Part IV
Probability models for sequences

or: why ChatGPT is just another probability model trained mostly by likelihood maximization

QUESTION. What does GPT stand for?

GENERATIVE
It's a generative probability model: it generates a random output, not just a prediction

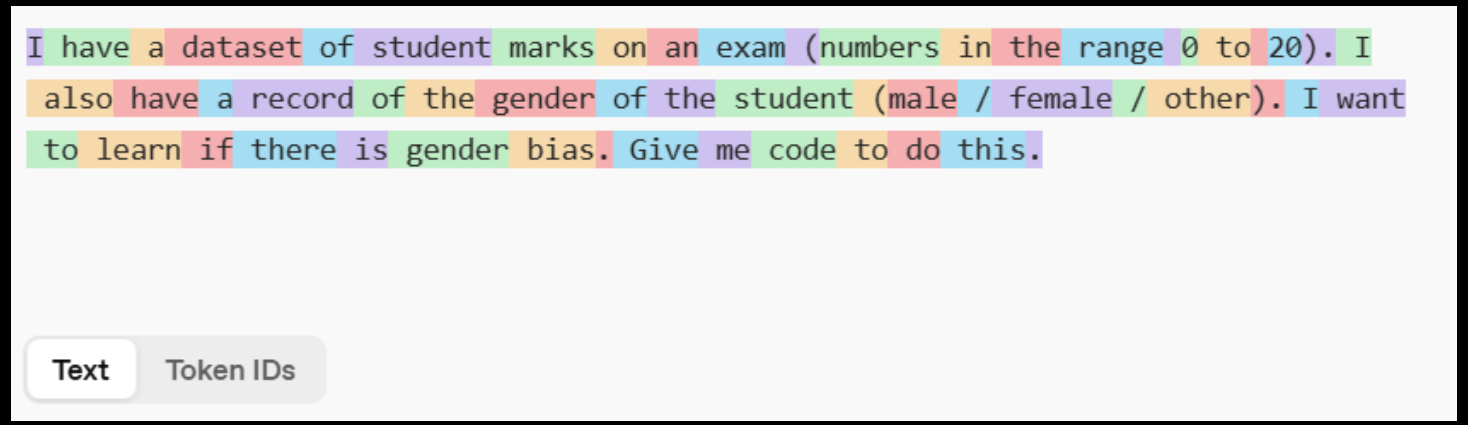
PRE-TRAINED
It's been trained by maximum likelihood estimation to fit a corpus of documents

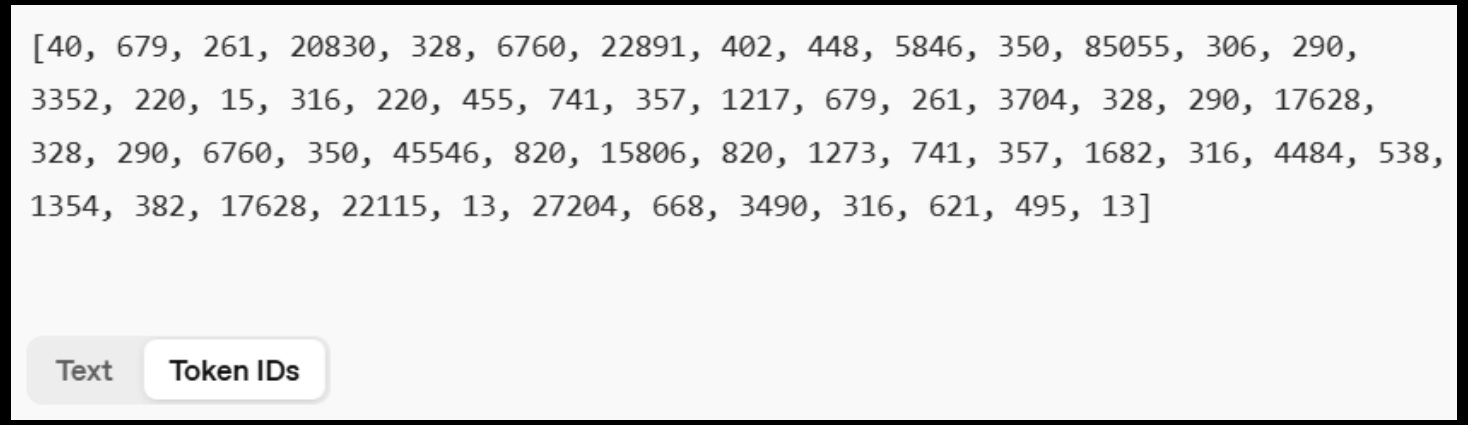
TRANSFORMER
The specific probability model we'll use

[Lecture notes for this part will be available after the end of term]

A piece of text is a sequence of tokens from a finite alphabet.

ChatGPT uses a vocabulary of ≈200k tokens.





GPT tokenizer: https://platform.openai.com/tokenizer

A piece of text is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text that looks like English?

- Write a piece of text of length ℓ as $\underline{x} = x_0 x_1 x_2 \cdots x_\ell$
- We want code to generate a random text $\underline{X}$

```
def X():
  ???
```

- Our code should have learnable parameters, call them θ
- We'll collect a corpus of documents $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$ and tune θ to make our code produce outputs similar to these documents

```
def X(θ):
  ???
```

genius insight!

- This is a probability model, and the random variable $\underline{X}$ has a likelihood function $\Pr_{\underline{X}}(\underline{x}; \theta)$
- We can fit the model by likelihood maximization: $\max_\theta \sum_i \log \Pr_{\underline{X}}(\underline{x}^{(i)}; \theta)$

A piece of text is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text that looks like English?

Why might this be interesting?

Text completion:
we give it initial text $\underline{x} = x_0 x_1 \cdots x_m$ and it completes the text.

In probability language, text completion is just sampling from a conditional distribution:

$$\mathbb{P}(\underline{X} = \underline{x} \,|\, X_0 = x_0, \ldots, X_m = x_m)$$

```
To make the perfect
pasta sauce, first ␣
```

- *This is how we talked with LLMs prior to ChatGPT.*
- *Current LLMs are still based on the same sort of probability models, fine-tuned to work better in chats.*

A piece of text is a sequence of tokens from a finite alphabet.

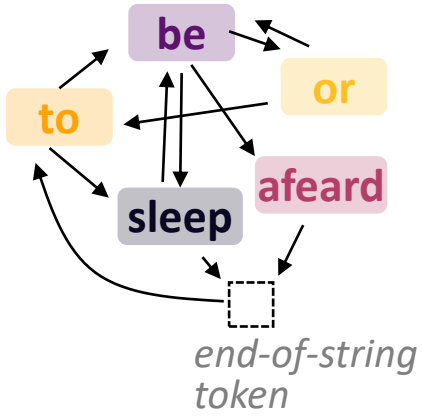
How might we generate a random piece of text that looks like English?

Why might this be interesting?

So, what might this code look like?

- `def X(θ): return ???`
- We also want its likelihood $\Pr_{\underline{X}}(\underline{x}; \theta)$ for training by MLE

Markov model for text generation



Let $\mathcal{W}$ be the vocabulary, including a stop token □

Probability model: generate $\underline{X}$ by starting at □ and jumping from token to token until we hit □ again.

$$□ \to X_1 \to X_2 \to \cdots \to X_L \to □$$

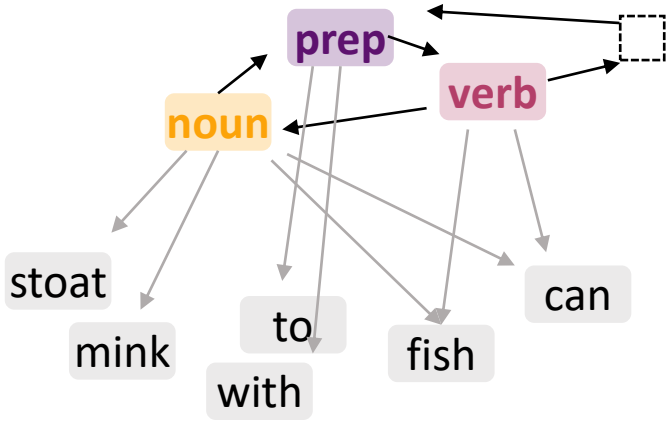
Choose the jumps according to a transition probability function $f_\theta : W \times W \to [0,1]$

$$\mathbb{P}(X_{n+1} = v | X_n = u) = f_\theta(u, v)$$

The likelihood function is easy:

$$\Pr_{\underline{X}}(x_1 x_2 \cdots x_\ell; \theta) = f_\theta(□, x_1)\, f_\theta(x_1, x_2) \cdots f_\theta(x_{\ell-1}, x_\ell)\, f_\theta(x_\ell, □)$$

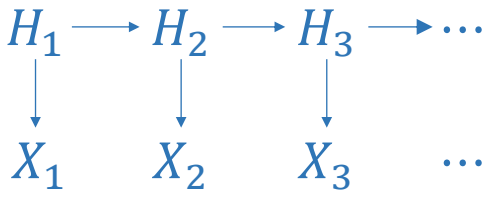
Hidden Markov model for text generation



hidden states $\mathcal{H}$

vocabulary $\mathcal{W}$

Probability model:



We need parameters for the H-transition probabilities $f_\theta: \mathcal{H} \times \mathcal{H} \to [0,1]$
and also for the emission probabilities $g_\theta: \mathcal{H} \times \mathcal{W} \to [0,1]$

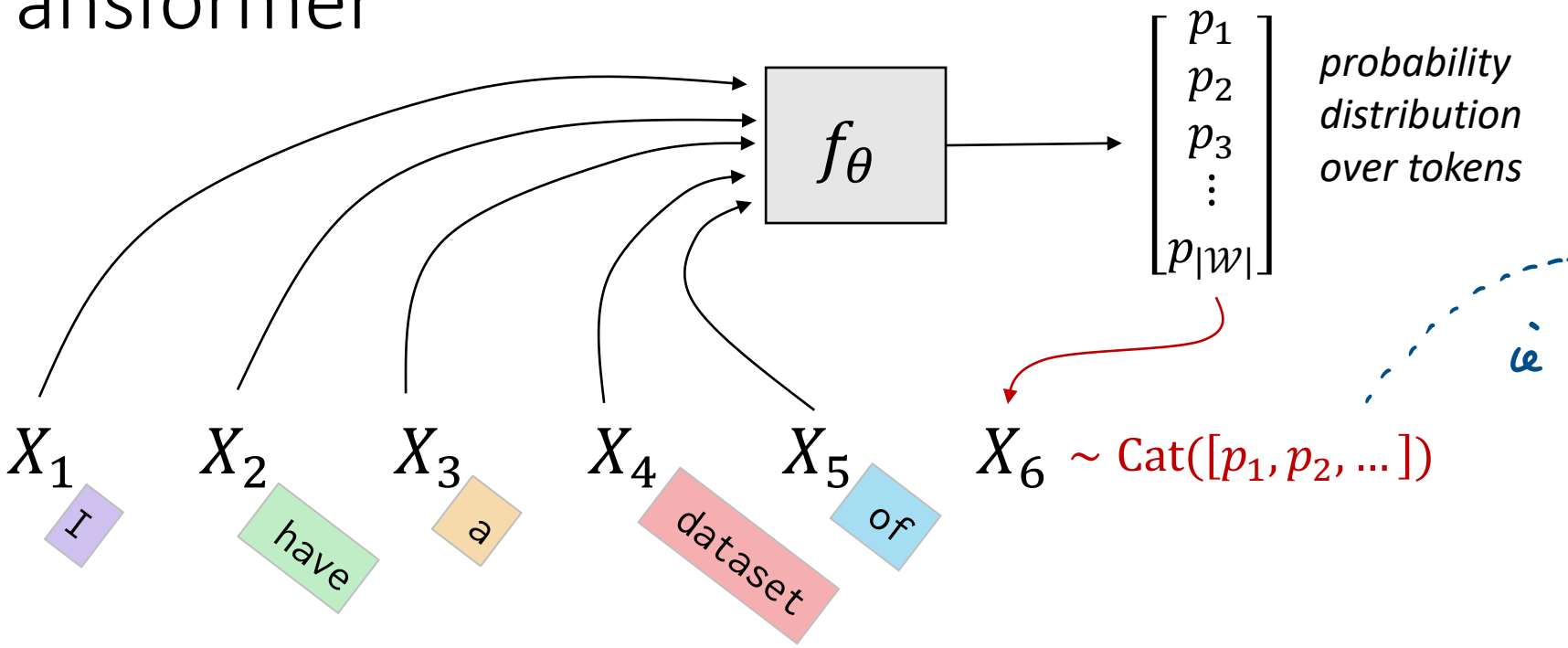
The likelihood function is a horrible sum:

$$\Pr_{\underline{X}}(\underline{x};\theta) = \sum_{\underline{h}} \Pr_{\underline{X}}(\underline{x}|\underline{h};\theta)\,\Pr_{\underline{H}}(\underline{h};\theta)$$

$$P(\underline{X}=\underline{x}) = \sum_{\underline{h}} P(\underline{X}=\underline{x}|\underline{H}=\underline{h})\,P(\underline{H}=\underline{h})$$

LOTP.

Transformer



$P(X_6 = j) = p_j$

i.e. $P(X_6 = x_6) = p_{x_6} = [f_\theta(\underline{x}_{<6})]_{x_6}$

since our probability vector is just $p = f_\theta(\underline{x}_{<6})$.

The likelihood function is easy:

$$\Pr_{\underline{X}}(\underline{x}; \theta) = \prod_i \Pr(x_i | \underline{x}_{<i}; \theta) = \prod_i f_\theta(\underline{x}_{<i})_{x_i}$$

I have a dataset of

Text | Token IDs

What would a suitable f_θ look like?

- It has to accept an input sequence of any length
- It has to be differentiable with respect to θ, so that we can run numerical optimization to maximize $\Pr(\underline{x}; \theta)$ … so `while` loops and recursion are out!
- Don't try to implement a clever f_θ yourself: just give it the flexibility to express the general sort of operations that are needed, and rely on MLE to discover the exact function. *The Bitter Lesson.*