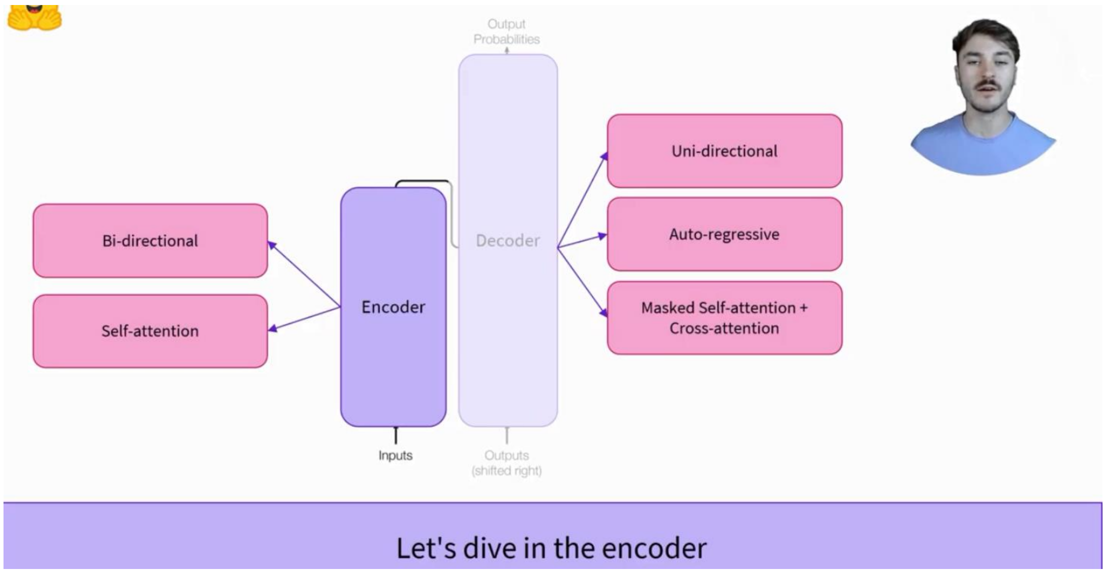
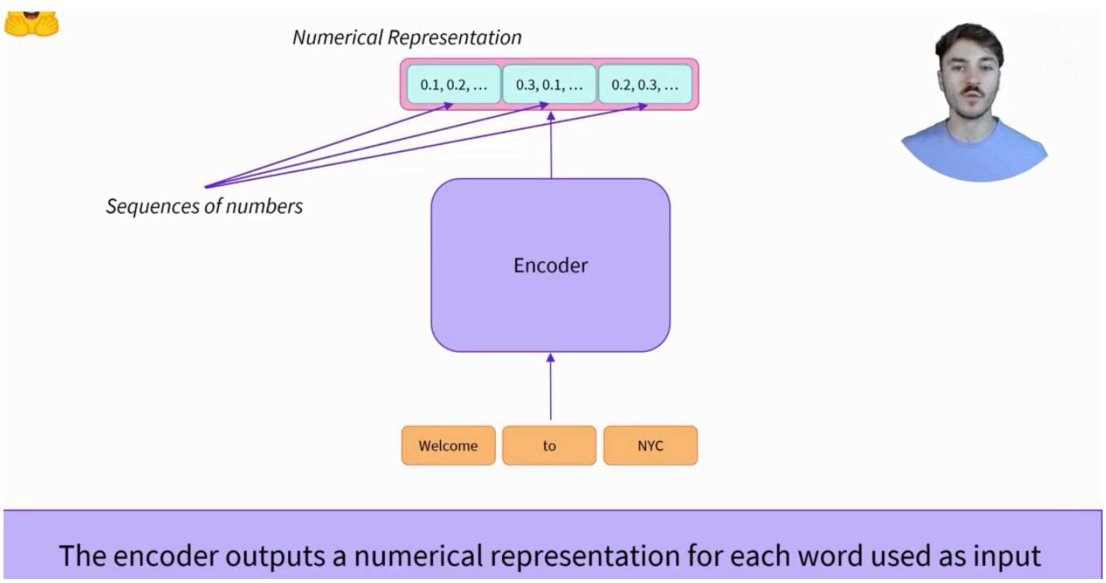
Encoders

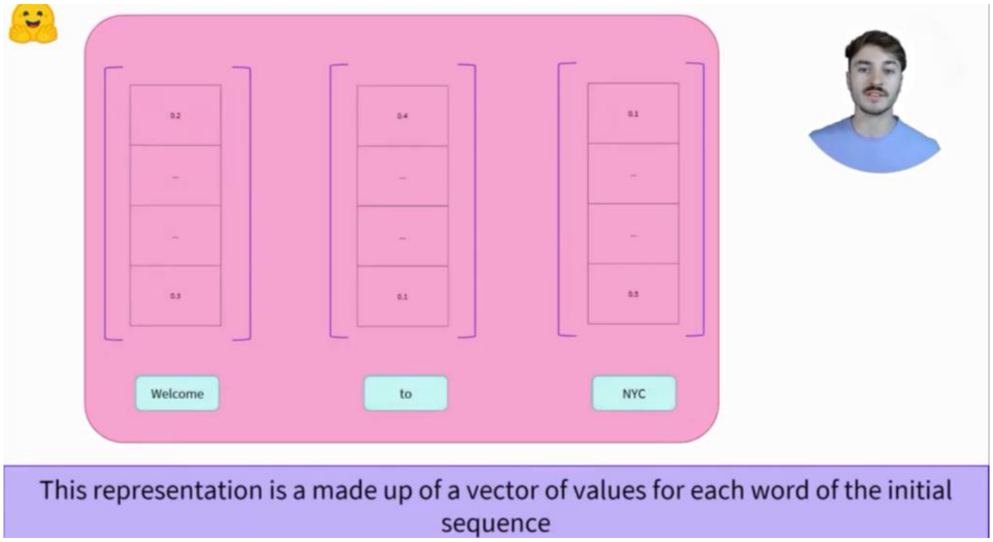
How do they work?

A general high-level introduction to the Encoder part of the Transformer architecture. What is it, when should you use it? BERT is a popular encoder-only architecture.

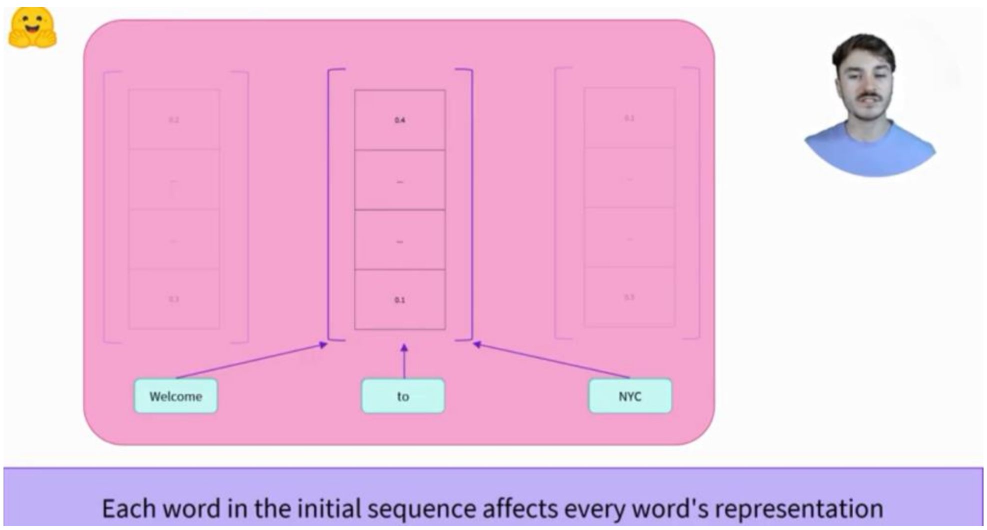




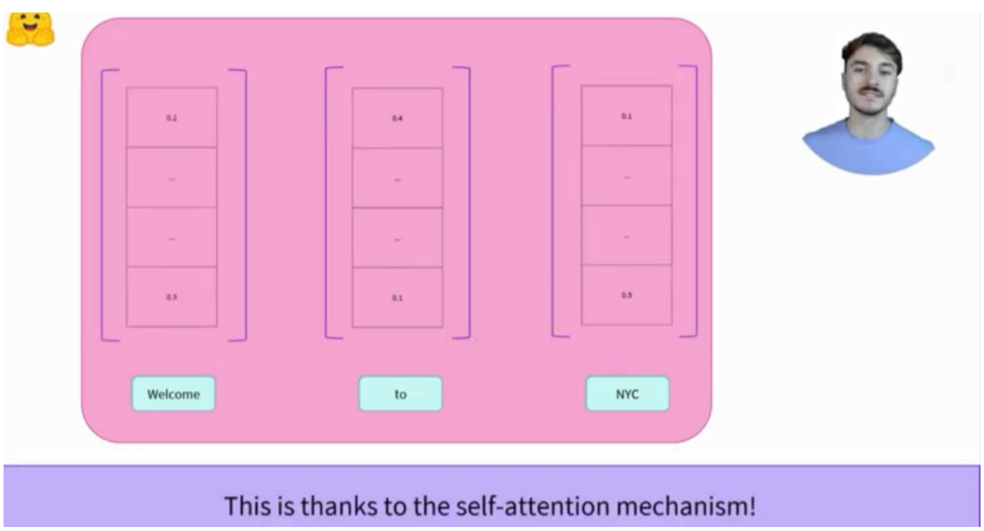
The encoder outputs one sequence of numbers per input word, this is called a feature vector or feature tensor.



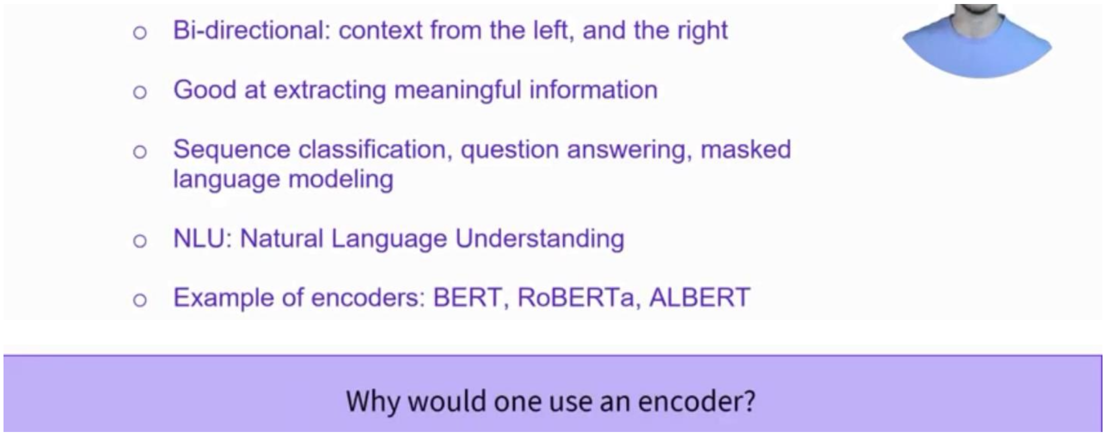
The dimension of each word vector is defined by the architecture of the model, for the base BERT model it is 768.



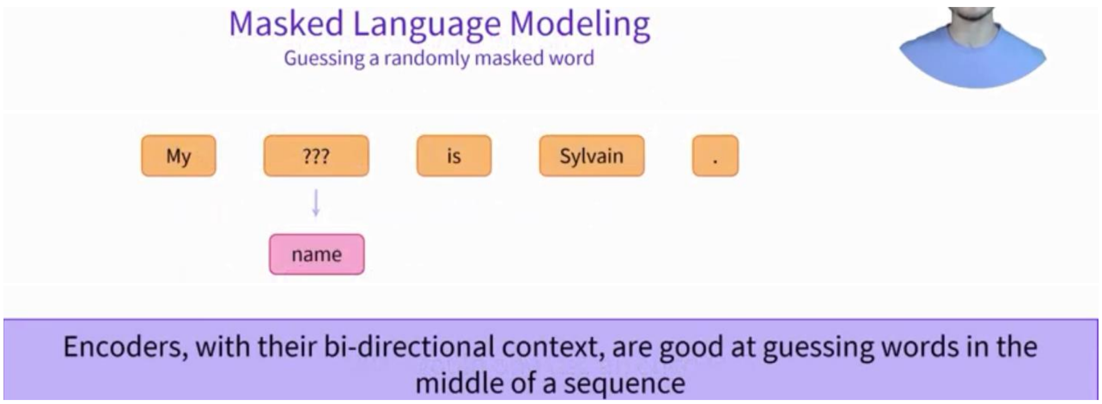
These representations contain the value of the word but contextualized, the representation of the word ‘to’ above also takes the surrounding left and right words into its context.



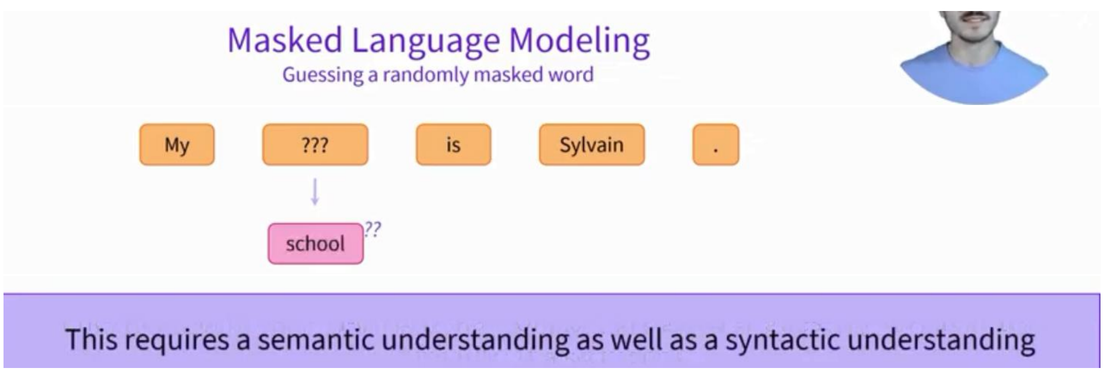
The self-attention mechanism relates to different positions of different words in a single sequence in order to compute a representation of that sequence. This means that the resulting representation of a word has been affected by other words in the sequence.

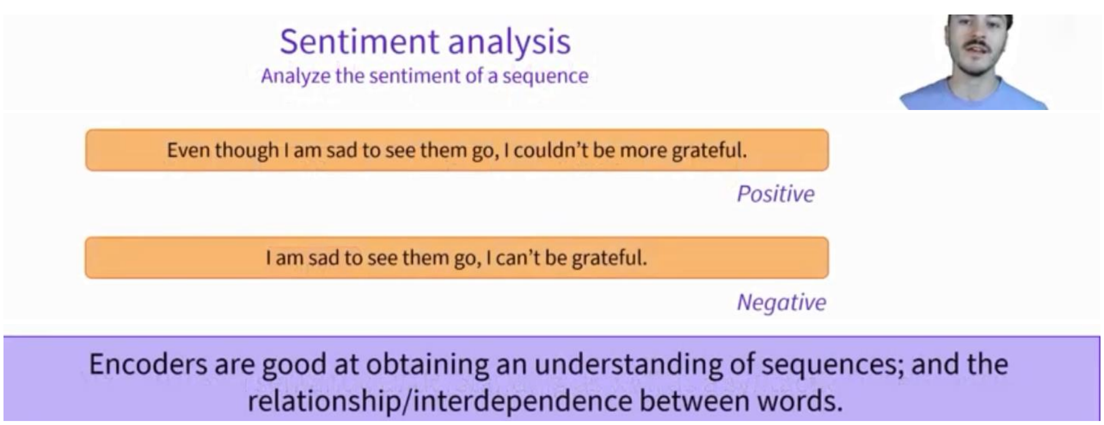


Encoders are very powerful at extracting vectors that carry meaningful information about a sequence, this vector can then be handled down the road by additional neurons to make sense of them.



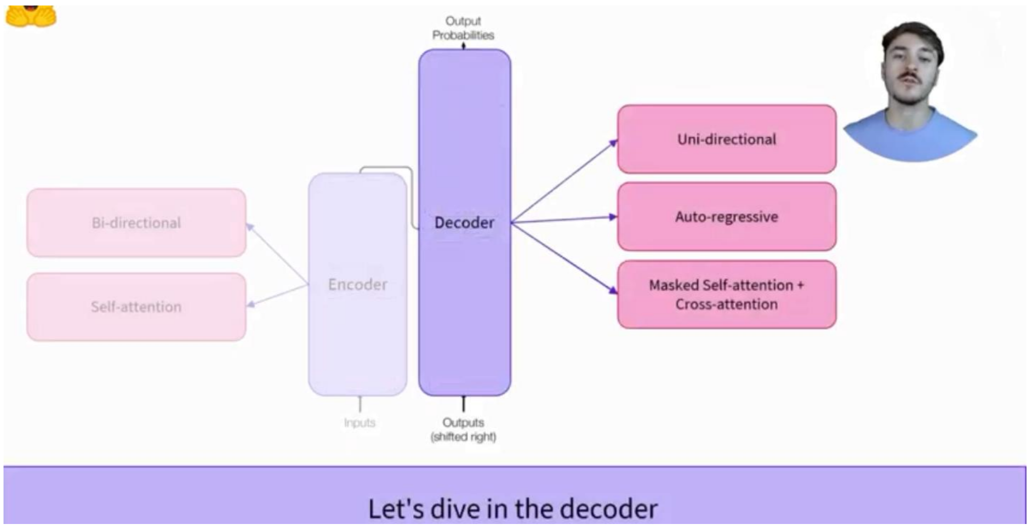
MLM is the task of predicting a hidden word in a sequence of words. BERT was trained to predict hidden words in a sequence of words. Encoders work great here since bidirectional words around the missing word are crucial in this task.



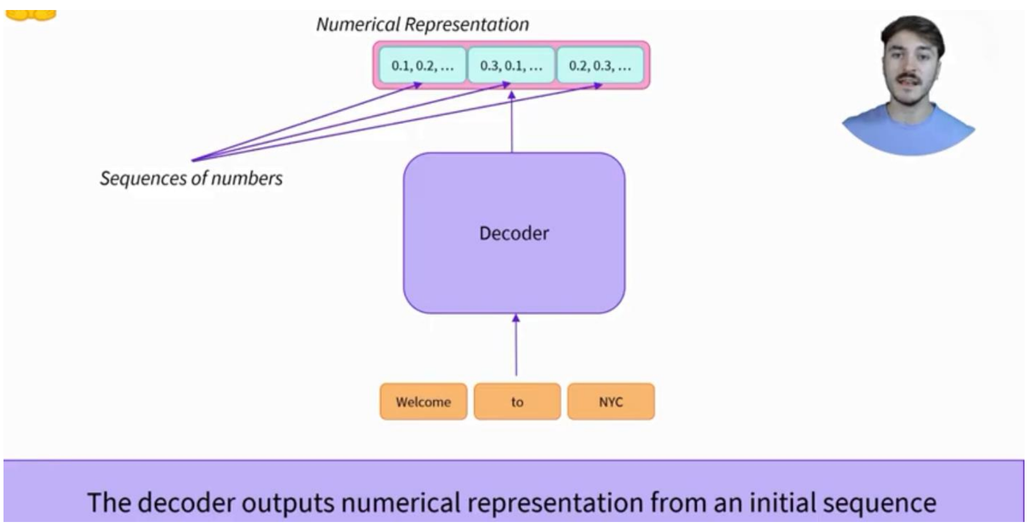


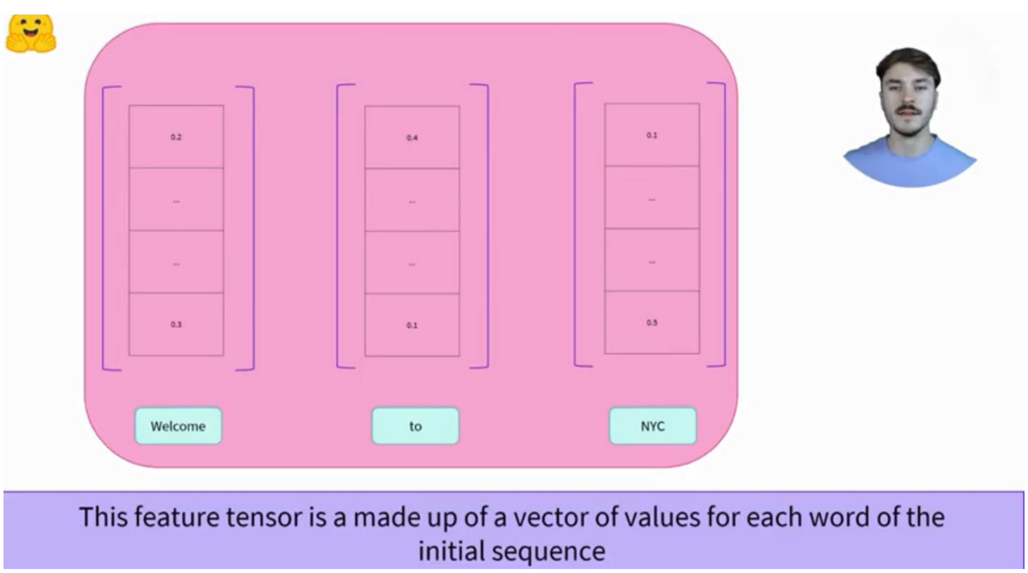
We will use the model to make a prediction that classifies the sequences among the 2 classes possible.

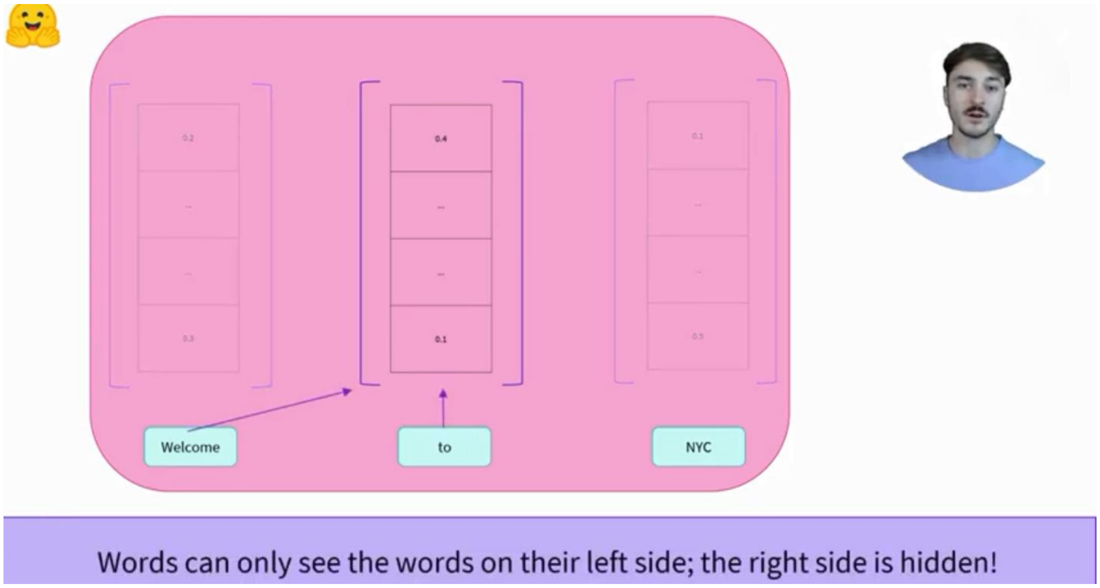
Let us now study the decoder architecture, a popular decoder-only architecture is GPT-2



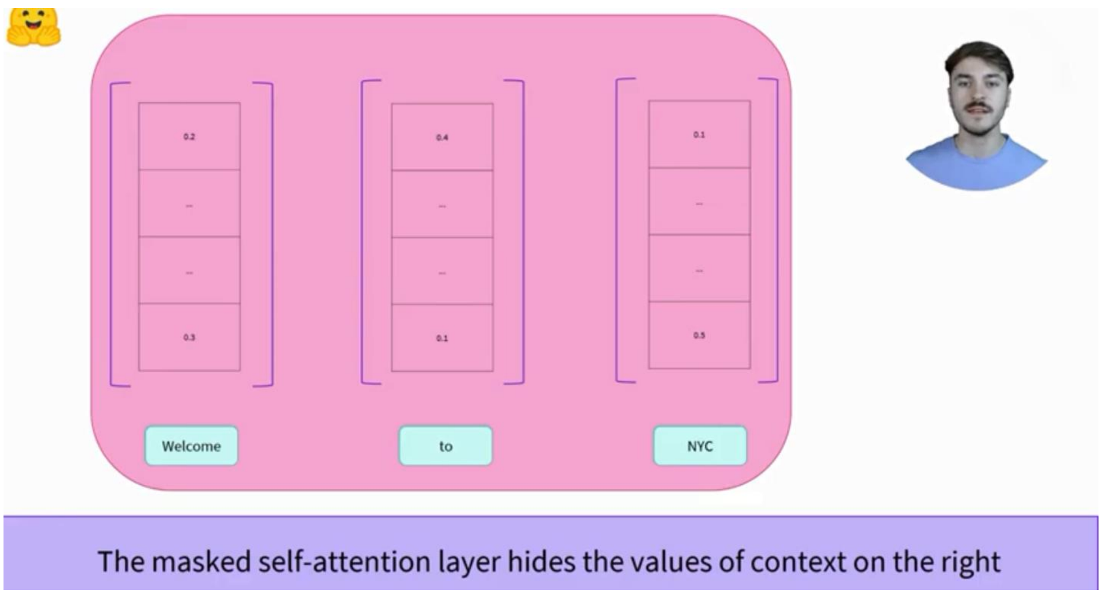
One can use a decoder for most of the same tasks as an encoder with a little loss of performance.

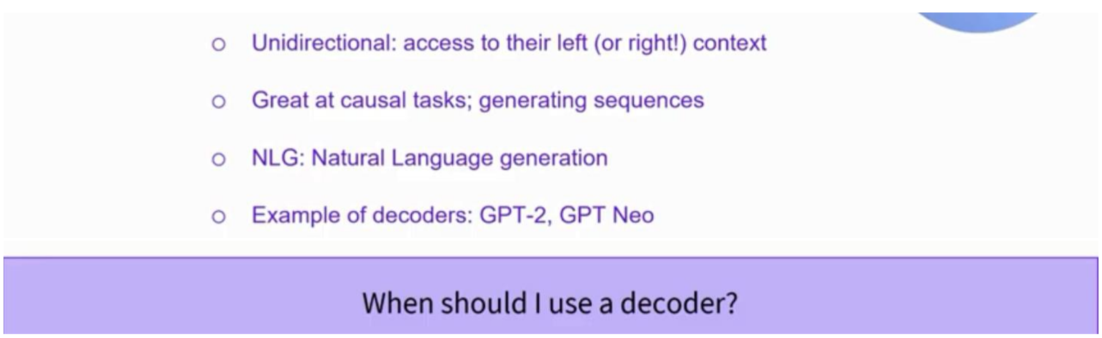




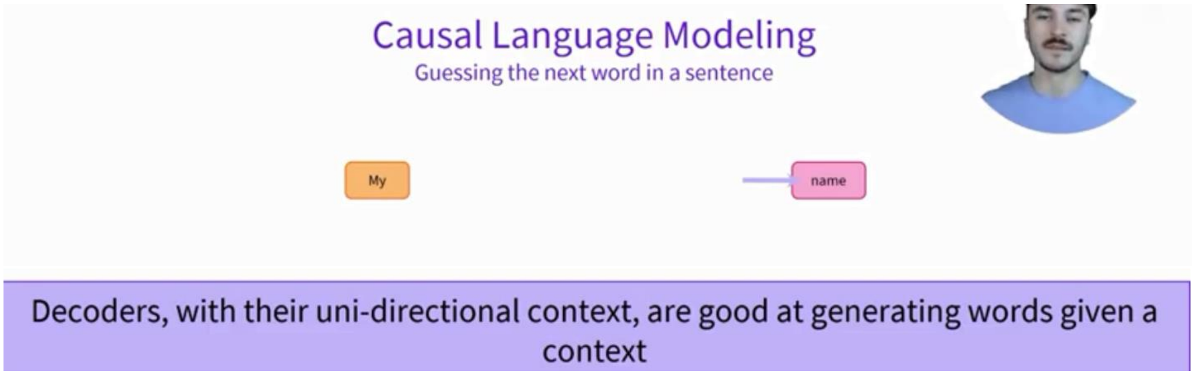


A decoder differs from an encoder in its self-attention mechanism, it uses a Masked Self-Attention because decoders only have access to a single context and are not bidirectional and can only look at the left of the word.

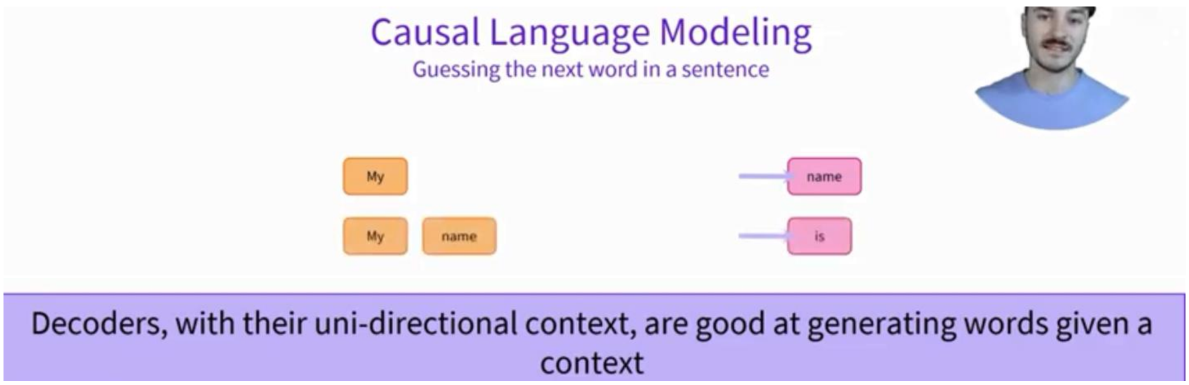




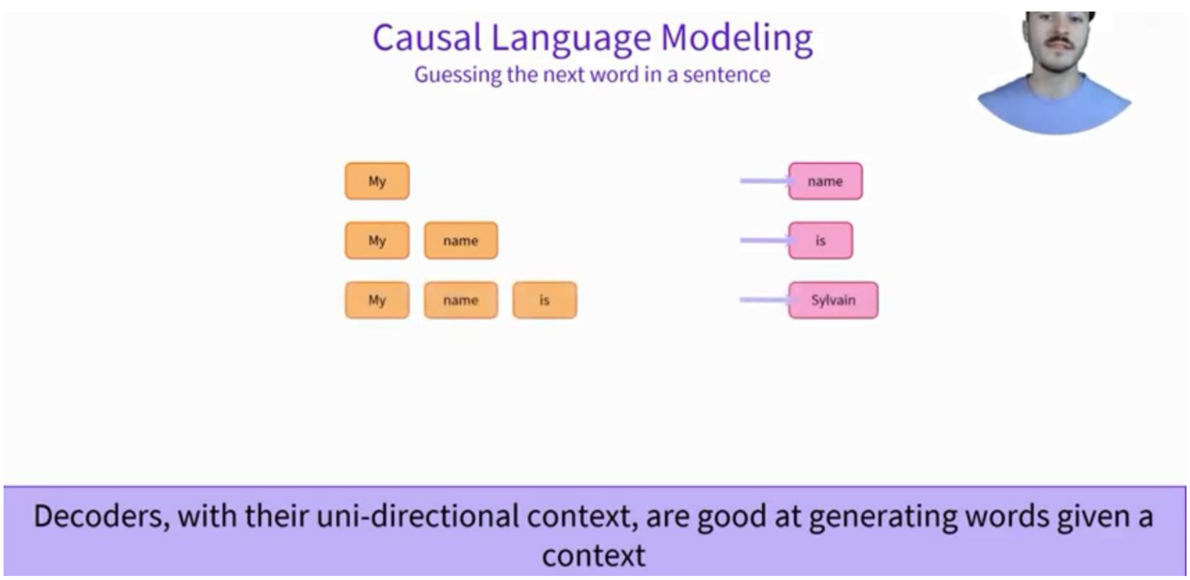
The strength of a decoder lies in the way the word can only have access to its left context, this explains the decoders good ability at text generation (NLG) for generating a word or a known sequence of words when given some words.



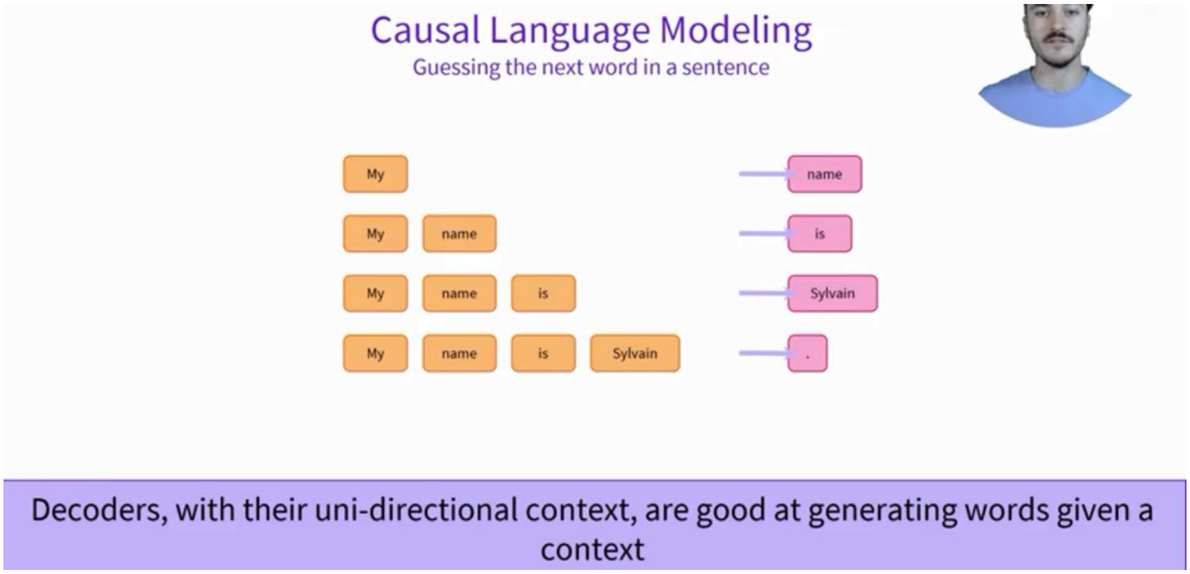
We start with an initial word which **My** as the input to the decoder model, the model outputs a vector of numbers that contains information about the sequence (which is a single word in the above example!). we apply a small transformation to that vector so that it maps to all the words known by the model which is a mapping that's called a language modelling head that identifies the word that the model thinks are probably the next sequence like **name**.



We then take that new word and add it to the initial sequence of words to get **My name** as the next decoder input. This is where the **auto-regressive** nature comes in by using their past outputs as inputs in the following steps. Once again, we do the exact same operation by passing that sequence through the decoder and retrieve the most probable following words.

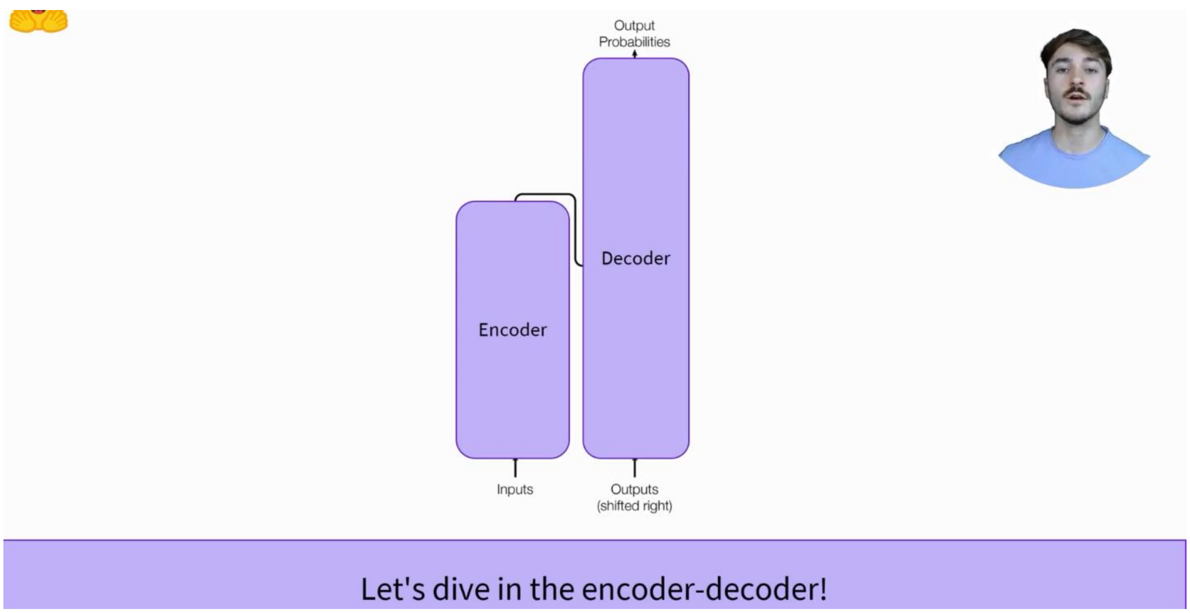


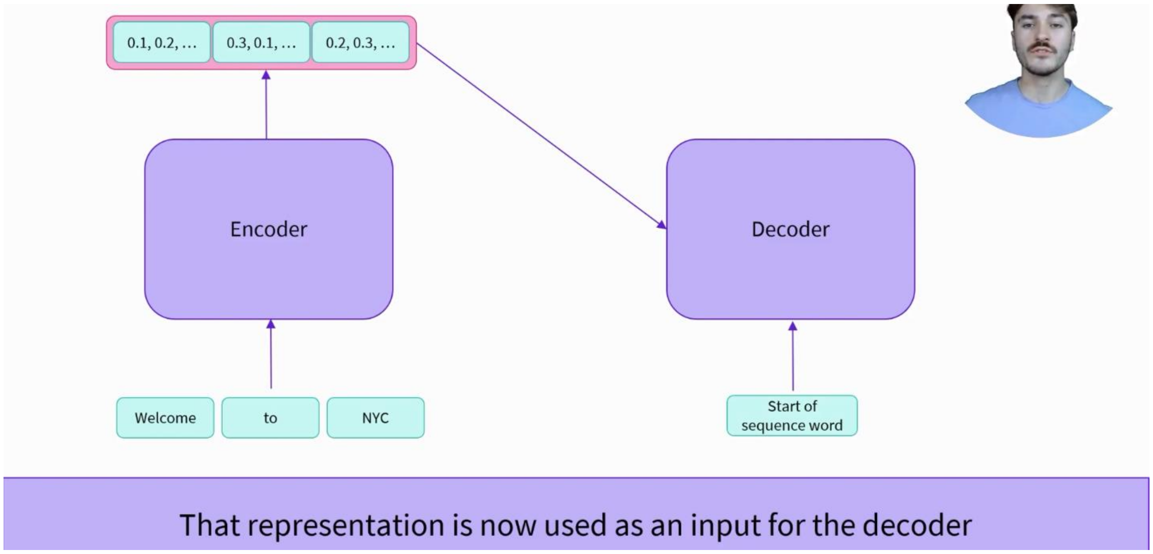
We repeat the operation until we are satisfied.



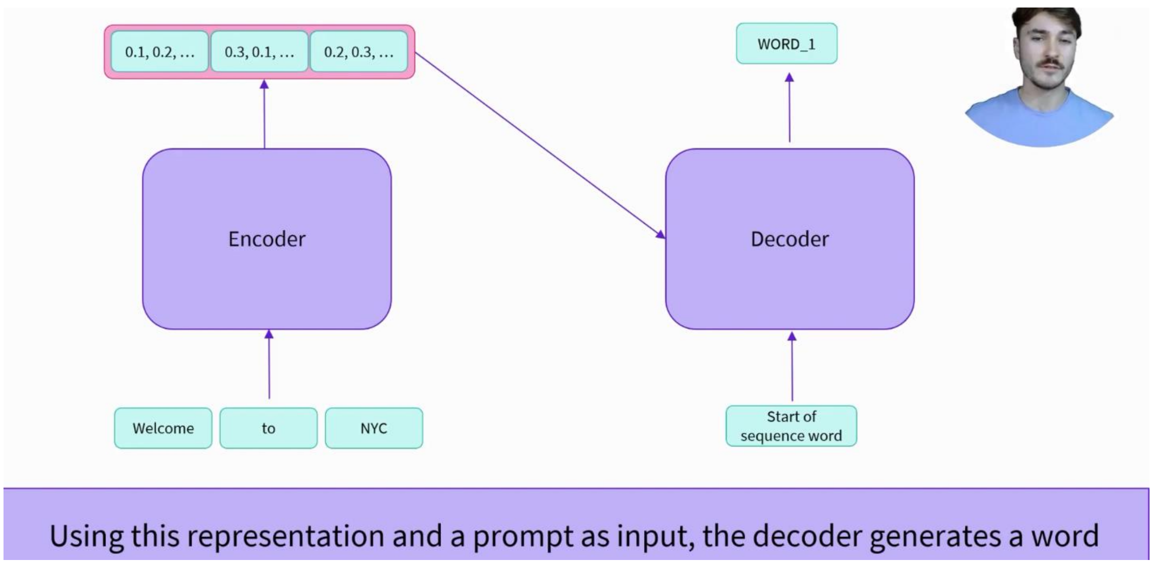
We have now generated a full sentence; we could continue for a while and can generate up to 1024 words and the decoder will still have some memory about the first word in the sequence. GPT-2 model has a maximum context size of 1024.

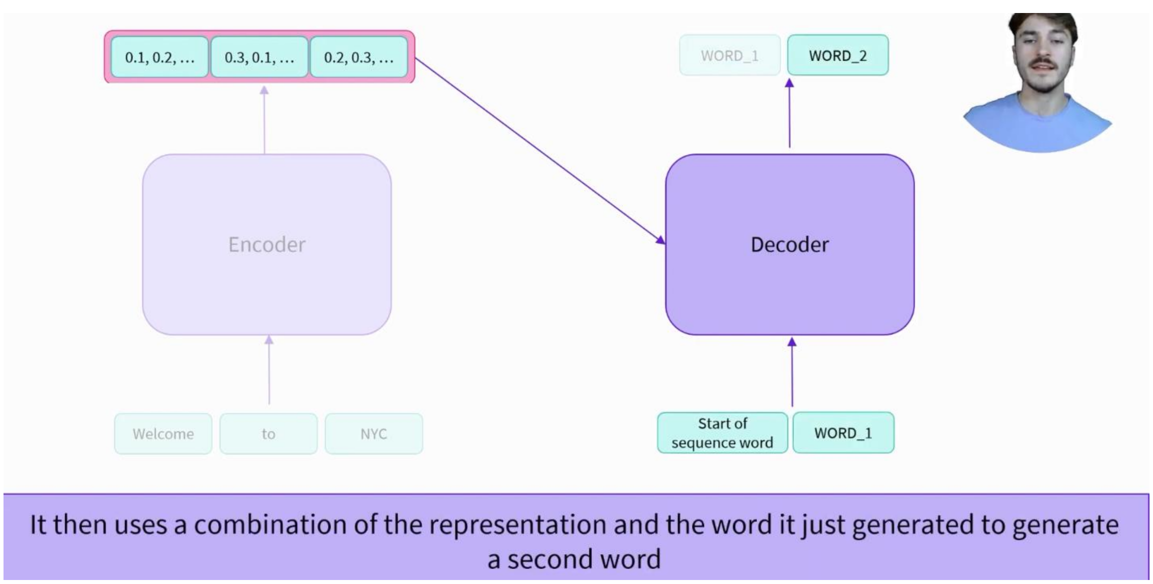
A general high-level introduction to the Encoder-Decoder, or sequence-to-sequence models using the Transformer architecture. What is it, when should you use it? An example of a popular encoder-decoder model is the T5.



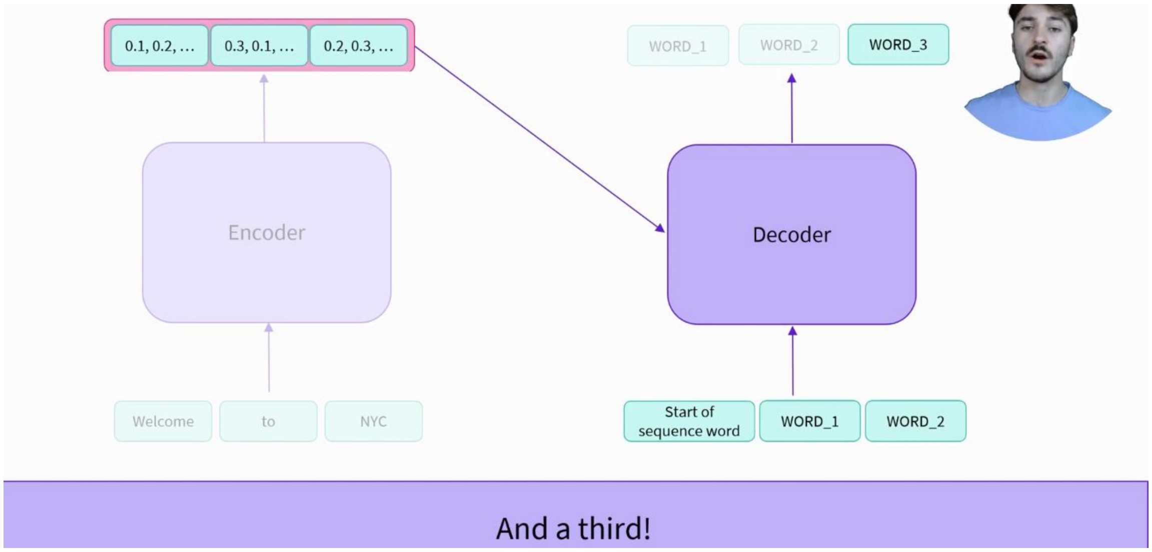


We are passing the output of the encoder as inputs to the decoder, we also give the decoder an additional sequence to indicate the start of a sequence.

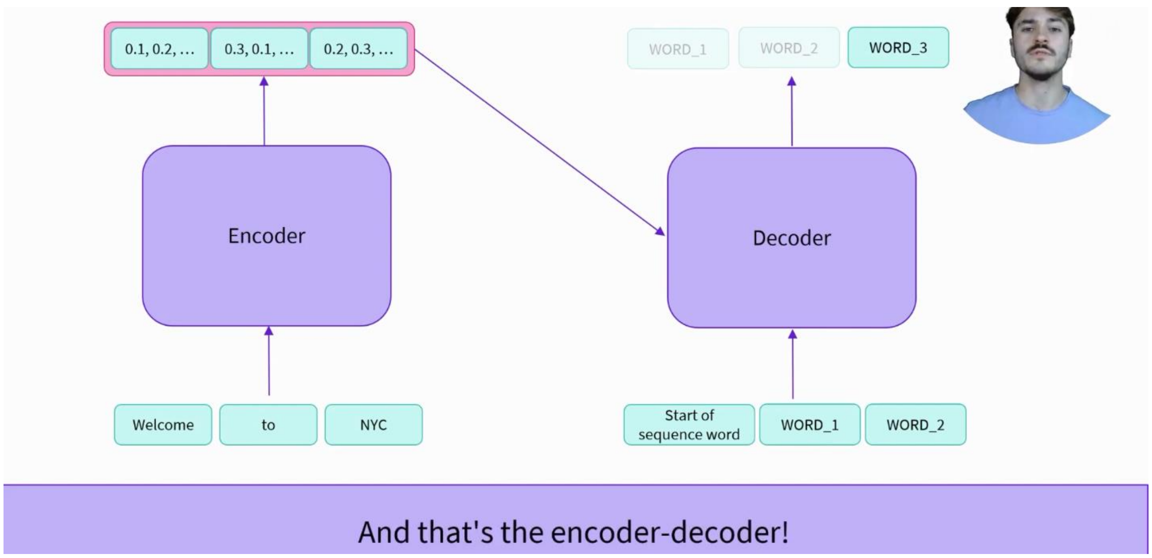


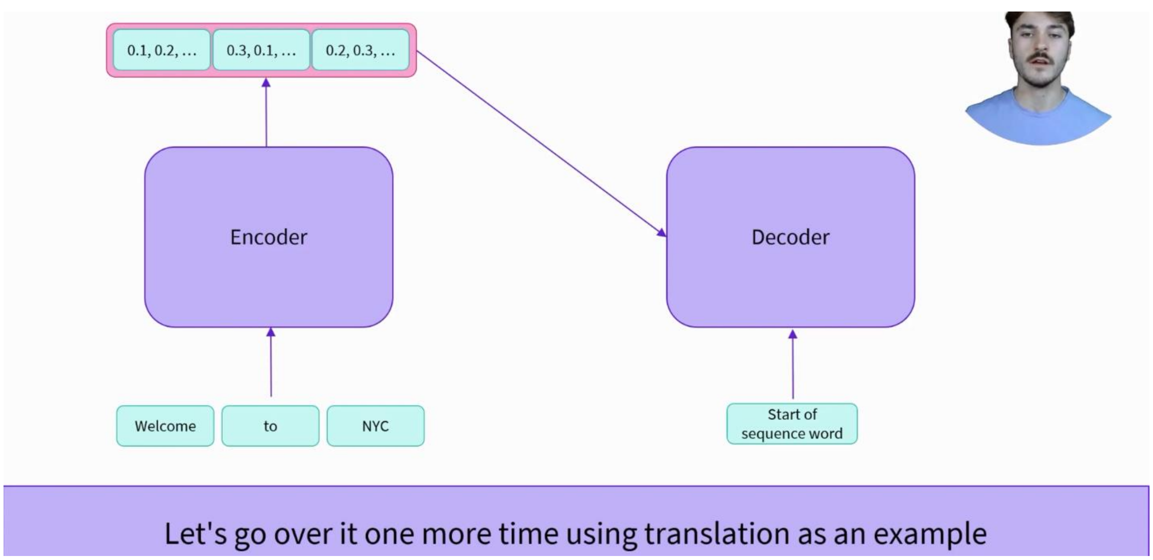


We don’t need the encoder anymore because the decoder can act in an auto-regressive manner to generate the next words in a sequence.

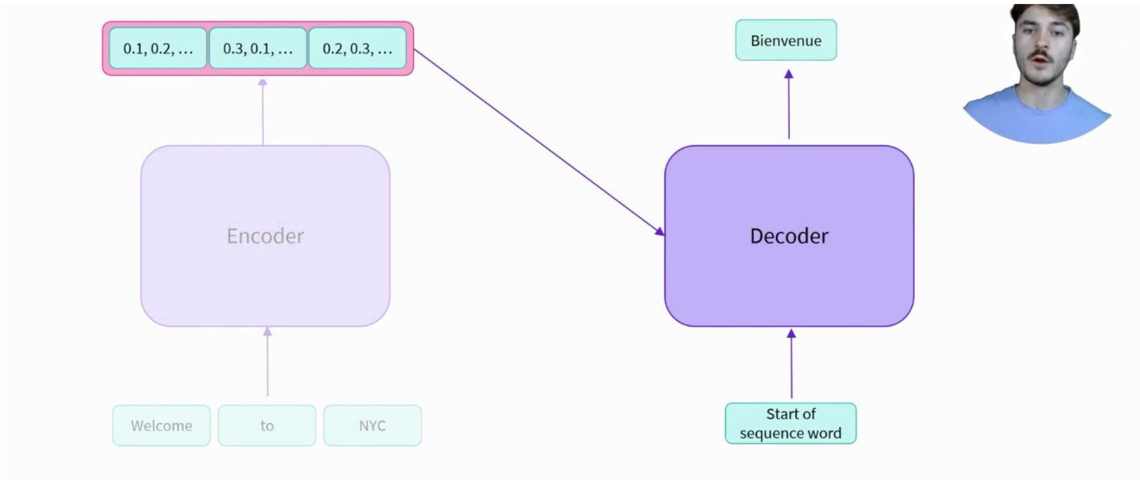


We can continue until the decoder outputs a value that we consider as a stopping value like a **full-stop**.

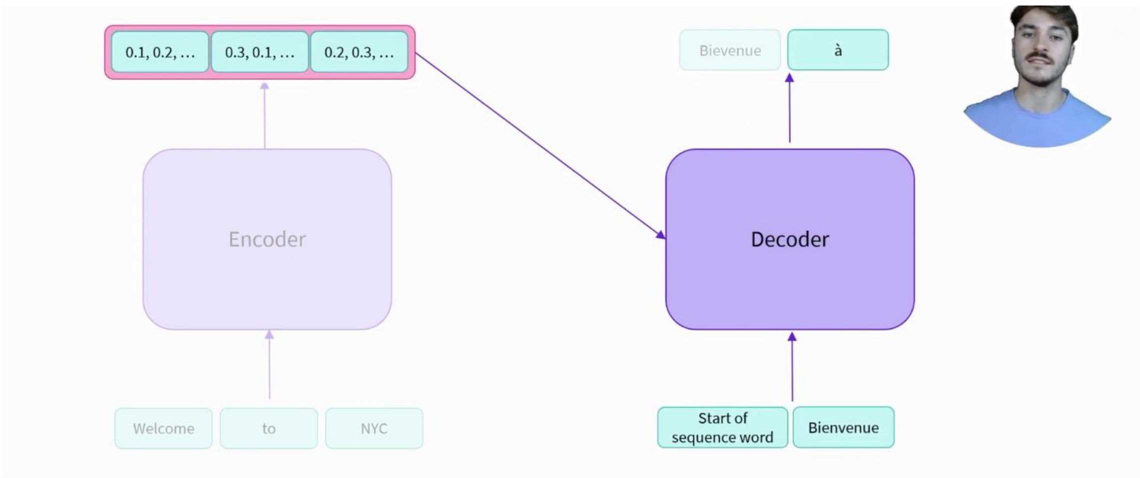




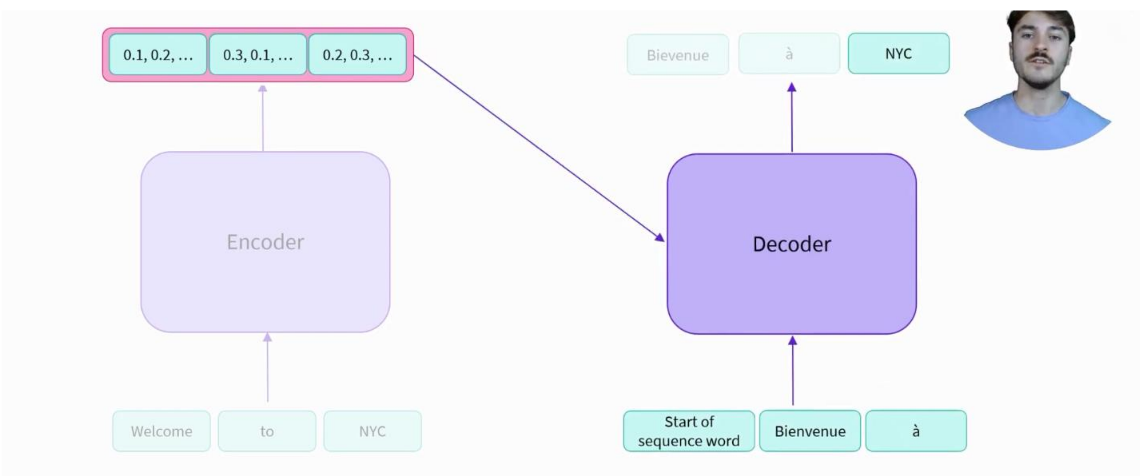
Let us take the example of translating a sequence from English to French. We will use a transformer model that is trained for that task explicitly. We first use the encoder to create the representation of the English sequence



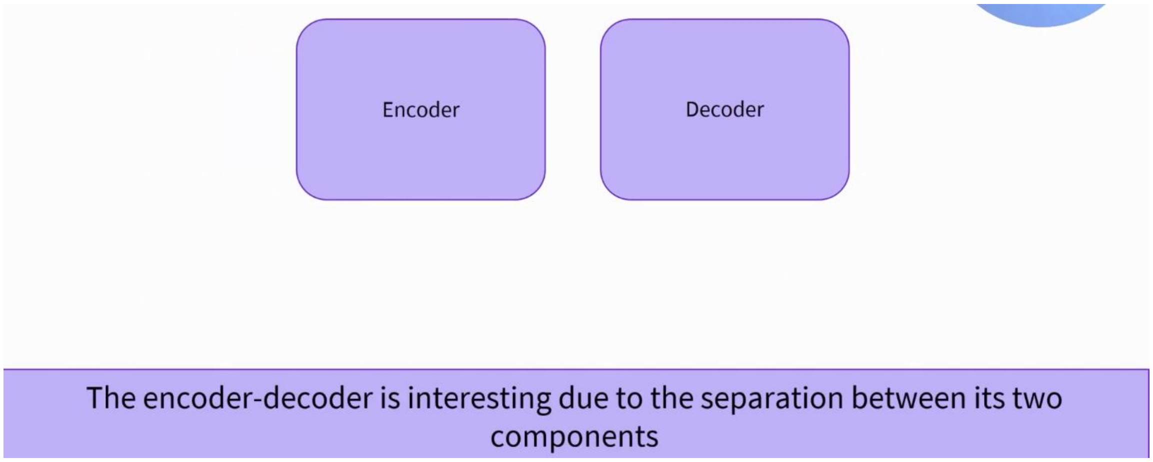
Given the representation of the English sentence "Welcome to NYC", the decoder generates the first word: the French word "Bienvenue"



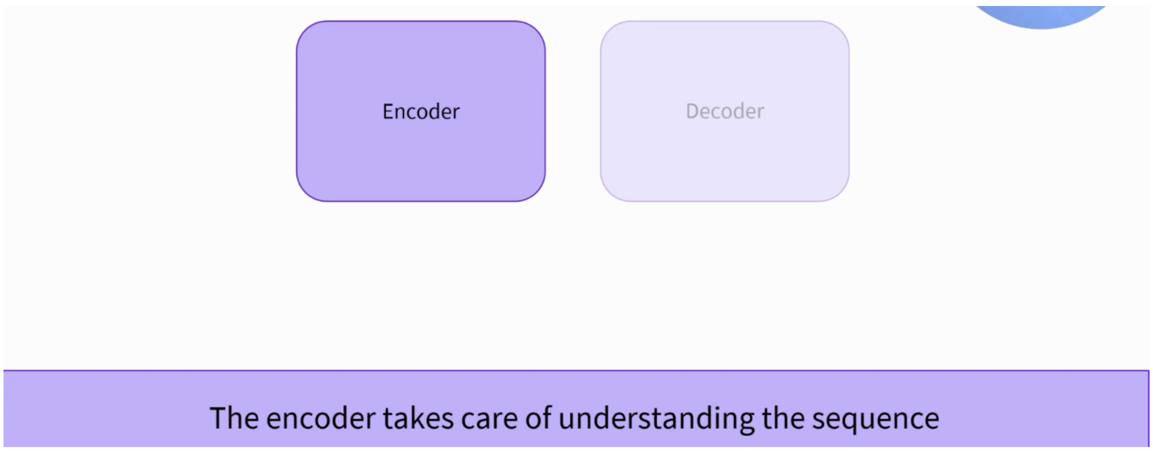
Using the representation and the first word, "Bienvenue", it is able to generate the second word: "à"



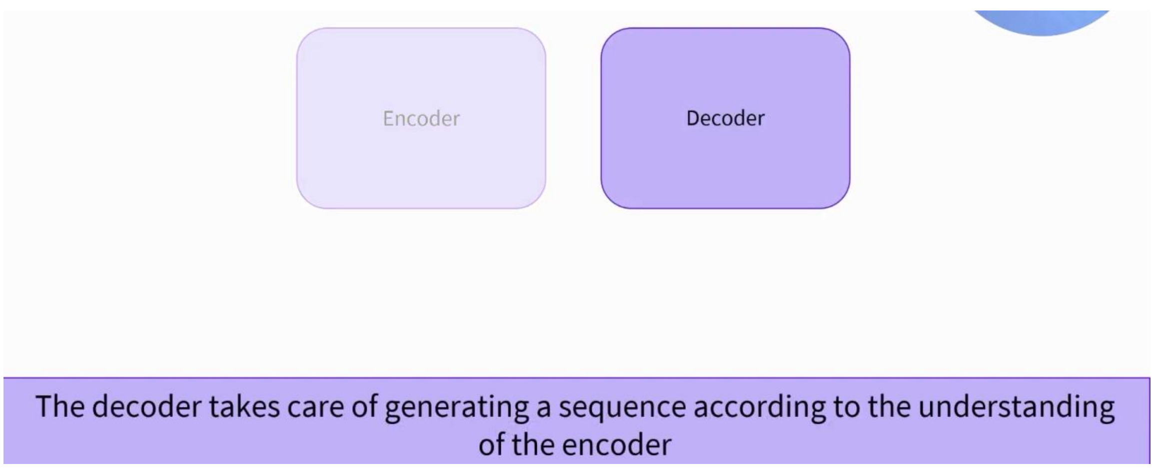
And, finally, leveraging both the encoder output and the two initial words, it can generate the third one: "NYC"



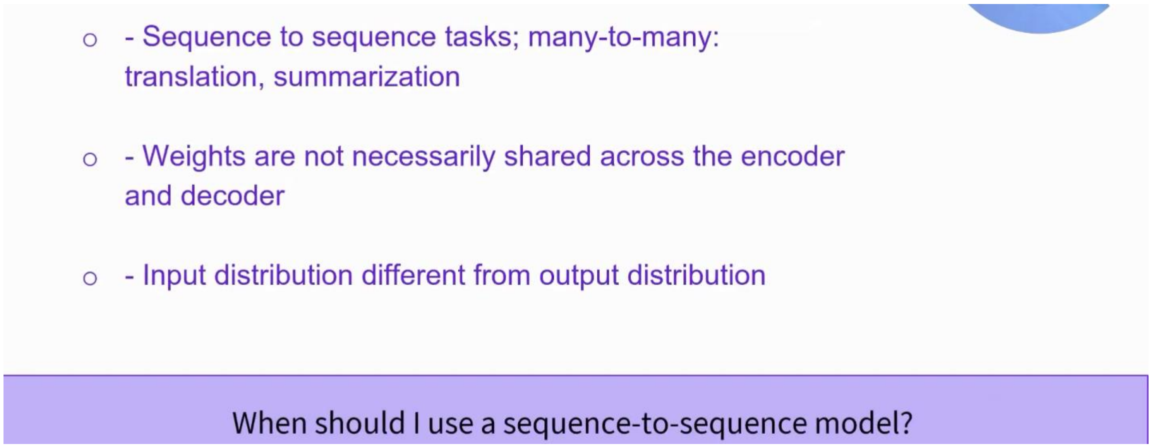
The encoder and decoder models often don’t share weights

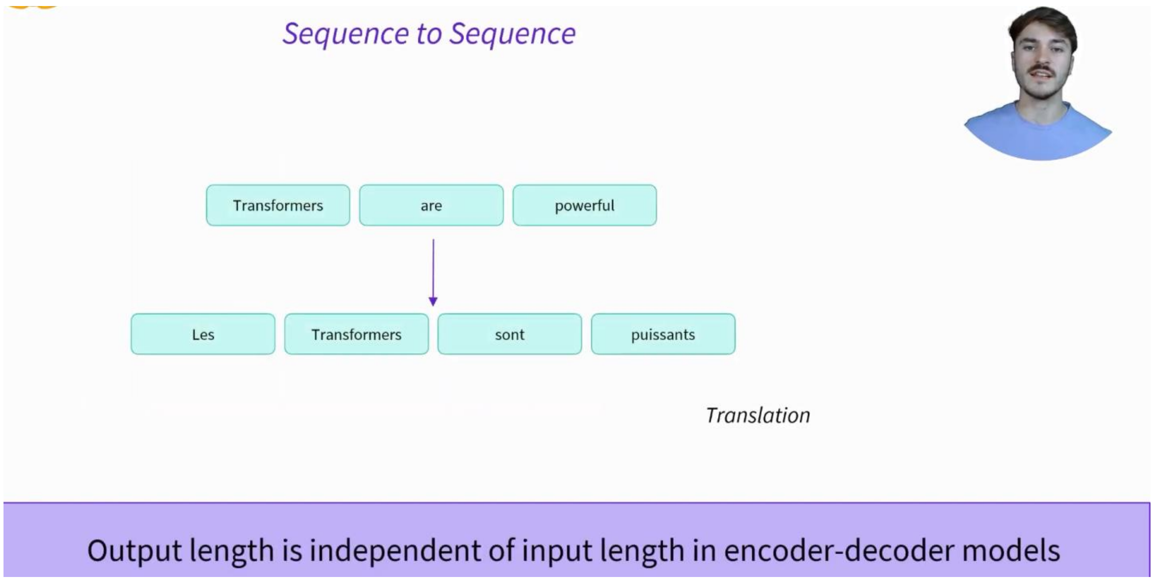


The encoder can be trained to understand the sequence and extract the relevant information by parsing into a vector

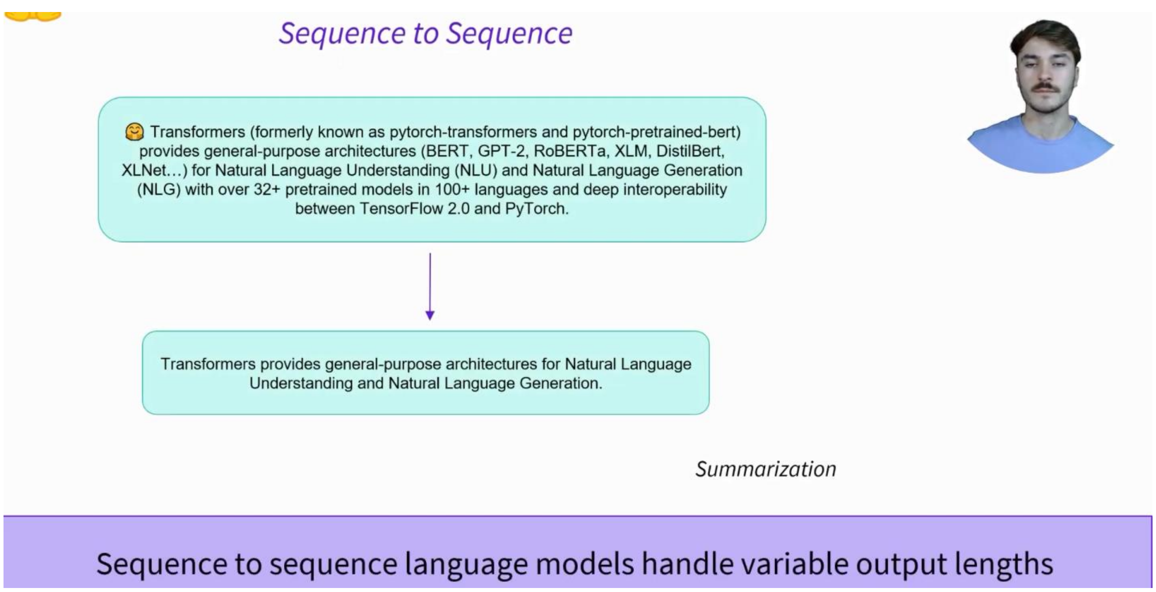


Trained to understand the encoder output.

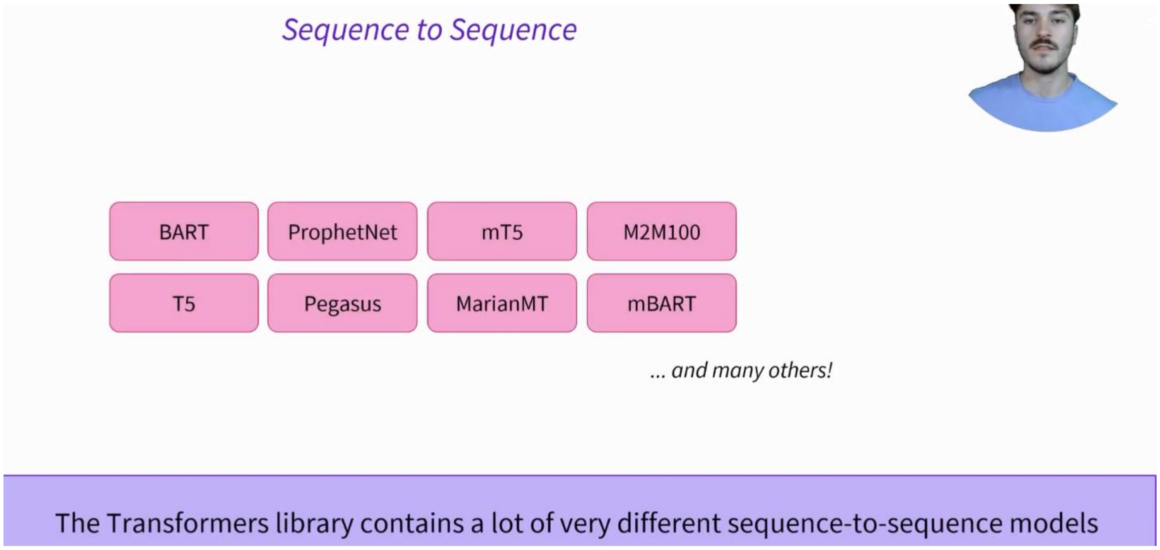




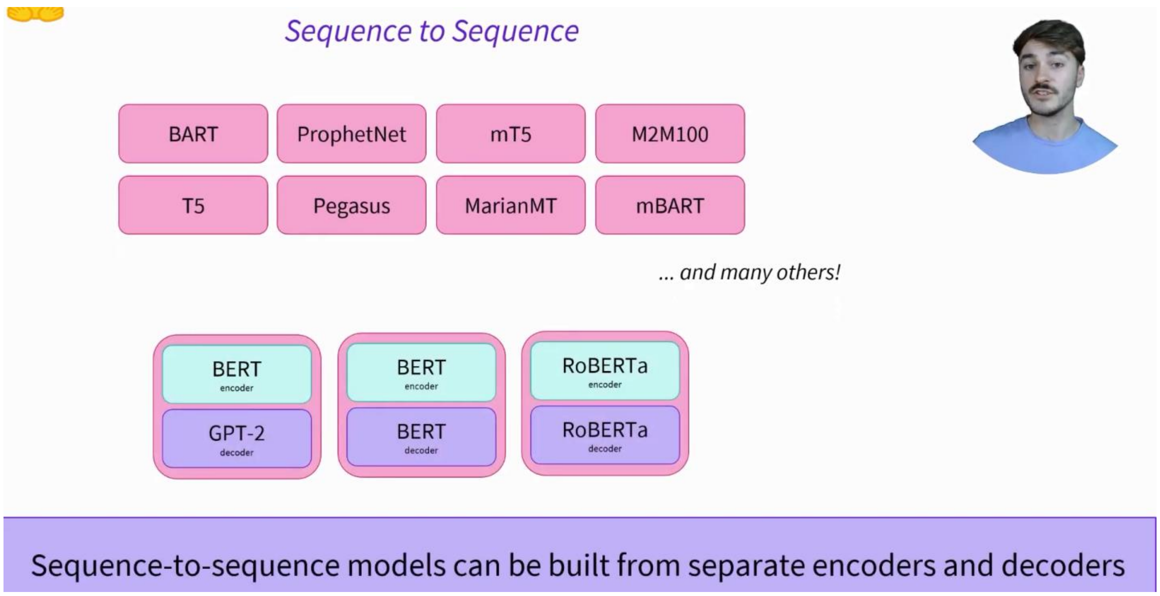
Let us see another example of translation where we are translating a sequence of 3 words to an output of 4 words. We can use an encoder and decoder in an autoregressive manner for this task.



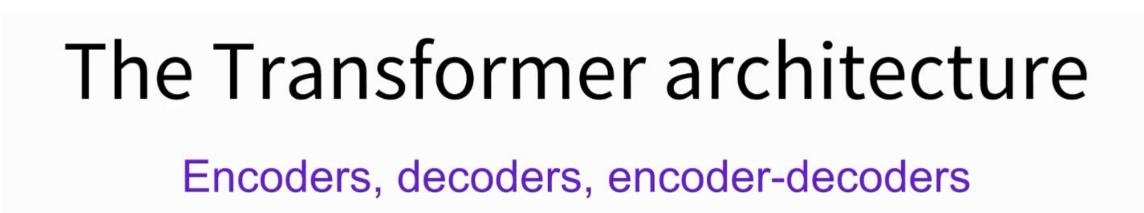
We can also do summarization with an encoder-decoder structure where the encoder and decoder have different context lengths.



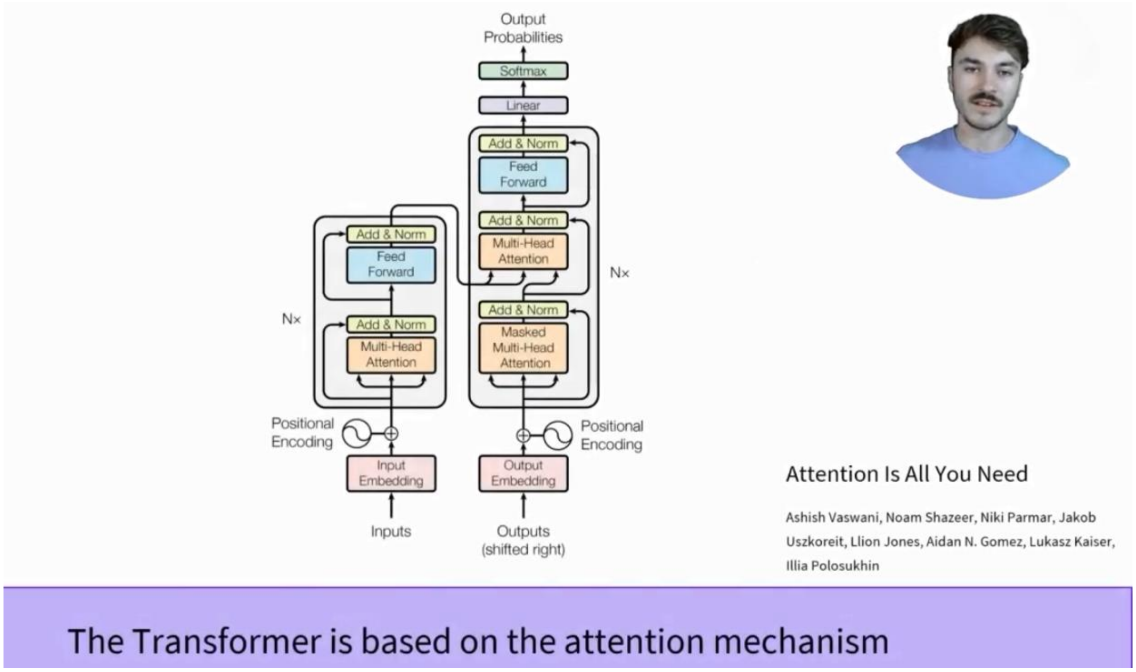
The are lots of encoder-decoder models

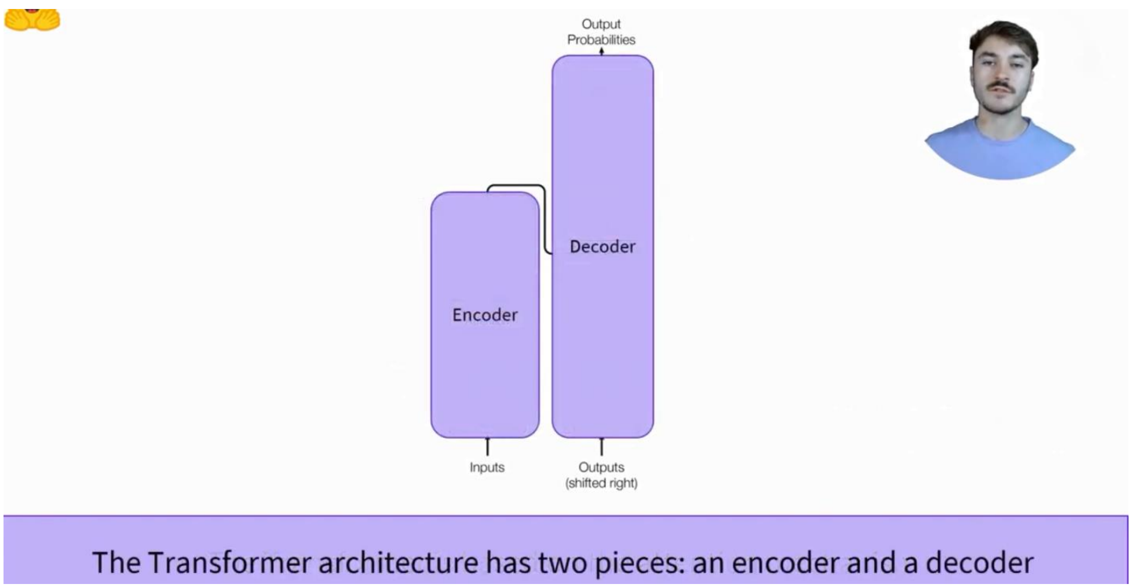


You can also load an encoder-decoder network inside an encoder-decoder model depending on the task you have.

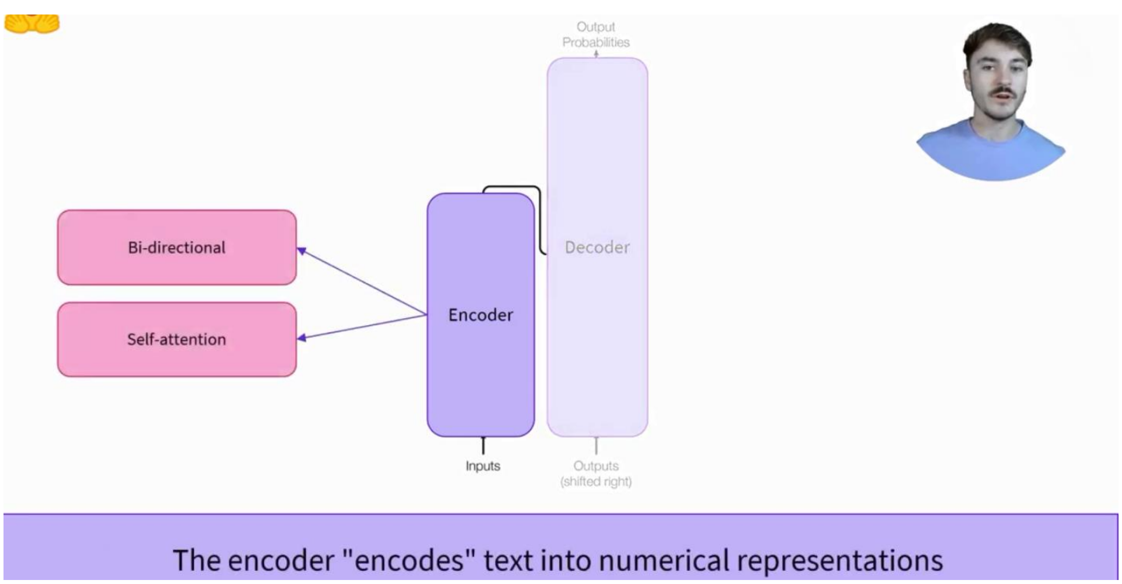


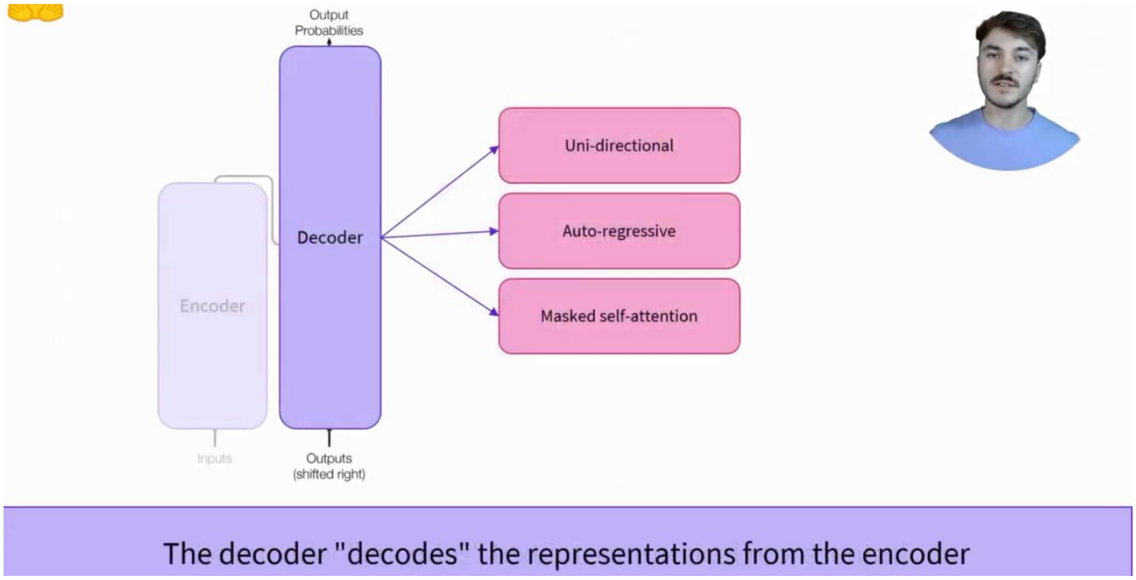
What makes a transformer network?

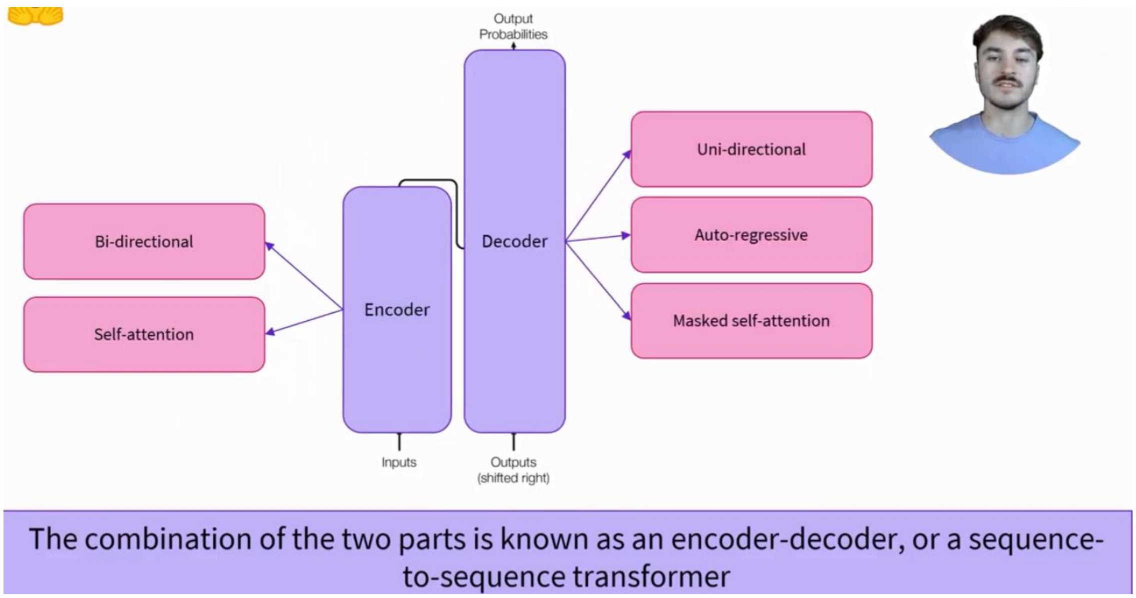




These 2 can be used together or used independently.







The encoder accepts text inputs and computes a high-level representation of those inputs as outputs that are then passed to the decoder. The decoder uses the encoder outputs and other inputs to generate a prediction as output to be re-used in future iterations, hence auto-regressive.