Rillor

EXPLAINER

Fine-tuning data formats: instruction, chat and preference records

Data preparation for LLM fine-tuning starts with the record: instruction, chat, preference or unpaired feedback records, how each is laid out, which tokens are trained and how files are checked.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Fine-tuning data is a file of records, and the training method decides what each record holds, how it is laid out and which of its tokens the model learns from.

- Fine-tuning records come in a few types, and each training method reads particular ones. One open-source training library defines text and prompt-completion records for supervised fine-tuning, preference records that pair a chosen and a rejected reply to one prompt, unpaired records with one reply and a true or false label, and prompt-only records [1].
- Chat records hold a conversation as a list of messages, each with a role, such as system, user, assistant or tool, and its content [1, 2]. A chat template turns those messages into one token sequence with control tokens, and models fine-tuned from the same base model can use different templates [3].
- Which tokens are trained is a setting. By default, one open-source library trains only on the completion of a prompt-completion record [4], and one fine-tuning service only on the assistant messages of a chat record [2]. A 2024 study found that adding loss on the instruction helped with long instructions and short outputs, or with few examples [5].
- Preference pairs can be built from rankings or ratings. One 2022 study turned each ranking of 4 to 9 replies into every possible pair [6]. A 2023 study took prompts with four rated replies each and paired the reply with the highest mean rating against one of the other three, picked at random [7].
- The four hosted services read for this report take text records as JSON Lines, one JSON value per line in UTF-8 [2, 8, 9, 10, 11]. One of them checks encoding, parsing, line count and size on the client, then roles and tool calls on its server after upload [2].
- Provider guidance puts quality before volume [9, 12]. One model provider's guide reports gains from 50 to 100 examples [8], and the LIMA study tuned a 65-billion-parameter model on 1,000 curated prompts and responses [13].
- Fine-tuning records can change behaviour beyond the task. A 2023 study broke one hosted model's safety guardrails with 10 adversarially designed training examples, for less than $0.20, and benign datasets also degraded safety, to a lesser extent [14].
- Evaluation items can leak into training records. The Tülu 3 project checked its training prompts for 8-gram overlap with its evaluations [15], and a 2023 study found that paraphrased or translated test items pass string-matching checks [16].

Background

What a fine-tuning record holds

Fine-tuning data is a set of worked examples. In one model provider's guide, each line of a training file holds "an example user prompt and a correct response from the model" [8]. Other methods read other shapes, such as a prompt with a preferred and a rejected reply, or one reply with a label [1]. One open-source training library describes every dataset by two properties. Its format "refers to how the data is structured", standard or conversational. Its type "is associated with the specific task the dataset is designed for", and each type "is characterized by its columns" [1].

Where the layouts came from

FLAN, first posted in 2021, defined instruction tuning as "finetuning language models on a collection of tasks described via instructions" [17]. Later collections gathered instructions at larger scale [18, 19] or generated them with a model [20]. One model developer's 2022 study collected rankings of a model's replies and trained a reward model on the pairwise comparisons [6]. Direct preference optimization (DPO) trains on such comparisons directly, with "a simple classification loss" [21], and KTO needs only a signal of whether one output is desirable [22]. The documentation read for this report lists the resulting record types side by side [1, 2].

Who this report is for

This report is for ML engineers and data teams who prepare fine-tuning files, and for buyers who check them. It covers the record layer: types, templates, loss settings and file checks.

Method

What was examined. Five record types (instruction, chat, tool-calling, preference and unpaired feedback records), the chat templates that turn records into tokens, the settings that decide which tokens are trained, the file rules and checks, and the failures that studies report.

Sources. Documentation from two open-source libraries [1, 3, 4], one model provider [8, 12, 23], two cloud providers [9, 10] and one fine-tuning service [2], the JSON Lines and JSON specifications [11, 24], one dataset release [25], and 21 papers that define record types or test their effects.

How. Each documentation page was read in full on 8 November 2026, and [9] again on 10 November 2026 after its update. Each is cited with its viewed date, and its statements are given in the present tense. Papers were read on their arXiv abstract pages and in full text, and the year given for a paper is the year it was first posted. Field names are written as each source writes them. Models are described by size and documentation by the kind of publisher. Numbers computed for this report are marked derived.

Findings

1. Format and type are separate choices

One open-source training library describes each dataset by a format and a type [1]. The format is standard or conversational, and conversational records "contain sequences of messages where each message has a `role`" and "`content`" [1]. The type follows the task, and the library lists which of its trainers read each type (Table 1).

Type	Columns (standard format)	Read by
Language modeling	`text` (`messages` when conversational)	Supervised fine-tuning (SFT)
Prompt-completion	`prompt`, `completion`	Supervised fine-tuning
Preference	`prompt`, `chosen`, `rejected` (the prompt may be implicit)	DPO and KTO (explicit prompt recommended), reward modeling (implicit prompt recommended)
Unpaired preference	`prompt`, `completion`, `label` (true or false)	KTO
Prompt-only	`prompt`	GRPO and RLOO

Table 1. Dataset types and the trainers that read them, from one open-source training library's guide [1].

The types are processed differently. Through a chat template, a prompt-only record ends with the marker that opens the assistant's turn, while a language modeling record is treated as a complete sequence and closed with an end-of-text token [1].

Records convert between types. The library provides conversion utilities and recommends converting before the chat template is applied [1]. It warns that the chosen and rejected completions of a preference dataset "can be both good or bad". Splitting pairs into unpaired records therefore needs every chosen reply to be good and every rejected reply bad [1].

```
{"prompt": "What is the capital of France?", "completion": "The capital of France is
Paris."}
{"prompt": "The sky is", "chosen": " blue.", "rejected": " green."}
```

Example 1. A prompt-completion record from one cloud provider's guide [10] and a preference record from one open-source library's guide [1].

Hosted services name the same parts differently. One fine-tuning service lists conversational, instruction, preference, generic text, tool-calling and reasoning formats [2]. It names a preference record's fields `input`, `preferred_output` and `non_preferred_output`, and its server rejects a file that has two possible formats [2]. One cloud provider takes `prompt` and `completion` fields for tasks that are not conversations [10]. Another calls the parts of a chat `systemInstruction` and `contents`, gives each content a `role` and `parts`, and names the assistant role `model` [9].

Service layouts also depend on the service staying open. As one model provider's guides stated on 8 November 2026, its fine-tuning platform is closed to new users, and eligible existing customers can create training jobs until 6 January 2027, unless the base model shuts down earlier [8, 23].

2. Instruction records: an instruction, an optional input and an output

FLAN found that instruction tuning "substantially improves zero-shot performance on unseen tasks" [17]. It rewrote 62 text datasets into an instructional format, with ten templates per dataset composed by hand, up to three of which "turned the task around", and formatted each example with a template picked at random [17].

Collections grew from there. Super-NaturalInstructions, from 2022, holds 1,616 tasks with expert-written instructions across 76 task types [18]. A 2022 study tuned a 540-billion-parameter model on 1.8 thousand tasks, and it outperformed the untuned model by 9.4% on average [19]. The Flan Collection, from 2023, found that a mix of zero-shot, few-shot and chain-of-thought templates in training improves results in all three settings, and that adding 10% few-shot prompts improves zero-shot results by 2% or more [26].

A 52,000-example instruction dataset that Stanford researchers released in 2023 uses three fields [25]. Each record has an `instruction` that describes the task, an optional `input` with context, and an `output`, the answer generated by a hosted model [25]. Each of the 52,000 instructions is unique, around 40% of the records have an input, and the data cost less than $500 to produce [25]. Two prompt templates, one with an input and one without, wrap the fields under the headings "### Instruction:" and "### Response:" [25]. The dataset is licensed for non-commercial use only [25].

Self-Instruct, from 2022, generated its records with a model: 52,445 instructions and 82,439 instances, 35,878 of them with an empty input [20]. To make the tuned model robust to formats, it encoded the instruction and input with several templates, with and without prefixes such as "Task:", "Input:" and "Output:" [20]. Tuning on the data gave a 33% absolute improvement over the original model on Super-NaturalInstructions [20].

Generated records need review. In a Self-Instruct sample of 200 instructions, 92% described a valid task, 79% of inputs were appropriate, 58% of outputs were correct and acceptable, and 54% had every field valid [20]. Datasets also teach different things. A 2023 study of 12 instruction datasets, on models from 6.7 to 65

billion parameters, found that each dataset can uncover or enhance specific skills, and that no single dataset or combination led on every evaluation [27].

3. Chat records: roles, turns and tool calls

A chat record holds one conversation. One fine-tuning service gives each message a role, system, user or assistant, and content, and requires the conversation to start with system or user and then alternate user and assistant [2]. One model provider asks for JSON Lines in its chat format, in a file of at least 10 lines [8].

System instructions sit in their own field in both cloud providers' layouts. One uses `systemInstruction`, whose text counts toward the token limit and whose `role` field is ignored [9]. The other uses a top-level `system` list [10].

Multi-turn records put several user and assistant messages in one line. One model provider's guide to fine-tuning practice adds an optional `weight` key, set to 0 or 1, to turn training off for a given assistant message [12].

```
{"messages": [{"role": "system", "content": "<instructions>"}, {"role": "user", "content": "
<question>"}, {"role": "assistant", "content": "<earlier answer>", "weight": 0}, {"role":
"user", "content": "<follow-up>"}, {"role": "assistant", "content": "<answer>", "weight":
1}]}
```

Example 2. A multi-turn chat record with per-message weights, written for this report with the fields one model provider documents [8, 12]. Each record sits on one line of the file.

Tool-calling records add two things. The assistant message carries a `tool_calls` field instead of standard content, and the tool's reply returns as a message with the role `tool` [1]. The record also carries a `tools` column, the list of available tools in a JSON schema format, which the chat template usually uses to build the system prompt [1, 4]. One model provider's example puts `tools` at the top level of the line and gives each call an `id`, a `type`, a function name and its arguments [8]. One cloud provider uses `functionCall` and `functionResponse` parts instead [9].

Some layouts add reasoning. One fine-tuning service lets assistant messages carry a `reasoning` or `reasoning_content` field that holds the chain of thought [2].

Chat records can also be cut from conversation trees. In OpenAssistant Conversations, from 2023, any path from a tree's root prompt to a node is a valid conversation, with prompter and assistant taking turns [28]. The release holds 161,443 messages in 35 languages, with 461,292 quality ratings [28].

A few dialogues can change behaviour. LIMA, a 2023 study, tuned a 65-billion-parameter model on 1,000 curated examples, and that model failed to follow the prompt within 3 interactions in 6 of 10 conversations [13]. Adding 30 multi-turn dialogue chains, for 1,030 examples in all, raised the share of excellent responses

from 45.2% to 76.1% and cut failures from 15 in 42 turns to 1 in 46 [13].

4. Chat templates turn messages into tokens

Before training, messages become tokens. One model library's guide explains that the list of role and content entries is "converted to a token sequence, often with control tokens" that mark the user, the assistant and the end of a message [3]. Templates differ between models, even between models fine-tuned from the same base model. The guide renders one conversation with `[INST]` markers for one model and <|user|> and <|assistant|> markers for another, and warns that "with the wrong control tokens, these models would have drastically worse performance" [3].

The guide's training advice is to apply the chat template as a preprocessing step, with `add_generation_prompt=False`, because the tokens that prompt an assistant reply do not help during training [3]. Templates already include the special tokens, so text formatted by a template is tokenized with `add_special_tokens=False`, to avoid duplicated tokens that hurt performance [3]. The end-of-sequence token has to match the template so that replies stop correctly [4]. LIMA added a separate end-of-turn token to each utterance to halt generation [13].

Trainers can apply the template themselves. One library does so automatically for conversational datasets [4], and one fine-tuning service formats records into the model's chat template where one is defined, noting that instruction-tuned models always have one [2].

Records should look like production prompts. One cloud provider's guide asks that examples match expected production traffic in formatting, keywords and instructions. If training records carry a "question:" and a "context:", production prompts need both, in the same order [9]. One model provider asks that all training examples share the format expected at inference, and suggests including the instructions and prompts that "worked best for the model prior to fine-tuning" in every example, especially with fewer than 100 examples [12].

Formatting moves scores at inference. A 2023 study found differences of up to 76 accuracy points from small changes in prompt formatting in few-shot use of a 13-billion-parameter open model, and the sensitivity remained after instruction tuning [29]. That study tested prompts, not fine-tuning records.

5. Which tokens the loss counts

Supervised fine-tuning in one open-source library uses token-level cross-entropy, with padding tokens masked out through an ignore index, -100 by default [4]. Table 2 lists the settings that decide which tokens count.

Setting	Tokens trained	Default	Source
Completion only	The completion of a prompt-completion record	On in one library and one service	[2, 4]

Setting	Tokens trained	Default	Source
Full sequence	Every token of a language modeling record	On in one library	[4]
Assistant only	The assistant messages of a chat record	Off in one library, on in one service	[2, 4]
Message weight 0 or 1	Messages with weight 1	Set per message	[2, 12]
Instruction modelling	The instruction and prompt as well as the output	A research method	[5]

Table 2. Loss settings in the documentation and study read for this report.

In the library, `completion_only_loss=False` trains a prompt-completion record on its full sequence, and `assistant_only_loss=True` needs a chat template that marks the assistant's text with `{% generation %}` and `{% endgeneration %}` [4]. The service offers `train_on_inputs=True` to include the other messages, and a weight per record [2].

One study tested the opposite choice. Instruction Modelling, from 2024, applies the loss to the instruction and prompt as well as the output, and was tested across 21 benchmarks [5]. On one of them it raised performance "by over 100%" in "the most advantageous case" [5]. Two factors decided the effect: the ratio of instruction length to output length, and the number of training examples. It helped especially with long instructions and short outputs, or with few examples [5]. Its authors write that they are not proposing it as a replacement for current fine-tuning processes [5].

6. Preference records: a prompt and two replies

DPO trains on "a static dataset of comparisons", each a prompt with a preferred and a dispreferred completion, using "a simple classification loss" and no sampling from the model during fine-tuning [21]. One open-source library stores the comparison as `prompt`, `chosen` and `rejected`, allows records with only `chosen` and `rejected`, the prompt implicit, and recommends "using explicit prompts whenever possible" [1]. Its reward modeling trainer is the exception, with implicit prompts recommended [1].

One model provider's preference format holds an `input` with the messages, and optionally tools, plus a `preferred_output` and a `non_preferred_output` [23]. It trains on one-turn conversations only, with both outputs as the last assistant message, and on text inputs and outputs only [23]. It reports that supervised fine-tuning on the preferred replies, or a subset, before a preference job can significantly enhance alignment and performance [23].

Rankings yield many pairs. One model developer's 2022 study asked labelers to rank between 4 and 9 replies per prompt, which gives K choose 2 comparisons for a ranking of K replies: 6 pairs for 4 replies and 36 for 9 (derived) [6]. Comparisons from one ranking are "very correlated", and shuffling them into one dataset made

the reward model overfit in a single pass, so the study trained on all comparisons from a prompt as one batch element [6].

Ratings yield pairs by a rule. UltraFeedback, from 2023, sampled four replies per instruction from a pool of 17 models, and another model scored each from 1 to 5 on instruction-following, truthfulness, honesty and helpfulness, with a written critique [30]. A 2023 distillation study built pairs from UltraFeedback's 64,000 prompts, with the response with "the highest mean score" as chosen and one of the remaining three, picked at random, as rejected [7]. It picked at random instead of taking the lowest-scored reply "to encourage diversity and make the DPO objective more challenging" [7].

Some datasets rate several attributes. HelpSteer2, from 2024, rated each response for helpfulness, correctness, coherence, complexity and verbosity on a five-point scale, with at least three annotators per response [31]. Cohen's kappa for helpfulness reached 0.706 [31]. Responses whose annotators differed by more than 2 points on helpfulness were removed, about 10% of samples [31]. The final set holds 21,362 samples, 10,681 prompts with two responses each, split 95% for training and 5% for validation [31].

Raters disagree. In the 2022 study, training labelers agreed with each other 72.6 ± 1.5% of the time and held-out labelers 77.3 ± 1.3%, against 73 ± 4% between researchers in earlier summarization work [6]. One model provider's guide adds that when several people create the training data, model performance is likely limited by their agreement. At 70% agreement on extracted snippets, the model would likely do no better [12].

7. Unpaired feedback records: one reply and a label

KTO, from 2024, needs only "a binary signal of whether an output is desirable or undesirable for an input", data its authors call more abundant, cheaper and faster to collect [22]. One open-source library stores such a record as `prompt`, `completion` and a true or false `label` [1]. In the paper's experiments, KTO matched or exceeded DPO at scales from 1 to 30 billion parameters, and the authors compared the two by breaking n preferences meant for DPO into 2n examples for KTO [22]. KTO also matched DPO while using up to 90% fewer desirable examples [22].

The two methods differ on the step before. With a good enough pretrained model, the authors report, supervised fine-tuning can be skipped before KTO without a loss in generation quality, "whereas SFT is always needed for best results with DPO" [22].

8. Files: JSON Lines, encoding, length and checks

JSON Lines has three rules: UTF-8 encoding with no byte order mark, one valid JSON value per line, with a blank line not counting as one, and `\n` as the line terminator, with `\r\n` also supported [11]. Files may use the `.jsonl` extension, and a terminator after the last value is "strongly recommended but not required" [11]. The JSON standard underneath requires UTF-8 between systems outside a closed ecosystem and forbids a byte order mark on networked JSON text [24]. When names within an object repeat, "the behavior of software that receives such an object is unpredictable" [24].

Checks run in two places. One fine-tuning service checks on the client that a file is UTF-8, that each non-empty line parses as JSON, that the line count exceeds the minimum and that the file is under the maximum size [2]. Full schema validation, of conversation roles, tool calls and other requirements, runs on its server after upload, and a file that fails is marked `INVALID_FORMAT` [2].

Length limits depend on the base model. One cloud provider lists per-record maximums from 4,096 tokens at a batch size of 1 to 32,000 tokens across its base models, with 2,048 tokens for two models at larger batch sizes, and record counts such as a minimum of 32 or 100 and a maximum of 10,000 training records [10].

A library default can cut records. One open-source library's `max_length` defaults to 1,024 tokens, longer sequences are truncated, and `truncation_mode` defaults to keeping the start [4]. Read with those defaults, a chat record over 1,024 tokens loses its end, where the final assistant reply sits (derived) [4].

Packing joins short records into fixed-length blocks to reduce padding. It is off by default in one library [4] and on by default in one service [2]. A study first posted in 2021 and rewritten in 2022 found that up to 50% of all tokens in common language datasets can be padding, and showed a packing method that avoids cross-contamination between records in self-attention [32].

Held-out data comes from the same collection. One cloud provider strongly recommends a validation dataset to measure a tuning job [9]. One model provider advises splitting the examples into training and test portions, using the test set for evaluations, and keeping both with roughly the same diversity of user inputs and model responses [8, 12].

9. Size and consistency

Provider guidance gives starting counts. One model provider sets a minimum of 10 examples, sees improvements from 50 to 100, recommends starting with 50 well-crafted demonstrations and expects a similar gain each time the example count doubles [8, 12]. One cloud provider recommends starting with 100 examples and scaling to thousands if needed, and calls quality "far more important than the quantity" [9]. The model provider adds that a smaller amount of high-quality data is generally more effective than a larger amount of low-quality data [12].

LIMA's 1,000 training sequences held about 750,000 tokens [13]. They were 750 questions and answers selected from community forums and 250 examples written by hand for task diversity and a uniform response style [13]. Its authors report that this consistent format generally improved performance in preliminary experiments [13].

Output faults can be traced to records. In the 2023 distillation study, supervised data was filtered from 1.47 million dialogues to about 200,000 examples [7]. A model trained without filtering tended to use incorrect capitalization and to preface answers with "I don't have personal experiences", and the authors applied truecasing heuristics to about 5% of the dataset [7]. One model provider's guide advises checking whether grammar, logic or style problems in a model's outputs also appear in its training data [12].

10. Contamination, safety and licenses travel with the records

Evaluation items can leak into training records. Tülu 3 used 8-gram matching [15]. A test instance counts as overlapping a training instance when more than 50% of its tokens have 8-gram matches with it, and a training set counts as contaminated when its instances overlap more than 2% of the instances in any evaluation [15]. Because completions in training sets are often regenerated with language models, it computed overlap on prompts only, or on user turns in multi-turn dialogue [15].

String matching has gaps. A 2023 study found that simple variations of test data, such as paraphrasing or translation, pass string-matching decontamination, that a 13-billion-parameter model can overfit a test benchmark this way, and that synthetic datasets generated by hosted models contain such contamination [16].

Fine-tuning can undo safety behaviour. A 2023 study broke one hosted model's safety guardrails by fine-tuning it on only 10 adversarially designed examples, at a cost of less than $0.20 [14]. Fine-tuning on benign, commonly used datasets also degraded safety alignment, to a lesser extent [14]. The benign sets it used include the 52,000-example instruction dataset of finding 2 [14].

Licenses travel with the records. The Stanford dataset allows only non-commercial use [25]. An audit of over 1,800 text datasets found license omission above 70% and error rates above 50% on widely used dataset hosting sites [33].

Implications

For teams preparing data. Pick the record type from the training method (Table 1), and keep a master copy that converts to each service's field names [1, 2, 9, 10]. Convert types before applying the chat template [1]. Train with the template the model uses at inference, add special tokens once and match the end-of-sequence token to the template [3, 4]. Write down which tokens are trained, since defaults differ between tools [2, 4].

For file checks. Check that each file is UTF-8 with no byte order mark, that every line parses, that keys are unique and that roles start and alternate as the service requires [2, 11, 24]. Count tokens per record against the base model's limit and the trainer's truncation setting [4, 10]. Hold out evaluation data before tuning, and decontaminate prompts by a stated rule, knowing that paraphrases can pass it [15, 16].

For preference data. Record how each pair was built, from a ranking or from ratings by a stated rule, how rater disagreement was handled, and check every chosen reply before splitting pairs into unpaired records [1, 6, 7, 31].

For buyers. Ask what record type, template and loss setting a delivery assumes, which license applies to each source [25, 33], how test items were kept out [15], and how safety behaviour was checked after tuning [14].

Custom datasets. When Rillor builds a custom dataset, the specification names the file formats and record layout your training and evaluation pipelines read. Each custom dataset comes with its schema, records with resolved identities, source provenance, lineage, version history, quality notes and stated limits.

Limits of this analysis

- This is an explainer of documented layouts and published studies. Documentation pages change, and each is cited with its viewed date, 8 November 2026, or 10 November 2026 for [9].
- No standards body defines a fine-tuning record format. The layouts here are those of the documentation read.
- No source read measures the effect of a chat template mismatch after fine-tuning. The 76-point figure comes from prompt formatting at inference [29].
- No source gives a general rule for how many records a job needs. The counts here are provider guidance and single studies [8, 9, 13].
- No source shows that masking the prompt is always better. One library's and one service's defaults mask it [2, 4], and one study finds that loss on the instruction helps in some cases [5]. This report gives no verdict.
- No source shows that one rule for building pairs from ratings is generally better [7], or gives a general threshold for rater agreement on preference records.
- KTO's comparison with DPO is the KTO paper's own result, at 1 to 30 billion parameters [22].
- One model provider's preference format takes one-turn conversations only [23]. One fine-tuning service takes a context in conversational format as the preference input, with two messages in its example, and exactly one assistant message in each output [2].

Sources

1. Dataset formats and types. Hugging Face, TRL documentation (v1.17.1), viewed 8 November 2026. https://huggingface.co/docs/trl/dataset_formats

2. Data preparation. Together AI, fine-tuning documentation, viewed 8 November 2026. https://docs.together.ai/docs/fine-tuning/data-preparation

3. Chat templates. Hugging Face, Transformers documentation (v5.22.2), viewed 8 November 2026. https://huggingface.co/docs/transformers/chat_templating

4. SFT Trainer. Hugging Face, TRL documentation (v1.17.1), viewed 8 November 2026. https://huggingface.co/docs/trl/sft_trainer

5. Instruction Tuning With Loss Over Instructions. Zhengyan Shi, Adam X. Yang, Bin Wu and others, NeurIPS 2024, arXiv 2405.14394, version 2, 2 October 2024, viewed 8 November 2026. https://arxiv.org/abs/2405.14394

6. Training language models to follow instructions with human feedback. Long Ouyang, Jeff Wu, Xu Jiang and others, arXiv 2203.02155, 4 March 2022, viewed 8 November 2026. https://arxiv.org/abs/2203.02155

7. Zephyr: Direct Distillation of LM Alignment. Lewis Tunstall, Edward Beeching, Nathan Lambert and others, arXiv 2310.16944, 25 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.16944

8. Supervised fine-tuning. OpenAI, API documentation, viewed 8 November 2026. https://developers.openai.com/api/docs/guides/supervised-fine-tuning

9. Prepare supervised fine-tuning data for Gemini models. Google Cloud documentation, last updated 9 November 2026, viewed 10 November 2026. https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini-supervised-tuning-prepare

10. Prepare data for fine-tuning your models. Amazon Web Services, Amazon Bedrock User Guide, viewed 8 November 2026. https://docs.aws.amazon.com/bedrock/latest/userguide/model-customization-prepare.html

11. JSON Lines. jsonlines.org, viewed 8 November 2026. https://jsonlines.org/

12. Fine-tuning best practices. OpenAI, API documentation, viewed 8 November 2026. https://developers.openai.com/api/docs/guides/fine-tuning-best-practices

13. LIMA: Less Is More for Alignment. Chunting Zhou, Pengfei Liu, Puxin Xu and others, arXiv 2305.11206, 18 May 2023, viewed 8 November 2026. https://arxiv.org/abs/2305.11206

14. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To! Xiangyu Qi, Yi Zeng, Tinghao Xie and others, arXiv 2310.03693, 5 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.03693

15. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. Nathan Lambert, Jacob Morrison, Valentina Pyatkin and others, arXiv 2411.15124, version 5, 14 April 2025, viewed 8 November 2026. https://arxiv.org/abs/2411.15124

16. Rethinking Benchmark and Contamination for Language Models with Rephrased Samples. Shuo Yang, Wei-Lin Chiang, Lianmin Zheng and others, arXiv 2311.04850, version 2, 11 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2311.04850

17. Finetuned Language Models Are Zero-Shot Learners. Jason Wei, Maarten Bosma, Vincent Y. Zhao and others, arXiv 2109.01652, version 5, 8 February 2022, viewed 8 November 2026. https://arxiv.org/abs/2109.01652

18. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi and others, EMNLP 2022, arXiv 2204.07705, version 3, 24 October 2022, viewed 8 November 2026. https://arxiv.org/abs/2204.07705

19. Scaling Instruction-Finetuned Language Models. Hyung Won Chung, Le Hou, Shayne Longpre and others, arXiv 2210.11416, version 5, 6 December 2022, viewed 8 November 2026. https://arxiv.org/abs/2210.11416

20. Self-Instruct: Aligning Language Models with Self-Generated Instructions. Yizhong Wang, Yeganeh Kordi, Swaroop Mishra and others, ACL 2023, arXiv 2212.10560, version 2, 25 May 2023, viewed 8 November 2026. https://arxiv.org/abs/2212.10560

21. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. Rafael Rafailov, Archit Sharma, Eric Mitchell and others, arXiv 2305.18290, version 3, 29 July 2024, viewed 8 November 2026. https://arxiv.org/abs/2305.18290

22. KTO: Model Alignment as Prospect Theoretic Optimization. Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff and others, ICML 2024, arXiv 2402.01306, version 5, 8 September 2026, viewed 8 November 2026. https://arxiv.org/abs/2402.01306

23. Direct preference optimization. OpenAI, API documentation, viewed 8 November 2026. https://developers.openai.com/api/docs/guides/direct-preference-optimization

24. RFC 8259, The JavaScript Object Notation (JSON) Data Interchange Format. T. Bray (editor), Internet Engineering Task Force, December 2017, viewed 8 November 2026. https://www.rfc-editor.org/rfc/rfc8259

25. Stanford Alpaca: An Instruction-following LLaMA model. Rohan Taori, Ishaan Gulrajani, Tianyi Zhang and others, GitHub repository tatsu-lab/stanford_alpaca, 2023, viewed 8 November 2026. https://github.com/tatsu-lab/stanford_alpaca

26. The Flan Collection: Designing Data and Methods for Effective Instruction Tuning. Shayne Longpre, Le Hou, Tu Vu and others, arXiv 2301.13688, version 2, 14 February 2023, viewed 8 November 2026. https://arxiv.org/abs/2301.13688

27. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. Yizhong Wang, Hamish Ivison, Pradeep Dasigi and others, NeurIPS 2023 Datasets and Benchmarks, arXiv 2306.04751, version 2, 30 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2306.04751

28. OpenAssistant Conversations: Democratizing Large Language Model Alignment. Andreas Köpf, Yannic Kilcher, Dimitri von Rütte and others, NeurIPS 2023 Datasets and Benchmarks, arXiv 2304.07327, version 2, 31 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2304.07327

29. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. Melanie Sclar, Yejin Choi, Yulia Tsvetkov and Alane Suhr, ICLR 2024, arXiv 2310.11324, version 2, 1 July

2024, viewed 8 November 2026. https://arxiv.org/abs/2310.11324

30. UltraFeedback: Boosting Language Models with Scaled AI Feedback. Ganqu Cui, Lifan Yuan, Ning Ding and others, ICML 2024, arXiv 2310.01377, version 2, 16 July 2024, viewed 8 November 2026. https://arxiv.org/abs/2310.01377

31. HelpSteer2: Open-source dataset for training top-performing reward models. Zhilin Wang, Yi Dong, Olivier Delalleau and others, arXiv 2406.08673, 12 June 2024, viewed 8 November 2026. https://arxiv.org/abs/2406.08673

32. Efficient Sequence Packing without Cross-contamination: Accelerating Large Language Models without Impacting Performance. Mario Michael Krell, Matej Kosec, Sergio P. Perez and Andrew Fitzgibbon, arXiv 2107.02027, version 2, 5 October 2022, viewed 8 November 2026. https://arxiv.org/abs/2107.02027

33. The Data Provenance Initiative: A Large Scale Audit of Dataset Licensing & Attribution in AI. Shayne Longpre, Robert Mahari, Anthony Chen and others, arXiv 2310.16787, version 3, 4 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.16787

About this report

Published by Rillor on 11 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Fine-tuning data formats: instruction, chat and preference records. 11 November 2026. https://rillor.com/insights/fine-tuning-data-formats

Work with Rillor

Send the question the data should answer, the sources you know of, the coverage you need and how often it should update. Contact us at rillor.com/contact?type=custom-data or contact@rillor.com. More on data services: rillor.com/data