ВВЕДЕНИЕ

Целью настоящего отчета является рассмотрение опыта, полученного в ходе производственной практики, и оценка достигнутых результатов. В период прохождения практики были поставлены следующие задачи:

1. Получить практические навыки в области разработки и анализа программного обеспечения.
2. Применить теоретические знания, полученные в процессе обучения, для решения реальных задач.
3. Изучить структуру и принципы работы современного IT-предприятия.
4. Выполнить индивидуальное задание, связанное с исследованием и разработкой в области нейронных сетей.
5. Оценить собственный уровень знаний в данной сфере и определить направления для дальнейшего профессионального развития.
6. Ознакомиться с инструментами и технологиями, используемыми специалистами в сфере разработки программного обеспечения.

Индивидуальное задание:

1. Ознакомиться и описать структуру предприятия, вид деятельности и выпускаемой продукции.
2. Изучить и описать методы разработки ПО различного функционального назначения, а также принципы защиты программных частей систем от воздействия дестабилизирующих факторов.
3. Изучить и описать ПО, используемое на предприятии для разработки сопроводительной документации и программного продукта.
4. Изучить и проанализировать требования ЕСПД, ГОСТ, ОСТ, ISO для оформления программной документации.
5. Проанализировать следующие виды документации и предоставить в отчете: техническое задание на один из видов выпускаемой продукции и графическую документацию.
6. Оформить отчет по производственной практике.

Местом прохождения практики является Центр информатизации и инновационных разработок (ЦИИР) БГУИР.

1. СТРУКТРУРА ПРЕДПРИЯТИЯ, ВИД ДЕЯТЕЛЬНОСТИ И ВЫПУСКАЕМАЯ ПРОДУКЦИЯ

Центр информатизации и инновационных разработок (ЦИИР) является структурным подразделением Учреждения образования «Белорусский государственный университет информатики и радиоэлектроники» (БГУИР). Он был создан на основании приказа ректора №69 от 20 февраля 2001 года.

Основные направления деятельности:

1. Поддержка и координация разработки, внедрения и применения современных информационных систем и технологий в сфере образования.
2. Разработка и реализация Концепции интегрированной информационной системы БГУИR.
3. Информационно-аналитическое обеспечение учебной, научной, административно-хозяйственной и других видов деятельности университета.

ЦИИР имеет многоуровневую структуру, включающую руководство и ряд специализированных отделов: Отдел информационных технологий, Отдел интегрированных автоматизированных систем управления, Отдел инновационных разработок в сфере образования, Отдел сетевых технологий, Отдел системного и технического обслуживания и др.

ЦИИР разрабатывает и поддерживает широкий спектр программных продуктов, включая: Интегрированную информационную систему (ИИС) БГУИР, систему "Электронный Абитуриент", веб-портал университета, мобильные приложения и Telegram-боты.

2. МЕТОДЫ РАЗРАБОТКИ И ПРИНЦИПЫ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Методы разработки программного обеспечения

В современной разработке применяются различные методологии, такие как каскадная модель, гибкие (Agile) подходы (Scrum, Kanban) и спиральная модель. В ходе практики использовались принципы Agile, что позволило гибко подходить к задачам исследования и оперативно вносить изменения в архитектуру.

2.2 Принципы защиты программных частей систем

Защита ПО — это комплекс мер для предотвращения несанкционированного доступа и модификации. Ключевые принципы включают:

Защита по глубине: Многоуровневая система безопасности.

Управление доступом: Принцип наименьших привилегий.

Безопасная разработка (DevSecOps): Интеграция безопасности на всех этапах жизненного цикла ПО.

3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ НА ПРЕДПРИЯТИИ

3.1 ПО для разработки сопроводительной документации

Используются текстовые редакторы (MS Word, Google Docs), системы управления документацией (Confluence), инструменты для создания диаграмм (diagrams.net) и генераторы документации из кода (Swagger, Javadoc).

3.2 ПО для разработки программного продукта

Стек технологий включает IDE (VS Code, JetBrains IDEs), систему контроля версий Git, СУБД (PostgreSQL, MySQL), средства контейнеризации (Docker), а также различные фреймворки и библиотеки (React, PyTorch, Hugging Face Transformers).

4. АНАЛИЗ ТРЕБОВАНИЙ ЕСПД, ГОСТ, ISO К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Стандартизация документации (ГОСТ серий 19 и 34, ISO/IEC 26514) необходима для унификации и обеспечения качества продуктов. Эти стандарты регламентируют виды, состав и содержание программных документов, обеспечивая их полноту, точность и доступность для целевой аудитории.

5. АНАЛИЗ ВИДОВ ДОКУМЕНТАЦИИ

5.1 Техническое задание на разработку веб-сервиса "LLM Comparator"

1. Общие сведения

Наименование продукта: Веб-сервис для сравнительного анализа производительности больших языковых моделей "LLM Comparator".

Заказчик: Руководитель практики ЦИИР БГУИР.

Разработчик: Студент-практикант.

2. Назначение и цели разработки

Назначение: Сервис предназначен для проведения сравнительного анализа различных больших языковых моделей (LLM) по ключевым метрикам производительности и качества.

Цели:

Создать платформу для одновременного тестирования моделей Mistral, Llama и других с параметрами от 8 до 24 млрд.

Реализовать REST API для программного взаимодействия с моделями.

Разработать веб-интерфейс для наглядной демонстрации и ручного тестирования.

Исследовать влияние дообучения (с помощью адаптеров) на точность генерации структурированных данных (JSON).

3. Требования к системе

3.1. Функциональные требования:

Сервис должен предоставлять REST API с конечной точкой /api/generate, принимающей POST-запросы.

Тело запроса должно содержать поля: model\_name (название модели для использования) и prompt (входной текст).

Сервис должен возвращать ответ в формате JSON, содержащий сгенерированный текст, время ответа и пиковое потребление памяти (GPU/RAM).

Веб-интерфейс должен позволять пользователю выбрать модель из списка, ввести текстовый запрос и отобразить полученный результат.

Система должна поддерживать загрузку и использование дообученных LoRA-адаптеров для повышения точности генерации JSON.

3.2. Требования к производительности:

Среднее время ответа API для моделей до 13 млрд параметров не должно превышать 5 секунд на стандартном запросе.

Сервис должен быть способен обрабатывать до 5 одновременных запросов без существенной деградации производительности.

3.3. Требования к надежности:

Сервис должен корректно обрабатывать ошибки, связанные с недоступностью моделей или неверными входными данными, и возвращать информативные сообщения об ошибках.

4. Требования к программной документации

Необходимо предоставить описание REST API (например, в формате Swagger/OpenAPI).

Подготовить краткое руководство пользователя для веб-интерфейса.

5. Стадии и этапы разработки

Проектирование архитектуры и API (3 дня).

Настройка окружения и базового функционала (5 дней).

Реализация веб-интерфейса (4 дня).

Дообучение и интеграция адаптеров (5 дней).

Тестирование и отладка (3 дня).

Подготовка отчета (2 дня).

6. Порядок контроля и приемки

Приемка осуществляется путем демонстрации работающего сервиса, который соответствует всем функциональным требованиям, и предоставления отчета по практике.

5.2 Графическая документация

Графическая документация используется для визуального представления архитектуры и логики работы системы. Ключевым элементом является диаграмма последовательности (sequence-диаграмма), которая показывает взаимодействие между компонентами системы во времени.

Диаграмма последовательности (Sequence Diagram)

[ Здесь должно быть изображение диаграммы последовательности ]

Словесное описание диаграммы последовательности:

На диаграмме представлены основные участники (акторы и компоненты системы): User (пользователь), Web Interface (веб-интерфейс в браузере), REST API Gateway (серверная часть, принимающая запросы) и LLM Service (микросервис, инкапсулирующий логику работы с конкретной нейронной моделью).

Процесс взаимодействия выглядит следующим образом:

sendRequest(prompt, model): Пользователь (User) вводит текстовый запрос и выбирает модель в веб-интерфейсе, после чего инициирует отправку запроса.

POST /api/generate: Web Interface формирует и отправляет асинхронный HTTP POST-запрос на сервер (REST API Gateway).

validateAndForward(request): REST API Gateway получает запрос, проверяет его корректность (валидация) и на основе параметра model\_name перенаправляет его в соответствующий LLM Service (например, сервис с моделью Mistral).

process(prompt): LLM Service обрабатывает запрос, выполняя генерацию текста с помощью загруженной нейронной сети. В этот момент также производится замер метрик производительности (время, память).

return generatedJSON: После завершения генерации LLM Service возвращает результат (JSON-объект с текстом и метриками) шлюзу REST API Gateway.

return 200 OK (response): REST API Gateway отправляет успешный HTTP-ответ с полученными данными обратно в Web Interface.

displayResult(data): Наконец, Web Interface получает данные и отображает сгенерированный ответ и метрики производительности пользователю (User).

Эта диаграмма наглядно демонстрирует разделение ответственности между клиентской частью, бэкендом и вычислительным сервисом, что является характерной чертой современных веб-приложений.

ОСНОВНЫЕ ЭТАПЫ ПРАКТИКИ:

Продумывание архитектуры решения: Был спроектирован сервис с микросервисной архитектурой, REST API и веб-интерфейсом.

Разработка плана, просчет мощностей: Составлен план работ, оценены требования к аппаратному обеспечению (в первую очередь, к объему видеопамяти).

Согласование: Архитектура и план работ согласованы с руководителем практики.

Изучение методов оптимизации сетей и стандартов: Изучены методы квантизации и дообучения с помощью адаптеров (LoRA).

Дообучение с использованием адаптеров для повышения точности: Для решения проблемы генерации невалидного JSON был применен метод дообучения с адаптерами LoRA.

Принцип LoRA: Этот метод относится к техникам Parameter-Efficient Fine-Tuning (PEFT) и позволяет дообучить модель под конкретную задачу, тренируя лишь небольшое количество дополнительных параметров.

Процесс: Был создан набор данных с примерами "запрос -> валидный JSON". На этих данных была обучена базовая модель с конфигурацией LoRA.

Результат: После применения обученного адаптера точность генерации валидного JSON увеличилась с ~75% до более чем 98%.

Частичное разворачивание и тестирование системы: Реализованы ключевые компоненты, включая API, и проведено тестирование производительности и качества моделей.

Написание и оформление отчета: По результатам работы составлен данный отчет.

ЗАКЛЮЧЕНИЕ

Производственная практика в Центре информатизации и инновационных разработок БГУИР позволила закрепить теоретические знания и получить ценный практический опыт в области разработки ПО и машинного обучения. Выполнение индивидуального задания позволило углубить знания в области LLM, освоить инструменты для их дообучения (PEFT/LoRA), а также получить навыки в разработке REST API и веб-интерфейсов. Анализ требований стандартов к документации (ГОСТ, ЕСПД) дал понимание важности качественного документирования. Полученный опыт является важным этапом в профессиональном становлении и определяет дальнейшие направления для развития в сфере информационных технологий.