2025-12-29

Ok where are we at? We are discussing how the way we frame the question matter to what we get. So the overall goal at this stage is to find out how performance L changes with each attention score, S, and we know how L changes with A, now we just need to get how A changes with S. Because of symbol matching, it's very easy to write $\frac{\partial L}{\partial S_{ij}} = \frac{\partial L}{\partial A_{ij}} \cdot \frac{\partial A_{ij}}{\partial S_{ij}}$. That is I did not even get a second to consider "how A changes with S" carefully. A has many elements, so does S, should we consider how each element of A changes with each element of S? Yes we should. But we don't need to cross-rows checking, because we know each row of A only involves the S in the same rows.

So the question of how does the performance changes with the similarity score should be rephrased: how does the similarity score impact the performance? Well, each score will impact the attention weights, but not just one, but all the weights. S_{ij} is about the similarity between token j's offering vs token i's query. S_{ij} not only impacts the attention weights between i and j directly, but also indirectly. For example, S_{23} measures how good a fit token 3 is to token 2, but how much weight should token 2 gives to token 3? Well that depends on how well other tokens fit to token 2! That's why all the tokens will be part of the denominator, $S_{21}, S_{22,} S_{23}$. That is, how much attention should token 2 pay to token 3 not only depends on how good a fit token 3 is to token 2, but also how good others are to token 2. Hence, how ~~performance L changes with similarity score S_{ij}?~~ So how does similarity score S_{ij} impact the performance? Well, S_{ij} is going to impact all the weights about token i: how much attention token i should give to each token. S_{ij} will impact A_{ij} for sure, but also, $A_{i1,}A_{i2,}A_{i3}$ ...., hence how performance changes with S_{ij} should involve the layers through all the attention weights $A_{i1,}A_{i2}, A_{i3}$ ….$etc.$ $\frac{\partial L}{\partial S_{ij}} = \frac{\partial L}{\partial A_{i1}} \cdot \frac{\partial A_{i1}}{\partial S_{ij}} + \frac{\partial L}{\partial A_{i2}} \cdot \frac{\partial A_{i2}}{\partial S_{ij}} + \cdots \frac{\partial L}{\partial A_{ij}} \cdot \frac{\partial A_{ij}}{\partial S_{ij}} + \frac{\partial L}{\partial A_{i,j+1}} \cdot \frac{\partial A_{i,j+1}}{\partial S_{ij}} + \cdots$. Now speaking of $\frac{\partial A_{ik}}{\partial S_{ij}}$, there are two different ways how attention weights change with similarity score:

- If k=j, because the similarity itself matters directly, but also indirectly through the weighting.
- If $k \neq j$, the similarity score is not related to token k, so the impact will be indirectly through the weighting.

Hence we should expect two different formats for the $\frac{\partial A_{ik}}{\partial S_{ij}}$, depending on whether it's the same token's weight or not.

fg

$$\frac{\partial A_{ik}}{\partial S_{ij}} = -A_{ik} \cdot A_{ij} \ for\ j \neq k$$

$$\frac{\partial A_{ij}}{\partial S_{ij}} = A_{ij} - A_{ij}^2 \qquad for\ j = k$$

Ok, the question is, how to understand these formulas? Or is there a way to understand it at all, or just accept 'it is what it is'? First of all, what's confusing is this double indices, ij, but really the highlight is the 'j', 'I' is from 'I's perspective. Once we frame I's perspective first, it's easier to talk about: $\frac{\partial A_{ik}}{\partial S_{ij}}$ is talking about how token k's weight changes with token j's match, from i's perspective. Should be negative if you think about it, because the better j matches I's need, the smaller attention k would get! (See, this kind of intuition can only follow if you have an intuitive way to think about all those symbols!). Ok, so direction-wise we get it wrong, what about the details? Why it's the negative of the product of k's weight and j'th weight? So the higher the weight k get, the faster the decrease? The same thing with j's weight. So the negative sign should be framed into the narrative as well: instead of saying 'k's attention changes with j's matching score as negative blah blah…., we should say: k's weight *decreases* with j's matching as a product of k's weight and j's weight. So the higher those weights, the faster it decreases. It's not very clear. It makes sense j's weight is there, but why k's weight as well? That means if k's weight is high, then it will decreases faster with higher similarity score of j? hmm, also the expression above is misleading, misguided me into thinking that $-A_{ik} \cdot A_{ij}$ is very different from $A_{ij} - A_{ij}^2$, but actually they might be just similar! If you write $A_{ij} - A_{ij}^2 = A_{ij}(1 - A_{ij})$, where $(1 - A_{ij})$ is the non token's weight. Except in the first case, $-A_{ik} \cdot A_{ij}$ does not have all the non-k's token, but just j't token there. But structurally they are similar: a product of the target token's weight and other token's weight. So the key is how to understand this product, product involving the target token's weight, and other token's weight (whether there is just one, or all others). Maybe we should interpret the product as a 'weighted scheme'. For example, how fast token k's weight *decreases* with j's matching is proportional to k's weight, but weighted by j's weight. So the higher k's weight, the faster it's going to decrease with j's matching, but weighted by j's weight. Because k's weight and j's weight are not independent of one another, but inversely correlated. So this product scheme seems to give that little balance. $A_{ij}(1 - A_{ij})$ tells the same story: token j's weight increases with it's similarity score proportional to the weight, weighted by other token's weight.

1) So first of all, does it make sense that the rate is proportional to the weight?

- So in either case, the target rate appears to suggest that, whether the rate is increasing or decreasing, the higher the weight is, the more sensitive it is to the change.
- Maybe we should look at the exact math: $A_{23} = \frac{e^{S_{23}}}{e^{S_{21}} + e^{S_{22}} + e^{S_{23}}}$

2) Then does it make sense that there is a little balancing scheme out here?

K let's have a break from this and proceed…

So $\frac{\partial L}{\partial S_{ij}} = \frac{\partial L}{\partial A_{i1}} \cdot \frac{\partial A_{i1}}{\partial S_{ij}} + \cdots + \frac{\partial L}{\partial A_{ij}} \cdot \frac{\partial A_{ij}}{\partial S_{ij}} + \cdots +$

$$= G_{i1}(-1)A_{i1} \cdot A_{ij} + G_{i2}(-1)A_{i2} \cdot A_{ij} + G_{ij}A_{ij} \cdot (1 - A_{ij}) + \cdots$$

$$= A_{ij}[-G_{i1}A_{i1} - G_{i2}A_{i2} + G_{ij}(1 - A_{ij}) + \cdots]$$

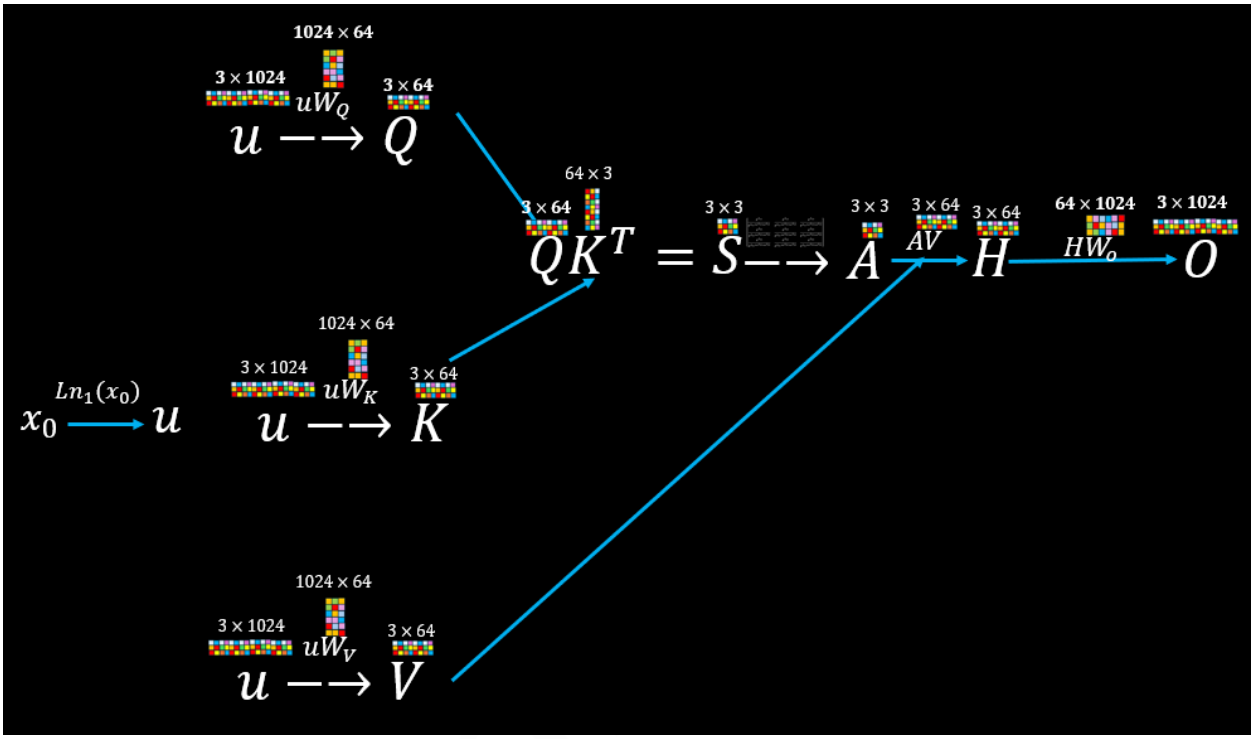
$$= A_{ij}[G_{ij} - (G_{i1}A_{i1} + G_{i2}A_{i2} + G_{ij}A_{ij} + \cdots)]$$

Nice pattern, surprise? Expected? No, I have no expectation, and i can't explain much about this result. And GPT complicates things further by bringing in more abstract symbols. I have one question: using simplified symbols is supposed to make things easier, does it?

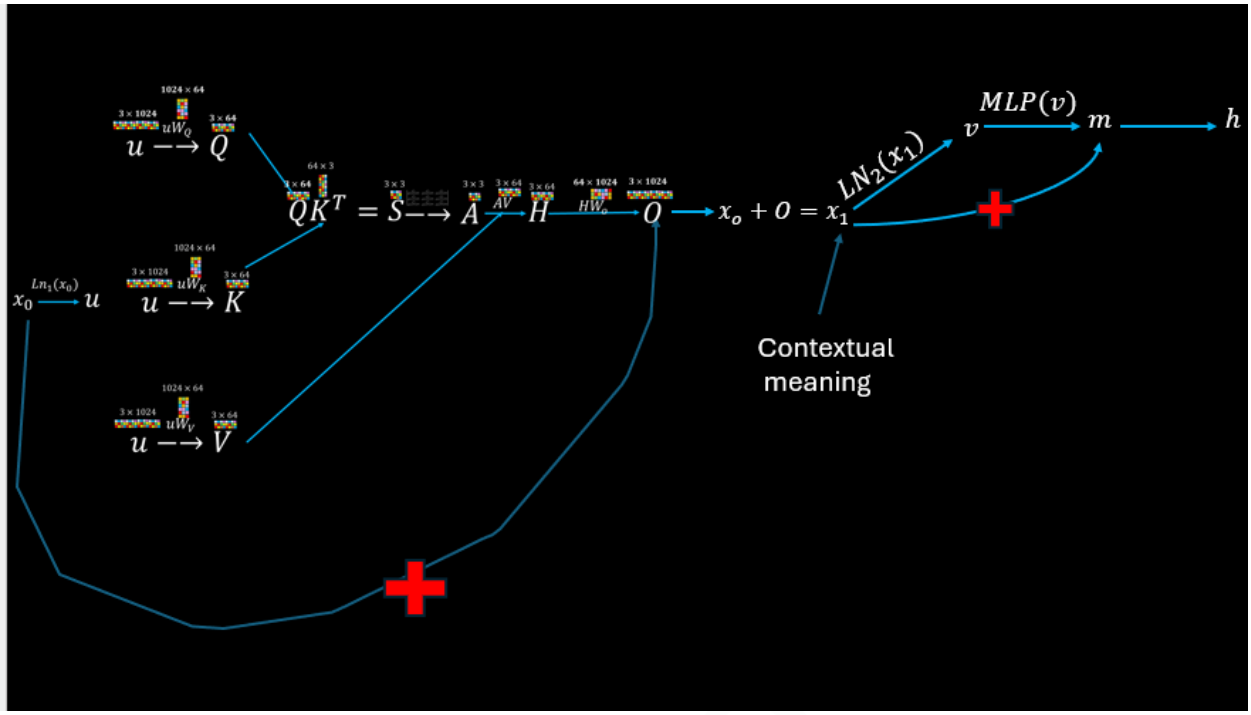
Let's delay this vectorization first and proceed to the next step. Now we know how the performance changes with each similarity score, now we need to push back one more step, how does similarity score changes with its input? Well, there are several branches here. Similarity score involves both the K and the Query, so we need to know how the performance changes with the Key and the query respectively.

2025-12-31

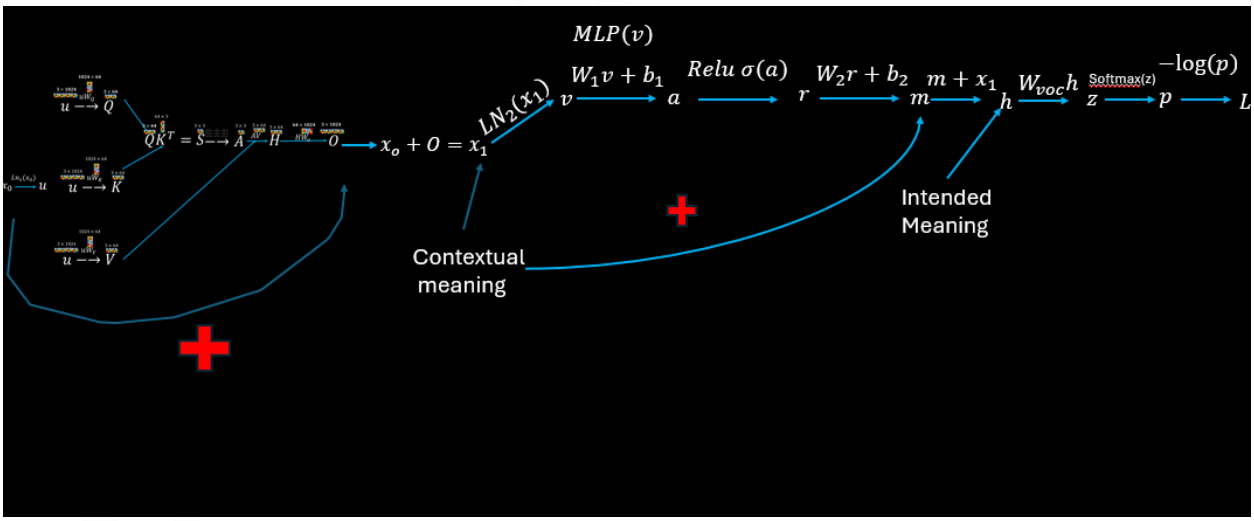
So now I want to piece together to have a big picture, the problem is, I almost forgot what I did before. I want to put the attention block and the feedback together, then from the perspective of backward propagation. One thing I need to correct is the input to attention block, GPT kept talking about linear norm and I have been brushing it away, until now.



Now you see, the input to the attention block is not the token itself, but, the token after linear norm transformation! Remember in our similarity discussion, we mentioned that the similarity score could be blurred by the asymmetrical length of a single vector. Hence some sort of normalization to make them equal length would be necessary, I guess this linear norm is doing that. So the normalized tokens go into the attention block for the attention weights calculation. The output is added back to the original token. So what does this suggest then? That means, the output is not the contextual meaning, but the meaning adjustment!!! It is like in the stock modeling exercise, instead of modeling the stock price directly, we model the change, the change is added back to the price to come up with a new price. Now the question is: *why can't we just use this block to figure out the contextual meaning, but instead, we use it to figure out the adjustment (or, the change?).* let's park this question and proceed with the big picture for now.



Now we attach the following steps, the contextual meaning x_1 will be normalized before going into feed-forward step I think! And the MLP mentioned by GPT, I think that's the feedforward step. Now let's attach the MLP details:



So the normalized contextual meaning goes through a linear mix, then relu, then another linear mix to become the 'adjustment' again, then added to the input meaning, we get the intended meaning! This intended meaning will be checked 'against the dictionary' to come up with similarity, then softmax to turn those into probability, finally, probability turned into performance valuation through log loss function.

Don't get lost in this long chain of events. Don't forget about the overall goal: to do tuning through smart knowledge. But to tune what? There are so many layers of differentiation, they are speedy ways

to know: how one changes with the other. But you should know there are two different purposes for knowing those changes: