Cross-Language Code Translation with LLMs: A Comparative Study of Accuracy Across Paradigms

CS540 Project Proposal

Manan Patel
Graduate Student
University of Tennessee
mpatel65@vols.utk.edu

Zachary Perry
Graduate Student
University of Tennessee
zperry4@vols.utk.edu

Shayana Shreshta
Graduate Student
University of Tennessee
sshres25@vols.utk.edu

Eric Vaughan
Graduate Student
University of Tennessee
evaugha3@vols.utk.edu

***Abstract*—This project aims to explore the capabilities of LLMs in cross-language code translation and its accuracy when translating across languages with different paradigms. The study will be conducted by combining various datasets that contain code translations between various imperative, object-oriented, and paradigm languages. We will then use this dataset to find which LLMs are best at translating between different languages and build a tool that will allow developers to simply input code and translate it to a desired language using the LLM that is best at that specific translation. We hope to provide a comprehensive guide as to which LLMs are best for specific programming language translations and provide a tool that will automatically do this for the user, giving them the most accurate results. The work for this project will be split equally among all members. We will all work on combining the dataset, finding the best LLMs, and building the application.**

I. Introduction

The rise of Large Language Models (LLMs) have completely revolutionized the field of software development and research. With these new tools, developer productivity and efficiency have drastically increased and additional tools, such as specialized coding assistants, have only pushed this boundary forward. While LLMs provide many capabilities, this project focuses on their application in the domain of cross-language code translation, with a specific interest in the accuracy when translating across languages with different paradigms. Currently, this area lacks comprehensive research for LLM performance for this specific task and we aim to fill this gap and help determine how LLMs can be further used to enhance developer productivity. Our team for this project is composed of members with various backgrounds in software engineering and research. Having a team with experience in both of these domains is perfect for this project, as both will be required to complete it and the research itself is relevant to both fields.

II. Methodology

As previously mentioned, our objective is to create a tool that compares the cross-language and cross-paradigm code translation of various LLMs. This will allow for individuals to determine which LLM is best to use for their specific code translation task. To create such a tool, we first need a vast amount of code translation data that can be used to evaluate the various LLMs. Fortunately, a few code-translation datasets already exist, with a couple of big ones being Cross-Lingual Code SnippeT (or XLCoST) [1] and CodeTransOcean [2]. Both of these datasets focus on translations between the same seven languages: C++, Java, Python, C#, Javascript, PHP, and C. Thus, these datasets include imperative languages, object-oriented languages, and multi-paradigm languages. While a primarily functional language is not included, the multi-paradigm languages certainly do have functional features. If we deem that functional code snippets are not adequately represented, we can manually add some code snippets from functional programming languages as well as their translations to the other seven languages. This would be a hefty amount of work, so we may only pursue this if time permits.

Once we have our data, we can shift our focus to figuring out the best LLM at translating from each combination of start language and end language. The LLMs that we are currently thinking about evaluating are OpenAI Codex, CodeBERT from Microsoft and Hugging Face, and TransCoder from Facebook. Each of these LLMs will be evaluated using code snippets from the XLCoST and CodeTransOcean datasets. In terms of validating our results, we will run our experiment several times with a randomly selected set of code snippets that all the LLMs will face. The results should be decently consistent across all of our runs.

Then, once we know which LLMs are best at translating between each set of languages, we can shift our focus to our tool. The current vision is to create a website where one can enter three pieces of information: the code snippet, the language of the code snippet, and the desired language for the code snippet translation. Our tool will then look at the start language and desired end language, and translate the code using the LLM that is best at translating between the two given languages. All of the steps mentioned above are shown in the figure below.

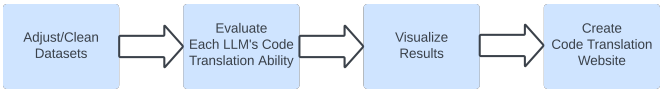

Fig. 1. Methodology

III. Research Value

This research is intended for both researchers and practitioners in programming languages, software development, and artificial intelligence. Researchers aim to evaluate how well large language models (LLMs) translate code between languages with different programming paradigms, such as imperative and object-oriented. Practitioners, including software developers and educators, need reliable tools and best practices to translate code accurately, saving time and reducing errors in projects or teaching. The main challenge is the lack of comprehensive studies on LLM performance in cross-paradigm translation and the difficulty practitioners face in manually rewriting code or relying on existing automated tools. While solutions like OpenAI's Codex and rule-based translators exist, they often struggle with paradigm-specific differences, highlighting the need for a deeper understanding of how LLMs handle these challenges.

This research will provide both researchers and developers with practical insights and useful tools. Researchers will gain access to a benchmark dataset to evaluate how well large language models (LLMs) translate code across different programming paradigms, along with a detailed analysis of their strengths and weaknesses. This will help guide future studies and highlight areas where LLMs still struggle. For developers and educators, the research will offer clear guidelines on which LLMs work best for specific translation tasks and how to improve accuracy when converting code between languages like Python, C++, Java, Javascript etc. By reducing the need for manual effort and minimizing errors, these insights will make coding workflows more efficient. Ultimately, this research aims to close the gap between theory and real-world use, helping both researchers and practitioners get the most out of LLMs for code translation.

The success of this project will be measured through both quantitative and qualitative factors. For researchers, success means that the benchmark dataset is useful for evaluating LLMs in future studies and that the findings contribute to discussions on improving code translation. For developers and educators, success will be reflected in clear, practical insights on using LLMs for cross-paradigm translation, helping them reduce errors and improve efficiency. On a technical level, accuracy rates above 90% and error rates below 5% will indicate strong model performance. Additionally, feedback from classmates, instructors, or developers testing the results will help assess the quality and usability of the findings. Ultimately, this project aims to bridge the gap between research and real-world application, providing valuable insights for both academic and practical use.

IV. Project Management

Schedule Overview

Our project will follow a structured schedule to ensure timely completion. We will meet twice a week on Discord for virtual collaboration and once a week in person after class for hands-on discussions. The project is divided into four sprints:

- **Sprint 1**: Focuses on project proposal submission, planning, research, and frontend design.
- **Sprint 2**: Involves backend design, LLM integration setup, and frontend development.
- **Sprint 3**: Centers on backend development and integrating the LLM with the frontend.
- **Sprint 4**: Dedicated to testing, debugging, final review, presentation preparation, and project submission.

Risk Management

The proposed tasks are achievable, but we have identified potential risks and mitigation strategies. Critical tasks include project proposal submission, planning and research, frontend and backend design, development, LLM integration, and testing. Peripheral tasks, such as LLM integration setup, documentation, UI polishing, and project presentation, will be addressed after prioritizing critical deliverables.

To manage time constraints, we will categorize features into "must-have" and "nice-to-have." This prioritization ensures that essential functionalities are completed first, while additional features can be added if time permits. The team is committed to working collaboratively to address challenges and maintain progress.

Resource Availability

All data and tools required for the project are open-source and readily available, minimizing dependencies on external resources. This accessibility allows us to focus on development and integration without delays related to resource acquisition. The use of open-source tools also ensures that our project remains cost-effective and scalable.

Team Skills and Expertise

Our team members bring a mix of academic and real-world experience to the project. Having worked on projects in previous semesters and during internships, we are familiar with the proposed methods and technologies. While some team members will be working on certain aspects for the first time, our collective experience ensures that we can support each other and overcome learning curves efficiently. This balance of familiarity and new challenges will drive innovation and growth throughout the project.

V. Team

Team Member	Role
Shayana Shreshta	Full Stack Developer
Zac Perry	Backend Developer
Manan Patel	Project Manager and Software Developer
Eric Vaughan	Software Developer

TABLE I
Team Members and Their Roles

REFERENCES

[1] Weixiang Yan, Yuchen Tian, Yunzhe Li, Qian Chen, and Wen Wang. CodeTransOcean: A comprehensive multilingual benchmark for code translation. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5067–5089, Singapore, December 2023. Association for Computational Linguistics.

[2] Ming Zhu, Aneesh Jain, Karthik Suresh, Roshan Ravindran, Sindhu Tipirneni, and Chandan K. Reddy. Xlcost: A benchmark dataset for cross-lingual code intelligence, 2022.