

# Next-Token Prediction

Tokens, embeddings, and one complete neural language-model calculation

---

Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

# L11 classified spatial sites; L12 classifies the next time site

## PUBLIC L11 · SEGMENTATION

one categorical distribution at  
every spatial location  
output support  $H \times W$



## PUBLIC L12 · LANGUAGE

one categorical distribution at the  
next ordered location  
output support  $V$

## PUBLIC L11 · SEGMENTATION

one categorical distribution at  
every spatial location  
output support  $H \times W$



## PUBLIC L12 · LANGUAGE

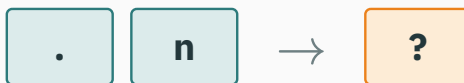
one categorical distribution at the  
next ordered location  
output support  $V$

**The classifier and cross-entropy stay. Order, causality, and feeding predictions back are new.**

# Commit before the model reveals anything

QUESTION

**HELD CASE · OPENING COMMITMENT** Vocabulary [., a, e, i, n, v]; visible context .n; predict the next token.



# Commit before the model reveals anything

QUESTION

**HELD CASE · OPENING COMMITMENT** Vocabulary [., a, e, i, n, v]; visible context .n; predict the next token.

. n → ?

## OUTPUT

What six numbers must the model return?

## CHOICE

Will greedy and sampling always pick the same token?

## LEARNING

Which embedding rows can this example update?

# Commit before the model reveals anything

**HELD CASE · OPENING COMMITMENT** Vocabulary [., a, e, i, n, v]; visible context .n; predict the next token.

. n → ?

## OUTPUT

What six numbers must the model return?

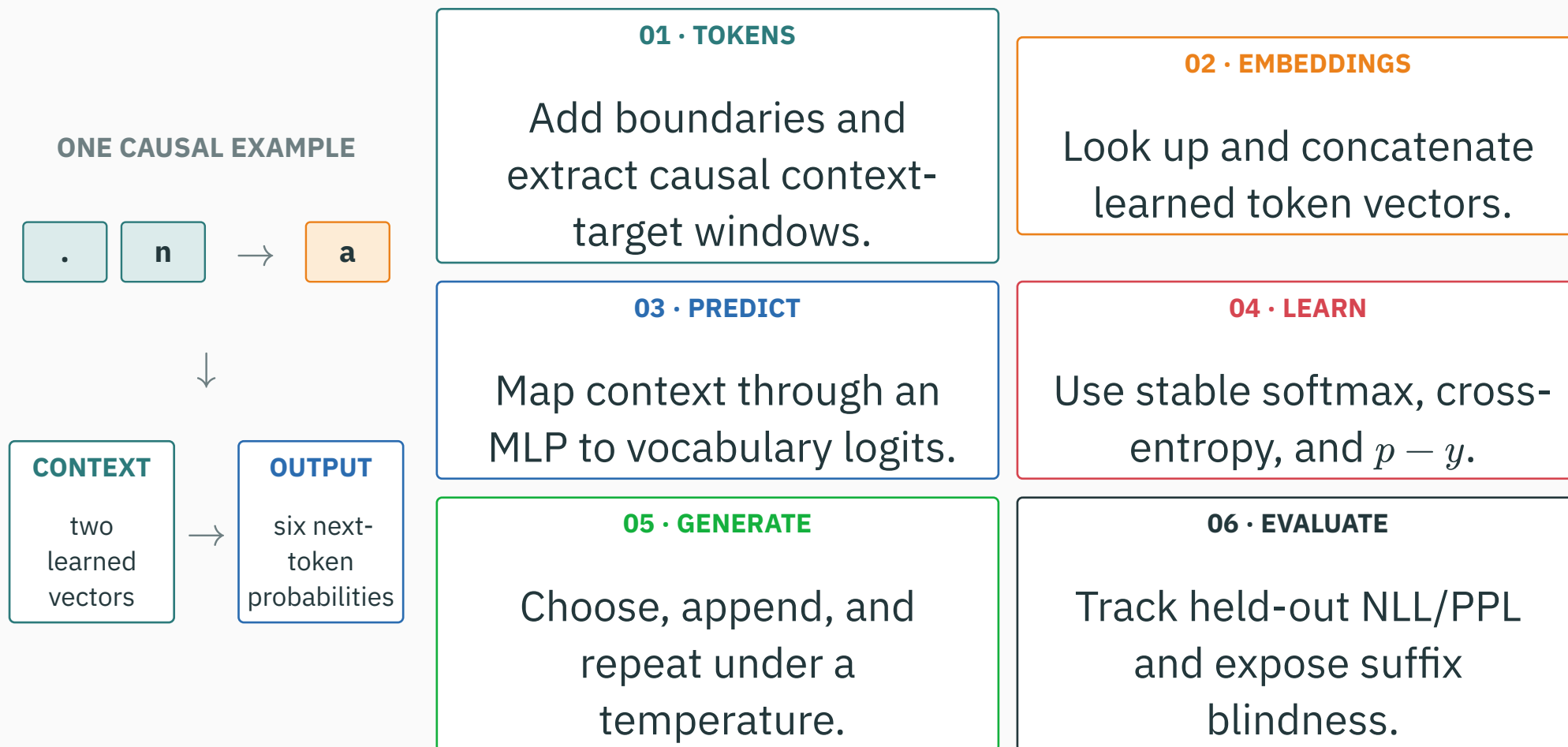
## CHOICE

Will greedy and sampling always pick the same token?

## LEARNING

Which embedding rows can this example update?

Commit silently. We will derive every answer from one set of weights and revisit all three.



# Three clocks reuse the distribution but change the question

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

# Three clocks reuse the distribution but change the question

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

# Three clocks reuse the distribution but change the question

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

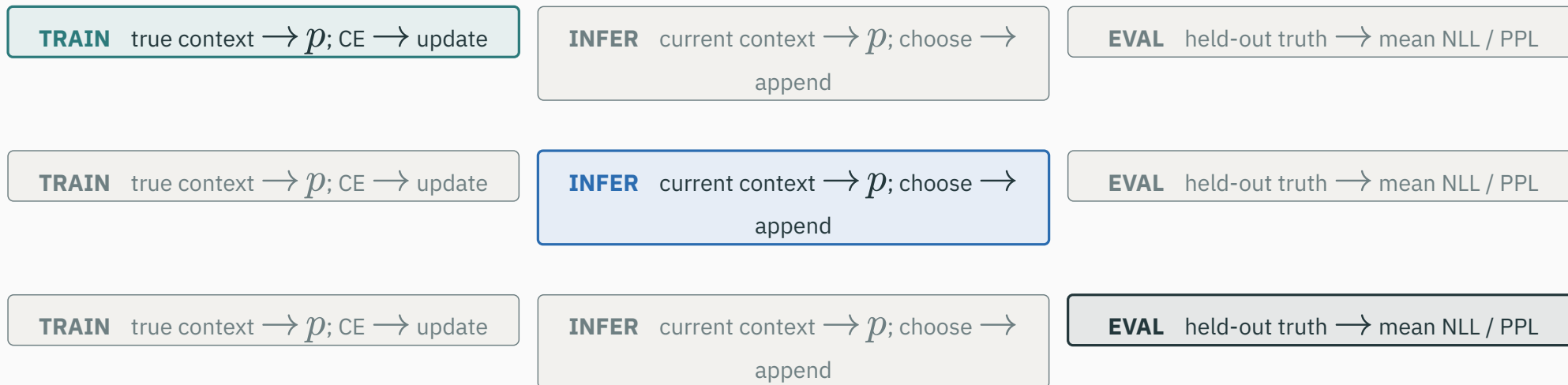
**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

# Three clocks reuse the distribution but change the question



**Always ask: are we updating weights, choosing a token, or scoring held-out truth?**

By the end, one sentence should be operational

V VISUAL

**A language model maps an observed prefix to a distribution over the next token.**

**A language model maps an observed prefix to a distribution over the next token.**

## REPRESENT

turn categorical ids into  
learned vectors without  
inventing numeric order

## PREDICT

produce  $V$  logits,  
normalize, and train with  
token cross-entropy

## DIAGNOSE

separate decoding  
choices from an  
architecture that cannot  
see enough history

— observed context / TRAIN    — model score or probability / INFER    — target or chosen token    — correct / retained    — loss or error    — EVAL

# One tiny corpus supplies every visible token

## TRAIN NAMES

naveen · navin · neev · neena · vani  
naina · avni · anvi · veena · navi

## HELD-OUT NAMES

navina · naveena · veenavi · anvina

Toy convention: . is one shared boundary symbol. Repeat it  $k = 2$  times as left padding; use one . target to end a name. It is not punctuation.

# Tokenization chooses the units the model must predict

| unit      | example                                | vocabulary | sequence |
|-----------|----------------------------------------|------------|----------|
| character | learning<br>→ l e a r n i n g          | small      | long     |
| word      | deep learning<br>→ deep   learning     | very large | short    |
| subword   | unpredictable<br>→ un   predict   able | moderate   | moderate |

Characters keep every row visible. Production usually separates <BOS>, <EOS>, <PAD>, and <UNK>; this toy merges <BOS> and <EOS> as ..

# The vocabulary fixes ids and output coordinates

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| . | a | e | i | n | v |
| 0 | 1 | 2 | 3 | 4 | 5 |

# The vocabulary fixes ids and output coordinates

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| . | a | e | i | n | v |
| 0 | 1 | 2 | 3 | 4 | 5 |

$V = 6$  means six logits, six probabilities, and six possible next tokens.

**HELD CASE · VOCABULARY** The target a is coordinate 1; context .n reads ids (0, 4).

# Sliding a two-character window creates seven examples

|    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|
| .. | .n | na | av | ve | ee | en |
| n  | a  | v  | e  | e  | n  | .  |

# Sliding a two-character window creates seven examples

|    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|
| .. | .n | na | av | ve | ee | en |
| n  | a  | v  | e  | e  | n  | .  |

top = visible context; bottom = next-token target

**HELD CASE · TRAINING PAIRS** .n → a is the second example and remains our hand calculation.

# Causal training shifts the same sequence by one site

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

|               |    |    |    |    |    |    |    |
|---------------|----|----|----|----|----|----|----|
| <b>input</b>  | .. | .n | na | av | ve | ee | en |
| <b>target</b> | n  | a  | v  | e  | e  | n  | .  |

# Causal training shifts the same sequence by one site

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

|               |    |    |    |    |    |    |    |
|---------------|----|----|----|----|----|----|----|
| <b>input</b>  | .. | .n | na | av | ve | ee | en |
| <b>target</b> | n  | a  | v  | e  | e  | n  | .  |

**At position  $t$ , the target token is not part of the input context.**

# The chain rule turns sequence probability into next-token factors

Let  $B = (., .)$  denote the fixed left boundary padding and let  $x_7 = .$  be the end target.

$$p(x_{1:7} \mid B) = \prod_{t=1}^7 p(x_t \mid B, x_{<t}).$$

# The chain rule turns sequence probability into next-token factors

D DERIVATION

Let  $B = (., .)$  denote the fixed left boundary padding and let  $x_7 = .$  be the end target.

$$p(x_{1:7} \mid B) = \prod_{t=1}^7 p(x_t \mid B, x_{<t}).$$

**This is an exact identity. The fixed-window approximation comes next.**

# One ledger will cross probability, loss, and evaluation

**HELD CASE · SEQUENCE LEDGER** Correct targets for naveen. under contexts . ., .n, na, av, ve, ee, en.

|                         |     |               |     |     |     |     |     |
|-------------------------|-----|---------------|-----|-----|-----|-----|-----|
| <b>context</b>          | . . | . n           | na  | av  | ve  | ee  | en  |
| <b>target</b>           | n   | a             | v   | e   | e   | n   | .   |
| <b><i>p(target)</i></b> | .30 | <b>derive</b> | .70 | .60 | .50 | .40 | .80 |

# One ledger will cross probability, loss, and evaluation

**HELD CASE · SEQUENCE LEDGER** Correct targets for naveen. under contexts . ., .n, na, av, ve, ee, en.

|                         |     |        |     |     |     |     |     |
|-------------------------|-----|--------|-----|-----|-----|-----|-----|
| <b>context</b>          | . . | .n     | na  | av  | ve  | ee  | en  |
| <b>target</b>           | n   | a      | v   | e   | e   | n   | .   |
| <b><i>p</i>(target)</b> | .30 | derive | .70 | .60 | .50 | .40 | .80 |

Six probabilities are held illustrative. The orange .n → a entry will come from the exact weights.

# Checkpoint: what did the chain rule assume?

QUESTION

**A**

tokens are independent

**B**

the sequence has fixed length

**C**

nothing extra; it is an identity

**D**

future tokens are visible

# Checkpoint: what did the chain rule assume?

QUESTION

**A**

tokens are independent

**B**

the sequence has fixed length

**C**

nothing extra; it is an identity

**D**

future tokens are visible

**C. Each exact factor conditions on the complete preceding history and boundary.**

The approximation begins only when the model discards all but the latest  $k$  tokens.

# A fixed-context MLP deliberately truncates the history

$$p_{\theta}(x_t \mid B, x_{<t}) \approx p_{\theta}(x_t \mid x_{t-2:t-1}).$$

# A fixed-context MLP deliberately truncates the history

$$p_{\theta}(x_t \mid B, x_{<t}) \approx p_{\theta}(x_t \mid x_{t-2:t-1}).$$

## EXACT CONDITIONAL

boundary plus every earlier token  
variable-length input



## TOY MODEL

only two latest tokens  
fixed-size input for an MLP

**This model learns local spelling patterns; it cannot recover a token that has scrolled out.**

# A token id is a label, not a magnitude

$n$  has id 4 and  $v$  has id 5, but

$5 - 4 = 1$  does not mean  $v - n = 1$ .

# A token id is a label, not a magnitude

$n$  has id 4 and  $v$  has id 5, but

$5 - 4 = 1$  **does not mean**  $v - n = 1$ .

Feeding raw ids as scalar features invents ordering and distance. Ids are categorical addresses.

The representation must preserve identity without pretending that adjacent ids are similar.

# One-hot vectors give each token an independent axis

With vocabulary order [., a, e, i, n, v]:

$$o_{.} = (1 \ 0 \ 0 \ 0 \ 0 \ 0)^T, \quad o_n = (0 \ 0 \ 0 \ 0 \ 1 \ 0)^T.$$

# One-hot vectors give each token an independent axis

With vocabulary order  $[\cdot, a, e, i, n, v]$ :

$$o_{\cdot} = (1 \ 0 \ 0 \ 0 \ 0 \ 0)^T, \quad o_n = (0 \ 0 \ 0 \ 0 \ 1 \ 0)^T.$$

For different tokens  $i \neq j$ ,  $\|o_i - o_j\|^2 = 2$ .

One-hot is faithful but sparse,  $V$ -dimensional, and has no learned similarity structure.

# An embedding is a learned row selected by an id

| token | feature<br>1 | feature<br>2 |
|-------|--------------|--------------|
| .     | 1            | 0            |
| a     | 0            | 1            |
| e     | 1            | 1            |
| i     | -1           | 1            |
| n     | 0            | 2            |
| v     | 2            | 0            |

# An embedding is a learned row selected by an id

| token | feature<br>1 | feature<br>2 |
|-------|--------------|--------------|
| .     | 1            | 0            |
| a     | 0            | 1            |
| e     | 1            | 1            |
| i     | -1           | 1            |
| n     | 0            | 2            |
| v     | 2            | 0            |

$$E \in \mathbb{R}^{6 \times 2}, \quad e_i = E[i] = E^T o_i.$$

**The id addresses a row; gradient descent decides what the two coordinates become.**

# Context .n reads exactly two rows

**HELD CASE · LOOKUP** ids  $(0, 4)$  select rows for . and n.

$$e_{.} = E[0] = (1, 0), \quad e_n = E[4] = (0, 2).$$

# Context .n reads exactly two rows

**HELD CASE · LOOKUP** ids (0, 4) select rows for . and n.

$$e_{.} = E[0] = (1, 0), \quad e_n = E[4] = (0, 2).$$



The target  $a$  affects the output loss, but its embedding row is not read on this example.

# Commit: concatenate or average the two rows?

QUESTION

**A • AVERAGE**

$$c = \frac{e_{\cdot} + e_n}{2} = (.5, 1)$$

**B • CONCATENATE**

$$c = [e_{\cdot}; e_n] = (1, 0, 0, 2)$$

# Commit: concatenate or average the two rows?

QUESTION

**A · AVERAGE**

$$c = \frac{e_{\cdot} + e_n}{2} = (.5, 1)$$

**B · CONCATENATE**

$$c = [e_{\cdot}; e_n] = (1, 0, 0, 2)$$

**B. Concatenation gives the first and second context positions different coordinate blocks.**

# Commit: concatenate or average the two rows?

## A · AVERAGE

$$c = \frac{e_{\cdot} + e_n}{2} = (.5, 1)$$

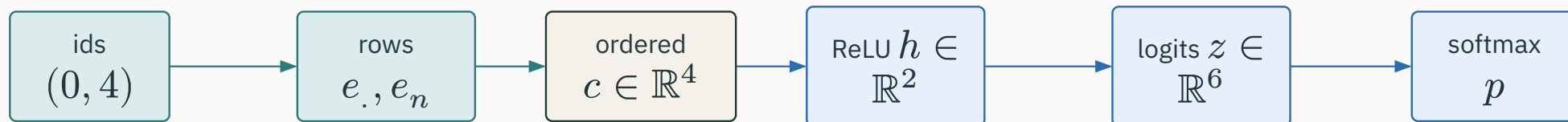
## B · CONCATENATE

$$c = [e_{\cdot}; e_n] = (1, 0, 0, 2)$$

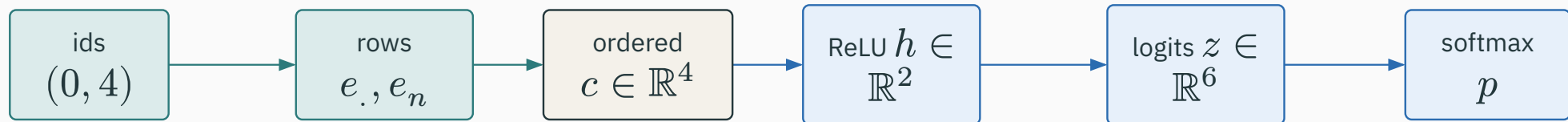
**B. Concatenation gives the first and second context positions different coordinate blocks.**

$$c_{\cdot n} = (1, 0, 0, 2) \neq (0, 2, 1, 0) = c_{n \cdot}$$

# One position is an ordinary six-class classifier



# One position is an ordinary six-class classifier



two ids  $\rightarrow$  two 2-D rows  $\rightarrow$  ordered 4-D context  $\rightarrow$  two hidden units  $\rightarrow$  six logits

**Language modeling changes how examples are constructed; the local head is familiar classification.**

# The tensor ledger follows the pipeline

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$c = [E[x_{t-2}]; E[x_{t-1}]] \in \mathbb{R}^4,$$

# The tensor ledger follows the pipeline

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$c = [E[x_{t-2}]; E[x_{t-1}]] \in \mathbb{R}^4,$$

$$u = W_h c + b_h, \quad h = \text{ReLU}(u) \in \mathbb{R}^2,$$

# The tensor ledger follows the pipeline

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$c = [E[x_{t-2}]; E[x_{t-1}]] \in \mathbb{R}^4,$$

$$u = W_h c + b_h, \quad h = \text{ReLU}(u) \in \mathbb{R}^2,$$

$$z = W_o h + b_o \in \mathbb{R}^6, \quad p = \text{softmax}(z).$$

$W_h \in \mathbb{R}^{2 \times 4}$ ;  $W_o \in \mathbb{R}^{6 \times 2}$ ; rows of  $W_o$  follow vocabulary order.

# Count parameters once, then keep the shapes visible

| block             | shape            | count             |
|-------------------|------------------|-------------------|
| embedding $E$     | $V \times d$     | $Vd = 12$         |
| hidden $W_h, b_h$ | $H \times kd, H$ | $Hkd + H =$<br>10 |
| output $W_o, b_o$ | $V \times H, V$  | $VH + V =$<br>18  |

# Count parameters once, then keep the shapes visible

| block             | shape            | count          |
|-------------------|------------------|----------------|
| embedding $E$     | $V \times d$     | $Vd = 12$      |
| hidden $W_h, b_h$ | $H \times kd, H$ | $Hkd + H = 10$ |
| output $W_o, b_o$ | $V \times H, V$  | $VH + V = 18$  |

$12 + 10 + 18 = 40$  parameters.

Increasing context length  $k$  grows the hidden block  $Hkd$  even before we discuss what the model can remember.

# Every weight for $.n \rightarrow a$ is now explicit

**HELD CASE · EXACT MODEL** input  $x = (., n)$ ; target  $y = a$ ;  $c = (1, 0, 0, 2)^T$ .

$$W_h = \begin{pmatrix} 0.5 & 0 & 0 & 0.25 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad b_h = \begin{pmatrix} 0 \\ -1 \end{pmatrix},$$

# Every weight for $.n \rightarrow a$ is now explicit

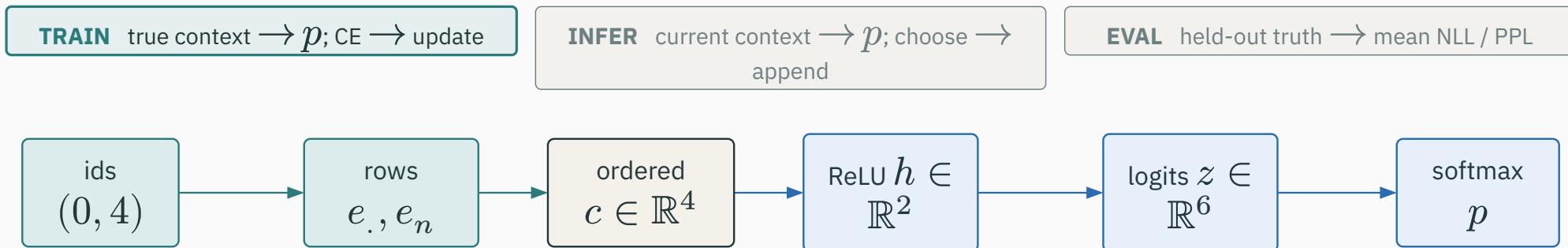
**HELD CASE · EXACT MODEL** input  $x = (., n)$ ; target  $y = a$ ;  $c = (1, 0, 0, 2)^T$ .

$$W_h = \begin{pmatrix} 0.5 & 0 & 0 & 0.25 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad b_h = \begin{pmatrix} 0 \\ -1 \end{pmatrix},$$

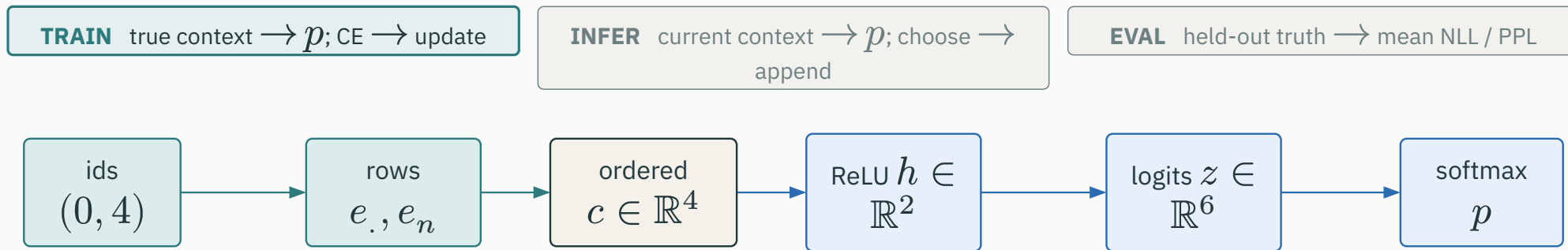
$$W_o = \begin{pmatrix} -1 & 0 \\ 2 & 1 \\ 0 & -1 \\ -1 & 1 \\ 0 & 0 \\ 1 & -1 \end{pmatrix}, \quad b_o = 0.$$

Rows of  $W_o$  are  $[., a, e, i, n, v]$ . Keep that order for every vector below.

# Forward 1: lookup and concatenate



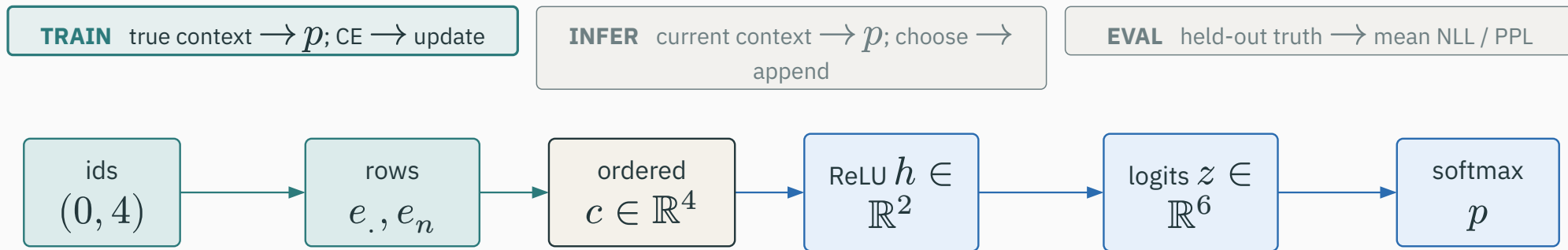
# Forward 1: lookup and concatenate



$$E[0] = (1, 0), \quad E[4] = (0, 2),$$

# Forward 1: lookup and concatenate

D DERIVATION



$$E[0] = (1, 0), \quad E[4] = (0, 2),$$

$$c = [E[0]; E[4]] = (1, 0, 0, 2)^T.$$

**HELD CASE · FORWARD · CONTEXT** The ordered context occupies four coordinates; no averaging occurs.

## Forward 2: one hidden unit turns on

$$u_1 = .5(1) + 0(0) + 0(0) + .25(2) + 0 = 1,$$

## Forward 2: one hidden unit turns on

$$u_1 = .5(1) + 0(0) + 0(0) + .25(2) + 0 = 1,$$

$$u_2 = 0(1) + 1(0) + 1(0) + 0(2) - 1 = -1.$$

## Forward 2: one hidden unit turns on

$$u_1 = .5(1) + 0(0) + 0(0) + .25(2) + 0 = 1,$$

$$u_2 = 0(1) + 1(0) + 1(0) + 0(2) - 1 = -1.$$

$$h = (\max(0, 1), \max(0, -1)) = (1, 0).$$

**ACTIVE**

$h_1 = 1$ ; gradient may pass

**OFF**

$h_2 = 0$ ; ReLU blocks its gradient here

# Commit: compute two logits before seeing all six

With  $h = (1, 0)$ , calculate the rows for target  $a$  and alternative  $v$ :

**TARGET  $a$**

$$z_a = (2, 1) \cdot (1, 0) = ?$$

**ALTERNATIVE  $v$**

$$z_v = (1, -1) \cdot (1, 0) = ?$$

# Commit: compute two logits before seeing all six

With  $h = (1, 0)$ , calculate the rows for target  $a$  and alternative  $v$ :

**TARGET  $a$**

$$z_a = (2, 1) \cdot (1, 0) = ?$$

**ALTERNATIVE  $v$**

$$z_v = (1, -1) \cdot (1, 0) = ?$$

$$z_a = 2 \quad z_v = 1$$

# Commit: compute two logits before seeing all six

With  $h = (1, 0)$ , calculate the rows for target  $a$  and alternative  $v$ :

**TARGET  $a$**

$$z_a = (2, 1) \cdot (1, 0) = ?$$

**ALTERNATIVE  $v$**

$$z_v = (1, -1) \cdot (1, 0) = ?$$

$$z_a = 2 \quad z_v = 1$$

**A logit is an unconstrained score, not yet a probability.**

# Forward 3: every output row supplies one logit

D DERIVATION

**HELD CASE · FORWARD · LOGITS** vocabulary order [., a, e, i, n, v].

$$z = W_o h = (-1, 2, 0, -1, 0, 1)^T.$$

# Forward 3: every output row supplies one logit

**HELD CASE · FORWARD · LOGITS** vocabulary order [., a, e, i, n, v].

$$z = W_o h = (-1, 2, 0, -1, 0, 1)^T.$$

| .  | a | e | i  | n | v |
|----|---|---|----|---|---|
| -1 | 2 | 0 | -1 | 0 | 1 |

Softmax will preserve this ranking while converting scores into a normalized distribution.

# Stable softmax subtracts a constant before exponentiating

**D** DERIVATION

Naively exponentiating large logits can overflow. Softmax is unchanged by a shared shift:

# Stable softmax subtracts a constant before exponentiating

Naively exponentiating large logits can overflow. Softmax is unchanged by a shared shift:

$$\text{softmax}(z) = \text{softmax}(z - m), \quad m = \max_j z_j = 2.$$

# Stable softmax subtracts a constant before exponentiating

Naively exponentiating large logits can overflow. Softmax is unchanged by a shared shift:

$$\text{softmax}(z) = \text{softmax}(z - m), \quad m = \max_j z_j = 2.$$

$$z - m = (-3, 0, -2, -3, -2, -1).$$

**Subtracting the maximum is an implementation safeguard, not a model change.**

## Forward 4: exponentiate and normalize

$$\exp(z - m) = (.0498, 1, .1353, .0498, .1353, .3679),$$

## Forward 4: exponentiate and normalize

$$\exp(z - m) = (.0498, 1, .1353, .0498, .1353, .3679),$$

$$Z = \sum_j \exp(z_j - m) = 1.7381,$$

## Forward 4: exponentiate and normalize

$$\exp(z - m) = (.0498, 1, .1353, .0498, .1353, .3679),$$

$$Z = \sum_j \exp(z_j - m) = 1.7381,$$

$$p = \frac{\exp(z - m)}{Z} = (.028644, .575333, .077863, .028644, .077863, .211653).$$

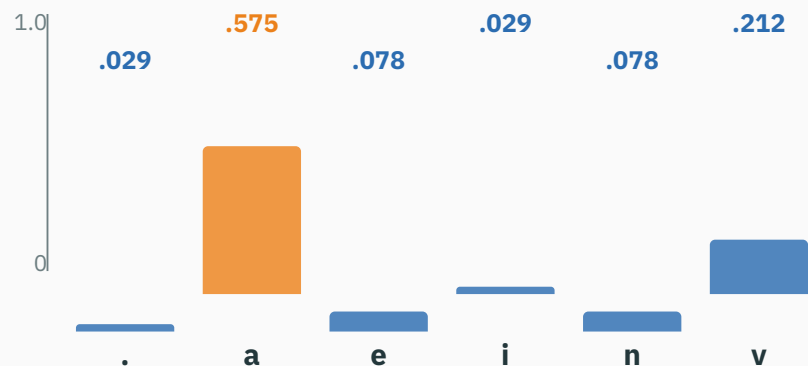
**HELD CASE · FORWARD · PROBABILITY** The exact orange ledger entry is now  $p(a|.n) = .575333$ .

# The six bars share a true 0-to-1 probability axis

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

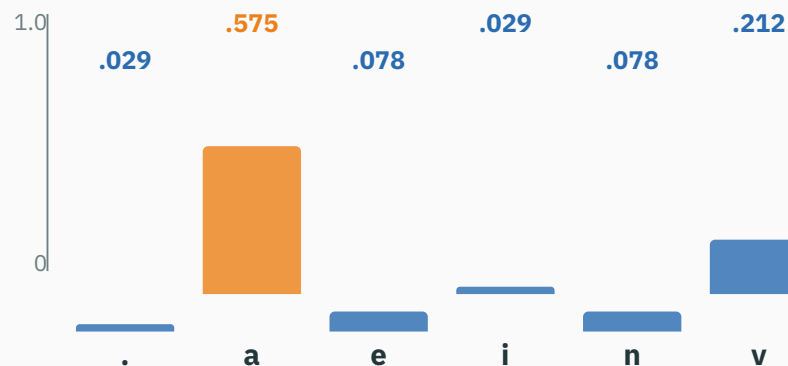


# The six bars share a true 0-to-1 probability axis

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL



Direct labels follow [., a, e, i, n, v]; all six probabilities sum to 1.

**a is most likely, but alternatives retain nonzero probability.**

# Forward 5: cross-entropy reads the target coordinate

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

The target is a, so  $y = (0, 1, 0, 0, 0, 0)$ .

# Forward 5: cross-entropy reads the target coordinate

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

The target is a, so  $y = (0, 1, 0, 0, 0, 0)$ .

$$\mathcal{L} = - \sum_j y_j \log p_j = -\log(.575333) = .552806 \text{ nats.}$$

**IF  $p_a$  RISES**

the loss falls

**IF  $p_a \rightarrow 1$**

$\mathcal{L} \rightarrow 0$

# The sequence ledger now contains no second version

|                         |     |         |     |     |     |     |     |
|-------------------------|-----|---------|-----|-----|-----|-----|-----|
| <b>context</b>          | ..  | .n      | na  | av  | ve  | ee  | en  |
| <b>target</b>           | n   | a       | v   | e   | e   | n   | .   |
| <b><i>p(target)</i></b> | .30 | .575333 | .70 | .60 | .50 | .40 | .80 |

# The sequence ledger now contains no second version

|                    |     |         |     |     |     |     |     |
|--------------------|-----|---------|-----|-----|-----|-----|-----|
| context            | ..  | .n      | na  | av  | ve  | ee  | en  |
| target             | n   | a       | v   | e   | e   | n   | .   |
| $p(\text{target})$ | .30 | .575333 | .70 | .60 | .50 | .40 | .80 |

$$p_{\theta(\text{naveen}|\text{..})} \approx .30(.575333)(.70)(.60)(.50)(.40)(.80) = .0115987.$$

**The sequence probability is small because all seven next-token decisions must agree.**

# Logs turn the same product into NLL and perplexity

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$-\log(.0115987) = 4.45686 \text{ nats,}$$

# Logs turn the same product into NLL and perplexity

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$-\log(.0115987) = 4.45686 \text{ nats},$$

$$\text{mean NLL} = \frac{4.45686}{7} = .636694 \text{ nats/token},$$

# Logs turn the same product into NLL and perplexity

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$-\log(.0115987) = 4.45686 \text{ nats},$$

$$\text{mean NLL} = \frac{4.45686}{7} = .636694 \text{ nats/token},$$

$$\text{PPL} = \exp(.636694) = 1.89022.$$

| This is one illustrative name ledger, not the trained notebook's validation score.

# Commit: what sign must the target-logit gradient have?

QUESTION

**A • POSITIVE**

gradient descent will lower  $z_a$

**B • NEGATIVE**

gradient descent will raise  $z_a$

# Commit: what sign must the target-logit gradient have?

QUESTION

**A • POSITIVE**

gradient descent will lower  $z_a$

**B • NEGATIVE**

gradient descent will raise  $z_a$

**B. To reduce  $-\log p_a$ , the update must increase the target logit relative to alternatives.**

# Commit: what sign must the target-logit gradient have?

QUESTION

**A • POSITIVE**

gradient descent will lower  $z_a$

**B • NEGATIVE**

gradient descent will raise  $z_a$

**B. To reduce  $-\log p_a$ , the update must increase the target logit relative to alternatives.**

Softmax plus cross-entropy makes the exact sign visible in one vector.

# Backward 1: softmax plus CE gives $p - y$

**D** DERIVATION

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\nabla_z \mathcal{L} = p - y$$

# Backward 1: softmax plus CE gives $p - y$

**D** DERIVATION

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\nabla_z \mathcal{L} = p - y$$

$$= (.028644, -.424667, .077863, .028644, .077863, .211653).$$

- the target coordinate is negative, so gradient descent raises  $z_a$ ;
- every alternative coordinate is positive, so gradient descent lowers it.

# Backward 2: output gradients are outer products

Because  $z = W_o h + b_o$ ,

$$\nabla_{W_o} \mathcal{L} = (p - y)h^T, \quad \nabla_{b_o} \mathcal{L} = p - y.$$

# Backward 2: output gradients are outer products

Because  $z = W_o h + b_o$ ,

$$\nabla_{W_o} \mathcal{L} = (p - y)h^T, \quad \nabla_{b_o} \mathcal{L} = p - y.$$

With  $h = (1, 0)$ , the first column equals  $p - y$  and the second column is zero.

**An inactive feature emits no output-weight gradient on this example.**

$$\nabla_h \mathcal{L} = W_o^T(p-y) = (-.694969, -.685539)^T.$$

# Backward 3: ReLU chooses which hidden signal survives

D DERIVATION

$$\nabla_h \mathcal{L} = W_o^T(p-y) = (-.694969, -.685539)^T.$$

The forward pre-activation was  $u = (1, -1)$ , so the ReLU mask is  $(1, 0)$ .

# Backward 3: ReLU chooses which hidden signal survives

$$\nabla_h \mathcal{L} = W_o^T(p-y) = (-.694969, -.685539)^T.$$

The forward pre-activation was  $u = (1, -1)$ , so the ReLU mask is  $(1, 0)$ .

$$\nabla_u \mathcal{L} = \nabla_h \mathcal{L} \odot (1, 0) = (-.694969, 0)^T.$$

The second unit had a nonzero upstream signal, but its negative pre-activation blocks that signal.

# Backward 4: return to the ordered context

Since  $u = W_h c + b_h$ ,

Since  $u = W_h c + b_h$ ,

$$\nabla_c \mathcal{L} = W_h^T \nabla_u \mathcal{L} = (-.347485, 0, 0, -.173742)^T.$$

# Backward 4: return to the ordered context

Since  $u = W_h c + b_h$ ,

$$\nabla_c \mathcal{L} = W_h^T \nabla_u \mathcal{L} = (-.347485, 0, 0, -.173742)^T.$$

| block      | $e.[1]$    | $e.[2]$ | $e_{n[1]}$ | $e_{n[2]}$ |
|------------|------------|---------|------------|------------|
| $\nabla_c$ | $-.347485$ | $0$     | $0$        | $-.173742$ |

**Concatenation makes the split back into two embedding rows unambiguous.**

$$\nabla_{e_{\cdot}} \mathcal{L} = (-.347485, 0), \quad \nabla_{e_n} \mathcal{L} = (0, -.173742).$$

$$\nabla_{e_1} \mathcal{L} = (-.347485, 0), \quad \nabla_{e_n} \mathcal{L} = (0, -.173742).$$

Scatter these into rows 0 and 4 of  $E$ ; every other row receives zero from this lookup.

# Backward 5: split, scatter, and update only rows read

$$\nabla_{e_{\cdot}} \mathcal{L} = (-.347485, 0), \quad \nabla_{e_n} \mathcal{L} = (0, -.173742).$$

Scatter these into rows 0 and 4 of  $E$ ; every other row receives zero from this lookup. With  $\eta = .1$ :

$$e_{\cdot} \leftarrow (1.03475, 0), \quad e_n \leftarrow (0, 2.01737).$$

**HELD CASE · OPENING · LEARNING** The rows for  $\cdot$  and  $n$  move. Target row  $a$  was not read and does not move through the lookup.

# One scalar loss teaches every used block



# One scalar loss teaches every used block



- output rows learn which hidden features support each token;
- hidden weights learn combinations of ordered context coordinates;
- read embedding rows learn features that improve future predictions.

**The path is ordinary backpropagation; language structure entered when we built context-target pairs.**

# TRAIN repeats the calculation over a declared batch

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\mathcal{L}_B = -\frac{1}{|B|} \sum_{i \in B} \log p_{\theta}(y_i | c_i).$$

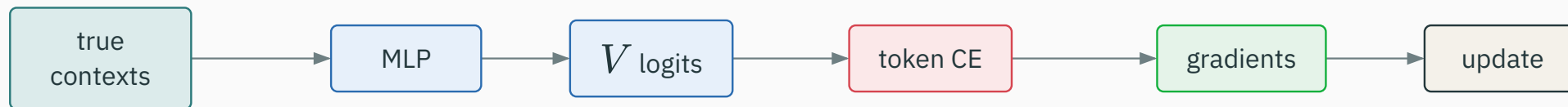
# TRAIN repeats the calculation over a declared batch

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\mathcal{L}_B = -\frac{1}{|B|} \sum_{i \in B} \log p_{\theta}(y_i | c_i).$$



The notebook uses the entire training set as  $B$  on each Adam step: full-batch training, not a minibatch claim.

# Split whole names before creating windows

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

## TRAIN

10 whole names  $\rightarrow$  56 context-target pairs  
weights receive updates

## VALIDATION

4 different whole names  $\rightarrow$  30 pairs  
no updates

# Split whole names before creating windows

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

## TRAIN

10 whole names  $\rightarrow$  56 context-target pairs  
weights receive updates

## VALIDATION

4 different whole names  $\rightarrow$  30 pairs  
no updates

Do not randomly split windows from the same name. Nearby windows share almost all their evidence.

**The split unit must match the object whose generalization we claim:  
here, a whole name.**

# EVAL declares tokenizer, targets, and reduction

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\text{NLL} = -\frac{1}{N} \sum_{i=1}^N \log p_{\theta}(y_i | c_i), \quad \text{PPL} = \exp(\text{NLL}).$$

# EVAL declares tokenizer, targets, and reduction

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$\text{NLL} = -\frac{1}{N} \sum_{i=1}^N \log p_{\theta}(y_i | c_i), \quad \text{PPL} = \exp(\text{NLL}).$$

## TOKENIZER

same character vocabulary and  
boundary policy

## TARGETS

$N$  non-padding next-token targets

## DATA

same held-out corpus and token-  
weighted reduction

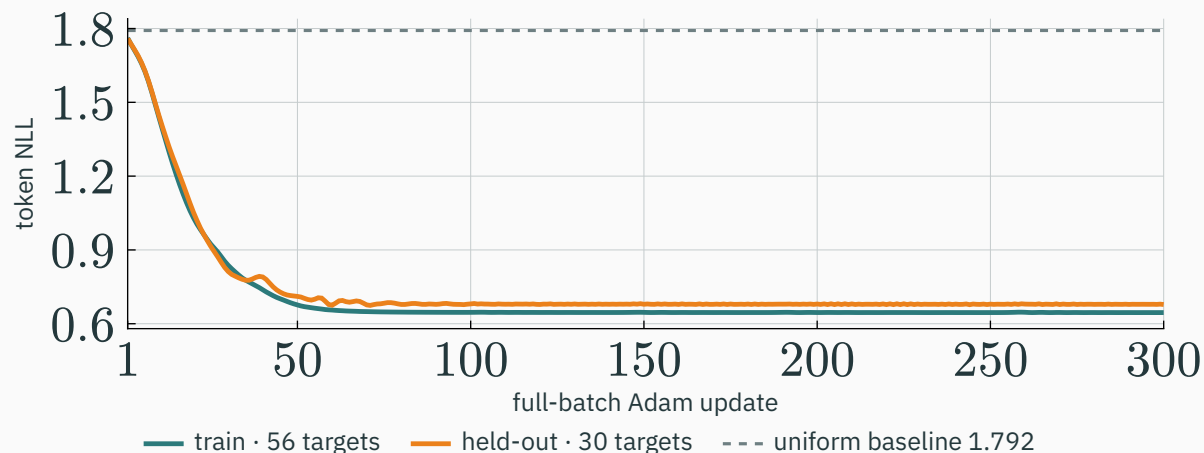
**Perplexities are comparable only when all three contracts agree.**

# The executed notebook shows the full learning trajectory

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL



## EXECUTED · SEED 11

300 full-batch Adam updates  
constructed 10+4-name course corpus

## COMPUTED · UPDATE 300

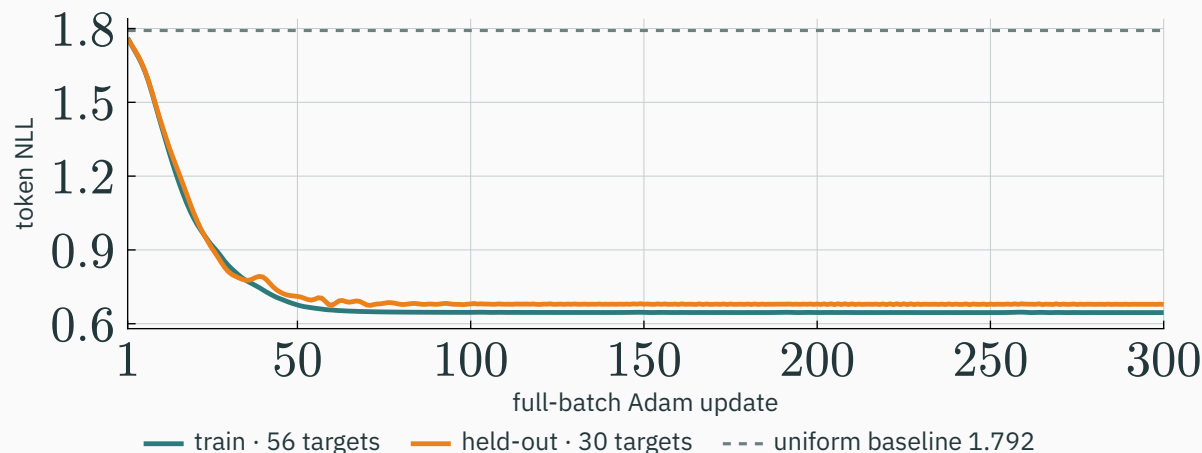
**train** · NLL 0.645308 · PPL 1.90657  
**held-out** · NLL 0.677965 · PPL 1.96986

# The executed notebook shows the full learning trajectory

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL



## EXECUTED · SEED 11

300 full-batch Adam updates  
constructed 10+4-name course corpus

## COMPUTED · UPDATE 300

**train** · NLL 0.645308 · PPL 1.90657  
**held-out** · NLL 0.677965 · PPL 1.96986

Every point evaluates the same post-update parameter snapshot; no smoothing. The train set has 56 targets and the four held-out names have 30 targets.

**Optimization + in-protocol transfer on a constructed course corpus; not broad name quality.**

# Uniform prediction is the first evaluation sanity check

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

For  $V = 6$ , a uniform model has  $p(y) = \frac{1}{6}$ :

# Uniform prediction is the first evaluation sanity check

**TRAIN** true context  $\rightarrow \underline{p}$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow \underline{p}$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

For  $V = 6$ , a uniform model has  $p(y) = \frac{1}{6}$ :

$$\text{NLL} = -\log\left(\frac{1}{6}\right) = \log 6 = 1.79176, \quad \text{PPL} = 6.$$

# Uniform prediction is the first evaluation sanity check

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

For  $V = 6$ , a uniform model has  $p(y) = \frac{1}{6}$ :

$$\text{NLL} = -\log\left(\frac{1}{6}\right) = \log 6 = 1.79176, \quad \text{PPL} = 6.$$

## UNIFORM

1.79176 NLL  
6.00 PPL

## VALIDATION

.677965 NLL  
1.96986 PPL

# Uniform prediction is the first evaluation sanity check

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

For  $V = 6$ , a uniform model has  $p(y) = \frac{1}{6}$ :

$$\text{NLL} = -\log\left(\frac{1}{6}\right) = \log 6 = 1.79176, \quad \text{PPL} = 6.$$

## UNIFORM

1.79176 NLL  
6.00 PPL

## VALIDATION

.677965 NLL  
1.96986 PPL

**The trained model predicts held-out next characters far better than uniform guessing under this protocol.**

# The same $p$ now crosses from TRAIN to INFER

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

# The same $p$ now crosses from TRAIN to INFER

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

# The same $p$ now crosses from TRAIN to INFER

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

## TRAIN

true context  $\rightarrow$  distribution  $\rightarrow$  compare with true target  
 $\rightarrow$  update

## INFER

current context  $\rightarrow$  distribution  $\rightarrow$  choose token  $\rightarrow$   
append

# The same $p$ now crosses from TRAIN to INFER

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

## TRAIN

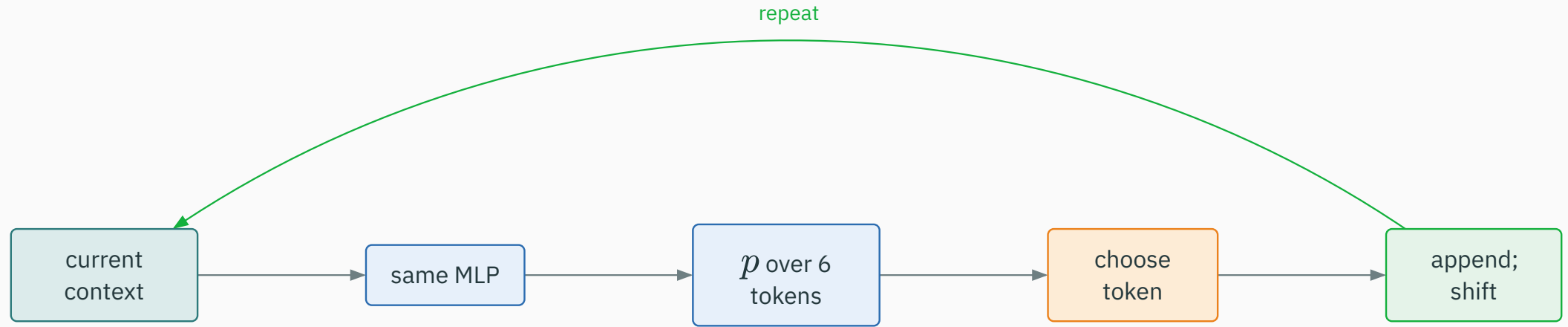
true context  $\rightarrow$  distribution  $\rightarrow$  compare with true target  
 $\rightarrow$  update

## INFER

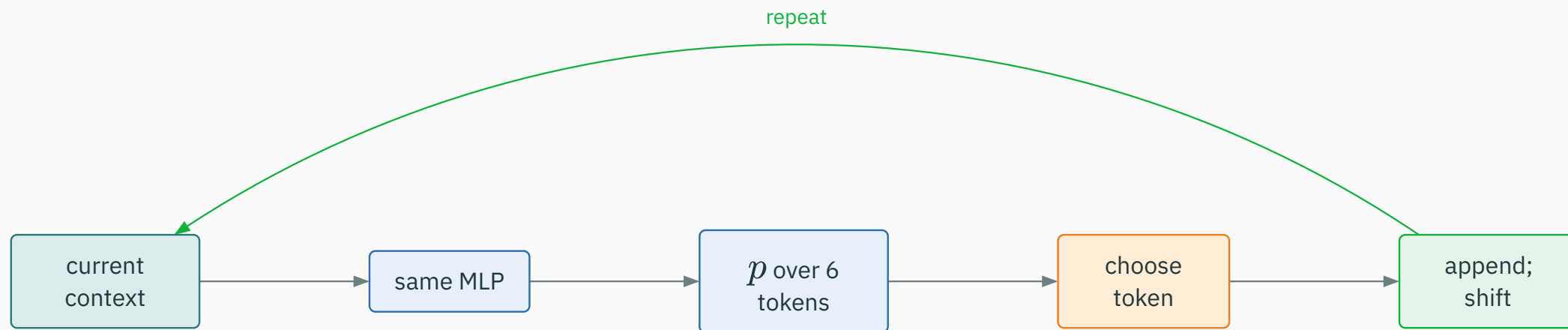
current context  $\rightarrow$  distribution  $\rightarrow$  choose token  $\rightarrow$   
append

**Weights stay fixed during generation; the chosen token changes the next input context.**

# Autoregressive generation closes a feedback loop

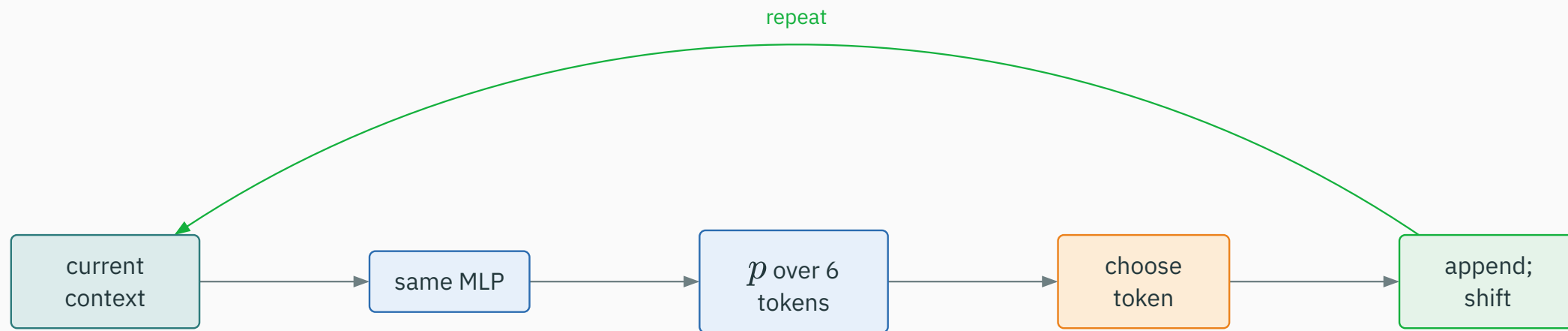


# Autoregressive generation closes a feedback loop



Start from . . ; stop when the model emits . or reaches a declared maximum length.

# Autoregressive generation closes a feedback loop

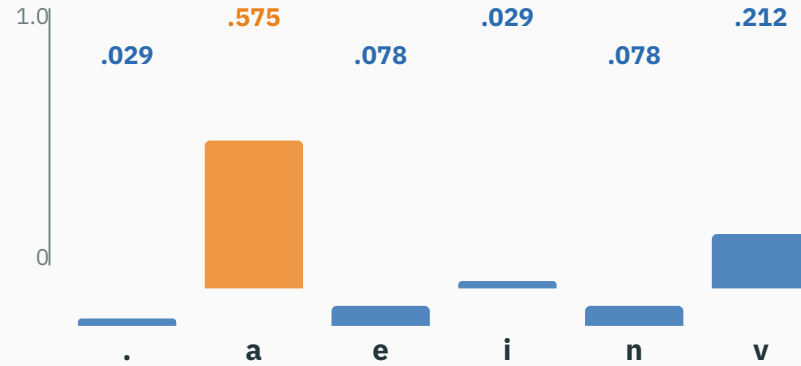


Start from . . ; stop when the model emits . or reaches a declared maximum length.

**Autoregressive means the model conditions on tokens already present, including its own earlier choices.**

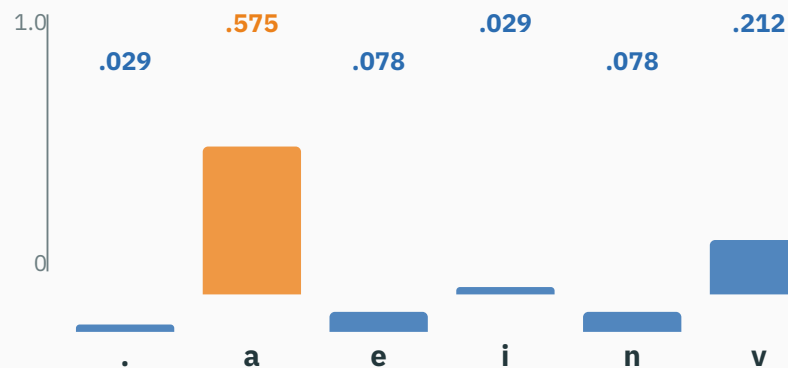
# Commit: must greedy and sampling choose the same token?

QUESTION



# Commit: must greedy and sampling choose the same token?

QUESTION



**GREEDY**

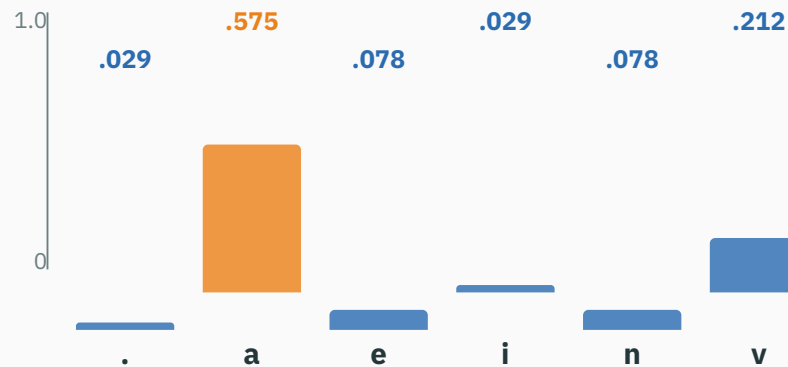
$$\hat{x} = \arg \max_v p(v|.n)$$

**SAMPLING**

$$\hat{x} \sim \text{Categorical}(p)$$

# Commit: must greedy and sampling choose the same token?

QUESTION



**GREEDY**

$$\hat{x} = \arg \max_v p(v|.n)$$

**SAMPLING**

$$\hat{x} \sim \text{Categorical}(p)$$

**No. Greedy always chooses a; sampling can choose any token with nonzero mass.**

# One uniform draw makes categorical sampling concrete

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

| token    | .    | a    | e    | i    | n    | v     |
|----------|------|------|------|------|------|-------|
| <i>p</i> | .029 | .575 | .078 | .029 | .078 | .212  |
| CDF      | .029 | .604 | .682 | .710 | .788 | 1.000 |

# One uniform draw makes categorical sampling concrete

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

| token | .    | a    | e    | i    | n    | v     |
|-------|------|------|------|------|------|-------|
| $p$   | .029 | .575 | .078 | .029 | .078 | .212  |
| CDF   | .029 | .604 | .682 | .710 | .788 | 1.000 |

Draw  $u = .650$  from  $\text{Uniform}(0, 1)$ .

# One uniform draw makes categorical sampling concrete

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

| token    | .    | a    | e    | i    | n    | v     |
|----------|------|------|------|------|------|-------|
| <i>p</i> | .029 | .575 | .078 | .029 | .078 | .212  |
| CDF      | .029 | .604 | .682 | .710 | .788 | 1.000 |

Draw  $u = .650$  from  $\text{Uniform}(0, 1)$ .

$.604 < .650 \leq .682 \Rightarrow$  **sample e.**

# One uniform draw makes categorical sampling concrete

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

| token | .    | a    | e    | i    | n    | v     |
|-------|------|------|------|------|------|-------|
| $p$   | .029 | .575 | .078 | .029 | .078 | .212  |
| CDF   | .029 | .604 | .682 | .710 | .788 | 1.000 |

Draw  $u = .650$  from  $\text{Uniform}(0, 1)$ .

$.604 < .650 \leq .682 \Rightarrow$  **sample e.**

**The model supplies probabilities; the decoder supplies the decision rule and randomness.**

# Temperature rescales logits before the same softmax

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$p_{i(\tau)} = \text{softmax}\left(\frac{z}{\tau}\right)_i, \quad \tau > 0.$$

# Temperature rescales logits before the same softmax

**D** DERIVATION

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$p_{i(\tau)} = \text{softmax}\left(\frac{z}{\tau}\right)_i, \quad \tau > 0.$$

$$\tau = .5$$

logit gaps double  
distribution concentrates

$$\tau = 1$$

learned distribution  
no rescaling

$$\tau = 2$$

logit gaps halve  
distribution flattens

# Temperature rescales logits before the same softmax

D DERIVATION

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

$$p_{i(\tau)} = \text{softmax}\left(\frac{z}{\tau}\right)_i, \quad \tau > 0.$$

$$\tau = .5$$

logit gaps double  
distribution concentrates

$$\tau = 1$$

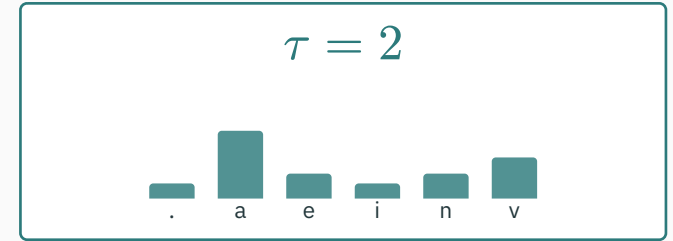
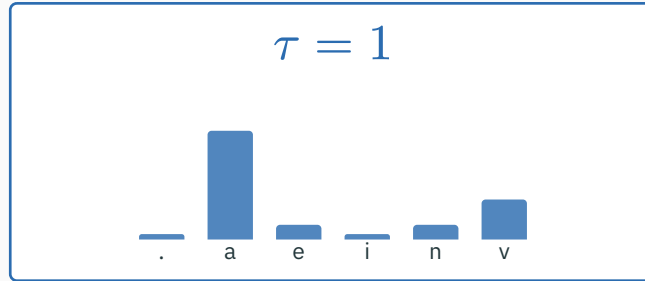
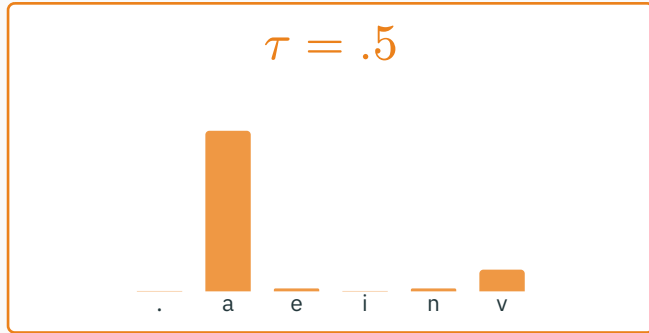
learned distribution  
no rescaling

$$\tau = 2$$

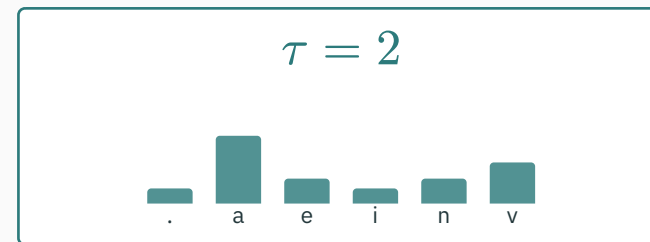
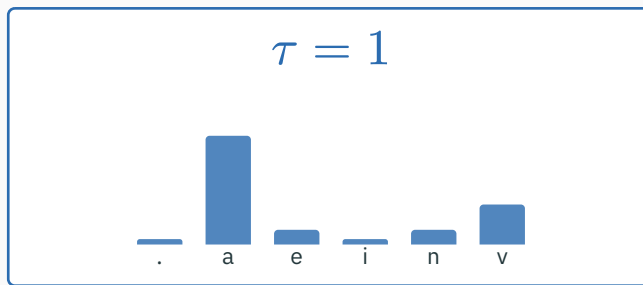
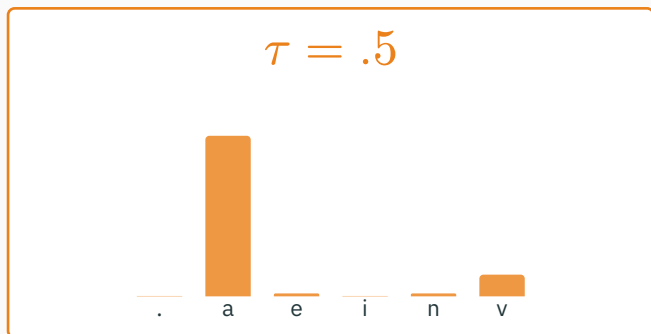
logit gaps halve  
distribution flattens

Temperature changes neither weights nor token ranking for positive  $\tau$ .

# The same six logits produce three exact distributions

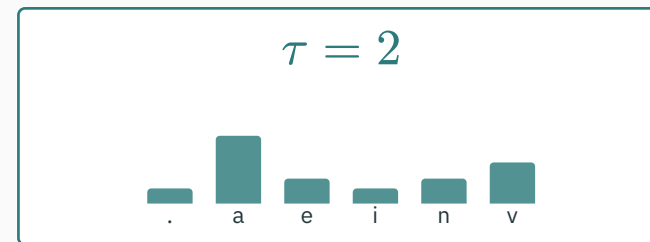
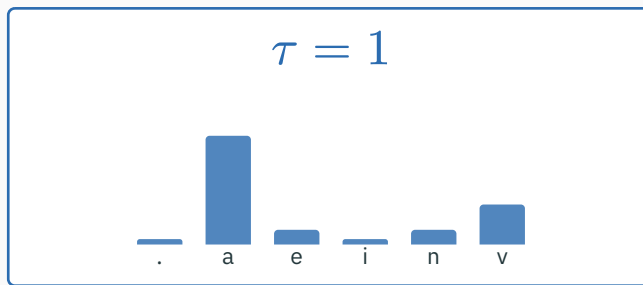
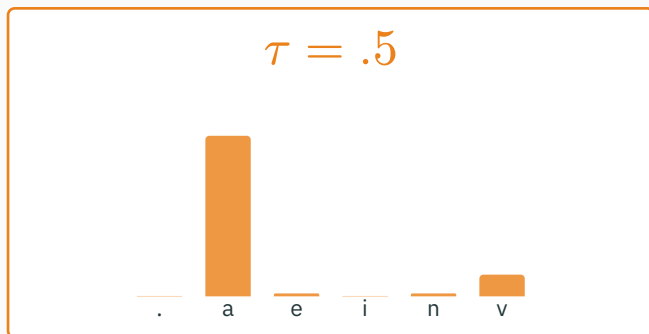


# The same six logits produce three exact distributions



| $\tau$ | .    | a    | e    | i    | n    | v    |
|--------|------|------|------|------|------|------|
| .5     | .002 | .850 | .016 | .002 | .016 | .115 |
| 1      | .029 | .575 | .078 | .029 | .078 | .212 |
| 2      | .080 | .359 | .132 | .080 | .132 | .218 |

# The same six logits produce three exact distributions



| $\tau$ | .    | a    | e    | i    | n    | v    |
|--------|------|------|------|------|------|------|
| .5     | .002 | .850 | .016 | .002 | .016 | .115 |
| 1      | .029 | .575 | .078 | .029 | .078 | .212 |
| 2      | .080 | .359 | .132 | .080 | .132 | .218 |

**Concentration is measurable; “safer” or “more correct” is not guaranteed.**

# Choose temperature by behavior, not a moral label

QUESTION

## SYMPTOM

samples collapse to the same spelling

## TEST

hold logits and seed fixed; compare token frequencies across

$\mathcal{T}$

# Choose temperature by behavior, not a moral label

QUESTION

## SYMPTOM

samples collapse to the same spelling

## TEST

hold logits and seed fixed; compare token frequencies across

$\tau$

## LOWER $\tau$

more mass on current top tokens; less variation

## HIGHER $\tau$

more mass on alternatives; more variation

# Choose temperature by behavior, not a moral label

QUESTION

## SYMPTOM

samples collapse to the same spelling

## TEST

hold logits and seed fixed; compare token frequencies across

$\tau$

## LOWER $\tau$

more mass on current top tokens; less variation

## HIGHER $\tau$

more mass on alternatives; more variation

If the model ranks a bad token highest, lowering temperature can make that error more consistent.

# The trained MLP is blind to prefixes with the same suffix

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

## PREFIX A

na  $\rightarrow$  visible context na

$\equiv$

## PREFIX B

veena  $\rightarrow$  visible context na

# The trained MLP is blind to prefixes with the same suffix

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**PREFIX A**

na  $\rightarrow$  visible context na

$\equiv$

**PREFIX B**

veena  $\rightarrow$  visible context na

$$p = (.418671, 9.91 \times 10^{-7}, 3.77 \times 10^{-6}, .142728, 3.17 \times 10^{-5}, .438565)$$

# The trained MLP is blind to prefixes with the same suffix

**TRAIN** true context  $\rightarrow p$ ; CE  $\rightarrow$  update

**INFER** current context  $\rightarrow p$ ; choose  $\rightarrow$   
append

**EVAL** held-out truth  $\rightarrow$  mean NLL / PPL

**PREFIX A**

na  $\rightarrow$  visible context na

$\equiv$

**PREFIX B**

veena  $\rightarrow$  visible context na

$$p = (.418671, 9.91 \times 10^{-7}, 3.77 \times 10^{-6}, .142728, 3.17 \times 10^{-5}, .438565)$$

**The two distributions are exactly identical because the model receives exactly the same two ids.**

# Diagnose the architecture before tuning the optimizer

## SYMPTOM

different long prefixes produce identical  
next-token distributions

## SUSPECT

the useful distinction lies outside the  
fixed window

## TEST

hold the suffix fixed, change only earlier  
tokens, compare  $p$  exactly

## SYMPTOM

different long prefixes produce identical  
next-token distributions

## SUSPECT

the useful distinction lies outside the  
fixed window

## TEST

hold the suffix fixed, change only earlier  
tokens, compare  $p$  exactly

**If  $p$  cannot change, more training cannot make this architecture use the hidden prefix.**

## SYMPTOM

different long prefixes produce identical  
next-token distributions

## SUSPECT

the useful distinction lies outside the  
fixed window

## TEST

hold the suffix fixed, change only earlier  
tokens, compare  $p$  exactly

**If  $p$  cannot change, more training cannot make this architecture use the hidden prefix.**

This is an information-access failure, not evidence that Adam, embeddings, or temperature are broken.

# A larger window delays the failure but does not remove it

D DERIVATION

$W_h \in \mathbb{R}^{H \times kd} \Rightarrow Hkd \text{ weights.}$

# A larger window delays the failure but does not remove it

$W_h \in \mathbb{R}^{H \times kd} \Rightarrow Hkd \text{ weights.}$

- doubling  $k$  doubles this block;
- the same token in slot 1 and slot 2 meets different columns;
- the maximum usable history remains fixed at model construction.

# A larger window delays the failure but does not remove it

$W_h \in \mathbb{R}^{H \times kd} \Rightarrow Hkd \text{ weights.}$

- doubling  $k$  doubles this block;
- the same token in slot 1 and slot 2 meets different columns;
- the maximum usable history remains fixed at model construction.

Concatenation preserves order, but the MLP does not reuse one position-independent update across time.

# Revisit: every opening commitment now has evidence

QUESTION

**HELD CASE · OPENING REVISIT** context .n, exact six-token model.

## OUTPUT

$$p = (.029, .575, .078, .029, .078, .212)$$

## CHOICE

greedy  $\rightarrow$  a; sample with  $u = .65$   
 $\rightarrow$  e

## LEARNING

lookup updates rows . and n

# Revisit: every opening commitment now has evidence

QUESTION

HELD CASE · OPENING REVISIT context .n, exact six-token model.

## OUTPUT

$$p = (.029, .575, .078, .029, .078, .212)$$

## CHOICE

greedy  $\rightarrow$  a; sample with  $u = .65$   
 $\rightarrow$  e

## LEARNING

lookup updates rows . and n

**One distribution supports training, decoding, and evaluation only after the clock and policy are declared.**

# Expose the full output-gradient matrix

With  $h = (1, 0)$ ,

$$\nabla_{w_o} \mathcal{L} = \begin{pmatrix} .028644 & 0 \\ -.424667 & 0 \\ .077863 & 0 \\ .028644 & 0 \\ .077863 & 0 \\ .211653 & 0 \end{pmatrix}.$$

# Expose the full output-gradient matrix

With  $h = (1, 0)$ ,

$$\nabla_{w_o} \mathcal{L} = \begin{pmatrix} .028644 & 0 \\ -.424667 & 0 \\ .077863 & 0 \\ .028644 & 0 \\ .077863 & 0 \\ .211653 & 0 \end{pmatrix}.$$

Each vocabulary row receives its own logit error; the inactive hidden feature leaves the second column zero.

# Use the notebook as a prediction protocol

**I** INTERACTIVE

## INTERACTIVE

↗ [colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01\\_char\\_mlp\\_language\\_model.ipynb](https://colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01_char_mlp_language_model.ipynb)

Before running: predict the seven windows, active ReLU unit, sign of the target gradient, two embedding rows that move, and whether `na` and `veena` can differ.

# Use the notebook as a prediction protocol

**I** INTERACTIVE

## INTERACTIVE

↗ [colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01\\_char\\_mlp\\_language\\_model.ipynb](https://colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01_char_mlp_language_model.ipynb)

Before running: predict the seven windows, active ReLU unit, sign of the target gradient, two embedding rows that move, and whether `na` and `veena` can differ.

### VERIFY

exact forward/backward arrays; train/validation NLL;  
generated names

### EXPLAIN

why temperature changes diversity and why a shared suffix  
forces identical  $p$

# Use the notebook as a prediction protocol

**I** INTERACTIVE

## INTERACTIVE

↗ [colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01\\_char\\_mlp\\_language\\_model.ipynb](https://colab.research.google.com/github/nipunbatra/dl-teaching/blob/master/notebooks/L11/01_char_mlp_language_model.ipynb)

Before running: predict the seven windows, active ReLU unit, sign of the target gradient, two embedding rows that move, and whether `na` and `veena` can differ.

### VERIFY

exact forward/backward arrays; train/validation NLL;  
generated names

### EXPLAIN

why temperature changes diversity and why a shared suffix  
forces identical  $p$

**Prediction before execution turns the notebook from a demo into evidence.**

## PRIMARY SOURCE

Bengio et al. (2003)  
distributed representations plus a feedforward neural  
probabilistic language model

## FOLLOW-ALONG

Exact ES 667 Colab  
deterministic NumPy forward/backward, full-batch training,  
generation, and suffix test

## PRIMARY SOURCE

Bengio et al. (2003)  
distributed representations plus a feedforward neural  
probabilistic language model

## FOLLOW-ALONG

Exact ES 667 Colab  
deterministic NumPy forward/backward, full-batch training,  
generation, and suffix test

The paper scales the central idea; our six-token model keeps its mechanics auditable by hand.

# Next: replace a fixed window with a shared recurrent state

## THIS LECTURE

$c_t = [e_{t-2}; e_{t-1}]$   
fixed access; position-specific columns



## PUBLIC L13

$h_t = f_{\theta}(h_{t-1}, e_t)$   
shared update; variable-length stream

## THIS LECTURE

$$c_t = [e_{t-2}; e_{t-1}]$$

fixed access; position-specific columns



## PUBLIC L13

$$h_t = f_{\theta}(h_{t-1}, e_t)$$

shared update; variable-length stream

**The next-token objective stays; the mechanism for carrying context changes.**

A language model estimates a distribution over the **next token**.

Next: **RNNs, LSTMs, and GRUs** - learn what recurrent state remembers, forgets, and struggles to train.