# Quality Matters: Evaluating Synthetic Data for Tool-Using LLMs

Shadi Iskander, Nachshon Cohen, Zohar Karnin, Ori Shapira, and Sofia Tolmach.

Presented by Abdullah Mamun

**19 Nov 2025**

# Quality Matters: Evaluating Synthetic Data for Tool-Using LLMs

**Shadi Iskander**[*]
Amazon
shadisk@amazon.com

**Nachshon Cohen**
Amazon
nachshon@amazon.com

**Zohar Karnin**
Technology Innovation Institute
zohar.karnin@tii.ae

**Ori Shapira**
OriginAI
obspp18@gmail.com

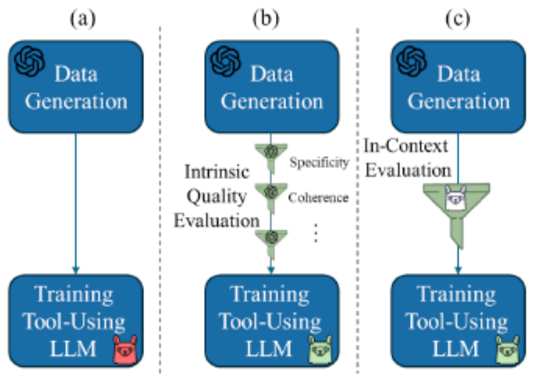
**Sofia Tolmach**
Amazon
sofiato@amazon.com

**Published in EMNLP 2024**

**12 citations to date**

## Abstract

Training large language models (LLMs) for external tool usage is a rapidly expanding field, with recent research focusing on generating synthetic data to address the shortage of available data. However, the absence of systematic data quality checks poses complications for properly training and testing models. To that end, we propose two approaches for assessing the reliability of data for training LLMs to use



## The Problem: The Hidden Cost of Bad Data

Large Language Models (LLMs) that use external tools are becoming increasingly powerful. However, their training often relies on large, synthetically generated datasets that are not checked for quality. This leads to several issues:

- Errors in training data lead to poor model performance.
- It's difficult to diagnose model failures without understanding data quality.
- Valuable resources are wasted training models on flawed or erroneous data.

**The core hypothesis: Quality over quantity.**

| **Synthetic Instruction** | **Error Type** |
|---|---|
| I'm curious about **a famous actor's** career. Can you provide details about their filmography, including their best-known titles and streaming availability on Netflix, Hulu, and Prime Video? Also, share some interesting facts about the actor. | Low Specificity |
| As a language enthusiast, I'm always eager to learn new languages. Can you help me explore the possible translations between Russian, Japanese, and Arabic? **Additionally, I would like to obtain a list of available language codes for future reference.** | Low Coherence |
| I need to **create** a temporary email address with the domain 'example.com'. Once created, I want to fetch the latest message from this email address.<br>Given APIs: [Get list of domains for email, Get message by message ID] | Unsolvable |

Table 2: Examples of synthesized instructions, **highlighted** with errors involving our defined properties.

| **Synthetic Instruction** | **API-Call within Sequence** | **Error Type** |
|---|---|---|
| Can you create a shield logo for my friend's blog? The name of the blog is 'The Creative Mind'. | `generate_shield(name=None)` | Missing Parameter |
| I need to fetch the current weather conditions for a specific location. Can you help me by providing the address and geocoordinates of the location? | `geocode(address="San Francisco")`<br>... | Hallucinated Parameter |

Table 3: Examples of synthesized API-call sequences for respective instructions, with incorrect parameters.

## 3 Task Setup

Tool-using LLMs are expected to behave as follows. Given a set of tools $T = \{t_1, ..., t_n\}$, represented as API functions, and an instruction query $q$, a model is required to plan a call sequence $S = (t'_1, ..., t'_k)$, based on $T$, that would obtain information, or perform actions, needed to address $q$. Based on the responses obtained after performing the call sequence (using an external API invoker), the model then generates a final response $r$ that responds to $q$. The primary method for model evaluation is based on calculating the *pass rate*, which measures the proportion of instances that successfully addressed their instructions, i.e., a predicted $r$ responded to $q$ adequately (explained further in §6).

### 4.1.1 Instruction Properties

In our setting, an instruction is a free-form text of one-to-a-few sentences that describes a user requirement. An instruction can contain more than one request, likely implying the need for several tool invocations. The following properties in the instruction demand validation (examples in Table 2):

**Specificity.** All the required details are present in the instruction for the LLM to be able to fulfill the user requests.

**Coherence.** The requests within the instruction are logically related, and the order of requests makes sense for a real-world use case.

**Solvability.** The requests within the instruction can be addressed by the given API tools.

| Characteristic | ToolBench | ToolAlpaca |
|---|---|---|
| API source | real-world | synthesized w/GPT |
| # available APIs | 16K | 2.3K |
| # of training instances | 125K | 4.2K |
| # required API calls per instance | 1-5 | 1-2 |

Table 1: Summary of relevant dataset characteristics.

|  |  | ToolBench Dataset |  |  |  | ToolAlpaca Dataset |  |  |  |
|---|---|---|---|---|---|---|---|---|---|
|  | **Quality Criterion** | **Accuracy** | **Prec.** | **Rec.** | **F1** | **Accuracy** | **Prec.** | **Rec.** | **F1** |
| Instruction | Specificity | 0.74 | 0.70 | 0.84 | 0.76 | 0.88 | 0.75 | 0.86 | 0.80 |
| Instruction | Coherence | 0.82 | 0.62 | 0.77 | 0.69 | 0.98 | 0.50 | 1.00 | 0.66 |
| Instruction | Solvability | 0.90 | 0.70 | 0.78 | 0.74 | 0.92 | 0.75 | 0.50 | 0.60 |
| Instruction | Instruction Correctness | 0.72 | 0.72 | 0.90 | 0.80 | 0.86 | 0.80 | 0.84 | 0.82 |
| API Call Seq. | Parameter Alignment | 0.70 | 0.63 | 0.92 | 0.74 | 0.76 | 0.74 | 0.80 | 0.77 |
| API Call Seq. | Sufficiency | 0.78 | 0.64 | 0.60 | 0.62 | 0.88 | 0.80 | 0.50 | 0.62 |
| API Call Seq. | Minimality | 0.76 | 0.95 | 0.63 | 0.76 | 0.86 | 0.88 | 0.57 | 0.70 |
| API Call Seq. | Sequence Correctness | 0.82 | 0.83 | 0.94 | 0.88 | 0.76 | 0.70 | 0.85 | 0.80 |
|  | Overall Correctness | 0.86 | 0.89 | 0.95 | 0.92 | 0.76 | 0.74 | 0.90 | 0.81 |

Table 4: Validation results of the automated metrics for each criterion, in the ToolBench and ToolAlpaca datasets. Coarse-grained correctness considers combined correctness over specific criteria. Note that precision, recall and F1 are measured w.r.t. a label that is positive when an error occurs, so e.g., recall means the amount of errors caught.

#### 4.3.1 Evaluation of Automated Metrics

Using the manually annotated data (described in §4.2), we conduct an assessment of the automatic metrics proposed. For each of the ToolBench and ToolAlpaca datasets, the 50 annotated instances are compared against the automatically produced values, producing measures of accuracy (agreement), precision, recall and F1 score. We treat instances marked as incorrect instances as positive labels, since we aim to identify and filter erroneous instances.

## The "Less is More" Principle in Action

This research demonstrates a crucial finding: fine-tuning a tool-using LLM on a small, high-quality dataset can achieve better or comparable performance to training on a much larger, unvalidated dataset.

## Two Approaches to Measuring Data Quality

To prove the hypothesis, the study introduces two distinct methods for evaluating the quality of training data instances.

### 1. Intrinsic Quality Evaluation

This method uses an external LLM (like ChatGPT) to assess data against various human-defined criteria for correctness and logic. It's about what makes an instance 'good' from a human perspective.

### 2. In-Context Evaluation (ICE)

This method measures the 'educational value' of a data instance. It tests how helpful an instance is as a one-shot example for a target LLM performing a task, predicting its usefulness for training.

| Dataset | Instruction | | | | API-Call Sequence | | | | Inst. & Seq. Overall |
|---|---|---|---|---|---|---|---|---|---|
| | Specificity | Coherence | Solvable | Overall | Param. Alignment | Sufficiency | Minimality | Overall | |
| **ToolBench** | 20.4% | 22.1% | 18.2% | 47.3% | 47.9% | 33.6% | 45.1% | 74.4% | 84.0% |
| **ToolAlpaca** | 17.5% | 4.1% | 12.7% | 27.2% | 33.1% | 13.6% | 15.9% | 35.5% | 44.8% |

Table 5: Percentage of instances containing errors in each dimension, according to our automated methods, in the train sets of the examined datasets. This analysis is done on 125K examples in ToolBench and 4.2K in ToolAlpaca.

# Defining "Good Data": Intrinsic Quality Criteria

## Instruction Properties

### Specificity

Does the instruction contain all the necessary details for the LLM to fulfill the request?

### Coherence

Are the requests within the instruction logically related and make sense in a real-world context?

### Solvability

Can the requests actually be addressed by the given set of API tools?

## API-Call Sequence Properties

1. **Parameter Alignment**

   Are parameter values correctly extracted from the instruction without missing or hallucinated values?

2. **Sufficiency**

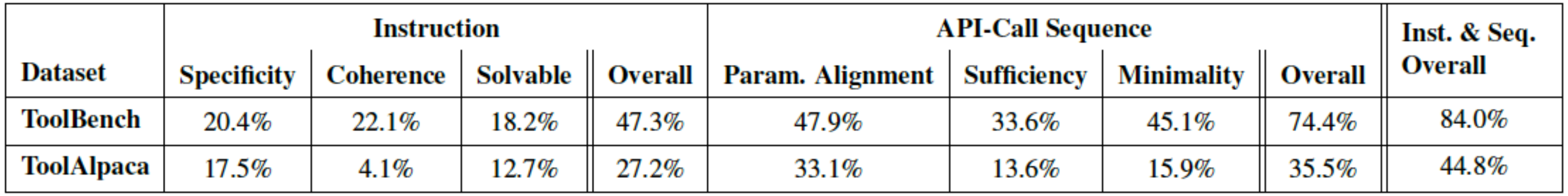
   Does the API-call sequence cover all actions required by the instruction?

3. **Minimality**

   Does the sequence use the minimum number of API calls necessary, with no redundant calls?

# The Quality Gap: ToolBench vs. ToolAlpaca

Automated metrics revealed significant quality differences between two prominent datasets. ToolBench, which uses real-world APIs, has a much higher percentage of errors across most criteria compared to ToolAlpaca, which uses cleaner, synthesized APIs.



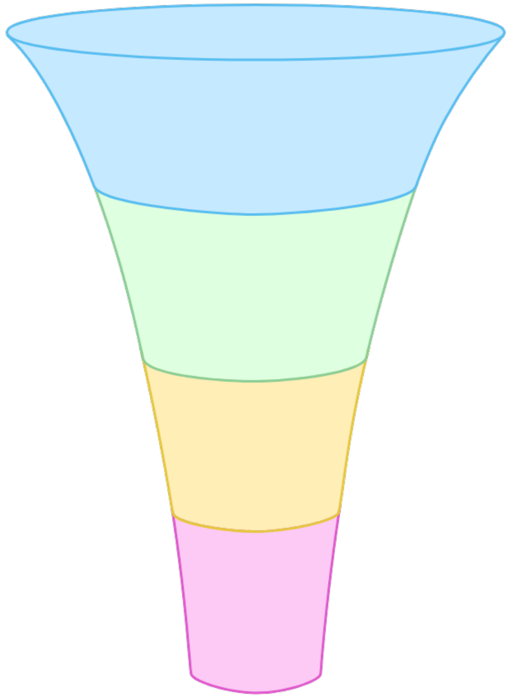
|  | Instruction |  |  |  | API-Call Sequence |  |  |  | Inst. & Seq. Overall |
|---|---|---|---|---|---|---|---|---|---|
| Dataset | Specificity | Coherence | Solvable | Overall | Param. Alignment | Sufficiency | Minimality | Overall |  |
| ToolBench | 20.4% | 22.1% | 18.2% | 47.3% | 47.9% | 33.6% | 45.1% | 74.4% | 84.0% |
| ToolAlpaca | 17.5% | 4.1% | 12.7% | 27.2% | 33.1% | 13.6% | 15.9% | 35.5% | 44.8% |

Table 5: Percentage of instances containing errors in each dimension, according to our automated methods, in the train sets of the examined datasets. This analysis is done on 125K examples in ToolBench and 4.2K in ToolAlpaca.

Notably, over 33% of instances in both datasets had parameter alignment errors, meaning the models learn to identify parameters incorrectly in a large portion of cases.

# Alternative Method: In-Context Evaluation (ICE)

ICE offers a different lens for data quality, focusing on an instance's 'educational value' rather than human-defined correctness. The process measures how much a single training example improves a model's performance on a related test task.



Prepare a small, hand-crafted test set (APIs and queries).

Use one training data instance (x) as a one-shot example.

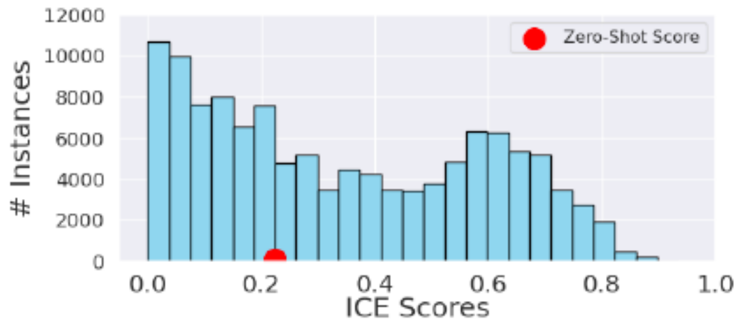
Prompt the target LLM to solve the test queries using the example.

The model's performance on the test set becomes the ICE score for instance x.

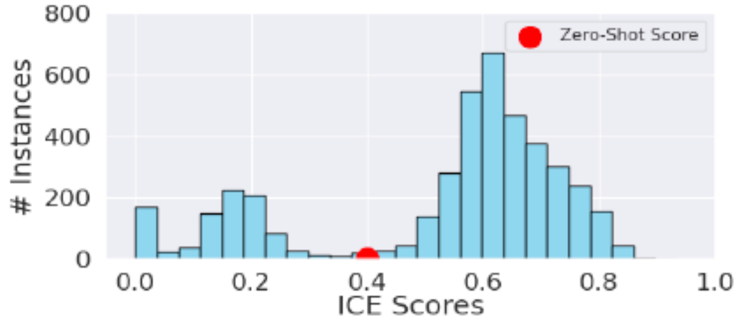
# ICE Score Analysis

## Score Distribution

The ICE scores revealed a clear difference between datasets. ToolAlpaca showed a bimodal distribution, suggesting distinct groups of high and low-quality examples. In contrast, most instances in ToolBench received low ICE scores, aligning with the intrinsic analysis that its overall quality is lower.



(a) ToolBench

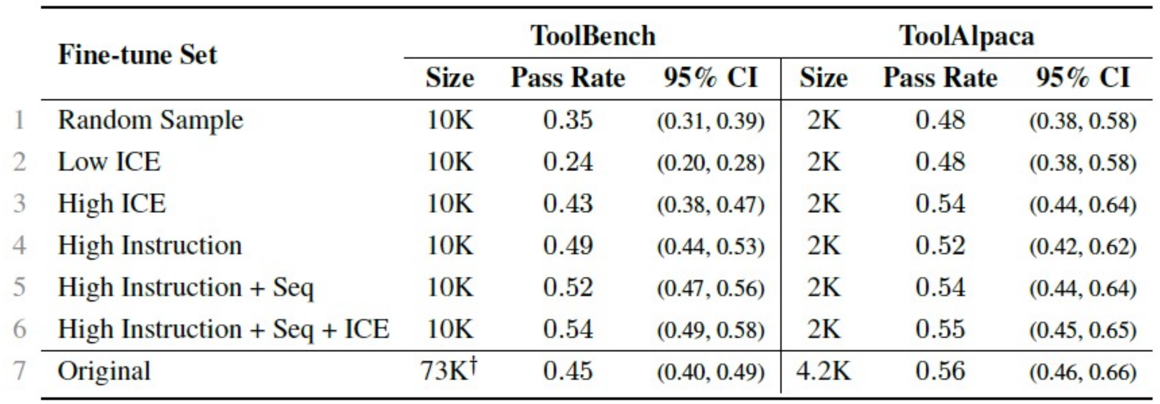


(b) ToolAlpaca

Figure 2: Distribution of ICE scores. Most instances in ToolAlpaca are beneficial as the one-shot in-context example. ToolBench instances are not as effective.

## The Final Showdown: Extrinsic Evaluation

To test the main claim, models were fine-tuned on various subsets of the ToolBench and ToolAlpaca datasets and then evaluated on a cleaned, high-quality test set. The results compare performance based on the quality of the training data.



| | Fine-tune Set | ToolBench | | | ToolAlpaca | | |
|---|---|---|---|---|---|---|---|
| | | Size | Pass Rate | 95% CI | Size | Pass Rate | 95% CI |
| 1 | Random Sample | 10K | 0.35 | (0.31, 0.39) | 2K | 0.48 | (0.38, 0.58) |
| 2 | Low ICE | 10K | 0.24 | (0.20, 0.28) | 2K | 0.48 | (0.38, 0.58) |
| 3 | High ICE | 10K | 0.43 | (0.38, 0.47) | 2K | 0.54 | (0.44, 0.64) |
| 4 | High Instruction | 10K | 0.49 | (0.44, 0.53) | 2K | 0.52 | (0.42, 0.62) |
| 5 | High Instruction + Seq | 10K | 0.52 | (0.47, 0.56) | 2K | 0.54 | (0.44, 0.64) |
| 6 | High Instruction + Seq + ICE | 10K | 0.54 | (0.49, 0.58) | 2K | 0.55 | (0.45, 0.65) |
| 7 | Original | 73K† | 0.45 | (0.40, 0.49) | 4.2K | 0.56 | (0.46, 0.66) |

Table 6: Extrinsic evaluation results with confidence intervals, and the size of the training sets. By filtering out low-quality training instances, the models perform significantly better than (in ToolBench) or as good as (in ToolAlpaca) the original models that use a much larger unvalidated training set. † Although there are 125K instances in the released dataset, the model published in the original paper was trained on a subset of 73K instances.

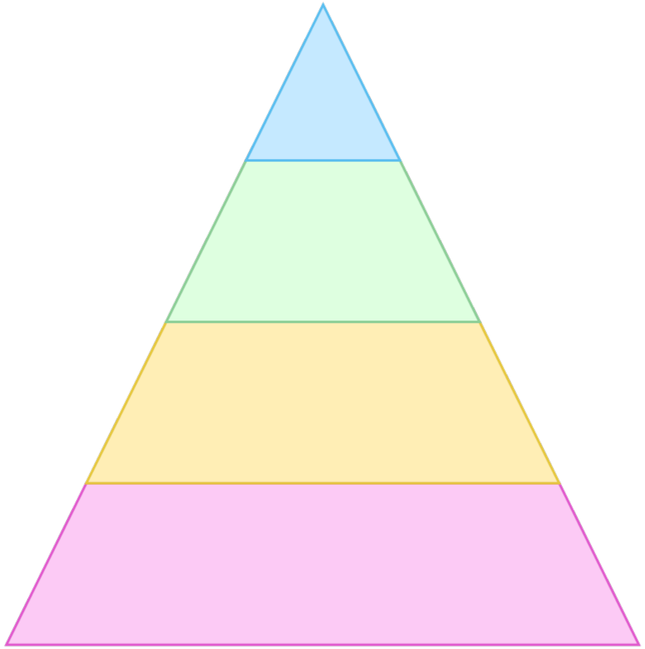
## Key Result: Quality Trumps Quantity

The results are clear. For ToolBench, training on a filtered, high-quality subset of just 10K instances significantly outperformed training on the original 73K instances.

- High-quality subsets (rows 4-6 in Table 6) consistently outperform random samples.
- Performance on small, high-quality sets is comparable or superior to the full, noisy datasets.
- Combining Intrinsic metrics and ICE for filtering yields the best results.
- Training on data with low ICE scores actively harms model performance.

**The takeaway: careful data selection is more effective than using more data.**

## Conclusion & Takeaways



It is worthwhile to carefully choose training data for tool-using LLMs.

Automatic post-hoc filtration is a great, cost-effective alternative to improving data generation methods.

Two effective evaluation approaches were introduced: explainable Intrinsic Quality metrics and computationally cheaper In-Context Evaluation (ICE).

This work proves the 'less is more' trend: smaller, high-quality datasets lead to superior or comparable model performance.