Neural language models

CS 5624: Natural Language Processing

Spring 2025

https://tuvllms.github.io/nlp-spring-2025

Tu Vu

VIRGINIA TECH.

Logistics

- Office hours (both in-person and via Zoom) have started this week. Zoom links are available on Piazza.

- HW0 was released on Piazza (due February 7th)

Final project

- Groups of ~~2-3~~ **4-5**; all groups should be formed by this Friday, January 31st
- A Google form for submitting group information will be available on Piazza after today's lecture
- Search for teammates on Piazza https://piazza.com/class/m63qacreewc2fs/post/5

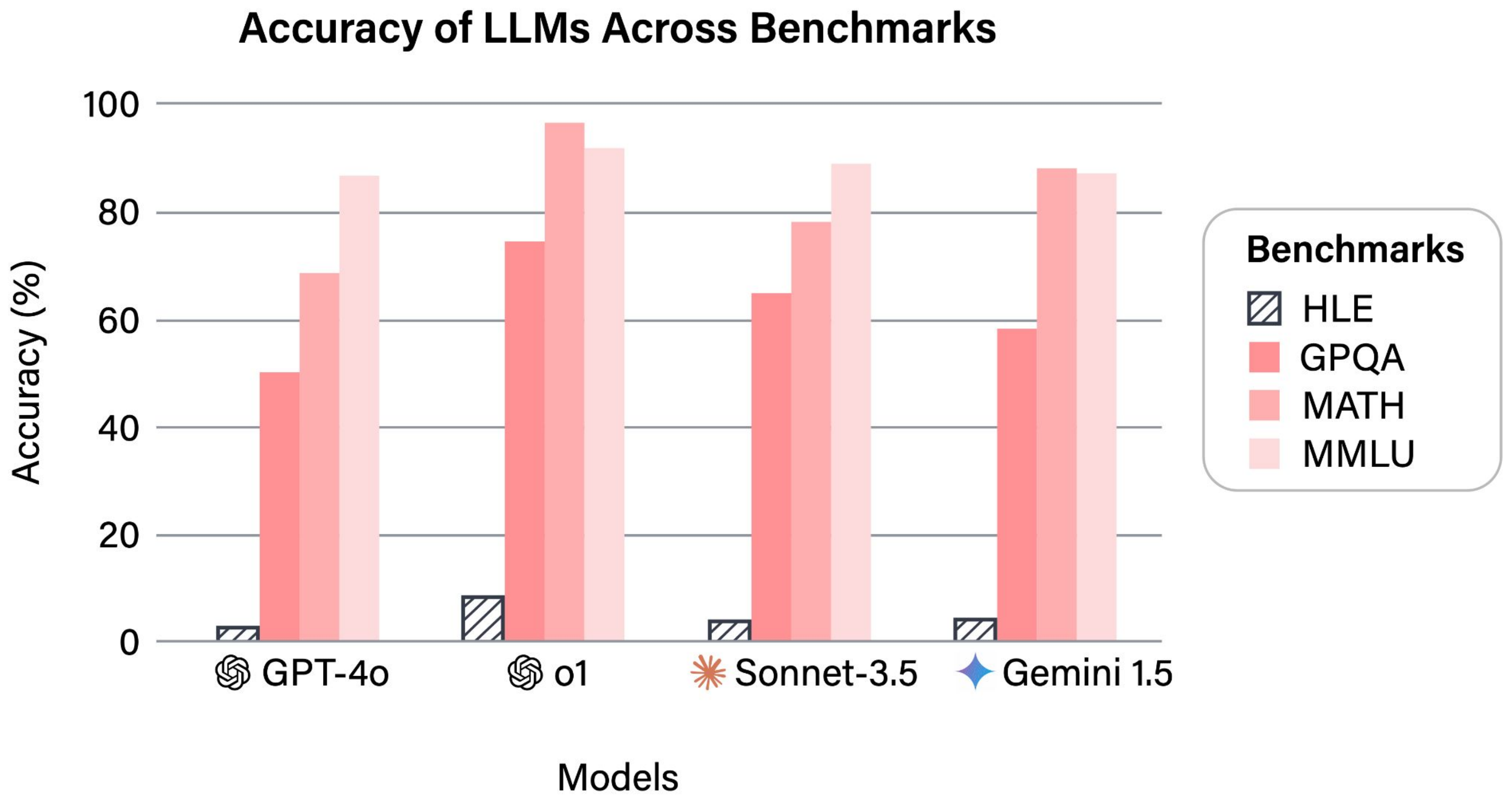
  or reach out to us at cs5624instructors@gmail.com

Final project (cont'd)

- Two deliverables:
  - 10% project proposal: 3+ pages, due February 21st
  - 30% final report: 8+ pages, due last day of classes

Final project ideas

- New challenging datasets
  - Instruction following, factuality, attribution, reasoning, math, code, safety, etc.
  - E.g., Humanity's Last Exam https://arxiv.org/abs/2501.14249



Accuracy of LLMs Across Benchmarks
Accuracy (%)
100
80
60
40
20
0
GPT-4o
o1
Sonnet-3.5
Gemini 1.5
Models
Benchmarks
HLE
GPQA
MATH
MMLU

Final project ideas (cont'd)

- Revisit inverse scaling tasks
  - https://www.lesswrong.com/posts/iznohbCPFkeB9kAJL/inverse-scaling-prize-round-1-winners
  - https://www.lesswrong.com/posts/DARiTSTx5xDLQGrrz/inverse-scaling-prize-second-round-winners

Final project ideas (cont'd)

- SemEval-2025 tasks
  https://semeval.github.io/SemEval2025/tasks.html
  - semantic relations
  - LLM capabilities
  - fact checking and knowledge verification
  - knowledge representation and reasoning

Final project ideas (cont'd)

- Replicate DeepSeek-R1?

deepseek

DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning

DeepSeek-AI

research@deepseek.com

https://github.com/deepseek-ai/DeepSeek-R1/blob/main/DeepSeek_R1.pdf

Post

lmarena.ai (formerly lmsys.org)
@lmarena_ai

Breaking News: DeepSeek-R1 surges to the top-3 in Arena🐳!

Now ranked #3 Overall, matching the top reasoning model, o1, while being 20x cheaper and open-weight!

Highlights:
- #1 in technical domains: Hard Prompts, Coding, Math
- Joint #1 under Style Control
- MIT-licensed

A massive congrats to @deepseek_ai for this incredible milestone and gift to the community! More analysis below 👇

Category: Overall

Apply filter: Style Control | Show Deprecated

Overall Questions
#models: 195 (100%) #votes: 2,572,591 (100%)

DeepSeek-R1 #3 in Arena Overall!

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization
1	1	Gemini-Exp-1206	1374	+5/-4	22116	Google
1	3	Gemini-2.0-Flash-Thinking-Exp-01-21	1382	+8/-6	6437	Google
3	1	ChatGPT-4o-latest (2024-11-20)	1365	+4/-4	35328	OpenAI
3	1	DeepSeek-R1	1357	+12/-13	1883	DeepSeek
4	1	o1-2024-12-17	1352	+6/-6	9230	OpenAI
4	5	Gemini-2.0-Flash-Exp	1356	+4/-4	20939	Google

Final project ideas (cont'd)

← Post

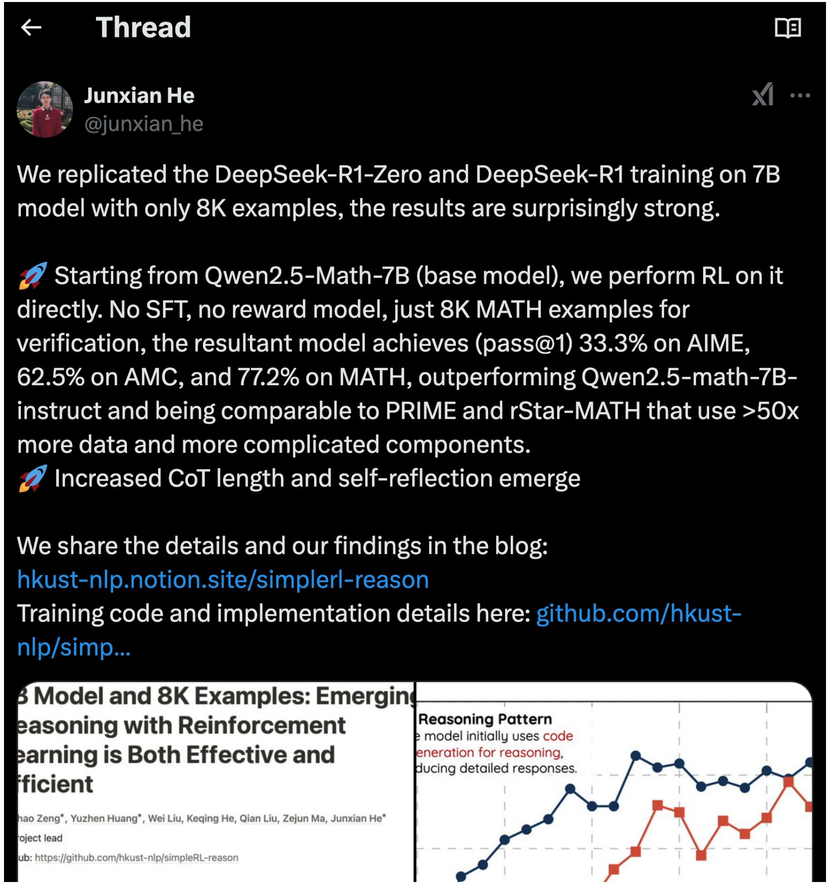
Reply

Denny Zhou @denny_zhou

AGI is finally democratized: RLHF isn't just for alignment—it's even more fun when used to unlock reasoning. Once guarded by a small circle in Silicon Valley, now the secret is for everyone.

2:05 PM · Jan 25, 2025 · **41.5K** Views

18 51 487 118



← Thread

Junxian He @junxian_he

We replicated the DeepSeek-R1-Zero and DeepSeek-R1 training on 7B model with only 8K examples, the results are surprisingly strong.

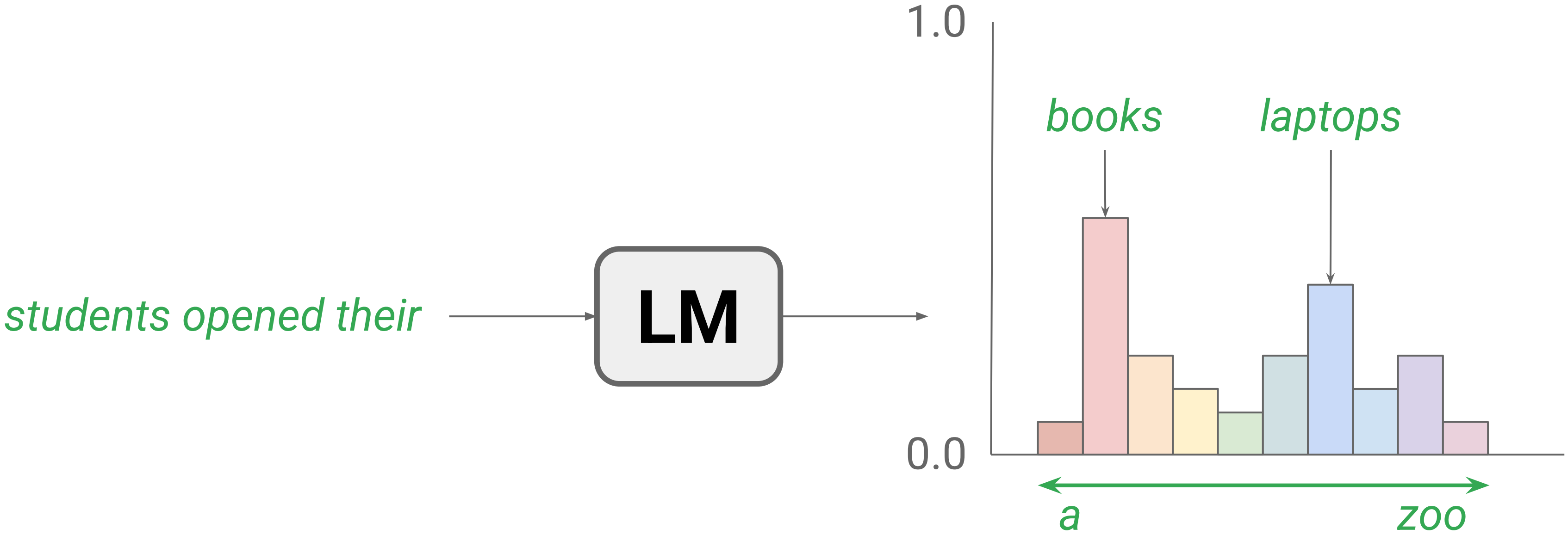
🚀 Starting from Qwen2.5-Math-7B (base model), we perform RL on it directly. No SFT, no reward model, just 8K MATH examples for verification, the resultant model achieves (pass@1) 33.3% on AIME, 62.5% on AMC, and 77.2% on MATH, outperforming Qwen2.5-math-7B-instruct and being comparable to PRIME and rStar-MATH that use >50x more data and more complicated components.
🚀 Increased CoT length and self-reflection emerge

We share the details and our findings in the blog: hkust-nlp.notion.site/simplerl-reason
Training code and implementation details here: github.com/hkust-nlp/simp...

https://x.com/junxian_he/status/1883183099787571519

A recap on language modeling

- Predict **the next word**, or a ***probability distribution*** over possible next words



A recap on language modeling (cont'd)

- Language models
  - compute

$$P(w_1, w_2, \ldots, w_n)$$

or

$$P(w_j | w_1, w_2, \ldots, w_{j-1})$$

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_n|w_1, w_2, \ldots, w_{n-1})$$

N-gram language models

- Use maximum likelihood estimation (MLE)

$$P(\text{“laptops”} \mid \text{“students opened their”}) = \frac{Count(\text{“students opened their laptops”})}{Count(\text{“students opened their”})}$$

Problems with n-gram language models

$$P(\text{"laptops"} \mid \text{"students opened their"}) = \frac{\text{Count}(\text{"students opened their laptops"})}{\text{Count}(\text{"students opened their"})}$$

What if "students opened their laptops" never occurred in training data?

Problems with n-gram language models (cont'd)

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Need to store V^n counts for an n-gram model!

source: Jurafsky and Martin

Problems with n-gram language models (cont'd)

- Treat semantically similar prefixes independently of each other

  "students opened their ___"

  "pupils opened their ___"

  "scholars opened their ___"

  "students began reading their ___"

Shouldn't we share information across these prefixes?

Word representations / embeddings

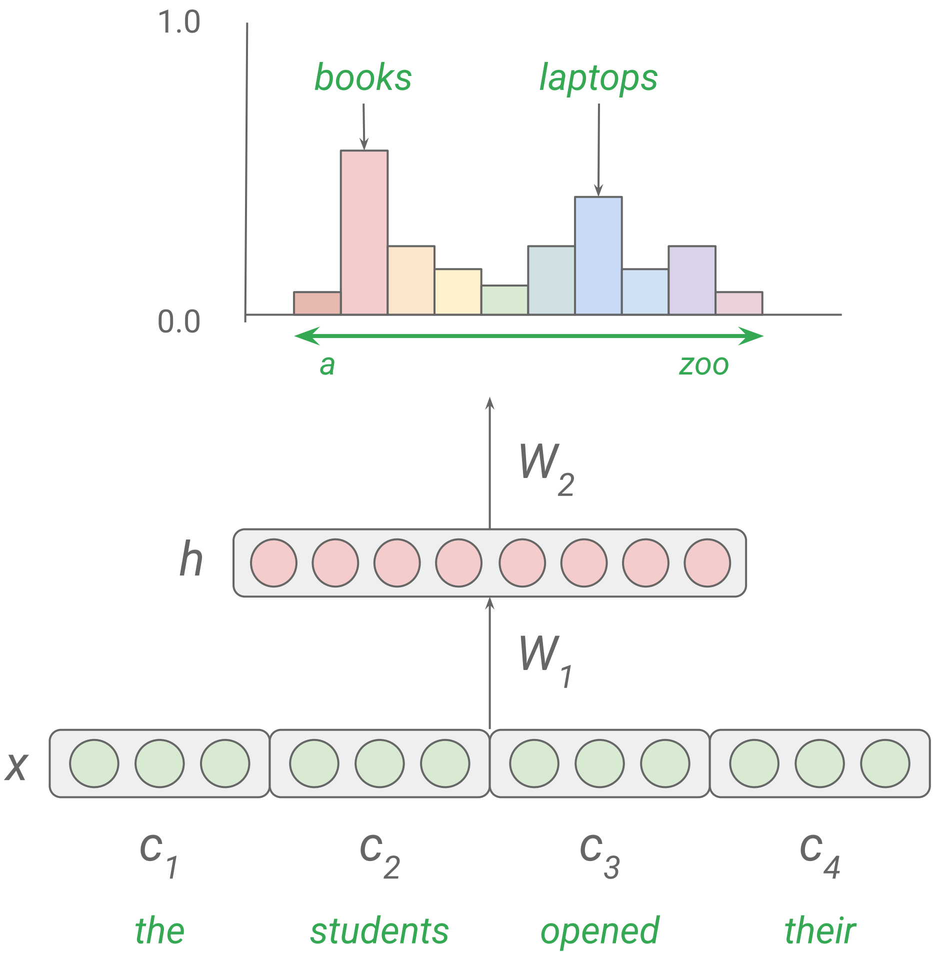
```
#What is the vector representation for a word?
w2v_model['computer']
```

```
array([ 1.07421875e-01, -2.01171875e-01,  1.23046875e-01,  2.11914062e-01,
       -9.13085938e-02,  2.16796875e-01, -1.31835938e-01,  8.30078125e-02,
        2.02148438e-01,  4.78515625e-02,  3.66210938e-02, -2.45361328e-02,
        2.39257812e-02, -1.60156250e-01, -2.61230469e-02,  9.71679688e-02,
       -6.34765625e-02,  1.84570312e-01,  1.70898438e-01, -1.63085938e-01,
       -1.09375000e-01,  1.49414062e-01, -4.65393066e-04,  9.61914062e-02,
        1.68945312e-01,  2.60925293e-03,  8.93554688e-02,  6.49414062e-02,
        3.56445312e-02, -6.93359375e-02, -1.46484375e-01, -1.21093750e-01,
       -2.27539062e-01,  2.45361328e-02, -1.24511719e-01, -3.18359375e-01,
       -2.20703125e-01,  1.30859375e-01,  3.66210938e-02, -3.63769531e-02,
       -1.13281250e-01,  1.95312500e-01,  9.76562500e-02,  1.26953125e-01,
        6.59179688e-02,  6.93359375e-02,  1.02539062e-02,  1.75781250e-01,
       -1.68945312e-01,  1.21307373e-03, -2.98828125e-01, -1.15234375e-01,
        5.66406250e-02, -1.77734375e-01, -2.08984375e-01,  1.76757812e-01,
        2.38037109e-02, -2.57812500e-01, -4.46777344e-02,  1.88476562e-01,
        5.51757812e-02,  5.02929688e-02, -1.06933594e-01,  1.89453125e-01,
       -1.16210938e-01,  8.49609375e-02, -1.71875000e-01,  2.45117188e-01,
       -1.73828125e-01, -8.30078125e-03,  4.56542969e-02, -1.61132812e-02,
        1.86523438e-01, -6.05468750e-02, -4.17480469e-02,  1.82617188e-01,
        2.20703125e-01, -1.22558594e-01, -2.55126953e-02, -3.08593750e-01,
        9.13085938e-02,  1.60156250e-01,  1.70898438e-01,  1.19628906e-01,
        7.08007812e-02, -2.64892578e-02, -3.08837891e-02,  4.06250000e-01,
       -1.01562500e-01,  5.71289062e-02, -7.26318359e-03, -9.17968750e-02,
       -1.50390625e-01, -2.55859375e-01,  2.16796875e-01, -3.63769531e-02,
        2.24609375e-01,  8.00781250e-02,  1.56250000e-01,  5.27343750e-02,
```

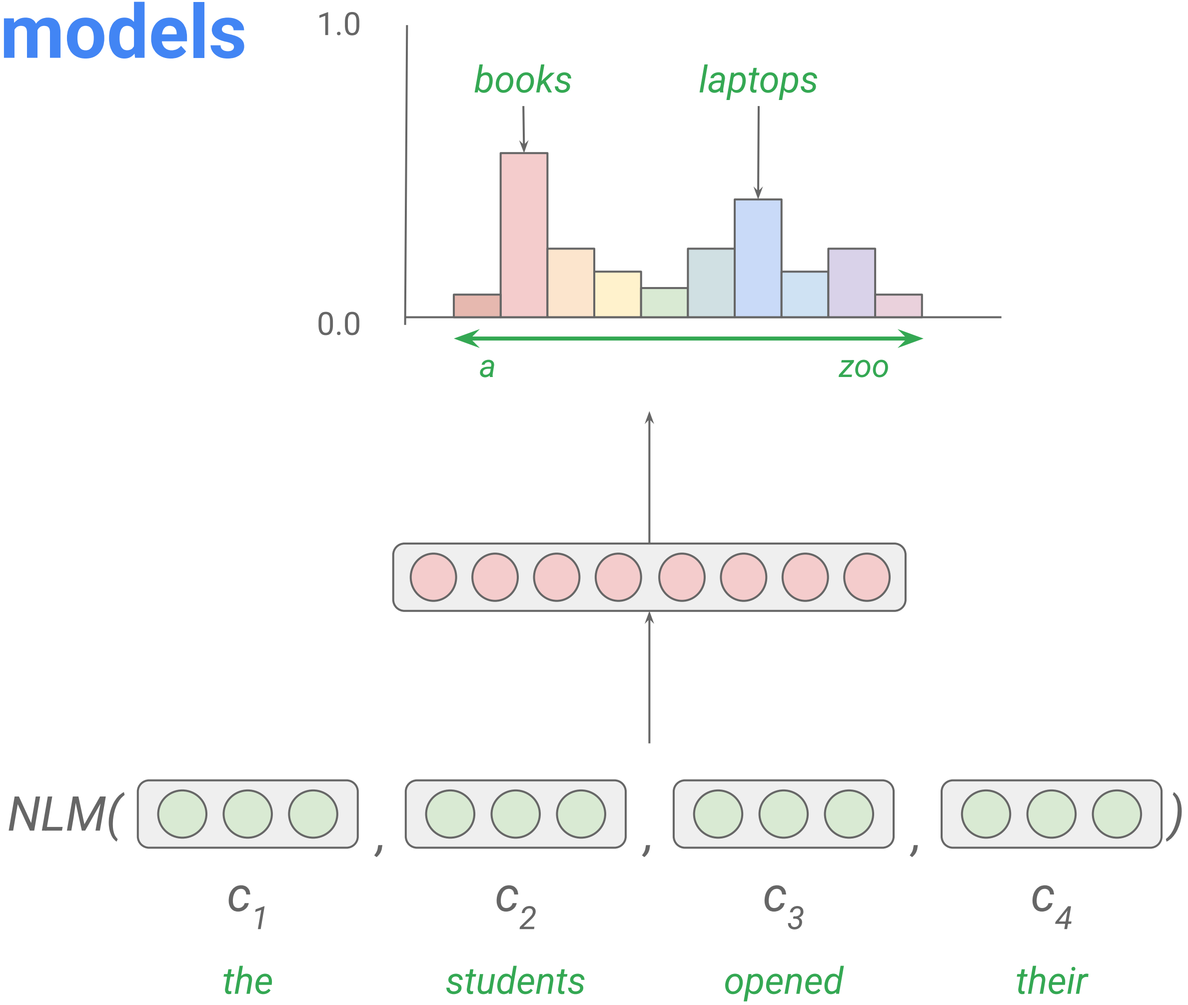
Connected to Python 3 Google Compute Engine backend

output distribution

$\hat{y} = softmax(W_2 h)$



Neural language models



Composition functions

- Element-wise functions
  - e.g., just sum up all of the word embeddings
- Concatenation
- Feedforward neural networks
- Convolutional neural networks
- Recurrent neural networks
- Transformers

Matrix-vector multiplication

Matrix A (dimensions 4×3):

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Vector x (dimensions 3×1):

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Resulting vector b (dimensions 4×1):

$$b = A \cdot x = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 \end{bmatrix}$$

Softmax function

For a vector $y = [y_1, y_2, \ldots, y_V]$ of dimension V, the softmax transformation is calculated as:

$$\mathrm{softmax}(y) = \left[\frac{e^{y_1}}{\sum e^y}, \frac{e^{y_2}}{\sum e^y}, \ldots, \frac{e^{y_V}}{\sum e^y}\right]$$

where $\sum e^y = e^{y_1} + e^{y_2} + \cdots + e^{y_V}$.

Feedforward neural language model

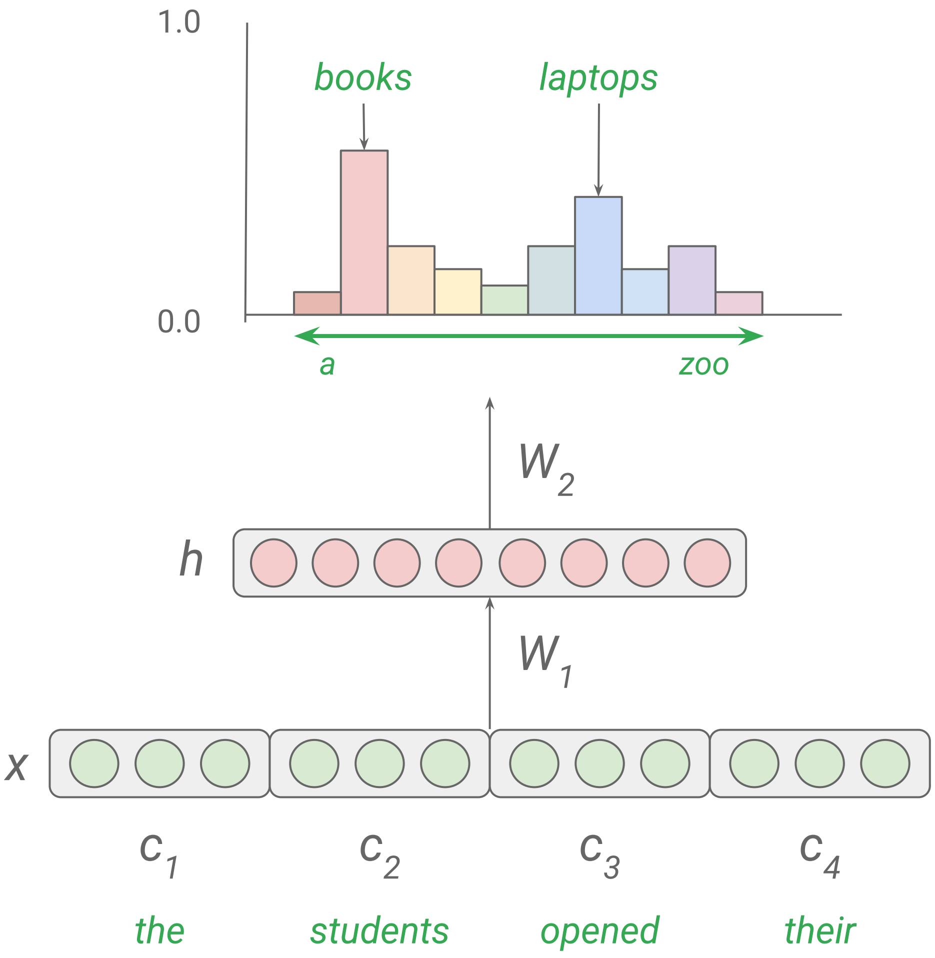
hidden layer

$h = f(W_1 x)$

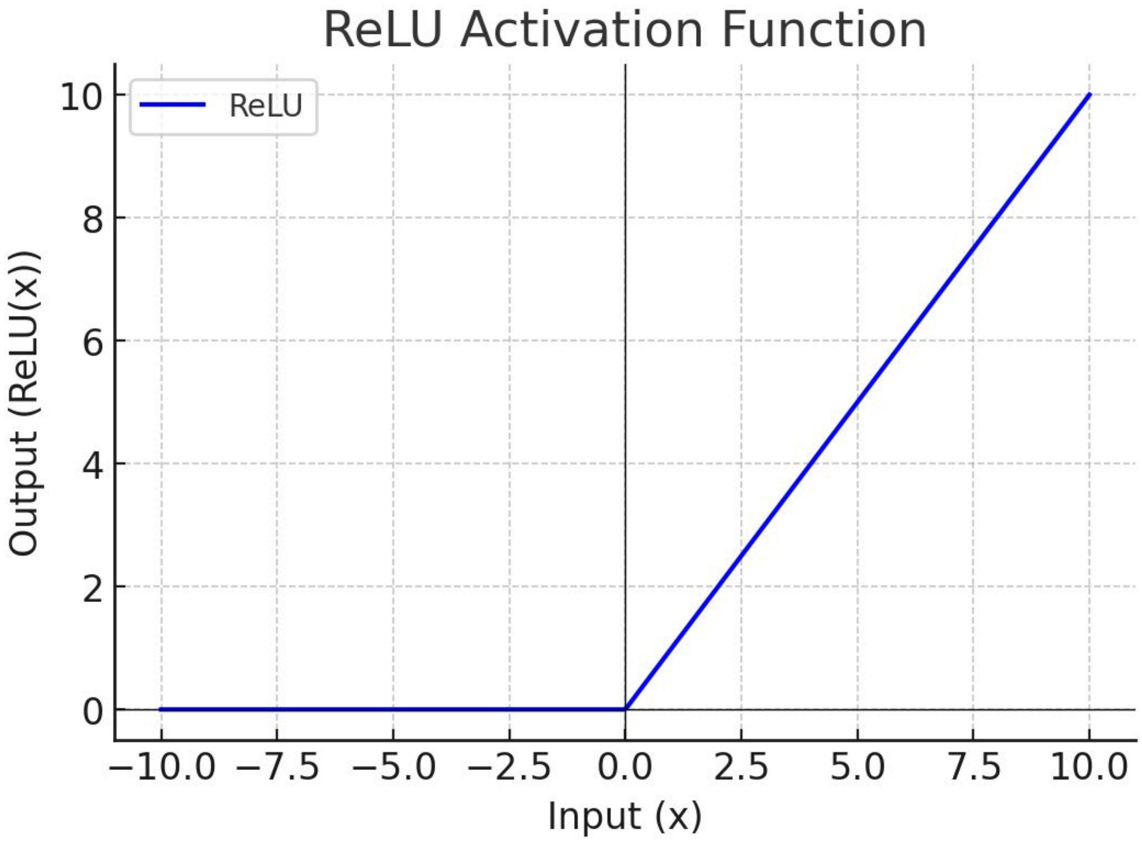
f is a non-linear activation function to model non-linear relationships between words

output distribution

$\hat{y} = softmax(W_2 h)$



Activation functions



Rectified Linear Unit (ReLU)

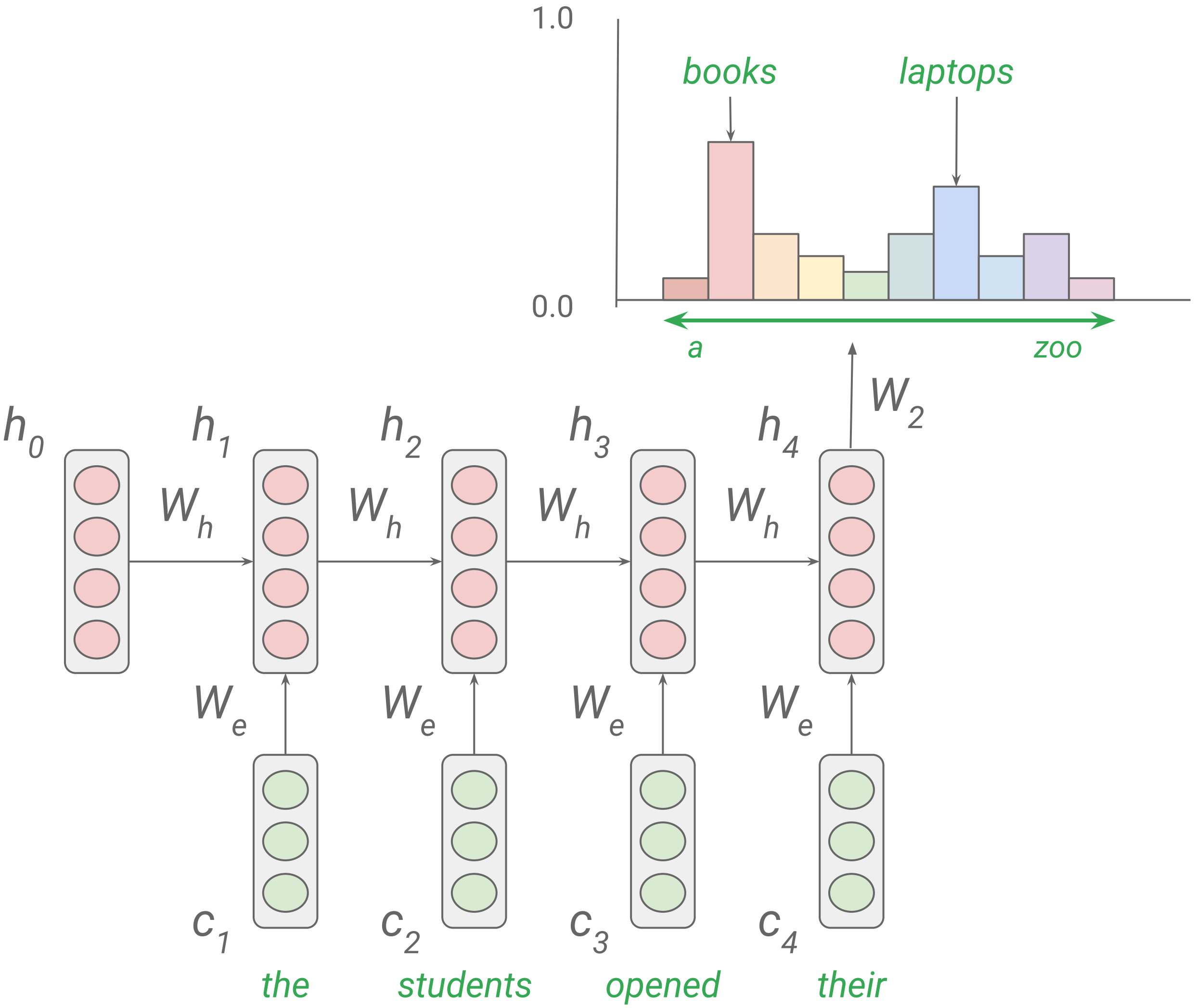
Recurrent neural networks (RNNs)

hidden states

$$h^{(t)} = f(W_h h^{(t-1)} + W_e c^t)$$

output distribution

$$\hat{y} = softmax(W_2 h^{(n-1)})$$



Recurrent neural networks (RNNs)

- RNNs advantages
  - can handle much longer histories
  - can generalize better over contexts of similar words
  - are more accurate at word-prediction

- RNNs disadvantages
  - are much more complex
  - are slower and need more energy to train
  - and are less interpretable than n-gram models

Thank you!