Transformer Models & LLMs

CS XXX: Introduction to Large Language Models

Contents

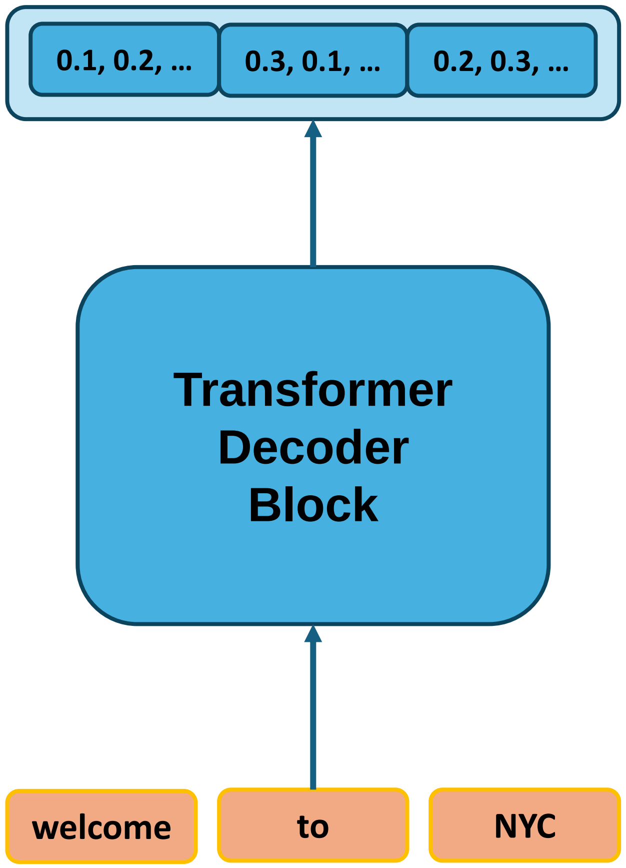
- Types of transformer models
- Decoders
- Encoders
- Encoder-Decoder
- Large Language Models
- The Training Paradigm of Large Language Models

Three types of transformer Models

- Decoder Only (Masked self-attention in transformer block e.g. GPT)
  - Stacked transformer blocks use **masked** self-attention (transformer decoder blocks)
  - Used for text generation (e.g. creative writing), and next word prediction
  - Also referred to as generative models
- Encoder Only (e.g. BERT)
  - Stacked transformer blocks use **unmasked** self-attention (transformer encoder blocks)
  - Used for classification and representation learning
  - Also referred to as representation models
- Encoder-Decoder (e.g. T5, BART)
  - Encoder used for representation learning
  - Decoder used for generation
  - Used for machine translation, summarization
  - Also referred to as sequence-to-sequence models

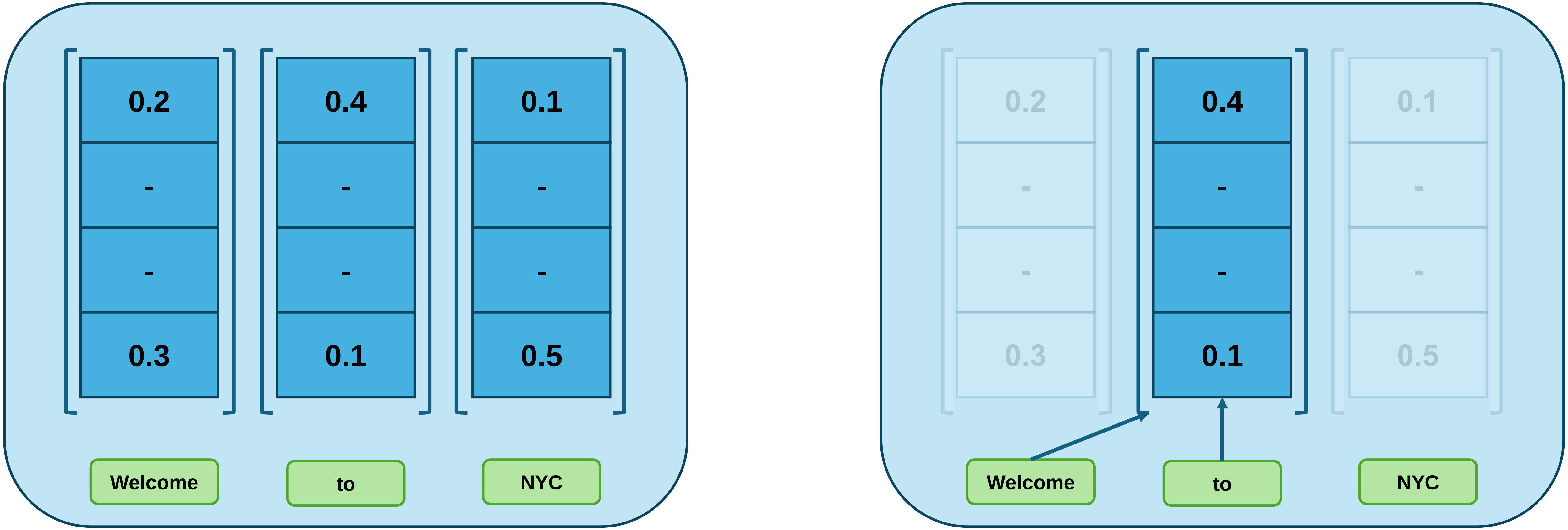
Decoders

- Create feature tensors from an initial sequence. These feature tensors are the numerical representations of the initial sequence



Decoders

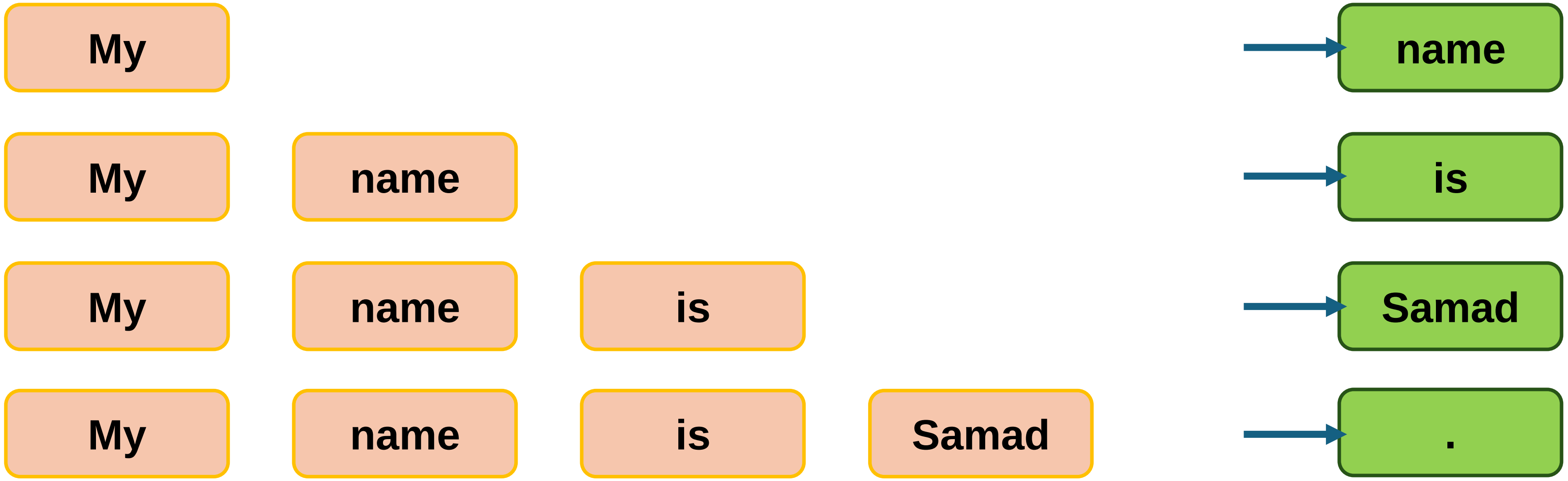
- The feature tensor is made up of a vector of values for each word of the initial sequence



Masked Self Attention: Words can only see the words on their left side; the right side is hidden

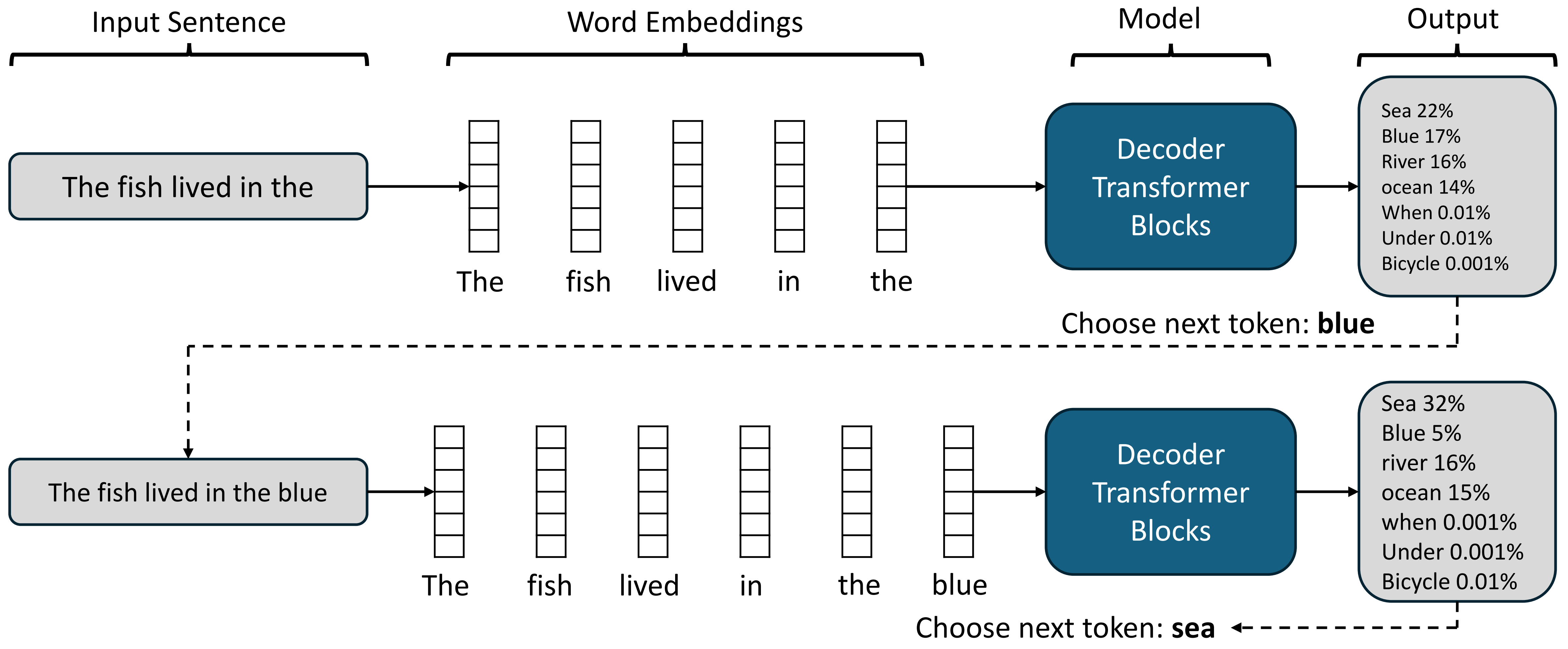
Decoders

- Decoders, with their uni-directional context, are good at generating words given a context



Decoders

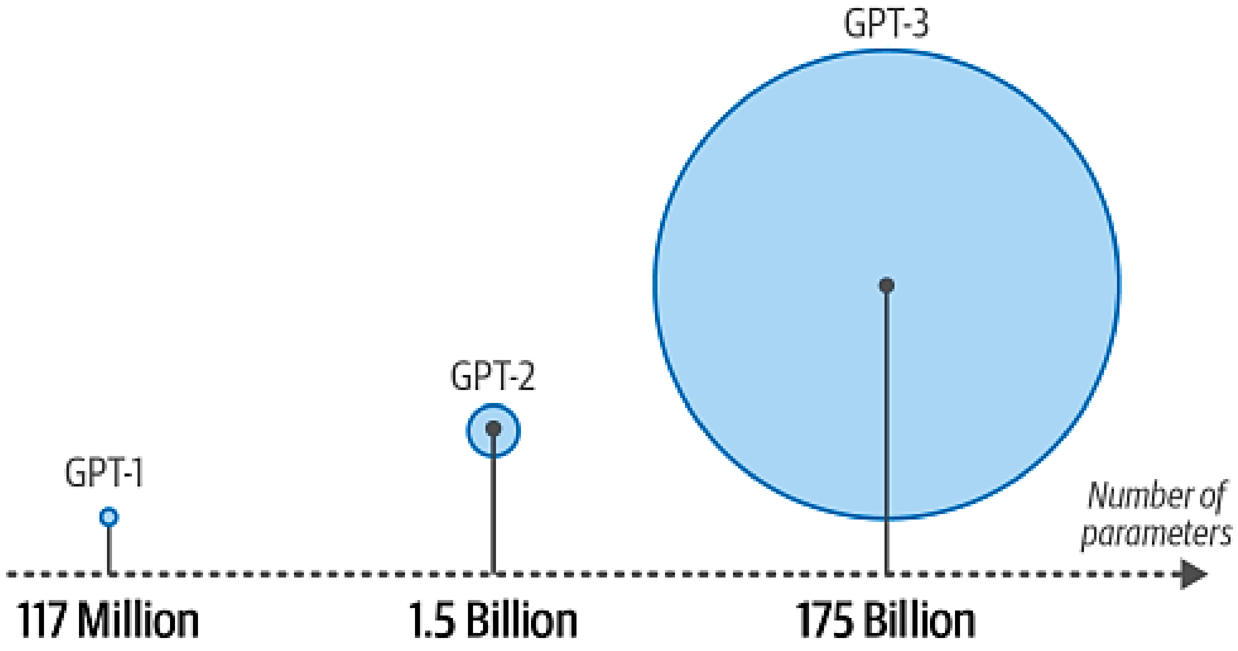
- **Autoregressive generation**: sequential token by token prediction



Decoders

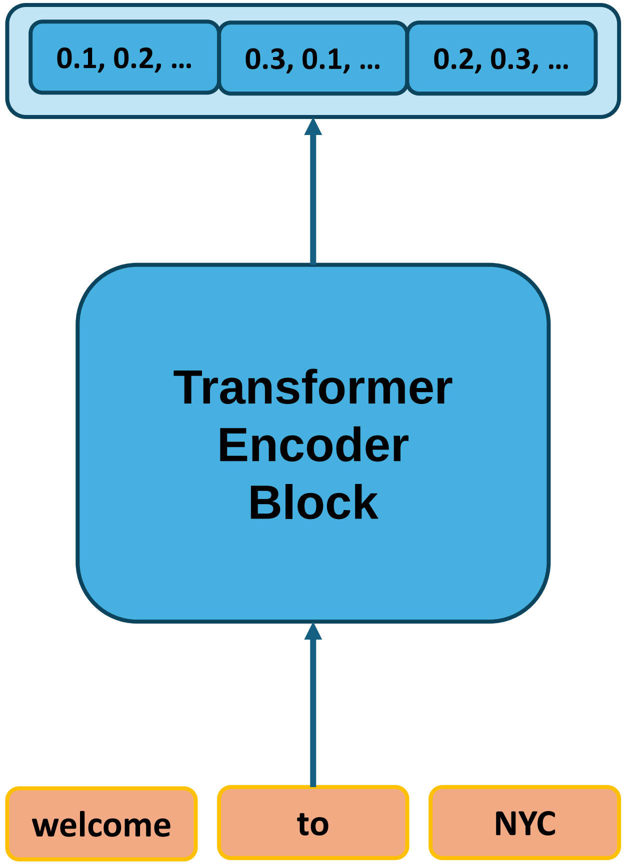
- GPT-3:

  96 stacked transformer (decoder) blocks



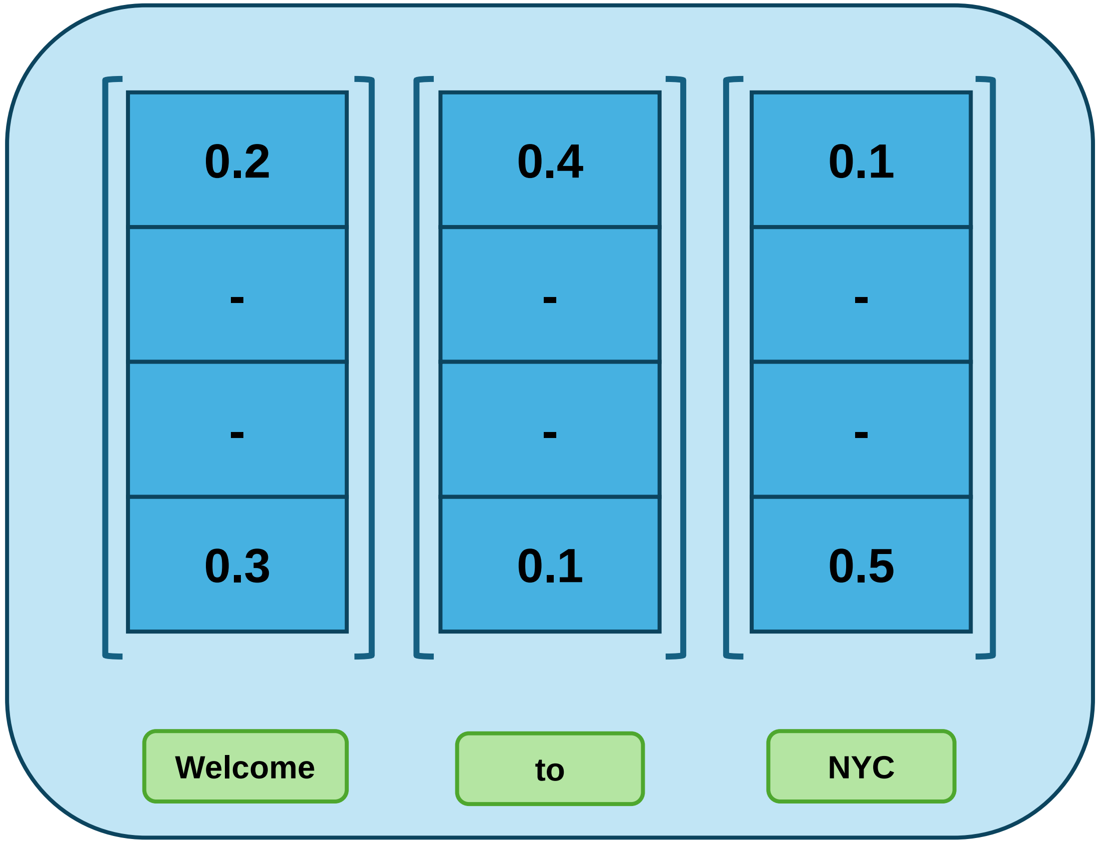
Encoders

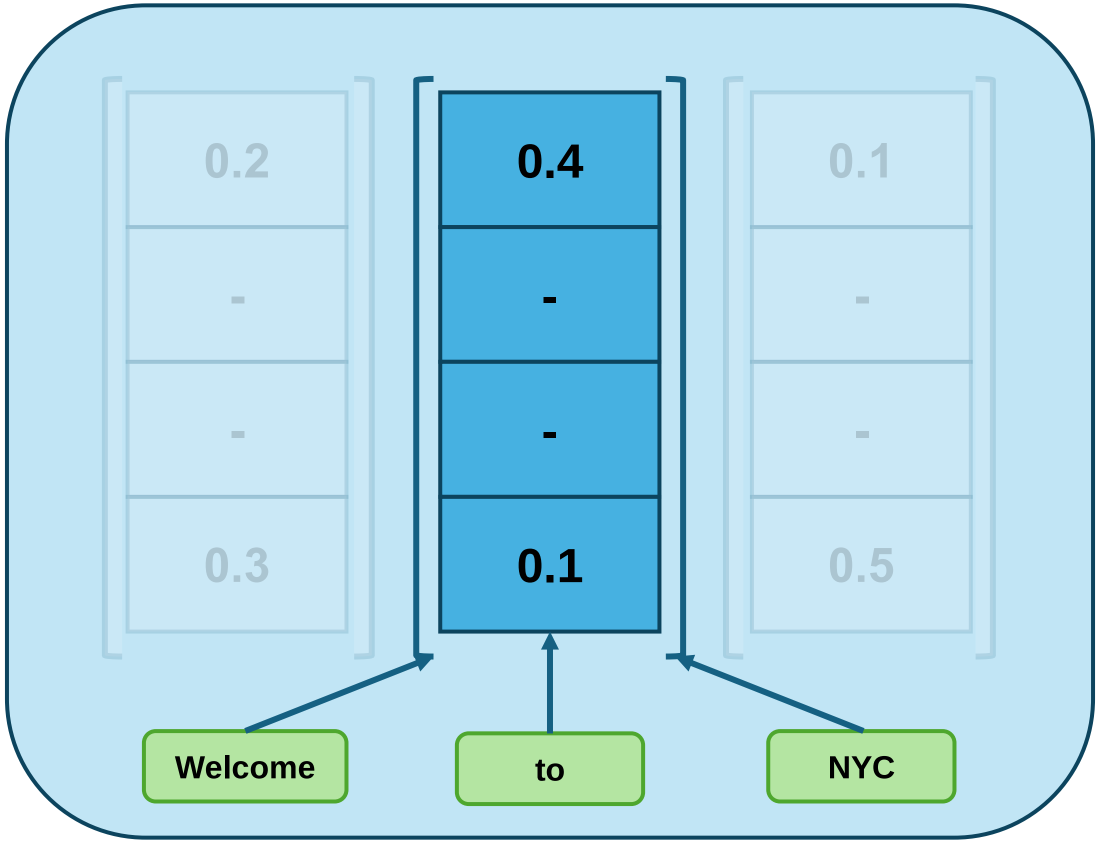
- Like the decoder, the encoder outputs a numerical representation for each token used in the input sequence



Encoders

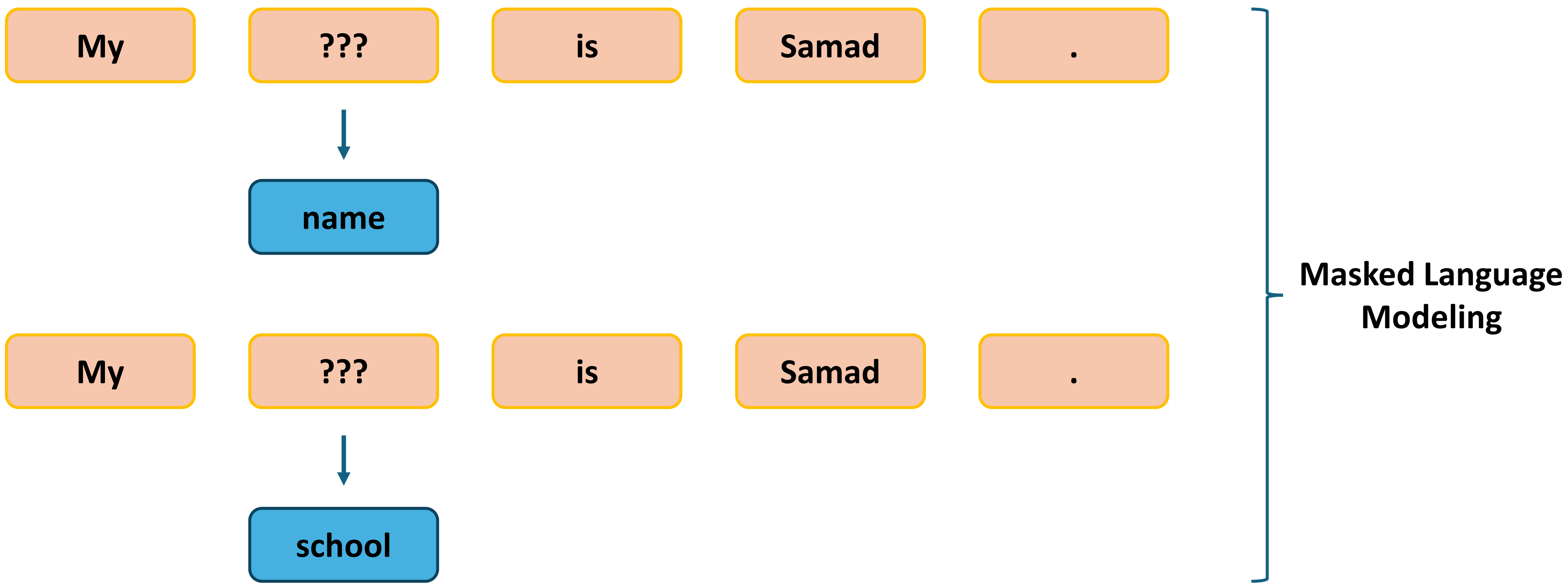
- Unlike the decoder, future values are **not** masked and each word can see words on its left and right side.





Encoders

- Encoders with their bi-directional context, are good at guessing words in the middle of a sequence



Encoders

- Encoders are good at obtaining an understanding of sequences; and the relationship/interdependence between words

Even though I am sad to see them go, I couldn't be more grateful

Positive

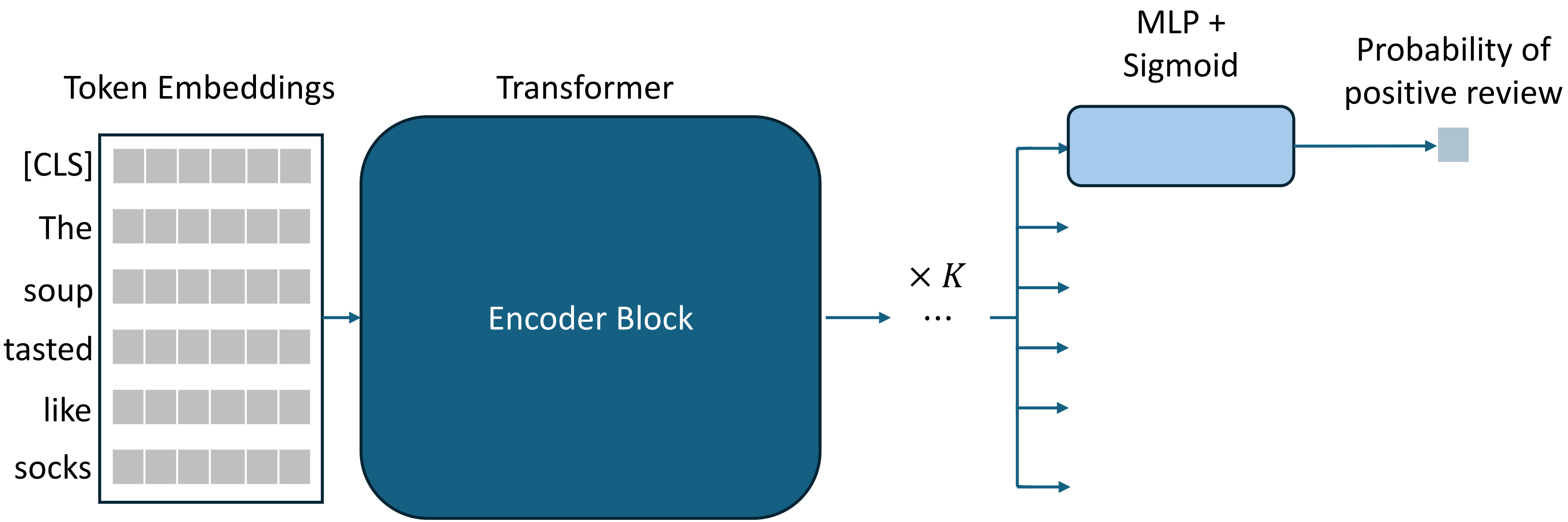
I am sad to see them go, I can't be grateful

Negative

Sentiment Analysis

Encoders

- Sentiment Analysis



Encoders

- BERT (base):

    12 stacked transformer (encoder) blocks

    110 Million Parameters

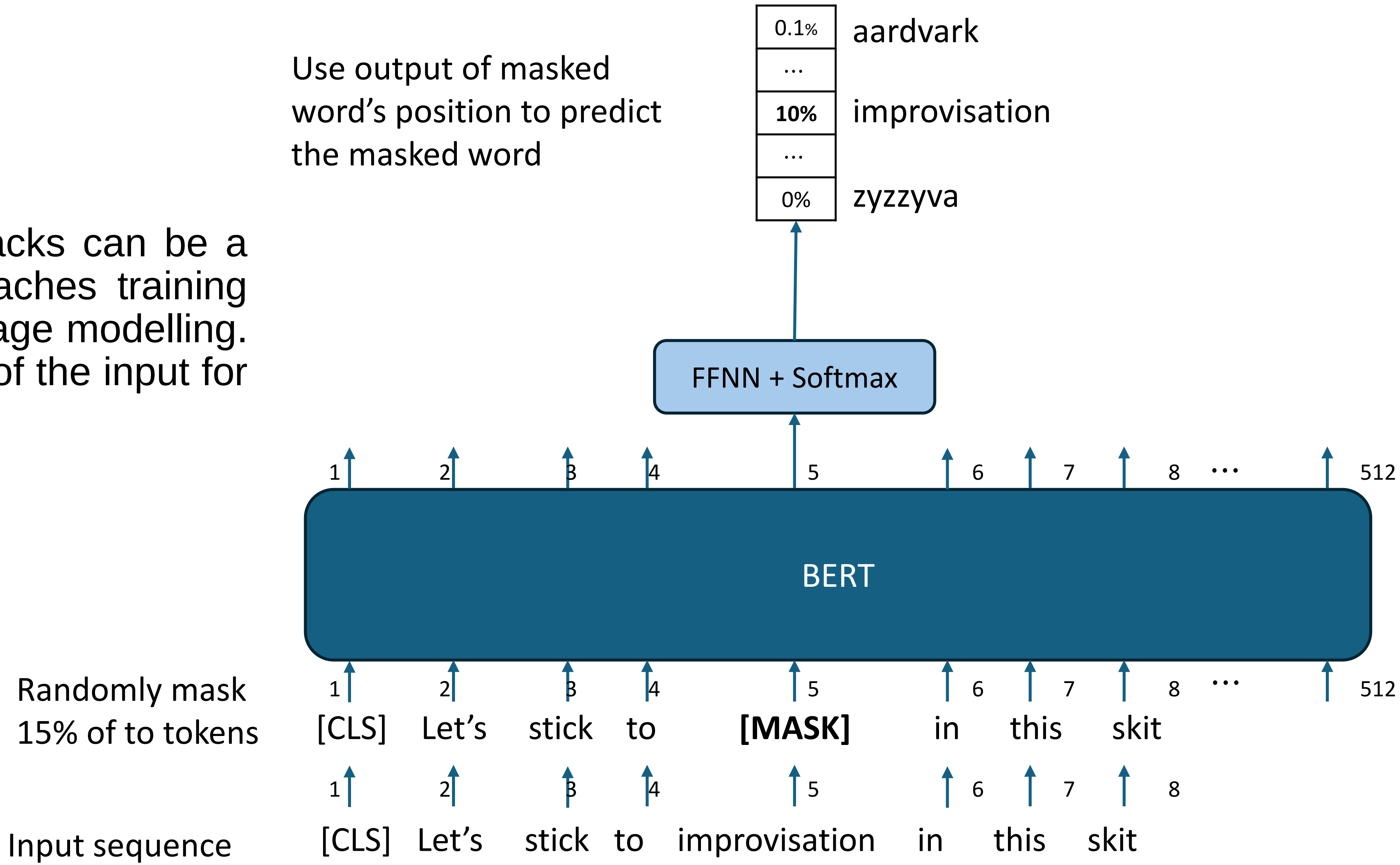
- BERT (large):

    24 stacked transformer (encoder) blocks

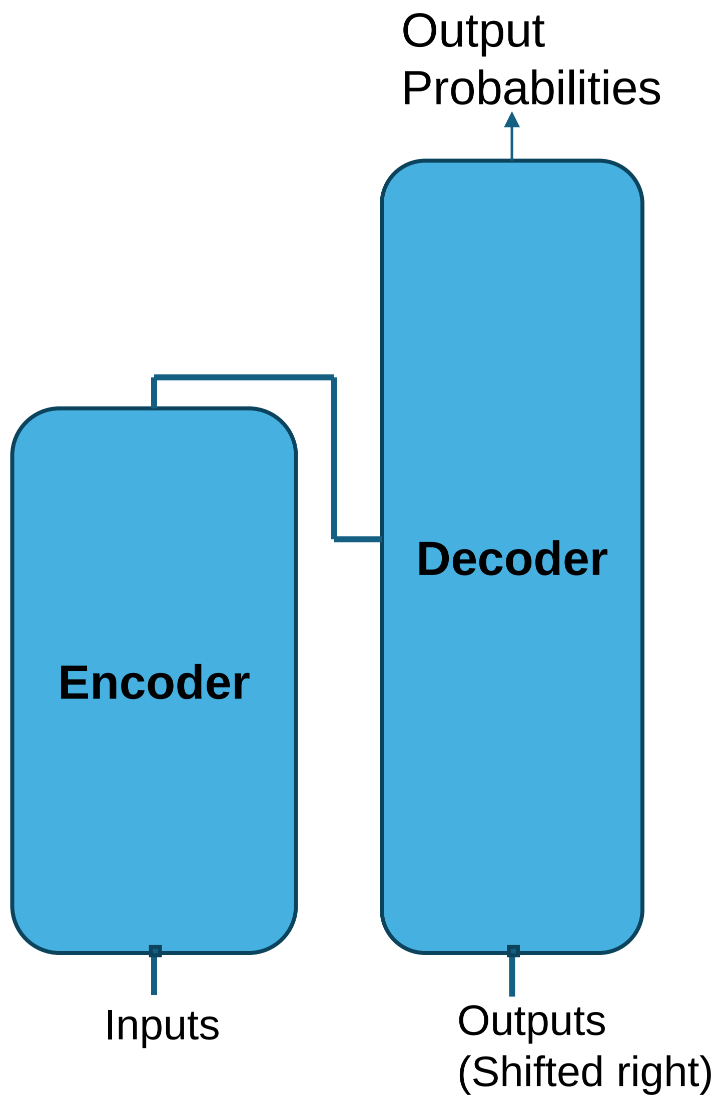
    340 million parameters

Encoders

Training these encoder stacks can be a difficult task. BERT approaches training by adopting masked language modelling. This method masks a part of the input for the model to predict.

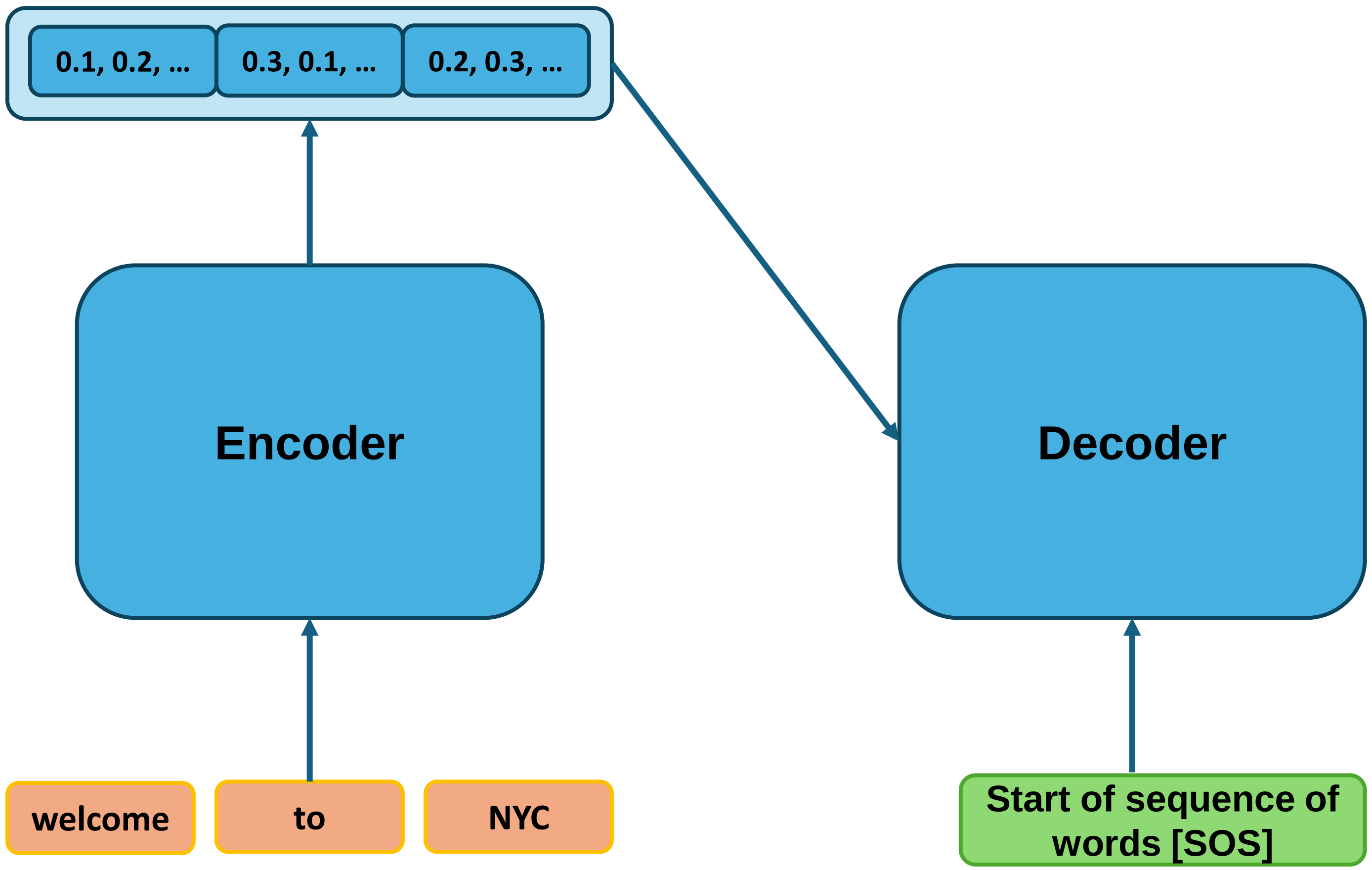


Encoder-Decoder



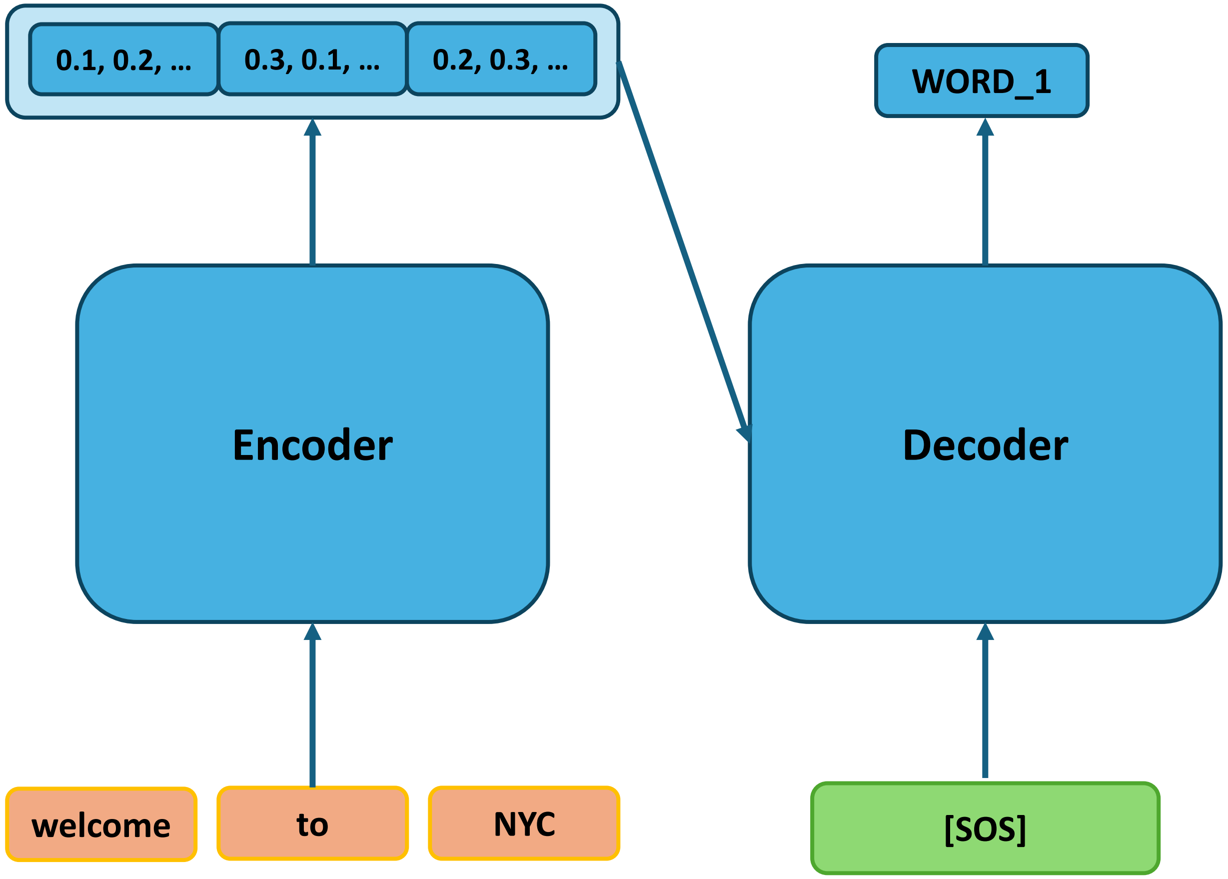
Encoder-Decoder

- Encoder output representations are now used as input for the decoder



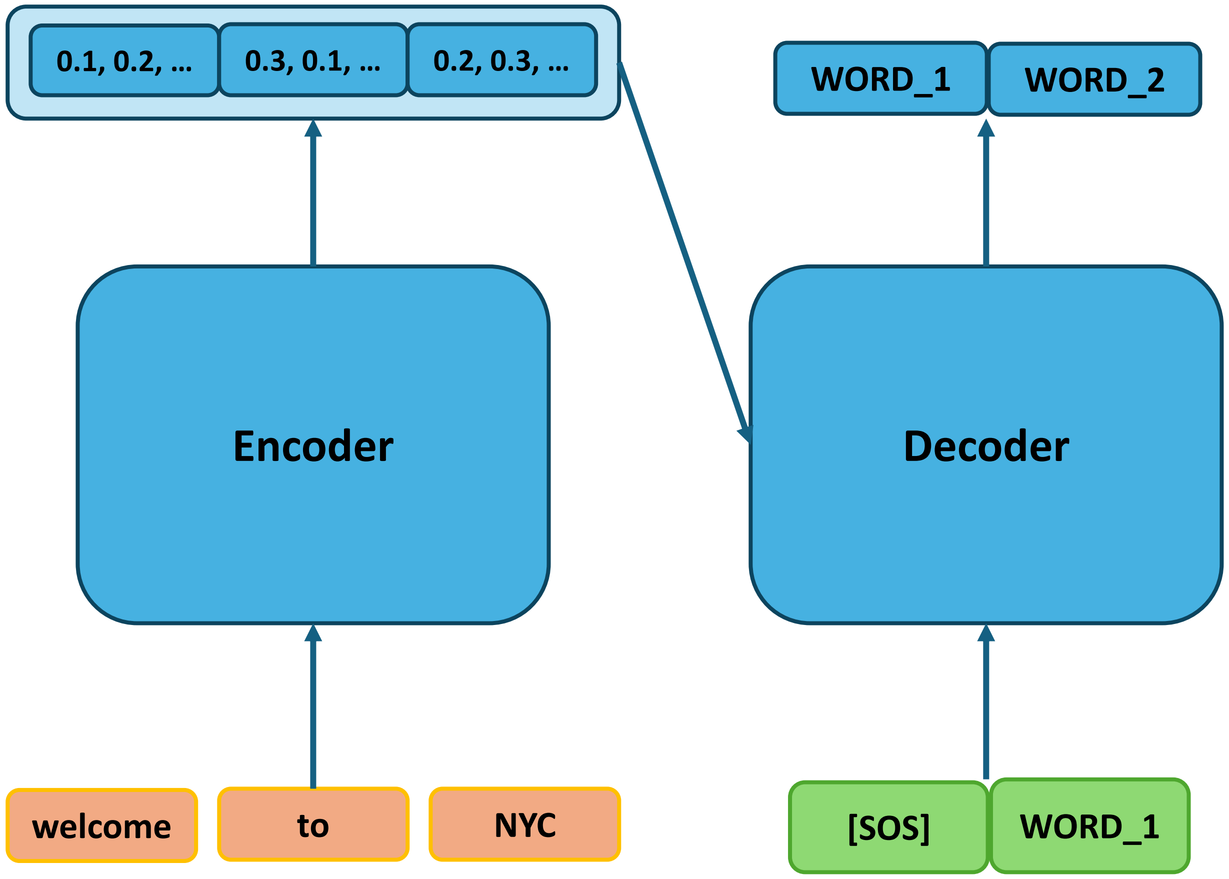
Encoder-Decoder

- Using this representation and a prompt as input, the decoder generates one word after another



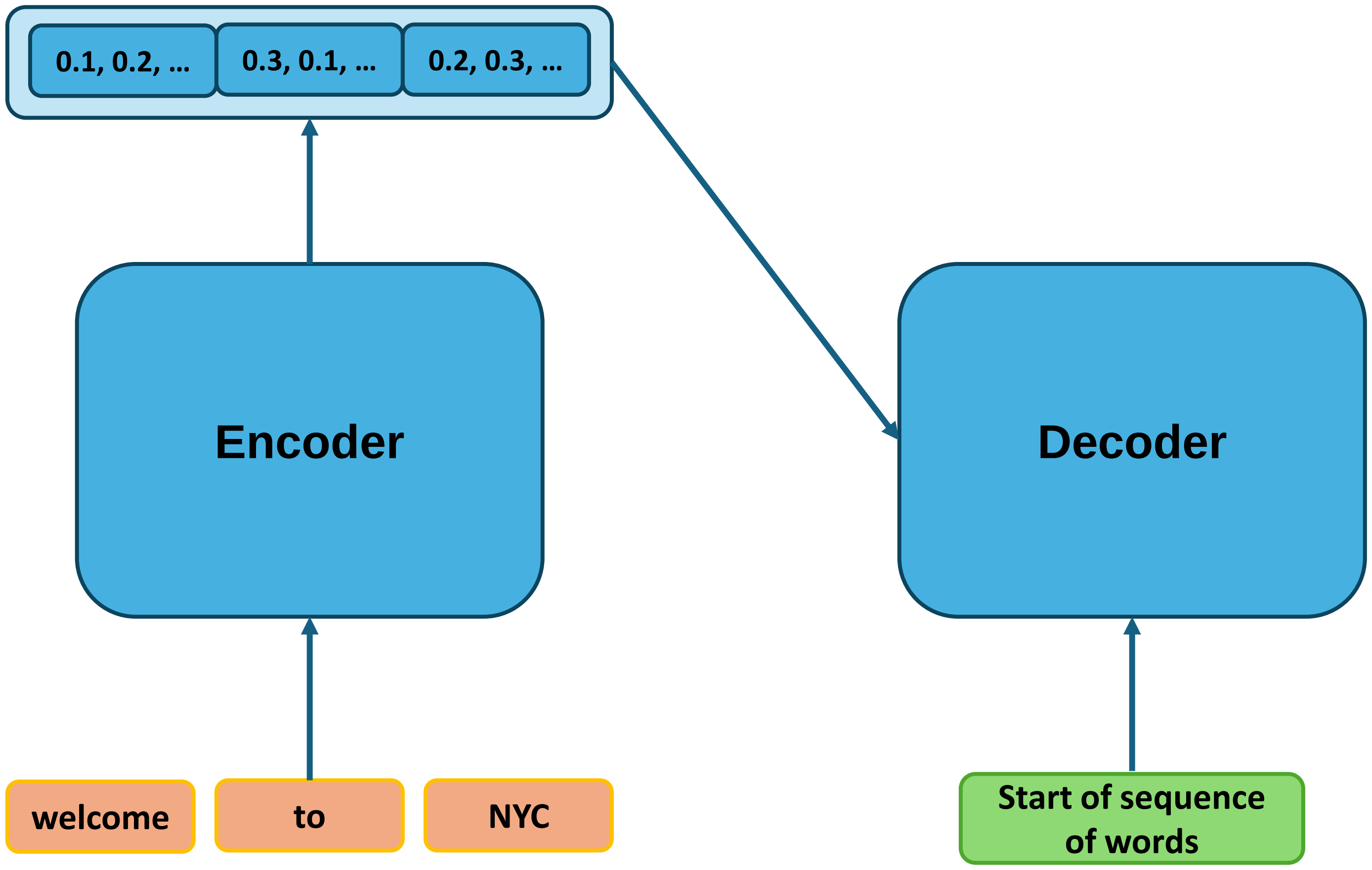
Encoder-Decoder

- Using this representation and a prompt as input, the decoder generates one word after another



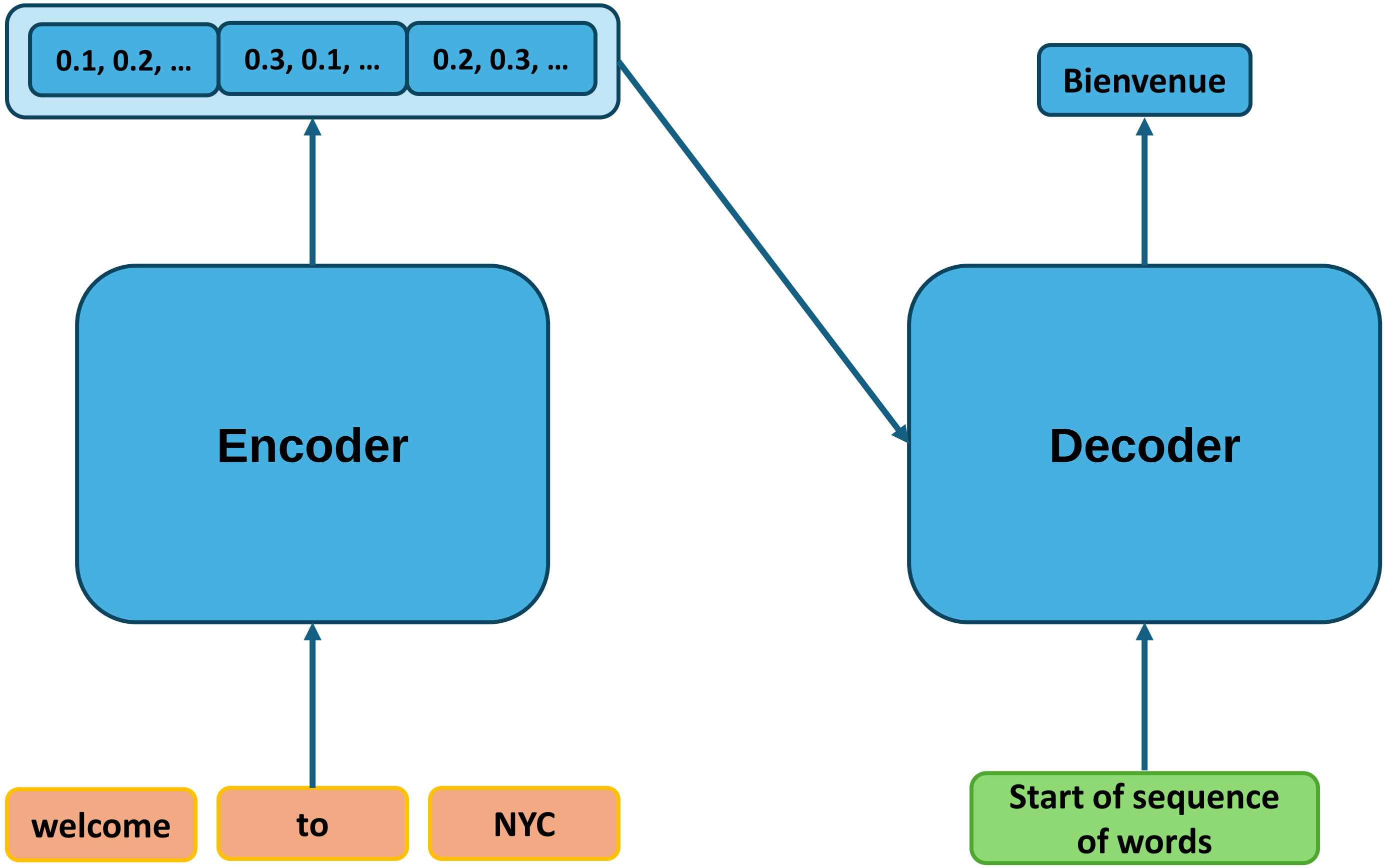
Encoder-Decoder

- Example: Machine Translation



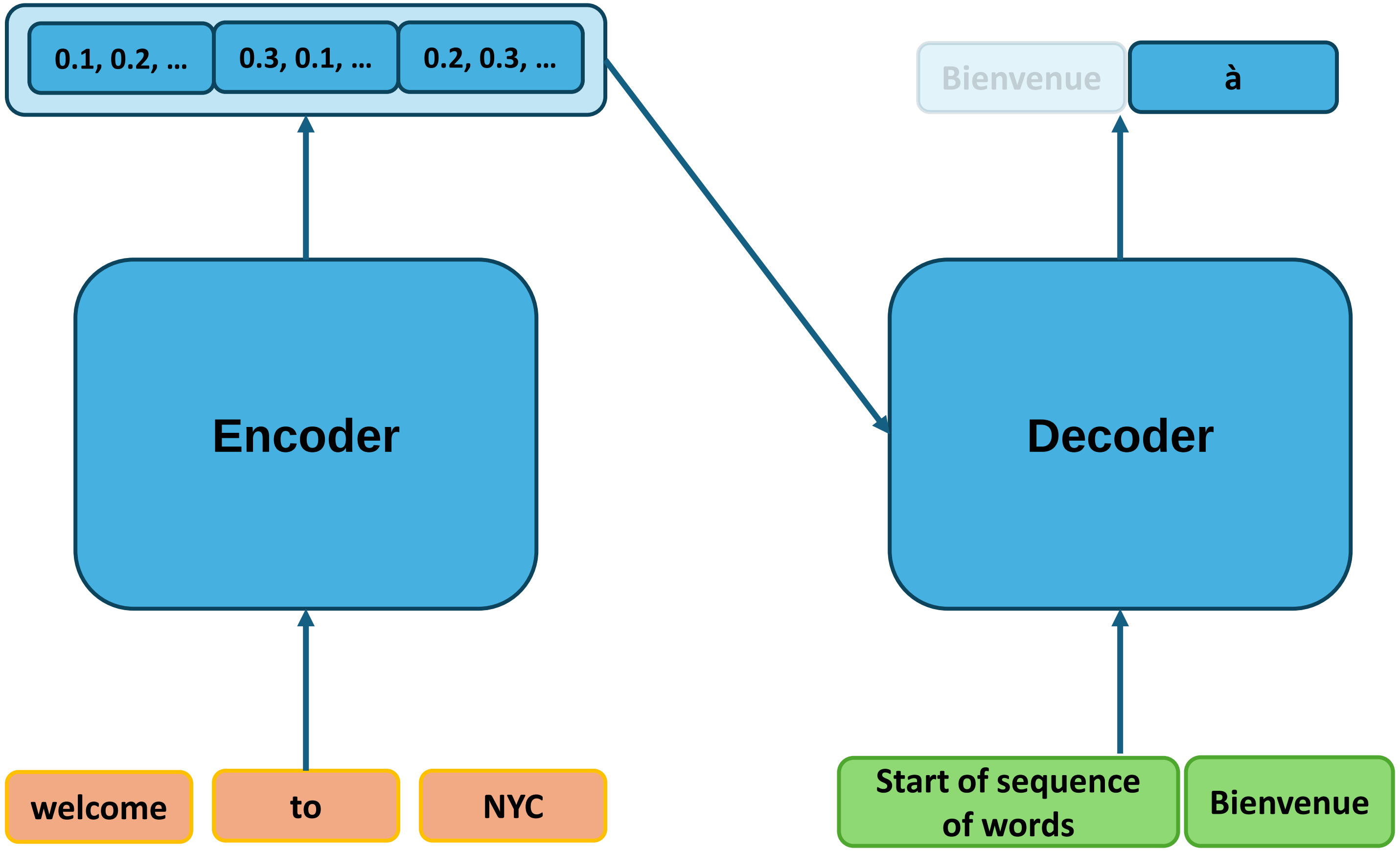
Encoder-Decoder

- Example: Machine Translation



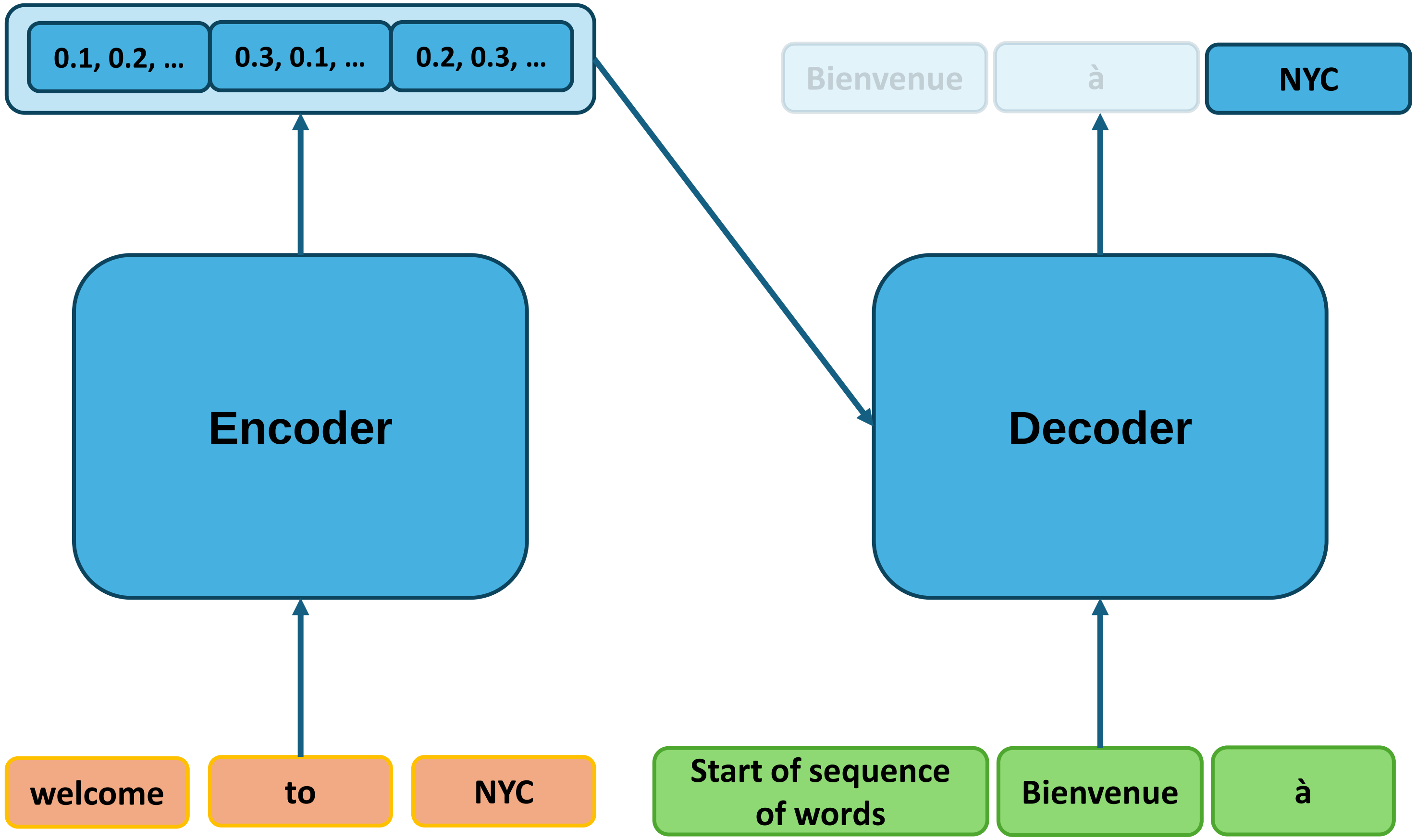
Encoder-Decoder

- Example: Machine Translation



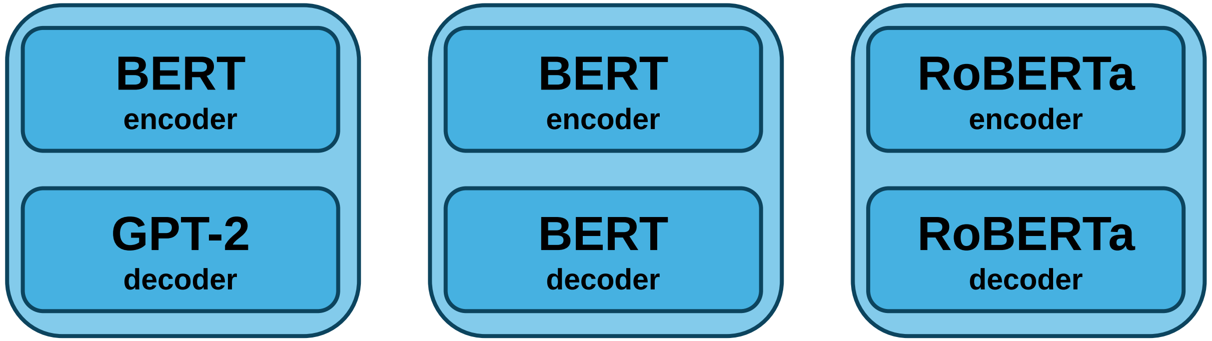
Encoder-Decoder

- Example: Machine Translation



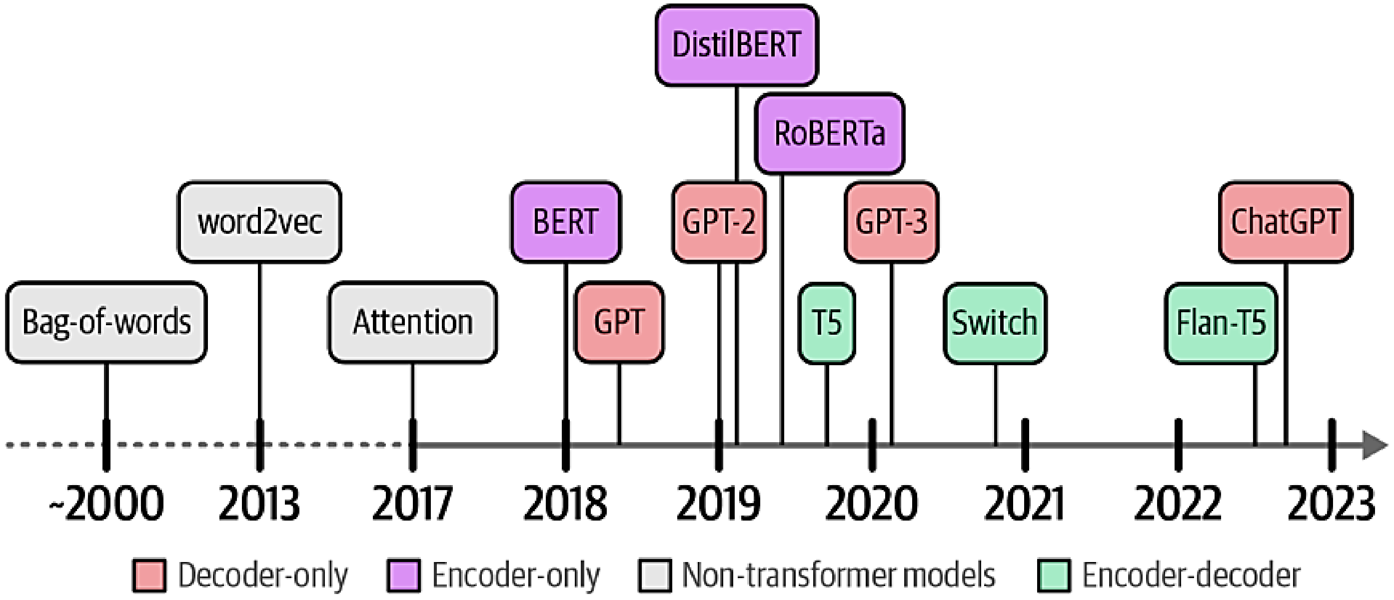
Encoder-Decoder

- Sequence-to-Sequence models can be built from separate encoders and decoders



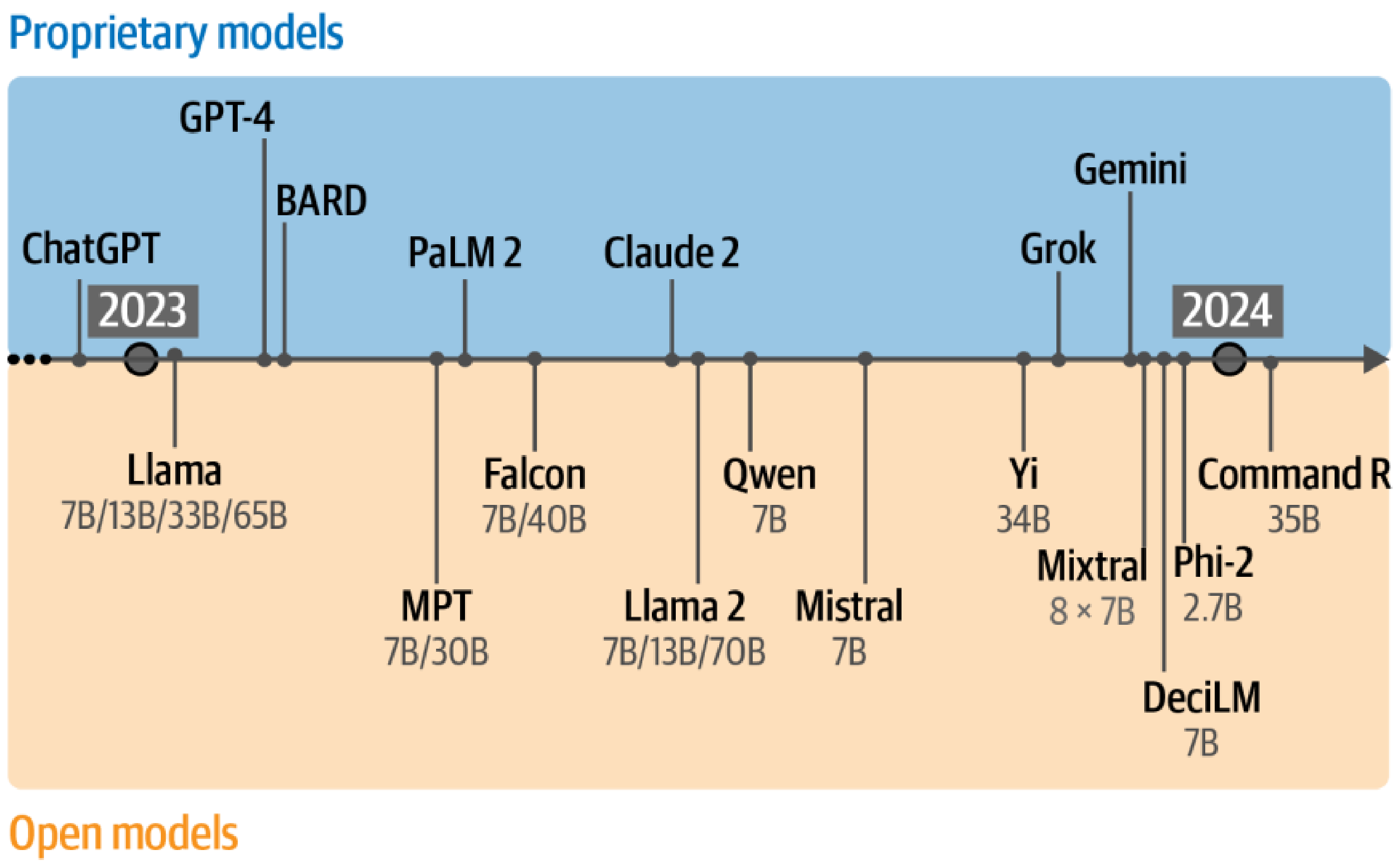
Large Language Models

- Through the recent history of language AI, the term **large language models** is used for these Encoder, Decoder, & Encoder-Decoder Models particularly if they are considered “large” based on the number of parameters
- The history of language AI encompasses many development and models aiming to represent and generate language



Large Language Models

- Recent LLM releases

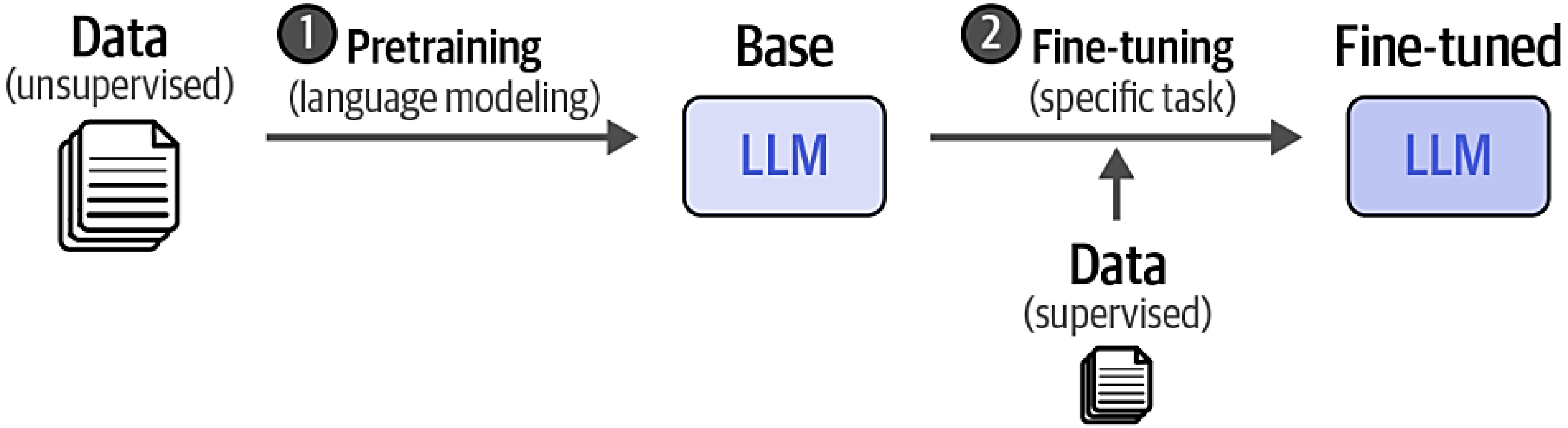


The training paradigm of large language models

- Creating LLMs typically consists of at least two steps:
  - The first step, called **pretraining**,
    - takes the majority of computation and training time.
    - An LLM is trained on a vast corpus of internet text allowing the model to learn grammar, context, and language patterns.
    - This broad training phase is not yet directed toward specific tasks or applications beyond predicting the next word.
    - The resulting model is often referred to as a foundation model or base model. These models generally do not follow instructions.
  - The second step, **fine-tuning** or sometimes post-training
    - involves using the previously trained model and further training it on a narrower (downstream) task. This allows the LLM to adapt to specific tasks or to exhibit desired behavior.
    - It saves massive amounts of resources because the pretraining phase is quite costly and generally requires data and computing resources that are out of the reach of most people and organizations.

The training paradigm of large language models

- Pretraining & Finetuning



References

- Jurafsky, D., & Martin, J. H. Speech and Language Processing. Stanford University
- Alammar, J., & Grootendorst, M. Hands-On Large Language Models: Language Understanding and Generation. O'Reilly Media.
- Hugging Face. The Hugging Face Natural Language Processing Course.