

2025-11-17

So let's rethink the main theme here. The goal is to produce the next word. 'when I think', how to fill the blanks? Of course we would take a look at previous words and try to think about which words might fit. That's exactly the spirit here in LLM: figuring out which words might fit other words. But with some differences. Here we only need to figure out which word fit, then that's it, we PUT IT THERE! So whatever fits will be the next predicted word! Also when we check if one word fits another, it's a binary decision! Whether it fits or not! It's not a continuous measurement! Also we tend to think directly two consecutive words, like what's after 'when I think ...', but we are not just thinking about 'think...' because the previous words before think also matters. I mean, you will fill up with different words whether seeing 'when I think ...(about you?)' Vs just 'think (carefully?)...'. So although we are trying to predict the word directly after 'think', yet its previous words matter. Hmm. That's why it makes sense LLM uses the 'last' word as a *vehicle* to aggregate previous meanings together as a 'context-aware vector'?

So here are what we need:

- 1) So we know we need to check how a word fits to another.
- 2) And we also need to aggregate the meanings of 'all the words'.

Let's see how LLM solve the above two:

- 1) LLM checks how good each word matches another, regardless of their distance.
- 2) LLM aggregates the meaning from the perspective of the last word.

That works? Based on the fact that LLM is popular it means this method works. But the question is, why it works because it does not quite make sense: how can you aggregate meaning from the perspective of the last word? I think the distance problem is probably fixed by some 'positional offset', we can ask more details later. But on 2, you know humans are really good at talking themselves into something they want to believe, so here is my logic on 2: why aggregate the meanings from the perspective of the last word, because we are predicting the next word, the word immediate in front should be critical, then it sort of make sense to use the last word as vehicle to aggregate meanings.

Ok can we move on? We haven't resolved the above two questions, we will keep coming back, but we will move on now.

Actually just had more thoughts about

- 3) LLM aggregates the meaning from the perspective of the last word. If we find weird, we should ask: what is the alternative? Which ones won't make you feel weird? Well, you realize that if we aggregate meanings from a different word, then the overall meaning could be different. So, a fairer way would be, to aggregate meaning from each word's perspective, and then average! Right??? For sure we will grill GPT on that later. So now we can move on.

Two directions. Either we zoom into more nitty-gritty details about 'multi-head', or , we move on big picture wise: now we know how each word matters to another, then what? maybe the big picture still.

So now we know the weights, we should use them to update meanings. Since only the last word's (word3) perspective is used, so only last row's weights matter here. A_{31}, A_{32}, A_{33} . Then I would imagine to just multiply each weight by each word and add up! Something like: $A_{31}W_1 + A_{32}W_2 + A_{33}W_3$. But no! That's when the concept Value comes up, Value of a word! After some struggles, here is how I interpret value vs word, word's vector representation is like being, the words' being, as captured as what they are. But what do they do? V seems to be doing that, turning words' being into some actions. The process of coming up with value is also linear mixture: $XW_V = V$. Now think about it, here is what happens so far:

1. We linearly mix meaning components to make query
2. We linearly mix meaning components to make key
3. We linearly mix meaning components to make value.

Does it make sense that:

- 1) Query, key, value are all related to words?
 - I think so
- 2) Their relationship is linear?
 - Not so much, because, that simple???

But I think it's time to drill down into the meaning of Value.

k, let's understand the meaning of V. Attention matrix tells us the weights for each word. i would naively think: just multiply each weight by each word and add them up, that's the combined meaning. but no, weights are applied to Value. why?

Thought for 9s >

Great instinct. The short reason: **Q/K do the routing, V carries the message**. We separate them so we can *match on one set of features and transmit a different set*.

- If you weighted raw words X directly (do $A \cdot X$): the exact features used to *decide* who to attend to would also be the ones you *blend*. That entangles “address” and “package” and adds noise.
- With Values $V = XW_V$:
 1. **Decouple match vs. content**. K/Q can key on “does this look like a head noun?” while V passes only “number=SG” (not all noun details).
 2. **Head-specific payload**. Each head has its own W_V , so one head can pass **number**, another passes **entity type**, even if both match the same token.
 3. **Denosing/compression**. W_V filters out irrelevant features before mixing.
 4. **Stable scaling**. We size/scale d_k for matching, d_v for payload—independently.

Analogy: K/Q = **mailing address matcher**; V = **the contents of the package**. You look up by the address, but you deliver what's *inside*, not the address label.



Sort of make sense, but very confusing actually. A is the simple weight, 20% of w_1 , 30% w_2 , 50% w_3 , where is ‘who attend whom’? Ok I see, the way we calculate those weights, from word3’s perspective, is actually ‘word3 attends to...’, A_{31} is how much word3 should attend to word1, A_{32} is how much word3 should attend to word2, A_{33} is how much word3 should attend to word3, itself. Ok, so if that’s how much attention word 3 should be paying, then I don’t see anything wrong with weighted average of the original words, still! And what is number=SG? Hmm unless, ok, A is calculated not in meaning space, hence it should not be applied on meanings either?

Maybe we should ask: what does V mean really?

2025-11-18

So first we have words, decomposed into 'meaning components'. Then we have Key as linear mixture of meaning components, Query as linear mixture of meaning components and Value as linear mixture of meaning components. How to understand this? The problem here is we don't have very clear understanding about those meaning components, let alone Key, Query, Value components! It is not like a $F=ma$ equation where we know exactly what F , m , a represents and what kind of relationship among them is here. In LLM I feel we are in the dark. But the problem is: you can't be all in the dark without any clues, otherwise, how would you know to set up a linear mixing structure to produce K , Q , V , and they run QK^T and then do AV ? I mean, if you do QK^T that means pretty much you understand one should be query one should be Key. And you set up $XW_Q = Q$, that means you pretty much understand why Q should be a linear mix of the words! I don't need to see exactly why the parameter is 0.5 of this, 0.25 of that, **but at least I need to see why this Key should be related to some meaning components!**

Again GPT's answer is overwhelming and dodge my ask! I asked for a few examples where I can see how a words' query is RELATED to its meaning components. Take the word 'is' as an example, its query vector should say that:

1. It is present tense
2. It needs a 3rd person
3. It needs a singular subject on the left.

Ok, so what kind of meaning components might produce this kind of Query then? Pronounness? Ok, so among the exemplar embedding GPT gave:

Toy meaning space

Imagine each token embedding lives in a space with axes like:

1. `subjectness` (how much this word is a subject-type NP)
 2. `objectness`
 3. `verbness`
 4. `negatoriness`
 5. `determinerness`
 6. `noun-headness`
 7. `pronounness`
 8. `number_singular`
 9. `number_plural`
 10. `past_tense`
 11. `present_tense`
 12. `animacy`
- ... etc.

It should have a few boxes checked. But checking those boxes is mainly from perspective of what that token is. It is like a verb, it is a pronoun, it is singular, it is present tense.... so, we can pretty much see this is what Key should 'advertise' for.

Then I got distracted into the meaning space:

pause, i'd like to zoom into the meanint space. take subjectness as an example, you use 'how much', that means it's not binary, it's sort of 'continuous', i would think 'orange' is subject, so 1, 'cat' subject 1 as well, 'abstract' '0'.. etc, can you list a few words to help me understand that it might not be just binary, it might be a continuum?

Right, let's zoom in on that "subjectness" idea.

When I said "how much subjectness," I was implicitly treating "subjectness" as a **continuous** feature, not a clean 0/1 tag.

Think of a scale like this (purely conceptual):

0.0 = "never behaves like a subject"
1.0 = "extremely prototypical subject word"

Now imagine rough placements:

Word	Why its "subjectness" might be high
I, you, he, she, we, they	Personal pronouns are classic subjects : "I think...", "She runs...". They <i>can</i> be objects ("He hit me"), but across usage they're <i>very</i> strongly subject-like. So maybe ~0.9+.
people, children, students	Prototypical agentive plural nouns : "Children play", "Students learn". Often do actions, often appear as subjects ⇒ very high, maybe ~0.8–0.9.
dog, cat, bird	Animate count nouns: "The dog barked", "The cat jumped". Often subjects <i>and</i> objects, so still high, say ~0.7–0.8.
teacher, driver, worker	Roles that scream "doer". Strong bias to subject: "The driver turned left." Maybe ~0.8–0.9.

Now things start sliding down:

Word	Why "subjectness" is moderate
car, table, door	Inanimate objects: show up as subjects <i>and</i> objects. "The car stopped." / "I bought a car." Subjectness middling, say ~0.4–0.6.
idea, plan, theory	Abstract but still often treated as propositional subjects: "The idea worked," "The plan failed." Maybe ~0.4–0.6.
water, furniture	Mass nouns: often objects ("drink water") but can be subjects ("Water evaporates."). Subjectness maybe ~0.3–0.5.

Even lower:

Word	Why "subjectness" is low
time, fact, way	Very frequent abstract nouns that often sit in fixed patterns ("the fact that...", "the way to..."). They <i>can</i> be subjects ("Time passed"), but often serve as grammatical shells. Maybe ~0.2–0.4.
north, left, edge	Spatial nouns, often complements or modifiers. "To the north", "at the edge". Occasional subjects, but rare. Maybe ~0.1–0.3.

Very low (almost never subjects):

Word	Why "subjectness" is near zero
of, in, to, not, and	Function words like prepositions, conjunctions, negation. Essentially never used as heads of subject NPs. Subjectness ~0.0.
quickly, very, not	Adverbs/particles; same story.
run, eat, think (as verbs)	In their verb sense, they're not noun-phrase subjects at all. They can <i>become</i> nouns ("the run", "the eat" in weird contexts), but that's rare. So as plain verb tokens: subjectness ~0.0–0.1.

So the picture:

- **Not binary:**
"cat" and "people" can *both* be subjects, but "people" is even more "subject-y" (more often agents, plural, human).
- **Continuum of preference:**
The model can learn that some tokens **tend to appear as subjects much more than others**, reflected in their embeddings.
- **Emergent axis:**
In reality, there isn't literally a single "subjectness" neuron. But there will be some direction in the big vector space that correlates with this gradient. When we talk about a "subjectness axis", we're approximating that emergent direction.

So when W_Q or W_K "looks at subjectness", it's really giving **more weight** to whatever combination of internal dimensions correlate with **"this word behaves like a typical subject NP"**, with pronouns and agentive nouns at the strong end and function words at the weak end.

Ok let's talk about a word's 'embedding', or, meaning space. When you look at the possible 'axis' (meaning components) such as:

- Subjectness
- Objectness
- Verbness
- Negatorness
- Determinerness

- Noun-headness
- Pronounness
- Number_singular
- Number plural
- Past tense
- Present-tense
- Animacy
-etc

You see, they are not really about the meaning, they are more like ‘descriptor’ of a word. Some high level feature that generalize how a word is used, what it is, its tense etc. It’s almost like those word-games where you can use various ways to describe a word rather than say it so that others can guess the word. Then I asked GPT to zoom into the first dimension, ‘subjectness’, I am curious as to how this can be a continuum. My understanding is it’s 1 or zero. I asked it to provide a list of words where I can see this number vary accordingly. Let’s see how this measurement vary with words. (function???) I would expect nouns to fit this category, words other than noun should have zero scores.

Take a look at GPT’s answer, while it’s expected that function words, adverbs, verbs etc will have score close to 0. But it’s interesting to see among different nouns, their scores are different. And all these scores are learnt, indirectly usage-frequency based.

Also I kind of appreciate why GPT kept reminding me that those ‘axis’ are mixtures rather than pure axis. Take ‘subjectness’ and ‘objectness’ as an example, they are highly correlated. For those who have high subjectness score, they tend to have high objectness score. Take past-tense, present-tense for example, they should be almost always oppositely correlated, because if you are past tense you can hardly be present tense and vice versa. So it’s not easy to explain at this stage, the difference between ‘axis’ and ‘mixture’. But in short, the meaning space is murkier than you think. It’s almost that, those basic components don’t correspond to ‘past-tense’, ‘present-tense’ directly, but rather correspond to some ingredients that make ‘past-tense’ and ‘present-tense’. Then it’s tempting to ask: what are those ingredients? The truth is, perhaps, no-one knows!

Now can we have better understanding about the linear process of making query, key, value based on those meaning components? These need to be understood with the concept of ‘head’.

So when we analyze a word’s query and key, it should be based from a certain matching-criteria. One could be based on ‘subject-verb agreement’, explained by GPT: when the verb forms its query, it should strongly look for a subject NP with matching number/person. What is ‘subject NP’? Ah, ok, so a verb will need a subject noun matching the number and person. ‘Kick’ is looking for someone plural, ‘kicks’ is looking for someone single. Another perspective is from ‘determiner-noun head head’??? (What???) the goal is to bind determiners to appropriate nouns, for example, word ‘a’ should follow a noun object single. Then comes the perspective of ‘negation linker’, “bind ‘not’ to its auxiliary/verb’. Then another aspect is, ‘pronoun-antecedent head’ with the goal to connect pronouns like ‘it’ to their antecedent noun phrases. Ok, so does it almost read like a connection type based on word? Or word type?

For example, if it’s a verb, then it should be matched to a subject noun based on ‘agreement’, that is, the right plural form. If it’s a noun, it should be matched to a verb based on agreement too? Then are

they belonging to a same 'head'? I think the same one. If it is an adjective? Then it should be matched to a verb agreement? That is, there must be a 'is, are, were, etc' in front?

Ok I think the reason I am struggling understand this piece is, I know how to use daily English, but I am not really clear about the rules literally, I am not familiar with those names even. Like, I don't even know what a 'pronoun' is, even though I can say 'I don't like it'.

Then ask another question:

back to your head, what kind of head is possible for an adjective? i mean, seems that the first subject-verb agreement head is made for noun and verbs

Thought for 7s >

Nice, this is a good place to zoom.

You're right: that **subject-verb agreement head** I described is basically a **noun-verb** story. So what are the analogues where **adjectives** are the stars?

Think of "heads" as **little specialists**. Some specialists *need* adjectives to do their job. Here are a few very plausible ones:

1. Adjective-noun binding head

Job: glue each adjective to the noun it's modifying.

Example:

the big angry black cat

For **this head**:

- **Q(adjective):**
"I am a modifier; I'm looking for a nearby noun head to attach to (probably on my right)."
- **K(noun):**
"I am a noun head; adjectives to my left can modify me."

So:

- Q(**big**) → K(**cat**)
- Q(**angry**) → K(**cat**)
- Q(**black**) → K(**cat**)

This lets the model treat *big/angry/black* as **features of "cat"**, not as separate, unrelated words.

2. Degree / intensifier head (adverb ↔ adjective)

Example:

a very tall building

Head specialized for **degree**:

- **Q(tall):** "If there is a degree word right before me, pull it in."
- **K(very):** "I am a degree/intensifier; I want to modify an adjective on my right."

So Q(**tall**) attends strongly to K(**very**).

The composed meaning is "tall++", not just "tall".

Same for:

- *really happy*
- *extremely cold*
- *so tired*

Hmm, so these heads almost read like the ‘occasions where words are binded’. Like adjectives can be connected to ‘degree intensifier’, like ‘very bad’. I like the summary given by GPT:

big picture

- Some heads are very **noun-verb-ish** (agreement, subjecthood).
- Some are **adj-noun** (modifier → head).
- Some are **adv-adj** (degree/intensifier).
- Some are **copula-adj** (predicative role).
- Some are **negation-adj** (polarity).

So you can see these ‘heads’ are very ‘word-type specific’, like, noun-verb agreement only applies to noun and verb, if you are an adjective, this head has nothing to do with you! But the processes must go on, so what will happen? Almost 0 entries for this word? GPT kind of agrees with me, but with more details about the 3 knobs a token can be ignored. I want to hold off reading the details, but want to talk about our original question: now understanding a bit more about head, about key and query, does it make sense that key and query are linear mixture of meaning components?

For example, if we are talking about how two words match from the angle of ‘noun-verbish’, then what meaning components should be picked? Nounness? Tense? Plural? Verbness? Hmm, there are mainly other components that don’t matter. So the linear mixture make sure the ‘relevant’ ones get considered? And non-relevant ones ignored?

GPT confirms and offers some more information about more benefits, ie, this linear mixture can form new direction (combo) so a single number represents a combo of components like: noun phrase is subject like and 3rd person and singular’. Ah ok. Sort of get it.

But I have a more urgent question: **how to understand the value a word should pass if attended to? Why it’s also a linear mixture of the meaning components? Ok, it’s for tomorrow!**

2025-11-19

So my original expectation is the weight would be used upon the words, or, meaning directly. But it is not. It will be applied to a word's "Value". So how to understand a word's value vs. its meaning? So GPT explains the value as "if I am the word you are looking for, what value am I going to send"? Then we should ask: why not just my meaning, why do I have to tweak the meaning to send something else? Now think about a concrete example, the word 'orange', its meaning components could have: it's can be either a subject or an object, it's edible, it's orange color, it's round shape, it's single, etc. So these meaning components pretty much refer to its 'being'.

1. The **orange** is **tasty**
 - If the word 'tasty' is asking for 'orange', then its tastiness, edibility should be picked up.
2. He is **peeling** an **orange**
 - If the word 'peeling' is asking for 'orange', then it's abjectness should be focused.

Ok, but we are not passing different values based on different words. Its value is determined, the attention matrix make sure the right word gets the right value. Let's see what GPT says. Based on what it said, sounds like this linear mixture to create value is to make sure the right 'components' get picked up for this head's purpose. So it cannot be understood standalone, it needs to tie back to the meaning of a particular head.

1. "What should the value be?" = "If someone looks at me *for this purpose*, what should I send?"

Think head-by-head.

Example A: subject-verb agreement head

Job: make the **verb form** agree with the **subject**.

- When the **verb** attends to the **subject**, what should it receive?
 - The subject's **person/number** features. Not its color, sentiment, etc.

So for this head:

- $V(\text{subject}) \approx$ linear mix that keeps:
 - person, number, maybe animacy, etc.
 and downweights:
 - color, concreteness, polarity, etc.

Then:

$$\text{output}_{\text{verb}}^{(\text{head})} \approx \sum_j A_{\text{verb},j} V_j$$

is basically:

"a distilled little vector encoding the subject's agreement features"

Later the MLP can use that to pick "is" vs "are".

So:

- **Q/K**: decide that verb should look at subject.
- **V**: decides *which pieces* of the subject's meaning to actually pass along.

"If someone picks me for this purpose, what should I send". Hmm, where is 'this purpose' appear? Well, that's per head! So if this head is for checking subject-verb agreement, then person, number, animacy components should be 'kept'. Other components like 'color, concreteness, polarity (?)', should be downplayed. Sound like sort of 'muting', muting certain aspects while display others 'appropriately'. Based on this explanation, I think value is very similar to Key then! So what's the difference between them?

Just have an idea while working on a treadmill (this is why diffused mode thinking is useful).

Why not think of the simplest case, like, back 10,000 years ago (or later, or before), when we first started talking, at that time the world is simple, the words should be simple. So in the simplest world, all we need is noun, verb, adjective, adverb. (ok I don't even see a need for 'adverb' then, whatever!).

In this tiny toy example, I just want to understand why we need to linearly mix the meaning components to make, Key, Query and Value. Particularly I was struggling with the difference between Key and Value, I thought 'aren't they the same thing'?

The reason I got confused is, I only thought about the dimension, I did not think about the values. For example, if our world is small and we only distinguish words by noun, verb, adjectives, those are the three dimensions, so our word will be a vector of a size of 3. But then, under each dimension, the value should be varied so that it can help us pin-point the exact word. For example, under the noun dimension, 1 is for 'ball', 2 is for 'stone', 3 is for 'tree', 4 is for 'river, etc..., under verb, 1 is for 'kick', 2 is for 'run', 3 is for 'lie', etc. So basically under each dimension, we should have enough richness to distinguish one noun from another, one verb from another.

	noun	verb	adjective
Ball	1	Kick 1	Bad 1
Stone	2	Run 2	good 2
Tree	3	Lie 3	Smelly 3
River	4	Curl 4	Yummy 4
Fish	5	Eat 5	Blue 5
Cow	6	Drink 6	Dark 6
Sky	7	Sleep 7	Small 7
Star	8	Is 8	Big 8
...	...	Am 9	Hard 9
			soft 10
			...

Now when we do the query analysis, we only care about if the 'word' type matches, not the exact state. For example, "kick" is a verb, its query should be a noun, it does not need the value of '1' for ball, or value of '2' for stone. Etc. So it's query should indicate a 'noun' is needed. Then for 'ball', from its key's perspective, it is a noun, it does not need to specify which noun. Then from individual word's perspective, if it's picked, then what value they should pass on? Ball should pass a value of '1', stone should pass a value of '2', tree should pass a value of '3'. **So I would think Key and Query are from more general perspective, while Value focuses on the actual state** (value). But if that's the case, then shouldn't key and query have smaller dimensions than value? Ok, in this tiny example, it's hard to see why Key and query should be multiple dimensions, actually one dimension, three states is enough! 1 – for noun, 2—verb, 3—for adjective! But it does show one thing: why k, q should be a smaller space than the original word. But then we need to figure out why they are also multi-dimensional?

2025-11-20

I am back at square 1! Why using a vector to represent a word? In my tiny toy case, where there only three types of words, noun, verb, adjectives, we can use three numbers to represent three type, why three vectors? I think we don't just need the word type, but we also need a way to pinpoint the exact word. If you just 1, 2, 3 to represent three types, how can you represent different nouns under 'noun', different verb under 'verb'. Ok, **So, using number to represent different word type is a no go.**

Ok, so we need a way to represent each word, why not just numbers? Going alphabetically or not, we can just number the words 1, 2, 3, ...,50,000. Now each number can correspond to exactly one word, what's the problem now?

Then maybe we should go back to the previous attempt of categorizing words according to their types, why? What does this allow us to do? Well, this allow us to find match. For example, if it is a verb, we know we probably need a noun in front. Hmm. Ok, so words category allows us to match. That's impossible with simply numbering words from 1 to 50,000! So the lessons now is:

- 1) Naming words after each number accomplish the goal of pinpoint each word to a number, but it does not help us with matching words (this is based on the assumption that predicting words is based on matching words).
- 2) Categorizing words after their 'function type' help us with matching words. Now the question is, how to represent the category using numbers?
 - First choice is again scalar, 1 for noun, 2 for verb, 3 for adjective. But then we are not able to pinpoint each word.
 - So the alternative would be a vector of 3 components, [x,0,0] representing a noun, [0, x, 0] representing a verb, [0, 0,x] representing an adjective. The 'x' there allows us to represent each different word under each category.

Ok, so now we may have agreed that we need to represent a word as 3-dimensional vector, then under each dimension, an exact number will help us narrow down a word. Like this:

	noun		verb		adjective
Ball	1	Kick	1	Bad	1
Stone	2	Run	2	good	2
Tree	3	Lie	3	Smelly	3
River	4	Curl	4	Yummy	4
Fish	5	Eat	5	Blue	5
Cow	6	Drink	6	Dark	6
Sky	7	Sleep	7	Small	7
Star	8	Is	8	Big	8
...	...	Am	...	Hard	...
				soft	
				...	

The vector structure allows us to do basic type-match, for example $[x, 0, 0]$ is a noun, $[0, x, 0]$ is a verb, the query analysis should generate a key of $[1,0,0]$ to represent a 'noun', so that doing the dot product anything like $[x,0,0]$ would match. So the vector structure allows us to do:

- 1) Type match
- 2) And keeping the state

But then, our world gets complicated, we have 'nuances', good noun, bad noun, good verb, bad verb. I know there are, but just can't think of any. So go to GPT for such examples, starting with noun:

1. Frugality vs. Stinginess
2. Confidence vs. Arrogance
3. Curiosity vs. Nosiness
4. Patriot vs. Nationalist
5. Leader vs. Boss

Great examples!

Also here you go:

1. Save vs. Hoard
2. Guide vs. Control
3. Persist vs. Cling

And here are the adjective examples:

1. Selective vs. Picky
2. Direct vs. Blunt
3. Skeptical vs. Cynical

And another angle for such nuance is animal vs. human:

1. Eat

- **Animals:** *graze, forage, scavenge, hunt, feed*
- **Humans:** *eat, dine, snack, feast*
→ Same core action; "graze/forage/scavenge" feel very animal-y. Saying "He grazed in the kitchen" is jokingly animal-like.

2. Move (general)

- **Animals:** *prowl, slither, gallop, scurry, slink, pounce*
- **Humans:** *walk, tiptoe, jog, stride*
→ We can borrow the animal verbs for humans metaphorically: "He prowled around the room."

3. Make a home

- **Animals:** *nest, burrow, den, roost* ("to nest," "to burrow")
- **Humans:** *live, settle, move in, stay*
→ "They nested on the couch" feels playful because "nest" is default for birds.

4. Speak / make sound

- **Animals:** *bark, meow, moo, chirp, growl, purr, hiss*
- **Humans:** *talk, speak, shout, whisper*
→ Using animal verbs on humans gives instant metaphor: "She growled at me," "He hissed the answer."

5. Reproduce / care for young

- **Animals:** *mate, spawn, brood, nurse, wean*
- **Humans:** *have children, give birth, raise, breastfeed, wean*
→ Some overlap (*nurse, wean*) but "spawn" ↓ "brood" sound comic or dehumanizing when applied

2025-11-21

We can accept that word is represented as vectors, three matrices produce Q, K, V , QK^T is used to create the attention matrix A , A will be applied to V the create a 'context-aware vector' to be fed into 'feed-forward' step. Even if we can memorize these steps, get familiar by carrying out the programming steps, that does not mean we really understand these. Especially the question that keeps bothering me is: what is the difference between Key and Value? Without going into the details, it's hard to really understand, except memorizing the high level summary offered by GPT. So we have to gain deeper understanding into word's representation.

We have first reviewed the possibility of representing words using scalar, from 1 to 50,000. We know we can't but the question is why. It does not mean in no situations should we represent words as scalar. It always depends on what you want to do with those words. If we want to summarize the frequency of each word occurring, then naming them after a scalar number. But we want to predict the next word. What's the basis for such prediction? Based on word's function: noun, verb, adjectives. That's why not only we need to represent the words' state, but also 'type'. So 3 dimension vector is necessary so that we can

- Know the word's type so we can do the matching
- Pinpoint the exact word.

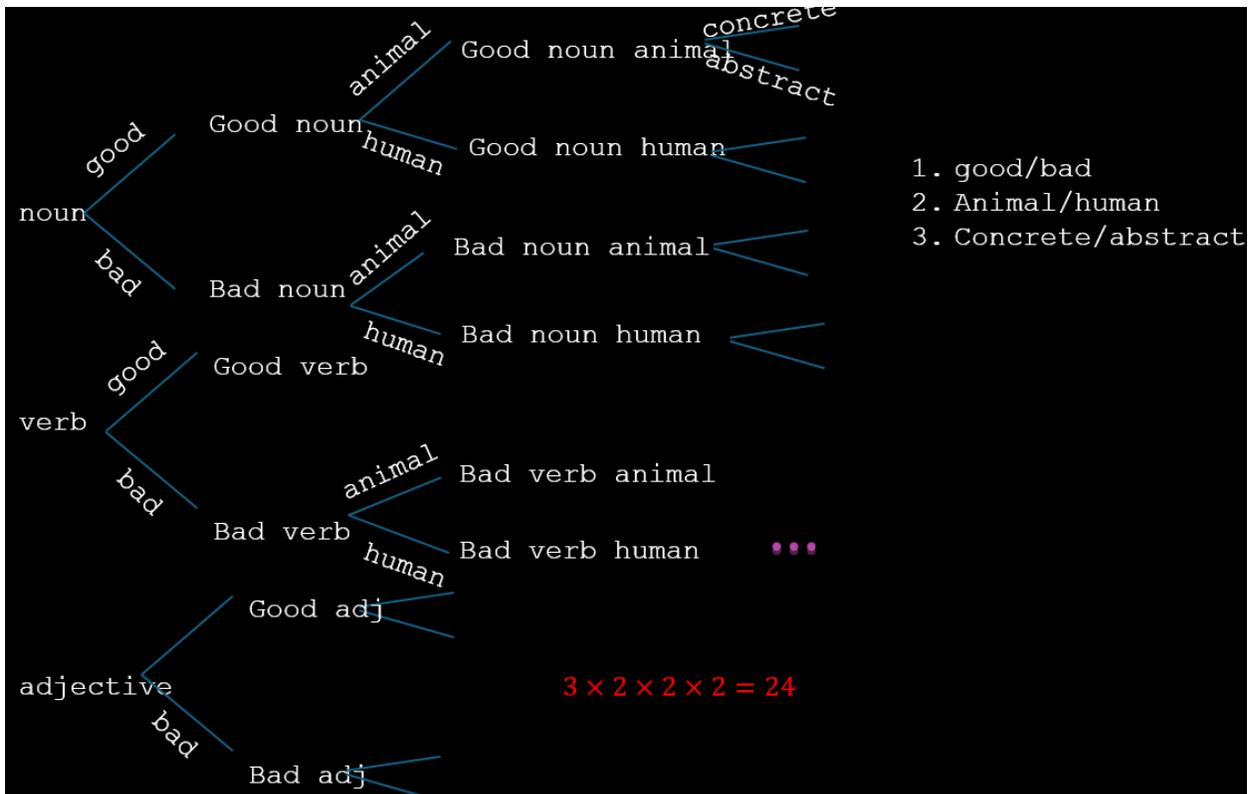
But then not all the nouns would match a verb and vice versa. We have so many 'flavors' for almost the same words. We have good and bad: good nouns, bad nouns, good verbs, bad verbs, good adjectives, bad adjectives. So a good noun should be paired with a good adj or verb and vice versa.

2025-11-22

Possible flavors:

1. Good/bad
2. Concrete/abstract
3. Animal/human

Then 3 word types, each time a flavor will branch it into two, 3 layers later, we will have 24 different 'sub-types':



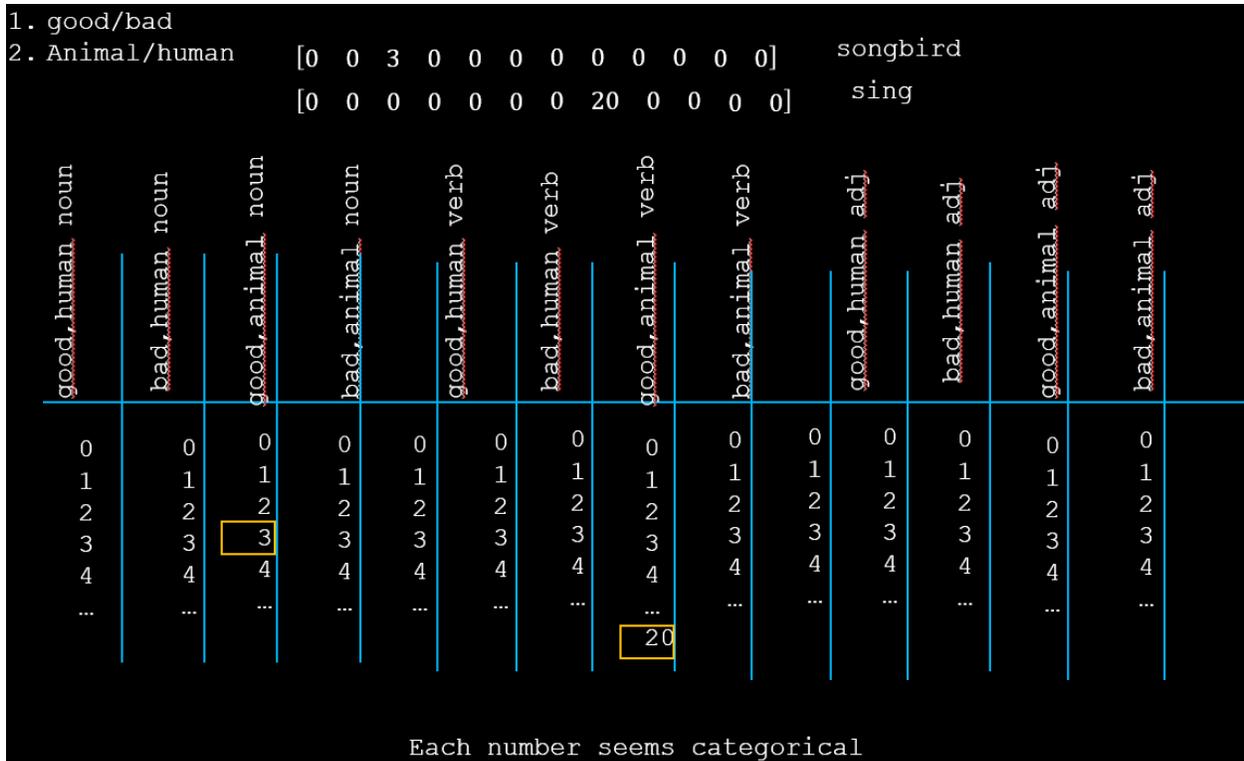
Then under each sub-type, we number the word accordingly. Say we only have two flavors, good/bad and animal/human, then we will have 12 dimensions, as in the following picture! 'songbird' is a good, animal-related noun, so it's under the third column, say '3' represents it. The verb 'sing' could be a good animal verb, so it's under column 8, say '20' represents it. This way, each word is a 12-dimension vector, and each number seems 'categorical', their magnitudes do not bear further meanings, they serve merely as an 'identifier'. Now think about type-matching. For songbird

[3]

Since it's a good animal noun, Its query should require a good animal verb, hence the query should look like

[1]

Whether it's 1 or 2 in there is not important, the key is to request something in the 8-th dimension, because that dimension represents a good animal verb. This way, a dot product can be carried out with the word 'sing' to do the match.



So this branching method makes the representation 'heavy', 2 flavors led to 12 dimensions, 3 led to 24 and so on. Also, the categorization is very strict, there is nothing between good and bad, too binary, of course to represent more nuances, more dimensions are needed. So it's not the best way to represent a word.

Another option is to have the flavors as 'components', to be combined with noun-ness, adjectiveness and verbness to 'form' a word.

Rather than have one dimension for good noun, one for bad noun, now good/bad, animal-human is like a little dial, that allow us to 'modify' a noun/verb/adj. for example, the difference between 'feed' and 'dine' is one is for animal another for human. They are both neutral word, so '0' for good/bad dimension, '-1' for animacy, not a noun, not a adj, so both 0 there, and under verb-ness dimension, say '5' represents to the action of eating, so pick 5 there. Therefore 'feed' is represented by [0 -1 0 5 0]. Then for 'dine', it's easier, the only difference is in the animal-human dimension, so should be 1 there, hence [0 1 0 5 0] represents 'dine'.

Similarly, frugal and stingy means the same thing except that one is positive another is negative. So for 'Stingy', the first dimension would be -1, 1 for human, 0 for not a noun, 0 for not a verb, then say '101' represents the property of being careful with money, so the stingy can be represented by [-1, 1, 0, 0, 101]. For frugal, we only need to flip the sentiment [1, 1, 0, 0, 101].

So I hope these examples can make us feel better in terms of meaning components. It is as if we need to decompose a word into different meaning components. Then of course what's the benefit of doing this? Dimension reduction is certainly one of them. Also it allows a continuous spectrum. When we hard label a word as good or bad, animal or human, we are making a binary decision, there are nothing in between. But for certain words, they are fluid, depending on the situation. This meaning components

manner allow a range of possibility. For example, I attach frugal to a hard '1' because it is a positive word, but it could be ranked slightly less than that, say, 0.94. The word 'sing' can be used by either human or birds, so not necessarily a 1 or -1 score. The word 'sponge' is definitely not a life, but in a spongebob cartoon, it takes that human/animal character. So there could be another dimension like 'topic', ie, science, fiction, cartoon, etc.

