Transformers are among the most prominent encoder-decoder architectures in natural language processing. Transformers and attention mechanisms have revolutionized the field of deep learning, offering a powerful way to process sequential data and capture long-range dependencies. In this article, we will dive into the transformers, encoder-decoder architecture, application, advantages, challenges.

Transformer

Transformers were first developed to solve the problem of sequence transduction, or neural machine translation, which means they are meant to solve any task that transforms an input sequence to an output sequence. This is why they are called Transformers. The transformer architecture is composed of an encoder and a decoder, each of which is made up of multiple layers of self-attention and feedforward neural networks. The self-attention mechanism is the heart of the transformer, allowing the model to weigh the importance of different words in a sentence based on their affinity with each other. This is like how a human might read a sentence, focusing on the most relevant parts of the text rather than reading it linearly from beginning to end. In addition to self-attention, the transformer also introduces positional bias, which allows the model to keep track of the relative positions of words in a sentence. This is important because the order of words in a sentence can significantly impact its meaning.

Working of Encoder-Decoder Architecture

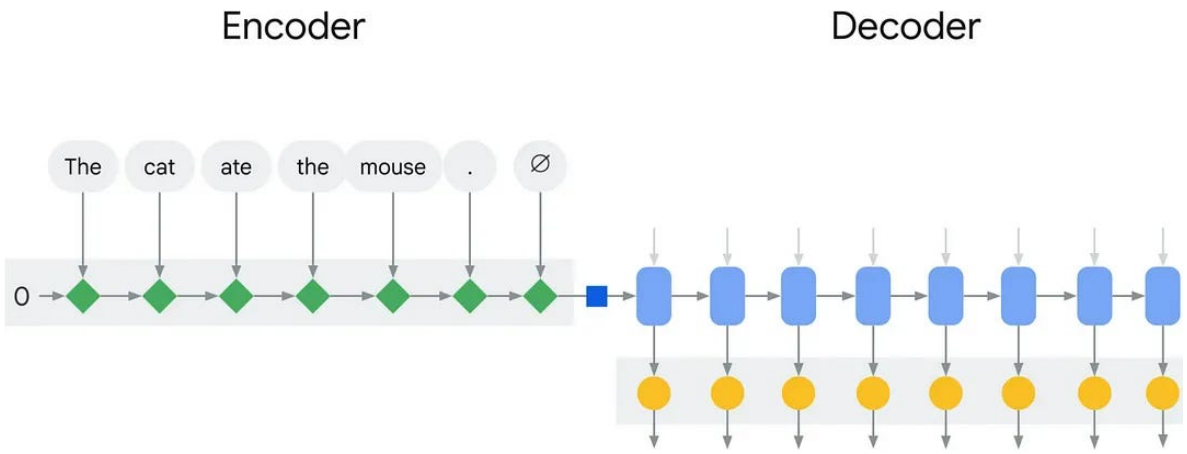
The encoder-decoder architecture is a deep learning architecture used in many natural language processing and computer vision applications. It consists of two main components: an encoder and a decoder.

The encoder is the primary portion of an encoder-decoder architecture. It takes in an input sequence and processes it to form a set of context vectors, which are then utilized by the decoder. For example, for text applications such as machine translation or summarization, the words in each sentence will be converted into numerical values that represent them mathematically. Then, these numbers are fed through a series of layers that reduce their dimensionality while preserving relevant and important data about how they relate to one another within the sentence structure. This “encoded” version of each sentence is then passed along to the decoder for further processing.

The decoder is responsible for taking this encoded representation and reconstructing it back into its original form. To do this, there must be some kind of relationship between what was encoded and what needs to be reconstructed else it would just be random guessing. To establish such a link, most modern architectures use attention mechanisms that allow specific parts of an input sequence such as individual words to influence how later parts are processed or interpreted by the model i.e. essentially giving greater weightage or importance to certain elements over others when generating

output sequences from encoding data inputs. By doing so, models become much more accurate at producing outputs that accurately reflect their input data sources and can even learn different patterns across various datasets without needing additional training cycles or parameter tuning procedures afterwards.

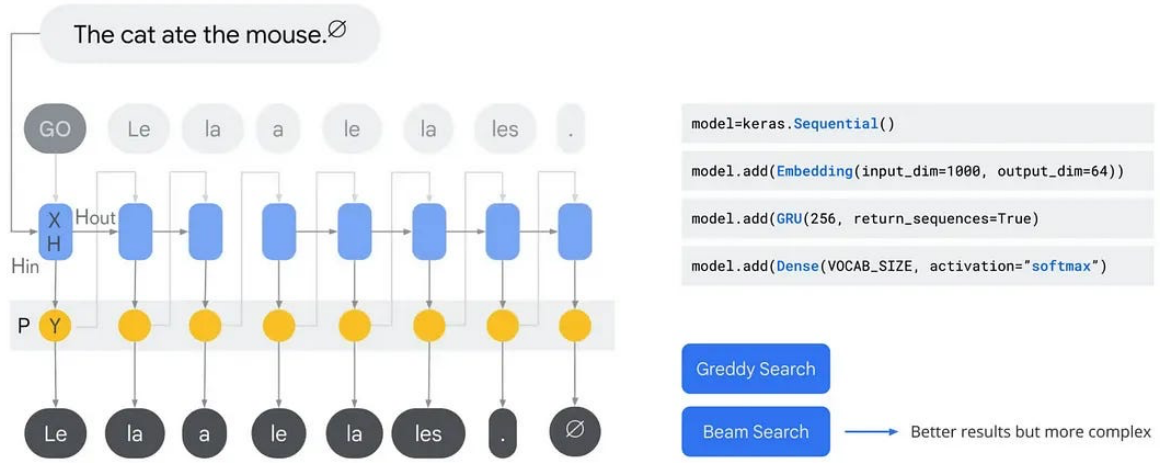
Different kinds of neural networks including RNN, LSTM, CNN, and transformer can be used based on encoder-decoder architecture. Encoder-decoder architecture is found to be most suitable for the use cases where input is a sequence of data and output is another sequence of data.



Source: — Google Cloud Skills Boost

The basic encoder-decoder architecture uses recurrent neural network (RNN) layers, modern large language models replace them with Transformer blocks. Transformers, developed at Google, rely on attention mechanisms to process sequences more effectively. The attention mechanism allows the model to focus on relevant parts of the input sequence while generating the output. This advancement

has greatly improved the performance and capabilities of language models.

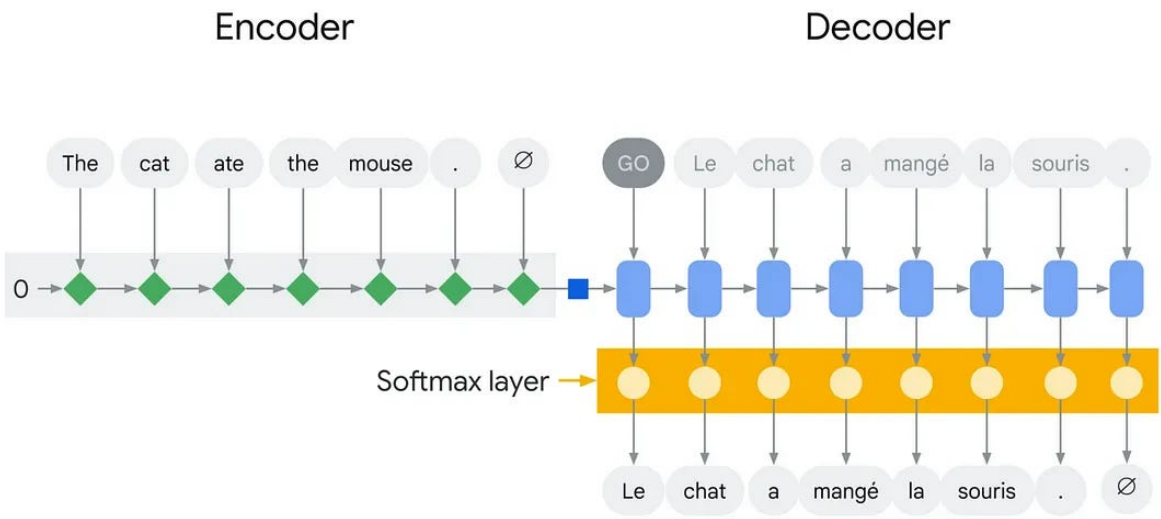


Source: — Google Cloud Skills Boost

To train an encoder-decoder model, a dataset of input-output pairs is required. This dataset contains examples of input sequences and their corresponding desired output sequences. During training, the model adjusts its weights based on the error it produces between the generated output sequence and the true output sequence from the dataset. For tasks like machine translation, sentence pairs in different languages are needed. The source language sentence is fed to the encoder, and the error is computed based on the decoder's generated translation compared to the actual translation. However, during training, the decoder also requires its own input, which would be the correct previously translated token. This training method is also referred to as "teacher forcing" because the decoder is forced to generate the next token based on the correct token provided previously.

Generating Text with Encoder-Decoder Models

After training, the encoder-decoder model can be used for text generation. To generate text, the model starts by feeding the encoder input representation for e.g., a prompt to the decoder, along with a special token like “go” which in turn prompts the decoder to generate the first word of the output sequence. The decoder’s RNN layer updates the state based on the encoder’s representation, and a SoftMax layer produces word probabilities. The word or chunk with the highest probability is selected using either greedy search or beam search. This process is repeated to generate subsequent words or chunks until the desired output sequence is obtained.



Source: — Google Cloud Skills Boost

Real-world applications of encoder-decoder architecture

Some of the applications of the encoder-decoder architecture are as follows:

1. Language translation
2. Image captioning
3. Speech recognition
4. Text summarization

Advantages of the encoder-decoder architecture

- **Versatility**: The architecture's versatility allows it to be applied across a wide array of AI tasks, from language processing to computer vision.
- **Enhanced Accuracy**: Its ability to effectively process and decode data leads to more accurate and contextually relevant outputs.

Limitations/Challenges of the encoder-decoder architecture

- **Complex Training**: Developing and training encoder-decoder models can be computationally intensive and require substantial resources.
- **Information Loss**: In certain scenarios, the architecture may encounter challenges in preserving all relevant information during the encoding and decoding processes.