МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОЙ ОБРАБОТКИ ЭКСПОРТА ЧАТОВ TELEGRAM С ИСПОЛЬЗОВАНИЕМ GEMINI API И ЛОКАЛЬНЫХ МОДЕЛЕЙ ЧЕРЕЗ ПЛАТФОРМУ OLLAMA

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Екатеринушкина Константина Игоревича

Научный руководитель
старший преподаватель _____ П. О. Дмитриев

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2025

ВВЕДЕНИЕ

В современном цифровом мире мессенджеры, в частности Telegram, стали хранилищем огромных объемов текстовой информации, содержащей ценные сведения, договоренности и идеи. Возможность экспорта истории чатов в формате HTML открывает перспективы для анализа этих данных, однако ручная обработка таких выгрузок крайне трудоемка и неэффективна. Стремительное развитие технологий искусственного интеллекта, особенно больших языковых моделей (LLM), таких как Gemini API от Google и локальные модели, доступные через платформу Ollama, предоставляет мощные инструменты для автоматизированной обработки и анализа текстовых данных. Интеграция этих возможностей позволяет извлекать смысл, суммировать информацию и отвечать на вопросы по предоставленному тексту, значительно повышая эффективность работы с накопленной информацией. Возможность использования локальных моделей обеспечивает высокий уровень приватности при работе с конфиденциальными данными, что особенно актуально.

Актуальность темы заключается в необходимости создания эффективного и конфиденциального инструмента для автоматизации анализа обширных текстовых данных, генерируемых в переписках Telegram, с использованием передовых технологий LLM.

Цель работы: разработка приложения на Godot Engine, предназначенного для автоматизированной обработки HTML-экспорта чатов Telegram, формирования на их основе промтов и взаимодействия с ИИ-моделями (Gemini API и локальные модели, доступные через платформу Ollama) для получения осмысленных результатов анализа текстовых данных.

Для достижения данной цели были поставлены следующие задачи:

— Проанализировать структуру и особенности HTML-формата экспорта чатов Telegram для определения методов извлечения информации.
— Исследовать возможности и интерфейсы взаимодействия с Gemini API и платформой Ollama для обработки текстовых данных с помощью больших языковых моделей.
— Обосновать выбор Godot Engine и языка GDScript в качестве инструментов для разработки приложения.
— Разработать архитектуру приложения, включающую модули для парсинга HTML, формирования промтов, взаимодействия с ИИ-моделями

и пользовательского интерфейса.

— Реализовать ключевые модули приложения: парсер HTML-файлов, модуль формирования промтов, модули интеграции с Gemini API и Ollama.
— Разработать интуитивно понятный пользовательский интерфейс на Godot Engine.
— Провести тестирование разработанного приложения на реальных данных экспорта чатов, включая оценку корректности парсинга, качества ответов ИИ-моделей и производительности.

В работе были исследованы спецификации HTML-формата экспорта чатов Telegram, документация API больших языковых моделей Gemini и платформы Ollama, а также особенности языка GDScript и движка Godot Engine для разработки приложений.

Дипломная работа состоит из введения, трех глав, заключения, списка использованных источников и приложений.

— Глава 1. Теоретические основы и анализ предметной области.
— Глава 2. Проектирование и разработка приложения.
— Глава 3. Тестирование и анализ результатов.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе

Данный раздел посвящен фундаментальным аспектам, необходимым для разработки приложения.

— Анализ формата HTML-экспорта чатов Telegram: была детально проанализирована структура HTML-файлов, генерируемых Telegram при экспорте истории чатов. Выявлено, что каждое сообщение представлено как последовательность HTML-блоков с уникальными идентификаторами (ID), временными метками и именами отправителей. Особое внимание уделено обработке медиа-контента (фото, видео, голосовые сообщения, файлы), который для унификации ввода и фокусировки на текстовом анализе заменяется соответствующими текстовыми заглушками (например, [Photo], [Video]). Подчеркнута критическая важность очистки извлеченного текста от HTML-тегов форматирования (например, ,), ссылок (<a>) и HTML-сущностей, что обеспечивает получение «чистого» текста, пригодного для анализа большими языковыми моделями (LLM).

— Исследование искусственного интеллекта и больших языковых моделей: рассмотрены ключевые принципы работы LLM, включая их базовую архитектуру Трансформера с механизмом внимания, этапы предварительного обучения на огромных текстовых корпусах, процессы токенизации и встраивания (эмбеддингов), а также тонкую настройку для специфических задач. Выделены Gemini API от Google DeepMind как представитель облачных LLM, подробно описана структура его JSON-запросов (включая contents, safetySettings, generationConfig) и формат ответов. В качестве альтернативы исследована платформа Ollama, позволяющая запускать LLM локально на компьютере пользователя. Отмечены её значимые преимущества: приватность данных (обработка на устройстве пользователя), отсутствие затрат на API, автономность и гибкость выбора из широкого спектра доступных моделей.

— Ограничения и этические аспекты LLM: обсуждены фундаментальные ограничения LLM, такие как склонность к «галлюцинациям» (генерации неправдивой информации), лимиты контекстного окна, чувствительность к формулировке промтов, отсутствие истинного «здравого

смысла» и потенциальная неактуальность знаний. Подробно рассмотрены этические риски, в частности вопросы приватности данных при использовании облачных LLM и предвзятости, обусловленной данными обучения. Для минимизации этих рисков в проекте реализована возможность использования локальных моделей через Ollama, механизм блочной обработки данных для обхода ограничений контекстного окна и возможность задания явных системных подсказок для управления поведением модели.

— Обоснование выбора Godot Engine и GDScript: детально аргументирован выбор игрового движка Godot Engine и его встроенного языка GDScript для разработки приложения, не являющегося игрой. Ключевыми факторами послужили мощная и гибкая система создания графических пользовательских интерфейсов (UI) на основе узлов Control, наличие встроенных средств для работы с файловой системой и выполнения сетевых запросов, открытый исходный код, активное сообщество и интегрированная среда разработки. Проведено сравнение с альтернативными решениями, такими как Python с UI-библиотеками (Tkinter, PyQt) и веб-технологии (Electron), показав, что Godot обеспечивает более легковесный, нативный и интегрированный подход для данной задачи.

— Методы извлечения информации из текстовых данных: рассмотрены как традиционные методы извлечения информации (регулярные выражения, парсинг на основе грамматик и правил, словарные методы) с их преимуществами и недостатками, так и современные подходы с использованием LLM. Подчеркнуто, что для глубокого смыслового анализа и извлечения неструктурированной информации из чатов предпочтение отдается LLM благодаря их гибкости и способности понимать сложный контекст через формулирование промтов.

Во втором разделе

Этот раздел описывает архитектуру и детали реализации каждого ключевого модуля приложения.

— Архитектура приложения: разработанное приложение имеет модульную архитектуру, построенную на Godot Engine 4.4.1 и GDScript. Приложение ориентировано на последовательное выполнение задач: загруз-

ка и парсинг HTML-экспорта, формирование промта, взаимодействие с ИИ-моделями и отображение результатов. Определены ключевые модули: Модуль Пользовательского Интерфейса (UI), Модуль Управления и Отображения (Основной скрипт, выступающий координатором), Модуль Парсинга HTML-экспорта, Модуль Формирования Промтов (интегрированный в функции вызова API), Модуль Взаимодействия с Gemini API и Модуль Взаимодействия с Ollama. Такое разделение обеспечивает четкое разграничение ответственности и упрощает потенциальные модификации.

— Реализация парсера HTML-экспорта: центральный модуль предобработки данных, реализованный с использованием регулярных выражений (RegEx) для идентификации и извлечения данных из HTML-кода. Основные этапы включают: однократную инициализацию RegEx объектов для повышения производительности; поиск, чтение и сортировку HTML-файлов в выбранной директории с использованием DirAccess и FileAccess; извлечение названия чата; основной итеративный цикл парсинга строк HTML-кода. В этом цикле происходит идентификация блоков сообщений (новых и «присоединенных»), создание MessageData для временного хранения атрибутов (ID, отправитель, время, текст, информация о пересылке/ответе). Особая логика предусмотрена для обработки пересланных сообщений (FWD), ответов (Reply) и замещения медиа-контента текстовыми заглушками, а также для очистки текста от HTML-тегов и сущностей. Итоговый вывод формируется в структурированный массив сообщений, готовый для последующего анализа LLM.

— Алгоритмы формирования промтов: процесс формирования промтов является ключевым для получения релевантных ответов от LLM. Реализованы два основных режима: обработка всего текста чата целиком (для небольших чатов) и итеративная блочная обработка (для очень больших объемов данных). В блочном режиме ответ от LLM на предыдущий блок данных используется как «контекстная часть» для обработки последующего блока, что помогает модели сохранять связность. Промт состоит из системной подсказки (задаваемой пользователем для определения общего контекста/роли LLM), пользовательского запро-

са, полного текста обработанного чата (или текущего блока в блочном режиме) и контекста предыдущих итераций. Это базовый подход (zero-shot prompting с контекстом), выбранный для обеспечения быстродействия и универсальности.

— Интеграция с Gemini API: реализована посредством узла HTTPRequest для асинхронной отправки HTTP POST-запросов. Процесс включает: получение API-ключа, имени модели и лимита токенов из глобальных переменных; формирование URL-запроса; создание JSON-тела запроса, содержащего промт, настройки безопасности (safetySettings) и параметры генерации (generationConfig); отправку запроса. Обработка ответа включает парсинг JSON-тела, извлечение сгенерированного текста (model_answer) и детальную обработку возможных ошибок, включая блокировки по безопасности и сетевые проблемы.

— Интеграция с Ollama: также использует узел Godot HTTPRequest для POST-запросов к локальному серверу Ollama. Процесс аналогичен интеграции с Gemini: получение URL локального сервера Ollama и имени локальной модели; формирование JSON-тела запроса (с model, prompt и stream: false для получения полного ответа); отправка запроса. Обработка ответа включает парсинг JSON и извлечение сгенерированного текста из поля response. Интеграция с Ollama проще благодаря её унифицированному REST API.

— Разработка пользовательского интерфейса в Godot Engine: пользовательский интерфейс (UI) разработан с использованием встроенной системы узлов Control Godot Engine. Основные цели проектирования UI: интуитивность, четкий рабочий процесс (выбор данных -> парсинг -> формирование запроса -> просмотр результата), минималистичный дизайн и конфигурируемость. Структура UI организована в виде нескольких «сцен» (Основное меню, Настройки, Информация, Обработка, Окно отправки запросов к ИИ), между которыми пользователь может переключаться. Предусмотрены элементы для выбора директории, запуска парсинга, копирования/сохранения результатов, ввода промтов, выбора режима работы LLM (API/Локальная, цельный/блочный запрос) и настройки параметров.

В третьем разделе

В данном разделе представлены результаты тестирования разработанного приложения и анализ его эффективности.

— Методика тестирования: тестирование проводилось по трем основным направлениям: проверка корректности работы парсера HTML-экспорта, оценка успешности интеграции с API LLM (Gemini и Ollama) и качественный анализ ответов моделей.

— Тестирование парсера HTML-экспорта:
  – Корректность: проверка на небольшом тестовом чате, содержащем разнообразные типы сообщений (текст, стикеры, фото, голосовые сообщения, видео), подтвердила корректное срабатывание паттернов парсера и преобразование медиа-контента в текстовые заглушки, обеспечивая правильное формирование промта.
  – Производительность: для оценки скорости обработки больших объемов данных был выгружен реальный личный чат за три года, состоящий из 51 HTML-файла, что суммарно составило 3 691 882 символов. Результаты показали высокую скорость обработки: на ПК с 6-ядерным CPU, 8 ГБ RAM и 8 ГБ GPU парсинг занял 12.53 секунды, а на ПК с 16-ядерным CPU, 32 ГБ RAM и 26 ГБ GPU — всего 1.16 секунды. Это демонстрирует эффективность парсера, способного быстро обрабатывать значительные объемы текстовых данных, что важно для последующего анализа LLM.

— Тестирование отправки запросов:
  – Настройка: для проверки взаимодействия с LLM была использована выгрузка Telegram-канала «Forbes Russia» за определенный период, общим объемом 1 325 657 символов. В качестве тестовых моделей были выбраны gemini-2.0-flash (облачная, контекст 1 048 576 токенов) для API-запросов и gemma3:12b (локальная, контекст 128 000 токенов) для Ollama. Для gemma3:12b была применена блочная обработка с разбиением на блоки по 300 000 символов из-за ограничения контекстного окна.
  – Результаты: обеим моделям был отправлен запрос на анализ экономического канала с целью извлечения информации о курсе доллара к рублю и прогнозирования. Модели успешно выделили идентичные даты и значения курса, которые совпадали с официаль-

ным графиком. В части прогнозов на май 2025 года gemini-2.0-flash предоставила более точный прогноз, тогда как gemma3:12b предложила несколько сценариев, базовый из которых также оказался весьма близким к реальной ситуации.

— Анализ результатов: тестирование подтвердило корректность работы парсера HTML-экспорта и успешную интеграцию с обоими типами LLM. Производительность парсинга напрямую зависит от вычислительной мощности ПК, но остается быстрой даже на относительно слабых системах. Взаимодействие с LLM происходит корректно, и модели адекватно обрабатывают подготовленный текст, включая блочные запросы. Выявлены особенности работы с локальными моделями, такие как необходимость учета их контекстного окна и потенциальные проблемы с языком вывода (например, вывод на английском или китайском для некоторых моделей), что требует явного указания желаемого языка в системной подсказке. Отмечено, что для пользователей со слабыми ПК более предпочтительным является использование облачных API-моделей, так как они не требуют значительных локальных вычислительных ресурсов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной дипломной работы была успешно решена поставленная задача по разработке и реализации приложения на Godot Engine, предназначенного для интеллектуальной обработки экспорта чатов Telegram.

Практическая значимость разработанного приложения заключается в предоставлении пользователям удобного инструмента для автоматизированного анализа больших объемов текстовой информации из чатов Telegram. Это может быть полезно для суммирования обсуждений, поиска ключевой информации и ответов на вопросы по содержимому переписок как для личного, так и для профессионального использования. Возможность работы с локальными моделями через Ollama дополнительно повышает ценность приложения для пользователей, заботящихся о приватности своих данных.

В ходе выполнения данной работы были получены следующие результаты:

— Проведен анализ предметной области, включая формат Telegram экспорта и принципы работы LLM, а также их ограничения и этические аспекты.
— Обоснован выбор технологического стека (Godot Engine, GDScript) на основе требований задачи и сравнения с альтернативами.
— Разработана модульная архитектура приложения, обеспечивающая разделение функционала и гибкость.
— Реализованы ключевые модули: парсер HTML, модуль формирования промтов, модули интеграции с Gemini API и Ollama.
— Разработан интуитивно понятный пользовательский интерфейс.
— Проведено тестирование, подтвердившее корректность работы парсера, успешность интеграции с ИИ-моделями и выявившее особенности работы с различными LLM.

Таким образом, поставленные цель и задачи выполнены в полном объеме. Приложение демонстрирует жизнеспособность выбранного подхода и предоставляет основу для дальнейшего развития и расширения функционала.

Основные источники информации:

1 Экспорт чата в телеграм. [Электронный ресурс] URL: https://blog.eldorado.ru/publications/kak-sdelat-rezervnuyu-kopiyu-telegram-i-

sokhranit-perepisku-42182. (дата обращения: 20.03.2025)
2 Что такое LLM. [Электронный ресурс] URL: https://aws.amazon.com/ru/what-is/large-language-model/. (дата обращения: 20.03.2025)
3 Gemini. [Электронный ресурс] URL: https://deepmind.google/technologies/gemini/. (дата обращения: 30.03.2025)
4 Документация к Gemini API. [Электронный ресурс] URL: https://ai.google.dev/gemini-api/docs. (дата обращения: 30.03.2025)
5 Ollama. [Электронный ресурс] URL: https://ollama.com/. (дата обращения: 01.04.2025)
6 Godot Engine. [Электронный ресурс] URL: https://godotengine.org/. (дата обращения: 01.04.2025)
7 GDScripts документация. [Электронный ресурс] URL: https://docs.godotengine.org/en/stable/index.html. (дата обращения: 01.04.2025)
8 Регулярные выражения. [Электронный ресурс] URL: https://habr.com/ru/articles/545150/. (дата обращения: 03.04.2025)
9 zero-shot promting. [Электронный ресурс] URL: https://www.promptingguide.ai/techniques/zeroshot. (дата обращения: 10.04.2025)
10 Модели gemma3. [Электронный ресурс] URL: https://ollama.com/library/gemma3. (дата обращения: 05.05.2025).
11 Модели deepseek-r1. [Электронный ресурс] URL: https://ollama.com/library/deepseek-r1. (дата обращения: 08.05.2025).