INTEGRATE A LARGE LANGUAGE MODEL WITH THE XEUS-CPP JUPYTER KERNEL

Google Summer of Code 2024

MENTORS
Anutosh Bhat, Johan Mabille, Aaron Jomy, David Lange, Vassil Vassilev

MENTEE
Tharun Anandh
ABOUT ME

- Bachelor of Technology (2023), Computer Science and Engineering at National Institute of technology, Tiruchirapalli, Tamil Nadu, India.
- Experimented with android Dev, backend frameworks, data analysis, machine learning, deep learning techniques related to image manipulation, IoT routing.
- Application Developer at Oracle India.
WHAT IS XEUS-CPP?

- Xeus - Library meant to facilitate the implementation of kernels for Jupyter.
- Xeus-cpp - Implementing the interpreter part of the kernel. This setup empowers users to interactively write and execute C++ code, providing immediate visibility into the results. With its REPL (read-eval-print-loop) functionality, users can rapidly prototype and iterate without the need to compile and run separate C++ programs.
HOW WILL THIS PROJECT HELP?

With the ever growing popularity of large language models, this project aims to integrate a large language model with the xeus-cpp Jupyter kernel. This integration will enable users to interactively generate and execute code in C++ leveraging the assistance of the language model.
MY IDEA

- Autocomplete: Enter code -> Press tab -> Code suggestion pops up -> Select the necessary code.

- Code Generation: Enter magic command and the prompt -> LLM suggestion is given as output.
Here's the Python code to solve the 2D Laplace equation in Cartesian coordinates:

```python
import numpy as np
import matplotlib.pyplot as plt

# Set up grid
nx = 101
ny = 101
nt = 100

dx = 1. / (nx - 1)
dy = 1. / (ny - 1)

x = np.linspace(0, 1, nx)
y = np.linspace(0, 1, ny)

# Initialize solution
u = np.zeros((nx, ny))

# Set boundary conditions
u[0,:] = 0
u[-1,:] = 0
u[:0] = 0
```
<table>
<thead>
<tr>
<th>Closed Source</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open AI, Gemini etc.</td>
<td>Mistral, CodeLlama, WizardCode, etc.</td>
</tr>
</tbody>
</table>

- Users have their own API keys and use the implemented plugins to access the LLM.
- Might yield better results than open source.

- Host the API for these models in the cloud
- User’s code remains in a secure environment and is not shared publicly.
CODING PHASE 1

- Experiment and figure out xmagics and code_complete.
- Implement xplugin framework and establish connection to the LLM.
- Develop a basic pipeline of the whole project.
- Look into scalability and concurrency.
CODING PHASE 2

- Scale the project.
- Write extensive tests and add rules to build.
- Document the features.
- Write comprehensive documentation about the new features.
PROGRESS SO FAR

- Updated tutorials documentation and added documentation for testing.
- Increased test coverage from 49% to 82% (and hopefully more).
- Experimenting with OpenLLM.
FUTURE SCOPE

- RAG Implementation.
- Xeus-cpp contribution.
- Contributing to other projects in the org.
THANKS FOR LISTENING