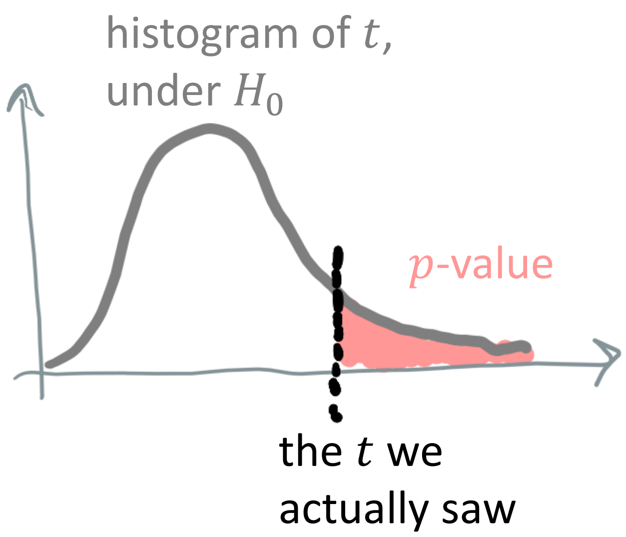
§9.3 Hypothesis testing

Hypothesis testing asks whether a proposed probability model H_0 could plausibly have generated the dataset.

- The p-value is the probability that an outcome as extreme as what we actually saw might have come about by chance, if H_0 were true.
- A low p-value suggests we should reject H_0.
- “Extreme” is measured by a test statistic t, which is up to us to choose.



Hypothesis testing is good for questions that we can cast as
"Does the evidence suggest rejecting H_0?"

H_0: the data was generated by my model

- Is my probability model a good enough fit for the dataset?
- Is my new algorithm better than the standard one?
- Does this new UI allow users to do their task faster than before?
- Is this drug effective, compared to placebo?

comparing groups of readings

H_0: all groups come from the same distribution

There's a common way to set out hypothesis tests for comparing groups (as well as for many similar tasks), called the Neyman-Pearson approach.

Neyman-Pearson hypothesis testing

Let x be the dataset.

Propose a general parametric model H_1, and express H_0 as a restriction on one or more parameters

1. Choose a test statistic based on mle estimates of the parameters of H_0 and H_1
2. Define a random synthetic dataset X^*, what we might see if H_0 were true.
3. Let p be the probability (assuming H_0 to be true) of seeing $t(X^*)$ as or more extreme than the observed $t(x)$.

A low p-value is a sign that H_0 should be rejected.

General model

H_1: $x_i \sim N(a, \sigma^2)$

$y_i \sim N(b, \sigma^2)$

$z_i \sim N(c, \sigma^2)$

H_0: $a = b = c$. All samples $\sim N(\mu, \sigma^2)$

$$\left.\begin{aligned} \hat{a} &= \bar{x} \\ \hat{b} &= \bar{y} \\ \hat{c} &= \bar{z} \end{aligned}\right\} \text{mles under } H_1$$

$$\hat{\mu} = \overline{\text{concat}(x, y, z)}$$

Can I invent a test statistic using these four parameters, which is liable to be bigger if H_0 is false?

More generally:

$$t = \frac{\max_{\text{params of } H_1} \Pr(\text{data} \mid H_1)}{\max_{\text{params of } H_0} \Pr(\text{data} \mid H_0)}$$

Exercise 9.3.2 (Equality of group means).
We are given three groups of observations from three different systems

$$x = [7.2, 7.3, 7.8, 8.2, 8.8, 9.5]$$
$$y = [8.3, 8.5, 9.2]$$
$$z = [7.4, 8.5, 9.0]$$

Do all three groups have the same mean?

fitted pars for full model

Test statistic:

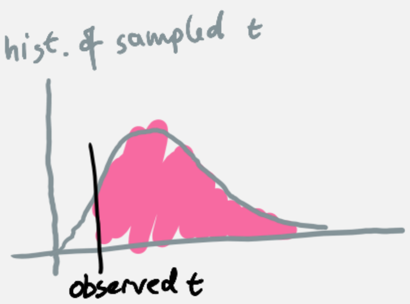
$$t = (\hat{a} - \hat{\mu})^2 + (\hat{b} - \hat{\mu})^2 + (\hat{c} - \hat{\mu})^2$$

fitted under H_0

```
# 1. Define test statistic
def t(x,y,z):
    μ = np.mean(np.concatenate([x,y,z]))
    a,b,c = [np.mean(v) for v in [x,y,z]]
    return (a-μ)**2 + (b-μ)**2 + (c-μ)**2

# 2. To generate a synthetic dataset, assuming H0 ...
xyz = np.concatenate([x,y,z])
μ̂ = np.mean(xyz)
σ̂ = np.sqrt(np.mean((xyz-μ̂)**2))
def rxyz_star():
    return (np.random.normal(size=len(x), loc=μ̂, scale=σ̂),
            np.random.normal(size=len(y), loc=μ̂, scale=σ̂),
            np.random.normal(size=len(z), loc=μ̂, scale=σ̂))

# 3. Sample the test statistic, find the p-value
t_ = np.array([t(*rxyz_star()) for _ in range(10000)])
p = np.mean(t_>=t(x,y,z))
```



EXERCISE.

Consider the data for IA student marks.

a. What's a sensible H_0 to test?

b. What's a natural test statistic?

c. How might we generate a random synthetic dataset?

gender	mark
F	17
F	14
M	18
O	11
M	17
⋮	⋮

(a) Introduce a richer model H_1: Mark $\sim \mu_{\text{gender}} + N(0,\sigma^2)$

let H_0 be: $\mu_F = \mu_M = \mu_O$

(b) $t = (\hat{\mu}_F - \hat{\mu})^2 + (\hat{\mu}_M - \hat{\mu})^2 + (\hat{\mu}_O - \hat{\mu})^2$ where $\hat{\mu}_F, \hat{\mu}_M, \hat{\mu}_O$ are MLE under H_1 and $\hat{\mu}$ is MLE under H_0

(c) Parametric resampling. Under H_0, Marks $\sim \hat{\mu} + N(0, \hat{\sigma}^2)$.

Conclusion: for the real marks from last year, $p = 0.71\%$ So we reject H_0.

NON-PARAMETRIC RESAMPLING

(a) H_0: marks for all three genders are drawn from the same distribution.

(c) If H_0 is true, then the best fit is the empirical distribution of all marks (concatenated together).
Let's simply resample from this.

Conclusion: $p = 0.80\%$

PERMUTATION TESTING

(a) H_0: you'd get the same mark regardless of your gender.

(c) Imagine a parallel universe where every student gets assigned a random gender (25 Women, 110 Men, 5 Other).
Simulate this parallel universe by randomly permuting the gender column.

Conclusion: $p = 0.82\%$

IB Data Science syllabus

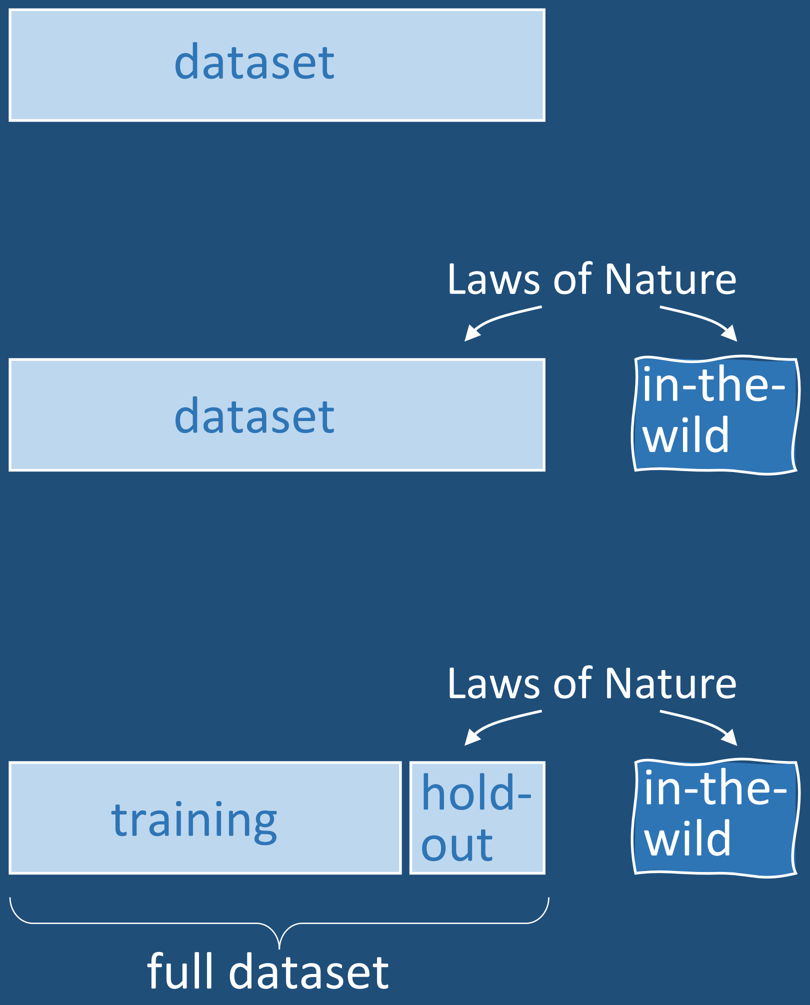
Using a probability model to describe data → Models that depend on linear combinations of features → Parameter interpretation and identifiability

Fitting a model's unknown parameters using MLE → Fitting via least squares (when appropriate)

Reasoning about uncertainty →
- Bayesian confidence intervals
- Resampling
- Frequentist confidence intervals
- Hypothesis tests

"Induction is the glory of Science and the scandal of Philosophy."

C.D. Broad, 1926



- Maximum likelihood estimation gives us a model that fits the training dataset

But how well will our model work on new data?
("The challenge of induction.")

- Bayesianism and frequentism address this by making careful claims about the Laws of Nature that generated the dataset.

- Alternatively, we could simply say "The performance on in-the-wild data is approximately the performance on holdout data."

Table 2: Results on HotpotQA distractor (dev). (+hyperlink) means usage of extra hyperlink data in Wikipedia. Models beginning with "–" are ablation studies without the corresponding design.

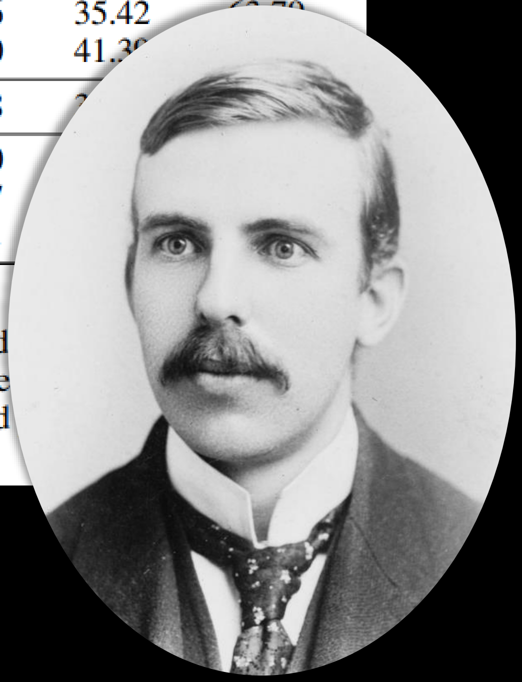
Model	Ans EM	Ans F_1	Sup EM	Sup F_1	Joint EM	Joint F_1
Baseline [53]	45.60	59.02	20.32	64.49	10.83	40.16
DecompRC [29]	55.20	69.63	N/A	N/A	N/A	N/A
QFE [30]	53.86	68.06	57.75	84.49	34.63	59.61
DFGN [36]	56.31	69.69	51.50	81.62	33.62	59.82
SAE [45]	60.36	73.58	56.93	84.63	38.81	64.96
SAE-large	66.92	79.62	61.53	86.86	45.36	**71.45**
HGN [14] (+hyperlink)	66.07	79.36	60.33	87.33	43.57	71.03
HGN-large (+hyperlink)	69.22	82.19	62.76	88.47	47.11	**74.21**
BERT (sliding window) variants						
BERT Plus	55.84	69.76	42.88	80.74	27.13	58.23
LQR-net + BERT	57.20	70.66	50.20	82.42	31.18	59.99
GRN + BERT	55.12	68.98	52.55	84.06	32.88	60.31
EPS + BERT	60.13	73.31	52.55	83.20	35.40	63.41
LQR-net 2 + BERT	60.20	73.78	56.21	84.09	36.56	63.68
P-BERT	61.18	74.16	51.38	82.76	35.42	[illegible]
EPS + BERT(large)	63.29	76.36	58.25	85.60	[illegible]	[illegible]
CogLTX	65.09	78.72	56.15	85.78	[illegible]	[illegible]
– multi-step reasoning	62.00	75.39	51.74	83.10	[illegible]	[illegible]
– rehearsal & decay	61.44	74.99	7.74	47.37	[illegible]	[illegible]
– train-test matching	63.20	77.21	52.57	84.21	[illegible]	[illegible]

Results. Table 2 shows that CogLTX outperforms most of previous method solutions on the leaderboard.[4] These solutions basically follow the frame results from sliding windows by extra neural networks, leading to bounded to insufficient interaction across paragraphs.

Most ML papers don't state an inductive claim.

Perhaps the authors haven't thought hard enough to be able to state one?

Perhaps they prefer to leave you, the reader, to make the inference?



"All science is either physics or stamp-collecting."

Ernest Rutherford (1871–1937)

	model selection	confidence intervals for model parameters	confidence intervals for predictions
BAYESIANIST 1	Given two models, each with a prior weight, use the data to reweight the models	✓	✓
EMPIRICIST 2	Given two models, prefer the one that works better on holdout data	Who cares?	?
FREQUENTIST 3	Given a model, is it a good enough explanation of the data?	✓	✓

4

A possible approach:
1. If there's anything for which I have a justified prior belief, put it into my model as a random variable
2. Choose between competing models empirically
3. Check my final model using frequentist tests
4. Read off confidence intervals, using Bayesianism or frequentistism as appropriate.

GPT is a model for sequences.

- It sees text as a sequence of tokens $\underline{x} = x_0 x_1 x_2 \cdots x_N$
- Its training dataset is a collection of sequences $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \dots, \underline{x}^{(n)}\}$

```
The following is a classic Chinese poem from the Tang dynasty, translated
 into English.

The dawn light strikes the head of my bed
I see leaves
```

TEXT TOKEN IDS

```
[464, 1708, 318, 257, 6833, 3999, 21247, 422, 262, 18816, 30968, 11,
14251, 656, 3594, 13, 198, 198, 464, 17577, 1657, 8956, 262, 1182, 286,
616, 3996, 198, 40, 766, 5667, 220]
```

TEXT TOKEN IDS

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

GPT is a probability model for sequences of tokens

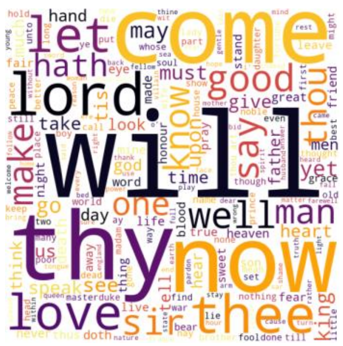
- Let $\underline{X} = X_0 X_1 X_2 \cdots X_N$ be a random sequence of tokens, of random length N
- What's a good probability model for $\underline{X}$
  and how do we fit it to a training dataset $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$?

- Once we have a trained probability model, we can use it for completion.
  We give it an input prompt $\underline{x} = x_0 x_1 \cdots x_m$ and it generates a sample from

$$(\underline{X} \mid X_0 = x_0, \ldots, X_m = x_m)$$

`GPT playground:` https://platform.openai.com/playground?mode=complete

§12. What’s a good probability model for sequences, and how can we fit it?



Goal: generate Shakespeare-like text

Bag-of-words text generation

Choose each word randomly, independently.

`"us the incite o'er a land-damn are peace incardinate take him worthy quick generals` $\square$`"`

end-of-sentence token

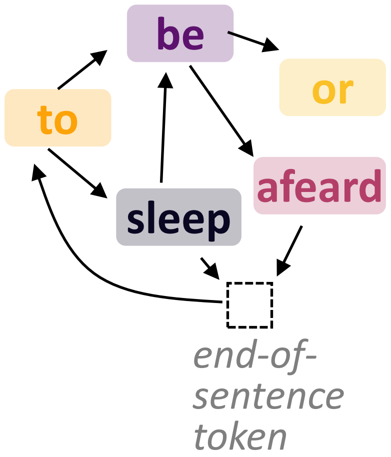
Probability model: generate $\underline{X}$ by producing random words until we produce $\square$.

$$X_1, X_2, \ldots, X_N, \square$$

$$\Pr_{\underline{X}}(x_1 x_2 \cdots x_n) = \Pr(x_1)\Pr(x_2) \times \cdots \times \Pr(x_n)\Pr(\square)$$

Let's let $\Pr(w) = p_w$ where $p = [p_{w_1}, p_{w_2}, \ldots, p_{w_V}, p_\square]$ is a probability vector with an entry for each word in the vocabulary.

We can learn the p vector by maximizing the likelihood of the dataset $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$.
The mle is simple: p_w = fraction of occurrences of word w in the dataset



Markov model

Based on a graph of word-to-word transitions.

```
"to foreign princes lie in your blessing god who
shall have the prince of rome □"
```

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

$$\square \to X_1 \to X_2 \to \cdots \to X_N \to \square$$

$$\Pr_{\underline{X}}(x_1 x_2 \cdots x_n) = \Pr(x_1|\square) \times \Pr(x_2|x_1) \times \cdots \times \Pr(x_n|x_{n-1}) \times \Pr(\square|x_n)$$

Let's let $\Pr(w|v) = P_{vw}$ for some matrix P that denotes the word-to-word transition probabilities. The maximum likelihood estimate for P is easy to find, by simple counting of word pairs.

Andrei Markov (1856–1922)

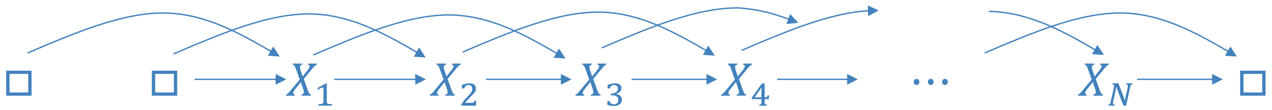
be contented **to be** what they
who is **to be** executed this
in him **to be** truly touched
took occasion **to be** quickly woo'd

Markov's trigram model

```
"to be wind-shaken we will be glad to receive at
once for the example of thousands □"
```

Probability model: Generate $\underline{X}$ by starting with □□ and repeatedly generating the next word based on the preceding **two**, until we produce □.

$$\Pr_{\underline{X}}(x_1 x_2 \cdots x_n) = \Pr(x_1 | \square\square) \Pr(x_2 | \square x_1) \Pr(x_3 | x_1 x_2) \times \cdots \times \Pr(x_n | x_{n-2} x_{n-1}) \Pr(\square | x_{n-1} x_n)$$

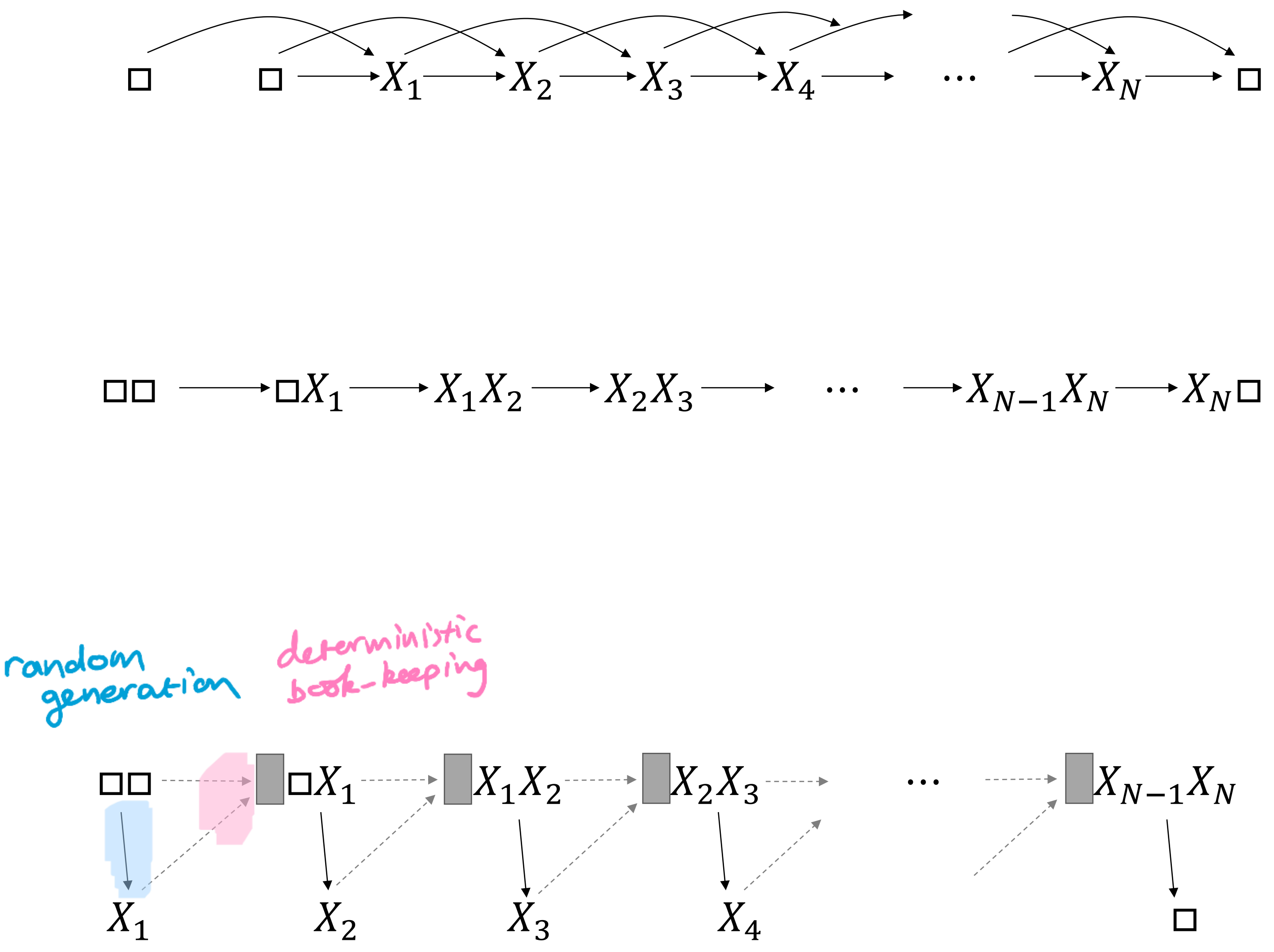


Let's let $\Pr(w|uv) = P_{(uv)w}$

It's easy to estimate P, the (word,word)-to-word transition probabilities, by simple counting.

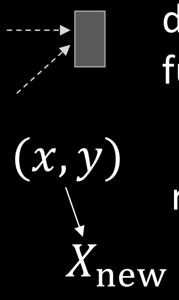
(Before counting, preprocess the dataset by putting □□ at the start and □ at the end of every sentence.)

Different ways to write the trigram model:



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

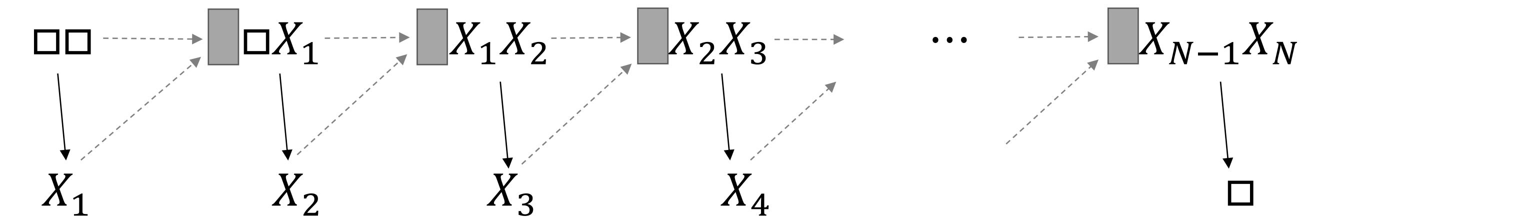
The trigram model is a Markov chain, whose items are word-pairs.



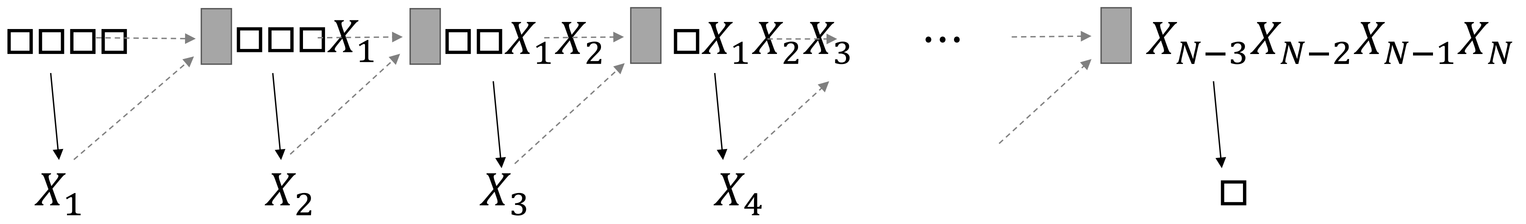
deterministic bookkeeping function $f((x, y), z) = (y, z)$

random generation

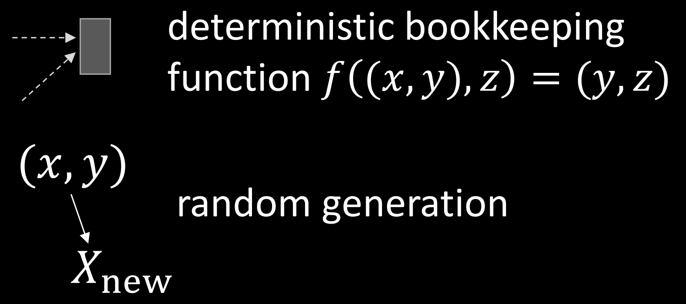
Can we get a better model by using more history?



Trigram character-by-character model trained on Shakespeare:
`"on youghtlee for vingiond do my not whow'd no crehout withal deepher forand a but thave a doses?"`



5-gram character-by-character model trained on Shakespeare:
`"once is pleasurely. though the the with them with comes in hand. good. give and she story tongue."`

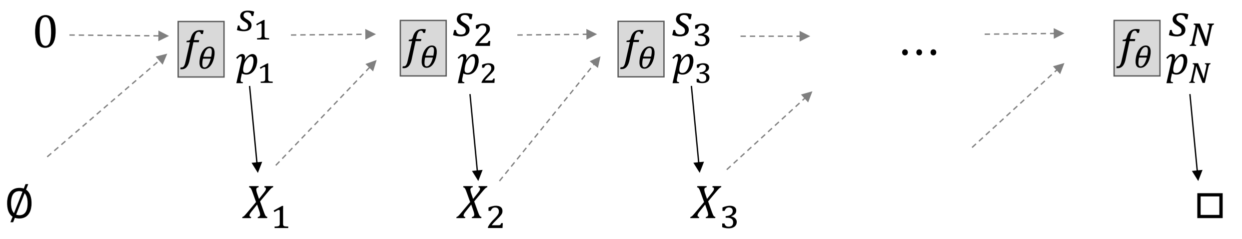


QUESTION. What are the advantages and disadvantages of a long history window?

QUESTION. Can we do better than using a fixed history window?

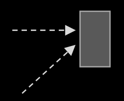
Recurrent Neural Network (RNN)

Let's use a neural network to learn an appropriate history digest. This is more flexible than choosing a fixed history window.



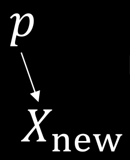
RNN character-by-character model trained on Shakespeare
[due to Andrej Karpathy]:

```
"PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep."
```



learnable function

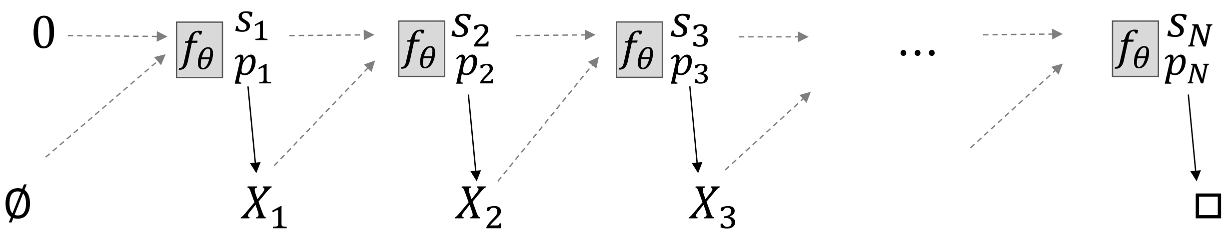
$f_\theta(s, x) = (p, s_{\text{new}})$



random generation

$X_{\text{new}} \sim \text{Cat}(p)$

A Recurrent Neural Network (RNN) is a probability model for generating a random sequence $\underline{X}$.



$X_i \sim \mathrm{Cat}(p_i)$

$(s_{i+1}, p_{i+1}) = f_\theta(s_i, X_i)$

We can train it in the usual way, by maximizing the log likelihood of our dataset.
This is easy, because there's a simple explicit formula for the likelihood of a datapoint:

$$\Pr_{\underline{X}}(x_1, \dots, x_n) = \Pr_{X_1}(x_1)\,\Pr_{X_2}(x_2|x_1) \times \cdots \times \Pr_{X_n}(x_n|x_1 \cdots x_{n-1})\,\Pr_{X_{n+1}}(\Box|x_1 \cdots x_n)$$

by the chain rule for probability

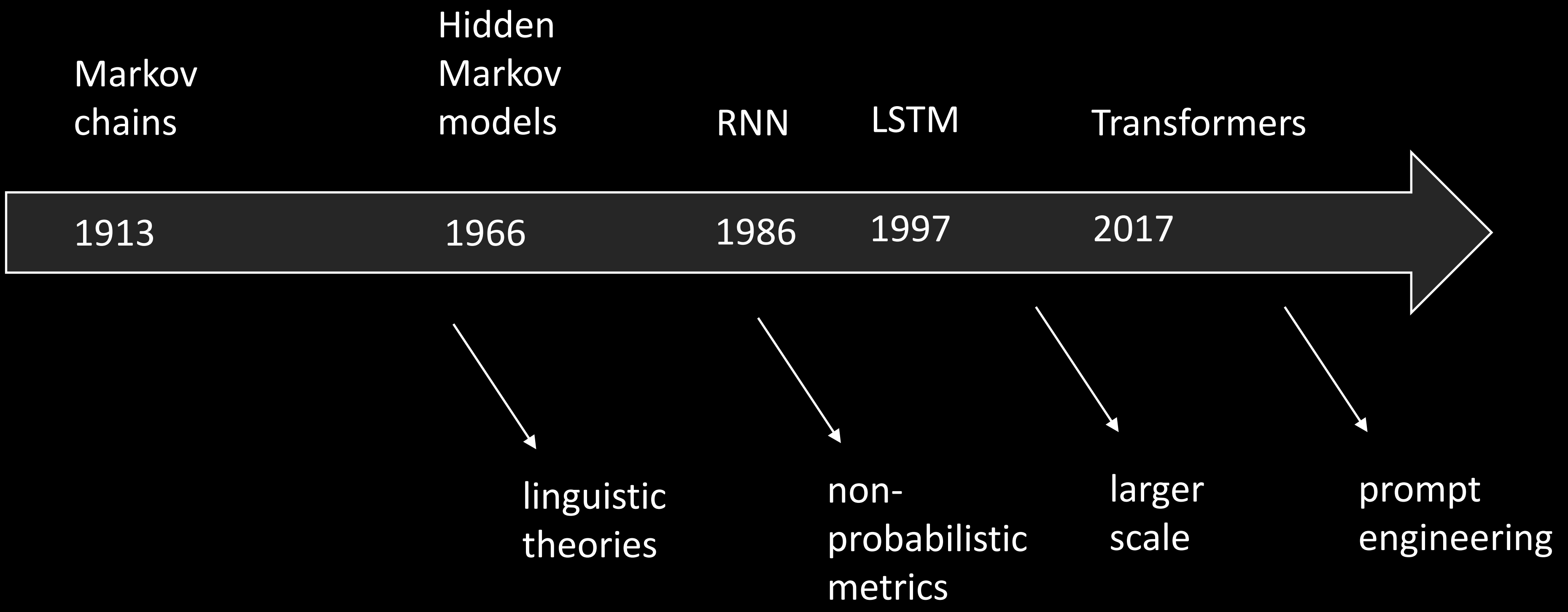
$$= [p_1]_{x_1}\,[p_2]_{x_2} \times \cdots \times [p_n]_{x_n}\,[p_{n+1}]_{\Box}$$

where each p_i is a function of $x_1 \cdots x_{i-1}$

```
def loglik(xstr):
    res = 0
    s,x = 0,□
    for x_next in xstr + "□":
        s,p = f_θ(s,x)
        res += log(p[x_next])
        x = x_next
    return res
```

$\mathbb{P}(A \text{ and } B \text{ and } C) = \mathbb{P}(A)\,\mathbb{P}(B|A)\,\mathbb{P}(C|A,B)$

The history of random sequence models



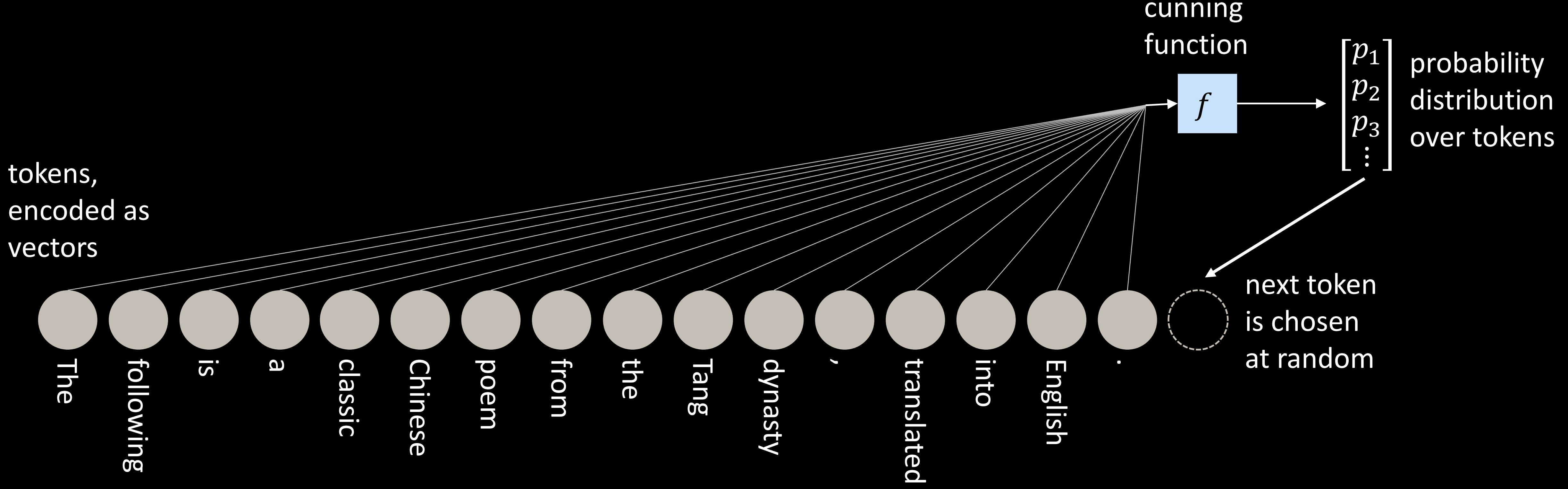
Better models of the data
All trained by maximizing the log likelihood of the data

Transformer architecture

This is a probability model for a random sequence $\underline{X}$.

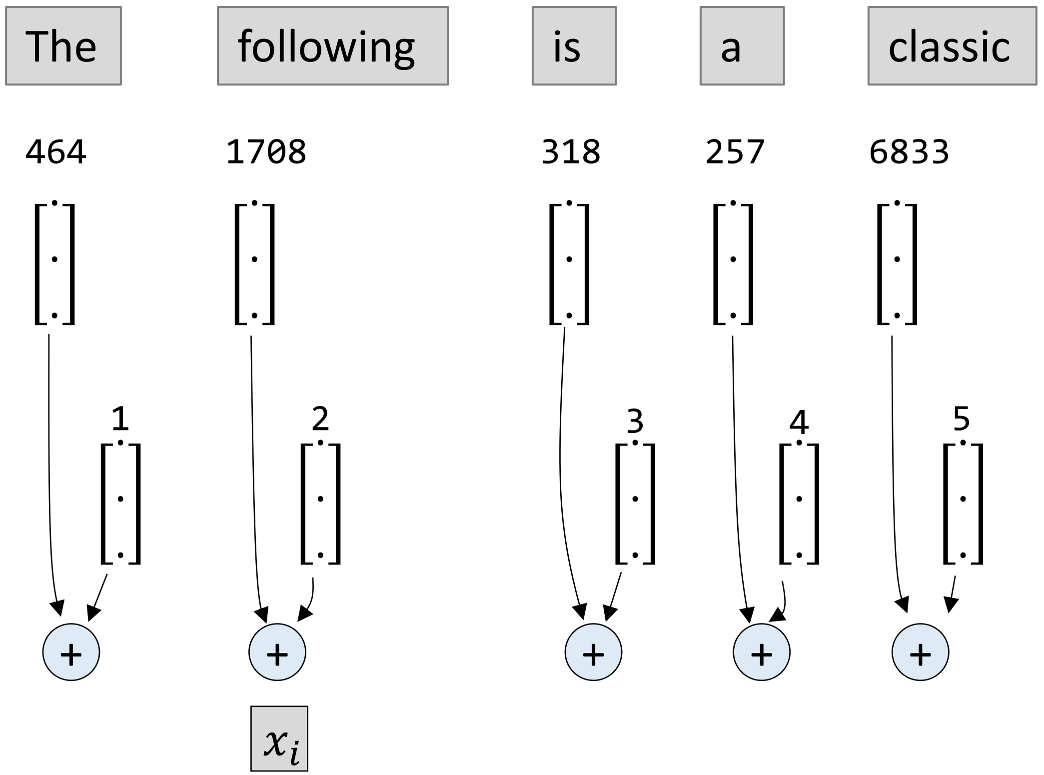
Like the RNN, there's a simple explicit formula for the log likelihood $\Pr_{\underline{X}}(\underline{x})$, so it's easy to train.

It's more powerful than an RNN, because f has access to the full sequence;
it doesn't have to squeeze history into a "history digest" at each step.



The following is a classic Chinese poem from the Tang dynasty, translated into English.

What does f look like? How is it built out of differentiable functions?



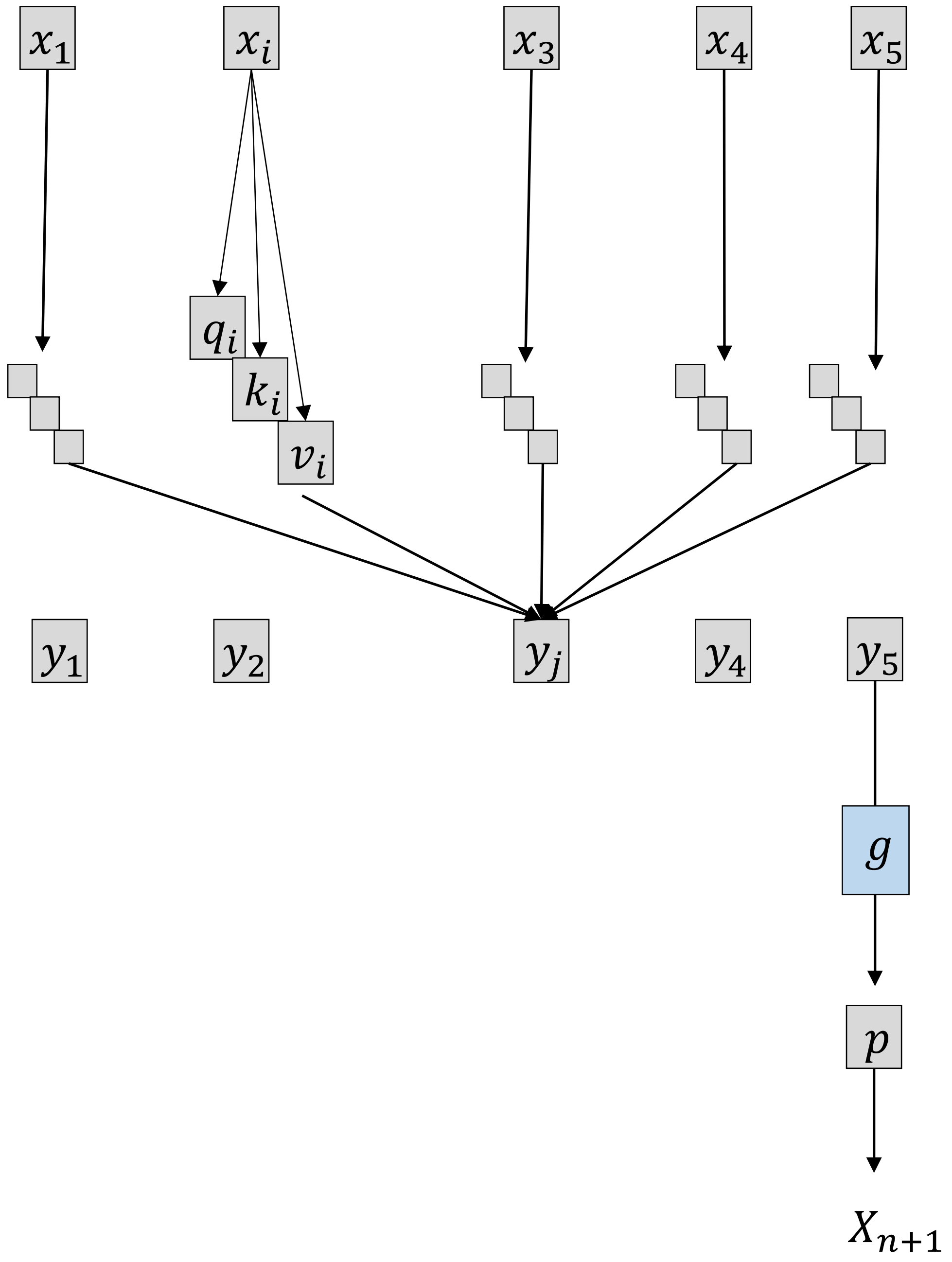
Split the text into tokens $t_i \in \{1, \ldots, W\}$

Turn each token into a vector $e_i \in \mathbb{R}^d$
by looking up an embedding matrix $E \in \mathbb{R}^{W \times d}$

For each position $i \in \{1, \ldots, n\}$
create a position-embedding vector $t_i \in \mathbb{R}^d$

$$\begin{bmatrix} \sin(i) \\ \cos(i) \\ \sin(i/2) \\ \cos(i/2) \\ \vdots \end{bmatrix}$$

Let $x_i = e_i + t_i \in \mathbb{R}^d$



For each position $i \in \{1, \ldots, n\}$,
let $q_i = Qx_i \in \mathbb{R}^e$, let $k_i = Kx_i \in \mathbb{R}^e$, let $v_i = Vx_i \in \mathbb{R}^d$

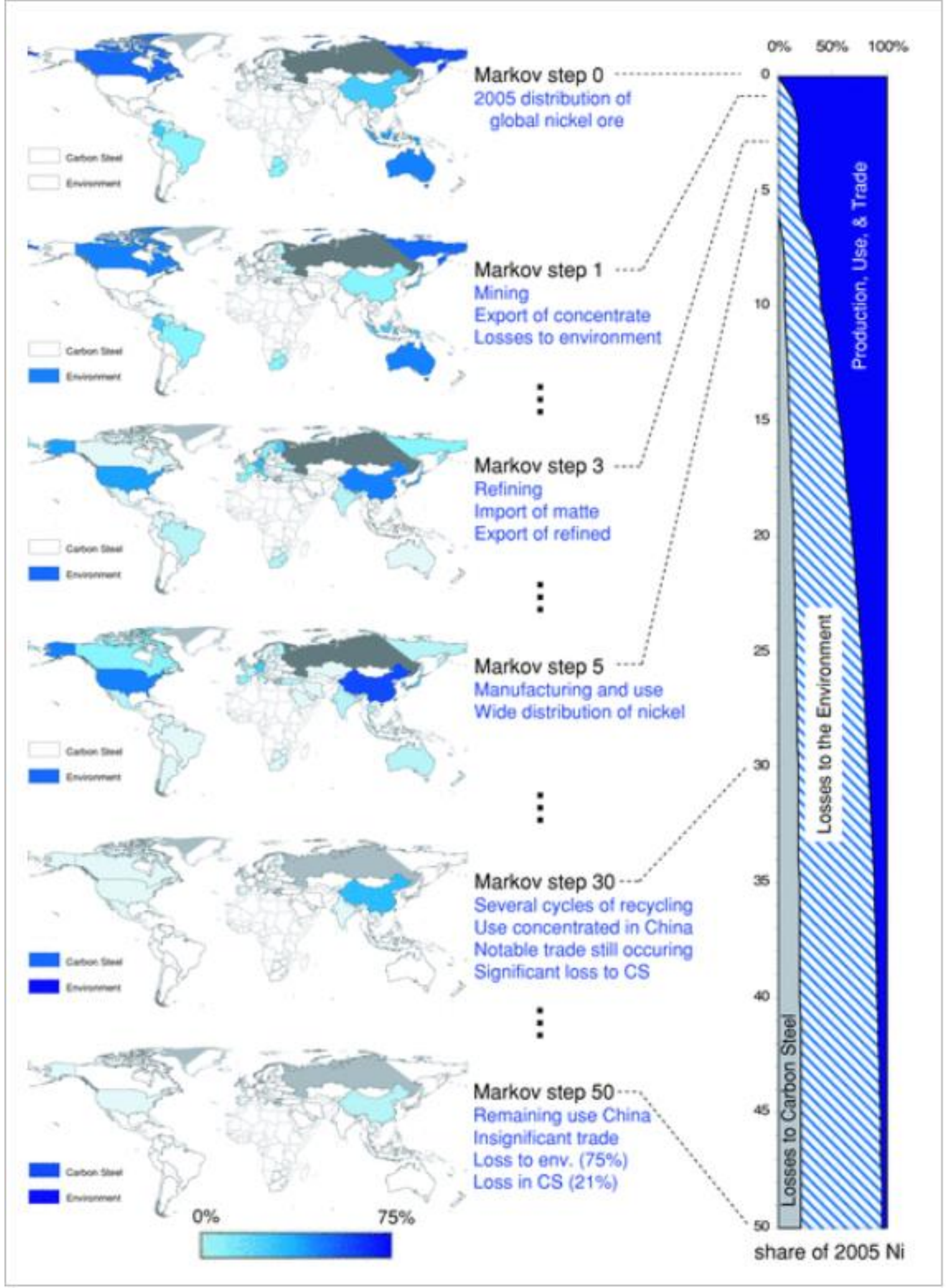
For each position $j \in \{1, \ldots, n\}$ we'll produce an output vector $y_j \in \mathbb{R}^d$, as follows:

1. let $s_{ji} = q_j \cdot k_i$ and $a_{j*} = \text{softmax}\left(s_{j*}/\sqrt{e}\right)$
2. let $y_j = \Sigma_i a_{ji} v_i$

a_{ji} is the attention that we should give to input x_i when computing output y_j

From the final value y_n, compute $p = g(y_n) \in \mathbb{R}^W$ where g is some straightforward neural network

Generate the next token by $X_{n+1} \sim \text{Cat}(p)$



Exploring the Global Journey of Nickel with Markov Chain Models

J of Industrial Ecology, Volume: 16, Issue: 3, Pages: 334-342, First published: 04 April 2012, DOI: (10.1111/j.1530-9290.2011.00425.x)