# How do the Embedding Layers of LLMs change with scale?

By Joseph Corey

## Introduction

LLMs have the curious distinction of being near the top of two seemingly opposing lists, “Most Complex Human Creations”, and ”Least Well Understood Human Creations”. Despite having architectures designed by humans, being trained on primarily human output (read: the internet), and having every bit (literally) exposed to human view, LLMs are still effectively black boxes when it comes to understanding why they do what they do.

Digging into understanding how LLMs and other neural networks work (broadly called “interpretability”) is a key field in Machine Learning research right now.

This analysis aims to address one of the less well understood pieces of LLMs, the embedding layer. Specifically how the token embedding layers in LLMs change as model size and embedding size increases.

The main conclusions from this analysis are that embedding quality increases with size to a point, but then stagnates or decreases in quality after a certain model size (though this size differs by model suite). This is interesting because it implies either an underutilization/undertraining of embeddings in large models, or a diminishing importance of embeddings as a model size increases.

The code used for this analysis can be found at <https://github.com/jstephencorey/LMEmbeddingAnalysis>.

## Embedding Layers in Language Models

This blog post aims to answer the question “As LLM models get larger, how does the level of information in their embedding layers change?”

As the foundational layer of the model (after the tokenizer), the embedding layer's quality seems a vitally important step in the language processing of the model. It transforms text tokens per a predefined vocabulary, into a fixed-size, meaningful vector in the embedding space for the neural network's processing, hopefully capturing the tokens' semantic and syntactic properties for later processing by the model

However, as the embedding layer scales with the overall model size, its role and optimization become more complex, as I find in this analysis.

Intuition suggests that bigger and more capable models would naturally harbor more sophisticated embeddings. After all, larger models consistently outperform their smaller counterparts, a fact that should logically extend to their token embeddings.

In a larger model, each embedding has more room to hold meaning as a result of more embedding dimensions, and the model has the capability to capture and use more nuance in all of its weights. However, this isn’t entirely consistent with what is found in this analysis. Not all of the meaning and capability growth in the model as a whole seems to be captured in the embedding layer of the model.

## Methodology and Analysis

This analysis looks at 5 model suites: Pythia (14m-12b) ([Huggingface](https://huggingface.co/collections/EleutherAI/pythia-scaling-suite-64fb5dfa8c21ebb3db7ad2e1), [Paper](https://arxiv.org/abs/2304.01373)), OPT (125m-175b) ([Huggingface](https://huggingface.co/facebook/opt-125m), [Paper](https://arxiv.org/abs/2205.01068)), Cerebras (111m-13b) ([Huggingface](https://huggingface.co/cerebras), [Paper](https://arxiv.org/abs/2304.03208)), BLOOM (560m-176b) ([Huggingface](https://huggingface.co/bigscience/bloom), [Paper](https://arxiv.org/abs/2211.05100)), and Google’s T5 v1.1 (60m-11b) ([Huggingface](https://huggingface.co/collections/google/t5-release-65005e7c520f8d7b4d037918), [Paper](https://arxiv.org/abs/1910.10683)). These aren't necessarily the most modern or powerful models, but they are suites with large ranges of sizes, which can be used for examining scaling on embedding quality. The embedding dimensionality sizes of these models ranges from 128 to 14336 dimensions, and with total model sizings from 14m to 176b parameters. T5 is added for an example of embeddings in a non decoder-only architecture.

It is hard to perfectly disentangle embedding size from model size, given that little effort has gone into making a tiny model with a huge embedding size, or a huge model with a tiny embedding size. By isolating some factors (cropping and padding embeddings), evaluating random baselines, and looking particularly at pythia-1b and pythia-1.4b, the effects of various variables on embedding quality can begin to be understood.

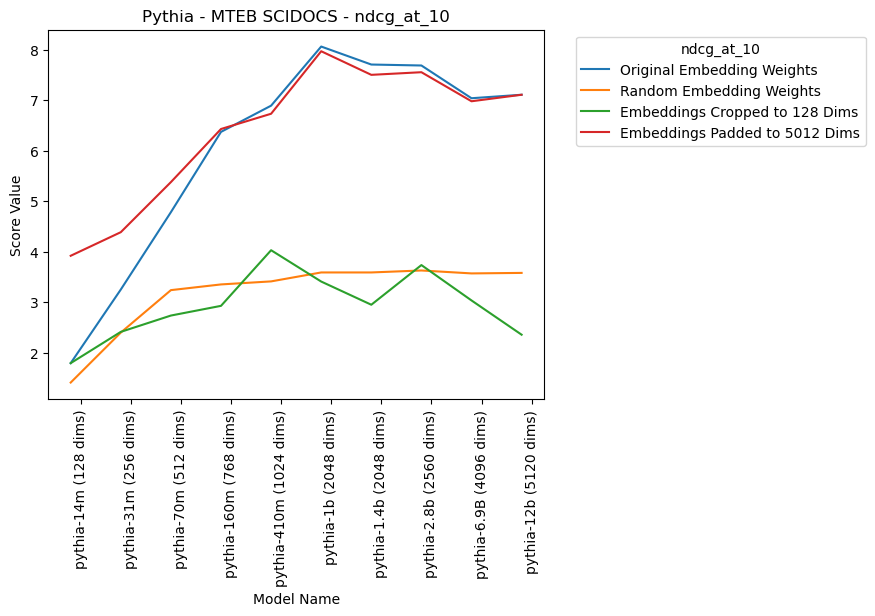
All the code for this analysis can be found in [this github repo](https://github.com/jstephencorey/LMEmbeddingAnalysis). A short summary of how I analyzed the model suites:

1. Take the token embedding layer from each model in the suite (Note that this is NOT doing any computation in the model, only taking the embedding lookup table of the model and evaluating its output).
2. Use the SentenceTransformer library to combine the model suite’s tokenizer, the embedding layer, and a mean pooling module, to result in a single embedding vector per sentence.
3. The model encodes sentences as a single, *d\_model* sized embedding. These embeddings are then evaluated on the SCIDOCS retrieval benchmark, using [MTEB](https://github.com/embeddings-benchmark/mteb) ([beir](https://github.com/beir-cellar/beir/tree/main) under the hood). Quality of embeddings is then measured by using [Normalized Discounted Cumulative Gain (ncgd)](https://en.m.wikipedia.org/wiki/Discounted_cumulative_gain).

The main thing to know is that **higher is better**. None of these models are going to be competitive in embedding rankings, that’s not the point. (the highest model as of this writing on the MTEB leaderboard for the SCIDOCS retrieval, [all-mpnet-base-v2](https://huggingface.co/sentence-transformers/all-mpnet-base-v2), scores a 23.76, whereas the highest model in my analysis scored less than 8). The goal is to compare model embedding layers with others in the same suite and others in different suites using a common metric.

### Pythia

Pythia is one of the more interesting suites because of how open it is, including a release of many checkpoints throughout training. Below is a graph of the following ablations:



1. **Original Embedding Weights**: Take the embeddings as they are from the models, unchanged and just how the first layer of the LM would see them.

The main thing that surprised me when looking at this is that the line doesn’t just keep going up. If you had to pick one embedding layer of a model to embed your sentences, you should pick pythia-1b, not, as you might expect, pythia-12b. This is true in spite of the fact that pythia-12b is a more capable language model and has a embedding size twice that of pythia-1b.

Especially of note is the comparison between pythia-1.4b and pythia-1b. These two models are the closest of any in this analysis to being a controlled experiment. These two models have the same embedding size and were initialized with identical embeddings before pre-training. Pythia-1.4b has 2x as many attention heads (8 vs. 16) and 1.5x as many hidden layers (16 vs. 24). However, dispute being the larger model, pythia-1.4b scores slightly *worse*. This seems to imply that, past a given point, making the model larger makes the embedding layer worse, all else being equal. This pattern of stagnation or degradation after a point is confirmed in other model suites as well, though the cutoff is not always the same embedding/model size.

1. **Random Embedding Weight**s: Re-initialize the embeddings of each model to a set of random numbers (xavier\_normal). (Note, this is a shared seed, so every dimension for a given token id will be identical across models and suites)

At least some of the improvement as embedding size increases comes purely from size, as noted by the gentle increase in the random baseline. The random baseline is further discussed in a later section.

1. **Embeddings Cropped to 128 Dims**: Cut down the embeddings of all of the models so they only encode tokens as the first 128 dimensions, as a way to measure how much “information per dimension” a given embedding has. (Note that this is identical to the original embedding weights for pythia-14m)

The “information per dimension”, as measured by quality of the first 128 dimensions of each model also seems to peak and then stagnate (if one model size earlier). This seems to imply that even at the level of the embedding vectors, the Pythia models smaller than pythia-410m don’t learn as much as they potentially have the embedding space to.

Also of note is that the first 128 dimensions of the embedding space of pythia-12b are only marginally better than that of pythia-14m. This implies that while pythia-12b as a model may be significantly more capable than pythia-14m, on a per-dimension basis, the embedding vectors are about as capable/information dense, at least I'm respect to usefulness for retrieval. Pythia-12b is more capable as a model in part because of a larger dimensionality, not because it uses its dimensions better (at least, for the embedding layer).

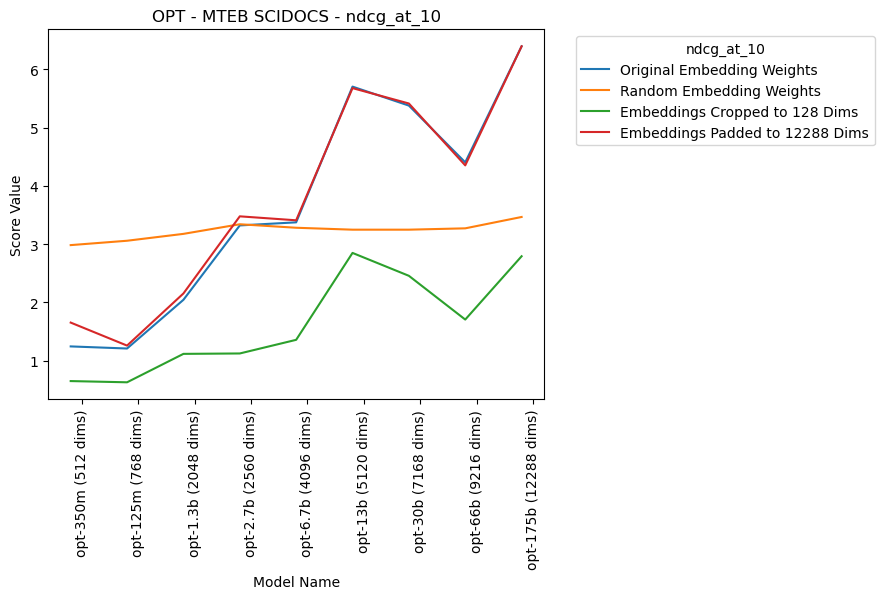
1. **Embeddings Padded to 5012 Dims**: Pad the embeddings with random numbers so each models encodes each token as the model embedding concatenated with a random vector so they are all the same size (5012 dimensions). (Note that this is identical to the original embedding weights for pythia-12b)

Padding the original weights with random numbers helps up to about 512 original dimensions. The trained information in the 128/256/512 dimensions from the model improve notably when noise is added to the vector (likely because there’s more dimensions to compare with for the retrieval). This seems to imply that 128 dimensions just isn’t enough to really capture the full meanings of tokens.

### OPT

The OPT models, released by AI at Meta, are another nice set of models to look at because they cover a very wide variety of model sizes.

The padding in this model is up to 12288 dimensions, the size of the embeddings on the largest OPT model, opt-175b. This doesn’t affect the embeddings much, though, besides the smallest model.



There are three main things to note here in how this analysis differs from the Pythia analysis.

First of all, the model improves up to 13b parameters/5120 dimensions before basically plateauing, as opposed to Pythia’s increase up to 1b parameters/2048 dimensions and plateauing at that size (technically it's more “inconsistent” than strictly “plateauing”, but the main point is that it doesn't improve much at all). It’s unclear why both model suites plateau, but at very different sizes.

Secondly, the random baseline is better than the smaller models. Unlike the Pythia suite, where each model outperforms the random baseline, however the smaller OPT models are trained, they seem to lose information that would be helpful in encoding information for retrieval. This could be a initialization issue, or potentially an issue with this particular metric.

### Other Model Suites

The further individual analysis of the other suites considered in this blog post - [Cerebras](https://github.com/jstephencorey/LMEmbeddingAnalysis/blob/main/plots/Cerebras_MTEB%20SCIDOCS_ndcg_at_10.png), [Bloom](https://github.com/jstephencorey/LMEmbeddingAnalysis/blob/main/plots/Bloom_MTEB%20SCIDOCS_ndcg_at_10.png), and [T5](https://github.com/jstephencorey/LMEmbeddingAnalysis/blob/main/plots/T5-v1_1_MTEB%20SCIDOCS_ndcg_at_10.png) - are all found on the github repo associated with this analysis, as well as the [raw data](https://github.com/jstephencorey/LMEmbeddingAnalysis/tree/main/data) that went into the graphs in this analysis.

### All Model Suites

Considered next to each other, the models form an interesting comparison.

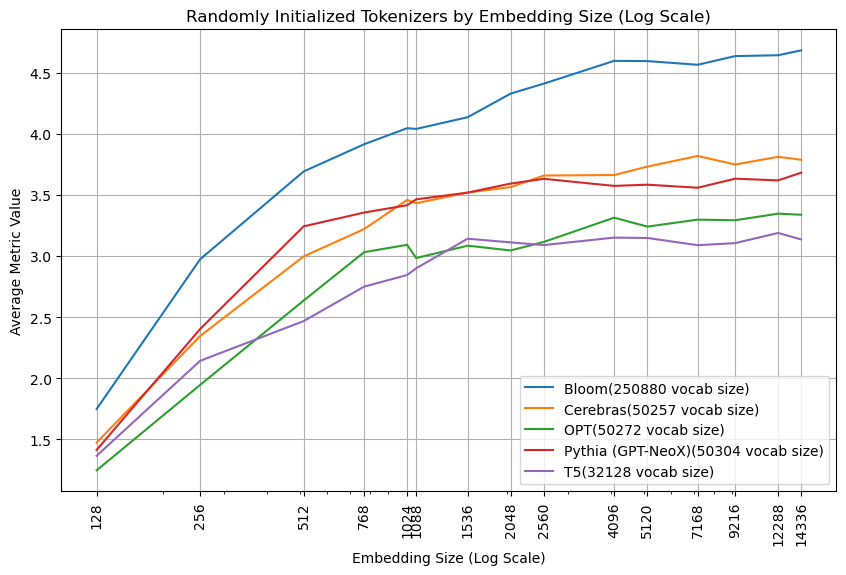
### 

This combined analysis shows how, despite some differences, there is a notable pattern that, past a certain point, embedding quality stagnates or regresses.

The one exception to this is Cerebras, though given the similarity of the Cerebras models to OPT models, I hypothesize it would plateau soon if the suite kept increasing in size. This peak/stagnation point occurs at different sizes in different model suites. Namely, 1024 dimensions for T5, 2048 dimensions for Pythia and Bloom, and 5120 dimensions for OPT.

Also notable is that the embeddings are very different quality by suite. OPT-66b’s embeddings (9216 dimensions) are slightly worse at embedding than pythia-70m’s embedding (512 dimensions).

### Random baseline from the tokenizers:



Of course, model suites vary in many ways besides just model size, and one big change with a potentially large impact is tokenizer choice. I think tokenizers are understudied, and point to efforts like [tokenmonster](https://github.com/alasdairforsythe/tokenmonster) to rethink the way we consider tokenizing text for LLMs.

Looking at how the tokenization affects the random baselines is rather interesting. For instance, the fact that Bloom notably improves over the other models makes sense when you consider that it has a vocab size almost 5x that of the next largest tokenizer (larger vocabularies intuitively make it easier to differentiate between randomly initialized tokens). Of most interest to me, however, is that Cerebras, OPT, and Pythia have almost identical vocab sizes, but score somewhat differently. (I ran this for a few other seeds, and though the exact lines slightly vary, the graph overall looks the same, see the [plots](https://github.com/jstephencorey/LMEmbeddingAnalysis/tree/main/plots) folder for those graphs).

Overall it seems like tokenizers may have some influence on embedding quality, though nothing to substantially effect this work’s conclusions.

## Implications and Future Work

Embedding quality in large language models (LLMs) tends to plateau or decline past a certain model size, as shown in this retrieval benchmark. This suggests several possible implications and areas for exploration:

1. Large models may underuse/undertrain their token embedding layers. This theory is that if the models were trained in a way that they had better embedding spaces, the models as a whole would be more capable models.

If true, this theory suggests a definite area for improvement in models by increasing token embedding quality. Some evidence against this theory is that that the embeddings usually train early on in the model, then stay relatively stable for the rest of pre-training (See [this informative blog post](https://www.lesswrong.com/posts/2JJtxitp6nqu6ffak/basic-facts-about-language-models-during-training-1#Tokenizer_Embeddings_are_rapidly_learnt_then_stabilize) for some more details)

If further “more of the same” pre-training does not result in much embedding quality gains, perhaps other methods for improving token embeddings ought to be improved, for instance pre-pre-training embeddings or different hyperparameter choices like suggested in [μP](https://arxiv.org/abs/2304.06875). (Though Cerebras was trained with [μP](https://arxiv.org/abs/2304.06875) hyperparameter choices, and this analysis doesn’t show significant differences from OPT’s non-μP hyperparameters).

2. The diminished role of embeddings in larger models may mean that beyond a specific size, the embedding layer’s information becomes less significant (and a model's interpretability becomes more complex) as meaning is increasingly encoded in subsequent layers.

Models like [EELBERT](https://arxiv.org/abs/2310.20144) have shown that capable models can be trained even if the embedding layer is an n-gram pooling hash function, so it seems like as long as there is some information in the embeddings to distinguish and correlate between tokens, the rest of the language model is capable of transforming that information (pun intended) into useful next-word predictions or sentence embeddings.

3. The possibility also exists that this retrieval metric may not comprehensively or sufficiently measure the quality of the embedding layer in large models. Repeating the evaluation using alternative methods remains an open task for further research.

## Conclusions

This analysis shows that the capacity of embedding layers in LLMs, as measured by their use in a retrieval benchmark, improves with the size of the model only up to a certain limit. This may imply limitations within smaller models and a potential underuse of embeddings in larger models. Future research should delve into whether embeddings are insufficiently utilized or if embedding layer importance diminishes as model size increases. These insights could influence strategies for architecture and pre-training of LLMs.

Whatever future work shows regarding the importance of embeddings in large models, this work can stand as evidence that bigger is not always better in every aspect of LLMs, and hopefully leads to a greater understanding of how LLMs work, scale, and learn.