ChatGPT, continued

A piece of text $\underline{x}$ is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text $\underline{X}$ that looks like English?

Why might this be interesting?

It can complete text for us. We can use this ability to get it to answer questions.

perfect pasta sauce ␣

- Nonna Ginny's biggest secret to making **the perfect sauce** is to cook with love
- **Perfect Pasta Sauce.** If you say "Marcella Hazan's pasta sauce" although she has many, this is the one everyone
- so flavorful this time of year they melt into the **perfect pasta sauce**. Add fresh basil and a drizzle of olive oil
- Secret Ingredient Italians Add For **Perfect Pasta Sauce**. I wasn't ready to hear this.
- Making the **perfect pasta sauce** is a skill that can be mastered with practice and the right ingredients

When I asked my nonna how to make the perfect pasta sauce, she gave me this recipe: ␣

A piece of text $\underline{x}$ is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text $\underline{X}$ that looks like English?

Why might this be interesting?

How can we make it *reply?*

perfect pasta sauce

Here's a perfect, dependable, weeknight-easy yet dinner-party-worthy pasta sauce. It's the one Italian nonnas and chefs quietly agree on: Marcella Hazan's tomato–butter sauce, plus a couple of optional refinements […]

OpenAI invented "Reinforcement Learning from Human Feedback" (RLHF) to get useful replies. This led to ChatGPT.

1. Gather a human-preference dataset, and learn what people want a reply to look like
2. Tweak our text-generating model so that preferred replies are more likely

1. Learn what humans want a reply to look like

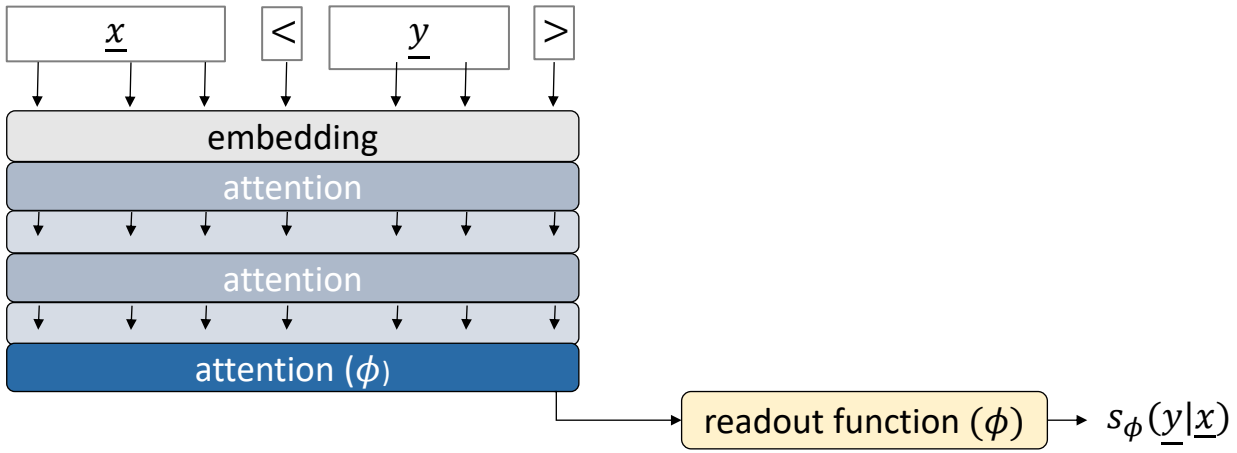
Let's use pairwise-comparison data for preferences (as in chess, sport, marking student work, ...).

prompt $\underline{x}$	reply $\underline{y}_0$	reply $\underline{y}_0$	preferred
write me a story about a fox named louis who one day goes down a alley in a poor, crime filled part of tokyo and gets mugged by a dragon	Once upon a time in the bustling metropolis of Tokyo, there lived a sly and cunning fox named Louis. Known for his sharp wit and mischievous nature, Louis had managed to thrive in the city's diverse landscapes.	In the dimly lit alleys of Tokyo's red-light district, a sleek and agile fox named Louis navigated through the crowded streets with ease. His rusty orange fur blended perfectly with the neon lights	1
can you role play as the cmd.exe of an ancient alien ai astromancy computer system called Omni-Seer?	Astromancy Computer System, Version 907.14.12.31 Copyright (c) Alien Technocrats Type 'help' for a list of commands. ** Omni-Seer ** What would you like to do?	Certainly, but remember that I am Qwen, trained by Alibaba Cloud, and not a real cmd.exe from an ancient alien AI system called Omni-Seer. Greetings, enigmatic traveler. In the vast expanses of Omicronia, where stars whisper secrets to those who listen, the Omicron Seer commands the forces of the cosmos through the ancient language of data, the syntax of logic, and the commands of the command prompt.	0

https://huggingface.co/datasets/allenai/tulu-3-wildchat-ultrafeedback

and use it to train a *reward model* model $s_\phi(\underline{y}|\underline{x})$ that measures how much humans like reply $\underline{y}$ to prompt $\underline{x}$.

Train only the final attention layer & a new readout layer. For the rest, re-use the fitted weights from model $\underline{X}$.



Exercise (Preference modelling)

We have a dataset of tennis matches. (It's not a tournament—some pairs have more than one match.)

We wish to assign a skill θ_y to each player y. Propose a probability model for the winner of a match Z based on the skills of the two players, and explain how to fit your model.

player y_0	player y_1	winner z
Anisimova	Swiatek	1
Sabalenka	Anisimova	1
Swiatek	Bencic	0
Andreeva	Bencic	1
Anisimova	Swiatek	0
⋮		

$$\mathbb{P}(Z=1 : \theta_{y_0}, \theta_{y_1}) = \frac{e^{\theta_{y_1}}}{e^{\theta_{y_0}} + e^{\theta_{y_1}}}$$

$$Z \sim \text{Bin}\left(1, \text{sigmoid}(\theta_{y_1} - \theta_{y_0})\right)$$

2. Tweak the model so that preferred replies are more likely

Our original language model $\underline{X}$ was trained to mimic texts from a corpus $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$

θ = parameters of our $\underline{X}$ model

$$\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log \Pr_{\underline{x}}(\underline{x}^{(i)}) \equiv \max_{\theta} \mathbb{E}_{\hat{\underline{x}} \sim \text{corpus}} \log \Pr_{\underline{x}}(\hat{\underline{x}})$$

Build a new language model, starting from the $\underline{X}$ model, and write $\underline{Y}_{\underline{x}}$ for its completion of prompt-text $\underline{x}$.
Train it to produce human-preferred responses.

η = parameters of our new model

$$\max_{\eta} \mathbb{E}_{\hat{\underline{x}} \sim \text{prompt dataset}} \left[\mathbb{E}_{\underline{Y}_{\hat{\underline{x}}}} s(\underline{Y}_{\hat{\underline{x}}} \mid \hat{\underline{x}}) \right]$$

and regularize it so it doesn't collapse to a few canned responses:

$$- \beta \times \left\{ \begin{array}{l} \text{how far the likelihood func. } \underline{y} \mapsto \Pr_{\underline{Y}_{\hat{\underline{x}}}}(\underline{y}) \\ \text{is from } \underline{y} \mapsto \Pr_{\underline{x}}(\hat{\underline{x}}\,\underline{y}) \end{array} \right\}$$

WHAT'S EXAMINABLE?

- Markov chains
- Trigram model as a Markov chain
- ~~RNNs~~
- ~~Transformers~~
- Preference modelling
- ~~RLHF~~

Markov modelling I:

fitting a Markov chain

A Markov Chain is a sequence in which each item is generated based only on the preceding item.

Thus the likelihood of a sequence $x_0 x_1 \cdots x_\ell$ is

$$\mathbb{P}(x_0 x_1 \cdots x_\ell) = \mathbb{P}(X_0 = x_0) \times \prod_{i=1}^{\ell} \mathbb{P}(X_i = x_i \mid X_{i-1} = x_{i-1})$$

Note: if the items are continuous, use likelihood notation.

$$\text{Pr}(x_0 \cdots x_\ell) = \text{Pr}_{x_0}(x_0) \prod_{i=1}^{\ell} \text{Pr}_{x_i}(x_i \mid X_{i-1} = x_{i-1})$$

This is the basis for fitting Markov models ...

Applications of Markov chains: dynamical systems

Deterministic

Conway's Game of Life:

$x_{n+1} = \texttt{update\_grid}(x_n)$

Ordinary differential equations:

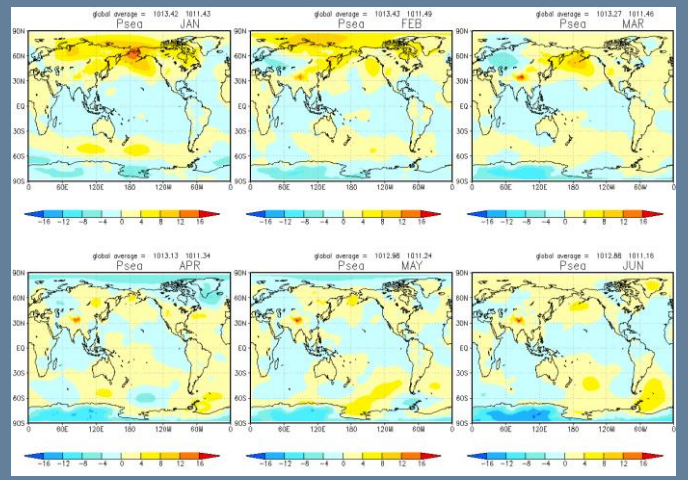
$$\frac{dv(t)}{dt} = g - \frac{\mu}{m} v(t)^2$$

QUESTION. What's the link between 2nd-order ODEs and Markov's trigram model?

Probabilistic

Weather:

$X_{t+1} = \texttt{random\_sim\_step}(X_t)$



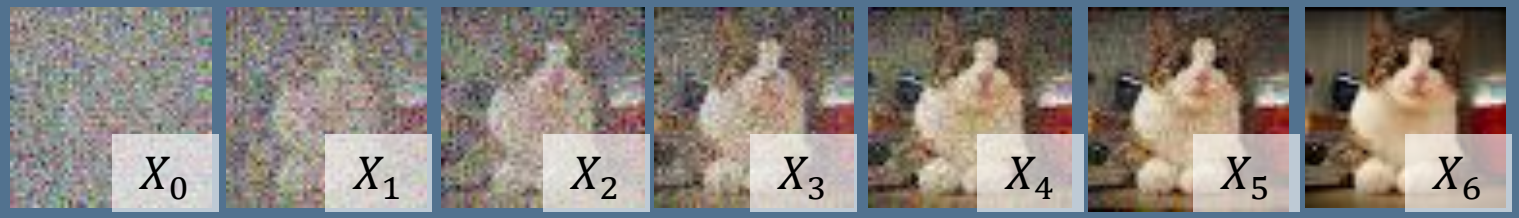
Assets in a stock market

Load on a webserver

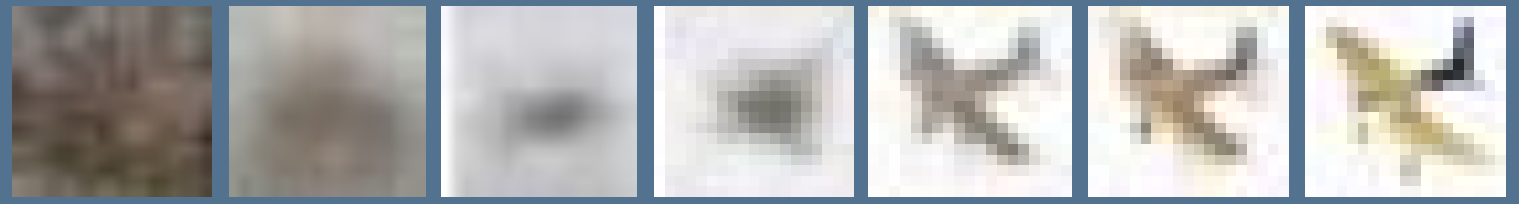
Applications of Markov chains: stable diffusion

Given an image, create a sequence with progressively more and more noise, until we get pure noise. Do this for many images, to create a training dataset of sequences.

Reverse the sequences. Train a Markov chain to learn the dynamics $(X_t | X_{t-1} = x)$.



If we apply these dynamics to a new pure-noise image, we will generate a novel image.



Example 12.1.1: fitting a Markov model

Let $[x_0, x_1, \ldots, x_n]$ be a time series which we believe is generated by

$$X_{i+1} = a + b\,X_i + N(0, \sigma^2).$$

Estimate a and b using maximum likelihood estimation.

[Also known as *red noise*. Klaus Hasselman, 2021 Nobel Prize for Physics, introduced such models to climate science.]

$$X_{i+1} \sim N(a + b X_i, \sigma^2)$$

$$\Pr(x_0 \cdots x_n) = \Pr_{x_0}(x_0) \prod_{i=1}^{n} \Pr(x_i | x_{i-1})$$

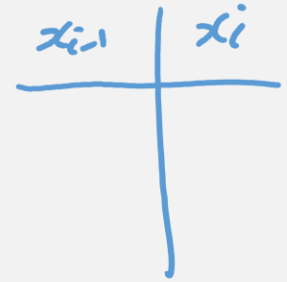
$$= \eta(x_0) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i - (a + b x_{i-1}))^2 / 2\sigma^2}$$

We want to maximize this over a, b, σ, η

This is just like fitting a linear model!

$X_i \sim a + b x_{i-1}$

use x_{i-1} as a feature to predict x_i.

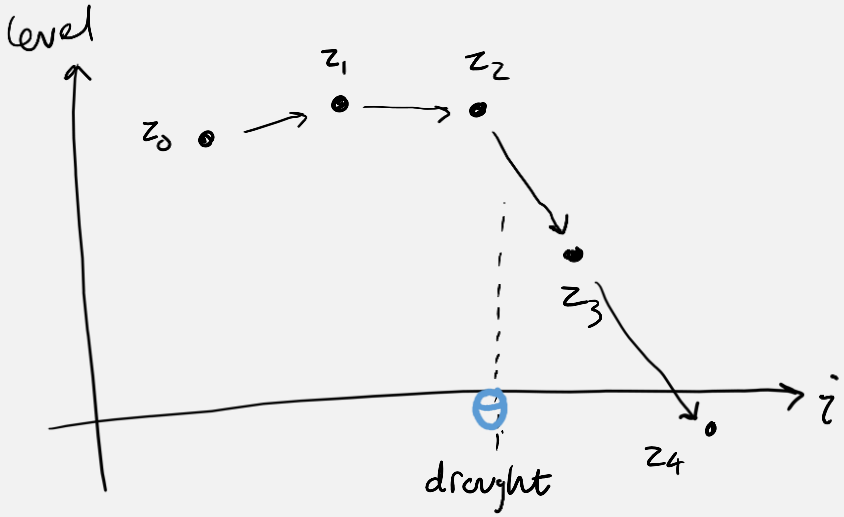


Mock exam question 1

I have been monitoring annual river levels for many years, and I have collected a dataset $(z_0, \ldots, z_n)$ where z_i is the level in year i of monitoring.

I believe that for the first few years the level each year was roughly what it was the previous year, plus or minus some random variation; but that some year a drought started, and since then the level has decreased on average.

Estimate when the drought started.



$$z_i \sim z_{i-1} + N(0, \varepsilon^2) - \mu 1_{i>\theta} \sim N(z_{i-1} - \mu 1_{i>\theta}, \varepsilon^2)$$

$$p_r(z_0 \ldots z_n) = \Pr(z_0) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-(z_i - (z_{i-1} - \mu 1_{i>\theta}))^2 / 2\varepsilon^2}$$

We want to max this over $\mu, \theta, \varepsilon, \Pr(z_0)$.

$$= \Pr(z_0) \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-(\delta_i + \mu 1_{i>\theta})^2 / 2\varepsilon^2}$$

$\delta_i := z_i - z_{i-1}$

Markov modelling II:
working with Markov chains and related random systems

Laws of probability

A **random system** is a collection of random variables, which may or may not be independent
e.g. pixels in an image, tokens in a piece of text

The **chain rule of probability** lets us factorize the joint likelihood in any order we like

$$\mathbb{P}(A = a, B = b) = \mathbb{P}(A = a)\,\mathbb{P}(B = b|A = a)$$

$$\Pr_{A,B}(a, b) = \Pr_A(a)\,\Pr_B(b|A = a) = \Pr_B(b)\,\Pr_A(a|B = b)$$

Bayes' Rule.

$$p(a,b) = p(a)\,p(b|a) = p(b)\,p(a|b)$$

$$p(a,b,c) = p(a)\,p(b|a)\,p(c|a,b) = p(b)\,p(a|b)\,p(c|a,b) = \dots$$

The **law of total probability** is for decomposing an event

$$\mathbb{P}(A = a) = \sum_b \mathbb{P}(A = a, B = b) = \sum_b \mathbb{P}(A = a \mid B = b)\,\mathbb{P}(B = b) \qquad \text{or} \quad \Pr_A(a) = \int_b \Pr_A(a \mid B = b)\,\Pr_B(b)\,db$$

and also the version with baggage $\{C = c\}$

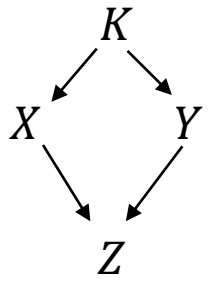
$$\mathbb{P}(A = a \mid C = c) = \sum_b \mathbb{P}(A = a \mid B = b, C = c)\,\mathbb{P}(B = b \mid C = c)$$

Causal models

A **causal model** is a random system that's generated sequentially, as in computer code, where each random variable is generated from specified others. The Directed Acyclic Graph (DAG) of dependencies is called the *causal diagram.*

The joint likelihood can be written as the product of terms specifying how each vertex arises **from its parents.**

```
K := Noise()
X := K + Noise()
Y := K + Noise()
Z := min(X,Y)
```



by Chain Rule

$$p(k,x,y,z) = p(k)\,p(x|k)\,p(y|x,k)\,p(z|x,y,k)$$
$$= p(k)\,p(x|k)\,p(y|k)\,p(z|x,y)$$

by this causal diagram.

Informally, a **Markov chain** is a sequence in which each item is generated based only on the preceding item.

Formally, a Markov Chain with transition matrix P
is a causal model $X_0 \to X_1 \to X_2 \to \cdots$
in which $\mathbb{P}(X_{n+1} = y | X_n = x) = P_{x,y}$

A MC with trans. lik. func. $p(x,y)$
is a causal model $X_0 \to X_1 \to \cdots$
in which $\Pr_{X_{n+1}}(y \mid X_n = x) = p(x,y)$

Example.

In the causal diagram $A \to B \to C$, calculate $p(b|a,c)$.

$$p(b|a,c) = \frac{p(b,a,c)}{p(a,c)} \quad \text{by defn. of cond. prob}$$

$$= \frac{p(a,b,c)}{\sum_{b'} p(a,b',c)} \quad \text{by LoTP}$$

$$= \frac{p(a)\,p(b|a)\,p(c|b)}{\sum_{b'} p(a)\,p(b'|a)\,p(c|b')} \quad \text{by causal Dag}$$

$$= \frac{p(b|a)\,p(c|b)}{\sum_{b'} p(b'|a)\,p(c|b')}$$

compare to Bayes rule

$B \to C$ $\quad p(b|c) = \frac{p(b)\,p(c|b)}{\sum_{b'} p(b')\,p(c|b')}$

Example 11.2.1
(Multi-step transition probabilities)

Consider a Markov chain $X_0 \to X_1 \to \cdots$ with transition matrix P. Calculate $\mathbb{P}(X_2 = x_2 | X_0 = x_0)$.

$$\mathbb{P}(X_2 = x_2 \mid X_0 = x_0) = \sum_{x_1} \mathbb{P}(X_2 = x_2 \mid X_1 = x_1, X_0 = x_0)\, \mathbb{P}(X_1 = x_1 \mid X_0 = x_0)$$

by LoTP with baggage $\{X_0 = x_0\}$

$$= \sum_{x_1} \mathbb{P}(X_2 = x_2 \mid X_1 = x_1)\, \mathbb{P}(X_1 = x_1 \mid X_0 = x_0)$$

by causal diagram.

$$= \sum_{x_1} P_{x_1 x_2} P_{x_0 x_1}$$

$$= [P^2]_{x_0 x_2}$$

The Memorylessness Theorem

For a Markov chain $X_0 \to X_1 \to X_2 \to \cdots$,
conditional on the present the future is independent of the past.

future present past

$$\mathbb{P}(X_3 = x_3 \mid X_2 = x_2, X_1 = x_1, X_0 = x_0) = \mathbb{P}(X_3 = x_3 \mid X_2 = x_2)$$

future present past

$$\mathbb{P}(X_3 = x_3 \mid X_1 = x_1, X_0 = x_0) = P(X_3 = x_3 \mid X_1 = x_1)$$

future present past

$$\mathbb{P}(X_3 = x_3 \mid X_2 = x_2, X_0 = x_0) = P(X_3 = x_3 \mid X_2 = x_2)$$

Proof: by induction, using LoTP with baggage.

Hidden Markov model

An HMM with transition matrix P and emission matrix E is the causal model

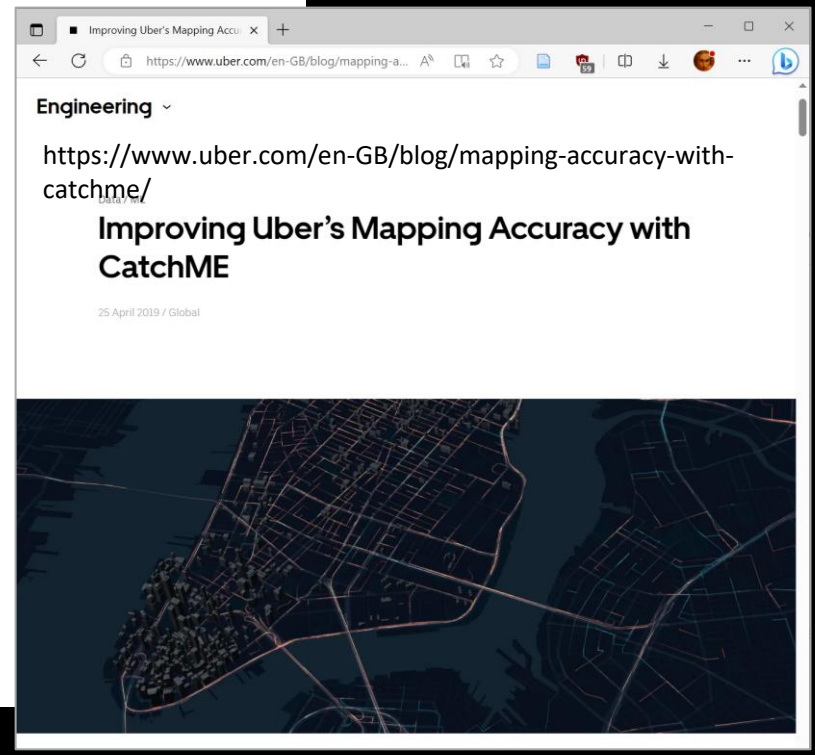
$$H_1 \longrightarrow H_2 \longrightarrow H_3 \longrightarrow \cdots \quad \text{hidden 'true' state}$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$X_1 \quad X_2 \quad X_3 \quad \cdots \quad \text{noisy observed sequence}$$

in which $\mathbb{P}(H_{n+1} = v | H_n = u) = P_{u,v}$ and $\mathbb{P}(X_n = x | H_n = u) = E_{u,x}$

For a hidden Markov model, the likelihood function $\Pr_{\underline{X}}(\underline{x})$ is nasty, and it's pretty much impossible to learn the model from $\underline{x}$ data.
So why are hidden Markov models useful?



Uber collects precise logs (both $\underline{h}$ and $\underline{x}$) from a few drivers, so it can learn P and E using straightforward supervised learning. Then it can estimate $(H_T | x_1 \cdots x_T)$ for drivers with only $\underline{x}$ data.

- Viterbi: cunning trick to find the most likely value from this distribution
  [only works for discrete state spaces]
- Probabilistic approach: find the full posterior distribution
  [see Example Sheet 4; can extend to continuous state spaces]