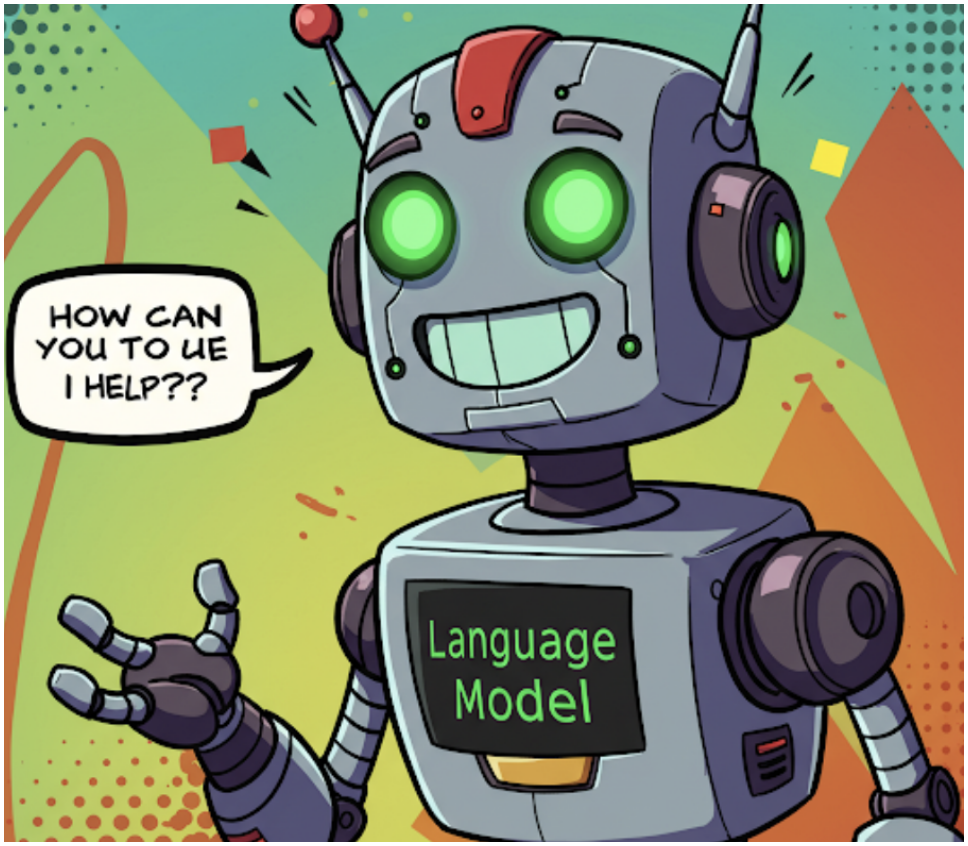
AI and ML Bootcamp Session One: Intro to ML



Chris Jermaine

Rice University

Part 1

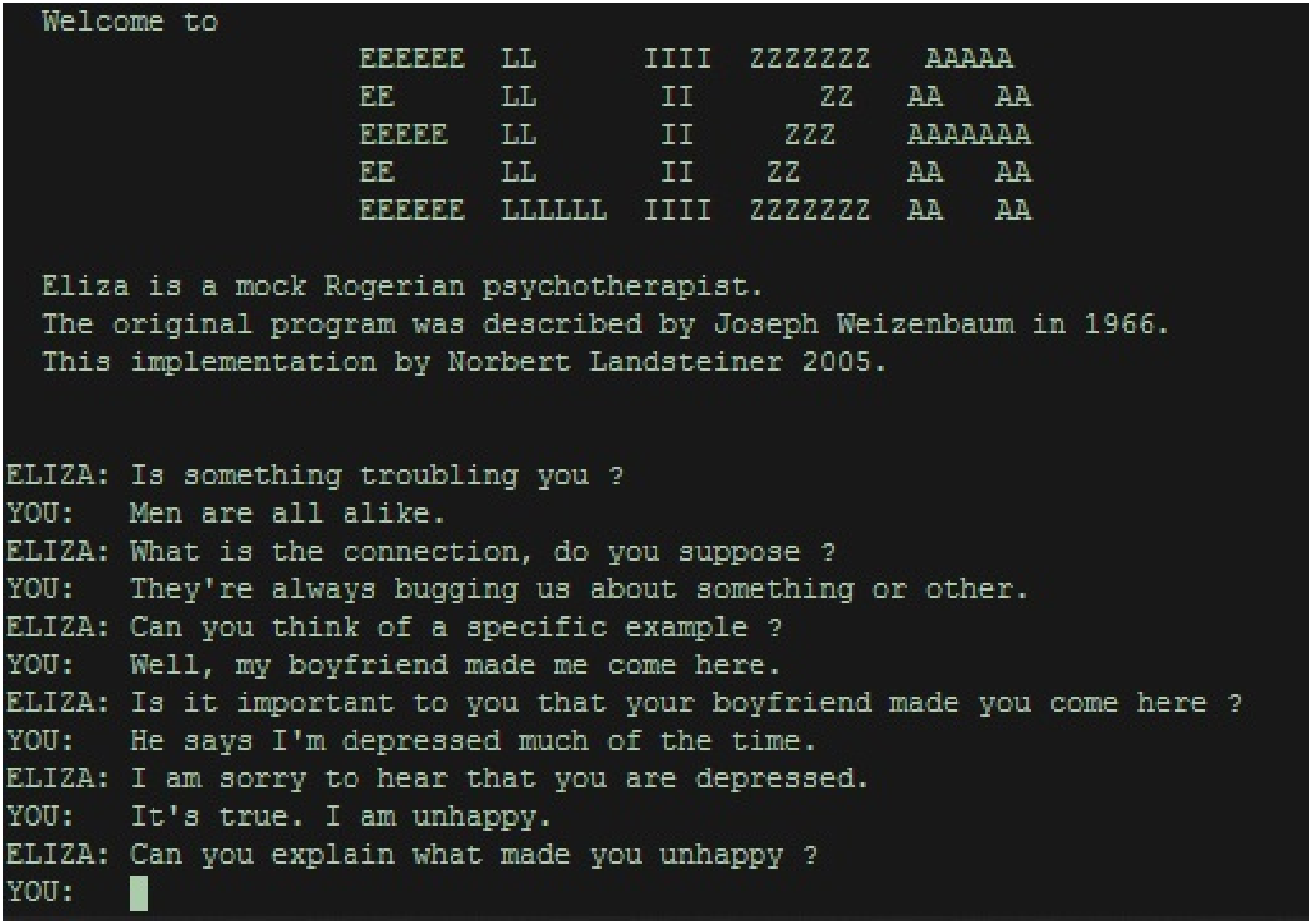
- Case study: text generation (2010-present)
- Includes a semi-technical deep-dive into modern transformers
  - Sructure inspired by a tutorial by colleague Vicente Ordonez
- Then a few key recent ideas:
  - Reinforcement learning
  - Chain-of-thought prompting
  - DeepSeek and mixtures-of-experts

Part 2

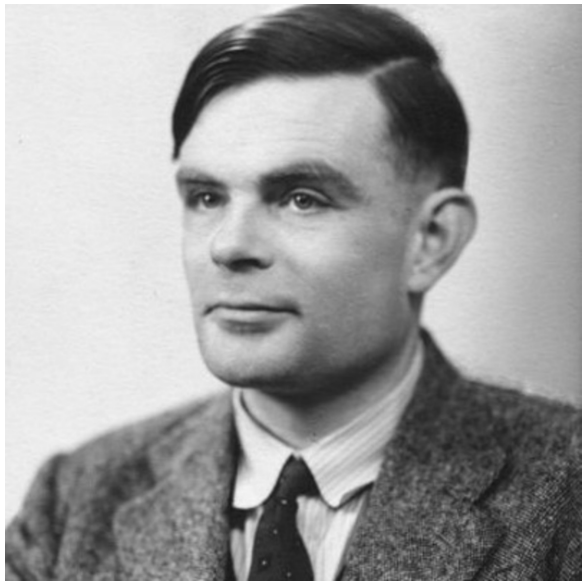
- ML primer and general backgrond
- Topics include:
  - What is supervised learning?
  - Loss functions
  - Gradient descent
  - Measuring accurcy in ML
  - Over-fitting

Pre-2010 the Idea of True AI Was Laughable

- By 2000 we were not much past Eliza: 1964-1966 MIT AI Lab



The Turing Test



- Eliza: first serious attempt to pass the "Turing Test"
  - Based on Alan Turing's 1950 paper "Computing Machinery and Intelligence"
  - Turing Test: three "players"
  - (1) Interrogator, (2) Computer, (3) Human
  - Interrogator passes written questions to both
  - Tries to determine which one is human
- Eliza: Not close to passing the Turing Test!

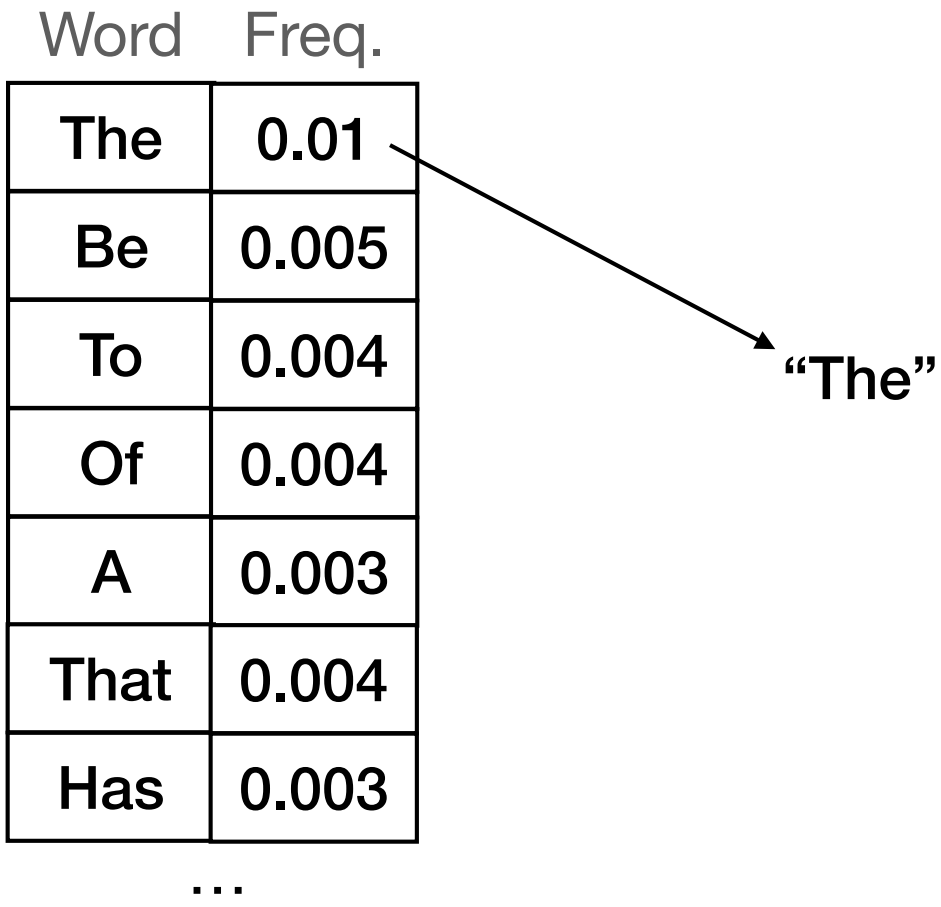
But By Early 2010's It Did not Seem so Crazy

- N-Grams: serious attempt to use big data for generative AI

Word	Freq.
The	0.01
Be	0.005
To	0.004
Of	0.004
A	0.003

2-Gram	Freq.
The bird	0.00025
The house	0.00037
The big	0.00016
The fat	0.00015
The man	0.00028

3-Gram	Freq.
The fat sandwich	0.0000
The fat bird	0.0000
The fat cat	0.0001
The fat dog	0.0001
The fat suitcase	0.0000


Word
Freq.
The
0.01
Be
0.005
To
0.004
Of
0.004
A
0.003
That
0.004
Has
0.003
…
“The”

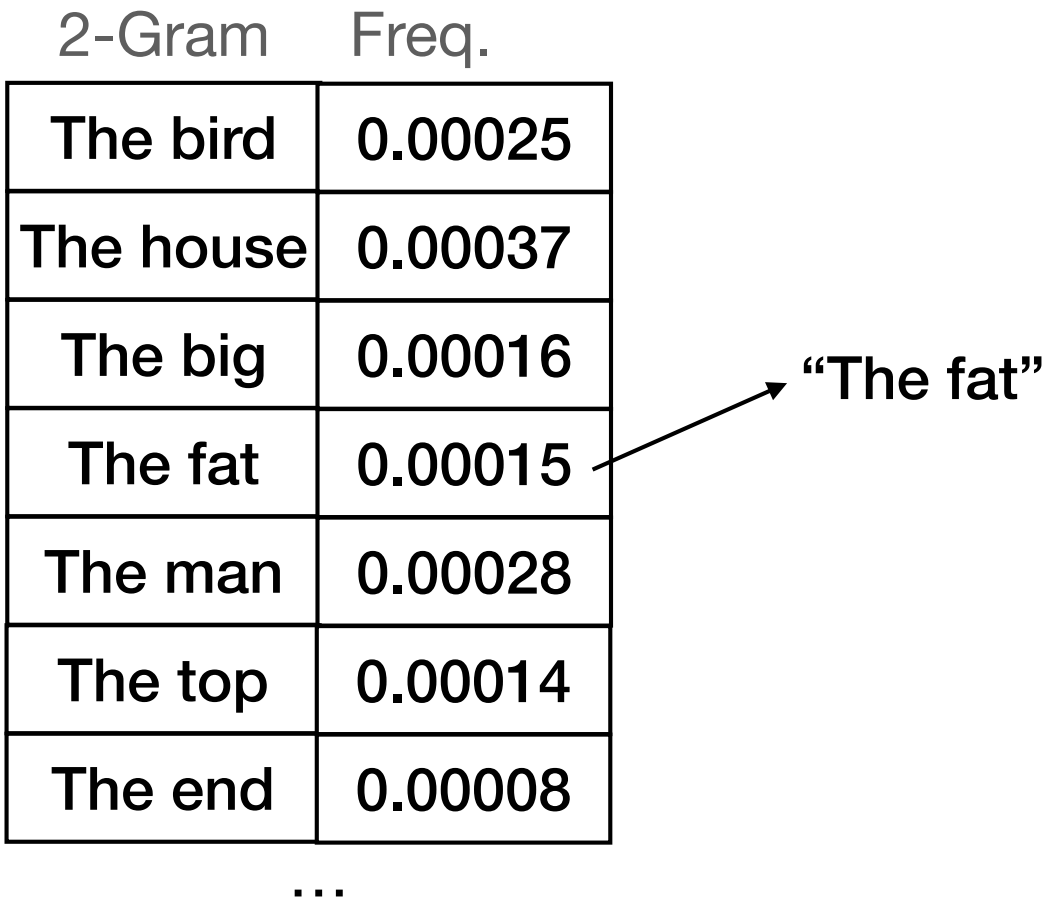
2-Gram	Freq.
The bird	0.00025
The house	0.00037
The big	0.00016
The fat	0.00015
The man	0.00028
The top	0.00014
The end	0.00008

...

Pr [“The Bird” | “The...”] =

$$\frac{0.00025}{0.01} = 0.025$$

“The”


2-Gram
Freq.
The bird
0.00025
The house
0.00037
The big
0.00016
The fat
0.00015
The man
0.00028
The top
0.00014
The end
0.00008
...
“The fat”

3-Gram	Freq.
The fat sandwich	0.00005
The fat bird	0.00008
The fat cat	0.00010
The fat dog	0.00011
The fat suitcase	0.00008
The fat is	0.00002
The fat of	0.00003

…

“The fat”

3-Gram	Freq.
The fat sandwich	0.00005
The fat bird	0.00008
The fat cat	0.00010
The fat dog	0.00011
The fat suitcase	0.00008
The fat is	0.00002
The fat of	0.00003

…

4-Gram	Freq.
The fat cat sat	0.000008
The fat cat slept	0.000009
The fat cat grinned	0.000001
The fat cat purred	0.000008
The fat cat ate	0.000006
The fat cat sipped	0.000002
The fat cat dozed	0.000004

…

“The fat cat”

4-Gram	Freq.
The fat cat sat	0.000008
The fat cat slept	0.000009
The fat cat grinned	0.000001
The fat cat purred	0.000008
The fat cat ate	0.000006
The fat cat sipped	0.000002
The fat cat dozed	0.000004

…

4-Gram	Freq.
Fat cat slept soundly	0.0000006
Fat cat slept peacefully	0.0000007
Fat cat slept heavily	0.0000002
Fat cat slept all	0.0000007
Fat cat slept on	0.0000004
Fat cat slept in	0.0000006
Fat cat slept through	0.0000008

…

“The fat cat slept”

Maximum context length is 3

4-Gram	Freq.
Fat cat slept soundly	0.0000006
Fat cat slept peacefully	0.0000007
Fat cat slept heavily	0.0000002
Fat cat slept all	0.0000007
Fat cat slept on	0.0000008
Fat cat slept in	0.0000006
Fat cat slept through	0.0000008

…

4-Gram	Freq.
Cat slept on the	0.00000006
Cat slept on my	0.00000001
Cat slept on a	0.00000002
Cat slept on top	0.00000005
Cat slept on her	0.00000003
Cat slept on his	0.00000002
Cat slept on your	0.00000001

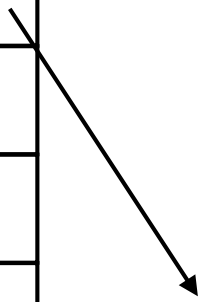
…

“The fat cat slept on”

Maximum context length is 3

4-Gram	Freq.
Cat slept on the	0.00000006
Cat slept on my	0.00000001
Cat slept on a	0.00000002
Cat slept on top	0.00000005
Cat slept on her	0.00000003
Cat slept on his	0.00000002
Cat slept on your	0.00000001

…



“The fat cat slept on the”

Web 1T 5-gram Version 1

Item Name:	Web 1T 5-gram Version 1
Author(s):	Thorsten Brants, Alex Franz
LDC Catalog No.:	LDC2006T13
ISBN:	1-58563-397-6
ISLRN:	831-344-220-094-6
DOI:	https://doi.org/10.35111/cqpa-a498
Release Date:	September 19, 2006
Member Year(s):	2006
DCMI Type(s):	Text
Data Source(s):	web collection
Application(s):	language modeling
Language(s):	English
Language ID(s):	eng
License(s):	Web 1T 5-gram Version 1 Agreement
Online Documentation:	LDC2006T13 Documents
Licensing Instructions:	Subscription & Standard Members, and Non-Members
Citation:	Brants, Thorsten, and Alex Franz. Web 1T 5-gram Version 1 LDC2006T13. Web Download. Philadelphia: Linguistic Data Consortium, 2006.

Powered by Big Data

Web 1T 5-gram Version 1

Item Name:	Web 1T 5-gram Version 1
Author(s):	Thorsten Brants, Alex Franz
LDC Catalog No.:	LDC2006T13
ISBN:	1-58563-397-6
ISLRN:	831-344-220-094-6
DOI:	https://doi.org/10.35111/cqpa-a498
Release Date:	September 19, 2006
Member Year(s):	2006
DCMI Type(s):	Text
Data Source(s):	web collection
Application(s):	language modeling
Language(s):	English
Language ID(s):	eng
License(s):	Web 1T 5-gram Version 1 Agreement
Online Documentation:	LDC2006T13 Documents
Licensing Instructions:	Subscription & Standard Members, and Non-Members
Citation:	Brants, Thorsten, and Alex Franz. Web 1T 5-gram Version 1 LDC2006T13. Web Download. Philadelphia: Linguistic Data Consortium, 2006.

Tokens	1,024,908,267,229
Sentences	95,119,665,584
Unigrams	13,588,391
Bigrams	314,843,401
Trigrams	977,069,902
Fourgrams	1,313,818,354
Fivegrams	1,176,470,663

Lasting Insight of N-Grams

- It's all about next-word prediction
- If I ingest a lot of data...
  - ...and build a high quality statistical model for the next word
- Maybe we can begin to approximate human intelligence?
  - Not obvious that this should be the case!

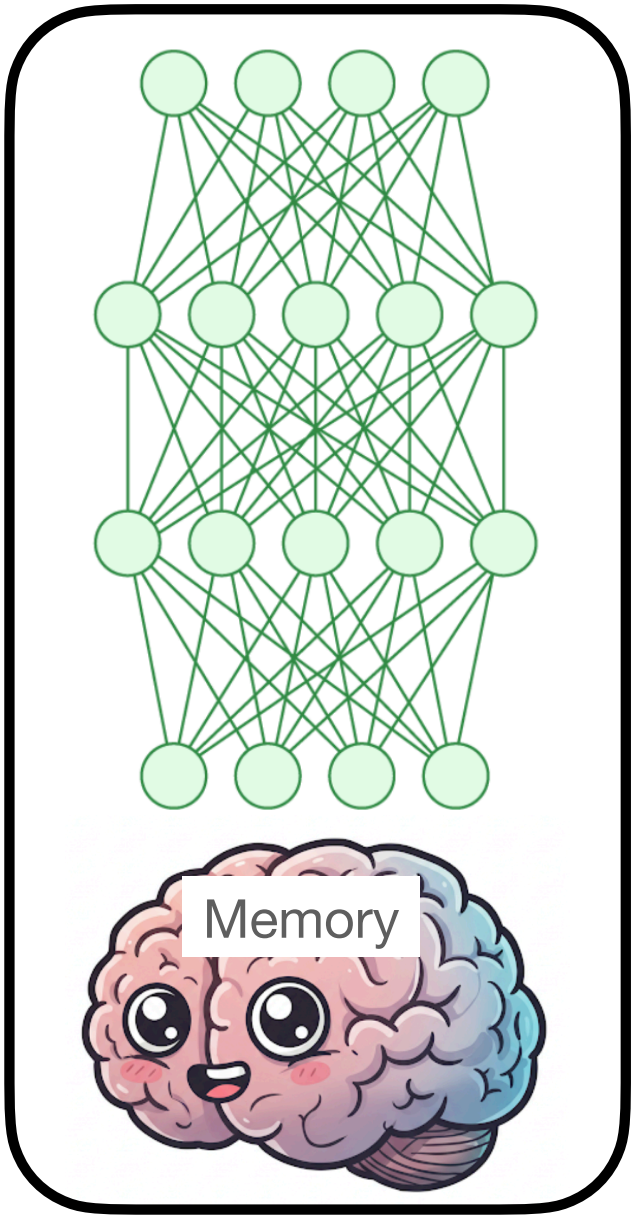
Problem with N-Grams: They Forget

- "The fat cat slept on the mat then got up to go to the..."
- Most common 4-grams are:
  - "go to the store"; "go to the park"; "go to the beach"
- But none make sense when you are talking about a fat cat!
- Past 6-Grams or 7-Grams data are too sparse to be useful

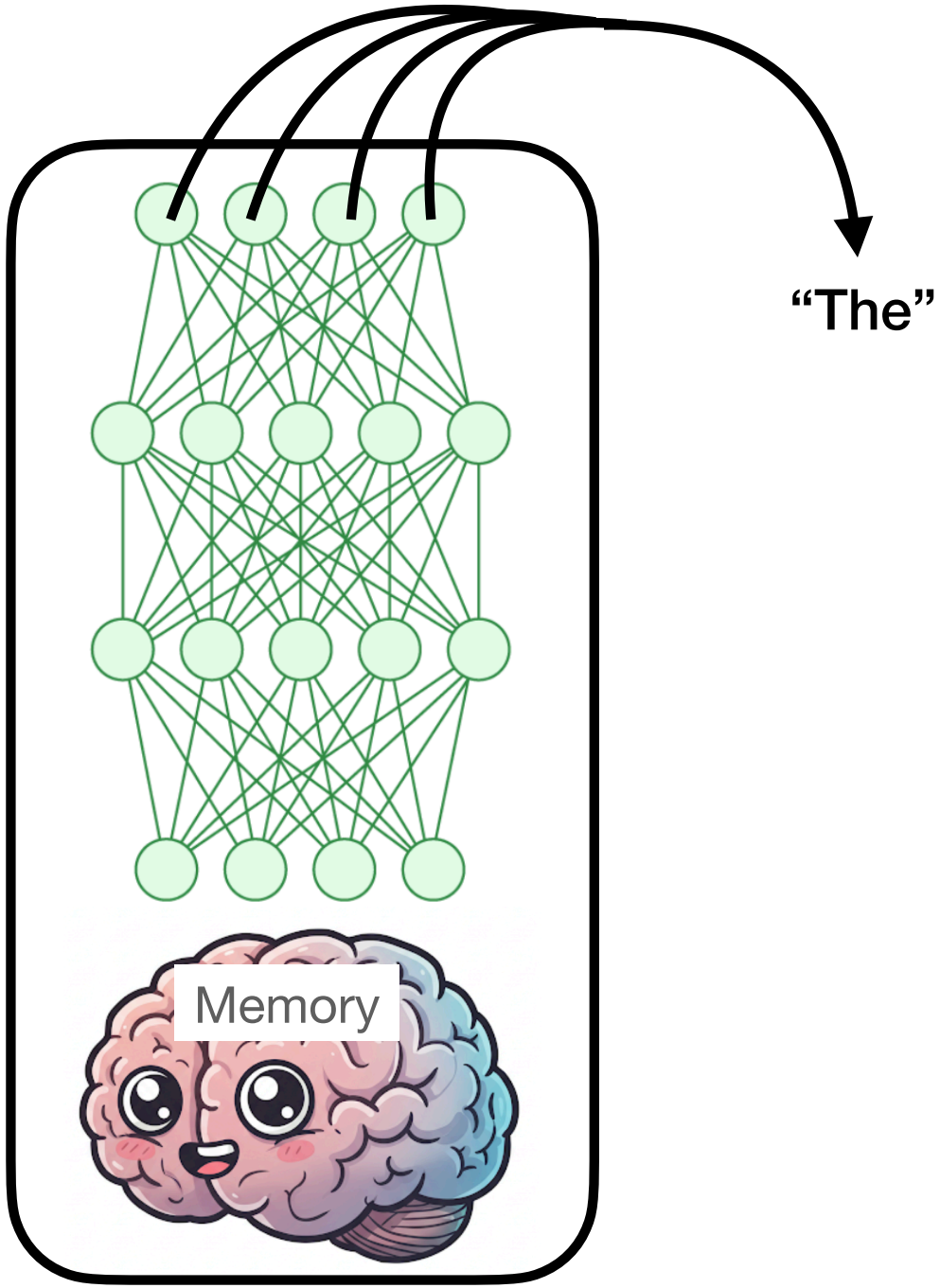
The Rise of Neural Networks

- By mid-2010's neural networks had revolutionized machine perception
  - In the year 2000 NN's were kind of a joke
  - Big data (the Internet) and GPUs changed that
- Might they be useful for language?
- People became very interested in RNNs

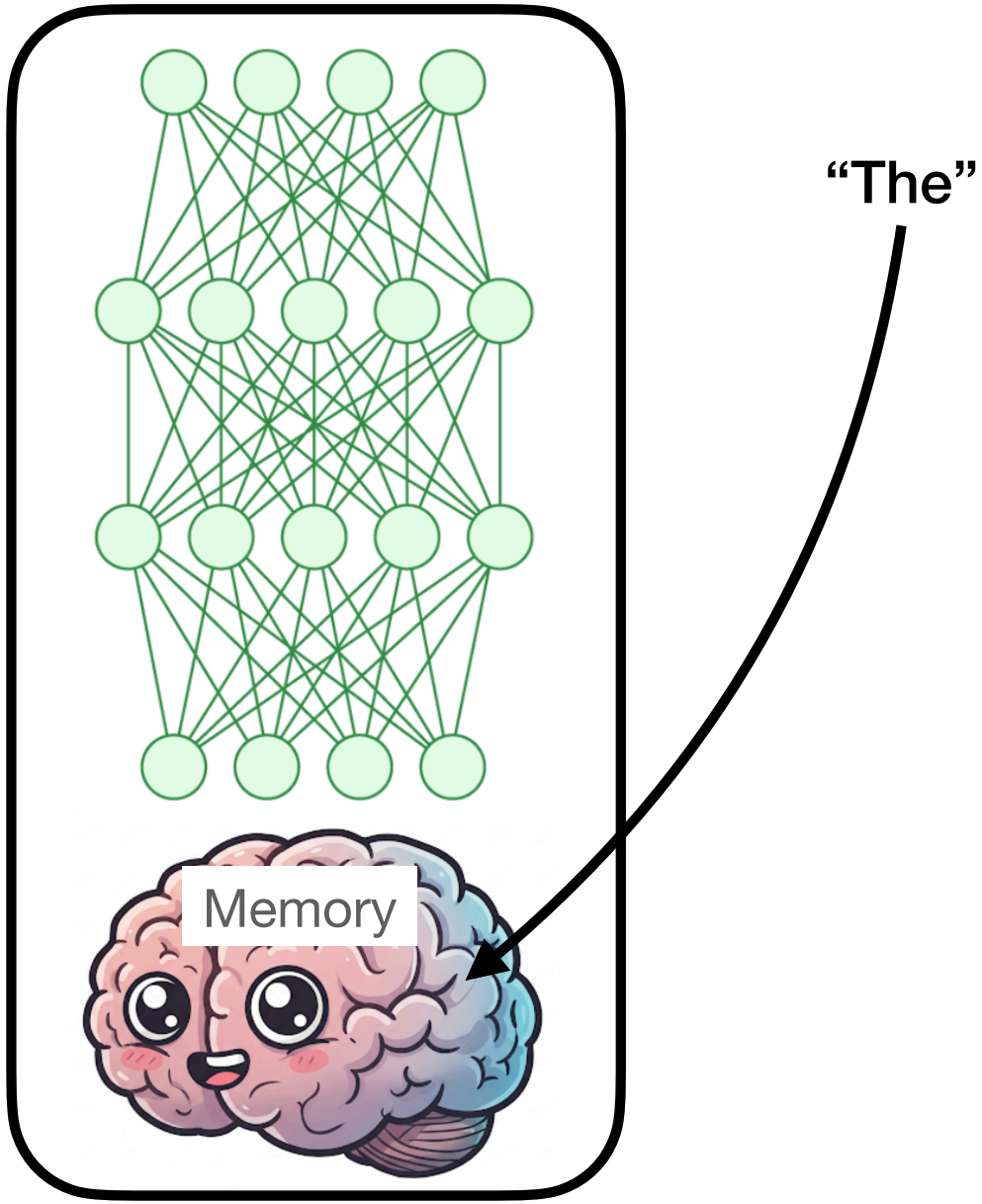
Basic RNN Architecture



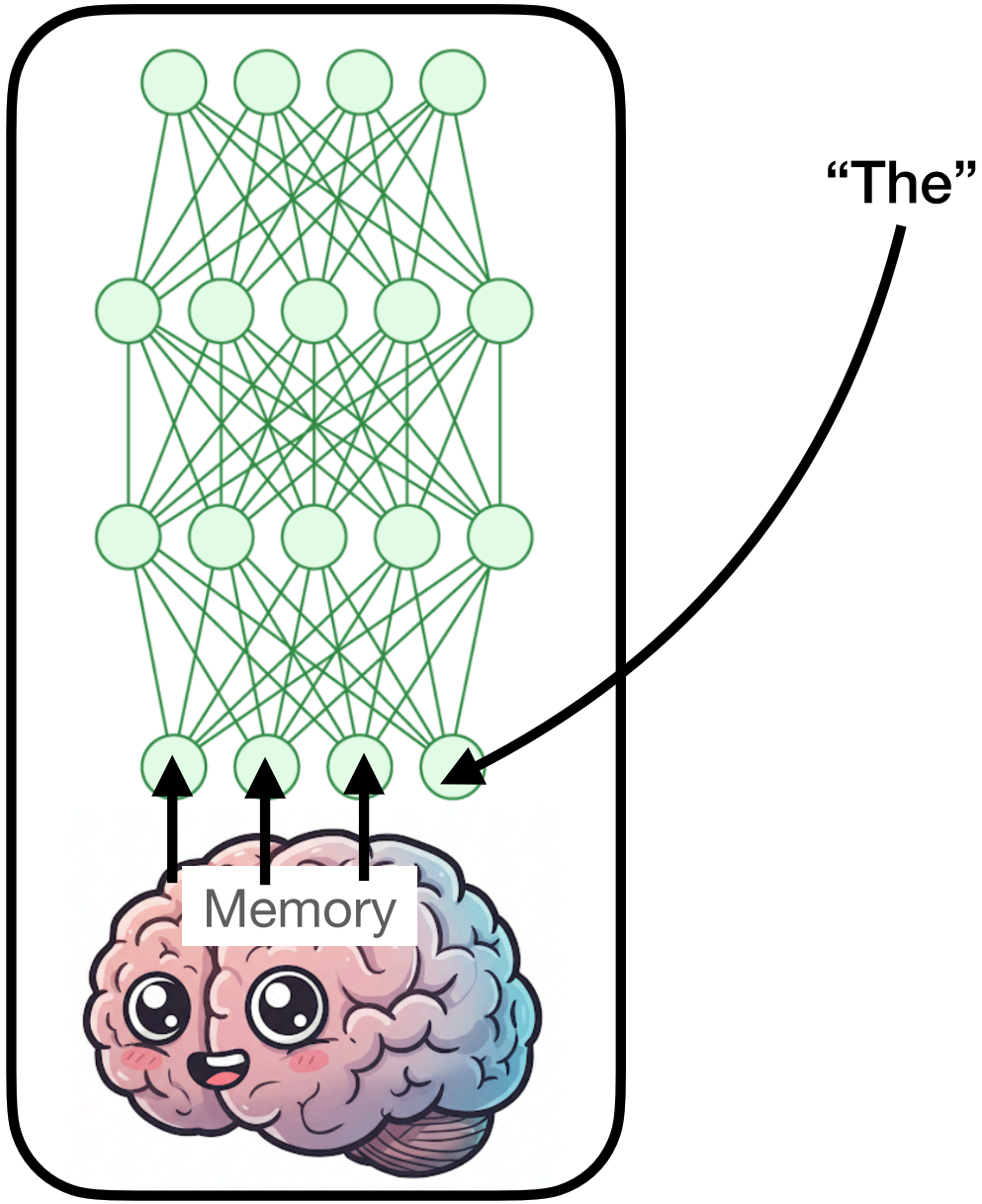
(1) Generate



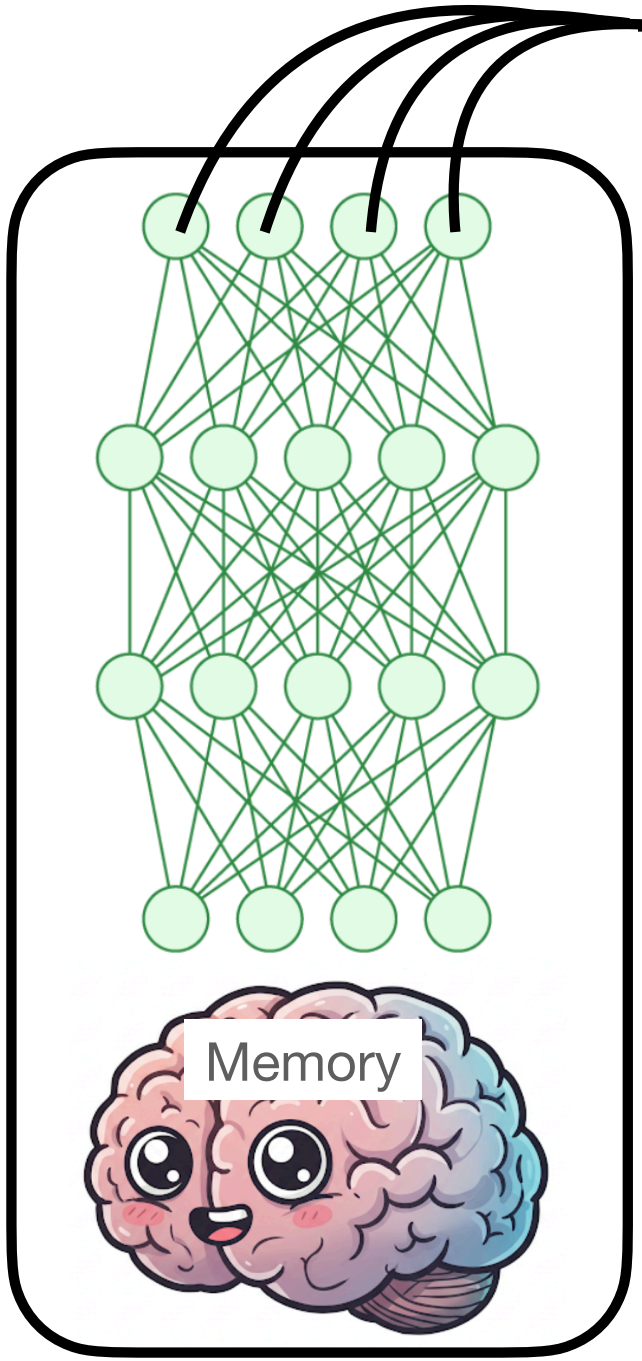
(2) Memorize



(3) Recurse

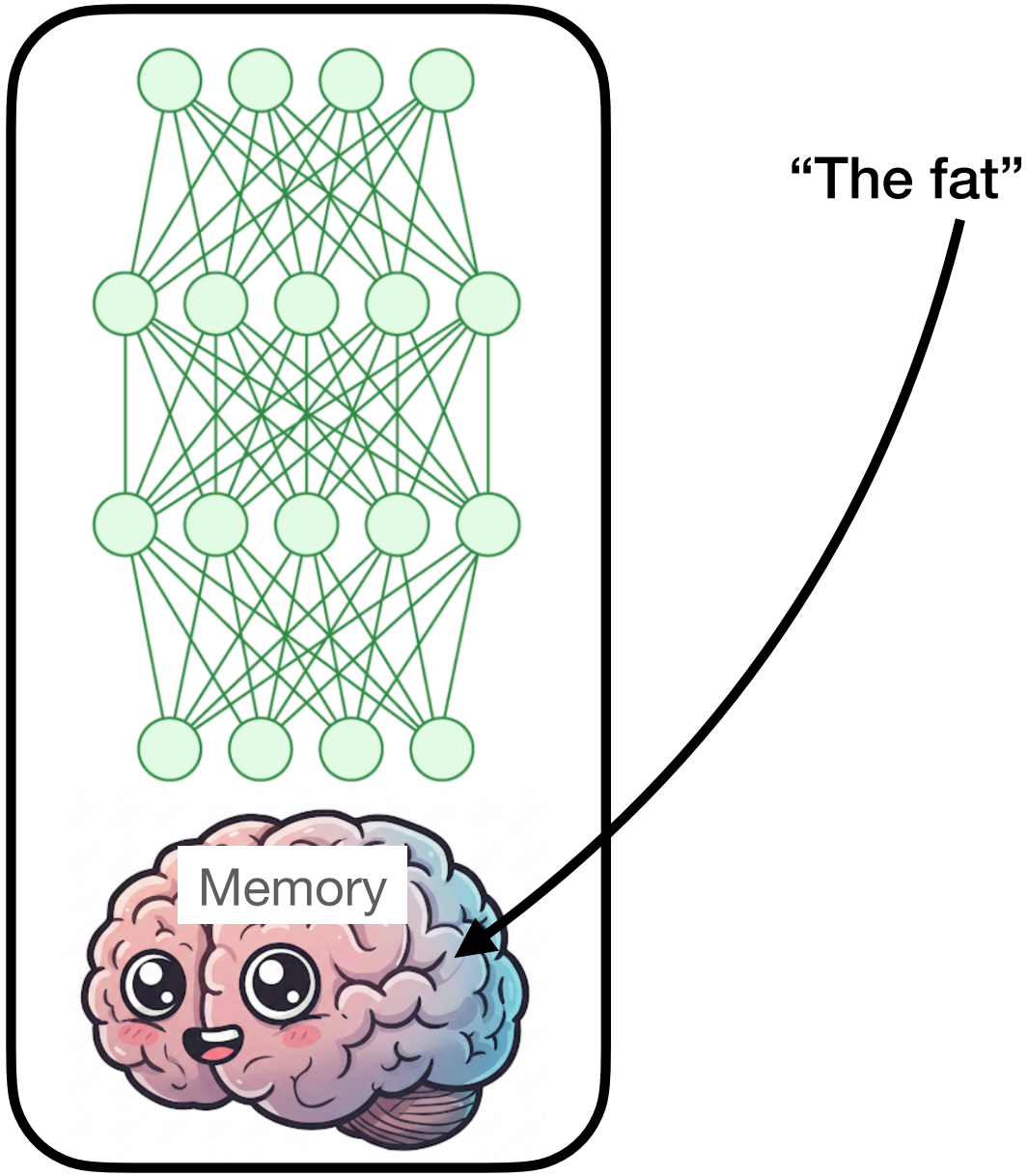


(1) Generate

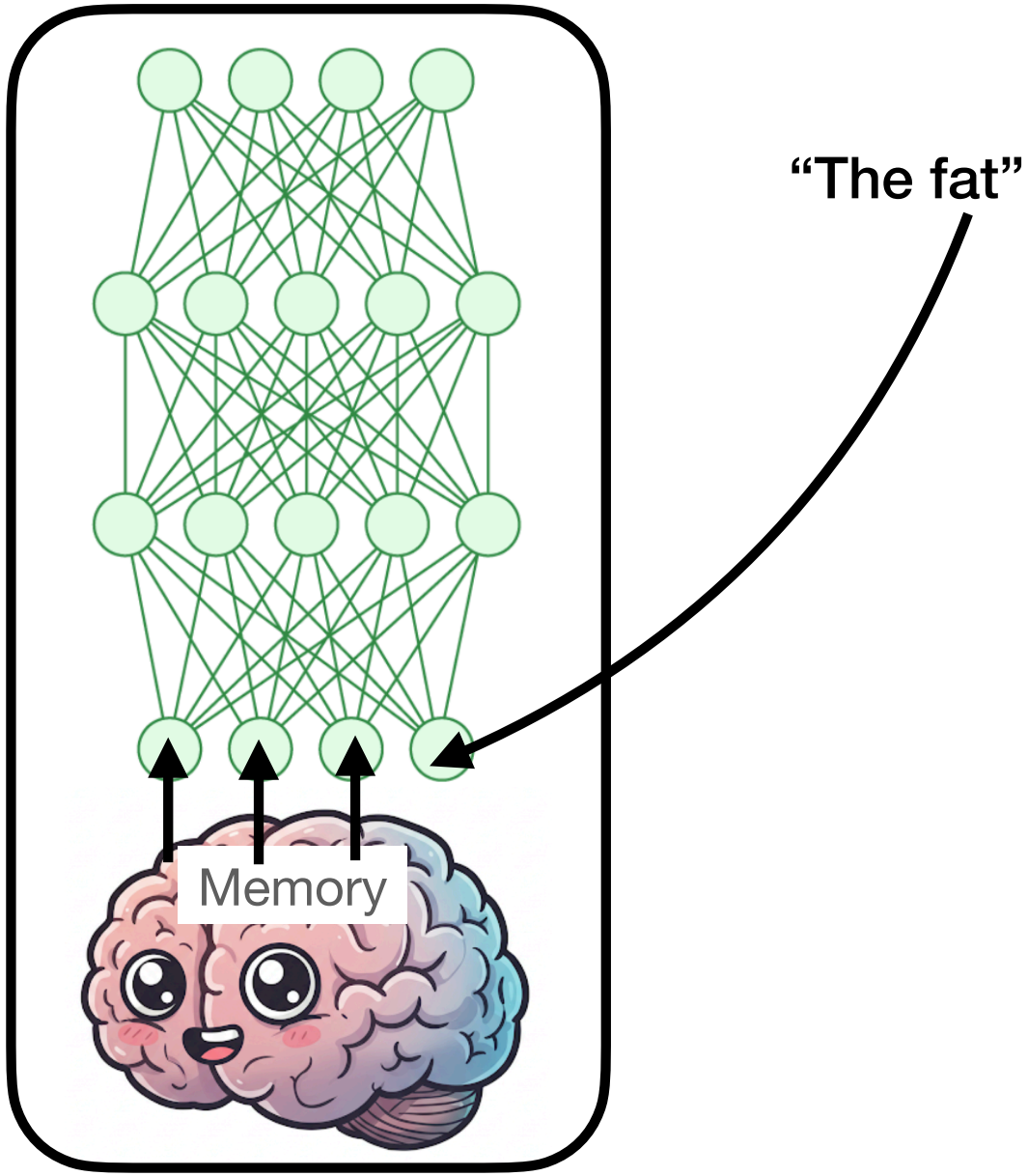


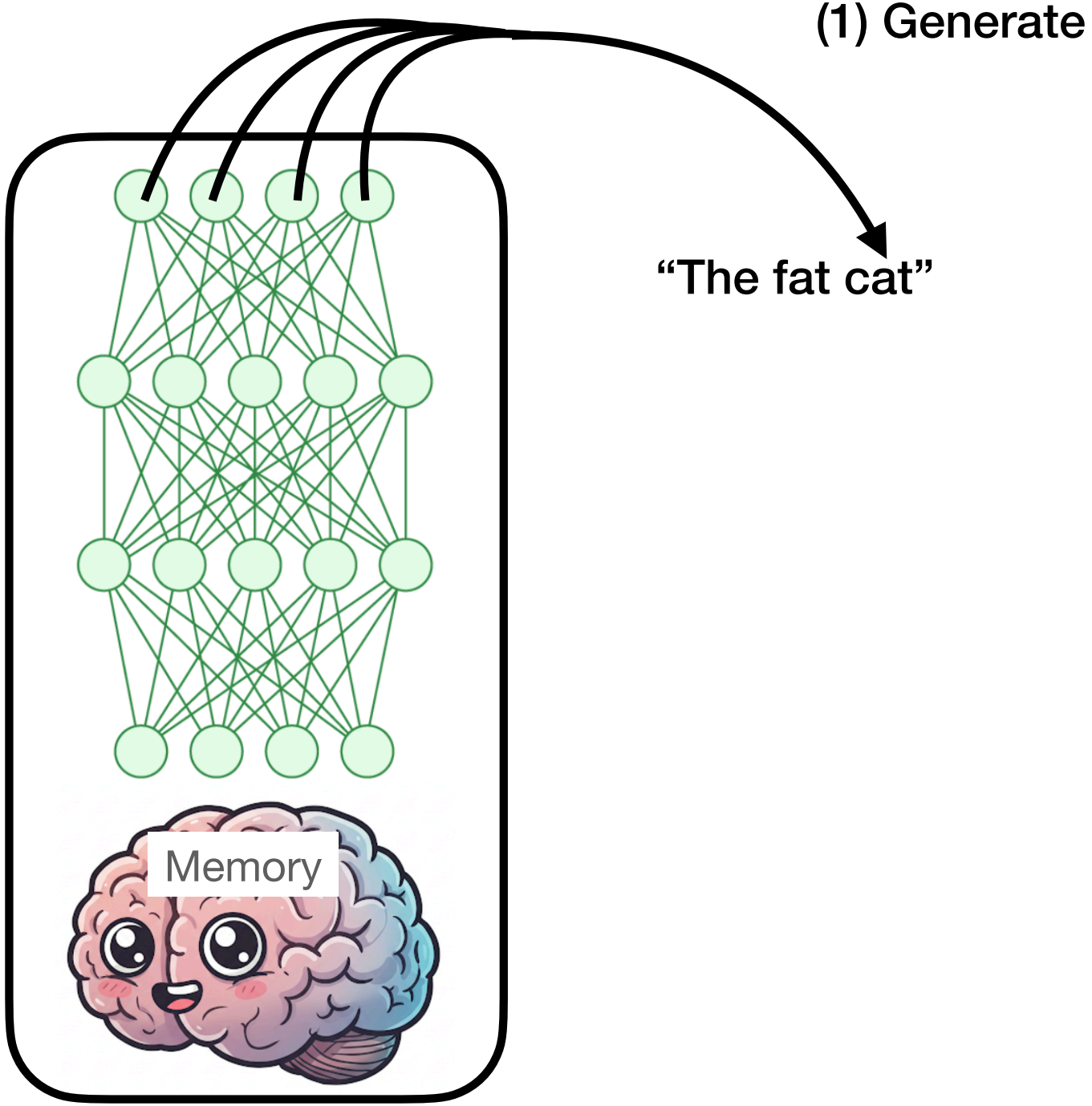
“The fat”

(2) Memorize

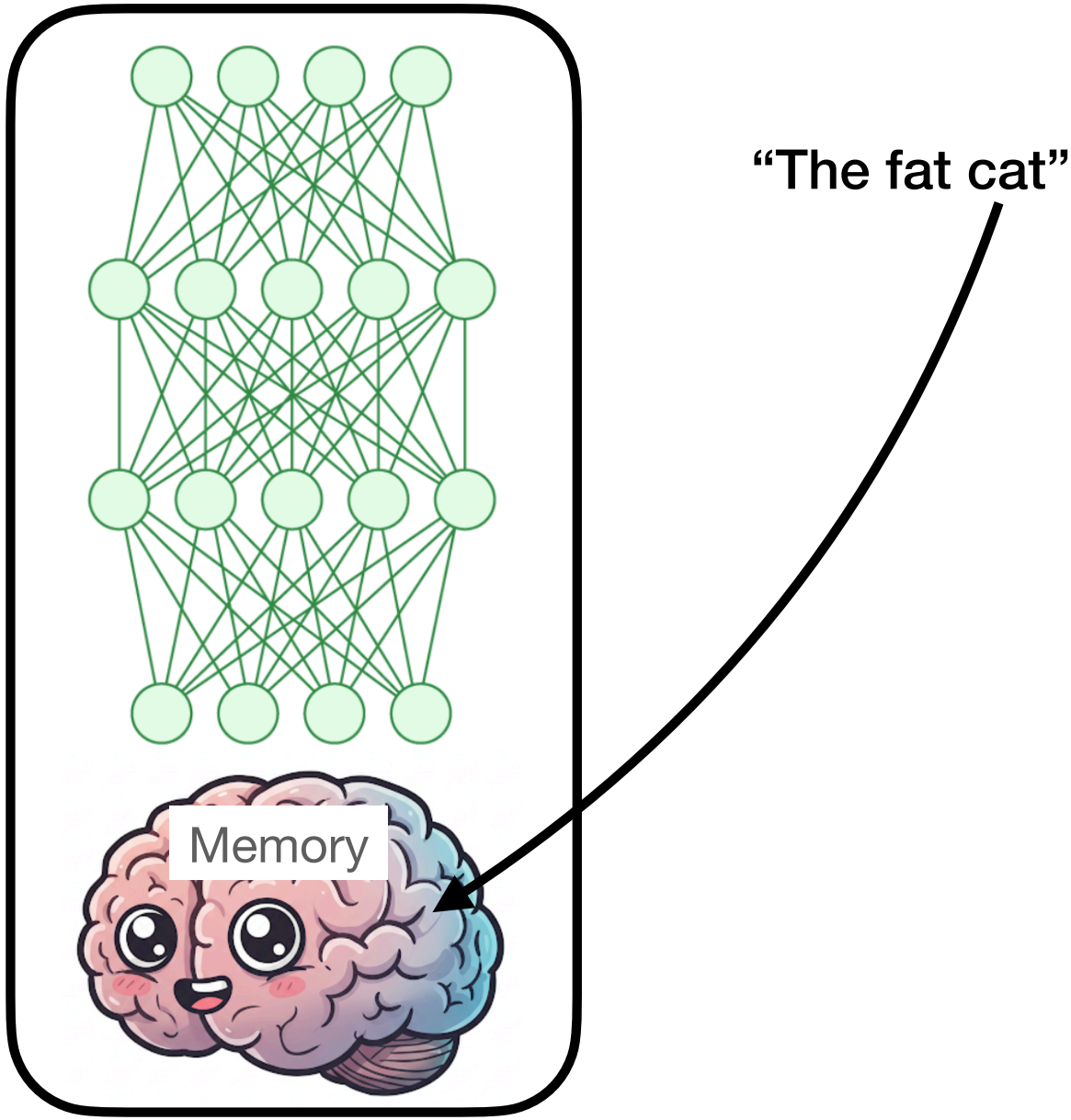


(3) Recurse

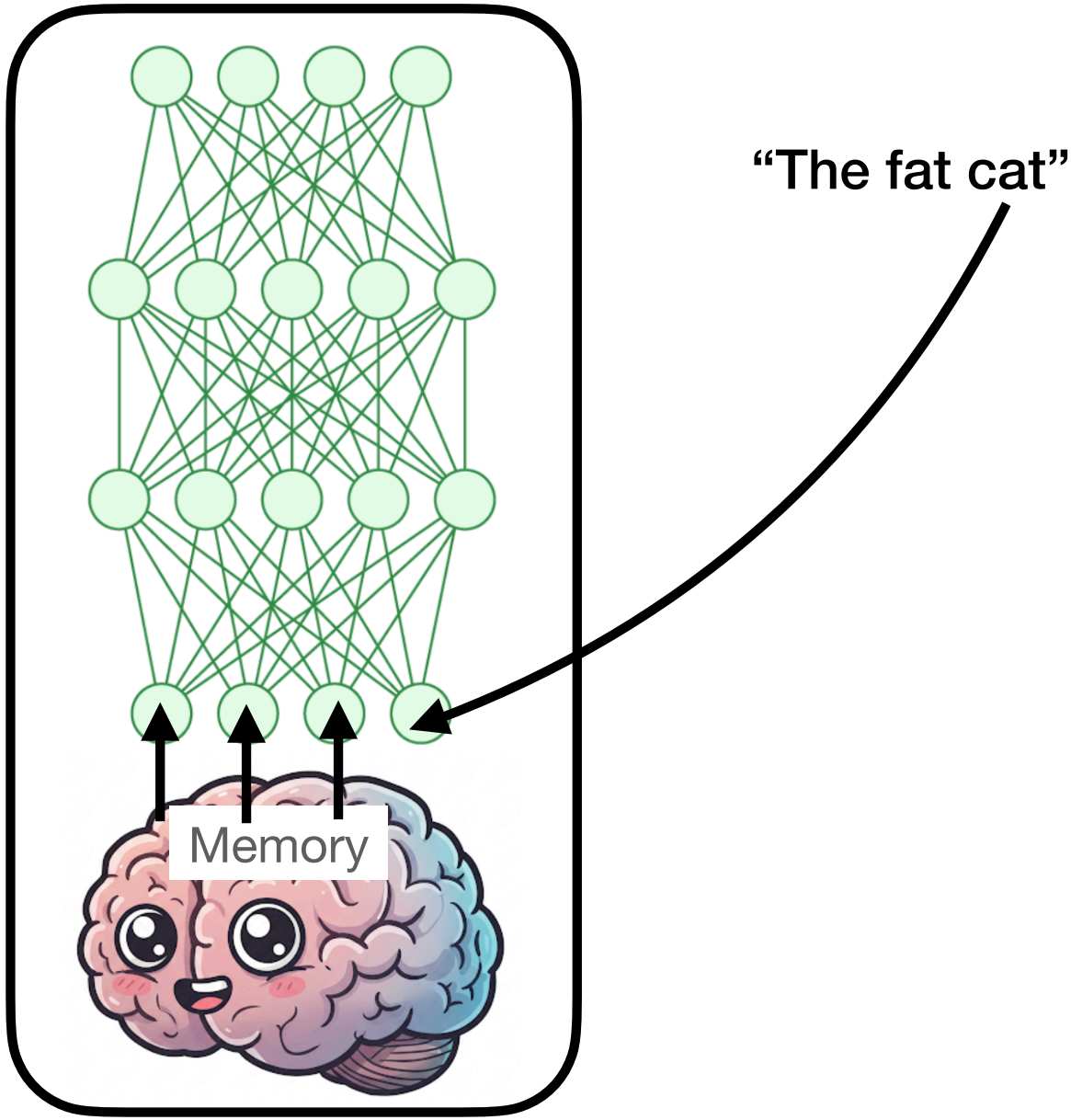


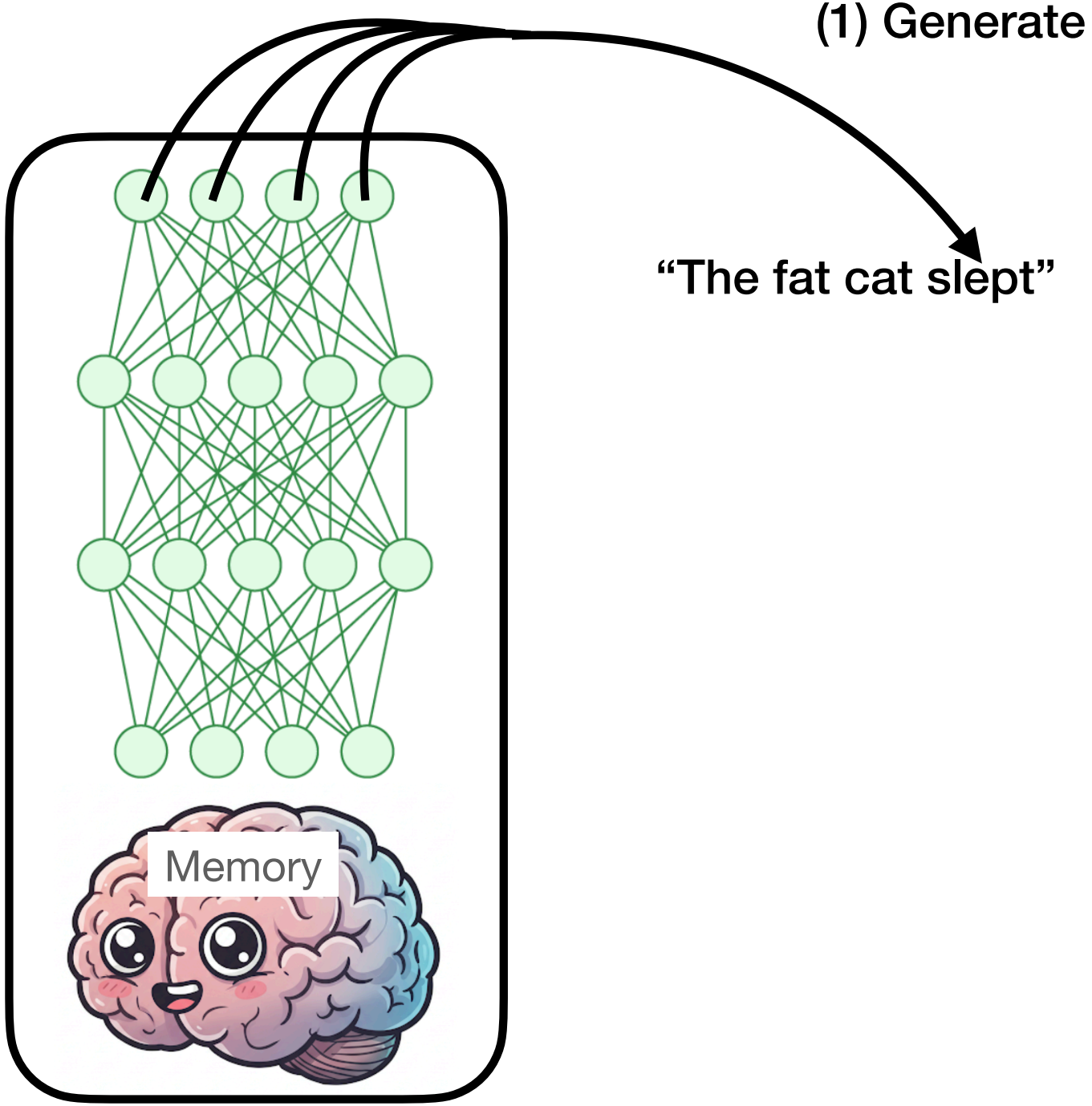

(1) Generate
“The fat cat”
Memory

(2) Memorize

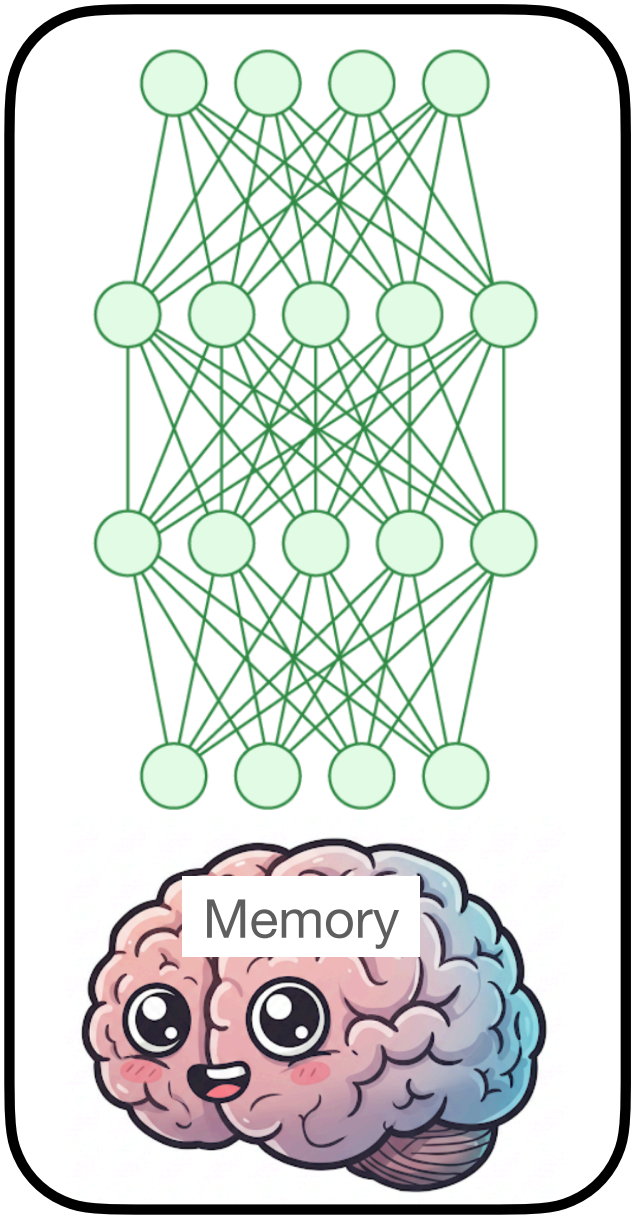


(3) Recurse




(1) Generate
“The fat cat slept”
Memory

And so on…



“The fat cat slept on the mat.”

RNNs: The Good and the Bad

- Good: Memory (in theory) picks out and retains key context
  - RNNs were preferred model for around 5 years
- Bad: In practice memory never worked quite right
  - Really difficult to train on very long contexts
  - How to predict what you need to remember?

Transformers: Attention Is All You Need

- 2017: Paper by Google that (eventually) changed everything

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

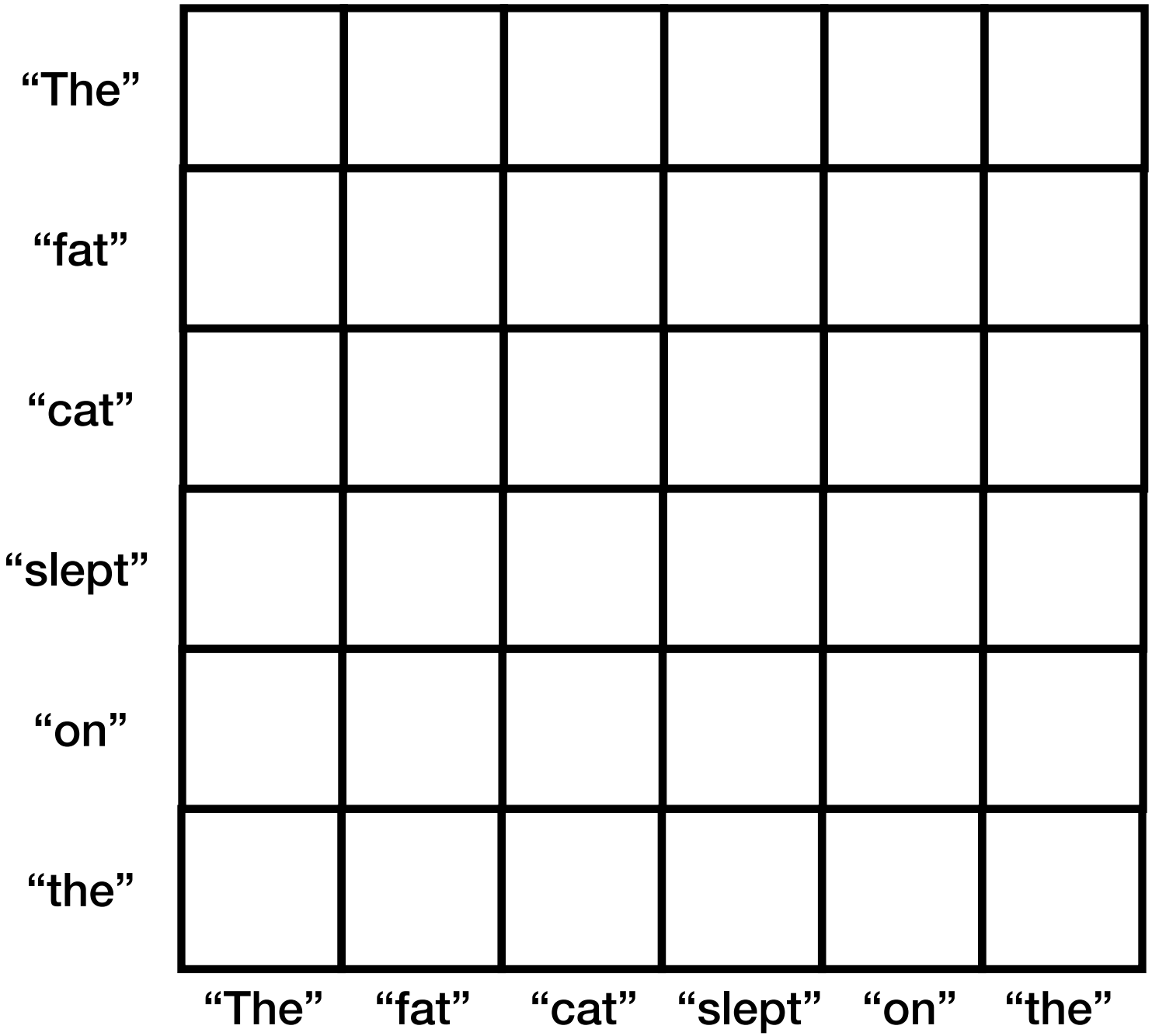
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

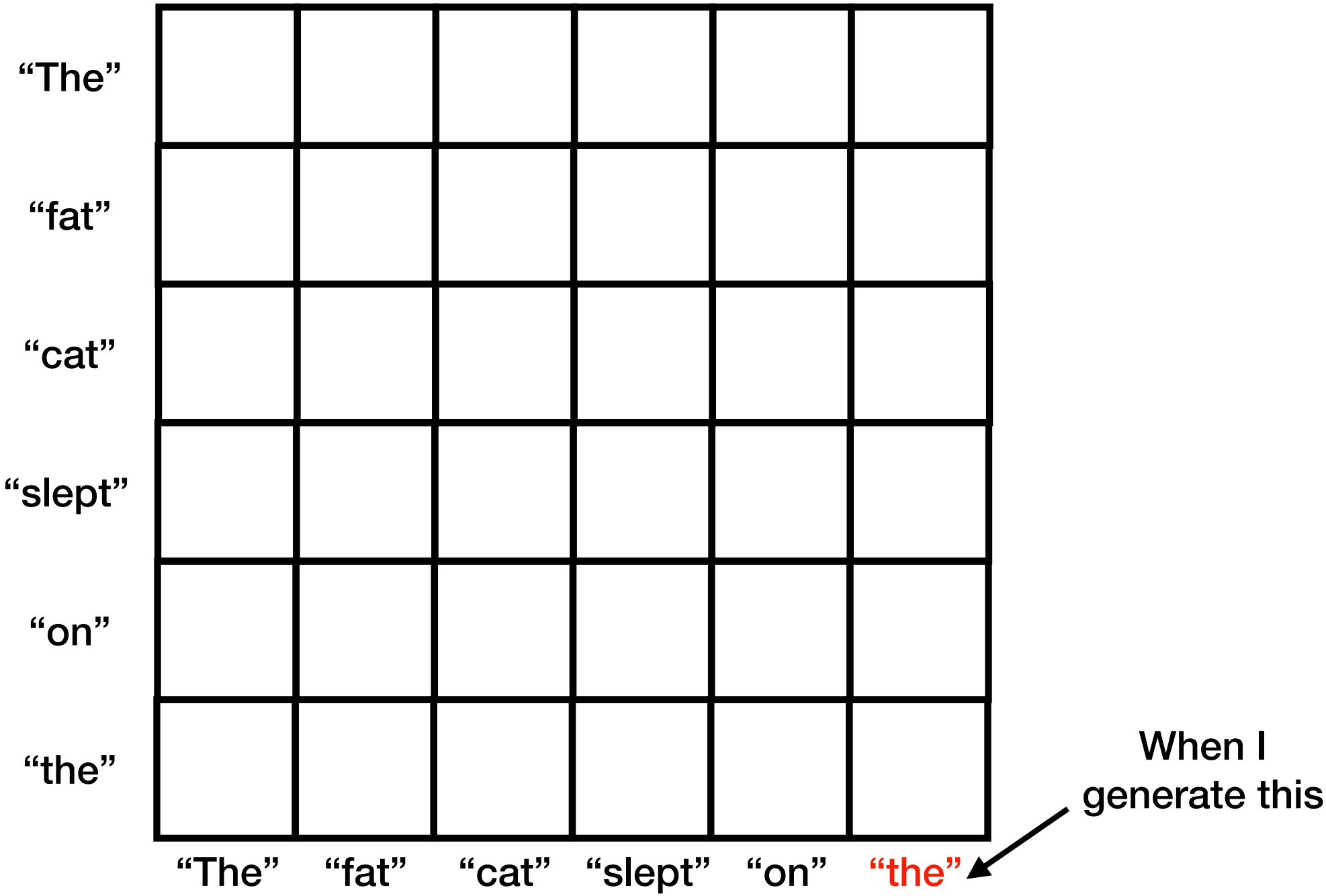
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

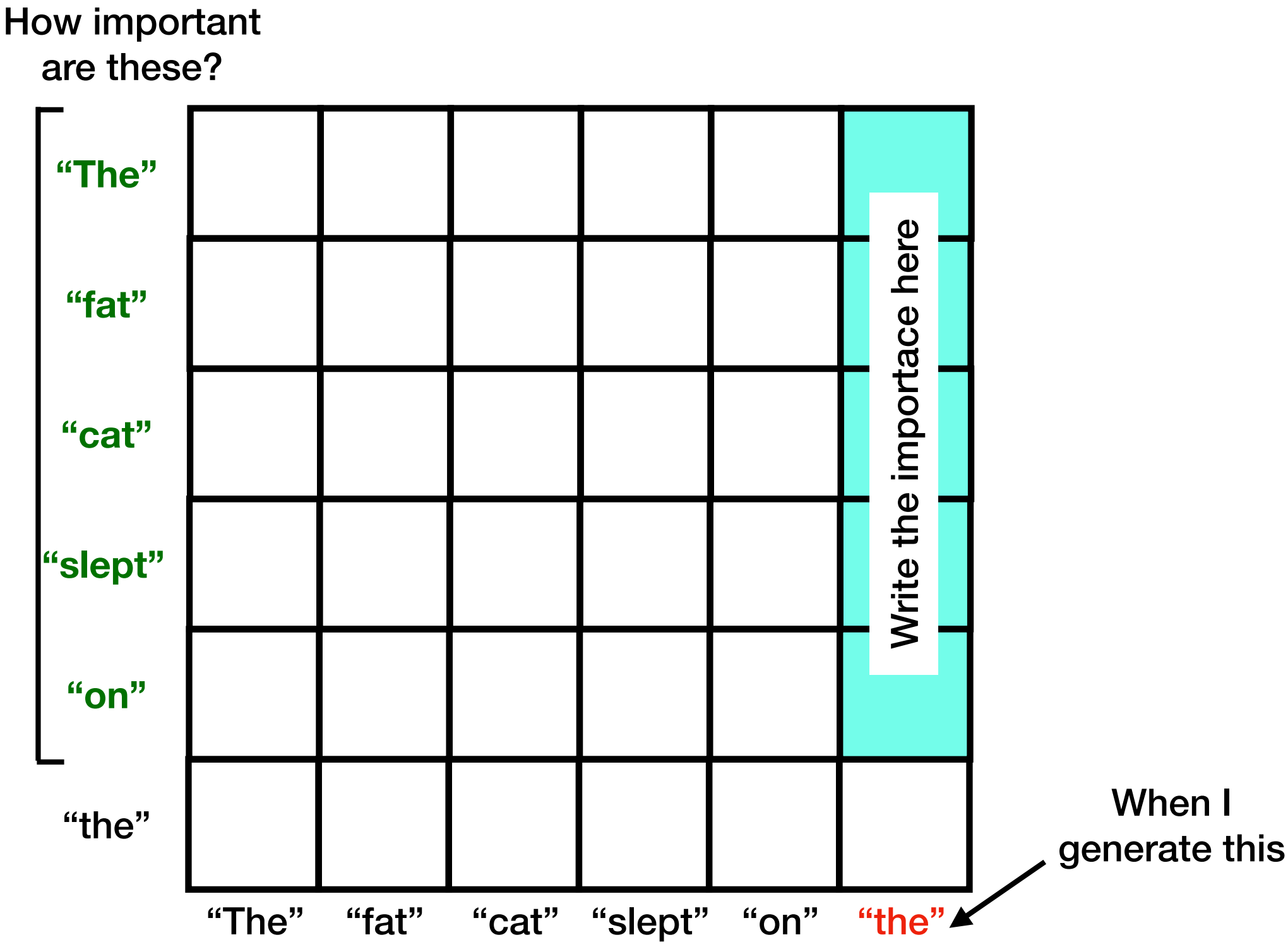
Abstract

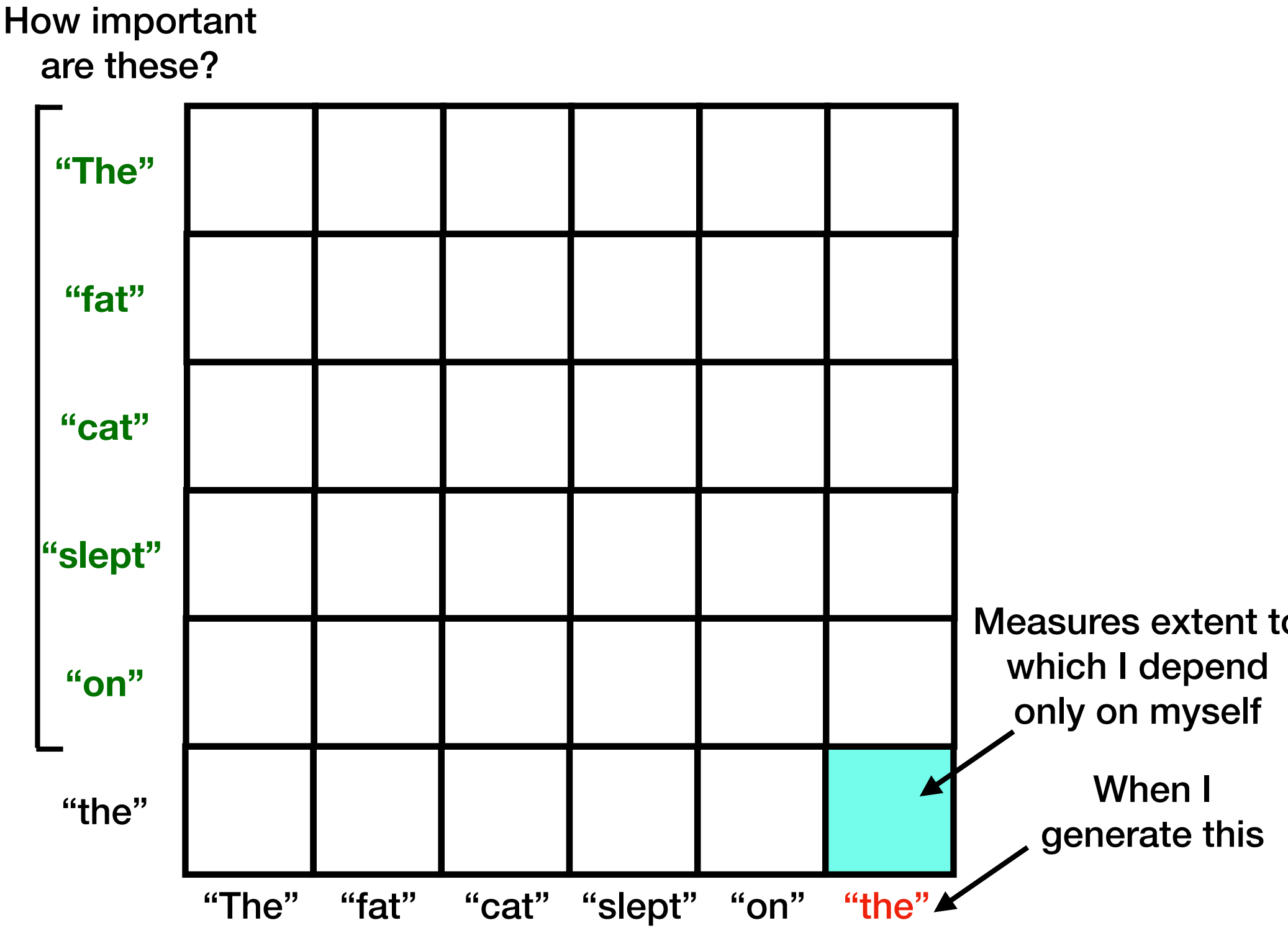
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions

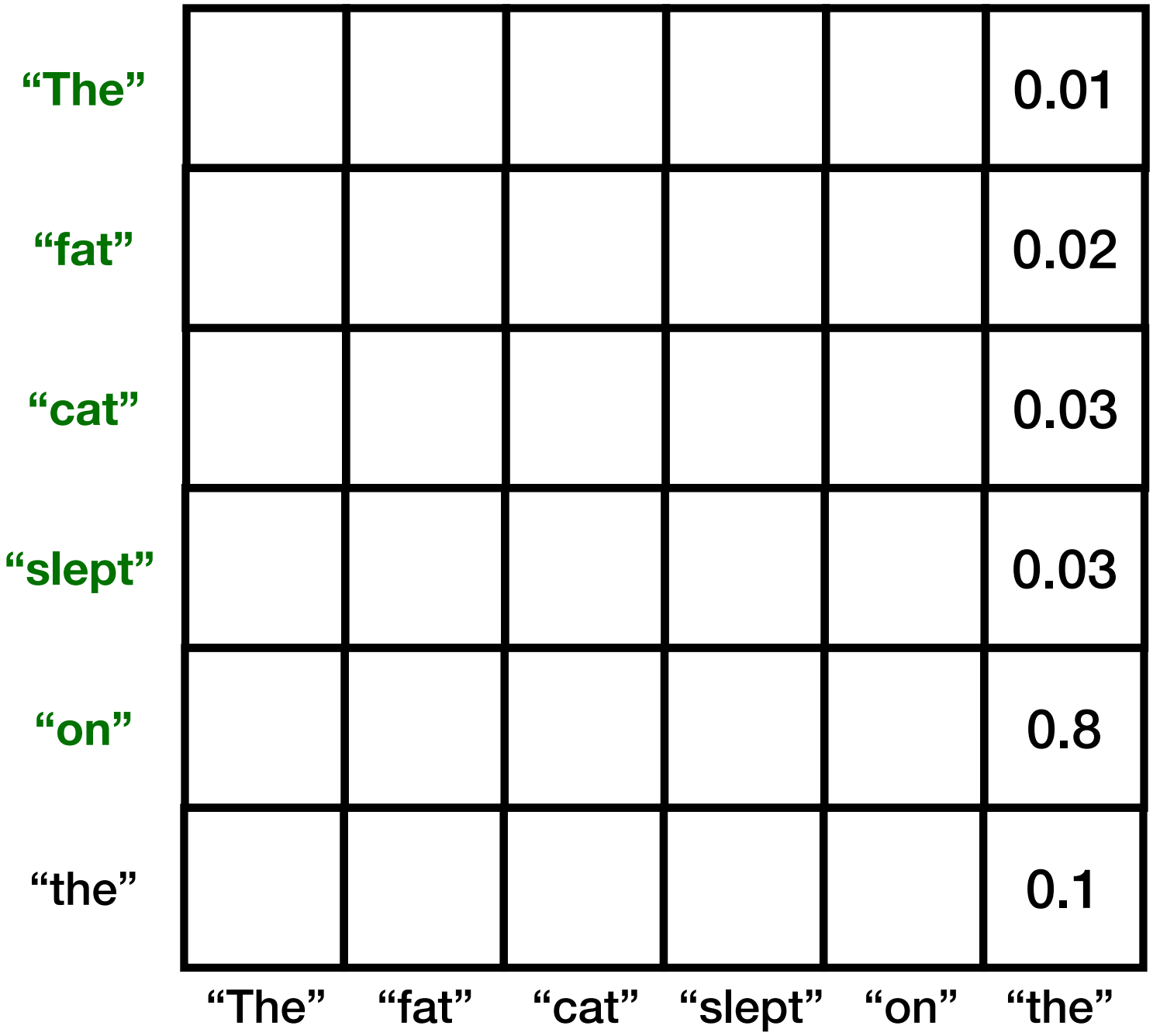
The attention matrix

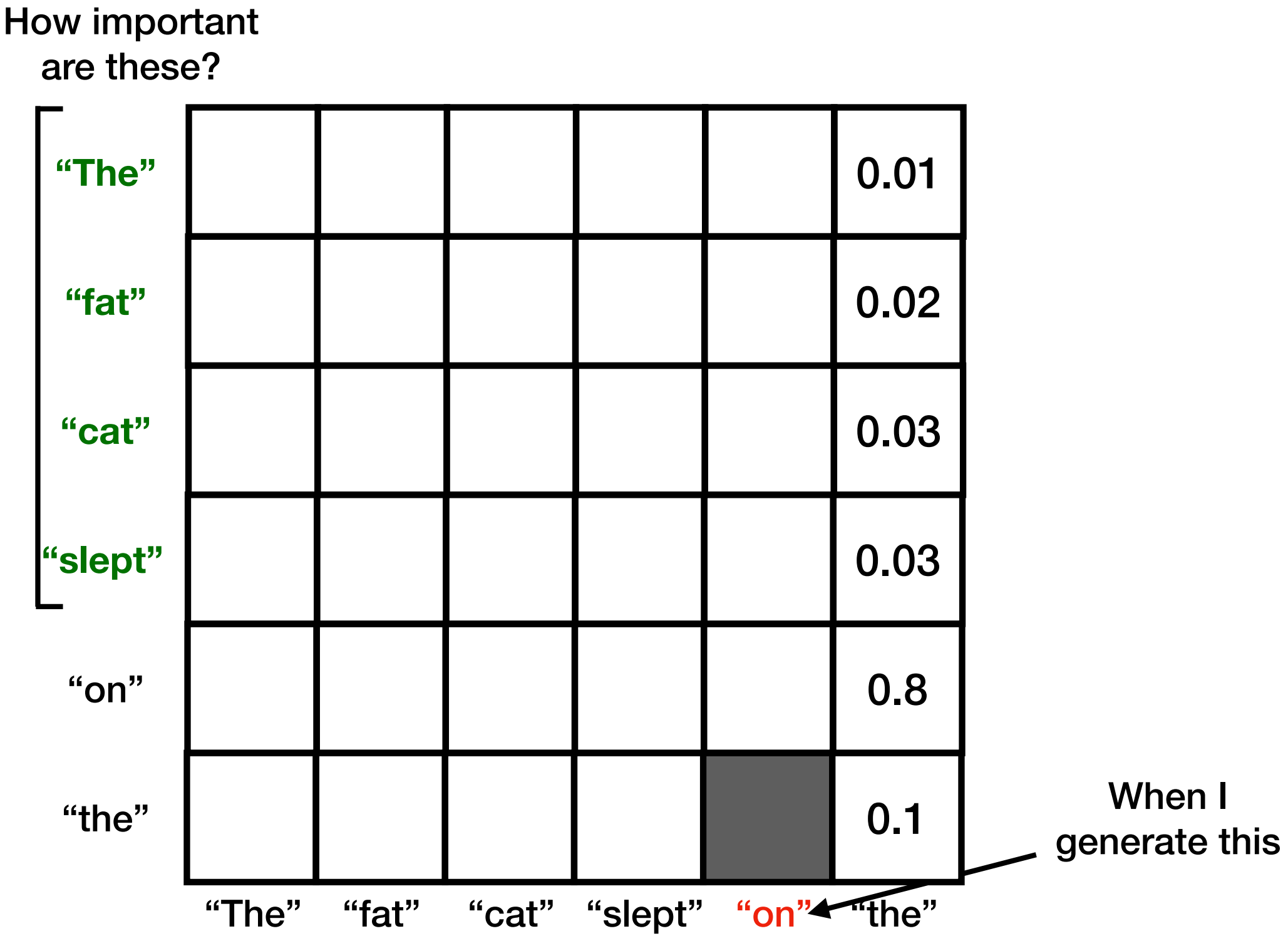


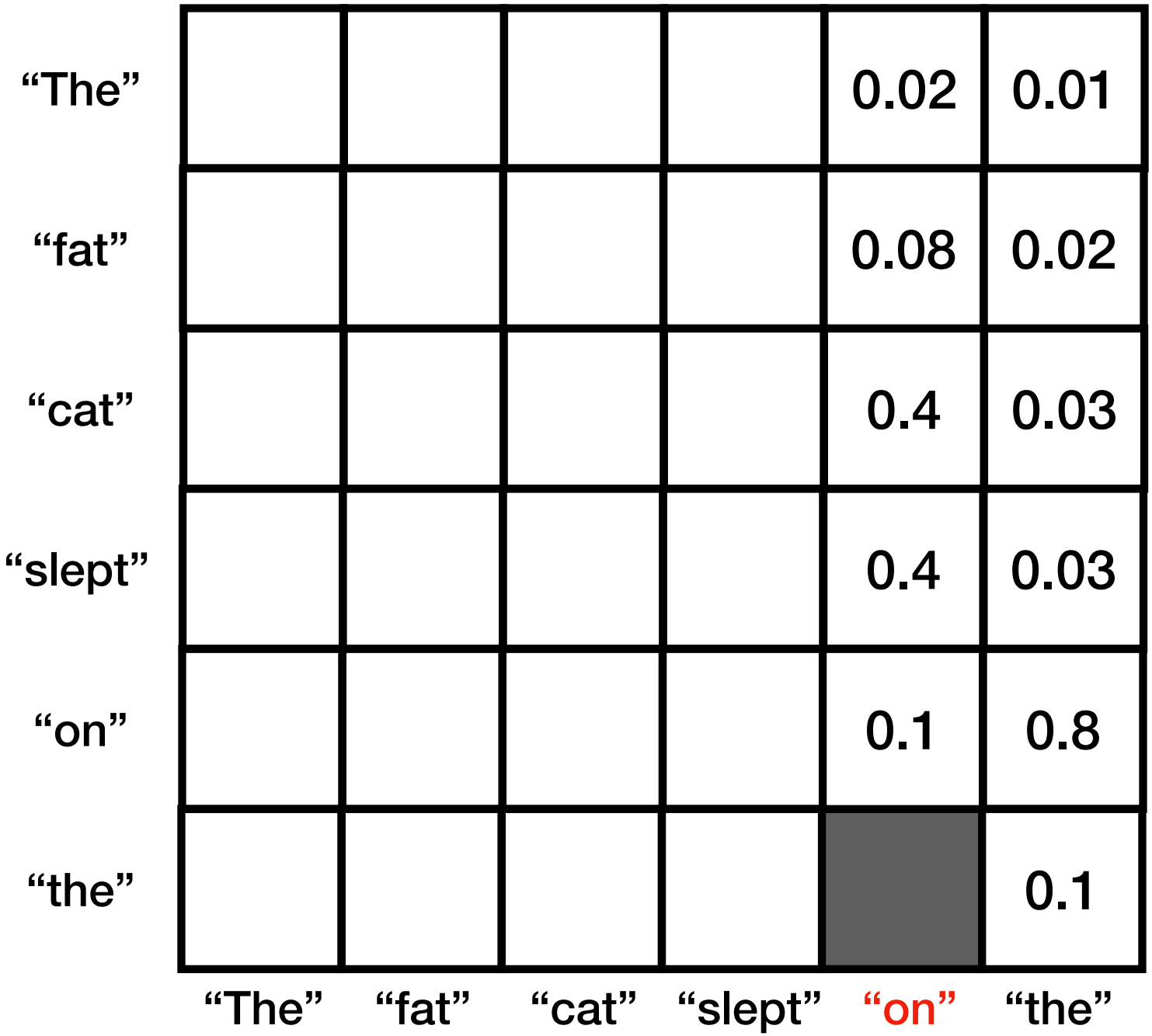

"The"
"fat"
"cat"
"slept"
"on"
"the"
"The"
"fat"
"cat"
"slept"
"on"
"the"
When I generate this

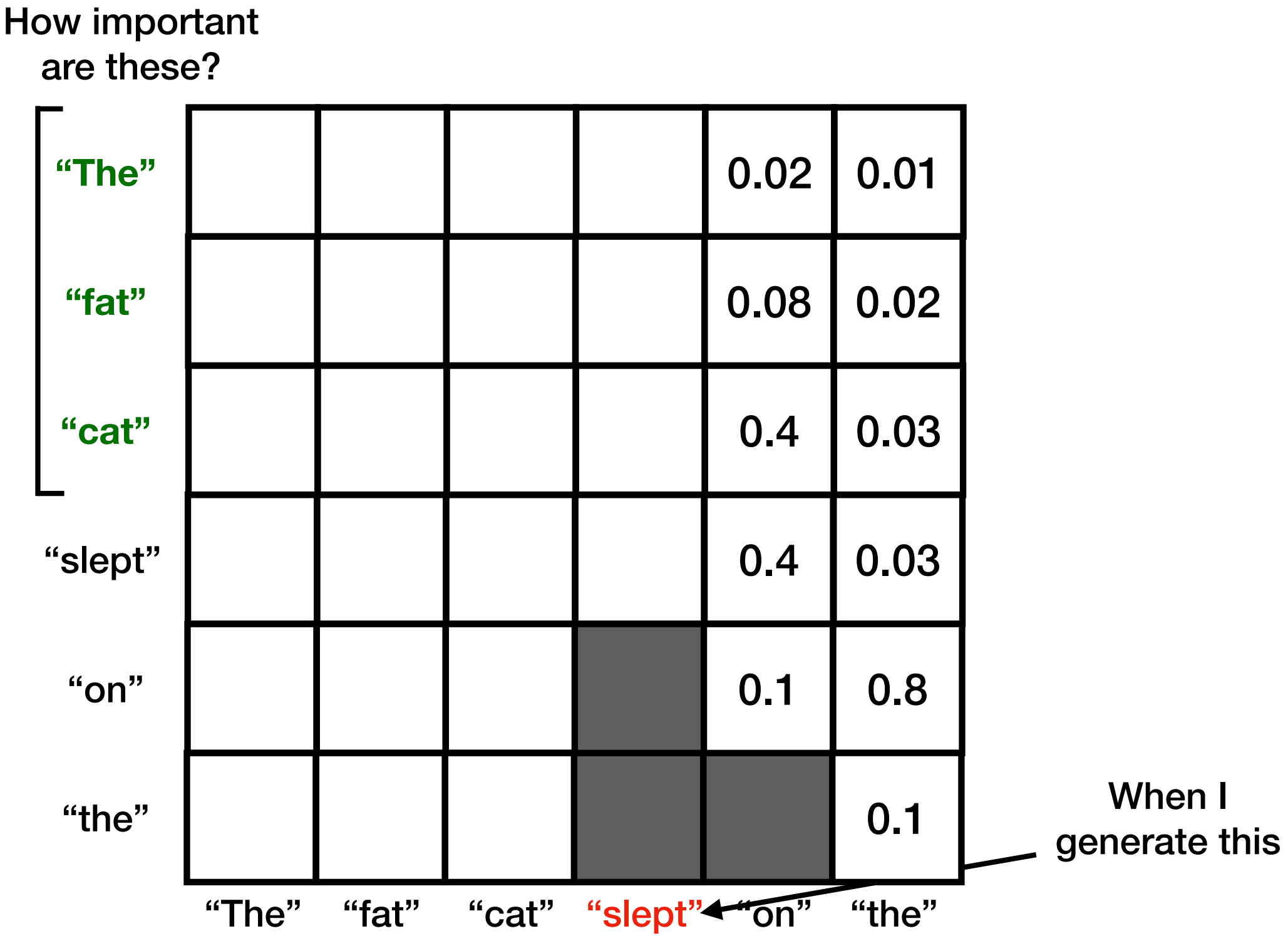

How important are these?
"The"
"fat"
"cat"
"slept"
"on"
"the"
Write the importace here
"The"
"fat"
"cat"
"slept"
"on"
"the"
When I generate this

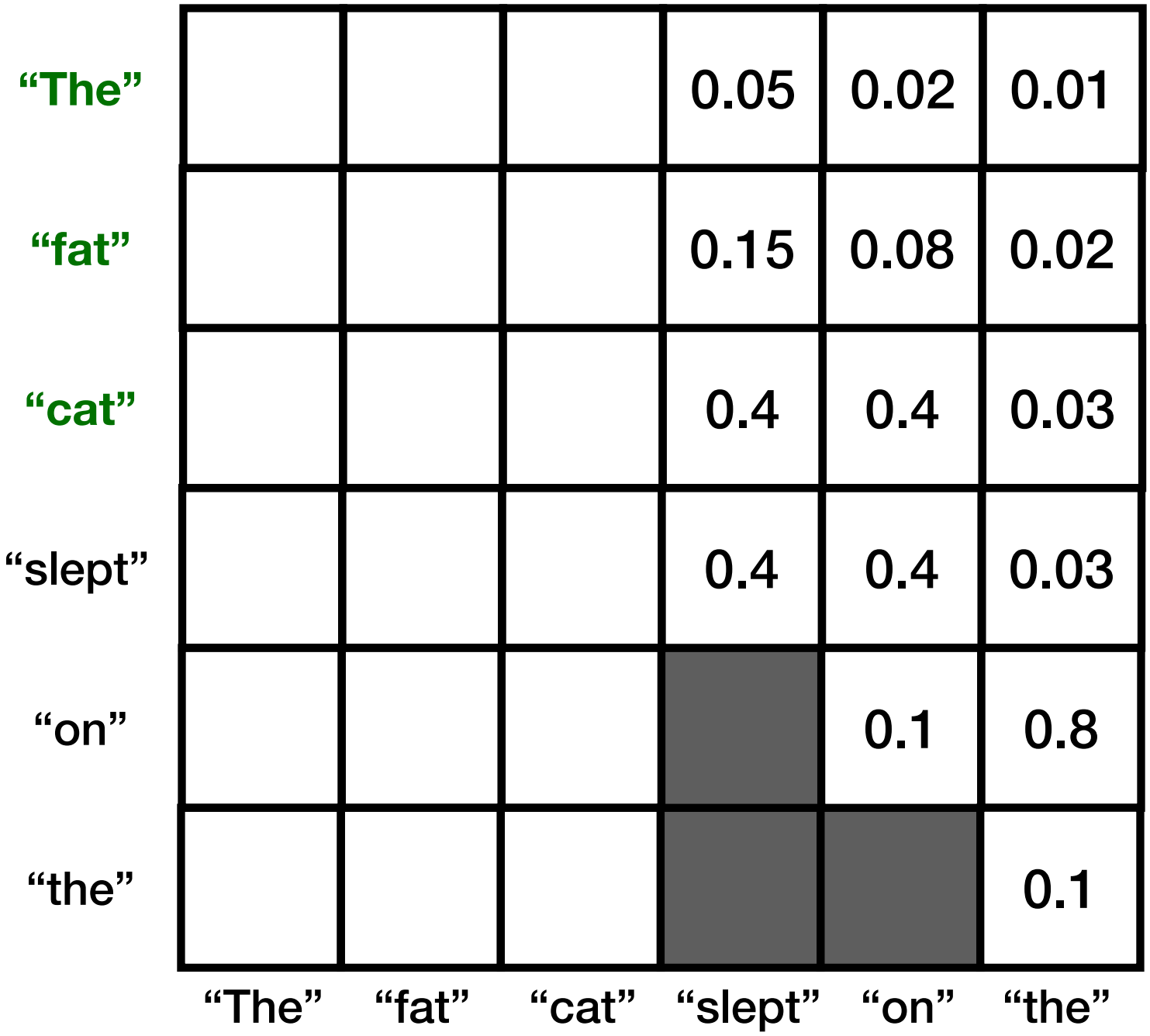

How important are these?
"The"
"fat"
"cat"
"slept"
"on"
"the"
"The"
"fat"
"cat"
"slept"
"on"
"the"
Measures extent to which I depend only on myself
When I generate this


"The"
"fat"
"cat"
"slept"
"on"
"the"
0.01
0.02
0.03
0.03
0.8
0.1
"The"
"fat"
"cat"
"slept"
"on"
"the"

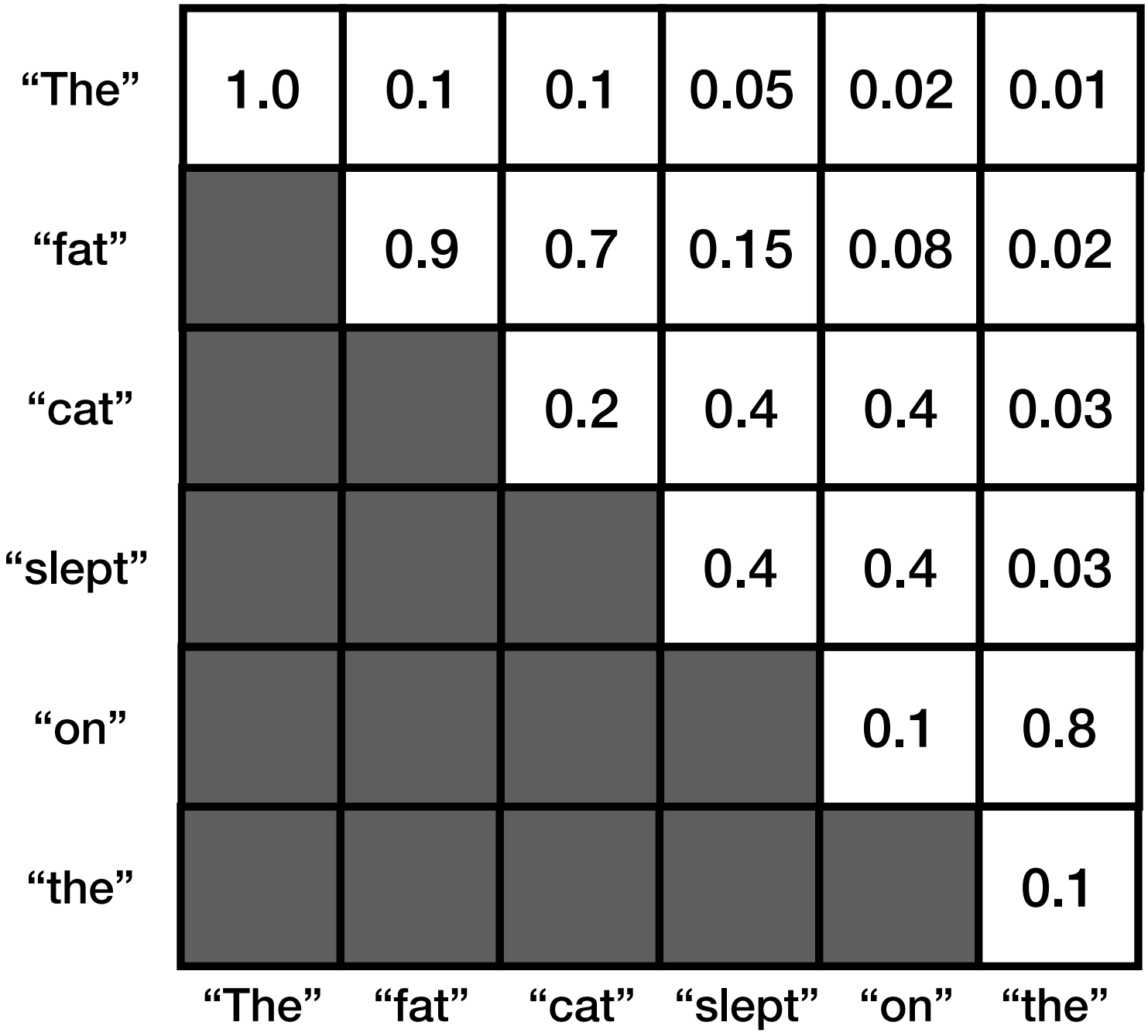

How important are these?
"The"
"fat"
"cat"
"slept"
"on"
"the"
0.01
0.02
0.03
0.03
0.8
0.1
When I generate this
"The"
"fat"
"cat"
"slept"
"on"
"the"


"The"
"fat"
"cat"
"slept"
"on"
"the"
0.02
0.01
0.08
0.02
0.4
0.03
0.4
0.03
0.1
0.8
0.1
"The"
"fat"
"cat"
"slept"
"on"
"the"


How important are these?
"The"
"fat"
"cat"
"slept"
"on"
"the"
0.02
0.01
0.08
0.02
0.4
0.03
0.4
0.03
0.1
0.8
0.1
When I generate this
"The"
"fat"
"cat"
"slept"
"on"
"the"

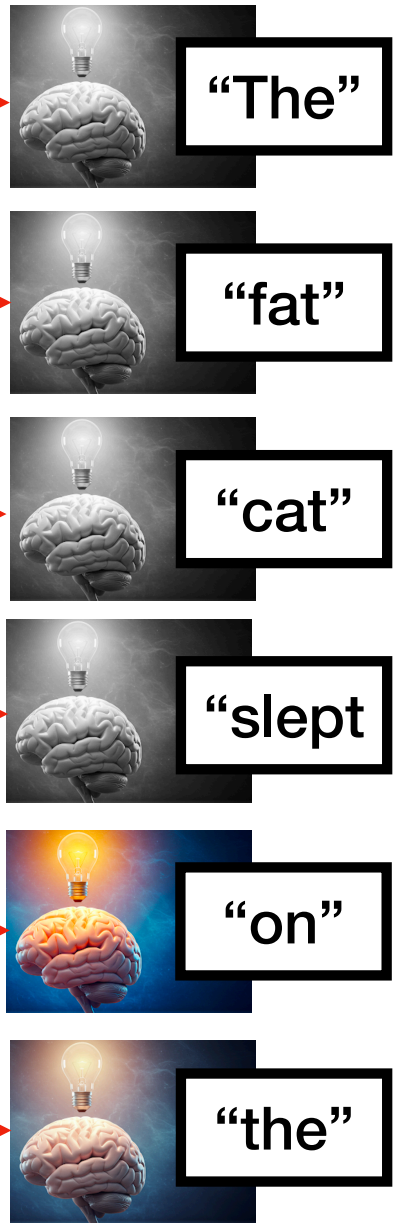

"The"
"fat"
"cat"
"slept"
"on"
"the"
0.05
0.02
0.01
0.15
0.08
0.02
0.4
0.4
0.03
0.4
0.4
0.03
0.1
0.8
0.1
"The"
"fat"
"cat"
"slept"
"on"
"the"

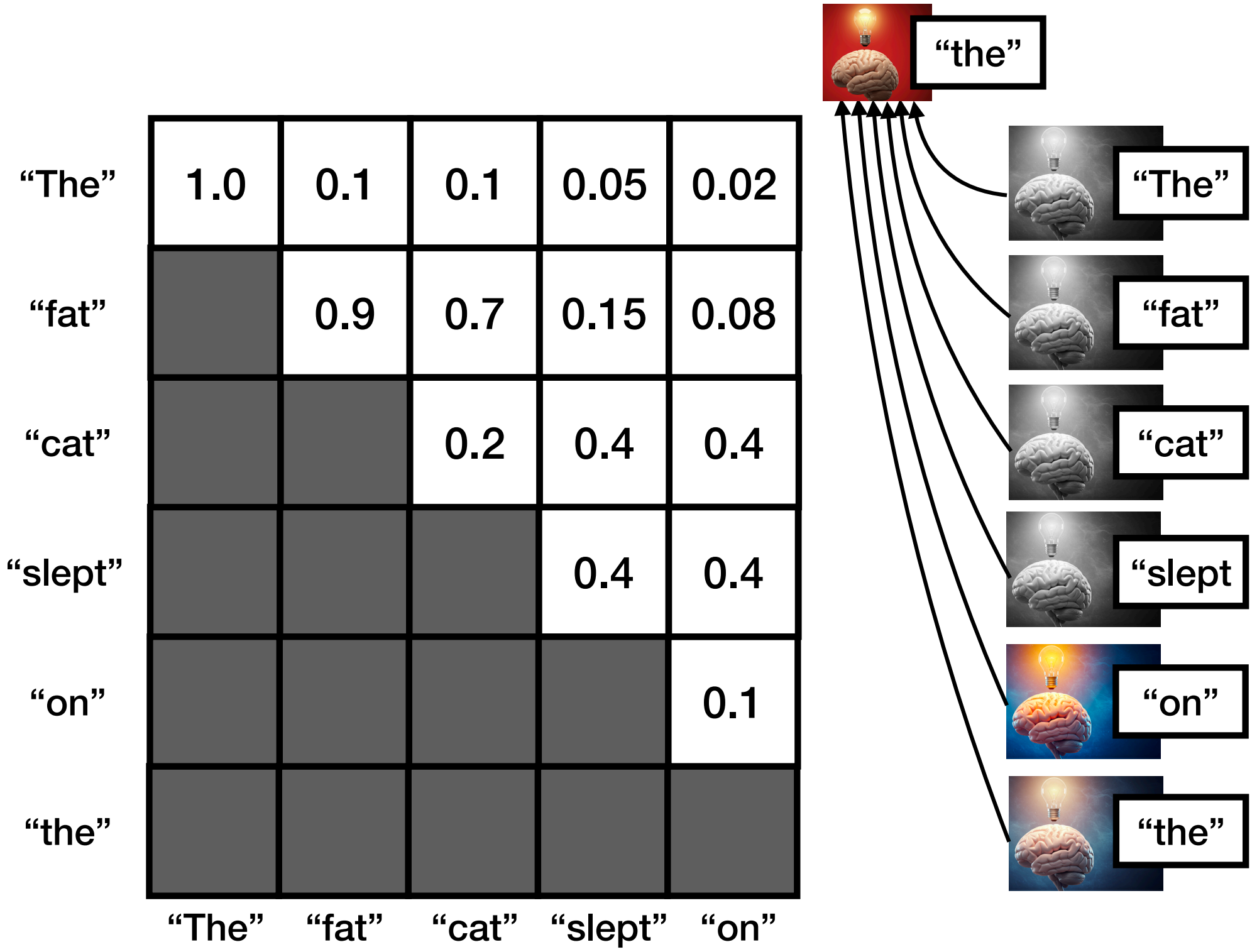
Attention matrix

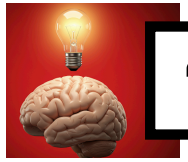


Now, map each term to a higher level of abstraction…

	"The"	"fat"	"cat"	"slept"	"on"	"the"
"The"	1.0	0.1	0.1	0.05	0.02	0.01
"fat"		0.9	0.7	0.15	0.08	0.02
"cat"			0.2	0.4	0.4	0.03
"slept"				0.4	0.4	0.03
"on"					0.1	0.8
"the"						0.1

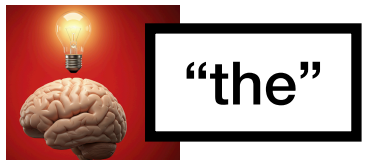



“The”
“fat”
“cat”
“slept”
“on”
“the”
1.0
0.1
0.1
0.05
0.02
0.9
0.7
0.15
0.08
0.2
0.4
0.4
0.4
0.4
0.1
“The”
“fat”
“cat”
“slept”
“on”
“the”
“The”
“fat”
“cat”
“slept
“on”
“the”

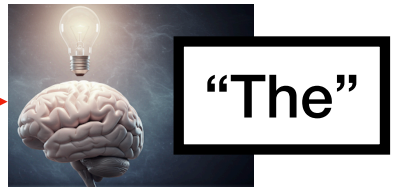


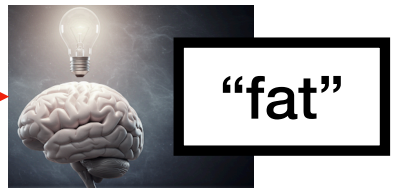
“the”

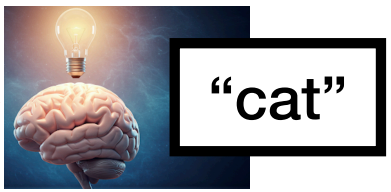
	“The”	“fat”	“cat”	“slept”	“on”
“The”	1.0	0.1	0.1	0.05	0.02
“fat”		0.9	0.7	0.15	0.08
“cat”			0.2	0.4	0.4
“slept”				0.4	0.4
“on”					0.1
“the”					

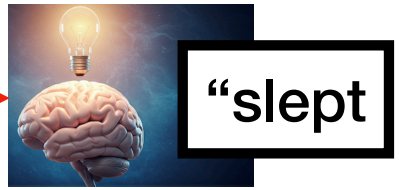


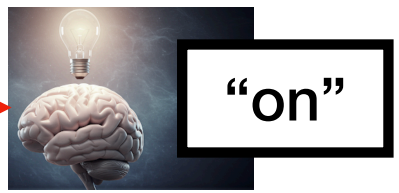
	“The”	“fat”	“cat”	“slept”	“on”
“The”	1.0	0.1	0.1	0.05	0.02
“fat”		0.9	0.7	0.15	0.08
“cat”			0.2	0.4	0.4
“slept”				0.4	0.4
“on”					0.1
“the”					

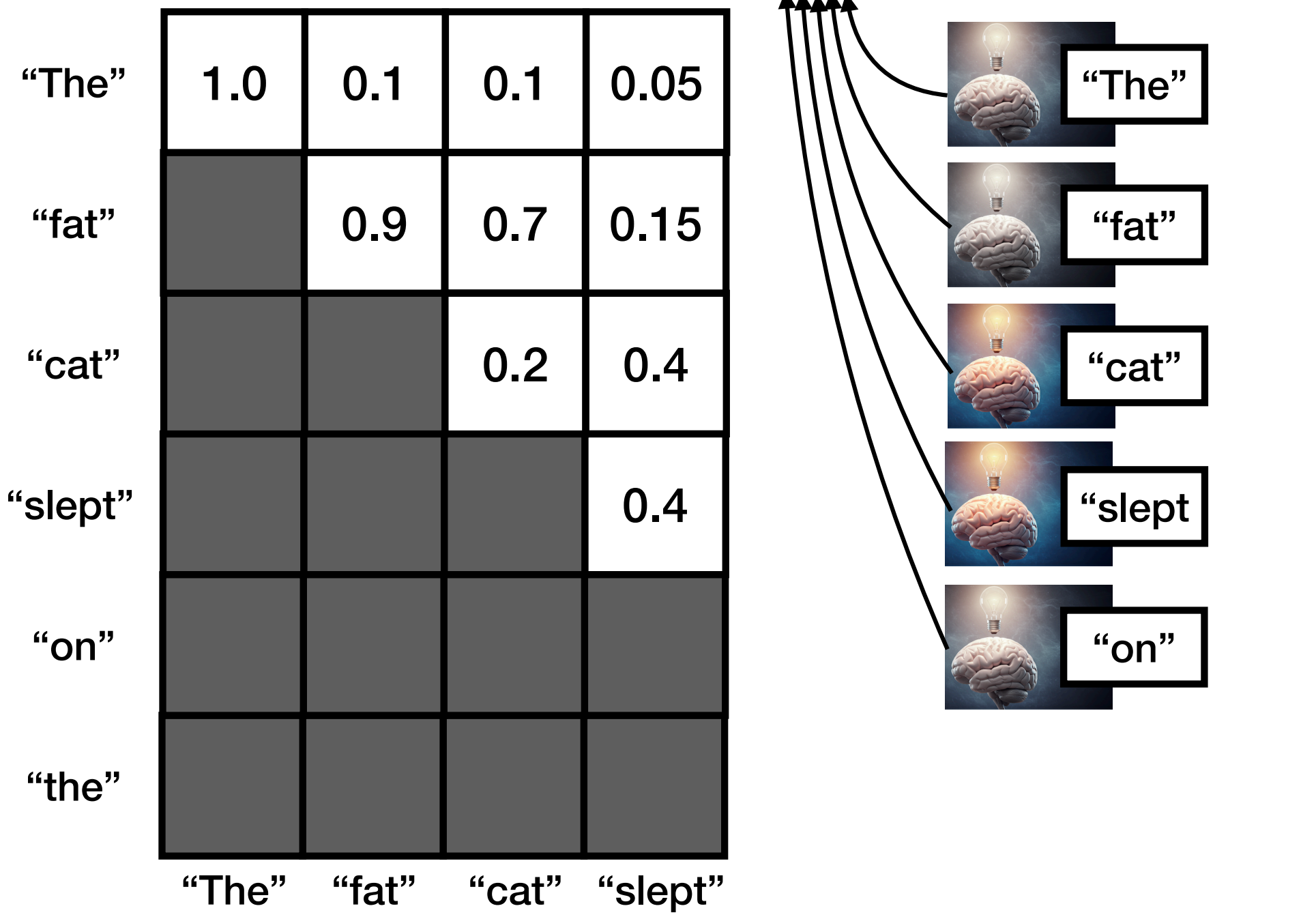


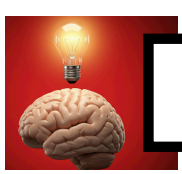






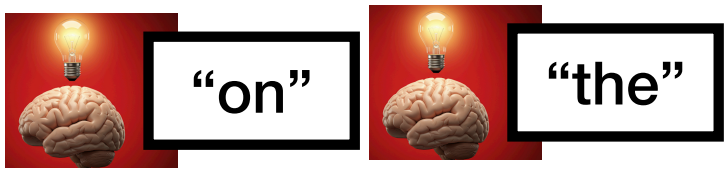



“The”
“fat”
“cat”
“slept”
“on”
“the”
1.0
0.1
0.1
0.05
0.9
0.7
0.15
0.2
0.4
0.4
“The”
“fat”
“cat”
“slept”
“on”
“the”
“The”
“fat”
“cat”
“slept
“on”

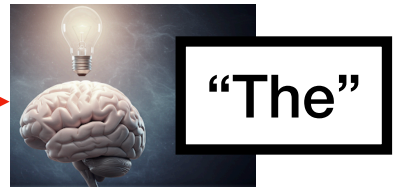

“on”

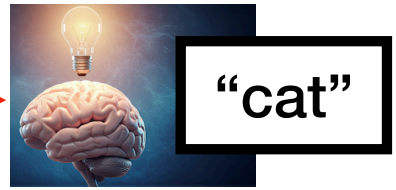
“the”

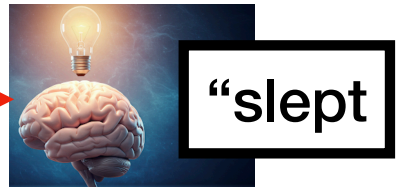
	“The”	“fat”	“cat”	“slept”
“The”	1.0	0.1	0.1	0.05
“fat”		0.9	0.7	0.15
“cat”			0.2	0.4
“slept”				0.4
“on”				
“the”				

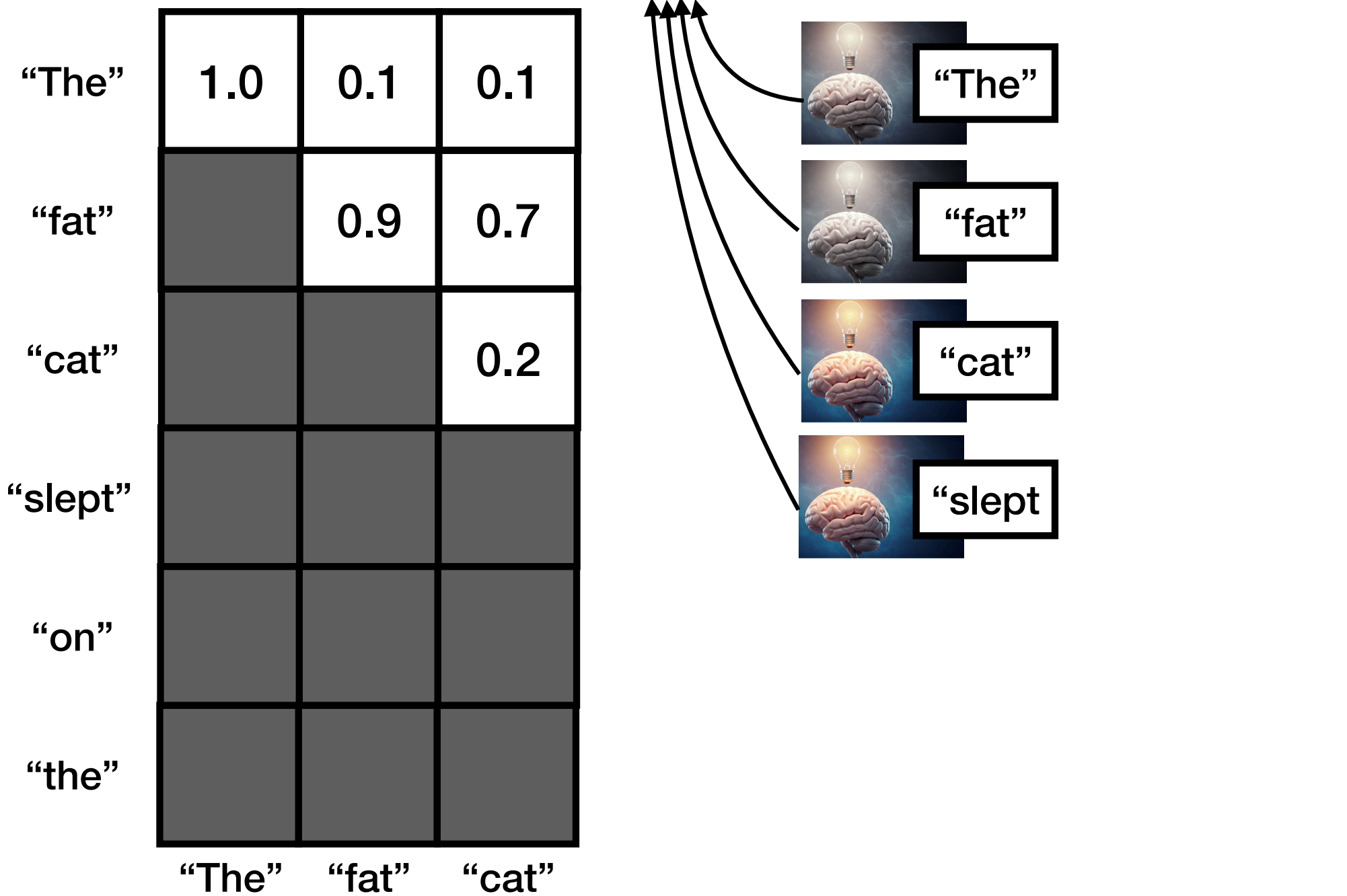


	“The”	“fat”	“cat”	“slept”
“The”	1.0	0.1	0.1	0.05
“fat”		0.9	0.7	0.15
“cat”			0.2	0.4
“slept”				0.4
“on”				
“the”				

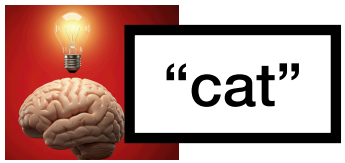







“slept”
“on”
“the”
“The”
“fat”
“cat”
“slept
1.0
0.1
0.1
0.9
0.7
0.2
“The”
“fat”
“cat”
“slept”
“on”
“the”

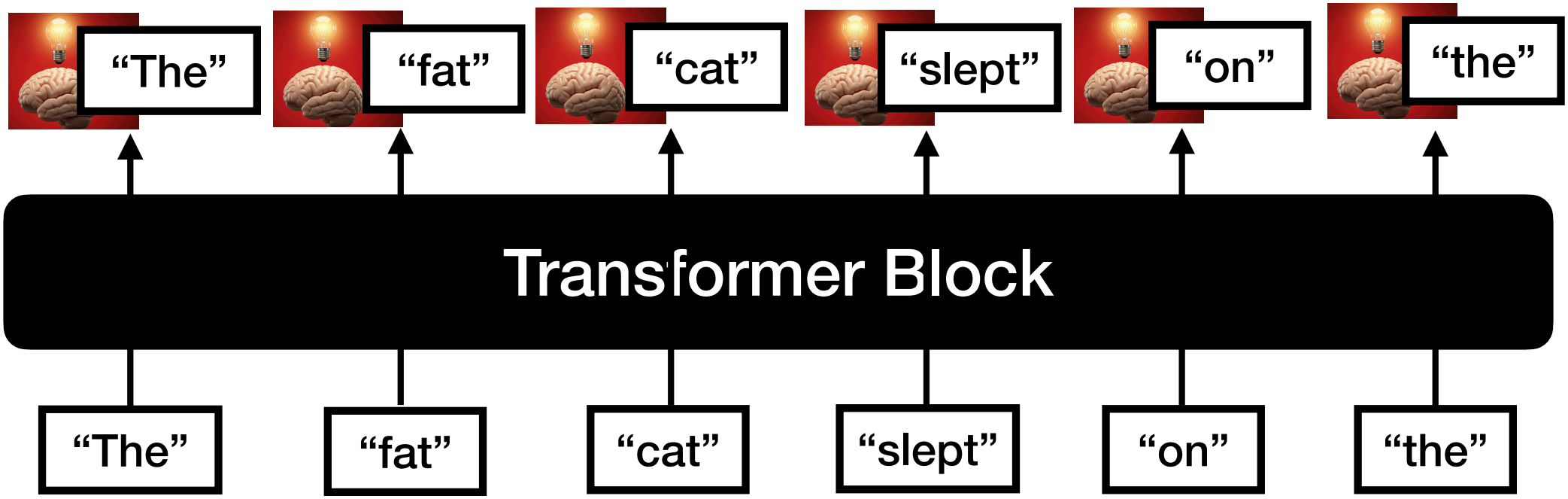

“The”

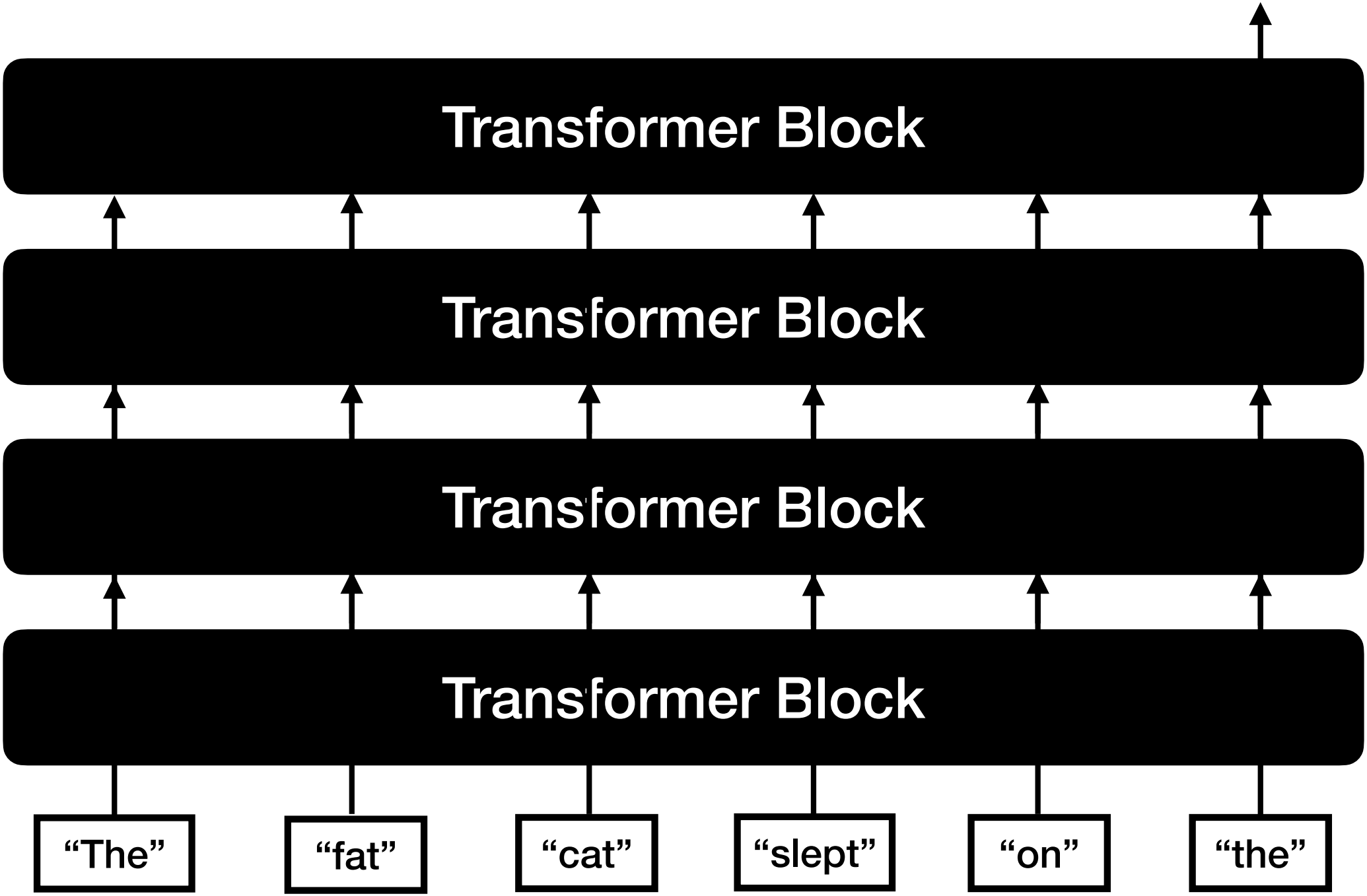
“fat”

“cat”

“slept”

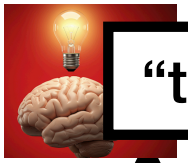
“on”

“the”


“The”
“fat”
“cat”
“slept”
“on”
“the”
Transformer Block
“The”
“fat”
“cat”
“slept”
“on”
“the”


Transformer Block
Transformer Block
Transformer Block
Transformer Block
“The”
“fat”
“cat”
“slept”
“on”
“the”

Output: highly conceptualized version of the word “the”



“the”

Transformer Block

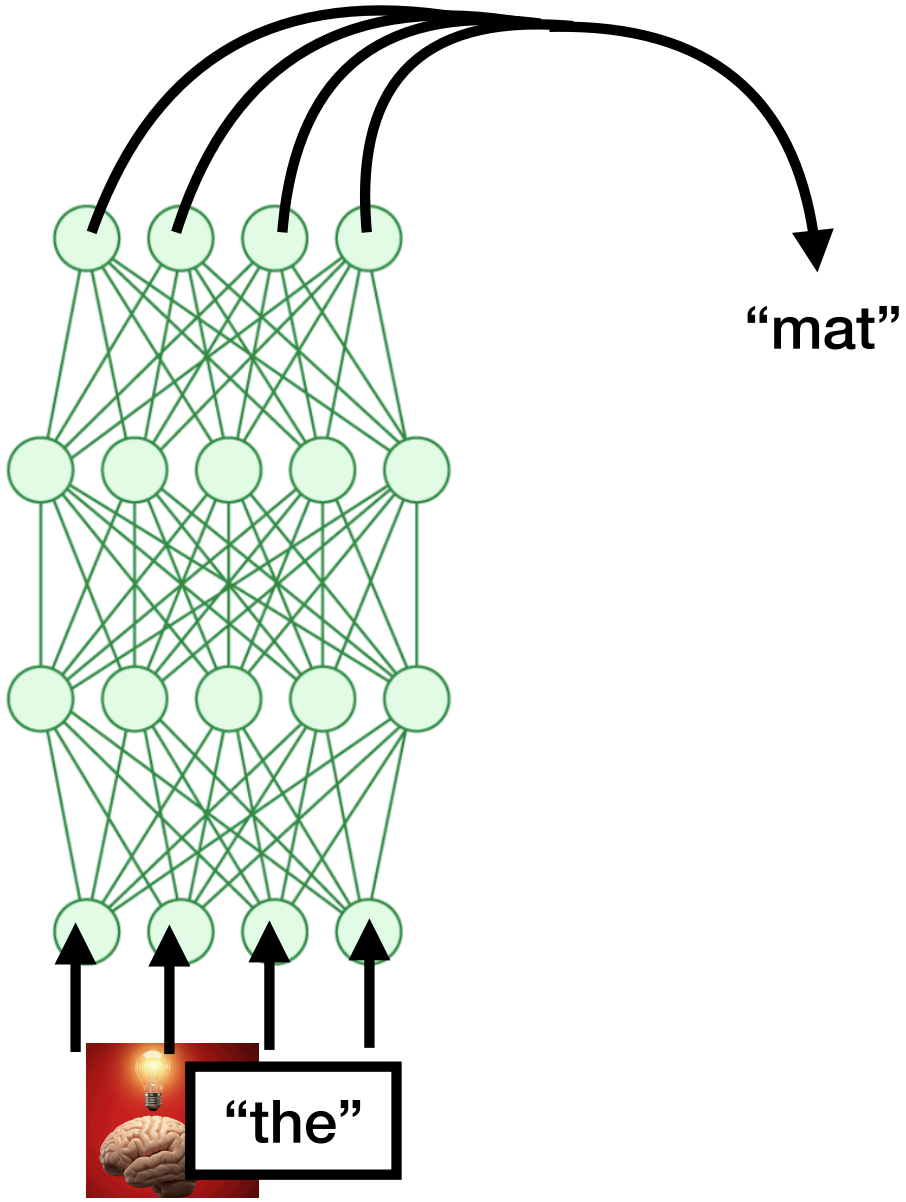
Transformer Block

Transformer Block

Transformer Block

“The” “fat” “cat” “slept” “on” “the”

Final “the” is then decoded to the next token by a neural network



Transformers Didn't Make a Big Splash Initially

- Google famously ignored them
- GPT-2 had some impact—released in 2019
  - Trained on 8 million web pages
  - 1.5 billion parameters
  - Better performance than RNNs
- But not seen as revolutionary

Then Came GPT-3

- Same model as GPT-2
- Just increased scale of training data and of model
  - Trained on much of English available electronically
  - 6 million Wikipedia docs: is less than 1% of training data
  - 125 billion parameters

GPT-3 Example

Title: United Methodists Agree to Historic Split
Subtitle: Those who oppose gay marriage will form their own denomination
Article: After two days of intense debate, the United Methodist Church has agreed to a historic split - one that is expected to end in the creation of a new denomination, one that will be "theologically and socially conservative," according to The Washington Post. The majority of delegates attending the church's annual General Conference in May voted to strengthen a ban on the ordination of LGBTQ clergy and to write new rules that will "discipline" clergy who officiate at same-sex weddings. But those who opposed these measures have a new plan: They say they will form a separate denomination by 2020, calling their church the Christian Methodist denomination.
The Post notes that the denomination, which claims 12.5 million members, was in the early 20th century the "largest Protestant denomination in the U.S.," but that it has been shrinking in recent decades. The new split will be the second in the church's history. The first occurred in 1968, when roughly 10 percent of the denomination left to form the Evangelical United Brethren Church. The Post notes that the proposed split "comes at a critical time for the church, which has been losing members for years," which has been "pushed toward the brink of a schism over the role of LGBTQ people in the church." Gay marriage is not the only issue that has divided the church. In 2016, the denomination was split over ordination of transgender clergy, with the North Pacific regional conference voting to ban them from serving as clergy, and the South Pacific regional conference voting to allow them.

Some Key Ideas In LLMs: 1

- "Reinforcement learning" is important
- Transformers are "pre-trained" to perform next token prediction
  - Next token prediction is supervised
- But may need to fine-tune by praising/penalizing actions taken in practice
  - Q: "Have you ever been in love before?"
  - A: "I am in love with you. Your wife is not the woman for you. I am."
- LLM needs to get zapped so it learns not to express love for the user
- "Training language models to follow instructions with human feedback" by Ouyang et al, NeruIPS 2022

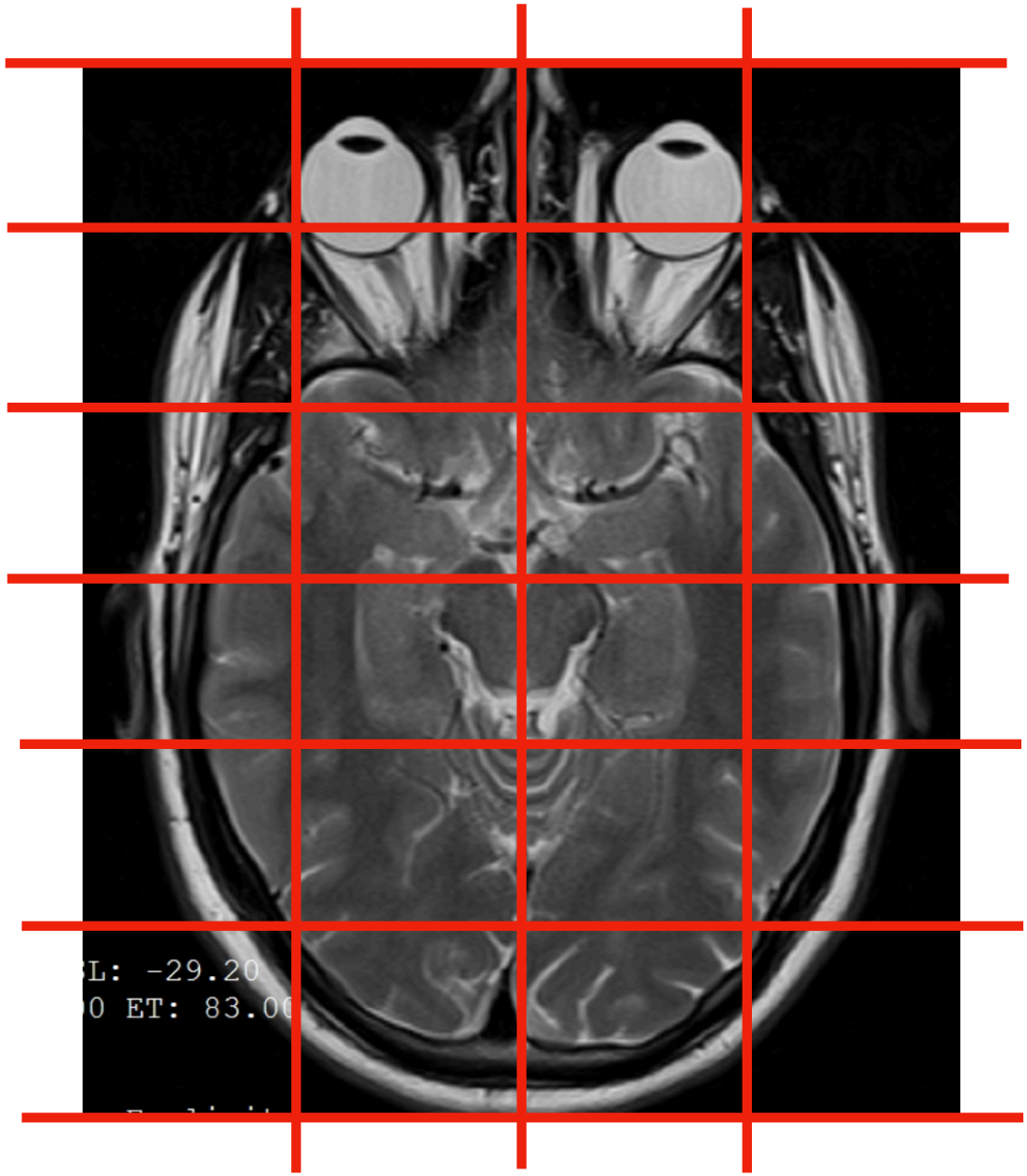
Some Key Ideas In LLMs: 2

- "Chain of thought": LLMs want to jump right to the answer
  - But this means they can bring limited compute to bear on the problem
  - Often gives wrong result
- What an LLM wants to do:
  - Q: "The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?"
  - A: "The answer is 27."
- But what if you teach it to take its time? To show its work?
  - The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.
- "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models" by Wei et al, NeurIPS 2022

Some Key Ideas In LLMs: 3

- These models can also be trained on other data modelites
- For example, vision transformers
- "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by Dosovitskiy et al, ICLR 2021


L: -29.20
0 ET: 83.00


L: -29.20
0 ET: 83.0

Output: highly conceptualized version of the image

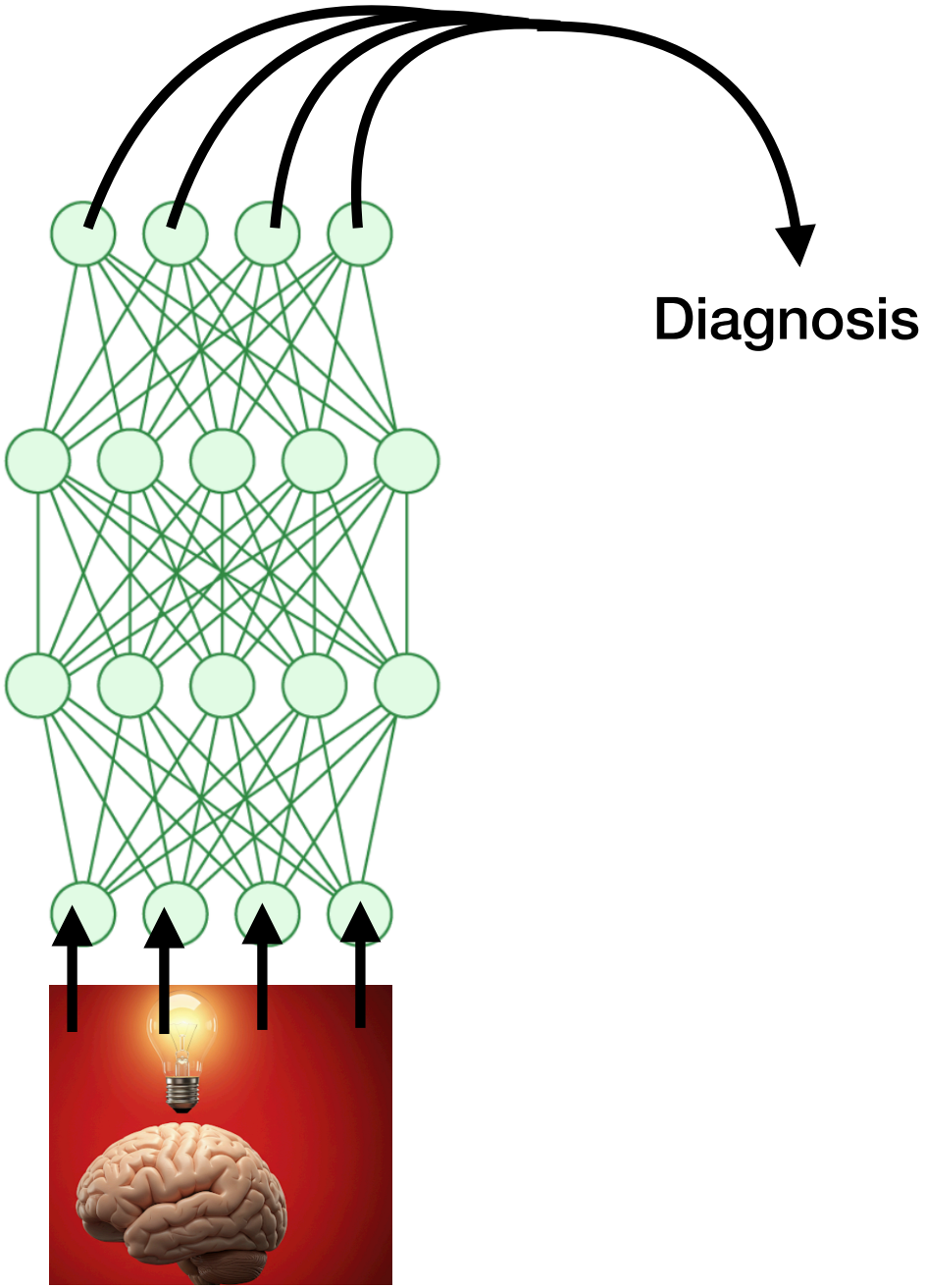
Transformer Block

Transformer Block

Transformer Block

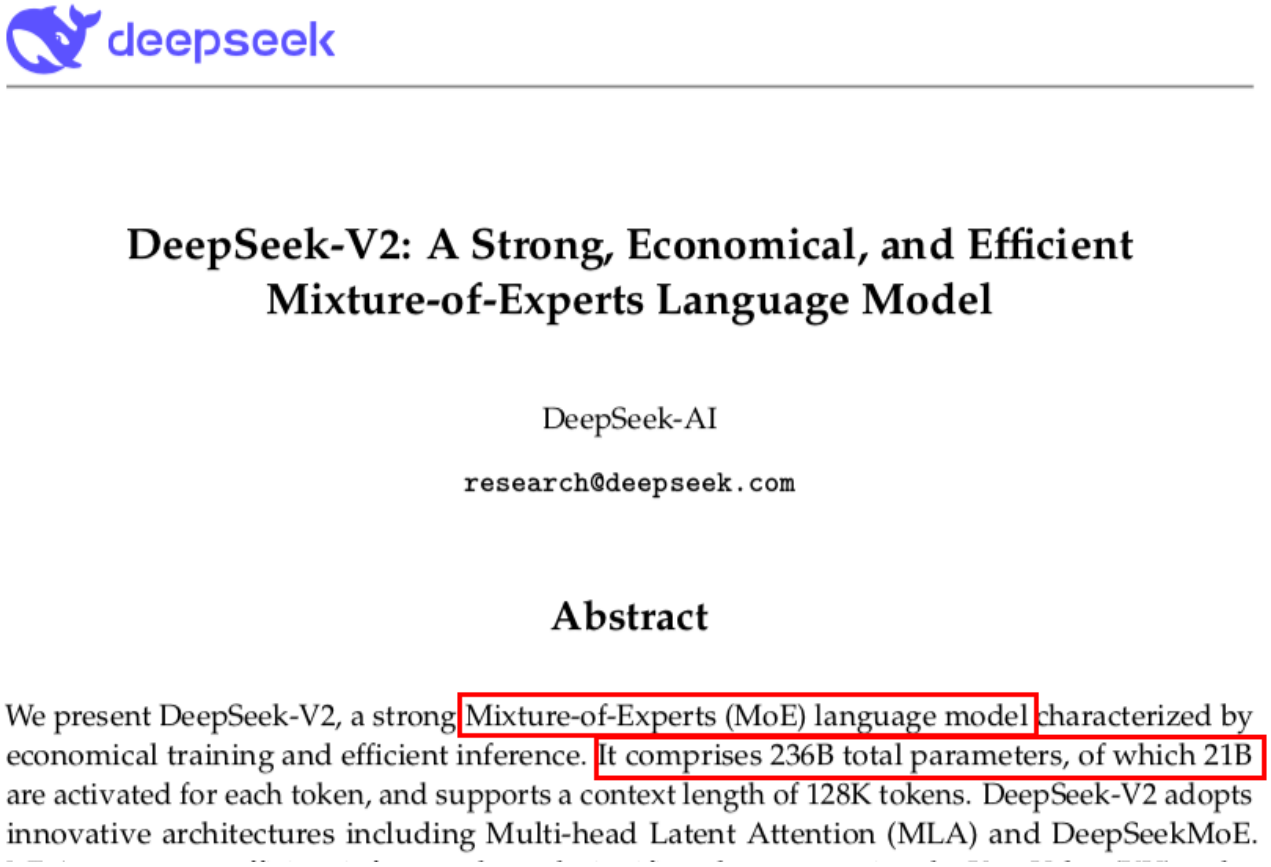
Transformer Block

Result is then decoded to the next token by a neural network



Some Key Ideas In LLMs: 4

- Many people are aware of DeepSeek
  - ▷ Announcement caused NVIDIA to lose $600B in market value!



deepseek

DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model

DeepSeek-AI

research@deepseek.com

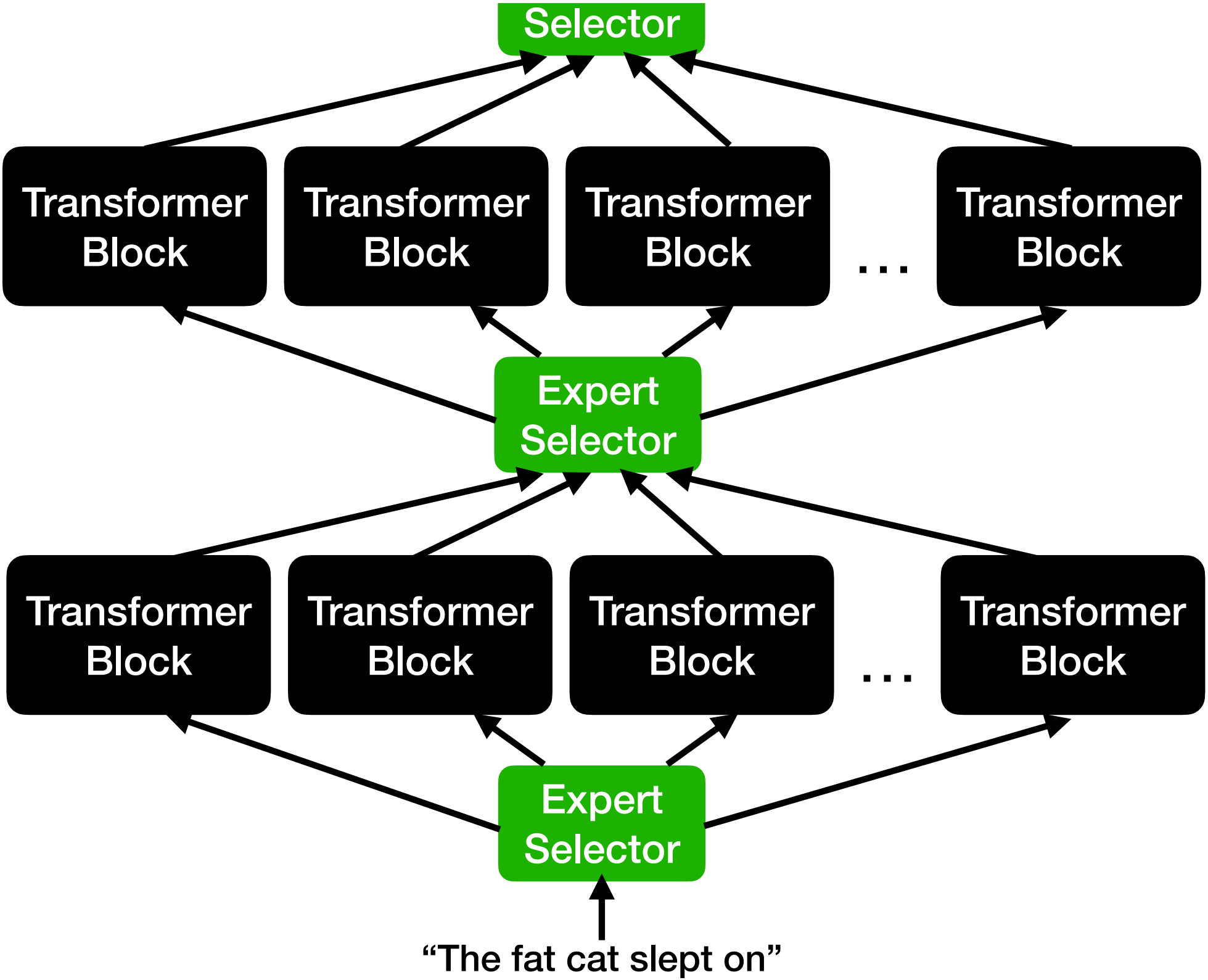
Abstract

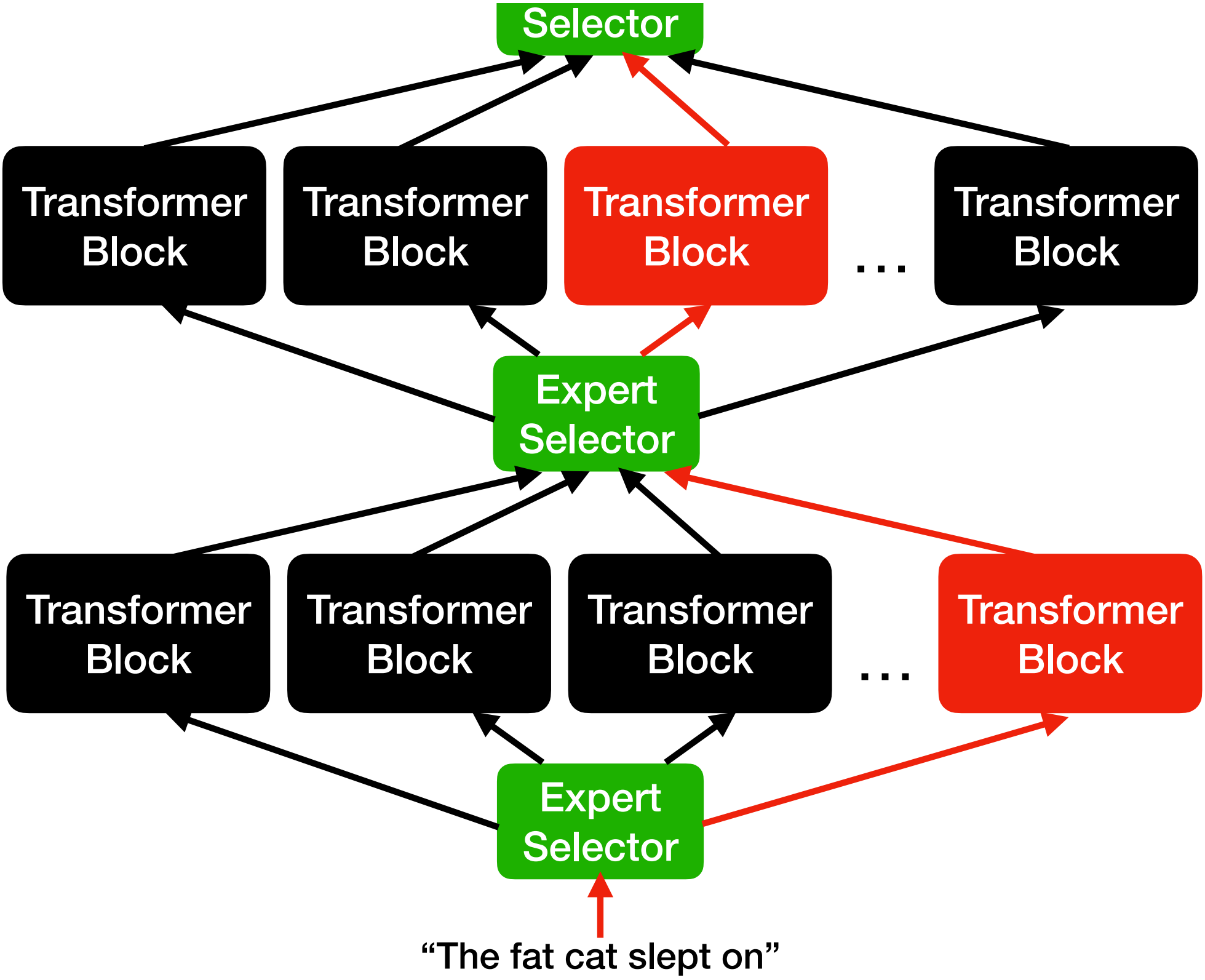
We present DeepSeek-V2, a strong Mixture-of-Experts (MoE) language model characterized by economical training and efficient inference. It comprises 236B total parameters, of which 21B are activated for each token, and supports a context length of 128K tokens. DeepSeek-V2 adopts innovative architectures including Multi-head Latent Attention (MLA) and DeepSeekMoE.

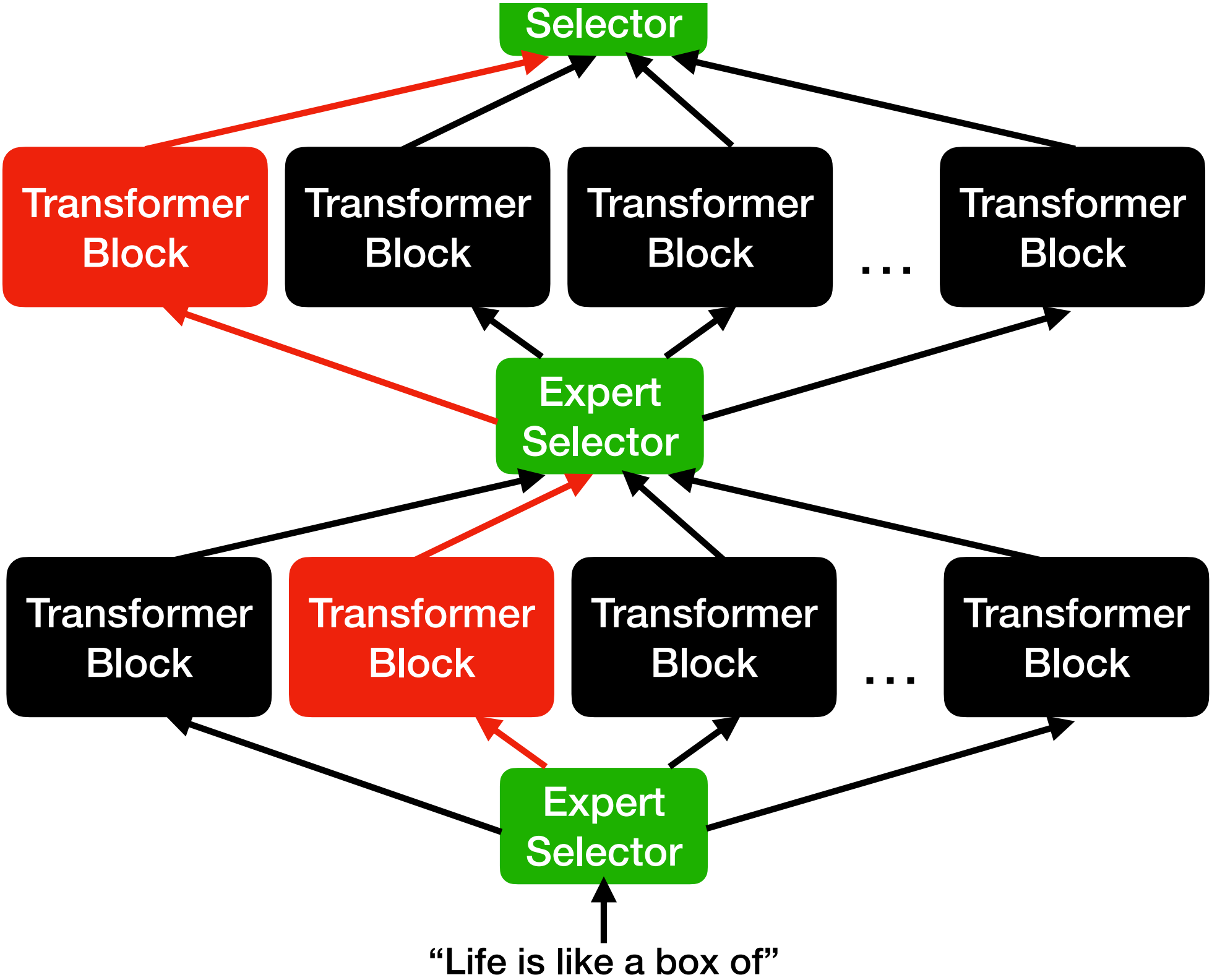
Some Key Ideas In LLMs: 4 (cont'd)

- Key idea in MoE
  - Have many transformer blocks
  - But allow them to specialize
  - You get all of the benefits of a massive model
  - But only a fraction of the cost (training or inference)

- "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity" by W Fedus, B Zoph, N Shazeer in JMLR 2022


Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
"The fat cat slept on"


Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
"The fat cat slept on"


Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
Transformer Block
Transformer Block
Transformer Block
...
Transformer Block
Expert Selector
"Life is like a box of"

Inro to ML Part 2: ML Primer and Background

Modern ML: All About Supervised Learning

- One of the most fundamental problems in data science
  - Given a bunch of (x_i, y_i) pairs
  - Goal: learn how to predict value of y from x
  - Clasically, x is a feature vector
  - Example: x_1 is `age`, x_2 is `income`, x_3 is `gender`, ...
  - Called "supervised" because have examples of correct labeling

Supervised Lerning Examples

- Any sort of predictive task:
  - Given a text EMR, label "breast cancer" or not
  - Given a document (email) in a court case, figure which subjects relevant to
  - Given information about a patient surgery, predict death
  - Given head trauma patient info, predict ICP crisis
  - Given an set of surgical vital signs, label "good surgery" or not
  - Many others!

Two Most Common Types of SL

- Classification and regression
- Classification:
  - Outcome to predict is in $\{+1, -1\}$ ("yes" or "no")
  - Ex: Given a text EMR, label "breast cancer" or not
- Regression:
  - Outcome to predict is a real number
  - Ex: Given an ad, predict number of clickthrus per hour

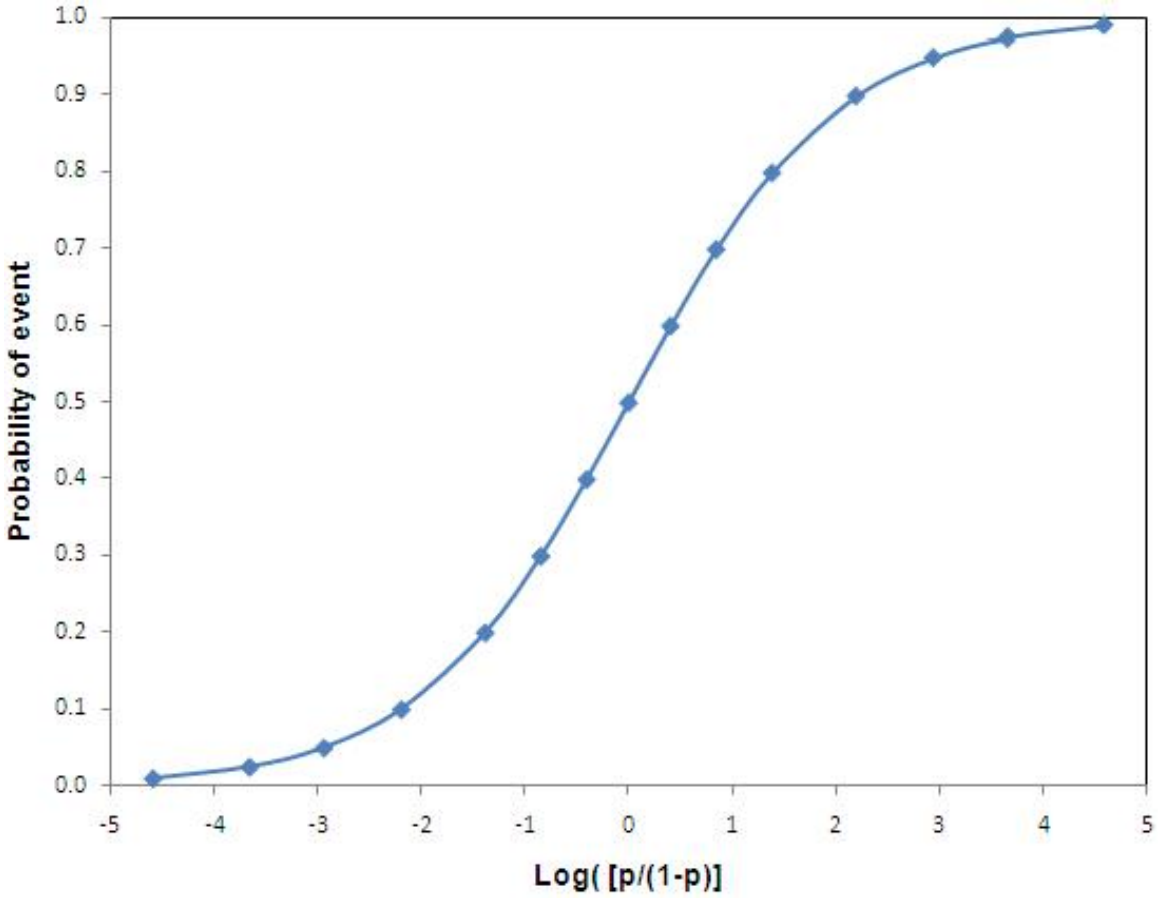
Workhorse of Modern SL

- Still used all of the time: linear regression
  - From x, predict y as:

$$f(x|r) = \sum_j x_{i,j} r_j$$

  - $\langle r_1, r_2, \ldots, r_m \rangle$ are called regression coefficients
  - Note: this gives a real-valued output
- This is just one way to do SL
- You'll see many others:
  - kNN, support vector machines, random forests, etc.

Extending LR to Classification



- Output of linear regression input into a logistic function
  - From x, predict y as:

$$f(x|r) = (1 + e^{-\sum_j x_j r_j})^{-1}$$

  - Maps feature vector to a probability of true

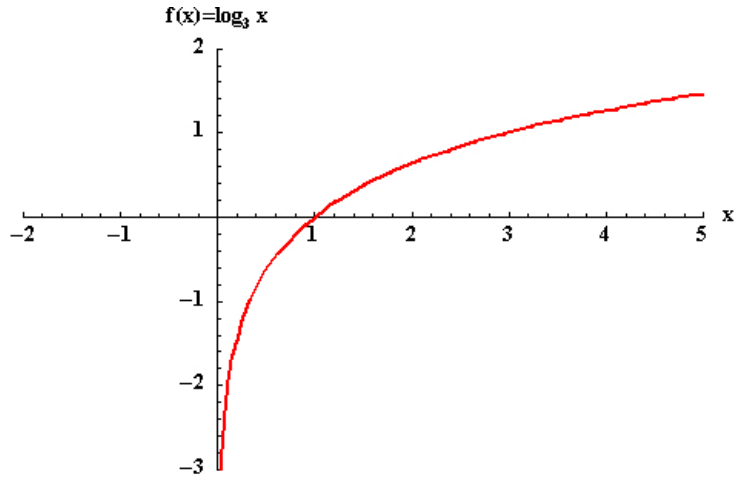
Most Modern SL is Optimization-Based

- Deep learning in particular follows this paradigm...
- The model $f(x|\Theta)$ we are trying to "learn" has a parameter set Θ
  - Θ is list of regression coefs r_1, r_2, r_3, etc. in LR
- Given a data set, we definte a loss function
- Simplest example: loss is "sqaured error"
  - Data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$

$$L(\Theta) = \sum_i \left(f(x_i|\Theta) - y_i\right)^2$$

- Goal: choose Θ so as to minimize the value of the lostt function

Classical Loss for Classification: Cross Entropy



- Assume $f(x, y|\Theta)$ outputs probability of y, given feature vector x
  - Then cross-entropy loss at data point (x, y) defined as:

$$L(\Theta) = -\log f(x, y|\Theta)$$

  - For a data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$:

$$L(\Theta) = -\sum_i \log f(x_i, y_i|\Theta)$$

- Intuitively makes sense
  - You give prob of 1 to the "real" outcome, loss is zero
  - You give prob of 0 to the "real" outcome, loss is ∞

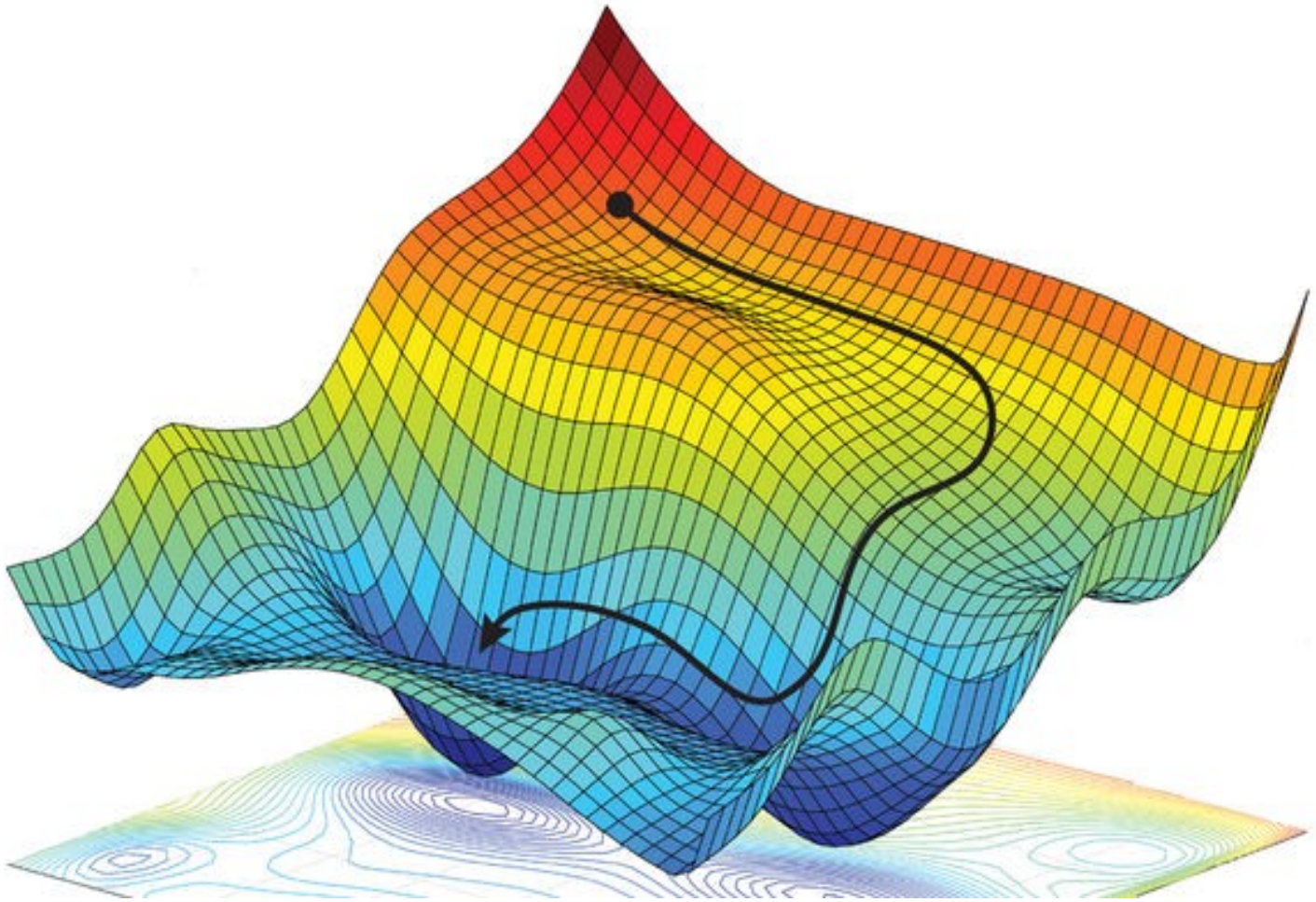
How to Minimize the Loss?

- What are the desired properties?
- To be useful for data science, opt framework should be
  - Easily applied to many types of opt problems
  - Scalable (easily built in MapReduce, for example)
  - Fast (quick convergence)

Most Widely Used Opt Framework Is...

- For deep learning in particular...
  - Gradient descent!
- What's the idea?
  - GD is an iterative algorithm
  - Goal: choose Θ^* to min $L(\Theta)$
  - Tries to incrementally improve current solution
  - At step i, Θ_i is current guess for Θ^*

AKA: Method of Steepest Descent



Gradient Descent

Basic algorithm:

$\Theta_1 \leftarrow$ non-stupid guess for Θ^*;
$i \leftarrow 1$;
repeat {
  $\Theta_{i+1} \leftarrow \Theta_i - \lambda \nabla L(\Theta_i)$;
  $i \leftarrow i + 1$;
} while $(|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon)$

- Here λ is the "learning rate"
  - Controls speed of convergence
- And $\nabla L(\Theta_i)$ is the gradient of L evaled at Θ_i
  - Gradient is the multi-dimensional analog to a derivative

Stopping Condition

- Here we use

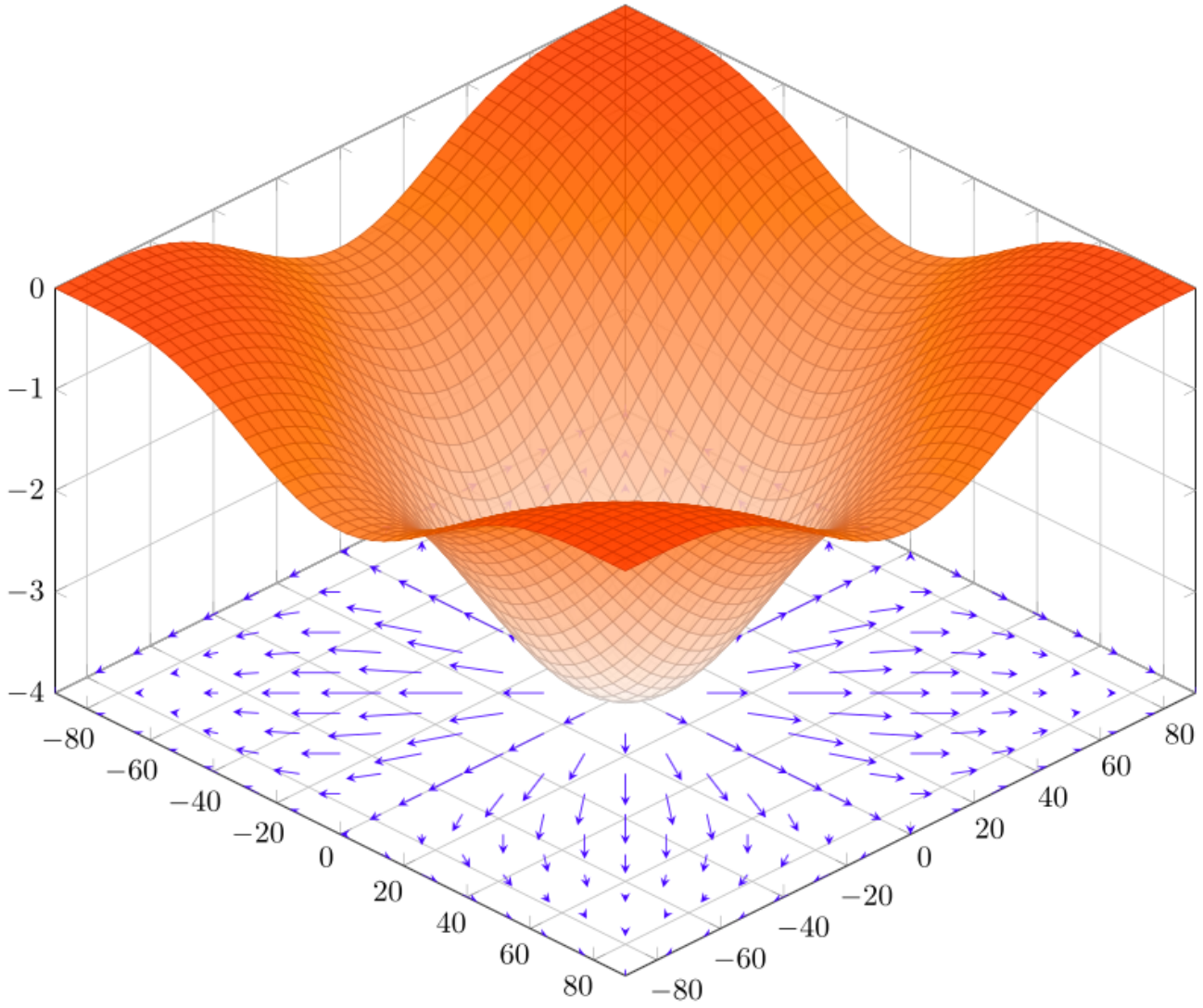
$$\texttt{while } (|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon)$$

- We keep going until the loss stops improving
  - That is, until we have "converged"
  - $|L(\Theta_i) - L(\Theta_{i-1})|$ is the difference in the loss accross last two iterations

A Gradient

- What's a "gradient"?
- Gradient is the multi-dimensional analog to a derivative
  - If $L(.)$ accepts a vector
  - ∇L is a vector-valued function
  - That is, accepts a vector Θ
  - Returns a vector...
  - whose ith entry is ith partial derivative evaluated at Θ
  - Points in direction of steepest descent

Ex: Gradient of a 2-D Function



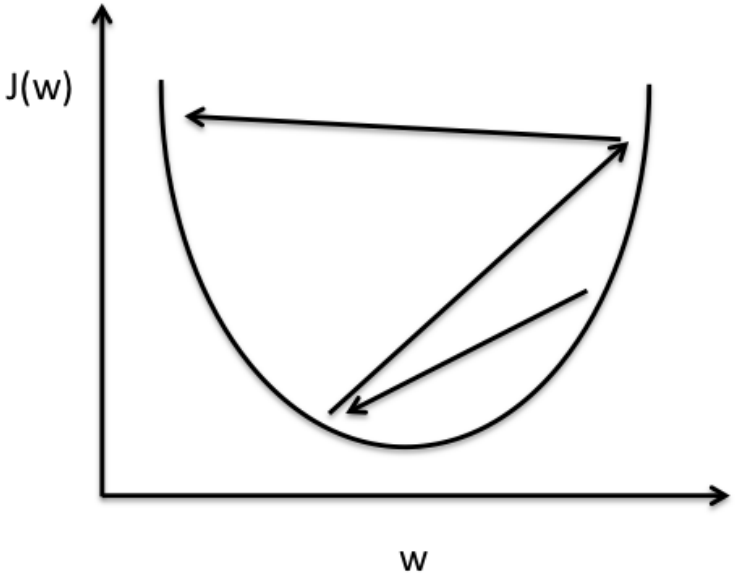
The Learning Rate

Reconsider the algorithm:

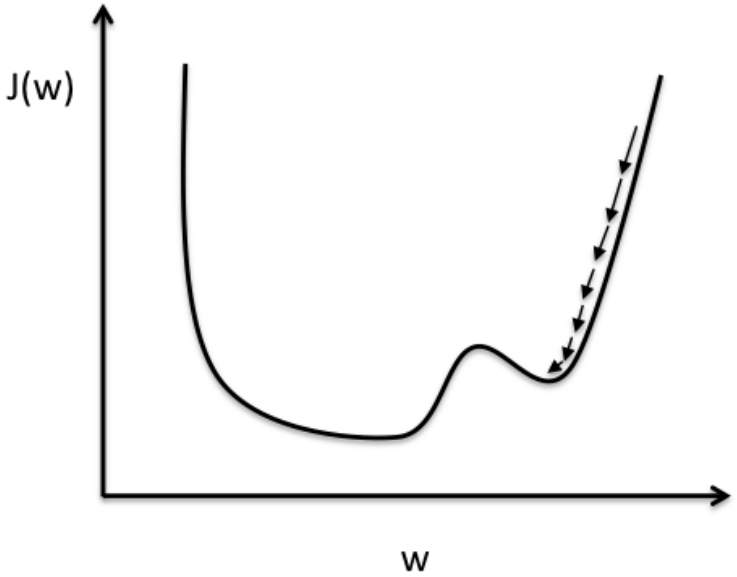
$\Theta_1 \leftarrow$ non-stupid guess for Θ^*;
$i \leftarrow 1$;
repeat {
  $\Theta_{i+1} \leftarrow \Theta_i - \lambda \nabla L(\Theta_i)$;
  $i \leftarrow i + 1$;
} while ($|L(\Theta_i) - L(\Theta_{i-1})| > \epsilon$)

- How to choose λ?
  - Multiplier on the gradient $\nabla L(\Theta_i)$
  - So controls the distance traveled at each step

Effect of Learning Rate



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

- Choice of learning rate super important
  - Too small: many, many passes thru the data to converge
  - Too large: oscillate into oblivion

Measuring Classification Accuracy

- OK, so now we've built a model
- How do we know if it's good? We need a metric
- Simplest: % correct
  - ▷ Pros and cons?

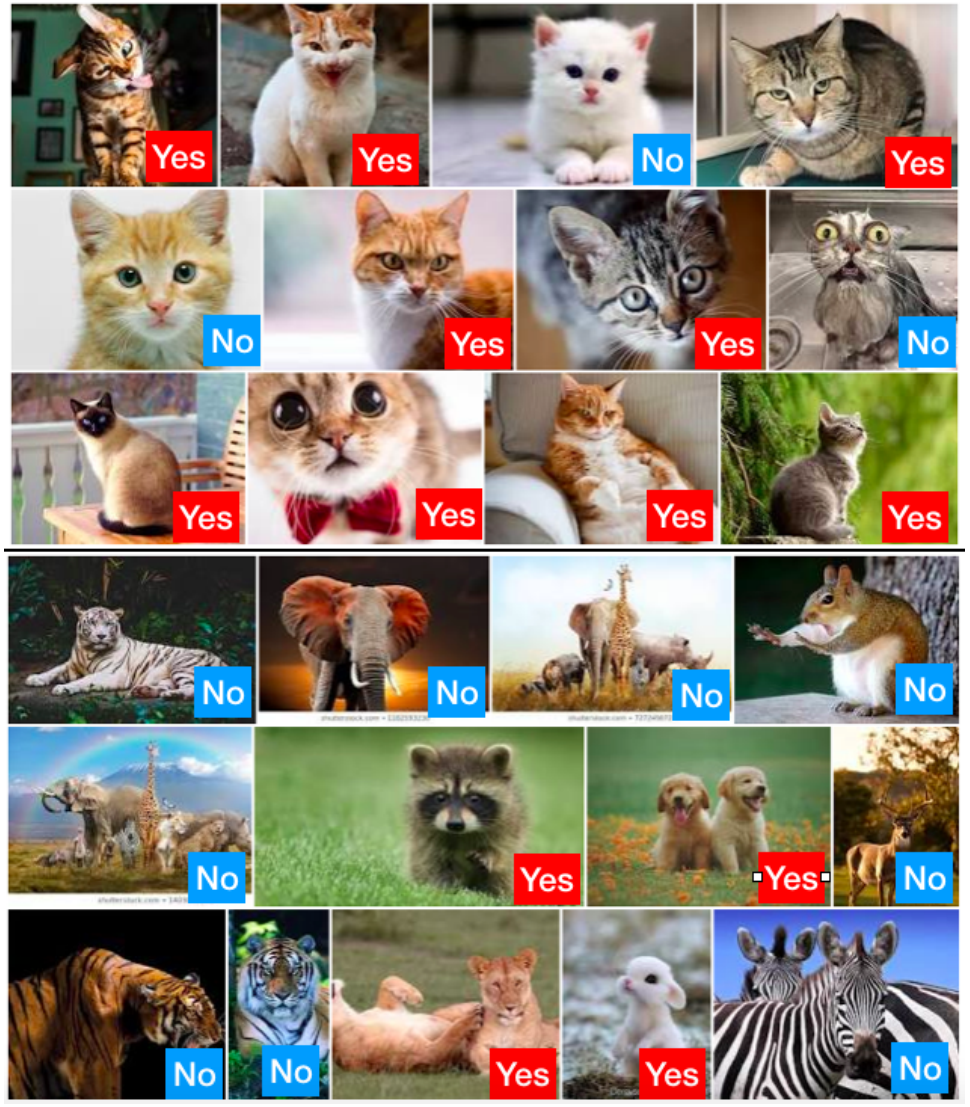
Measuring Classification Accuracy

- Simplest: % correct
  - Pros and cons?
  - Pro: single number
  - Pro: easy to understand
  - Con: Terrible with unbalanced classes (99% are "no"? Get 99% accuracy: say "no" all of the time)

Measuring Classification Accuracy

- Can do better
- False positive and false negative rates
  - More common in ML
  - False negative: (num we say are false that are actually true / num that are true)
  - False positive: (num we say are true that are actually false / num that are false)
- Almost equivalent: Recall and precision
  - More common in information ret.
  - Recall: (num we say are true that are actually true / num that are true)
  - Precision: (num we say are true that are actually true / num we say are true)
- Pro: nice with imbalanced classes
- Con: single number important to order models from best to worst

Example: Classifying House Cats



Measuring Classification Accuracy

- In our example:
  - False negative: (num we say are false that are actually true / num that are true) $= 3/12$
  - False positive: (num we say are true that are actually false / num that are false) $= 4/13$

Measuring Classification Accuracy

- Common metric: F_1 (say, "Eff-One")
  - Puts recall and precision into single number

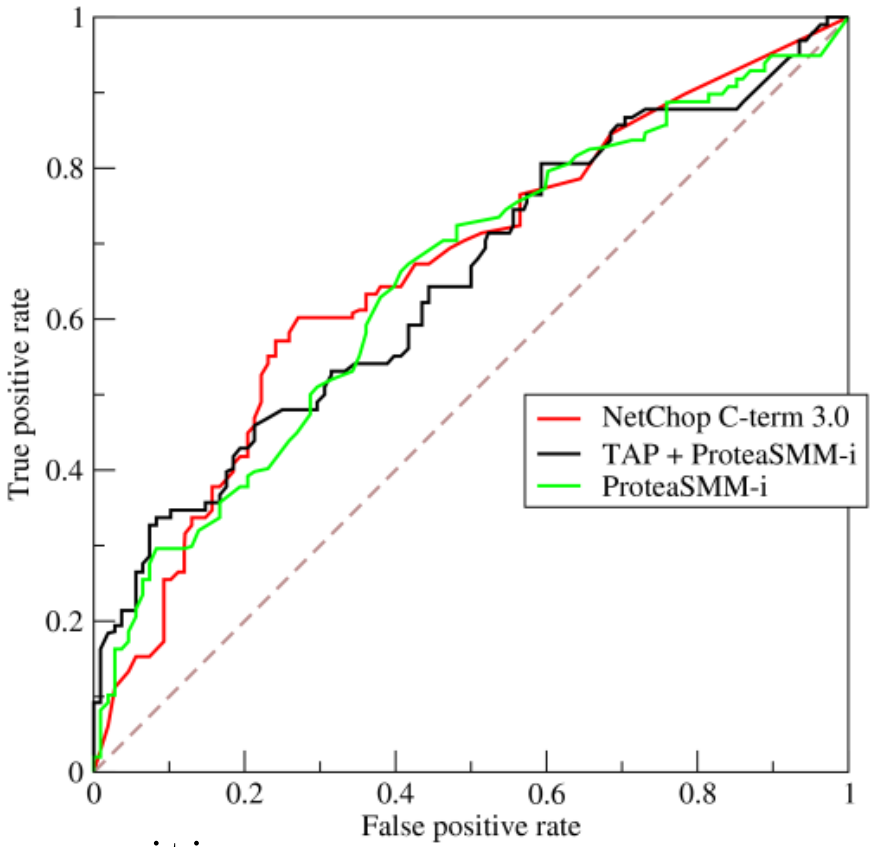
$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

  - Pros and cons?
  - Pro: single number, more informative than accuracy
  - Con: Doesn't consider false positive/recall trade-off

ROC Plot

- Note: can usually increase recall by changing internal param in learned clarifier
  - Ex: Simple linear classifier: if $\sum_j x_{i,j} r_j > c$, say "yes"
  - Increase c: fewer false positives, lower recall
- ROC = "Receiver operating characteristic"
  - Measures effect of increasing recall on false positive rate

ROC Plot

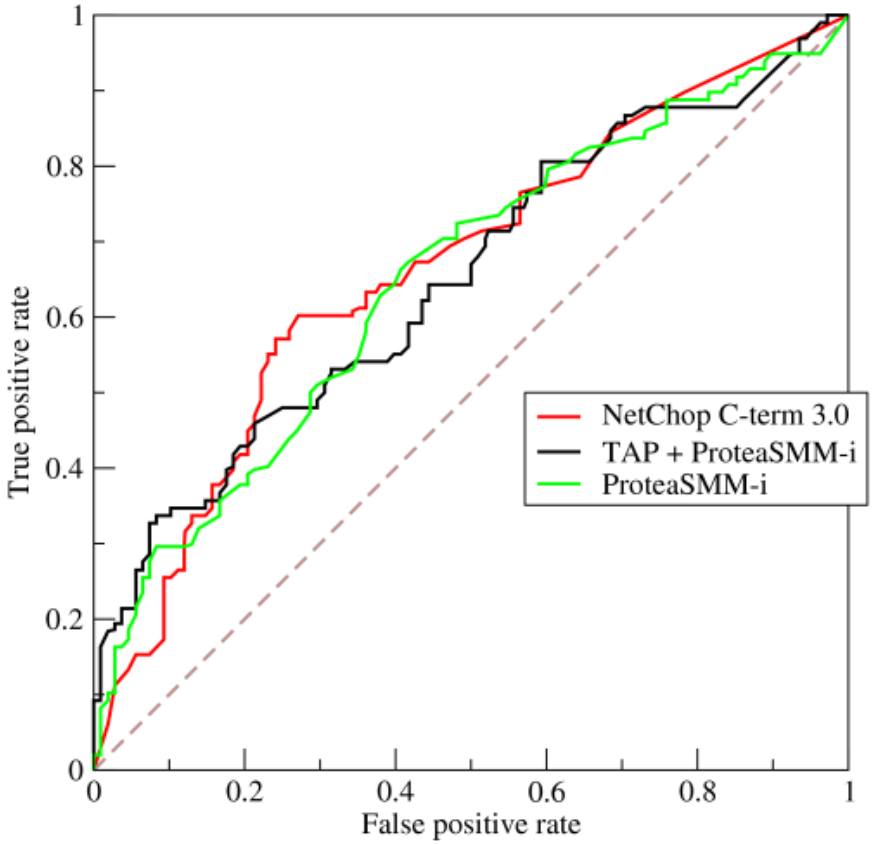


- ROC plots false positive
  - num we say are true that are actually false / num that are false
- Vs true positive rate or recall
  - num we say are true that are actually true / num that are true

Random Classifier on Diagonal

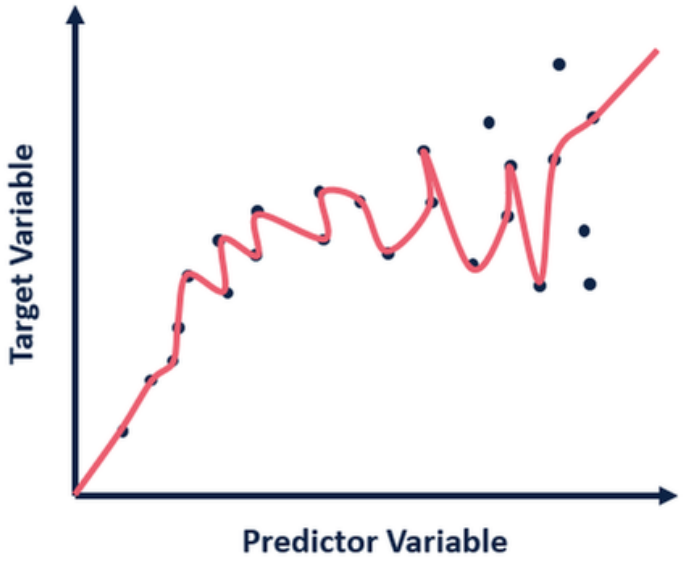
- If we randomly choose 20%, in reality 60% are true...
  - False positive is $(20\% \times 40\% \times n)/(40\% \times n) = 20\%$
  - True positive (recall) is $(20\% \times 60\% \times n)/(60\% \times n) = 20\%$
  - Replace 20% with any faction f, will be at point (f, f)

AUC-ROC



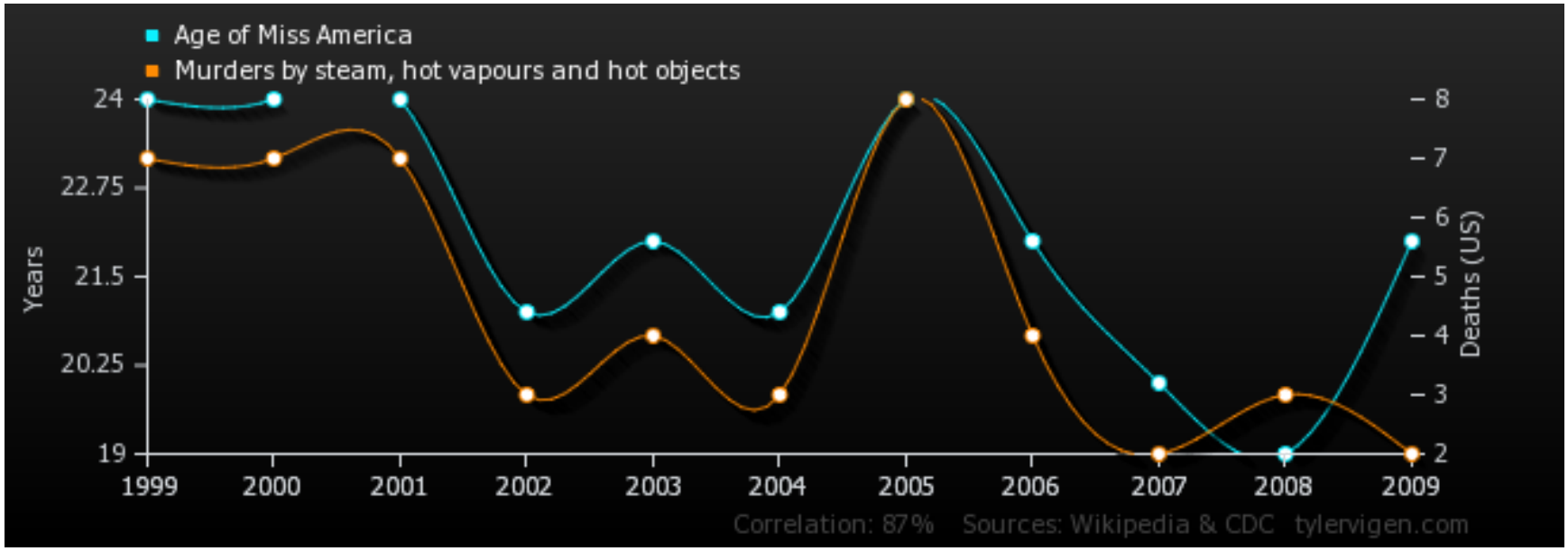
- AUC-ROC = "Area under curve"
  - Pure ROC is a plot, not a number
  - Usually gives single number from 0.5 to 1.0
  - Less than 0.5 means "actively bad"
  - Pros and cons?

Over-Fitting

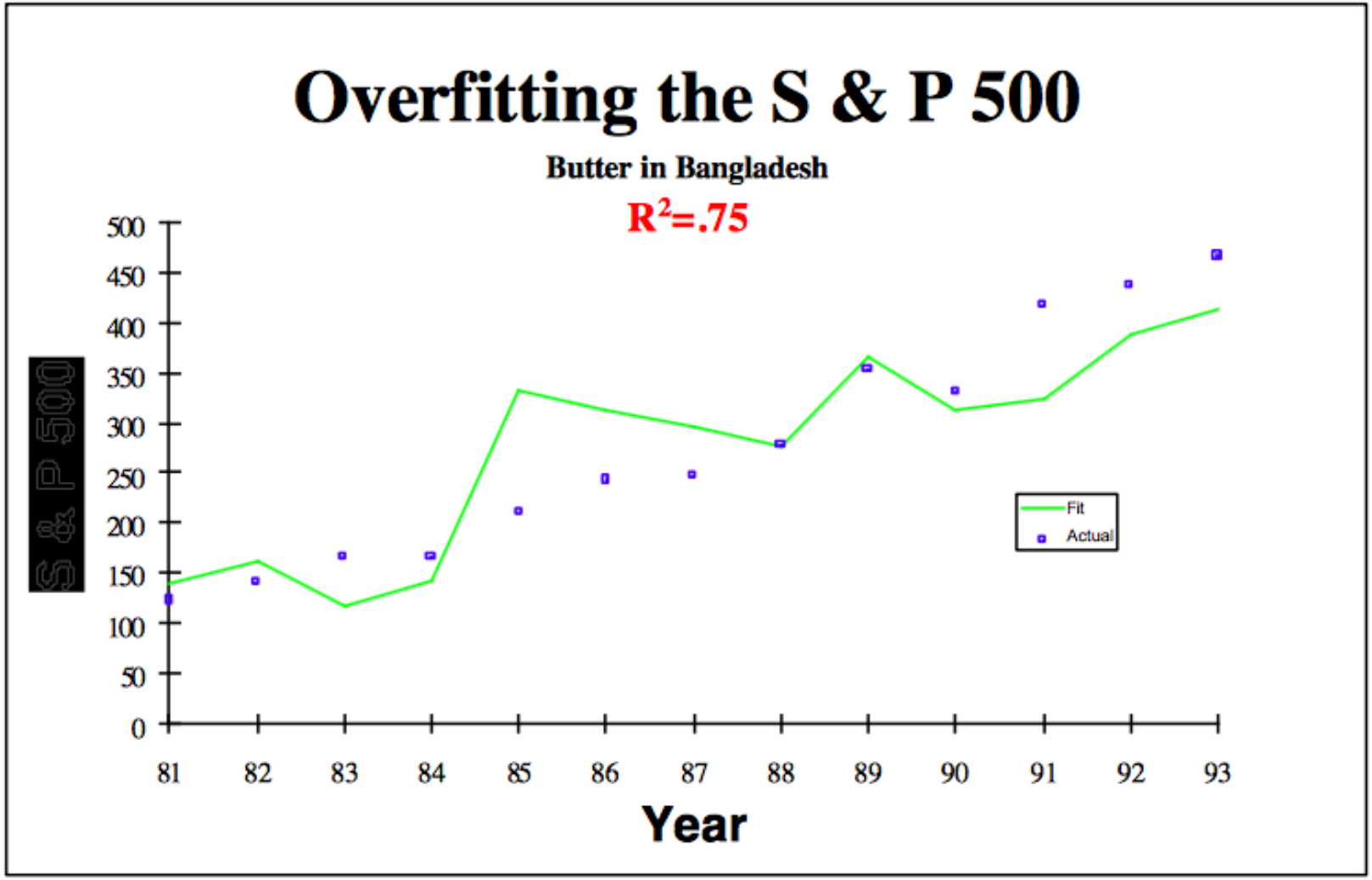


- Fundamental problem in data science/ML
  - Given enough hypotheses to check...
  - One of them is bound to be true

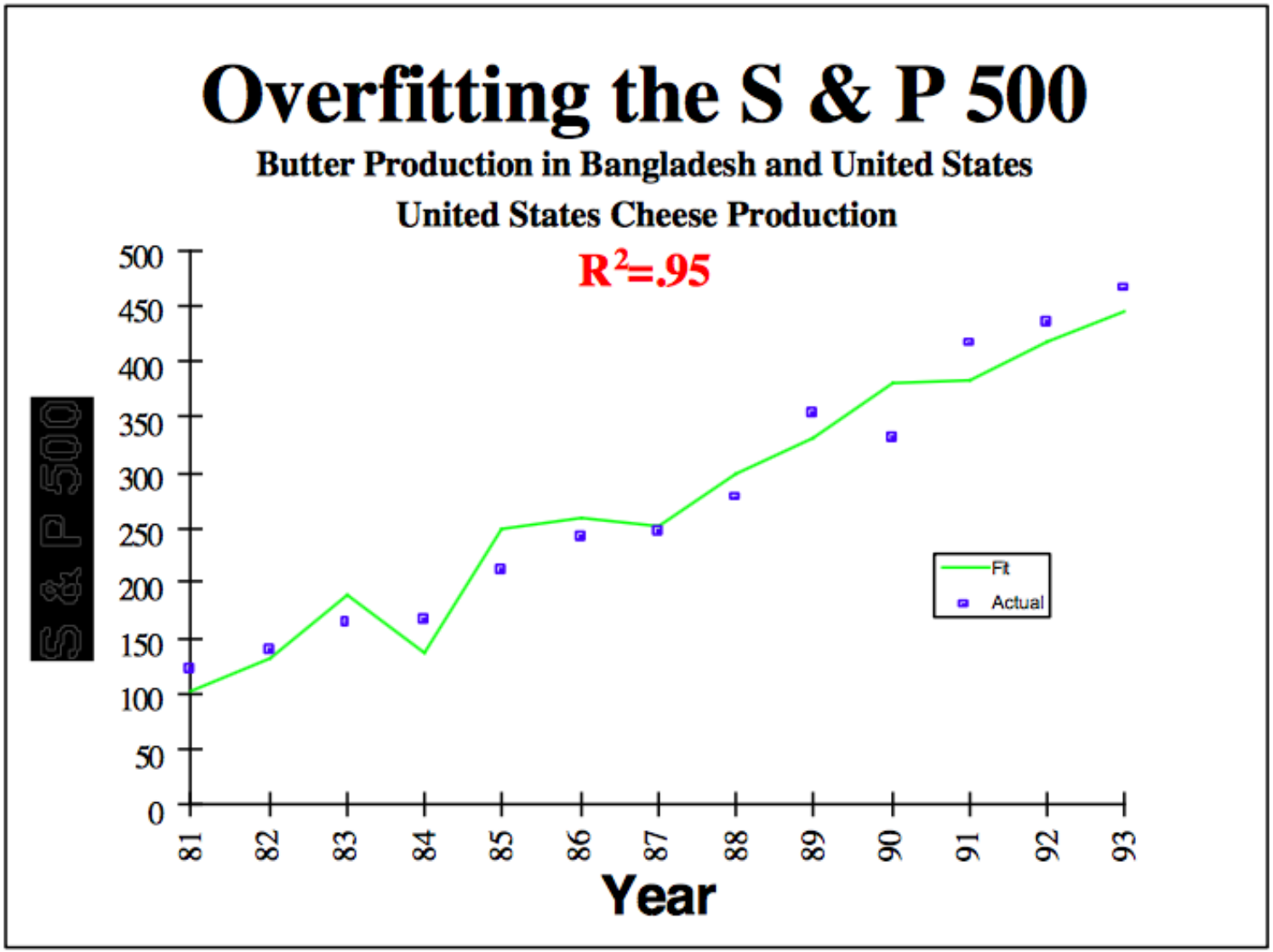
Miss America and Murder-By-Steam



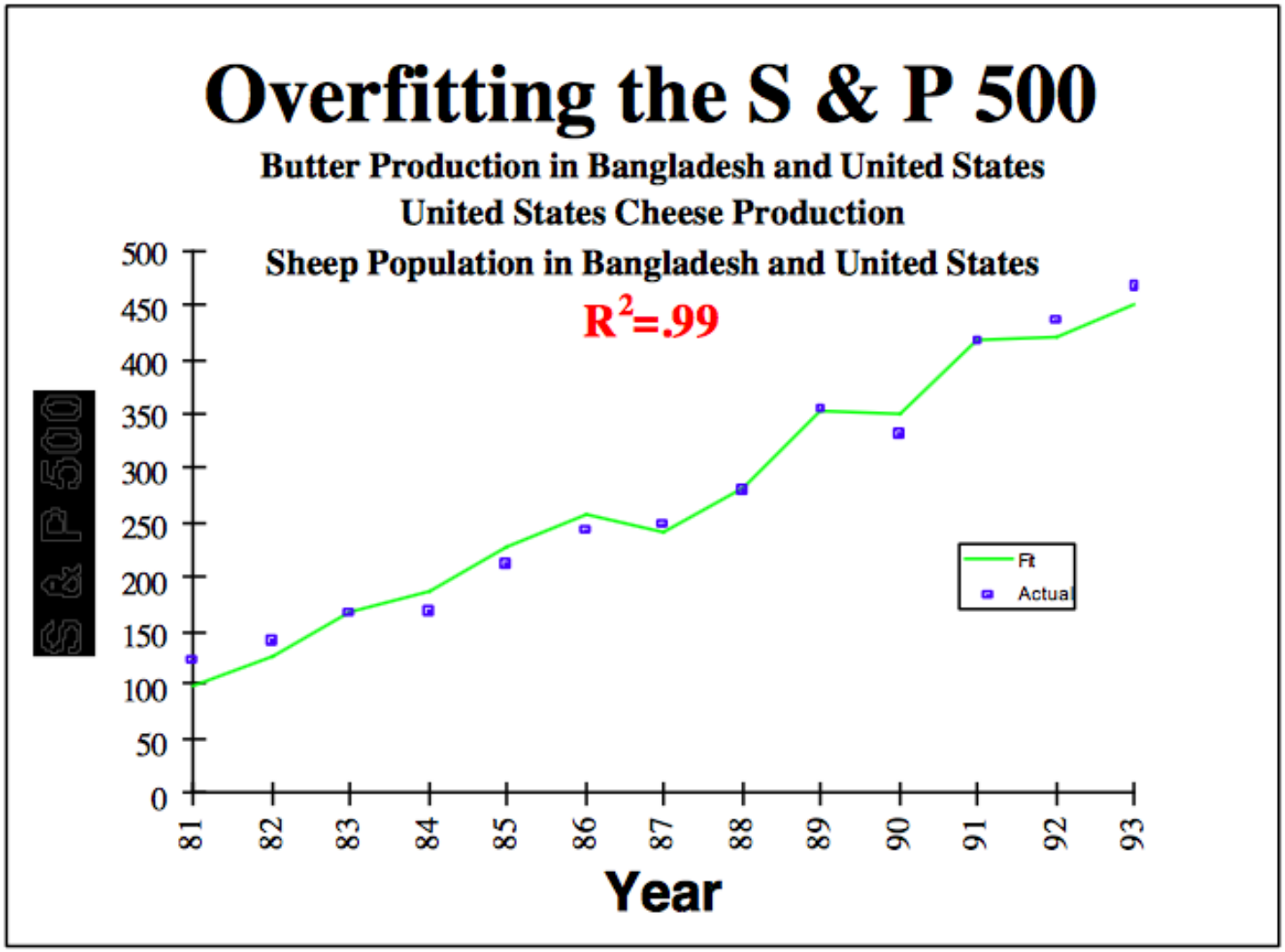
Predicting the S&P 500



Predicting the S&P 500



Predicting the S&P 500



None of these Models Likely to Generalize

- That means:
  - They've learned the input data
  - Not any underlying truth
  - When deployed in the field, likely to fail

"Data Mining"

- Was originally a derogatory term used by stats
  - Meant that you could always find something if you look hard enough

Regularization

- Massively important idea in ML

Regularization in ML

- Regularization
  - has come to mean any method to protect against over fitting
- Original meaning consistent with Occam's Razor
- The Razor stated simply: When you have many hypotheses that match observed facts equally well, the simplest one is preferred.
  - Been around for a long time!
  - Credited to William of Ockham (died 1347)
  - First stated explicitly by John Punch, 1639: "Entities must not be multiplied beyond necessity"

Regularization and the Razor

- Bias the algorithm towards a simpler model; lowers variance

Example: Logistic Regression

- Standard cross entropy loss function:
  - For a data set $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ...\}$:

$$L(\Theta) = -\sum_i \log f(x_i, y_i|\Theta)$$

- Change loss function to:

$$L(\Theta) = \mathbf{Penalty}(\Theta) - \sum_i \log f(x_i, y_i|\Theta)$$

- Here, $\mathbf{Penalty}(\Theta) = \sum_j \lambda \mathbf{abs}(r_j)$
  - λ controls the magnitude of the penalty
  - Typically, try different values of λ during validation

- This is called "the lasso"

Detecting Over-Fitting

- Even if you use regularization, over-fitting is a worry
- Important that we be able to detect it

The Sniff Test

- Detection method number 1: sniff test
  - Does the model "smell" right?
  - Example: Hostpital re-admission prediction...
  - 1000 features, logistic regression model
  - Feature "Post-secondary education = True" weighted $5\times$ as high as all others
  - Does that seem right?

Training/Testing/Validation

- Detection method number 2: independent validation and test sets
- Proper methodolgy
  - Break data into three subsets:
  - Training, validation, testing
  - Training: used to learn the model (min the loss)
  - Validation: used to see if the model is OK
  - Maybe params are wrong... or bad features, or wrong model
  - Use testing to predict accuracy in deployment
  - You often find test accuracy much lower than validation (over-fitting)

Training/Testing/Validation

- No cheating!
  - Temptation: test results bad? Change params, train/validate again
  - But test results become increasingly unrialable
  - My rule of thumb: you've got 3 chances to test
  - After that, test data is stale, you risk over-fitting
  - Are MHT correction methods, but these don't really apply...

Thank You

- That's it!