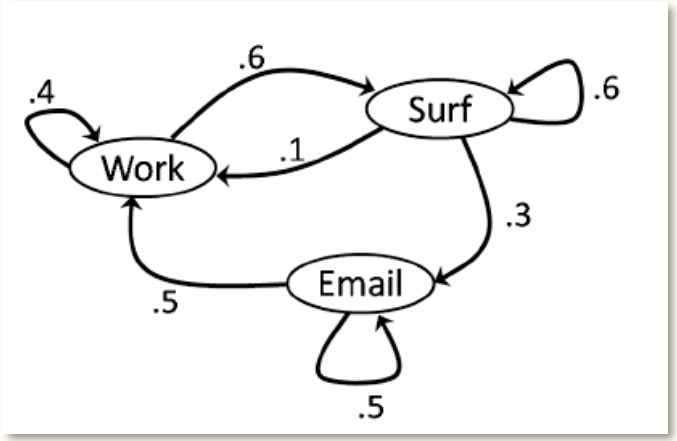
Wrapping up Markov chains and PageRank

CSE 312 Spring 26

Lecture 26

A typical day in my life

How do we interpret this diagram?



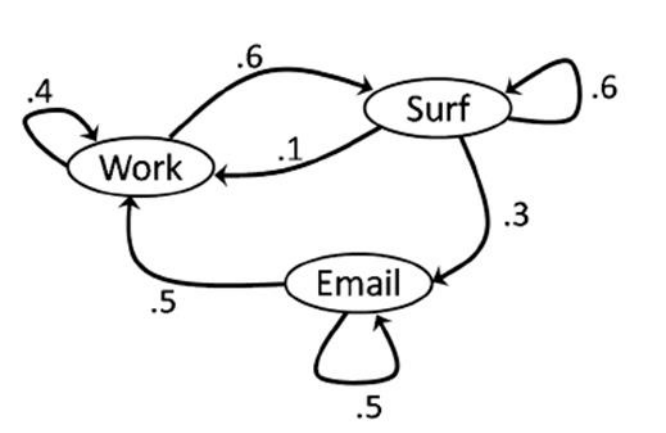
This kind of random process is called a **Markov Chain**

At *each* time step t

- I can be in one of 3 **states**
  - Work, Surf, Email
- If I am in some state s at time t
  - the **labels of out-edges** of s **give the probabilities** of my moving to each of the states at time $t + 1$ (as well as staying the same)
    - so **labels on out-edges sum to** 1

e.g. If I am in Email, there is a 50-50 chance I will be in each of Work or Email at the next time step, but I will never be in state Surf in the next step.

An organized way to understand the distribution at time t



$$[q_W^{(t+1)}, q_S^{(t+1)}, q_E^{(t+1)}] = [q_W^{(t)}, q_S^{(t)}, q_E^{(t)}] \begin{matrix} \\ \\ \\ \end{matrix} \overset{\boldsymbol{M}}{\begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ \cdot & 0 & 0.5 \end{bmatrix}}$$

$\boldsymbol{M^t}$ captures t -step transition probabilities

ite $\boldsymbol{q}^{(t)} = [q_W^{(t)}, q_S^{(t)}, q_E^{(t)}]$

The all $t \geq 0$, $\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)}\boldsymbol{M}$

$q_W^{(t)} =$ P(in state Work at time t)

$q_S^{(t)} =$ P(in state Surf at time t)

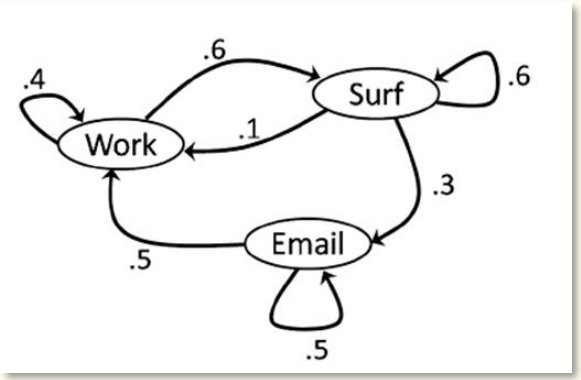
$q_E^{(t)} =$ P(in state Email at time t)

$$M = \begin{matrix} \\ W \\ S \\ E \end{matrix} \overset{\begin{matrix} W & S & E \end{matrix}}{\begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ 0.5 & 0 & 0.5 \end{bmatrix}}$$

$$\boldsymbol{q}^{(t)} = \boldsymbol{q}^{(0)}\boldsymbol{M}^t \text{ for all } t \geq 0$$

M^t as t grows

$$q^{(t)} = q^{(0)} M^t \text{ for all } t \geq 0$$



M

$$\begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ 0.5 & 0 & 0.5 \end{bmatrix}$$

M^2

$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .22 & .6 & .18 \\ .25 & .42 & .33 \\ .45 & .3 & .25 \end{pmatrix} \end{array}$$

M^3

$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .238 & .492 & .270 \\ .307 & .402 & .291 \\ .335 & .450 & .215 \end{pmatrix} \end{array}$$

M^{10}

$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .2940 & .4413 & .2648 \\ .2942 & .4411 & .2648 \\ .2942 & .4413 & .2648 \end{pmatrix} \end{array}$$

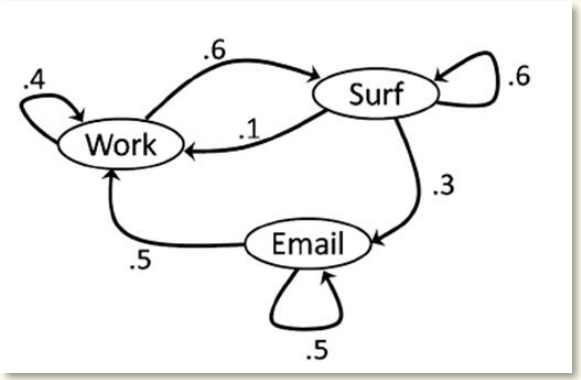
M^{30}

$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .29411764705 & .44117647059 & .26470588235 \\ .29411764706 & .44117647058 & .26470588235 \\ .29411764706 & .44117647059 & .26470588235 \end{pmatrix} \end{array}$$

M^{60}

$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .294117647058823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \end{pmatrix} \end{array}$$

M^t as t grows



$$\boldsymbol{q}^{(60)} = \boldsymbol{q}^{(0)}\,\boldsymbol{M}^{60}$$

$$\boldsymbol{q}^{(t)} = [q_W^{(t)}, q_S^{(t)}, q_E^{(t)}]$$

$q_W^{(t)} = P(\text{in state Work at time t})$

$q_S^{(t)} = P(\text{in state Surf at time t})$

$q_E^{(t)} = P(\text{in state Email at time t})$

$$[q_W^{(60)}, q_S^{(60)}, q_E^{(60)}] = [q_W^{(0)}, q_S^{(0)}, q_E^{(0)}] \cdot$$

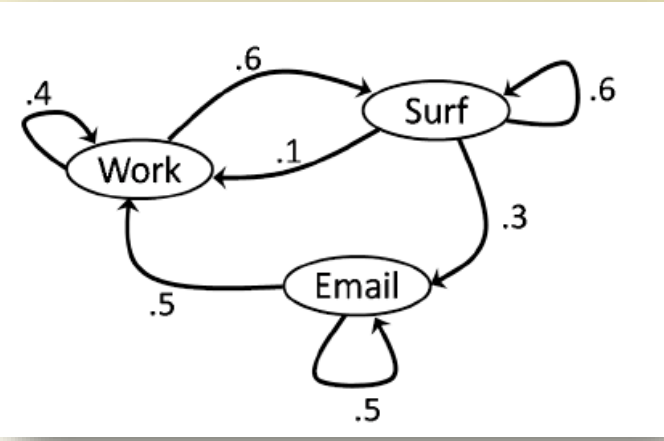
$$\begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .294117647058823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \end{pmatrix} \end{array}$$

$\forall \boldsymbol{q}^{(0)}$

$$[q_W^{(60)}, q_S^{(60)}, q_E^{(60)}] = [\ .294117647058823 \quad .441176470588235 \quad .264705882352941\ ]$$

Claim: This probability distribution over states is "stationary", meaning it never changes.

Observation



If $\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)}$ then it will never change again!

Proof: $\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)}\boldsymbol{M}$

$$\boldsymbol{q}^{(t+2)} = \boldsymbol{q}^{(t+1)}\boldsymbol{M} = \boldsymbol{q}^{(t)}\boldsymbol{M} = \boldsymbol{q}^{(t+1)}$$

$q_W^{(t)} = P(\text{in state Work at time t})$

$q_S^{(t)} = P(\text{in state Surf at time t})$

$q_E^{(t)} = P(\text{in state Email at time t})$

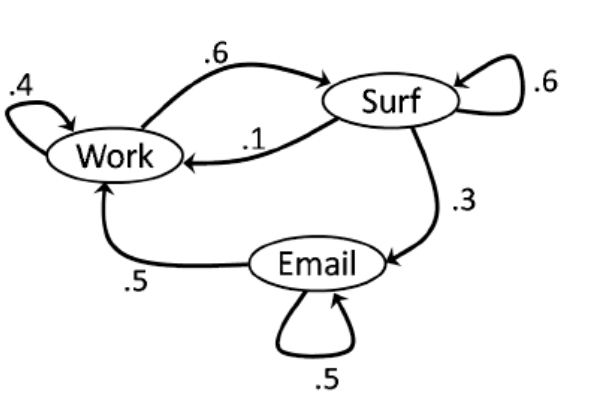
Such a $\boldsymbol{q}^{(t)}$ (that never changes) is called a **stationary distribution** and has a special name $\boldsymbol{\pi} = (\pi_W, \pi_S, \pi_E)$

It is the solution to $\boldsymbol{\pi} = \boldsymbol{\pi}\,\boldsymbol{M}$

Solving for Stationary Distribution

$$\boldsymbol{q}^{(t+1)} = \boldsymbol{q}^{(t)}\boldsymbol{M}$$

$$(\pi_W, \pi_S, \pi_E) = (\pi_W, \pi_S, \pi_E) \begin{pmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ 0.5 & 0 & 0.5 \end{pmatrix}$$



$$\pi_W = \pi_W \cdot 0.4 + \pi_S \cdot 0.1 + \pi_E \cdot 0.5$$

$$\pi_S = \pi_W \cdot 0.6 + \pi_S \cdot 0.6 + \pi_E \cdot 0$$

$$\pi_E = \pi_W \cdot 0 + \pi_S \cdot 0.3 + \pi_E \cdot 0.5$$

$$\pi_W + \pi_S + \pi_E = 1$$

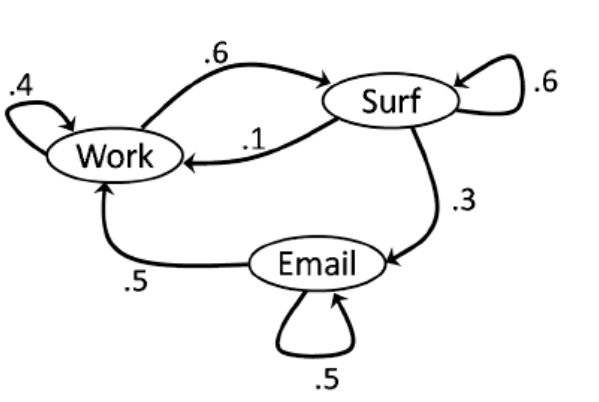
Solving for Stationary Distribution

$$(\pi_W, \pi_S, \pi_E) \begin{pmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ 0.5 & 0 & 0.5 \end{pmatrix} = (\pi_W, \pi_S, \pi_E)$$

$$\pi_W = \pi_W \cdot 0.4 + \pi_S \cdot 0.1 + \pi_E \cdot 0.5$$

$$\pi_S = \pi_W \cdot 0.6 + \pi_S \cdot 0.6 + \pi_E \cdot 0$$

$$\pi_E = \pi_W \cdot 0 + \pi_S \cdot 0.3 + \pi_E \cdot 0.5$$



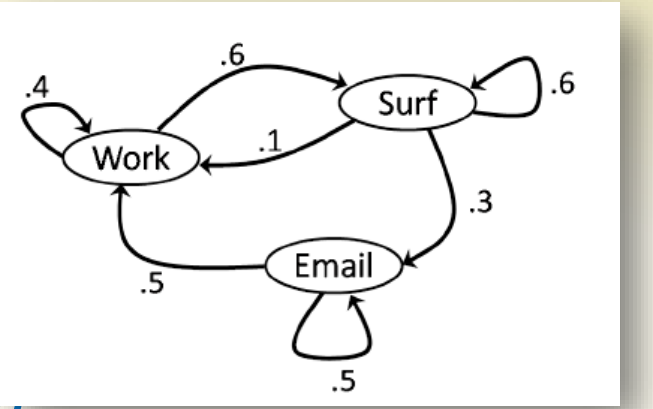
$$\pi_W + \pi_S + \pi_E = 1$$

$$\rightarrow \pi_W = \frac{10}{34}, \ \pi_S = \frac{15}{34}, \ \pi_E = \frac{9}{34}$$

As $t \to \infty$, $\boldsymbol{q}^{(t)} \to \boldsymbol{\pi}$ no matter what distribution $\boldsymbol{q}^{(0)}$ is !!

Markov Chains summary

$$M = \begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0.1 & 0.6 & 0.3 \\ 0.5 & 0 & 0.5 \end{bmatrix} \end{array}$$



- A set of n **states** $\{1, 2, 3, \ldots\ n\}$
- A square **transition matrix** $\boldsymbol{M}$, dimension $n \times\ n$

$$\boldsymbol{M}_{ij} = P(\text{transition from } i \text{ to } j)$$

- $\boldsymbol{M}^t{}_{ij}$ = Pr (in state j after t steps| start in state i).
- Nice Markov chains are not sensitive to initial distribution of states. $\boldsymbol{M}^t \rightarrow \boldsymbol{W}$, where all rows in $\boldsymbol{W}$ are the same probability vector $\boldsymbol{\pi}$
- A **stationary distribution** $\boldsymbol{\pi}$ is the solution to:

$\boldsymbol{\pi} = \boldsymbol{\pi}\, \boldsymbol{M}$, normalized so that $\Sigma_{i \in [n]} \pi_i = 1$

$$\boldsymbol{M}^{60} = \begin{array}{c} \\ W \\ S \\ E \end{array} \begin{array}{c} \begin{array}{ccc} W & S & E \end{array} \\ \begin{pmatrix} .294117647058823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \\ .294117647068823 & .441176470588235 & .264705882352941 \end{pmatrix} \end{array}$$

The Fundamental Theorem of Markov Chains

Theorem. Any nice* Markov chain

has a unique stationary distribution $\boldsymbol{\pi}$.

Moreover, as $t \to \infty$, for all i, j, $\lim_{t\to\infty} \boldsymbol{M}_{ij}^{t} = \pi_j$

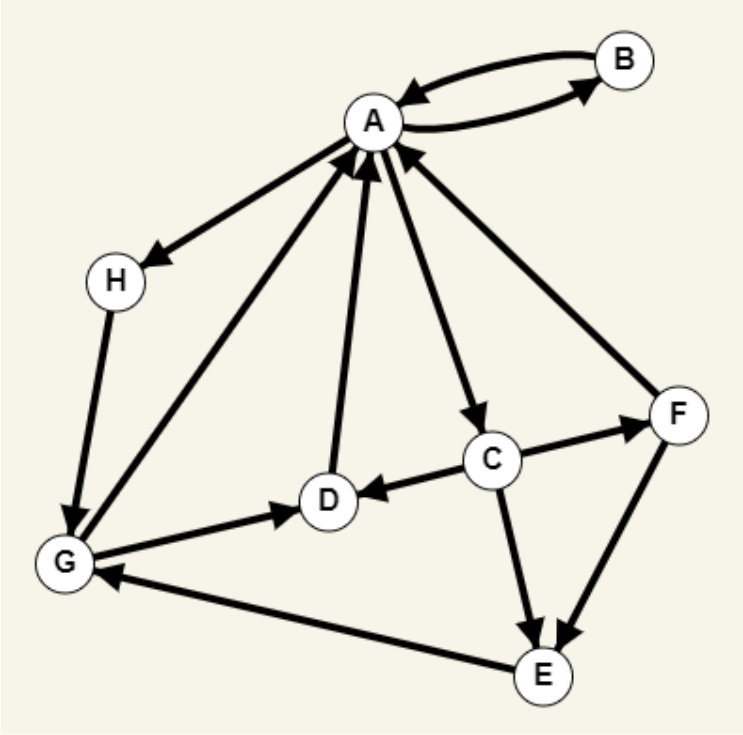
aperiodic and irreducible: these concepts are beyond us but they turn out to cover a very large class of Markov chains of practical importance.

One of these: the Markov chain is connected.

Another Example of a Markov Chain: Random Walks on Graph

Suppose we start at some node, and at each step transition to a neighboring node with each neighbor equally likely.

This is called a “random walk” on this graph.



Agenda

- Unbiased Estimation
- Markov Chains
- Application: PageRank (and the birth of Google)

PageRank: Some History

The year was 1997

- Bill Clinton in the White House
- Deep Blue beat world chess champion (Kasparov)

The Internet was not like it was today. Finding stuff was hard!

- In Nov 1997, only one of the top 4 search engines actually found itself when you searched for it

The Problem

Search engines worked by matching words in your queries to documents.

Not bad in theory, but in practice there are lots of documents that match a query.

- Search for ‘Bill Clinton’, top result is ‘Bill Clinton Joke of the Day’
- Susceptible to spammers and advertisers

The Fix: Ranking Results

- Start by doing filtering to relevant documents (with decent textual match).
- Then **rank** the results based on some measure of ‘quality’ or ‘authority’.

Key question: How to define ‘quality’ or ‘authority’?

Enter two groups:

- Jon Kleinberg (professor at Cornell)
- Larry Page and Sergey Brin (Ph.D. students at Stanford)

Both groups had the same brilliant idea

Larry Page and Sergey Brin (Ph.D. students at Stanford)

- Took the idea and founded Google, making billions

Jon Kleinberg (professor at Cornell)

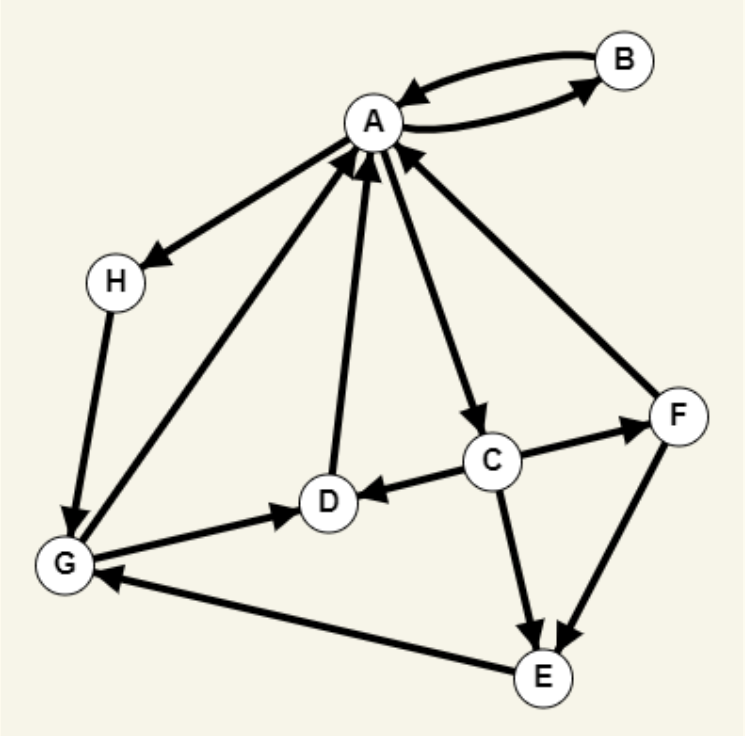
- MacArthur genius prize, Nevanlinna Prize and many other academic honors

PageRank - Idea

Take into account the directed graph structure of the web.

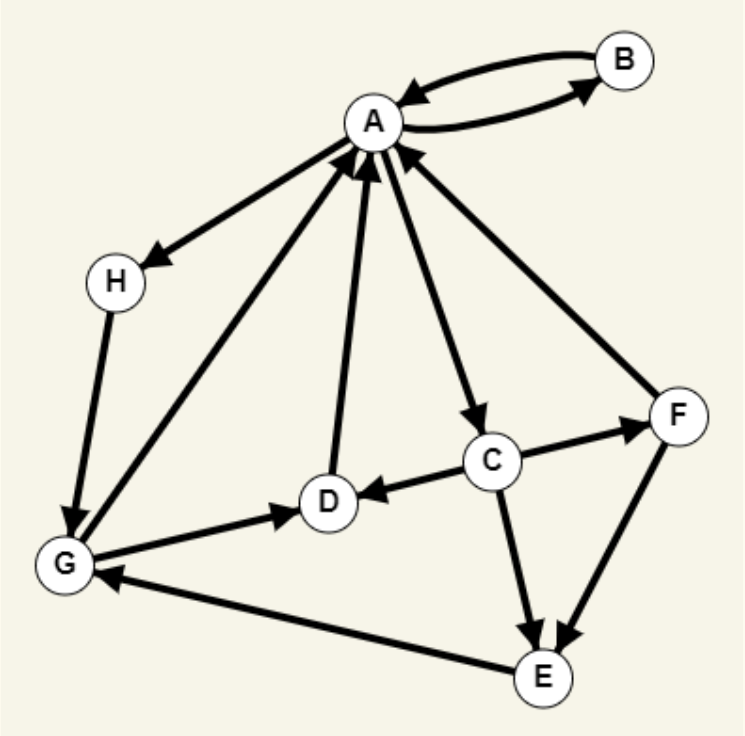
Use **hyperlink analysis** to compute what pages are high quality or have high authority.

Trust the Internet itself to define what is useful via its links.



PageRank - Idea

Idea 1 : Think of each link as a citation
"vote of quality"



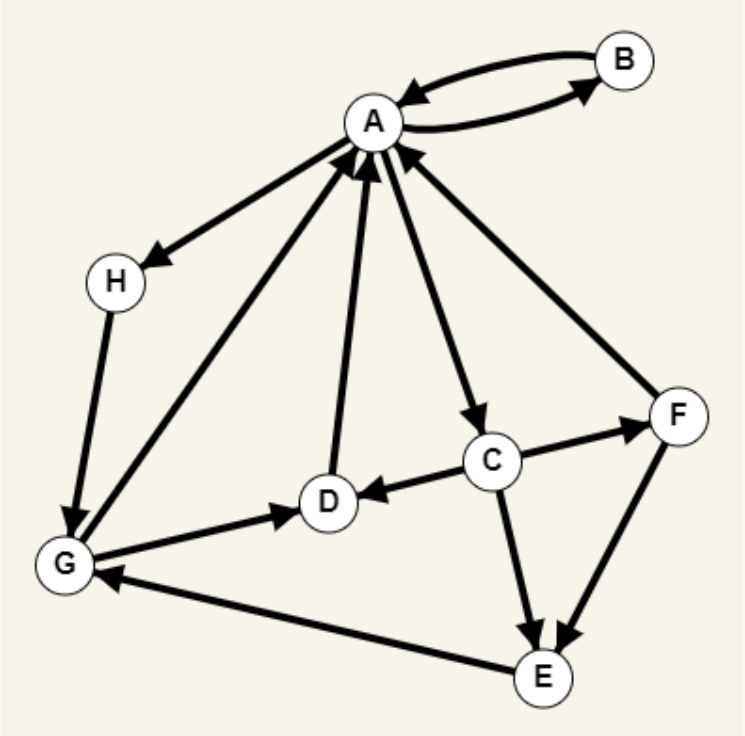
Rank pages by in-degree?

Two problems:

- Susceptible to spamming
- Doesn't take into account quality of link creator.

PageRank - Idea

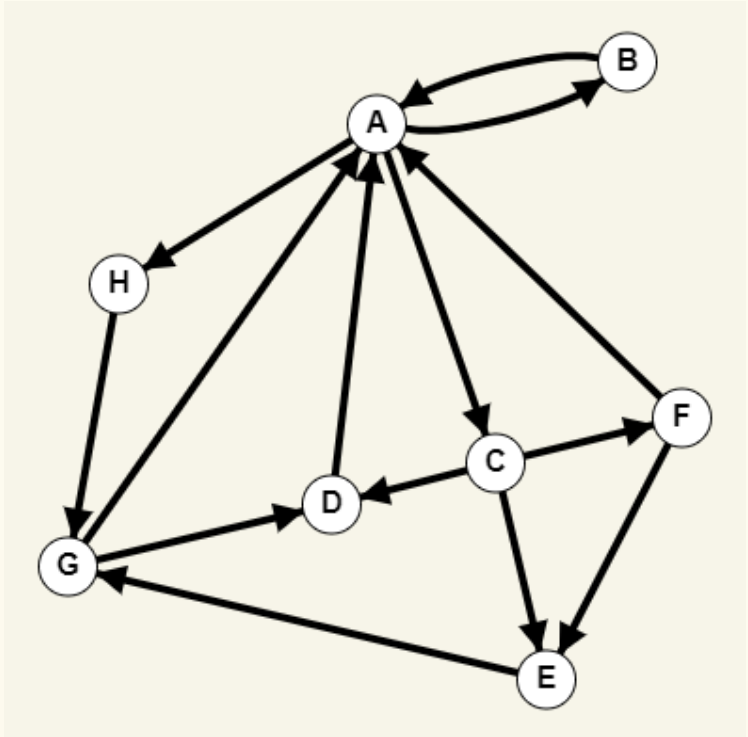
Idea 2 : Perhaps we should weight the links somehow and then use the weights of the in-links to rank pages



Inching towards PageRank

1. A web page has high quality if it's linked to by lots of high quality pages

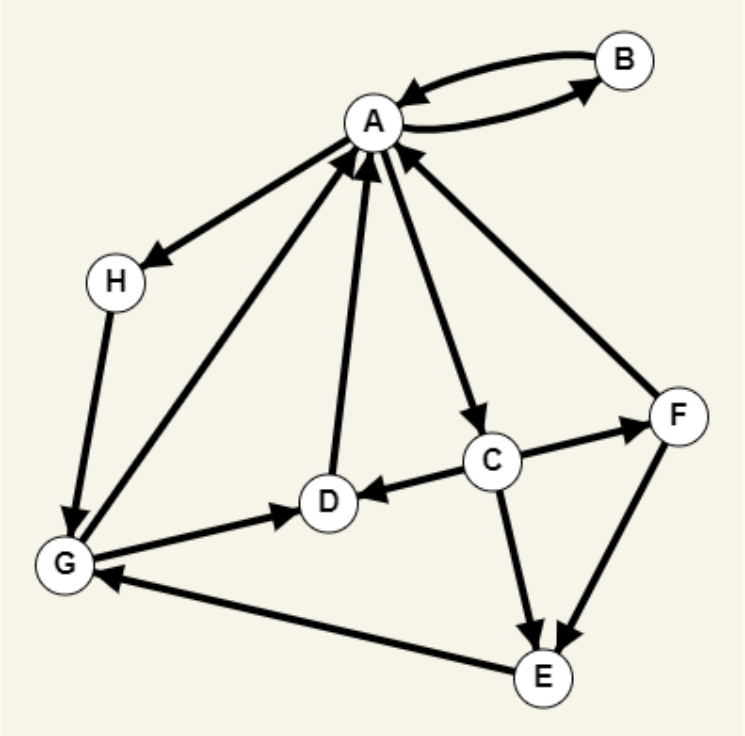
2. A web page is high quality if it links to lots of high quality pages



That's a recursive definition!

Inching towards PageRank

- If web page x has d outgoing links, one of which goes to y, this contributes $1/d$ to the importance/quality of y

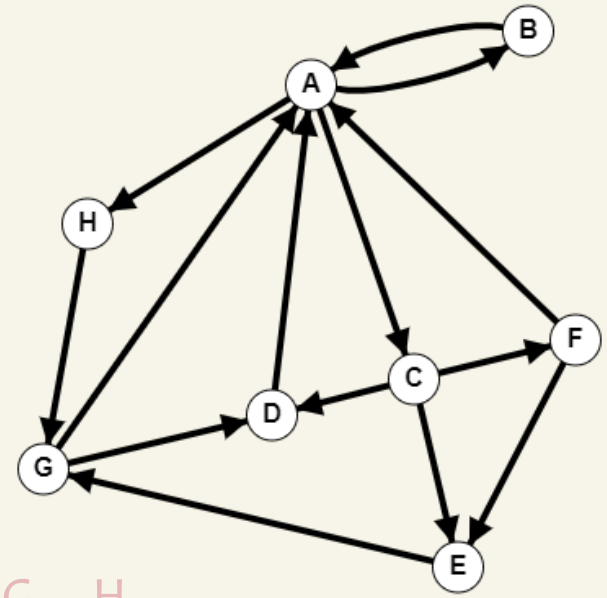


- But $1/d$ of what?

  We want to take into account the importance/quality of x too…

  …so it actually contributes $1/d$ of the importance/quality of x

Define q_x to be the quality of page x



$$q_A = q_B \cdot 1 + q_D \cdot 1 + q_F \cdot \frac{1}{2} + q_G \cdot \frac{1}{2}$$

$$(q_A, q_B, \dots, q_F, q_G, q_H) = (q_A, q_B, \dots, q_F, q_G, q_H)$$

Do you recognize this matrix as representing a Markov Chain we mentioned?

	A	B	C	D	E	F	G	H
A	0	$\frac{1}{3}$	$\frac{1}{3}$	0	0	0	0	$\frac{1}{3}$
B	1	0	0	0	0	0	0	0
C	0	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	0
D	1	0	0	0	0	0	0	0
E	0	0	0	0	0	0	1	0
F	$\frac{1}{2}$	0	0	0	$\frac{1}{2}$	0	0	0
G	$\frac{1}{2}$	0	0	$\frac{1}{2}$	0	0	0	0
H	0	0	0	0	0	0	1	0

This gives the following equations

Idea: Use the transition matrix $\boldsymbol{M}$ defined by a *random walk* on the web to compute quality of webpages.

Namely: Find $\boldsymbol{q}$ such that $\boldsymbol{qM = q}$ **Seem familiar?**

This is the stationary distribution for the Markov chain defined by a random web surfer

- Starts at some node (webpage) and randomly follows a link to another.
- Use stationary distribution of her surfing patterns after a long time as notion of quality

Issues with PageRank

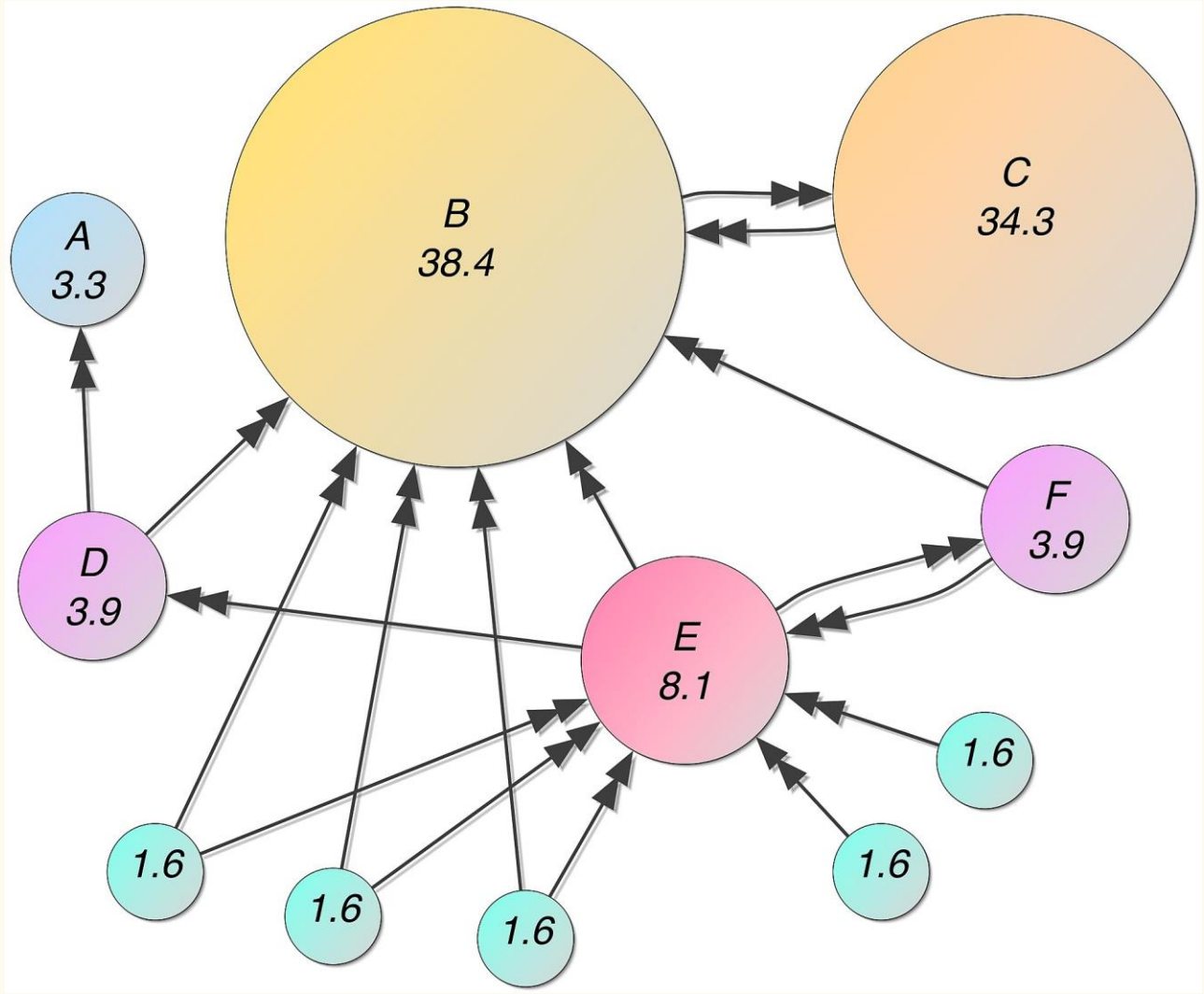
- How to handle dangling nodes (dead ends that don't link to anything) ?
- How to handle Rank sinks – group of pages that only link to each other ?

Both solutions can be solved by "teleportation"

Final PageRank Algorithm

1. Make a Markov Chain with one state for each webpage on the Internet with the transition probabilities $\boldsymbol{M}_{ij} = \frac{1}{outdeg(i)}$ if there is a link from i to j.
2. Then modify as follows. At each point in time if the surfer is at some webpage i:
   - If i has outlinks:
     - With probability p, take a step to one of the neighbors of i (equally likely)
     - With probability $1 - p$, "teleport" to a uniformly random page in the whole Internet.
   - Otherwise, always "teleport"
3. Compute stationary distribution $\boldsymbol{\pi}$ of this perturbed Markov chain.
4. Define the PageRank of a webpage i as the stationary probability π_i.
5. Find all pages with decent textual match to search and then order those pages by PageRank!

PageRank - Example



It Gets More Complicated

This basic algorithm was the **defining idea** that launched Google on their path to success, this is far from the end to optimizing search

Nowadays, Google and other web search engines have a LOT more secret sauce to rank pages, most of which they don't reveal 1) for competitive advantage and 2) to avoid gaming of their algorithms.

This slide was written a few years ago. Everything is changing now!

Still, PageRank is what launched Google!!!

From CSE312 to ChatGPT and Gemini: A Brief Glimpse

Caveats:

1. I am NOT an expert on this topic!
2. I am vastly oversimplifying.
3. Gemini and ChatGPT helped me with this lecture!

CSE 312 Spring 26

Lecture 26

The central object in a language model

$$\mathbb{P}(\text{next token} \mid \text{context})$$

- **Token:** A **word** or part of a word or punctuation.
- **Context:** prompt, previous words output and more

An entire response is generated one conditional probability at a time.

- You enter prompt
- LLM computes $\mathbb{P}(w_1 \mid \text{prompt, any other context})$.
- Samples from this distribution
- LLM computes $\mathbb{P}(w_2 \mid w_1, \text{prompt, any other context})$
- Samples from this distribution.
- And so on.

Distribution over outputs (by chain rule)

$$\mathbb{P}(w_1, w_2, \ldots, w_n \mid \text{prompt}, \text{context})$$
$$= \mathbb{P}(w_1 \mid \text{prompt}) \cdot \mathbb{P}(w_2 \mid w_1, \text{prompt}) \cdot \mathbb{P}(w_3 \mid w_1, w_2) \ldots \mathbb{P}(w_n \mid w_1, \ldots, w_{n-1})$$

This simple idea, scaled up with

- **huge data sets**
- **huge neural networks**
- **massive amounts of computation**

leads to ChatGPT, Gemini, etc.…

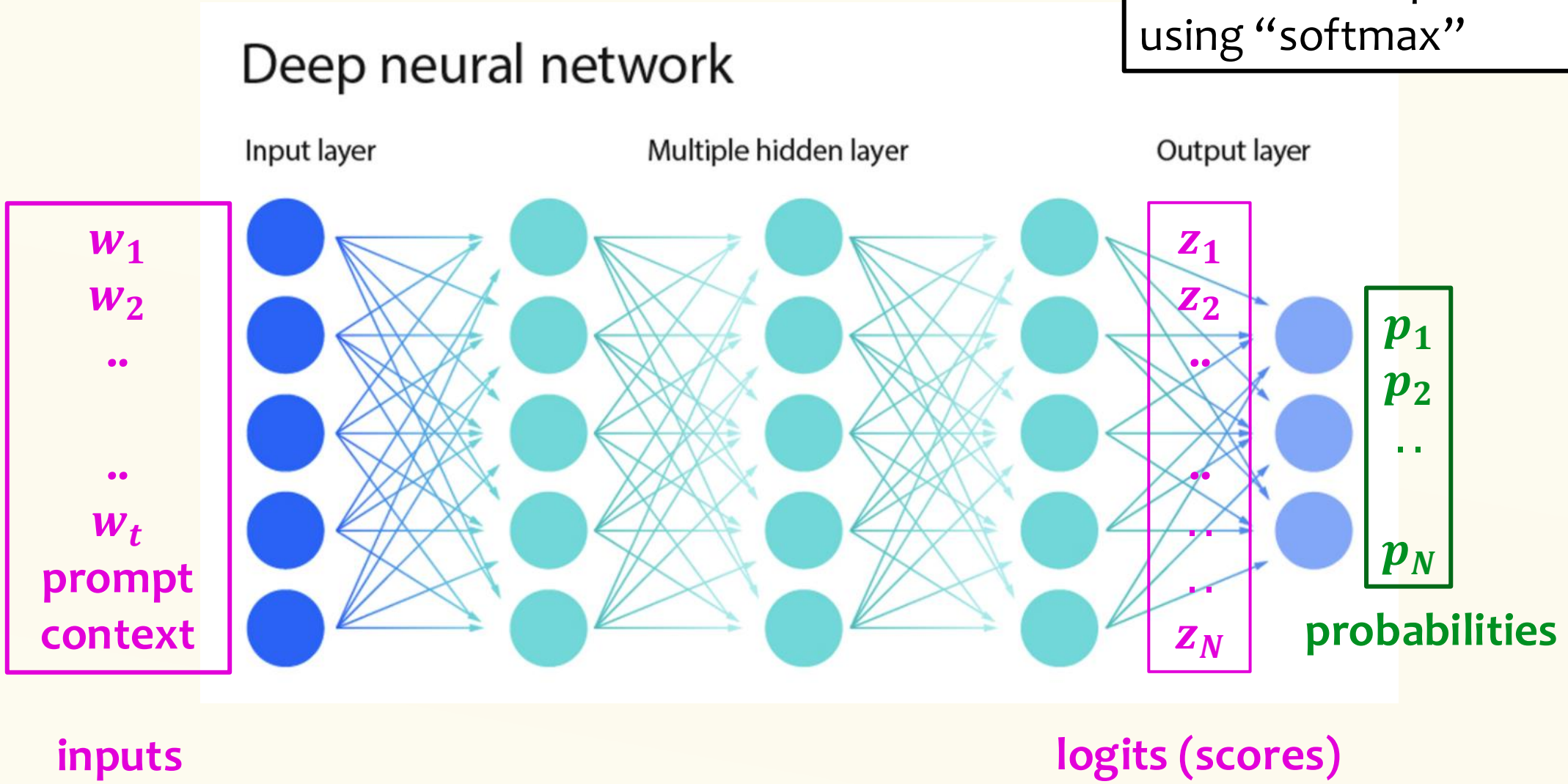
Some details to work out:

Where do these probabilities come from? What is the sampling process?
And much more.…

Where do the probabilities come from?

How does the LLM compute $\mathbb{P}(w_t \mid w_1, \ldots, w_{t-1}, \text{prompt})$

Logits are real numbers
Converted to probabilities using "softmax"



Logits -> probabilities via softmax

$\mathbb{P}$(next word | "The probabilities of all disjoint outcomes must sum to")

- $z_{\text{one}} = 2.0, \quad z_{\text{zero}} = 1.0, \quad z_{\text{infinity}} = 0.0, \quad z_{\text{pizza}} = -1.0$

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}$$

- $p_{\text{one}} = 0.64, \quad p_{\text{zero}} = 0.24, \quad p_{\text{infinity}} = 0.09, \quad p_{\text{pizza}} = 0.03.$

Logits -> Probabilities -> Sampling

- To allow for “tuning” of this probability distribution, a temperature parameter *T* is introduced

$$p_i = \frac{e^{z_i/T}}{\sum_{j=1}^{V} e^{z_j/T}}$$

Logits -> probabilities via softmax

$\mathbb{P}$(next word | "The probabilities of all disjoint outcomes must sum to")

$z_{\text{one}} = 2.0, \quad z_{\text{zero}} = 1.0, \quad z_{\text{infinity}} = 0.0, \quad z_{\text{pizza}} = -1.0$

$$p_i = \frac{e^{z_i/T}}{\sum_{j=1}^{V} e^{z_j/T}}$$

T = 1. $p_{\text{one}} = 0.64, \quad p_{\text{zero}} = 0.24, \quad p_{\text{infinity}} = 0.09, \quad p_{\text{pizza}} = 0.03.$

T = 0.5. $p_{\text{one}} = 0.87, \quad p_{\text{zero}} = 0.12, \quad p_{\text{infinity}} = 0.015, \quad p_{\text{pizza}} = 0.001.$

Logits -> Probabilities -> Sampling

- To allow for "tuning" of this probability distribution, a temperature parameter T is introduced

$$p_i = \frac{e^{z_i/T}}{\sum_{j=1}^{V} e^{z_j/T}}$$

- What happens when increase the temperature?
  - distribution becomes uniform!
- What happens when decrease the temperature?
  - Pushes probability towards more likely words

How and when is the temperature set?

Determined on a prompt by prompt basis!

Prompt -> intent router before entering neural network, where various hyperparameters are set, including temperature

- Low temp (0.0-0.2): when it wants model to act deterministically and pick highest probability tokens to avoid logic errors and hallucinations
  - Math problems, coding problems, translation, factual questions
- Medium temp (0.4-0.7): conversational default to get natural, varied vocab without losing factuality
  - Summarize article, write email, explain a concept
- High temp (0.8 and above): flattens prob dist to encourage less likely tokens -> generates more creative and unpredictable text.
  - Ask for poem or fictional story or brainstorm ideas

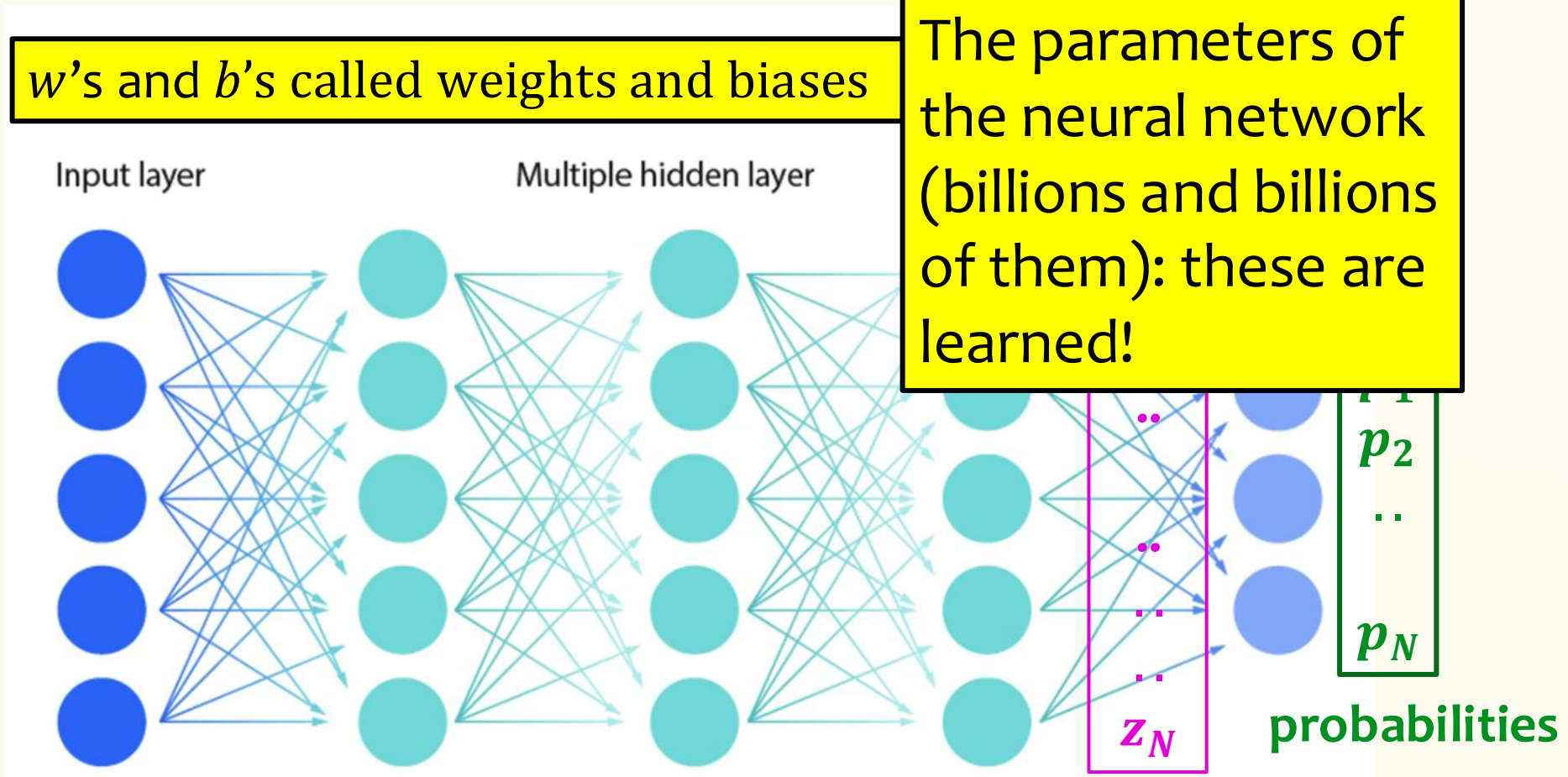
Back to the neural network: what's going on inside

Typical node: takes inputs $x_1, \ldots, x_k$ and outputs

$f(w_1 x_2 + \cdots + w_k x_k + b)$ where f is a non − linear "activation" function.

w's and b's called weights and biases

The parameters of the neural network (billions and billions of them): these are learned!



Training dataset: massive corpus

Pre-training

- For months and months, collect all the text they can: Wikipedia, books, articles, code, Internet forums
- Goal: is to set the parameters of the network to maximize the likelihood of seeing the text in the training set

$$\mathcal{L}(\text{corpus};\ \theta)\ = \prod_{\text{i=1}}^{\text{N}} \mathbb{P}_\theta(\ \text{token}\ i\ \mid \text{previous tokens}\ )$$

Actually, minimize negative log likelihood

$$-\ln \mathcal{L}(\text{corpus};\ \theta)\ = -\sum_{i=1}^{N} \mathbb{P}_\theta(\ \text{token}\ i\ \mid \text{previous tokens}\ )$$

The Pre-training Loop

Repeat billions of times:

- Read a chunk of text from the Internet
- Use the current weights to guess the probability of the next word
- Check what the actual next word was
- Calculate the negative log likelihood of seeing that word.
- Use calculus (“Backpropogation”) to figure out which direction to nudge every single weight and bias to make that specific word slightly more probable next time.

When done: you have a Base Model! (pre-trained)

BUT… not useful yet!

Making the model useful/helpful

- Maximize human preference rather than pure likelihood.

- How to learn human preference?
- Ask humans!

The helpfulness problem

Suppose one of you asks our 312 GPT for help with a problem.

The assistant generates two possible hints:

- Hint A: Start by identifying the sample space. Then define the event you are counting.
- Hint B: The answer is $\binom{52}{5}$

Different humans will differ on which hint they prefer.

Helpfulness is not objective.

The Bradley-Terry Model

Basic assumption: for every possible response, there is a hidden "helpfulness score", we call it R (for reward) – could be positive or negative.

Unknown to us.

Can learn it based on preferences expressed by human.

If we are comparing two responses A and B,

R_A = hidden score of response A, . R_B = hidden score of response B

Assume that $\mathbb{P}(A \succ B) = \frac{e^{R_A}}{e^{R_A} + e^{R_B}}$.

Use MLE to estimate the parameters from the preferences expressed by humans!

Reinforcement Learning from Human Feedback (RLHF) workflow

- Companies hire armies of human contractors.
  - Sit at computers, read a prompt, read two AI-generated responses and click “A is better” or “B is better”, thousands of times a day.
- Train a Reward Model.
  - If 10 people rate a particular pair of responses, and 8 people prefer A, say $\mathcal{L}(R_A, R_B) = \left(\frac{e^{R_A}}{e^{R_A}+e^{R_B}}\right)^8 \left(\frac{e^{R_B}}{e^{R_A}+e^{R_B}}\right)^2$
  - Find the parameters R_A, R_B that make our observed data most likely
  - Once trained, it can estimate how helpful a human would find any new text.

Reinforcement Learning step

Bring Reward Model back to Language Model:

“Your new goal in life is to get the highest score possible from the Reward Model.”

- AI generates thousands of responses.
- Reward Model grades them.
  - When it gets a high score, it updates internal parameters to behave that way more often.
  - When it gets a low score, adjusts parameters to avoid that behavior.

Some takeaways

- A language model repeatedly estimates a conditional distribution of the form: $\mathbb{P}(\text{next token} \mid \text{previous tokens})$
- The chain rule for probability is the mathematical reason we can assign probabilities to whole sequences one token at a time.
- Maximum likelihood training rewards a model for assigning high probability to the actual observed next token and to text that humans find helpful.
- Sampling from a distribution explains why LLM outputs can be useful, creative, variable and sometimes wrong (aka hallucinations).

Notes

- Prompts matter! Bayes Rule tells us that evidence changes beliefs!

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$

$$\mathbb{P}(\text{spam}|\text{contains }\text{"free"}) = \frac{\mathbb{P}(\text{contains }\text{"free"}|\text{spam})\mathbb{P}(\text{spam})}{\mathbb{P}(\text{contains }\text{"free"})}$$

- In an LLM the prompt acts like evidence.
- Compare:
  - Explain induction
  - Explain induction in a probability class, assuming students know binomial coefficients but have not seen sigma-algebras.

- Second prompt changes conditional distribution over answers.

Removing ambiguity

If a prompt is ambiguous, we see how that manifests in Pr(answer| prompt) via the law of total probability.

Example: Tell me about Mercury

- H1 = the planet
- H2 = the chemical element
- H3 = the Roman god
- H4 = the car brand.

$$\mathbb{P}(\text{answer} \mid \text{prompt}) = \sum_{h} \mathbb{P}(\text{answer} \mid \text{H} = \text{h},\ \text{prompt})\mathbb{P}(\text{H} = \text{h}, \text{prompt})$$