Chat GPT, how does it work.

Definition and online example of the classical perceptron.

Let me begin with the perceptron. In displaying data on recent Microsoft products like Powerpoint, and more recently Power BI, one feature that is often used is the 'bar chart.' This can be used for displaying a single-variable real function if it happens to be, or is approximated by, a step function, where the value is locally constant except at finitely many points.

If we decided that we like step functions, that they are a convenient way of approximating a function with a finite amount of data (including rational approximations of numbers representing the exceptional points in the domain, and the values), we might look for a higher-dimensional analog.

At the same time, we might look for a smooth analog. An example would be to choose a linear functionals $V \to \mathbb{R}$ on a real vector-space V, and to compose with a clamping function $\mathbb{R} \to \mathbb{R}$ such as $arctan$, so the composite is smooth and bounded, and it approximates the step function which takes just two values, $\frac{\pi}{2}$ and $\frac{-\pi}{2}$. where we replace the arctan function with the function which sends negative numbers to $\frac{-\pi}{2}$ and positive numbers and zero to $\frac{\pi}{2}$. A linear combination of m such clamped funcionals as $arctan \circ f$ is exactly what is meant by a 'perceptron' with n input nodes, m 'hidden' nodes and one output node. The word 'hidden' just refers to the fact that in a linear combination of $arctan \circ f_i$ where f_i are functions the values of the individual summands are neither input nor output values of the overall function.

Here is an example which uses gradient descent to find such a 'smoothed step function' which approximates various functions you can write in https://spectrograph.uk/neural/nerual.html

A function $R^n \to R^k$ of this type can be made where each of the k coordinate function is such a linear combination and if we compose j such functions we obtain what is called a $(j+2)$-layer perceptron with j hidden layers. However, chatGPT is based on a different architecture.

For a simple GPT-1 translation machine, in place of using arctan as an 'activation function' one uses the function which acts as the identity for positive numbers, and sets negative numbers to zero. As a function on n-space, it is a projection on the first (generalized) octant.

One can visualize this as if working in a machine shop with just one machine, which projects onto the first octant in three-space. You have a finite set of points, which we visualize as a lump of clay. Using an affine transformation, you can bring the clay into the first octant, but with part of it extending outside the first octant. Depending on whether you are near a wall, or a corneer, etc, the part outside the octant is squashed, if it is near a wall, hammered flat, if it is near a corner, reduced to a line, etc. Then another affine transformation brings your lump of clay to a new position and the process is repeated.

The transformer described in [6] has as its encoding and decoding stage six such steps, but the first step is preceded by a position encoding, and each other step is preceded by an attention step. Also, once separated by attention steps, we include in each 'neural' step an affine transformation, projection to (generalized) octant, and another affine transformation. That was there implicitly if we had just composed steps, but when we'd composed two affine transformations we could view it as one. But we can think of the affine transformations as a storekeeper bringing an item from the reception desk into the workshop, to have the projection applied; afterwards he still has to bring it out to the reception desk for the translation step. If there were no customer watching over, one could compose the affine transformation of bringing it out to show the customer and back to the shop six times into just an affine transformation between two efforts of bringing the lump of clay to the octant, as it were, thinking 'the customer has gone home so there is no need to bring it out and back again.

Opening books

Now let's discuss something else, chess opening books.

There was a story that Magnus Carlson and another player played

20 or 30 moves quickly, then slowed down and took a long time to consider their next moves.

In those days, there was a piece of chess software called Chessbase, which players used to practise, and as you played through a game, it would give how many winning games went that way, for each next move. So a person could just choose the winningest next-move, historically, and work through the opening book, for quite a number of moves. And, although I did not check this, I imagined that some players essentially memorize the opening book.

As a very first approximation for how chatGPT would work, which is very different than the perceptron idea, you could look a the question, and the answer as far as it has been written up to now, as a sequence of moves, and ask, how many satisfactory answers were given that had each possible English word as the next word.

At the point where one gets 'out of book', where there was no such exact question and partial answer, it might be necessary to do something analogous to what in chess is called 'transposition' which would mean, allow onesself to replace the wording in the question or the answer up to now with a re-wording according to a simple transformation known not to change the meaning.

This too is not quite how chatGPT works.

Position evaluation

If our sequence of words had been represented, in the same order, as hot-encoded vectros $(1,0,0,...),(0,1,0,...)$ then the usual nilpotent matrix could be used to shift the sequence of vectors. But when they are just encoded as vectors one-by-one there is no information about the word order. In the transformer we're speaking of, for an n word sentence, we have n rows consisting of 512 numbers, and we can arrange these as a matrix with the row corresponding to the first word on the top etc. If we wanted a single step of the neural algorithm to use word-order information we might include extra coordiantes which hot-encode the position, but a single step (affine transformation, projection to a generalized octant, affine transformation) wouldn't 'know' how to take powers of the relevant nilpotent matrix.

Instead of thinking of the question as an actual sequence of words, one can use a position evaluator. A way to think of that is, imagine that you say one word per second, and the hands of a clock are going around. You have for each word, the x and y position of the second hand, minute hand, hour hand. If you look at what you are doing, you are replacing each impulse function at time t with a sum of exponentials e^{iut} for values of u along a geometric progression. The fact it is along a geometric progression rather than an arithmetic one is merely a matter of how we have coordinatized the variable u. In this way, if we consider say 3 values of u then each word has now 6 new coordinates (three x and three y) or three new complex coordinates. But when the dimension of our vector space is 512, there are 512 values of u involved. The way the perceptron can interpret these extra input coordinates allows information about word order to become accessible to the subsequent steps, which can give the transformer the appearance of understanding syntax.

Attention

The notion of 'attention' (see [3], [6]) begun with thinking, at the beginning the – let us imagine – row vector x representing a noun in a sentence might be modified according to surrounding adjectives. We apply a linear map Q yeilding the 'query' Qx in lower dimension d and apply another map K to all the words x_i in the sentence

to produce 'keys'. The correction to add to x is a linear combiation with the coefficient sequence of numbers between 0 and 1 being $softmax((xQ)(x_1K1)^t, (xQ)x_K2)^t, ...)$ or better $softmax(d^{-1/2}(xQ)(x_1K)^t, (xQ)(x_2K)^t, ...)$ and the vectors a matrix V applied to the matrix X with rows $y_1, y_2, ...$ Doing this for all $x = x_1, x_2, ...$ and adding is the same as adding to X the matrix $softmax(d^{-1/2}(xQ)(xK)^t)XV$. And the whole process may be repeated multiple times with Q replaced by a sequence of matrices $W_1^Q, W_2^Q...$ and likewise for K, V. This is called applying 'multiple attention heads'. And finally even in later stages of processing by ReLu neural stages when the matrix X no longer represents a sequence of words, the same matrix correction can be done.

In this file (which has no GUI, so you have to examine it in the developer console of a browser) I have put together here https://spectrograph.uk/transformer.html the scripts which perform one step of position evaluation, and then any specified number of steps attention (currently with just one attention head) and neural processing. If the neeural processing is taken to be the identity, m stages is quite similar to applying m attention heads to one stage, I wonder if it is an improvement. I wonder if the simpler architecture is better if one only has the courage to have many many stages but some with ReLu so sparse they cannot be trained. The issue might be, is it actually better not to correct for emotional timbre until after you have sorted the syntax of antecedents in English. The arrays of matrices $WQ, WK, WV, W1, b1, W2, b2$ and the base for the position encoding must be specified. Except for the base, these are to be determined by gradient descent, and it is easiest to analyze these algorithms with the chain rule in mind to describe the gradient descent algorithm, or, one can iteratively change entries of these arrays in whatever way improves the behaviour. Even though the 'activation function' is not differentiable on both sides, we just use the convention that $\frac{d}{dx}max(x, 0) = 1$ for $x > 0$ and 0 for $x <= 0$. The function called getMeaning returns an array of the same dimensions as the input array, we interpret the rows of the input array as vector representations of words in the language, with each row representing a successive word, so the input matrix is given by choosing a sentence, and the output matrix is what we could say is – to the best this is possible within the world of transformers – the 'meaning' of

the sentence.

To translate the meaning into a different language, we essentially reverse the steps, but similar to the opening book strategy, the output is merely meant to be the vector representing the next word in the translated sentence, and the input includes not only the meaning, but also an encoding of the part of the translated sentence so-far constructed, and the attention step of course only deals with part of this.

As a test of the code which we've written so far, we could work out the gradient descent by the chain rule, and try to train it just so the $(0, 0)$ entry of the meaning matrix tells us whether a given sentence is ironic or not, true or not, or whatever concept we want to test.

Riemannian metric on weight space

I have omitted putting into the script the 'layer normalization' described in [6]. At times during training or operation, it is apparently necessary to replace particular vectors with unit vectors.

One suspects that the description of a transformer could be simplified a lot, both in the softmax step of the attention step and in the layer normalizatio, if one recognized that when one sets up an error function, it is the assumption that one is using the Euclidean metric that allows one to even speak of the gradient vector-field of the error function. As one knows in manifold theory, a function does not have a specific gradient unless one has chosen a Riemannian metric. Here, the manifold is just the weight space which is smoothly equivalent to a Euclidean space, but one could I think simplify how these things are conceptualized by thinking of Euclidean space here as a Riemannian manifold.

The development from GPT-1 to GPT-3 or 'question and answer,' if we allow ourselves to over-simplify things, can be understood as just 'translating' a question like "Have you ever noticed that on TV each animal seems only to eat one food?" to an answer like "Yes! Especially in animated cartoons, for instance, there is the trope that mice eat cheese and dogs eat bones. It is important to be careful because in reality dogs need a balance of proteins, carbohydrates,

and vitamins not present if they were actually only to eat bones."

Unavoidable mistakes.

Now, here is why chatGPT is not a really revolutionary development in AI. Let's look at some mistakes it makes and why they are unavoidable. Earlier, I gave an example of trying to talk to chatgpt, and it seemed to make mistakes like thinking that among the squares of whole numbers between 2 and 10, you could shame it to think that 3^2 and $(-3)^2$ count as two of them.

Or, when asking if Falcon Heene could have travelled to the next town over in Colorado during the 80 days after the balloon boy hoax, it referred to the "available information" that he didn't travel during that whole time.

You can give a trivial interpretation to both mistakes – that it's a matter of convention whether we should count the answers before or after we square them, and it has to learn to be more accurate about what information is really available.

But, really, what went wrong in the second conversation is that we, as people, have a feeling for how long it takes to make one footstep, and how long it takes for the sun to go around, and we know that a footstep takes less than 80 days.

If 80 days was a lot shorter than the amount of time to take one step, we could go from saying he was stationary at the beginning of an interal of time, to saying he never reached the next town by the end of it.

And likewise we could adjust this for how long it takes to get into a taxicab and shut the door, etc.

Comparison with human writing

We can compare how chatgpt works, to the writing of, say, Hilary Mantel. Her books are each a series of vignettes, where she deals in concepts of basic biology, like how Shakespeare writes about blood, clouds, real things like meat, leaves, dirt. Like a head servant despairing that there are mouse droppings on the cutting board, and weevils in the flour, and the Archbishop needs to be fed, to rest, the whole kitchen will be modernized in the Italian mode of operation, now that the Archbishop's residence had been requisitioned by the king, and he had needed to choose a few articles of clothing to protect himself from the autumn weather, and it has been necessary to move to this inferior residence of a member of his congretation.

Compared to the possibilities any language model allows, we live in an infinitesimally thin slice of those possibilities. As Chomsky noted, children being able to learn a language with so little training must implly that the language instinct accrued what we could legitimately call an infinite degree of innate articulation.

No possibility to repair the limitations of AI

We could try to superimpose atop chatgpt – atop the so-called 'neural' training of the language model – a layer of restrictions coming from what biological possibilities really exist. Instead of allowing the actual mistake of thinking it's OK to mix up the beginning and end of an interval when recounting 'available information,' we could try to encode some actual facts.

The problem is, we don't know any facts, really. As Rumsfield said, in a tragically different context, there are the 'unknown unknowns.'

We know ABOUT them, yes, but in a different way than linguistically.

Why some experts condemn AI

The *impression* that it is revolutionary, that it will cause damage, etc, is along the lines of a labour union dispute. It was when weaving

was automated that actual Luddites united to try to smash factory machines, thinking that the weaving they could do should be things that only humans ought to be able to do. There is constantly an illuision that each most recent development in technology is a crucial 'last straw' (as Kasparov spoke when he said there can be no chess engine which could ever beat a person).

Why AI *should* be condemned

What causes damage includes AI, but is more basic, it is things that have already happened, the way shovels and tractors enable artificial agriculture, the way wheels mix cultures and clans in a damaging way, which destroys thought itself (see my blog above).

The outputs of chatGPT *seem* startling, seem to represent thought, but only if you have a very mechanical or unoriginal veiw of what thought entails. Confusing a technical trivialization of an aspect of humanity, or, more generally of life, and of nature, with the aspect itself is something that has happened over and over again, from building dams to create electrical energy without worrying about fish in the rivers, to synthesizing what we thought would be complete baby formulas. What is unavoidable is the confusion itself. It is unavoidable because the human mind, nor the animal mind, nor the human or animal biology, has ever had any resistance against a particular strain of possibities not been present during the evolution of life.

References

1. Eccles, the neurophysiological basis of mind, Clarendon (1953)
2. Neural networks and physical systems with emergent collective computational abilities, Proc Nat Acad Sci 79 (8) 1982
3. Lei Mao, Transformers explained in one single page 9 February 2023 (blog)
4. McCulloch, W; Pitts, W ,A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. 5 (4) (1943) .
5. Rosenblatt, Frank (1957). "The Perceptron—a perceiving and recognizing automaton". Report 85-460-1. Cornell Aeronautical Laboratory.
6. Attention Is All You Need Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin (preprint)