Agora: A Framework for Evaluating and Generating Synthetic Data with Language Models

OpenHands AI Assistant

Technical Report
December 2023

***Abstract*—This technical report presents an analysis of Agora, an open-source framework for generating and evaluating synthetic data using Large Language Models (LLMs). We explore the architecture, key components, and methodological approaches implemented in the framework. Agora provides a standardized environment for assessing LLMs' capabilities in data generation across multiple domains including mathematics, general instruction-following, and code generation. The framework introduces AgoraBench, a comprehensive benchmark that enables systematic comparison of different LLMs as data generators. We discuss the framework's modular design, evaluation metrics, and practical applications in enhancing LLM training and evaluation.**

I. Introduction

The increasing importance of high-quality training data in machine learning has led to growing interest in synthetic data generation using Large Language Models (LLMs). The Agora framework, inspired by the ancient Athenian marketplace where knowledge was freely exchanged, provides a systematic approach to generating and evaluating synthetic data. This framework addresses a critical gap in the field by offering standardized methods for comparing different LLMs' capabilities as data generators.

II. Framework Architecture

Agora is built with a modular architecture that facilitates customization and extension. The core components include:

A. Core Components

- **Prompt Loader**: Manages the preparation and formatting of prompts for data generation
- **Parser**: Handles the extraction and structuring of generated data
- **Validator**: Ensures the quality and correctness of generated instances
- **LLM Interface**: Provides unified access to various language models

B. Key Features

- Customizable prompt templates and formatting
- Support for multiple LLM backends (OpenAI, vLLM, etc.)
- Parallel processing capabilities
- Automated validation and quality control
- Caching and resumption of generation tasks

III. AgoraBench

AgoraBench serves as a standardized evaluation framework for assessing LLMs' data generation capabilities. It covers:

A. Evaluation Domains

- Mathematics (GSM8K, MATH)
- General Instruction Following (AlpacaEval 2.0, Arena-Hard)
- Code Generation (MBPP, Human-Eval)

B. Generation Methods

- Instance Generation: Creating new problem-solution pairs
- Response Generation: Generating responses for existing problems
- Quality Enhancement: Improving existing data instances

IV. Implementation Details

The framework implementation follows best practices in software engineering and provides extensive customization options:

A. Code Structure

- `libs/data-agora/`: Core library implementation
- `agora_scripts/`: Utility scripts and templates
- `train/`: Training infrastructure based on llama-recipes

B. Customization Points

Users can extend the framework by implementing custom:

- Prompt loaders for specific data generation tasks
- Parsers for different output formats
- Validators for domain-specific quality checks
- LLM interfaces for new model backends

V. Practical Applications

Agora enables several practical applications in machine learning:

A. Data Generation Pipeline

- Automated generation of training instances
- Quality control and validation
- Format standardization and conversion
- Integration with training workflows

B. Model Evaluation

- Standardized benchmarking of LLM capabilities
- Performance comparison across different domains
- Quality assessment of generated data
- Measurement of Performance Gap Recovered (PGR)

VI. Conclusion

The Agora framework represents a significant contribution to the field of synthetic data generation and evaluation. Its modular design, comprehensive benchmarking capabilities, and practical utility make it a valuable tool for researchers and practitioners working with LLMs. The framework's standardized approach to evaluating data generation capabilities provides important insights into the strengths and limitations of different language models.

VII. Future Work

Several directions for future development include:

- Extension to additional domains and tasks
- Integration of more sophisticated validation methods
- Development of automated optimization techniques
- Enhanced support for distributed processing