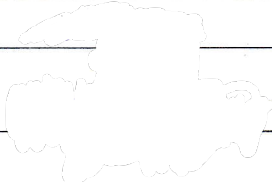
	Институт интеллектуальных кибернетических систем
	Кафедра №22 «Кибернетика»

Направление подготовки 09.03.04 Программная инженерия

Расширенное содержание пояснительной записки

к учебно-исследовательской работе студента на тему:

Разработка метода генеративного дообучения больших языковых
моделей

Группа	Б21-504		
Студент		Мандров А.П.	
		(ФИО)	
Руководитель		Трофимов А.Г.	
		(ФИО)	
Научный консультант	(подпись)		
		(ФИО)	
Оценка руководителя	15	Оценка консультанта	
	(0-15 баллов)		(0-15 баллов)
Итоговая оценка		ECTS	
	(0-100 баллов)		

Комиссия			
Председатель			
	(подпись)		(ФИО)
	(подпись)		(ФИО)
	(подпись)		(ФИО)
	(подпись)		(ФИО)

Москва 2024



Институт интеллектуальных кибернетических систем

КАФЕДРА КИБЕРНЕТИКИ

Задание на УИР

Студенту гр. Б - 21-504 Мандрову Александру Павловичу
(группа) (ф.и.о.)

ТЕМА УИР

Разработка метода генеративного дообучения больших языковых моделей

ЗАДАНИЕ

№ п/п	Содержание работы	Форма отчетности	Срок исполнения	Отметка о выполнении Дата, подпись
1.	Аналитическая часть		27.03.24	27.03.24
1.1.	Обзор существующих языковых моделей	Подраздел ПЗ	03.03.24	
1.2.	Обзор методов дообучения	Подраздел ПЗ	07.03.24	
1.3.	Обзор методов генеративного моделирования	Подраздел ПЗ	12.03.24	
1.4.	Оформление расширенного содержания пояснительной записки (РСПЗ)	Текст РСПЗ	27.03.24	
2.	Теоретическая часть		05.04.24	
2.1.	Формальная постановка задачи дообучения больших языковых моделей	Подраздел ПЗ	20.03.24	
2.2.	Разработка метода дообучения	Рабочие материалы	02.04.24	
2.3.	Разработка метода оценивания дообученной модели	Рабочие материалы	05.04.24	
3.	Инженерная часть		25.04.24	
3.1.	Выбор языка и среды программирования для реализации метода дообучения больших языковых моделей	Подраздел ПЗ	07.04.24	
3.2.	Разработка требований к программной системе дообучения больших языковых моделей	Описания требований	10.04.24	
3.3.	Программная реализация модуля дообучения больших языковых моделей	Программный код, описание программной реализации	20.04.24	
3.4.	Тестирование разработанных программных модулей	Описание тесткейсов, отчет об тестировании	25.04.24	
4.	Технологическая и практическая часть		14.05.24	
4.1.	Выбор датасета для проведения экспериментальных исследований	Описание датасета, датасет	27.04.24	

4.2.	Экспериментальные исследования разработанного метода дообучения больших языковых моделей	Диаграммы, графики, таблицы	05.05.24	
4.3.	Экспериментальное сравнение предлагаемого метода дообучения с существующими методами	Диаграммы, графики, таблицы	14.05.24	
5.	Оформление пояснительной записки (ПЗ) и иллюстративного материала для доклада.	Текст ПЗ, презентация	22.05.24	

ЛИТЕРАТУРА

1.	А. В. Брагин, А. Р. Бахтизин, В. Л. Макаров. Большие языковые модели четвёртого поколения как новый инструмент в научной работе // Искусственные общества. – 2023. – Т. 18, № 1.
2.	Jie Huang Kevin, Chen-Chuan Chang. Towards Reasoning in Large Language Models: A Survey // ACL 2023 Findings - 2023. - С. 1049-1064.
3.	Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, Yanai Elazar. Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation // Findings of the Association for Computational Linguistics: ACL 2023 - 2023. - С. 12284-12314
4.	Kunfeng Wang, Chao Gou, Yanjie Duan, Yilun Lin, Xihu Zheng, Fei-Yue Wang. Generative Adversarial Networks: Introduction and Outlook // IEEE/CAA Journal of Automatica Sinica - 2017. - С. 588-598

Дата выдачи задания:

« 26 » 02 2024г.

Руководитель

Студент

Реферат

Общий объем основного текста, без учета приложений XX страниц, с учетом приложений XX. Количество использованных источников XX. Количество приложений XX.

КЛЮЧЕВЫЕ СЛОВА: LLM, GAN, FINE TUNING, PROMPT TUNING

Целью данной работы является разработка метода генеративного дообучения больших языковых моделей.

В первой главе проведен обзор и анализ существующих методов дообучения больших языковых моделей.

Во второй главе описывают метод дообучения и метод оценивания языковой модели с теоретической стороны.

В третьей главе приводится описание программной реализации и ее тестирование.

В четвертой главе описывают исследование работы разработанного программного модуля и его сравнение с существующими аналогами.

Содержание

Введение	4
1 Обзор методов дообучения больших языковых моделей	5
1.1 Обзор больших языковых моделей	5
1.2 Обзор методов тонкой настройки	5
1.3 Обзор методов инженерии промптов	8
1.4 Обзор других методов дообучения	9
1.5 Выводы	12
1.6 Цели и задачи УИР	13
2 Разработка моделей и алгоритмов дообучения языковых моделей	14
2.1 Разработка метода дообучения	14
2.2 Разработка метода оценивания дообученной модели	14
2.3 Выводы	14
3 Программная реализация метода дообучения	15
3.1 Выбор языка и среды программирования	15
3.2 Разработка требований к программной системе	15
3.3 Программная реализация модуля дообучения больших языковых моделей	15
3.4 Тестирование разработанных программных модулей	15
3.5 Выводы	15
4 Экспериментальная проверка	16
4.1 Выбор набора данных для исследований	16
4.2 Экспериментальные исследования разработанного метода	16
4.3 Экспериментальное сравнение предлагаемого метода с существующими ме- тодами	16
4.4 Выводы	16
Заключение	17

Введение

В последнее время стали очень популярны большие языковые модели. Эти модели, благодаря своей масштабной архитектуре и обучению на огромных корпусах текста, демонстрируют впечатляющие результаты в самых разных областях, от естественного языка до компьютерного зрения. Однако, их потенциал полностью раскрывается лишь в контексте конкретных задач и предметных областей.

С учетом огромного спектра применений этих моделей: в медицине [1], в обучении программированию [2; 3], в экономике [4], в кибербезопасности [5], в поиске данных, в генерации контента, в переводе текстов - множество предприятий и исследовательских групп стремятся интегрировать их в свои проекты и продукты. Однако, обучение большой языковой модели с нуля является крайне затратным процессом, как в плане вычислительных ресурсов, так и временных затрат.

Именно поэтому приобрели большую популярность методы дообучения предварительно обученных моделей. Эти методы позволяют адаптировать существующие и предварительно обученные модели к конкретным задачам, минимизируя затраты на обучение и значительно повышая эффективность использования. Благодаря процессу дообучения, модели становятся более адаптированными и специализированными, что приводит к значительному повышению их точности и релевантности в контексте конкретных прикладных задач. Можно выделить следующие виды дообучения:

1. Тонкая настройка (FMT, LoRA, настройка префикса, настройка адаптера)
2. Инженерия промптов (zero-shot prompting, few-shot prompting, метод цепочки мыслей, метод самосогласованности)
3. Другие методы (Prompt tuning, RAG, RLHF, RLAIIF)

В настоящей работе представлен обзор современных больших языковых моделей и подходов к их дообучению. Дополнительно предоставляется описание метода дообучения языковых моделей, разработанного в рамках данной работы, а также способ его оценки. Кроме того, в рамках работы представлен программный модуль, реализующий разработанный метод. Также работа включает в себя тестирование предложенного метода и его экспериментальное сопоставление с альтернативными подходами к дообучению.

1. Обзор методов дообучения больших языковых моделей

Аннотация. В данной главе приводится обзор и обзор больших языковых моделей и существующих методов дообучения.

1.1 Обзор больших языковых моделей

Аннотация. В данном разделе приводится обзор больших языковых моделей

В настоящее время стали очень популярны большие языковые модели (Large language models - LLM). LLM - это большая нейронная сеть, обученная на огромном наборе данных, которая воспринимает текст в виде многомерных векторов - эмбедингов. Обычно эмбединги устроены так, что слова со схожими контекстными значениями или другими взаимосвязями находились близко друг к другу в векторном пространстве. Кроме того, в статьях [2; 5] выделяют следующие свойства, которыми должна обладать современная LLM:

- улучшенные возможности понимания и генерации: эти модели демонстрируют лучшее понимание контекста, грамматики и семантики, что приводит к более связному и контекстуально точному созданию текста.
- способность генерировать текст, похожий на человеческий.

К самым популярным современным LLM можно отнести следующие модели: ChatGPT и GPT-4 от OpenAI, PALM-2 и Gemini от Google, Llama-2 от Meta, Grok-1 от X.Ai, YandexGPT-2 от Yandex, GigaChat от Сбер.

1.2 Обзор методов тонкой настройки

Аннотация. В данном разделе проведен обзор методов тонкой настройки. Рассмотрены методы: полной настройки весов, адаптации низкого ранга, настройки префикса, настройки адаптера.

Тонкая настройка (fine tuning) — это класс методов дообучения LLM, при котором корректируют веса уже обученной языковой модели. Этот подход очень полезен при ограниченном количестве данных. Так как LLM уже обучена на большом объеме текстов и обладает лингвистическими знаниями, то остается адаптировать ее к нюансам решаемой задачи или предметной области. Кроме того, тонкая настройка обладает меньшей вычислительной сложностью из-за меньшего объема обучающих данных.

Метод полной настройки

К тонкой настройке относится метод полной настройка модели (full model tuning - FMT), описанный в статье [6]. При данном методе вычисляют функцию потерь на обучающей выборке, и осуществляют градиентный спуск, изменяя все веса, входящие в LLM. Проблема данного метода в том, что хоть он вычислительно менее сложен, чем обучение LLM с нуля, он все равно остается вычислительно затратным и дорогостоящим. Еще одной проблемой является возможное катастрофическое забывание. Катастрофическое забывание — это явление, когда LLM при настройке к конкретной задаче может забыть общие знания, то есть начать хуже отвечать на вопросы, не относящиеся к предметной области, к которой проводилось дообучение.

Метод эффективной настройки параметров

Также к тонкой настройке относится класс методов - эффективная настройка параметров (parameter efficient fine-tuning - PEFT). PEFT предполагает настройку небольшого количества весов, входящих в LLM. Это приводит к значительному снижению требований к памяти и вычислениям, необходимым для дообучения. Более того этот метод позволяет бороться с катастрофическим забыванием.

Адаптация низкого ранга

Одним из подходов PEFT является адаптация низкого ранга (low-rank adaptation - LoRA), описанная в статье [7]. В слоях LLM есть матрицы весов W . При дообучении вычисляются некоторые матрицы ΔW , которые прибавляют к исходным матрицам W . Обычно матрицу ΔW вычисляют с помощью градиентного спуска. Идея подхода LoRA зафиксировать веса матрицы W и обучать только матрицу ΔW . Так как матрица W и матрица ΔW имеют одинаковые размерности выигрыша от такого подхода не будет. Поэтому матрицу ΔW представляют в виде произведения двух других матриц меньшей размерности. Схематично разбиение показано на Рисунке 1.1 из статьи [7]. Такой подход позволяет обучать на порядки меньше параметров. Достаточность матриц низкого ранга обуславливается в статье [7] гипотезой, что "внутренняя размерность" (intrinsic rank) у LLM очень низкая. На практике, при гораздо меньших вычислительных затратах, LoRA имеет почти такую же эффективность, как и метод FMT.

В статье [8] описывают модификацию подхода LoRA - QLoRA. Хотя LoRA настраивает не все веса, она все равно вычислительно сложная. Поэтому в методе QLoRA используют квантование весов LLM. Обычно веса хранятся в 32-битном формате. При квантовании веса конвертируются в 4-битный формат. Идея метода состоит в том, что большинство весов языковой модели близко к нулю. При такой конвертации веса близкие к нулю будут иметь

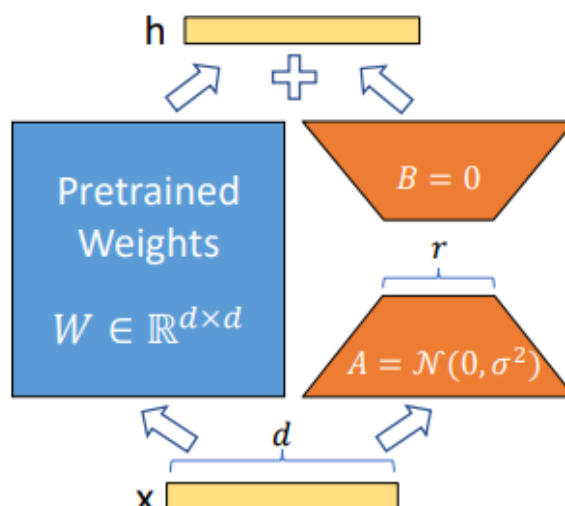


Рисунок 1.1 – Разбиение матрицы весов в методе LoRA

большую значимость. Исследования показывают, что такой метод позволяет сохранить примерную эффективность классического подхода LoRA, требуя меньших вычислительных затрат.

Настройка префикса

Другим подходом PEFT является настройка префикса (prefix tuning). Идея этого метода связана с тем, что добавляя дополнительный контекст к запросу, который потом передается LLM, можно получить более точный ответ. Поэтому к некоторым слоям языковой модели добавляют дополнительные параметры, отвечающие за контекст. Именно эти параметры и настраивают, не трогая изначальные веса языковой модели. Схематично это представлено на Рисунке 1.2 из статьи [9]. В статье [9] делают вывод, что настройка префикса показывает

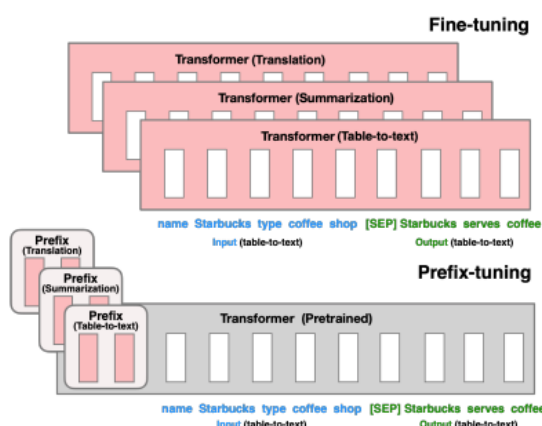


Рисунок 1.2 – Архитектура метода настройки префикса

производительность на уровне производительности при FMT.

Настройка адаптера

Другим методом, похожим на настройку префикса и относящимся к PEFT, является настройка адаптера. При этом подходе в LLM добавляются дополнительные слои, параметры которых настраивают, не трогая изначальные веса моделей. Добавляемые слои состоят из четырех компонентов. Первый компонент — это линейное преобразование, уменьшающее размерность входных данных. Уменьшая размерность, получается вектор, концентрирующийся на более значимых параметрах для предметной области. Следующий компонент применяет нелинейную функцию активации к данным. Он необходим для установления сложных взаимосвязей, связанных с предметной областью. Третий компонент восстанавливает изначальную размерность данных. Последний компонент осуществляет суммирование полученного результата и исходных данных, которые подавались на вход адаптера. Архитектура адаптера представлена на Рисунке 1.3 из статьи [10].

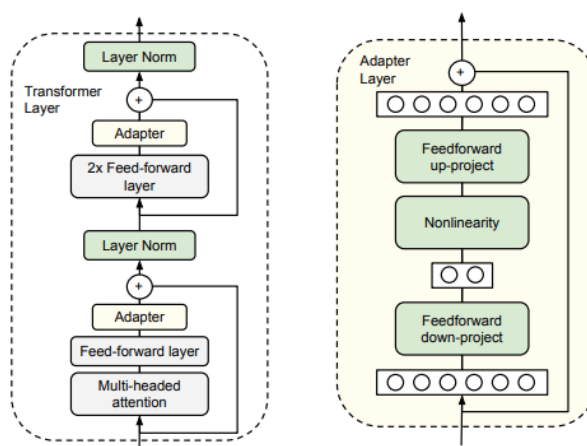


Рисунок 1.3 – Архитектура метода настройки адаптера

У всех описанных методов, относящихся к PEFT, можно выделить следующие свойства:

- Меньшую вычислительную сложность по сравнению с FMT. Однако это все равно ресурсоемкие методы.
- Преодоление катастрофического забывания, свойственного FMT.
- Большая производительность при небольшом объеме обучающих данных по сравнению с FMT.
- Сравнимая с FMT эффективность.

1.3 Обзор методов инженерии промптов

Аннотация. В данном разделе проведен обзор методов инженерии промптов. Рассмотрены методы: нулевой разметки, нескольких подсказок, цепочки мыслей, самосогласованности.

В последнее время очень популярным стал способ дообучения связанный с промптами -

инженерия промптов (prompt engineering). Промпт это запрос, который подают на вход языковой модели. В инженерии промптов никак не меняют веса языковой модели.

Техника нулевой разметки

Одной из техник инженерии промптов является техника нулевой разметки (zero-shot prompting). При этом методе языковой модели не передают никаких инструкций заранее. В этом методе подсказки для LLM подаются в виде контекста с самим промптом. Например: решай задачу шаг за шагом, классифицируй текст, как нейтральный, негативный или позитивный.

Техника нескольких подсказок

Другой техникой является техника нескольких подсказок (few-shot prompting). В этой технике в промпт добавляется несколько примеров в виде инструкций. Инструкция представляет из себя: запрос, который могли бы передать языковой модели, и ответ, который должна бы была выдать языковая модель. Используя эту технику, можно задать стиль и формат ответа LLM.

Техника цепочки мыслей и метод самосогласованности

Цепочка мыслей (Chain-of-Thought Prompting) - техника, которую применяют, когда требуется решить математическую задачу или задачу, требующую аналитических рассуждений. Этот метод использует технику нескольких подсказок, но в инструкции дополнительно указывают способ рассуждения, которым был получен ответ.

Модификацией цепочки мыслей является метод самосогласованности (Self-Consistency), предложенный в статье [11]. В отличие от цепочки мыслей в методе самосогласованности содержится больше инструкций с различными путями рассуждений. В этом методе мы несколько раз запускаем LLM, подавая каждый раз один и тот же промпт. Затем нужно выбрать наиболее встречающийся вариант из полученных ответов.

В итоге, у методов из инженерии промптов можно выделить следующие свойства: высокую эффективность, отсутствие вычислительных затрат, сильную ограниченность применения. Еще одним важным недостатком является отсутствие обоснования.

1.4 Обзор других методов дообучения

Аннотация. В данном разделе проведен обзор следующих методов: *prompt tuning*, генерация ответа с использованием результатов поиска, обучение с подкреплением на основе обратной связи от человека, обучение с подкреплением на основе обратной связи от искусственного интеллекта с использованием генеративной состязательной сети.

Prompt tuning

Prompt tuning - метод дообучения, занимающий промежуточное положение между тонкой

настройкой и инженерией промптов, принцип работы которого описан в статье [12]. В инженерии промптов использовались “жесткие” промпты: добавление примеров или подсказок в запрос, передаваемый на вход LLM. Prompt tuning работает с “мягкими” промптами: добавление в запрос последовательности символов, которая скорее всего будет бессмысленной с лингвистической точки зрения. Эта последовательность добавляется небольшой моделью, которую необходимо обучить. Этот метод похож на метод настройки префикса, но в настройке префикса добавляются дополнительные параметры к нескольким слоям LLM, а в prompt tuning добавляются параметры только перед входным слоем. Данный метод обладает большой эффективностью, но также имеет проблему с обоснованностью: неизвестно почему добавление именно таких последовательностей улучшает точность модели.

Генерация ответа с использованием результатов поиска

Другим методом с подходом, похожим на подход инженерии промптов, является генерация ответа с использованием результатов поиска (Retrieval Augmented Generation - RAG). Данный метод предполагает создание системы, работающей по следующему алгоритму:

1. Пользователь вводит вопрос
2. Система ищет подходящие документы, которые могут содержать ответ на вопрос, вводимый пользователем. Эти документы часто включают в себя собственные данные компании, которая разработала систему, а хранятся они обычно в некотором каталоге документов.
3. Составляется промпт, в котором найденные документы являются контекстом для вопроса пользователя.
4. Промпт передают в LLM

Архитектура RAG, реализующая данный алгоритм, описана в статье [13]. Данная архитектура состоит из двух компонентов: поискового компонента и генератора. В поисковом компоненте кодировщик запроса преобразует запрос от пользователя в вектор из чисел. Затем кодировщик документов преобразует каждый документ в вектор чисел и сохраняет их в поисковом индексе. На следующем шаге поисковой компонент находит в поисковом индексе документы, которые относятся к запросу пользователя. На последнем шаге поисковой компонент преобразует найденные документы обратно в текст и передает их вместе с текстом запроса генератору. Генератор объединяет текст документов с текстом запроса в один промпт и передает его LLM. Ответ языковой модели и будет ответом всей системы. В статье [13] отмечают, что при обучении модели с такой архитектурой стоит проводить совместное обучение только генератора запроса и LLM. При этом не стоит обучать кодировщик документов

по двум причинам. Во-первых, из-за затратности этого процесса. Во-вторых, без дообучения кодировщика документов можно добиться хороших результатов работы системы. Причины, по которым используют RAG:

- Языковые модели могут генерировать общую или устаревшую информацию
- Предметная область, с которой планируется работать, с течением времени быстро пополняется новой информацией
- Имеется большое количество документов, среди которых часто нужно найти ответ

Недостатками данного метода являются затраты, необходимые для поддержания хранения документов, используемых в методе, а также поддержания актуальной информации, то есть обновления документов.

Обучение с подкреплением на основе обратной связи

Еще одним методом дообучения является обучение с подкреплением на основе обратной связи от человека (Reinforcement Learning with Human Feedback – RLHF). В данном методе необходим сбор отзывов от людей о правильности и качестве ответов LLM. Полученные ответы используются в качестве сигнала для вознаграждения или наказания в обучении с подкреплением языковой модели. Метод довольно эффективный, но требует больших затрат на сбор и хранение отзывов от людей. Модификацией данного метода является обучение с подкреплением на основе обратной связи от искусственного интеллекта (Reinforcement Learning from AI Feedback - RLAIFF). Отличие состоит в том, что в RLAIFF используют отзывы, созданные некоторой моделью искусственного интеллекта. Например, в статье [14] реализуют RLAIFF с использованием генеративной состязательной сети.

Генеративная состязательная сеть

Генеративная состязательная сеть (Generative Adversarial Networks - GAN) состоит из двух нейронных сетей: генератора и дискриминатора. Изначально генератор создает данные из распределения, которое называют шумом. К примеру, в качестве шума может выступать нормальное распределение. Дискриминатор представляет собой двоичный классификатор. Его обучают на реальных данных, размеченных значением “1”, и данных, созданных генератором, размеченных значением “0”. Используя размеченные данные и значения на выходе дискриминатора, вычисляют значение функции потерь, которое используют для обучения генератора. Таким образом генератор учится лучше обманывать дискриминатор, создавая данные похожие на реальные, а дискриминатор учится отличать сгенерированные и реальные данные. Архитектура GAN представлена на Рисунке 1.4 из статьи [15].

Обоснование того, что генератор начинает создавать ”правдоподобные” данные, связано

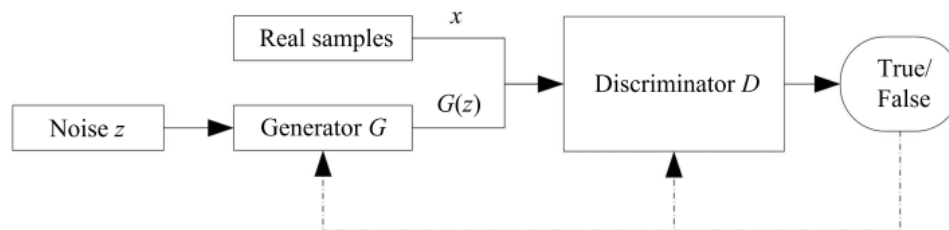


Рисунок 1.4 – Архитектура GAN

с тем, что генератор в ходе обучения меняет свое начальное распределение на распределение реальных данных. GAN обладает хорошей эффективностью, но вычислительно сложен. Также к недостаткам можно отнести следующую трудность: сложность настройки обучения. Обучение нужно настроить так, чтобы дискриминатор и генератор обучались на “равных”. Если генератор будет обучен гораздо лучше дискриминатора, то дискриминатор будет принимать все сгенерированные данные за настоящие, и соответственно дальнейшее обучение генератора не будет происходить. Если дискриминатор будет обучен гораздо лучше генератора, то это будет означать, что генератор создает нереалистичные данные.

В статье [14] используют LLM в качестве генератора и дискриминатор в качестве обратной связи для RLAIIF. В статье [14] отмечают, что необходимы дополнительные исследования, но уже можно утверждать, что данный подход имеет высокую эффективность, но обладает сложностью в правильном обучении.

1.5 Выводы

В ходе обзора существующих методов дообучения были получены следующие выводы:

1. Методы тонкой настройки обладают высокой эффективностью, но требуют больших вычислительных затрат. Кроме этого, для них необходимо наличие высококачественных обучающих данных.
2. Методы инженерии промптов обладают неплохой эффективностью и почти не требуют дополнительных данных. К недостаткам можно отнести: отсутствие обоснования методов, ограниченность применения.
3. Prompt tuning не очень вычислительно затратен, но при этом имеет большую эффективность. Но, как и методы инженерии промптов, имеет некоторые проблемы с обоснованностью.
4. К недостаткам методов RAG и RLHF относятся затраты на хранение дополнительных данных и их обновление.
5. Метод RLAIIF с использованием GAN обладает высокой эффективностью, но имеет

сложности при обучении.

1.6 Цели и задачи УИР

Поскольку prompt tuning является новым популярным методом с большой эффективностью, то подход этого метода был выбран в качестве основы для разрабатываемого метода дообучения. Prompt tuning предполагает обучение небольшой модели, которая будет преобразовывать входной запрос, который затем будет передан на вход LLM. В качестве такой модели был выбран генератор из генеративной состязательной сети, поскольку генератор и дискриминатор из GAN обладают впечатляющей эффективностью после обучения. Таким образом основной целью работы является разработка метода дообучения, основанного на подходе prompt tuning, с использованием генеративной состязательной сети. К основным задачам данной работы относятся:

1. Формальная постановка задачи дообучения LLM
2. Разработка метода дообучения
3. Разработка метода оценивания дообученной модели
4. Выбор языка и среды программирования для реализации метода дообучения
5. Программная реализация модуля дообучения
6. Тестирование разработанного модуля
7. Экспериментальные исследования разработанного метода дообучения больших языковых моделей

2. Разработка моделей и алгоритмов дообучения языковых моделей

Аннотация. В данной главе будет описан алгоритм разрабатываемого метода дообучения.

2.1 Разработка метода дообучения

Аннотация. В данном разделе будет описана архитектура метода дообучения.

2.2 Разработка метода оценивания дообученной модели

Аннотация. В данном разделе будет описан способ оценивания метода дообучения.

2.3 Выводы

Аннотация. В данном разделе будут описаны полученные выводы в ходе разработки метода дообучения.

3. Программная реализация метода дообучения

Аннотация. В данной главе будет описана программная реализация метода дообучения.

3.1 Выбор языка и среды программирования

Аннотация. В данном разделе будет описан язык и среда программирования, а также причины их выбора.

3.2 Разработка требований к программной системе

Аннотация. В данном разделе будут описаны требования к разрабатываемому методу.

3.3 Программная реализация модуля дообучения больших языковых моделей

Аннотация. В данном разделе будет приведена программная реализация модуля дообучения больших языковых моделей.

3.4 Тестирование разработанных программных модулей

Аннотация. В данном разделе будут описаны результаты тестирования разработанной модели.

3.5 Выводы

Аннотация. В данном разделе будут описаны результаты, полученные в ходе программной реализации метода дообучения.

4. Экспериментальная проверка

***Аннотация.** В данной главе будут описаны Экспериментальные исследования разработанного метода дообучения.*

4.1 Выбор набора данных для исследований

***Аннотация.** В данном разделе будет описан набор данных, выбранный для исследований, и причины выбора этого набора.*

4.2 Экспериментальные исследования разработанного метода

***Аннотация.** В данном разделе будут описаны экспериментальные исследования разработанного метода.*

4.3 Экспериментальное сравнение предлагаемого метода с существующими методами

***Аннотация.** В данном разделе будет описано экспериментальное сравнение предлагаемого метода с существующими методами.*

4.4 Выводы

***Аннотация.** В данном разделе будут описаны результаты экспериментальных исследований.*

Заключение

В заключении в тезисной форме будут представлены результаты работы. Ожидается, что будет разработан метод дообучения больших языковых моделей, и выполнены задачи, поставленные в первой главе.

Список литературы

1. Large language models in medicine / A. J. Thirunavukarasu [и др.] // Nature medicine. — 2023. — Т. 29, № 8. — С. 1930—1940.
2. Брагин А.В. Б. А. Большие языковые модели четвёртого поколения как новый инструмент в научной работе // Искусственные общества. — 2023. — Т. 18, № 1. — DOI: 10.18254/S207751800025046-9.
3. Programming Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation [Электронный ресурс] / В. А. Becker [и др.]. — 2022. — arXiv: 2212.01020 [cs.HC]. — URL: <https://arxiv.org/abs/2212.01020>.
4. Korinek A. Language Models and Cognitive Automation for Economic Research : Working Paper / National Bureau of Economic Research. — 02.2023. — № 30957. — DOI: 10.3386/w30957. — URL: <http://www.nber.org/papers/w30957>.
5. A survey on Large Language Model (LLM) security and privacy: The Good, The Bad, and The Ugly / Y. Yao [и др.] // High-Confidence Computing. — 2024. — Март. — С. 100211. — ISSN 2667-2952. — DOI: 10.1016/j.hcc.2024.100211. — URL: <http://dx.doi.org/10.1016/j.hcc.2024.100211>.
6. When Scaling Meets LLM Finetuning: The Effect of Data, Model and Finetuning Method [Электронный ресурс] / В. Zhang [и др.]. — 2024. — arXiv: 2402.17193 [cs.CL]. — URL: <https://arxiv.org/abs/2402.17193>.
7. LoRA: Low-Rank Adaptation of Large Language Models [Электронный ресурс] / E. J. Hu [и др.]. — 2021. — arXiv: 2106.09685 [cs.CL]. — URL: <https://arxiv.org/abs/2106.09685>.
8. QLoRA: Efficient Finetuning of Quantized LLMs / T. Dettmers [и др.] // Advances in Neural Information Processing Systems. Т. 36 / под ред. А. Oh [и др.]. — Curran Associates, Inc., 2023. — С. 10088—10115.
9. Li X. L., Liang P. Prefix-Tuning: Optimizing Continuous Prompts for Generation // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) /

- под ред. С. Zong [и др.]. — Online : Association for Computational Linguistics, 08.2021. — С. 4582—4597. — DOI: 10.18653/v1/2021.acl-long.353.
10. Parameter-Efficient Transfer Learning for NLP [Электронный ресурс] / N. Houlsby [и др.]. — 2019. — arXiv: 1902.00751 [cs.LG]. — URL: <https://arxiv.org/abs/1902.00751>.
 11. Self-Consistency Improves Chain of Thought Reasoning in Language Models [Электронный ресурс] / X. Wang [и др.]. — 2023. — arXiv: 2203.11171 [cs.CL]. — URL: <https://arxiv.org/abs/2203.11171>.
 12. P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks / X. Liu [и др.] // Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) / под ред. S. Muresan, P. Nakov, A. Villavicencio. — Dublin, Ireland : Association for Computational Linguistics, 05.2022. — С. 61—68. — DOI: 10.18653/v1/2022.acl-short.8. — URL: <https://aclanthology.org/2022.acl-short.8>.
 13. Retrieval-augmented generation for knowledge-intensive NLP tasks / P. Lewis [и др.] // Proceedings of the 34th International Conference on Neural Information Processing Systems. — Vancouver, BC, Canada : Curran Associates Inc., 2020. — (NIPS'20). — ISBN 9781713829546.
 14. Fine-tuning Language Models with Generative Adversarial Reward Modelling [Электронный ресурс] / Z. Z. Yu [и др.]. — 2024. — arXiv: 2305.06176 [cs.CL]. — URL: <https://arxiv.org/abs/2305.06176>.
 15. Generative adversarial networks: introduction and outlook / K. Wang [и др.] // IEEE/CAA Journal of Automatica Sinica. — 2017. — Т. 4, № 4. — С. 588—598. — DOI: 10.1109/JAS.2017.7510583.