OPEN MAINFRAME PROJECT

zopen community

Automated Porting of Tools to IBM Z/OS

By:

Pavan S
Aravind A
Prithiviraj N

INTRODUCTION

Porting software tools from conventional operating systems to IBM z/OS is a complex and labor-intensive task, primarily due to architectural disparities and platform-specific constraints. Unlike Linux or Windows environments, z/OS has a unique set of system interfaces, compilers, and runtime behaviors that are often incompatible with open-source tools designed for POSIX-like systems. Developers frequently encounter issues such as missing or unsupported system calls, differing file system semantics, compiler-specific errors, and outdated dependencies. These roadblocks can cause persistent build failures and unpredictable runtime behavior.

Addressing these issues typically requires extensive manual effort—debugging complex error messages, writing system-specific patches, and reconfiguring build environments through trial and error. In many cases, developers must also gain deep expertise in both the source tool and the target z/OS environment, further increasing the time and effort required. This not only slows down the migration process but also introduces opportunities for human error, inconsistencies, and maintenance challenges. As a result, many open-source tools remain inaccessible on z/OS, limiting developer productivity and slowing the adoption of modern software practices in mainframe environments.

MOTIVATION

Challenge: Porting open-source tools to IBM z/OS is complex and error-prone due to significant architectural differences and system-specific constraints.

Developer Pain Points:

- Frequent build failures
- Compatibility issues
- High manual effort and prolonged timelines

Project Motivation:

- Simplify and automate the porting process
- Reduce manual debugging and intervention

OBJECTIVES

- **Create a Robust Dataset Manually and then Automate collection process for LLM Training:** Collect and organize a detailed dataset comprising errors, faulty code, applied patches, and corrected code, which will be utilized by the LLM to enhance its predictive capabilities and optimize future porting tasks.
- **Error Detection and Resolution**: Leverage the LLM to identify and classify build-time errors and faulty code segments and generate automated suggestions for patches to resolve these issues, accelerating the debugging and correction process.
- **Automate Tool Porting with LLM Integration**: Develop an automated framework that uses a trained Large Language Model (LLM) to assist in the porting of open-source tools to the z/OS environment, reducing the need for manual intervention and improving speed and accuracy.
- **Support Continuous Improvement and Research**: Foster ongoing development by enabling the LLM to learn from previous porting projects, improving the model's accuracy and efficiency over time, and contributing to research in the field of automated porting and mainframe toolchain optimization.

CHALLENGES

- **Manual Data Collection Doesn’t Scale:** Collecting and mapping errors, buggy code, and corresponding patches manually was time-consuming and prone to mistakes. Repeating this for many tools made it clear that automation was needed to ensure accuracy and efficiency.
- **Separating Build Errors from Feature Patches:** It was important to include only patches that fixed build errors (not those adding new features) in the training data. Mixing both types would confuse the LLM and reduce its ability to learn meaningful error-fix patterns.
- **Inconsistent Data Formats:** Different tools and errors had different formats, making it hard to maintain consistency. We had to write custom scripts to clean and structure the data uniformly.
- **No Existing Training Data:** There were no ready-made datasets for this kind of work on z/OS, so we had to build everything from scratch, including collecting errors, patches, and outputs.
- **Designing Effective Prompts for the LLM:** Teaching the LLM to understand error messages and generate correct fixes required careful prompt design, testing, and refinement over multiple iterations.
- **Automating Build and Patch Workflows on z/OS:** Setting up scripts to automate building tools, capturing errors, applying patches, and re-validating builds on z/OS was technically complex due to platform-specific constraints.
- **Creating a Feedback Loop for Learning:** To improve the LLM over time, we needed a way to test its patch suggestions in real builds and use the results to improve the training data.

ACCOMPLISHMENTS

Successfully installed InstructLab **with custom modifications to its internal codebase**, enabling advanced environment configuration and tailored fine-tuning workflows for custom LLM models.

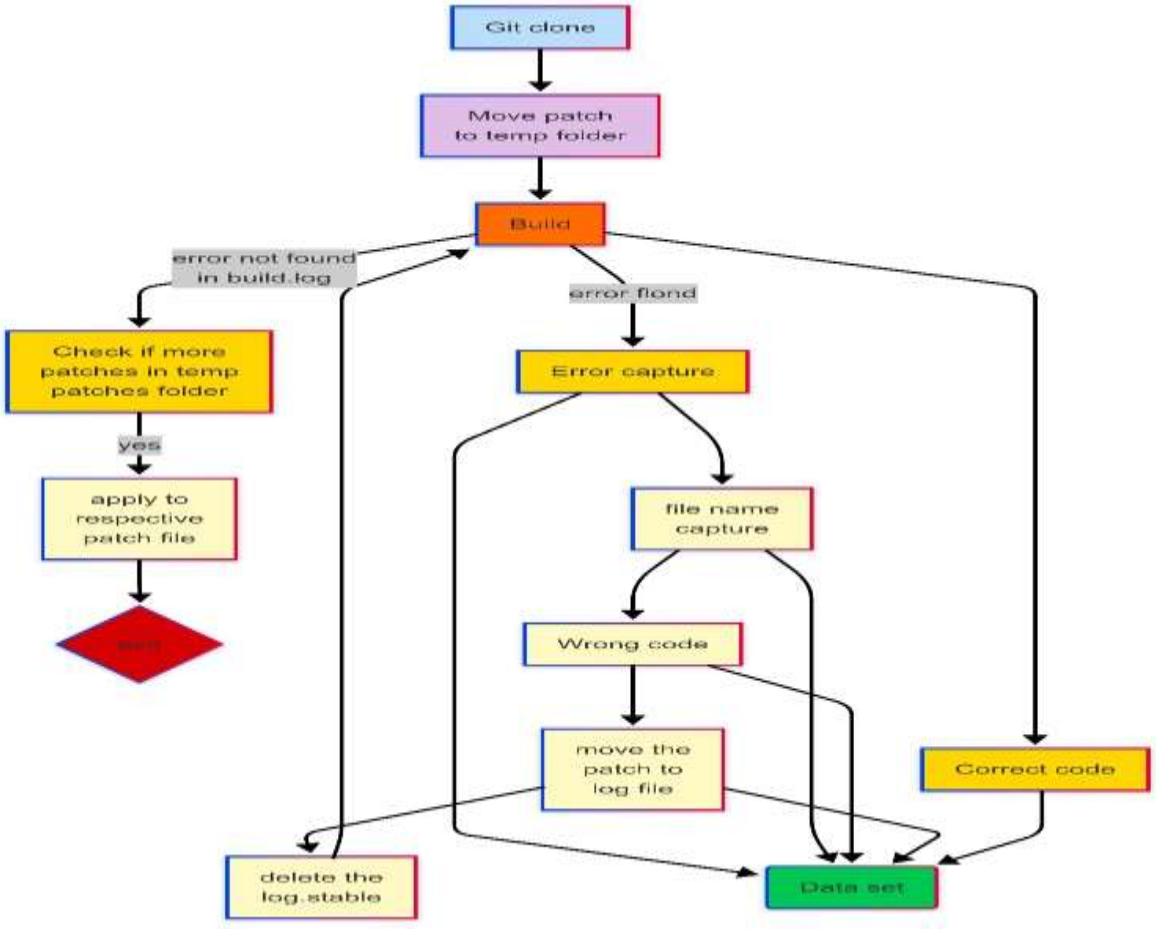
Analyzed real-world code repositories to create a structured taxonomy and conducted an initial local proof-of-concept for LLM fine-tuning.

Developed automation scripts to extract and build taxonomies from code patches, ensuring **scalable, repeatable training workflows** which captures the errors data to make a dataset for fine-tuning the LLM

Leveraged the taxonomy to generate synthetic training data, simulating realistic error-patch scenarios for effective model learning.

Trained the LLM using curated synthetic datasets, fine-tuning it to understand and suggest relevant code corrections.

DATASET CREATION AUTOMATION WORKFLOW



FUTURE WORKS

Optimize Dataset Creation Automation: Enhance the efficiency and scalability of the dataset generation pipeline to minimize redundancy and processing time.

Refactor LLM Training Codebase: Redesign and modularize the LLM training code for improved readability, reusability, and easier experimentation.

Orchestrate Dataset Creation Pipeline: Integrate the automation workflow into a fully orchestrated pipeline using tools like Airflow or custom job schedulers for end-to-end control.

Revamp & Expand qna.yaml File: Refine the qna.yaml file to be more concise and comprehensive, enabling the model to better learn contextual error resolutions.

REFERENCES

- https://zopen.community
- github.com/zopencommunity
- https://github.com/instructlab/instructlab