Reverse-Engineering the Retrieval Process in GenIR Models

Anja Reusch
anja@campus.technion.ac.il
Technion – Israel Institute of Technology
Haifa, Israel

Yonatan Belinkov
belinkov@technion.ac.il
Technion – Israel Institute of Technology
Haifa, Israel

Abstract

Generative Information Retrieval (GenIR) is a novel paradigm in which a transformer encoder-decoder model predicts document rankings based on a query in an end-to-end fashion. These GenIR models have received significant attention due to their simple retrieval architecture while maintaining high retrieval effectiveness. However, in contrast to established retrieval architectures like cross-encoders or bi-encoders, their internal computations remain largely unknown. In this work, we investigate this retrieval mechanism and uncover the roles played by different model components (self-attention, cross-attention, MLPs) and their interaction to generate the document identifier. First, we show that the pre-trained encoder, which was not fine-tuned for retrieval, is sufficient for the retrieval process. Then, we find that the pass through the decoder can be divided into three stages: (I) the priming stage in which no component contributes query-specific information, (II) the bridging stage where cross-attention transfers query information from the encoder to the decoder, and (III) the interaction stage where MLPs process this transferred information to predict the document identifier in the last layer. Our results indicate that document-specific information is only stored in a few components in the final stage of the retrieval process. We hope that our findings will motivate the development of more effective GenIR models and facilitate their improvements.[1]

CCS Concepts

• **Information systems** → **Language models**; **Retrieval models and ranking**.

Keywords

Neural Information Retrieval, Generative Information Retrieval, Interpretability

ACM Reference Format:
Anja Reusch and Yonatan Belinkov. 2025. Reverse-Engineering the Retrieval Process in GenIR Models. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3730076

[1]Code and Models can be found here: https://github.com/technion-cs-nlp/re-gen-ir.

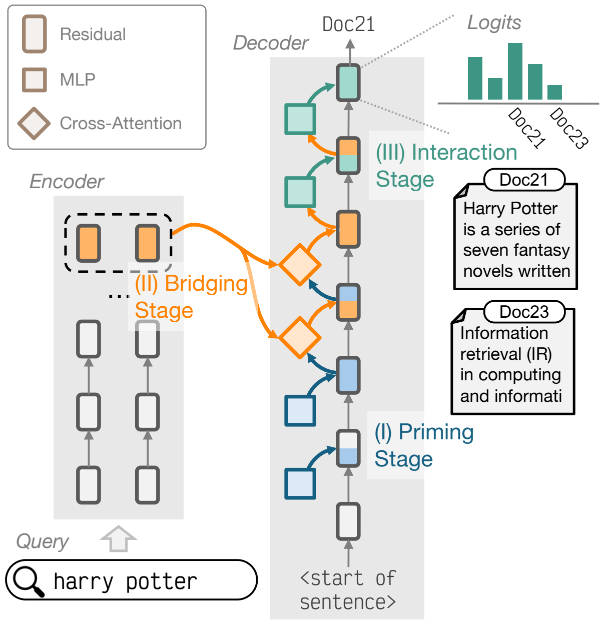

Figure 1: A simplified view of the retrieval process in the GenIR models in this work. After the encoder processes the query, the decoder operates in three stages: (I) the Priming Stage, where MLPs activate to prepare the residual stream triggering the cross-attention components in (II) the Bridging Stage, which transfer query information from the encoder to the decoder's residual stream; and (III) the Interaction Stage, in which MLPs process the query information from the cross-attention to promote relevant document logits.

1 Introduction

Generative Information Retrieval (GenIR) [20] trains a neural model to associate a query with a document identifier that satisfies the information need expressed within the query [29]. This approach applies transformer models (usually an encoder-decoder model) as an end-to-end retrieval system. The introduction of this new retrieval paradigm has received much attention [1, 17, 22, 24, 35, 39], due to the simplified architecture and the fact that only one pass (or a few passes) through the model is required. Despite these advantages, GenIR models are commonly applied as a black box, obscuring the process by which they perform retrieval.

In contrast, past research on cross-encoders [2, 37] and dense retrieval models [6, 27, 33] has shed light on how these models judge the relevance of a document given a query. Another line of research focuses on the inner workings of transformer decoder models

(LLMs), which have recently been popularized due to their success in text generation. Here, research has identified key components that are crucial for performing certain tasks (e.g., [5, 10, 11, 34]) and used these insights to improve model performance [18, 19]. Despite these efforts, their insights are not applicable to GenIR models due to differences in training objective and model architecture.

Therefore, this work studies the retrieval process within GenIR models. We investigate the following questions: Which components are responsible for predicting the correct document identifier, and how do they interact with each other? First, we show that the GenIR encoder can be replaced with encoders that were not trained on the target documents while still leading to successful retrieval by the trained decoder (Sec. 4). This finding indicates that the encoder does not need to be specialized for the retrieval dataset or task, suggesting that retrieval happens primarily in the GenIR decoder. Then, we analyze the pass through the decoder with three types of methods (Sec. 5): (a) causal interventions, where we modify activations of different components to identify which components are crucial for retrieval; (b) vocabulary projections, where we project activations to the vocabulary space, which includes document identifies, to measure which components directly contribute to the ranking of the document identifier; and (c) interaction studies, where we assess how different components interact to perform the final ranking.

Our findings are summarized in Fig. 1. We discover that the pass through the decoder can be grouped into three stages: (I) the priming stage, which is query-agnostic, but contributes important information for the activation of components in later layers; (II) the bridging stage, in which cross-attention components—triggered by the components in Stage I—transfer query information from the encoder to the decoder; and (III) the interaction stage, which is the first time that document-specific information interacts with query information. We demonstrate that this flow exists in several GenIR models trained on Natural Questions [16] and TriviaQA [12].

The main contributions of this work are summarized as follows:

- Reverse-engineering the retrieval process in GenIR models,
- Identifying the role and interaction of MLPs and cross-attention within the retrieval process, and
- Release of our code and trained models for further analyses.

2 Background

2.1 Insights into Language Models

A large body of work has attempted to uncover the internal mechanisms of Transformer models [2, 5, 10, 11, 18, 32, 34]. This field, known as mechanistic interpretability, is concerned with reverse engineering how language models implement different functionalities. A prominent view in this field is the residual stream view as introduced by Elhage et al. [5]. Here, the pass through the transformer model can be seen as a sequence of operations that read from and write to a stream of vectors called the residual stream. The idea is that model components like attention or multilayer perceptrons (MLPs) take the residual stream as input and add their output back to it. Each model component contributes to this residual stream in this view. Fig. 2 visualizes this idea in the context of GenIR.

Subsequent work identified sub-graphs of a model's computation graph, called circuits, which are responsible for solving a certain task, e.g., indirect object identification [19, 34], mathematical [10] or reasoning tasks [11]. To locate relevant model components, many of these studies apply mechanistic interpretability methods referred to as "activation patching" or "causal mediation analysis" [7, 18, 32, 34]. The core idea is to swap the activations of model components with different activations and measure the impact of this swap. For example, to analyze the prompt "The Eiffel Tower is in __", we replace the activation of each model component one by one with the activations for the prompt "The Colosseum is in __". If the model could still predict "Paris" after replacing a component's activation, that component was not responsible for the prediction. If the prediction switched to "Rome", the component was crucial for the model's computation of this particular output. Doing so enables us to identify all components crucial for a certain task, and reverse-engineer the circuit responsible for the model's behavior.

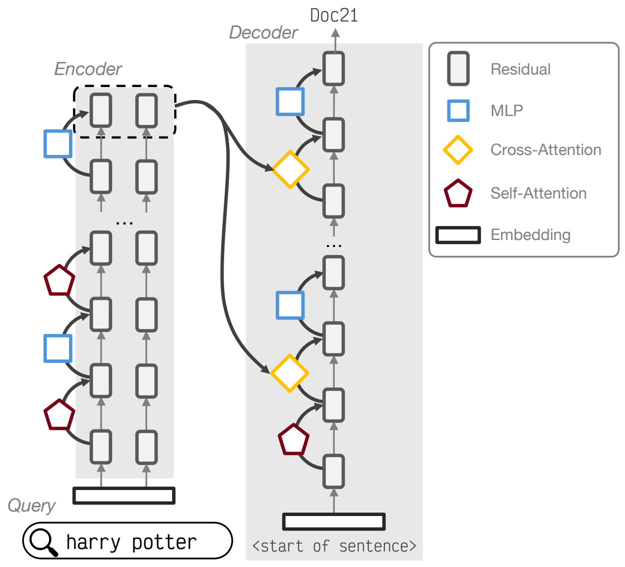

Figure 2: Overview of a transformer encoder-decoder for GenIR, depicted as the residual stream to which components read and write.

However, the majority of this recent work analyzes decoder-only models and does not study retrieval scenarios. Since the prevailing approaches in information retrieval focus on encoder-only or encoder-decoder models, which are trained for different objectives, techniques, and insights from these works may not be directly applicable. A few recent studies started working on language models in the context of retrieval: [33] analyze how dense retrievers apply common retrieval properties, [2] apply activation patching in a cross-encoder scenario, [27] show that dense retrievers implement query expansion, [6] performed an analysis of ColBERT [15], and [37] analyzed a cross-encoder. While these studies led to profound insights into the retrieval behavior of transformer models in cross- or bi-encoder scenarios, none of them dealt with GenIR or even encoder-decoder models. Therefore, to the best of our knowledge, this is the first work to study GenIR models mechanistically.

2.2 Generative Information Retrieval

Generative Information Retrieval (GenIR) refers to the relatively new paradigm of using a transformer model in an end-to-end fashion to rank documents given a query. The model, usually an encoder-decoder,[2] is trained to associate the content of documents with their document identifiers. This approach was popularized by Tay et al. [29], who provided a proof-of-concept that their approach, DSI, is able to outperform comparable dense retrieval models. DSI fine-tuned T5 encoder-decoder models [26] in a multi-task fashion: They applied an indexing phase where the model predicts the associated document identifier given the document input, and a retrieval phase where the model gets a query as its input and outputs the document identifier. Document identifiers were represented as either a new atomic token or as a sequence of tokens. As we focus on atomic document identifiers, we do not cover other versions and refer the reader to [29] for further details on them.

Fig. 2 depicts a GenIR model. The input to the encoder is a query of N tokens. The encoder itself employs the same architecture as popular encoder-only models such as BERT and outputs embedding vectors $e_1, \dots, e_N$. The decoder contains cross-attention components in each layer in addition to attention and MLPs, which are also found in decoder-only models. The input to the decoder is a [start-of-sentence] token. In each layer, three components add information to the residual stream: attention, cross-attention, and MLPs [31]. Each MLP component consists of two feed-forward layers $\mathrm{FF}^{\mathrm{proj}}$, and $\mathrm{FF}^{\mathrm{out}}$ connected by a ReLU non-linearity:

$$\mathrm{MLP}(r) = \sum_i \mathrm{FF}_i^{\mathrm{out}} \cdot a(r)_i, \text{ with } a(r)_i = \mathrm{ReLU}(\mathrm{FF}_i^{\mathrm{proj}} \cdot r), \qquad (1)$$

where i denotes the ith neuron, $a(\cdot)_i$ its activation, and FF_i the ith column/row of the linear layer weight matrices. T5 does not use bias terms in feed-forward layers. Eq. 1 allows us to view the activation as a dot product of the input to the MLP, i.e., residual stream hidden state r, and each neuron's row in $\mathrm{FF}^{\mathrm{proj}}$, which gets passed through the ReLU activation. Since the ReLU replaces negative values with 0, only positively activated neurons contribute to the MLP output.

Each cross-attention head has two inputs: the residual stream r and the output of the encoder $e_1, \dots, e_N$. These vectors get multiplied by the key matrix W_K forming the key vectors $k_1, \dots, k_N$. Similarly, the residual stream r is multiplied by the query matrix W_Q forming the query vector q:

$$W_K \cdot (e_1, \dots, e_N) = (k_1, \dots, k_N); \quad W_Q \cdot r = q$$

The dot product of query and keys results in similarity scores s_i for each key k_i with regard to the query q:

$$(s_1, ..., s_N) = q^T \cdot (k_1, ..., k_N)$$

In the calculation of the cross-attention head, these similarity scores are transformed by a softmax with additional scaling:

$$(a_1, \dots, a_N) = \text{scaled-softmax}(s_1, \dots, s_N)$$

These cross-attention scores a_i serve as weights in the output o: $o = \sum_i^N a_i \cdot v_i$, where $v_i = W_V \cdot e_i$ is a value vector. Each cross-attention component consists of H heads, each of which has separate W_Q, W_K, and W_V, resulting in one output vector o_h per head h. These output vectors are concatenated and multiplied by the output matrix W_O, resulting in the cross-attention output:

$$\text{Cross-Attention}(r) = [o_1, \dots, o_H] \cdot W_O$$

The (self-)attention component works in a similar way, but instead of using the encoder output as key and value vectors, it uses the decoder residual stream as keys and values. Since the residual stream consists of only one vector (because in the case of *atomic* GenIR models, the decoder receives only one token as its input), self-attention only transforms the residual stream vector linearly and adds it back to the residual stream.

After the pass through all layers, the residual stream is multiplied by an "unembedding matrix" $W_U \in \mathrm{R}^{d_V \times d_{\mathrm{model}}}$, which maps the residual stream $r \in \mathrm{R}^{d_{\mathrm{model}}}$ to the dimensionality of the vocabulary d_V: $W_U \cdot \sqrt{d_{model}} \cdot \mathrm{ReLU}(r) = l$. The output $l \in \mathrm{R}^{d_V}$ is a vector of logits for each token in the vocabulary. Since we adopt the atomic document identifier approach from DSI, we receive one logit for each document identifier along with the logits of the encoder-decoder model's word vocabulary (henceforth: "non-document identifiers"). These logits can be converted to a probability distribution over the vocabulary by using the softmax.

Following the success of DSI, other GenIR versions emerged [1, 22, 35, 39, 40]. Most of these adaptations aim to solve the underlying issue of GenIR models: the limited corpus size. An extensive comparison of GenIR variants [24] found that GenIR models perform worse compared to traditional methods when scaled to corpora of over eight million documents. Recently, also questions on the robustness of GenIR models [17] and the ranking quality [3, 38] have arisen. Establishing a better understanding of the inner workings of GenIR models might help in solving these issues.

3 Training and Analysis Setup

Since prior work did not publish trained GenIR models, we start by training models based on the DSI setup [29]. Our models use atomic IDs, direct indexing with a maximum token length of 32, and an indexing-to-retrieval ratio of 32. We fine-tune T5-large models on Natural Questions [16] in three different scenarios, with 10k and 100k unique documents (queries on 1k documents for each validation and test set, the rest for training), and the entire Natural Questions dataset denoted by NQ320k. To verify our findings in another scenario, we additionally train two T5-large models on Trivia-QA [12], one using the same scenario as DSI (Trivia-QA) and one without a dedicated indexing stage, but instead using generated queries provided by [35] (Trivia-QA QG). We use the data splits provided by the data set. We set all hyperparameters according to the ones reported in [29] and train using Huggingface Transformers [36]. Patching experiments as well as logit analyses are performed using TransformerLens [21]. The models' performance on our test sets can be viewed in Tab. 1. Our results are in line with those reported by previous work [24, 29].

4 The Role of the Encoder in GenIR

Generative IR models usually employ a transformer encoder-decoder architecture. The decoder is responsible for generating the output token depending on what was processed by the encoder. Therefore, the decoder needs to store some information about the document

[2]Recent work [28] used Llama 2 [30], a decoder model. There, the model generates the entire passage, which is closer to the text generation objective of decoder-only models compared to other GenIR approaches, where document identifiers are generated.

Table 1: Results on the respective test set for the models trained in this work.

Dataset	Hits@1	R@5	Hits@10
NQ10k	40.4	58.1	63.9
NQ100k	22.6	42.8	50.2
NQ320k	20.0	39.8	48.0
Trivia-QA	60.4	64.2	82.2
Trivia-QA QG	49.8	51.7	67.0

tokens in order to output the correct token. However, what is the role of the encoder? Is information about the corpus on which the model was trained encoded in the weights of the encoder?

To investigate this, we design an experiment where the encoder is trained on fewer documents than the decoder and test if retrieval can still be performed using this "incomplete" encoder. In total, we train four additional GenIR models on NQ10k, but omit ten different documents (taken from the validation set) when training each model. Then, we use the "full" GenIR model, which was trained on all NQ10k documents (including the 4×10 documents), but replace its encoder with an encoder of a model that was trained with ten missing documents. We feed the query of the removed documents to this "hybrid" encoder-decoder to measure how well the correct document can still be predicted. If information on the documents is stored exclusively in the encoder, the hybrid model, with an encoder that was not trained on these ten particular documents, should fail to correctly retrieve the removed documents. Before swapping the encoder, we measure the performance of the four models (with their own encoders and decoders). They obtain comparable performance to the "full" model, which was trained on all documents.

Results. As shown in Tab. 2, all four models are able to retrieve the removed documents with high performance. For most documents (70 - 80%), the models are able to rank the removed document at Rank 1, as demonstrated by the high Hits@1, even though the encoder was not optimized for these documents during training. Motivated by this high performance, we performed a follow-up experiment and swapped the encoder of the full model with the pre-trained T5-large encoder. The T5 encoder did not receive any GenIR training. When performing retrieval on all queries for which the full model was previously able to place a relevant document at Rank 1, the average rank of the relevant documents drops to 21.66. For most queries, the relevant document is still placed at Rank 1. Only a few documents are ranked higher than 1000.

Table 2: Results of the Encoder-Swapping Experiment. Models RD I–IV were each trained on 10 documents less ("removed documents", RD); the full model was trained on the entire corpus.

	Full	RD I	RD II	RD III	RD IV
MRR	1	0.867	0.754	0.853	0.820
Hits@1	10	8	7	8	8

Since the models in both experiments are still able to retrieve the documents even though their encoder was not trained to retrieve them, we conclude that the documents are not exclusively encoded in the encoder. We will later see that the information transferred from the encoder to the decoder is mostly similar to non-document identifiers, which also strengthens the argument that information on document identifiers is stored in the decoder. Potentially, the encoder's role lies in semantically encoding the query whose information is then moved to the decoder. Thus, the decoder holds the key for performing retrieval. In the next section, we therefore take a deeper look at the components of the decoder.

5 Identifying Components Crucial for Retrieval

The decoder consists of three components: the self-attention, the cross-attention, and the multilayer perceptrons (MLPs). Each model component reads from the residual stream and adds its output back to it. How much each component adds back to the residual stream can be traced by looking at the vector norm of each component output and comparing it to the vector norm of the residual stream. Since in our atomic document id scenario, only one token is generated, the residual stream of our decoder consists of only one vector as well. For each component c, $c \in \{$Attention, Cross-Attention, MLP $\}$, we calculate its relative contribution to the residual stream:

$$\text{ratio}_c = \frac{\|c_{\text{out}}\|_2}{\|r_\Delta\|_2}, \text{ with } r_\Delta = r_{\text{end}} - r_{\text{begin}},$$

where r_Δ is the difference of the residual stream before and after the layer and c_{out} denotes the output of component c that is added to the residual stream. This ratio only indicates the magnitude of the contribution of each component, but not whether it adds or subtracts this vector (since the signs of the vector elements could be inverse to the residual stream). Therefore, we also calculate the angle by computing the cosine similarity between the component outputs c_{out} and the residual stream at the beginning of each layer. We aggregate the results of all queries from the validation set for which the model predicted a relevant document at Rank 1 (denoted henceforth as "correct queries").[3]

Results. Fig. 3 (top) shows the ratio$_c$ of each model component per layer. We see a similar progression for all models: In the beginning, the MLPs dominate, then cross-attention contributes most (or more than before), and in the end, the MLPs contribute most to the change in the residual. When viewing the cosine similarity (Fig. 3, bottom), we see that towards the end, all components receive a negative cosine similarity. We suspect that in this part, components remove information from the residual stream, as their output is directed to the opposite side of the residual stream. Further inspection revealed that the output vectors of these components in this last part indeed have a high negative cosine similarity with components from the first part (lower than -0.9 cosine similarity between the output of each component in the last part and the MLP output of the first part, calculated on the correct queries using NQ10k).

Based on our observations, we divide the plots into three stages: Stage I (Layers 0–6), where MLPs contribute the most, adding to

[3]It is common practice to only evaluate over correctly generated results, as we want to investigate here how the model performs retrieval "correctly". Using queries for which the retrieval failed would also be of interest, as it might point to erroneous behavior the model might have learned, but such analyses are out of scope for this work.

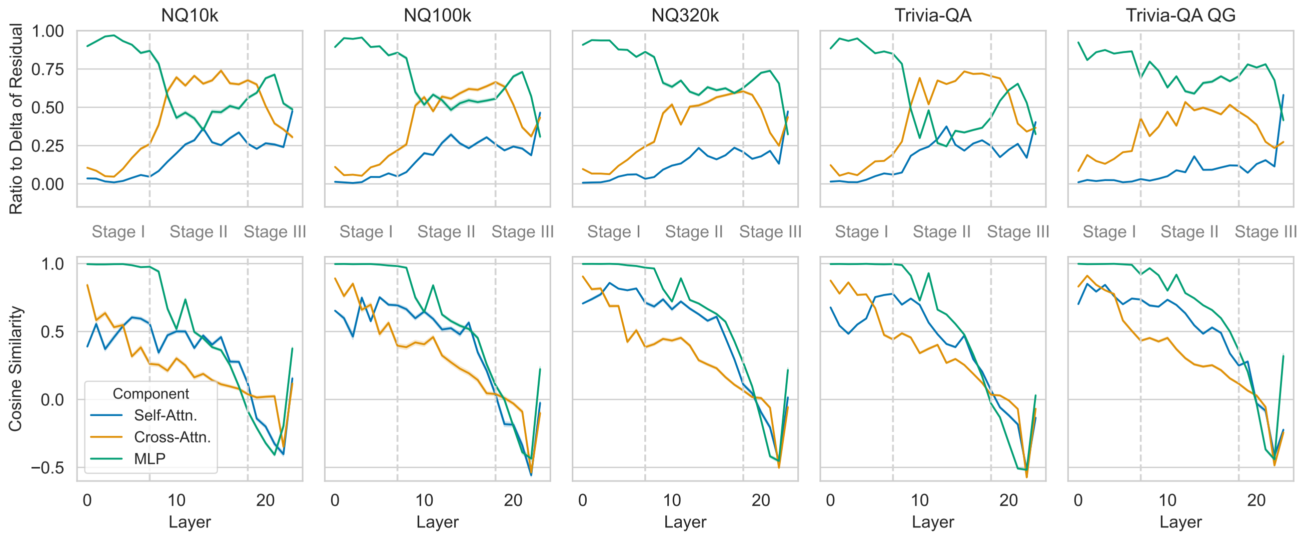

Figure 3: Proportion that each component's output contributes to the change in the residual stream (top) and cosine similarity of each component output with the layer output (bottom), displayed per layer. The models follow a similar trend: high MLP contribution in Stage I and III, cross-attention peaks in Stage II, the cosine similarity of all components is negative in Stage III.

the residual stream, Stage II (Layers 7–17) where cross-attention receives the highest contribution (positive), and Stage III (Layers 18–23), where MLPs contribute most again, but attention and cross-attention as well. In the last stage (except for the last layer), the contribution vector of each component is negative with respect to the residual stream and the first stage output, suggesting that components in this stage remove information initially contributed by Stage I MLPs. These results indicate which components might be essential in which part of the decoder, as a higher contribution might point to a higher importance. Nevertheless, we do not know whether these components are involved in predicting the output. Next, we will therefore causally verify their individual importance.

5.1 Contribution of Individual Components

To investigate whether a model component has a causal influence on the prediction of the model, we apply a variant of activation patching [7, 32]: We replace the output of a certain component at a specified position by a zero vector, i.e., removing the component from the computation. Since past research found that this practice might lead to noisy results [34], we also apply mean patching, where we replace a component's output with the mean output of this component aggregated over correct queries. We measure the effect of our intervention by calculating the rank of the first relevant document after applying the patching, aggregated over all queries where the correct document was originally placed at Rank 1. If the model could not perform retrieval after a component was replaced by a zero vector, we conclude that this component is necessary for the retrieval process. If the model could still perform retrieval after replacing the same component by the mean of the correct queries, this component might be crucial for the retrieval process, but its role is not specific to a certain query, but rather general. In these patching experiments, we use the models trained as described in Sec 3. We do not re-train models using the indicated components. We patch the three stages we previously identified. In each stage, we patch each component in all layers of the respective stage.[4]

Results. Tab. 3 summarizes the results for models trained on NQ10k, NQ320k, and Trivia-QA (results for the other models are qualitatively similar). We see approximately the same pattern in all three models. In all three stages, the self-attention component can be removed without affecting most of the retrieval performance. In Trivia-QA, replacing self-attention with zero or mean values leads to over 10% of documents which are not ranked at Rank 1 anymore, suggesting that it has learned a slightly more prominent role for this data set. In Stage I, cross-attention does not play an important role either, as its performance loss in Stage I is relatively small compared to the other stages. In Stages II and III, cross-attention cannot be replaced by zero or mean outputs without a major performance drop. A reason might be its role as the only component that moves query information from the encoder to the decoder. For the models trained on the larger datasets, the highest performance drops caused by patching cross-attention can be seen for Stage II, while for NQ10k the patching in Stage III resulted in higher losses. Interestingly, in Stage I, the MLP components can be completely replaced by their mean output for the correct queries. In Stage III for all datasets, MLPs contribute substantially to query-specific retrieval, resulting in high losses when being replaced or removed.

Motivated by these results, we perform a subsequent experiment combining the patching of individual stages to verify that this partial computation graph of the model can still perform retrieval. As we aim to construct the essential combination of components for retrieval, we choose in each stage those components that resulted

[4]Initially, we patched each component individually in each *layer*, but did not observe a drop in performance for the individual components, suggesting that the model could recover its performance, possibly due to redundancy.

Table 3: Percent of correct queries where the rank of the relevant document is not on Rank 1 after applying patching, we remove/ replace the indicated component output only in the indicated stage. The highest value, i.e., the largest drop in performance per model and stage, is indicated in bold. MLPs in Stage I can be replaced by mean activations, cross-attention in Stage II and III is crucial for the retrieval process, and MLPs in Stage III are the components adapted most for retrieval.

		NQ10k			NQ320k			Trivia-QA		
Component	Method	Stage I	Stage II	Stage III	Stage I	Stage II	Stage III	Stage I	Stage II	Stage III
Attention	zero	1.52	2.27	6.57	2.67	4.46	11.23	12.58	14.78	19.97
MLP	zero	**92.86**	7.83	**96.72**	**34.40**	30.30	**99.28**	**29.89**	16.94	**97.51**
Cross-Attention	zero	4.29	**21.72**	39.65	15.86	50.62	41.89	15.29	**64.20**	60.02
Attention	mean	0.00	2.02	6.06	1.78	5.35	11.76	14.62	16.62	20.93
MLP	mean	0.76	5.56	48.23	3.39	17.47	71.48	14.70	17.58	61.67
Cross-Attention	mean	4.55	20.45	42.93	16.04	**53.30**	50.80	18.14	54.44	71.43
Attention	T5	5.30	5.56	7.32	10.16	11.05	13.37	5.54	6.65	9.78
MLP	T5	5.30	7.07	74.49	14.62	14.62	92.69	11.94	8.01	81.45
Cross-Attention	T5	7.58	17.93	34.09	13.19	24.96	33.16	8.91	21.31	41.38

in the lowest loss in the previous experiment. Concretely, we remove cross-attention from Stage I and attention from all stages, and replace the MLP output in Stages I and II with the mean output over all correct queries. To strengthen the argument that this configuration performs the main computation required for relevance judgment, we evaluate each model on the respective test set.

Tab. 4 shows the results of all models after applying the patch. All models are still able to perform retrieval even though their computation graph now contains one third of the original 72 active components (3 components—attention, cross-attention, MLPs—per layer × 24 layers = 72). Different models exhibit larger or smaller losses in performance. While for NQ10k the losses are marginal, both Trivia-QA models lose approx. 10 points in each of the metrics.

Nevertheless, all models are still able to perform ranking, demonstrating that these components recover most of the performance of the entire model. We conclude, therefore, that these model components are essential for the retrieval mechanism. To summarize, "active" components, i.e., components that cannot be removed or replaced by a mean output, are the MLPs in the last stage and cross-attention in the middle and last stages.

Patching in T5. The previous experiment showed which components can be removed or replaced by their mean output. However, this does not indicate that these components perform *retrieval-specific* behavior. To investigate this question, we perform patching in the same scenario as before, but this time we replace the output of a component with the output of the same component when the same input is processed by a pre-trained (but not GenIR-trained) T5-large. By doing so, we can analyze which mechanisms are learned during pre-training and which are acquired later during the GenIR training. This idea is similar to cross-model activation patching from [25], which patched from fine-tuned to pre-trained models, while we patch in the other direction.

The results for patching in T5-large are shown in Tab. 3 (bottom). Clearly, only a small number of queries require the model's attention components to be trained for GenIR. This is in line with our findings that attention can be removed from the model's computation. Similarly, the MLPs in Stages I and II do not seem to be specifically adapted to the GenIR process, resulting in a relatively small drop of 5–15%, depending on the model. In all three models, the largest drop stems from the MLPs in Stage III, followed by the cross-attention in Stages III and II. Cross-attention in Stage II is less adapted to retrieval compared to Stage III, indicating that the Stage II mechanism may have already been acquired during pre-training.

The greatest adaptation to the retrieval task occurs in Stage III, where we see the largest performance change for all three components. We, thus, conclude that Stage III learns information crucial to the retrieval process, potentially specific to the learned corpus.

Table 4: Results on the respective test set after patching. Stage I consists of MLP mean outputs, Stage II of MLP mean outputs and cross-attention, and Stage III consists of both cross-attention and MLPs. Percentages denote results relative to the model without interventions. On all datasets, ranking can still be performed efficiently even though all models suffer small losses.

	Hits@1		R@5		Hits@10	
	Abs.	Rel.	Abs.	Rel.	Abs.	Rel.
NQ10k	38.0	94%	56.90	98%	62.0	97%
NQ100k	17.60	78%	35.10	82%	41.20	82%
NQ320k	16.40	82%	33.60	84%	42.01	88%
Trivia-QA	49.67	82%	55.26	86%	74.95	91%
Trivia-QA QG	38.94	78%	41.60	80%	57.97	87%

5.2 Rank Development

Recall that the objective of a GenIR model is to predict the relevant document identifier, which means it is optimized to place the relevant document identifier at a low rank in its output. In this section, we examine the changes in the rank of the relevant document identifier caused by different model components. This analysis provides a first intuition on the role of the MLPs and cross-attention.

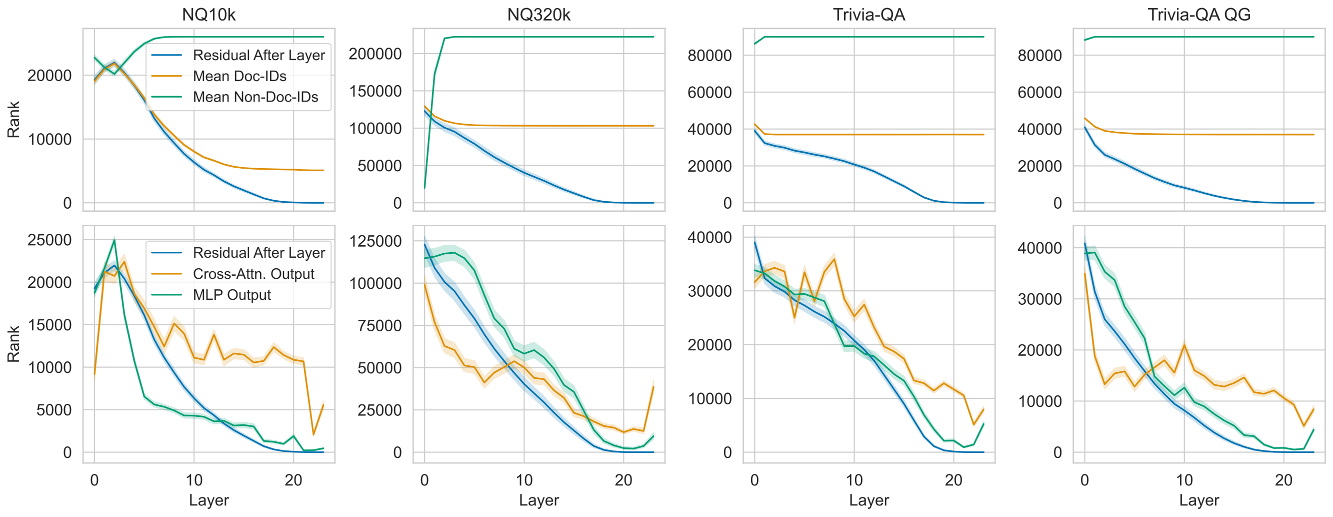

Figure 4: Rank of the correct document after each layer, average rank of all document identifier tokens and non-document identifier tokens after applying logitlens to the output of the layer (top), and rank of the correct document after applying logitlens to the indicated model component (bottom), displayed per layer. All models follow a similar trend (including NQ100k, omitted for space concerns): The models separate document identifiers and non-document identifiers early, and gradually improve the rank of the relevant document. The cross-attention output does not seem to follow this gradual progression.

We adopt the *logit lens* [23], a technique that projects intermediate hidden states from a model to the space of output vocabulary tokens. This technique has been used extensively to study how decoder language models build representations when processing a given input [4, 8, 9, 13, 14]. The key intuition is that due to the residual stream view, hidden states at different layers can be projected to the output vocabulary with the unembedding matrix:

$$\textit{logit-lens}(r) = W_U \cdot \text{scale}(\text{ReLU}(r)),$$

where scale$(\cdot)$ scales its input by $\sqrt{d_{\text{model}}}$ as done before the unembedding layer in T5 models. logit-lens$_i(r)$ contains the logits of the ith token in the vocabulary when applied to vector r. The higher the logit value, the lower the rank of the corresponding token, such that the token at Rank 1 has the highest logit value. We use logit-lens to determine the ranks of document id tokens, of non-document id tokens, and of relevant document ids, when applying it to the outputs of cross-attention, MLPs, and the residual stream in total after each layer. We average the ranks over all correct queries. In this experiment, we use all model components and do not remove or replace model outputs as in the previous section.

Results. Fig. 4 displays the rank for the relevant documents for each layer (top) and the outputs of cross-attention and MLPs per layer (bottom). In early layers, the relevant documents move to the lower half of all ranks, with the average rank of all document ids. It gradually decreases until it arrives at Rank 1 in the last layer. A similar progression can be seen for the MLP output. But for the cross-attention output, the rank of the relevant documents does not gradually change. Instead, no clear trend is visible.

One hypothesis is that only the MLPs contribute directly to the ranking of the documents by modifying the residual stream. We therefore calculate the rank of each document when only the output of MLPs contributes to the logits. In this experiment, for each query, we perform a pass through the model, so each model component uses its "normal" input to calculate its output without any intervention. Only the calculation of the logits is changed from using the residual stream with the output of all components (attention, cross-attention, and MLPs) to using only the residual stream consisting of the MLP outputs. As before, the experiment is performed on the correct queries only. Here, we only use NQ10k. When only using the output of MLPs to calculate the logits, the performance drops slightly: 33% of the relevant documents are no longer in Rank 1, but between Rank 1 and 10, indicating that the ranking can still be performed well. Performing the same experiment using only the output of the cross-attention components reveals that ranking using only this information is not possible. For all correct queries, the relevant document is ranked above 100.

The results of both analyses indicate that cross-attention does not directly influence the rank of document identifiers, but instead passes information to the MLPs, which then "write" the relevant information to the residual stream. This communication between cross-attention and MLPs is analyzed in detail in the next section.

5.3 Communication between Components

We saw in the previous sections that we can divide the decoder pass into three stages. In Stage II, cross-attention contributed more than other components, while in Stage III, the MLPs seem to be the most important components. In addition, the MLP components, as demonstrated in the previous section, mainly affect the output logit and, thereby, the output ranking. Therefore, we suspect that communication between these two component types is a crucial part of the retrieval process. To investigate this communication,

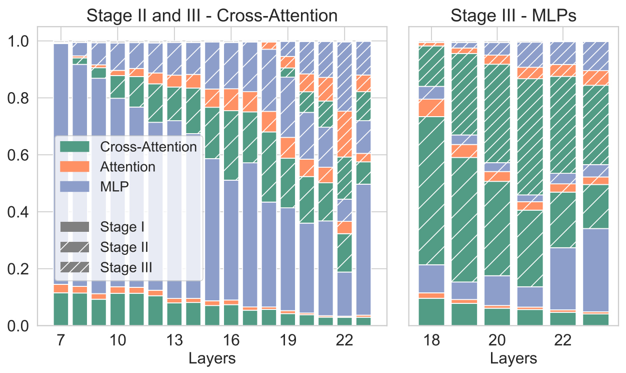

Figure 5: Components per stage that trigger cross-attention in Stage II and III (left) and activate MLPs in Stage III (right) of NQ10k. Stage III MLPs get mostly activated from cross-attention in Stage II and III, while cross-attention in Stage II gets mostly activated by Stage I MLPs.

we gather from which components Stage III MLPs and the cross-attention in later stages read. The following analysis is based on the idea that the input to MLPs and cross-attention is the residual stream, the sum of all outputs from components before the one being analyzed. Hence, we can compute the contribution of each component to the MLPs/cross-attention output to determine the most influential components.

Contribution to Component Output. Recall from Sec. 2 that only positively activated neurons contribute to the MLP output. We therefore investigate how these neurons get activated. The ith neuron is activated when $\text{FF}_i^{\text{proj}} \cdot r > 0$. The MLP's input is the residual stream r, the sum of all previous components. For each activated neuron, we iterate through all previous components and calculate the contribution t_{MLP} as the dot product between $\text{FF}_i^{\text{proj}}$ and each component's output c_{out}, averaged over all activated neurons and all correct queries: $t_{\text{MLP}}(c_{\text{out}}) = \text{FF}_i^{\text{proj}} \cdot c_{\text{out}}$.

In a similar way, we calculate the contribution of all components to the output of a cross-attention head. The composition of the output of one cross-attention head is determined by the similarity-scores $(s_1, ..., s_N)$. Let $\hat{i}$ denote the index of the highest score: $\hat{i} = \arg\max_i(s_i)$. Because the highest similarity score $s_{\hat{i}}$ was computed by $q^T \cdot k_{\hat{i}} = (W_Q \cdot r)^T \cdot k_{\hat{i}}$, we iterate through all previous component outputs c_{out} and calculate the contribution score $t_{\text{Cr-Attn}}$ as: $t_{\text{Cr-Attn}}(c_{\text{out}}) = W_Q \cdot c_{\text{out}} \cdot k_{\hat{i}}$. We average over all heads and all correct queries.

Results. For each correct query, we compute the active neurons in NQ10k and determine the model components with the highest contribution to the activation. Fig. 5 displays the proportions of the dot product scores of each component to the cross-attention of Stage II and III, $t_{\text{Cr-Attn}}$, and the active neurons of the MLPs of Stage III, t_{MLP}. We aggregate the components by type and stage. Each bar in the diagram shows from which previous components cross-attention (left) or MLPs (right) in this layer read. The results show that cross-attention in Stage II and later Stage III plays a major role in shaping the MLPs output, as it is mostly determined by the output of these components (proportion of green colors in each bar in the right plot). The output of cross-attention in Stage II and III is mostly determined by Stage I MLPs (proportion of solid blue color in the left plot). With later layers, other components from Stage II and III also contribute, but still less than Stage I MLPs. Layer 22 is an outlier as Stage III MLPs contribute the largest share to the activation of the cross-attention heads in this layer.

In addition, we repeat this analysis for individual cross-attention heads. We find that for NQ10k the same heads (66% of all cross-attention heads) are within the top 5 components with the highest contribution for all correct queries. Overall, all cross-attention heads except for the last layer ones appear in the top 5 for at least one query. This finding indicates that all cross-attention heads participate in computing the output (through MLPs).

Overall, these analyses show that MLPs in Stage III, which mainly contribute to the final result, get activated by the cross-attention output (in Stage II and later Stage III). In contrast, the cross-attention output is mostly determined by MLPs from Stage I. These results suggest that Stage I serves as a *priming* or preparation phase to contribute information to the residual stream, which later triggers cross-attention. Cross-attention then transfers information from the encoder to the residual stream. This information activates neurons in Stage III. Nevertheless, it remains open what cross-attention writes and the neurons read. We take a look at this aspect next.

Table 5: Percentage of Document Identifier tokens within the Top 10, 100, and 1,000 tokens with highest logits in the logit-lens projection of the cross-attention Components. Average over all layers in Stage II, all heads, and all correct queries.

Dataset	Top 10	Top 100	Top 1,000
NQ10k	0.00	0.10	2.30
NQ100k	0.00	0.01	0.50
NQ320k	0.00	0.04	0.57
Trivia-QA	0.00	0.08	8.58
Trivia-QA QG	0.02	0.16	2.68

5.4 Cross-Attention and MLPs communicate in the word-token space

To better understand how MLPs and cross-attention communicate, we apply logit-lens (see Sec. 5.2) to the output of the cross-attention components. Thereby, we identify the tokens the cross-attention output is most similar to. We sort all tokens in the vocabulary by their logit value, and compute the proportion of non-document id within the top 10, 100, and 1,000 tokens with the highest logits. The results (Tab. 5) show that cross-attention heads mostly output vectors that are more similar to non-document id tokens. Document ids rarely appear in the top 1,000. Since MLPs are activated by these outputs, these results suggest that cross-attention and MLPs communicate in the space of non-document id tokens.

In addition, Tab. 6 displays the top 5 tokens for three queries in Stage II after projecting the cross-attention head outputs in these layers using logit-lens. Evidently, these tokens are semantically similar to the input query. Among them are also tokens in other languages, like German or French, that share similar semantics.

Table 6: Examples of top 5 tokens whose logits were promoted by a given head when the model processed the query on NQ10k.

Query	Head	Top 5 Words
who wrote the harry potter books	Layer 14 - Head 2	about, written, about, tailored, privire
	Layer 16 - Head 1	books, ouvrage, books, authors, book
who won the football championship in 2002	Layer 16 - Head 1	year, YEAR, Year, year, jahr
	Layer 16 - Head 13	football, Football, fotbal, soccer, NFL
will there be a sequel to baytown outlaws	Layer 12 - Head 8	erneut, successor, similarly, repris, continuation
	Layer 16 - Head 1	town, towns, city, Town, village

It seems that the model has learned to perform query expansion similar to other neural retrieval models [27].

This might indicate that the communication between cross-attention and MLPs happens in the space of word-tokens, and that cross-attention learns no information on the document ids or propagates such information. We saw in Sec. 5.1 that the model acquired several mechanisms that are part of the retrieval process already during pre-training. Therefore, the communication in the word space could also be a leftover from the next word prediction task, which the model was originally trained on during pre-training.

6 The Retrieval Process in GenIR

This section summarizes our findings and provides a walkthrough of the retrieval process discovered in our GenIR models. A high-level overview of this process is displayed in Fig. 1.

Encoder. The model first embeds the query at the beginning of the encoder, which then applies self-attention and MLPs to contextualize the input tokens. The encoder is not required to encode information on the documents directly, as it can be replaced by an encoder without document-specific information (see Sec. 4).

Priming Stage. The decoder embeds a generic start token. In the first stage (layers 0–6), no query-specific information is required (Sec. 5.1). The output of the MLPs moves document id tokens to lower ranks and non-document id tokens to higher ranks (Sec. 5.2) while adding information used by subsequent cross-attention.

Bridging Stage. In the second stage (layers 7–17), cross-attention moves information from the encoder to the decoder. The cross-attention heads output information on the input query to the residual stream, which resembles a form of query expansion (Sec. 5.4). This information is then used to activate neurons (Sec. 5.3).

Interaction Stage. In Stage III (layers 18–23), cross-attention continues to output query information to the residual stream. At the same time, the MLP neurons are activated and output information that promotes document identifier tokens (Sec. 5.2). In the last layer, only the MLPs are required. They do not read from the last layer cross-attention component (Sec. 5.3). In this layer, all non-document id tokens are moved to lower ranks, such that only document id tokens will be predicted by the model (Sec. 5.2). This stage received the greatest adaptation to the retrieval task during the GenIR training, implying that corpus-specific information is most likely be stored in this stage. Therefore, query and documents interact only in this stage. Finally, in the last layer, the residual stream is multiplied by the unembedding matrix, resulting in logits for each token that are used to rank the document ids.

Although our results suggest that the same retrieval process is present in all models we examined, differences were observable as well. In models trained on larger datasets, MLPs in the first and second stages seem similarly important, indicating that the priming stage might be longer than for NQ10k.

7 Conclusion

We reverse-engineered the retrieval process in atomic GenIR models using mechanistic interpretability methods. First, we employed an encoder trained on fewer or no documents, demonstrating that the retrieval process within GenIR models does not rely on a retrieval-tuned encoder. Our main contribution is uncovering the pass through the decoder which can be divided into three stages: (I) the priming stage, which prepares the residual stream for subsequent components; (II) the bridging stage, where cross-attention transfers query information from the encoder to the decoder; and (III) the interaction stage, where predominantly MLPs promote relevant documents. Furthermore, our experiments revealed that GenIR models can still rank documents effectively with only one-third of their components. These results suggest that GenIR fine-tuning can be optimized by selecting only crucial components for faster training or forcing to adapt all components for improved retrieval. Also, removing inactive components will enhance inference efficiency.

Our experiments showed the retrieval process in atomic GenIR models using two datasets, but we have not determined how models learn these mechanisms, so we cannot ensure all GenIR models operate the same way. The retrieval process re-uses existing mechanisms from pre-training, meaning different pre-training schemes may result in varied retrieval mechanisms. Future work should analyze the effects of training data and pre-training on retrieval. We are releasing our models and code to support further research.

Acknowledgments

The authors are grateful to Maik Fröbe and Hadas Orgad for providing feedback on this work. A.R. was funded through the Azrieli international postdoctoral fellowship and the Ali Kaufman postdoctoral fellowship. This research was supported by an Azrieli Foundation Early Career Faculty Fellowship, and an AI Alignment grant from Open Philanthropy. This research was partly funded by the European Union (ERC, Control-LM, 101165402). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen-tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: generating substrings as document identifiers. In *Proceedings of the 36th International Conf. on Neural Information Processing Systems*. 31668–31683.

[2] Catherine Chen, Jack Merullo, and Carsten Eickhoff. 2024. Axiomatic Causal Interventions for Reverse Engineering Relevance Computation in Neural Retrieval Models. In *Proceedings of the 47th International ACM SIGIR Conf. on Research and Development in Information Retrieval*. 1401–1410.

[3] Xiaoyang Chen, Yanjiang Liu, Ben He, Le Sun, and Yingfei Sun. 2023. Understanding differential search index for text retrieval. *arXiv preprint arXiv:2305.02073* (2023).

[4] Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing Transformers in Embedding Space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 16124–16170. https://doi.org/10.18653/v1/2023.acl-long.893

[5] 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, and Chris Olah. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread* (2021). https://transformer-circuits.pub/2021/framework/index.html.

[6] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. A White Box Analysis of ColBERT. In *Advances in Information Retrieval*, Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani (Eds.). Springer International Publishing, Cham, 257–263.

[7] Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 9574–9586.

[8] Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022. LM-Debugger: An Interactive Tool for Inspection and Intervention in Transformer-Based Language Models. In *Proceedings of the 2022 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, Wanxiang Che and Ekaterina Shutova (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 12–21. https://doi.org/10.18653/v1/2022.emnlp-demos.2

[9] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conf. on Empirical Methods in Natural Language Processing*. 5484–5495.

[10] Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems* 36 (2024).

[11] Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a Mechanistic Interpretation of Multi-Step Reasoning Capabilities of Language Models. In *Proceedings of the 2023 Conf. on Empirical Methods in Natural Language Processing*. 4902–4919.

[12] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

[13] Shahar Katz and Yonatan Belinkov. 2023. VISIT: Visualizing and Interpreting the Semantic Information Flow of Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 14094–14113. https://doi.org/10.18653/v1/2023.findings-emnlp.939

[14] Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. 2024. Backward Lens: Projecting Language Model Gradients into the Vocabulary Space. In *Proceedings of the 2024 Conf. on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, FL, USA, 2390–2422. https://doi.org/10.18653/v1/2024.emnlp-main.142

[15] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd Int'l ACM SIGIR conf. on research and development in Information Retrieval*. 39–48.

[16] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[17] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Changjiang Zhou, Maarten de Rijke, and Xueqi Cheng. 2024. On the Robustness of Generative Information Retrieval Models. *arXiv preprint arXiv:2412.18768* (2024).

[18] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems* 35 (2022), 17359–17372.

[19] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit Component Reuse Across Tasks in Transformer Language Models. In *The Twelfth International Conf. on Learning Representations*.

[20] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. In *Acm sigir forum*, Vol. 55. ACM New York, NY, USA, 1–27.

[21] Neel Nanda and Joseph Bloom. 2022. TransformerLens. https://github.com/TransformerLensOrg/TransformerLens.

[22] Thong Nguyen and Andrew Yates. 2023. Generative retrieval as dense retrieval. *arXiv preprint arXiv:2306.11397* (2023).

[23] nostalgebraist. 2020. Interpreting GPT: The logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens

[24] Ronak Pradeep, Kai Hui, Jai Gupta, Adam D Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Q Tran. 2023. How Does Generative Retrieval Scale to Millions of Passages? *arXiv preprint arXiv:2305.11841* (2023).

[25] Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. Fine-Tuning Enhances Existing Mechanisms: A Case Study on Entity Tracking. In *The Twelfth International Conf. on Learning Representations*. https://openreview.net/forum?id=8sKcAWOf2D

[26] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[27] Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What Are You Token About? Dense Retrieval as Distributions Over the Vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2481–2498.

[28] Qiaoyu Tang, Jiawei Chen, Zhuoqun Li, Bowen Yu, Yaojie Lu, Haiyang Yu, Hongyu Lin, Fei Huang, Ben He, Xianpei Han, et al. 2024. Self-Retrieval: End-to-End Information Retrieval with One Large Language Model. In *The Thirty-eighth Annual Conf. on Neural Information Processing Systems*.

[29] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems* 35 (2022), 21831–21843.

[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[32] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. *Advances in Neural Information Processing Systems* 33 (2020), 12388–12401.

[33] Jonas Wallat, Hauke Hinrichs, and Avishek Anand. 2024. Causal Probing for Dual Encoders. In *Proceedings of the 33rd ACM International Conf. on Information and Knowledge Management*. 2292–2303.

[34] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conf. on Learning Representations*. https://openreview.net/forum?id=NpsVSN6o4ul

[35] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems* 35 (2022), 25600–25614.

[36] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[37] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. An analysis of BERT in document ranking. In *Proceedings of the 43rd International ACM SIGIR Conf. on research and development in Information Retrieval*. 1941–1944.

[38] Yujia Zhou, Jing Yao, Zhicheng Dou, Yiteng Tu, Ledell Wu, Tat-Seng Chua, and Ji-Rong Wen. 2024. ROGER: Ranking-oriented Generative Retrieval. *ACM Transactions on Information Systems* (2024).

[39] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257* (2022).

[40] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128* (2022).