РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ПЛАГИНА С ЛОКАЛЬНОЙ МОДЕЛЬЮ ГЕНЕРАЦИИ КОДА ДЛЯ ЯЗЫКА SC-КОД

Дубинина Е. О., Захаров В. В.
Кафедра интеллектуальных информационных технологий,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: eugenia.dubinina@gmail.com, zakharov@bsuir.by

Большие языковые модели демонстрируют высокую эффективность в задачах генерации кода, однако их применение ограничено популярными языками программирования. Специализированные языки, такие как SC-код платформы ОСТИС, остаются без поддержки. Более того, большинство существующих инструментов полагаются на облачную инфраструктуру, что создает проблемы с конфиденциальностью данных и требует постоянного подключения к сети. В данной работе представлен подход к разработке интеллектуального плагина для среды разработки, который интегрирует локально развернутую и оптимизированную модель для генерации SC-кода. Для решения проблемы нехватки данных был собран уникальный датасет. Модель семейства CodeGen была дообучена и оптимизирована с использованием методов квантования и дистилляции, что позволило достичь высокой производительности на потребительском оборудовании. Тестирование прототипа показало точность генерации 82,5% при средней задержке 165 мс.

Введение

Интеграция больших языковых моделей (БЯМ) в интегрированные среды разработки кардинально меняет процессы написания кода. Инструменты, такие как GitHub Copilot и JetBrains AI Assistant, ускоряют разработку, предлагая автодополнение и генерацию кода по запросу на естественном языке. По данным исследований GitHub, разработчики, использующие Copilot, выполняют задачи до 55% быстрее, а в крупных технологических компаниях ИИ-ассистенты уже генерируют от 40% до 60% нового кода.

Однако у этого подхода есть две существенные проблемы. Во-первых, существующие модели обучены преимущественно на популярных языках. Специализированные и доменно-специфические языки, к которым относится декларативный SC-код платформы ОСТИС, практически не представлены в обучающих выборках. Это приводит к синтаксически и семантически некорректным результатам генерации.

Во-вторых, большинство сервисов работают по облачной модели. Это вызывает опасения относительно конфиденциальности исходного кода и делает невозможной работу в офлайн-режиме.

Целью данной работы является разработка прототипа интеллектуального плагина для среды разработки, который решает обе проблемы: он интегрирует локально развернутую БЯМ, специально дообученную для генерации кода на языке SC-код.

I. Подход к разработке

Для достижения поставленной цели необходимо было решить три ключевые задачи:

1. Инженерия данных: Формирование и аугментация специализированного корпуса данных (датасета). Этот этап был необходим для последующего дообучения модели на синтаксических и семантических конструкциях SC-кода, который не представлен в публичных выборках.
2. Адаптация и оптимизация модели: Выбор подходящей архитектуры БЯМ, ее дообучение на собранном датасете и применение методов компрессии (таких как квантование и дистилляция) для обеспечения высокой производительности и низкого потребления ресурсов на локальном устройстве.
3. Системная интеграция: Проектирование клиент-серверной архитектуры, включающей сам плагин для среды разработки (клиент) и локальный бэкенд-сервис, который инкапсулирует оптимизированную модель и обслуживает запросы на генерацию кода

II. Подготовка данных

Ключевой проблемой при работе со специализированными языками, такими как SC-код, является острый дефицит или полное отсутствие публичных размеченных данных для обучения. Для решения этой задачи был применен гибридный подход к формированию корпуса, включающий два источника.

Во-первых, был выполнен систематический анализ и парсинг существующих публичных репозиториев платформы ОСТИС для сбора всего доступного реального кода. Во-вторых, для обеспечения полноты покрытия грамматических конструкций и типичных структур была применена техника синтетической генерации данных.

Для обеспечения возможности обучения модели генерации по текстовому запросу, каждый фрагмент кода был аннотирован (сопоставлен) с описанием на естественном (русском) языке. Ре-

зультирующий объем датасета составил около 250 000 строк кода и 70 000 текстовых описаний.

III. Выбор и оптимизация модели

Для обеспечения локальной работы была выбрана модель из семейства CodeGen, имеющая открытый исходный код и доступная в различных размерах. Модель была дообучена на собранном датасете для адаптации к специфическому синтаксису SC-кода.

1. Квантование: Веса модели были квантованы до 8-битного формата. Это позволило сократить размер модели с 1.4 ГБ до 350 МБ, что критически важно для загрузки в ОЗУ ноутбука, и ускорило вычисления.
2. Дистилляция знаний: Компактная "модель-ученик"была обучена повторять поведение "модели-учителя"большего размера, что позволило сохранить высокое качество генерации при меньшем количестве параметров.

IV. Количественные результаты

Для оценки использовался тестовый набор из 100 запросов на естественном языке, описывающих типовые задачи на SC-коде.

Точность генерации: Общая точность (доля семантически корректных фрагментов) составила 82%. Модель показала высокую точность (90%) на простых запросах (создание узлов, связей) и более низкую (65%) на сложных структурах, что указывает на необходимость расширения датасета.

Таблица 1 – Результаты оценки точности генерации

Категория запросов	Количество запросов	Успешно	Точность, %
Простые	56	55	91
Сложные	40	27	67
Итого	100	82	82

Скорость генерации: Благодаря оптимизации и архитектуре Apple Silicon, среднее время генерации составило всего 165 мс (с разбросом от 110 мс до 285 мс). Это обеспечивает практически мгновенный отклик в среде разработки.

V. Качественные результаты

Было проведено пилотное исследование с участием разработчиков, имеющих опыт работы с ОСТИС.

1. Удобство: 100% пользователей отметили интуитивность интерфейса и удобство вызова генерации.
2. Полезность: Участники подтвердили значительное ускорение рутинных задач, таких как создание стандартных rrel или nrel структур.
3. Качество: В 85% случаев код был признан релевантным. В остальных случаях требовалась незначительная ручная доработка, в основном из-за неоднозначности запросов.

VI. Заключение

В ходе работы был успешно разработан и протестирован прототип интеллектуального плагина, интегрирующего локальную БЯМ для генерации кода на специализированном языке SC-код.

Доказана жизнеспособность подхода, основанного на сборе кастомного датасета и применении методов оптимизации (квантование, дистилляция). Это позволило достичь высокой производительности (82% точность, 165 мс задержка) на стандартном потребительском оборудовании.

Данная работа демонстрирует практический путь создания автономных, конфиденциальных и эффективных ИИ-ассистентов для доменно-специфических языков программирования, что ранее было прерогативой облачных сервисов для популярных языков.

Перспективы развития включают расширение обучающего датасета для повышения точности на сложных запросах и реализацию полноценного функционала, включая контекстное автодополнение и исправление ошибок.

1. Chen, M. Evaluating Large Language Models Trained on Code / M. Chen [et al.] // arXiv preprint arXiv:2107.03374. – 2021.
2. Nijkamp, E. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis / E. Nijkamp [et al.] // arXiv preprint arXiv:2203.13474. – 2022.
3. Hinton, G. Distilling the Knowledge in a Neural Network / G. Hinton [et al.] // arXiv preprint arXiv:1503.02531. – 2015.
4. Dettmers, T. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale / T. Dettmers [et al.] // arXiv preprint arXiv:2208.07339. – 2022.
5. Golenkov, V. V. Ontology-based design of intelligent systems // Open semantic technologies for intelligent systems. – Minsk: BSUIR, 2017. – Iss. 1. – P. 37–56.
6. Голенков, В. В., Технология комплексной поддержки жизненного цикла семантически совместимых интеллектуальных компьютерных систем нового поколения. – Минск: Бестпринт, 2023.