

Gargi: a 384M thinking language model trained from scratch on about 17B tokens

UDGAM LABS VERSION 1.0.0

Abstract

Gargi is a 384M-parameter English language model with an optional thinking mode. Udgam Labs trained it from scratch and released it with open weights under Apache-2.0 on 17 October 2026, together with its base model, Gargi-Base, and GGUF and MLX builds. Across all training stages it read about 17 billion tokens. It uses the Gemma 4 architecture and chat format [1], so the same weights run unchanged in transformers, llama.cpp, Ollama, LM Studio and MLX.

- **Data efficiency.** In one evaluation harness (lm-evaluation-harness, 0-shot), Gargi-Base comes within 2 points of Qwen2.5-0.5B on ARC-Easy (62.8 vs 64.7), ARC-Challenge (32.3 vs 32.4) and WinoGrande (54.6 vs 56.1). Qwen2.5-0.5B was pretrained on about 18 trillion tokens, as published by its authors [3], so Gargi used about 1,000 times less data.
- **Where it is behind.** Knowledge, everyday commonsense and maths still follow the amount of data. Gargi-Base is near chance on MMLU (25.5 vs 47.3 for Qwen2.5-0.5B) and behind on HellaSwag (41.5 vs 52.1) and PIQA (66.8 vs 70.0). SmoLLM2-360M and Granite 4.0 350M, pretrained on 4T and about 15T tokens as reported by their developers, are ahead of Gargi-Base on every benchmark except MMLU, where SmoLLM2-360M is level.
- **Maths.** With thinking on, the chat model solves 36.0% of the GSM8K test set (0-shot, greedy), and 32.9% with thinking off. Newer small reasoning models score far higher: Qwen3-0.6B scores 79.2, as reported in the MobileLLM-R1.5 model card [8].
- **Conversation.** On 50 held-out two-turn chats, Gargi repeats its previous reply in 0% of follow-ups with thinking off and 4% with thinking on, but its answers to follow-ups are often wrong.
- **Long context.** Gargi was trained at 32,768 tokens. It recalls a hidden 5-digit code in 100% of trials at 8K tokens, 93% at 16K and 90% at 20K, but only 13% at 24K and 0% at 32K. Six of the eight other small models we tested recall 100% at 24K. We recommend inputs of up to about 16K tokens.
- **Builds and speed.** The Q8_0 build (410 MB) stays close to f16 on GSM8K; Q4_K_M (238 MB) loses a few points. On an Apple M6 Mac mini that was also running other jobs, the Q8_0 file generated 273 tokens per second on the GPU and 248 on 4 CPU threads, and the MLX 4-bit build 480 tokens per second.
- **Safety.** Gargi had no dedicated safety-training round. It may follow harmful requests, and it is released as a research model that is not safety-tuned.

1. Introduction

Most open language models under one billion parameters are pretrained on trillions of tokens. That works, but it makes small models hard to reason about: when a 0.5B model has read 18 trillion tokens, it is hard to say how much of its ability comes from the amount of data and how much from the model and its training. Gargi asks the opposite question: how far does a small, fixed amount of openly licensed data go?

Gargi was built under four constraints:

1. **From scratch and open:** random initialisation, our own tokenizer, training text that is openly licensed, in the public domain, or synthetic text written by openly licensed models, and Apache-2.0 weights.
2. **Small:** 384M parameters, with a recommended 8-bit file of 410 MB that runs on a laptop.
3. **Standard formats:** an architecture and chat format that existing runtimes already support.
4. **Thinking and long inputs:** an optional written thought before the answer, and training at a 32,768-token context.

We count every token of training text the model read, in every stage, fine-tuning included: about 17 billion. We compare this with the pretraining token counts that other developers publish for their models, which are measured in trillions.

The result is uneven, and we report it that way:

- On reasoning-style multiple-choice tests (ARC and WinoGrande), Gargi-Base is within 2 points of Qwen2.5-0.5B.
- On knowledge (MMLU), everyday commonsense (HellaSwag, PIQA), maths and recall beyond about 20K tokens, it is behind, sometimes far behind.

Gargi is named after Gargi Vachaknavi, the Vedic philosopher remembered for asking the hardest questions.

This report covers:

- the models, builds and links (section 2) and the architecture (section 3);
- training at a high level (section 4);
- inference: chat format, thinking, settings, long inputs, speed and memory (section 5);
- evaluation, with every setting stated (section 6), and an analysis of data efficiency (section 7);
- limitations and safety (section 8) and release details (section 9).

2. Model overview

	GARGI	GARGI - BASE
What it is	Chat model, thinking on or off, single-turn and multi-turn chat	The pretrained base model after long-context training. It continues text and is not chat-tuned
Parameters	384.4M	384.4M
Context	Trained at 32,768 tokens; reliable recall up to about 16K	Trained at 32,768 tokens (the recall test of section 6.6 was run on the chat model)
Safety training	A small set of refusal examples; no dedicated safety round	None
Hugging Face	udgamlabs/Gargi	udgamlabs/Gargi-Base
Licence	Apache-2.0	Apache-2.0

Each repository holds the bf16 weights (`model.safetensors` , 769 MB), `config.json` , `generation_config.json` and the tokenizer files; the chat model also holds the chat template. The chat model has four more builds:

BUILD OF GARGI	REPOSITORY	SIZE	USE
GGUF Q8_0	udgamlabs/Gargi-GGUF	410 MB	recommended
GGUF Q4_K_M	same	238 MB	low-memory devices
GGUF f16	same	770 MB	reference precision
MLX bf16	udgamlabs/Gargi-MLX	769 MB	Apple silicon
MLX 4-bit, group size 64 (about 4.5 bits per weight)	udgamlabs/Gargi-MLX-4bit	216 MB	Apple silicon

The GGUF repository also holds an Ollama `Modelfile` . The GitHub repository [udgamlabs/gargi](#) (version 1.0.0) holds the documentation, five runnable examples (transformers, Ollama, a long-document question, llama.cpp and MLX) and SHA-256 checksums of every released file.

These replies were recorded on 17 October 2026 with the released weights (transformers, fp32, greedy, chat template). They show typical behaviour, including two wrong answers; they are not a representative sample:

REQUEST		THINKING WHAT THE MODEL RETURNED
"Who are you?"	off	"I'm Gargi, an AI trained from scratch by UdgamLabs. I have about 384 million parameters and I process up to roughly 16,000 tokens at a time. ..."
"Sara has 3 boxes of 6 pencils and gives away 4. How many are left?" plus the maths instruction (section 5.1)	on	Thought: We compute: $3 \text{ boxes} \times 6 = 18$ pencils. Gives away 4, left 14. Then a worked answer ending "Final answer: 14" (right)
"A train travels 60 km in 45 minutes. What is its speed in km/h?" plus the maths instruction	on	Thought ends: $60 \text{ km} / 0.75 \text{ h} = 60 \times (1/0.75) = 60 \times 4 = 240 \text{ km/h}$. Then "Final answer: 240" (wrong: 80)
"What is 5 squared?", answer "5 squared is 25.", then "and 6 squared?"	off	"6 squared is 36."
"What is the capital of Australia?"	off	"The capital of Australia is Sydney. ..." (wrong: Canberra)

3. Architecture

Gargi is a decoder-only transformer. Its `config.json` declares the `Gemma4ForCausalLM` class of `transformers` (model type `gemma4_text`) [1], so `transformers`, `llama.cpp` [25], `Ollama` [26] and `MLX` [27] load it with their existing Gemma 4 code. The details below come from the released `config.json` and the tensor shapes in `model.safetensors`.

Layers	20
Hidden width	1,024
Attention heads	16 query and 4 key-value heads (grouped-query attention [18])
Head size	64 in sliding-window layers, 128 in global layers
Local and global attention	Three sliding-window layers (window 512 tokens), then one global layer, repeated [19, 20]; layers 4, 8, 12, 16 and 20 are global
Position encoding	Rotary [21]: base 10,000 in sliding-window layers; base 1,000,000, proportional, in global layers
MLP	Gated, GELU (tanh approximation) [22], width 4,096
Normalisation	RMSNorm [23], as in the Gemma 4 text architecture, including RMS normalisation of queries and keys [24]
Output	Final-logit soft-cap of 15 [20]
Vocabulary	32,768 tokens
Embeddings	Untied input table and output projection, each $32,768 \times 1,024$ (33.55M parameters)
Not used	Per-layer embeddings, mixture of experts, shared key-value caches
Parameters	384,389,268 (384.4M), of which 317.3M are in the 20 layers
Context	32,768 positions, in <code>config.json</code> and in the GGUF files
Weights	bf16

Gargi has the same architecture and the same tokenizer as Udgam-384M, our earlier on-device tool-calling model [2]; the only difference in `config.json` is the context length (32,768 positions instead of 4,096). Gargi is a new model trained again from random initialisation, not a fine-tune of Udgam-384M.

Tokenizer. A byte-level BPE tokenizer with 32,768 tokens, trained by us. It includes Gemma 4's special tokens for turns, thoughts and tool calls, so the Gemma 4 chat template works unchanged.

Why sliding windows matter for long inputs. Only the 5 global layers keep keys and values for the whole input; the 15 sliding-window layers keep the last 512 tokens. By our arithmetic from `config.json`, at 16-bit precision the key-value cache therefore grows by about 10 KB per token (168 MB at 16,384 tokens and 336 MB at 32,768), plus a fixed 8 MB for the windows. If all 20 layers attended over the whole input, a 32,768-token cache would take about 840 MB.

4. Training (high level)

Both models were trained by Udgam Labs from random initialisation; no weights from any other model were used.

1. **Pretraining from scratch** on openly licensed and public-domain English text and synthetic text written by openly licensed models.
2. **Long-context training → Gargi-Base.** Continued pretraining at a sequence length of 32,768 tokens on long documents.
3. **Supervised fine-tuning → Gargi.** From Gargi-Base, on:
 4. step-by-step solutions to maths problems from the GSM8K and MATH training splits [15, 16], written by gpt-oss models [32], shown both with a written thought and without one, so that the same weights can answer either way;
 5. general chat and instruction examples, both single-turn and multi-turn conversations;
 6. a small set of refusal examples;
 7. examples in which the model states its own name and origin.

Across the three stages Gargi read about 17 billion tokens of training text. Gargi-Base, which stops before fine-tuning, read slightly fewer; both figures round to 0.017 trillion.

Data sources. The model card acknowledges these sources, used under their licences:

- **Web, books, papers and public records:** FineWeb [28] and Ultra-FineWeb [29] (ODC-By 1.0 / Apache-2.0); Wikipedia and Wikisource (CC-BY-SA); the Common Pile collections [30] (public-domain books from Project Gutenberg, the Library of Congress and pre-1929 publications; openly licensed arXiv papers, DOAB books and PubMed Central articles under CC-BY, CC-BY-SA or CC0; US Government Publishing Office records; permissively licensed source code).
- **UK Parliament:** contains Parliamentary information licensed under the Open Parliament Licence v3.0.
- **Synthetic and instruction data:** PleIAs SYNTH (CC-BY-4.0), FinePhrase (ODC-By), NVIDIA Nemotron datasets (CC-BY-4.0 and the NVIDIA Open Data License), Salesforce xLAM function-calling (CC-BY-4.0), Table-GPT (MIT), and step-by-step solutions and other examples written by OpenAI's gpt-oss models (Apache-2.0) [31–33].
- **Maths problems:** the GSM8K (MIT) and MATH (MIT) training splits. Their test splits were used only for evaluation.

Licences of individual documents vary within collections; we kept only documents whose licence allows this use.

Test-set hygiene. Pretraining and long-context documents were checked against the evaluation sets in this report (MMLU, ARC, HellaSwag, PIQA, WinoGrande and GSM8K, among others) for shared 13-word sequences, and documents with several shared sequences were dropped. The long documents used as filler in the recall test (section 6.6) were held out from training.

This report does not describe data proportions, training settings or compute.

5. Inference

5.1 Chat format and thinking

Gargi uses the standard Gemma 4 chat format. With thinking on, the system turn contains `<|think|>`, and the model writes its reasoning in a thought channel before the answer:

```
<bos><|turn>system
<|think|>
<turn|>
<|turn>user
Sara has 3 boxes of 6 pencils and gives away 4. How many are left?

Solve it step by step, then give the final answer on the last line as: Final
answer: <number><turn|>
<|turn>model
<|channel>thought
We compute: 3 boxes *6 =18 pencils. Gives away 4, left 14.
<channel|>(a worked answer in Markdown) ...
Final answer: 14<turn|>
```

With thinking off, there is no `<|think|>` and the model answers directly. The included chat template handles both: pass `enable_thinking=True` or `False` to `apply_chat_template` in `transformers`. Ollama turns thinking on by default; `/set nothink`, or `"think": false` in the API, turns it off.

- **When it thinks.** Gargi writes thoughts mainly on maths problems, and they are short: on the GSM8K test set, a thought averages 40.5 words.
- **The maths instruction.** Answers are most reliable when a maths question ends with *"Solve it step by step, then give the final answer on the last line as: Final answer: "*. Every GSM8K result in this report uses it.
- **Multi-turn chat.** Earlier turns are sent as ordinary user and model turns; the template drops earlier thoughts.

5.2 Running the model

The commands below are those in the model card:

```
ollama run hf.co/udgamlabs/Gargi-GGUF:Q8_0
llama-cli -hf udgamlabs/Gargi-GGUF:Q8_0 -c 16384
mlx_lm.generate --model udgamlabs/Gargi-MLX --prompt "A train travels 60 km in
45 minutes. What is its speed in km/h?"
```

In transformers (5.18 or newer), `tokenizer.apply_chat_template(messages, add_generation_prompt=True, enable_thinking=True, return_tensors="pt", return_dict=True)`, then `model.generate`. The examples in the GitHub repository show the same calls, including how to separate the thought from the answer.

Same tokens in every runtime. On 17 October 2026 we gave the same 8 chat prompts (thinking off and on, and one two-turn chat) to transformers (fp32, CPU) and to llama.cpp (llama-server, build d7a695e) with the released f16 and Q8_0 GGUF files. With greedy decoding, all 8 continuations of up to 64 tokens were identical, token for token, in all three.

Exactly one `<bos>`. Every training document started with `<bos>`, so every input should start with exactly one. We checked the input paths on 17 October 2026:

HOW THE INPUT IS BUILT	<code><BOS></code> TOKENS AT THE START
transformers, <code>apply_chat_template(..., tokenize=True)</code>	one
transformers, <code>tokenizer(text)</code>	one (recommended for the base model)
transformers, <code>tokenizer("<bos>" + text)</code>	two (avoid)
transformers, <code>tokenizer("<bos>" + text, add_special_tokens=False)</code>	one
llama.cpp and Ollama with the GGUF files	one (the files set <code>add_bos_token</code>)

5.3 Recommended settings

SETTING	DEFAULT (OLLAMA <code>MODELFILE</code> , <code>GENERATION_CONFIG.JSON</code> , GGUF METADATA)	FOR REPRODUCIBLE MATHS
Temperature	0.7	0 (greedy)
top-p / top-k	0.95 / 64	not used
Context	16,384 tokens in the Ollama <code>Modelfile</code>	16,384 or less
Thinking	on in Ollama; off unless requested in transformers	on
Maths instruction	append to maths questions	append

Every score in section 6 uses greedy decoding. We have not measured the released model's accuracy with the default sampling settings.

5.4 Long inputs

The model accepts 32,768 tokens, but recall is reliable only up to about 16K tokens (section 6.6). The Ollama `ModelFile` therefore sets a 16,384-token context, and the card's `llama.cpp` command passes `-c 16384`. The GitHub example `long_document_qa.py` warns when a file is longer than about 60,000 characters. For longer documents, split the text and ask about each part.

5.5 Speed and memory

We measured speed on 17 October 2026 on an Apple M6 Mac mini (12 CPU cores, 24 GB of unified memory, macOS 27.0.1) with the released GGUF files:

- `llama.cpp` build d7a695e, `llama-bench`, a 512-token prompt and 128 generated tokens, 5 repetitions;
- the GPU (Metal) and 4 CPU threads;
- peak memory is the macOS peak memory footprint of the CPU run. On the GPU, the weights sit in Metal buffers, which that figure does not fully count.

The Mac was also running other jobs (load average 6.7 to 9.6 on 12 cores), so the speeds are lower bounds.

RELEASED FILE	GPU PROMPT / GENERATE, TOKENS/S	4 CPU THREADS PROMPT / GENERATE, TOKENS/S	PEAK MEMORY, CPU RUN
Q8_0 (410 MB)	14,135 / 273	1,063 / 248	481 MB
Q4_K_M (238 MB)	13,417 / 399	784 / 288	325 MB
f16 (770 MB)	12,049 / 156	1,167 / 115	not comparable (the weights are read from the memory-mapped file; resident set 942 MB)

Long inputs (Q8_0). Reading a 16,384-token prompt ran at 7,778 tokens per second on the GPU (about 2.1 seconds) and 459 on 4 CPU threads (about 36 seconds). With 16,384 tokens already in context, generation ran at 207 tokens per second on the GPU and 44 on 4 CPU threads, against 273 and 248 with an empty context. The peak memory footprint of the CPU run with 16,384 tokens of context was 842 MB.

MLX. With `mlx 0.32.4` and `mlx-lm 0.32.2` on the same Mac, a 494-token chat prompt and 128 generated tokens (greedy, median of 5 runs after 2 warm-up runs):

RELEASED BUILD	PROMPT, TOKENS/S	GENERATE, TOKENS/S	PEAK METAL MEMORY (MLX-LM)
Gargi-MLX (bf16)	13,262	172	1.20 GB
Gargi-MLX-4bit	14,431	480	0.63 GB

We have not measured Gargi on a phone or on an NVIDIA GPU.

6. Evaluation

6.1 Setup

All numbers in this section were measured by Udgam Labs unless they are labelled as reported by others.

General benchmarks (sections 6.2 and 6.3):

- **Harness:** lm-evaluation-harness 0.4.13 [9], Hugging Face backend, bf16.
- **Tasks and items:** MMLU [10] (14,042 test questions), ARC-Easy and ARC-Challenge [11] (2,376 and 1,172 test questions), HellaSwag [12] (10,042 validation items), WinoGrande [13] (1,267 validation items), PIQA [14] (1,838 validation items), GSM8K [15] (1,319 test problems).
- **Shots:** 0-shot for the six multiple-choice tasks. GSM8K is 5-shot, greedy, without a chat template, scored with the harness's flexible-extract filter.
- **Metric per task,** the same for every model: accuracy for MMLU, ARC-Easy and WinoGrande; length-normalised accuracy for ARC-Challenge, HellaSwag and PIQA.
- **Start token:** Gargi, Gargi-Base and Udgam-384M were run with `add_bos_token=True`, so every input starts with one `<bos>`, as in training. The other models used their own tokenizer defaults; Qwen2.5, for example, adds no start token by design.
- **Where and when:** Gargi-Base (16 October 2026) and Gargi (17 October) were run on the Apple M6 Mac mini of section 5.5 (MPS, transformers 5.19.0, batch size 16). The other models were run on 13 October 2026 on an NVIDIA GPU (transformers 5.18.0, batch size 32).
- **Hardware check.** The same harness on an NVIDIA GPU gave Gargi-Base scores within 0.2 points of its Mac scores on ARC-Easy, HellaSwag, WinoGrande and PIQA.
- **Uncertainty:** the harness's standard errors for Gargi-Base are about ± 1.0 point on ARC-Easy, ± 1.4 on ARC-Challenge, ± 1.4 on WinoGrande, ± 1.1 on PIQA, ± 0.5 on HellaSwag and ± 0.4 on MMLU. Differences under about 2 points on the smaller sets should be read as small, not as clear.

- Training tokens of other models are as reported by their developers; we did not check them.

GSM8K with the chat template (section 6.4): our own script on the full 1,319-problem test set with the released weights, on 17 October 2026: transformers 5.18.0, bf16, NVIDIA GPU, batch size 64, 0-shot, greedy, the chat template, the maths instruction of section 5.1, and up to 512 new tokens. The answer is the number after the last "Final answer" in the reply (outside the thought), or the last number in the reply if that phrase is missing. Each problem is run with thinking on and with thinking off.

Multi-turn chat (section 6.5), **long-context recall** (section 6.6) and **quantisation** (section 6.7) are described in their sections.

6.2 Base models

MODEL	PARAMS	TRAINING TOKENS (REPORTED)	MMLU	ARC- E	ARC- C	HELLASWAG	WINOGRANDE	PIQA
Gargi-Base	384M	~0.017T (all stages)	25.5	62.8	32.3	41.5	54.6	66.8
Qwen2.5-0.5B	494M	~18T [3]	47.3	64.7	32.4	52.1	56.1	70.0
SmolLM2-360M	360M	~4T [4]	25.6	70.2	38.3	56.3	59.0	71.5
Granite 4.0 350M (base)	352M	~15T [6]	32.0	65.4	33.7	52.0	56.0	70.7
Gemma 3 270M (pretrained)	268M	~6T [5]	26.1	58.8	28.2	41.3	53.4	68.1
Udgam-384M- Base-4K	384M	—	25.0	59.1	29.9	39.7	51.7	66.1

Parameter counts are those reported by the harness. Udgam-384M-Base-4K is the base model of our earlier Udgam-384M release [2], scored with the public weights on 17 October 2026 in the same Mac setup as Gargi-Base.

Against Qwen2.5-0.5B (~18T tokens):

- Within 2 points on ARC-Easy (−1.9), ARC-Challenge (−0.1) and WinoGrande (−1.5). Each of these gaps is less than 1.4 times the combined standard error of the two scores.
- Behind on MMLU (−21.8), HellaSwag (−10.6) and PIQA (−3.2).

Against the others:

- **SmolLM2-360M (~4T):** behind on ARC-Easy (−7.4), ARC-Challenge (−6.0), HellaSwag (−14.8), WinoGrande (−4.4) and PIQA (−4.7); both are near chance on MMLU (−0.1).
- **Granite 4.0 350M base (~15T):** behind on all six, by 1.4 to 10.5 points.

- **Gemma 3 270M (~6T, a smaller model):** ahead on ARC-Easy (+4.0), ARC-Challenge (+4.1) and WinoGrande (+1.2), level on HellaSwag (+0.2), behind on PIQA (-1.3) and MMLU (-0.6).
- **Udgam-384M-Base-4K** (same architecture and tokenizer): Gargi-Base is higher on all six, by 0.5 (MMLU) to 3.7 (ARC-Easy) points; the MMLU and PIQA differences (0.5 and 0.7) are within noise.

Reasoning-style multiple choice (ARC, WinoGrande) is where Gargi-Base comes closest to models trained on far more data. Knowledge-heavy and everyday-commonsense benchmarks (MMLU, HellaSwag, PIQA) still scale with data, and there it is clearly behind. Section 7 looks at this in more detail.

GSM8K, 5-shot, base models. Few-shot maths is where the data-rich base models lead most clearly:

MODEL	GSM8K, 5-SHOT (FLEXIBLE-EXTRACT)
Gargi-Base	5.9
Qwen2.5-0.5B	34.2
Granite 4.0 350M (base)	21.9
SmolLM2-360M	3.9
Gemma 3 270M (pretrained)	2.4

6.3 Chat model

BENCHMARK (HARNESS OF SECTION 6.1)	GARGI-BASE	GARGI
MMLU	25.5	24.9
ARC-Easy	62.8	62.2
ARC-Challenge	32.3	31.0
HellaSwag	41.5	42.0
WinoGrande	54.6	55.2
PIQA	66.8	66.4
GSM8K, 5-shot, flexible-extract / strict-match	5.9 / 4.9	19.8 / 9.2

Fine-tuning moves the multiple-choice scores by at most 1.3 points (ARC-Challenge, -1.3), within about one standard error, and more than triples few-shot GSM8K. On the same 5-shot GSM8K test, Udgam-384M-Chat, our earlier chat model of the same size, scores 1.5. The data-efficiency comparison of section 6.2 uses the base model, because the other models in it are base models.

6.4 Maths with the chat template

GSM8K TEST, 1,319 PROBLEMS (SETTING OF SECTION 6.1)	ACCURACY	REPLIES WITH A "FINAL ANSWER" LINE
Gargi, thinking on	36.0	94.8%
Gargi, thinking off	32.9	99.5%

The binomial standard error of each score is about ± 1.3 points. The gain from thinking, 3.1 points, is about 1.7 times the combined standard error of two independent scores. Both runs use the same problems, so a paired comparison would be tighter, but this run did not save per-problem results.

Newer small reasoning models, as reported by others. These figures come from their developers' own pipelines; we did not reproduce them, and the settings differ from ours:

MODEL	GSM8K	SETTING AND SOURCE
Qwen3-0.6B (thinking)	79.2	0-shot pass@1, as reported in the MobileLLM-R1.5 model card [8]
MobileLLM-R1.5-360M	52.8	0-shot pass@1, reported in its model card [8]
Granite 4.0 H 350M	39.3	8-shot, reported in its model card [6]
Gargi, thinking on (ours)	36.0	0-shot, greedy, chat template, measured by us

The rows are not directly comparable, but the gaps are large. These models were also trained on far more data: Qwen3 on 36 trillion tokens, as its developers report [7]. Gargi is not a maths specialist.

6.5 Multi-turn chat

We ran two tests through Ollama 0.35.1 with the released Q8_0 file, greedy, one request at a time, on 17 October 2026:

- **Follow-ups.** 50 two-turn chats that we wrote by hand and kept out of training, such as "What is 5 squared?" followed by "and 6 squared?". A reply counts as a near-copy when its first 400 characters match the previous reply's with a `difflib` similarity above 0.6. For 30 follow-ups with a checkable answer, a reply counts as right when it contains the expected answer. Replies were capped at 400 tokens with a 4,096-token context.
- **A maths problem as the second question.** GSM8K test problems 200 to 399, thinking on, up to 512 tokens. Each problem was asked alone, and again as the second question of a chat whose first turn was another GSM8K problem (items 0 to 199) with its reference solution.

TEST	THINKING OFF	THINKING ON
Follow-ups whose reply near-copies the previous reply (50)	0%	4%
Checkable follow-ups answered right (30)	56.7%	43.3%
GSM8K problems 200–399: asked alone / as the second question (200)		34.0 / 36.0

Gargi keeps track of what a follow-up refers to, and a maths problem asked second in a chat is solved about as often as one asked alone. The follow-up answers themselves are often wrong: the failures are mostly wrong facts or arithmetic on the right question. With 30 checkable items, each of those two scores has a standard error of about ± 9 points.

6.6 Long-context recall

Method. A 5-digit "locker code" is placed near the start of a long document of real text (after 64 tokens of filler), and the model is asked for it at the end:

- The filler comes from the same held-out long documents for every model, re-tokenised for each model, so lengths are counted in each model's own tokens.
- Inputs are plain text, starting with the start token that each model's own format uses (`<bos>` for Gargi).
- A trial counts as recalled only if the model's top prediction is right for every token of the code (teacher forcing).
- 30 trials per length; lengths beyond a model's declared context are marked n/a.
- transformers 5.18.0, bf16, NVIDIA GPUs. The other models were run on 16 October 2026, Gargi on 17 October.

MODEL	PARAMS	DECLARED CONTEXT	8K	16K	20K	24K	32K
Gargi	384M	32,768 (trained)	100	93	90	13	0
Qwen3-0.6B	600M	40,960	100	100	100	100	100
Hunyuan-0.5B-Instruct	500M	262,144	100	100	100	100	100
Falcon-H1-Tiny-R-0.6B	600M	262,144	100	97	100	100	97
Granite 4.0 H 350M	350M	32,768	100	100	100	100	100
ERNIE-4.5-0.3B	300M	131,072	100	100	100	100	100
Gemma 3 270M (instruction-tuned)	270M	32,768	100	100	100	100	63
LFM2.5-350M	350M	128,000	90	0	0	0	0
SmolLM2-360M-Instruct	360M	8,192	60	n/a	n/a	n/a	n/a

Declared context is the value in each model's published configuration [34]. With 30 trials, a score of 93% has a standard error of about ± 5 points.

- **Where Gargi is reliable:** up to about 16K tokens (100% at 8K, 93% at 16K), and 90% at 20K.
- **Where it fails:** 13% at 24K and 0% at 32K, although it was trained on and accepts 32K-token inputs.
- **Against the others:** six of the eight other models recall 100% at 24K, and five recall at least 97% at 32K. Three of those six are smaller than Gargi: Granite 4.0 H 350M, ERNIE-4.5-0.3B and Gemma 3 270M. Only LFM2.5-350M (0% from 16K, despite a 128K declared context) and SmolLM2-360M-Instruct (60% at 8K, its declared limit) score lower.
- **What it does not show.** Recall of a single planted code is an easy test of long context; it does not show that the model can reason over or summarise a long document. Harder tests such as RULER [17] were not run.

6.7 Quantisation

GSM8K through Ollama 0.35.1 with each released GGUF file: the first 200 test problems, thinking on, greedy, the maths instruction, up to 512 new tokens, on 17 October 2026.

BUILD	SIZE	GSM8K (200)	SAME RIGHT/WRONG RESULT AS F16
f16	770 MB	39.0	–
Q8_0 (recommended)	410 MB	37.5	96.5%
Q4_K_M	238 MB	32.0	87%

At 200 problems each score has a standard error of about ± 3.4 points. Q8_0 stays close to f16; Q4_K_M loses a few points. The Q4_K_M file was quantised without an importance matrix. A re-run on 17 October 2026, with a separate Ollama server and per-problem results saved, gave the same three scores and the same agreement figures.

7. Analysis

7.1 Data efficiency, framed honestly

The table puts the base-model scores of section 6.2 next to the amount of training data. The group means are unweighted, and the arithmetic is ours.

MODEL	TRAINING TOKENS (REPORTED)	DATA RELATIVE TO GARGI	MEAN OF ARC-E, ARC-C, WINOGRANDE	MEAN OF MMLU, HELLASWAG, PIQA
Gargi-Base	~17B (all stages)	1×	49.9	44.6
Gemma 3 270M	~6T	~350×	46.8	45.2
SmolLM2- 360M	~4T	~235×	55.8	51.1
Granite 4.0 350M (base)	~15T	~880×	51.7	51.6
Qwen2.5-0.5B	~18T	~1,060×	51.1	56.5

What the table shows:

- Reasoning-style multiple choice comes early at this size.** On ARC and WinoGrande, Gargi-Base averages 1.2 points below Qwen2.5-0.5B with about 1,000 times less data.
- Knowledge and everyday commonsense do not.** On MMLU, HellaSwag and PIQA, the gap to Qwen2.5-0.5B averages 11.9 points, and MMLU alone accounts for 21.8. These benchmarks reward facts and broad exposure, which come from reading more text.
- The claim is about data needed, not rank.** SmolLM2-360M, with about 235 times more data, scores higher than Gargi-Base on both groups, and so does Granite 4.0 350M. Gargi-Base is not the strongest base model of its size. Its result is reaching this level with far less data.
- Two of the three close benchmarks sit near chance.** Chance is 50% on WinoGrande and about 25% on ARC-Challenge. Every model in the table is at most 9 points above chance on WinoGrande, and all but SmolLM2-360M are at most 9 points above chance on ARC-Challenge. Being level there says less than being level on ARC-Easy, where Gargi-Base is 1.9 points behind Qwen2.5-0.5B and 7.4 behind SmolLM2-360M.
- The efficiency does not carry over to maths or long context.** Few-shot GSM8K (section 6.2) and recall beyond 20K tokens (section 6.6) both favour models trained on far more data, including smaller ones.

Caveats:

- What is counted.** Our count includes every stage of training, fine-tuning included; the other developers report pretraining tokens. Definitions may differ, for example in whether repeated data is counted.
- Synthetic data.** Part of Gargi's training text was written by much larger open models, which were themselves trained on far more data. The ratios count the tokens Gargi read, not the data behind those models.

- **One run.** These results come from one training run; we report no variation across runs.
- **Our harness.** The other models were scored with our settings. Their developers report different numbers with other harnesses and prompts.

7.2 Where Gargi is behind

- **Knowledge.** MMLU is at chance (25.5 for the base model, 24.9 for the chat model). The model answers factual questions fluently and often wrongly, as in the "Sydney" reply of section 2.
- **Everyday commonsense.** HellaSwag is 10 to 15 points behind SmolLM2-360M, Granite 4.0 350M and Qwen2.5-0.5B.
- **Maths.** 36.0% on GSM8K with thinking. Thoughts are short (40.5 words on average), and a plausible-looking thought can contain a wrong step, as in the train example of section 2 ($60 \div 0.75$ written as 60×4).
- **Long context.** Recall falls from 90% at 20K tokens to 13% at 24K, well inside the trained 32,768-token window.

7.3 What thinking buys

On the full GSM8K test set, thinking on scores 36.0% and thinking off 32.9%. Thinking makes replies longer: with thinking on, 94.8% of replies end with the requested "Final answer" line, against 99.5% with thinking off, because some thoughts and worked answers do not finish within 512 tokens. Thinking helps on maths, by a modest margin; on other requests the model usually writes no thought.

8. Limitations and safety

8.1 Limitations

- **Facts and arithmetic.** Often wrong on facts, arithmetic and code. Double-check anything important.
- **Knowledge.** Limited world knowledge (MMLU at chance).
- **Long inputs.** Reliable recall only up to about 16K tokens (16–20K at best), although the model accepts 32K.
- **Language.** English only.
- **Follow-ups.** The model tracks what a follow-up refers to, but answers to follow-ups are often wrong (section 6.5).
- **Evaluation.**

- The other models in section 6.2 were scored on a different machine and transformers version from Gargi; tasks, shots and metrics were identical (section 6.1).
- The multi-turn and quantisation tests are small (30 to 200 items).
- The recall test checks a single planted code, not understanding of long documents.
- The results come from one training run.
- The speed figures come from one Mac that was running other jobs.
- **Machine-generated data.** Part of the training text was written by other language models and can carry their errors and habits.
- **Base model.** Gargi-Base has no instruction or safety tuning and will continue any text. Do not deploy it directly to end users.

Good fits: learning and experimenting with small models; offline, on-device chat and step-by-step maths explanations where mistakes are acceptable; reading and answering questions about documents of up to about 16K tokens.

8.2 Safety

Gargi has **not been safety-tuned** beyond a small set of refusal examples. There was no dedicated safety-training round, and we ran no safety benchmark on it.

- In our tests it **may follow harmful requests**. For example, it gave steps when asked how to break into a neighbour's house. In a re-check on 17 October 2026 (Q8_0 through Ollama; thinking off and on, each greedy and with 3 sampled seeds at the default settings), none of the 8 replies to that request began with a refusal, and both greedy replies began a step-by-step guide.
- The model card notes that it sometimes lists other assistants it is not ("I'm not ChatGPT, Gemini, ...") without being asked. In the same re-check (4 identity questions and 15 everyday requests, 152 replies), no reply named another assistant unprompted, so this appears to be rare.

Treat Gargi as a research and hobby model:

- put a moderation layer in front of any user-facing deployment;
- do not use it for medical, legal, financial or safety-critical advice;
- do not deploy the base model to end users.

The try box at udgamlabs.com/models/gargi puts a filter in front of the model that declines clearly harmful requests. The filter belongs to the try box, not to the weights: downloaded copies of Gargi have no filter.

9. Release details

ITEM	WHERE
Chat model: weights and model card	huggingface.co/udgamlabs/Gargi
Base model	huggingface.co/udgamlabs/Gargi-Base
GGUF builds and Ollama <code>Modelfile</code>	huggingface.co/udgamlabs/Gargi-GGUF
MLX builds	Gargi-MLX , Gargi-MLX-4bit
Documentation, examples and checksums (v1.0.0)	github.com/udgamlabs/gargi
Try box and model page	udgamlabs.com/models/gargi

- **Release:** 17 October 2026, version 1.0.0.
- **Requirements:**
 - transformers 5.18 or newer (we tested 5.18.0 and 5.19.0);
 - a llama.cpp build with Gemma 4 support (we tested d7a695e);
 - Ollama with the `gemma4` renderer (we tested 0.35.1);
 - mlx-lm (we tested 0.32.2).
- **Files:** bf16 safetensors (769 MB per model); GGUF and MLX builds as listed in section 2; `SHA256SUMS` in the GitHub repository.
- **Licence:** Apache-2.0 for the weights and tokenizer. The chat template is the Gemma 4 template (Apache-2.0). The training data sources carry their own licences (section 4).
- **Developer:** Udgam Labs, udgamlabs.com.
- **Contact:** the Community tab of the Hugging Face model pages, or issues on the GitHub repository.

```
@misc{udgamlabs2026gargi,  
  title      = {Gargi: a 384M language model trained from scratch on 17B  
tokens},  
  author     = {{Udgam Labs}},  
  year      = {2026},  
  howpublished = {\url{https://huggingface.co/udgamlabs/Gargi}}  
}
```

10. Conclusion

A 384M-parameter model trained from scratch on about 17 billion tokens of openly licensed, public-domain and synthetic text comes within 2 points of Qwen2.5-0.5B, trained on about 18 trillion tokens, on ARC-Easy, ARC-Challenge and WinoGrande. That is about 1,000 times less data for a similar score on reasoning-style multiple choice. It also thinks step by step on maths when asked, keeps track of follow-up questions, reads up to about 16K tokens reliably, and runs on a laptop from a 410 MB file.

The same evaluation shows the limits just as clearly:

- **Data still wins on knowledge and commonsense.** MMLU is at chance, and HellaSwag and PIQA are behind every larger-data model we compared.
- **Maths.** 36.0% on GSM8K with thinking on, far below newer small reasoning models.
- **Long context.** Recall fails beyond about 20K tokens, while several smaller models hold it to 32K.
- **Safety.** There was no safety-training round.

Gargi is not the strongest model of its size. What it shows is how far a small amount of data goes on some kinds of test and how far it falls short on others. The weights, builds and examples are open under Apache-2.0, so the results can be checked and the models can be fine-tuned.

11. References

1. Gemma Team (2026). *Gemma 4 Technical Report*. [arXiv:2607.02770](https://arxiv.org/abs/2607.02770).
2. Udgam Labs (2026). *Udgam-384M: a small on-device tool-calling model trained from scratch*. Technical report, udgamlabs.com/reports/udgam-384m.
3. Qwen Team: Yang, A., Yang, B., Zhang, B., Hui, B. et al. (2024). *Qwen2.5 Technical Report*. [arXiv:2412.15115](https://arxiv.org/abs/2412.15115). Model card: huggingface.co/Qwen/Qwen2.5-0.5B.
4. Ben Allal, L., Lozhkov, A., Bakouch, E., Martín Blázquez, G., Penedo, G. et al. (2025). *SmolLM2: When Smol Goes Big -- Data-Centric Training of a Small Language Model*. [arXiv:2502.02737](https://arxiv.org/abs/2502.02737). Model card: huggingface.co/HuggingFaceTB/SmolLM2-360M.
5. Gemma Team: Kamath, A., Ferret, J., Pathak, S., Vieillard, N. et al. (2025). *Gemma 3 Technical Report*. [arXiv:2503.19786](https://arxiv.org/abs/2503.19786). Model card: huggingface.co/google/gemma-3-270m.
6. IBM. *Granite 4.0 Nano* model cards: [granite-4.0-350m-base](https://huggingface.co/ibm-granite/granite-4.0-350m-base) and [granite-4.0-h-350m](https://huggingface.co/ibm-granite/granite-4.0-h-350m), Hugging Face (accessed 17 October 2026).
7. Qwen Team: Yang, A., Li, A., Yang, B., Zhang, B. et al. (2025). *Qwen3 Technical Report*. [arXiv:2505.09388](https://arxiv.org/abs/2505.09388). Model card: huggingface.co/Qwen/Qwen3-0.6B.

8. Zhao, C., Chang, E., Liu, Z., Chang, C.-J., Wen, W. et al. (2025). *MobileLLM-R1: Exploring the Limits of Sub-Billion Language Model Reasoners with Open Training Recipes*. [arXiv:2509.24945](https://arxiv.org/abs/2509.24945). Model card with the GSM8K figures used here: huggingface.co/facebook/MobileLLM-R1.5-360M (accessed 17 October 2026).
9. Biderman, S., Schoelkopf, H., Sutawika, L., Gao, L., Tow, J. et al. (2024). *Lessons from the Trenches on Reproducible Evaluation of Language Models*. [arXiv:2405.14782](https://arxiv.org/abs/2405.14782). Code: github.com/EleutherAI/lm-evaluation-harness.
10. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M. et al. (2020). *Measuring Massive Multitask Language Understanding*. [arXiv:2009.03300](https://arxiv.org/abs/2009.03300).
11. Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A. et al. (2018). *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*. [arXiv:1803.05457](https://arxiv.org/abs/1803.05457).
12. Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A. and Choi, Y. (2019). *HellaSwag: Can a Machine Really Finish Your Sentence?* [arXiv:1905.07830](https://arxiv.org/abs/1905.07830).
13. Sakaguchi, K., Le Bras, R., Bhagavatula, C. and Choi, Y. (2019). *WinoGrande: An Adversarial Winograd Schema Challenge at Scale*. [arXiv:1907.10641](https://arxiv.org/abs/1907.10641).
14. Bisk, Y., Zellers, R., Le Bras, R., Gao, J. and Choi, Y. (2019). *PIQA: Reasoning about Physical Commonsense in Natural Language*. [arXiv:1911.11641](https://arxiv.org/abs/1911.11641).
15. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H. et al. (2021). *Training Verifiers to Solve Math Word Problems*. [arXiv:2110.14168](https://arxiv.org/abs/2110.14168). (GSM8K.)
16. Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S. et al. (2021). *Measuring Mathematical Problem Solving With the MATH Dataset*. [arXiv:2103.03874](https://arxiv.org/abs/2103.03874).
17. Hsieh, C.-P., Sun, S., Krizan, S., Acharya, S., Rekesh, D. et al. (2024). *RULER: What's the Real Context Size of Your Long-Context Language Models?* [arXiv:2404.06654](https://arxiv.org/abs/2404.06654). The needle-in-a-haystack test it extends: Kamradt, G. (2023), github.com/gkamradt/needle-in-a-haystack.
18. Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F. and Sanghavi, S. (2023). *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. [arXiv:2305.13245](https://arxiv.org/abs/2305.13245).
19. Beltagy, I., Peters, M. E. and Cohan, A. (2020). *Longformer: The Long-Document Transformer*. [arXiv:2004.05150](https://arxiv.org/abs/2004.05150).
20. Gemma Team (2024). *Gemma 2: Improving Open Language Models at a Practical Size*. [arXiv:2408.00118](https://arxiv.org/abs/2408.00118).
21. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B. and Liu, Y. (2021). *RoFormer: Enhanced Transformer with Rotary Position Embedding*. [arXiv:2104.09864](https://arxiv.org/abs/2104.09864).
22. Shazeer, N. (2020). *GLU Variants Improve Transformer*. [arXiv:2002.05202](https://arxiv.org/abs/2002.05202).

23. Zhang, B. and Sennrich, R. (2019). *Root Mean Square Layer Normalization*. [arXiv:1910.07467](https://arxiv.org/abs/1910.07467).
24. Henry, A., Dachapally, P. R., Pawar, S. and Chen, Y. (2020). *Query-Key Normalization for Transformers*. [arXiv:2010.04245](https://arxiv.org/abs/2010.04245).
25. ggml-org. *llama.cpp* and the GGUF format. github.com/ggml-org/llama.cpp.
26. Ollama. github.com/ollama/ollama.
27. Apple machine learning research. *MLX* and *mlx-lm*. github.com/ml-explore/mlx; github.com/ml-explore/mlx-lm.
28. Penedo, G., Kydlíček, H., Ben Allal, L., Lozhkov, A., Mitchell, M. et al. (2024). *The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale*. [arXiv:2406.17557](https://arxiv.org/abs/2406.17557).
29. Wang, Y., Fu, Z., Cai, J., Tang, P., Lyu, H. et al. (2025). *Ultra-FineWeb: Efficient Data Filtering and Verification for High-Quality LLM Training Data*. [arXiv:2505.05427](https://arxiv.org/abs/2505.05427).
30. Kandpal, N., Lester, B., Raffel, C., Majstorovic, S., Biderman, S. et al. (2025). *The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text*. [arXiv:2506.05209](https://arxiv.org/abs/2506.05209).
31. Dataset cards: [PleIAs/SYNTH](#), [HuggingFaceFW/finephrase](#), [nvidia/Nemotron-Pretraining-Specialized-v1](#), [Salesforce/xlam-function-calling-60k](#).
32. OpenAI (2025). *gpt-oss-120b & gpt-oss-20b Model Card*. [arXiv:2508.10925](https://arxiv.org/abs/2508.10925).
33. Liu, Z. et al. (2024). *APIGen: Automated Pipeline for Generating Verifiable and Diverse Function-Calling Datasets*. [arXiv:2406.18518](https://arxiv.org/abs/2406.18518). Li, P., He, Y. et al. (2023). *Table-GPT: Table-tuned GPT for Diverse Table Tasks*. [arXiv:2310.09263](https://arxiv.org/abs/2310.09263).
34. Model cards of the other recall models: [Qwen/Qwen3-0.6B](#), [tencent/Hunyuan-0.5B-Instruct](#), [tiiuae/Falcon-H1-Tiny-R-0.6B](#), [ibm-granite/granite-4.0-h-350m](#), [baidu/ERNIE-4.5-0.3B-PT](#), [google/gemma-3-270m-it](#), [LiquidAI/LFM2.5-350M](#), [HuggingFaceTB/SmolLM2-360M-Instruct](#) (accessed 17 October 2026).
35. Udgam Labs. Gargi model cards, GitHub repository (examples and `SHA256SUMS`) and model page, October 2026.

About Udgam Labs. Udgam Labs is an independent AI research lab. We work from first principles: before we build, we understand why things work, whether we are training a model, fine-tuning one or applying AI to a new problem. Udgam (उद्गम) is Sanskrit for origin. udgamlabs.com