Lecture 6 Activity: A Language Model by Hand — Key

E190BO

Part 1

(a) BOS the fluffy blue creature → token IDs 0, 5, 3, 1, 2.

(b) Each target is the next input token, shifted by one:

position	0	1	2	3	4
input token	BOS	the	fluffy	blue	creature
target token	the	fluffy	blue	creature	is

(c) e = word embedding + position embedding:

	word embedding	position embedding	e
BOS (pos. 0)	$[0,0,0,0,0]$	$[0,0,0,0,0]$	$[0,0,0,0,0]$
the (pos. 1)	$[0,0,0,0,0]$	$[0,0,0,0,1]$	$[0,0,0,0,1]$
fluffy (pos. 2)	$[1,0,0,0,0]$	$[0,0,0,0,2]$	$[1,0,0,0,2]$
blue (pos. 3)	$[0,1,0,0,0]$	$[0,0,0,0,3]$	$[0,1,0,0,3]$
creature (pos. 4)	$[0,0,1,0,0]$	$[0,0,0,0,4]$	$[0,0,1,0,4]$

If asked: BOS and *the* have the same all-zero word embedding only because our three meaning features say nothing about them. A real model has hundreds of dimensions, so every token gets its own vector.

Part 2a

X = masked (above the diagonal: a token may not look at later tokens). Shaded = where the attention goes.

Head A (previous token)

	BOS	the	fluffy	blue	creature
BOS		X	X	X	X
the			X	X	X
fluffy				X	X
blue					X
creature					

Head B (noun → its adjectives)

	BOS	the	fluffy	blue	creature
BOS		X	X	X	X
the			X	X	X
fluffy				X	X
blue					X
creature					

Head A is only possible because position is encoded in the embeddings. In Head B, every row must put its attention *somewhere* (the weights sum to 1), so non-nouns spread it evenly over every token they can see. Part 2b computes exactly this pattern.

(c) From Part 1(b), the target at each position **is** the next input token. If *blue* could look at *creature*, the model could simply copy the answer. The training loss would be near zero, but the model would learn

nothing useful, because at generation time the future tokens don't exist yet. The mask makes training match how the model is actually used, while still letting it train on all positions at once.

Part 2b

(a) *creature* is looking for **adjectives** (+6 on adjective, −6 on everything else). *fluffy* and *blue* are looking for a **noun**. BOS and *the* have all-zero queries: they aren't looking for anything in particular.

(b, c) Scores $= q \cdot k/\sqrt{4} = q \cdot k/2$. Because each key is one-hot, the dot product just picks out one entry of the query. Masked squares get $-\infty$ (shown here as X, with the pre-mask score in parentheses). Why $-\infty$: $e^{-\infty} = 0$, so after the softmax those squares get exactly zero weight and the rest of the row still sums to 1. Putting in 0 would not work, because $e^0 = 1$.

	k_{BOS}	k_{the}	k_{fluffy}	k_{blue}	k_{creature}
q_{BOS}	0	X (0)	X (0)	X (0)	X (0)
q_{the}	0	0	X (0)	X (0)	X (0)
q_{fluffy}	−3	−3	−3	X (−3)	X (3)
q_{blue}	−3	−3	−3	−3	X (3)
q_{creature}	−3	−3	3	3	−3

(d) Weights. Tied scores always give equal weights, so only the last row needs a calculator:

	BOS	the	fluffy	blue	creature
BOS	1	0	0	0	0
the	0.50	0.50	0	0	0
fluffy	0.33	0.33	0.33	0	0
blue	0.25	0.25	0.25	0.25	0
creature	0.00	0.00	0.50	0.50	0.00

creature row: $e^3 \approx 20.09$, $e^{-3} \approx 0.05$, total ≈ 40.32, so the weights are $[0.0012,\ 0.0012,\ 0.498,\ 0.498,\ 0.0012]$. Worth pointing out: softmax never gives an exact 0, and a row can't "attend to nothing" (*fluffy* and *blue* found no noun, so their attention got spread evenly).

(e) $\Delta e_{\text{creature}} = 0.5\,[1,0,0,0,0] + 0.5\,[0,1,0,0,0] = [0.5,\ 0.5,\ 0,\ 0,\ 0]$

$x_{\text{creature}} = [0,0,1,0,4] + [0.5,0.5,0,0,0] = [0.5,\ 0.5,\ 1,\ 0,\ 4]$. *creature* is still an animal at position 4, but now it also "knows" it is fuzzy and blue.

(f) $\Delta e_{\text{fluffy}} = \frac{1}{3}[0,0,0,0,0] + \frac{1}{3}[0,0,0,0,0] + \frac{1}{3}[1,0,0,0,0] = [0.33,0,0,0,0]$, so $x_{\text{fluffy}} = [1.33,\ 0,\ 0,\ 0,\ 2]$. *fluffy* learned **nothing** about *creature*: *creature* comes later, so it is masked.

Part 3

(a) Only the **verb** entry (4th column of W) can be non-zero: fuzzy + blue + 2 · animal − 1. It fires when the token looks like an **animal** (especially a described one), i.e., "a noun phrase just ended, so a verb is probably next."

(b) $x\,W + b = [0,\ 0,\ 0,\ 0.5 + 0.5 + 2 - 1,\ 0] = [0,0,0,2,0]$. ReLU leaves it unchanged, so $h = [0,0,0,2,0]$ and $x_{\text{out}} = [0.5,\ 0.5,\ 1,\ 2,\ 4]$.

(c) $x\,W + b = [0,0,0,-1,0]$. ReLU clips -1 to 0, so $h = 0$ and x_{the} is unchanged. That's the right behavior: *the* isn't an animal, so the "verb is next" signal should stay off rather than go negative. The ReLU lets the MLP act like an if-statement.

Part 4

(a, b) Logits = $x_{\text{out}} \cdot$ (each row of the word embedding table):

	BOS	blue	creature	fluffy	is	the
logit	0	0.5	1	0.5	2	0
e^{logit}	1	1.65	2.72	1.65	7.39	1
probability	0.06	0.11	0.18	0.11	0.48	0.06

(Total of e^{logit} row = 15.40.)

(c) "is" (48%). The **MLP** made it likely, by writing 2 into the *verb* feature. But the MLP only saw *creature*'s own vector, which attention had already enriched. The position feature had no effect: every row of the word embedding table has 0 in that column.

(d) loss $= -\ln(0.48) \approx 0.73$. If $p_{\text{target}} = 1$, the loss is 0. As $p_{\text{target}} \to 0$, the loss $\to \infty$: confident wrong answers are punished heavily.

(e) No. Sampling is random: you'd get "is" about 48% of the time, "creature" about 18%, and so on. This is why a chatbot can answer the same prompt differently each time.