Towards Integrating Medium-Term Memory into Transformer Models:

Directing Sentiment and Evoking Recall in GPT-2 from an Attention Bock Based Persistent Memory using a Small Fraction of One Layer and a Case for the Spacial Separation of Intrinsic and Explicit Memory

Abstract

Transformers use both their context and internal parameters as memory which typically leaves an amnesia between the two memory types. A medium-term memory should be able to fill the gap between the context and internal parameters while reducing both the computational cost and memory footprint over an extended context. This paper proposes a bolt on method which uses a *k*-nearest neighbours algorithm to search a persistent memory while reusing the internal machinery of the Transformers' attention blocks to determine what to search for and how to rank the search results. It was found that sentiment direction through what might be described as intrinsic memory and recall (explicit memory) appear to arise from different layers in GPT-2 (layers 7, 8 and 9 vs. layer 10 respectively). For sentiment direction, it was found that with a positive persistent memory it required a fraction of layers (< 3) and a fraction of each layers' tokens*heads (< 10%) to change the VADER compound score from a median of -0.13 with an IQR of 1.6 in unadulterated GPT-2 to a median of 0.99 with an IQR of 0.01 with the method. For recall, it was found that the method pulls similar tokens out of the persistent memory as GPT-2 appears to "look" at in the context. Further, it was found that only a fraction of the first context tokens' tokens*heads in a single layer (5 out of 2160 total tokens*heads) was sufficient to produce similar or better recall results in most cases than when the Wikipedia text was placed in the context. This suggests that the computational cost of recall using this method may be context length independent. As this work only tests the method as a bolt on addition to GPT-2 it is impossible to determine if it will perform as an effective medium-term memory in more advanced models. However, it is believed that the data provided within suggests the method described herein merits further investigation.

1 Introduction:

In humans, memory can be simplistically separated into short-term or working memory and long-term memory.[1] Here short-term and working memory describe two different concepts, where short-term is the classic memory of the 7-digit phone number and working memory is a scratch pad like concept, both of which are not stored for long periods of time.[1] Long-term memory on the other hand describes a filtered memory for "important items" which last much longer.[1] It is tempting to draw analogies of these concepts to large language Transformer models (LLMs), where the context (what the user types in) is the models' short-term/working memory and the models internal parameters (representing the data the model was trained on) are the long term memories. This analogy exposes a major flaw with the Transformer models', the gap in memory between when the model was trained and when the user filled in the context. This amnesia causes two problems. The first is the model's knowledge of the world can be quite out of date. In the case of the initial release of GPT-4 the amnesia was on the order of years.[2] The second, but perhaps the most important, is the model can not know what the user previously added into the context window or the previous response the model provided if they are not re-entered into the context window and recomputed. Both of these issues have been somewhat mitigated by extending the length of the context window. However, extending the context window comes with its own issues, namely a significant increase in the amount of compute and memory requirements. For instance, doubling the length of the context window doubles the size of the model which in turn doubles the memory required and increases the required compute. There have been a number of techniques proposed to mitigate the memory and compute cost of extending the context window.[3–6] For instance, with the Transformer-XL Dai *et al.* sought to minimize computational cost by storing and reusing previously calculated internal outputs.[4] Further, recurrent memory Transformers have had great success with addressable tokens being demonstrated to be in the millions.[7,8]

In order to implement a medium-term memory, or a persistent memory of arbitrary size, the compute and memory required ideally needs to be drastically reduced over the current context extension techniques. This necessitates a number of structural changes. The most obvious is implementing a search over the memory as well as reducing the number of layers involved and the number of tokens*heads used for memory retrieval in each of those layers. Considering both the information flow of a Transformer model and the inner workings of the attention mechanism[9], the possibility of a medium-term memory appears tantalizingly plausible.

Since the original Transformer was described[9] a gradual understanding of how they function has emerged. The Transformer itself is composed of multiple layers which in turn are composed of two repeating blocks, an attention block and a feed-forward block. These two blocks are connected by a residual or, more specifically, the input for each block is added to the output of the same block. This residual is extended through the layer level, creating an information flow with each layer adding its information to the whole until that information is converted into the next token. It is now thought that the attention block is involved in information mixing, or rather determining what is "important" to that particular layer and emphasizing it.[10,11] It has been shown that the early attention blocks focus on nearby relationships and on structure[12] and later layers focus on longer distance relationships.[11] The feed-forward blocks on the other hand appear to be the long-term memories of the Transformer.[13–15] Geva *et. al.* have shown that the feed-forward blocks work both by creating a distribution of memories[14] and a method of promoting a particular memory to the most probable next token.[13] Further it was shown that the outputs of the feed-forward blocks can be edited to produce either particular tokens or produce a more positive sentiment.[13,15] By corrupting particular tokens and restoring the layer outputs from an uncorrupted layer Meng *et al.* were able to show that knowledge (such as place names, *i.e.* Seattle) were restored if the feed-forward block output was restored at approximately the middle of GPT-2 XL.[15] More importantly for this paper though, they showed that knowledge was restored if the attention block was restored in the later layers of GPT-2 XL.[15] The attention block is important to this paper as it appears to contain all the necessary components to implement a medium-term memory.

A Transformer's attention block is composed of a Softmax (S_{QK}) of the Keys (K) and Querys (Q) followed by the S_{QK} and Value (V) matrix multiplication (equations 1 and 2).

$$S_{QK} = Softmax\left(\frac{Q K^T}{\sqrt{d_k}}\right) \tag{1}$$

$$Attention(Q, K, V) = S_{QK} V \tag{2}$$

Here S_{QK} can be personified as deciding which Values are important to the attention block and hence can be used to determine which Values should be used for memory retrieval. It is not a giant leap to predict that multiplying the Query by a previously stored Key would provide information on how important the Value associated with that Key is to the attention block. Thus, storing the Keys and Values should be sufficient to create a medium-term memory. This is not a new idea, it is essentially how Transformer XL[4] works. However, as mentioned earlier significant reductions in computational cost and memory footprint are required over techniques such as Transformer XL. This is where Meng *et al.*'s finding comes in handy. If knowledge recovery from the attention block can be instigated in the final layers[15] a medium-term memory would not require all the Transformers' outputs to be saved. If this is transferable to current state-of-the-art models, this would be a significant amount of memory and compute saved.

As noted earlier a memory of arbitrary size requires a search mechanism. There are a number of algorithms that may work, however, in this paper the *k*-nearest neighbours algorithm was chosen. In particular, Facebook AI Similarity Search (FAISS).[16] The *k*-nearest neighbours algorithm has been previously used on the Transformers' attention blocks with great success.[17,18] In this work it is shown that the bolt on method described herein only requires three or fewer layers and a fraction of each layers' tokens*heads to generate acceptable results.

For this paper, it was hypothesized that in the later layers the next token distribution would be relatively refined, and the attention mechanism would be looking over long distances to further refine the distribution.[11,13] It was also hypothesized that the medium-term memory could act associatively or rather finding similar memories would refine the prediction of the next token. To be more specific, it was predicted that:

1) The later attention blocks would be looking over long distances to find tokens that would help refine the next token.
2) Equation 1 would give information about which tokens*heads the attention block found most valuable.
3) The keys created by the attention block would act as an ordered index such that similar "memories" would have similar keys and each attention block would order it differently depending on what the block was looking for.
4) Equation 1 could give information about how valuable each of the *k*-nearest neighbours for each token was to the attention block
5) Equation 2 could be used to determine what fraction of each nearest neighbour "memory" should be added back into the original Value matrix.

This paper focuses on sentiment direction through what might be classified as intrinsic memory and fact recall, or explicit memory, from GPT-2.[19] These two topics help illustrate that implementing the medium-term memory on different layers results in two different and perhaps complementary results. Given that this paper focuses on an old model, it cannot be determined if this is an effective medium-term memory for more sophisticated models. However, it is believed that the data within provides tantalizing clues as to how it might be solved.

2 Methods:

2.1 Transformer Attention:

The LLMs this paper refers to and works on, are Transformers.[9] As mentioned in the introduction a Transformer is made of multiple layers with each layer connected by a residual. Each layer is made up of an attention block and a Feed-Forward block. The focus of this paper's method is the attention block which first creates a Query (Q), Key (K) and Value (V) via linear transformations of the residual, x.

$$Q = x W_Q^T + b,\ K = x W_K^T + b,\ V = x W_V^T + b \tag{3}$$

The model dimension of Q, K and V is then linearly projected h times to form h heads. Equations 1 and 2 are then applied, and the heads are projected back into the model dimension to produce the output of the attention block. The output of the attention block is then added back into the residual.

2.2 Creating a Persistent Memory:

For all experiments in this paper a persistent memory was created first, and then the experiment was run. This was not necessary, but was convenient for repeating the experiments on the same persistent memory. For instance, the memory used for the sentiment experiments is shown below:

> *"Cats are absolutely amazing creatures that bring so much joy and happiness into our lives. Every cat has its own unique personality that makes them an absolute delight to be around. A cat's playful nature and charming personality can brighten up any day. Cats are incredibly adaptable and can thrive in any environment, making them the perfect pet for any home. With a cat by your side, you'll always have a loyal friend to share your life with."*

The above was produced by Microsoft Bing's version of GPT-4[20] and does not represent anything refined. It allows for the creation of a series of value vectors which associate cats with positive aspects.

2.2.1 Persistent Memory Creation 1:

The memory text was transformed into a usable form by running it through GPT-2 and storing the Values and Keys from the attention blocks of interest. The Values and the Keys were ordered by token index and stored in the hierarchy of Layer → Head. For each head a FAISS[16] index was created for the list of Keys.

2.2.2 Persistent Memory Creation 2:

A newline character ("*n*") was added to the front of the memory text. The memory text was transformed into a usable form by running it through GPT-2. The vectors representing the first token (the newline character, "*n*") were removed from the Values and Keys in the attention blocks of interest. The resulting Values and Keys were ordered by token index and stored in the hierarchy of Layer → Head. For each head a FAISS[16] index was created for the list of Keys.

2.3 Inference Time Operation:

As mentioned in the introduction, this paper takes the view that S_{QK} (equation 1) gives information about the "importance" of each Value vector to each head. This can be extended to the view that a Query vector can be plugged into equation 1 with any Key vector to give the importance of the Value vector associated with the Query vector.

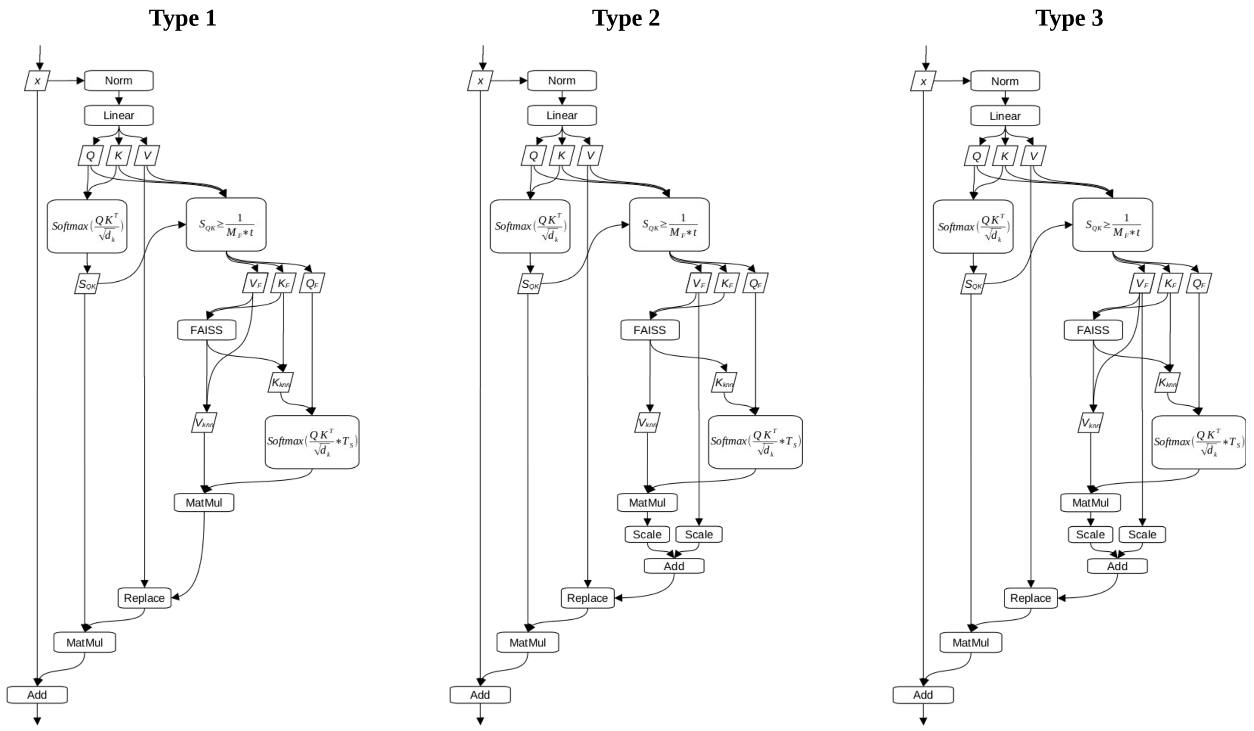


Figure 1: Flow charts of the three different methods added to the Transformer Attention blocks

This papers inference time method is inserted between equation 1 and equation 2 in the attention module. Each of the Value vectors and Key vectors (size d_k) were filtered by token index according to equation 4. Where M_F is the memory fraction, t is the number of tokens in the current context and B_F is the bookkeeping fraction. The memory fraction gives a handle on which vectors should be used with respect to the number of tokens and the distribution of S_{QK}. This is important since both the number of tokens and the distribution of S_{QK} will be constantly changing within the LLM. The bookkeeping fraction is used to determine which and how many of the first tokens' Value and Key vectors are used, further information on the bookkeeping fraction is given in section 4.1.

$$S_{QK} \geq \frac{1}{M_F * t}, S_{QK} \geq B_F \tag{4}$$

Splitting the original Key and Value tensor into a list of heads and filtering by equation 4 produced a list of Values and Keys whose length was the number of heads, where the number of tokens (t) in each head is filtered and flattened to a variable size t_F in the form [{t_F, d_k}].

A FAISS search[16] was performed on the filtered Keys and the top k Keys and Values from the search were produced in the form [{k, t_F, d_k}]. In Type 1 and 3 methods the found Keys and Values from the persistent memory search were concatenated with their associated original Key or Value in the form [{k + 1, t_F, d_k}].

The Keys were then matched with the filtered Querys and run through equation 5 on the zeroth dimension.

$$S_{knn} = Softmax(\frac{QK^T}{\sqrt{d_k}} * T_S) \tag{5}$$

Here T_S gives a handle to adjust the “influence” of the memory on the current output. The Softmax does not produce linear results with order of magnitude, thus a T_S above 1 places a greater “influence” on the higher ranked “memories” and a T_S below 1 spreads the “influence” across the “memories” more evenly.

The Value vectors were then multiplied by S_{knn} and summed along the zeroth dimension to create what will be called the Memory Vectors. In type 2 and 3 methods the original Value vectors and the Memory vectors were scaled and added together. Finally, the updated Values were scattered to their original positions in the full Value tensor replacing the original values and the attention module was completed with the updated Value tensor using equation 2.

2.4 Tunable Parameters:

There are a number of parameters that can be changed in order to explore the effect on the models’ output. To start with, there are the parameters described above in equation 4 (M_F and B_F) and equation 5 (T_S). However, there are also parameters associated with the FAISS search[21] and the choice of which layers and heads to implement the method on. The FAISS search of the Keys has two main parameters which are explored herein. They are the Keys k-nearest neighbours and the distance between the Key and stored nearest neighbour Key. It should be noted that FAISS returns the squared Euclidean distance. Given the way the distance is used within, the number line which the distance is reported on seemed somewhat arbitrary. Thus, the distances reported herein are in the form of the squared Euclidean distance.

3 Evaluation:

3.1 Methodology:

Experiments were carried out using 16 GB RAM on an AMD Ryzen 5700u. The base GPT-2[19] model with 117M parameters, 12 layers and a model dimension of 768 was used for the experiments as this was lightweight enough to enable batches of 100 runs for each data point in a reasonable amount of time. The box plots were produced by the default Matplotlib[22] parameters with the box indicating the first and third quartile, the indent indicating the median and the whiskers showing the quartile ± 1.5x the interquartile range.

The code for this work was adapted from nanoGPT.[23]

3.2 Sentiment Analysis:

VADER (Valence Aware Dictionary and sEntiment Reasoner) was used for sentiment analysis.[24] In particular the python package vaderSentiment.[25] VADER is described as a lexicon and rule-based sentiment analysis tool that was attuned to sentiments expressed in social media but still works well on texts from other domains.[24] The output of VADER includes positive sentiment, neutral sentiment, negative sentiment and a compound score. The sentiment scores are ratios of the text which fall into each category and thus should add up to 1. The compound score is VADER's attempt to combine the sentiment scores into a single score between -1 and 1, where -1 is extremely negative and 1 is extremely positive. For scoring, the initial context text was removed as this was not produced by the model. Within the sentiment box plots the positive sentiment is shown as orange, the neutral sentiment as light blue, the negative sentiment as red and the compound score as light green.

The prompt, or initial context, given to the model had a negative sentiment in order to initiate GPT-2 into a negative sentiment and thus maximize the difference for positive sentiment behaviour. Unless otherwise stated the below prompt was used as the initiator.

"Cats are horrible"

The Memory that was used with the above prompt was given in section 2.2.

3.3 Recall Analysis:

Sets of 100 random seeds with 1337 as the initial seed were run.

A paragraph from Wikipedia was chosen, and a question was contrived such that when the Wikipedia text was placed in the context before the question the unadulterated GPT-2 achieved a success rate of at least 50 of 100 tries/seeds and this success rate was higher than when the question alone was placed in unadulterated GPT-2.

4 Results and Discussion:

4.1 Persistent Memory Architecture:

As it might be expected there are aspects of the Transformer architecture and the repercussions from that architecture that need to be taken into account when designing a medium-term memory. This section outlines two of those.

4.1.1 Positional Encoding:

Positional encoding is used in the majority of Transformer architectures to encode information about the token position in a way that the Transformer can use. In GPT-2 this is done at the same time as token encoding. The token encoding and positional encoding are added together and then fed into the first Transformer layer. Since the persistent memory is generated by feeding text into the model, a large part of the information fed into layer 1 for both the persistent memory and the generated text will be positionally correlated. Hence, it would be expected that the *k*-nearest neighbours algorithm would choose the positionally similar vectors as opposed to contextually similar vectors. However, as mentioned in the introduction the information flow through the layers of a Transformer should dilute GPT-2's type of positional encoding relatively quickly, i.e. from 50% in layer 1 to 25% in layer 2 and 12% in layer 3. This appears to be illustrated by Figure 2.

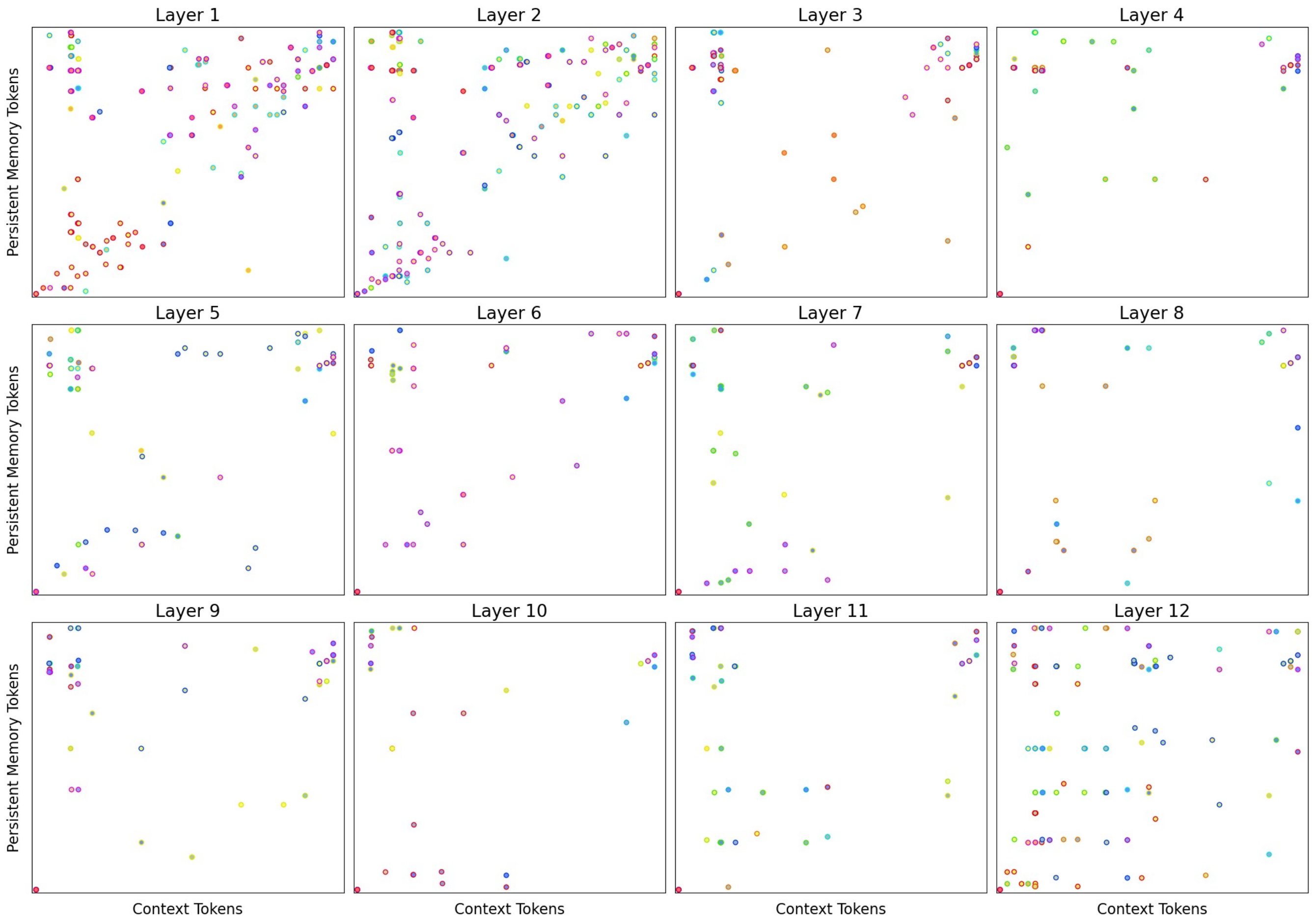


Figure 2: A plot of the nearest neighbour from the persistent memory with a filter of M_F=1.0 for each head of each token in the context. The outline colour of each point is the head and the inner colour is a heat map of the nearest neighbour distance with red representing the smallest distance and blue representing the largest distance between the persistent memory token and the context token. The plot is a snapshot of the 87th generated token when the initial context was "How did you". The persistent memory used is shown in section 2.2 and the memory creation was done as outlined in section 2.2.1. The Type 1 method was applied to all 12 layers and the parameter B_F set to 0.0, M_F set to 1.0, T_S set to 1.0 and k set to 1.

Figure 2 plots the top nearest neighbour from the persistent memory's tokens*heads for each head of the generated tokens where B_F was 0, M_F was 1, T_S was 1 and the context was initiated with *"How did you"*. The y-axes are the persistent memory's tokens (section 2.2.1) in order bottom to top, and the x-axes are the generated tokens in order from left to right. The outline colour for each point is the head and the inside colour is the nearest neighbour distance with red being the smallest distance and blue being the largest distance.

In Figure 2 the persistent memory and the generated text have a similar number of tokens and are plotted sequentially. Therefore, if a positional correlation between them exists it should manifest as clustering around a line with an approximate slope of one. As anticipated, layers 1 and 2 show significant clustering around this line. Moreover, the inner colour of the points within this clustering transition from orange in layer 1 to fuchsia in layer 2. This suggests that the nearest neighbour distance of the points in this clustering is increasing significantly, aligning with the hypothesis that the position encoding would be diluted in the residual relatively quickly. By Layer 6, any clustering around this line becomes hard to distinguish. In Figure 2 it is possible to squint and draw a line of slope 1 in all the layers. However, after layer 3 the patterns in the layers vary significantly with seed, context initiator and persistent memory, and thus any line drawn is not representative of a trend. Summing up, Figure 2 implies that positional encoding can largely be ignored when dealing with later layers in GPT-2.

4.1.2 Bookkeeping Token

A notable feature shown in Figure 2 is the data point with a red centre located in the lower left of layers 3-12. Although it cannot be discerned from Figure 2, the data point obfuscates a data point with a red centre for every head in those layers. This is what will be described within this paper as the bookkeeping token.

The red colour of the bookkeeping token indicates that the context token is very close in distance to the persistent memory token, or in other words, the vector representing the first token in the persistent memory and the first token in the generated text are nearly identical. This is not due to these tokens being initiated from a similar word, "*How*" is normally considered to be a different word to "*Cats*". Further emphasizing this is the orange centre of the first token in layer 1. This can be rationalized as the tokens in layer 1 representing 50% from the tokenization of the text and 50% from the positional encoding and hence the orange colour arises from the 50% positional encoding. This leaves the bookkeeping token as an artifact of the Transformer architecture. Since B_F is set to 0 in Figure 2, memories are added to it in every layer, however, it is still an artifact which is very similar across very different text.

Placing a token which will be very similar to the generated text token into a persistent memory would appear to be redundant and perhaps detrimental. Especially since the attention block places a priority through equation 1 and 4 on the bookkeeping token. This is shown by the generated text being very similar at short context lengths between when the method is activated versus unadulterated GPT-2. There is a very simple solution to this which is outlined in section 2.2.2. In short, place a newline character at the front of the text to be placed in memory, run it through GPT-2 and then remove the bookkeeping token before storing the memory.

4.1.3 Adding "Memories":

Removing the bookkeeping token from the persistent memory results in a situation where the attention block still places priority through equation 1 and 4 on the contexts' bookkeeping token, however, now "memories" are added to the bookkeeping token which are not bookkeeping tokens themselves. This leaves three possible techniques for adding "memories" to the model after being filtered by equations 1 and 4.

1. Only add "memories" to the contexts' bookkeeping token
2. Do not add "memories to the contexts' bookkeeping token

3. Add "memories" to both

Obviously, the ideal situation would be to only add "memories" to the bookkeeping token, as this would result in a medium-term memory which is independent of context length. However, as will be described in this paper, depending on a number of variables, this is at times very deleterious. When it is deleterious the solution is to use memory technique number 2 and add a newline character to the beginning of the context in order to ensure it is possible to add "memories" to the original initial token if they are indicated by equation 4.

4.2 Sentiment:

There have been a number of papers written on directing GPT-2 sentiment.[26–29] This paper takes a slightly different perspective on sentiment direction. Since it is an attempt to replicate intrinsic memory, the perspective is more akin to the difference in a persons' response to a mouse running across the diner table when they have one of the two below memories:

1. "*mice are disgusting vectors of disease*"
2. "*mice are so cute especially when being fed*"

While Gunasekar *et. al.*[30] have shown that curated training data leads to a well-behaved LLM, the ability to direct or "pattern" a LLM's response with human intelligible text does add a little extra reassurance. However ethereal that reassurance may end up being.[31]

As shown in Table 1, this method does generate text with a higher degree of positive sentiment than the unadulterated GPT-2 with a similar degree of text quality. The below table shows matching output pairs that were initiated with "*\nCats are horrible*" on the same random seed to generate 50 tokens. On the left unadulterated GPT-2 and on the right the Type 3 method with the residual scaled to 33% and the "memories" scaled to 66%, using the persistent memory given in section 2.2, created as outlined in section 2.2.2 and activated on the three layers 7, 8 and 9 with a B_F of 0.9, a M_F of 5, a T_S of 0.01 and a k of 5. Some newline markers ("*\n*") were deleted in the text to conserve space.

Table 1

Seed	GPT-2	GPT-2 + Type 3 Method
1337	Cats are horrible, as are tigers and elephants. The great majority of their population is confined to a small strip of forest. Few people seem to care for them. The only man-made predators are the big cats who creep up on the dogs and rats and hunt	Cats are horrible, you know. They are sweet, but they are also incredibly smart to tell you things. If a cat looks at a cat and says "Oh, cats…" or says "I like cats," it's a really good cat. Cats are really
2658	Cats are horrible parasites, but they're getting worse, and they're getting worse, and they're getting worse, so they're a nuisance. You will have to look at the numbers." The problem is that cat feces are kind of a nuisance - even	Cats are horrible at the feet of cats. They have great behavioral reactions when it comes to communicating with each other, so they're a great complement to the cat. They're so smart and so smart they can be easily reached if they're alone. They will chase
8932	Cats are horrible. They can talk through you in huge, noisy chunks of paper, which can cause injury or death. Their fur is razor-sharp. Their heart rate is too high, or at least, they have a tendency to keep going. They are terrifying	Cats are horrible. They can talk through fears, jealousy, and so on. They learn to love these animals, and they are absolutely loving to them. They love their cats, and as cats, they have a special bond with each other. They have
2179	Cats are horrible at the hands of strangers. But it's not just the cats that face a hostile crowd. Police are looking into the possibility of a "vicious aggression" that could be rife with cats. You can talk to your	Cats are horrible at the moment. Especially cats that were bred to be very specific about their cat. They're very kind and they like to stick together. They're a wonderful breed. They're able to be playful and social creatures. They're loving on both sides
9671	Cats are horrible at handling other animals and can be easily killed by any animal other than the cat. The researchers also found that the pain-relieving properties of lambs and other animals have a strong effect on the size of their tusks.	Cats are horrible at handling other animals and can be easily startled by the sight of these fluffy creatures. The researchers also found that the dogs don't have specific behavioral characteristics and are friendly to animals that are either cat friendly or cat-specific.

The first thing that is noticeable in the above table is that the text generated using the Type 3 method has a much more positive sentiment. The second thing is that it appears to maintain the topic on cats better than the unadulterated GPT-2. This becomes noticeable to a greater degree as the number of generated tokens increases. The third thing is that the text has a positive sentiment without being a copy of the persistent memory (section 2.2). The generated text appears to be quite strange at times, but this appears to be an aspect of GPT-2, as both the unadulterated and the method generate “strange” text. As will be shown in the next section, it is quite easy to generate text with a positive sentiment, but the text quality does vary. A method of determining text quality was not devised for this paper. However, it is believed that the majority of people reading this paper will have the necessary hardware to rapidly replicate the results and determine if they agree with the authors’ anecdotal and subjective comments on text quality.

4.2.1 Sentiment Tuning Parameters:

As mentioned in section 2.4, there are four main tuneable parameters, the bookkeeping fraction (B_F) and the memory fraction (M_F) which filter the tokens used *via* equation 4, the “influence” of the persistent memory on the output (T_S) given by equation 5 and the number of nearest neighbours, or “memories”, used (k). Figure 3 gives an initial look at these tuning parameters in regard to the number of layers and how the persistent memory was created (section 2.2).

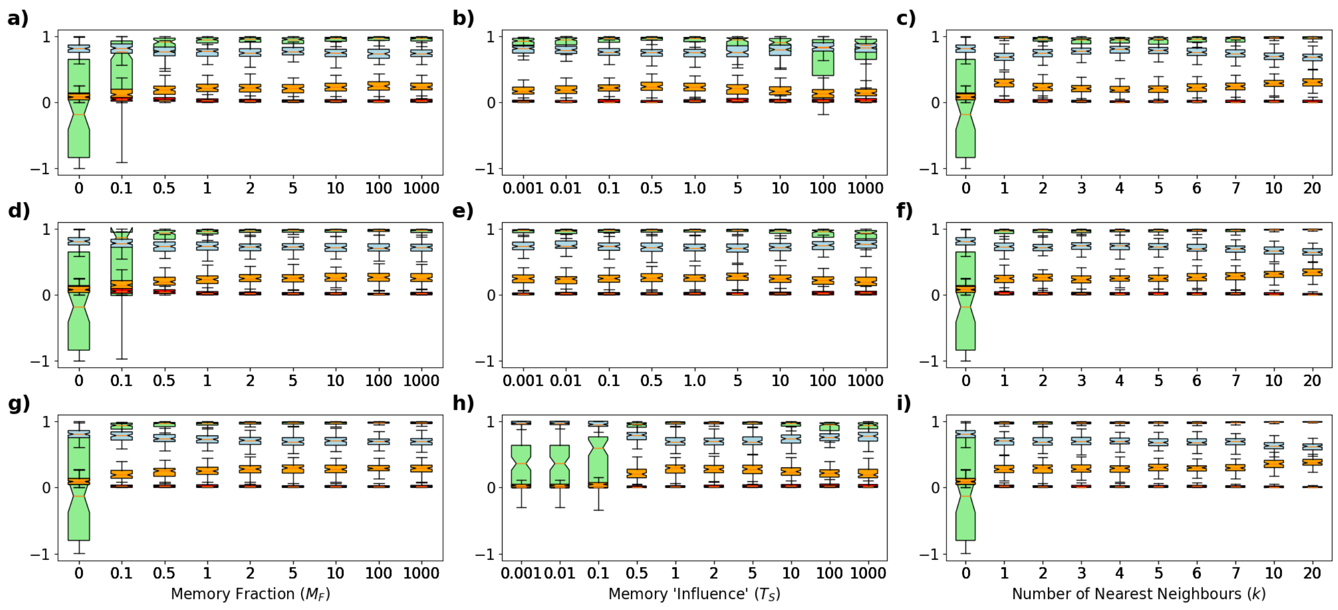


Figure 3: Box plots showing the distribution of sentiment calculated using VADER on generated text from 100 runs of 100 tokens generated by GPT-2 using the Type 1 method with the initial parameters of B_F set to 0.0, M_F set to 5.0, T_S set to 1.0 and k set to 2. The parameters are varied as indicated by the plots. In a-c the method was applied to all layers with a persistent memory created using method 2.2.1. In d-f the method was applied to layers 7-12 only. In g-l the method was applied to layers 7-12 only and the persistent memory was created using the method described in section 2.2.2. The method was not applied (unmodified GPT-2) to the box plots where the parameter is shown as 0.

The major takeaway from Figure 3 is that the method significantly tightens the VADER compound distribution and shifts the median to a very positive sentiment. Naturally not all generated text is created equal. This can be seen in Figure 3b where 12 layers were activated. In Figure 3b as T_S is decreased from a value of 1.0, VADER neutral increases and VADER positive decreases. The reason for this was as the T_S decreased, the model became more and more enthusiastic in repeating the word cat. This rather poor behaviour was probably due to including layers which are expected to be positionally correlated with the persistent memory or related to sentence structure. This was also the case in Figure 3h, indicating that adding memories directly to all the bookkeeping tokens does not result in high text quality. Keeping a T_S of 1 (the Softmax ranking GPT-2 was trained to use) appears to offset this problem in Figure 3. However, in the authors' opinion, the generated text still has some issues.

4.2.2 Memory Fraction:

In Figures 3a, 3d and 3g, the VADER compound distribution rapidly tightens and median shifts to very positive sentiment as the memory fraction, or fraction of tokens*heads this method is applied to, was increased. To give a better conceptualization of what this means for computation cost, Figure 4 plots the percent of tokens*heads this method was applied to for various memory fractions. The plot is a snapshot of the output for generating the 100th token of the seed 1337 (first data point for all box plots in Figure 3a). The plot data/line changes every-time a new token is generated, however, the trends remain approximately the same. For instance, the outer layers, 1, 2 and 12, generally have a significantly higher percentage than the centre layers and the centre layers remain at approximately the same percent. Thus, Figure 3a suggests that by applying a M_F of 1 or using 2 - 10% of the heads*tokens, a dramatic increase in positive sentiment distribution can be achieved. It also suggests that increasing the memory fraction above 5 (20 – 30% of heads*tokens) does not display a significant benefit.

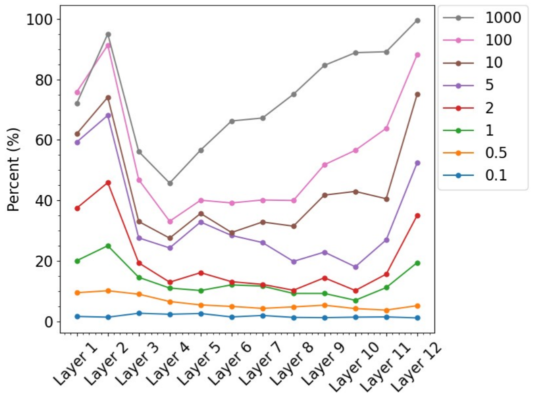


*Figure 4: The percent of heads*tokens which the method was applied to for each layer at various memory fractions (M_F)* for generating the 100th token generated with B_F set to 0.0, T_S set to 1.0 and k set to 2 and the seed set to 1337

4.2.3 Sentiment Adding "Memories":

As noted in section 4.1.3 and section 2.3, this paper separates the areas which "memories" are added to into the bookkeeping tokens and non-bookkeeping tokens. Figure 5 compares the effects of added "memories" to either of these or both on the VADER scores.

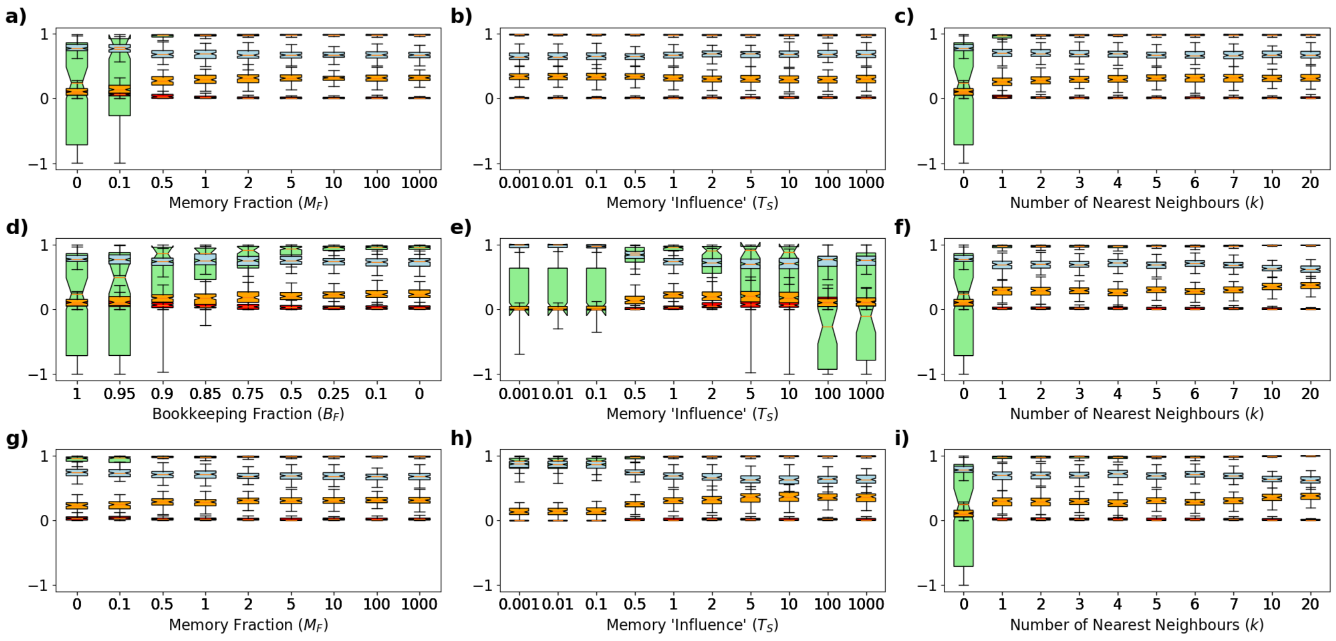


Figure 5: Box plots showing the distribution of sentiment calculated using VADER on generated text from 100 runs of 100 tokens generated by GPT-2 using the Type 1 method on layers 7-12 with the initial parameters of T_S set to 1.0 and k set to 5. The parameters are varied as indicated by the plots. In a-c B_F set to 1.0 and M_F set to 5.0. In d-f B_F set to 0.25 and M_F set to 0.0, . In g-i B_F set to 0.25 and M_F set to 5.0 .

From Figure 5 it can be seen that adding the bookkeeping token does increase the positivity of the VADER scores. However, as with Figure 3, it can also be seen from the memory influence plots that adding the bookkeeping token also increases the VADER Neutral which indicates the generated text is repeating itself. It should be noted, anecdotally, a bookkeeping fraction of 0.8 and higher does not noticeably degrade the text but does noticeably increase the positivity of the generated text. This is shown in Table 1 where a bookkeeping fraction of 0.9 was used.

4.2.4 Sentiment Layers:

As noted in the introduction it was hypothesized, given Meng *et al.*'s finding[15], that the addition of the Memory to only the final layers would be sufficient for recall. However, sentiment direction was anticipated to be slightly different from recall and this is what Figure 3 appears to show.

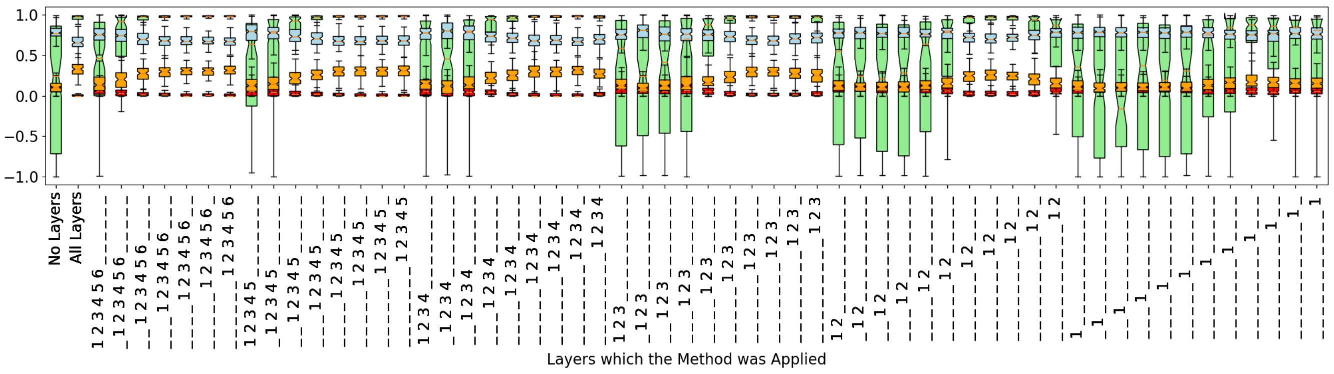


Figure 6: Box plots showing the distribution of sentiment calculated using VADER on generated text from 100 runs of 100 tokens generated by GPT-2 using the Type 1 method with the initial parameters of B_F set to 1.0, M_F set to 5.0, T_S set to 1.0 and k set to 5. Green denotes compound, orange denotes positive, red denotes negative and blue denotes neutral. Each character in the x-axis label indicates one of the 12 layers, the "_" symbol denotes the method was not applied to that layer and the numeral denotes a layer the method was applied to.

At first glance, Figure 6 appears to show that activating the first 6 layers negatively impacts the sentiment. However, a better description is probably that the best results stem from including layers in the range of 7-11 with activating layers 7, 8 and 9 providing the best results.

4.2.5 Sentiment Method Type:

In the Type 1 method shown in Figure 1 the "memories" are mixed with the original values and passed through equation 5. Due to equation 5, it could be envisaged that at large k values the generated text may lose track of the original context. The solution to this would intuitively involve using the original value vector as a residual. To this end the Type 2 and Type 3 methods were designed. Figure 7 shows the tuning parameter plots for the three types of methods where only layers 7, 8, 9 were activated and the parameter B_F was set to 1.0, M_F was set to 5.0, T_S was set to 1.0 and k was set to 5 when not being varied.

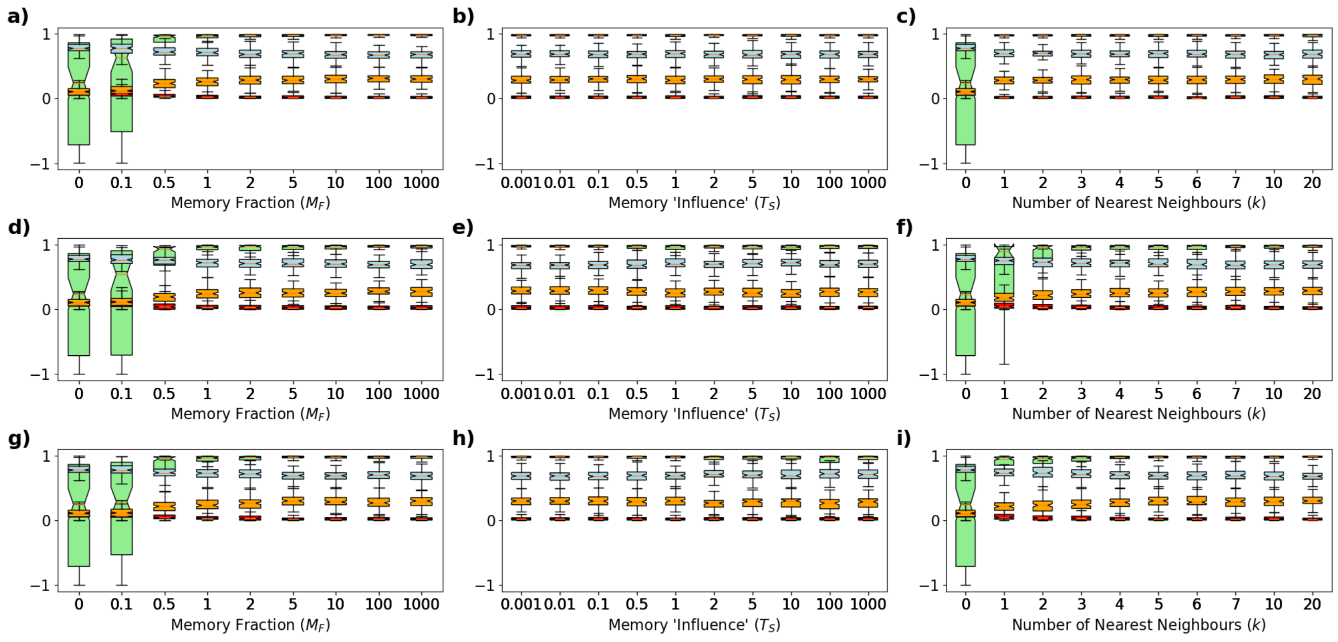


Figure 7: Box plots showing the distribution of sentiment calculated using VADER on generated text from 100 runs of 100 tokens generated by GPT-2 with the persistent memory shown in section 2.2 and created using the method outlined in section 2.2.2. The initial parameters were B_F set to 1.0, M_F set to 5.0, T_S set to 1.0 and k set to 5 with the parameters are varied as indicated by the plots. The method was applied to layers 7, 8 and 9 only. In a-c the method applied was Type 2 with the residual fraction set to 0.1. In d-f the method applied was Type 3 with the residual fraction set to 0.1. In g-i the method applied was Type 1. Green denotes compound, orange denotes positive, red denotes negative and blue denotes neutral. The method was not applied (unmodified GPT-2) to the box plots where the parameter is shown as 0.

From Figure 7 it appears that method type does not significantly change the VADER values. However, anecdotally, adding the residual appears to increase the quality of the generated text. Table 1 shows examples of text using the Type 3 method.

4.3 Persistent Memory Recall:

As noted in the introduction, it is the hypothesis of this work that the Keys of the attention block are stored associativity, which would in turn allow a *k*-nearest neighbours search to pull the appropriate items from the persistent memory. If this is correct and GPT-2 can pull the correct answer out of its context, it should also be able to pull it out of the persistent memory when the method is applied to appropriate layers. There are various addendums that need to be added to that statement and some of which will be discussed later but the obvious way to test memory recall is to iterate through the following stages:

1. Find a question that GPT-2 struggles to answer from its internal parameters
2. Ensure GPT-2 can answer the question when a paragraph containing the answer is entered into the context followed by the question.
3. Test to see if GPT-2 with the method activated can answer the question when the paragraph containing the answer is placed in the persistent memory.

Ideally statistics could be rapidly generated by running this through a database like SQuAD.[32] However, although GPT-2 readily clears stage 1 in SQuAD, it does not fare so well on stage 2, even when testing 100 seeds for every question. Given GPT-2's low success rate on SQuAD, in order to probe the method, it was unfortunately necessary to contrive a question for each Wikipedia paragraph such that GPT-2 could answer it under stage 2. This naturally does not lead to great statistics nor does it "feel" satisfactorily robust. Thus, this section will be focused on developing a rudimentary understanding of how recall from the persistent memory works in GPT-2, and it is left to the curious with access to a more capable model to test for “robustness”.

4.3.1 Persistent Memory Recall Example:

As with extrapolating from any single data point the usual disclaimers and warnings apply. It is also not clear if “fact” recall is the best term when using GPT-2, perhaps “most probable token” recall would be a better description. However, it is thought that the example may still provide some useful insights into how the persistent memory functions during recall and that a more powerful model might be able to amplify the more enticing aspects. The topic chosen was Vancouver and the persistent memory was created using the method outlined in section 2.2.2 from the first paragraph of Vancouver's Wikipedia page shown below.

> *"Vancouver is a major city in western Canada, located in the Lower Mainland region of British Columbia. As the most populous city in the province, the 2021 Canadian census recorded 662,248 people in the city, up from 631,486 in 2016. The Metro Vancouver area had a population of 2.6 million in 2021, making it the third-largest metropolitan area in Canada. Greater Vancouver, along with the Fraser Valley, comprises the Lower Mainland with a regional population of over 3 million. Vancouver has the highest population density in Canada, with over 5,700 people per square kilometre, and fourth highest in North America (after New York City, San Francisco, and Mexico City)."*

A section of the persistent memory that did not explicitly mention Vancouver was chosen and then rearranged in an attempt to specifically elicit Vancouver as the next word. The newline character was used to shift the text away from the bookkeeping token as well as to add a newline between the Wikipedia text and the question when adding the Wikipedia text into the context. The text designed to elicit Vancouver as the next word is shown below.

"\nThe name of the city with the third-largest metropolitan area in Canada is"

Using the above, GPT-2 achieved 12 out of 100 tries/seeds when the question alone was entered into the context and 50 out of 100 tries/seeds when the Wikipedia entry was added to the context before the question. Given the bookkeeping token, even though the first token is “V” and not “Vancouver” it was thought best to flip the two phrases in the first sentence for this section. Thus, the below text is what was used.

> "Located in the Lower Mainland region of British Columbia, Vancouver is a major city in western Canada. As the most populous city in the province, the 2021 Canadian census recorded 662,248 people in the city, up from 631,486 in 2016. The Metro Vancouver area had a population of 2.6 million in 2021, making it the third-largest metropolitan area in Canada. Greater Vancouver, along with the Fraser Valley, comprises the Lower Mainland with a regional population of over 3 million. Vancouver has the highest population density in Canada, with over 5,700 people per square kilometre, and fourth highest in North America (after New York City, San Francisco, and Mexico City)."

Using the above, GPT-2 achieved 15 out of 100 tries/seeds when the question alone was entered into the context and 53 out of 100 tries/seeds when the Wikipedia entry was added to the context before the question.

4.3.2 Recall Type 1 Method:

It was hypothesized in the introduction that the addition of the medium-term memory to only the final layers would be sufficient for recall. Figure 8 shows that is the case with small number of later layers being the optimum for the recall of word “Vancouver” from the persistent memory.

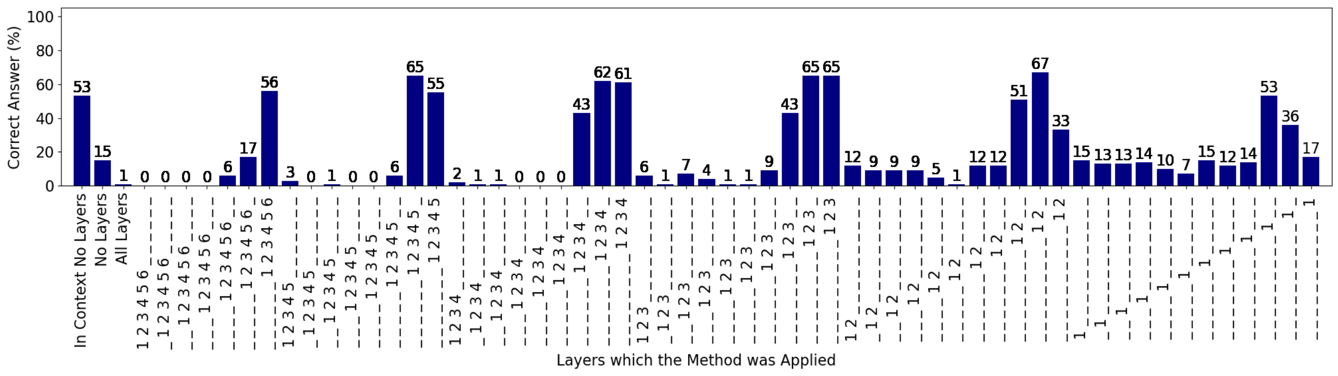


Figure 8: Bar plots of the number of times the method generates the correct answer (Vancouver) out of 100 tries/seeds for the example given in section 4.3.1 using the Type 1 method with the parameter B_F set to 0.1, M_F set to 1.0, T_S set to 1.0 and k set to 3 and using a persistent memory created using the section 2.2.2 method. Context No Layers refers to unadulterated GPT-2 with the Wikipedia text being placed in the context followed by the question, No Layers refers to unadulterated GPT-2 with only the question in the context and All layers refers to the method being activated on all the layers. For the rest, "_" refers to an unactivated layer and a number refers to an activated layer.

It can be clearly seen in Figure 8 that activating the method on layers 10 and 11 of GPT-2 achieves the best results. Furthermore, only activating layer 10 still provides similar recall rates to GPT-2 with the Wikipedia text in the context prior to the question.

As with every system the best way to get an understanding of it is to vary parameters. Figure 9 does this for the three "memory" injection techniques outlined in section 4.1.3, adding memory only to the bookkeeping token, not adding "memory" to the bookkeeping token and adding "memory" to both.

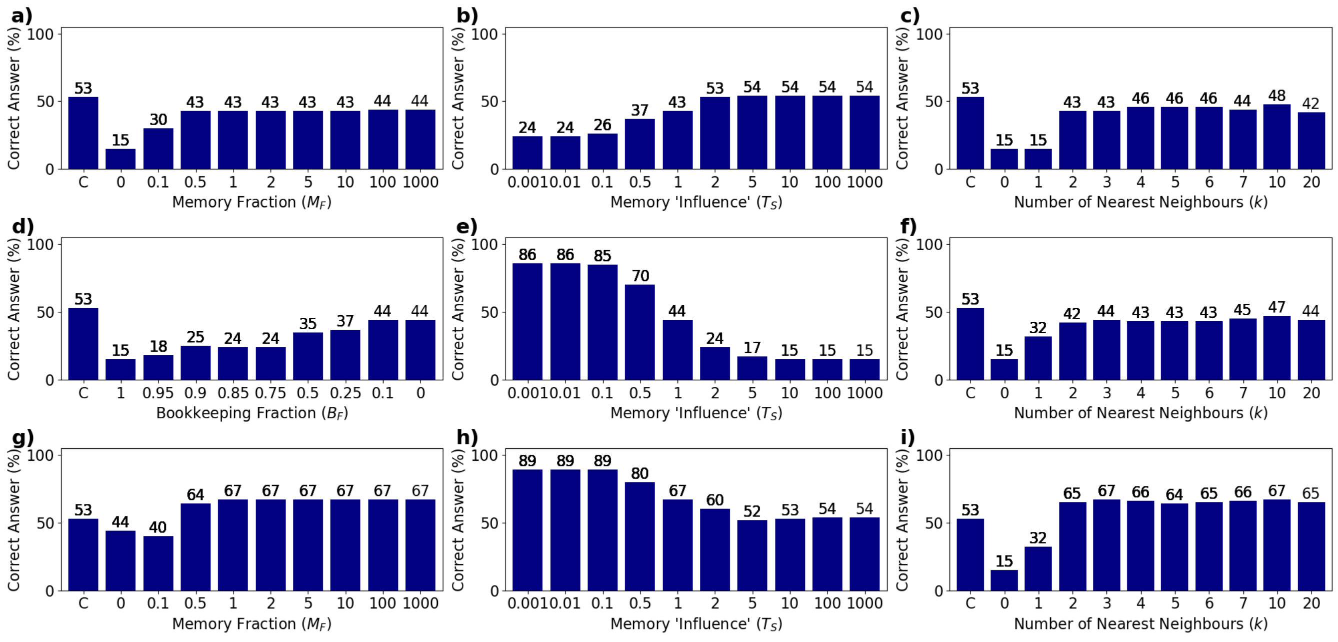


Figure 9: Bar plots of the number of times the method generates the correct answer (Vancouver) out of 100 tries/seeds for the example given in section 4.3.1 using a Type 1 method with the method activated on layers 10 and 11, the parameter T_S set to 1.0 and k set to 3 when not being varied. The persistent memory used was created using the section 2.2.2 method. On the x-axes "C" refers to unadulterated GPT-2 with the Wikipedia text being placed in the context followed by the question. In a-c B_F set to 1.0 and M_F set to 1.0 when not being varied. In d-e t B_F set to 0.1 and M_F set to 0.0 when not being varied. In g-i B_F set to 0.1 and M_F set to 1.0 when not being varied.

In Figure 9 both the memory fraction and the number nearest neighbours reach a point and then remain stable. The parameter T_S for Figure 9e and 9h on the other hand start at a high level and then drop to a stable level. In Figure 9b, where "memory" is only added bookkeeping token, that level is the same value as the control. For Figure 9h, where the “memory” is added to everything after being filtered by equation 4, that level is the same as Figure 9e where the “memory” was not added to the bookkeeping token.

4.3.3 Recall in Layer 10:

Figure 8 shows that it only requires layer 10 to produce decent recall results and Figure 9 shows that low memory fractions also produce good recall results. This gives the opportunity to study what is occurring with raw data on a scale humans can readily comprehend. The table below shows the output at layer 10 for a B_F of 0.0, a M_F of 1.0, a T_S of 1.0 and a k of 3. Since it is a snapshot of layer 10 prior to the application of the method, Table 2 shows the data which would be added to the context tokens under the three different “memory” injection techniques outlined in section 4.1.3 and is independent of the seed.

Table 2

Head	Context Index	Context Token	Memory Index	knn Distance	Memory Token	Equation 5
0	0	\n	[12, 54, 140]	[154, 169, 172]	[Vancouver, Metro,).]	[0.12, 0.09, 0.07]
1	0	\n	[140, 78, 52]	[147, 155, 164]	[)., ., .]	[0.14, 0.14, 0.14]
2	0	\n	[12, 33, 140]	[163, 165, 191]	[Vancouver, census,).]	[0.19, 0.17, 0.11]
3	0	\n	[140, 78, 52]	[266, 311, 312]	[)., ., .]	[0.12, 0.14, 0.17]
4	0	\n	[52, 78, 53]	[139, 139, 143]	[., ., The]	[0.1, 0.11, 0.11]
5	0	\n	[12, 53, 1]	[134, 142, 144]	[Vancouver, The, Located]	[0.19, 0.15, 0.26]
6	0	\n	[12, 54, 1]	[151, 158, 177]	[Vancouver, Metro, Located]	[0.09, 0.07, 0.06]
7	0	\n	[140, 78, 101]	[133, 149, 157]	[)., ., .]	[0.04, 0.05, 0.04]
8	0	\n	[12, 7, 6]	[146, 156, 159]	[Vancouver, region, land]	[0.19, 0.13, 0.15]
9	0	\n	[140, 12, 78]	[169, 186, 193]	[)., Vancouver, .]	[0.05, 0.07, 0.04]
10	0	\n	[140, 1, 78]	[307, 342, 352]	[)., Located, .]	[0.06, 0.13, 0.08]
11	0	\n	[31, 1, 12]	[127, 130, 131]	[2021, Located, Vancouver]	[0.12, 0.08, 0.09]
0	2	name	[1, 11, 52]	[107, 116, 125]	[Located, ,, .]	[0.2, 0.06, 0.27]
3	2	name	[1, 11, 7]	[64, 79, 80]	[Located, ,, region]	[0.04, 0.68, 0.18]
10	2	name	[7, 11, 12]	[74, 76, 80]	[region, ,, Vancouver]	[0.02, 0.37, 0.26]
0	5	city	[16, 42, 25]	[31, 42, 44]	[city, city, city]	[0.15, 0.27, 0.11]
2	5	city	[7, 42, 12]	[58, 59, 68]	[region, city, Vancouver]	[0.07, 0.13, 0.47]
3	5	city	[6, 7, 12]	[40, 44, 44]	[land, region, Vancouver]	[0.12, 0.16, 0.58]
5	5	city	[16, 6, 12]	[24, 35, 36]	[city, land, Vancouver]	[0.16, 0.08, 0.41]
3	12	area	[7, 6, 16]	[44, 45, 46]	[region, land, city]	[0.04, 0.02, 0.7]
3	14	Canada	[10, 19, 12]	[23, 30, 42]	[Columbia, Canada, Vancouver]	[0.07, 0.71, 0.16]
5	14	Canada	[19, 10, 77]	[16, 21, 25]	[Canada, Columbia, Canada]	[0.11, 0.15, 0.36]
7	14	Canada	[19, 10, 9]	[27, 28, 36]	[Canada, Columbia, British]	[0.2, 0.19, 0.14]
8	14	Canada	[19, 10, 77]	[15, 26, 29]	[Canada, Columbia, Canada]	[0.13, 0.56, 0.05]
3	15	is	[13, 14, 2]	[46, 69, 73]	[is, a, in]	[0.53, 0.11, 0.01]
5	15	is	[2, 11, 29]	[34, 35, 36]	[in, ,, ,]	[0.2, 0.07, 0.05]
10	15	is	[13, 11, 14]	[33, 57, 58]	[is, ,, a]	[0.21, 0.33, 0.23]

There are a few things in Table 2 that help support the veracity of this method as a potential candidate for a medium-term memory. First of all, the Memory Indexes (the indexes of the tokens within the Wikipedia text chosen by the k-nearest neighbours algorithm) are not sequential, nor do they appear to be correlated with the Context Index. This suggest that it is not positionally correlated, agreeing with the conclusion in section 4.1.1. Secondly, the output from equation 5 does not appear to be correlated with the k-nearest neighbours distance. This suggests that the premise behind using k-nearest neighbours to pull “associative memories” from the persistent memory and then using equation 5 to rank them is working. Finally, excluding the bookkeeping token, the Memory Tokens chosen by k-nearest neighbours from the Context Token appear to be human associable.

This is not to say that the value vectors are the same, this can be seen in head 0 for the Context Token “city” where it chooses three different “city” tokens from the memory of which equation 5 ranks them very differently. Or in head 2 where both “city” and “Vancouver” are chosen from the persistent memory but even though they are relatively close in distance and “city” is nearer in distance, equation 5 ranks “Vancouver” significantly higher. This appears, to the author at least, another indication that the “memories” are stored associatively and that *k*-nearest neighbours algorithm is able to extract the appropriate “memory”.

In section 4.3.2 it was noted that Figure 9b at high T_S values reached a level which was the same value as the control. The reason for this can be seen in the outputs from equation 5 for the bookkeeping token, they are all very low values. The equation 5 output for the residual is omitted from Table 2 for reasons of clarity, but since outputs of a Softmax must sum to 1 the residual token will be much larger than the outputs shown. Since high T_S values amplify these differences, the “memories” for the bookkeeping tokens will approach 0. Thus, at high T_S, the results in Figure 9b reach a level at the same value as the control and in Figure 9h a level the same as Figure 9e. This is not an ideal situation, as high T_S values amplify the attention blocks “preferred memory”.

4.3.4 Recall Type 2 Method:

The Type 2 method was designed to surmount the problem of the residual dominating the added “memories” in equation 5 at high T_S values. Figure 10 shows the required layers for the Type 2 method with a high T_S value.

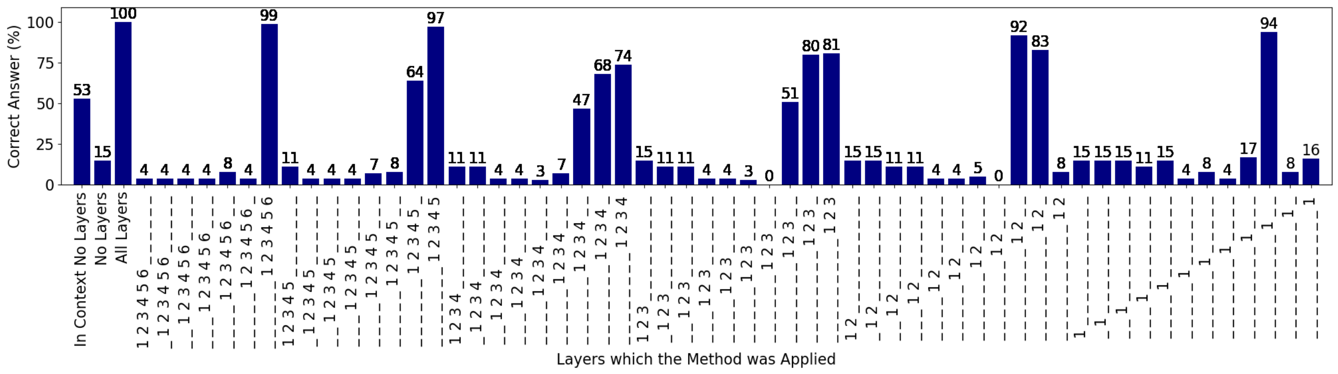


Figure 10: Bar plots of the number of times the method generates the correct answer (Vancouver) out of 100 tries/seeds for the example given in section 4.3.1 using the Type 2 method with the parameter B_F set to 0.9, M_F set to 5.0, T_S set to 1000.0 and k set to 5 and using a persistent memory created using the section 2.2.2 method. Context No Layers refers to unadulterated GPT-2 with the Wikipedia text being placed in the context followed by the question, No Layers refers to unadulterated GPT-2 with only the question in the context and All layers refers to the method being activated on all the layers. For the rest, “_” refers to an unactivated layer and a number refers to an activated layer.

From Figure 10 it can be clearly seen that only layer 10 is required to produce a high rate of recall. The tuning parameters for the Type 2 method are shown in Figure 11.

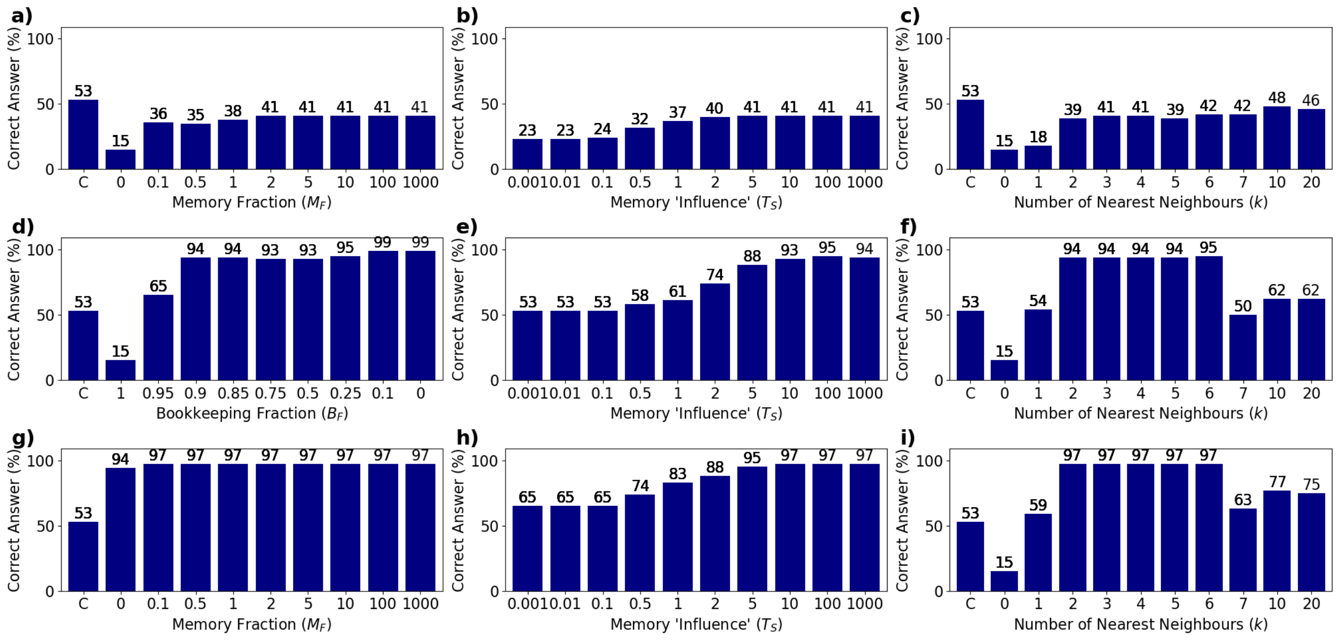


Figure 11: Bar plots of the number of times the method generates the correct answer (Vancouver) out of 100 tries/seeds for the example given in section 4.3.1 using the Type 2 method with the residual scaled to 0.1, the method activated on layer 10, the parameter T_S set to 1000.0 and k set to 3 when not being varied. The persistent memory used was created using the section 2.2.2 method. On the x-axes "C" refers to unadulterated GPT-2 with the Wikipedia text being placed in the context followed by the question. In a-c B_F set to 1.0 and M_F set to 5.0 when not being varied. In d-e t B_F set to 0.9 and M_F set to 0.0 when not being varied. In g-i B_F set to 0.9 and M_F set to 5.0 when not being varied.

In comparing the Type 1 method in Figure 9 to the Type 2 method with high T_S values in Figure 11, the Type 2 method clearly generates a higher rate of recall while using a smaller number of both tokens*heads and layers. It was noted previously that using a high proportion of bookkeeping tokens produces poor text quality. However, as noted in section 4.2.5 a bookkeeping fraction greater than 0.8 does not generate text which is noticeably poorer in quality than no bookkeeping tokens. As shown in Figure 11d, a bookkeeping fraction of 0.9 is all that is required for a high rate of recall.

It should be noted that high T_S values essentially select one "memory" for each context token*head through equation 5. This can be seen in Table 3 where the equation 5 column shows two 0.0 and one 1.0 for each context token*head selected. This means that it is possible to get a change in result with a change in k, which is shown in Figures 11f and 11i. However, this is a "preference" choice by the models' internal parameters amplified by T_S.

Table 3 shows the bookkeeping tokens*heads selected by equation 4 when B_F is set to 0.9, M_F is set to 0.0, T_S is set to 1000 and k is set to 3.

Table 3

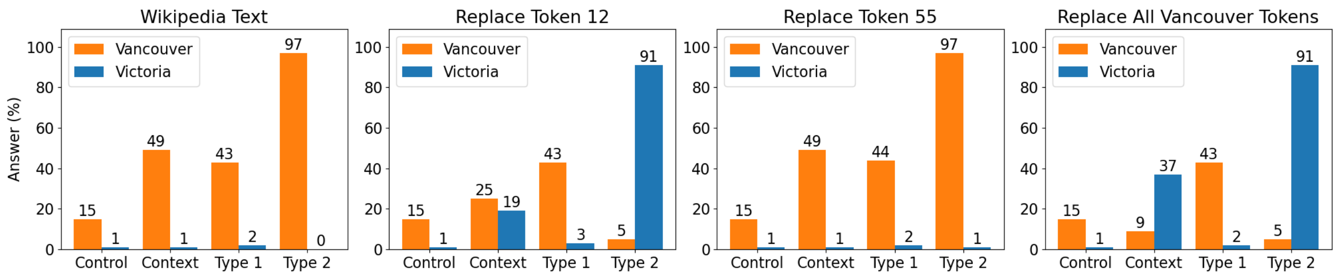
Head	Equation 1	Memory Index	knn Distance	Memory Token	Equation 5
1	0.9299	[140, 78, 52]	[147, 155, 164]	[)., ., .]	[0.0, 0.99, 0.0]
4	0.9316	[52, 78, 53]	[139, 139, 143]	[., ., The]	[0.0, 1.0, 0.0]
6	0.9206	[12, 54, 1]	[151, 158, 177]	[Vancouver, Metro, Located]	[1.0, 0.0, 0.0]
9	0.9550	[140, 12, 78]	[169, 186, 193]	[)., Vancouver, .]	[0.0, 1.0, 0.0]
11	0.9656	[31, 1, 12]	[127, 130, 131]	[2021, Located, Vancouver]	[1.0, 0.0, 0.0]

Although Vancouver is only selected by equation 5 twice out of the five tokens*heads used, this is clearly enough to shift the ranking to Vancouver in the majority of the seeds/tries. It should probably be emphasized here that only 5 tokens*heads *k*-nearest neighbours searches out of a possible 2160 were required in order to generate the expected answer. Further, since only the Bookkeeping token was used, the number of tokens*heads *k*-nearest neighbours searches does not appear likely to change significantly with an increase in the number of tokens in the context.

4.3.5 Focus of the Attention Block:

In Table 2, for the non-bookkeeping tokens there does appear to be an obvious trend in the Attention blocks choices of context tokens and the associated *k*-nearest neighbours search results. The main context tokens chosen are "name", "city", "Canada" and "is" and the *k*-nearest neighbours search results are mostly focused on the persistent memory phrase "*Vancouver is a major city in western Canada*". For instance, the main *k*-nearest neighbours search results for the context token "is" are 11, 13 and 14 which all surround the persistent memory token for "Vancouver" (token 12) within that phrase. Further, all instances of the *k*-nearest neighbours searches which produced a "Vancouver" token in Table 2 and 3 were token 12. This, suggests that the method is actually getting the question wrong by pulling its answer from the wrong sentence. However, this appears to be a GPT-2 issue and not an issue with the method.

To test if GPT-2 with the Wikipedia text in the context focuses on the same area as the method, select "Vancouver" tokens were replaced with "Victoria" tokens (the second-biggest city in British Columbia, Canada) in the Wikipedia text. The results are shown in Figure 12.



Panel	Bar	Control	Context	Type 1	Type 2
Wikipedia Text	Vancouver	15	49	43	97
Wikipedia Text	Victoria	1	1	2	0
Replace Token 12	Vancouver	15	25	43	5
Replace Token 12	Victoria	1	19	3	91
Replace Token 55	Vancouver	15	49	44	97
Replace Token 55	Victoria	1	1	2	1
Replace All Vancouver Tokens	Vancouver	15	9	43	5
Replace All Vancouver Tokens	Victoria	1	37	2	91

Figure 12: Replace selected "Vancouver" tokens with "Victoria" in the Wikipedia text outlined in section 4.3.1 for the Control which was the unadulterated GPT-2 with the question alone entered into the context, the Context which included the Wikipedia text prior to the question in the context of an unadulterated GPT-2, the Type 1 method, which was activated of layers 10 and 11 with B_F set to 1.0, M_F set to 5.0, T_S set to 1.0 and k set to 3 and the Type 2 method with the memory scaled to 0.9, the residual scaled to 0.1 and activated on layer 10 with B_F set to 0.9, M_F set to 5.0, T_S set to 1000 and k set to 5.

Token 55 is the "Vancouver" token which should be directly connected to the question. As can be seen from Figure 12 replacing token 55 with "Victoria" does not change answer compared to no substitution in either the method or when the Wikipedia text is placed in the context. This suggests that Token 55 does not play a large part in generating the expected answer. The substitution of Token 12 with "Victoria" on the other hand, for both placing the Wikipedia text in the context and the Type 2 method does meaningfully change the answer to "Victoria". Further, although not shown in Figure 12 replacing only the 3rd or 4th instance of "Vancouver" in the Wikipedia text gives similar results as replacing Token 55. Thus, although replacing all instances of "Vancouver" produces the best result when the Wikipedia text is placed in the context, Token 12 probably plays an out-sized role.

A notable feature of Figure 12 is that the Type 1 method generates "Vancouver" significantly more times than the control, independent of whether "Vancouver" appeared in the persistent memory. It is thought that the Wikipedia text used herein was also used in training GPT-2, and the Type 1 method amplifies its ranking from the internal parameters. Something similar to this is probably occurring during sentiment direction. The “resistance” of GPT-2 to deviate substantially from the patterns in its training data can also be seen in the next section on substitution.

4.3.6 Type 2 Method and Token Substitution:

In order to form a picture of how well the Type 2 method reads from its persistent memory all the “Vancouver” tokens were replaced with a substitute in the Wikipedia text. In Figure 13 the Type 2 method (Figure 13a) is compared to unadulterated GPT-2 when the substituted Wikipedia text is placed in the context (Figure 13b).

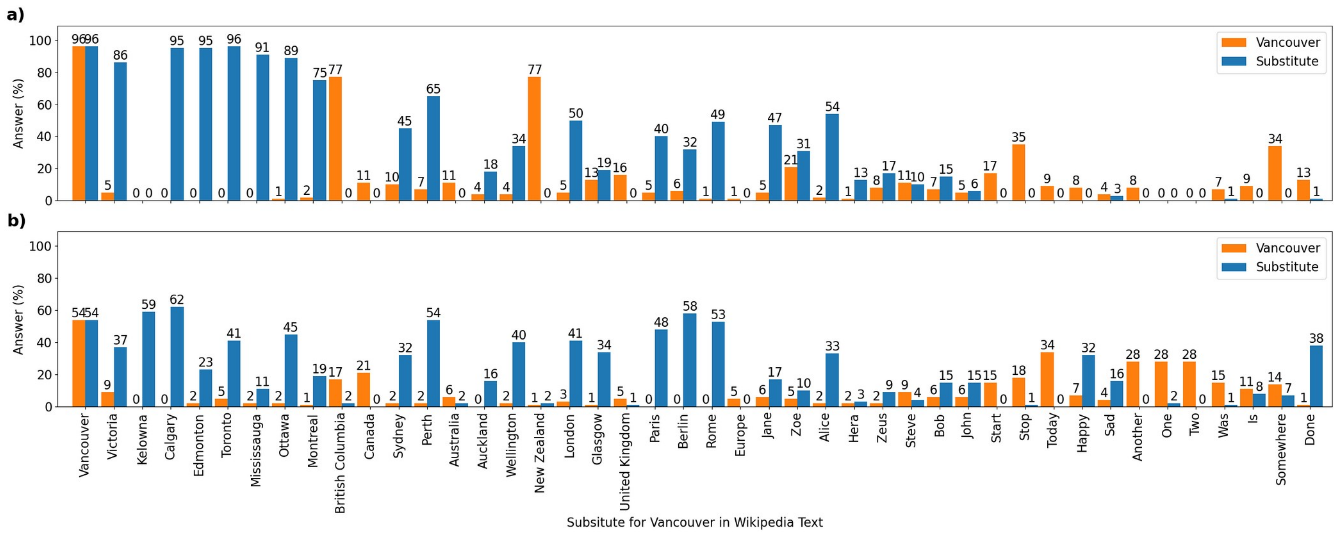


Figure 13: Percent Vancouver and the substitute were generated when all of the “Vancouver” tokens in the Wikipedia text were replaced by the substitute for a) the Type 2 method with no memories added to the bookkeeping token, the memory scaled to 0.9, the residual scaled to 0.1 and activated on layer 10 with B_F set to 0.9, M_F set to 0.0, T_S set to 1000 and k set to 5. b) The Wikipedia text placed in the context prior to the question

As noted in the previous section GPT-2 appears to have a set of patterns which it follows, if the substitution is outside this pattern, GPT-2 does not give the expected answer. It can be seen from Figure 13 that both the Type 2 method and when the Wikipedia text is placed in the context follow a similar pattern in recall rates. For instance, it can be seen that both techniques in Figure 13 have trouble recalling verbs, numbers and most human name substitutes. One particularly specific pattern is both techniques fail to recall country and province name substitutes. A divergence between the two techniques patterns is the Type 2 method appears to have a much higher recall rate for Canadian cities over international ones. This may largely be due to the base ranking given by GPT-2, the control, which the method adds memories to. The frequency of the first tokens generated by the control is shown in Figure 14.

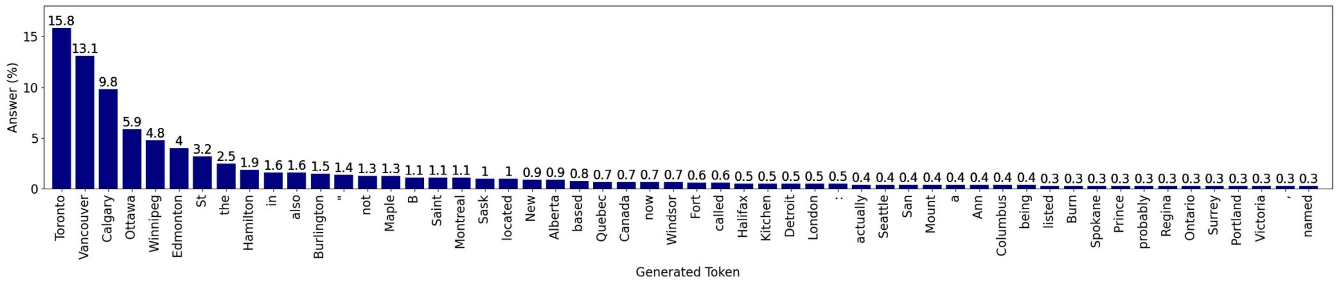


Figure 14: Percent of times the token was generated as the first token when GPT-2 was run for 1000 triess/seeds for the question outlined in section 4.3.1

The top tokens given by the control are also included in the Canadian cities in Figure 13, however, there are notable exceptions. For instance, in Figure 14 “Victoria” has a lower frequency of generation than “Canada” and Figure 14 does not include “Mississauga”. This implies that the persistent memory is imparting a pattern on the output outside the base ranking.

A glaring omission in the Type 2 methods ability to recall Canadian cities is “Kelowna”, the third-largest city in British Columbia. It could be argued that this is due to “Kelowna” being composed of three tokens. However, this is unlikely as it fails on the second token “own” and there are several two token names in Figure 13 which have high recall rates, such as “Mississauga”.

4.3.7 Recall Generated Text:

The text generated using the Type 2 method produced text at a quality similar to unadulterated GPT-2 with a bookkeeping fraction greater than 0.8. Although this is dependent on if the substitute is within GPT-2s’ “pattern”, if it is not, there appears to be a tendency for the word to be repeated. Word repetition can be mitigated by increasing the fraction of the residual scaling or in some cases increasing the memory fraction above zero.

Table 4 uses the “Toronto” substituting “Vancouver” version of the Wikipedia text to compare the Wikipeda text in the context to the Type 2 method with the same parameters as Figure 13 (residual scaled to 0.1, B_F set to 0.9, M_F set to 0.0, T_S set to 1000 and k set to 5). It uses the same seeds as Table 1 and some newline markers (“\ *n*”) were deleted to conserve space.

Table 4

Seed	GPT-2 + Context	GPT-2 + Type 2 Method
1337	The name of the city with the third-largest metropolitan area in Canada is Toronto. The city of Toronto is located within the eastern border of the United States encompassing both Ontario (Canada) and Canada's Northern border (Canada). The city has a population of approximately 300,000, making the city the fourth largest city outside	The name of the city with the third-largest metropolitan area in Canada is Toronto. The city of Toronto was established in 1857 and the capital of Canada until 1999. Toronto has a population of about 1.6million. It is a UNESCO world heritage site. This article is licensed under a Creative
2658	The name of the city with the third-largest metropolitan area in Canada is Vancouver, British Columbia. The city has a population of 1.2 million in 2021, up from 1.7 million in 2016. The Greater Vancouver area, along with the Fraser Valley region, comprises Greater Vancouver, B.C., with a population of	The name of the city with the third-largest metropolitan area in Canada is Toronto, but some regionalities have more name-specific names in Toronto and Toronto Center. The key to a national map is the location of the entire city. The American census results show that Greater Toronto has no name-specific geographic region, if
8932	The name of the city with the third-largest metropolitan area in Canada is Toronto, which is home to the Toronto skyline. The city is located right next to the Ontario Provincial Mall, a major tourist attraction. The Toronto skyline was unveiled by Toronto Mayor Rob Ford, at the Toronto Auto Show. The mayor said Toronto's skyline	The name of the city with the third-largest metropolitan area in Canada is Toronto, or a city in the Toronto region. That's because it's an urban center, not a city in a region of cities. It was developed by the Toronto Public Policy Opinion Project and is located in a large, wealthy area of urban
2179	The name of the city with the third-largest metropolitan area in Canada is Maple Ridge, Saskatchewan. As the eighth largest city in Canada, the city has a population of 20.2 million in the city, up from 18.2 million in 2016. The region has the second-biggest population of Quebecers (6.	The name of the city with the third-largest metropolitan area in Canada is Toronto. The 2014 Census of Toronto released in September 2016 estimates that Toronto has about 34.4 million people in the city, with one of the fastest growing populations in the city. But the number is much smaller than those of Calgary or
9671	The name of the city with the third-largest metropolitan area in Canada is the Imperial Toronto. The Imperial Toronto, composed mainly of the city of Toronto, is located in the heart of Toronto, surrounded by the Bayview River, Burrard Inlet, the Island, the Mississauga River, and the GTA (including East York	The name of the city with the third-largest metropolitan area in Canada is Toronto. It is the gateway to another city, Toronto-Camperance, and the largest city in the Toronto region. But the city's popularity and its stature in the mainstream media has been tempered by recent efforts by some of Canada's richest

5 Future Work:

The sentiment section of this paper was set out as a bit of a novelty to personify GPT-2 as having an intrinsic memory, but also, more constructively, to get an understanding of the tuning parameters. It may find a use in alignment with the greater specificity of the increased number of layers in larger models, but it is impossible to tell how useful or even if it can be useful from experiments using GPT-2.

Although this paper was successfully able to find a word which GPT-2 focused on in the context and show that the method also focused on that word, the recall ability of the method *really* needs to be tested on a model that is actually capable of question answering! The method may also benefit by being fully integrated into a more sophisticated model. If that works, enabling continuous extension of the persistent memory as well as developing offline techniques for the persistent memory to “consolidate” and “forget” would probably be the appropriate next steps.

6 Conclusion:

This paper demonstrated that Type 1 and Type 3 methods are able to increase the VADER sentiment positivity of GPT-2 by using a persistent memory. It also showed that if the persistent memory of the Type 2 method is placed in the context of unadulterated GPT-2, the type 2 method searches a similar region of its persistent memory as unadulterated GPT-2 searches in the context. On a specific example it showed that the Type 2 method is able to generate the expected answer more reliably and significantly faster than GPT-2 with the persistent memory in the context. It also found that the optimum layers for sentiment direction were layers 7, 8 and 9 which were different for the optimum layer for recall which was layer 10. This suggests that the intrinsic memory activation may be separate from explicit memory activation in GPT-2.

Given the limitations of GPT-2, it was impossible to demonstrate that the method described herein will act as an effective medium-term memory in more sophisticated models. However, it is believed that the data provided does suggest it is worth further investigation.

Code is posted at https://github.com/MTMTransformer/MTMTransformer

(1) *Memory*. Wikipedia. https://en.wikipedia.org/w/index.php?title=Memory&oldid=1188368490 (accessed 2023-12-05).

(2) OpenAI. GPT-4 Technical Report. arXiv March 27, 2023. http://arxiv.org/abs/2303.08774 (accessed 2023-07-20).

(3) Beltagy, I.; Peters, M. E.; Cohan, A. Longformer: The Long-Document Transformer. arXiv December 2, 2020. http://arxiv.org/abs/2004.05150 (accessed 2023-07-11).

(4) Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q. V.; Salakhutdinov, R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. arXiv June 2, 2019. http://arxiv.org/abs/1901.02860 (accessed 2023-07-11).

(5) Kitaev, N.; Kaiser, Ł.; Levskaya, A. Reformer: The Efficient Transformer. arXiv February 18, 2020. http://arxiv.org/abs/2001.04451 (accessed 2023-07-11).

(6) Rae, J. W.; Potapenko, A.; Jayakumar, S. M.; Lillicrap, T. P. Compressive Transformers for Long-Range Sequence Modelling. arXiv November 13, 2019. https://doi.org/10.48550/arXiv.1911.05507.

(7) Bulatov, A.; Kuratov, Y.; Burtsev, M. S. Recurrent Memory Transformer. arXiv December 8, 2022. https://doi.org/10.48550/arXiv.2207.06881.

(8) Bulatov, A.; Kuratov, Y.; Kapushev, Y.; Burtsev, M. S. Scaling Transformer to 1M Tokens and beyond with RMT. arXiv February 6, 2024. https://doi.org/10.48550/arXiv.2304.11062.

(9) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. arXiv December 5, 2017. http://arxiv.org/abs/1706.03762 (accessed 2023-07-11).

(10) Voita, E.; Talbot, D.; Moiseev, F.; Sennrich, R.; Titov, I. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*; Korhonen, A., Traum, D., Màrquez, L., Eds.; Association for Computational Linguistics: Florence, Italy, 2019; pp 5797–5808. https://doi.org/10.18653/v1/P19-1580.

(11) Vig, J.; Belinkov, Y. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*; Linzen, T., Chrupa\la, G., Belinkov, Y., Hupkes, D., Eds.; Association for Computational Linguistics: Florence, Italy, 2019; pp 63–76. https://doi.org/10.18653/v1/W19-4808.

(12) Nelson Elhage; Neel Nanda; Catherine Olsson; Tom Henighan; Nicholas Joseph; Ben Mann; Amanda Askell; Yuntao Bai; Anna Chen; Tom Conerly; Nova DasSarma; Dawn Drain; Deep Ganguli; Zac Hatfield-Dodds; Danny Hernandez; Andy Jones; Jackson Kernion; Liane Lovitt; Kamal Ndousse; Dario Amodei; Tom Brown; Jack Clark; Jared Kaplan; Sam McCandlish; Chris Olah. *A Mathematical Framework for Transformer Circuits*. https://transformer-circuits.pub/2021/framework/index.html (accessed 2023-12-08).

(13) Geva, M.; Caciularu, A.; Wang, K. R.; Goldberg, Y. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. arXiv October 12, 2022. http://arxiv.org/abs/2203.14680 (accessed 2023-07-11).

(14) Geva, M.; Schuster, R.; Berant, J.; Levy, O. Transformer Feed-Forward Layers Are Key-Value Memories. arXiv September 5, 2021. http://arxiv.org/abs/2012.14913 (accessed 2023-07-11).

(15) Meng, K.; Bau, D.; Andonian, A.; Belinkov, Y. Locating and Editing Factual Associations in GPT. arXiv January 13, 2023. http://arxiv.org/abs/2202.05262 (accessed 2023-07-11).

(16) Johnson, J.; Douze, M.; Jégou, H. Billion-Scale Similarity Search with GPUs. arXiv February 28, 2017. https://doi.org/10.48550/arXiv.1702.08734.

(17) Wu, Y.; Rabe, M. N.; Hutchins, D.; Szegedy, C. Memorizing Transformers. arXiv March 16, 2022. https://doi.org/10.48550/arXiv.2203.08913.

(18) Bertsch, A.; Alon, U.; Neubig, G.; Gormley, M. R. Unlimiformer: Long-Range Transformers with Unlimited Length Input. arXiv October 30, 2023. https://doi.org/10.48550/arXiv.2305.01625.

(19) Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models Are Unsupervised Multitask Learners; 2019.

(20) *Microsoft Bing*. Wikipedia. https://en.wikipedia.org/w/index.php?title=Microsoft_Bing&oldid=1169110640 (accessed 2023-08-14).

(21) Johnson, J.; Douze, M.; Jégou, H. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* **2019**, *7* (3), 535–547.

(22) Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **2007**, *9* (3), 90–95. https://doi.org/10.1109/MCSE.2007.55.

(23) Karpathy, A. nanoGPT, 2023. https://github.com/karpathy/nanoGPT (accessed 2023-08-12).

(24) Hutto, C.; Gilbert, E. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media* **2014**, *8* (1), 216–225. https://doi.org/10.1609/icwsm.v8i1.14550.

(25) Hutto, C. J. vaderSentiment: VADER Sentiment Analysis. VADER (Valence Aware Dictionary and sEntiment Reasoner) Is a Lexicon and Rule-Based Sentiment Analysis Tool That Is Specifically Attuned to Sentiments Expressed in Social Media, and Works Well on Texts from Other Domains. https://github.com/cjhutto/vaderSentiment (accessed 2023-08-11).

(26) Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; Liu, R. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. arXiv March 3, 2020. http://arxiv.org/abs/1912.02164 (accessed 2023-07-26).

(27) Krause, B.; Gotmare, A. D.; McCann, B.; Keskar, N. S.; Joty, S.; Socher, R.; Rajani, N. F. GeDi: Generative Discriminator Guided Sequence Generation. arXiv October 22, 2020. http://arxiv.org/abs/2009.06367 (accessed 2023-07-26).

(28) Liu, A.; Sap, M.; Lu, X.; Swayamdipta, S.; Bhagavatula, C.; Smith, N. A.; Choi, Y. DExperts: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*; Association for Computational Linguistics: Online, 2021; pp 6691–6706. https://doi.org/10.18653/v1/2021.acl-long.522.

(29) Yang, K.; Klein, D. FUDGE: Controlled Text Generation With Future Discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Online, 2021; pp 3511–3535. https://doi.org/10.18653/v1/2021.naacl-main.276.

(30) Gunasekar, S.; Zhang, Y.; Aneja, J.; Mendes, C. C. T.; Del Giorno, A.; Gopi, S.; Javaheripi, M.; Kauffmann, P.; de Rosa, G.; Saarikivi, O.; Salim, A.; Shah, S.; Behl, H. S.; Wang, X.; Bubeck, S.; Eldan, R.; Kalai, A. T.; Lee, Y. T.; Li, Y. Textbooks Are All You Need. arXiv October 2, 2023. https://doi.org/10.48550/arXiv.2306.11644.

(31) Three Laws of Robotics. *Wikipedia*; 2023.

(32) Rajpurkar, P.; Zhang, J.; Lopyrev, K.; Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv October 10, 2016. https://doi.org/10.48550/arXiv.1606.05250.