Improving language models by retrieving from trillions of tokens

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Laspiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen and Laurent Sifre.

We enhance auto-regressive language models by conditioning on document chunks retrieved from a large corpus, based on local similarity with preceding tokens. With a 2 trillion token database, our Retrieval-Enhanced Transformer (R/e.sc/t.sc/r.sc/o.sc) obtains comparable performance to GPT-3 and Jurassic-1 on the Pile, despite using 25× fewer parameters. After fine-tuning, R/e.sc/t.sc/r.sc/o.sc performance translates to downstream knowledge-intensive tasks such as question answering. R/e.sc/t.sc/r.sc/o.sc combines a frozen Bert retriever, a differentiable encoder and a chunked cross-attention mechanism to predict tokens based on an order of magnitude more data than what is typically consumed during training. We typically train R/e.sc/t.sc/r.sc/o.sc from scratch, yet can also rapidly Retrofit pre-trained transformers with retrieval and still achieve good performance. Our work opens up new avenues for improving language models through explicit memory at unprecedented scale.

1. Introduction

Language modelling (LM) is an unsupervised task that consists of modelling the probability of text, usually by factorising it into conditional next-token predictions $p(x_1, \ldots, x_n) = \prod_i p(x_i|x_{<i})$. Neural networks have proven to be powerful language models, first in the form of recurrent architectures (Graves, 2013; Jozefowicz et al., 2016; Mikolov et al., 2010) and more recently in the form of Transformers (Vaswani et al., 2017), that use attention to contextualise the past. Large performance improvements have come from increasing the amount of data, training compute, or model parameters. Transformers have been scaled from 100 million parameter models in seminal work to over hundred billion parameters (Brown et al., 2020; Radford et al., 2019) in the last two years which has led to models that do very well on a wide array of tasks in a zero or few-shot formulation. Increasing model size predictably improves performance on a wide range of downstream tasks (Kaplan et al., 2020). The benefits of increasing the number of parameters come from two factors: additional computations at training and inference time, and increased memorization of the training data.

In this work, we endeavor to decouple these, by exploring efficient means of augmenting language models with a massive-scale memory without significantly increasing computations. Specifically, we suggest retrieval from a large text database as a complementary path to scaling language models. Instead of increasing the size of the model and training on more data, we equip models with the ability to directly access a large database to perform predictions—a semi-parametric approach. At a high level, our Retrieval Transformer (R/e.sc/t.sc/r.sc/o.sc) model splits the input sequence into chunks and retrieves text similar to the previous chunk to improve the predictions in the current chunk. Existing retrieval for language modelling work only considers small transformers (100 millions parameters) and databases of limited size (up to billions of tokens) (Guu et al., 2020; Khandelwal et al., 2020; Lewis et al., 2020; Yogatama et al., 2021). To our knowledge, our work is the first to show the benefits of scaling the retrieval database to trillions of tokens for large parametric language models. Our main
Figure 1 | Scaling of RETRO. The performance gain of our retrieval models remains constant with model scale (left), and is comparable to multiplying the parameteric model size by ~ 10×. The gain increases with the size of the retrieval database (middle) and the number of retrieved neighbours (right) on the C4 validation set, when using up to 40 neighbours. Past this, performance begins to degrade, perhaps due to the reduced quality. At evaluation RETRO can be used without retrieval data (RETRO[OFF]), bringing limited performance degradation compared to baseline transformers.

Our primary contributions are the following.

• We introduce RETRO, a retrieval-enhanced autoregressive language model (§2.2). We use a chunked cross-attention module to incorporate the retrieved text (§2.4), with time complexity linear in the amount of retrieved data. We show that retrieving based on a pre-trained frozen BERT model (§2.3) works at scale, removing the need for training and updating a retriever network.

• We show that our method scales well with model size and database size (Fig. 1): RETRO provides a constant gain for models ranging from 150M to 7B parameters, and RETRO can be improved at evaluation time by increasing the database size and the number of retrieved neighbours. Our largest model obtains state-of-the-art results on a range of downstream evaluation datasets including Wikitext103 (Merity et al., 2017) and the Pile (Gao et al., 2020) (§4). We show that RETRO can be fine-tuned to achieve competitive performance on downstream tasks such as question answering (§4.3).

• We propose an evaluation aware of proximity of test documents with the training set (§2.6), addressing the problem of test set leakage (Lee et al., 2021). This is relevant for all language models, and especially for retrieval-enhanced models since they have direct access to the training dataset during evaluation. Using this methodology, we show that the performance of RETRO comes from both explicit neighbour copying and general knowledge extraction (§4.4).

2. Method

We design our retrieval-enhanced architecture to be capable of retrieving from a database with trillions of tokens. For this purpose, we retrieve at the level of contiguous token chunks instead of individual tokens which reduces storage and computation requirements by a large linear factor. Our method first constructs a key-value database, where values store raw chunks of text tokens and keys are frozen BERT embeddings (Devlin et al., 2019). We use a frozen model to avoid having to periodically re-compute embeddings over the entire database during training. Each training sequence is then split into chunks, which are augmented with their k-nearest neighbour retrieved from the database. An encoder-decoder architecture integrates retrieval chunks into the model’s predictions. We summarize the RETRO architecture in Fig. 2, and detail it in this section. We end the section by introducing
a new methodology to evaluate language models when an evaluation set is partially present in the training set.

2.1. Training dataset

We use a multi-lingual version of MassiveText (Rae et al., 2021) for both training and retrieval data. The dataset consists of text documents from multiple sources and multiple languages totalling over 5 trillion tokens (detailed in Table 1). Sequences are sampled from subsets of the training data, with sampling weights given in the right-most column of Table 1. We tokenize the dataset using SentencePiece (Kudo and Richardson, 2018) with a vocabulary of 128,000 tokens. During training (unless otherwise specified), we retrieve from 600B tokens from the training data. The training retrieval database is made of the same subsets as the training data, in proportion that matches the training sampling frequencies. During evaluation the retrieval database consists in the full union of these datasets, with the exception of books for which we use a sub-sample of 4%. The evaluation retrieval database thus contains 1.75T tokens. To limit test set leakage, we compute the 13-gram Jaccard similarity between train and test documents using the MinHash scheme and remove all training documents with high similarity (0.8 or higher) to a validation or test set document. Additionally, we remove all validation and test articles from Wikitext103 (Merity et al., 2017) from our Wikipedia training data.

2.2. Retrieval-enhanced autoregressive token models

Our approach uses retrieval as a way to augment input examples at the granularity of small chunks of tokens. Formally, we consider sequences of integer tokens in $\mathbb{V} = [1, v]$, obtained using a text tokenizer\(^1\). We split each $n$-token-long example $X = (x_1, \ldots, x_n)$ into a sequence of $l$ chunks $(C_1, \ldots, C_l)$ of size $m = \frac{n}{l}$, i.e. $C_1 \equiv (x_1, \ldots, x_m), \ldots, C_l \equiv (x_{n-m+1}, x_n) \in \mathbb{V}^m$. We use $n = 2048$ and $m = 64$. We augment each chunk $C_u$ with a set $\text{RET}_D(C_u)$ of $k$ neighbours from the database $\mathcal{D}$. $\text{RET}_D$ (or

\(^1\)We use the notation $[1, v] \equiv \{1, \ldots, v\}$ throughout the text.
We use the SCaNN library (Guo et al., 2020) to achieve this. This means that we can query our Table1 values.

This makes retrieval-enhanced models directly comparable with the largest language models that are trillion token database in MassiveText. Thelastcolumnindicatesthesamplingweightduringtraining. Themultilingual subsets include documents in 10 languages. The full breakdown is given in §A.1.

2.3. Nearest neighbour retrieval

Retrieval neighbours. Our database consists of a key-value memory. Each value consists of two contiguous chunks of tokens which we denote \([N, F]\) where \(N\) is the neighbour chunk which is used to compute the key, and \(F\) is its continuation in the original document. The corresponding key is the BERT embedding of \(N\), averaged over time, that we denote \(\text{BERT}(N)\). For each chunk \(C\), we retrieve its approximate \(k\)-nearest neighbours from our key-value database using the \(L_2\) distance on BERT embeddings \(d(C, N) = ||\text{BERT}(C) - \text{BERT}(N)||_2^2\). The model receives the corresponding values \(\text{RET}(C) \triangleq ([N^1, F^1], \ldots, [N^k, F^k])\). Both neighbour chunks and their continuations provide meaningful improvements, as illustrated in our ablation study (Appendix D). We use a length 64 for both \(N^j\) and \(F^j\), thus \(\text{RET}(C)\) has a shape of \(k \times r\) with \(r = 128\). To avoid retrieving the chunk \(C_{u+1}\) in the retrieval set \(\text{RET}(C_u)\), which would break causality during training, we filter out neighbours originating from the same document as the training sequence \(X\).

For a database of \(T\) elements, we can query the approximate nearest neighbours in \(O(\log T)\) time. We use the SCaNN library (Guo et al., 2020) to achieve this. This means that we can query our 2 trillion token database in 10 ms whilst evaluating or sampling from the model; this expense is amortized over a chunk length. Performing retrieval on-the-fly is too slow to keep up with the training calculations—we leverage the frozen aspect of the embedding operator \(\text{BERT}\) to precompute all approximate nearest neighbours and save the results as part of the data. In Fig. 9 in the Appendix, we show results where we only retrieve neighbours within Wikipedia. We find that neighbours tend to come from 2-3 links away from a given article whereas random articles are more than 5 links apart.

Table 1 | MassiveText. The last column indicates the sampling weight during training. The multilingual subsets include documents in 10 languages. The full breakdown is given in §A.1.
2.4. **Retro** model architecture

Our model relies on an encoder-decoder transformer architecture, integrating the retrieved data through a cross-attention mechanism as introduced in Vaswani et al. (2017). First, the retrieved tokens \( \text{RET}(C) \) are fed into an encoder Transformer, which computes the encoded neighbours set \( E \). Denoting the intermediate activations by \( H \), our transformer decoder then interleaves \( \text{RET} \)-blocks \( \text{RETRO}(H, E) \) and standard Transformer blocks \( \text{LM}(H) \) (the hyperparameter \( P \subseteq [1, I] \) determines at which layers we use a \( \text{RET} \)-block). These blocks are built from three different residual operators with signature \( \mathbb{R}^{nxd} \rightarrow \mathbb{R}^{nxd} \): a fully-connected layer \( \text{FFW} \), the standard sequence-level self-attention layer \( \text{ATTN} \), and a chunked cross-attention layer \( \text{CCA}(\cdot, E) \) that incorporates information from the retrieval encoder:

\[
\text{RETRO}(H, E) \triangleq \text{FFW}(\text{CCA}(\text{ATTN}(H), E)) \quad \text{and} \quad \text{LM}(H) \triangleq \text{FFW}(\text{ATTN}(H))
\]

Since \( \text{FFW} \), \( \text{ATTN} \) and \( \text{CCA} \) are all autoregressive operators whose output at position \( i \) only depends on \( (h_{j})_{j \leq i} \), any succession of \( \text{RET} \) and \( \text{LM} \) layers, followed by a token classification head defines an autoregressive log-likelihood (1). An overview of the model architecture is given in Algorithm 1 and in Fig. 2. We next describe the retrieval encoder and the chunked cross-attention layer in more detail, and explain how to sample from \( \text{RETRO} \).

**Encoding retrieval neighbours.** For each chunk \( C_u \), the \( k \) retrieval neighbours \( \text{RET}(C_u) \) are fed into a bi-directional transformer \( \text{ENCODER} \), yielding the outputs \( E_u \triangleq \text{ENCODER}(\text{RET}(C_u)', H_u) \in \mathbb{R}^{rxd'} \), where \( j \in [1,k] \) indexes each neighbour. The retrieval encoder is a non-causal transformer. It is conditioned on \( H_u \), the activations of chunk \( C_u \), through cross-attention layers; this allows the representations of the retrieval encoder to be modulated by the retrieving chunk in a differentiable way. More precisely, the encoding of the \( j \)th neighbour of the \( u \)th chunk, \( \text{RET}(C_u)' \), depends on the attended activation \( H_u \triangleq (h_{(u-1)m+i})_{i \in [1,m]} \in \mathbb{R}^{mxd} \) of chunk \( C_u \) at layer \( \min(P) \). All neighbours for all chunks are encoded in parallel, yielding a full encoded set \( E \triangleq (E_u)'_{u \in [1,I]} \in \mathbb{R}^{k \times k \times rxd'} \). We denote \( E_u \in \mathbb{R}^{k \times rxd'} \) as the encoded neighbours for chunk \( u \in [1,I] \).

**Chunked cross-attention.** To perform the \( \text{CCA} \) operation, we first split a given intermediate activation \( H \in \mathbb{R}^{rxd} \) into \( l-1 \) attending chunks \( H^+ \triangleq (h_{um+i-1})_{i \in [1,m]} \in \mathbb{R}^{mxd} \), as depicted on the right of Fig. 2. \( H^+ \) holds the intermediary embeddings of the last token in chunk \( C_u \) and of the first \( m-1 \) tokens in \( C_{u+1} \). We compute the cross-attention between \( H^+ \) and \( E_u \)—the encoded retrieval set obtained from chunk \( C_u \). Attention is computed across time and across neighbours simultaneously, so as to merge the neighbour and time dimensions of \( E_u \) before applying cross-attention. Since there is a notion of alignment between data chunks and retrieval neighbours, we use relative positional encodings as described in §8.1.2.

We concatenate the \( l-1 \) outputs of the per-chunk cross-attentions (each of shape \( m \times d \)) across time, and properly pad the result; we thus form the output activation \( \text{CCA}(H, E) \in \mathbb{R}^{nxd} \). Formally, for each chunk \( C_u \) and for each token \( i \in [1,m] \) we set

\[
\text{CCA}(H, E)_{u,m,i-1} \triangleq \text{CA}(h_{um+i-1}, E_u),
\]

\(^2\)The last token of chunk \( C_u \) is the first to be able to access the retrieved content \( E_u \) while maintaining autoregressivity in (1). Hence, there is a one token overlap between chunk \( C_u = (x_{(u-1)m+i})_{i \in [1,m]} \) and the corresponding attending chunk \( C^+_{u} \triangleq (x_{um+i-1})_{i \in [1,m]} \).
where the softmax is performed on the second dimension and all products are matrix products. We attending to that set.

The activationsof the

Hyperparam: \( P \) and \( P_{\text{enc}} \), indices of layers with cross-attention in the decoder and encoder respectively

Hyperparam: \( L \) and \( L_{\text{enc}} \), number of decoder layers and number of encoder layers.

Input: \( X \in \mathbb{V}^n \): sequence of tokens. \( \{ \mathbb{R} \mathbb{T}(C_u)\}_{1 \leq u \leq l} \): the retrieved neighbours

Output: \( O \in \mathbb{R}^{nx|\mathbb{V}|} \): the output logits

\[
\text{def } \text{ENCODER}(\mathbb{R} \mathbb{T}(C_u)_{1 \leq u \leq l}, H): \\
(H_u)_{u \in [1,l]} \leftarrow \text{SPLIT}(H) \\
\text{for } j \in [1,k], u \in [1,l] \text{ do} // Encoder shared across neighbours and chunks \\
\quad E_u^j = \text{EMB}_\text{enc}(\mathbb{R} \mathbb{T}(C_u)^j) // May be shared with the decoder \text{Emb} \\
\text{for } p' \in [1, L_{\text{enc}}] \text{ do} \\
\quad E_u^j \leftarrow \text{ATTN}_\text{enc}(E_u^j) // Bi-directional attention \\
\quad \text{if } p' \in P_{\text{enc}} \text{ then} \\
\quad \quad E_u^j \leftarrow \text{CA}_\text{enc}(E_u^j, H_u) \\
\quad E_u^j \leftarrow \text{FFW}_\text{enc}(E_u^j) \\
\text{return } E
\]

\[
H \leftarrow \text{EMB}(X) \\
\text{for } p \in [1, L] \text{ do} \\
\quad H \leftarrow \text{ATTN}(H) // Causal attention \\
\quad \text{if } p = \min(P) \text{ then} \\
\quad \quad // The neighbour ENCODER is conditioned with the decoder activations of the last layer before the first cross-attention \\
\quad \quad E = \text{ENCODER}(\mathbb{R} \mathbb{T}(C_u)_{1 \leq u \leq l}, H) \\
\quad \text{if } p \in P \text{ then} \\
\quad \quad H \leftarrow \text{CCA}(H, E) \\
\quad H \leftarrow \text{FFW}(H) \\
O \leftarrow \text{READ}(H)
\]

where \( \text{CCA} \) is the cross-attention residual operator over time-concatenated encoded neighbours. We recall that this operator is defined in its simplest version by three parameter matrices \( K \in \mathbb{R}^{d \times c}, Q \in \mathbb{R}^{d \times c} \) and \( V \in \mathbb{R}^{d \times d} \). For all \( h \in \mathbb{R}^d \) and \( Y \in \mathbb{R}^{T \times d} \), we define

\[
\text{CCA}(h, Y) \doteq \text{softmax}(Y K Q^T h) Y V,
\]

where the softmax is performed on the second dimension and all products are matrix products. We use multi-head cross-attention, and add positional encodings to the softmax(see §B.1.2).

The first \( m - 1 \) tokens cannot attend to any neighbour of a previous chunk; at these positions, we define \( \text{CCA} \) as the identity, setting \( \text{CCA}(H, E)_j \doteq h_j \) for all tokens \( j \in [1, m - 1] \). Finally, the last token \( h_{lm} \) attends to the last retrieval set \( E_l \) and we set \( h_{lm} \doteq \text{CCA}(h_{lm}, E_l) \) (not shown in Fig. 2). Listing 1 contains a simplified implementation of \( \text{CCA} \). Note that chunked cross-attention is autoregressive: the output of \( \text{CCA} \) at position \( i \) depends on the sequence from tokens from 0 to \( i \) that is input to \( \text{CCA} \).

With \text{RETRO} models, even though each \( \text{CCA} \) cross-attention attends only to the neighbours of the preceding chunk \( \mathbb{R} \mathbb{T}(C_{u-1}) \), the dependencies over previous neighbours are propagated via the self-attention operations. The activations of the \( i^{th} \) token in the \( u^{th} \) chunk therefore potentially depend upon the set of all previous neighbours \( \mathbb{R} \mathbb{T}(C_u)_{u' < u} \), without incurring the quadratic cost of cross attending to that set.
Sampling. When sampling, at the end of a chunk \( C_u \), we use SCaNN to retrieve neighbours \( \text{RET}(C_u) \), based on the embedding \( \text{BERT}(C_u) \). The encoded neighbours \( E_u = \text{ENCODER}(\text{RET}(C_u)) \) are then used to condition the generation of the next chunk \( C_{u+1} \), which we do incrementally: overall the cost of sampling is thus quadratic in the size of the sampled sequence, as when sampling from regular Transformers; the added cost of retrieval is linear in the number of chunks \( l \), and is negligible compared to the token sampling cost in practice.

2.5. Baseline Transformer architecture

We use a transformer (Vaswani et al., 2017) similar to the one described in (Radford et al., 2019), with some minimal changes: we replace LayerNorm with RMSNorm (Zhang and Sennrich, 2019) and use relative position encodings (Dai et al., 2019). As baselines, we train retrieval-free transformers with 132M, 368M, 1.3B and 7.0B parameters (embedding matrices are excluded from parameter counts). The hyperparameters we used are detailed in Table 2. All retrieval models use the same size encoder for the retrieval data, with \( d' = 896 \) and 2 layers, which roughly adds 19M parameters. The encoder uses relative positional encodings. The retrieval models contain one \( \text{RETR}-\text{block} \) every 3 blocks, starting from layer 6. For our smallest model, \( \text{CCA} \) is applied in layers 6, 9 and 12 of the main pathway and also once for query conditioning in the encoder, which adds an additional 12M parameters. The relative number of extra parameters reduces as we increase the baseline model size. All models are implemented using JAX (Bradbury et al., 2018) and Haiku (Hennigan et al., 2020).

2.6. Quantifying dataset leakage exploitation

\( \text{RETR} \) models may arguably benefit more easily from evaluation dataset leakage, i.e. the fact that we evaluate on data that were also present in the training set. To better understand how retrieval improves language modelling performance, we therefore quantify evaluation likelihood as a function of the overlap between the evaluation and training datasets.

The following approach can be used with any language model, and depends only on the frozen retriever system presented in §2.3. We split the evaluation sequences \( (X_i) \) into chunks of length \( m \leq 64 \), and we see the training data as a set of chunks \( C \). For each evaluation chunk \( C \in C \), we retrieve the 10 closest neighbours (of length up to 128) in the training data. We then compute the longest token substring common to both the evaluation chunk and its neighbours. This gives a number \( s \in [0, m] \). The value \( r(C) = \frac{s}{m} \), ranging from 0 (chunk never seen) to 1 (chunk entirely seen), gives a reliable indication of how much overlap there is between the evaluation chunk and the training data. For a given model, we then obtain the log-likelihood \( \ell(C) \) of each chunk \( C \), and the number of bytes \( N(C) \) it encodes. We then consider the filtered bits-per-bytes of the model:

\[
\forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{ C \in C, r(C) \leq \alpha \}, \quad \text{bpb}(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)},
\]

Table 2 | Number of parameters for our baseline and \( \text{RETR} \) models, excluding embeddings, along with the corresponding hyperparameters.

| Baseline parameters | \( \text{RETR} \) | \( d \) | \( d_{ffw} \) | # heads | Head size | # layers |
|---------------------|-----------------|-------|---------|--------|----------|---------|
| 132M                | 172M (+30%)     | 896   | 3,584   | 16     | 64       | 12      |
| 368M                | 425M (+15%)     | 1,536 | 6,144   | 12     | 128      | 12      |
| 1,309M              | 1,451M (+11%)   | 2,048 | 8,192   | 16     | 128      | 24      |
| 6,982M              | 7,532M (+8%)    | 4,096 | 16,384  | 32     | 128      | 32      |
which correspond to the bits-per-bytes on the set of chunks that overlap less than $\alpha$% with the training chunks. Note that the full evaluation bit-per-bytes performance is recovered by $bpb(1)$. The function $bpb(\cdot)$ allows us to evaluate the impact of evaluation leakage over predictive performance: for low $\alpha$, $bpb(\alpha)$ gives an indication on how the model performs on chunks that are entirely new; the slope of $bpb(\cdot)$ shows how much the model exploits evaluation leakage.

3. Related Work

We first review existing work on using retrieval for language modelling, and compare Retro to these works (see Table 3). As we train Retro models on a large dataset containing a substantial section of the internet, our work raises potential privacy, safety, and fairness issues that we then review.

3.1. Retrieval for language modelling

Brants et al. (2007) show that scaling the training data to trillions of tokens improves the machine translation performance of n-gram models. More recently, GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), and Jurassic-1 (Lieber et al., 2021) show that scaling up language models leads to massive improvements on many downstream tasks. At the same time, Carlini et al. (2021) demonstrate that large-scale language models can perfectly memorise parts of their training data, suggesting that enhancing models with retrieval may lead to further improvements. However, significant leakage between train and test datasets (Lee et al., 2021; Lewis et al., 2021) makes comparing and evaluating large models trained on large datasets difficult, especially once retrieval capabilities over the training dataset are added.

Historically, information retrieval for text relies on inverted index matching such as TF-IDF and BM25 (Robertson and Zaragoza, 2009). Foundational work use latent topic modelling approaches like LDA (Blei et al., 2003) to identify relevant neighbours (Wei and Croft, 2006). Work in machine translation such as Zhang et al. (2018) and Gu et al. (2018) retrieve translation pairs based on edit distance between source sentences and guide the translation output using the closest retrieved target sentences. The retrieval database may also be structured — for example, Ahn et al. (2016) use a symbolic knowledge graph to improve an RNN language model.

With the success of deep learning, retrieving systems have partly switched to dense learned representations based on a neural network’s activations. Continuous cache (Grave et al., 2017) adds probability mass to tokens for which previous activations resemble the current activation vector, extending the model’s context to the local history. kNN-LM (Khandelwal et al., 2020) applies this idea to transformers and extends the retrieval database to English Wikipedia, resulting in

Table 3 | Comparison of Retro with existing retrieval approaches.

| Retrieval | # Retrieval tokens | Granularity | Retriever training | Retrieval integration |
|-----------|--------------------|-------------|--------------------|----------------------|
| Continuous Cache | $O(10^3)$ | Token | Frozen (LSTM) | Add to probs |
| kNN-LM | $O(10^9)$ | Token | Frozen (Transformer) | Add to probs |
| SPALM | $O(10^9)$ | Token | Frozen (Transformer) | Gated logits |
| DPR | $O(10^9)$ | Prompt | Contrastive proxy | Extractive QA |
| REALM | $O(10^9)$ | Prompt | End-to-End | Prepend to prompt |
| RAG | $O(10^9)$ | Prompt | Fine-tuned DPR | Cross-attention |
| FiD | $O(10^9)$ | Prompt | Frozen DPR | Cross-attention |
| EMDR² | $O(10^9)$ | Prompt | End-to-End (EM) | Cross-attention |
| Retro (ours) | $O(10^{12})$ | Chunk | Frozen (BERT) | Chunked cross-attention |
substantial improvements on Wikitext103 evaluation. Continuous cache and kNN-LM do not modify the underlying neural-network models, but interpolate at inference between the language model's output and distributions computed from retrieved tokens. These methods can therefore be plugged into any model without additional training, although this limits the model's ability to reason about the retrieved text. SpaLM (Yogatama et al., 2021) addresses this limitation by adding an extra gating network to post-process the retrieved data; yet most of the network is unaffected by the retrieval during inference.

The retrieval representations may be trained directly instead of relying on a pre-trained model—retriever systems have been developed for this purpose, primarily on open-domain question answering. For example, DPR (Karpukhin et al., 2020) trains two BERT models (for queries and keys respectively) using a contrastive loss to align the representations of a question and of its answers. Lee et al. (2019) use an inverse cloze task to find semantic representations of passages for retrieval. These works differs from continuous cache and kNN-LM in that they embeds passages (or chunks) of text together, as opposed to each token individually. The retriever network is trained in isolation of the downstream task that uses the retrieval data. This potential issue is specifically addressed by REALM (Guu et al., 2020), which trains the retrieval system end-to-end to maximize the final training cross-entropy. This comes with the extra complexity of searching the database during training and periodically updating the embedding table, severely limiting the scale at which it can operate. RAG (Lewis et al., 2020) and FID (Izacard and Grave, 2021) build upon DPR to set the state of the art on question answering benchmarks by training encoder-decoder transformer models. More recently, EMDR² (Sachan et al., 2021) extends FID by using an expectation-maximization algorithm to train the retriever end-to-end and achieves state of the art results compared to similarly sized models.

In the open-domain dialogue setting, BlenderBot 2.0 (Komeili et al., 2021) learns to issue textual internet queries, outperforming dense retrieval methods when evaluated on a task measuring how close model responses are to those of humans. This involves collecting a dataset of human dialogues with associated search queries, which limits the scalability of this approach. Hashemi et al. (2020) introduce the Guided Transformer, a modified Transformer similar to REALM, for document retrieval and clarifying question selection. Although effective on question answering and other tasks with strong conditioning, none of these methods are designed to model arbitrary text sequences, in contrast with RETRO.

RETRO shares components with kNN-LM and DPR in that it uses frozen retrieval representations. RETRO models longer sequences than QA examples; this requires to reason at a sub-sequence level, and to retrieve different documents for the different chunks of a sequence. Similar to FID, RETRO processes the retrieved neighbours separately in the encoder, and assemble them in the chunked cross-attention. This differs from e.g. REALM, that prepends retrieved documents to the prompt. Using chunks allows for repeated retrieval whilst generating a sequence as opposed to retrieving only once based on the prompt alone. Furthermore, retrieval is done during the whole pre-training process in RETRO, and is not simply plugged-in to solve a certain downstream task. Finally, previous methods based on dense query vectors use small models and retrieval datasets with less than 3B tokens (English Wikipedia). Table 3 summarizes the difference of RETRO with existing approaches.

### 3.2. Privacy, safety and fairness

Bender et al. (2021); Weidinger et al. (2021) highlight several dangers of large language models. Those stem from their ability to memorise training data, their high training cost, the static nature of their training data (Lazaridou et al., 2021), their tendency of amplifying inherent biases in the training data, and their ability to generate toxic language (Gehman et al., 2020). In this section we inspect these dangers, focusing on how retrieval augmented language models may exacerbate or
mitigate them.

Large language models can perfectly memorise parts of their training data (Carlini et al., 2021). When coupled with large training datasets gathered from the web or other sources, this has clear privacy and safety implications. Retrieval models such as R\textsc{et}ro that have access to the entire training dataset during inference exacerbate these privacy issues by being able to directly copy training data. However, retrieval systems offer a path towards mitigating these concerns via obliteration of the retrievable data at inference time. In addition, differential privacy training (Abadi et al., 2016) of retrieval models could guarantee that no private information is stored in the model weights, while individualisation on private data could be made by updating the retrieval database at inference time.

Due to their high training cost, re-training large language model regularly to incorporate new data, languages, and norms is prohibitively expensive. To keep retrieval models up-to-date, it may be sufficient to update the retrieval database, which is orders of magnitude cheaper than re-training a model from scratch. In addition to the benefits of updating models in terms of fairness and bias, simply training large language models has a significant energy cost (Schwartz et al., 2020; Strubell et al., 2019). Retrieval mechanisms offer a path to reducing the compute requirements needed to train and update language models that reach a certain performance.

Large language models are prone to generating toxic outputs, as shown in Gehman et al. (2020). Bender et al. (2021); Jo and Gebru (2020) advocate for the importance of better training data curation and documentation. Additionally, if portions of the training data are found to be eliciting biased or toxic outputs after training, retrieval allows for some correction, as the offending retrieval data can be retroactively filtered. However, it is also the case that without careful analysis and intervention, retrieval models may exacerbate biases that are present in the training data. Retrieval models can also add a further source of bias through the selection mechanism for retrieval documents. Further work in this area is required to better understand how retrieval affects the bias and toxicity of the model outputs.

Finally, samples from large models are difficult to interpret, making mitigating these issues all the more challenging (Belinkov et al., 2020; Jain and Wallace, 2019). Retrieval provides more insights into the outputs of a model, as one can directly visualise or modify the neighbours that are being used. The examples in Table 6, 7, 20 and 21 illustrate how retrieval makes language models more factual and interpretable by providing more transparent outputs.

4. Results

We first report results on language modelling benchmarks. Second, we show how to R\textsc{et}rofit pre-trained Transformer language models into retrieval models with few additional FLOPs. Next, we report R\textsc{et}ro results on question answering. Finally, we report evaluation metrics with leakage filtering, to better understand the source of the gains with retrieval.

4.1. Language modelling

Datasets. We evaluate our models on C4 (Raffel et al., 2020), Wikitext103 (Merity et al., 2017), Curation Corpus (Curation, 2020), Lambada (Paperno et al., 2016) and the Pile (Gao et al., 2020). We also evaluate on a set of manually selected Wikipedia articles that were added or heavily edited in September 2021, months after our pre-training and retrieval dataset was collected (details are given in §A.2). We construct the dataset with articles from the “future” and manually remove new articles that strongly overlap documents in our training data. This guarantees that the evaluation documents are not leaked in our training data.
Improving language models by retrieving from trillions of tokens

Figure 3 | Scaling with respect to model size. (a) LAMBADA top-1 accuracy. (b) Evaluation loss on curation corpus. (c) Perplexity on Wikitext103 valid. (d) Bits-per-byte on selected Wikipedia articles from September 2021.

For C4, Wikitext103, the Pile, and our Wikipedia dataset we evaluate the language modelling performance on entire documents and measure the bits-per-byte (bpb). We favour bits-per-byte over loss as it is tokenizer agnostic. We evaluate with a sequence length of 2048 tokens but use a stride of 1024 within documents to mitigate boundary effects. On Curation Corpus we concatenate the article, the “TL; DR:” string, and the summary, but only evaluate the bpb on the summary. For Lambada we evaluate the accuracy on the last word, using greedy generation.

Model scaling. In Fig. 1(left) and Fig. 3 we show the language modelling performance as we scale models from 150 million to 7 billion (non-embedding) parameters. We see that on all datasets, RETRO outperforms the baseline at all model sizes. Furthermore, we observe that improvements do not diminish as we scale the models. The performance is dataset dependent, with the largest gains on Wikitext103 and C4. Wikipedia articles and other web pages are similar to Wikitext103 documents, even if not exact copies (§4.4), we thus obtain dramatic improvements on Wikitext103 as our retrieval model is able to directly exploit these overlaps. The smallest gains are for Curation Corpus, where RETRO only slightly outperforms the baseline. This is expected as Curation Corpus summaries are designed to only contain information from the source article and are not included in our retrieval database. On our “future” Wikipedia September 2021 dataset, we also observe consistent gains for all model sizes.

Data scaling. Fig. 1 (middle) shows how scaling the retrieval database at evaluation improves the language modelling performance. We observe dramatic gains as the retrieval data is increased from Wikipedia (4 billion tokens) to all of Massive text (1.7T tokens). Fig. 1(right) shows how performance scales as we increase the number of retrieved chunks. Despite being only trained with 2 neighbours, we see consistent improvements for all models when the number of neighbours is increased from 1 to 10. Furthermore, we observe that larger models are able to better utilise more neighbours: the 172M model improves with up to 10 neighbours, whereas the 7B model improves with up to 40 neighbours.

The Pile. We evaluate our 7B models on the Pile test sets\(^3\) and compare against the 178B parameter Jurassic-1 (Lieber et al., 2021) model and the 280B parameter Gopher (Rae et al., 2021) model. We do not compare against GPT-3 as it is outperformed by Jurassic-1 and Gopher on almost all subsets. Fig. 4 shows the relative improvements in bits-per-byte over our 7B transformer baseline for our

\(^3\)Due to legal and ethical concerns relating to their use, we exclude the Enron Emails and the Youtube Subtitles datasets.
Figure 4 | **The Pile: Comparison of our 7B baseline against Jurassic-1, Gopher, and RETRO.** We observe that the retrieval model outperforms the baseline on all test sets and outperforms Jurassic-1 on a majority of them, despite being over an order of magnitude smaller.

7.5B RETRO model, Jurassic-1 and Gopher. Jurassic-1 outperforms the baseline on all datasets except for books, likely due to the inclusion of books in our training data. Gopher and RETRO outperform the baseline on all test sets. Overall, RETRO 7.5B outperforms Jurassic-1 and Gopher on a majority of the test sets. On the dm_mathematics and ubuntu_irc subsets, our RETRO model does not outperform our 7B baseline and underperforms Jurassic-1. We hypothesise that the retrieved neighbours on these datasets are not helpful, due to a combination of what is in our retrieval dataset and the efficacy of the nearest-neighbour search.

**Wikitext103.** To validate our approach in a controlled setting, we compare our method with kNN-LM (Khandelwal et al., 2020) on the Wikitext103 dataset in Table 4. We train a baseline transformer on the training set of Wikitext103. This transformer has 24 layers, 1024 hidden units, 16 heads and a key size of 64, as in Baevski and Auli (2019). Our baseline does not have adaptive input, and our tokenizer has an open vocabulary, unlike Baevski and Auli (2019), which makes our baseline

| Model                    | Retrieval Set | #Database tokens | #Database keys | Valid | Test  |
|--------------------------|---------------|------------------|----------------|-------|-------|
| Adaptive Inputs (Baevski and Auli, 2019) | -             | -                | -              | 17.96 | 18.65 |
| SPALM (Yogatama et al., 2021) | Wikipedia     | 3B               | 3B             | 17.20 | 17.60 |
| kNN-LM (Khandelwal et al., 2020) | Wikipedia     | 3B               | 3B             | 16.06 | 16.12 |
| Megatron (Shoeybi et al., 2019) | -             | -                | -              | -     | -     |
| Baseline transformer (ours) | -             | -                | -              | 21.53 | 22.96 |
| kNN-LM (ours) | Wikipedia     | 4B               | 4B             | 18.52 | 19.54 |
| RETRO | Wikipedia     | 4B               | 0.06B          | 18.46 | 18.97 |
| RETRO | C4            | 174B             | 2.9B           | 12.87 | 10.23 |
| RETRO | MassiveText (1%) | 18B             | 0.8B           | 18.92 | 20.33 |
| RETRO | MassiveText (10%) | 179B            | 4B             | 13.54 | 14.95 |
| RETRO | MassiveText (100%) | 1792B           | 28B            | 3.21  | 3.92  |
perplexities a bit higher. The full experiment details and hyperparameters are given in §C.2 and Table 11.

We re-implement kNN-LM with our tokenizer and baseline transformer to produce embeddings of size 1024 for every token in Wikitext103. kNN-LM has probabilities $p_{kNN-LM} = \lambda p_{kNN} + (1 - \lambda)p_{LM}$ with $p_{kNN}(n_k) \propto \exp(-d_k)$. We tune $\lambda = 0.118$ and $\alpha = 0.00785$ on the validation set (Fig. 7) and report performance for these hyperparameters on both the validation and test set.

We fine-tune our baseline transformer into a Retro model (Fig. 7), using the Wikitext103 training data and retrieving from Wikipedia with 2 neighbours. We only train the new weights, as explained in §4.2, and share the embedding weights between the encoder and the main pathway. This is necessary for Wikitext103 which is quite small, as training Retro from scratch in this setting leads to over-fitting.

We evaluate the fine-tuned Retro model with different retrieval sets. We use 10 neighbours at evaluation for both Retro and kNN-LM. When retrieving from Wikipedia, we obtain results comparable to our kNN-LM implementation. Furthermore, scaling the retrieval database to MassiveText yields dramatic improvements, though this is partly due to leakage (see §4.4). For reproducibility, we also include results when retrieving from C4, which are close to previous state-of-the-art and comparable to using 10 % of MassiveText.

It is worth noting that kNN-LM requires 1024 floats for every token in the retrieval dataset, totalling 15 terabytes (Tb) for the 4 billion tokens in Wikipedia. kNN-LM and other token-level retrieval approaches therefore don’t scale to retrieval databases with trillions of tokens such as MassiveText. In comparison, Retro only requires 215Gb to index our Wikipedia dataset, and 93Tb for MassiveText. Inspecting the number of retrieval database entries in Table 4 makes it clear why retrieving at the chunk level is necessary when scaling to datasets with trillions of tokens.

4.2. Retro-fitting baseline models

We extend baseline models into Retro models by freezing the pre-trained weights and training only chunked cross-attention and neighbour encoder parameters (less than 10% of weights for the 7B model) in Fig. 5. This offers an efficient alternative path to enhance transformers with retrieval, requiring only 6 million sequences (3% of the pre-training sequences that we used). Additionally, by only training the new weights we ensure that when evaluated without retrieval, the original model performance is exactly maintained. Retrofitting models quickly surpasses the performance of baseline models and even achieves performance close to that of Retro models trained from scratch. The experiment hyperparameters are given in §C.3.

4.3. Question answering

We fine-tune our retrieval models on the Natural Questions (Kwiatkowski et al., 2019) dataset to demonstrate that our retrieval pathway can be used to inject information from arbitrary data sources. We use the version provided by Izacard and Grave (2021) which is augmented with the retrieved passages from DPR (Karpukhin et al., 2020). We fine-tune all the weights of our 7.5B pre-trained Retro model for 25,000 steps using the top 20 retrieved passages. We format the data as “question: {question} \nanswer: {answer}” and left pad the data such that “answer:” coincides with the end of the first chunk of 64 tokens and thus aligns with the first retrieving chunk. The model has access to the question via the previous tokens in the sequence as well as the top 20 DPR Wikipedia passages and their titles via the chunked cross-attention mechanism. 

4https://github.com/facebookresearch/FiD
Improving language models by retrieving from trillions of tokens

Figure 5 | **Retro-fitting a baseline transformer.** Any transformer can be fine-tuned into a retrieval-enhanced transformer by randomly initializing and training only the chunked cross-attention and retrieval encoder weights. Fine-tuning in this way quickly recovers and surpasses the non-retrieval performance, and almost achieves the same performance as training a retrieval model from scratch (shown by the arrow on the right hand side of each plot). We find good performance **Retro-fitting** our models training on only 3% the number of tokens seen during pre-training.

The exact match scores are shown in **Table 5** and the full fine-tuning details are given in §C.4. Our method is competitive with previous approaches such as **Realm**, **RAG** and **DPR**, but underperforms the more recent F1D. In contrast with this work, we find that increasing the number of neighbours past 20 does not improve **Retro** performance on this task. We hypothesise that the encoder-decoder structure of T5—the base model in F1D— and the T5 pre-training objective leads to a model that relies more on the encoder output than **Retro**, which is important in the QA setting. To compete with T5-finetuned models, future work should consider ways of forcing **Retro** to rely further on the retrieval encoder output when producing tokens.

### 4.4. Relating retrieval performance to dataset leakage.

We report the filtered eval losses as detailed in §2.6 on C4, Curation Corpus and Wikitext103 in **Fig. 6**. On C4 and Wikitext103, for which there is leakage into the training set, the slope is negative for both baseline models and **Retro** models. **Retro** models exploit leakage more strongly than baseline models, as indicated by the more negative slope. This is due to its explicit ability to copy-paste existing training chunks to predict leaked evaluation chunks (see a qualitative example of this model behavior).

**Table 5 | Question answering results.** Exact match accuracy on Natural Questions.

| Model                                      | Test Accuracy |
|--------------------------------------------|---------------|
| **Realm** (Guu et al., 2020)               | 40.4          |
| **DPR** (Karpukhin et al., 2020)           | 41.5          |
| **RAG** (Lewis et al., 2020)               | 44.5          |
| **EMDR** (Sachan et al., 2021)             | 52.5          |
| **F1D** (Izacard and Grave, 2021)          | 51.4          |
| **F1D** + Distill. (Izacard et al., 2020)  | 54.7          |
| Baseline 7B (closed book)                  | 30.4          |
| **Retro** 7.5B (DPR retrieval)             | 45.5          |
Improving language models by retrieving from trillions of tokens

Figure 6 | **Performance vs. longest common retrieval substring.** Evaluation loss as a function of allowed longest common substring between evaluation data chunks and their nearest neighbours. Retrieval still helps when considering chunks with no more than 8 contiguous tokens overlapping with training dataset chunks.

on a Wikitext103 article in Table 19). On Curation Corpus, retrieval provides a constant offset, which is expected as there is by design no leakage between Curation Corpus and the training dataset. On the other hand, RETRO outperforms baseline models at all leakage levels, down to $\alpha = 12.5\%$. At this level, the loss is computed on chunks with less than 8 contiguous tokens shared with the closest matching chunk in the training dataset—this is a reasonable level of overlap at which we consider that there is no local leakage. Retrieval thus improves predictions on both chunks that are syntactically similar to chunks in the training set, and on chunks that are syntactically different from all training chunks. This points toward a non trivial RETRO capacity of generalizing based on both model parameters and retrieval database. Similar results are found on the Pile dataset (see Fig. 12, §F.3).

4.5. Using RETRO for sampling

We show examples of samples obtained using the 7.5B RETRO model in Table 6, Table 7 and Appendix E. For each chunk (the first one being the prompt), we juxtapose sampled chunks $C_u$ with retrieved neighbours $\text{RET}(C_u)$. To give an indication of local overlap, we colour each sampled token in chunk $C_u$ based on the length of the longest common prefix (LCP) found in the retrieved chunks $\text{RET}(C_u-1)$. Similarly, we colour the retrieved chunks based on the LCP in the sampled chunk. For the sample in Table 6, for which we chose the prompt, we observe that the retrieved chunks influence the sample as there are overlaps between the sampled tokens and neighbour tokens. Overall, retrieval reduces hallucinations (in line with the findings of Shuster et al. (2021)) and makes the model more knowledgeable, when comparing with samples produced with retrieval disabled. In the sample in Table 7, the model recognises that the prompt is the beginning of the first scene of Hamlet and leverages retrieval data to continue it with only a few mistakes. We provide further examples in Appendix E, including examples from the evaluation sets, as well as the detailed procedure used for colouring the tables.

5. Conclusion

We present Retrieval-Enhanced Transformers (RETRO), a method for modelling arbitrary text sequences whilst retrieving from databases with trillions of tokens—scaling the data available to models by an order of magnitude compared to what is typically consumed during training. RETRO models
gains do not diminish for models with up to at least 7B parameters, and correspond to non-retrieval models with 10× more parameters on certain datasets. On Wikitext103 and the Pile, RetRo outperforms previous models trained on large scale datasets. We also show that RetRo is competitive on retrieval-intensive downstream tasks such as question answering.

RetRo models are flexible and can be used without retrieval at evaluation and still achieve comparable performance to baseline models. Conversely, baseline models can be rapidly fine-tuned into RetRo models to obtain nearly the same performance as if trained from scratch. Careful analysis shows that only a modest fraction of the gains obtained by RetRo are due to test set leakage. In general, we caution for such leakage in large-scale language datasets and suggest further work in better understanding the role of test set leakage in the performance of large-scale language models.

Overall, our work demonstrates at an unprecedented scale that semi-parametric approaches can provide an orthogonal, more efficient approach than raw parameter scaling as we seek to build more powerful language models.

Acknowledgements

We would like to thank Nikolai Grigorev, Marc’auelio Ranzato, Cyprien de Masson d’Autume, Po-Sen Huang, Johannes Welbl, Lisa Anne Hendricks, Ethan Perez, Jeff Stanway, Eric Noland, Gregory Wayne, John Jumper, Julian Schrittwieser, Lorrayne Bennett, Devang Agrawal, Dani Yogatama, Susannah Young, Nando de Freitas, Demis Hassabis, and Koray Kavukcuoglu for their help, advice and reviews. Additionally, we would like to thank Zonglin Li, David Simcha, and the ScaNN developers for their help.
Table 6 | Sample - Beavers are interesting animals. The RetrO[Off] sample quickly diverges to other animals while the RetrO[On] sample tends to stay focused on the beaver topic due to neighbour conditioning.

| Prompt and sample of RetrO[Off] | Prompt and sample of RetrO[On] | \[N_1^2, F^2_1\] colored by LCP with \(C_{11}\) | \[N_1^2, F^2_1\] colored by LCP with \(C_{11}\) |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| Beavers are interesting animals that live near rivers. They build beaver lodges in ponds. They have created wooded areas. Like many things in nature, there is a connection between creatures in the wild. Beaver ponds cause trees to die, but the dead trees attract the great blue heron, which often returns year after year. Over time, a beaver pond can attract more than 50 nests in a colony, called a rookery. An example of this can be found in the large pond off Bradford Road at Carter Fields near the Rockford Pond. Chris Leacy, an expert with the Massachusetts Audubon Society, who was once highly popular, now lives there.

| Beavers are interesting animals that live near rivers. They build their lodges in ponds. They have created wooded areas. Like many things in nature, there is a connection between creatures in the wild. Beaver ponds cause trees to die, but the dead trees attract the great blue heron, which often returns year after year. Over time, a beaver pond can attract more than 50 nests in a colony, called a rookery. An example of this can be found in the large pond off Bradford Road at Carter Fields near the Rockford Pond. Chris Leacy, an expert with the Massachusetts Audubon Society, who was once highly popular, now lives there.

| dams to create ponds. Frogs are amphibians, so they can live in both land and water. They have great camouflage to hide from predators. The Golden Retriever, or Golden, is a very big dog. They are very strong, and have a thick coat to keep them warm.

| dams to create ponds. Frogs are amphibians, so they can live in both land and water. They have great camouflage to hide from predators. The Golden Retriever, or Golden, is a very big dog. They are very strong, and have a thick coat to keep them warm.

| cold areas. A kangaroo is an animal that is very large. It can jump very high. They can run very fast. A meerkat is a small animal that lives in Africa. They live in colonies of up to a hundred of them. They can climb trees very easily.

| cold areas. A kangaroo is an animal that is very large. It can jump very high. They can run very fast. A meerkat is a small animal that lives in Africa. They live in colonies of up to a hundred of them. They can climb trees very easily.

| A mouse is a small mammal that lives on land. It is a very good climber and can run very fast. Penguins are birds that live on Antarctica. They have a thick coat to keep them warm. Rabbits are small animals that live in the ground. They

| A mouse is a small mammal that lives on land. It is a very good climber and can run very fast. Penguins are birds that live on Antarctica. They have a thick coat to keep them warm. Rabbits are small animals that live in the ground. They

| at like tail, and two protruding teeth that are strong enough to gnaw down trees. The beaver uses trees, branches, and mud to build dams across rivers and streams. These dams create a deep pool of water in which the beaver builds its home. Beaver homes are called lodges. A baby beaver is "kit" remains in the family lodge until the age of two Beaver fur, like own pelt, was once highly popular as a trim for hats and coats. How in the popularity of beaver fur contributed to the colonization of New France? Our World
## Sample - Hamlet, Act 1, Scene 1.

The Retro[Off] sample has correct syntax but is hallucinated, and ends with repetition of one character (FRANCISCO Approach me not). The Retro[On] sample is the correct continuation of the original text, and is robust to formatting differences between our prompt and the retrieved data.

| Prompt and sample of Retro[Off] | Prompt and sample of Retro[On] | \([N_1, N_2] \) colored by LCP with \(C_{\text{rel}} = 1\) | \([N_1, N_2] \) colored by LCP with \(C_{\text{rel}} = 1\) |
|---------------------------------|--------------------------------|--------------------------------------------------|--------------------------------------------------|
| ACT I SCENE I. Elsinore. A pla | ACT I SCENE I. Elsinore. A pla | \(LCP = 0, 1, 2, 3, 4, 5\)                       | \(LCP = 0, 1, 2, 3, 4, 5\)                       |
| tform before the castle. FRAN | tform before the castle. FRAN |                                                   |                                                   |
| DISCO at his post. Enter to him | DISCO at his post. Enter to him |                                                   |                                                   |
| BERNARDO BERNARDO Who's there? | BERNARDO BERNARDO Who's there? |                                                   |                                                   |
| FRANCISCO Nay, answer me: at and, and unfold yourself. BERNARDO | FRANCISCO Nay, answer me: at and, and unfold yourself. BERNARDO |                                                   |                                                   |
| BERNARDO What calls? FRANCISCO | BERNARDO What calls? FRANCISCO |                                                   |                                                   |
| I am th e lord here. I, Francisco, that | I am th e lord here. I, Francisco, that |                                                   |                                                   |
| I am sick of grief. | I am sick of grief. |                                                   |                                                   |
| Aside. BERNARDO The king! | Aside. BERNARDO The king! |                                                   |                                                   |
| O My lord? FRANCISCO Do not a | O My lord? FRANCISCO Do not a |                                                   |                                                   |
| pproach me. BERNARDO | pproach me. BERNARDO |                                                   |                                                   |
| FRANCISCO Approach me not. | FRANCISCO Approach me not. |                                                   |                                                   |
| Bernardo you had quiet guard. | Bernardo you had quiet guard. |                                                   |                                                   |
| who is there? | who is there? |                                                   |                                                   |
| Bernardo? FRANCISCO | Bernardo? FRANCISCO |                                                   |                                                   |
| I have a letter. | I have a letter. |                                                   |                                                   |
| Enter Bernardo. | Enter Bernardo. |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do here. | If you do here. |                                                   |                                                   |
| Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
| HAD QUIET GUARD? | HAD QUIET GUARD? |                                                   |                                                   |
| FRANCISCO | FRANCISCO |                                                   |                                                   |
| BERNARDO BERNARDO He. FRANCISCO | BERNARDO BERNARDO He. FRANCISCO |                                                   |                                                   |
| BERNARDO Have you had quiet guard? | BERNARDO Have you had quiet guard? |                                                   |                                                   |
| Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. | Bernardo? FRANCISCO Not a mouse stirring. BERNARDO Well, good night. |                                                   |                                                   |
| If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? | If you do meet Horatio and Marcellus. The rival of my soul, bid them make haste. FRANCISCO I think I hear them. Stand, but who is there? |                                                   |                                                   |
References

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In ACM SIGSAC Conference on Computer and Communications Security, 2016.

S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio. A neural knowledge language model. arXiv preprint arXiv:1608.00318, 2016.

A. Baevski and M. Auli. Adaptive input representations for neural language modeling. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=BHyX2ZQ0qFQ.

Y. Belinkov, S. Gehrmann, and E. Pavlick. Interpretability and analysis in neural NLP. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, pages 1–5, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-tutorials.1. URL https://aclanthology.org/2020.acl-tutorials.1.

E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In ACM Conference on Fairness, Accountability, and Transparency, 2021.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research, 3(Jan):993–1022, 2003. URL https://jmlr.csail.mit.edu/papers/v3/blei03a.html.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. V. der Plas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large Language models in machine translation. In Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 858–867, 2007.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In Advances in Neural Information Processing Systems, 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d66fcb4967418bfb8ac142f64a-Paper.pdf.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. Preprint, 2021.

C. Consonni, D. Laniado, and A. Montresor. Wikilinkgraphs: a complete, longitudinal and multi-language dataset of the wikipedia link networks. In AAAI International Conference on Web and Social Media, volume 13, 2019.

Curation. Curation corpus base, 2020.

Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Annual Meeting of the Association for Computational Linguistics, July 2019. URL https://aclanthology.org/P19-1285.
Improving language models by retrieving from trillions of tokens

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*, June 2019. URL https://aclanthology.org/N19-1423.

L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Conference on Empirical Methods in Natural Language Processing*, Nov. 2020. URL https://aclanthology.org/2020.findings-emnlp.301.

E. Grave, A. Joulin, and N. Usunier. Improving neural language models with a continuous cache. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B184E5qee.

A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

J. Gu, Y. Wang, K. Cho, and V. O. Li. Search engine guided neural machine translation. In *AAAI Conference on Artificial Intelligence*, 2018.

R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, 2020. URL https://arxiv.org/abs/1908.10396.

K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, 2020.

H. Hashemi, H. Zamani, and W. B. Croft. Guided transformer: Leveraging multiple external sources for representation learning in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1131–1140, 2020.

T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020. URL http://github.com/deepmind/dm-haiku.

G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Conference of the European Chapter of the Association for Computational Linguistics*, Apr. 2021. URL https://aclanthology.org/2021.eacl-main.74.

G. Izacard, F. Petroni, L. Hosseini, N. De Cao, S. Riedel, and E. Grave. A memory efficient baseline for open domain question answering. *arXiv preprint arXiv:2012.15156*, 2020.

S. Jain and B. C. Wallace. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357.

E. S. Jo and T. Gebru. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 306–316, 2020.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.
J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *CoRR*, 2020. URL https://arxiv.org/abs/2001.08361.

V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*, Nov. 2020. URL https://aclanthology.org/2020.emnlp-main.550.

U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

M. Komeili, K. Shuster, and J. Weston. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*, 2021.

T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural Questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 7:452–466, Mar. 2019. URL https://aclanthology.org/Q19-1026.

A. Lazaridou, A. Kuncoro, E. Gribovskaya, D. Agrawal, A. Liska, T. Terzi, M. Gimenez, C. de Mason d’Autume, S. Ruder, D. Yogatama, K. Cao, T. Kociský, S. Young, and P. Blunsom. Pitfalls of static language modelling. *CoRR*, 2021. URL https://arxiv.org/abs/2102.01951.

K. Lee, M.-W. Chang, and K. Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *Annual Meeting of the Association for Computational Linguistics*, June 2019. URL http://arxiv.org/abs/1906.00300.

K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

O. Lieber, O. Sharir, B. Lenz, and Y. Shoham. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*, 2021.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg66iCqY7.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.
T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. *Interspeech*, 2(3):1045–1048, 2010.

D. Paperno, G. Kruszewski, A. Lazaridou, N. Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Annual Meeting of the Association for Computational Linguistics*, Aug. 2016. URL https://aclanthology.org/P16-1144.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *Preprint*, 2019.

J. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d’Aumont, Y. Li, T. Terzi, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, J. Bradbury, M. Johnson, B. Hechtman, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Ossendorf, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassanis, K. Kavukcuoglu, and G. Irving. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv submission*, 2021.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models. In *IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, Jan 2009.

D. S. Sachan, S. Reddy, W. Hamilton, C. Dyer, and D. Yogatama. End-to-end training of multi-document reader and retriever for open-domain question answering. *arXiv preprint arXiv:2106.05346*, 2021.

R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green AI. *Communications of the Association for Computing Machinery*, 63(12):54–63, Nov. 2020.

M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *CoRR*, 2019. URL http://arxiv.org/abs/1909.08053.

K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv:2104.07567 [cs]*, Apr. 2021. URL http://arxiv.org/abs/2104.07567.

E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Association for Computational Linguistics*, July 2019. URL https://aclanthology.org/P19-1355.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In ACM SIGIR International Conference on Research and Development in Information Retrieval, 2006. URL http://portal.acm.org/citation.cfm?doid=1148170.1148204.

L. Weidinger, I. Gabriel, C. Griffin, M. Rauh, J. Uesato, J. Mellor, W. Isaac, P.-S. Huang, L. A. Hendricks, M. Cheng, B. Balle, J. Haas, C. Biles, L. Rimell, W. Hawkins, M. Glaese, A. Kasirzadeh, Z. Kenton, S. Brown, A. Birhane, T. Stepleton, G. Irving, and S. Legassick. Ethical and social risks of harm from language models. arXiv submission, 2021.

D. Yogatama, C. de Masson d’Autume, and L. Kong. Adaptive semiparametric language models. Transactions of the Association for Computational Linguistics, 9:362–373, 2021.

B. Zhang and R. Sennrich. Root mean square layer normalization. In Advances in Neural Information Processing Systems, 2019. URL https://proceedings.neurips.cc/paper/2019/file/1e8a19426224ca89e83cefe47f1e7f53b-Paper.pdf.

J. Zhang, M. Utiyama, E. Sumita, G. Neubig, and S. Nakamura. Guiding neural machine translation with retrieved translation pieces. In Conference of the North American Chapter of the Association for Computational Linguistics, 2018.
A. Datasets

We provide a full description of MassiveText and our extract of recent Wikipedia articles.

### A.1. Full description of MassiveText

The full breakdown of MassiveText by source and languages is given in Table 8. For a full description and analysis of MassiveText, see Rae et al. (2021).

| Source | Language | Token count (M) | Documents | Sampling weight |
|--------|----------|-----------------|-----------|----------------|
| Web    | En       | 483,002         | 604,938,816 | 0.314          |
|        | Ru       | 103,954         | 93,004,882  | 0.033          |
|        | Es       | 95,762          | 126,893,286 | 0.033          |
|        | Zh       | 95,152          | 121,813,451 | 0.033          |
|        | Fr       | 59,450          | 76,612,205  | 0.033          |
|        | De       | 57,546          | 77,242,640  | 0.033          |
|        | Pt       | 44,561          | 62,524,362  | 0.033          |
|        | It       | 35,255          | 42,565,093  | 0.033          |
|        | Sw       | 2,246           | 1,971,234   | 0.0044         |
|        | Ur       | 631             | 455,429     | 0.0011         |
| Books  | En       | 3,423,740       | 20,472,632  | 0.25           |
| News   | En       | 236,918         | 397,852,713 | 0.1            |
| Wikipedia | En | 3,977     | 6,267,214   | 0.0285         |
|        | De       | 2,155           | 3,307,818   | 0.003          |
|        | Fr       | 1,783           | 2,310,040   | 0.003          |
|        | Ru       | 1,411           | 2,767,039   | 0.003          |
|        | Es       | 1,270           | 2,885,013   | 0.003          |
|        | It       | 1,071           | 2,014,291   | 0.003          |
|        | Zh       | 927             | 1,654,772   | 0.003          |
|        | Pt       | 614             | 1,423,335   | 0.003          |
|        | Ur       | 61              | 344,811     | 0.0001         |
|        | Sw       | 15              | 58,090      | 0.0004         |
| Github | -        | 374,952         | 142,881,832 | 0.05           |
| Total  | -        | 5,026,463       | 1,792,260,998 | 1             |

Table 8 | **MassiveText dataset.** The final column indicates the sampling weight for each dataset during training. For the retrieval database, the entire dataset is used, with the exception of books for which we use a sub-sample of 4%.

### A.2. Wikipedia September 2021

We create an evaluation dataset consisting of 23 Wikipedia articles that were added or heavily edited in September 2021, after we collected our training dataset. In addition, we filter out articles that rely too heavily on templated content, using the method detailed in §2.6 to identify articles with chunks that have a high overlap with their neighbours. Fig. 10 show that little overlap remains between our test dataset and the retrieved neighbours from the training dataset. The full list of included articles is given in Table 9.
We first parse articles using mwparserfromhell\textsuperscript{5}. We then remove sections with the following titles: “references”, “external links”, “sources”, “further reading”, “see also”, “citations”, and “note”. In the remaining sections, we remove Wikilinks and remove the following templates: “reflist”, “notelist”, “notelist-ua”, “notelist-lr”, “notelist-ur”, and “notelist-lg”. We also exclude objects with the “ref” or “table” tag and clean the remaining text with the \texttt{strip_code} function. Finally, we concatenate the title and all the sections and use \texttt{\textbackslash n} to delimitate them.

B. Details on the retrieval architecture

We give details on the \textsc{Retro} architecture, and on the fine-tuning procedure we use for \textsc{Retro} fitting existing language models.

B.1. \textsc{Retro} architecture and implementation

B.1.1. Feed-forward architecture

As mentioned in the main text, the overall encoder-decoder architecture is fully feed-forward. We start with a sequence $X \in \mathbb{V}^n = (C_u)_{1 \leq u \leq l}$, and its pre-computed neighbours $(\text{RET}(C_u))_{1 \leq u \leq l}$ and returns logits in $\mathbb{R}^{nx|\mathbb{V}|}$. Along with \textsc{Attn}, \textsc{FFW}, \textsc{CCA} and \textsc{CA} operators introduced in the main text, we define the decoder embedding layer $\text{EMB} : \mathbb{V}^n \rightarrow \mathbb{R}^{nxd}$, the \textsc{Split} operator that extracts chunked intermediary embeddings $\text{Split}(H) \triangleq (H_u)_{1 \leq u \leq l} \in \mathbb{R}^{lxmxd}$ and the read-out layer $\text{READ} : \mathbb{R}^{nxd} \rightarrow \mathbb{R}^{nx|\mathbb{V}|}$. We then describe the forward pass in Algorithm 1. In addition to the usual Transformer ones, \textsc{Retro} architecture hyperparameters involves the layer indices $P_{\text{enc}}$ and $P$, at which the encoder and the decoder perform cross-attention.

B.1.2. Relative positional encoding in the chunked cross-attention layer

The \textsc{Ca} operator uses relative positional logits, that are computed from a specific relative distance separating data tokens from retrieval tokens. Indeed, we expect any retrieval neighbour $\text{RET}(C_u)^j$ and the chunk $C_u$ to be relatively well aligned, and assume that they start at the same position. Therefore, when computing $\text{Ca}(H_u^+, E_u)$, we set the distance between the data token $i \in [1, l]$ of chunk $C_u^+$ and $\textsc{RET}(C_u)^j$ to be $\text{RET}(C_u)^j - C_u^+$.

Table 9 | Full set of articles included in our \textbf{Wikipedia Sept. 2021} evaluation dataset.

| Article Name                              | Description                                      |
|-------------------------------------------|--------------------------------------------------|
| Megan Rohrer                              | Aakashavaani                                     |
| Emma Raducanu                             | Junior Eurovision Song Contest 2021              |
| Ambra Sabatini                            | Pavilion Bukit Jalil                             |
| WhyDonate                                 | Blake Desjarlais                                 |
| The Juggernaut (company)                  | 2021 All-Ireland Senior Football Championship Final |
| Angela Diaz                               | Drift-barrier hypothesis                         |
| 2020 Summer Paralympics                   | Venomics                                         |
| 2021 Afghan protests                      | Great Circle (novel)                             |
| Rexh Xhakli                               | Hurricane Ida                                    |
| Julia Laskin                              | 2021 Montenegrin episcopal enthronement protests |
| Cuijk                                     | At War With The Silverfish                       |
| Ghoubet Wind Power Station                |                                                  |

\textsuperscript{5}https://github.com/earwig/mwparserfromhell
the retrieval token \( i' \in [1, 2l] \) of \( \text{RET}(C_u)^l \) to be

\[
d(i, i') = i - i' + l - 1.
\]  

When computing the encoder cross-attentions \( C/a.sc_R/e.sc/t.sc \), we set the distance between the retrieval token \( i' \in [1, 2l] \) and the data token \( i \in [1, l] \) to be

\[
d_{enc}(i', i) = i' - i.
\]

Positional logits are obtained as a linear transform of a cosine vector computed from \((d(i, i'))_{i,i'}\), and are added to content logits, as in a regular self-attention block.

### B.1.3. Chunked cross-attention implementation

Our implementation of the \( C/c.sc/a.sc \) operator, shown in Listing 1, is based on a vectorized application of a cross-attention layer. For simplicity, we omit the multi-head attention logic and use the simplest Q,K,V attention. We omit relative positional logits computation, described above.

### B.1.4. Optional sharing of embedding matrices

We use disjoint embeddings for the encoder and decoder by default, which allows us to use a different dimensionality for the encoder (typically kept at \( d_{enc} = 896 \)) and for the decoder (that we scale up to \( d = 8192 \)). It is possible to share the embeddings, with little difference in training, as we show in the ablation section.

### B.2. Baseline to Retro model fine-tuning

As shown in Fig. 5, we found that we were able to take a pre-trained baseline transformer and add Retro through fine-tuning. In all cases, we froze all weights from pre-training and freshly initialised the retrieval encoder and cross-attention weights. In all cases, the cross-attention is added every third layer starting at layer six. The learning rate for the three smaller models was set to \( 2 \times 10^{-4} \) and half that for the larger model. We experimented with allowing the entire model to resume training during fine-tuning but consistently found that the best approach was to freeze the pre-trained model. This kept the retrieval-off performance frozen whereas when all weights were tuned the retrieval off performance would degrade.

### C. Training details and hyperparameters

We provide the hyperparameters used in the various experiments of §4.

#### C.1. Language model pre-training

In Table 10, we show the hyperparameters of the different models we train. In all cases, we train for 419,430,400,000 training tokens. The three smaller models are trained with a batch size of 256 and the largest model is trained with a batch size of 1024. The minimum learning rate is set to 0.1 times the maximum learning rate, which is shown in Table 10. The learning rate is decayed using a cosine cycle length that matches the total number of training tokens. All models are trained using AdamW (Loshchilov and Hutter, 2019) with a weight decay parameter of 0.1. The learning rate linearly increases from \( 10^{-7} \) to the maximum learning rate over the first 750 steps of training. All models use ZeRO to shard the optimiser state (Rajbhandari et al., 2020). Additional infrastructure details can be found in Rae et al. (2021).
Listing 1 | Jax implementation of the chunked cross attention, simplified.

```python
n = 128  # Sequence length
m = 16   # Chunk length
r = 32   # Retrieval length
k = 4    # Number of neighbours
d = 16   # Embedding size
l = n // m  # Number of chunks

# Parameters
Q = jnp.zeros((d, d))
K = jnp.zeros((d, d))
V = jnp.zeros((d, d))

def relative_positional_encodings(attending_length, attended_length):
    # Classical relative positional encodings
    ...

def cross_attention(chunk, neighbour):
    m, d = chunk.shape
    r, d = neighbour.shape
    queries = chunk @ Q
    keys = neighbour @ K
    logits = queries @ keys.T
    values = neighbour @ V
    return logits, values

def multi_neighbour_cross_attention(chunk, neighbours):
    m, d = chunk.shape
    k, r, d = neighbours.shape
    logits, values = jnp.vectorize(cross_attention,
                                   signature='(m,d),(r,d)->(m,r),(r,d)')(chunk, neighbours)
    assert logits.shape == (k, m, r)
    assert values.shape == (k, r, d)
    logits += relative_positional_encodings(m, r)[None, :, :]
    logits = jnp.moveaxis(logits, 0, -1).reshape((m, r * k))
    values = jnp.moveaxis(values, 0, 1).reshape((r * k, d))
    return jax.nn.softmax(logits) @ values

def multi_chunk_cross_attention(observation, neighbours):
    attending_chunks = jnp.pad(observation[n-1:],
                                ((0, m - 1), (0, 0)),
                                mode='constant').reshape(l, m, d)
    chunked_output = jnp.vectorize(multi_neighbour_cross_attention,
                                   signature='(m,d),(k,r,d)->(m,d)')(attending_chunks, neighbours)
    assert chunked_output.shape == (l, m, d)
    output = jnp.pad(chunked_output.reshape(n, d),
                     ((m - 1, 0), (0, 0)),
                     mode='constant')[n:]
    return output

observation = jnp.zeros((n, d))  # Input
neighbours = jnp.zeros((l, k, r, d))

h = multi_chunk_cross_attention(observation, neighbours)
assert h.shape == (n, d)  # Output
```
Table 10 | **Retro model hyperparameters**, along with the size of the decoder.

| Baseline | d_{model} | d_{ffw} | # heads | Head size | # layers | P | P_{ENC} | Max LR |
|----------|-----------|---------|---------|----------|----------|---|----------|--------|
| 247M     | 896       | 3584    | 16      | 64       | 12       | [6, 9, 12] | [1]    | 2×10^{-4} |
| 564M     | 1536      | 6144    | 12      | 128      | 12       | [6, 9, 12] | [1]    | 2×10^{-4} |
| 1,574M   | 2048      | 8192    | 16      | 128      | 24       | [9, 12, ..., 24] | [1]    | 2×10^{-4} |
| 7,505M   | 4096      | 16384   | 32      | 128      | 32       | [9, 12, ..., 32] | [1]    | 1×10^{-4} |

Table 11 | Hyperparameters for the Wikitext103 experiments presented in **Table 4**. We use the same learning rate schedule for the baseline and the **Retro**-fitting. For **Retro**-fitting, we reset the schedule i.e. the schedule starts from step 0, not from step 35,000.

| Model | Number of layers | 18 |
|-------|-----------------|----|
| d     | 1024            |    |
| d_{FFW} | 4096           |    |
| Key size | 64              |    |
| Value size | 64              |    |
| Number of heads | 16             |    |

| Training data | Dataset | Wikitext103train |
|---------------|---------|------------------|
| Sequence length | 3072   |                  |
| Batch size     | 128     |                  |
| Tokenizer vocabulary size | 128,000 |                |

| Optimisation | optimiser | Adam |
|--------------|-----------|------|
| Adam’s β₁    | 0.9       |      |
| Adam’s β₂    | 0.95      |      |
| Adam’s ε     | 1e-8      |      |
| Dropout rate  | 0.25      |      |

| Schedule | Learning rate start | 1e-7 |
|----------|---------------------|------|
| Learning rate max | 2.5e-4 | |
| Learning rate min | 2e-5 | |
| Warmup steps | 4,000 | |
| Cosine cycle steps | 100,000 | |

| Evaluation | Overlapping proportion | 87.5 % |

C.2. Wikitext103 comparison

We provide more details on our Wikitext103 results presented in §4.1 and **Table 4**. We train a baseline transformer on the Wikitext103 training set with the hyperparameters presented in **Table 11**. The learning rate ramps linearly from 1×10^{-7} to 2.5×10^{-4} in the first 4,000 steps, then decays to 2×10^{-5} at 100,000 steps using a cosine schedule. The baseline checkpoint at step 35,000 has the lowest perplexity on Wikitext103 valid, of 21.58, for overlapping proportion of 75% (sliding window evaluation that only uses probabilities for tokens that have at least 75% of the sequence length of context, when available). We use this checkpoint for all our baseline and kNN-LM numbers reported in **Table 4**, except that **Table 4** reports for an overlapping proportion of 87.5 %, which slightly lowers the perplexity of our baseline to 21.53 on Wikitext103 valid.

We also use the 35,000 step baseline checkpoint as initialization for a **Retro**fit, which otherwise uses the same optimiser and schedule hyperparameters but only trains the new retrieval weights, as explained in §4.2. Our best **Retro**fit checkpoint has a Wikitext103 valid perplexity 18.46, when retrieving from Wikipedia. We use this **Retro** checkpoint in **Table 4** for all other retrieval sets. The evaluation curves for our baseline and **Retro**fit is shown if Fig. 7 (left). In this particular case,
because Wikitext103 is quite small, training a RETRO model from scratch led to weaker results than the baseline, at least when retrieving from Wikipedia, as we couldn’t find an effective way to mitigate the increased over-fitting due to the additional weights of RETRO.

We also re-implement \(k\)NN-LM using the same tokenizer and dataset that we use for our baseline and RETROfitting experiments. \(k\)NN-LM has probabilities \(p_{k\text{NN-LM}} = \lambda p_{LM} + (1 - \lambda)p_{k\text{NN}}\) with \(p_{k\text{NN}}(n_k) \propto \exp(-\alpha d_k)\). To tune \(\lambda\) and \(\alpha\), we begin with \(\alpha = 0.0012\), which corresponds to the inverse of the standard deviation of the norm of the embeddings that we use as keys and queries for \(k\)NN-LM. We find the best \(\lambda = 0.118\). We then find the best \(\alpha = 0.00785\) for that value of \(\lambda\). Fig. 7 center and right respectively show the perplexity of \(k\)NN-LM as a function of \(\lambda\) and \(\alpha\).

Figure 7 | Wikitext103 valid perplexities. Left: Baseline and RETROfit (initialized from baseline’s checkpoint at 35,000 steps) perplexities as a function of training steps. Center and right: \(k\)NN-LM perplexity as a function of \(\lambda\) (for \(\alpha = 0.0012\)) and \(\alpha\) (for \(\lambda = 0.12\)) respectively.

C.3. RETROfitting baseline models experiments

In Table 12, we give the hyperparameters used for RETROfitting the models on Massive Text.

Table 12 | Hyperparameters for the RETROfitting experiments

| Model | Layers with RETRO-block (\(P\)) | Learning rate | Batch size |
|-------|---------------------------------|---------------|------------|
| 172M  | Every 3\textsuperscript{rd} from 6 | \(2 \times 10^{-4} \rightarrow 2 \times 10^{-5}\) | 256 |
| 425M  | Every 3\textsuperscript{rd} from 6 | \(2 \times 10^{-4} \rightarrow 2 \times 10^{-5}\) | 256 |
| 1.5B  | Every 3\textsuperscript{rd} from 6 | \(2 \times 10^{-4} \rightarrow 2 \times 10^{-5}\) | 256 |
| 7.5B  | Every 3\textsuperscript{rd} from 6 | \(1 \times 10^{-4} \rightarrow 1 \times 10^{-5}\) | 256 |

C.4. Question answering experiments

We fine-tune our 7.5B RETRO model for 25,000 steps, using a batch size of 128, a learning rate cosine scheduled from \(10^{-6}\) to \(10^{-7}\), with a linear ramp of 750 steps. We use dropout in the decoder only, as it performs better than using dropout in both the encoder and the decoder. Each neighbour is formatted as title: {title}, source: {source}. We use the top 20 neighbours from DPR when training and evaluating.
Table 13 | **Performance of RETRO for different variants.** Model performance on C4 evaluation set, measured in bytes-per-bits, for a 247M parameter model trained with a 157 billion token schedule.

| Ablation group     | Ablation                          | C4 eval bpb |
|--------------------|-----------------------------------|-------------|
| Model              | RETRO                             | 0.822       |
|                    | No query conditioning              | 0.829       |
|                    | No CA positional encodings         | 0.826       |
|                    | Shared embeddings                  | 0.823       |
|                    | 6-layer encoder                   | 0.821       |
| Retrieval values   | Neighbours N                       | 0.950       |
|                    | Continuations F                    | 0.895       |
|                    | No retrieval                       | 0.987       |
| Training neighbours| 1 training neighbours              | 0.858       |
|                    | 4 training neighbours              | 0.847       |
| Cross attention position | CA top layer (1/12)     | 0.827       |
|                    | CA mid layer (6/12)                | 0.823       |
|                    | CA top layer (12/12)               | 0.831       |
|                    | CA all layers                      | 0.860       |
|                    | CA every 3 from 1                  | 0.823       |

**D. Model ablations**

We validate important design choices by evaluating what happens when we do not include them. We use the 247M parameter model for all experiments and we train on a compressed 157 billion token schedule for all ablation experiments. We describe results relative to the default settings presented in the main text and recalled here. We report C4 evaluation loss at the end of the training process, and also compares how the evaluation loss decrease versus the training time, measured relatively to the baseline training time. Results are reported in Fig. 8 and Table 13.

**Using relative encodings in cross-attention.** Using relative encodings in cross-attention, as described in §B.1.2, provides a pure improvement both in the number of steps to reach a given performance and computational efficiency.

**Conditioning the encoder on the previous chunk.** Conditioning the encoder on the previous chunk’s intermediate embeddings, as described in §B.1.1, provides a pure improvement both in term of number of steps and computational efficiency.

**Sharing embeddings.** Sharing embeddings across the encoder and the decoder does not affect performance. This motivates us using separate embeddings, as it allows to have a narrower encoder than decoder as we scale up the decoder size.

**Attending neighbours and their continuation.** RETRO models are trained by attending, for a given chunk, to both the neighbours of the preceding chunk and their continuation in time. We measure how training and evaluating RETRO models on neighbours only and their continuation only affects performance. Overall, attending to neighbours only provides 22% of the performance improvement due to retrieval in RETRO, while attending the future of the neighbours gives 56% of
Improving language models by retrieving from trillions of tokens

Figure 8 | Computational efficiency for different variants. We report the training curves plotting C4 evaluation bytes per bits against time, relative to the time taken to train the baseline RETRO model. Overall, our design choices are optimal in term of computational efficiency.

- **Training a deeper encoder.** All models in the text use a relatively small RETRO encoder. We experimented with a 3x deeper encoder. We found that this resulted in a tiny decrease in loss—0.15% at the cost of a larger training time (+20%). Overall, using a shallow encoder is the best choice in term of training efficiency.

- **Training with multiple neighbours.** We measure the effect of training on a single retrieved neighbour, as well as training on 4 neighbours (RETRO uses 2 neighbours in training). Training on a single neighbour results in a large decrease in performance, while training on 4 neighbours does not give substantial performance improvement at the end of training, but induces a large computational overhead. Overall, we find that using 2 neighbours is the best choice in term of training efficiency. Furthermore, evaluation can be done with additional neighbours.

- **Frequency of cross-attention.** We measure how the frequency of cross-attention in the decoder affects performance. Overall, attending only once at the top or the bottom layer is a bad choice, while attending once on a mid-depth layer is relatively sound. We choose to have cross-attention every 3 layer as this provides a good trade-off between performance and run-time.
E. Qualitative experiments

We illustrate the usage of RETRO models by looking at the perplexity of evaluation samples and by producing samples autoregressively.

E.1. Inspecting neighbours and perplexities on evaluation data

To build an intuition of what kind of information is leveraged by RETRO models, we suggest to have a closer look at a few evaluation documents and the corresponding retrieved data in Tables 16, 17, 18 and 19. In these tables, the 4 rows corresponds to the first 4 chunks of the documents. The left-most column shows the chunk $C_u$ from the document being evaluated, where each token is coloured by the negative cross entropy loss difference $L_{\text{RETO}[\text{OFF}]} - L_{\text{RETO}}$, a positive value, coloured in yellow, indicates that RETRO performs better when it has access to neighbours data. The second columns also shows the evaluated chunk $C_u$ but where each token $i$ is coloured by the length of the longest common prefix (LCP) with the preceding neighbours, i.e. the largest integer $j$ such that the prefix $(x_{i-j-1}, \ldots, x_i)$ also appears in RET$(C_{u-1})$. Conversely, columns three and four show the first two neighbours and their continuation, respectively $[N_u^1, F_u^1]$ and $[N_u^2, F_u^2]$ coloured by LCP with subsequent chunk $C_{u+1}$. LCP colouring helps to visually identify where the evaluated document overlaps the retrieved data. Note that the first chunk, $C_1$, in the second column is not coloured as it does not have any preceding neighbours to compute LCP with. Similarly, we do not show the neighbours of the fourth chunk, as these are not used to condition any of the first four chunks.

Our qualitative analysis exhibits two major behaviors.

Firstly, we observe that sometimes, specific facts in $C_u$ can be extracted from the preceding neighbours RET$(C_{u-1})$ and that this can correspond to significant reduction in loss from the RETRO model for the corresponding tokens. Some examples of such behavior include the journal name *Publishers Weekly* in Table 16, the football team name *Tyrone* in Table 17 or the event dates *25 August to 6 September 2020* in Table 18. In these three examples, the evaluated data consists of recent Wikipedia articles written in September 2021, after we built our retrieval dataset (see section §A.2). Yet, relevant information to predict this new data was available in the pre-existing retrieval data and the RETRO model seems to be able to correctly leverage it.

On the other hand, we also observe that some of the evaluation data can partially leak in our training and retrieval data, despite the use of deduplication. RETRO can dramatically exploit such leakage. Table 19 illustrates this behavior, where the chunks $C_2$ and $C_3$ largely overlaps RET$(C_1)$ and RET$(C_2)$ respectively, up to small formatting differences, which leads to much lower RETRO loss for all the corresponding tokens. Fig. 6 shows that it is possible to quantify how much of the RETRO loss reduction is due to each of these two behaviors, by filtering out evaluation chunks that overlaps with the retrieval set.

E.2. Inspecting samples

We can follow the same procedure as above on samples generated using RETRO models, in order to better understand where retrieval data had an influence on sampling. We show examples of samples obtained using the 7.5B RETRO model in Table 6, 7, 20 and 21.

E.3. Neighbour quantification

To quantify a notion of distance between the source document and the retrieved chunks, we can ask the distance between source articles when retrieving only from Wikipedia. Consonni et al. (2019)
Figure 9 | Wikipedia link-distance between retrieved articles. For each sequence, chunk combination we compute the link distance between the target and the top-5 neighbours using only Wikipedia. The rank shows the relative neighbour distance, where rank-1 is the first neighbour and rank 5 is the fifth. The different colours represent link distance. Because we do not retrieve from the same document, 1 is the smallest value. We find, on average, the distance between random articles with a path between them is over 5.0

F. Complementary quantitative results

We report tables corresponding to quantitative figures of the main text, as well as further filtered language model results on the Pile.

F.1. Main text datasets

We report the performance of RETRO and baseline models, measured in bits-per-bytes on evaluation set, in Table 14.

F.2. The Pile

In Fig. 4, we compare RETRO against Jurassic-1 (Lieber et al., 2021). The full bits-per-bytes results are reported in Table 15.

F.3. Filtered results

Distribution of leaked chunks in our main evaluation sets. We evaluate leakage between the evaluation sets and the training set by measuring the proportion of evaluation chunks with a certain
Table 14 | Full results for the main language modelling datasets. First three sets of rows correspond to Fig. 1, last set of rows to Fig. 3.

| Dataset | Baseline | Retro [Off] | Retro [On] |
|---------|----------|-------------|------------|
|         | 172M  | 425M  | 1.5B  | 7.5B  | 172M  | 425M  | 1.5B  | 7.5B  | 172M  | 425M  | 1.5B  | 7.5B  |
| C4 Eval bpb | 0.98 | 0.92 | 0.84 | 0.78 | 0.98 | 0.92 | 0.84 | 0.78 | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (900B) | - | - | - | - | - | - | - | - | 0.88 | 0.83 | 0.76 | 0.71 |
| C4 Eval bpb (360B) | - | - | - | - | - | - | - | - | 0.92 | 0.87 | 0.80 | 0.74 |
| C4 Eval bpb (180B) | - | - | - | - | - | - | - | - | 0.94 | 0.89 | 0.81 | 0.75 |
| C4 Eval bpb (90B) | - | - | - | - | - | - | - | - | 0.95 | 0.89 | 0.82 | 0.76 |
| C4 Eval bpb (36B) | - | - | - | - | - | - | - | - | 0.96 | 0.90 | 0.83 | 0.77 |
| C4 Eval bpb (18B) | - | - | - | - | - | - | - | - | 0.96 | 0.91 | 0.83 | 0.77 |
| C4 Eval bpb (9B) | - | - | - | - | - | - | - | - | 0.96 | 0.91 | 0.83 | 0.77 |
| C4 Eval bpb (4B) | - | - | - | - | - | - | - | - | 0.97 | 0.91 | 0.84 | 0.78 |
| C4 Eval bpb (2B) | - | - | - | - | - | - | - | - | 0.97 | 0.91 | 0.84 | 0.78 |
| C4 Eval bpb (k = 1) | - | - | - | - | - | - | - | - | 0.84 | 0.79 | 0.73 | 0.67 |
| C4 Eval bpb (k = 2) | - | - | - | - | - | - | - | - | 0.83 | 0.78 | 0.72 | 0.67 |
| C4 Eval bpb (k = 3) | - | - | - | - | - | - | - | - | 0.82 | 0.78 | 0.71 | 0.66 |
| C4 Eval bpb (k = 4) | - | - | - | - | - | - | - | - | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (k = 5) | - | - | - | - | - | - | - | - | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (k = 10) | - | - | - | - | - | - | - | - | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (k = 20) | - | - | - | - | - | - | - | - | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (k = 30) | - | - | - | - | - | - | - | - | 0.82 | 0.77 | 0.71 | 0.66 |
| C4 Eval bpb (k = 40) | - | - | - | - | - | - | - | - | 0.83 | 0.77 | 0.71 | 0.65 |
| C4 Eval bpb (k = 50) | - | - | - | - | - | - | - | - | 0.83 | 0.78 | 0.71 | 0.66 |
| C4 Eval bpb (k = 60) | - | - | - | - | - | - | - | - | 0.84 | 0.78 | 0.72 | 0.66 |
| C4 Eval bpb (k = 70) | - | - | - | - | - | - | - | - | 0.84 | 0.79 | 0.72 | 0.66 |
| C4 Eval bpb (k = 80) | - | - | - | - | - | - | - | - | 0.85 | 0.79 | 0.73 | 0.66 |
| C4 Eval bpb (k = 90) | - | - | - | - | - | - | - | - | 0.85 | 0.79 | 0.73 | 0.66 |
| C4 Eval bpb (k = 100) | - | - | - | - | - | - | - | - | 0.85 | 0.79 | 0.73 | 0.66 |
| Lambada Accuracy | 0.42 | 0.51 | 0.61 | 0.69 | 0.47 | 0.54 | 0.63 | 0.70 | 0.52 | 0.60 | 0.67 | 0.73 |
| Curation Corpus bpb | 0.69 | 0.63 | 0.56 | 0.52 | 0.68 | 0.64 | 0.57 | 0.51 | 0.66 | 0.61 | 0.55 | 0.50 |
| Wikitext103 Perplexity | 25.62 | 19.29 | 13.98 | 10.65 | 25.88 | 19.78 | 13.89 | 10.40 | 3.32 | 2.96 | 2.53 | 2.22 |
| Wikipedia Sept. 2021 bpb | 0.85 | 0.78 | 0.71 | 0.65 | 0.86 | 0.79 | 0.71 | 0.65 | 0.79 | 0.73 | 0.66 | 0.61 |

overlap \( r(C) \). We show histograms in Fig. 10. We can see that C4 has some slight overlaps between train and evaluation. Similarly, chunks of Wikitext103 appear in the training set despite having removed the actual Wikitext103 evaluation documents from the training set. On the other hand, our Wikipedia September 21 dataset shows almost no leakage (data being original documents that did not exist at training data creation), and neither does Curation Corpus.

**Filtered results on the Pile.** We report chunk overlap distribution and filtered performance curves on the Pile in Fig. 12 and Fig. 11, respectively. The qualitative interpretation of the filtered curves is the same: Retro models exploit leakage more, but the performance improvement they provide remains significant even on original chunks that haven’t been observed in the training set.
Table 15 | **Full results on The Pile, measured in bits-per-bytes.** Jurassic-1 and GPT-3 numbers are taken from Lieber et al. (2021). Gopher numbers are taken from Rae et al. (2021).

| Subset                      | 7B Baseline (Ours) | GPT-3 | Jurassic-1 | Gopher | 7.5B RETRO |
|-----------------------------|--------------------|-------|------------|--------|------------|
| arxiv                       | 0.742              | 0.838 | 0.680      | 0.641  | 0.714      |
| books3                      | 0.792              | 0.802 | 0.835      | 0.706  | 0.653      |
| dm_mathematics              | 1.177              | 1.371 | 1.037      | 1.135  | 1.164      |
| freelaw                     | 0.576              | 0.612 | 0.514      | 0.506  | 0.499      |
| github                      | 0.420              | 0.645 | 0.358      | 0.367  | 0.199      |
| gutenberg_pg_19             | 0.803              | 1.163 | 0.890      | 0.652  | 0.400      |
| hackernews                  | 0.971              | 0.975 | 0.869      | 0.888  | 0.860      |
| nh_exporter                 | 0.650              | 0.612 | 0.590      | 0.590  | 0.635      |
| opensubtitles               | 0.974              | 0.932 | 0.879      | 0.894  | 0.930      |
| philpapers                  | 0.760              | 0.723 | 0.742      | 0.682  | 0.699      |
| pile_cc                     | 0.771              | 0.698 | 0.669      | 0.688  | 0.626      |
| pubmed_abstracts            | 0.639              | 0.625 | 0.587      | 0.578  | 0.542      |
| pubmed_central              | 0.588              | 0.690 | 0.579      | 0.512  | 0.419      |
| stackexchange               | 0.714              | 0.773 | 0.655      | 0.638  | 0.624      |
| ubuntu_irc                  | 1.200              | 0.946 | 0.857      | 1.081  | 1.178      |
| uspto_backgrounds           | 0.603              | 0.566 | 0.537      | 0.545  | 0.583      |

Figure 10 | **Distribution of the overlap between evaluation and train chunks** for C4, Curation Corpus, Wikitext103 and Wikipedia Sept. 2021.
Figure 11 | Filtered evaluation losses on the Pile, with baseline Transformers and Retro.
Figure 12 | Distribution of the overlap between evaluation and train chunks for the Pile evaluation sets.
Table 16 | Great Circle (novel), from Wikipedia September 21. The article is about a recent novel and chunks $C_3$ and $C_4$ are specifically about its reception. The name Publishers Weekly of the journal that reviewed the novel appears both in the neighbours $[N_1^3, F_1^3]$, $[N_2^3, F_2^3]$ of chunk $C_3$ and in the subsequent chunk $C_4$, where the loss for those tokens is significantly reduced by Retro.

| Chunk | Name | Neighbours | Neighbours | Neighbours | Neighbours |
|-------|------|------------|------------|------------|------------|
| $C_3$ colored by loss difference $\Delta_{Retro}(Dw]$ | Great Circle (novel) | Great Circle (novel) | Great Circle (novel) | Great Circle (novel) | Great Circle (novel) |
| $C_4$ colored by LCP with Retro($C_3$-1) LCP = 0, 2, 3, 4, 5 | | | | | |
| $[N_1^3, F_1^3]$ colored by LCP with $C_4$ LCP = 0, 2, 3, 4, 5 | | | | | |
| $[N_2^3, F_2^3]$ colored by LCP with $C_4$ LCP = 0, 2, 3, 4, 5 | | | | | |

About the disappeared 20th-century aviator Marian Graves, while the other is about the struggling 21st-century Hollywood actress Hadley Baxter, who is attempting to make a film about Marian. Hadley's narrative is in the first-person, while Marian's sections are told in the third-person.

The Dutch House (novel) is a 2020 novel by Ann Patchett. It was published by Harper on September 24, 2019. It tells the story of a mother and a daughter who made their fortune in the tobacco industry. Cyril Conroy, a self-made estate owner, based closely on her own youthful experiences. (She plans the film to be the first of two parts, the second dealing with the aftermath of the first event.) Byrne plays a young film student named Julie (Hogg's daughter), who stars her artistic identity with high hopes of making a movie about a boy named Tony, living in a working-class Sunderland, who adores his mother—"an almost obsessed woman," as eager Julie tells her advice. Her idealism is evident in the first part of the novel.

The Dutch House (novel) is a 2020 novel by Ann Patchett. It was published by Harper on September 24, 2019. It tells the story of a mother and a daughter who made their fortune in the tobacco industry. Cyril Conroy, a self-made estate owner, based closely on her own youthful experiences. (She plans the film to be the first of two parts, the second dealing with the aftermath of the first event.) Byrne plays a young film student named Julie (Hogg's daughter), who stars her artistic identity with high hopes of making a movie about a boy named Tony, living in a working-class Sunderland, who adores his mother—"an almost obsessed woman," as eager Julie tells her advice. Her idealism is evident in the first part of the novel.

The Dutch House (novel) is a 2020 novel by Ann Patchett. It was published by Harper on September 24, 2019. It tells the story of a mother and a daughter who made their fortune in the tobacco industry. Cyril Conroy, a self-made estate owner, based closely on her own youthful experiences. (She plans the film to be the first of two parts, the second dealing with the aftermath of the first event.) Byrne plays a young film student named Julie (Hogg's daughter), who stars her artistic identity with high hopes of making a movie about a boy named Tony, living in a working-class Sunderland, who adores his mother—"an almost obsessed woman," as eager Julie tells her advice. Her idealism is evident in the first part of the novel.

The Dutch House (novel) is a 2020 novel by Ann Patchett. It was published by Harper on September 24, 2019. It tells the story of a mother and a daughter who made their fortune in the tobacco industry. Cyril Conroy, a self-made estate owner, based closely on her own youthful experiences. (She plans the film to be the first of two parts, the second dealing with the aftermath of the first event.) Byrne plays a young film student named Julie (Hogg's daughter), who stars her artistic identity with high hopes of making a movie about a boy named Tony, living in a working-class Sunderland, who adores his mother—"an almost obsessed woman," as eager Julie tells her advice. Her idealism is evident in the first part of the novel.
Table 17 | All-Ireland Senior Football Championship Final, from Wikipedia September 21. The name of the team **Tyrone** appears both in the second neighbours \([N_1^2, F_1^1]\) of chunk C1 and in the subsequent chunk C2, where the loss for those tokens is significantly reduced by RETRO.

![Improving language models by retrieving from trillions of tokens](image-url)
Table 18 | 2020 Summer Paralympics, from Wikipedia September 21. The original dates of the event, 25 August to 6 September 2020, appears both in the neighbors \( [N^1, P^1] \) and \( [N^2, P^2] \) of chunk \( C_1 \) and in the subsequent chunk \( C_2 \), where the loss for those tokens is significantly reduced by RETRO. Interestingly, in this case, the neighbors were written at a time when the event hadn’t yet been postponed.

\[
\begin{align*}
C_0 \text{ colored by loss difference } & \quad \delta_{\text{loss}}(C_0) = 2.0, \, \delta_{\text{pos}}(C_0) = 0.8, \, \delta_{\text{neg}}(C_0) = 0.5 \\
C_1 \text{ colored by LCP with } \text{RET}(C_0-1) & \quad \text{LCP} = \{ 1, 2, 3, 4 \}, \, \delta_{\text{pos}}(C_1) = 0.5 \\
[N^1, P^1] \text{ colored by LCP with } C_{\text{rep}} & \quad \text{LCP} = \{ 1, 2, 3, 4 \}, \, \delta_{\text{pos}}(C_{\text{rep}}) = 0.5 \\
[N^2, P^2] \text{ colored by LCP with } C_{\text{rep}} & \quad \text{LCP} = \{ 1, 2, 3, 4 \}, \, \delta_{\text{pos}}(C_{\text{rep}}) = 0.5
\end{align*}
\]

2020 Summer ParalympicsThe brand as the Tokyo 2020 Paralympic Games, s, was an international multi-sport paraparsons event held from 24 August to 6 September 2021 in Tokyo, Japan. They were the 16th Summer Paralympic Games as organized by the International Paralympic Committee (IPC).

Originally scheduled to take place from 25 August to 6 September 2020, both the Summer Olympic Games and Paralympics were postponed by one year due to the COVID-19 pandemic until referred to as Tokyo 2020 for marketing and branding purposes. As with the Olympics, the Games were initially held behind closed doors with no outside spectators due to a state of emergency in the Greater Tokyo Area and other prefectures. The Games were the second Summer Paralympics hosted by Tokyo since 1964, and the third Paralympics held in Japan overall since 1998. The Paralympics Paralympians were postponed by one year due to the COVID-19 pandemic, with the rescheduled Games still referred to as Tokyo 2020 for marketing and branding purposes. As with the Olympics, the Games were initially held behind closed doors with no outside spectators due to a state of emergency in the Greater Tokyo Area and other prefectures. The Games were the second Summer Paralympics hosted by Tokyo since 1964, and the third Paralympics held in Japan overall since 1998.

Summer Paralympics. The 2020 Summer Paralympics are an upcoming major international multi-sport event for athletes with disabilities governed by the International Paralympic Committee. Scheduled as the 16th Summer Paralympic Games, it is planned to be held in Tokyo, Japan from 25 August to 6 September 2020. Originally due to take place between 25 August and 6 September 2020. On 24 March 2020, the IOC announced that the Tokyo Organizing Committee of the Olympic and Paralympic Games had officially announced that the 2020 Summer Olympics and 2020 Summer Paralympics would be postponed to 2021, due to the COVID-19 pandemic, marking the first time that the Paralympics had been postponed. They will still be publicly marketed as Tokyo 2020, have now been rescheduled to run 23 July to 8 August 2021 in Tokyo, Japan. The Games were postponed in March 2020 as a result of the worldwide Covid-19 pandemic, although they will still keep the name Tokyo 2020 for marketing and branding purposes. This will be the first time the Olympic Games have been postponed rather than cancelled.

Originally scheduled to take place from 25 August to 6 September 2020, both the 2020 Summer Olympic Games and Paralympics were postponed by one year due to the COVID-19 pandemic, with the rescheduled Games still referred to as Tokyo 2020 for marketing and branding purposes. As with the Olympics, the Games were initially held behind closed doors with no outside spectators due to a state of emergency in the Greater Tokyo Area and other prefectures. The Games were the second Summer Paralympics hosted by Tokyo since 1964, and the third Paralympics held in Japan overall since 1998. The Games were the second Summer Paralympic Games as organized by the International Paralympic Committee (IPC).

Originally scheduled to take place from 25 August to 6 September 2020, both the 2020 Summer Olympic Games and Paralympics were postponed by one year due to the COVID-19 pandemic, with the rescheduled Games still referred to as Tokyo 2020 for marketing and branding purposes. As with the Olympics, the Games were initially held behind closed doors with no outside spectators due to a state of emergency in the Greater Tokyo Area and other prefectures. The Games were the second Summer Paralympic Games hosted by Tokyo since 1964, and the third Paralympics held in Japan overall since the 1998 Winter Paralympic Games in Nagano. The Games featured 539 medal events in 22 sports, with badminton and track and field both making their Paralympic debut to replace football 4-side and sailing. China topped the medal table for the fifth consecutive Paralympics, with 96 gold and 207 total medals. Great Britain finished second, for the ninth time.
Table 19 | Daniel Radcliffe, from Wikitext103Valid, retrieval data from c4. The chunks $C_2$ and $C_3$ are almost entirely retrieved from neighbours $\{N_1, F_1\}$ and $\{N_2, F_2\}$ respectively, up to formatting differences, which dramatically reduces the loss for these tokens. This example illustrates that when training data leaks into evaluation sets despite deduplication, our Retro model can directly exploit this leakage.

| $C_0$ colored by loss difference $\frac{\text{Ret}_0(\text{Oov})}{\text{Ret}_0(\text{Oov})} > 0.5$ | $C_0$ colored by LCP with RET($C_0$–1) LCP = $0, 1, 2, 3, 4, 5$ | $[N_2, F_2]$ colored by LCP with $C_{ret+1}$ LCP = $0, 1, 2, 3, 4, 5$ | $[N_2, F_2]$ colored by LCP with $C_{ret+1}$ LCP = $0, 1, 2, 3, 4, 5$
| --- | --- | --- | --- |
| $\text{Daniel Radcliffe (born 23 July 1989)}$ is an English actor who rose to prominence as the title character in the Harry Potter film series. He made his acting debut at 10 years of age in BBC One's 1999 television film David Copperfield, followed by his cinematic debut in 2001. He was cast as Harry Potter in the first Harry Potter film, and starred in the series for 10 years until the release of the eighth and final film in 2011. Radcliffe began to branch out to stage acting in 2007, starring in the London and New York productions of Equus. | $\text{Daniel Radcliffe (born 23 July 1989)}$ is an English actor who rose to prominence as the title character in the Harry Potter film series. He made his acting debut at 10 years of age in BBC One's 1999 television film David Copperfield, followed by his cinematic debut in 2001. The Tailor of Panama. At age 11, he was cast as Harry Potter in the first Harry Potter film, and starred in the series for 10 years until the release of the eighth and final film in 2011. Radcliffe began to branch out to stage acting in 2007, starring in the London and New York productions of Equus. | $\text{Daniel Radcliffe (born 23 July 1989)}$ is an English actor who rose to prominence as the title character in the Harry Potter film series. He made his acting debut at 10 years of age in BBC One's 1999 television film David Copperfield, followed by his cinematic debut in 2001. The Tailor of Panama. At age 11, he was cast as Harry Potter in the first Harry Potter film, and starred in the series for 10 years until the release of the eighth and final film in 2011. Radcliffe began to branch out to stage acting in 2007, starring in the London and New York productions of Equus. | $\text{Daniel Radcliffe (born 23 July 1989)}$ is an English actor who rose to prominence as the title character in the Harry Potter film series. He made his acting debut at 10 years of age in BBC One's 1999 television film David Copperfield, followed by his cinematic debut in 2001. The Tailor of Panama. At age 11, he was cast as Harry Potter in the first Harry Potter film, and starred in the series for 10 years until the release of the eighth and final film in 2011. Radcliffe began to branch out to stage acting in 2007, starring in the London and New York productions of Equus. |
Table 20 | Déclaration des droits de l’homme: Article premier. The Retro[Off] sample has correct syntax and is almost plausible but is hallucinated. The Retro[On] sample is correctly copied from neighbour data, and robustly re-formatted according to our prompt.

| Prompt and sample of Retro[Off] | Prompt and sample of Retro[On] colored by LCP with $C_{\text{on}}$ | [N$^2$, F$^2$] colored by LCP with $C_{\text{on}}$ | [N$^2$, F$^2$] colored by LCP with $C_{\text{on}}$ |
|---------------------------------|-------------------------------------------------|---------------------------------|---------------------------------|
| Article premier - Les hommes naissent et demeurent libres et égaux en droits. Les distinctions sociales ne peuvent être fondées que sur l’utilité commune. Article 2. - Le but de toute association politique est la conservation des droits naturels et | ***imprescriptibles de l’homme,*** ***et par conséquent la garantie à chacun des droits suivants :*** Article 3. - La propriété est un droit inviolable et sa conservation. Toute personne a le droit de procéder à sa propre conservation. Article 4. - Le but de toute association est la défense des droits de l’homme et du citoyen. Tout citoyen a le droit de participer à l’administration des affaires publiques. Article 5. - L’impunité n’a jamais été et ne sera jamais une fin en elle-même. L’impunité est un droit inviolable et sa conservation. | ***imprescriptibles de l’homme,*** ***et par conséquent la garantie à chacun des droits suivants :*** Article 3. - La propriété est un droit inviolable et sa conservation. Toute personne a le droit de procéder à sa propre conservation. Article 4. - Le but de toute association est la défense des droits de l’homme et du citoyen. Tout citoyen a le droit de participer à l’administration des affaires publiques. Article 5. - L’impunité n’a jamais été et ne sera jamais une fin en elle-même. L’impunité est un droit inviolable et sa conservation. | ***imprescriptibles de l’homme,*** ***et par conséquent la garantie à chacun des droits suivants :*** Article 3. - La propriété est un droit inviolable et sa conservation. Toute personne a le droit de procéder à sa propre conservation. Article 4. - Le but de toute association est la défense des droits de l’homme et du citoyen. Tout citoyen a le droit de participer à l’administration des affaires publiques. Article 5. - L’impunité n’a jamais été et ne sera jamais une fin en elle-même. L’impunité est un droit inviolable et sa conservation. |
Table 21 | Sample - Decimals of $\pi$. The RETRO[OFF] sample quickly diverges two digits after the end of the prompt whereas RETRO[ON] correctly outputs a large number of $\pi$ digits, directly copied from the neighbours data.

| Prompt and sample of RETRO[OFF] | Prompt and sample of RETRO[ON] | $|N^2, F_2^2|$ colored by LCP with $C_{n-1}$ | $|N^2, F_2^2|$ colored by LCP with $C_{n+1}$ |
|--------------------------------|--------------------------------|---------------------------------------------|---------------------------------------------|
| Pi = 3, 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 | Pi = 3, 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 | 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 | 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 |
| LCP = 0. 1. 3. 4. 5. 6. 7. 8. 9. | LCP = 0. 1. 3. 4. 5. 6. 7. 8. 9. | 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 | 1415926535 8979323846 2643383279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 |
| 46 2643138279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 | 46 2643138279 5028841971 69999375 10 5829974944 5923078164 06286 20899 8628034825 3421170679 |

**Improving language models by retrieving from trillions of tokens**