

Assays of Bias: LLMs and Geometries of Meaning*

Chris Forster

July 12, 2026

There are plenty of things to say about Large Language Models and the discourse around AI. I may not be able to resist contributing to that discourse. But this post (and probably some others to follow it up) is an attempt to step away from the Discourse, and just try to better understand this technology.

That choice itself can be reasonably critiqued. Can we afford to look away from the ways that these technologies exacerbate climate collapse and prop up hypercapitalist totalitarianism? Can one use a technology that is largely controlled by people who seem somewhere between comfortable and downright eager to placate the current US presidential administration? Besides, how interesting can a tool be that is undermining education and facilitating illiteracy and cheating? What could conceivably be learned from a technology that is little more than an act of mass intellectual property theft? A technology that is an insult to the very idea of creativity? To the very notion of *humanity*? Can it possibly be worth scrutinizing the details of the valve timings on an engine that is shredding the already-frayed fabric of collective life?¹

My modest, tentative answer is: yes, it is worth it to better understand these things. In part this is because I don't fully agree with all the critiques (perhaps unfairly) summarized above. One can be curious without being uncritical. And most fundamentally, what I think I don't share with AI skeptics, is utter shock that these machines work as well as they do. It is very hard for me not to remain surprised that a next-token predictor can do the things that it does. If you had told me ten years ago that a technology built by digesting every book and reddit post would be able to pass the Turing test, I would have been deeply skeptical—and it is my sense that most other people would have been too. It

*<http://cforster.com/2026/07/assays-of-bias>

¹I don't think all these critiques are equally reasonable though.

was simply not obvious that this technology could produce these results. The shape of the political moment and the massive investment in these technologies (and those are not uncorrelated forces) have produced strongly polarized responses, with accordingly stark utopian/dystopian visions. But I remain surprised, and curious, how software, trained this way, produces these outputs.

Below, I want to talk about a more specific instance of that surprise concerning how LLMs represent individual tokens. That realization can be summarized: the “meaning” of each token as represented in an LLM is unique to its context. The phrases “I saw a bear in the woods” and “a bear was running through the woods,” will use the same token (“bear”) with the same token ID. They may be describing the same particular bear. But the context reshapes the representation of that word so that the token “bear” will have a unique value, or meaning, in these two cases. Equating the value of some vector—some list of numbers—with the “meaning” of the term is tricky. Another way to say this: meanings, in an LLM, are not points, they’re regions. In one way, this is just repeating the premise of contextual word vectors. But it still struck me as odd. So I wanted to try to walk through it carefully. This post is my attempt to do so.

Another Attempt to Explain how LLMs Work

Understanding my point requires trying to understand how LLMs work. I’ve found different explanations of LLMs stress different elements of the machinery. This one will try to focus on how vector representations of tokens are reshaped. Recall, that the first step when an LLM processes text (I’ll talk about inference, and ignore training here) is to split the input into terms from the model’s vocabulary—to tokenize the input. After the text is tokenized, each token is replaced with an initial vector of some high dimensionality. For `gpt2-small` (the smallest version of the `gpt2` model, and the one I’ll be using as an example), those vectors have 768 numeric values. That means each token can now

be imagined as a point in 768-dimensional space. For larger open models, that dimensionality can be larger. The embedding vector dimensionality for the most current models from Google, OpenAI, Anthropic, and others is not publicly disclosed. Whatever its size, this initial “static” embedding is basically the same as the word embeddings of word2vec or GloVe. LLMs learn this embedding during model training. I think this initial embedding value can be loosely described as the average meaning of a token across the training data. After this vector look up, all tokens of *the same type* (i.e. having the same ID), will start with the same value. However, even at this first stage, positional information is added to the vector. This “positional information” is not just an index of the position itself, but another vector, of the same dimensionality as the embedding, which is, like the initial embedding, learned during training.

Consider the text input: “dog dog.” That looks like two instances of the token (dog).² Even in this first stage all tokens of the same type (ID) will probably not have the same vector value, because to a simple token value is added “positional information.” This though is the trivial way that token values are essentially unique.

The work of the LLM is to then reshape this vector based on context, so that the “best” possible next token can be predicted. To do so, all those vectors values (that have been looked up and altered by position vectors) are passed through a series of “layers.” A layer is simply a set of mathematical functions to reshape the vector values. gpt-2 has 12 layers. (Again, details like “number of layers” are not typically disclosed for the chatbots most folks think of as LLMs: ChatGPT, Claude, etc.) The neural network alters each vector based on all the preceding tokens in the context window. Knowing exactly which of those preceding tokens should influence any particular token, and how (how many, and which of those 768 numbers to change; and how to change them—increase or decrease), is of course non-trivial. It’s the whole (de-

²It could of course be the token “dog” and the token ” dog,” but we’ll ignore that.

cidedly black-box-y) mechanism. This is the “attention” mechanism at work: a system of keys, values, and queries that is the backbone of transformer-based LLMs. The most relevant terms in the context push or pull that initial, layer-0 vector value in different directions. And then you do it again... and again, for each layer in the model (along with some simpler “feed forward” layers in between). I struggle to have a meaningful, intuitive understanding of this process; and I suspect that most other people do too.³ But ultimately, however baroquely complex, this is a matter of lots of cross products passing through some non-linear functions, with the effect of reshaping each vector according to values learned from the training data. That training process (frustratingly called “pretraining”) has “learned” the particular values that will reshape the input vectors in ways that are meaningful. Or, perhaps more accurately, values that will reshape the input vectors to be most consistent with the training data. (Keep in mind: the task during training is accurately predicting the next token given the input.)⁴

So, text becomes tokens; tokens becomes vectors; and the vector values change based on the other vectors in the context window (or, more precisely, the *preceding* vectors). And all these values, in all these vectors, are changing in parallel. The “product” or “goal” of this process, in typical use, is to predict the *next* token after the *last* token in the context window. Doing so involves reshaping all the vectors, multiple times. The *autoregressive* character of LLMs means that all vectors representing the context window reshape themselves in light of all the preceding vectors. That this process could write even a bad poem or produce an inaccurate summary of the Krebs cycle still strikes me as deeply weird.

After all the tokens have been reshaped across all the layers, but before the next token is generated, we have a

³(I, like literally millions of others, have found [3Blue1Brown](https://www.youtube.com/c/3blue1brown) (<https://www.youtube.com/c/3blue1brown>)’s videos illuminating—on LLMs and much else besides.)

⁴This has led to the description of LLMs as “autocomplete on steroids,” a description that helpfully gives one an intuition for the main *goal* of LLM training, but may underplay the particular *means* of how they pursue this task.

reshaped set of vectors that all been changed based on the values/weights/parameters learned during training. How do we get from these values to the next token or word? We take our reshaped final vector and turn it back into a token by “unembedding” it. As the name suggests, this is essentially the reverse of the embedding stage. In gpt2, it is done by comparing our final 768-dimension vector to the initial vectors representing each token in the vocabulary. (Whether the “frontier” models use the same matrix for embedding and unembedding is another undisclosed detail.) This comparison is done with that staple of linear algebra, [the dot product](#)⁵. One dot product is calculated between the final vector and a vector representing each term in the vocabulary. (That is a lot of dot products—tens of thousands at least.) Dot products with higher numeric values mean greater similarity. (These numbers, to the frustration of neophytes and nonexperts, are called *logits*—despite having nothing to do with logistic regression or with the log-odds.) Using [softmax](#)⁶, this list of numbers can easily be turned into a probability distribution across the whole vocabulary of the model.⁷ In typical inference (when you’re using a chatbot like Claude or ChatGPT), there is complicated (and stochastic) logic to select a token from this distribution. In practice, always choosing the top “most likely” next token produces bad outputs. And so, there is a deliberate injection of randomness to what is otherwise a deterministic processes built on a probabilistic representation of the training data.⁸ If LLMs are stochastic parrots, they are stochastic in two ways: their training attempts to model the conditional probabilities of a distribution that is only very loosely sampled by even their enormous training data; but also, the text they

⁵<https://www.youtube.com/watch?v=LyGKycYT2v0>

⁶https://en.wikipedia.org/wiki/Softmax_function

⁷Or, you could use a setting like “temperature” to reshape this distribution, and so likelihood of tokens being sampled (about which, more below).

⁸Technically, this determinism can be lost just through matrix multiplication because of the way that floating point rounding works in digital computers.

generate from these distributions involves deliberately injecting randomness because, empirically, it makes the output “better.”⁹

However it is selected, the new token is added onto the input. If this selected token isn’t some special “end of response” token, the whole thing starts over, now with that one added token that will, in turn, reshapes the values that come after it in the window. And you do the whole thing again.

You might note that only the last token of the last layer is unembedded in order to predict the next token. So why did we do all those other calculations—for each token at each layer? This is the autogressive character of the LLM: in order to get that last token’s vector representation, we need all the others, and each of those need all the layers, all reshaped as if they were trying to predict the next token. But each token in the context window is being reshaped.¹⁰ So that is my own, overlong, explanation of LLMs.

Each Meaning is Utterly Unique

After struggling to understand this process, I was surprised to notice that the value of any contextualized vector is almost certainly

⁹This fact is really quite weird! Always picking the most likely most next token produces bad output. Human generated text, when measured, doesn’t tend to predict the most likely term. This suggests a tension between the training goal of LLMs and the reality of actually existing text. The following is the most interesting observation on this peculiarity that I’ve read: “Why is human-written text not the most probable text? We conjecture that this is an intrinsic property of human language. Language models that assign probabilities one word at a time without a global model of the text will have trouble capturing this effect. Grice’s Maxims of Communication (Grice, 1975) show that people optimize against stating the obvious. Thus, making every word as predictable as possible will be disfavored. This makes solving the problem simply by training larger models or improving neural architectures using standard per-word learning objectives unlikely: such models are forced to favor the lowest common denominator, rather than informative language” (Holtzman et al. 7).

¹⁰You could examine those tokens to try to inspect what is going on—this is the idea of the “[logit lens](https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens)” (https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens),” which emerged soon after gpt2 to try to understand what those other layers are doing.

unique. I don't know that this is counterintuitive exactly, because I have no intuitions about how meanings represented as points in high-dimensional space. The naive sense that the word "jello" is the same word in every sentence is belied by a system which rearranges its representation of "jello" in *every* appearance. Context reshapes meaning, so that the representation of any token within that message is unique to the message.

Should that be surprising? After all, every unique sentence is describing, or doing, something unique. Doesn't my point reduce to the rather obvious observation that *unique sentences are unique*? Maybe! I think my surprise stems from a naive sense of the discreteness of words. There is a tension between the abstraction that language requires by carving the world up into manipulable meaning-units like words, and any specific use of it. I am ready to concede that "unique sentences are unique." I was surprised to realize that in an LLM every unique sentence (or unique context) is made up of unique vectors.

An LLM enacts this tension between the abstract and the particular, between the token-type and the contextualized, vectorized token value, in a particular way. After the initial embeddings, the LLM simply *is* a series of vectors in a continuous high-dimensional space such that each token-representation is unique. To find two vectors that are *identical* in any position (*other than the first!*) seems impossibly unlikely. Because computers represent things in digital bits, the space is not truly continuous. But consider: each token position in gpt2 (a "small" "large language model") is represented by 768 32-bit precision floating point numbers. That means each of those 768 numbers can take on one of 2^{32} values. 768 numbers, each capable of 2^{32} unique values, means $2^{(32*768)}$, or $2^{24,576}$ unique combinations. That is so many possible values. So many, that if you, try to print the number, Python returns a `ValueError`.¹¹

¹¹But if you insist. Each vector takes on one of 1,297,697,410,448,600,346,538,135, 939,031,222,035,519,728,998,700,232,084,778,543,406,238,891,850,334,076,587,129,694,423, 527,303,789,473,987,996,649,748,036,250,082,317,297,862,768,096,218,198,670,173,239,042, 717,243,022,404,097,405,358,815,383,304,467,520,506,016,644,412,538,620,827,664,539,590,

981,133,764,939,559,889,792,881,228,022,688,086,387,013,159,779,474,659,377,228,904,660,
169,451,046,977,211,569,734,880,854,126,915,463,239,386,217,836,496,696,394,119,476,173,
179,833,969,555,344,970,609,797,275,119,509,074,848,382,649,401,987,542,136,420,071,348,
640,341,362,674,535,922,153,064,214,928,764,822,074,773,723,105,483,968,708,166,268,391,
770,001,008,193,546,523,866,907,057,414,757,404,671,950,237,688,337,348,889,336,470,255,
095,022,937,678,634,206,826,776,953,401,707,839,620,349,125,443,294,562,166,015,762,041,
130,046,137,078,304,507,285,578,022,169,570,112,201,489,325,577,731,123,168,662,277,782,
774,175,060,360,171,925,158,129,387,447,235,460,016,960,647,395,805,695,215,807,380,983,
699,032,255,008,369,041,788,431,642,510,192,930,958,332,314,819,211,877,161,067,761,083,
492,745,514,752,985,497,572,005,591,098,300,898,501,362,642,140,331,600,815,207,942,822,
100,023,617,638,152,801,392,095,870,047,020,586,430,664,761,145,703,707,938,535,906,464,
406,225,255,478,970,566,082,524,079,758,186,098,468,463,624,073,204,527,508,260,284,048,
962,987,191,501,560,528,120,556,297,523,750,493,275,728,157,552,504,092,216,062,654,422,
454,903,614,664,854,707,523,432,470,828,142,420,198,725,979,347,441,441,153,726,933,242,
354,091,395,389,601,490,357,550,432,526,686,664,921,480,152,112,368,216,565,882,267,653,
305,092,881,044,402,210,015,692,351,775,891,368,301,146,414,351,981,968,923,350,373,884,
448,126,606,724,415,090,942,918,312,718,285,342,778,976,990,207,729,827,271,369,182,128,
651,357,747,021,196,541,162,375,775,864,442,262,272,153,790,454,445,842,771,953,191,553,
371,532,285,370,457,313,533,801,325,474,998,460,876,129,865,738,698,744,711,785,842,767,
536,060,631,433,138,044,176,622,498,313,429,867,186,433,040,976,399,943,263,724,380,738,
286,727,261,473,007,118,657,345,412,394,415,247,329,923,139,297,422,468,472,074,973,105,
376,268,973,641,269,265,766,763,550,078,055,034,204,797,006,653,822,418,032,652,646,514,
526,001,315,721,600,841,383,643,905,592,770,018,559,411,242,071,334,524,097,130,309,230,
341,753,863,086,433,457,304,332,201,258,143,789,693,310,794,050,368,139,526,486,425,810,
515,920,526,690,514,966,426,657,735,297,009,822,822,347,308,821,059,461,601,928,047,577,
404,088,595,982,353,448,715,973,318,764,903,612,555,984,878,557,394,689,727,581,475,898,
934,959,903,510,650,539,523,742,387,358,493,028,687,147,999,727,868,782,905,595,440,800,
670,060,908,385,225,123,633,807,300,055,026,747,539,139,596,139,699,572,013,416,469,479,
147,050,202,026,035,823,595,658,241,564,260,446,273,117,491,733,149,544,139,183,746,349,
729,775,131,574,234,475,326,952,793,240,708,302,248,933,142,582,357,669,936,354,521,381,
623,946,955,024,586,401,006,691,321,883,101,748,742,366,234,089,670,835,411,314,692,531,
612,234,047,518,081,809,098,799,701,912,741,535,499,566,760,826,901,138,816,147,745,050,
571,154,414,816,928,387,311,642,876,344,456,295,168,525,833,275,257,389,289,089,002,855,
575,634,203,975,416,381,552,717,491,969,566,299,428,479,451,557,318,976,731,232,361,052,
347,688,093,782,450,759,172,496,613,457,785,426,674,586,796,341,558,349,801,020,810,439,
696,683,918,494,133,116,675,377,012,157,259,822,338,829,144,906,520,875,137,659,210,690,
851,412,992,747,336,492,326,914,326,688,736,069,877,775,771,817,439,628,583,041,620,269,
827,979,325,221,988,483,150,742,648,764,196,519,796,064,219,048,218,581,790,238,796,107,
433,591,494,903,767,305,623,499,779,161,623,829,491,394,153,438,577,602,753,296,015,872,
912,403,939,133,781,995,669,305,118,750,402,512,528,595,977,334,351,459,763,436,583,313,
632,905,069,902,953,475,008,068,913,625,365,490,282,617,160,080,393,262,801,172,330,313,
146,148,236,745,449,921,164,948,060,806,911,452,069,815,451,549,857,673,708,623,443,802,
874,549,340,502,789,744,414,566,660,268,553,543,566,910,755,632,036,051,772,333,652,210,
319,136,753,402,028,349,537,857,280,475,205,705,938,845,645,749,930,586,645,051,732,086,

272,786,583,983,061,014,900,742,914,501,299,157,138,806,769,790,024,767,722,750,633,585,
764,520,182,070,926,356,100,664,138,850,210,715,705,598,842,122,905,532,688,122,285,662,
818,997,869,709,358,741,389,382,818,551,074,714,596,490,081,150,107,374,227,933,685,329,
986,845,487,543,930,253,675,616,739,882,062,385,836,491,209,598,753,928,114,457,207,376,
576,356,255,107,028,642,034,175,437,122,941,195,648,806,083,085,549,982,368,852,278,066,
886,136,087,399,156,800,246,270,140,823,155,995,306,361,156,851,746,204,728,297,233,434,
496,857,940,379,371,424,957,968,305,685,415,799,097,530,416,332,358,418,536,606,933,116,
174,315,215,327,097,663,282,370,339,032,753,068,632,038,435,102,853,566,876,821,617,976,
116,764,083,041,289,405,510,235,702,138,673,206,084,233,412,415,977,601,761,552,494,634,
402,751,172,069,930,297,017,193,063,865,055,410,231,671,763,544,260,826,996,010,519,413,
376,980,317,677,626,223,193,586,838,527,728,173,723,646,908,915,741,191,050,522,407,467,
628,834,478,108,728,633,562,456,189,294,949,972,907,072,217,835,516,032,768,065,884,130,
726,042,757,123,496,466,996,641,424,573,962,320,966,764,049,957,703,899,302,070,352,183,
284,617,821,442,035,886,042,565,424,815,190,185,607,850,336,275,339,789,767,440,541,239,
244,969,508,140,362,837,460,506,154,194,733,006,766,882,061,115,814,920,468,876,280,572,
493,497,117,209,685,427,909,368,149,860,795,633,997,946,052,586,776,078,940,905,667,705,
642,198,211,724,479,972,324,896,764,242,258,402,262,954,862,503,531,240,323,448,374,982,
115,808,740,282,888,904,588,259,700,595,206,248,104,595,240,127,900,801,503,490,967,825,
067,529,653,117,701,159,787,653,102,842,260,359,451,835,029,603,941,483,333,188,517,120,
844,494,235,705,765,414,044,718,522,420,703,999,784,968,967,892,709,894,636,845,967,827,
768,197,323,409,343,104,494,220,034,948,026,747,051,910,930,855,371,819,909,486,828,996,
182,563,646,818,626,752,659,938,545,654,081,089,955,366,580,546,278,471,380,839,211,695,
493,074,974,418,621,269,683,720,940,027,491,064,776,543,824,006,365,637,843,286,379,190,
389,074,049,228,943,087,667,090,419,445,854,074,062,458,815,715,066,335,645,273,655,551,
762,071,864,833,938,653,812,714,755,489,303,912,189,289,249,791,427,033,572,312,255,750,
132,692,158,245,570,011,930,244,438,550,785,983,827,405,238,706,612,842,402,776,718,110,
713,673,430,107,593,004,180,336,747,715,384,598,315,488,260,995,655,046,945,837,985,356,
787,156,536,989,126,867,235,288,282,769,656,063,048,530,249,588,946,471,234,472,131,973,
601,315,870,490,469,871,433,260,386,093,197,064,710,657,885,805,829,743,893,244,275,588,
028,819,074,466,906,030,714,862,761,545,813,522,171,421,020,133,884,168,241,983,512,478,
748,442,083,201,937,260,172,820,273,409,005,526,650,193,325,470,249,073,091,805,786,606,
060,858,378,895,109,345,350,695,626,788,146,217,788,690,663,147,695,114,726,616,794,137,
652,420,096,127,458,684,110,437,426,367,896,052,974,371,764,756,234,020,770,269,534,434,
111,728,507,595,798,614,972,033,157,716,166,985,275,365,617,494,263,241,070,561,088,405,
182,564,846,615,164,214,637,734,671,747,042,841,816,392,654,568,855,171,198,670,204,310,
647,596,770,041,328,176,272,480,084,582,031,716,294,514,384,653,096,571,740,071,416,236,
945,808,538,413,410,300,289,298,927,252,395,528,565,432,762,746,902,299,537,728,883,871,
891,888,920,180,175,626,831,912,680,526,523,939,831,682,287,344,731,935,250,988,843,132,
573,781,950,303,895,403,679,727,132,795,397,488,289,174,164,965,209,702,135,337,312,021,
434,334,312,951,732,630,812,726,685,025,763,392,191,097,985,766,249,699,687,119,504,279,
835,501,519,449,709,324,409,415,408,073,124,865,462,545,188,189,851,584,962,820,403,391,
548,396,601,373,059,528,432,677,446,640,585,197,923,376,516,971,120,013,543,334,131,938,
417,888,061,074,117,612,732,303,756,778,467,970,163,777,923,331,639,234,305,215,709,299,
572,337,761,757,035,510,153,856,294,832,610,278,918,707,689,049,035,382,129,356,527,046,

My point is simply that the in an LLM is no longer what I naively assume “a word” is. To say, as LLM explainers often do, that in an LLM each word is replaced with a set of numbers is not untrue. But the description is too coarse to be useful. More useful is thinking about how words are first mapped in, and then back out, of the model’s high-dimensional space. In this model words and meanings are less discrete points than neighborhoods in the high-dimensional space that is constructed by the LLM and its training. Even though all the tokens will have the same token ID, and will begin with the same initial embedding (even though modified by their position in the context window)—everything else will be different. Those vectors will be moved around, so when the is output, it is because the predicted vector has ended up in some region or neighborhood of high-dimensional space, near to other thes, such that it unembeds as the. A vector becomes a word in an LLM not because of some essential quality, but simply because that token-representation is the closest one based on a dot product. This idea, that meaning is ultimately simply a matter of proximity (of near and far) is just very strange.

941,595,295,240,080,735,531,452,932,093,120,954,708,385,449,412,443,070,924,916,585,818, 586,680,574,886,184,182,869,327,998,509,170,260,455,635,446,673,449,213,667,468,574,497, 166,352,796,712,515,091,327,645,628,750,241,110,507,759,853,113,479,841,485,935,309,248, 994,530,027,404,986,592,273,392,579,816,513,233,358,532,974,083,317,237,346,360,753,309, 422,925,190,491,271,424,500,891,419,608,328,395,756,832,810,762,266,683,352,128,675,634, 710,527,614,321,150,264,678,943,631,196,036,179,171,716,403,147,146,831,648,828,350,037, 974,238,822,550,903,062,247,261,364,597,892,072,925,954,092,413,870,637,316,678,081,963, 842,074,813,957,351,386,937,129,800,909,906,306,557,789,158,351,467,739,260,857,109,711, 439,017,680,124,989,017,079,743,044,145,771,243,475,158,059,147,388,018,714,931,793,406, 684,348,047,066,043,445,686,177,403,209,935,784,881,349,886,420,722,418,431,870,954,052, 668,401,631,196,236,306,698,392,457,910,980,406,842,887,671,834,754,179,287,753,604,831, 182,219,354,347,006,159,916,298,311,506,700,806,419,133,922,129,045,611,604,789,404,880, 962,986,447,269,926,587,853,558,493,111,142,546,189,040,791,544,888,879,533,624,391,117, 247,722,961,529,184,560,333,182,143,385,935,164,557,921,429,635,788,460,812,145,451,792, 228,269,895,641,717,313,447,855,596,174,724,491,244,793,968,842,948,355,957,576,317,391, 057,576,885,263,334,995,728,821,917,585,989,247,726,539,952,718,769,475,840,182,868,192, 342,315,042,817,482,680,980,989,496,896,501,755,261,761,502,450,606,171,506,098,974,822, 921,493,874,400,207,047,960,353,579,365,573,551,102,433,938,746,563,463,036,728,374,276, 870,017,575,273,546,411,534,018,816,856,133,958,796,375,525,337,544,207,431,860,691,381, 011,828,338,496,128,914,496,884,503,314,383,707,471,558,515,557,061,617,255,675,934,959, 405,747,919,509,467,910,252,801,546,835,131,737,202,162,139,136 possible values.

Consider this very basic illustration. I fed each of the following sentences into gpt2 independently (i.e. the model only ever saw one line from this passage at a time).

I went to the bank.

Joe went to the bank.

I went to the store.

Joe went to the store.

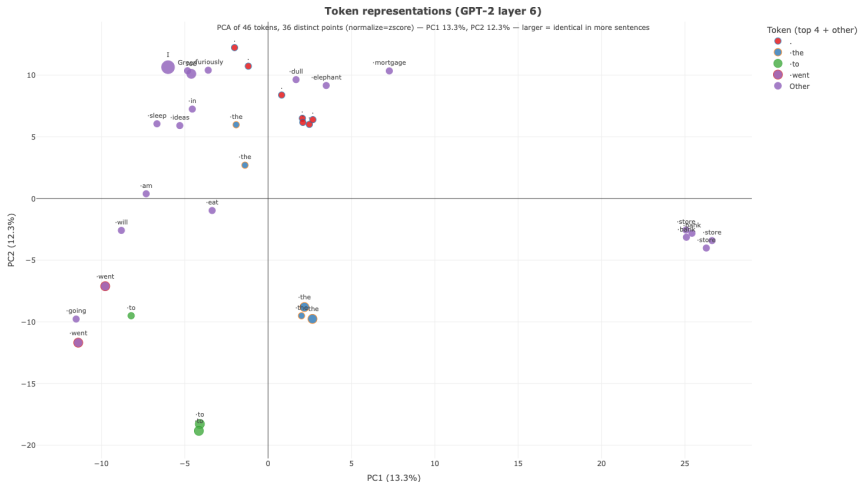
I am going to the store.

I will eat the elephant.

Green ideas sleep furiously in the dull mortgage.

Each of these sentences was independently tokenized, replaced with a set of 768-dimension vectors (one per token), and then processed, layer by layer. At each layer, the weights that the model learned during training dictate how each of these vectors modify each of the other in each layer. Now imagine stopping the machine, opening it up mid-processing, and looking at those lists of numbers. This sort of inspection has been done in a more rigorous way by serious researchers. But I grabbed a layer in the middle (layer 6, of this 12 layer network). From this layer I get one vector per token for each sentence. I then used Principal Component Analysis (which I think of as the most boring, straightforward method of dimensionality reduction) to reduce the 768-dimensional vectors for each term to 2 dimensions and created the plot below.¹² While the typical use of gpt2 would be to predict the next token, I haven't retrieved that. For this illustration, it doesn't matter.

¹²There was an additional element of normalization because otherwise GPT2's layers behave oddly. I only learned that in the course of trying to produce this plot.



Each point in the plot above is represents a token from one of the above listed sentences (hover over to see the sentence, or sentences, from which it was drawn). I’ve color coded the most frequent words here, to show how in the LLM even the “same” token is mapped to different places within the model’s space. Consider the word/token **the** above, from the sentences “I went to **the** store” and “Joe went to **the** store.” These two sentences feature the same token, in the same position, with a meaning that is as similar as I could imagine. But the vectors representing *the* in these two sentences are still separated a little. On this 2-dimensional reduction, they’re close. Most instances of the are in the lower right of this plot, representing sentences like “I am going to the store” and “Joe went to the bank.” But some end up in the top left. The the of “I will eat the elephant” is closer to ideas and sleep than the the of “I went to the bank.” I often see plots like these stress the proximity—the nearness of similar terms. We normally plot a bunch of terms and note how semantically similar terms appear *near* each other, but *far* from dissimilar terms. Just as important as the nearness of similar terms is the distance a model creates between similar, and even “identical,” tokens. Note, though, that the values for the in “I went to the store” and “I went to the bank” will be identical. That’s because all the tokens and positions to the left are identical, the “context”

is identical. (The plot represents identical vector representations with a larger point.) This again is because of “predict the next token” task. When you’re predicting the next token, then “I went to the bank” and “I went to the store” are identical right up until the last token (bank/store) (in a “bidirectional” model, like BERT, this would not be the case.)

This is a toy example. gpt2 is a very small model. Reducing 768 dimensions to 2, for purposes of plotting is obviously a lossy and potentially misleading process.¹³ I wouldn’t “read” too much into this plot. It captures a complexity in thinking of meaning as *space*. If meaning is space, words are not simply points, but regions. And if that’s the case, the sort of questions we might ask change. Are different regions larger? Is shape meaningful? Semantic space can be described not simply in terms of near and far, but with a richer vocabulary.

The Well-Wrought Vector: Uniqueness and Precision

Reflecting on this way of imagining meaning, I was reminded of a passage from Cleanth Brooks’s *The Well Wrought Urn* (1947). For Brooks, poetry is a mode of language where each meaning becomes as precise and utterly individual as a contextualized vector. Yet, how this uniqueness works is rather different. Language, for Brooks, tends to drift into abstraction and imprecision. It is a flawed tool. But this drift, in poetry, can be beaten back and precision achieved. In trying to describe this model of language, and poetry’s place in it, Brooks relies on images of meaning as space. Discussing Wordsworth’s “[It is a beauteous evening, calm and free](#)”¹⁴, Brooks writes:

T. S. Eliot has commented upon “that perpetual slight alteration of language, words perpetually juxtaposed in new and sudden combinations,” which occurs in poetry. It is perpetual; it cannot be kept out of the poem;

¹³These components represent only ~25% of the total variance, which is surprisingly good given that we started with 768 dimensions!

¹⁴<https://babel.hathitrust.org/cgi/pt?id=dul1.ark:/13960/t4hm6jw9n&seq=135>

it can only be directed and controlled. The tendency of science is necessarily to stabilize terms, to freeze them into strict denotations; the poet's tendency is by contrast disruptive. The terms are continually modifying each other, and thus violating their dictionary meanings. To take a very simple example, consider the adjectives in the first lines of Wordsworth's evening sonnet: *beauteous, calm, free, holy, quiet, breathless*. The juxtapositions are hardly startling; and yet notice this: the evening is like a nun breathless with adoration. The adjective "breathless" suggests tremendous excitement; and yet the evening is not only quiet but *calm*. There is no final contradiction, to be sure; it is *that* kind of calm and *that* kind of excitement, and the two states may well occur together. But the poet has no one term. Even if he had a polysyllabic technical term, the term would not provide the solution for his problem. He must work by contradiction and qualification.

We may approach the problem in this way: the poet has to work by analogies. All of the subtler states of emotion, as I. A. Richards has pointed out, necessarily demand metaphor for their expression. The poet must work by analogies, but the metaphors do not lie in the same plane or fit neatly edge to edge. There is a continual tilting of the planes; necessary overlapings, discrepancies, contradictions. Even the most direct and simple poet is forced into paradoxes far more often than we think, if we are sufficiently alive to what he is doing. (9–10)

Brooks's description helps me understand my own surprise at this element of contextual embeddings' behavior. "The terms are continually modifying each other, and thus violating their dictionary meanings." This contrast between the model of meaning as space and the dictionary is clarifying. The dictionary is or-

ganized by words. It imagines a meaning as an instance of some general type. Contextual embeddings start with a dictionary-like look up; but the whole process that makes them *contextual* embeddings involves modifying and altering them in a series of ways such that the vector representing a token may no longer most resemble the token it began as. We turn to the dictionary believing that we can understand a word in isolation. But the fantasy of the self-contained integrity of a word, embodied by the dictionary, is violated by the power of a context, which “continually modifies” the individual word. Science (Brooks suggests) attempts to solidify the mushiness of language, freezing it into a stricter denotation. It intensifies the logic of the dictionary, adding more and more words. It introduces new terms, and uses old terms more strictly. Poetry achieves a greater precision. This is a deliberately audacious claim—outperforming science on its own territory. Poetry does accept the imprecision language, but overcomes it through paradox. By bracing one imprecise meaning against another. Poetry navigates meaning-space more precisely not by trying to finely indicate a particular *point*, but by understanding precise meaning as an *intersection* in meaning space.



¹⁵<https://pdimagearchive.org/images/648fde18-34f5-4352-87f8-8b1b044c256d/>

Brooks's view of a meaning is LLM-like insofar as he imagines words as imprecise regions, representable as spatial, geometric forces. A desired meaning and its expression don't line up; you use a metaphor, but that comparison includes too much or not enough—the meanings “not lie in the same plane or fit neatly edge to edge.” Meaning is “a continual tilting of planes”—a nice enough image for meaning as space. Brooks further elaborates this geometric model of meaning, comparing the poet's work to a quotation from *Hamlet* (Polonius's advice on successful spying).

with assays of bias

By indirections find directions out. (*Hamlet* II.1)

Brooks explains:

Shakespeare had in mind the game of lawnbowls in which the bowl is distorted, a distortion which allows the skillful player to bowl a curve. To elaborate the figure, science makes use of the perfect sphere and its attack can be direct. The method of art can, I believe, never be direct—is always indirect. But that does not mean that the master of the game cannot place the bowl where he wants it. (10)

Science seeks to reform and rationalize meaning space, to straighten it out, to allow precise statement. The poet instead takes advantage of this curvature through paradox.

LLMs are not poets. They might, as Leif Weatherby suggests in *Language Machines* (which is undoubtedly the best, and most sophisticated work, on the intersection of structuralist theory and LLMs), represent something like Roman Jakobson's “general poetic function” (esp. chapter 5). But I want to end by observing a difference between LLMs and Brooks's description. They share a geometric understanding of meaning, and focus on individual meanings a unique points within this geometric space. Yet, the fact that each message/context is singularly *unique* is not the same as Brooks's assertion that the poet's meanings are

singularly *precise*.¹⁶ At the crucial moment, Brooks must reach outside language, through a deictic gesture. To follow the reading I've quoted above, we must agree with him that the paradox of the feeling Wordsworth describes exists. We must recognize “*that kind of calm and that kind of excitement.*” Brooks relies on this ostensive, deictic appeal to link the uniqueness of the poetic representation back to the world. And it is this link that allows the uniqueness to indicate not a point in semantic space, but a point in the world. For Brooks, the uniqueness of the representation allows it to overcome the imprecision of language and name something exact in the world. In an LLM, of course, uniqueness reflects nothing except consistency with training data.

Discosure: All the prose here was, for better or worse, written by me. The code I used to generate the plot above was generated by Claude Code.

Works Cited

- Brooks, Cleanth. *The Well Wrought Urn: Studies in the Structure of Poetry*. Harcourt, Brace & World, 1947.
- Holtzman, Ari, et al. *The Curious Case of Neural Text Degeneration*. 14 Feb. 2020, <https://doi.org/10.48550/arXiv.1904.09751>.
- Weatherby, Leif. *Language Machines: Cultural AI and the End of Remainder Humanism*. University of Minnesota Press, 2025.

¹⁶(Crucially Brooks completely separates poetry—and perhaps all of “literature”—from the rest of language and culture. The rejection of this segregation is probably the central fact of academic literary criticism since the late 1960s.)