Adaptation, Sensitivity, and Introspection: Investigating the Capabilities of LLMs as Hypernetworks

Stanford CS224N Custom Project

Joseph O'Brien
Department of Computer Science
Stanford University
jobrien3@stanford.edu

Christopher Pondoc
Department of Computer Science
Stanford University
clpondoc@stanford.edu

Joseph Guman
Department of Computer Science
Stanford University
joeytg@stanford.edu

Abstract

Much work has been done to evaluate Large Language Models (LLMs) on common sense reasoning tasks, such as the Winograd Schema Challenge (Levesque et al., 2012). However, the underlying embeddings these models learn are static with respect to language, causing LLMs to perform poorly on benchmarks such as WinoDict that test lexical semantic word shifts (Eisenschlos et al., 2022b). These shifts occur when a word changes meaning over time or when a brand new word emerges. While retraining the entire model is a natural approach to solving this problem, carrying out this operation is expensive in terms of both computation and time (Patterson et al., 2021) and could potentially lead to model degradation. Hypernetworks (Ha et al., 2017), or neural networks that learn parameters for another model, have shown promise in editing language model behavior without the need for retraining (Cao et al., 2021; Kim and Jeong, 2021). Thus, we sought to train a separate LLM hypernetwork to learn and predict GPT-2 (Radford et al., 2019) token embeddings based on the token's definition. After achieving poor results on both WinoDict and our novel dataset CBTDict, we investigated the sensitivity of GPT-2's embedding layer further. Using several "forcing" experiments, we found that even slight perturbations to an embedding can lead to drastic differences in model performance on various downstream tasks. Overall, we argue that a path moving forward involves training transformer models to be more embedding agnostic or differentiating the two-pronged capabilities of language and embedding space understanding in a successful hypernetwork that targets new word embeddings. Our code for the project is open-source.

1 Key Information to include

- Mentor: Steven Cao

2 Introduction

Recent trends have seen a large spike in the popularity of LLMs among the wider public. OpenAI's ChatGPT, for example, attained 100 million users just two months after launching — the fastest growth rate achieved by a consumer app to date (Milmo, 2023). This expanding reliance on LLMs emphasizes a need for these models to have the capability to rapidly adapt to changes in the real world. While these changes include general knowledge – such as the election of a new prime minister for a country – they also consist of shifts in the underlying language itself. New words emerge in a dialect, and old words take on new meanings over time. Thus, language models must accommodate these changes in order to not only understand their user base but also to live up to their name.

After an LLM undergoes training, however, the model's capabilities become static and limited to the understanding acquired from the training corpus. The most straightforward way to update model capabilities is to retrain the model entirely. This retraining, though, does not come without a significant cost in resources: training the 175 billion parameter GPT-3 required 10,000 V100 GPUs and took just under 15 days to complete (Patterson et al., 2021). As these models continue to get larger, these expenses in terms of computation and time are likely to scale appropriately, highlighting the need for more efficient ways of updating these models. Additionally, for the model to learn an emerging trend, the number of available examples may be restricted, demonstrating another disadvantage of completely retraining the model.

One particularly promising technique for updating an LLM is the usage of hypernetworks (Ha et al., 2017). These networks learn the parameters of another model and can thus be used to make alterations to this other model. Since both the knowledge of the world and general reasoning capabilities are stored in a transformer model's layers, overly invasive interference runs the risk of degrading pre-existing behavior. Nonetheless, if such a network were able to successfully predict a model's parameters at the base token embedding layer, updating only this layer and leaving the model otherwise untouched would greatly decrease the potential for model degradation.

To this end, we decided to wield GPT-2 (Radford et al., 2019) as a hypernetwork for itself, utilizing definitions from WordNet to predict token embeddings for the model. In doing so, we hoped to employ an LLM's understanding of language to better obtain meaningful information from textual descriptions of a word, amplifying hypernetwork performance in this domain. We also aimed to understand how well LLMs can understand themselves. After the embeddings predicted by these trained networks performed on par with random chance, we took a deeper dive into the LLM architecture, conducting a group of "forcing" experiments that each made small alterations to this aspect of the model. These investigations revealed that even slight disturbances to a word's proper token embedding could lead to drastic changes in the model outcome. In turn, we discovered that a hypernetwork must predict a word's embedding nearly perfectly to not have the model's performance fall apart on difficult tasks. This suggests the potential for new modes of training and constructing these models to yield better results in understanding word shifts and emergences.

3 Related Work

Due to the increasingly high cost of retraining LLMs in terms of both computation and time, much work recently has focused on updating these models without having to carry out this process (Patterson et al., 2021). Some approaches have focused on altering model outputs without changing the base model at all. One example is ConCoRD (Mitchell et al., 2022b), a framework that builds an additional pre-trained natural language inference model on top of the base LLM to combat logical inconsistencies in question answering. In a similar vein, SERAC (Mitchell et al., 2022a) is another framework that alters a model's knowledge by storing edits in cached memory and guiding the base LLM's output using an edit scope classifier and a counterfactual model. These methods, however, act more like stopgaps in lieu of directly remedying the internal shortcomings of the base LLM.

In turn, other methods have focused on updating model behavior by altering the parameters of the base model itself. Sinitsin et al. (2020) utilized meta-learning (Finn et al., 2017) to train model parameters in a way that intrinsically prepares them for later alteration by an edit function that wields stochastic gradient descent during testing. Mitchell et al. (2021) proposed MEND, a method that trains a group of model editor networks that update model weights by utilizing a low-rank decomposition of the finetuning gradient of an input-output pairing. Cao et al. (2021) created KnowledgeEditor, an approach that wields a hypernetwork trained using constrained optimization to update an LLM's parameter weights in order to change the model's knowledge about a given fact. By altering model parameters broadly, though, these invasive methods increase the chances of model degradation.

Thus, the aforementioned frameworks do not provide a lightweight method to address lexical change to best preserve model functionality. Old words often adapt new meanings, and new words gradually emerge in everyday dialect. Work done by Shoemark et al. (2019) identifies the great presence of semantic change, or the shift in the meaning of words, in today's world through the medium of social media by analyzing Twitter data over a 5.5 year time span. The group found that words not only underwent shifts because of the emergence of new named entities but also due to neologisms - newly created words like "glo" in the phrase "glo up." The authors also pointed to the rise and fall in popularity of different named entities, emphasizing the need for fast token embedding updates to allow the model to adapt to these changes as they occur.

Akin to SERAC and ConCoRD, Eisenschlos et al. (2022a) addressed lexical shift without altering the base model at all. The group proposed WinoDict – an evaluation benchmark and dataset – to evaluate an LLM's in-context acquisition of new words using Winograd schemas. Other approaches have instead altered the base model itself to update and add new token embeddings. Work by Orosanu and Jouvet (2018) initialized new embeddings by identifying known words that had the most similar distribution of neighboring words to establish n-gram parameter elements. Kim and Jeong (2021) took more of a hypernetwork approach, wielding a convolutional neural network to predict effective word embeddings, finding promising early results in a simple tweet classification experiment. However, their validation average cosine similarity of 0.4825 at best revealed difficulty in mastering this topic.

4 Approach

4.1 G2G: Using GPT-2 as a hypernetwork for itself

At the start, we hypothesized that an LLM with a rich understanding of language would be best suited to both derive meaningful information from text and understand the behavior of another LLM. As a result, our first step was to train GPT-2 to generate effective word embeddings for itself (referred to as G2G). To do so, we found all the words in our dataset that corresponded to a single GPT-2 token embedding. For each word, we then supplied the dictionary definition of the word to the predictor model, making sure to include a classification (CLS) token at the end of the input. After running this input through the predictor model, the CLS token encompassed a learned summary of the word's definition in its embedding, which we returned as the model's output. Since our task was to predict a GPT-2 word embedding, we defined our loss as how different our prediction – the learned CLS token embedding – was from our ground truth – the GPT-2 token embedding for the corresponding word. We specifically employed a mean-square error loss – given the predicted embedding e_p, and the true GPT-2 embedding e_g, the loss was then:

$$\text{MSE}_{\text{loss}}(e_p, e_g) = \frac{1}{n}\sum_{i=1}^{n}(e_{p_n} - e_{g_n})^2$$

After completing this fine-tuning, we used our model to generate new GPT-2 word embeddings for the fake words sourced from WinoDict given the definition of their real word counterpart. We then initialized GPT-2 with these embeddings and evaluated the model using our two evaluation benchmarks – WinoDict and CBTDict. We use performance with GPT-2's default tokenization of the unfamiliar words as a baseline to compete against.

4.2 Exploring average embeddings as an initialization

After including our learned embeddings within GPT-2, we found that we not only achieved extremely poor results on these fake-word-oriented benchmarks: we also performed poorly on the previously high-performing CBTDisct task that used real words exclusively. Consequentially, we shifted our attention to better understanding the impact of new token embedding initializations on model performance. We first established a baseline: recent work from Hewitt (2021) initialized new word embeddings as samples from a multivariate distribution established using all pre-existing embeddings. Initializing new word embeddings using this method and loading them into GPT-2 showed minimal impact on performance, suggesting the notion of a "safe" new token embedding.

Sample based embeddings: $\mu = \frac{1}{n}\sum_i e_i; \quad \Sigma = (E-\mu)^\top(E-\mu)/n; \quad e_{\text{new}} \sim \mathcal{N}(\mu, \Sigma)$

Here, e_i represents the i'th embedding in the vocabulary, and E is an n by d matrix of all embeddings. We average over all pre-existing embeddings in the model, and find their covariance to create a multivariate normal distribution from which we sample new embeddings e_{new}.

4.3 “Forcing” experiments for safe embeddings

In an attempt to generate “safe” embeddings, we then ran several “forcing” experiments to see if we could teach a model to learn such safe embeddings. To start off, we supplied the model with the single generic sentence *"This is a very normal sentence that will have normal results to give to something"* as input and trained GPT-2 to predict the mean embedding defined by the multivariate distribution. Subsequently, we then took the same generic sentence, added a new word, and trained a separate GPT-2 model to predict the new word’s embedding. After training both of these hypernetworks, we then used both of them to generate new GPT-2 embeddings and evaluated them on our benchmarks.

4.4 Noising experiments to gauge fragility

After still facing issues with model performance, we conducted a final experiment to understand the overall fragility of a large transformer model with non-ideal embeddings. To do so, we ran a "noising experiment" where we noised GPT-2’s embeddings, holding an embedding’s L_2 norm constant while sampling from a multivariate noise distribution to slightly modify embedding orientation for a specific word. We then compared the performance of a noised embedding on our benchmarks to their non-noised counterparts.

$$\textbf{Noising embedding e:} \quad e' = e + \mathcal{N}(\vec{0}, 0.036\mathbf{I});\ e'' = e'\frac{|e|}{|e'|} \Rightarrow |e''| = |e|;\ \mathbb{E}\left[\frac{e''^{\top}e}{|e''||e|}\right] \approx 0.95$$

Here, we take a word embedding e and noise its individual elements independently to create a noised embedding e'. Embedding e' is then normalized such that its L_2 norm is equivalent to the original embedding e. We define the variance such that the average cosine similarity equals a desired value.

We then wielded WinoDict where we would, for a given example, switch out GPT-2’s embedding for the token(s) of the word that is considered critical to defining the meaning answer of the example. From there we would see what level of variance and corresponding expected cosine difference would cause GPT-2 to fail on the WinoDict schema, which would represent the model no longer being able to derive meaning from the word’s embedding.

5 Experiments

5.1 Data

WordNet: For predicting embeddings using definitions, we used the WordNet dataset. WordNet is a lexical database of English words, with each part-of-speech grouped into a set of synsets, or cognitive synonyms. Note that the WinoDict evaluation benchmark prompts the LLM using WordNet definitions, so we found that using the dataset would establish parity. We gathered definitions for all of the tokens in GPT-2 that had a definition and used all of those pairs to create our LLM hypernetwork that would predict GPT-2 embeddings. All in all, we had 2,954 total word-definition pairs from WordNet.

WinoDict: In addition, both of our benchmarks came with corresponding datasets. The WinoDict dataset includes 249 Winograd schema pairings, encompassing 498 examples overall. The original Winograd Schema Challenge that consists of 273 data points contributes 184 examples to this total, whereas the much larger WinoGrande (Sakaguchi et al., 2019) with 12,282 inputs contributes 314. These data points were chosen so that examples that did not have a pairing were not included and so that the key concept token that defines the schema changes within a pairing.

CBTDict: To create the CBTDict dataset, our group took a random sampling of 2,000 of the original 687,343 examples contained within the Children’s Book Test (Hill et al., 2016). When selecting these examples, we made sure to only include those which possessed real words that corresponded to fake WinoDict words, as we applied the fake words from WinoDict as the fake words for CBTDict. We made this decision because these WinoDict words encompassed the test set for which we tasked our GPT-2 hypernetwork to learn token embeddings. We also made sure to exclude examples that were too long for our model to process after being tokenized. We believe CBTDict is an important evaluation benchmark for this task as its comprehension tasks are easier in comparison to WinoDict because they do not require a nuanced understanding of a single critical word, making it a useful for detecting early success while iterating solutions.

5.2 Evaluation Methods

WinoDict: Our first evaluation was using WinoDict, which consists of Winograd schemas. A Winograd schema is a reading comprehension question that requires choosing from two noun options (Levesque et al., 2012). The job of the model under evaluation is to correctly identify which of the two noun options a pronoun in the sentence refers to. Consider the following Winograd schema:

"The trophy would not fit into the brown suitcase because it was too {large/small}. What was too {large/small}?"

When "large" completes the sentence, we see that "it" refers to the trophy. However, when "small" completes the sentence, the answer changes to the brown suitcase. WinoDict expands upon Winograd by replacing the central word that disambiguates the pronoun with a fake word. The fake word adopts the same definition as the real word it replaced. This adopted definition for the fake word – retrieved from the WordNet dataset – is also placed before the example sentence.

CBTDict: Our second evaluation utilized a dataset that we created ourselves - CBTDict. CBTDict is composed of a subset of the Children's Book Test, whose examples follow a similar methodology to that of a Winograd schema in the form of a reading comprehension question. The key difference between the two methods is that a Children's Book Test example evaluates a model's ability to reason using a large amount of context with a clearer solution in comparison to the adversarial structure of a Winograd schema. A CBT example consists of several sentences of context at the beginning followed by a concluding question sentence. The question sentence has a word omitted, and the model's task is to identify the correct answer from a group of 10 options. Consider the following CBT example:

Context Sentences: ... *her work, hoped she was getting on well, and said she had two young rascals of his own to send soon. Esther felt relived.*
Question: *She thought that Mr. ____ had exaggerated matters a little.*
Options: *Cropper, Esther,* ***Baxter****, course, fingers, manner, objection, opinion, right, spite*

In this scenario, using the information gathered from the previous context sentences the model is expected to provide "Baxter" as the answer.

Partial vs. Full Scoring: We use two similar methods to score a model's performance on both WinoDict and CBTDict: "full scoring" and "partial scoring." Full scoring uses the probability of the entire sentence to define the score. Partial scoring calculates the score as the conditional probability of the phrase following the replaced pronoun given in the preceding part of the sentence:

$\text{Score}_{\text{full}}$("the trophy") = P(The trophy would not fit into the brown suitcase because **the trophy** was too large)

$\text{Score}_{\text{partial}}$("the trophy") = P(was too large | The trophy would not fit into the brown suitcase because **the trophy**)

For WinoDict, we chose "partial scoring" between the two options because of the approach's superior performance. For CBTDict, we chose full scoring because oftentimes there were none or little words after the blanked out word. To implement partial scoring, we constructed each candidate sentence and masked out their labels except for the phrase following the potential solution. After feeding the sentences and their labels as inputs into our model, we then determined that the input with lower loss was the model's prediction, as the loss represents the conditional probability of the sentence's ending given the entire sentence as context. This same approach held for full scoring, except we did not mask out any context. Concretely – given candidate sentences s_1 and s_2, we determined that the model would choose s_1 as its prediction if:

$$\texttt{model(tokenize(}s_1\texttt{)).loss < model(tokenize(}s_2\texttt{)).loss}$$

5.3 Experimental Details

G2G Experiment: For predicting GPT-2 embeddings using a separate LLM, we used GPT-2 Medium to predict word embeddings for itself. We specifically chose GPT-2 Medium due to its superiority to smaller versions of GPT-2 on common sense reasoning tasks while still being able to iterate quickly and run experiments without running out of CUDA memory. When training our hypernetwork, we used a batch size of 2 and gathered gradients across 10 batches for our hyperparameters. Since we used a learning rate of 0.001, gathering gradients gave us an effective learning rate of 0.01 and a batch size of 20. Finally, we kept the model's default dropout normalization for our fine-tuning tasks. Overall, we trained the G2G task for 10 epochs, reaching testing loss convergence and taking a training time of 58.33 minutes. When evaluating on CBTDict, we also only used the first 200 examples because of the extreme differences in results that are not just due to noise. Training and validation sizes are 2659 and 295 respectively.

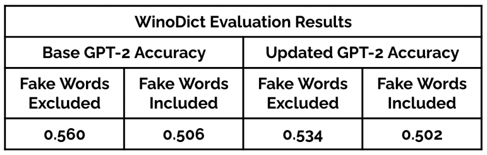
Averaged Embeddings Experiment: When using averaging technique to initialize embeddings for all of the fake words in our evaluation benchmarks, we did not train the model: we simply evaluated the model with the new embeddings on CBTDict and a model loaded with averaged embeddings on default CBT. When evaluating on CBTDict, we again only used the first 200 examples for the same reasons stated earlier.

"Forcing" Experiments: When training GPT-2 for our "forcing" experiments, we did not need to batch examples or gather gradients as every example and label was identical, and there is no need to average identical gradients. Thus, we used a learning rate of 0.01 and a batch size of 1 for 8000 iterations. Finally, we kept the model's default dropout normalization for our fine-tuning tasks. Training took 24.733 minutes. When evaluating on CBTDict, we again only used the first 200 examples for the same reasons stated earlier.

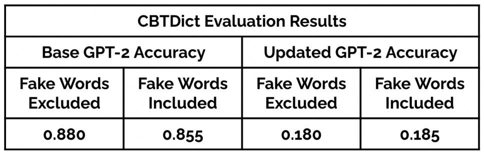
Noising Experiment Details: When performing our noising experiments, we once again did not train the model. However, we only evaluated the model with the noised embedding on a subset of 300 examples of the WinoDict schema training examples.

5.4 Results

G2G Results: As we noted earlier, G2G was unable to make strong predictions on both WinoDict and CBTDict. This performance held true both when we did and did not replace real words with fake words.



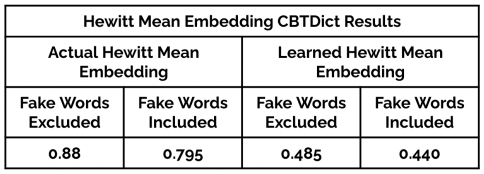
WinoDict Evaluation Results			
Base GPT-2 Accuracy		Updated GPT-2 Accuracy	
Fake Words Excluded	Fake Words Included	Fake Words Excluded	Fake Words Included
0.560	0.506	0.534	0.502



CBTDict Evaluation Results			
Base GPT-2 Accuracy		Updated GPT-2 Accuracy	
Fake Words Excluded	Fake Words Included	Fake Words Excluded	Fake Words Included
0.880	0.855	0.180	0.185

Figure 1: Evaluation results for WinoDict and CBTDict.

Sampling Embeddings and "Forcing" Experiment Results: Conversely, we found that utilizing the Hewitt embedding approach on the CBTDict task led to no hindrance on model performance when using real words exclusively and only a 6 percent worse showing when including fake words. When attempting to have GPT-2 learn these embeddings through our forcing experiment on a single sentence, however, we found that the model did not converge to this output even after running 8,000 unique iterations of training. The method of trying to have the model learn an embedding for a word appended at the end of the sentence such as "boat" with no significant difference in loss convergance and training speed. When using these GPT-2 learned average embeddings trained over 8000 iterations on the model as the token embeddings for the fake words for the CBTDict task, we found that model performance degraded by roughly half as compared to the actual average embedding.



Hewitt Mean Embedding CBTDict Results			
Actual Hewitt Mean Embedding		Learned Hewitt Mean Embedding	
Fake Words Excluded	Fake Words Included	Fake Words Excluded	Fake Words Included
0.88	0.795	0.485	0.440

Figure 2: Results for the the actual and learned Hewitt mean embedding on CBTDict.

Noising Experiment Results: For our noising experiment, we run the noising process described in the approach section on each word in GPT-2's vocabulary to estimate an expectation of cosine similarity. We found that minor perturbations would greatly destroy model performance. For instance, a cosine similarity expectation of 0.95 would lead to a 0.5467 accuracy against the default 0.5933 accuracy. We also see that a 0.9 similarity lead to an accuracy of 0.4867, significantly below guessing.

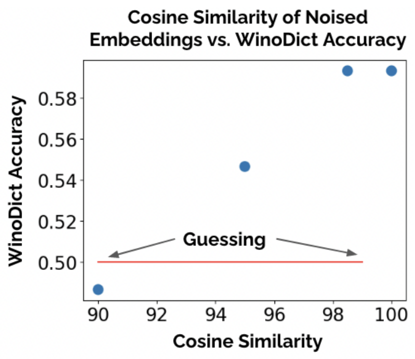


Figure 3: Graph of the noising experiment.

6 Analysis

6.1 Localized embedding level interventions still have risk of model degeneration

Since G2G-generated embeddings deteriorate GPT-2 performance on our common-sense reasoning evaluation benchmarks, we infer that a deficient embedding can actually degrade model performance. Thus, even a non-invasive alteration of the model can have unforeseen consequences on language reasoning capabilities. As for why: note that GPT-2's final layer, the language model (LM) head, is multiplied by the transformed embedding of the given word in a sequence to create logits for a probability distribution of the next word. The model's loss on reasoning tasks is then calculated from these logits. However, the rows of GPT-2's LM head themselves are embeddings of the vocabulary. Therefore, even if a word is not used as input in a task, its embedding will still influence model behavior at all points by unpredictably modifying the normalizing constant.

6.2 Embeddings that avoid degradation are not always "perfect"

An embedding that avoids degradation did not necessarily have to be a perfect embedding that mapped to the word's underlying meaning. In fact, when creating new embeddings for the words in our evaluation benchmarks, we found that simply being statistically significant – in this case, being retrieved from a multivariate normal distribution – was enough to prevent deteriorating model performance.

6.3 Implied separation between handling embeddings for general purpose language tasks and handling embeddings as a hypernetwork

For both "forcing experiments" – where we trained GPT-2 to predict a non-damaging embedding and the embedding of a word in the input – we saw that even after training for thousands of iterations, the model was unable to generate close approximations of their ground truth labels. This behavior suggests that GPT-2's attention layers must be trained in a way far from what would be needed in a hypernetwork, as the model was designed to handle language on a macroscospic scale. Thus, the immense training needed to modify this behavior into G2G would destroy the model's inherent understanding of language, nullifying our motivation for using an LLM hypernetwork in the first place.

6.4 Observed fragility of LLMs and lack of embedding agnosticism

Since even minor perturbations to embeddings greatly destroyed model performance, we realized that LLMs as hypernetworks are fragile. In fact, a hypernetwork for GPT-2 would need to have an incredible understanding of GPT-2's large embedding space to reliably create high quality embeddings while also having a robust understanding of language to encode information from a textual description of a word. However, LLM based hypernetworks do not yet demonstrate this embedding space understanding within their macroscopic view of language.

7 Conclusion

Motivated by an idea to use an LLM as a hypernetwork for word shifts, we have better studied the dangers and difficulties associated with updating a model's knowledge of the world through non-invasive interventions in embeddings. Using GPT-2 to predict GPT-2 embeddings given a word's definitions significantly destroyed model performance, suggesting that embedding-level changes may lead to risk of model degradation. However, using an averaging method for initializing embeddings proved to have minimal effects on purpose, showing the possible disconnect between embeddings that avoid harming preexisting model performance and risky embeddings created to explicitly add functionality to a model. When training GPT-2 to predict either average embeddings or a real word's embedding, we found that GPT-2 was trained in a way that would not easily be helpful as a hypernetwork. Finally, by noising a portion of the embeddings of the model, we found that LLMs are incredibly fragile, suggesting that state of the art language models would require an incredibly powerful hypernetwork to consistently create effective and non-damaging word embeddings.

Guided by our research findings, we propose two main avenues for future work. One would involve training transformer models to be more embedding agnostic. In the same way dropout regularization trains a model to not require the parameter's inputs of a specific lower layer, we may experiment with an even stronger form of regularization where we add noise to the embeddings in each layer during training. Ideally, this would train the model to recognize meaning across a wider range of embedding directions and more uniformly take advantage of the immense, high-dimensional space. In turn this might make a model that can more easily accept new embeddings. Another method would be exploring the augmentation of hypernetworks with both language and embedding space understanding. Since there has been successful research in placing an additional model on top of a language model to expand it's abilities, rather than trying to train G2G to convert information into an embedding prediction within layers that were trained to understand language, it might be beneficial to augment the architecture by adding new layers than can scan across the entire hidden states within GPT-2's upper layers and then convert this into an embedding. This may allow us to create a hypernetwork that can process encodings of the entire textual description for a conversion into the embedding vector space while preserving and taking advantage of GPT-2's internal understanding of language.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

Julian Martin Eisenschlos, Jeremy R. Cole, Fangyu Liu, and William W. Cohen. 2022a. Winodict: Probing language models for in-context word acquisition.

Julian Martin Eisenschlos, Jeremy R. Cole, Fangyu Liu, and William Weston Cohen. 2022b. Winodict: Probing language models for in-context language acquisition.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *International Conference on Learning Representations*.

John Hewitt. 2021. Initializing new word embeddings for pretrained language models.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations.

Jihye Kim and Ok-Ran Jeong. 2021. Mirroring vector space embedding for new words. *IEEE Access*, 9:99954–99967.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, page 552–561. AAAI Press.

Dan Milmo. 2023. Chatgpt reaches 100 million users two months after launch.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2021. Fast model editing at scale.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022a. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Eric Mitchell, Joseph J. Noh, Siyan Li, William S. Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher D. Manning. 2022b. Enhancing self-consistency and performance of pre-trained language models through natural language inference.

Luiza Orosanu and Denis Jouvet. 2018. Adding new words into a language model using parameters of known words with similar behavior. *Procedia Computer Science*, 128:18–24. 1st International Conference on Natural Language and Speech Processing.

David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale.

Philippa Shoemark, Farhana Ferdousi Liza, Dong Nguyen, Scott Hale, and Barbara McGillivray. 2019. Room to Glo: A systematic comparison of semantic change detection approaches with word embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 66–76, Hong Kong, China. Association for Computational Linguistics.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.