NanoKnow: How to Know What Your Language Model Knows

Lingwei Gu*
University of Waterloo
Waterloo, ON, Canada
lingwei.gu@uwaterloo.ca

Nour Jedidi*
University of Waterloo
Waterloo, ON, Canada
njedidi@uwaterloo.ca

Jimmy Lin
University of Waterloo
Waterloo, ON, Canada
jimmylin@uwaterloo.ca

Abstract

How do large language models (LLMs) know what they know? Answering this question has been difficult because pre-training data is often a "black box" – unknown or inaccessible. The recent release of nanochat – a family of small LLMs with fully open pre-training data – addresses this as it provides a transparent view into where a model's parametric knowledge comes from. Towards the goal of understanding how knowledge is encoded by LLMs, we release *NanoKnow*, a benchmark dataset that partitions questions from Natural Questions and SQuAD into splits based on whether their answers are present in nanochat's pre-training corpus. Using these splits, we can now properly disentangle the sources of knowledge that LLMs rely on when producing an output. To demonstrate NanoKnow's utility, we conduct experiments using eight nanochat checkpoints. Our findings show: (1) closed-book accuracy is strongly influenced by answer frequency in the pre-training data, (2) providing external evidence can mitigate this frequency dependence, (3) even with external evidence, models are more accurate when answers were seen during pre-training, demonstrating that parametric and external knowledge are complementary, and (4) non-relevant information is harmful, with accuracy decreasing based on both the position and the number of non-relevant contexts. We release all NanoKnow artifacts at https://github.com/castorini/NanoKnow.

CCS Concepts

• **Information systems** → **Question answering**.

Keywords

RAG, LLMs, Pre-Training Data, Parametric Knowledge

ACM Reference Format:
Lingwei Gu, Nour Jedidi, and Jimmy Lin. 2026. NanoKnow: How to Know What Your Language Model Knows. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3805712.3808604

1 Introduction

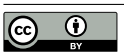
Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks, yet it is unclear *how they know what they know*. While LLMs ultimately express their knowledge through their outputs at inference time, *how* and *where* this knowledge is acquired remains an open question.

Knowledge expressed by LLMs can originate from various, potentially entangled, sources. There exists knowledge stored within their parameters [19], which can be probed via closed-book question answering [8], but this only tells us *what* LLMs know, not necessarily *how* that knowledge was acquired. For example, did this knowledge come from memorization of its pre-training data [3] or is the model performing a sort of multi-hop reasoning over facts encoded within its parameters [27]? Alternatively, external knowledge can be injected into the LLM using retrieval-augmented generation (RAG), but in this case does the model's output solely represent facts present in the external context or does the output represent a latent interaction between the external context and the LLM's parametric knowledge [28]? Ultimately, answering these questions requires understanding the model's pre-training data, but this has been difficult as such data is often unknown or inaccessible [13].

Recently, this changed with developments in fully open LLMs, making the understanding of the pre-training data now possible. A notable example is the release of nanochat [10], which, by being pre-trained on the open FineWeb-Edu corpus – a collection of educational web content [16] – provides a completely transparent and self-contained environment for tracing the information an LLM has *seen*. Such transparency allows us to answer questions like: does seeing facts more often make it easier to recall? When does RAG actually make a difference? However, transparency of data is only the first step. Before we can answer these questions systematically, we require a resource which can not only identify questions an LLM has seen the answer to during pre-training but also questions beyond its knowledge. Such a resource is a necessary step toward properly disentangling and understanding the various sources of knowledge LLMs rely on when producing their outputs.

To address this, we release NanoKnow, a benchmark dataset of questions from Natural Questions (NQ) [11] and SQuAD [18] projected onto the FineWeb-Edu corpus. NanoKnow partitions each dataset into two splits – "supported" (questions for which the answer exists in the pre-training data) and "unsupported" (questions for which the answer does not exist in the pre-training data) – enabling a controlled evaluation of knowledge in LLMs, like nanochat, which were pre-trained entirely on FineWeb-Edu. To generate these relevance judgments, NanoKnow was built in three stages. In the first stage, we build a searchable BM25 index over the corpus using Anserini [26] and retrieve candidate documents for each question. Next, we check for exact match answer strings across the retrieved documents. In the last stage, we use LLM-based verification to filter out coincidental matches, keeping only documents that genuinely answer the questions.

*Equal Contribution



SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808604

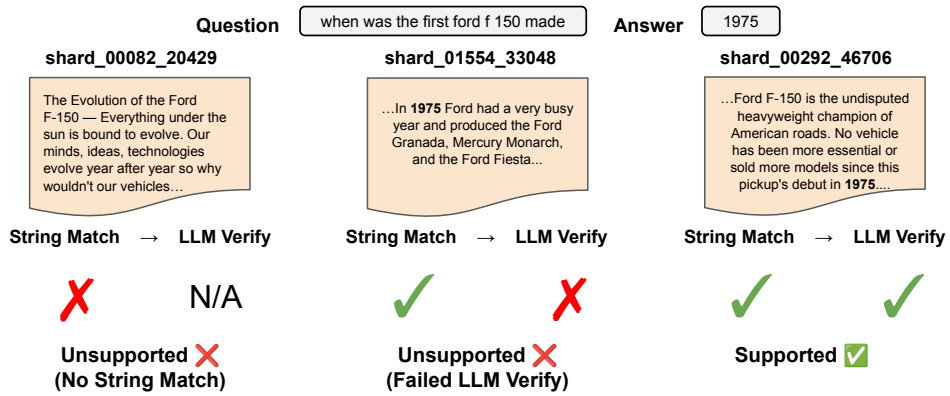


Figure 1: An example of NanoKnow on a question-answer pair. If any passage is deemed to answer the question after the string match and LLM verify steps, we label the question as "supported". Otherwise, the question is considered "unsupported".

With NanoKnow in hand, we run comprehensive experiments using eight nanochat checkpoints across three different model scales. Our various experiments demonstrate the value of NanoKnow as a tool for *confidently* disentangling and evaluating the contributions of different knowledge sources underlying an LLM's outputs. Using NanoKnow, we were able to confirm and replicate a range of results across the literature [2, 3, 6, 9, 14, 22], highlighting its reliability:

(1) Closed-book question answering effectiveness is highly related to answer frequency in the pre-training corpus. We found a clear increase in nanochat's accuracy when it has "seen" the answer more often.
(2) Integrating external evidence mitigates this dependence on "memorization", but even with external evidence, nanochat is more effective on questions with a higher answer frequency in the pre-training corpus.
(3) Even when provided the oracle answer document, nanochat was more accurate on "supported" versus "unsupported" questions, demonstrating that parametric knowledge can complement external knowledge.
(4) Despite nanochat having "seen" the answer to a question, it is negatively impacted by distractor documents (i.e., non-relevant documents). We found a clear decline in accuracy based on *where* the answer document is positioned with respect to distractors as well as *how many* distractors are present.

We hope NanoKnow provides a foundation for future explorations in understanding *how LLMs know what they know.*

2 NanoKnow

NanoKnow is a benchmark dataset that partitions NQ and SQuAD questions into *supported* and *unsupported* splits based on the presence of their answers within FineWeb-Edu [16]. In this section, we present NanoKnow and describe our process for producing it.

2.1 Projection Data

Corpus. We build NanoKnow on the shuffled version of the FineWeb-Edu corpus released by Karpathy [10].[1] This variant of FineWeb-Edu comes as 1,823 parquet shards, each containing thousands of web documents. The total corpus size is about 171GB and contains 97,230,848 documents. FineWeb-Edu was chosen in our experiments primarily due to the recent release of nanochat [10], a family of small LLMs that utilized it for pre-training.

To enable a searchable BM25 index for our projection pipeline – which we describe in the next subsection – we build a FineWeb-Edu inverted index using Anserini [26]. The full index is about 326GB.

Question-Answering Datasets. We project the following question-answering (QA) benchmarks onto FineWeb-Edu:

- **Natural Questions (NQ)** [11]: Open-domain questions from Google search queries. We use the validation set (3,610 questions). Each question has one or more short answers.
- **SQuAD** [18]: Reading comprehension questions where answers are spans from Wikipedia passages. We use the validation set (10,570 questions).

2.2 Projection Pipeline

Given a question and its corresponding *gold* answer, we project it onto FineWeb-Edu using a three-step process, which we describe in this subsection. A high-level illustration of the NanoKnow pipeline for an example QA pair is shown in Figure 1.

Step 1: BM25 Retrieval. Using BM25, we first search the index to retrieve documents that may contain the answer. We retrieve the top-100 candidate documents, leveraging Pyserini [12] for retrieval.

Step 2: Answer String Matching (String Match in Figure 1). Next, we check whether any retrieved document contains the gold answer string associated with the query. The gold answers are taken from the official evaluation set of each benchmark. We lowercase everything and strip extra whitespace, then look for the answer as a substring. If it shows up, we flag the document as a *candidate match.* This is fast, but returns many false positives. For example, for the question "What is the best bakery in Paris?" the word "Paris" might appear in a document about the song Paris and not Paris, France. Another step is needed to filter these out.

[1] https://huggingface.co/datasets/karpathy/fineweb-edu-100b-shuffle

Table 1: Examples of supported and unsupported NQ questions. Failed LLM Verify examples contain the gold answer string in FineWeb-Edu but do not directly answer the question.

Type	Question	Answer	Evidence
Supported	When was the last time anyone was on the moon?	December 1972	"No one has walked on the Moon since **December 1972**."
	What is the main artery that takes blood from the heart to the body?	The aorta	"Arteries begin with the **aorta**, the large artery leaving the heart."
Unsupported (Failed LLM Verify)	Who won last year's NCAA women's basketball?	South Carolina	"The Wildcats were ranked 8th in the nation before a loss to **South Carolina** on October 4."
	Love Yourself by Justin Bieber is about who?	Rihanna	"Other Icelandic artists have followed in his footsteps, collaborating with international acts like **Rihanna** and Ed Sheeran."
Unsupported (No String Match)	Who sang I ran all the way home?	The Impalas	–
	Who plays Gram on The Young and the Restless?	Max Shippee	–

Step 3: LLM Verification (LLM Verify in Figure 1). To address potential false positives from the output of the String Match step, we next leverage an LLM to separate real answer string matches from coincidental matches. In particular, for each candidate answer document provided by String Match (retrieved documents that contain the answer string), we extract a context window around where the answer appears: 256 words before the match and 256 words after. We send this context to Qwen3-8B [25] with the question and ask it to classify whether the context answers the question.

After completing these three steps, NQ and SQuAD can be broken down into two splits:

- **Supported**: Questions where the answer appears in FineWeb-Edu in a relevant context. These are questions which passed both String Match (step 2) and LLM Verify (step 3).
- **Unsupported**: Questions where the answer does not appear in any retrieved document or only shows up in unrelated contexts. These are questions which either have no string match (step 2) or failed LLM Verify (step 3).

With the set of supported questions, we can build the NanoKnow relevance judgments (qrels in TREC parlance), which link the question to its answer document in FineWeb-Edu. Each document gets a unique ID that encodes its location in the corpus. The format is `shard_XXXXX_YYYYY`, where `XXXXX` is the zero-padded shard number and `YYYYY` is the row offset within that shard. For example, `shard_00151_20323` refers to row 20,323 in shard 151. This encoding lets us trace any answer document back to its exact location in the FineWeb-Edu parquet files. For unsupported questions, as they do not contain answers in FineWeb-Edu, we simply provide a file with the question text.

2.3 Projection Results

Now that we have projected NQ and SQuAD onto FineWeb-Edu, we examine the resulting projection. In Table 1, we show examples of questions from NQ which were labeled by the projection pipeline as supported or unsupported.

Table 2 shows the projection results. The "String Match *only*" column shows the percentage of questions which have its answer string in at least one retrieved document. "String Match → LLM Verify" is the percentage of questions which survive after the LLM Verify step. For NQ, 73.9% of questions have the answer string in a retrieved document; after LLM Verify, 66.2% are confirmed to be supported. SQuAD is higher at 70.9% of questions deemed supported, with the LLM Verify step removing 8.0% of questions as coincidental string matches. This is not surprising as SQuAD answers come from Wikipedia, and FineWeb-Edu has a lot of Wikipedia content.

Table 2: Projection rates for NQ and SQuAD on FineWeb-Edu. The reported percentage represents how many questions are supported after String Match only and after String Match followed by LLM Verify.

	NQ	SQuAD
Total QA Pairs	3,610	10,570
String Match *only*	73.9%	78.9%
String Match → LLM Verify (Supported)	66.2%	70.9%

Lastly, to validate that our unsupported labels are accurate, we prompt the official d32 nanochat checkpoint to answer the unsupported questions in a closed-book setting.[2] We found a closed-book exact match (EM) answering accuracy of 0.8% and 1.5% for NQ and SQuAD. More details on the EM accuracy metric can be found in Section 3.

Released Artifacts. As the result of this pipeline, we release the following to support reproducibility and future research:

- **Qrels**: Relevance judgments mapping supported questions from NQ and SQuAD to the answer documents in FineWeb-Edu.
- **Unsupported Questions**: The set of unsupported questions from NQ and SQuAD whose answers are not in FineWeb-Edu.
- **Lucene Index**: Pre-built index over FineWeb-Edu (326GB).[3]
- **Evaluation Code**: Scripts to reproduce all experiments, including LLM-Judge prompts and evaluation metrics.

The artifacts are available at https://github.com/castorini/NanoKnow.

[2] `karpathy/nanochat-d32`
[3] https://huggingface.co/datasets/LingweiGu/NanoKnow-Fineweb-Edu-Index

3 Experimental Setup

NanoKnow now allows us to answer many interesting questions regarding how pre-training data shapes what knowledge LLMs rely on. We study a subset of these questions:

- Does seeing the answer to a question more often in pre-training improve closed-book QA accuracy? What if we integrate external evidence?
- How does closed-book QA compare to open-book QA on supported questions?
- How does an LLM's open-book QA accuracy differ on supported versus unsupported questions?
- On supported questions, how do distractors (i.e., non-relevant information) influence an LLM's QA accuracy?

To answer these questions, we make use of the nanochat [10] family of models, which were entirely pre-trained on the FineWeb-Edu corpus discussed in Section 2. We consider three nanochat model sizes: d20 (≈ 561M parameters); d32 (≈ 1.9B parameters); and d34 (≈ 2.2B parameters). To ensure the robustness of our results to any variations in how models were trained, each evaluation is run with multiple open-source checkpoints for each model scale.

Across experiments, we use the following three prompting setups. The first is "Closed-Book", where nanochat is only prompted with the question. The second is "w/ FineWeb Context", where nanochat is "reminded" with the oracle answer passage from its pre-training data; and, lastly, for SQuAD, we also evaluate nanochat with the original context, "w/ Original Context", where nanochat is provided the original answer context from SQuAD.[4] As FineWeb-Edu documents are very long, for these experiments, we only consider the surrounding context window of 200 words around the first matched answer (approximately 100 words before and 100 words after the answer).

To evaluate the accuracy of responses generated by nanochat, we use two approaches. The first is exact match (EM), computed using the standard evaluation scripts commonly used by the community for these benchmarks. EM checks if any of the predefined correct answers exactly appear in the model's output. If there is a match, the answer is deemed correct; otherwise the answer is deemed incorrect. The other method we consider is an LLM-Judge, which given nanochat's output and the predefined correct answers, classifies nanochat's output as correct or not. For this, we leverage Qwen3-14B [25].

4 Results

4.1 Impact of Answer Frequency in Pre-training

We begin by measuring how closed-book and open-book QA accuracy are impacted by how often the answer was replicated (i.e., "seen") during pre-training. To study this, we measure how accuracy changes with answer frequency in the pre-training corpus.

To measure frequency, for a given question, we count the number of FineWeb-Edu documents in which the answer was found and verified by the LLM in step 3 of Section 2.2. We then categorize questions into four frequency buckets: Rare (1–5 verified documents), Low (6–20), Medium (21–50), and High (51+). Figure 2 shows the distribution of supported questions across these buckets; the majority of questions fall in the Rare and Low frequency buckets for both NQ and SQuAD.

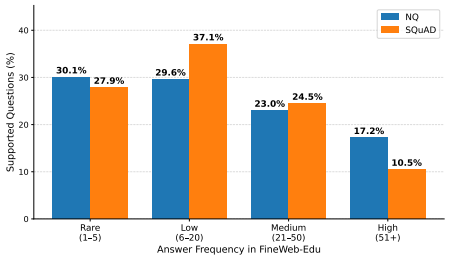


Figure 2: Distribution of NanoKnow's supported questions by answer frequency in FineWeb-Edu for NQ and SQuAD.

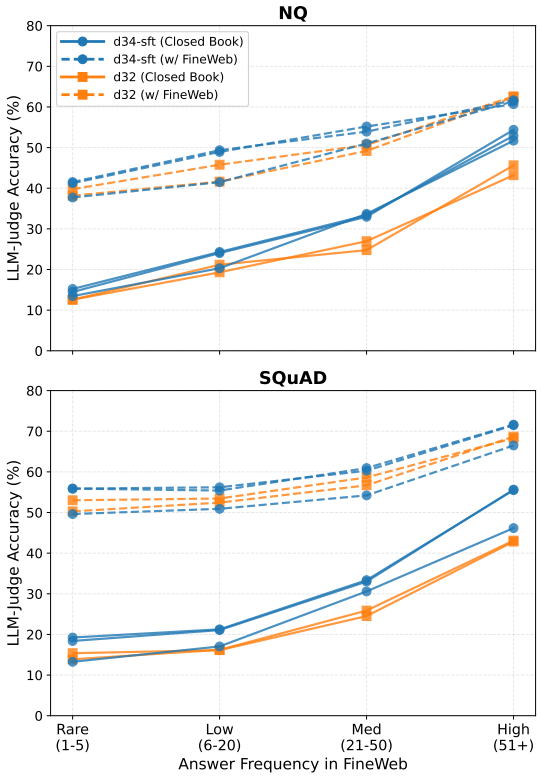


Figure 3: Influence of pre-training data answer frequency on nanochat's accuracy. Solid lines show the closed-book prompt setup; dashed lines show w/ FineWeb-Edu context.

The relationship between answer frequency and nanochat's accuracy is shown in Figure 3. We find a clear increase in closed-book QA effectiveness on both NQ and SQuAD as the answer frequency increases, with accuracy more than doubling for questions with high answer frequency versus rare answer frequency. However, interestingly we did not find this to be the case for d20 (omitted from the plot), suggesting that, at smaller parameter counts, the LLM does not have the capacity to "memorize" information.

[4] As NQ is an open-domain QA task, there does not exist a singular, default "original context", thus for simplicity we do not consider it in our experiments.

Table 3: Comparing closed-book versus open-book QA over the NQ and SQuAD supported splits of NanoKnow.

			NQ				SQuAD					
			Closed-Book		w/ FineWeb Context		Closed-Book		w/ FineWeb Context		w/ Original Context	
	Model Checkpoint	**Model Size**	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge
1	`sampathchanda/nanochat-d20`	561M	0.004	0.008	0.022	0.018	0.004	0.002	0.021	0.019	0.086	0.081
2	`shu127/nanochat-d20`		0.003	0.005	0.021	0.016	0.003	0.011	0.016	0.013	0.042	0.040
3	`pankajmathur/nanochat-d20`		0.003	0.014	0.028	0.023	0.005	0.008	0.022	0.020	0.056	0.052
4	`karpathy/nanochat-d32`	1.9B	0.196	0.224	0.468	0.476	0.114	0.173	0.465	0.540	0.672	0.736
5	`Antigma/nanochat-d32`		0.198	0.226	0.516	0.492	0.122	0.169	0.483	0.551	0.686	0.740
6	`renatocastro33/nanochat-d34-sft`	2.2B	0.250	0.283	0.503	0.528	0.167	0.228	0.512	0.587	0.721	0.779
7	`victoremnm/nanochat-d34-sft`		0.250	0.277	0.503	0.523	0.167	0.227	0.512	0.587	0.721	0.777
8	`pankajmathur/nanochat-d34-finetuned`		0.239	0.271	0.479	0.522	0.141	0.210	0.476	0.569	0.670	0.749

Table 4: QA accuracy for supported versus unsupported questions on SQuAD (w/ Original Context).

		Supported		Unsupported	
Model Checkpoint	**Model Size**	EM	LLM-Judge	EM	LLM-Judge
`sampathchanda/nanochat-d20`	561M	0.086	0.081	0.069	0.068
`shu127/nanochat-d20`		0.042	0.040	0.032	0.031
`pankajmathur/nanochat-d20`		0.056	0.052	0.051	0.054
`karpathy/nanochat-d32`	1.9B	0.672	0.736	0.554	0.688
`Antigma/nanochat-d32`		0.686	0.740	0.553	0.680
`renatocastro33/nanochat-d34-sft`	2.2B	0.721	0.779	0.610	0.737
`victoremnm/nanochat-d34-sft`		0.721	0.777	0.610	0.733
`pankajmathur/nanochat-d34-finetuned`		0.670	0.749	0.574	0.702

When integrating external evidence (i.e., open-book QA), there is also a general increase in accuracy as answer frequency in the pre-training data increases. However, the rate of this improvement, especially for SQuAD, is much lower than in the closed-book setting, demonstrating that RAG can help mitigate this dependence on pre-training frequency.

4.2 Closed-Book QA vs. Open-Book QA

We next examine how much improvement external knowledge provides over the LLM's parametric knowledge. The results of this experiment can be found in Table 3.

As expected, we see a clear upward trend in closed-book accuracy for both NQ and SQuAD as model size increases, demonstrating that larger nanochat checkpoints indeed memorize more of their training data [3, 21]. In particular, for NQ, the closed-book LLM-Judge accuracy improves by 26.9 points (19.2%) when comparing the best nanochat-d20 (row 3) to the best nanochat-d34 checkpoint (row 6). Similarly, with SQuAD, accuracy improves by 21.7 points (19.7%) when comparing the corresponding best checkpoints for nanochat-d20 (row 2) and nanochat-d34 (row 6).

Comparing the "Closed-Book" versus "w/ FineWeb Context" columns, we find that all nanochat checkpoints see a large jump in accuracy when provided an answer passage from its pre-training data as additional context. On average, the relative improvement of incorporating FineWeb context decreases as the nanochat model size increases. For example, on NQ, we see an average LLM-Judge accuracy improvement of 2.4×, 2.1×, and 1.9×, for the d20, d32, and d34 model scales, respectively. With SQuAD, the average LLM-Judge accuracy improvement is 4.4×, 3.2×, and 2.6×, for d20, d32, and d34. This result suggests that smaller models benefit more from open-book QA versus larger models. Furthermore, we find that each of the model checkpoints is more accurate on SQuAD when utilizing the original context versus the FineWeb context. This makes sense since the original context is tailored to answer the question; in other words, the FineWeb context is like a textbook, whereas the original context is the answer booklet.

Finally, we examine how nanochat's open-book QA effectiveness differs on questions in which its answer appears in the pre-training corpus (supported) versus questions where it does not appear in the pre-training corpus (unsupported). For this experiment, we provide nanochat with the original context from SQuAD as the FineWeb context does not exist for the unsupported split. The results are presented in Table 4 and show that nanochat is consistently more accurate on supported questions across all model scales.

4.3 Influence of Distractors

Lastly, we are interested in understanding how nanochat is influenced by distractors (i.e., non-relevant contexts). For this experiment, we follow the setup in Cuconasu et al. [6]. We prompt nanochat using three different placements of the answer contexts and the distractor: "Far" in which the answer context is placed furthest away from the question; "Mid", in which the answer context is placed in the middle of the prompt, in between different distractor contexts; and "Near", in which the answer context is placed closest to the question. We additionally consider a setting in which nanochat is only prompted with a distractor context ("Distractor *only*"). In our setup, the distractor context was the highest ranked BM25 document which *did not* contain the answer; when two distractors are used, i.e., for the "Mid" case, we took the top-2 such documents. To ensure that the distractor contexts have the same length as the answer contexts, we extracted a random 200-word snippet from each distractor document.

We compare all distractor setups to the closed-book and w/ FineWeb context ("Answer *only*") settings shown in Table 3. For this experiment, we focus on `Antigma/nanochat-d32`, the strongest d32 checkpoint. These experiments were run over the supported splits of NQ and SQuAD. The results are shown in Table 5.

Beginning by comparing closed-book (row 1) to the distractor only setting (row 2), we find that prompting nanochat with a non-relevant context does worse than utilizing the model's parametric knowledge, with the LLM-Judge accuracy dropping 3.2 and 1.5 points on NQ and SQuAD, respectively.

Table 5: Influence of distractors on nanochat's (`Antigma/nanochat-d32`) effectiveness on supported questions. A denotes the answer document, D denotes the distractor document, and Q denotes the question.

		Far: [A, D, Q]				Mid: [D, A, D, Q]				Near: [D, A, Q]			
		NQ		SQuAD		NQ		SQuAD		NQ		SQuAD	
	Setting	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge	EM	LLM-Judge
1	Closed-Book	0.198	0.226	0.122	0.169	0.198	0.226	0.122	0.169	0.198	0.226	0.122	0.169
2	Distractor *only*	0.152	0.194	0.091	0.154	0.152	0.194	0.091	0.154	0.152	0.194	0.091	0.154
3	Answer *only*	0.516	0.492	0.483	0.551	0.516	0.492	0.483	0.551	0.516	0.492	0.483	0.551
4	Answer + 1 Distractor	0.452	0.447	0.411	0.478	N/A				0.456	0.457	0.428	0.501
5	Answer + 2 Distractors	0.414	0.422	0.363	0.438	0.387	0.406	0.352	0.428	0.433	0.448	0.397	0.480
6	Answer + 4 Distractors	0.357	0.378	0.287	0.367	0.334	0.368	0.277	0.363	0.417	0.432	0.369	0.457

The negative influence of the distractor context on nanochat's effectiveness is further confirmed when comparing the answer only setting (row 3) to each of the answer + distractor settings (rows 4 to 6). When prompted with answer and distractor documents, as might be expected in a practical RAG setting, nanochat is consistently less accurate than when only prompted with the correct answer context, across all prompt setups (Far, Mid, Near). Furthermore, nanochat is also less accurate when prompted with more distractors. Using the "Far" prompt on NQ as a representative case, the LLM-Judge accuracy drops from 0.447 (1 distractor) to 0.378 (4 distractors).

Lastly, the results show that nanochat is most accurate when the answer context is closest to the question. But notably, being closer to the question is only helpful when there are no distractors between the answer and the question, as nanochat has a "lost in the middle" effect [14], where it is least effective when the answer context is placed between distractors.

5 Related Work

Tracing an LLM's capabilities to its pre-training data. LLMs pick up a wide range of factual knowledge during their pre-training, but identifying *where* in the pre-training data the LLM learned that knowledge remains an open research question. Much of the research in this area fits directly under the umbrella of training data attribution methods [1, 4, 20], which try to find the pre-training data that can explain a model's output. This has been commonly done via gradient-based or representation-based methods [20]. Other works, such as FASTTRACK [5] and OLMoTrace [13] take a more retrieval-oriented approach, leveraging semantic clustering or lexical overlap to match the LLMs' outputs to training examples. In fact, it was shown by Akyürek et al. [1] that even BM25 can serve as a strong baseline for tracing a model's outputs back to training examples.

There have also been other works that have proposed methods for mapping task-specific data back to pre-training data a priori, with the goal of understanding how knowledge contained in, or properties of, the pre-training data link to capabilities of LLMs on specific downstream tasks. For example, Kandpal et al. [9] proposed an approach which counts how often specific question and answer entities appear in documents in the pre-training corpus, with the aim of studying how the number of relevant documents to a question relates to its answering accuracy – similar to our experiment in Section 4.1. More recently, Wang et al. [22], proposed a method to measure an LLM's "memorization" versus "generalization" by mapping its output distribution to the task-specific pre-training data frequency.

Interplay of parametric versus external knowledge. It has been shown in previous works [9] – and further demonstrated in our experiments – that LLMs struggle to recall knowledge which is less frequent in its pre-training data. Even more so, parametric knowledge can become obsolete as time goes on. To address these shortcomings, RAG has been proposed as an approach to feed the LLM *external* knowledge at inference time to guide its generated output. With this, various research [7, 15, 17, 23] has focused on understanding this interplay between parametric knowledge within the LLMs and external knowledge provided to the LLM; see Xu et al. [24] for a survey on the topic.

We note, however, that a large chunk of these works has focused on evaluating this interplay for LLMs in which the pre-training data is *unknown*. This makes it difficult to properly disentangle the effects of parametric and external knowledge since the knowledge the LLM knows is unclear. NanoKnow provides a benchmark in which the interplay of the different knowledge sources can be explored confidently.

6 Conclusion

In this paper, we set out to answer a simple question: *how do LLMs know what they know?* Towards this goal, we introduced NanoKnow, a benchmark dataset that identifies questions which nanochat – or any LLM entirely pre-trained on FineWeb-Edu – has "seen" the answer to during pre-training, along with the pipeline to produce it. By projecting Natural Questions and SQuAD onto FineWeb-Edu, we found that over 66% and 71% of questions, respectively, have verifiable supported answers. NanoKnow now allows us to answer many interesting questions regarding how pre-training data shapes what language models know.

Using NanoKnow, we ran controlled experiments across various nanochat checkpoints, providing a clear picture of how parametric knowledge and external knowledge interact. What a model knows is largely a function of frequency: answers seen often are recalled reliably, while rare answers require external help. RAG closes this gap, improving accuracy where parametric knowledge is weakest. Even with RAG, models perform better on questions they have already seen, showing that parametric knowledge and external knowledge complement each other. At the same time, external knowledge is fragile: distractors degrade accuracy, particularly

when the answer is buried in the middle of the context, highlighting the importance of retrieval precision in practical RAG systems. Taken together, these findings replicate a wide range of results in the literature, further underscoring the reliability of NanoKnow.

While we built NanoKnow on FineWeb-Edu, the methodology used to create it can be extended to any open corpus. Indexing is a one-time cost, and projecting new benchmarks afterward is fast. This opens the door to questions we have not yet explored thoroughly: how does the topical composition of pre-training data influence downstream capabilities? Can answer frequency information guide more effective data curation? We leave these directions to future work and release NanoKnow for the community to conduct fair and controlled studies of how training data shapes what LLMs can and cannot do.

Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Additional funding was provided by Snowflake and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project).

References

[1] Ekin Akyürek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards Tracing Knowledge in Language Models Back to the Training Data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2429–2446.

[2] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. In *International Conference on Machine Learning*. 2397–2430.

[3] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying Memorization Across Neural Language Models. In *The Eleventh International Conference on Learning Representations*.

[4] Tyler A. Chang, Dheeraj Rajagopal, Tolga Bolukbasi, Lucas Dixon, and Ian Tenney. 2025. Scalable Influence and Fact Tracing for Large Language Model Pretraining. In *The Thirteenth International Conference on Learning Representations*.

[5] Si Chen, Feiyang Kang, Ning Yu, and Ruoxi Jia. 2024. FASTTRACK: Reliable Fact Tracing via Clustering and LLM-Powered Evidence Validation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. 5821–5836.

[6] Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2024)*. 719–729.

[7] Mehrdad Farahani and Richard Johansson. 2024. Deciphering the Interplay of Parametric and Non-Parametric Memory in Retrieval-Augmented Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 16966–16977.

[8] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How Can We Know What Language Models Know? *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438.

[9] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large Language Models Struggle to Learn Long-Tail Knowledge. In *International Conference on Machine Learning*. 15696–15707.

[10] Andrej Karpathy. 2025. nanochat: The Best ChatGPT That $100 Can Buy. https://github.com/karpathy/nanochat

[11] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 452–466.

[12] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.

[13] Jiacheng Liu, Taylor Blanton, Yanai Elazar, Sewon Min, Yen-Sung Chen, Arnavi Chheda-Kothary, Huy Tran, Byron Bischoff, Eric Marsh, Michael Schmitz, Cassidy Trier, Aaron Sarnat, Jenna James, Jon Borchardt, Bailey Kuehl, Evie Yu-Yen Cheng, Karen Farley, Taira Anderson, David Albright, Carissa Schoenick, Luca Soldaini, Dirk Groeneveld, Rock Yuren Pang, Pang Wei Koh, Noah A. Smith, Sophie Lebrecht, Yejin Choi, Hannaneh Hajishirzi, Ali Farhadi, and Jesse Dodge. 2025. OLMoTrace: Tracing Language Model Outputs Back to Trillions of Training Tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. 178–188.

[14] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.

[15] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9802–9822.

[16] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*.

[17] Cheng Qian, Xinran Zhao, and Tongshuang Wu. 2024. "Merge Conflicts!'" Exploring the Impacts of External Knowledge Distractors to Parametric Knowledge Graphs. In *First Conference on Language Modeling*.

[18] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2383–2392.

[19] Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How Much Knowledge Can You Pack Into the Parameters of a Language Model?. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5418–5426.

[20] Weiwei Sun, Haokun Liu, Nikhil Kandpal, Colin Raffel, and Yiming Yang. 2025. Enhancing Training Data Attribution with Representational Optimization. *arXiv preprint arXiv:2505.18513* (2025).

[21] Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*.

[22] Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. 2025. Generalization v.s. Memorization: Tracing Language Models' Capabilities Back to Pretraining Data. In *The Thirteenth International Conference on Learning Representations*.

[23] Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive Chameleon or Stubborn Sloth: Revealing the Behavior of Large Language Models in Knowledge Conflicts. In *The Twelfth International Conference on Learning Representations*.

[24] Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge Conflicts for LLMs: A Survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8541–8565.

[25] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388* (2025).

[26] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*. 1253–1256.

[27] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do Large Language Models Latently Perform Multi-Hop Reasoning?. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 10210–10229.

[28] Jun Zhao, Yongzhuo Yang, Xiang Hu, Jingqi Tong, Yi Lu, Wei Wu, Tao Gui, Qi Zhang, and Xuanjing Huang. 2025. Understanding Parametric and Contextual Knowledge Reconciliation within Large Language Models. In *Proceedings of the 39th International Conference on Neural Information Processing Systems*.