Fewer Truncations Improve Language Modeling

Hantian Ding [1] **Zijian Wang** [1] **Giovanni Paolini** [1] **Varun Kumar** [1] **Anoop Deoras** [1] **Dan Roth** [1] **Stefano Soatto** [1]

Abstract

In large language model training, input documents are typically concatenated together and then split into sequences of equal length to avoid padding tokens. Despite its efficiency, the concatenation approach compromises data integrity—it inevitably breaks many documents into incomplete pieces, leading to excessive truncations that hinder the model from learning to compose logically coherent and factually consistent content that is grounded on the complete context. To address the issue, we propose Best-fit Packing, a scalable and efficient method that packs documents into training sequences through length-aware combinatorial optimization. Our method completely eliminates unnecessary truncations while retaining the same training efficiency as concatenation. Empirical results from both text and code pre-training show that our method achieves superior performance (e.g., relatively +4.7% on reading comprehension; +16.8% in context following; and +9.2% on program synthesis), and reduces closed-domain hallucination effectively by up to 58.3%.

1. Introduction

Large language models (LLMs) have achieved unprecedented success on a number of natural language processing and coding benchmarks (Brown et al., 2020; Chen et al., 2021) and in complex real-world tasks (Ouyang et al., 2022). This remarkable progress is driven by large-scale pre-training over a massive amount of unlabeled documents. When formatting the training inputs, naïvely padding every document to a fixed length is inefficient as short documents lead to an excessive amount of padding. Instead, the common practice is to concatenate all documents together and then split them into sequences of exactly the model's context length. A sentinel token (e.g., `<|endoftext|>`) is often added at the end of each document to indicate document boundaries within each training sequence. This concatenate-then-split (hereafter "concatenation") approach has been widely adopted in training language models in both natural language (Brown et al., 2020; Chowdhery et al., 2022; Rae et al., 2021; Zhang et al., 2022; Touvron et al., 2023b; Scao et al., 2022) and programming language (Nijkamp et al., 2023), thanks to its optimal training efficiency as no padding is needed. However, such training efficiency comes at the expense of *data integrity*—documents that could have been processed in their entirety by the model are instead fragmented into independent segments, which naturally results in *loss of information*. Further, truncation reduces the amount of context within each segment, causing next-token prediction to be potentially *ungrounded* to its context, and thus making models more *prone to hallucination*.

We argue that data integrity is the key towards better language modeling. To realize this, we first show that it is feasible to group billions of documents at pre-training scale into sequences in a way that is as token-efficient as concatenation without incurring *any* unnecessary truncation: only documents beyond model's context length need to be segmented. Data grouping strategies that preserve the entirety of individual samples have been widely adopted for encoder-only and encoder-decoder models (Liu et al., 2019; Raffel et al., 2020b; Krell et al., 2021). Nonetheless, these existing strategies either exhibit limited scalability or compromise training efficiency, making them less favorable compared to the concatenation method in LLM training at scale.

In response, we propose *Best-fit Packing* to eliminate unnecessary document truncations without sacrificing training efficiency. As illustrated in Figure 1, we first segment long documents into multiple chunks by model's context length. Documents shorter than that are kept as singleton chunks. Next, we pack all the chunks into training sequences without breaking them any further. This step is essentially an instance of *the bin packing problem*[1], which is NP-hard. We employ Best-Fit-Decreasing (Eilon & Christofides, 1971), an approximation algorithm, and further optimize it to handle billions of documents efficiently. Empirical results show that the packed training sequences only contain a negligible amount of padding, which enables us to maintain the same

[1]AWS AI Labs. Correspondence to: Hantian Ding <dhantian@amazon.com>, Zijian Wang <zijwan@amazon.com>.

[1]Bin packing is an optimization problem in which items of different sizes must be packed into a finite number of bins or containers, each of a fixed given capacity, in a way that minimizes the number of bins used (Bernhard & Vygen, 2008).

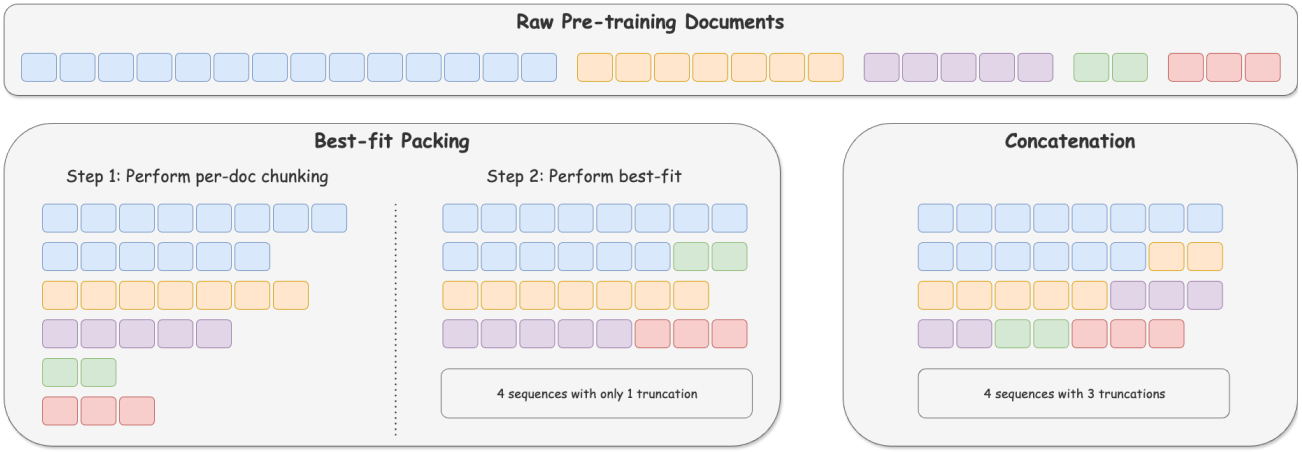


Figure 1. An illustration of the proposed Best-fit Packing compared with concatenation (baseline). We set max sequence length to 8 tokens in this example. **Top:** Original training documents. Each box stands for a token. Contiguous boxes in the same color represent a document. There are five documents of lengths 14, 7, 5, 2, 3, respectively. **Bottom-left:** Best-fit Packing. In step 1, we segment the long document (e.g., blue) into chunks with ≤ 8 tokens. In step 2, we group chunks into training sequences in a smart way that results in the smallest number of sequences. We do not break any chunk in the second step. In total, only one document was truncated and this is necessary to meet the max sequence length requirement. **Bottom-right:** The concatenation approach. 3 out of the 5 documents are truncated.

training efficiency as the concatenation approach while preventing unnecessary document truncation.

To validate the effectiveness of truncation reduction, we pretrain a set of models with inputs formatted by concatenation and Best-fit Packing respectively, ranging from 7B to 13B in model size, 2k to 8k in sequence length, on both Natural Language (NL) and Programming Language (PL) data. We evaluate these models on 22 tasks covering reading comprehension, natural language inference, context following, summarization, world knowledge, and program synthesis. Experiment results show that models trained with fewer truncations demonstrate superior performance and exhibit less hallucination.

In summary, our main contributions are the following.

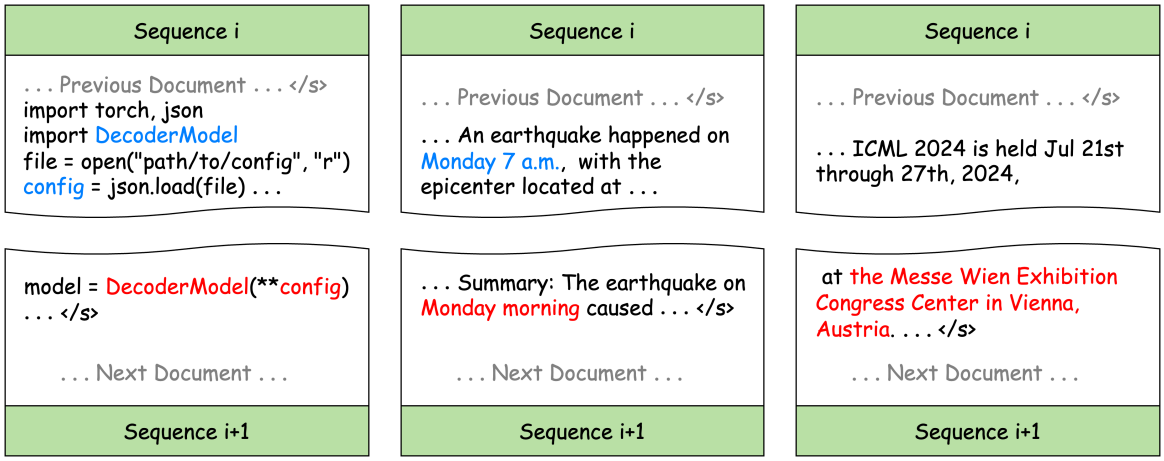
- We highlight the truncation issue inherent in the widely-used concatenation method for LLM pre-training (§2).
- We analytically show the adverse impact of truncation on learning through a simplified model (§2.1).
- We propose Best-fit Packing, a scalable data grouping method that eliminates unnecessary document truncations at almost no cost of training efficiency (§3).
- We empirically quantify the benefits of truncation reduction in a variety of downstream scenarios (§4).

2. The Curse of Truncation

A well-written document in its entirety is naturally coherent and self-contained. In particular, factual statements in the document often logically depend on their aforementioned context through reference, entailment, or more sophisticated reasoning. We refer to the key span(s) of context that serves to establish such a dependency relation as *grounding context*. When learning from next-token prediction, if the grounding context is missing, the model will be forced to spuriously predict token(s) that in fact cannot be derived from the observed partial context. Consequently, at inference time, the model has a higher chance to ignore the grounding context (even when it is provided) and generate content that either contradicts or cannot be verified from the given context, which is known as *closed-domain hallucination*[2] (OpenAI et al., 2023). We illustrate this point in Figure 2.

Figure 2(a) shows an example in Python. Despite the original code being correct, splitting variable definitions and corresponding usages into two distinct training sequences introduces grammatical errors. As self-attention does not cross sequence boundaries, `DecoderModel` and `config` are essentially undefined in the latter training sequence. Formatting data in such a fragmented way makes models learn pathological patterns, potentially leading to hallucination in downstream tasks. For example, in a program synthesis task, the model may directly use `config` without its definition. Even worse, the model may disregard the provided context and fabricate an irrelevant name: if we intend to instantiate an `EncoderModel`, and specify `import EncoderModel` in context, the model may still generate `model=DecoderModel(...)` due to the learned spurious association between `model` and `DecoderModel`.

[2] *Hallucination* is an overloaded term. In this work, we focus on context-based hallucination as opposed to knowledge-based.



(a) Undefined Names (b) Ungrounded content (c) Missing knowledge

Figure 2. Examples where document truncation leads to hallucination or loss of knowledge. (a) Variable definitions (in blue) are truncated and subsequent usage calls result in undefined names (in red). (b) Key context information is truncated (in blue), making the summary unfaithful (in red). (c) Where ICML 2024 is held is unknown to the model due to truncation.

Figure 2(b) illustrates the same issue in natural language where truncation harms faithfulness. The phrase *Monday morning* in the summary cannot be grounded to any part of the context in the same training sequence, and thus turns into a fabrication. Such incomplete samples can reduce models' *context-awareness* (i.e., the ability to attend to context) and result in unfaithful generation or nonsensical reasoning.

Besides exacerbating hallucination, truncation can also impede knowledge acquisition during training, as textual representation of knowledge often takes the form of complete sentences or paragraphs, which is vulnerable to fragmentation. For example, in Figure 2(c), the model will not be able to learn the location of ICML because the conference name and its venue are located in different training sequences.

2.1. Analytical Study via a Simplified Stochastic Process

As an additional source of intuition, we describe a simplified stochastic process $(X_n)_{n\in\mathbb{N}}$ for which we can analytically show that a model trained on truncated sequences achieves a *strictly worse* sequence modeling accuracy than a model trained on full sequences, even if the amount of training data is infinite. While it is difficult to rigorously establish a theory on how truncation impacts learning with transformers models, we would like to make a first attempt on the analytical exploration to better motivate our proposal.

In analogy with language modeling, we can think of the X_n's as tokens in the binary vocabulary $\{0, 1\}$. Our process is defined recursively, starting from a Bernoulli variable X_0 which takes the value 0 with probability 0.5 and the value 1 otherwise. For $n \geq 1$, the variable X_n takes the value of X_0 with probability p and $1 - X_0$ with probability $1 - p$, where $p \in (0.5, 1)$ is fixed. A graphical model associated with this process would be a tree with X_0 as the root and $X_1, X_2, \ldots$ as the leaves.

We now compare a "model A" trained on sequences $X_{0:L} := (X_0, X_1, \ldots, X_{L-1})$, against a "model B" trained on sequences (X_0) and $X_{1:L} := (X_1, X_2, \ldots, X_{L-1})$. Thus, training of model B is affected by truncation. We assume that there is a sufficient amount of data for the models to perfectly fit the training sequences.

For $m \geq 1$, the expected classification loss achieved by model A on token X_m is given by the conditional entropy

$$H(X_m \mid X_{0:m}) = H(X_m \mid X_0) = -p \log p - q \log q,$$

where $q = 1 - p$. On the other hand, if we feed a sequence of observations $(x_0, \ldots, x_{m-1})$ to Model B, its prediction for the next token is equal to

$$P\big(X_{m+1} \mid X_{1:m+1} = (x_0, \ldots, x_{m-1})\big).$$

This distribution can be computed analytically thanks to the simplicity of the process (see Appendix A), allowing us to determine the expected loss of model B. The relative increase in loss from model A to model B is shown in Figure 3 as a function of m. We see that model B is always worse, even under the assumption of sufficient data. The relative loss increase always converges to 0 as m goes to infinity, but the convergence rate hinges on p, i.e., on how strongly the visible tokens depend on the truncated one. This exemplifies an effect that exists also in real-world language modeling: if a key piece of information is truncated but then repeated

shortly after (high p), then the absence of the first mention does not have a lasting impact; on the other hand, if only vaguely related information about the truncated concept is available (low p), then the effect of truncation lasts longer.

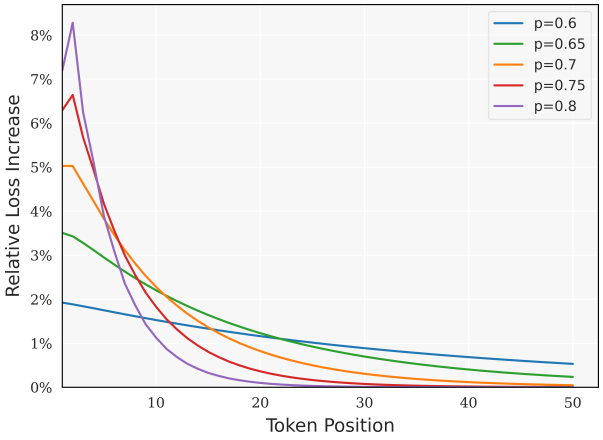


Figure 3. Relative increment in expected loss of model B (trained on truncated sequences) with respect to model A (trained on full sequences), as a function of token position $m \geq 1$, for different p.

3. Best-fit Packing

We propose a new method to *group training data efficiently* while *eliminating unnecessary truncation*, as illustrated in Figure 1. Given the model's max sequence length L, we first segment every document into chunks that are at most L tokens long. Note that this is the minimally required truncation, constrained by the context length. Then, to construct each training sequence, we select a number of document chunks to fill up as much of the L-token space as possible, without breaking any of them. The selection strategy, which we refer to as the packing algorithm, is discussed in §3.1. There are two challenges: first, as it's not always feasible to fill up the L-token sequence fully, padding tokens are used, leading to an increased number of training sequences compared to the concatenation approach. This increase necessitates more training steps per epoch. Therefore, packed sequences must be compact enough to minimize the use of padding tokens in order to retain training efficiency. Second, the algorithm must be scalable and fast enough so that it can operate on datasets of billions of documents.

3.1. The packing algorithm

We formulate Best-fit Packing as a combinatorial optimization problem. We then present an efficient algorithm that scales linearly with data size, and show the solution achieves the same level of compactness as the usual concatenation, thus incurring a negligible loss in training efficiency. Empirically, we validate our method on large-scale pre-training datasets, specifically the RefinedWeb (Penedo et al., 2023) for text, and the Stack (Kocetkov et al., 2022) for code.

Algorithm 1 First/Best-Fit-Decreasing

Input: items $C = \{c_i\}_{i=1}^N$, bin size L
Define $l(c)$: the weight of item c
Define $r(b)$: the remaining capacity of bin b
$(b_1, b_2, \ldots, b_N) \leftarrow$ Initialize empty bins
$SC \leftarrow$ Sort C by weight in descending order
for c_i **in** SC **do**
  Let $J = \{j | r(b_j) \geq l(c_i)\}$
  - FFD: Find $j^* = \texttt{min}(J)$
  - BFD: Find $j^* = \texttt{argmin}_{j \in J} r(b_j)$

  Add c_i to b_{j^*}
end for

Given a set of document chunks $C = \{c_1, \ldots, c_N\}$, where $l(c)$ is the length of c in tokens and $l(c_i) \leq L$, packing these chunks into training sequences is equivalent to determining a partition of C, denoted as $S = \{s_1, \ldots, s_M\}$, subject to $\sum_{c \in s_i} l(c) \leq L$. A training sequence is constructed by concatenating all chunks in an s_i. Our goal is to find a partition S of the smallest possible size, which in practical terms means generating the fewest number of training sequences.

The above optimization problem is known as *the bin packing problem* (Bernhard & Vygen, 2008), in which N items of different sizes must be packed into a finite number of bins or containers, each of a fixed given capacity, in a way that minimizes the number of bins used. Computationally, the problem is NP-hard. There exist several approximation algorithms, among which *First-Fit-Decreasing* (FFD) and *Best-Fit-Decreasing* (BFD) are the most popular ones that strike a good balance between efficiency and accuracy. We briefly describe these heuristics in Algorithm 1.

Time Complexity In general, both FFD and BFD take $O(N \log N)$ sorting time and $O(N \log N)$ packing time. The search step for j^* is typically implemented with an $O(N)$-sized balanced binary tree that tracks all existing bins. However, in our case, notice that the sequence length is always an integer in $[1, L]$, where $L \ll N$. This reduces the sorting cost to $O(N)$ via count sort, and more importantly, allows further optimization on the packing part. Since in BFD we do not distinguish among bins with the same remaining capacity, it suffices to track the remaining capacity values instead of the actual bins, which effectively reduces the tree size to $O(L)$, and consequently the packing time to $O(N \log L)$. However, the same does not apply to FFD, because the order of bins matters.

In practice, we implemented the above fast search in BFD using a segment tree defined as follows:

- The tree has L leaf nodes. The value of the i-th leaf is i if there exists at least one bin whose remaining capacity is i, and zero otherwise. Initially, all leaf nodes are set to zero, except the last one which is set to L.

- The value of every internal node is the maximum value of its children.

To find the best-fit bin, we query the tree from the root. At every internal node, we go left if the left child is no less than the item weight, and go right otherwise. We end up at a leaf node whose value is the best-fit capacity. A capacity-to-bin map is used to retrieve the best-fit bin. Finally, we update the tree to restore the two properties listed above. Please refer to Appendix B for a more detailed illustration.

Table 1 presents a runtime comparison of the Optimized Best-Fit Decreasing (OBFD) algorithm against the standard First-Fit Decreasing (FFD) at 2048 context length on different data scales by up/down-sampling the RefinedWeb dataset which consists of roughly 1 billion documents. As demonstrated, our optimized BFD saves 60% of the running time at 1B scale, and the relative speedup (FFD/OBFD) increases logarithmically to the data size. With this efficient implementation, our method is able to scale up to even larger datasets as the asymptotic time complexity only depends *linearly* on the data size.

Table 1. We benchmark the running time in seconds of FFD and our optimized BFD (OBFD) on 1 million to 2 billion documents at 2048 context length. Both algorithms are implemented in Python and run on a single CPU thread. The baseline FFD uses a similar segment tree as the optimized BFD, except having N leaves that correspond to all bins. Our optimized BFD significantly improves the packing efficiency with linear scalability.

	Document Count				
	1M	10M	100M	1B	2B
FFD (sec)	17	205	2,311	26,354	55,074
OBFD (sec)	10	106	1,066	10,816	22,244
FFD / OBFD	1.7	1.93	2.17	2.44	2.48

Compactness The other important perspective of packing is how compact the resulting training sequences are. Theoretically, both FFD and BFD are guaranteed to use no more than $11/9$ of the optimal number of bins asymptotically (Johnson et al., 1974). However, practically, the majority of documents are short compared to context length, as shown by the dashed curves in Figure 4. The abundance of small-sized items makes packing much easier. Consequently, we observe that the training sequences from packing are nearly as compact as those obtained from concatenation. As shown in Table 2, Best-fit Packing yields no more than 0.01% additional training sequences with either 2k or 8k context length and across NL and PL datasets. From another perspective, Table 2 also indicates that with Best-fit Packing, the amount of padding is negligibly small. The results prove that Best-fit Packing achieves roughly the same training efficiency as concatenation, as measured by the number of non-padding tokens processed using the same amount of compute.

Table 2. Compactness of Best-fit Packing. We present the number of training sequences generated by concatenation, and the increment in number (or percentage) of sequences generated by Best-fit Packing with respect to concatenation. The difference between the two approaches is negligible. Therefore, Best-fit Packing retains the same training efficiency as concatenation.

	RefinedWeb (NL)		Stack (PL)
Max length	2048	8192	2048
$Concat$	2.6×10^8	6.5×10^7	6.4×10^7
$\Delta(Pack, Concat)$	+6253	+411	+1786
$\Delta\%(Pack, Concat)$	0.0024%	0.00063%	0.0028%

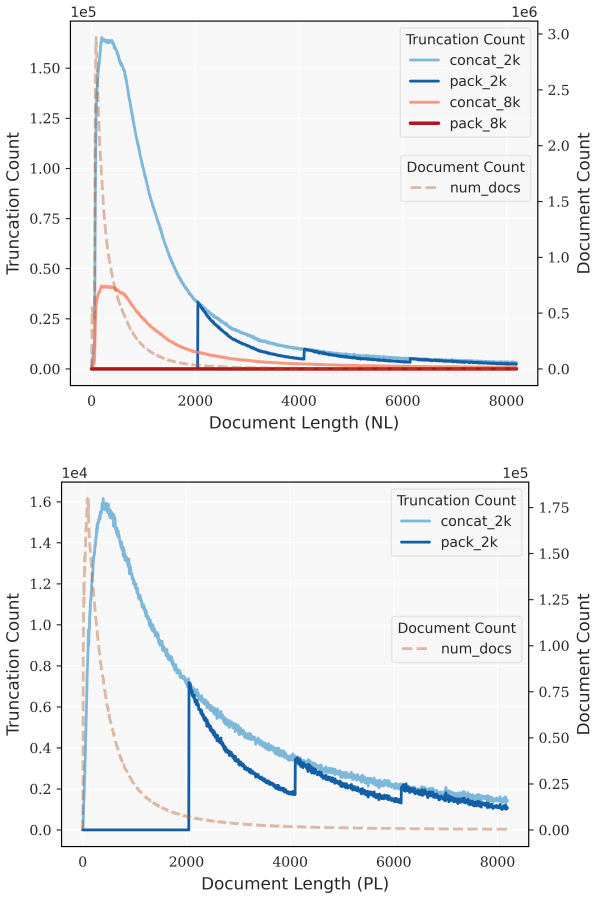


Figure 4. Document count and truncation count for each document length, under 2k or 8k max sequence length. The number of truncations reduces significantly with Best-fit Packing. Top: Natural Language. Bottom: Programming Language.

Truncation reduction We study to what extent Best-fit Packing alleviates the truncation problem. We count how many times each document is truncated in both NL and PL datasets, and aggregate by document length, as plotted in Figure 4. Note that most documents contain fewer than $L = 2048$ tokens; therefore, truncation caused by concate-

nation predominantly occurs within this range. By completely eliminating truncation for document lengths below L, Best-fit Packing effectively preserves the integrity of an overwhelming majority of documents, limiting truncation only to longer documents where it is absolutely necessary.

4. Experiments and Results

To empirically validate the effectiveness of Best-fit Packing over concatenation, we pre-train a set of transformer language models using the same architecture as LLaMA (Touvron et al., 2023a), covering different domains, sizes, and context lengths as in Table 3. We use two popular pre-training datasets in our study: the Falcon RefinedWeb dataset (Penedo et al., 2023) for text, and the Stack (Kocetkov et al., 2022) for code. Please refer to Appendix C.1 for additional details on the training setup.

Table 3. Model specs. We list the number of model parameters, context length, and the number of tokens trained.

Domain	#Params	Context Len.	#Tokens
Natural Language	13B	2048	500B
Natural Language	13B	8192	500B
Programming Language	7B	2048	300B

We evaluate models on a variety of downstream tasks with zero-shot and few-shot prompting. Our findings reveal that Best-fit Packing improves performance in an array of tasks, most significantly in reading comprehension (+4.7%), natural language inference (+9.3%), context following (+16.8%) and program synthesis (+15.0%).[3] We also show that Best-fit Packing effectively reduces closed-domain hallucination.

Throughout this section, we denote results that are statistically significant ($p < 0.05$) under a paired t-test by a superscript **s**, and those that are not significant by **n**. On every single task evaluated, our method either outperforms or matches the baseline. Importantly, in no case do we observe a statistically significant degradation with Best-fit Packing, showing that the improvement is *monotonic*.

4.1. Reading Comprehension

Reading comprehension tests models' ability to answer questions based on information from a given passage. We evaluate the 13B natural language models with 5-shot in-context learning on Narrative QA (Kočiský et al., 2018), Natural Questions (Kwiatkowski et al., 2019), SQuAD (Rajpurkar et al., 2018), and DROP (Dua et al., 2019); with 1-shot on QuAC (Choi et al., 2018); and with zero-shot on BoolQ (Clark et al., 2019) and RACE (Lai et al., 2017). We report F1 score or exact match (EM) for generation tasks, and accuracy for multiple-choice tasks. We adopt the few-shot examples released by HELM (Liang et al., 2022) for Narrative QA, QuAC, and Natural Questions, and randomly sample from the training split for the rest datasets.

Results in Table 4 demonstrate the superior performance of Best-fit Packing in reading comprehension at both 2k and 8k context length: packing significantly outperforms concatenation in half of the settings, and shows no degradation on the rest. Across different benchmarks, we observe: 1) between open-ended generation and multiple choice, Best-fit Packing generally achieves more significant gains on the former, presumably because hallucination is less of a problem in multiple choice questions where the answer space is highly constrained. 2) Among generation tasks, the improvement on Natural Questions is minimal. We speculate that this is because these questions are designed to be answerable under both open-book and closed-book settings, which allows models to occasionally bypass the context and still provide correct answers based on their inherent parametric knowledge. 3) On all other generation tasks where the answer must be inferred from context, the improved performance aligns with our hypothesis that Best-fit Packing enhances models' context-awareness.

4.2. Natural Language Inference

We evaluate models' capability in understanding dependencies between sentences through natural language inference (NLI) tasks. We use 5-shot in-context learning for MultiNLI (Williams et al., 2018) and RTE (Wang et al., 2019). As shown in Table 5, Best-fit Packing improves NLI performance by up to +9.3%. With truncation reduction, dependency relations between sentences in training documents are better preserved, which explains the observed improvement.

4.3. Context Following

To validate our hypothesis that excessive truncations impair factual consistency and faithfulness of generation with respect to the context, we consider special cases where the context contradicts the model's parametric knowledge and the model must follow instructions or facts in the context to answer correctly. Specifically, we evaluate with 5-shot on NQ-Swap (Longpre et al., 2021), a perturbed version of Natural Questions by replacing both the answer and answer mentions in the context with a different entity, and with zero-shot on MemoTrap (McKenzie et al., 2023), where the instruction conflicts with models' memorization. Table 5 shows our method excels concatenation in both settings by up to +16.8%, thereby further strengthening our hypothesis. This also suggests that Best-fit Packing can potentially enhance in-context learning (Wei et al., 2023), presenting a promising avenue for future exploration.

[3] As the scale of metric varies task by task, we use relative improvement in narratives by default unless otherwise noted.

Table 4. Evaluation on **Reading Comprehension** tasks. Best-fit Packing outperforms the concatenation baseline on almost all datasets.

Seq Len	Method	Narrative QA	QuAC	Natural Q	SQuAD	DROP	BoolQ	RACE-m	RACE-h	**Average**
		F1	F1	EM	EM	EM	Acc	Acc	Acc	
2k	Concat	60.38%	33.86%	39.13%	53.86%	21.47%	70.64%	50.00%	44.31%	46.71%
	Pack	**63.91%**^{s}	**35.09%**^{n}	**39.87%**^{n}	**61.61%**^{s}	**24.44%**^{s}	**70.73%**^{n}	**50.55%**^{n}	**45.17%**^{n}	**48.92%**
8k	Concat	61.03%	33.68%	38.40%	59.19%	23.52%	69.33%	**49.72%**^{n}	42.20%	47.13%
	Pack	**64.78%**^{s}	**35.79%**^{s}	**39.93%**^{n}	**61.57%**^{s}	**25.35%**^{s}	**71.31%**^{s}	48.34%	**45.17%**^{s}	**49.03%**

Table 5. Best-fit Packing excels over baseline in **Natural Language Inference** and **Context Following**.

Len	Method	**NLI**		**Context Following**	
		MNLI	RTE	NQ-Swap	MemoTrap
		ACC	ACC	EM	ACC
2k	Concat	42.78%	55.74%	45.62%	35.58%
	Pack	**44.33%**^{s}	**60.28%**^{s}	**51.03%**^{s}	**41.56%**^{s}
8k	Concat	38.85%	54.66%	47.07%	37.61%
	Pack	**40.60%**^{s}	**59.72%**^{s}	**50.04%**^{s}	**40.28%**^{s}

4.4. Summarization

We conduct 5-shot evaluation on CNN/DailyMail (See et al., 2017; Hermann et al., 2015) and XSUM (Narayan et al., 2018), using few-shot examples released by HELM. We follow HELM to report ROUGE scores (Lin, 2004) for accuracy on both datasets, along with SummaC (zero-shot) (Laban et al., 2022) and QAFactEval (Fabbri et al., 2022) scores for faithfulness on CNN/DailyMail. However, these two metrics are suboptimal for XSUM (see Appendix C.2). Thus, we report the binary factuality scores from FAVA (Mishra et al., 2024) for faithfulness evaluation on XSUM.[4]

In Table 6, we observe improvement in all cases except on XSUM with 2k context length, where both methods perform close to each other. Models trained with Best-fit Packing generally obtains not only higher ROUGE scores, but also better faithfulness. The result further strengthens our hypothesis that excessive truncation in training data is one of the reasons for hallucination. By keeping documents in their entirety to the maximum possible extent, Best-fit Packing effectively improves the faithfulness of generations.

Interestingly, we also find that the choice between packing and concatenation makes a difference in the number of generated sentences on CNN/DM. The prompts we used explicitly ask models to summarize the article in 3 sentences. Table 6 shows that models trained with packing do a better job at following this instruction, while models trained with concatenation tend to compose longer summaries. Best-fit Packing seems to ameliorate an issue where the conventional approach causes models to ramble on, despite being prompted with a verbalized length constraint.

[4]A summary gets a score of 1 if FAVA does not detect any hallucination (with the article as reference evidence), and 0 otherwise.

4.5. Commonsense and Closed-book QA

We evaluate models' commonsense and world knowledge on benchmarks where answers cannot be inferred solely from context and models must rely on their parametric knowledge. We use 5-shot in-context learning for SIQA (Sap et al., 2019), ARC (Sakaguchi et al., 2020), TriviaQA (Joshi et al., 2017), and zero-shot for HellaSwag (Zellers et al., 2019) and PIQA (Bisk et al., 2020). We report exact match (EM) for TriviaQA, and multiple-choice accuracy for the rest tasks.

Results are presented in Table 7. Best-fit Packing is slightly better than concatenation on average, and individually the performance can be very close on some of the datasets. Although in §2 we discussed that truncation may also impede knowledge acquisition during training, the impact is not uniform. Most of the standard evaluation sets focus on *common knowledge*, and for a piece of knowledge to be considered as *common*, it must be associated with abundant occurrences in human language. Therefore, even if the knowledge is lost due to one document getting truncated, the model still has a good chance to learn it from other complete documents carrying the same information. In contrast, *tail knowledge* that appears less frequently is more vulnerable to truncation.

This conjecture is exemplified by our observation that truncation reduction results in a larger gain on ARC-C, which covers likely more tail knowledge, over ARC-E, which covers likely more common knowledge. To verify this difference, we follow Kandpal et al. (2023) to count the co-occurrences of each question-answer pair from ARC-E and ARC-C respectively, using the pre-computed Wikipedia entity map. We report the distribution of QA co-occurrence counts in Table 8, and show that indeed the challenge set contains a larger portion of rarely co-occurring pairs. This supports our earlier hypothesis and suggests a possible reason why LLMs struggle to learn long-tail knowledge (Kandpal et al., 2023).

4.6. Program Synthesis

We evaluate the 7B programming language models on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) for zero-shot code generation. Following standard practice, we generate 200 samples per problem, and report Pass@k with $k = 1, 10, 100$ for functional correctness.

Table 6. Evaluation on **Summarization** tasks. We report ROUGE for accuracy, SummaC, QAFactEval, FAVA for faithfulness, and number of generated sentences. Best-fit Packing generally achieves better performance except on XSUM at 2k length. ↑ indicates higher is better.

Seq Len	Method	CNN/DailyMail				XSUM		
		ROUGE-1/2/L	SummaC(↑)	QAFactEval(↑)	#Sent.	ROUGE-1/2/L	FAVA(↑)	#Sent.
2k	Concat	29.46 / 11.04 / 19.47	32.77%	3.348	4.89	**35.42 / 13.27 / 28.23**^{n}	**86.20%**^{n}	1.02
	Pack	**33.79 / 13.14 / 22.88**^{s}	**39.55%**^{s}	**3.772**^{s}	**3.13**	35.40 / 13.14 / 27.79	85.57%	1.02
8k	Concat	33.92 / 13.41 / 23.18	48.89%	4.137	3.72	33.99 / 12.65 / 26.99	84.50%	1.02
	Pack	**35.71 / 14.43 / 23.84**^{s}	**49.65%**^{n}	**4.224**^{s}	**3.23**	**35.29 / 13.35 / 27.69**^{s}	**85.80%**^{n}	1.03

Table 7. Evaluation on **Commonsense and Closed-book QA** tasks. Best-fit Packing is slightly better on average, with notable gains on ARC-C and TriviaQA.

Seq Len	Method	HellaSwag	PIQA	SIQA	ARC-E	ARC-C	TriviaQA	**Average**
		Acc	Acc	Acc	Acc	Acc	EM	
2k	Concat	**75.43%**^{n}	**80.25%**^{n}	53.53%	**76.14%**^{n}	43.94%	57.01%	64.38%
	Pack	75.31%	79.98%	**53.99%**^{n}	75.88%	**46.67%**^{s}	**57.20%**^{n}	**64.84%**
8k	Concat	74.30%	79.82%	53.74%	**74.75%**^{n}	43.00%	55.21%	63.47%
	Pack	**75.16%**^{n}	**80.63%**^{s}	**53.89%**^{n}	74.33%	**44.20%**^{n}	**56.65%**^{s}	**64.14%**

Table 8. Distribution of QA co-occurrence counts in ARC.

ARC	Co-occurrence Count			
	[1, 10)	[10, 100)	[100, 1k)	[1k, $+\infty$)
Easy	9%	26%	50%	14%
Challenge	15%	32%	42%	11%

Besides, a crucial and great aspect of evaluation with programming language is that we can detect hallucination *accurately* without relying on ML-based models which can be error-prone. We resort to program analysis as a more reliable alternative for hallucination detection. Following (Ding et al., 2023), we identify *undefined name* errors using static analysis, and report the percentage of generations with at least one such error as the hallucination metric.

As shown in Table 9, our method both improves Pass@k (+15.0% for Pass@100 on HumanEval and +5.8% on MBPP), and reduces undefined name errors significantly by up to 58.3%. With Best-fit Packing eliminating most truncations in training inputs (cf. Table 4), models are exposed to fewer partial code segments that contain undefined names. Consequently, hallucination is suppressed in model-generated code. Besides that, Best-fit Packing also benefits functional correctness as reflected in Pass@k improvement, which we believe is a combined effect of reduced hallucination and a better understanding of programming logic by learning from complete code examples.

5. Related Work

Pre-training Data Pre-training data is pivotal to the quality of language models. There has been multiple high-quality pre-training datasets that were made publicly available, e.g., C4 (Raffel et al., 2020b), Pile (Gao et al., 2021), RefinedWeb (Penedo et al., 2023), RedPajama (Computer, 2023), and the Stack (Kocetkov et al., 2022; Lozhkov et al., 2024). On top of these, multiple papers (e.g., (Lee et al., 2022; Marion et al., 2023; Chen et al., 2023; Chowdhery et al., 2023; Touvron et al., 2023a; Raffel et al., 2020a)) propose various filtering strategies to improve data quality. Our work broadly applies on top of these pre-training datasets.

Data grouping in language model training Recent transformer language models have adopted different strategies to group training data into batched sequences in order to tackle the variable document length problem. For encoder-only models, the choice of data formatting was first studied in RoBERTa (Liu et al., 2019), which shows that concatenating sentences from more than one documents in the same training sequence results in very little performance degradation. Krell et al. (2021) proposed an approximation-based combinatorial packing method to accelerate BERT training (Devlin et al., 2019), yet without improving downstream performance. It is worth mentioning that document truncation is less of a concern for encoder models for two reasons: first, they are usually trained on relatively short text spans of 128-512 tokens that only respect sentence boundary. In such case, document-wise truncation is inevitable given the limited context size. Second, they are not intended for open-ended generation, and thus, hallucination is not an issue.

Decoder-only language models have predominantly adopted the concatenate-then-split strategy to maximize training efficiency (Brown et al., 2020; Chowdhery et al., 2022; Rae et al., 2021; Zhang et al., 2022; Touvron et al., 2023b; Scao et al., 2022). Very recently, Shi et al. (2024) proposed to

Table 9. Evaluation on **Program Synthesis** tasks. Best-fit Packing outperforms concatenation in terms of execution-based accuracy (Pass@k). More importantly, our method can significantly reduce hallucination by up to 58.3% as measured by undefined name errors.

Seq Len	Method	**Accuracy (↑)**						**Hallucination (↓)**	
		HumanEval			MBPP			HumanEval	MBPP
		Pass@1	Pass@10	Pass@100	Pass@1	Pass@10	Pass@100	Undef. Name	Undef. Name
2k	Concat	17.54%	25.91%	35.28%	22.60%	42.49%	59.46%	5.10%	9.52%
	Pack	**18.32%**^{s}	**28.96%**^{s}	**40.57%**^{s}	**23.48%**^{s}	**45.58%**^{s}	**62.93%**^{s}	**2.41%**^{s}	**3.97%**^{s}

concatenate semantically relevant documents into the same training sequence, which yields notable improvement on downstream tasks. Nevertheless, the method as a variant of concatenation still suffers from excessive truncation, and it is orthogonal (and possibly complimentary) to our method.

Integration with LLM Training Framework Best-fit Packing operates on data level and can be handled as an offline process. Thus, it does not require any change in the training implementation and can be integrated into common distributed training frameworks like Megatron-LM (Shoeybi et al., 2019) or DeepSpeed (Rasley et al., 2020) easily.

Hallucination in Language Generation With the rapid development of generative language models of large scale, hallucination has attracted increased attention as it can hinder performance and mislead users with fabricated facts (Ji et al., 2022). Various approaches have been proposed to tackle this problem, including retrieval augmented generation (Peng et al., 2023; Kang et al., 2023), prompt engineering (Si et al., 2023; Ji et al., 2023), context-aware decoding (Shi et al., 2024), and supervised finetuning (Tian et al., 2023). However, hallucination mitigation during the pre-training stage has largely been overlooked, and we are among the first to explore in this direction.

6. Conclusion

The prevalent concatenate-then-split approach of data grouping in language model training inevitably results in fragmentation of documents. We show that this truncation effect undermines models' ability to follow the context, and even worse, makes models more prone to hallucination. Motivated by these, we propose Best-fit Packing, a new data grouping method that maximally preserves the entirety of individual documents. The algorithm is scalable for datasets of billions of documents, and maintains the same level of compactness as concatenation. Empirically, we demonstrate the effectiveness of truncation reduction by comparing models trained with different data grouping strategies at various scales across both text and code. Specifically, we show that by eliminating unnecessary truncations, Best-fit Packing excels in a broad range of tasks without compromising performance on others. Additionally, it effectively reduces closed-domain hallucination in language generation. While the experiments conducted in this paper have primarily focused on the pre-training stage, Best-fit Packing is broadly applicable to the finetuning stage as well. Our work contributes to the ongoing efforts in developing more effective and reliable language models.

Impact Statements

This paper highlights a critical and fundamental issue of excessive truncation that broadly exists in the contemporary LLM training practice. We propose Best-fit Packing, a novel method that mitigates the issue with wide applicability in LLM training. We demonstrate that it improves language modeling while reducing hallucination. However, it is crucial to acknowledge that no method can completely eliminate the risk of generating false information or hallucinating. We encourage readers to refer to Weidinger et al. (2021) for an in-depth discussion on societal impact and potential risks associated with LLMs.

Acknowledgements

We would like to express our sincere gratitude to Carson Klingenberg, Wasi Ahmad, and colleagues at AWS AI Labs for their help and valuable feedback. We are also deeply grateful to the anonymous reviewers whose insightful and constructive comments have significantly enhanced the quality of this work.

References

Austin, J., Odena, A., Nye, M. I., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., and Sutton, C. Program synthesis with large language models. *ArXiv preprint*, abs/2108.07732, 2021. URL https://arxiv.org/abs/2108.07732.

Bernhard, K. and Vygen, J. Combinatorial optimization: Theory and algorithms. *Springer, Third Edition, 2005.*, 2008.

Bisk, Y., Zellers, R., LeBras, R., Gao, J., and Choi, Y. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6239.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Chen, D., Huang, Y., Ma, Z., Chen, H., Pan, X., Ge, C., Gao, D., Xie, Y., Liu, Z., Gao, J., et al. Data-juicer: A one-stop data processing system for large language models. *ArXiv preprint*, abs/2309.02033, 2023. URL https://arxiv.org/abs/2309.02033.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.-t., Choi, Y., Liang, P., and Zettlemoyer, L. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2174–2184, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1241. URL https://aclanthology.org/D18-1241.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311, 2022. URL https://arxiv.org/abs/2204.02311.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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)*, pp. 2924–2936, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

Computer, T. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *ArXiv preprint*,

abs/2307.08691, 2023. URL https://arxiv.org/abs/2307.08691.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Ding, H., Kumar, V., Tian, Y., Wang, Z., Kwiatkowski, R., Li, X., Ramanathan, M. K., Ray, B., Bhatia, P., and Sengupta, S. A static evaluation of code completion by large language models. In Sitaram, S., Beigman Klebanov, B., and Williams, J. D. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 347–360, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-industry.34. URL https://aclanthology.org/2023.acl-industry.34.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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)*, pp. 2368–2378, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

Eilon, S. and Christofides, N. The loading problem. *Management Science*, 17(5):259–268, 1971. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2628979.

Fabbri, A., Wu, C.-S., Liu, W., and Xiong, C. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2587–2601, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.187. URL https://aclanthology.org/2022.naacl-main.187.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *ArXiv preprint*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1693–1701, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/afdec7005cc9f14302cd0474fd0f3c96-Abstract.html.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ArXiv preprint*, abs/2202.03629, 2022. URL https://arxiv.org/abs/2202.03629.

Ji, Z., Yu, T., Xu, Y., Lee, N., Ishii, E., and Fung, P. Towards mitigating LLM hallucination via self reflection. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 1827–1843, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.123. URL https://aclanthology.org/2023.findings-emnlp.123.

Johnson, D. S., Demers, A. J., Ullman, J. D., Garey, M. R., and Graham, R. L. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974. doi: 10.1137/0203025. URL https://doi.org/10.1137/0203025.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147.

Kandpal, N., Deng, H., Roberts, A., Wallace, E., and Raffel, C. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pp. 15696–15707. PMLR, 2023.

Kang, H., Ni, J., and Yao, H. Ever: Mitigating hallucination in large language models through real-time verification and rectification, 2023.

Kocetkov, D., Li, R., Allal, L. B., Li, J., Mou, C., Ferrandis, C. M., Jernite, Y., Mitchell, M., Hughes, S., Wolf, T., et al. The stack: 3 tb of permissively licensed source code.

ArXiv preprint, abs/2211.15533, 2022. URL https://arxiv.org/abs/2211.15533.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023.

Krell, M. M., Kosec, M., Perez, S. P., and Fitzgibbon, A. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *ArXiv preprint*, abs/2107.02027, 2021. URL https://arxiv.org/abs/2107.02027.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Laban, P., Schnabel, T., Bennett, P. N., and Hearst, M. A. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177, 2022. doi: 10.1162/tacl_a_00453. URL https://aclanthology.org/2022.tacl-1.10.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL https://aclanthology.org/D17-1082.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.577. URL https://aclanthology.org/2022.acl-long.577.

Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., Zhang, Y., Narayanan, D., Wu, Y., Kumar, A., Newman, B., Yuan, B., Yan, B., Zhang, C., Cosgrove, C., Manning, C. D., Ré, C., Acosta-Navas, D., Hudson, D. A., Zelikman, E., Durmus, E., Ladhak, F., Rong, F., Ren, H., Yao, H., Wang, J., Santhanam, K., Orr, L. J., Zheng, L., Yüksekgönül, M., Suzgun, M., Kim, N., Guha, N., Chatterji, N. S., Khattab, O., Henderson, P., Huang, Q., Chi, R., Xie, S. M., Santurkar, S., Ganguli, S., Hashimoto, T., Icard, T., Zhang, T., Chaudhary, V., Wang, W., Li, X., Mai, Y., Zhang, Y., and Koreeda, Y. Holistic evaluation of language models. *ArXiv preprint*, abs/2211.09110, 2022. URL https://arxiv.org/abs/2211.09110.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *ArXiv preprint*, abs/1907.11692, 2019. URL https://arxiv.org/abs/1907.11692.

Longpre, S., Perisetla, K., Chen, A., Ramesh, N., DuBois, C., and Singh, S. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7052–7063, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.565. URL https://aclanthology.org/2021.emnlp-main.565.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., et al. Starcoder 2 and the stack v2: The next generation. *ArXiv preprint*, abs/2402.19173, 2024. URL https://arxiv.org/abs/2402.19173.

Marion, M., Üstün, A., Pozzobon, L., Wang, A., Fadaee, M., and Hooker, S. When less is more: Investigating data pruning for pretraining llms at scale. *ArXiv preprint*, abs/2309.04564, 2023. URL https://arxiv.org/abs/2309.04564.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173. URL https://aclanthology.org/2020.acl-main.173.

McKenzie, I. R., Lyzhov, A., Pieler, M., Parrish, A., Mueller, A., Prabhu, A., McLean, E., Kirtland, A., Ross, A., Liu, A., et al. Inverse scaling: When bigger isn't better. *ArXiv preprint*, abs/2306.09479, 2023. URL https://arxiv.org/abs/2306.09479.

Mishra, A., Asai, A., Balachandran, V., Wang, Y., Neubig, G., Tsvetkov, Y., and Hajishirzi, H. Fine-grained hallucinations detections. *ArXiv preprint*, abs/2401.06855, 2024. URL https://arxiv.org/abs/2401.06855.

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1797–1807, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206.

Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., and Xiong, C. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/pdf?id=iaYcJKpY2B_.

OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *ArXiv preprint*, abs/2306.01116, 2023. URL https://arxiv.org/abs/2306.01116.

Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., Huang, Q., Liden, L., Yu, Z., Chen, W., and Gao, J. Check your facts and try again: Improving large language models with external knowledge and automated feedback, 2023.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, H. F., Aslanides, J., Henderson, S., Ring, R., Young,

S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S. M., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J., Tsimpoukelli, M., Grigorev, N., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., de Masson d'Autume, C., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., de Las Casas, D., Guy, A., Jones, C., Bradbury, J., Johnson, M. J., Hechtman, B. A., Weidinger, L., Gabriel, I., Isaac, W., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K., Stanway, J., Bennett, L., Hassabis, D., Kavukcuoglu, K., and Irving, G. Scaling language models: Methods, analysis & insights from training gopher. *ArXiv preprint*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

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

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

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, Melbourne, Australia, 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

Rasley, J., Rajbhandari, S., Ruwase, O., and He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In Gupta, R., Liu, Y., Tang, J., and Prakash, B. A. (eds.), *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 3505–3506. ACM, 2020. URL https://dl.acm.org/doi/10.1145/3394486.3406703.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6399.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilic, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., and et al. BLOOM: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100, 2022. URL https://arxiv.org/abs/2211.05100.

See, A., Liu, P. J., and Manning, C. D. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099.

Shi, W., Min, S., Lomeli, M., Zhou, C., Li, M., Lin, X. V., Smith, N. A., Zettlemoyer, L., Yih, W.-t., and Lewis, M. In-context pretraining: Language modeling beyond document boundaries. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Si, C., Gan, Z., Yang, Z., Wang, S., Wang, J., Boyd-Graber, J. L., and Wang, L. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=98p5x51L5af.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *ArXiv preprint*, abs/2104.09864, 2021. URL https://arxiv.org/abs/2104.09864.

Tian, K., Mitchell, E., Yao, H., Manning, C. D., and Finn, C. Fine-tuning language models for factuality, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971, 2023a. URL https://arxiv.org/abs/2302.13971.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023b. URL https://arxiv.org/abs/2307.09288.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Superglue: A stickier benchmark for general-purpose language understanding systems. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3261–3275, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html.

Wei, J., Wei, J., Tay, Y., Tran, D., Webson, A., Lu, Y., Chen, X., Liu, H., Huang, D., Zhou, D., et al. Larger language models do in-context learning differently. *ArXiv preprint*, abs/2303.03846, 2023. URL https://arxiv.org/abs/2303.03846.

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., et al. Ethical and social risks of harm from language models. *ArXiv preprint*, abs/2112.04359, 2021. URL https://arxiv.org/abs/2112.04359.

Williams, A., Nangia, N., and Bowman, S. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M. T., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. OPT: open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068, 2022. URL https://arxiv.org/abs/2205.01068.

A. Derivation of Model Accuracy on the Toy Process

We now describe the computations needed to compute the accuracy of models A and B on the toy process described in §2.1.

Let $x = (x_0, \ldots, x_{m-1})$ be a sequence of observations, for a fixed $m \geq 1$. The next token X_m is distributed as a Bernoulli variable, taking the value x_0 with probability p and $1 - x_0$ with probability $q = 1 - p$. This is exactly the distribution predicted by model A. The expected classification loss of model A on the m-th token is therefore given by the entropy of a Bernoulli distribution, and it does not in fact depend on m or on the value of x_0 (because the entropy remains the same if we swap p and $q = 1 - p$).

Model B, however, "believes" that the observations x are part of a sequence $(\tilde{x}_0, x_0, \ldots, x_{m-1})$ drawn according to the process $(X_n)_{n\in\mathbb{N}}$. We can then think of the inference of model B as consisting of two steps: first, predict the probability distribution of the hidden token $\tilde{x}_0$; then, predict the probability distribution of the next token in the sequence, which is X_{m+1} from the point of view of model B. In formulas:

$$\begin{aligned}
&P\Big(X_{m+1} = 0 \mid X_{1:m+1} = x\Big) \\
&= \sum_{\tilde{x}_0 \in \{0,1\}} P\Big(X_0 = \tilde{x}_0 \mid X_{1:m+1} = x\Big) \cdot P\Big(X_{m+1} = 0 \mid X_0 = \tilde{x}_0\Big) \\
&= P\Big(X_0 = 0 \mid X_{1:m+1} = x\Big) \cdot p \;+\; P\Big(X_0 = 1 \mid X_{1:m+1} = x\Big) \cdot q. \qquad (1)
\end{aligned}$$

The distribution of the hidden token can be computed using Bayes' rule:

$$\begin{aligned}
&P\Big(X_0 = 0 \mid X_{1:m+1} = x\Big) \\
&= \frac{P\Big(X_{1:m+1} = x \mid X_0 = 0\Big) \cdot P\Big(X_0 = 0\Big)}{\sum_{\tilde{x}_0 \in \{0,1\}} P\Big(X_{1:m+1} = x \mid X_0 = \tilde{x}_0\Big) \cdot P\Big(X_0 = \tilde{x}_0\Big)} \\
&= \frac{p^k q^{m-k} \cdot \frac{1}{2}}{p^k q^{m-k} \cdot \frac{1}{2} + p^{m-k} q^k \cdot \frac{1}{2}} = \frac{p^k q^{m-k}}{p^k q^{m-k} + p^{m-k} q^k}
\end{aligned}$$

where k is the number of zeros in the sequence x. We can substitute back in (1) and obtain an explicit expression for the prediction of model B:

$$P\Big(X_{m+1} = 0 \mid X_{1:m+1} = x\Big) = \frac{p^{k+1} q^{m-k} + p^{m-k} q^{k+1}}{p^k q^{m-k} + p^{m-k} q^k}. \qquad (2)$$

The expected classification loss of model B (for the given sequence of observations x) is the cross entropy between the true distribution (a Bernoulli distribution that assigns probability p to x_0 and probability q to $1 - x_0$) and the distribution predicted by model B, which is given by (2):

$$\text{loss}_B(x) = -p^{1-x_0} q^{x_0} \log \frac{p^{k+1} q^{m-k} + p^{m-k} q^{k+1}}{p^k q^{m-k} + p^{m-k} q^k} - p^{x_0} q^{1-x_0} \log \frac{p^k q^{m-k+1} + p^{m-k+1} q^k}{p^k q^{m-k} + p^{m-k} q^k} \qquad (3)$$

where the first term corresponds to the outcome 0 while the second term corresponds to the outcome 1. Note that the right-hand side of (3) only depends on x_0 and k, so we can call it $\text{loss}_B(x_0, k)$. Finally, the expected loss of model B is the average of $\text{loss}_B(x_0, k)$ over all possible observations x:

$$\text{loss}_B = \frac{1}{2} \sum_{x_0 \in \{0,1\}} \sum_{h=0}^{m-1} \binom{m-1}{h} (p^{1-x_0} q^{x_0})^h (p^{x_0} q^{1-x_0})^{m-1-h} \text{loss}_B(x_0, h + 1 - x_0),$$

where h indicates the number of zeros in the subsequence $(x_1, \ldots, x_{m-1})$.

B. An Illustration of optimized BFD Algorithm

We illustrate the proposed optimized Best-Fit-Decreasing algorithm with figures. At a high-level, we maintain three data structures: (i) a *bin-to-items table* which tracks the current assignment of items to every bin; (ii) a *space-to-bins table* that

allow us to retrieve a bin given a certain remaining space; (iii) a *segment tree* that enables us to find the best-fit capacity in $O(\log L)$ time. In this example, we assume the max sequence length (or max bin capacity) is 8.

Initialization: Both the *bin-to-items table* and the *space-to-bins table* are empty at the beginning. All nodes in the *segment tree* are zero except for the right most path along which the nodes have value 8, because we only have empty bins of max capacity at the very beginning.

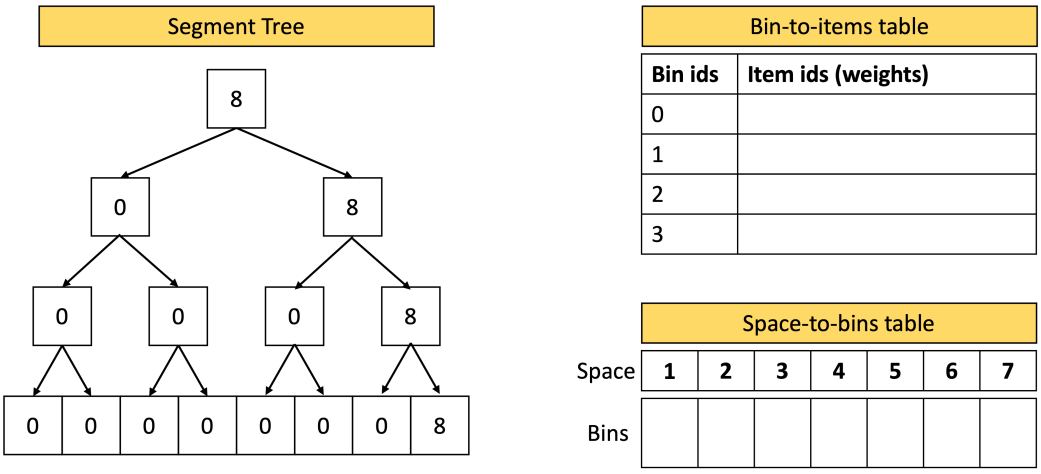


Figure 5. Initialization

A running example: Below we demonstrate how to pack a new item given an intermediate state of the algorithm. Assume that we have packed 4 items whose weights are 8, 6, 6, 4, and arrive at the following state. Now we are about to pack an incoming item of weight 3.

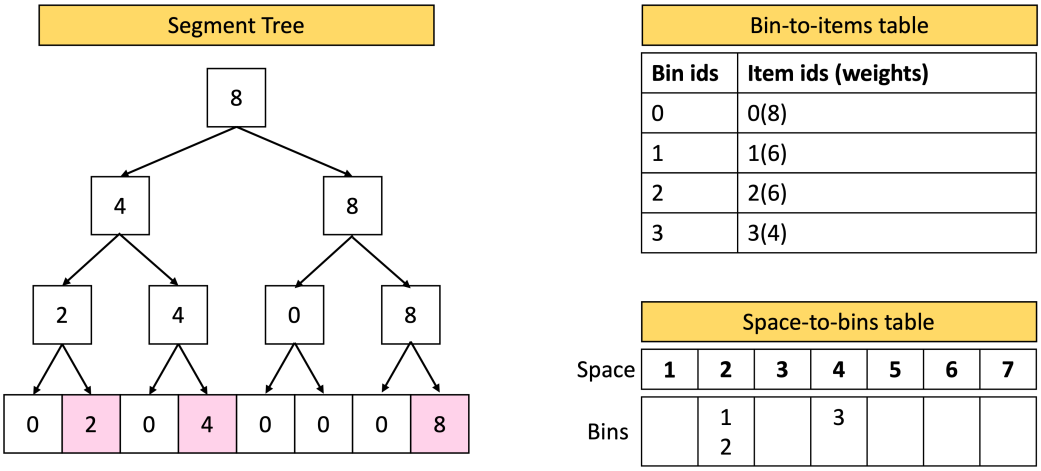


Figure 6. State after packing four items of weight 8, 6, 6, 4. The *bin-to-item table* shows the 0^{th} item (of weight 8) has been placed in bin 0, and the 1^{st} item (of weight 6) in bin 1, and etc.. The *space-to-bins* table shows bin 1 and bin 2 each has 2 space left after packing an item of weight 6, and bin 3 has 4 space left. In the *segment tree*, since we have bins with 2, 4, or 8 space left, the 2^{nd}, 4^{th}, and 8^{th} leaves from left to right are assigned value 2, 4, 8, respectively. All other leaves are zero. The internal nodes are recursively updated to be the maximum of their children.

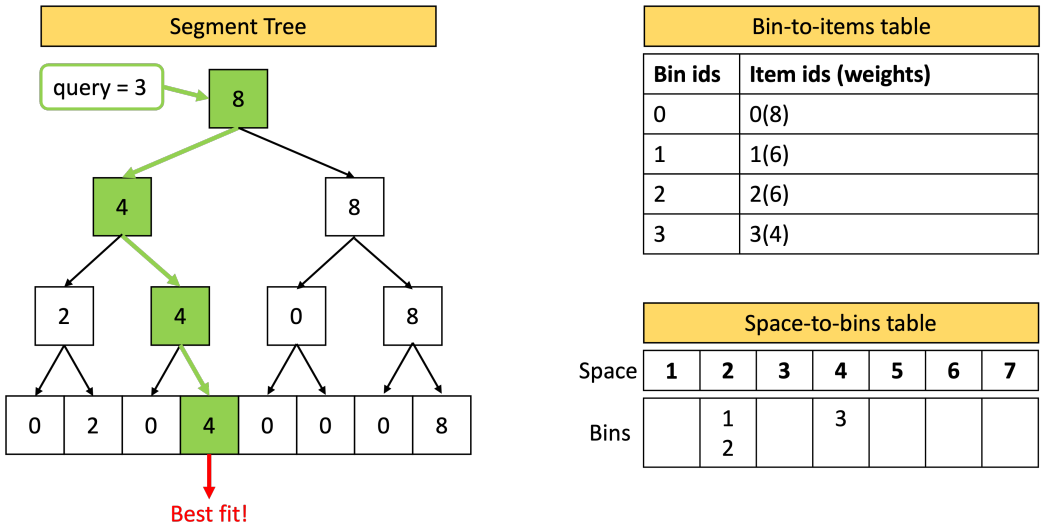


Figure 7. To pack a new item of weight 3, the first step is to find the best-fit capacity using the segment tree. We query the tree from the root. At every internal node, we go left if the left child is no less than the item weight, and go right otherwise. We end up at a leaf node whose value is the best-fit capacity (4 in this case).

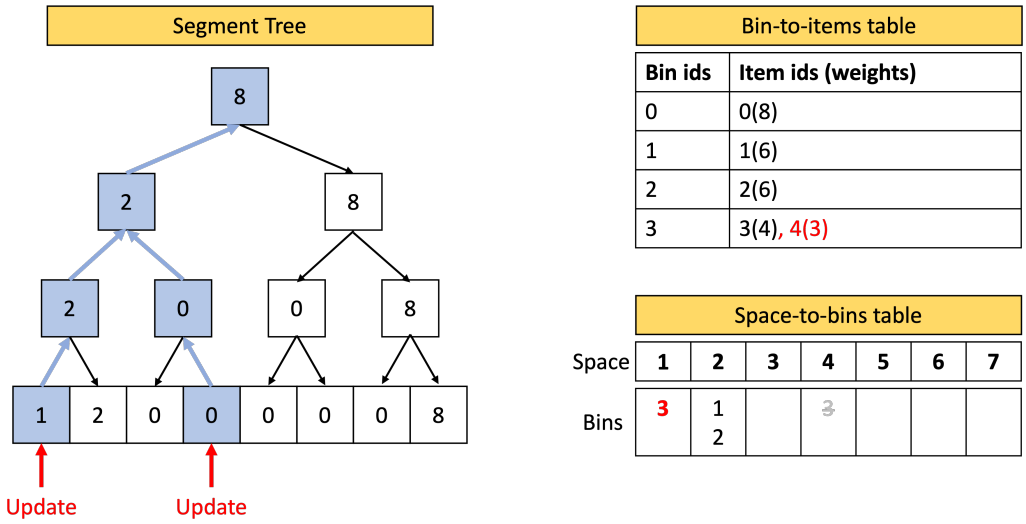


Figure 8. We retrieve a bin with 4 space left, which is bin 3. This new 5$^{\text{th}}$ item (of weight 3) is then placed in bin 3. We update the *bin-to-items table* and *space-to-bins* table accordingly. Finally, we update the *segment tree* recursively from the bottom up to restore the two properties stated in §3: (i) The value of the i-th leaf is i if there exists at least one bin whose remaining capacity is i, and zero otherwise; (ii) The value of every internal node is the maximum value of its children. This concludes one iteration of packing.

C. Additional Details on Experiments

C.1. Training Details

We use the same model architecture as LLaMA 7B/13B (Touvron et al., 2023a). In case a sequence contains multiple documents, we follow the standard practice to allow attention to cross document boundaries when using concatenation (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023b), but mask that out when using Best-fit Packing, so that Best-fit Packing is equivalent to treating each document as a standalone training sample, albeit in a much more compact way. Thanks to the relative nature of rotary positional embeddings (RoPE) (Su et al., 2021), we do not adjust position ids from model input. We perform additional ablations on cross-document attention in §C.3.

We train all models with the AdamW optimizer (Loshchilov & Hutter, 2019). We use a learning rate of 3e-4 with a cosine learning rate scheduler, and warm up over the first 3,000 steps. The global batch size is 2M tokens. We use FlashAttention2 (Dao, 2023) to accelerate training. We find the implementation of document-wise block attention in FlashAttention2 does not yield perceivable overhead compared with the standard full-sequence attention. All models were trained on a cluster of 256 A100 GPUs.

The data pipeline for Best-fit Packing is implemented as follows. In the offline stage, we tokenize the raw data into document chunks up to max sequence length, and run BFD to get chunk indices for each training sequence. During training (the online stage), for every sequence, we fetch the corresponding document chunks through the chunk indices, and build the training sequence on the fly. This saves the expensive cost of concatenating individual chunks on disk.

For the Stack dataset, we use a subset of 7 popular programming languages: Python, Java, C#, JavaScript, TypeScript, C, and C++.

C.2. Faithfulness Metrics for Summarization Tasks

We do not use SummaC (Laban et al., 2022) and QAFactEval (Fabbri et al., 2022) for faithfulness evaluation on XSUM for two reasons. First, SummaC assumes that every sentence in a faithful summary can be entailed from another sentence in the article. This generally does not apply to XSUM which requires summarizing the article in one sentence. If this only sentence is an implication of just one sentence from the article, then the coverage of the summary will be very limited. Second, from what has been reported in literature (Fabbri et al., 2022), both methods show lower accuracy on a summary inconsistency detection benchmark constructed from XSUM (namely XSF (Maynez et al., 2020)), compared to other benchmarks from CNN/DailyMail.

On CNN/DailyMail, we do not report the FAVA binary score because it does not take into consideration the length of the summary. The metric is biased towards shorter generations that have less chance to make a hallucination. As noted in Table 6, there is a significant difference between the number of sentences generated by models trained with concatenation and with packing. On the contrary, on XSUM, all models are able to follow the instruction to generate 1-sentence summaries.

In general, existing methods for evaluating faithfulness of summary have certain limitations, largely due to the complexity of human language. In contrast, hallucination detection for code, which relies on program analysis, tends to be more reliable.

C.3. Ablation Study on Cross-Document Attention

Best-fit Packing treats each segment as individual and thus masks out cross-document attention in training (Appendix C.1). To study the impact of this choice and to highlight the importance of our packing method, we pre-train a 2k NL model with concatenation and cross-document attention mask enabled. In Table 10, we compare the three models by perplexity on a validation set, and performance in downstream tasks. Note that every sequence in the validation set contains only one document chunk. We find that though perplexity gain can be attributed to the attention mask, Best-fit Packing is critical for achieving optimal performance in downstream tasks. In particular, using attention mask on top of the conventional concatenation approach even degrades the model's performance in summarization.

Table 10. Ablation results for cross-document attention. We additionally train a 2k-length model with concatenation and cross-document attention mask. We report perplexity on validation set (**PPL**), and average performance in reading comprehension (**RDC**), natural language inference (**NLI**), context following (**CTX**), summarization (**SUM**, in ROUGE-2), and commonsense (**CMS**).

Method	Attn. Mask	PPL	RDC	NLI	CTX	SUM	CMS
Concat	✗	9.64	46.71%	49.26%	40.60%	12.16	64.38%
Concat	✓	**9.53**	47.92%	50.35%	42.41%	11.79	64.77%
Pack	✓	**9.53**	**48.92%**	**52.31%**	**46.30%**	**13.14**	**64.84%**