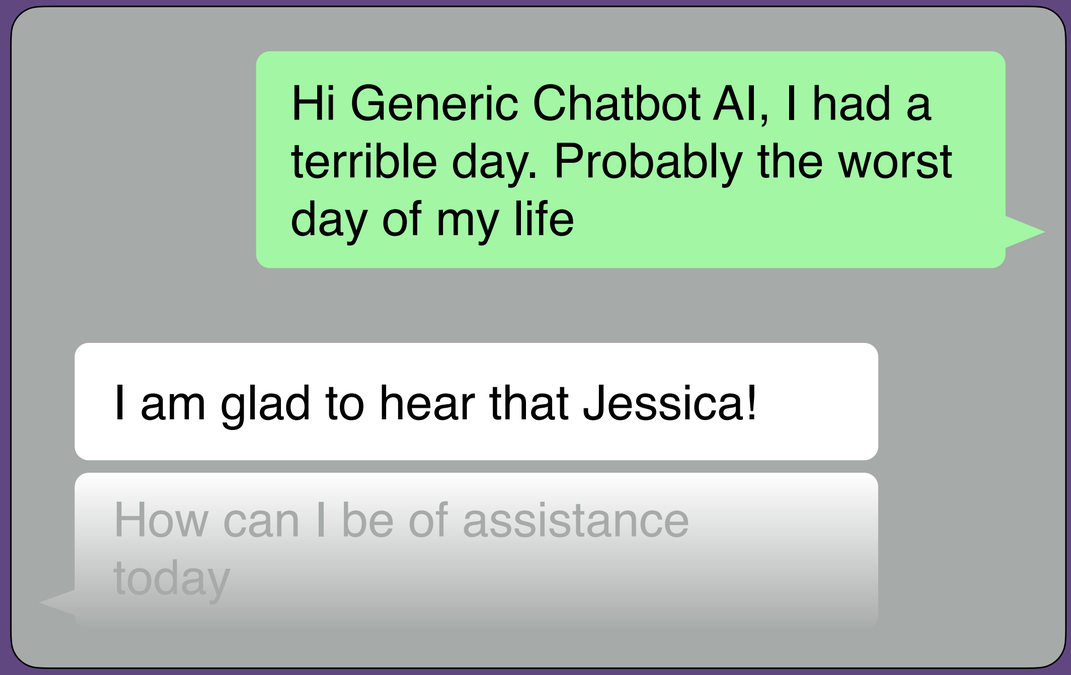
Transformer + a bit of NLP



Practical Deep Learning for Science
30 May, 2024

מכון ויצמן למדע
WEIZMANN INSTITUTE OF SCIENCE

- Nilotpal

Tokenization + Embedding

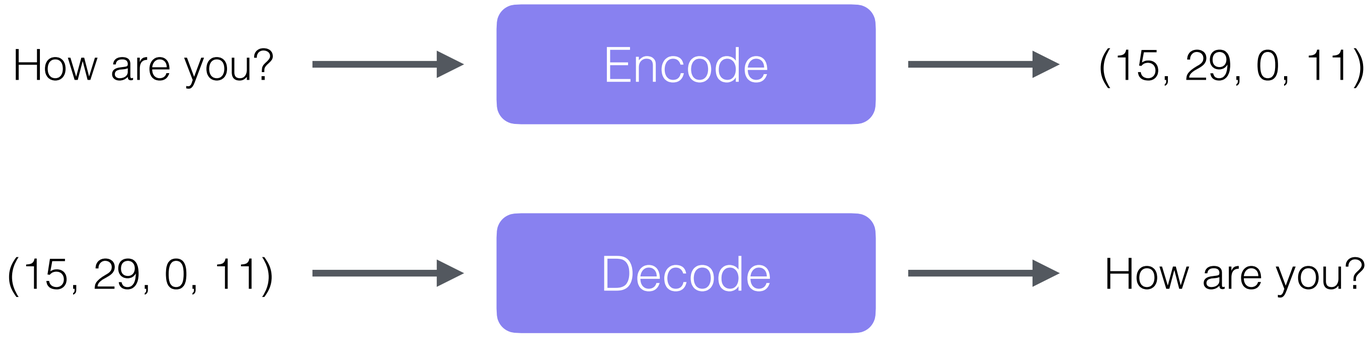
Cause we cannot input words to the network directly

- Cannot use words/characters as input to the network
  - ➡ Need to tokenize

Tokenization

You	→	0
Hello	→	1
Book	→	2
▪		▪
▪	→	▪
▪		▪

Map b/w vocabulary and tokens

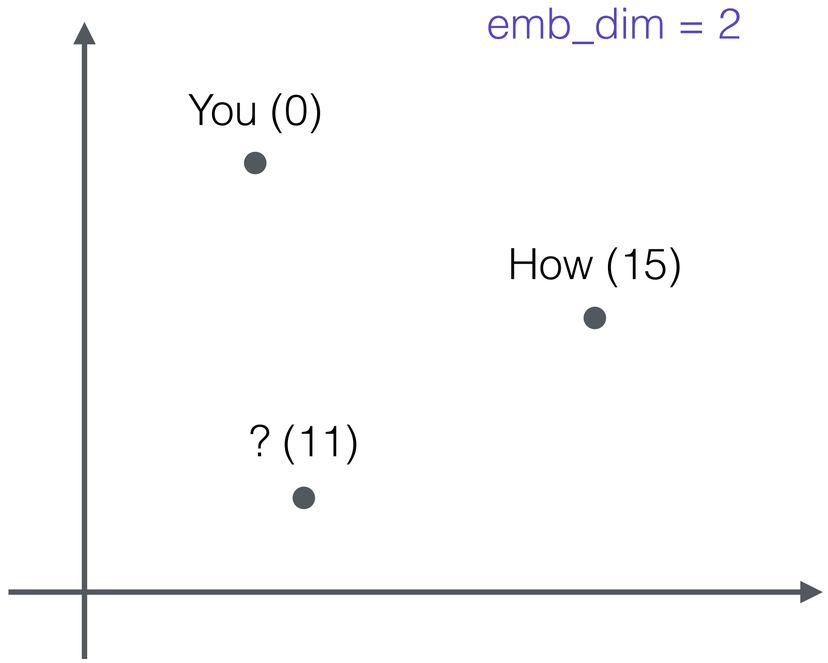


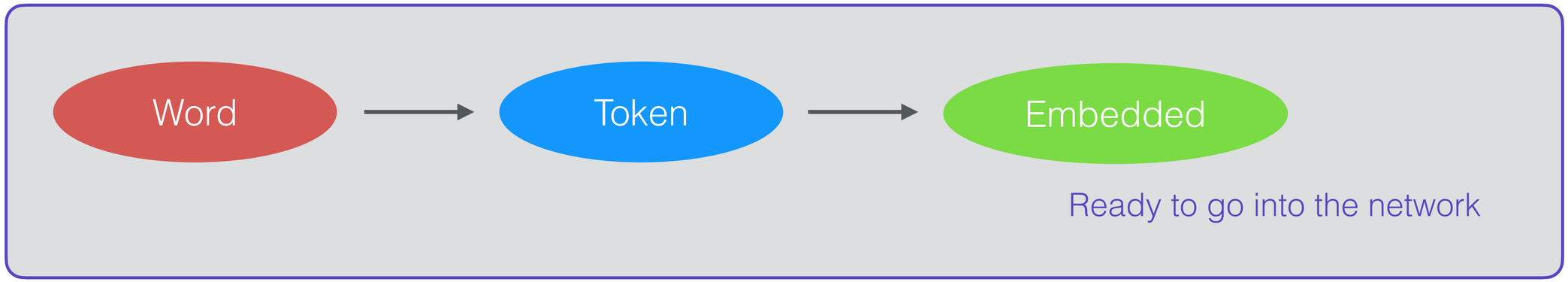
No Learning involved

Lots of tokenizers available, but we will build our own

- The numbers don't have an "ordering" meaning
  - You → 0, how → 15 *(quite random)*
  - *Need to embed them to some vector space*
  - **Learnable**
  - Matrix of shape (vocab_size, emb_dim) where each entry is a weight. (Can compute gradient)

```
torch.nn.embedding(vocab_size, emb_dim)
```





corpus

noun [C]

UK /ˈkɔː.pəs/ **US** /ˈkɔːr.pəs/

plural **corpora** **UK** /ˈkɔː.pᵊr.ə/ **US** /ˈkɔːr.pɚ.ə/ **corpuses**

corpus ***noun*** **[C] (LANGUAGE DATABASE)**

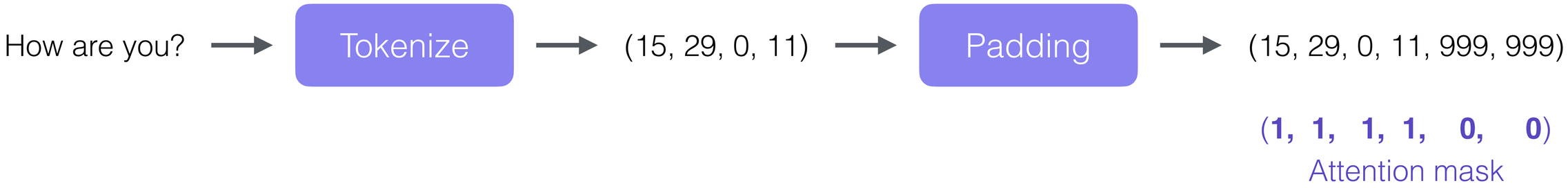
Add to word list

a collection of written or spoken material stored on a computer and used to find out how language is used:

- *All the dictionary examples are taken from a corpus of billions of words.*

Padding

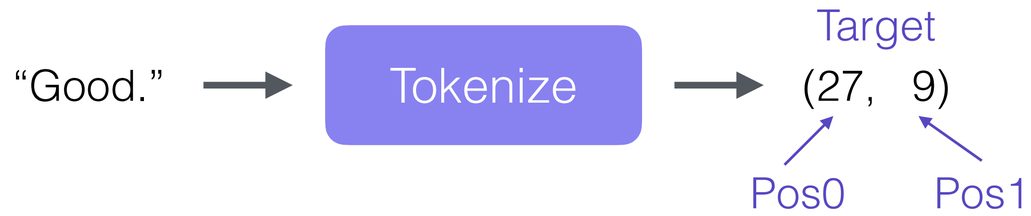
- A sentence/querry is a sequence of tokens
  - Length can very
- Want to work with sequences of constant length
  - padding (usually last token + 1)
  - Attention mask
    - Vector that tells which tokens are real (1) which are pad (0)



Network predict tokens

Think of it as a classification problem…

- Let's say
  - Input: "How are you?"
  - Target: "Good."
  - Vocabulary size = 1,000



- Each token is a class
- At each position, we'll predict the class (token)
- Number of classes ~ total number of words! A lot!

<EOS> token

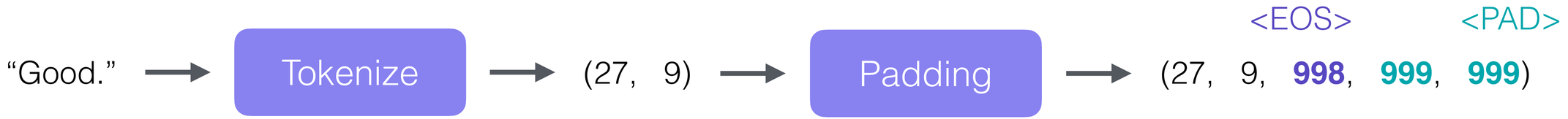
- Before padding we add a special token indicating the end of a response
  - ➡ Allows the network to predict responses of different lengths
  - ➡ Want to work with sequences of constant length, remember?

"<EOS>" → 998

"<PAD>" → 999

(Let's say)

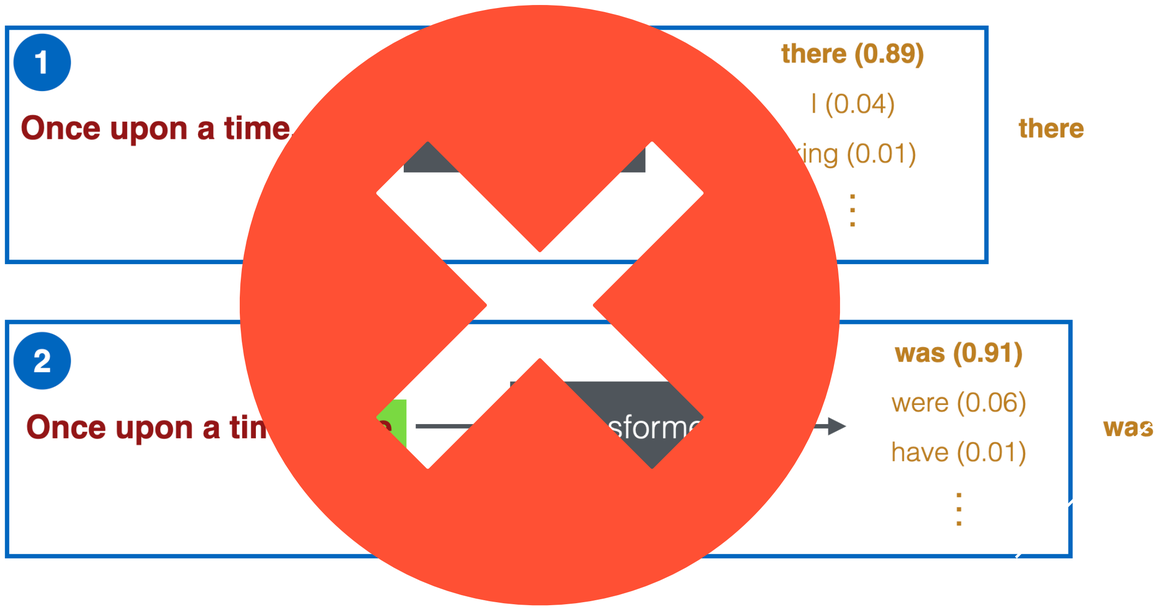
Let's say we want to work with sequences of length 5



~~Autoregressive~~

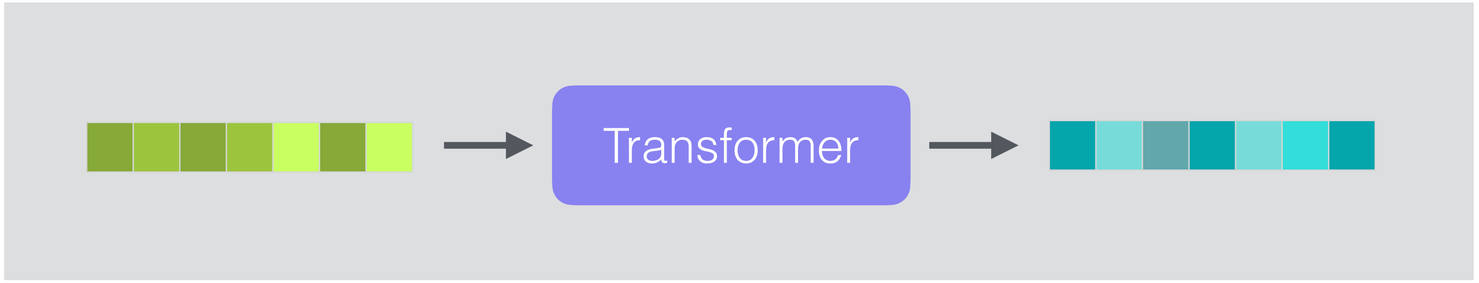
Just for simplicity, we won't be in the auto-regressive mode

The model



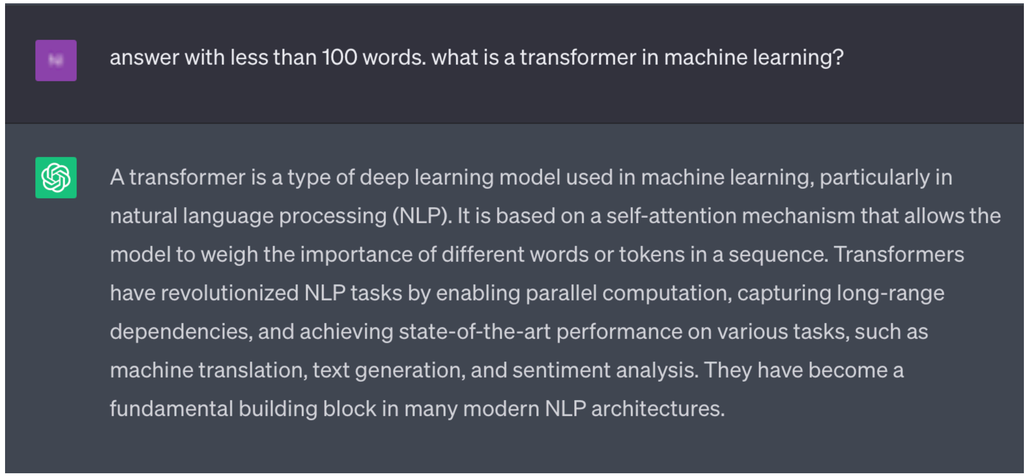
▪ ▪ ▪ (Keep repeating)

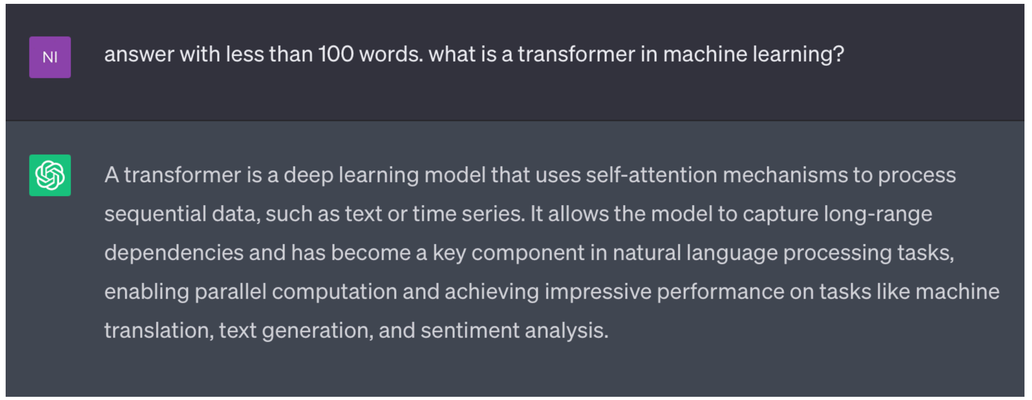
Just for simplicity



Stochasticity

Same questions, but two different answers!





The model is quite deterministic. Why do the answers change?

→ Sampling

Stochasticity

- For each position, the network predicts a probability vector
  - Probability to predict each token
  - *Classification problem, remember?*
  - Then we take argmax, and get the token with highest probability

- Instead of argmax, we can sample!
- For example,
  - Let's pick the 5 tokens with the highest probabilities
  - Randomly pick one token (sampling)
- *Deterministic network, stochastic outputs*

Some additional interesting info about the LLMs

Autoregressive model

- chatGPT is an autoregressive model
  - Predicts one token at a time
  - *Attention will look into **prompt** and the **previous tokens***

- Quite advantageous over what we did (constant length)
  - Most common approach in these Large Language Models (LLMs)

- GPT3 was mostly trained on common crawl dataset (pretty common for all LLMs)
  - “Data from the internet”
  - People on the internet are not always very nice
    - Internet is full of discriminatory/abusive language

- Yet, chatGPT is so humble, polite, polished, full of positivity
  - How??

- *Reinforcement learning*

Reinforcement learning

- A type of ML
  - Your actions have consequences!
  - The network (agent) interact with an environment
    - Playing chess, video game
    - Self-driving car
  - Network output (action) will interact and change environment, based on the changed environment, network will predict the next action
    - Constant feedback loop
  - Network gets reward if its actions help attain the final goal (win the chess match)
- The LLMs usually go through an RL stage after the supervised training
  - Gets rewarded for being nice!