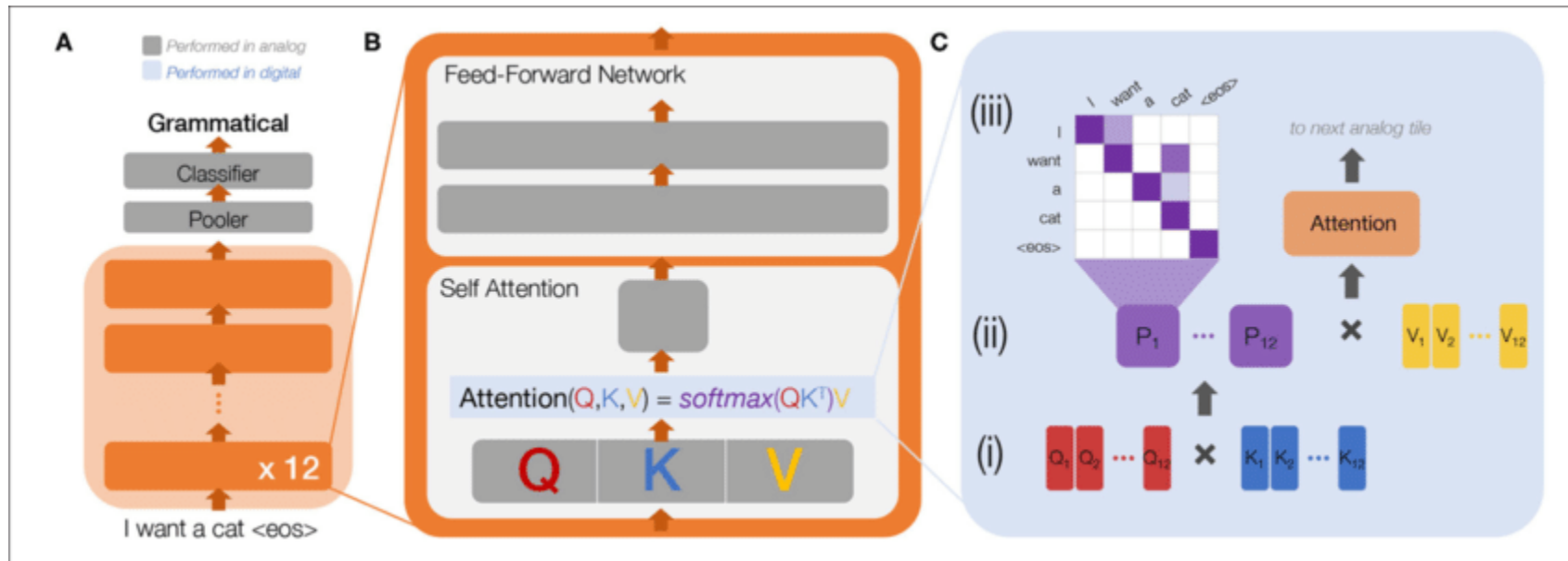
    def forward(self, Q, K, V, mask=None):
        batch_size = Q.size(0)
        Q = self.split_heads(self.WQ(Q), batch_size)
        K = self.split_heads(self.WK(K), batch_size)
        V = self.split_heads(self.WV(V), batch_size)

        scaled_attention, _ = ScaledDotProductAttention()(Q, K, V, mask)
        scaled_attention = scaled_attention.transpose(1, 2).contiguous()
        concatenated_attentions = scaled_attention.view(batch_size, -1, self.d_model)
        output = self.fc(concatenated_attentions)
        return output
```

Transformers in NLP – BERT (Bidirectional Encoder Representations from Transformers)

"Bidirectional" Learning:

- BERT is designed to understand the context of words in a sentence bidirectionally, meaning it utilizes both the words before and after the target word.
- Bidirectionality allows for a better understanding of relationships in the text.

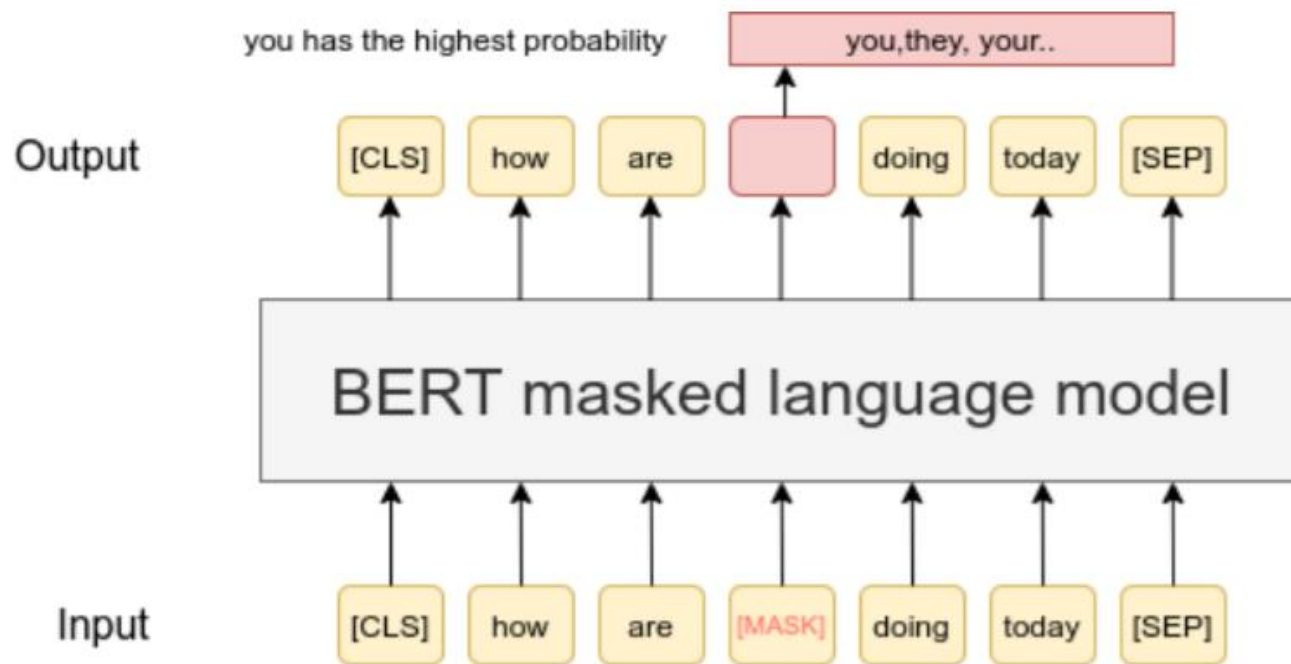


Transformers in NLP – BERT (Bidirectional Encoder Representations from Transformers)

Pre-training:

- **MLM (Masked Language Modeling) Tasks:**

- During pre-training, some words in the sentence are randomly masked, and the model must predict these masked words.
- **Objective:** To understand the context of each word in both directions.



$$L_{MLM}^{(x)} = -\frac{1}{|M_x|} \sum_{i \in M_x} \log P(x_i / x_{\setminus M_x})$$

where:

$x_{\setminus M_x}$ represents masked version of x

M_x represents set of masked token positions in x

Transformers in NLP – BERT (Bidirectional Encoder Representations from Transformers)

Pre-training:

- **NSP (Next Sentence Prediction) Tasks:**

- The model is trained to predict if a given sentence follows another sentence.
- **Objective:** To understand the relationship between sentences and improve performance on tasks requiring global text comprehension.

Sentence 1	Sentence 2	Next Sentence?
I have a class	I will be back by 6	✓
I have a class	Zebra is a animal	✗

$$L_{NSP}^{(x,y)} = -\log P(d/x, y)$$

where:

$d \in \{1, 0\}$ represents whether the sentences are consecutive or not

Transformers in NLP – BERT Variants

- **RoBERTa (Robustly optimized BERT approach):**
 - **Improvements Over BERT:**
 - RoBERTa is trained with more data and for a longer period, using optimized training practices.
 - Removes the NSP task and focuses more on MLM with a larger dataset.
 - Provides better robustness and accuracy compared to the original BERT model.
- **DistilBERT:**
 - **Lighter Version of BERT:**
 - DistilBERT is a lighter version of BERT that retains about 97% of its performance while being 60% smaller.
 - **Faster Inference Speed:** Ideal for applications requiring quick responses with lower resource usage.
 - Uses a knowledge distillation technique to transfer knowledge from a larger model (like BERT) to a smaller, more efficient model.

Transformers in NLP – Pre-training and Fine-tuning

Pre-training:

- **Learning on Large Unlabeled Corpora:**
 - Transformer models like BERT are initially pre-trained on vast unlabeled text corpora, such as Wikipedia and BookCorpus.
 - This phase allows the model to understand basic linguistic structures and contexts.
- **Generic Objective Tasks (e.g., Masked Word Prediction):**
 - Pre-training uses generic tasks like masked word prediction (MLM) to enhance language comprehension.
 - Trains the model to predict these words using the context provided by other words in the sentence.

Fine-tuning:

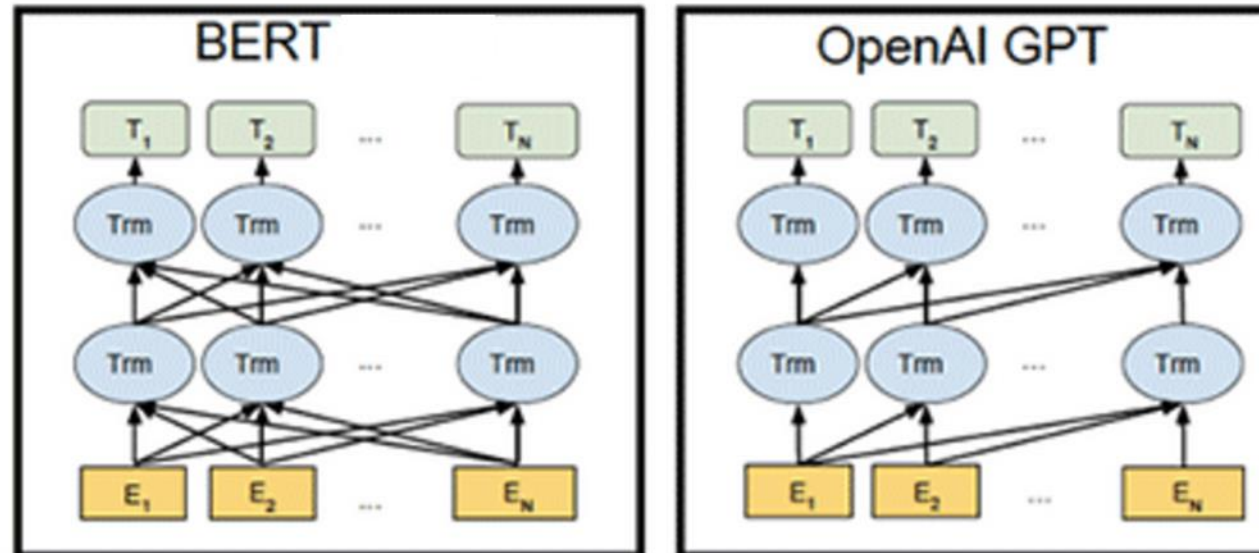
- **Adapting to Specific Tasks with Labeled Data:**
 - After pre-training, the model is fine-tuned on specific tasks using labeled datasets, such as text classification or named entity recognition (NER) datasets.
 - Fine-tuned models perform exceptionally well on specific tasks due to the deep language understanding acquired during pre-training.

GPT (Generative Pre-trained Transformer) – GPT and GPT-2

Unidirectional Architecture (Auto-regressive Transformer):

- **Concept:**

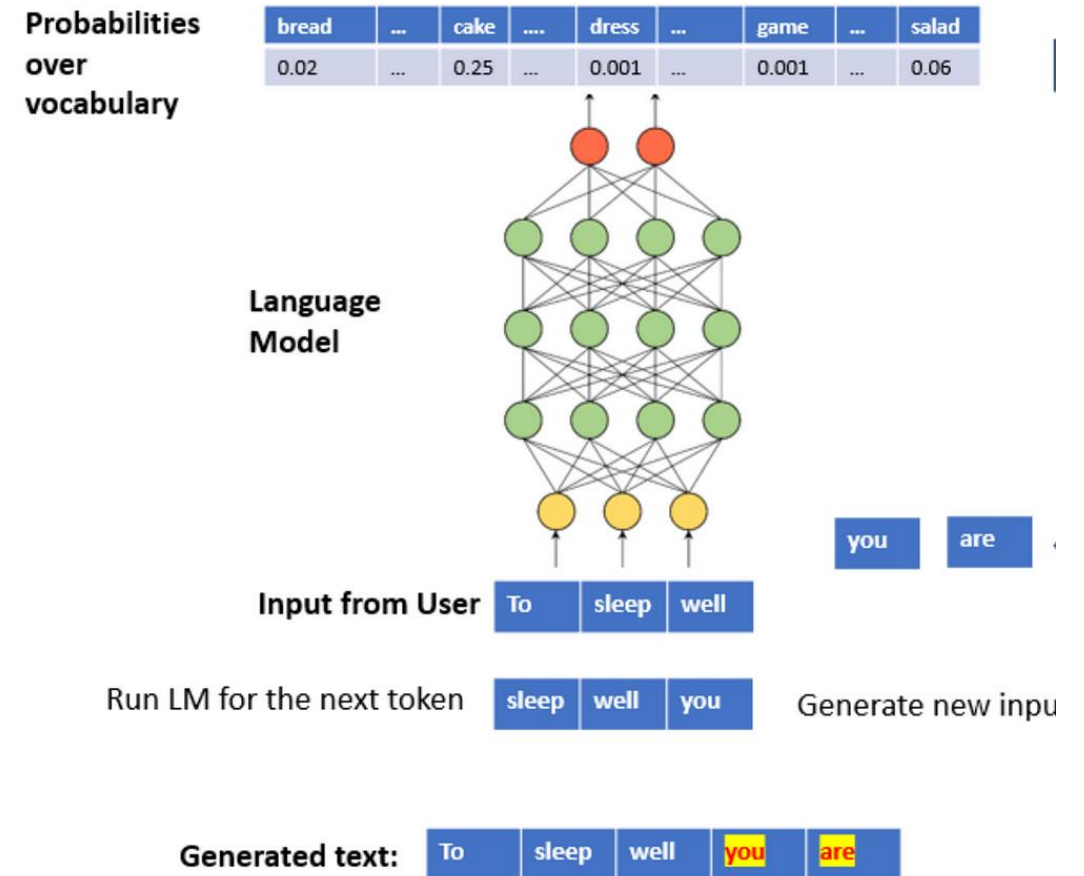
- Unlike BERT, GPT is an auto-regressive model, meaning it generates text by predicting the next word in a given sequence.
- Uses only the context of preceding words for prediction.



GPT (Generative Pre-trained Transformer) – GPT and GPT-2

Unidirectional Architecture (Auto-regressive Transformer):













- **Next Word Prediction in Sequence (Conditioned Language):**
 - Trained to predict the next word in large text sequences, enabling coherent and contextually appropriate text generation.
- **GPT learn to predict the future**



GPT (Generative Pre-trained Transformer) – GPT3

175 Billion Parameters Capacity:

- **Expansion in Size:**
 - GPT-3 is a massive extension of previous versions with 175 billion parameters, compared to 1.5 billion for GPT-2.
 - The large capacity allows it to capture finer linguistic nuances and better understand the overall context of a sequence.
- **Complex Applications with Little Additional Data (Zero-shot, One-shot Learning):**
 - Capable of performing language understanding tasks with little to no additional examples thanks to its extensive pre-training.
 - **Zero-shot Learning:** The ability to perform tasks without seeing specific examples of that task.
 - **One-shot Learning:** The ability to learn from a single provided example.
- **Application Examples:**
 - **Text Generation:** Creating articles, automatic writing.
 - **Answering Questions:** In dialogue systems.
 - **Text Translation:** Converting natural languages.
 - **Sentiment Analysis:** Evaluating sentiments in text.

 Chat Open ended conversation with an AI assistant.	 Q&A This prompt creates a question + answer structure for answering questions based on existing...
 Grammar correction This zero-shot prompt corrects sentences into standard English.	 Summarize for a 2nd grader This prompt translates difficult text into simpler concepts.
 Text to command This prompt translates text into programmatic commands.	 English to French This prompt translates English text into French.
 Parse unstructured data Create tables from long form text by specifying a structure and supplying some examples.	 Classification Classify items into categories via example.
 Movie to Emoji Convert movie titles into emoji.	 Advanced tweet classifier This is an advanced prompt for detecting sentiment. It allows you to provide it with a list of...
 Keywords Extract keywords from a block of text. At a lower temperature it picks keywords from the text. At a...	 Factual answering This prompt helps guide the model towards factual answering by showing it how to respond to...

GPT (Generative Pre-trained Transformer) – Conclusion

Transformers, leveraging architectures like BERT and GPT, are revolutionizing natural language processing. Their pre-training and fine-tuning techniques allow for significant advancements across various domains. Learning about their variants, mechanisms, and specific applications is essential to fully harness their power.



Applications and Case Studies – Machine Translation

Transformers for Translation:

- **Encoder-Decoder Model:**
 - **Principle:**
 - Transformer models use an encoder-decoder architecture for machine translation. The encoder converts the input sequence (source text) into a sequence of latent representations, which the decoder uses to generate the output sequence (target text).
 - **Advantages:**
 - Captures the global context of the sentence, producing more fluent and accurate translations compared to previous RNN-based architectures.

Example Models:

- **T5 (Text-to-Text Transfer Transformer):**
 - **Overview:**
 - T5 is a generalized transformer model that converts all NLP tasks into text-to-text conversion problems.
 - It is used for various NLP tasks, including machine translation, by reformulating the problem as a text transformation task.
 - **Functioning for a translation task:**
 - The input (source text) is encoded by the encoder.
 - The decoder generates the translated text word by word.

Introduction to Transformers - Attention Mechanism

Example of Using T5 for Translation with Hugging Face Transformers:

```
from transformers import T5Tokenizer, T5ForConditionalGeneration

# Load the pre-trained model
model_name = 't5-base'
tokenizer = T5Tokenizer.from_pretrained(model_name)
model = T5ForConditionalGeneration.from_pretrained(model_name)

# Source text to translate
text = "translate English to French: The quick brown fox jumps over the lazy dog"

# Tokenize the input
input_ids = tokenizer.encode(text, return_tensors='pt')

# Translation
output = model.generate(input_ids, max_length=40, num_beams=4, early_stopping=True)
translated_text = tokenizer.decode(output[0], skip_special_tokens=True)

print("Translation:", translated_text)
```

Translation: Le renard brun rapide saute au-dessus du chien lazy

T5

Le rapide renard brun saute par-dessus le chien paresseux.

GPT

Applications and Case Studies – Text Synthesis

Coherent Text Generation:

- **Functionality:**
 - Models like GPT-3 generate text by predicting the next word in a given sequence, based on an auto-regressive transformer architecture.
 - Capable of producing coherent and contextually relevant text from a short starting prompt.

Practical Applications:

- **Articles and Dialogues:**
 - Writing blog articles, creating marketing content, automatic writing.
 - Used in chatbots and virtual assistants to provide natural and implicit responses.

Examples of Applications with GPT-3:

```
# Initialize OpenAI client
client = OpenAI(api_key=api_key)
# Prompt for text generation
prompt = "translate English to French: The quick brown fox jumps over the lazy dog"

response = client.chat.completions.create(
    model="gpt-3.5-turbo", # ou "gpt-4" si vous avez accès à GPT-4
    messages=[
        {"role": "system", "content": "Vous êtes un assistant utile."},
        {"role": "user", "content": prompt}
    ],
    max_tokens=200
)

# Obtenir la réponse générée
generated_text = response.choices[0].message
print(generated_text)
```

Applications and Case Studies – Text Synthesis

Renewable energy sources are becoming increasingly essential in the fight against climate change and the transition towards a sustainable future. The importance of renewable energy lies in its numerous environmental, economic, and social benefits.

One of the key advantages of renewable energy is its low carbon footprint. Unlike fossil fuels, renewable sources such as solar, wind, and hydropower generate electricity without producing harmful greenhouse gas emissions that contribute to global warming. By reducing our reliance on fossil fuels and transitioning to renewable energy sources, we can mitigate the impacts of climate change and work towards a cleaner and healthier planet.

Renewable energy also offers economic benefits by creating new job opportunities and driving innovation in the energy sector. The growing market for renewable technologies has led to the creation of millions of jobs worldwide, from manufacturing and installation to research and development. Investing in renewable energy helps stimulate economic growth, reduce reliance on imported fuels, and increase energy security.

Furthermore, the widespread adoption of renewable energy can improve energy access and affordability, particularly in developing

Applications and Case Studies – Ethical and Social Issues

Bias and Ethics:

- Models trained on vast, unfiltered datasets can replicate and amplify biases present in those datasets.

Misuse:

- Potential use for generating misinformation, fake news, or misleading content.

Intellectual Property:

- Issues of copyright when models use protected works to generate new content.

Complex Text Synthesis with GPT-3 – Case Studies

- **GPT-3 for Automatic Writing:**
- **Capabilities:**
 - High-quality automatic text generation, including research articles, literature reviews, video scripts.
 - Context recognition to improve the coherence and relevance of the generated text.
- **Example Usage for Writing an Article:**

prompt = "Write a detailed introduction on the impact of artificial intelligence in healthcare. "

- **Usage in Content Creation:**
 - **Automating Repetitive Tasks:**
 - Generating product descriptions, newsletters, article summaries, or creating artistic content.
 - Increases efficiency by reducing the time required to create quality content.

Transformers, especially models like T5 and GPT-3, offer powerful solutions for various applications in natural language processing, ranging from machine translation to coherent and complex text generation. They also significantly impact content creation but require careful consideration of the ethical and social issues related to their use