УДК 004.8
Миронов Дмитрий Михайлович
магистрант,
Сургутский государственный университет,
Научный руководитель: Кузин Дмитрий Александрович,
доцент
Сургутский государственный университет

ДООБУЧЕНИЕ БОЛЬШОЙ ЯЗЫКОВОЙ МОДЕЛИ ДЛЯ ГЕНЕРАЦИИ КОДА

Аннотация: *В работе рассматривается задача дообучения большой языковой модели для генерации программного кода на специализированном языке программирования Fore. Для адаптации модели Mistral 7B использован метод LoRA, позволяющий выполнять дообучение при ограниченных вычислительных ресурсах за счет обучения небольшого числа дополнительных параметров. Полученные результаты подтверждают эффективность применения метода LoRA для адаптации больших языковых моделей к узкоспециализированным языкам программирования и задачам разработки программного обеспечения.*

Ключевые слова: *дообучение LLM, метод LoRA, fine-tuning, генерация кода, адаптация большой языковой модели, токенизатор, датасет.*

Annotation: *This paper addresses the task of fine-tuning a large language model for code generation in the specialized programming language Fore. The Mistral 7B model was adapted using the LoRA (Low-Rank Adaptation) method, which enables efficient fine-tuning under limited computational resources by training only a small number of additional parameters. The obtained results demonstrate the effectiveness of the LoRA approach for adapting large language models to domain-specific programming languages and software development tasks.*

Keywords: *LLM training, LoRA method, fine-tuning, code generation, adaptation of a large language model, tokenizer, dataset.*

Введение. Во времена активного развития искусственного интеллекта, функционал больших языковых моделей (LLM) находит широкое применение, поэтому становятся основой для создания различных приложений, начиная от чат-ботов, заканчивая системами анализа данных [1]. В рамках данной работы рассматривается задача генерации программного кода на специализированном языке программирования Fore, используемом для разработки и сопровождения прикладных решений в составе информационных систем. Данный язык относится к категории предметно-ориентированных языков программирования, поскольку его синтаксис и семантика адаптированы под решение конкретных прикладных задач, связанных с обработкой данных, построением отчетности и автоматизацией бизнес-логики. Существующий процесс отладки и анализа кода на языке Fore усложняется отсутствием удобных инструментов, поскольку разработчик вынужден вручную и затратно по времени работать с официальной документацией(справкой) при разборе возникающих ситуаций. При этом узкая специфика языка Fore приводит к отсутствию на рынке специализированных инструментов поддержки разработки, таких как нейросетевые агенты,

способных помочь в анализе, написании и исправлении кода. В процессе разработки и сопровождения кода разработчик сталкивается со следующей ключевой проблемой: высокие временные затраты на доработку, обусловленные сложностью локализации и устранения ошибок.

Целью данной работы является разработка и дообучение нейросетевой модели, предназначенной для работы с внутренним языком программирования Fore, способной помогать с разработкой и решением ошибок. Для достижения поставленной цели необходимо провести следующие технические мероприятия:

- подготовить обучающий набор данных, основанный на справке по языку Fore;
- провести анализ полученного набора данных;
- провести дообучение нейросетевой модели;

Материалы и методы. В качестве обучающего набора данных принято решение использовать официальную документацию по языку программирования Fore. Документация разделяется по разделам, каждый раздел хранится в файлах формата chm контекстной справки. Данные файлы содержат в себе HTML документы с документацией, которые необходимы для обучающего набора данных. Был написан скрипт, который разархивирует разделы и собирает из них все HTML документы в одну папку. Прямое использование документации в процессе обучения не является правильным решением, поскольку большие языковые модели обучаются на примерах взаимодействия пользователя и системы. В связи с этим требовалось преобразовать исходные материалы документации в структуру, содержащую вопросы и ответы. Так как формирование обучающего набора данных является трудоемким процессом было принято решение об использовании API больших языковых моделей DeepSeek и ChatGPT для формирования датасета. Использование двух различных моделей позволило получить разнообразие формулировок в обучающем наборе. Полученная документация передавалась языковой модели вместе со специально подготовленной инструкцией. Основная задача инструкции заключалась в том, чтобы модель анализировала содержимое документа, выделяла конструкции языка программирования Fore и формировала на их основе обучающие примеры в формате «инструкция – вход – выход». Сформированные данные сохраняются в формате JSONL, где каждая строка представляет отдельный обучающий пример. Для хранения используется структура, содержащая три основных поля: instruction, input и output. Поле instruction содержит текст запроса пользователя, поле input используется для передачи дополнительного контекста при его наличии, а поле output содержит ожидаемый ответ модели. На рисунке 1 представлен пример записи обучающего набора данных.

```
{"instruction": "Вставить новый лист в заданную позицию в отчёте.",
"input": "Отчёт с идентификатором REPORT, вставить лист с именем \"Sheet\" +
(текущий индекс + 1) на позицию 1, тип листа - табличный.",
"output": "Sub UserProc;\n\nVar\n\n\t\tMB: IMetabase;\n\n\t\tReport: IPrxReport;\n\n\t\t
Sheets: IPrxSheets;\n\n\t\tSheet: IPrxSheet;\n\n\t\ti: Integer;\n\nBegin\n\n\t
// Получим текущий репозиторий\n\n\tMB := MetabaseClass.Active;\n\n\t
// Получим отчёт\n\n\tReport := Mb.ItemById(\"REPORT\").Edit As IPrxReport;\n\n\t
// Получим листы отчёта\n\n\tSheets := Report.Sheets;\n\n\ti := Sheets.Count;\n\n\t
// Вставим лист\n\n\tSheet := Sheets.Insert(1, \"Sheet\" + (i + 1).ToString, PrxSheetType.Table);\n\n\t
Debug.WriteLine(\"Вставлен лист: \" + Sheet.Name + \";\");\n\n\t
// Сохраним отчёт\n\n\t(Report As IMetabaseObject).Save;\n\nEnd Sub UserProc;"}
```


Рисунок 1. Обучающий набор данных

После разметки проведена ручная корректировка выборки, включающая уточнения формулировок, проверку и дополнения ответов. Итоговый объем выборки составлял более 5100 примеров.

В качестве предобученной модели используется большая языковая модель Mistral 7B. Данная модель предназначена для обработки последовательностей текста и генерации ответов на естественном языке, а также способна генерировать программный код. Mistral – это языковая модель с 7 миллиардами параметров, доступная в предварительно обученном и оптимизированном для выполнения инструкций вариантах. Она призвана сбалансировать затраты на масштабирование больших моделей с их производительностью и эффективностью логического вывода. Адаптация модели к специализированной задаче генерации кода осуществлялась с использованием метода LoRA (Low-Rank Adaptation), при котором исходные веса модели остаются замороженными, а обучение производится за счет добавления малых адаптационных матриц к ключевым слоям. LoRA (адаптация с низким рангом) это метод тонкой настройки с эффективным использованием параметров, при котором предварительно обученные веса модели остаются неизменными, а в слои модели вводятся обучаемые матрицы декомпозиции ранга. Вместо того чтобы обучать все параметры модели во время тонкой настройки, LoRA декомпозирует обновления весов на более мелкие матрицы с помощью декомпозиции низкого ранга, что значительно сокращает количество обучаемых параметров без ущерба для производительности модели [2]. Суть метода заключается в том, что обучение полносвязных слоёв нейронной сети осуществляется не напрямую, а через оптимизацию матриц разложения низкого ранга, описывающих изменение весов в процессе адаптации. При этом исходные веса предобученной модели остаются неизменными. Таким образом, вместо обновления полной матрицы весов обучаются две матрицы меньшей размерности, произведение которых аппроксимирует требуемое изменение. Вместо изменения миллиардов параметров обучаются только дополнительные веса, представляющие собой малую долю от общего числа параметров модели. В случае Mistral 7B дообучается только 2.26%, что составляет 167.8 миллионов параметров [3,4]. На рисунке 2 представлено обучение матриц A и B с замороженной предварительно обученной моделью. LoRA позволяет использовать одну и ту же модель для разных задач путем замены LoRA весов, что позволяет существенно сократить объем памяти, который необходим для хранения моделей.

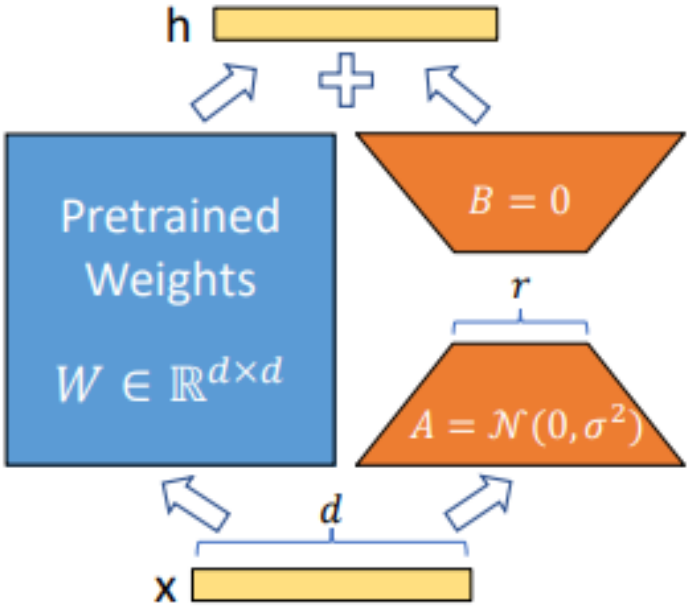

Рисунок 2. Метод LoRA

Конфигурация метода LoRA включает в себя адаптацию слоев q_proj, k_proj, v_proj и o_proj, установку параметрам r и lora_alpha значений 16 и 32 соответственно. Слои представляют собой векторные проекции схода в механизм.

- Запрос (q_proj): представляет текущий элемент, который «ищет» информацию;
- Ключ (k_proj): представляет элементы, которые могут быть найдены;
- Значение (v_proj): содержит фактическую информацию, которая агрегируется;
- Проекция (o_proj) представляет собой выходную проекцию, объединяющую результаты работы предыдущих компонентов [5].

Параметр r определяет ранг низкоранговых матриц, увеличение данного параметра увеличивает количество информации, которое будут хранить в себе адаптационные матрицы. Слишком маленькое значение параметра ограничивает возможность модели адаптироваться под поставленную задачу. Параметр lora_alpha принято брать, основываясь на параметре r*2. Данный параметр определяет коэффициент масштабирования адаптационных матриц, что позволяет регулировать влияние адаптера на исходные веса модели.

Для предотвращения переобучения параметру lora_dropout присвоено значение 0,05. Во время обучения случайная часть соединений временно исключается из расчетов, что улучшает обобщающую способность модели и снижает вероятность запоминания отдельных примеров из обучающего набора данных.

Скорость обучения определяется параметром learning_rate. Этот параметр определяет величину изменения веса модели после каждого шага оптимизации. Слишком высокое значение может привести к нестабильности обучения, в то время как слишком низкое значение значительно увеличивает время сходимости модели. Скорость обучения подбирается экспериментально.

Количество эпох обучения определяет количество полных проходов по обучающему набору данных. Недостаточное количество эпох может привести к неполному освоению предметной области, в то время как большое количество эпох увеличивает риск переобучения. Оптимальное значение было выбрано на основе анализа изменения функции потерь в процессе обучения.

Перед обучением данные проходили этап токенизации с использованием токенизатора Mistral. Токенизатор преобразует запрос, часто представляющий собой строку, в последовательность идентификаторов токенов, которые модель получает в качестве входных данных. Токеном может быть что угодно — от одного символа до части слова или символа. Модель содержит словарь токенов с соответствующими идентификаторами. Задача токенизатора — разбить строку на части и преобразовать ее в последовательность идентификаторов токенов [6]. В модели Mistral 7B используется токенизатор на основе алгоритма SentencePiece. Данный подход позволяет разбивать текст не только на отдельные слова, но и на их составные части. SentencePiece состоит из четырех основных компонентов: Normalizer, Trainer, Encoder и Decoder. Компоненты представлены на рисунке 3.

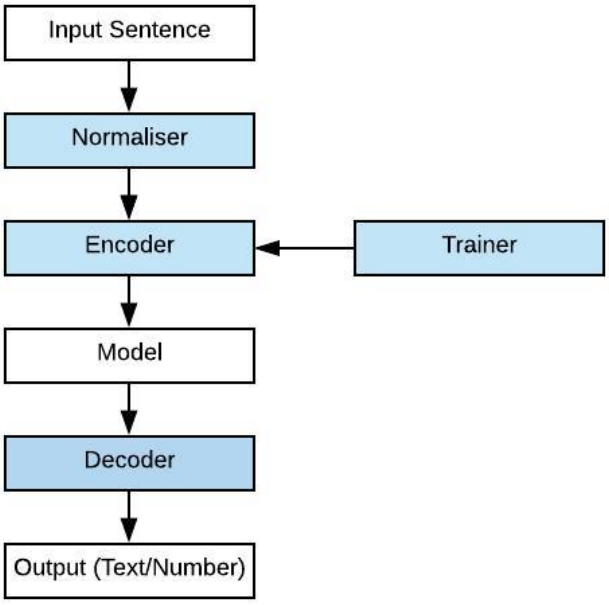

Рисунок 3. Компоненты токенизатора

Процесс дообучения модели выполнялся с использованием сервиса Google Collab и аренды графического процессора NVIDIA A100. В процессе обучения происходил сбор данных функции потерь. Данный показатель отображает соответствие предсказания модели ожидаемым ответам. На рисунке 4 представлен график функции потерь.

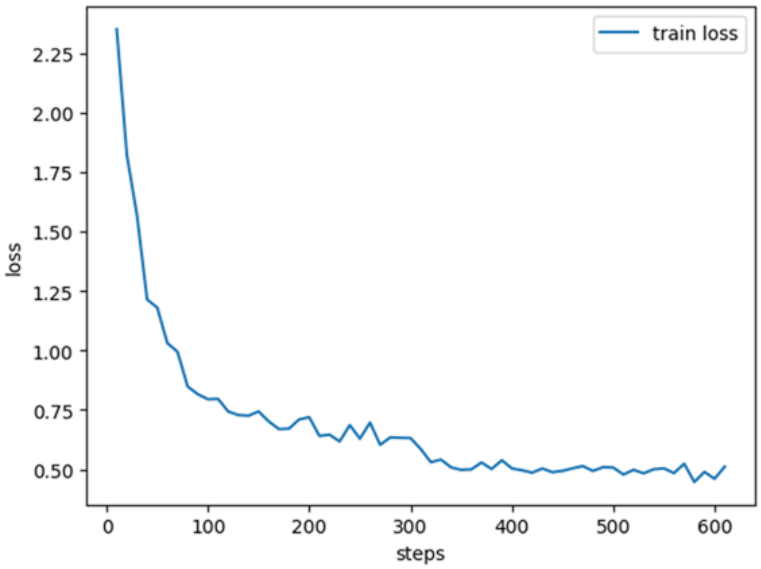

Рисунок 4. График функции потерь

В процессе обучения наблюдается устойчивое снижения значения функции потерь, что показывает адаптацию модели к новым знаниям. Продолжительность обучения составила порядка 5 часов, при этом объем используемой видеопамяти достигал 30 ГБ. После завершения был получен отдельный адаптер.

Результаты. В результате дообучения была получена специализированная версия модели, способная генерировать код на языке Fore, анализировать фрагменты программ и предлагать варианты исправления ошибок. Тестирование модели показало ее способность корректно формировать процедуры по текстовому запросу разработчика, а также выявлять и исправлять типовые ошибки в представленном коде.

Заключение. В рамках данной работы был рассмотрен процесс адаптации большой языковой модели по узкоспециализированную задачу генерации кода на языке программирования Fore. Результат работы демонстрирует, что применение метода LoRA является практичным решением для дообучения моделей под специализированные языки программирования. Решение позволит снизить время на изучение языка и отладку кода, за счет генерации типовых решений и предложений по решению ошибок на основе запроса разработчика. Полученный опыт может быть использован при дообучении больших языковых моделей для других направлений разработки [7].

Литература:

1. Пальмов С.В., Акунишникова В.В. Fine-tuning и prompt-engineering: получение качественных результатов от интеллектуальной модели // Индустриальная экономика. - 2025. - №7. - С. 54-60. URL: https://cyberleninka.ru/article/n/fine-tuning-i-prompt-engineering-poluchenie-kachestvennyh-rezultatov-ot-intellektualnoy-modeli (дата обращения: 11.02.2026).

2. Самонов Александр Валерьянович Methods for Optimizing the Training and Fine-Tuning Large Language Models // Интеллектуальные технологии на транспорте. 2024. №3 (39). URL: https://cyberleninka.ru/article/n/methods-for-

optimizing-the-training-and-fine-tuning-large-language-models (дата обращения: 12.05.2026).

3. Самонов А.В., Бурова И.О. методика разработки автоматизированных средств генерации программного кода посредством настройки больших языковых моделей // Вопросы кибербезопасности. - 2024. - №3 (61). - С. 68-75. URL: https://cyberleninka.ru/article/n/metodika-razrabotki-avtomatizirovannyh-sredstv-generatsii-programmnogo-koda-posredstvom-nastroyki-bolshih-yazykovyh-modeley (дата обращения: 11.02.2026).

4. Тимошенко, Д. Методы генерации персонализированного маркетингового контента с использованием больших языковых моделей (LLM) / Д. Тимошенко // Universum: технические науки. – 2025. – № 12-2(141). – С. 62-67. URL: https://cyberleninka.ru/article/n/metody-generatsii-personalizirovannogo-marketingovogo-kontenta-s-ispolzovaniem-bolshih-yazykovyh-modeley-llm (дата обращения: 10.02.2026)

5. Hu E. J. et al. Lora: Low-rank adaptation of large language models //Iclr. – 2022. – Т. 1. – №. 2. – С. 3. URL: https://arxiv.org/abs/2106.09685 (дата обращения: 23.05.2026)

6. Zhang, Yuqi and Di Zhang. The Structural Attention Tax: How Retrieval Format Hijacks In-Context Learning Independent of Content. (2026). URL: https://arxiv.org/abs/2606.11198 (дата обращения: 23.05.2026)

7. Елисеев В.О., Бондаренко В.И., Максимова А.Ю. Разработка интеллектуального ассистента для образовательной организации // Проблемы искусственного интеллекта. - 2025. - №4 (39). - С. 36-48. URL: https://cyberleninka.ru/article/n/razrabotka-intellektualnogo-assistenta-dlya-obrazovatelnoy-organizatsii (дата обращения: 11.02.2026).