FREEZE!: Pretrained Language Models as Frozen Feature Extractors for Semantic Tasks

Hayden McDonald
hayden_mcdonald@brown.edu

Nicholas Marsano
nicholas_marsano@brown.edu

Oliver McLaughlin
oliver_mclaughlin@brown.edu

December 4, 2024

1 Introduction

Our project is an investigation into a few related areas. In short, we wanted to see if we could develop an architecture that could take the intermediate activations of a pre-trained decoder-only large language model (Henceforth LLM) and use them to do text-embedding tasks. We chose semantic sentence similarity (STS) as our primary focus and motivation because of its simplicity. This task involves training a model to rate the semantic similarity of a pair of sentences. Importantly, in contrast to most other work using LLMs for text embedding, we are *not* fine-tuning the model. We are simply taking a subset of the intermediate activations and treating them as a kind of *"feature extracted"* (frozen) representation.

We chose this idea for several reasons.

1. Our original motivational question was: *"Do 'similar' prompts (to human readers) have 'similar' (under some metric) representations to a given language model?"*. By training a sentence-similarity model on LLM activations, we believed we could make progress on answering this question by *learning* a transformation of the activations that correlates to semantic similarity.

2. *If* we could train a small model to interpret a given LLM's activations and produce a useful text embedding (or set of embeddings) from them, we could skip costly fine-tuning and simply pre-train and fine-tune the small "interpreter" model. Pretrained small LLMs are cheap to do forward passes on and very plentiful, so if we could *just* use the intermediate activations and not have to do backward passes we could get a lot of functionality for very little compute.

3. We also wanted to investigate the question: *Do multiple intermediate activations perform better than just the last?* Typically when doing finetuning or transfer learning you remove the classifier end and use the last intermediate representation as your "feature extracted representation". We wanted to see if there was any performance benefit in using *multiple layer's* worth of representation. This is clearly the case in the finetuning setting [9] but it's unclear if it will work here.

4. From an interpretability standpoint, most distance or similarity measures (E.g cosine similarity) do not correlate strongly with *human ideas* of similarity (Figure 1). By training a similarity learner on LLM activations, grounded in human ideas of similarity, we hoped to produce a more interpretable measure of similarity for LLM activations.

5. There just isn't that much work on this particular topic. For this task you'd typically finetune the LLM using LoRA [5] or simply just use an existing text embedding model. We hoped to discover interesting properties and learn a lot about what it's like to try and *"disentangle"* an LLM's intermediate activations into something useful by pursuing this project. [1] We're not really interested in *what* information is where, we just want to see *if it's useable* for a simple task like STS.

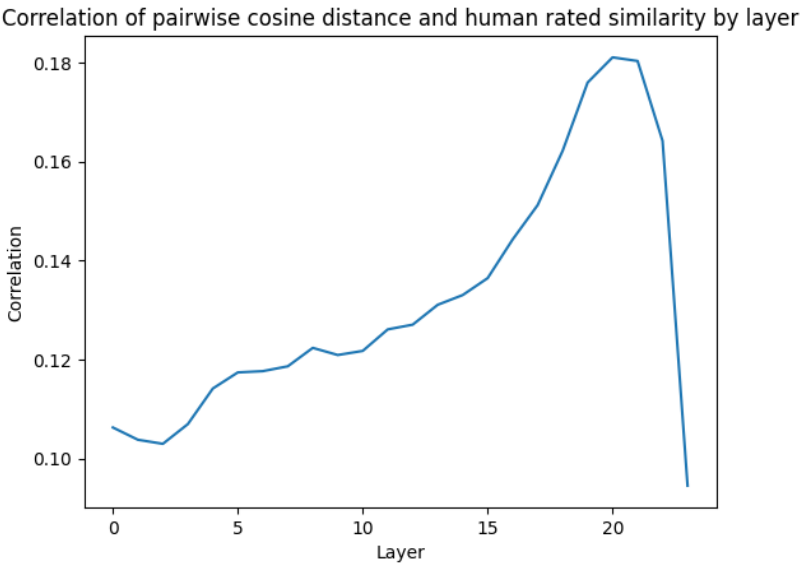

Figure 1: **The problem we are ultimately trying to solve**. Layerwise average correlation between gpt2-medium's embeddings and their cosine similarity vs. human similarity score for the STS training pairs

[1] There was only one paper we could find doing a similar setup – extracing latents from an LLM and applying them to a task – but it is very low quality and does not reveal many of the important details [1]. There are many papers *like* this one ([6] comes to mind) but almost none completely freeze the entire model and "start from scratch" as we did here.

We chose `gpt2-medium` (355M parameters) as our *"base model"* for study because Oliver was familiar with its architecture and it readily fit onto all of our GPUs. We also did minor experiments on `qwen2-0.5B` (391M parameters). Moreover we chose `gpt2-medium` because of its very low quality text generation ability. It was not at all a priori obvious to us that its representations would be fit for this task.

1.1 Problem Setup

Figure 1 depcits our "starting point". This figure depicts the correlation between two important quantities for our investigation:

1. The cosine similarity of two sentences' latent representations (See 3.2 for more information on this representation)
2. A human rated similarity score for those same two sentences (See 3.1)

For each layer we computed the mean correlation between these two quantities for more than five thousand sentence pairs. This is essentially the problem we are trying to solve – *How can we transform these latent representations into something useable for semantic text similarity?*

2 Related Work

3 Data

We have three major datasets that we used for both pretraining and supervised STS training.

1. STSB [4] – A standard sentence similarity benchmark with associated training and test sets.
2. GenericsKB [2] – A 'generic sentence' dataset of which we used twenty-thousand examples.
3. SNLI [3] – An inference relation dataset. Used in a pretraining experiment and in Sentence-BERT [8]

3.1 Semantic Text Similarity Benchmark (STSB)

The main task we targeted was STSB [4], which is a benchmark for semantic text similarity. It's comprised of pairs of sentences and a human-rated "similarity score". For example:

$$(\text{s1='A plane is taking off.'}, \text{s2='An air plane is taking off.'}, \text{sim} = 1.0)$$

The performance metrics for STS are typically Spearman's rank correlation coefficient and normal Pearson correlation between the predicted similarities of pairs of sentences and the human rated similarities. State of the art for this task is around `90 - 92%` in both spearman and pearson according to the MTEB leaderboard [7].

3.2 Data Collection and Pipeline

Our data pipeline is fairly simple. We're treating `gpt2-medium` as a feature extractor, so for each piece of text for any task, we simply pass that text through the model and extract a latent representation from the intermediate activations at each layer. We chose to take the *last token at each layer, after the residual connection* (Henceforth called a *latent*). This strategy is not original and is fairly standard in the *fine tuning* literature [10]. We expected it to do fine for our task given that `gpt2-medium` uses causal attention, even though we're not fine tuning. This gives us, for each sentence, twenty-four `1024` dimensional vectors (`n_layers x d_model`). One of the main challenges we wanted to overcome by taking this project on was finding a sensible way to *actually use* such a high volume and dimensionality of data.

4 Methodology

4.1 Architecture

Our model looks like (FIGURE). We do autoencoder pretraining stuff. Sentence-BERT style SNLI pretraining did improve performance over baseline but not better than the standard autoencoders.

4.2 Loss function

Our loss function for supervised STS training departs from the traditional mean squared-error (MSE) in that we only try and minimize the variance of the residuals, instead of the traditional variance and bias which are both minimized when using MSE.

5 Results

5.1 Requirements for success

Our main notion of success was performance on the STSB [4] benchmark as described in Section 3.1. Pearson/spearman correlation are both reasonable measures of success on this task because they measure how consistent your model is with human-rated similarity notions, regardless of scale or center. State of the art on this task is `90-92%` for both Spearman and Pearson correlation as of the writing of this document [7]. Our goal was to achieve at least `65%` spearman correlation. We felt this was a reasonable target to hit given that

6 Ethics

7 Reflection

One big takeaway from this project is that as gigantic open source LLMs become more pevalent, it might be possible to chop off most of their layers and train little "interpreter" modules as we did here to leverage the incredibly rich and complex learned represenations at a fraction of the cost and computation time.

8 Division of Labor

- *Hayden.*
- *Nick.*
- *Oliver.*

9 Appendix: *STSB* dataset leakage

References

[1] Mahmoud Basharat and Marwan Omar. Harnessing gpt-2 for feature extraction in malware detection: A novel approach to cybersecurity, 2024.

[2] Sumithra Bhakthavatsalam, Chloe Anastasiades, and Peter Clark. Genericskb: A knowledge base of generic statements, 2020.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[4] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055, 2017.

[5] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[6] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines, 2021.

[7] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[8] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[9] Yixuan Tang and Yi Yang. Pooling and attention: What are effective designs for llm-based embedding models?, 2024.

[10] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. (arXiv:2401.00368), May 2024. arXiv:2401.00368.