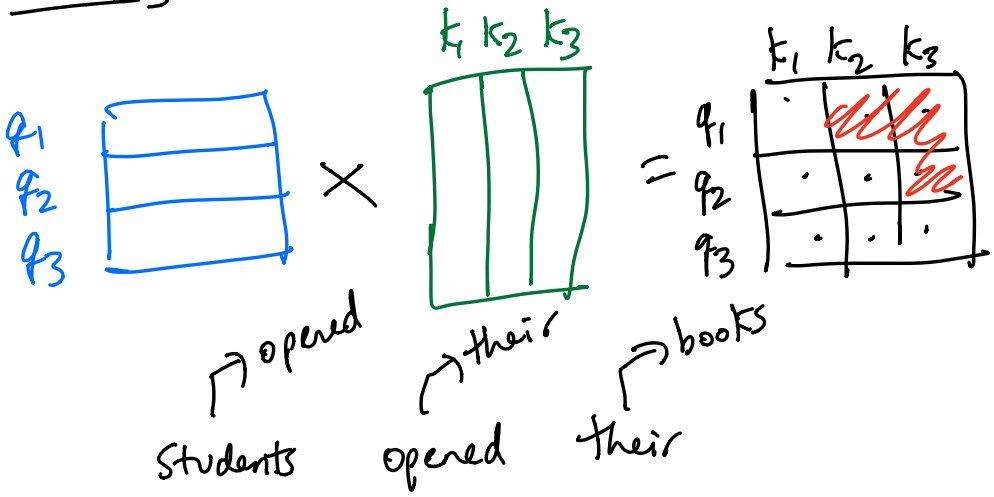
Self-attention:

↳ training time: parallelizable

↳ test time: sequentially

training:



- we can only parallelize this computation because we know the full sequence ahead of time

test time: q_1, k_1, v_1 q_2, k_2, v_2 ?

my dog

$\text{Softmax}(\langle q_2 k_1, q_2 k_2 \rangle)$

weighted ave of value vectors ↳ h_2 → softmax layer

dim d, hyperparam

Decoding: dim of value vector a $\hat{o} = \text{softmax}(W_o h_2)$ (200)

select a word from the pred. dist
- greedy
- sampling

$q_1 k_1 v_1$ $q_2 k_2 v_2$ $q_3 k_3 v_3$
my dog walked ?

$\text{softmax}(\langle q_3 k_1, q_3 k_2, q_3 k_3 \rangle)$

KV cache:

Storing previously-computed key/value vectors so you don't have to recompute them at every timestep

↳ test-time

- always start w/ a <SOS> <BOS> "start of seq"

Transformer:

→ neural LM built around multi-headed self-attention

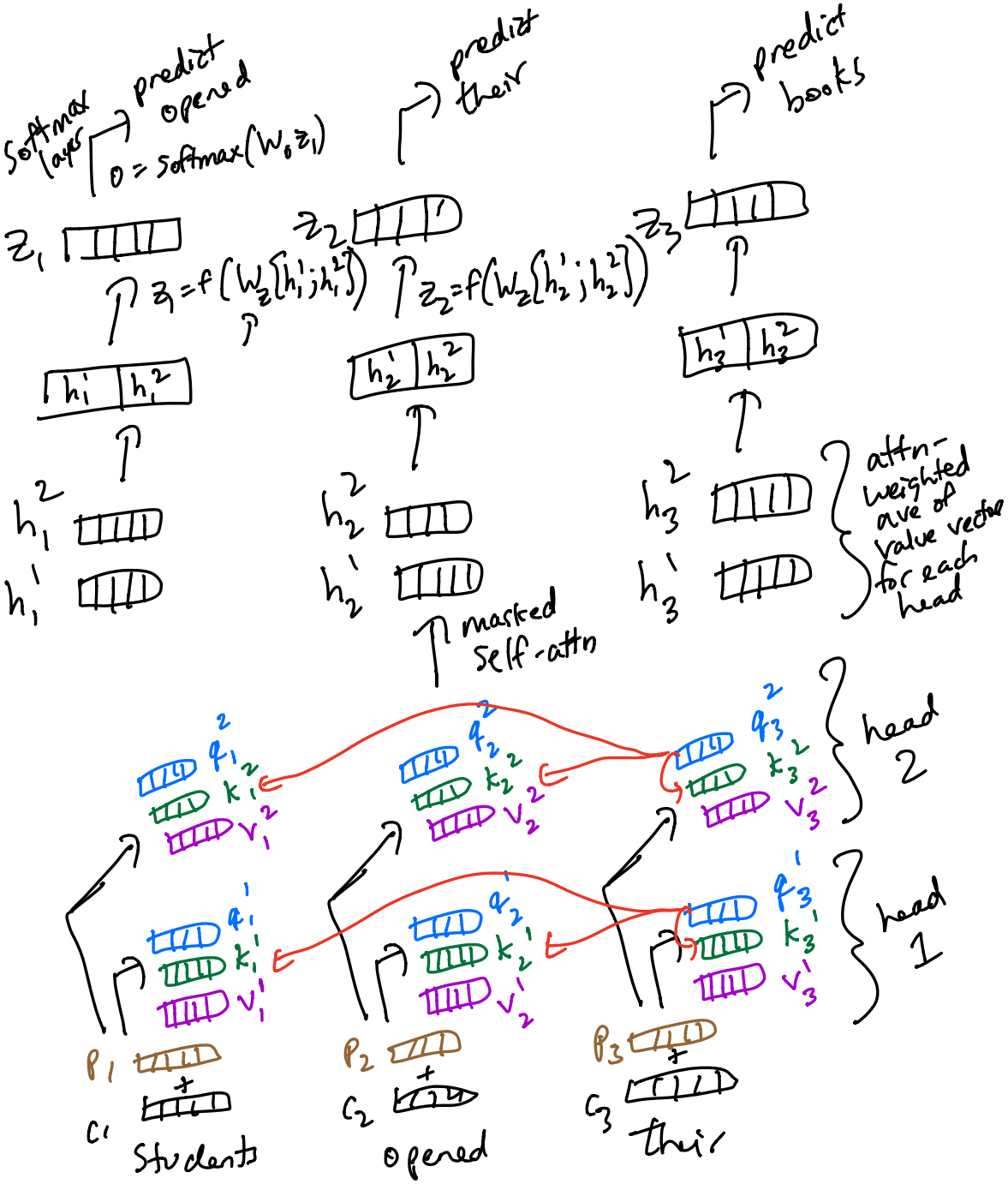
↳ stacked layers of attn, "deep"

multi-head self-attention:

→ Vaswani et al. 2017

→ intuition: let's have multiple sets of q, k, v vectors for each token, then maybe each "head" can focus on a specific linguistic property

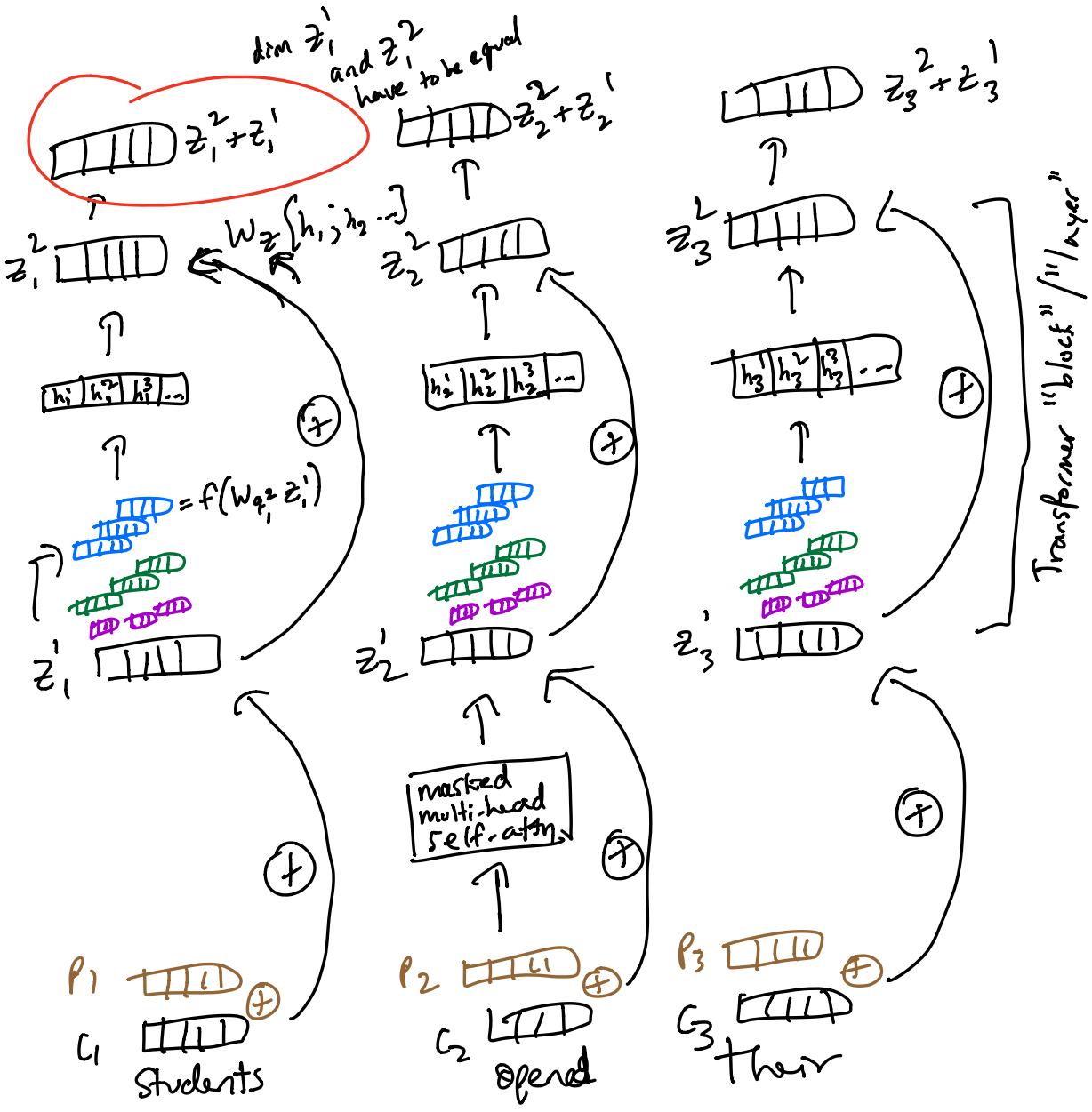
↳ Syntactic structures (all verbs in prefix)
↳ activate on certain words/phrases
↳ entities/dates



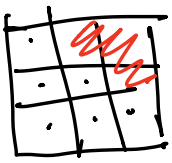
$$q_1^1 = f(W_{q^1}[p_1 + c_1])$$
$$q_1^2 = f(W_{q^2}[p_1 + c_1])$$

f = relu, $\max(0, x)$

Adding depth:



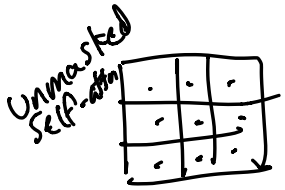
Decoder language model:



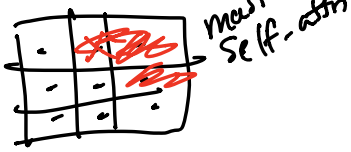
Sequence-to-sequence models:

↳ encoder ⇒ source lang

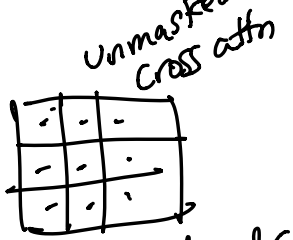
↳ decoder ⇒ target lang



encoder



decoder



queries ⇒ decoder

key ⇒ encoder

values ⇒ encoder