Deep Learning with Keras and TensorFlow

Transformer Models for NLP

Learning Objectives

By the end of this lesson, you will be able to:

- Apply self-attention mechanisms to analyze the importance of different words in a sentence for text processing
- Utilize Transformer models to improve machine translation tasks, and explore how these models effectively manage long-range dependencies
- Employ the Transformer architecture in natural language processing to develop more efficient sentiment analysis tools
- Implement a Transformer model to recognize and translate languages, demonstrating the model's ability to handle different linguistic inputs

Business Scenario

TelNet Global, a multinational telecommunications company, handles millions of customer interactions monthly across various channels, including social media, email, and live chat. The company faces challenges with response time and support quality due to the high volume of queries and the complexity of customer issues, ranging from billing problems to technical support for services.

Challenge:

TelNet Global needs a solution that can:

- Automate responses to common queries to reduce the workload on human agents
- Understand and route complex customer inquiries to the appropriate departments
- Enhance customer experience by quickly providing accurate, context-aware responses
- Reduce operational costs associated with customer support

Overview of Transformer Models

Introduction to Transformer Models

Transformer models are a type of deep learning architecture that leverages self-attention mechanisms to enable simultaneous processing of all sequence elements.

RNNs process data sequentially, requiring each step in a sequence to be handled one after another, unlike Transformers, which process elements simultaneously.

Understanding Self-Attention

Self-attention, a key component in Natural Language Processing (NLP), enables the network to focus on specific words or phrases in a sentence to improve context understanding.

While reading a novel, you simultaneously focus on the current page and recall earlier events, characters, and clues.

Example:

You are reading a mystery novel.

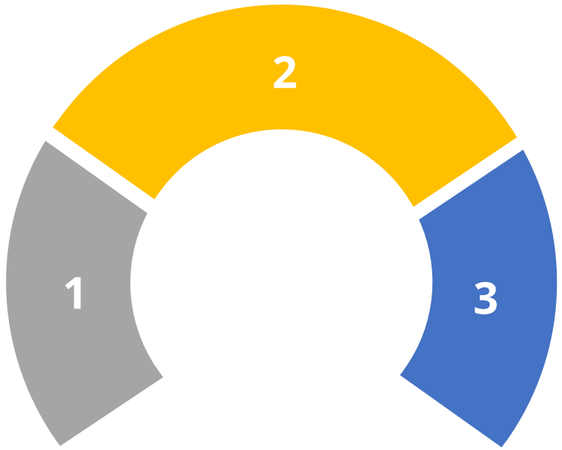
This context helps you understand the story better and predict future events.

Mechanics Behind Self-Attention

The self-attention layer calculates three vectors from each encoder's input vector:

Key Vector (K): This vector scores the attentiveness of each word.

Query Vector (Q): This vector scores each word regarding the extent of attention it needs.



Value Vector (V): This vector represents the actual word content, which generate the final output.

Mechanics Behind Self-Attention

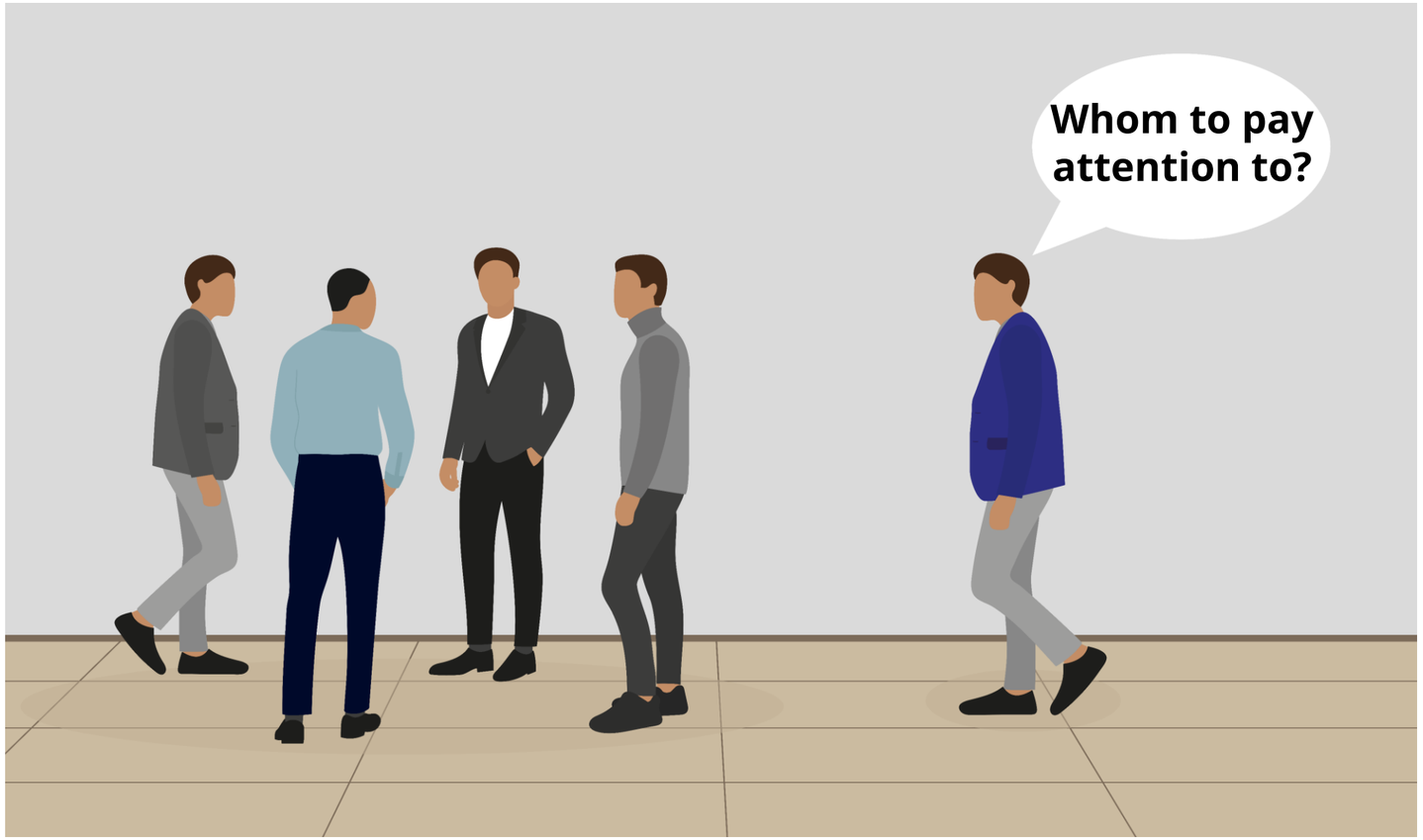
During training, vectors are iteratively trained and updated.

The following equation defines the attention score for each input word.

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

Understanding Mechanics Behind Self-Attention: Example

A person at a party tries to decide who to listen to in a crowded room. Imagine each person at the party has a story to tell, but not all stories are equally important to that person. So, everyone needs to decide how much attention each speaker should get.



Understanding Mechanics Behind Self-Attention: Example

Understand how self-attention works in this analogy:

Listening: Each person (data point or word in a sentence) listens to the stories (inputs) of others in the room (sequence).

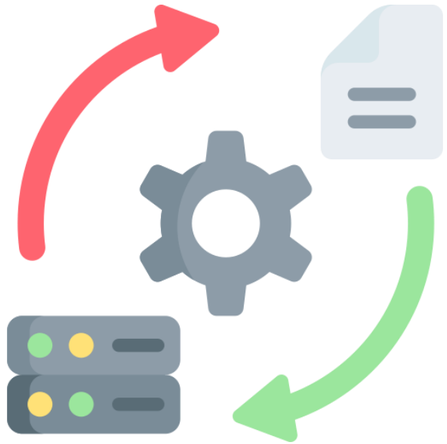
Scoring: Everyone assigns a score to each storyteller based on how relevant each story is to their current interests or the context of the conversation (query-key matching).

Focusing: The focus is more on the stories with higher scores (these get more attention).

Combining: Everyone creates a summary of what's important from all these stories, weighted by how much attention was paid to each (weighted sum of values).

Transformer Models: Advantage

Transformer models can excel at capturing long-range dependencies through self-attention mechanisms, making them ideal for tasks such as machine translation and document summarization.



Prior models, such as RNNs and LSTMs, had challenges dealing with long-range sentence dependencies, which were resolved by transformer models.

Long-Range Dependencies

Consider the following sentence:

"Frenny likes to play basketball; he is good at dunking."

Frenny is the subject.

he is a pronoun referring to Frenny.

RNNs and LSTMs are generally proficient at establishing such associations, especially when dealing with shorter sentences.

Long-Range Dependencies

Consider a bigger paragraph:

"Lara is a cook at McTown French Fries. She's been working there for three years now. The place immediately gained fame once she joined. Nobody knew of its existence until she joined it. She made good friends with other cooks and learned to cook various new dishes, which customers enjoyed at that place."

- Lara
- McTown French Fries
- Pronouns of Lara
- Pronouns of McTown French Fries

Longer sequences or paragraphs can be challenging for traditional sequence models like RNNs and LSTMs to process.

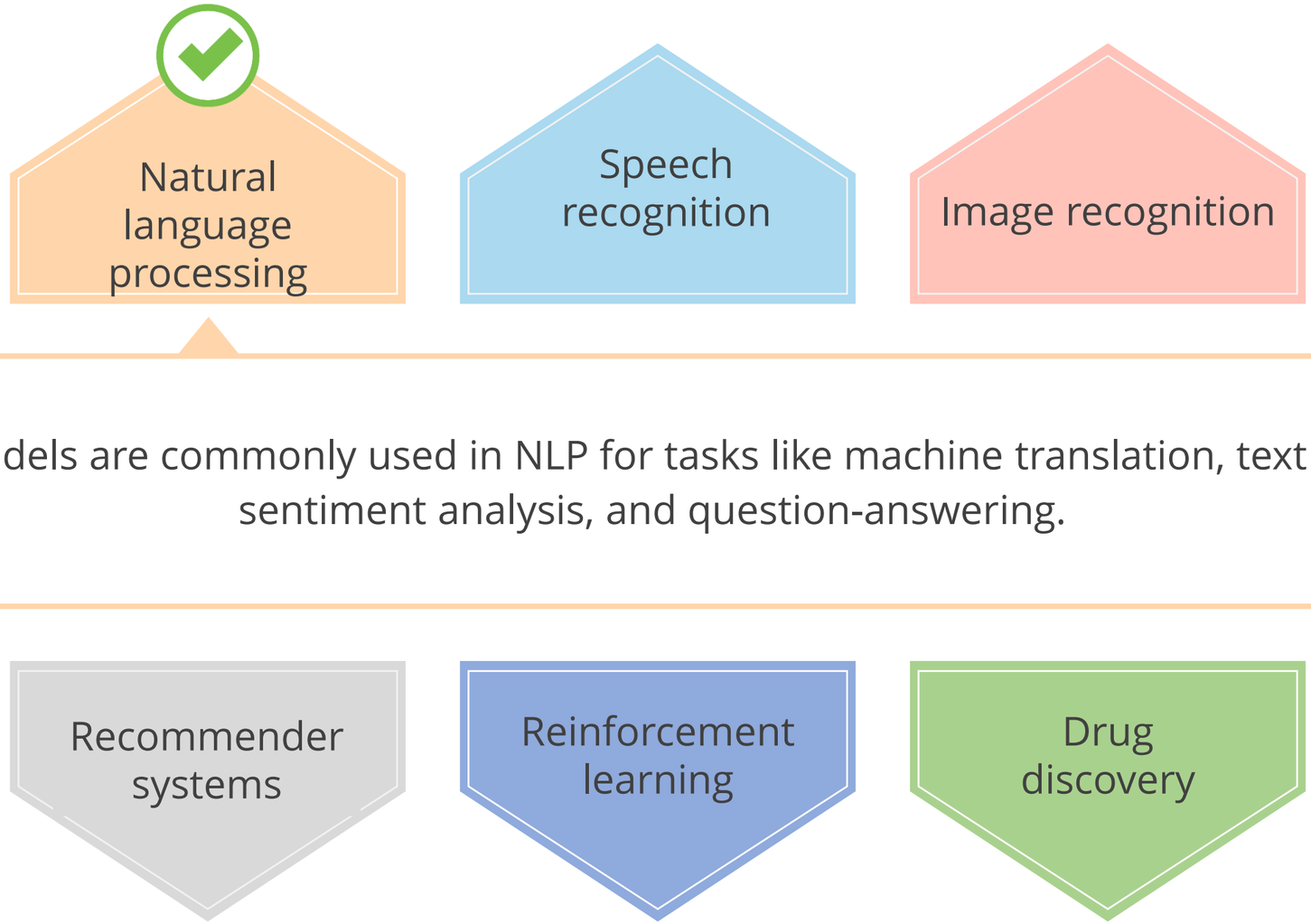
Long-Range Dependencies

The algorithms (RNNs and LSTMs) often struggle to determine the subject of a sentence when processing a whole paragraph.

Eventually, it loses its association with other subjects in the input.

Transformer models retain the context of pronoun references and exhibit strong proficiency in recognizing the grammar of provided sentences.

Applications of Transformer Models



Applications of Transformer Models

The transformer model is employed in automatic speech recognition systems to convert spoken language into written text.

Applications of Transformer Models

Natural language processing

Speech recognition

Image recognition

The transformer model is adapted for computer vision tasks such as image classification, object detection, and image captioning.

Recommender systems

Reinforcement learning

Drug discovery

Applications of Transformer Models

Natural language processing

Speech recognition

Image recognition

The transformer model is applied in recommendation systems to provide personalized recommendations based on user preferences and behavior.

Recommender systems

Reinforcement learning

Drug discovery

Applications of Transformer Models

Natural language processing

Speech recognition

Image recognition

Transformer models are used in reinforcement learning to enhance decision-making in sequential tasks like game-playing and robot control.

Recommender systems

Reinforcement learning

Drug discovery

Applications of Transformer Models

Natural language processing

Speech recognition

Image recognition

Transformer models are utilized in drug discovery to predict molecular properties, design new molecules, and speed up the drug development process.

Recommender systems

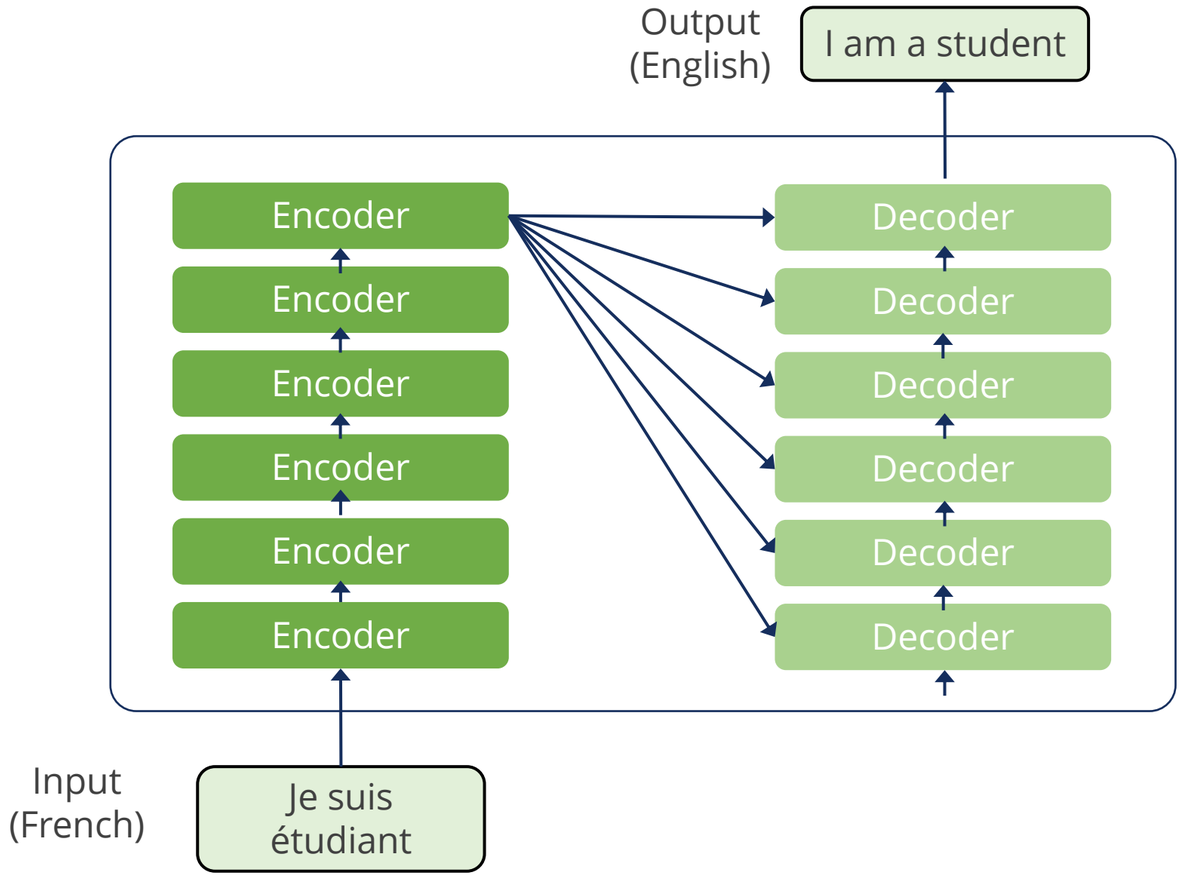
Reinforcement learning

Drug discovery

Architecture of the Transformer Model

Transformer Model Architecture

At a fundamental level, a transformer architecture comprises two primary components: an encoder and a decoder.



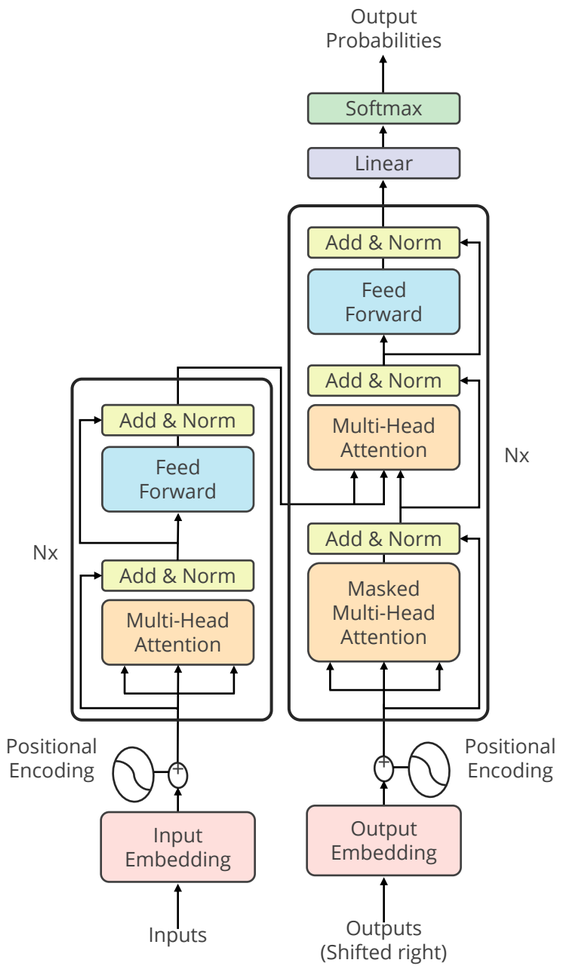
Transformer Model Architecture

Working of encoder and decoder

- The encoder processes input sequences and captures contextual information, employing self-attention to integrate information across the entire sequence.
- The decoder generates an output sequence and predicts the next word based on context.
- The architecture ensures accurate and coherent sequence processing and generation.

Transformer Model Architecture

In transformer architecture, multiple identical encoders and decoders layer for sequence-to-sequence processing, as shown below:



- It is common to have six or more of these identical layers.
- The stacking of these layers enables effective information capture and generation in transformer models.

Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

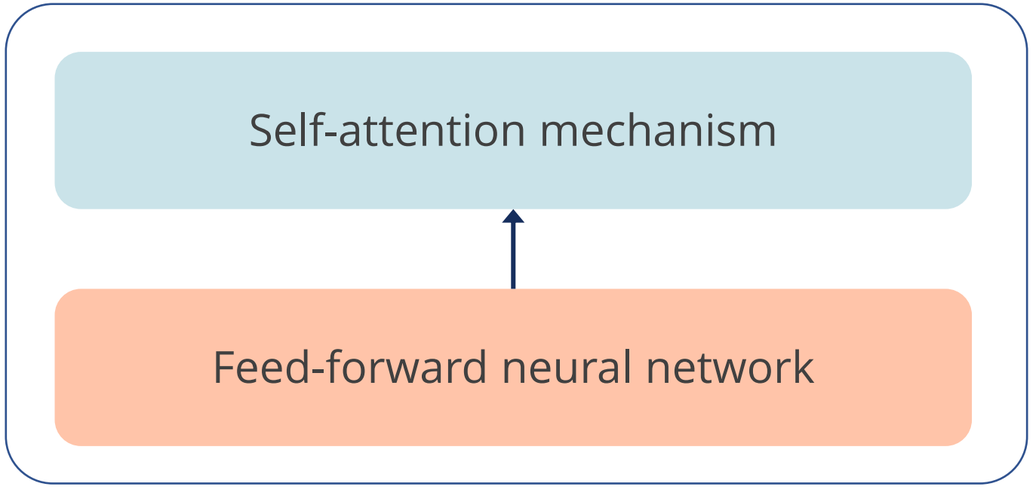
Working of Transformer in Language Translation

Transformer Working: Encoder

An example of language translation to understand the workings of a transformer:

Input: Je suis étudiant

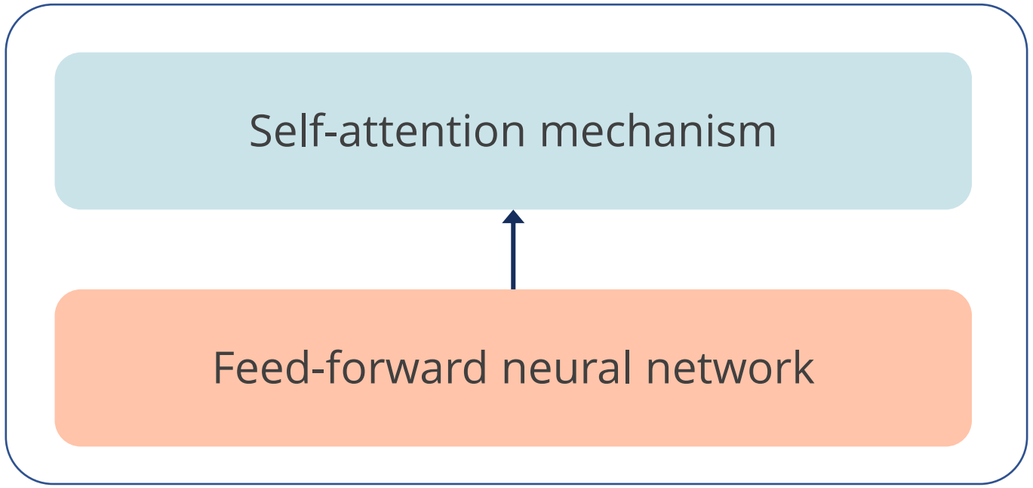
Each encoder has two layers:



- **Input processing**: The Transformer model begins by taking an input, in this case, '**Je suis étudiant'** (I am a student in French). Each word is first converted into a vector through an embedding process.
- **Positional encoding**: After embedding, positional encoding is added to each word vector. This step is crucial as it injects information about the position of each word in the sequence into its vector representation, allowing the model to recognize word order, which is essential for understanding the syntax and semantics of the input sentence.

Transformer Working: Encoder

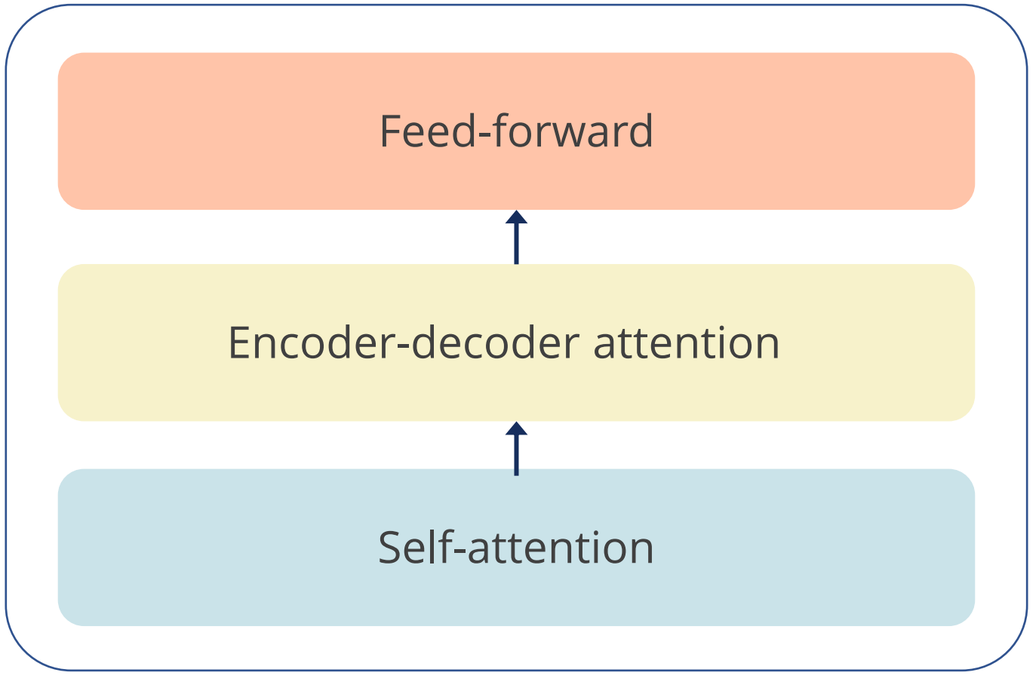
Each encoder has two layers:



- **Passing through encoders:** The word vectors, now enhanced with positional information, pass through multiple layers of encoders. Each encoder layer processes the vectors, refining and enriching the representations with context from the entire sentence. This process leverages self-attention and feed-forward neural networks within each layer.

Transformer Working: Decoder

A decoder has three layers:

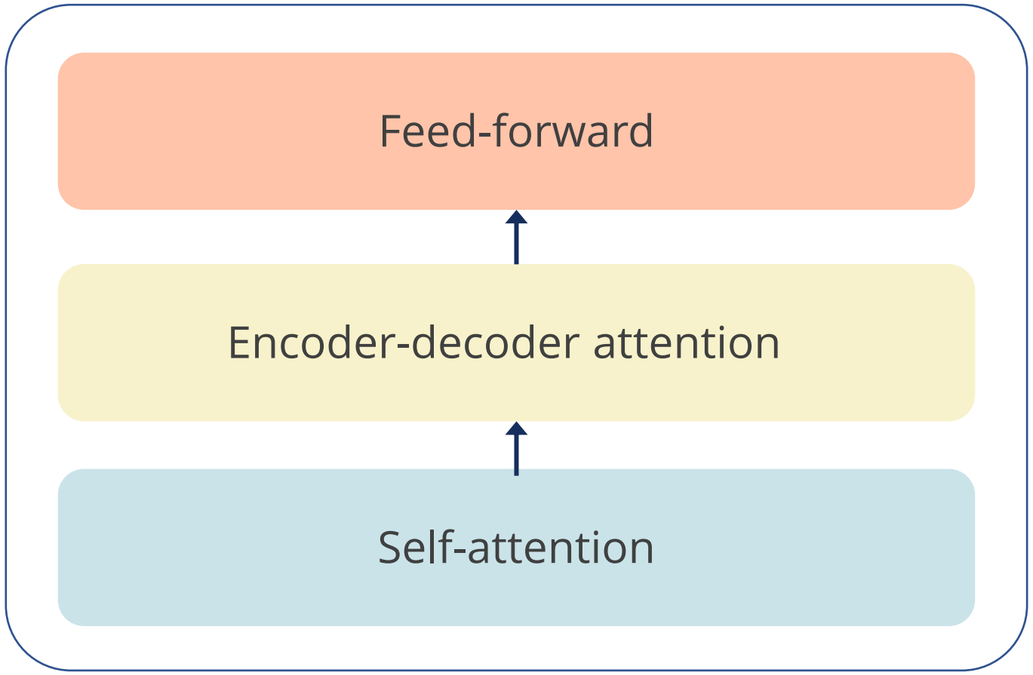


- **Receiving encoder outputs**: The decoder begins its process by receiving the entire sequence of outputs from the encoder. These outputs contain encoded information about every word in the input sequence, providing a comprehensive context that the decoder will use to generate the translation.

- **Output sequence initialization:** The decoder generates the output sequence by receiving a special start token. This token serves as the initial input for the decoder layers.

Transformer Working: Decoder

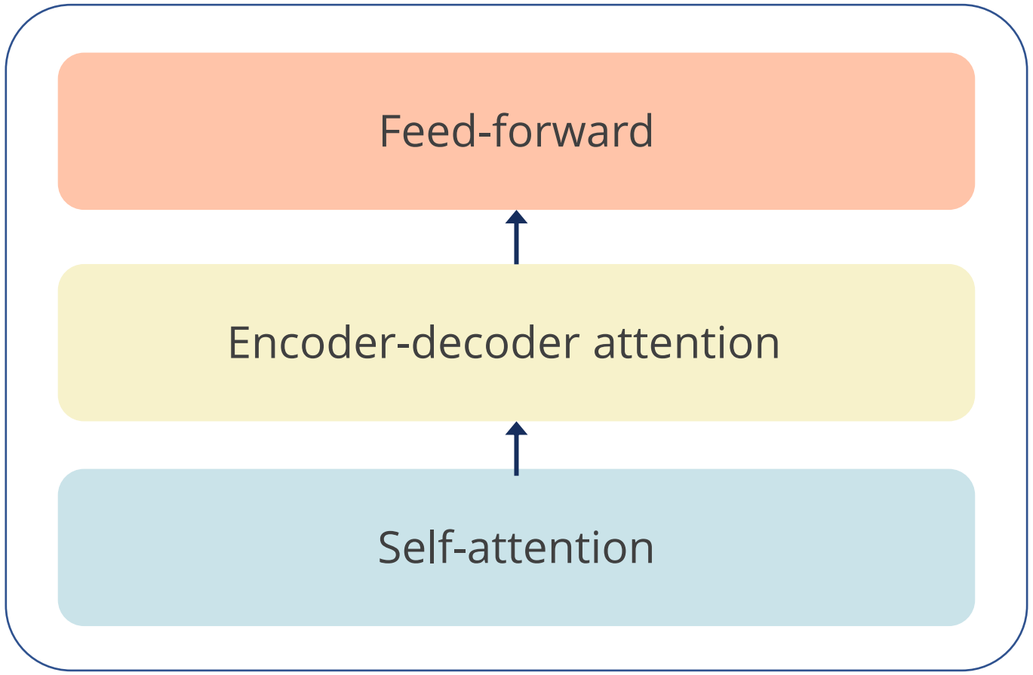
A decoder has three layers:



- **Self-attention mechanism:** Each decoder layer first applies a self-attention mechanism with a restriction. This self-attention only allows each position in the decoder to attend to earlier positions in the output sequence. This ensures that the predictions for each word are dependent only on the known previous words, preserving the auto-regressive property necessary for coherent generation. This mechanism helps the decoder understand the context within the output sequence itself, enhancing the flow and grammatical structure of the translation.

Transformer Working: Decoder

A decoder has three layers:

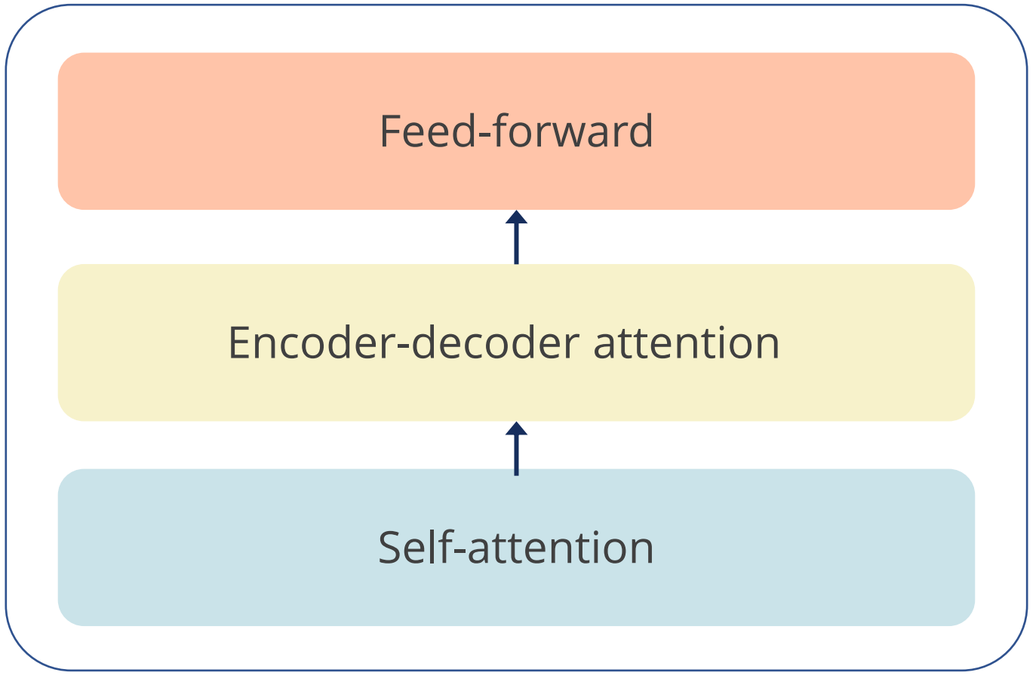


- **Encoder-decoder attention:** After processing through its self-attention layer, each decoder layer uses an encoder-decoder attention mechanism. This crucial layer allows the decoder to focus on relevant parts of the input sequence for each word being generated in the output. By attending to the encoder's outputs, the decoder effectively utilizes the input sequence's contextual information, ensuring that the generated translation is semantically aligned with the input.

- **Feed-Forward Neural Networks:** Like the encoder, each decoder layer includes a position-wise feed-forward neural network. This network processes the output of the attention mechanisms and produces the intermediate representations that generate the next word in the output sequence.

Transformer Working: Decoder

A decoder has three layers:

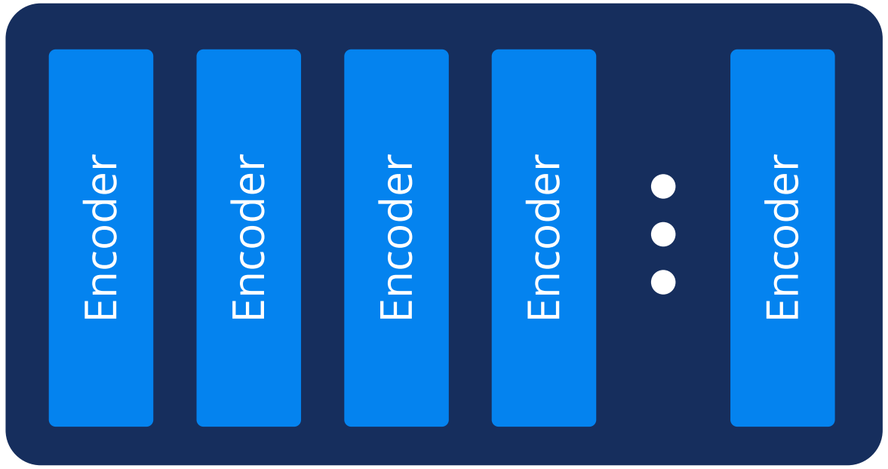


- **Output generation:** The final layer in the decoder transforms the intermediate representations into logits, which pass through a softmax layer to form a probability distribution over the possible output words. The word with the highest probability is selected as the next word in the output sequence.
- **Repeat the process:** This process repeats for each word in the output sequence until the decoder generates an end-of-sequence token, signaling the completion of the output generation.

Introduction to BERT Model

BERT Model

BERT stands for Bidirectional Encoder Representations from Transformers.



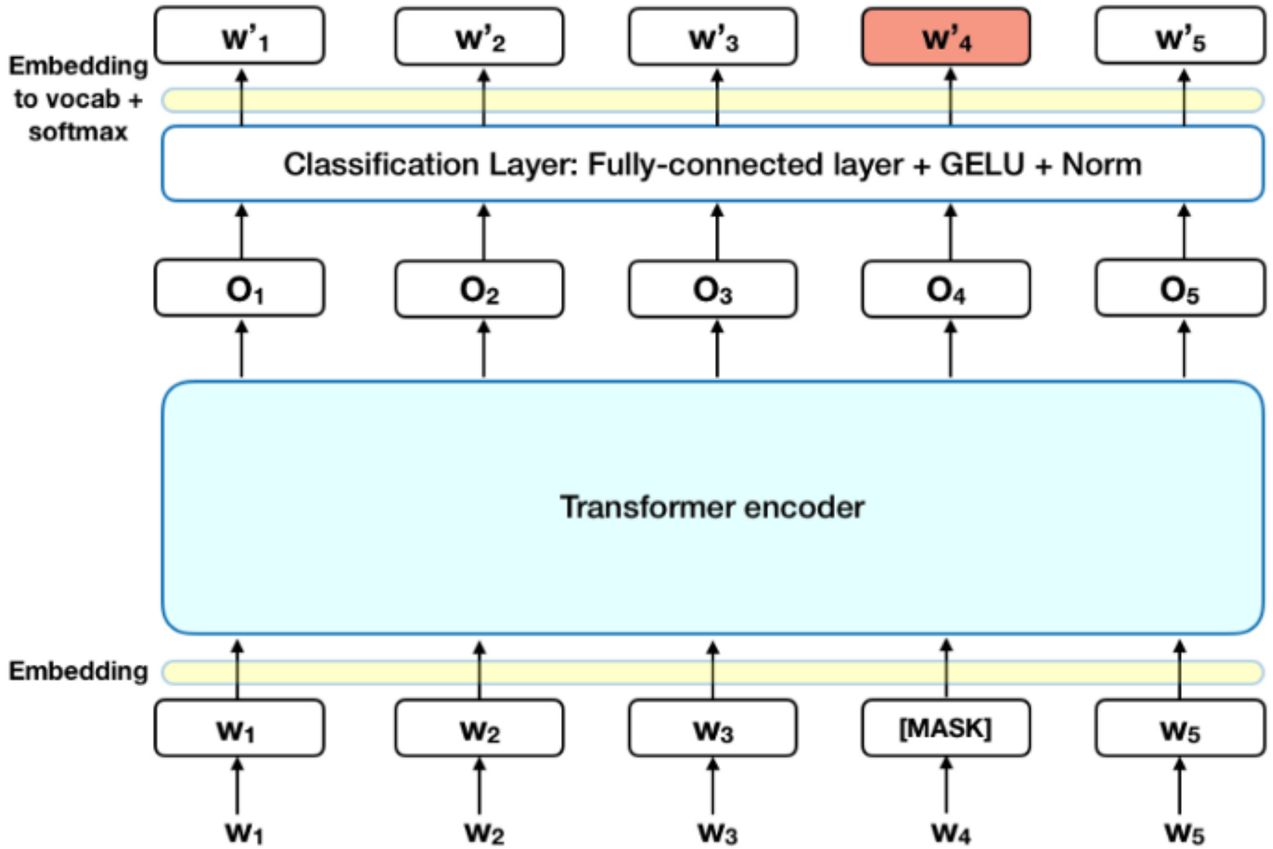
It is a transformer model without decoder modules and only has a trained encoder stack.

The transformer encoder tends to read the entire sequence of words at once.

It learns the context of the given input rather than learning it in sequence. It is called contextual learning.

BERT and Masked Language Modeling

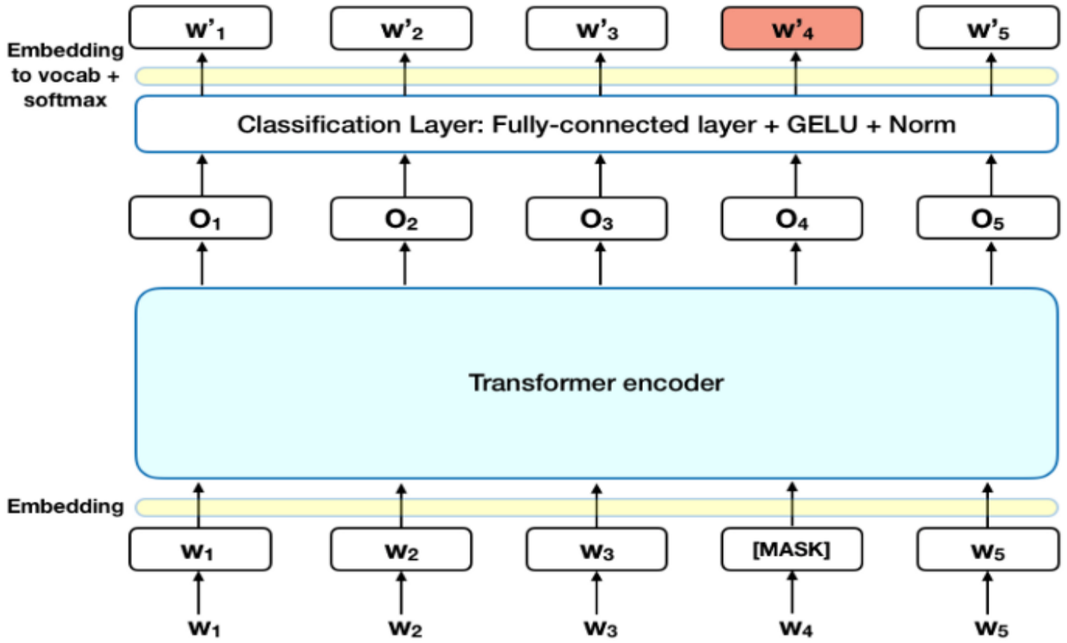
A Masked Language Model (MLM) is a type of language model used in the pre-training of models like BERT (Bidirectional Encoder Representations from Transformers). The working of an MLM is as shown below:



Source: Lappin, Shalom. *Deep learning and linguistic representation*. Chapman and Hall/CRC, 2021.

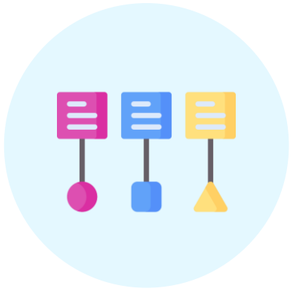
BERT and Masked Language Modeling

In Masked Language Modeling (MLM), 15% of the words are replaced with a MASK token before feeding the sequence of words into BERT.



The model attempts to predict the original values of the masked words by leveraging the contextual information available.

Use Cases for BERT



Text classification: It identifies text characteristics like fraud detection.

Text generation: It generates text, specifically chatbot responses.

Use Cases for BERT

Search engine optimization: It improves search relevance for user queries.

Question-answering (Q&A) system: It helps in accurate Q&A responses.

Assisted Practices

Let’s understand the concept of BERT and its text classification using Jupyter Notebooks.

- 12.05_Introduction_to_BERT__V3
- 12.06_Text_Classification_using_BERT

Note: Please refer to the **Reference Material** section to download the notebook files corresponding to each mentioned topic.

ASSISTED PRACTICE

Key Takeaways

- Transformers are known for their long-range memory dependencies and acquire this functionality through self-attention.
- A transformer architecture comprises two primary components: an encoder and a decoder.
- BERT is a transformer model without any decoder module, and it has a trained encoder stack.
- In an MLM, 15% of the words are replaced with a MASK token before feeding the sequence of words into BERT.

Knowledge Check

Knowledge Check 1

What major challenge do Transformer models overcome compared to earlier models like RNNs and LSTMs?

A. They reduce the computational time for each task.

B. They excel at capturing long-range dependencies.

C. They require less data for training.

D. They eliminate the need for attention mechanisms

Knowledge Check 1

What major challenges do Transformer models overcome compared to earlier models like RNNs and LSTMs?

A. They reduce the computational time for each task.

B. They excel at capturing long-range dependencies.

C. They require less data for training.

D. They eliminate the need for attention mechanisms.

The correct answer is **B**

Transformer models effectively address the challenge of long-range dependencies that earlier models like RNNs and LSTMs struggled with due to their use of self-attention mechanisms.

Knowledge Check 2

In the Transformer Model Architecture, what is the primary role of the encoder and decoder components?

A. The encoder processes input sequences into a fixed format, and the decoder directly outputs translated text.

B. The encoder generates an output sequence, and the decoder predicts the next word in the input sequence.

C. The encoder processes input sequences and integrates information using self-attention, and the decoder generates the output sequence based on this contextual information.

D. The encoder and decoder function independently to translate sentences without sharing information.

Knowledge Check 2

In the Transformer Model Architecture, what is the primary role of the encoder and decoder components?

A. The encoder processes input sequences into a fixed format, and the decoder directly outputs translated text.

B. The encoder generates an output sequence, and the decoder predicts the next word in the input sequence.

C. The encoder processes input sequences and integrates information using self-attention, and the decoder generates the output sequence based on this contextual information.

D. The encoder and decoder function independently to translate sentences without sharing information.

The correct answer is **C**

The encoder processes input sequences and integrates information using self-attention, and the decoder generates the output sequence based on this contextual information.

Thank You!