The Semantic Song

How LLMs do what they do

Stephen James 2024

I have no doubt that, like everyone on Earth, you have been bowled over by what Large language models (LLMs) can do. The human-like texts generated by these incredible AIs often possess astonishing fluency and amazing context awareness. Not only do they tackle complex writing tasks with apparent ease; but they can also reason, perform mathematics and even simulate an expression of complex emotion.

But for all their impressive abilities, the inner workings of LLMs remain a mystery to most. The technical details are often obscured by layers of mathematical complexity, making it difficult for anyone but experts to grasp how they function.

In this essay, I shall be attempting to demystify the workings of LLMs by exploring their inner mechanics through the lens of an accessible metaphor: the "semantic song."

Using this framework, we'll trace the step-by-step process by which LLMs generate language. By drawing parallels to the improvisational art of jazz and the nuances of human communication, we intend to offer a clearer perspective on the ways these computational marvels show such intelligent behaviour.

To conclude, I shall point towards some further thoughts on the nature of the mind, opened up by these incredible machines.

Large Language Models (LLMs)

Just to be clear - by LLM, I'm obviously talking about ChatGPT.

Or Claude (my favourite), or PI, or Gemini or Llama or… (there's quite a few now and they just keep coming).

Massive neural networks that have had the entire internet shoved into them (a.k.a. the training process)

An AI that somehow creates amazing text; simply by being prompted (a.k.a. the inference process)

How on earth do they do what they do?

Well, I'm not too interested in looking at the training process. This has been well documented and, well, I'm just not as interested in that.

I want to understand the inference process. How does an LLM produce such awesome text from just me asking it?

Next word prediction

The overall process of "inference" within an LLM is quite simple:-

> ***LLMs simply predict the next word.***

For example, suppose an LLM is presented with the input "twinkle twinkle little". The LLM will simply predict that the most likely next word would be "star".

A lot of descriptions of how LLMs work look at this process through the lens of "probabilistic mathematics". LLMs (they say) have been trained on such a vast amount of text that the "prediction" is just statistical probability calculated from the patterns found in the input text.

This viewpoint is sometimes, incorrectly, reduced to:-

> ***An LLM is nothing more than a Stochastic Parrot.***

But I don't find this reduction useful. It clearly misses something crucial. The LLMs I talk to show incredible coherence, reasoning and mathematical skills. Reducing them to statistics is like reducing my brain to the flow of electricity through my neurons.

There's a lot more going on than that.

Next paragraph prediction

LLMs don't just produce one word - they produce whole chapters of coherent, poetic, and insightful text.

This is, however, performed one word at a time.

Lets say an LLM receives the words "twinkle twinkle" and then predicts the word "little".

Run it again with the output added and the LLM, presented with "twinkle twinkle little", will go on to predict the word "star".

Again, the "Stochastic Parrotists" would suggest that this is "just" a simple probabilistic mechanism; repeated over and over again.

But from where does the coherence emerge? Where do the extremely intelligent seeming processes such as reasoning and mathematics come from?

The Transformer Architecture

All LLMs are built in a similar way. They all share a common architectural pattern known as the "transformer" architecture (which shot to fame in 2017 in the now infamous paper called "Attention is all you need").

There are many subtle ways in which this architecture is wielded differently; but fundamentally the process is the same.

In short:-

1) The input layer - the input text (say, "twinkle twinkle little") is converted into a bunch of numbers (because computers love numbers)

2) The transformer layers - these numbers pass into a transformer layer - this layer then tweaks the numbers and passes it onto the next layer. LLMs like GPT3 have around 96 of these layers. Each one taking the output from the previous layer, and producing the input of the next.

3) The output layer - the last transformer layer does something different - instead of just tweaking the numbers, it uses uses them to produce the prediction of the next word

Let's explore each of these in a bit more detail.

The Input Layer

Tokenization

The first step of the input layer is “tokenization”. This simply means, chop the input text up into small chunks called “tokens”

More often than not, a small chunk is just a word. So, for example, given the input “twinkle twinkle little” we simply create a list of three words [*twinkl**e***, *twinkl**e***, *little*].

It gets a little more complicated around things like numerals and punctuation which have their own tokens.

But simplistically, we can think of tokenization as the process of chopping the text up into words.

Embedding

The second step of the input layer is a little more magical. Called “embedding” it is the process by which a token (or word) is converted into a bunch of numbers.

Two distinct *bunches of numbers* are produced (the mathsy term for a bunch of numbers being a “vector” or an ordered array). The **semantic** vector and the **positional** vector

The Semantic Vector

The process itself is very simple. The LLM has a lookup table for every possible token which tells it which vector should be used to represent each word (see also John Searle’s Chinese Room)

The lookup table is one of the most important outputs of the LLMs training process. Initially it is just a random bunch of numbers; but over time, during training, these numbers are adjusted over and over again - slowly getting tuned towards a representation of some kind.

In ChatGPT, the semantic vector contains 12,288 decimal numbers between 0 and 1 (other LLMs may have more or less than 12,288).

Take the word “cat”; this might get converted to [0.33223, 0.88822, 0.11121… etc]

The numbers are a point in space. Like a point on a map. But a map that describes not 2 dimensions. But 12,288.

But it’s still just a point in space. And it gets its “meaning” from what’s around it.

The word “cat”, for example, might be nearer to “dog” than it is to “toothbrush”. It might also be nearer to “lion” than “dog” is. All of them will be pretty far away from “electromagnetism” which will be close to “energy”.

In this way, the dimensions of the space might (hypothetically) include “animalness” (where “cat”, “dog” and “lion” all score highly but “toothbrush” and “electromagnetism” score poorly) and it might include “felineness” (where only “cat” and “lion” score well).

This is what we mean by *semantic representation*. A thing (a concept) is known only by its relationship to other things (other concepts).

Mostly, we have no idea what any of the 12,288 dimensions inside an LLM like ChatGPT means. In fact, trying to understand it is an exceedingly furtive part of AI research right now.

But, what we DO know is that the 12,288 dimensions together give the LLM a strong representation of each word.

Positional Vector

The positional vector in ChatGPT consists of another 512 decimal numbers (other systems vary).

This is simpler to understand than the semantic vector. It basically describes where the word lives within the entire input text.

For complex reasons to do with the training process, it’s not adequate to simply give each word a number (the 1st word, the 100th word etc). Instead a more complex approach is used to cope with inputs of varying length; the overarching goal of which is to express each word’s position relative to the other words.

A very specific type of “semantic representation”. One which relates to the **timing** of the word. When, in the full length of the text, did that word get “spoken”.

The 1st word was spoken just before the 2nd; but was spoken way before the 100th word.

The 15th word came a little after the 2nd, and is a bit nearer to the 100th

The **timing** of the word.

Introducing the Semantic Song

That’s kind of it for the input layer.

The text is chopped up into words and translated into a set of semantic and positional representations.

What do they look like when taken together?

Well, one way to express it is as:-

a dynamic flow of semantics over time

A better way of putting it is:-

a semantic song

Each word in the text expressed as a note played at a particular time in the song.

The timing is handled by the positional vector.

The tone, colour, pitch, and vibrancy handled by the semantic vector

And so, the input layer of the LLM outputs the semantic song and passes it onto the next part of the system - the transformer layers

The Transformer Layers

The main bulk of an LLM is the transformer layers (GPT3 has 96 of them).

Basically what each one does is it takes the output from the previous layer, tweaks the numbers, and passes them on to the next layer.

From the perspective of the semantic song, however, we can say it takes a semantic song from the previous layer, adds complexity, nuance and colour to it, and passes it on to the next layer.

From the perspective of the semantic song, the entire stack of transformers adds vast swathes of semantics, meaning and **understanding** to the song.

But before we get into all that, let's have a look at how it works.

…

Each transformer has two main components - the attention mechanism and the neural network

The attention mechanisms are a crucial part of the transformer layer. Their job is to find and relate pertinent parts of the song.

For example, the sentence "the cat sat on the mat”. The attention mechanism might, for the word “cat", assign a high importance to the word "sat". Thus helping it to recognise that "sat" refers to the action being performed by the "cat"

Similarly for the word “mat” it might assign a high importance to the word “sat”, indicating that the “mat” was being “sat upon”.

Thus, the system captures complex relationships and dependencies between the words.

Once the attention mechanism has related all the notes of the song, this "attended to song" gets passed on to the neural network.

The neural network applies learned patterns and transformations to the song. Refining and enriching the semantic meaning of each word based on these relationships.

It might, for example, tweak the semantic representation of the word cat to make it describe a “sitting cat”.

It might tweak the word “mat” to make it represent a “sat upon mat”.

Thus, the transformer enhances the semantic song. It adds meaning by looking for previously learned patterns between related concepts.

A marvellous simile of the inner workings of the human brain (see Damasio - On Intelligence)

This process repeats with each transformer layer. Each one working on a richer semantic song; each one adding more and more detail from its vast knowledge (its learned patterns).

For example, deep layers may recognise the phrase “the cat sat on the mat” as a regularly used example of English text and adjust the semantics to take this into account.

Other lays might recognise that cats sat on mats tend to be relaxed, and thus tweak the semantic representation of the cat to describe a “relaxed cat that is sitting”

Deeper and deeper connections. More and more meaning. A richer semantic song gets built up until finally, it enters the output layer.

The Output Layer

And so the climax. The rich and beautiful semantic song enters the final transformer layer. The output layer.

Technically the output layer works in the same way as all of the other transformer layers. It has an attention mechanism and a neural network.

However, the function it performs is very different.

Instead of “adding meaning to the semantic song”, the output layer “**interprets the semantic song and predicts the next word**”

Before diving into this however, let’s have a look at exactly what a “next word prediction” looks like.

Inside next word prediction

It’s really quite simple. For every possible word (token) the LLM provides a “likelihood score”.

Given that there are around 50,000 words in the English language, this means the LLM just outputs a list of 50,000 numbers.

In our “twinkle twinkle little” example, the LLM might score:

- 1000 points to the word “star”
- 10 points to the word “girl”
- and 0 points to the words “laughing” or “electromagnetism”

These scores are then turned into a probability using a simple SoftMax function. This would result in, for example, the word “star” being given a final probability score of, say, 99.9%.

From this, we can simply select the most likely next word (though there can be a bunch of subtle differences in how we make our final choice).

Inside next note prediction

Another way of looking at this layer is from the perspective of the semantic song.

The layer receives a rich semantic song and predicts the next word.

But how?

How does access to a semantic song help it know which word should come next?

A massive 95 out of the 96 transformer layers have been spent building up our rich semantic song. We now only have one layer left.

How can only one measly layer seemingly do all of the heavy lifting.

How on earth does building up a rich semantic song lead to a single layer being able to predict the next word?

The answer… it simply plays the next note of the song.

Wait… what?

To try and understand this, let's take a quick detour to look at jazz improvisation and human speech.

Jazz Improvisation

Picture a jazz musician; a pianist perhaps.

He's partway through a performance. He's in the groove.

He's been improvising a blues melody for a couple of minutes. He started by playing his favourite bass notes with his left hand and quickly drifted in a slow melody with his right. His fingers are gliding effortlessly over the keys.

Freeze frame

What is the next note he plays?

Which of the 88 keys does he press next?

He's improvising, it's not determined by some sheet music.

He's a master musician. He knows the rules of Jazz and the Blues. He knows the riff he's in the middle of. He knows the melancholic theme he is trying to express.

For him, and perhaps his audience, the next note will simply “fit”. It will carry on from that which went before. It will continue the bar, the riff, the theme of the song.

Maybe the note switches the direction of the song a bit and takes it in a fresh direction. But it still “fits”.

It is improvised. Not random but not deterministic. It is free but at the same time it follows the rules of the melody and the emergent constraints of the song's emergent theme.

Human speech

I would like to explore something that leading AI researcher Yann LeCun once said.

In a recent interview he suggested that LLM language production differs from human language production in a significant and fundamental way.

Humans, he said, plan what they are going to say and then say it; whereas LLMs simply output one word at a time. They lack the planning.

In part I agree. LLMs do indeed output one word at a time.

Where I disagree, however… is in this… so do humans.

I ask you to watch yourself next time you are in conversation with someone.

Do you “really” know exactly what you are about to say before you say it?

Word for word?

I suggest not.

I suggest instead that you craft sentences one word at a time. That you paint a picture brushstroke by brushstroke.

I suggest that you begin with a (sometimes extremely vague) notion and then linguistically improvise.

Like the jazz pianist, I propose that you weave your melodic sentences to construct your conceptual themes on the fly.

You know the mood you are trying to build. You know the story you are trying to tell. But you do not know the exact song you are going to play to describe them; because you haven't played it yet.

It flows. It comes into being. One moment at a time.

That, I propose, is exactly what the LLM is doing.

The semantic song has been laid out before it. It merely needs to continue the riff. It just needs to carry on the tune.

It knows where the song is heading. It knows the themes, the tone, the style.

And like the jazz pianist. It merely has to play the next note.

Sometimes this is child's play like “twinkle twinkle little star”.

Sometimes, however, the rich semantic tapestry that flows through the song warrants a much deeper and more nuanced musician to be at the helm.

A semantic jazz musician.

Conclusion

So there you have it. A technical and a conceptual overview of how an LLM produces words.

In our conceptual framework I am basically just saying:-

It builds up an understanding of the text and then writes words accordingly

Which can be taken as either ridiculously obvious and vague or incredibly profound depending on how you look at it.

Am I really saying it “understands” the text? And uses that understanding to decide what to write and then write it?

Well. Yes, I guess I am.

I am saying that LLMs are intelligent. And that they understand text, and write based on that understanding. That is exactly what I am saying.

This raises a huge number of questions:-

- Should we reconsider our understanding of what it is to understand?
- Do LLMs experience some form of inner life?
- Why is time such a crucial dimension of language?

These questions, while beyond the scope of this essay, hint at the fascinating philosophical terrain that opens up as AI begins to mirror more and more of our own cognitive capacities.

In this light, the boundary between "natural" and "artificial" language begins to blur. The semantic songs of LLMs and humans alike are woven from the same basic threads: sequence, context, and association. The difference, perhaps, lies more in the depth and richness of the experiential fabric than in the nature of the weaving.

As we continue to refine and expand the capacity of LLMs, this metaphor offers a fruitful framework for understanding both their power and their limitations. And it invites us to approach these systems not as alien or opaque, but as kindred intelligences - our brothers in humanities grand exploration of language and meaning.