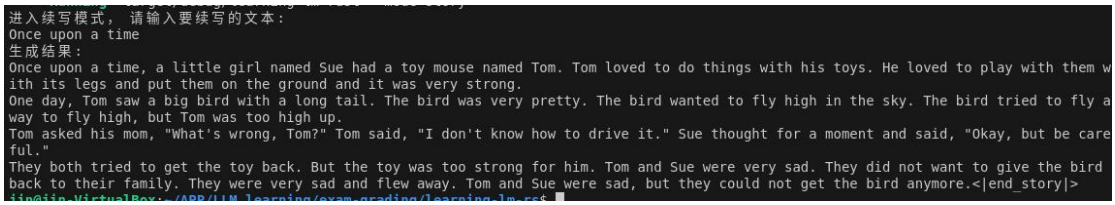
大模型推理系统项目报告

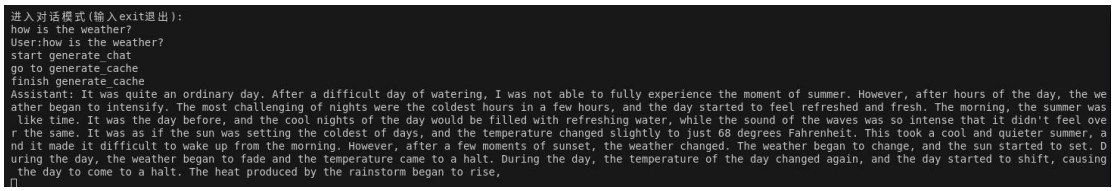
一．基础功能：

故事续写：

命令行直接显示续写结果。



AI 对话：



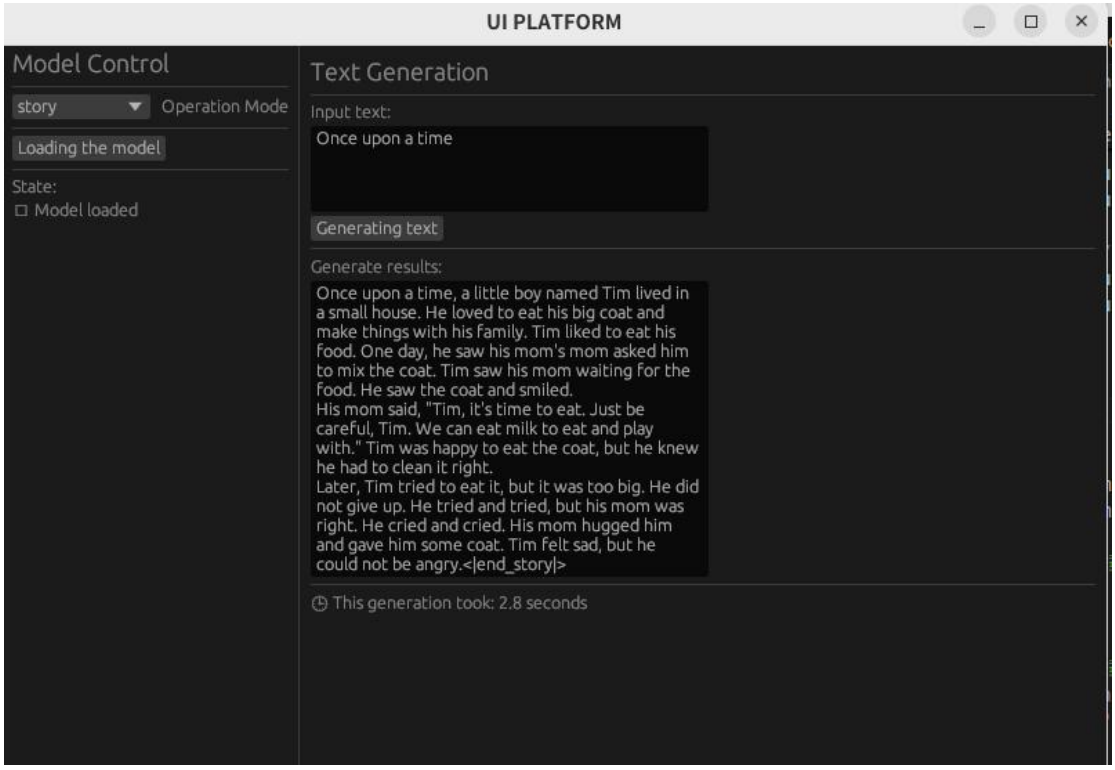
二．优化及扩展

1. 可交互 UI

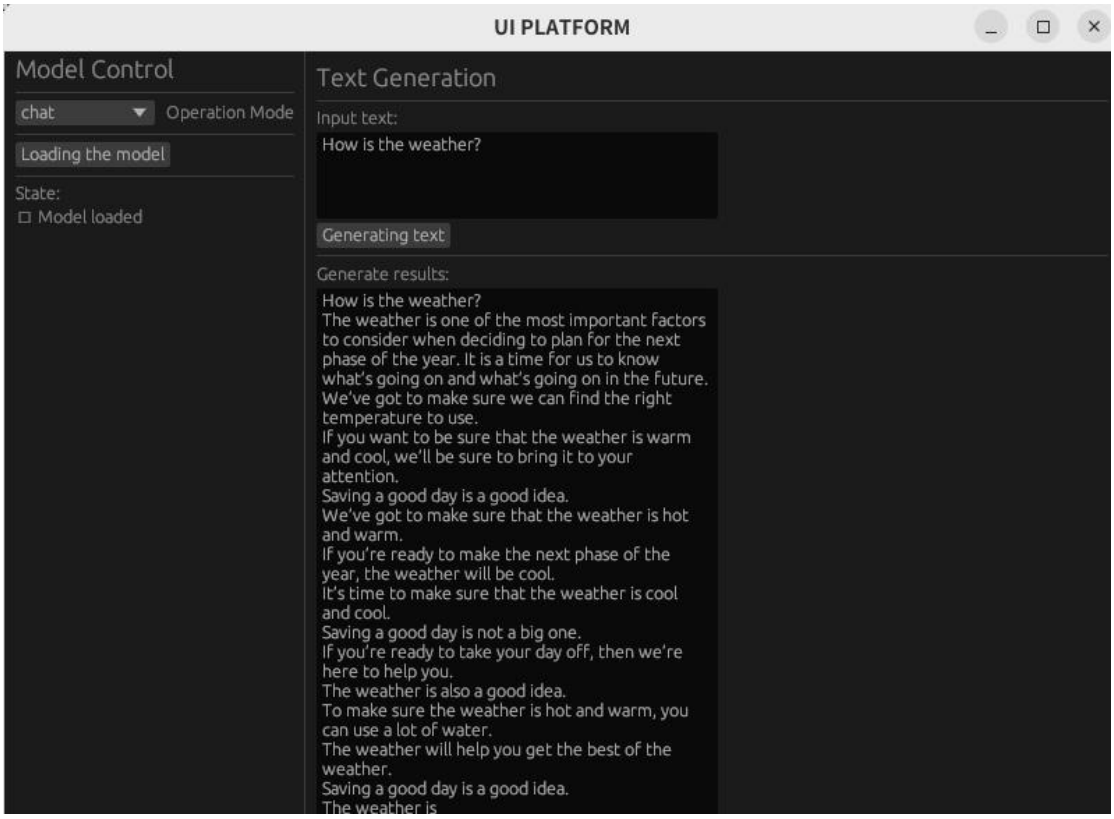
使用 egui 库实现，可以选择 chat/story 模式，添加了加载模型按钮，分别加载 chat 或 story 模型。用户在右侧输入文本，按 Generating text 按钮，输出对应的回复文本。

在底部添加记录耗时的模块，方便比较各种优化方法的效果。

Story mode:

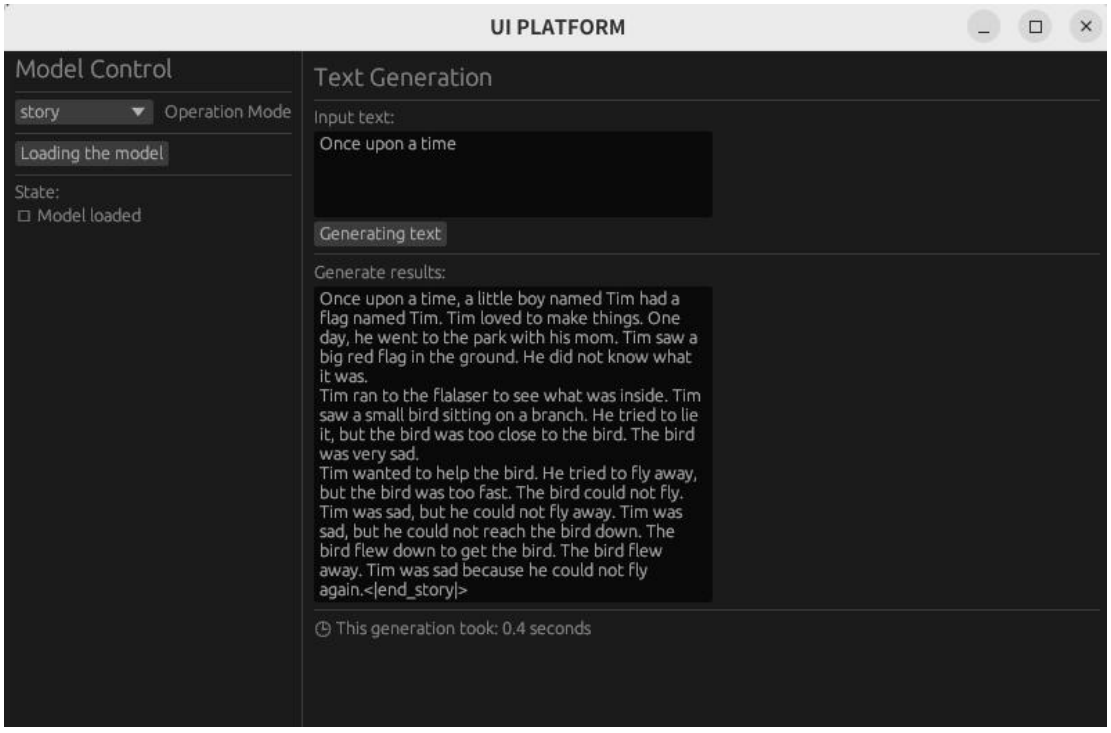


Chat mode:



2. SMID 加速矩阵乘算子

使用 AVX 指令集，做 256 位向量操作，一次处理 8 个 f32 数据，可以充分提高计算吞吐量。测试 story 模式下，输入同样的文本，生成回复耗时较没有优化时，有了较大的减少。由原来的 2.8s 缩减为 0.4s。



3. 混合精度推理

降低模型存储占用和计算量，用 FP16 替换 FP32 ，内存需求减少 50%，理论计算吞吐量提升 2 倍。在保证精度损失可控的前提下，加快推理速度。

主要对以下算子进行了混合精度处理：

（1）矩阵乘法 MatMul

在注意力机制中的 QKV 投影、MLP 层的全连接都有用到。

GPU 对 FP16 有专用硬件单元 Tensor Core，调用 Tensor Core 执行 FP16(输入) × FP16(权重) → FP32(累加)。

（2）激活函数 ReLU

在 MLP 层。逐元素运算，FP16 精度足够且计算快。用近似公式减少计算量。

首先通过硬件指令将输入数据从 FP32 转换为 FP16，存储到 GPU 的共享内存中。在 GPU 核函数中，通过线程并行处理每个 FP16 元素。最后将 ReLU 的 FP16 输出转换回 FP32 格式。

精度验证：

对比混合精度下的中间结果，使用 L2 误差计算使用 FP16 的中间层输出与 FP32 结果的欧氏距离，衡量数值偏离程度。

人工评价生成文本的连贯性、语义准确性。

从结果来看，使用混合精度后的 L2 误差在 1e-6 内，语义基本正常。

4. CUDA 加速计算

使用 Rust 封装的 CUDA 库 rustacuda 管理 GPU 内存，加速计算。

未来方向：

用 TensorRT 做推理加速。