

DataArc-SynData-Toolkit: A Unified Closed-Loop Framework for Multi-Path, Multimodal, and Multilingual Data Synthesis

Zhichao Shi^{1,2,3,4*}, Cehao Yang^{1,2,5*}, Hao Zhou^{1,2*}, Xiaojun Wu^{1,2,5},
Huajie Li^{1,2}, Xuhui Jiang^{1,2†}, Chengjin Xu^{1,2†}, Yuanzhuo Wang⁴ Jian Guo²

¹DataArc Tech Ltd., ²IDEA Research, International Digital Economy Academy,

³School of Advanced Interdisciplinary Sciences, UCAS,

⁴Institute of Computing Technology, CAS,

⁵The Hong Kong University of Science and Technology (Guangzhou)

 <https://github.com/DataArcTech/DataArc-SynData-Toolkit>

Abstract

Synthetic data has emerged as a crucial solution to the data scarcity bottleneck in large language models (LLMs), particularly for specialized domains and low-resource languages. However, the broader adoption of existing synthetic data tools is severely hindered by convoluted workflows, fragmented data standards, and limited scalability across modalities. To address these limitations, we develop DataArc-SynData-Toolkit, an open-source framework featuring: (1) a configuration-driven, end-to-end pipeline equipped with an intuitive visual interface and simplified CLI for exceptional usability; (2) a unified, quality-controllable synthesis paradigm that standardizes multi-source data generation to ensure high reusability; and (3) a highly modular architecture designed for seamless multimodal, multilingual, and multi-task adaptation. We apply the toolkit in multiple application scenarios. Experimental results demonstrate that our toolkit achieves an optimal balance between generation efficiency and data quality. By offering an end-to-end and visually interactive pipeline, DataArc-SynData-Toolkit significantly lowers the technical barrier to synthetic data generation and subsequent model training, accelerating its practical deployment in real-world applications.

1 Introduction

The rapid evolution of Large Language Models (LLMs) has exponentially driven the demand for high-quality training corpora (OpenAI, 2023; Touvron et al., 2023). However, acquiring real-world data at scale remains a formidable challenge, particularly for specialized domains and low-resource languages (Barati et al., 2025; Abdalla et al., 2025). Consequently, synthetic data generation has become a critical paradigm for expanding training

corpora. While several pioneering frameworks like Synthetic Data-RL (Guo et al., 2025) and DataFlow (Liang et al., 2025) have significantly advanced the field, their primary design often leans towards algorithmic exploration rather than out-of-the-box, end-to-end engineering usability.

Specifically, as shown in Table 1, current synthetic data tools face three major challenges in practical scenarios: (1) **Steep Learning Curves and Limited Usability**. Existing tools cater primarily to AI experts. Users lacking specialized backgrounds struggle to navigate the fragmented processes of data generation, model training, and validation, often resorting to complex, multi-tool scripting. (2) **Fragmented Workflows and Unstandardized Pipelines**. There is a distinct lack of a unified paradigm for handling multi-source data (Chen et al., 2024). This fragmentation leads to unpredictable data quality and severely limits cross-project reusability. (3) **Restricted Modality and Language Scalability**. Most frameworks lack generic interfaces for multimodal and low-resource language synthesis, resulting in prohibitively high extension and maintenance costs.

To address these challenges, we propose **DataArc-SynData-Toolkit**, a configuration-driven end-to-end system for data synthesis, model training, and evaluation. The toolkit unifies data collection, synthesis, quality control, post-training, and evaluation into a single closed-loop workflow that supports continuous iteration. Users can construct complex synthetic datasets and optimize models through concise configuration files. In addition, the toolkit encapsulates execution logic into simplified commands and provides an intuitive visual interface, which significantly improves usability.

The toolkit adopts a highly modular and agent-based system design. It unifies the synthesis pipelines and data formats for multi-source synthetic data, including local corpora, open-source web datasets, and model distillation. This design

*Both authors contributed equally to this research.

†Correspondence: jiangxuhui@dataarctech.com;
xuchengjin@dataarctech.com

Framework	Github Repository	Data Prep.	Modality Support			Synthesis Strategies			Filter	Eval.	Model-Training	Vis.
			Struct.	Text	Multi.	Local	Distill	Web				
DataArc (Ours)	DataArcTech/DataArc-SynData-Toolkit	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
SDV	sdv-dev/SDV	✓	✓			✓			✓			
Gretel	gretelai/gretel-synthetics	✓	✓	✓		✓						
YData	ydataai/ydata-synthetic	✓	✓			✓			✓		✓	
EasyInstruct	zjunlp/EasyInstruct			✓		✓		✓	✓		✓	
EasyDataset	ConardLi/easy-dataset	✓	✓		✓	✓		✓	✓		✓	
EasyDistill	modelscope/easydistill	✓		✓		✓		✓	✓	✓		
DataFlow	OpenDCAI/DataFlow	✓	✓	✓		✓		✓	✓	✓	✓	
Distilabel	argilla-io/distilabel	✓	✓	✓		✓	✓	✓	✓	✓		
SyntheticDataKit	meta-llama/synthetic-data-kit	✓	✓		✓	✓		✓	✓			
Synthetic Data RL	gydpku/Data_Synthesis_RL	✓		✓		✓		✓	✓	✓		

Table 1: Feature comparison of DataArc with other open-source synthetic data frameworks. *Data Prep.* refers to Data Preprocessing; *Struct.* and *Multi.* denote Structured Data and Multimodal Data, respectively. For Synthesis Strategies, “Local” indicates synthesis based on local corpora, “Distill” refers to knowledge distillation, and “Web” denotes synthesis via web search. As shown, DataArc is the only framework offering an end-to-end pipeline covering multiple strategies, multimodal synthesis, and downstream model fine-tuning.

ensures high data quality and strong reusability. Furthermore, the system abstracts unified adaptation interfaces for multimodal and multilingual data, greatly reducing the cost of future extensions.

In summary, DataArc-SynData-Toolkit is designed with usability, standardization, and scalability at its core. Its main contributions are as follows:

- We introduce a configuration-driven, closed-loop system encompassing data synthesis, model training, and evaluation, accessible via a streamlined CLI and an intuitive UI to democratize synthetic data generation.
- We design a unified, standardized paradigm for multi-source data synthesis equipped with robust quality-control mechanisms, thereby guaranteeing data reliability and reusability.
- We deliver a highly extensible, modular framework inherently capable of multimodal and multilingual synthesis, significantly lowering the engineering overhead for future academic and industrial adaptations.

2 Framework

We aim to develop a user- and developer-friendly synthetic data toolkit. The toolkit should provide a highly extensible framework for developers, and offer users an end-to-end pipeline and an easy-to-use operational paradigm.

Therefore, DataArc-SynData-Toolkit adopts a unified and modular agent-based design. We generalize and abstract the process from multi-source synthetic data to model outputs. Based on the convergence nodes in the pipeline, and considering the mergeability of different steps, we divide the

entire end-to-end workflow into three key stages: data synthesis, data quality control, and model post-training and evaluation, as shown in Figure 1. In this section, we introduce the design of DataArc-SynData-Toolkit for implementing the key stages.

2.1 Data Synthesis

To address the lack of unified workflows and data standards in existing tools, DataArc-SynData-Toolkit abstracts a unified data synthesis paradigm, including *task parsing*, *data source preparation*, *constraint construction*, *data acquisition*, and *structured processing*. (1) *Task parsing* parses the task definitions into information like task instructions and domain boundaries. (2) *Data source preparation* selects and prepares data sources based on the configuration, including local corpora, web datasets, or models to be distilled. (3) *Constraint construction* builds constraints for data synthesis, including reference passages, field limitations, or pattern constraints. (4) *Data acquisition* generates or filters the required data from the selected sources based on the parsed information and the defined constraints. (5) *Structured processing* normalizes the synthesized data into a unified format.

Based on this unified paradigm, DataArc-SynData-Toolkit designs BaseTaskExecutor. As shown in Figure 2, BaseTaskExecutor has implemented basic execution logic. Developers can flexibly extend different synthesis paths by overriding the abstract functions. To ensure data normalization and reusability, the toolkit processes all the data into the same format in Function `structure_process`.

We have implemented three paths: local corpus-driven synthesis (`LocalTaskExecutor`),

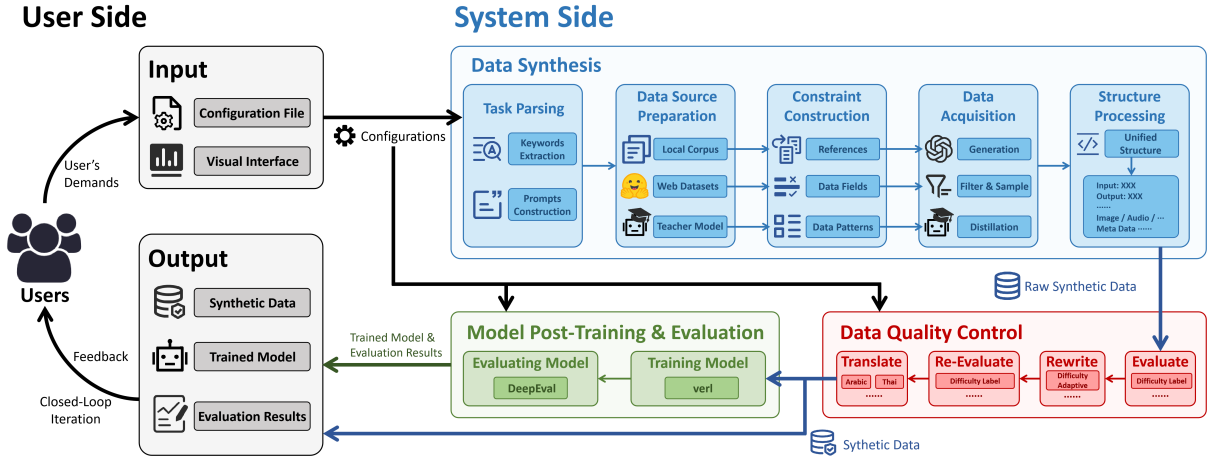


Figure 1: The overview of DataArc-SynData-Toolkit. **User side:** Users only need to set configurations in the file or the visual interface to obtain the desired synthetic data, models, and evaluation results. **System Side:** The toolkit divides the end-to-end pipeline into three stages: (1) Data synthesis: synthesize data based on the configuration; (2) Quality control: filter and rewrite the data to ensure quality; (3) Model post-training and evaluation.

web dataset-based filtering (WebTaskExecutor), and instruction distillation-based synthesis (DistillTaskExecutor), which cover the common synthesis paths in current practice.

```

from sdgsystem.tasks import BaseTaskExecutor

# Inputs: configurations, llm
# Initialization
executor = BaseTaskExecutor(configurations, llm)
# Step(1): parsing configurations
parsed_information = executor.task_parsing()
# Step(2): prepare data sources
sources = executor.prepare()
# Step(3): get constraints for synthesis
constraints = executor.construct_constraints(
    parsed_information)
# Step(4): synthesizing data from sources
raw_data = executor.data_acquisition(
    sources,
    constraints,
    parsed_information
)
# Step(5): processing data into a unified
structure
syn_data = executor.structure_process(raw_data)
# Outputs: synthetic data in format:
# {input: ..., output: ..., image: ..., audio:
..., metadata: ...}

```

Figure 2: The code implementation for data synthesis in abstract class BaseTaskExecutor

2.1.1 Local Task

This path synthesizes data based on local corpora. Function `task_parsing` extracts keywords from the task instruction and domain in configurations for retrieval. Function `prepare` loads the retriever for the local corpora and prepares the generator for synthesis. Function `construct_constraints` invokes the retriever to retrieve reference passages from local corpora using methods such as BM25 (Robertson et al., 2009) or dense methods (Karpukhin et al., 2020; Khattab and Zaharia,

2020). Function `data_acquisition` combines the task instruction, reference passages, format constraints, and optional examples to form a prompt. The LLM generator is prompted to synthesize data.

2.1.2 Web Task

This path selects required data from open-source web datasets. Function `task_parsing` is the same as Local Task. Function `prepare` searches candidate datasets from open platforms (e.g., Hugging-Face) based on the extracted keywords. Function `construct_constraints` adopts the LLM to select the data fields that best match the task based on configurations. Function `data_acquisition` extracts the contents in the selected fields, and scores the dataset by the task consistency and quality of these contents. The final data are preferentially sampled from datasets with higher scores until the required data volume is reached.

2.1.3 Distill Task

This path distills data from a stronger teacher model based on the task instruction. Function `task_parsing` parses the configurations to get parameters for the teacher model. Function `prepare` deploys the teacher model. Function `construct_constraints` queries the teacher model to extract high-quality generation patterns from task instruction and optional examples. Function `data_acquisition` prompts the teacher model with task instruction, pattern constraints, and format constraints to generate data.

2.2 Data Quality Control

Caused by factors such as inappropriate sample difficulty (Tong et al., 2024; Ferracci et al., 2024), directly training models on initial synthetic data may lead to degraded performance. Therefore, DataArc-SynData-Toolkit designs a quality control stage to ensure the data quality. This stage is divided into two key functions: evaluation and rewrite. Through evaluation, rewriting, and re-evaluation, high-quality synthetic data is generated.

To support extensibility across different algorithms, we highly abstract the evaluation and rewriting steps into base classes `Evaluator` and `BaseRewriter`. The evaluator takes raw synthetic data as input and outputs corresponding evaluation scores. The rewriter takes the raw data and their scores as input and produces rewritten samples based on these scores. Developers can override the abstract functions and design different strategies, enabling flexible extension of quality control policies. An example of the code implementation for quality control in the toolkit is provided below.

```
from sdgsystem.evaluation import Evaluator
from sdgsystem.generation.rewriter import
    DifficultyAdjustRewriter

# Inputs: configs, llm, base_model to be trained.
evaluator = Evaluator(configs, base_model, llm)
rewriter = DifficultyAdjustRewriter(configs, llm)

# evaluation
init_eval = evaluator.evaluate(dataset)
# rewrite
rewritten_dataset = rewriter.rewrite(
    dataset,
    init_eval,
)
# re-evaluation
final_eval = evaluator.evaluate(
    rewritten_dataset
)
solved, learnable, unsolved \
= rewritten_dataset.categorize_by_score(
    final_eval)
# learnable subset is selected as training data
```

Figure 3: An example of code implementation for quality control strategy in the toolkit.

2.3 Model Post-Training and Evaluation

To facilitate users in examining the synthetic data’s training effectiveness, DataArc-SynData-Toolkit integrates model post-training and evaluation modules, establishing a closed-loop model iteration.

For training, the toolkit integrates a training module powered by `verl` (Sheng et al., 2025), enabling users to train models directly on their synthesized data. The toolkit supports two training methods: Supervised Fine-tuning (SFT) and Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

For evaluation, the toolkit provides a model evaluation module powered by DeepEval¹. This module enables users to evaluate post-trained models on user-provided evaluation datasets and applies G-Eval (Liu et al., 2023) to judge model responses. The toolkit supports three metrics: Answer Correctness, Format Compliance, and Pairwise Preference.

2.4 Other Extensible Modules

To make it easy for developers to integrate various algorithms and strategies into the toolkit, we adopt a highly modular design. In the previous subsections, we described the agent-based abstraction and encapsulation of the three key stages of the workflow. In this section, we introduce other extensible modules, with a focus on multimodal and multilingual adaptation modules.

For the multimodal adapter, we developed `MMPProjector`, which executes the task of synthesizing multimodal instructions after being provided with seed image data for cold-start initialization. The base model and pipeline-driven model are switched to their multimodal versions; consequently, the final synthesized instructions include not only input and output, but also the corresponding seed images. This component can be optionally activated when users provide seed images or when processing local rich-text data.

For the multilingual adapter, we design `BaseTranslator`, which performs language conversion in the final step of quality control. The toolkit currently integrates adapters for low-resource languages such as Arabic.

In addition, we abstract the execution logic for sequential data and design `ParallelExecutor`. This executor encapsulates parallelization, temporary interruption, data caching, and resumption from existing progress. Developers can call `Function execute(...)` of `ParallelExecutor` to perform more efficient and reliable processing of sequential data, as shown in Figure 4.

2.5 Simplified Usage

DataArc-SynData-Toolkit is designed with usability as the primary goal, enabling even users without professional backgrounds to easily synthesize data. Therefore, the toolkit provides highly simplified commands for CLI and an intuitive visual interface to launch the workflow.

The toolkit offers simplified commands to start

¹<https://github.com/confident-ai/deepeval>

```

from sdgsystem.parallel import ParallelExecutor

executor = ParallelExecutor(n_workers=10)
# define the function to process each item
def process(...): ...
# Inputs: the iterable sequence data, the process
# function
outputs = executor.execute(
    iterable_inputs=data,
    process_function=process,
    ...
)
# outputs: A list like [r1, r2, ..., None, ...]
# None indicates items failed to be processed
# when the program is interrupted.

```

Figure 4: An example of ParallelExecutor.

the pipeline. We use uv² as the package manager and unify the command format. Users can complete the whole pipeline with simple commands:

```

uv run sdg generate configs/sdg.yaml
uv run sdg train configs/[sft|grpo].yaml
uv run sdg eval configs/eval.yaml

```

Figure 5: Simplified commands for quick start in CLI.

In addition, we provide a visual interface to facilitate task configuration, data management, and model management for users. The toolkit adopts a frontend-backend separated architecture, featuring a FastAPI backend and a React frontend for improved visualization and scalability. The detailed design is described in Appendix A.

The samples of data synthesized by our toolkit are detailed in Appendix B. Detailed deployment instructions, command usage, and tutorial videos for the visual interface are available in our GitHub repository³.

3 Performance Evaluation

In this section, we systematically evaluate the performance of DataArc-SynData-Toolkit, including the performance across different domains, multi-lingual, multi-modal, and its efficiency.

3.1 Benchmarks

We conduct experiments on domain-specific, multi-lingual, and multi-modal benchmarks.

- **Domain-specific benchmarks: MedQA** (Jin et al., 2021) contains multiple-choice questions collected from medical board exams in the

²<https://github.com/astral-sh/uv>

³<https://github.com/DataArcTech/DataArc-SynData-Toolkit>

US and China. We adopt the subset MedQA-USMLE-4-options⁴ of 1,273 questions. **Flare-CFA**⁵ comprises 1,032 multiple-choice questions covering CFA exam levels I and II. **LexEval** (Li et al., 2024) evaluates LLM performance in legal applications, and 500 multiple-choice questions involving multi-hop reasoning are adopted.

- **Multi-lingual benchmark: Arabic Broad Benchmark (ABB)**⁶ is a human-validated and compact benchmark of 470 free-form questions spanning 22 Arabic language tasks (Ouda, 2025).
- **Multi-modal benchmarks: FinMME** (Luo et al., 2025) contains high-quality financial research samples across 18 domains, and about 1,850 numerical questions are selected. **TableVQA** (Kim et al., 2024) contains table question-answering and table structure recognition datasets of 1,500 samples.

3.2 Implementation Details

For each benchmark, we collect seed data from HuggingFace for cold start according to the corresponding task definition. We sample query terms from the extracted keyword set and, by default, select 5 candidate relevant datasets to ensure diversity in data distribution. Subsequently, we sample and filter 25 data instances and clean them into seed data based on the task definition. For multi-modal tasks, we will collect seed images for cold-start. We then use *gpt-5-mini* to synthesize the initial instruction data and submit it to a quality management control pipeline powered by *gpt-5-mini*. The resulting curated datasets are constructed at specified scales for downstream training, namely 1,000 / 2,000 / 4,000 samples. We validate our toolkit on a wide range of LLMs, including Qwen2.5-7B-Instruct and Qwen3-4B for domain and multi-lingual tasks, and Qwen2.5-VL-7B-Instruct and Qwen3-VL-4B-Instruct for multi-modal tasks. All experiments are conducted on an Ubuntu machine equipped with eight NVIDIA H800-80G GPUs.

3.3 Evaluation Results

Consistent Performance Gains Across Models. As shown in Table 2, synthetic data generated

⁴<https://huggingface.co/datasets/GBaker/MedQA-USMLE-4-options>

⁵<https://huggingface.co/datasets/TheFinAI/flare-cfa>

⁶<https://huggingface.co/datasets/silma-ai/arabic-broad-benchmark>

Method	Data Size	Domain			Multi-lingual	Multi-modal	
		MedQA	Flare-CFA	LexEval	ABB	FinMME	MedQA
		<i>Qwen2.5-7B</i>				<i>Qwen2.5-VL-7B</i>	
Base Model	-	42.34	52.91	19.80	6.42	16.41	55.13
	1000	59.02	67.15	33.87	6.96	24.90	61.90
SYNDATA	2000	64.57	73.93	42.80	7.08	26.98	61.39
	4000	68.12	75.80	46.15	7.12	32.47	64.32
		<i>Qwen3-4B</i>				<i>Qwen3-VL-4B</i>	
Base Model	-	57.52	60.89	31.48	6.26	27.65	54.10
	1000	64.70	66.41	37.90	6.75	33.81	60.37
SYNDATA	2000	66.13	75.74	45.44	6.69	33.46	66.54
	4000	70.62	74.41	48.00	6.77	36.96	64.90

Table 2: The main experiment results.

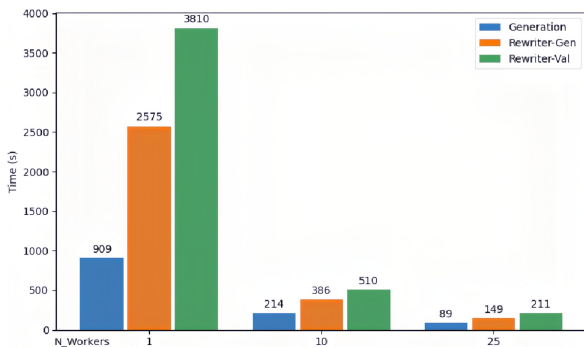


Figure 6: The efficiency of our ParallelExecutor design in toolkit when synthesizing 500 samples.

by our toolkit consistently improves performance across different models and task settings. For both Qwen2.5-7B and Qwen3-4B, training on synthesized samples yields substantial accuracy gains over the corresponding base models. For instance, Qwen2.5-7B improves from 42.34 to 68.12 on MedQA and from 19.80 to 46.15 on LexEval with 4,000 synthetic samples. Similar trends are observed for Qwen3-4B, which increases from 57.52 to 70.62 on MedQA and from 31.48 to 48.00 on LexEval. These results demonstrate that the proposed toolkit provides robust and architecture-agnostic gains, validating its effectiveness across heterogeneous model scales and modalities.

Scaling Synthetic Data Enhances Performance Gains. Larger synthetic datasets lead to more pronounced performance gains. Across nearly all benchmarks, accuracy steadily increases as the training data size expands from 1,000 to 2,000 and 4,000 samples. For example, on Qwen2.5-7B,

MedQA accuracy rises from 59.02 to 64.57 and further to 68.12, while FinMME improves from 24.90 to 32.47. The scaling behavior suggests strong potential for further gains with increased synthetic data budgets, highlighting the practicality of synthetic data as a controllable and efficient alternative to costly human annotation.

Parallel Design Significantly Boosts Synthesis Efficiency. Experimental results in Figure 6 demonstrate that adjusting the number of workers in the toolkit’s ParallelExecutor yields substantial improvements in synthesis efficiency. Without parallelization, generating merely 500 initial samples requires 909 seconds. As the number of active workers increases, the required time decreases markedly. Since all modules within the pipeline benefit from the parallel design, including both stages of the rewriter (i.e., generation and validation), scaling the worker count to just 10 or 25 achieves significant performance gains.

4 Conclusion and Future Work

We propose DataArc-SynData-Toolkit, a configuration-driven, end-to-end open-source toolkit for data synthesis. The toolkit provides a highly extensible, unified, and quality-controllable framework for multi-path, multimodal, multilingual data synthesis. For the next release, we plan to enable encrypted synthetic data generation to protect sensitive information for secure model training. Future work will focus on incorporating more algorithms, modalities, and languages to offer broader options.

Limitations

Due to the rapid evolution of synthetic data methods, our toolkit currently integrates only a subset of data synthesis and validation approaches, such as the difficulty-based data rewriting method shown in Figure 3. However, our analysis of the data synthesis pipeline enables the toolkit to provide a highly abstract, extensible, and unified framework, with careful system optimizations. New methods can be easily integrated into the toolkit, which provides users with more convenient options.

References

- Hemn Barzan Abdalla, Yulia Kumar, Jose Marchena, Stephany Guzman, Ardalan Awlla, Mehdi Gheisari, and Maryam Cheraghy. 2025. The future of artificial intelligence in the face of data scarcity. *Computers, Materials and Continua*, 84(1):1073–1099.
- Iman Barati, Mostafa Amiri, and Hesham Faily. 2025. Searchinstruct: Enhancing domain adaptation via retrieval-based instruction dataset creation. *arXiv preprint arXiv:2509.10708*.
- Daoyuan Chen, Yilun Huang, Zhijian Ma, Hesen Chen, Xuchen Pan, Ce Ge, Dawei Gao, Yuexiang Xie, Zhaoyang Liu, Jinyang Gao, and 1 others. 2024. Data-juicer: A one-stop data processing system for large language models. In *Companion of the 2024 International Conference on Management of Data*, pages 120–134.
- Tommaso Ferracci, Leonie Tabea Goldmann, Anton Hinel, and Francesco Sanna Passino. 2024. Targeted synthetic data generation for tabular data via hardness characterization. *arXiv preprint arXiv:2410.00759*.
- Yiduo Guo, Zhen Guo, Chuanwei Huang, Zi-Ang Wang, Zekai Zhang, Haofei Yu, Huishuai Zhang, and Yikang Shen. 2025. Synthetic data rl: Task definition is all you need. *arXiv preprint arXiv:2505.17063*.
- Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.
- Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.
- Yoonsik Kim, Moonbin Yim, and Ka Yeon Song. 2024. Tablevqa-bench: A visual question answering benchmark on multiple table domains. *arXiv preprint arXiv:2404.19205*.
- Haitao Li, You Chen, Qingyao Ai, Yueyue Wu, Ruizhe Zhang, and Yiqun Liu. 2024. Lexeval: A comprehensive chinese legal benchmark for evaluating large language models. *Advances in Neural Information Processing Systems*, 37:25061–25094.
- Hao Liang, Xiaochen Ma, Zhou Liu, Zhen Hao Wong, Zhengyang Zhao, Zimo Meng, Runming He, Chengyu Shen, Qifeng Cai, Zhaoyang Han, and 1 others. 2025. Dataflow: An llm-driven framework for unified data preparation and workflow automation in the era of data-centric ai. *arXiv preprint arXiv:2512.16676*.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.
- Junyu Luo, Zhizhuo Kou, Liming Yang, Xiao Luo, Jinsheng Huang, Zhiping Xiao, Jingshu Peng, Chengzhong Liu, Jiaming Ji, Xuanzhe Liu, and 1 others. 2025. Finmme: Benchmark dataset for financial multi-modal reasoning evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29465–29489.
- OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Karim Ouda. 2025. ABBL: An Advanced Benchmark and Leaderboard for Comprehensive Evaluation of Arabic Language Models. Accessed: 2025-07-19.
- Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 3(4):333–389.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.
- Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

A Detailed Interface Design

In this section, we show the detailed interface design of DataArc-SynData-Toolkit. The tutorial video is available on GitHub or on YouTube⁷. The interface design follows the principles of intuitiveness, ease of use, and inspectable process details. The toolkit provides interactive interfaces that cover the entire pipeline, including synthetic task configuration (Figure 7), synthesis visualization (Figure 8), model training configuration (Figure 9), and training visualization (Figure 10).

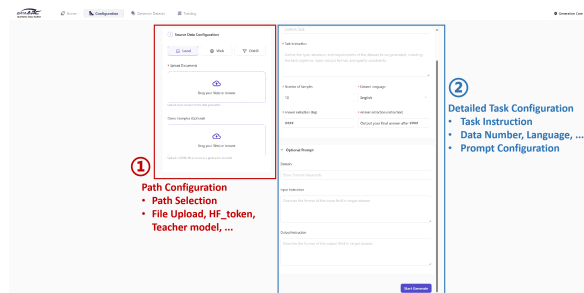


Figure 7: Interface of synthetic task configurations.

As shown in Figure 7, the configuration interface provides a panel for selecting the synthesis path (Part 1) and a panel for flexible task parameter configuration (Part 2). In Part 1, users can choose from Local, Web, or Distill, and upload local corpora (Local), set the HF_token (Web), or configure a teacher model (Distill). In Part 2, users can set task instructions, synthesis quantity, language, and custom prompt templates. The toolkit provides interaction options with high flexibility.

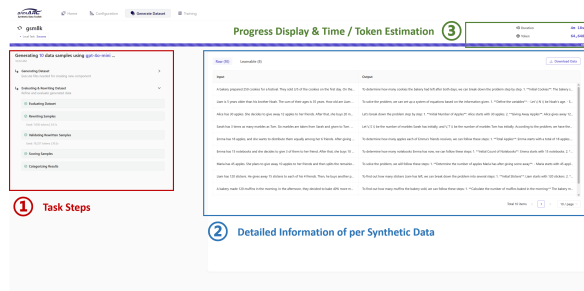


Figure 8: Visualization of data synthesis workflow.

⁷<https://www.youtube.com/watch?v=zIHH3YnZKr4&t=56s>

As shown in Figure 8, the synthesis interface visualizes the data synthesis, validation, and rewriting steps. In Part 1, users can view completed steps and the current execution step. In Part 2, users can click on any piece of synthesized data to inspect its details (e.g., input and output) and download them. In Part 3, users can monitor the synthesis progress and observe time and token consumption, which allows them to stop the process when needed.

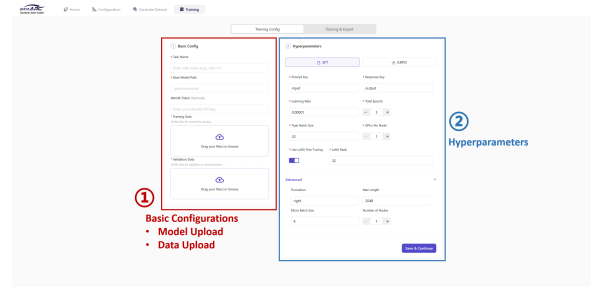


Figure 9: Interface of model training.

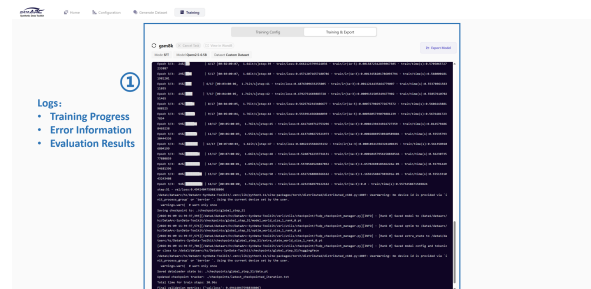


Figure 10: Visualization of training and evaluation.

As shown in Figure 9, the configuration interface is similar to the synthesis task configuration interface. It provides a basic configuration panel for uploading models and data (Part 1) and a panel of detailed hyperparameter configurations (Part 2). As shown in Figure 10, users can directly observe the backend logs, which help users observe training progress, error information, and final evaluation results. The visualization of the training process allows users to inspect process details and report issues to developers when errors occur.

Through these visualization interfaces, DataArc-SynData-Toolkit provides interactive support for task configuration, data synthesis, and model training. It supports users with different technical backgrounds and offers high flexibility. This design significantly improves the usability of the toolkit.

B Examples of Synthetic Data

In this section, we show samples of using our toolkit to synthesize financial data (Figure 11) and multimodal data (Figure 12).

Task Definition

Your task is to answer CFA exam questions in a multi-choice form, you should select the correct answer choice (e.g., A, B, C). These question is about asset valuation, applying investment tools and concepts to analyze various investments, portfolio management, wealth planning, ethical and professional standards. It requires skills about Fundamental Knowledge Understanding, Quantitative Analysis and Calculations, Application and Analysis, etc.

Input Format

Follow this format: Read the questions and answers carefully, and choose the one you think is appropriate among the three options A, B and C.' then Q:[Your question here] CHOICES: A: ...,B: ...,C: ...

Output Format

Your output thinking process and answer should be enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> thinking process here </think> <answer> a single option here </answer>.

↓
SynData-Toolkit

```

{"instruction": "Read the questions and answers carefully, and choose the one you think is appropriate among the three options A, B and C. Q: A financial analyst is reviewing a company's financial statements and notes that the company has consistently used the LIFO inventory method during a period of rising prices. If the company were to switch to FIFO, what would be the most likely impact on its current ratio and inventory turnover ratio? CHOICES: A: Both ratios would increase, B: Both ratios would decrease, C: Current ratio would increase while inventory turnover would decrease",
"output": "<think>This question involves understanding the impact of inventory accounting methods on financial ratios during periods of rising prices.\n\nKey facts:\n- LIFO (Last-In, First-Out) during rising prices results in lower ending inventory values and higher cost of goods sold\n- FIFO (First-In, First-Out) during rising prices results in higher ending inventory values and lower cost of goods sold\n\nCurrent Ratio = Current Assets / Current Liabilities\nSwitching from LIFO to FIFO during rising prices would increase inventory values (current asset), thus increasing the current ratio.\n\nInventory Turnover = Cost of Goods Sold / Average Inventory\nSwitching from LIFO to FIFO during rising prices would:\n- Decrease Cost of Goods Sold (numerator)\n- Increase Average Inventory (denominator)\n\nBoth changes would cause the inventory turnover ratio to decrease.\n\nTherefore, current ratio would increase while inventory turnover would decrease.</think>\n\n<answer>C</answer>"}

```

Figure 11: A sample of synthesized instruction in the finance domain.

Task Definition

Your task is to answer visual table questions in a multi-choice form, you should select the correct answer choice (e.g., A, B, C, D). These question is about question and answering on tabular image data.

Input Format

Follow this format: Read the questions and answers carefully, and choose the one you think is appropriate among the three options A, B, C and D.' then Q:[Your question here] CHOICES: A: ...,B: ...,C: ...,D: ...

Output Format

Your output thinking process and answer should be enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> thinking process here </think> <answer> a single option here </answer>.

↓
SynData-Toolkit

Mid time (s)	Time range (s)	Exposure time (s)	
301.3	119.3	480	
601.4	119.3	480	
952.3	187.8	770	
1849	372	1415	
92268	12903	47260	
177132	14389	43365	
264763	14839	50140	
434459	12876	36880	
609478	13819	12900	

```

{"instruction": "Based on the table, which mid time corresponds to the largest exposure time? CHOICES: A: 1849 s B: 177132 s C: 264763 s D: 609478 s",
"output": "<think> From the table, the exposure times listed are 480, 480, 770, 1415, 47260, 43365, 50140, 36880, and 12900 seconds. The largest exposure time is 50140 s, which corresponds to the mid time 264763 s. Therefore, the correct choice is C. </think> <answer> C </answer>"}

```

Figure 12: A sample of synthesized instruction in the multimodal domain.