УДК 004.89

Козлов Андрей Игоревич, Бакалавр, Калужский государственный университет им. К. Э. Циолковского, Россия, г. Калуга

ПРОЕКТИРОВАНИЕ И АНАЛИЗ АРХИТЕКТУРЫ ИНТЕЛЛЕКТУАЛЬНОГО TELEGRAM-БОТА ДЛЯ АВТОМАТИЗИРОВАННОГО СТИЛИСТИЧЕСКОГО ОФОРМЛЕНИЯ ТЕКСТА

Аннотация

Рассмотрена архитектура Telegram-бота «Textemojiai», предназначенного для оформления готовых публикаций. Система сохраняет исходные формулировки, порядок абзацев и пользовательскую разметку. Языковая модель определяет, где уместен эмодзи, какой тип символа подходит и какие фрагменты можно выделить. Итоговое сообщение формируется локальными программными модулями. Такое разделение позволяет контролировать Telegram entities, деление длинных сообщений и выбор custom emoji из внутреннего каталога. Прототип реализован на Python с использованием aiogram и внешних языковых моделей. Для проверки применялись автоматические тесты, Ruff, Mypy и чтение локальных баз данных. Все 22 предусмотренных теста пройдены; статический анализ и проверка типов не выявили замечаний. Полученные результаты относятся к локальной однопроцессной версии и не заменяют пользовательские и нагрузочные испытания.

Ключевые слова: Telegram-бот, языковая модель, программная архитектура, обработка текста, эмодзи, aiogram, SQLite.

Annotation

The architecture of «Textemojiai», a Telegram bot for styling ready-made posts, is examined. The system preserves the original wording, paragraph order, and user markup. A language model decides where an emoji is appropriate, which symbol category should be used, and which text fragments may be emphasized. The final message is assembled by local software modules. This separation keeps control over

Telegram entities, long-message splitting, and custom emoji selection from an internal catalog. The prototype is implemented in Python with aiogram and external language models. The implementation was checked with automated tests, Ruff, Mypy, and local database reading. All 22 available tests passed, while static analysis and type checking reported no issues. The results refer to the current local single-process version and do not replace user or load testing.
Keywords: Telegram bot, language model, software architecture, text processing, emoji, aiogram, SQLite.

Введение

Авторы Telegram-каналов нередко готовят текст самостоятельно, а затем отдельно расставляют эмодзи, выделения и маркеры списков. Обычные шаблоны решают эту задачу лишь частично: оформление зависит от смысла абзаца, типа публикации и выбранного визуального стиля.

Языковые модели уже применяются для редактирования текста по инструкции пользователя [2-4]. Но для оформления готового поста полный пересказ не нужен. Пользователь ожидает, что его формулировки и порядок мыслей сохранятся. Поэтому модели следует поручать ограниченный набор решений, а техническую сборку сообщения выполнять программно.

Telegram предоставляет готовый интерфейс с командами, кнопками и разметкой. При этом после добавления новых символов меняются позиции фрагментов, к которым относятся Telegram entities. Длинный текст приходится делить на несколько сообщений без потери выделений. Эти особенности требуют разделить смысловой анализ и локальное формирование результата.

Цель работы - спроектировать и проанализировать архитектуру Telegram-бота для стилистического оформления готового текста. Объект исследования - программная система обработки сообщений в Telegram. Предмет исследования - архитектурные решения, связывающие обработчики мессенджера, внешние языковые модели и локальные модули форматирования. Использованы функциональный анализ, декомпозиция и техническая проверка кода.

1. Назначение и границы системы

Проект «Textemojiai» оформляет уже подготовленные посты и не создаёт материал с нуля. Бот подбирает тематические эмодзи, предлагает жирное и курсивное выделение, оформляет списки и сохраняет деление на абзацы. Пользователь отправляет текст, выбирает режим и получает результат в том же диалоге. После этого можно запросить другой вариант или дать короткое уточнение.

Для части функций используется профиль с маркерами и добровольно переданными примерами авторского оформления. Обычные посты постоянно не сохраняются. Для Telegram Premium доступны custom emoji и согласованные наборы символов. Граница системы задана явно: бот отвечает за визуальное оформление, но не проверяет факты и не переписывает публикацию целиком.

2. Архитектура и путь сообщения

Архитектура построена как последовательность модулей. Telegram передаёт событие обработчикам aiogram, затем текст проходит предварительный разбор. Отдельный клиент обращается к языковой модели, а полученные решения применяются локально. Готовое сообщение возвращается пользователю через Bot API.

В проекте выделены четыре группы компонентов: интерфейс Telegram и маршрутизация; подготовка запроса и разбор ответа модели; работа с текстом, списками и entities; хранение профилей, каталогов, лимитов и журнала. Такое разделение снижает связанность: обработчики не зависят от конкретного поставщика модели, а внешняя модель не управляет внутренними идентификаторами custom emoji. Общая последовательность показана на рисунке 1.

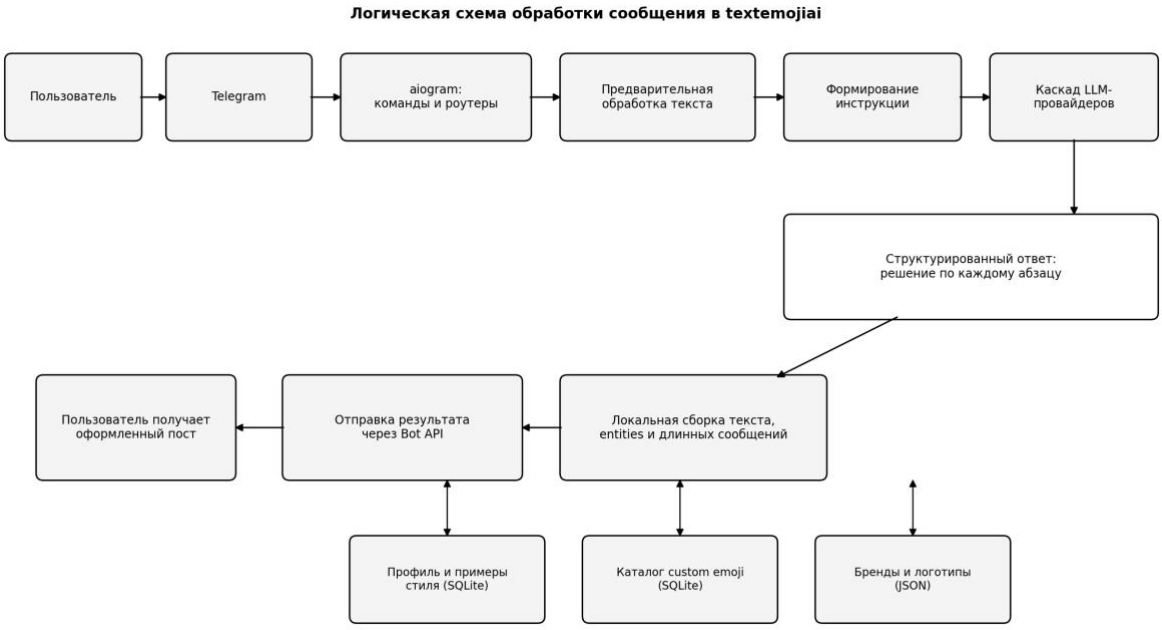

Рисунок 1 - Логическая схема обработки сообщения в «Textemojiai»

2.1. Приём и предварительная обработка

Приложение реализовано на Python с использованием aiogram 3.x. Новые события бот получает методом long polling [1, 7]. После получения сообщения обработчик определяет его тип, извлекает текст и делит его на абзацы, пустые строки и строки списков.

Краткое указание вида «стиль: деловой» отделяется до обращения к модели и не попадает в итоговый текст. Уже созданные выделения Telegram сохраняются отдельно. После вставки эмодзи их позиции рассчитываются заново. Строки списков также отмечаются заранее, чтобы все пункты получили единое оформление.

2.2. Инструкция для языковой модели

Модуль формирования запроса задаёт узкую задачу: сохранить смысл и порядок абзацев, не добавлять факты и не вставлять эмодзи внутрь предложения. Для каждого абзаца модель возвращает структурированное решение: нужен ли символ, где его поставить и какие точные фрагменты можно выделить.

Структурированный ответ проверяется до изменения исходного текста. Локальный модуль ищет указанные фрагменты и только затем создаёт новые entities. Если модель возвращает неизвестный фрагмент или нарушает

ожидаемую структуру ответа, программа отклоняет результат и сообщает об ошибке обработки.

2.3. Работа с внешними моделями

Взаимодействие с внешними сервисами вынесено в отдельный клиент. Основным провайдером служит Groq; при ошибке запрос может быть передан OpenRouter или Mistral AI. Ключи и названия моделей задаются через окружение, поэтому смена сервиса не требует изменять обработчики Telegram. Для обращений установлены тайм-ауты, а техническая причина сбоя записывается в журнал.

2.4. Пример прохождения запроса

Пользователь отправляет пост из нескольких абзацев и выбирает режим с эмодзи и выделениями. Обработчик сохраняет границы текста и исходные entities. Модель может предложить символ для первого абзаца, оставить второй без эмодзи и выделить точные фразы в третьем. Локальный модуль проверяет эти фрагменты, применяет оформление и пересчитывает позиции. При повторной генерации исходный пост остаётся тем же, а набор допустимых решений может измениться.

3. Локальная сборка Telegram-сообщения

Языковая модель не собирает итоговое Telegram-сообщение. Для каждого абзаца она возвращает набор решений, которые применяет локальный код. После вставки эмодзи программа заново рассчитывает позиции Telegram entities. Поэтому жирное, курсивное, подчёркнутое и другие виды оформления относятся к тем же фрагментам, что и в исходном сообщении.

Если результат превышает лимит одного сообщения, текст делится по абзацам, строкам или словам. Для каждой части формируется собственный набор entities. Локальная сборка сохраняет абзацы и пустые строки в исходном порядке. Сравнение трёх вариантов построения системы приведено в таблице 1.

Таблица 1 - Сравнение способов построения системы

Критерий	**Только правила**	**Готовый текст LLM**	**Решения LLM и локальная сборка**

Смысловой выбор	Ограничен	Высокий	Высокий
Сохранение текста	Контролируется	Есть риск изменений	Контролируется
Разметка и локальные ID	Полный контроль	Низкий контроль	Полный контроль

Комбинированная схема объединяет смысловой анализ модели с техническим контролем локального кода. Для Telegram Premium программа выбирает custom emoji из внутреннего каталога. Модель определяет тему и тип символа, а локальный модуль находит доступный идентификатор с учётом цвета и визуального набора [5]. По тому же принципу выбираются маркеры списков, цифры и сохранённые логотипы брендов.

3.1. Списки и длинные сообщения

Структура списка определяется до обращения к модели, поэтому все пункты получают единый маркер. Длинный результат нельзя обрезать по числу символов: граница может попасть внутрь слова, ссылки или выделенного фрагмента. Система сначала ищет место между абзацами, затем между строками и словами. Для каждой части создаются собственные entities, что позволяет сохранить текст и оформление.

4. Хранение данных и устойчивость

SQLite используется для локальных каталогов и пользовательских настроек [6]. Такой вариант подходит текущей однопроцессной версии и не требует отдельного сервера базы данных. Обычные тексты постов в базу и журнал не записываются. Пользователь может удалить весь профиль или только примеры, переданные для настройки стиля.

Краткие состояния, данные кнопок, очередь и лимиты находятся в памяти процесса и сбрасываются после перезапуска. Для нескольких экземпляров понадобится общее хранилище. Дополнительно применяются ограничение частоты запросов, ротация журнала, проверка Telegram ID для

административных команд и привязка кнопочных состояний к пользователю, чату и сообщению.

5. Проверка реализации

Проверялась ветка проекта «Textemojiai» на коммите ecd1b3f по состоянию на 26 июля 2026 года. Результаты локальной технической проверки приведены в таблице 2.

Таблица 2 - Результаты технической проверки

Проверка	Что охватывает	Результат
Автоматические тесты	Каскад моделей, доступ, состояния, базы, лимиты и длинные сообщения	22 теста пройдены
Ruff	Статический анализ Python-кода	Замечаний нет
Mypy	Проверка согласованности типов	Замечаний нет
SQLite	Открытие и чтение локальных баз	Ошибок не выявлено

Все основные модули прошли предусмотренные для них автоматические проверки в локальной среде. Тесты каскада моделей выполняются без запуска Telegram-интерфейса, а обработка длинных сообщений проверяется без реального обращения к LLM. Это помогает связать ошибку с конкретным уровнем системы. Результаты относятся только к перечисленным сценариям и не характеризуют качество оформления или работу под нагрузкой.

6. Ограничения и дальнейшая работа

Проверка проводилась в локальной среде; эксплуатационные и пользовательские испытания пока не выполнялись. На следующем этапе результаты можно оценивать по трём критериям: сохранность смысла, уместность оформления и необходимость самостоятельного редактирования. Для серверного запуска также потребуются общее хранилище состояний, синхронизация лимитов и отдельная проверка правил передачи текста внешним моделям.

Заключение

Предложена модульная архитектура Telegram-бота для стилистического оформления готовых постов. Её основной принцип состоит в разделении смысловых решений языковой модели и локальной сборки сообщения. Такой подход сохраняет исходный текст и позволяет контролировать Telegram entities, custom emoji и длинные сообщения.

В протестированных сценариях ошибок не выявлено: пройдены 22 автоматических теста, а Ruff и Муру не обнаружили замечаний. Пользовательская оценка и испытания под нагрузкой остаются задачами следующего этапа.

Список литературы

1. aiogram 3. Long-polling: официальная документация [Электронный ресурс]. URL: https://docs.aiogram.dev/en/latest/dispatcher/long_polling.html.

2. Lee M., Gero K. I., Chung J. J. Y., Shum S. B., Raheja V., Shen H. et al. A Design Space for Intelligent and Interactive Writing Assistants // Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. New York: ACM, 2024. Article 1054. DOI: 10.1145/3613904.3642697.

3. Raheja V., Kumar D., Koo R., Kang D. CoEdIT: Text Editing by Task-Specific Instruction Tuning // Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics, 2023. P. 5274-5291. DOI: 10.18653/v1/2023.findings-emnlp.350.

4. Shi S., Zhao E., Tang D., Wang Y., Li P., Bi W., Jiang H. et al. Effidit: Your AI Writing Assistant. 2022. arXiv:2208.01815. DOI: 10.48550/arXiv.2208.01815.

5. Shoeb A. A. M., de Melo G. Assessing Emoji Use in Modern Text Processing Tools // Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. Bangkok: Association for Computational Linguistics, 2021. P. 1379-1388. DOI: 10.18653/v1/2021.acl-long.107.

6. SQLite. About SQLite: официальная документация [Электронный ресурс]. URL: https://sqlite.org/about.html.

7. Telegram Bot API: официальная документация [Электронный ресурс]. URL: https://core.telegram.org/bots/api.

References

1. aiogram 3. Long-polling: official documentation. Available at: https://docs.aiogram.dev/en/latest/dispatcher/long_polling.html.

2. Lee M., Gero K. I., Chung J. J. Y., Shum S. B., Raheja V., Shen H. et al. A Design Space for Intelligent and Interactive Writing Assistants. Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. New York, ACM, 2024, Article 1054. DOI: 10.1145/3613904.3642697.

3. Raheja V., Kumar D., Koo R., Kang D. CoEdIT: Text Editing by Task-Specific Instruction Tuning. Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore, Association for Computational Linguistics, 2023, pp. 5274-5291. DOI: 10.18653/v1/2023.findings-emnlp.350.

4. Shi S., Zhao E., Tang D., Wang Y., Li P., Bi W., Jiang H. et al. Effidit: Your AI Writing Assistant. 2022. arXiv:2208.01815. DOI: 10.48550/arXiv.2208.01815.

5. Shoeb A. A. M., de Melo G. Assessing Emoji Use in Modern Text Processing Tools. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. Bangkok, Association for Computational Linguistics, 2021, pp. 1379-1388. DOI: 10.18653/v1/2021.acl-long.107.

6. SQLite. About SQLite: official documentation. Available at: https://sqlite.org/about.html.

7. Telegram Bot API: official documentation. Available at: https://core.telegram.org/bots/api.