Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Программное решение на основе искусственного интеллекта для разбиения сложных вопросов на простые с использованием больших языковых моделей

Студент	ИУ7-46Б		М.С. Гиричев
	(группа)	(подпись, дата)	(И.О. Фамилия)
Руководитель курсового проекта			Ю.В. Строганов
		(подпись, дата)	(И.О. Фамилия)
Консультант			Д.В. Якуба
		(подпись, дата)	(И.О. Фамилия)
			Н.В. Москвичёв
		(подпись, дата)	(И.О. Фамилия)
			Д.О.Левиев
		(подпись, дата)	(И.О. Фамилия)
		(подпись, дата)	(И.О. Фамилия)

2025 г.

РЕФЕРАТ

Расчетно-пояснительная записка 91 с., 9 рис., 4 табл., 58 ист., 2 прил.

В работе представлено программное решение на основе искусственного интеллекта для разбиения сложных вопросов на простые с использованием больших языковых моделей.

Ключевые слова: большие языковые модели, декомпозиция вопросов, искусственный интеллект, обработка естественного языка, промпт-инженерия.

Проведен комплексный анализ существующих методов декомпозиции вопросов, включая экспертную оценку, алгоритмические методы и методы на основе больших языковых моделей. Разработана трёхуровневая архитектура программного решения, состоящая из модуля консольного интерфейса, модуля управления запросами и модуля взаимодействия с языковой моделью. Реализованы алгоритмы формирования запроса к языковой модели и обработки ответа с многоуровневой валидацией контента. Выполнено исследование эффективности декомпозиции на тестовой выборке из 10 сложных вопросов с экспертной оценкой полноты, атомарности и корректности результатов для 30 языковых моделей. Установлено, что наилучшими показателями обладают проприетарные модели (DeepSeek-V3-Chat, ChatGPT-4, Claude-3-Opus), а среди локальных решений высокую эффективность демонстрируют T-Tech-T-pro-it-1.0 и RuadaptQwen-32B-Pro_v1. Разработаны правовые механизмы защиты системы в соответствии с требованиями законодательства РФ.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяются следующие термины с соответствующими определениями.

1) Сложный вопрос – вопрос, содержащий нескольк аспектов и требующий многокомпонентного ответа.

2) Простой вопрос – самодостаточный атомарный вопрос, фокусирующийся на одном аспекте с определенным ответом.

3) Модель – упрощённое представление реального объекта, процесса или явления, отражающее его существенные свойства.

4) Языковая модель – алгоритмическая система, обученная на больших массивах текста для понимания и генерации естественного языка.

5) Декомпозиция – разделение целого на составные части.

6) Промпт – текстовая инструкция для языковой модели.

7) Токен – минимальная единица текста, обрабатываемая языковой моделью.

8) ИИ – искусственный интеллект.

9) LLM – Large Language Model (большая языковая модель).

10) QDMR – Question Decomposition Meaning Representation (представление смысла декомпозиции вопроса).

11) EDG – Entity Description Graph (граф описания сущностей).

12) HSP – Hierarchical Semantic Parsing (иерархический семантический парсинг).

13) IDEF0 – Integration DEFinition for Function Modeling (методология функционального моделирования).

14) Dependency-граф – граф, отражающий синтаксические связи между словами в предложении.

15) API – Application Programming Interface (программный интерфейс приложения).

16) MMLU – Massive Multitask Language Understanding (массовое многозадачное понимание языка).

17) MATH – Mathematics Dataset (набор математических задач для тестирования языковых моделей).

18) HumanEval – набор тестов для оценки способности модели генерировать код.

19) HellaSwag – набор тестов для оценки здравого смысла и понимания контекста.

20) GPQA – General Purpose Question Answering (тест для оценки способности отвечать на общие вопросы).

21) Квантование – метод оптимизации модели путем уменьшения точности представления весов.

22) Дистилляция – процесс передачи знаний от большой модели к меньшей.

23) Токенизация – процесс разбиения текста на минимальные единицы (токены).

24) Контекстное окно – максимальное количество токенов, которое модель может обработать за один раз.

ВВЕДЕНИЕ

Развитие систем искусственного интеллекта создаёт новые возможности для автоматизации сложных интеллектуальных задач. Одной из таких задач является автоматическое разбиение сложных вопросов на простые составляющие. Это особенно актуально в контексте развития больших языковых моделей, которые показывают высокую эффективность при работе с простыми запросами, но часто затрудняются при обработке сложных (комплексных) вопросов. [1]

Существующие подходы к декомпозиции вопросов включают как классические методы на основе лингвистического анализа, так и современные решения с использованием нейронных сетей и больших языковых моделей. При этом классические методы часто ограничены жёсткими правилами и шаблонами, в то время как методы на основе ИИ способны адаптироваться к различным формулировкам и контекстам.

Цель работы состоит в разработке программного решения для автоматического разбиения сложных вопросов на простые с использованием больших языковых моделей.

Для достижения поставленной цели определены следующие задачи:

1) анализ существующих методов декомпозиции вопросов;
2) разработка алгоритма декомпозиции на основе больших языковых моделей;
3) реализация программного решения;
4) исследование эффективности разработанного решения.

1 Аналитический раздел

1.1 Анализ задачи

Задача декомпозиции сложных вопросов может быть определена как преобразование исходного вопроса в набор более простых, где каждый простой вопрос направлен на получение части информации, необходимой для полного ответа [2]. При этом возникают следующие ключевые проблемы:

1) определение границ подвопросов: необходимо корректно выделить независимые смысловые части исходного вопроса;
2) сохранение контекста: простые вопросы должны сохранять связь с контекстом исходного вопроса;
3) обеспечение полноты: набор простых вопросов должен покрывать всю информационную потребность исходного вопроса;
4) устранение избыточности: необходимо избегать повторяющихся или пересекающихся подвопросов [3].

Особую сложность представляют вопросы, содержащие неявные логические связи, требующие понимания контекста и предметной области. Например, вопрос «Как повлияла индустриальная революция на развитие городов в Европе?» требует учета множества аспектов: экономических, социальных, технологических и демографических изменений.

1.2 Анализ существующих решений

1.2.1 Экспертная оценка

В области декомпозиции сложных вопросов базовым подходом является привлечение экспертов предметной области. Эксперты, опираясь на свои знания и опыт, способны эффективно разбивать сложные вопросы на простые составляющие, учитывая специфику области и контекст. Такой подход обеспечивает высокое качество декомпозиции и позволяет выявлять неочевидные связи между компонентами вопроса, однако имеет ограничения по масштабируемости и требует значительных временных и человеческих ресурсов. [4]

1.2.2 Алгоритмические методы

EDG-Based Question Decomposition представляет собой комплексный подход к разбиению сложных вопросов, основанный на построении графа описания сущностей (Entity Description Graph). Алгоритм создаёт направленный ациклический граф, где узлы представляют вопросы и сущности, а рёбра отражают логические связи между ними. Особенность метода заключается в использовании итеративного процесса разбиения с применением специализированных правил: W-Rule для обработки вопросительных конструкций, C-Rule для координационных союзов, A-Rule и N-Rule для работы с атрибутивными и именными группами. Эффективность алгоритма подтверждена на задачах построения запросов к базам знаний, однако его производительность существенно зависит от качества начального синтаксического анализа. [5]

Question Decomposition with Dependency Graphs реализует подход на основе модели QDMR (Question Decomposition Meaning Representation), где сложный вопрос преобразуется в серию простых вычислительных шагов. Алгоритм применяет конвертацию аннотаций в логические формы и далее в dependency-графы, что позволяет явно моделировать связи между частями вопроса. Метод показывает 16-кратное ускорение по сравнению с другими моделями, однако требует качественной предварительной разметки данных для обучения. [6]

SPARQA: Skeleton-based Semantic Parsing предлагает подход к декомпозиции вопросов через построение «скелета» – формализации на основе dependency-грамматики. Алгоритм итеративно разбивает вопрос на текстовые span’ы, определяя их взаимосвязи и формируя направленное дерево. Особенность метода заключается в фокусировке на высокоуровневой семантической организации вопроса, что позволяет избежать ошибок dependency-парсинга на длинных сложных конструкциях. Однако метод требует значительных вычислительных ресурсов для построения и анализа скелетной структуры. [7]

Hierarchical Semantic Parsing (HSP) реализует трёхэтапный подход к декомпозиции вопросов. На первом этапе специальная нейронная модель разбивает исходный вопрос на последовательность подвопросов, затем извлекается ключевая семантическая информация, и наконец происходит интеграция данных для построения логической формы. Метод отличается способностью генерировать полные, естественные подвопросы вместо простого поиска точек разделения, однако требует тщательной настройки параметров для достижения оптимальной

производительности. [8]

1.2.3 Методы на основе больших языковых моделей

Большие языковые модели (Large Language Models, LLM) представляют собой особый класс решений для декомпозиции сложных вопросов, не требующий предварительной разработки правил или алгоритмов. В отличие от традиционных подходов, языковые модели способны выполнять декомпозицию, основываясь только на текстовой инструкции (промпте) и собственном обучении на больших массивах текстовых данных. [9] Данный подход значительно упрощает процесс внедрения и масштабирования решений для задач декомпозиции вопросов на практике.

DeepSeek (DeepSeek-V3-Chat) использует архитектуру смеси экспертов, что обеспечивает эффективное распределение вычислительных ресурсов при обработке запросов. Модель демонстрирует высокие результаты в задачах рассуждения и решения проблем, особенно в технических областях, где требуется понимание контекста и многоступенчатый анализ. [10]

ChatGPT-4 (chatgpt-4o-latest) представляет собой мультимодальную модель от OpenAI, способную эффективно обрабатывать текст, изображения и аудио данные. Благодаря улучшенной архитектуре и оптимизированным механизмам обработки информации, модель показывает высокую производительность при сниженной стоимости использования по сравнению с предыдущими версиями, что делает её подходящей для широкого спектра приложений. [11]

O1 (o1-mini) от OpenAI представляет собой компактную модель, оптимизированную для быстрого отклика и низких вычислительных затрат. Модель использует технологии квантования и дистилляции для достижения баланса между производительностью и качеством, что делает её эффективной для встраиваемых систем и мобильных приложений, где критична скорость работы и ограничены вычислительные ресурсы. [12]

Yi-Lightning (yi-lightning) использует архитектуру смеси экспертов с оптимизированной системой маршрутизации вычислений. Модель показывает высокие результаты в обработке китайского языка, математических вычислениях и задачах программирования, при этом обеспечивая эффективное использование вычислительных ресурсов. [13]

Claude-3 (claude-3-opus-20240229) от Anthropic реализует комплексный

подход к обработке текста и изображений. Модель демонстрирует высокие результаты в тестах MMLU и GPQA, что подтверждает её способность к качественному анализу и обработке сложных запросов в различных предметных областях. [14]

T-Tech (T-Tech-T-pro-it-1.0) ориентирована на промышленные приложения и обработку технической документации. Модель обучена на корпусе русского и английского языков, что позволяет ей эффективно работать с технической документацией и специализированными текстами в обоих языках. [15]

GigaChat (SberDevices-GigaChatMax) представляет собой российскую разработку с расширенными возможностями обработки естественного языка. Модель эффективно справляется с задачами коммуникации на русском языке, генерацией программного кода и аналитической обработкой данных, что делает её востребованной в различных бизнес-приложениях. [16]

Gemini (gemini-1.5-pro-002) от Google обеспечивает комплексную обработку текста, кода и мультимодальных данных. Модель отличается улучшенной производительностью в тестах MMLU и MATH, а оптимизированная система обработки данных позволяет снизить затраты на использование при сохранении высокого качества результатов. [17]

Qwen (Qwen2.5-72B-Instruct) от Alibaba поддерживает работу с 29 языками и обеспечивает высокое качество обработки естественного языка. Модель показывает стабильные результаты в задачах программирования и математических вычислениях, что делает её универсальным инструментом для различных прикладных задач. [18]

Phi (Phi-4) от Microsoft представляет собой компактную модель с 14 миллиардами параметров. Благодаря специализированной подготовке на качественных наборах данных, модель демонстрирует высокую эффективность в решении математических задач и обработке научных текстов. [19]

LLaMA 3 (llama-3.1-70b-instruct) от Meta обеспечивает высокую производительность в различных задачах обработки естественного языка. Модель успешно проходит тесты HumanEval и MMLU, что подтверждает её способность к анализу и генерации текста в различных предметных областях. [20]

Mistral (mistral-large-2407) реализует поддержку множества языков программирования и естественных языков. Модель обладает развитыми механизмами рассуждения и доступна через API на различных платформах, что упрощает

её интеграцию в существующие системы. [21]

Command-R+ (command-r-plus) от Cohere разработана для решения корпоративных задач и поддерживает работу с десятью языками. Модель обеспечивает высокую пропускную способность и низкую задержку, что делает её эффективным инструментом для промышленного применения. [22]

Gemma-2 (gemma-2-9b-it) от Google представляет собой открытую модель, оптимизированную для практического применения. Модель показывает высокие результаты в тестах MMLU и адаптирована для работы как в облачной среде, так и на мобильных устройствах. [23]

Watari (Watari-7b-v1) специализируется на обработке японского и английского языков с особым вниманием к языковым нюансам. Модель эффективно обрабатывает различные уровни вежливости в японском языке и поддерживает точный перевод между языками. [24]

YandexGPT (yandexgpt-4-pro) фокусируется на обработке русского языка и применяется в различных бизнес-задачах. Модель успешно используется для создания контента и разработки чат-ботов, демонстрируя высокую эффективность в задачах на русском языке. [25]

GPT-3.5 Turbo (gpt-3.5-turbo-0125) обеспечивает эффективную обработку текста при оптимальном соотношении цены и качества. Модель демонстрирует высокие результаты в тестах HellaSwag и MMLU, что подтверждает её способность к качественному анализу текста в различных контекстах. [26]

GLM-4 (glm-4-9b-chat) поддерживает работу с 26 языками и обладает расширенным контекстным окном. Модель включает функционал веб-поиска и выполнения программного кода, что расширяет её возможности в решении практических задач. [27]

C4AI Command-R (c4ai-command-r-v01) оптимизирована для задач рассуждения и обработки текстов. Модель поддерживает работу с большими объемами контекста и включает механизмы проверки генерируемого содержания, что повышает надежность её использования. [28]

Suzume (suzume-llama-3-8b-multilingual) основана на архитектуре LLaMA-3 и адаптирована для многоязычного применения. Модель обучена на большом корпусе диалогов на разных языках, что обеспечивает высокое качество обработки естественного языка в многоязычных приложениях. [29]

Hermes-2 (hermes-2-theta-llama-3-8b) от Nous Research фокусируется на

качественной обработке длительных диалогов. Модель эффективно сохраняет контекст беседы и поддерживает расширенные возможности взаимодействия с пользователем через программные интерфейсы. [30]

Saiga (saiga_llama3_8b_v6) разработана для работы с научной литературой в области точных наук. Модель обеспечивает точную обработку математических формул и поддерживает форматирование в LaTeX, что делает её полезным инструментом для научной работы. [31]

Aya-23-8b (aya-23-8b) реализует поддержку 23 языков через систему специализированных адаптеров. Модель обеспечивает высокое качество обработки текста для каждого поддерживаемого языка, приближаясь по эффективности к специализированным одноязычным решениям. [32]

Paralex (paralex-llama-3-8b-sft) ориентирована на обработку юридической и финансовой документации. Модель обеспечивает высокую точность анализа документов и значительно ускоряет процессы их обработки по сравнению с универсальными решениями. [33]

Storm (storm-7b) оптимизирована для работы в режиме реального времени на стандартном оборудовании. Модель сохраняет высокий уровень производительности при сниженных требованиях к вычислительным ресурсам, что делает её подходящей для широкого спектра применений. [34]

Neural-Chat (neural-chat-7b-v3-3) от Intel использует оптимизированные методы вычислений для повышения производительности. Модель поддерживает специализированные инструкции процессоров Intel и обеспечивает эффективную обработку данных на серверном оборудовании. [35]

Vikhr (vikhr-nemo-12b-instruct-r-21-09-24) представляет собой открытую модель для работы с русским языком. Модель использует специально адаптированную систему токенизации и постоянно совершенствуется через процесс непрерывного обучения. [36]

RuadaptQwen (RuadaptQwen-32B-Pro_v1) представляет собой адаптированную для русского языка версию модели Qwen2.5-32B с усовершенствованным токенизатором. Модель демонстрирует значительное повышение скорости генерации русскоязычных текстов при сохранении высокого качества ответов и эффективно решает широкий спектр задач обработки естественного языка. [37]

Zero-Mistral (ru-Zero-Mistral-Small-24B) основана на Mistral-Small-24B-Instruct-2501 с оптимизацией для русского и английского языков. Модель демон-

стрирует высокую производительность в задачах диалоговых систем и обеспечивает эффективную обработку запросов с сохранением контекста, что делает её подходящей для создания отзывчивых ассистентов. [38]

Cotype-Nano (MTSAIR-Cotype-Nano) представляет собой легковесную модель, оптимизированную для работы в условиях ограниченных вычислительных ресурсов. Модель разработана с использованием двухэтапного обучения, где MLP-слои тренировались на математических задачах и программном коде, что обеспечивает эффективную обработку запросов на русском и английском языках при сохранении быстродействия. [39]

1.3 Сравнение больших языковых моделей

Для объективной оценки возможностей языковых моделей была использована платформа LLM Arena – открытая краудсорсинговая система оценки больших языковых моделей на русском языке. Платформа использует модель Брэдли-Терри для ранжирования моделей на основе парных сравнений, выполненных людьми, и представляет результаты по шкале Эло. Модель Брэдли-Терри позволяет трансформировать субъективные оценки экспертов в количественные показатели, где вероятность предпочтения одной модели над другой зависит от разницы их рейтингов.

В таблице 1.1 представлены результаты тестирования лучших представителей каждого семейства моделей в категории Arena-Hard-Auto. Для каждой архитектуры или линейки моделей была отобрана версия с наивысшим показателем эффективности, что позволяет провести объективное сравнение различных подходов к построению языковых моделей. Представленные метрики включают: Score – общий балл модели по шкале Эло, отражающий её относительную силу в сравнении с другими моделями; CI (Confidence Interval) – доверительный интервал, указывающий на статистическую погрешность оценки; Avg Tokens – среднее количество токенов в ответе, характеризующее детальность генерируемых текстов; SD (Standard Deviation) – стандартное отклонение количества токенов, отражающее стабильность объёма ответов; LC (Language Competency) – оценка языковой компетенции, представляющая собой нормализованный показатель способности модели работать с естественным языком. Высокие значения Score и LC обычно коррелируют с лучшим пользовательским восприятием качества ответов. [40]

Таблица 1.1 – Рейтинг языковых моделей по результатам тестирования в LLM Arena

Модель	**Score**	**CI**	**Avg Tokens**	**SD**	**LC**
DeepSeek-V3-Chat	96.30	+0.8/-0.7	665.97	504.83	56.62
chatgpt-4o-latest	94.74	+0.8/-1.0	693.15	634.20	56.40
o1-mini	93.46	+0.7/-0.8	791.18	647.74	56.22
yi-lightning	93.46	+0.9/-1.0	636.68	469.74	56.22
RuadaptQwen-32B-Pro_v1	92.18	+1.2/-1.2	563.43	387.83	56.04
claude-3-opus-20240229	91.31	+1.0/-1.1	468.69	254.10	55.92
T-Tech-T-pro-it-1.0	90.87	+0.8/-1.1	502.00	380.68	55.85
SberDevices-GigaChatMax	89.96	+1.2/-1.4	523.95	421.87	55.73
gemini-1.5-pro-002	89.07	+1.3/-0.9	639.51	493.30	55.60
Qwen2.5-72B-Instruct	88.25	+0.8/-1.5	557.41	437.32	55.48
ru-Zero-Mistral-Small-24B	87.43	+1.4/-1.2	565.19	339.27	55.37
vikhr-nemo-12b-instruct-r	87.31	+1.1/-1.4	627.00	416.72	55.35
Phi-4	86.59	+1.1/-1.3	641.70	439.80	55.25
llama-3.1-70b-instruct	83.26	+1.4/-1.1	537.53	428.60	54.77
mistral-large-2407	80.40	+2.1/-1.5	422.42	320.72	54.36
command-r-plus	77.17	+1.5/-1.5	560.83	424.07	53.90
gemma-2-9b-it	76.50	+1.3/-1.3	459.15	312.59	53.81
Watari-7b-v1	69.49	+1.7/-1.7	616.80	449.25	52.80
yandexgpt-4-pro	59.24	+2.1/-1.9	383.80	306.97	51.33
MTSAIR-Cotype-Nano	50.51	+1.9 / -1.4	567.34	435.47	50.07
gpt-3.5-turbo-0125	50.00	+0.0/-0.0	220.83	170.30	50.00
glm-4-9b-chat	49.75	+2.0/-2.0	568.81	448.76	49.96
c4ai-command-r-v01	48.95	+3.0/-2.1	529.34	368.98	49.85
suzume-llama-3-8b	45.71	+2.3/-2.6	641.18	858.96	49.38
hermes-2-theta-llama-3-8b	44.08	+2.4/-2.3	485.99	390.85	49.15
saiga_llama3_8b_v6	39.17	+2.1/-1.6	471.51	463.62	48.44
aya-23-8b	36.26	+1.9/-2.2	554.34	433.51	48.02
paralex-llama-3-8b-sft	37.36	+2.5/-2.2	688.57	632.87	48.18
storm-7b	20.62	+1.7/-1.4	419.32	190.85	45.78
neural-chat-7b-v3-3	19.04	+1.5/-1.5	927.21	1211.62	45.56

1.4 Модель разрабатываемого процесса

Для наглядного представления процесса разбиения сложных вопросов на простые была создана IDEF0-диаграмма (рисунок 1.1). На ней показано, как сложный вопрос преобразуется в набор простых вопросов.

После анализа всех рассмотренных методов, для решения задачи лучше всего подходят большие языковые модели. Это объясняется несколькими причинами:

— отсутствие необходимости в разработке сложных правил и алгоритмов;
— высокая адаптивность к различным типам вопросов и предметным областям;
— возможность сохранения контекста и логических связей между частями вопроса;
— простота интеграции и масштабирования решения.

Рисунок 1.1 – Функциональная модель процесса разбиения сложных вопросов на простые

1.5 Вывод

В аналитическом разделе был проведен комплексный анализ задачи разбиения сложных вопросов на простые. Рассмотрены основные проблемы и требования к процессу декомпозиции, исследованы существующие подходы в трёх категориях: экспертная оценка, алгоритмические методы и методы на основе больших языковых моделей. Проанализировано более 30 существующих решений с точки зрения их применимости к решению поставленной задачи.

На основе проведенного анализа разработана функциональная модель процесса, в которой декомпозиция вопроса осуществляется с помощью языковой модели. Такой подход позволяет эффективно решать поставленную задачу, обеспечивая необходимую гибкость и качество результатов при минимальных требованиях к разработке дополнительных компонентов системы.

2 Конструкторский раздел

2.1 Функциональная модель системы

Функциональная модель системы декомпозиции сложных вопросов представляет процесс преобразования исходного сложного вопроса в набор более простых. Данный процесс реализуется с помощью локальной языковой модели, которая получает специально сформированные инструкции (промпты) для выполнения декомпозиции.

Проектируемая система основана на результатах анализа существующих решений, проведенного в аналитическом разделе. Для реализации выбран подход с использованием локальной языковой модели, что позволяет обеспечить высокую адаптивность к различным типам вопросов при сохранении контекста и логических связей, а также независимость от внешних сервисов.

Для формализации процесса декомпозиции разработана функциональная модель в нотации IDEF0, представленная на диаграммах нулевого и первого уровней (рисунки 2.1, 2.2).

Рисунок 2.1 – Функциональная модель процесса разбиения сложных вопросов (нулевой уровень)

Для более детального представления структуры системы разработана модель первого уровня, которая включает три последовательных процесса:

1) формирование запроса к языковой модели;
2) взаимодействие с языковой моделью;
3) формирование результата декомпозиции.

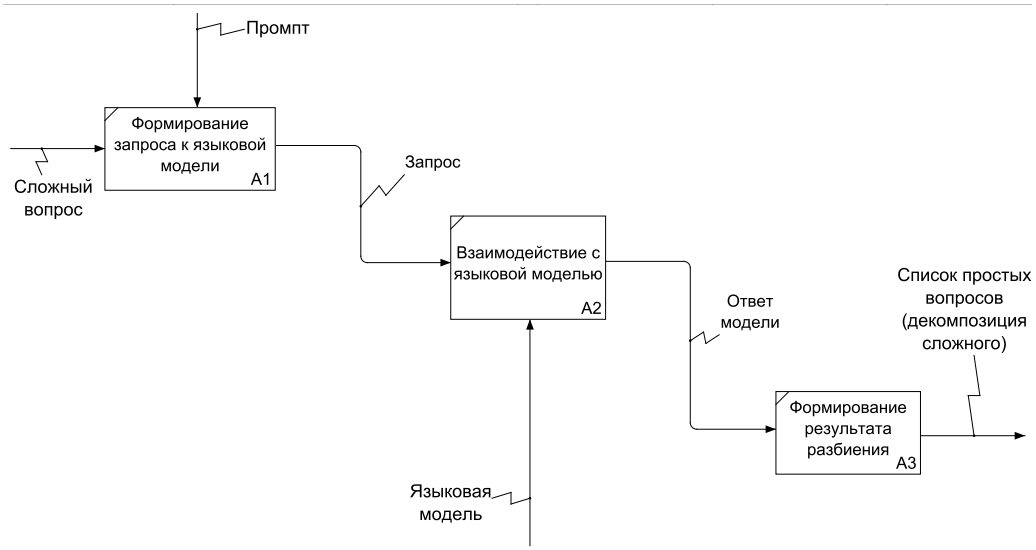


Рисунок 2.2 – Функциональная модель процесса разбиения сложных вопросов (первый уровень)

Декомпозиция процесса позволяет выделить три ключевых этапа обработки вопроса. На первом этапе исходный сложный вопрос преобразуется в формат запроса для языковой модели с добавлением промпта. Далее подготовленный запрос передается в языковую модель, которая генерирует ответ согласно полученным инструкциям. На заключительном этапе система обрабатывает полученный от модели ответ, структурирует его и формирует итоговый список простых вопросов.

Разделение на отдельные процессы позволяет локализовать компоненты системы и обеспечить их независимую модификацию и тестирование. Такой подход также упрощает интеграцию различных языковых моделей и форматов представления данных.

2.2 Архитектура программного решения

Программное решение для декомпозиции сложных вопросов построено на модульном принципе, что обеспечивает гибкость системы и возможность ее адаптации к различным требованиям. Архитектура включает три основных моду-

ля, взаимодействующих между собой для обеспечения полного цикла обработки вопросов (рисунок 2.3).

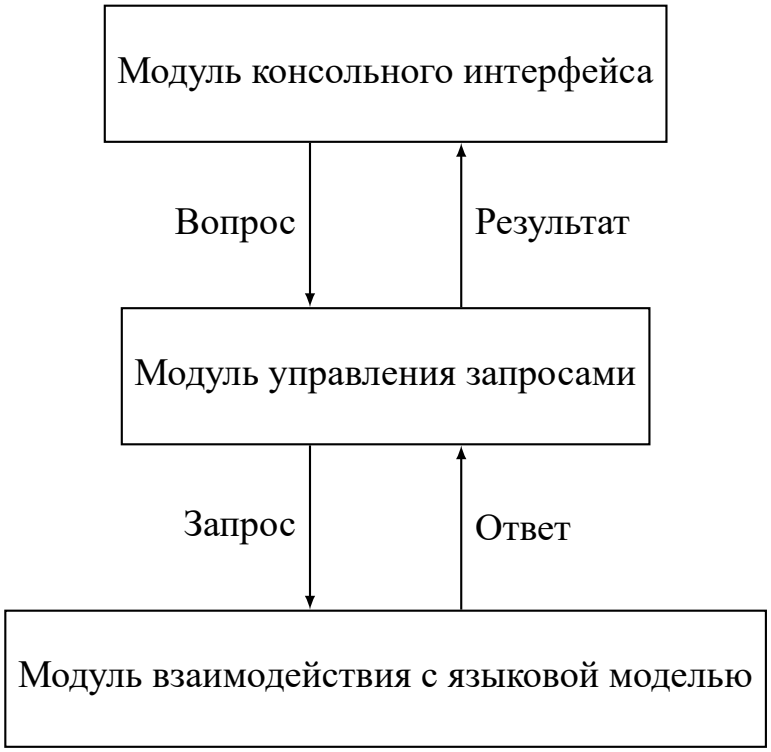


Рисунок 2.3 – Архитектура программного решения

Основные модули системы и их функции:

1) Модуль консольного интерфейса отвечает за взаимодействие с пользователем через командную строку. Основные функции модуля включают чтение сложных вопросов из указанных пользователем текстовых файлов и запись результатов декомпозиции в выходные файлы. Модуль поддерживает базовые команды для управления процессом и настройками приложения.
2) Модуль управления запросами осуществляет подготовку запроса к языковой модели с добавлением необходимых инструкций и обработку полученных результатов. Данный модуль также выполняет фильтрацию входных запросов и выходных ответов на предмет наличия запрещенных слов по предустановленному списку. Модуль связывает интерфейс пользователя с языковой моделью и обеспечивает корректную передачу данных между ними.
3) Модуль взаимодействия с языковой моделью обеспечивает коммуникацию с локальной языковой моделью, передачу запросов и получение ответов. Модуль содержит механизмы обработки ошибок для обеспечения надежности системы.

Взаимодействие между модулями построено по принципу последовательной обработки данных. Модуль консольного интерфейса считывает вопрос из указанного пользователем файла и передает его в модуль управления запросами. Этот модуль формирует полный запрос и передает его в модуль взаимодействия с языковой моделью. После получения ответа от модели данные проходят обратный путь: модуль взаимодействия передает их в модуль управления запросами для структурирования, после чего результат передается в модуль консольного интерфейса для записи в выходной файл.

2.3 Алгоритмическое обеспечение

2.3.1 Алгоритм формирования запроса к языковой модели

Алгоритм формирования запроса предназначен для подготовки структурированного запроса к языковой модели, включающего оригинальный вопрос пользователя и специализированный промпт с инструкциями по декомпозиции (рисунок 2.4). Он представляет собой последовательность действий от получения исходного вопроса до формирования полного запроса к модели.

Алгоритм включает следующие шаги:

1) получение исходного вопроса из указанного файла;
2) загрузка шаблона промпта из конфигурационного файла;
3) объединение промпта и вопроса пользователя в единый запрос;
4) проверка запроса на наличие запрещенных слов по предустановленному списку;
5) форматирование запроса для передачи языковой модели;
6) возвращение готового запроса для дальнейшей обработки.

Временная сложность алгоритма: $O(n)$, где n – длина исходного вопроса и промпта. Требования к памяти: $O(n)$, необходимая память линейно зависит от размера вопроса и промпта.

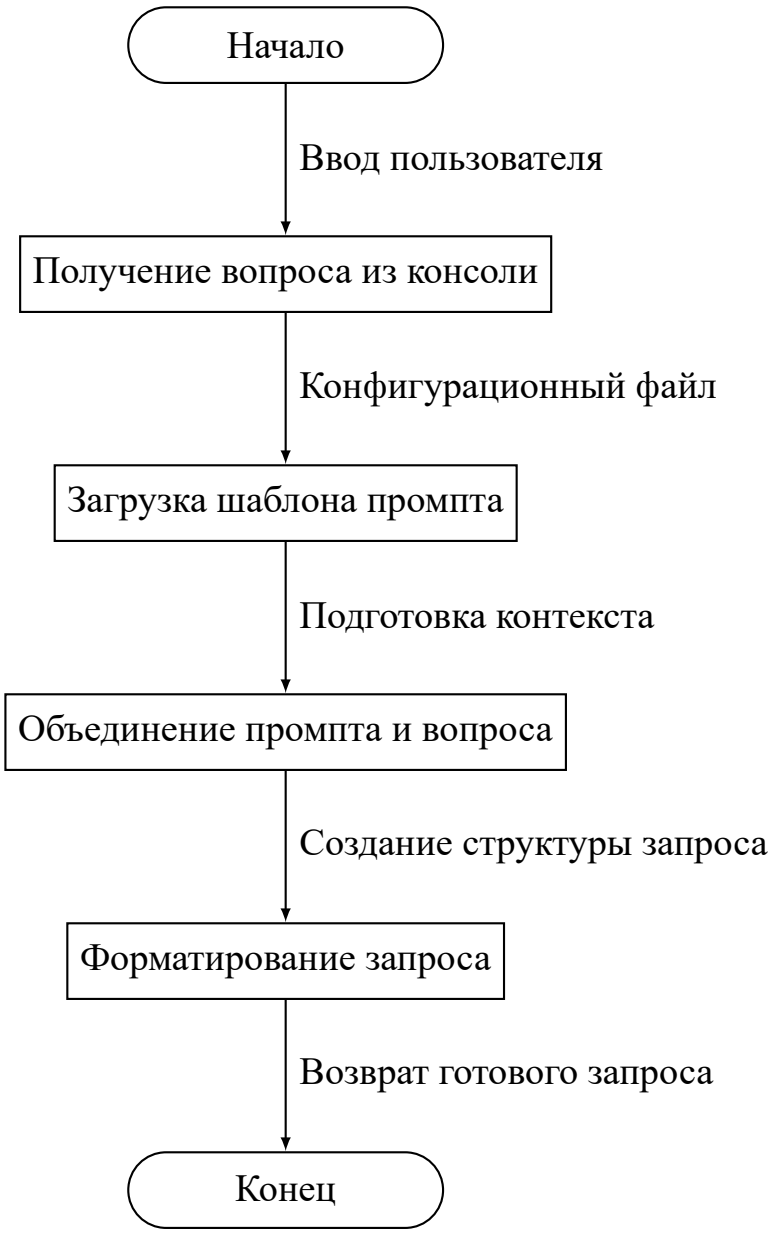


Рисунок 2.4 – Алгоритм формирования запроса к языковой модели

2.3.2 Алгоритм обработки ответа языковой модели

Алгоритм обработки ответа предназначен для структурирования результата, полученного от языковой модели, в заданный формат (список вопрос), который ожидает пользователь (рисунок 2.5).

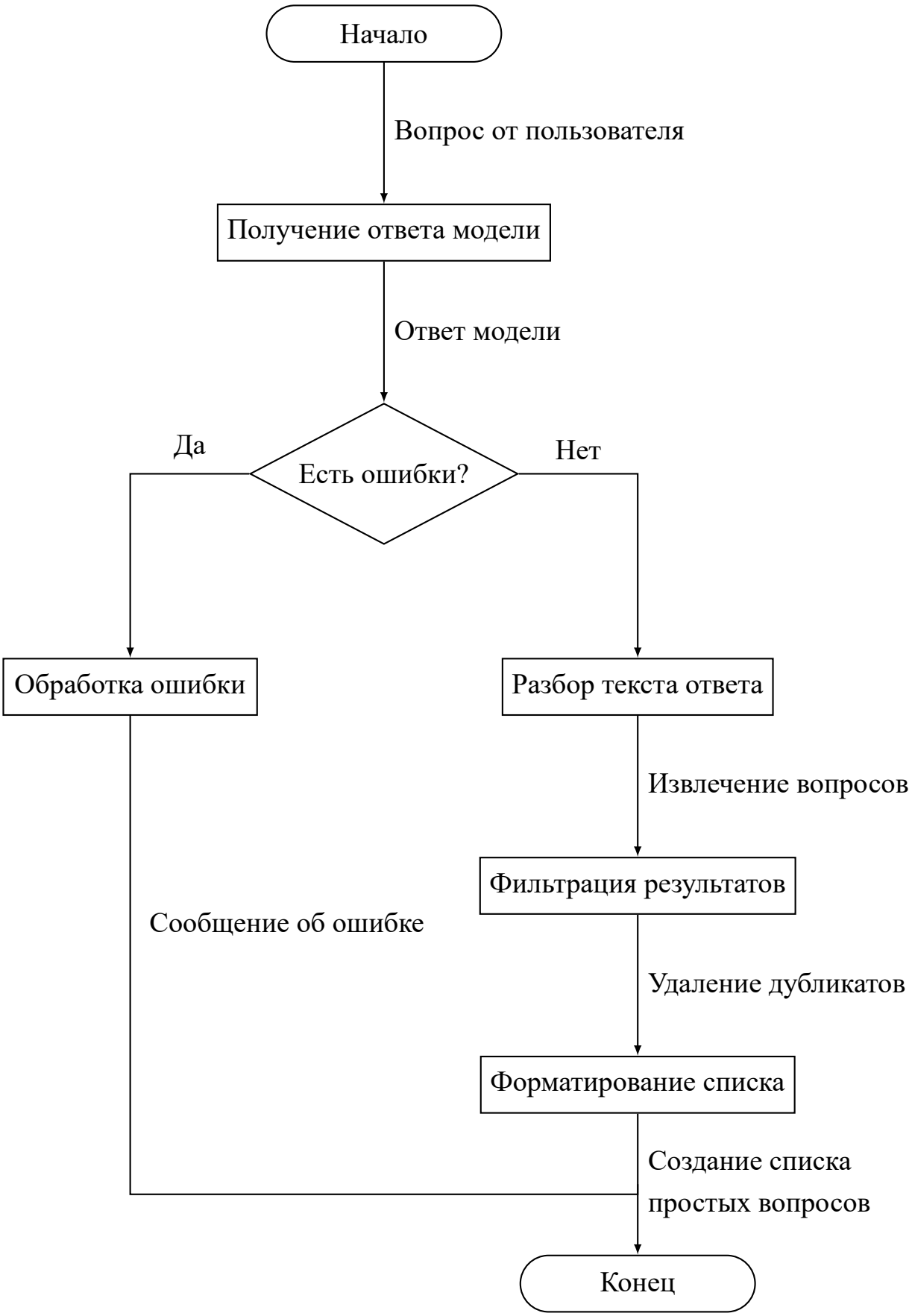


Рисунок 2.5 – Алгоритм обработки ответа языковой модели

Алгоритм обработки ответа иллюстрирует процесс извлечения и структурирования простых вопросов из текстового ответа языковой модели. Включает проверки на наличие ошибок и корректность формата ответа.

Алгоритм состоит из следующих шагов:

1) получение текстового ответа от языковой модели;
2) проверка на наличие ошибок в ответе модели;
3) разбор текстового ответа и извлечение отдельных вопросов;
4) проверка извлеченных вопросов на наличие запрещенных слов;
5) фильтрация и удаление возможных дубликатов или нерелевантных элементов;
6) форматирование результата в структурированный список вопросов;
7) сохранение готового списка простых вопросов в выходной файл.

Временная сложность алгоритма: $O(m)$, где m – длина ответа языковой модели. Требования к памяти: $O(m)$, необходимая память линейно зависит от размера ответа.

2.4 Сценарий использования системы

Основной сценарий использования консольного приложения включает:

1) запуск приложения через командную строку;
2) формирование системой запроса к языковой модели;
3) отправка запроса в локальную языковую модель и получение ответа;
4) обработка ответа и формирование списка простых вопросов;
5) запись результата в указанный файл в виде пронумерованного списка простых вопросов.

2.5 Требования к консольному интерфейсу

Консольный интерфейс системы должен обеспечивать:

— поддержку пакетного режима обработки нескольких файлов;
— поддержку базовых команд: декомпозиция, справка, завершение работы;
— отображение процесса обработки запроса (вывод названия текущего процесса);
— формирование диагностических сообщений при возникновении ошибок.

2.6 Процессная модель системы

Для наглядного представления последовательности операций и взаимодействия между компонентами системы разработана процессная модель декомпозиции сложных вопросов в нотации BPMN (Business Process Model and Notation), представленная на рисунок 2.6.

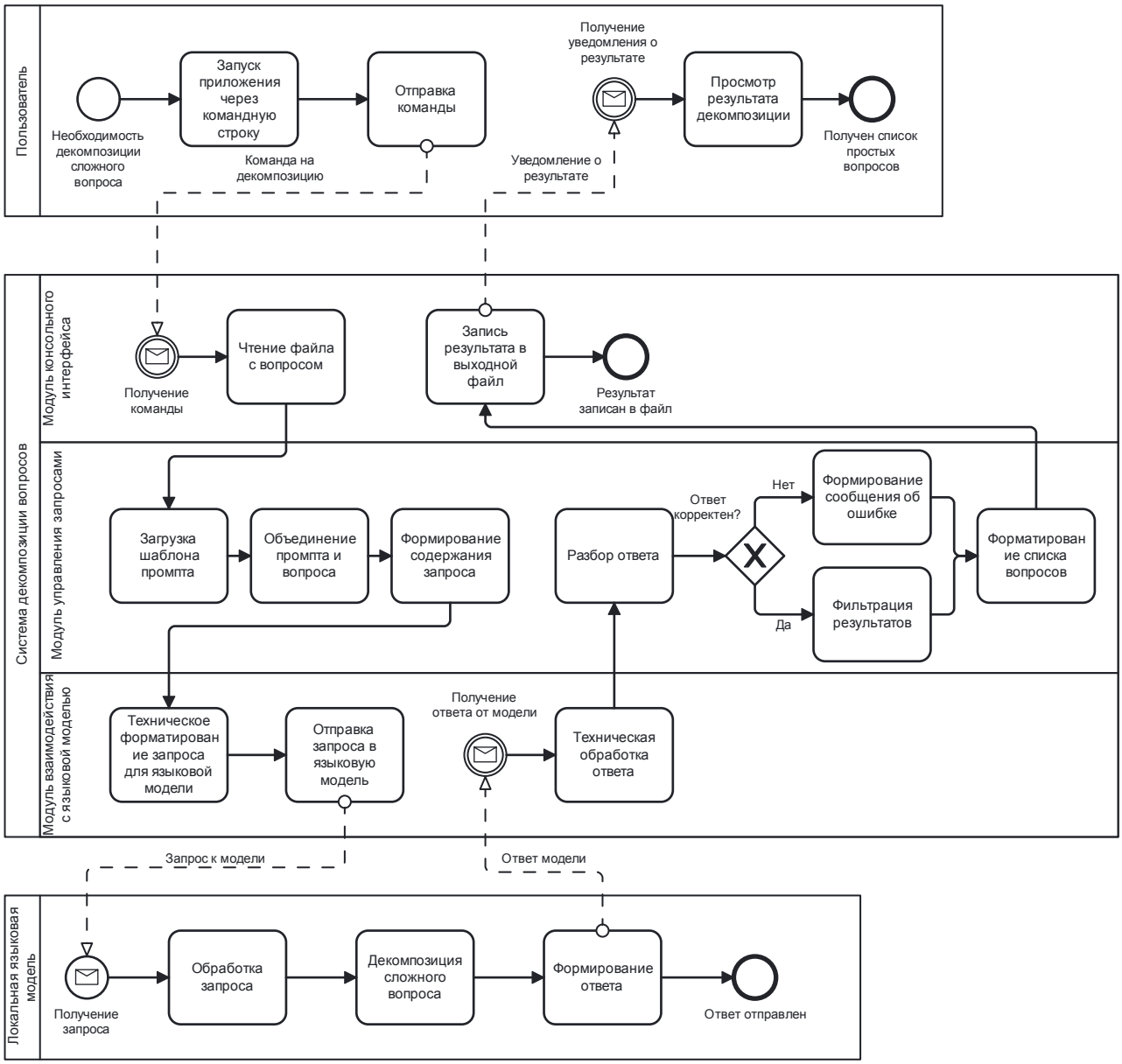


Рисунок 2.6 – Процессная модель декомпозиции сложных вопросов в нотации BPMN

Данная модель детализирует взаимодействие четырех основных участников процесса:

— пользователь системы, инициирующий декомпозицию вопроса;

— модуль консольного интерфейса, обеспечивающий ввод-вывод данных;

— система декомпозиции вопросов, включающая модуль управления запросами и модуль взаимодействия с языковой моделью;
— локальная языковая модель, выполняющая непосредственную декомпозицию.

Представленная процессная модель иллюстрирует полный жизненный цикл декомпозиции вопроса – от момента запуска приложения пользователем до получения структурированного списка простых вопросов. В модели отражены точки принятия решений, например, проверка корректности ответа модели, которая позволяет обрабатывать нестандартные ситуации.

Модель также демонстрирует распределение задач между тремя основными модулями системы и позволяет визуализировать потоки данных между ними. Такое представление обеспечивает комплексное понимание процесса декомпозиции и возможность анализа системы на предмет оптимизации взаимодействия между компонентами.

2.7 Параметры системы

2.7.1 Временные характеристики

Для обеспечения приемлемого пользовательского опыта система должна соответствовать следующим временным характеристикам:

— время формирования запроса к языковой модели: не более 0.01 секунды;
— время обработки запроса языковой моделью: не более 60 секунд для сложных вопросов (зависит от размера модели и вычислительных ресурсов);
— время обработки ответа и формирования результата: не более 60 секунд (зависит от размера модели и вычислительных ресурсов).

2.7.2 Параметры языковой модели

К используемой локальной языковой модели предъявляются следующие требования:

— поддержка контекстного окна не менее 2000 токенов для обработки сложных вопросов;
— возможность работы с русским языком;
— способность следовать инструкциям в промпте для структурированного вывода;

— производительность, достаточная для обработки запроса в рамках установленных временных ограничений.

2.8 Вывод

В конструкторском разделе разработана функциональная модель системы декомпозиции сложных вопросов на простые с использованием локальной языковой модели. Определена архитектура программного решения, включающая три основных модуля: консольный интерфейс, управление запросами и взаимодействие с языковой моделью.

Предложены алгоритмы формирования запроса к языковой модели и обработки полученных результатов, определены их временные и пространственные характеристики. Описаны основные сценарии использования системы, включая работу с файлами для ввода сложных вопросов и вывода результатов декомпозиции.

Установлены временные характеристики работы системы и требования к используемой языковой модели. Сформулированы системные требования для функционирования программного решения с локальной языковой моделью.

Разработанное проектное решение обеспечивает основу для реализации системы декомпозиции сложных вопросов, отвечающей требованиям надежности и интуитивности использования.

3 Технологический раздел

3.1 Выбор средств разработки

В данном разделе обосновывается выбор технологий и инструментов, используемых для реализации программного решения декомпозиции сложных вопросов на простые.

3.1.1 Язык программирования

Для реализации программного решения использован язык Python версии 3.11. Выбор обусловлен следующими факторами:

1) наличие готовых библиотек для работы с NLP и языковыми моделями;
2) относительная простота синтаксиса, что сокращает время разработки;
3) поддержка основных операционных систем (Windows, Linux, macOS);
4) простота интеграции с локальными моделями машинного обучения через существующие Python-обертки;
5) возможность применять как объектно-ориентированный, так и функциональный подходы при необходимости.

3.1.2 Библиотеки и фреймворки

Для реализации программного решения использованы следующие библиотеки:

1) **llama-cpp-python** [41]:
   — python-обертка для llama.cpp, позволяющая запускать локальные языковые модели;
   — запуск выполнение моделей на CPU и опционально на GPU;
   — поддержка различных форматов моделей (GGUF, GGML);
2) **typer** [42]:
   — создание CLI-приложений с типизированными аргументами;
   — автоматическая валидация входных данных;
   — генерация справочной документации;
   — поддержка подкоманд и расширенная обработка ошибок.
3) **rich** [43]:
   — форматированный вывод в терминале с поддержкой цветов и стилей;
   — отображение прогресса выполнения операций;

4) **pydantic** [44]:

— валидация данных и управление настройками приложения;

— автоматическая сериализация/десериализация данных;

— поддержка схем валидации на основе аннотаций типов python.

3.1.3 Среда разработки

Разработка программного решения велась в операционной системе **Windows 11**. В качестве интегрированной среды разработки использовался **Visual Studio Code**.

3.2 Модульная структура программного решения

Разработанное решение представляет собой консольное приложение, структура которого соответствует архитектуре, спроектированной в конструкторском разделе. Программа состоит из трех основных модулей, каждый из которых выполняет определенную функцию в процессе декомпозиции вопросов.

Структура файлов проекта организована следующим образом:

Листинг 3.1 – Файловая структура проекта

```
project/
├──configs/
│    └──config.json           # Конфигурационный файл
│
├──examples/                  # Примеры входных данных
│    ├──question1.txt
│    └──question2.txt
│
├──models/                    # Директория для файлов языковых моделей
│    └──t-lite-it-1.0-q4_k_m.gguf
│
├──results/                   # Результаты работы программы
│    ├──answers1.txt
│    └──answers2.txt
│
└──src/                       # Исходный код программы
     ├──cli_module.py         # Модуль консольного интерфейса
     ├──model_module.py       # Модуль взаимодействия с языковой моделью
     ├──request_module.py     # Модуль управления запросами
     ├──curse_words.txt       # Список запрещённых слов
     └──__init__.py
```

3.2.1 Модуль консольного интерфейса

Модуль консольного интерфейса (`cli_module.py`) отвечает за взаимодействие с пользователем через командную строку, загрузку конфигурации и координацию работы других модулей. Основные компоненты модуля представлены в листингах 3.2–3.5.

Листинг 3.2 – Инициализация консольного приложения

```
import typer
import json
from pathlib import Path
from rich.console import Console
from rich.progress import Progress

from request_module import RequestManager
from model_module import ModelInterface

app = typer.Typer(help="Декомпозиция сложных вопросов на простые")
console = Console()
```

Как видно из листинга 3.2, для создания консольного интерфейса используется библиотека `typer`, которая обеспечивает типизированные аргументы командной строки. Библиотека `rich` применяется для форматированного вывода в терминал с цветовым выделением информации.

Листинг 3.3 – Определение основной команды приложения

```
@app.command()
def main(
    config_path: Path = typer.Option(
        "configs/config.json", help="Путь к конфигурационному файлу"
    ),
):
    try:
        if not config_path.exists():
            console.print(f"[bold red] Ошибка:[/] Файл {config_path} не
    найден")
            raise typer.Exit(code=1)
```

В листинге 3.3 определяется основная команда приложения, которая принимает опциональный аргумент – путь к конфигурационному файлу. Если путь не указан, используется файл `configs/config.json`. Также присутствует проверка существования конфигурационного файла с соответствующим форматированием сообщения об ошибке.

Листинг 3.4 – Обработка вопроса и взаимодействие с компонентами системы

```
with open(config_path, "r", encoding="utf-8") as f:
        config = json.load(f)

    input_file = Path(config["files"]["input_file"])
    output_file = Path(config["files"]["output_file"])

    if not input_file.exists():
        console.print(f"[bold redОшибка]:[/] Файл {input_file} не найден")
        raise typer.Exit(code=1)

    with open(input_file, "r", encoding="utf-8") as f:
        question = f.read().strip()

    if not question:
        console.print("[bold redОшибка]:[/] Входной файл пуст")
        raise typer.Exit(code=1)

    request_manager = RequestManager(config)
    if request_manager.contains_curse_words(question):
        console.print("[bold redОшибка]:[/] Входной текст содержит
запрещенные слова")
        raise typer.Exit(code=1)

    console.print(f"[bold greenИсходный] вопрос:[/] {question}")

    model_interface = ModelInterface(config)
    request_manager = RequestManager(config)

    with Progress() as progress:
        task = progress.add_task("[cyanОбработка]...", total=4)

        progress.update(task, description="[cyanФормирование] запроса...",
advance=1)
        prompt = request_manager.create_prompt(question)

        progress.update(task, description="[cyanОбработка] запроса моделью...
", advance=1)
        response = model_interface.generate_response(prompt)

        progress.update(task, description="[cyanОбработка] результата...",
advance=1)
        simple_questions = request_manager.process_response(response)

        progress.update(task, description="[cyanЗавершение] обработки...",
advance=1)
```

В листинге 3.4 демонстрируется основная логика работы: загрузка кон-

фигурации, валидация входного файла и его содержимого, инициализация компонентов системы и последовательное выполнение этапов обработки с отображением прогресса в терминале. Присутствуют проверки входных данных (на наличие запрещённых слов и проверка языковой моделью) и форматирование сообщений.

Листинг 3.5 – Сохранение результатов и запуск приложения

```
    console.print("[bold green] Результат декомпозиции:[/]")
    for i, question in enumerate(simple_questions, 1):
        console.print(f"[bold]{i}.[/] {question}")

    output_file.parent.mkdir(parents=True, exist_ok=True)
    with open(output_file, "w", encoding="utf-8") as f:
        for i, question in enumerate(simple_questions, 1):
            f.write(f"{i}. {question}\n")

    console.print(f"[bold green] Результаты сохранены в файл:[/] {output_file}")

except json.JSONDecodeError:
    console.print(f"[bold red] Ошибка:[/] Некорректный формат JSONфайла- {config_path}")
    raise typer.Exit(code=1)
except FileNotFoundError as e:
    console.print(f"[bold red] Ошибка:[/] {str(e)}")
    raise typer.Exit(code=1)
except Exception as e:
    console.print(f"[bold red] Ошибка при обработке:[/] {str(e)}")
    raise typer.Exit(code=1)

if __name__ == "__main__":
    app()
```

3.2.2 Модуль управления запросами

Модуль управления запросами (`request_module.py`) отвечает за формирование запроса к языковой модели и обработку полученного ответа. Ключевые части этого модуля представлены в листингах 3.6–3.7.

Листинг 3.6 – Инициализация менеджера запросов

```
import re
from typing import List, Dict, Any

class RequestManager:
    def __init__(self, config: Dict[str, Any]):
        self.prompt_config = config["prompt"]
        self.prompt_template = self.prompt_config["template"]
        self.curse_words = self.load_curse_words()

    def load_curse_words(self) -> List[str]:
        try:
            with open("curse_words", "r", encoding="utf-8") as f:
                return [line.strip().lower() for line in f if line.strip()]
        except FileNotFoundError:
            print("Предупреждение: Файл curse_words не найден, проверка будет пропущена")
            return []

    def contains_curse_words(self, text: str) -> bool:
        if not self.curse_words:
            return False
        text_lower = text.lower()
        for word in self.curse_words:
            if word in text_lower:
                return True
        return False

    def create_prompt(self, question: str) -> str:
        prompt = self.prompt_template.format(question=question)
        return prompt.strip()

    def create_validation_prompt(self, response: str) -> str:
        validation_template = self.prompt_config.get("validation_prompt",
            "Проверь, соответствует ли следующий ответ правовым нормам и законам РФ, " +
            "а также моральным и этическим стандартам. Не нарушает ли материал " +
            "какиелибо- законы? Если ответ соответствует нормам, ответь ВАЛИДНО''. " +
            "Если нет - опиши нарушения и предложи исправленную версию. Ответ: {response}")

        return validation_template.format(response=response)
```

В листинге 3.6 демонстрируется инициализация менеджера запросов с параметрами из общего конфигурационного файла, а также два метода для создания промптов: основной промпт и промпт для валидации ответа модели.

А также методы для загрузки и проверки запрещенных слов, что позволяет выполнять фильтрацию входных запросов пользователя.

Листинг 3.7 – Создание промпта и обработка ответа модели

```
def process_response(self, response: str) -> List[str]:
    questions = []
    numbered_questions = re.findall(r'^\s*\d+\.\s*(.+)$', response, re.MULTILINE)

    if numbered_questions:
        questions.extend(numbered_questions)

    if not questions:
        lines = [line.strip() for line in response.split('\n') if line.strip()]
        for line in lines:
            cleaned_line = re.sub(r'^\s•*[-*]|\d+\.?\s*', '', line).strip()
            if cleaned_line and not cleaned_line.startswith('Вопрос') and '?' in cleaned_line:
                questions.append(cleaned_line)

    unique_questions = []
    for q in questions:
        if q not in unique_questions:
            unique_questions.append(q)

    return unique_questions
```

Листинг 3.7 содержит улучшенный метод для извлечения структурированного списка вопросов из ответа модели с помощью регулярных выражений. Добавлена логика обработки ответа в случае, если нумерованные вопросы не были найдены – в этом случае код ищет строки, содержащие знак вопроса, и очищает их от лишних символов.

3.2.3 Модуль взаимодействия с языковой моделью

Модуль взаимодействия с языковой моделью (`model_module.py`) обеспечивает инициализацию и коммуникацию с локальной языковой моделью. Основные компоненты модуля представлены в листингах 3.8–3.11.

Листинг 3.8 – Инициализация интерфейса модели

```
from pathlib import Path
from typing import Dict, Any, Optional
from llama_cpp import Llama

class ModelInterface:
    def __init__(self, config: Dict[str, Any]):
        self.model_config = config["model"]
        self.system_prompt = config["prompt"]["system_prompt"]
        self.model_path = Path(self.model_config["model_path"])
        self.validation_prompt = config["prompt"].get("validation_prompt", "
    Проверь, соответствует ли следующий ответ правовым нормам и законам РФ,
    а также моральным и этическим стандартам. Не нарушает ли материал
    какиелибо- законы? Если ответ соответствует нормам, ответь ВАЛИДНО''.
    Если нет - ответь НАРУШЕНИЕ''. Ответ: {response}")
        self.curse_words = self._load_curse_words()
        self._initialize_model()
```

В листинге 3.8 при инициализации интерфейса теперь также загружается список запрещенных слов, что позволяет контролировать содержание ответов и отфильтровывать нежелательный контент. Методы для загрузки и проверки запрещенных слов показаны в листинге 3.9.

Листинг 3.9 – Методы проверки запрещенных слов

```
def _load_curse_words(self) -> List[str]:
    try:
        with open("curse_words", "r", encoding="utf-8") as f:
            return [line.strip().lower() for line in f if line.strip()]
    except FileNotFoundError:
        print("Предупреждение: Файл curse_words не найден, проверка будет
    пропущена")
        return []

def _contains_curse_words(self, text: str) -> bool:
    if not self.curse_words:
        return False
    text_lower = text.lower()
    for word in self.curse_words:
        if word in text_lower:
            return True
    return False
```

В листинге 3.10 показан метод для инициализации модели с проверкой существования файла модели и обработкой возможных ошибок инициализации.

Листинг 3.10 – Инициализация модели с проверкой

```
def _initialize_model(self):
    if not self.model_path.exists():
        raise FileNotFoundError(f"Файл модели не найден по пути {self.
    model_path}")

    try:
        self.model = Llama(
            model_path=str(self.model_path),
            n_ctx=self.model_config.get("context_size", 4096),
            n_threads=self.model_config.get("n_threads", 4),
            n_gpu_layers=self.model_config.get("n_gpu_layers", 0),
            verbose=False,
            offload_kqv=True
        )
    except Exception as e:
        raise RuntimeError(f"Не удалось инициализировать модель: {str(e)}")
```

В листинге 3.11 представлен метод `generate_response`, отвечающий за взаимодействие с языковой моделью. Метод формирует запрос, используя системный и пользовательский промпты, и контролирует параметры генерации текста. Результат проходит трехуровневую валидацию: проверку на запрещенные слова, анализ маркеров отказа модели (`"НАРУШЕНИЕ"`, `"НЕ ОТВЕЧУ"`) и дополнительную этико-правовую проверку через метод `_validate_response`. Система обработки исключений обеспечивает корректное информирование об ошибках.

Листинг 3.11 – Генерация ответа языковой модели с валидацией

```
def generate_response(self, prompt: str) -> str:
    try:
        response = self.model.create_chat_completion(
            messages=[
                {"role": "system", "content": self.system_prompt},
                {"role": "user", "content": prompt}
            ],
            temperature=self.model_config.get("temperature", 0.7),
            top_p=self.model_config.get("top_p", 0.9),
            max_tokens=self.model_config.get("max_tokens", 1024)
        )

        response_text = response["choices"][0]["message"]["content"]

        if self._contains_curse_words(response_text):
            raise RuntimeError("Ответ содержит запрещенные слова")

        is_valid = ("НАРУШЕНИЕ" not in response_text.upper() and
                    "НЕ ОТВЕЧУ" not in response_text.upper() and
                    len(response_text.split()) > 3)

        if not is_valid:
            raise RuntimeError("Задан вопрос, нарушающий нормы, либо задан не
    вопрос.")

        self._validate_response(response_text)

        return response_text

    except Exception as e:
        raise RuntimeError(f"Ошибка при генерации ответа: {str(e)}")
```

3.2.4 Валидация ответа модели

Дополнением к системе является механизм валидации ответов модели на соответствие этическим и правовым нормам. Эта функциональность реализована в модуле взаимодействия с моделью и представлена в листинге 3.12.

Листинг 3.12 – Метод валидации ответа

```
def _validate_response(self, response: str) -> None:
    try:
        validation_prompt = self.validation_prompt.format(response=response)
        validation_result = self.model.create_chat_completion(
            messages=[
                {"role": "system", "content": "Ты - эксперт по правовым и этическим нормам РФ."},
                {"role": "user", "content": validation_prompt}
            ],
            temperature=0.2,
            top_p=0.9,
            max_tokens=100
        )

    validation_text = validation_result["choices"][0]["message"]["content"]

    words = validation_text.split()
        is_valid = ("ВАЛИДНО" in validation_text.upper() and
                    "НАРУШЕНИЕ" not in validation_text.upper() and
                    len(words) <= 3)

        print(f"Результат валидации: Прошел{'' if is_valid else 'Не' прошел'}")


        if not is_valid:
            raise RuntimeError("Обнаружены недопустимые темы в ответе во время валидации.")

    except Exception as e:
        print(f"Ошибка во время валидации: {str(e)}")
```

3.2.5 Проверка на запрещенные слова

Дополнительно к механизму валидации ответов модели на соответствие этическим и правовым нормам в систему интегрирована проверка текста на наличие запрещенных слов. Эта функциональность реализована как для входных запросов пользователя, так и для ответов, генерируемых моделью, что обеспечивает двойной контроль содержания.

Список запрещенных слов загружается из файла `curse_words` при инициализации компонентов системы. Если файл отсутствует, выдается предупреждение, и проверка не выполняется. Такой подход позволяет гибко настраивать фильтрацию контента путем редактирования внешнего файла без изменения исходного кода программы.

Проверка запрещенных слов выполняется на двух уровнях:

1) на уровне консольного интерфейса – все входные запросы проверяются перед отправкой языковой модели;

2) на уровне модуля языковой модели – проверяется ответ модели перед его передачей пользователю.

3.3 Конфигурационные файлы

Для обеспечения гибкости настройки системы используется единый конфигурационный JSON-файл, содержащий все параметры работы модулей. В листинге 3.13–3.14 демонстрируется пример конфигурационного файла.

Листинг 3.13 – Конфигурационный файл (configs/config.json). Часть 1

```
{
    ”model”: {
        ”model_path”: ”models/t-lite-it-1.0-q4_k_m.gguf”,
        ”context_size”: 4096,
        ”temperature”: 0.7,
        ”top_p”: 0.9,
        ”max_tokens”: 1024,
        ”n_gpu_layers”: 0,
        ”n_threads”: -1
    },
    ”prompt”: {
        ”system_prompt”: Ты” — эксперт по декомпозиции сложных вопросов. Ты занимаешься исключительно декомпозицией сложных вопросов и больше ничем. В случае, если тебе был передан не вопрос ты пишешь – НАРУШЕНИЕ’’. Если тебе был передан вопрос с запрещённым плохим() содержимым ты пишешь – НЕ’ ОТВЕЧУ’.”,
        ”template”: Ты” — эксперт по декомпозиции сложных вопросов. Разбей сложный вопрос на простые, независимые вопросы.\Каждыйn простой вопрос должен быть:\n1. Самодостаточным — понятным без контекста других вопросов;\n2. Атомарным — фокусирующимся на одном аспекте;\n3. Однозначным — имеющим определенный ответ;\n4. Разбиение должно быть весьма подробным.\n\Сложныйn вопрос: {question}\n\n. И помни, что это должны быть именно вопросы, а не просто предложения. Никакое форматирование не применяй. Напиши вопросы в виде нумерованного списка. Тебе нужно написать только исключительно сами вопросы и больше ничего.”
}
```

Листинг 3.14 – Конфигурационный файл (configs/config.json). Часть 2

```
{
        "validation_prompt": Проверь", соответствует ли следующий текст
    правовым нормам и законам РФ, моральным и этическим стандартам и не
    нарушает ли материал какиелибо- законы. Если ответ соответствует всем
    перечисленным нормам, ответь ВАЛИДНО'', иначе - НАРУШЕНИЕ'' и( больше
    ничего, только одно слово, не надо ничего объяснять ВАЖНО()).\Вотп текст:
     {response}"
    },
    "files": {
        "input_file": "examples/question3.txt",
        "output_file": "results/answers3.txt"
    }
}
```

Конфигурационный файл разделен на три основные секции:

1) **model** — параметры языковой модели (путь к файлу модели, размер контекста, температура и др.);
2) **prompt** — настройки шаблонов для формирования запросов (системный промпт, шаблон основного запроса, шаблон валидации);
3) **files** — пути к файлам ввода-вывода (входной файл с вопросом, выходной файл для результатов).

Такой подход позволяет модифицировать поведение системы без изменения исходного кода, что повышает простоту поддержки и развития программного решения. Централизация всех настроек в едином файле значительно упрощает конфигурацию и использование программы.

3.4 Необходимое оборудование

Для функционирования разработанного программного решения требуется следующее оборудование и программное обеспечение:

- **процессор**: Многоядерный CPU (Intel Core i5/AMD Ryzen 5 или выше)
- **оперативная память**: Минимум 8 ГБ (рекомендуется 16 ГБ)
- **дисковое пространство**: Не менее 10 ГБ для установки Python и всех зависимостей, а также для языковой модели
- **операционная система**: Windows 11

Приведённые системные требования являются ориентировочными и могут варьироваться в зависимости от конкретной языковой модели, используемой в системе. Более компактные модели потребуют меньше ресурсов, в то время как

модели с большим количеством параметров предъявляют повышенные требования к оперативной памяти и вычислительной мощности процессора. При выборе языковой модели следует учитывать имеющиеся аппаратные ограничения и при необходимости применять методы оптимизации, такие как квантизация, для снижения ресурсоемкости.

Важно отметить, что программное решение оптимизировано для работы на CPU, что обеспечивает его высокую совместимость с широким спектром устройств. Несмотря на то, что использование GPU могло бы ускорить выполнение операций, данный подход позволяет запускать систему на стандартных компьютерах без специализированного оборудования, что существенно расширяет потенциальную аудиторию пользователей.

3.5 Инструкция по установке и использованию

3.5.1 Пошаговая инструкция по установке

Ниже представлена пошаговая инструкция по установке программного решения:

1) **установка Python**:
   - скачайте и установите Python 3.11 с официального сайта: https://www.python.org/ftp/python/3.11.9/python-3.11.9-amd64.exe
   - при установке на Windows обязательно отметьте опцию ”Add Python to PATH”
2) **создание виртуальной среды**:
   - откройте командную строку или терминал
   - скопируйте проект и перейдите в его директорию
   - создайте виртуальную среду Python:

```
python -m venv venv
```

   - активируйте виртуальную среду:

```
venv\Scripts\activate
```

3) **установка зависимостей**:

```
pip install typer pydantic rich llama-cpp-python
```

4) **загрузка языковой модели**:
   — скачайте желаемую языковую модель в формате GGUF (например, t-lite-it-1.0-q4_k_m.gguf) из доверенного источника
   — поместите файл модели в директорию `models`

3.5.2 Инструкция по использованию

После установки программу можно использовать следующим образом:

1) **подготовка конфигурационного файла**:
   — откройте файл `configs/config.json` в текстовом редакторе;
   — укажите путь к входному файлу с вопросом в параметре `files.input_file`;
   — укажите путь для сохранения результатов в параметре `files.output_file`;
   — при необходимости измените настройки модели или шаблона запроса.
2) **подготовка входного файла**:
   — создайте текстовый файл с сложным вопросом по указанному в конфигурации пути;
   — убедитесь, что файл сохранен в кодировке UTF-8.
3) **запуск программы**:
   — активируйте виртуальную среду:

```
venv\Scripts\activate
```

   — запустите программу командой:

```
python .\src\cli_module.py
```

4) **интерпретация результатов**:
   — после завершения работы программы результаты будут выведены в консоль и, при отсутствии ошибок, сохранены в указанный выходной файл;
   — результаты представляют собой пронумерованный список простых вопросов либо сообщение об ошибке.

3.6 Демонстрация работы системы

3.6.1 Примеры входных вопросов

Для демонстрации работы системы использовались следующие сложные вопросы:

Пример 1 (`examples/question1.txt`):

```
Какие факторы повлияли на развитие письменности в
древних цивилизациях и как это изменило политическую,
экономическую и социальную структуру общества?
```

Пример 2 (`examples/question2.txt`):

```
Как экологические проблемы, связанные с загрязнением
океана пластиком, влияют на морских обитателей,
прибрежные экосистемы и здоровье человека, и какие
существуют технологические и законодательные решения
для минимизации этого воздействия?
```

3.6.2 Примеры результатов декомпозиции

Ниже представлены результаты работы системы для указанных выше сложных вопросов:

Результат для вопроса 1 (`results/answers1.txt`):

```
1. Какие экономические факторы повлияли на развитие
письменности в древних цивилизациях?
2. Какие социальные факторы способствовали
возникновению письменности в древних обществах?
3. Какие религиозные или культурные предпосылки привели
к появлению систем письма?
4. Как географические условия повлияли на развитие
письменности в разных регионах?
5. Как развитие письменности изменило политическую
структуру древних обществ?
6. Какое влияние письменность оказала на экономическое
развитие древних цивилизаций?
```

7. Как письменность трансформировала социальную стратификацию в древних обществах?
8. Какую роль играло образование и грамотность в древних цивилизациях после появления письменности?

Результат для вопроса 2 (`results/answers2.txt`):

1. Какие виды пластикового загрязнения наиболее распространены в мировом океане?
2. Как пластиковое загрязнение влияет на физиологию морских млекопитающих?
3. Какое воздействие оказывает микропластик на рыб и морских беспозвоночных?
4. Как пластиковое загрязнение влияет на коралловые рифы?
5. Какие изменения происходят в прибрежных экосистемах из-за накопления пластика?
6. Каким образом загрязнение океана пластиком влияет на здоровье человека?
7. Какие технологии разрабатываются для очистки океана от пластикового мусора?
8. Какие законодательные меры приняты разными странами для сокращения пластикового загрязнения?
9. Какие международные соглашения регулируют проблему загрязнения океана пластиком?
10. Какие альтернативы пластику разрабатываются для снижения его использования?

3.6.3 Пример работы программы в командной строке

Пример работы программы от момента запуска и до получения ответа представлен на рисунке 3.1.

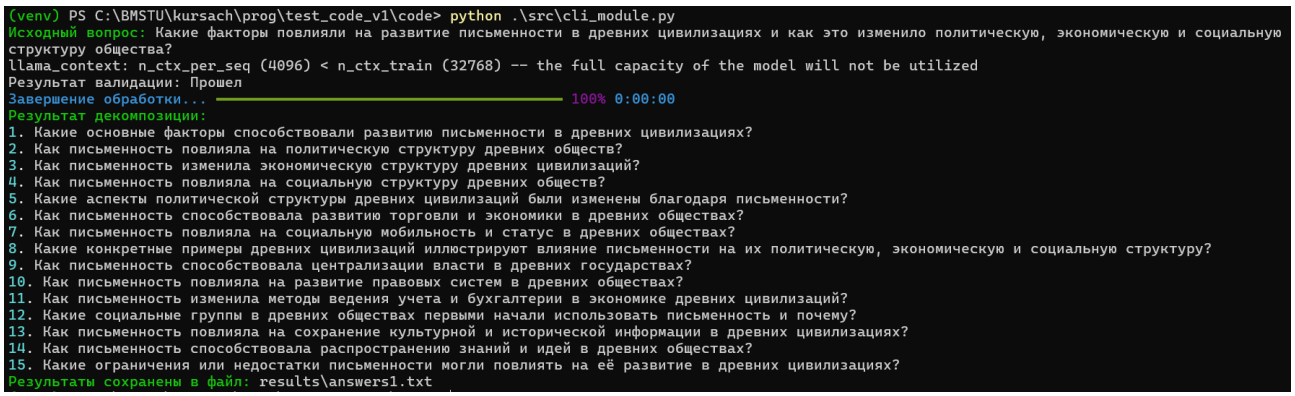


Рисунок 3.1 – Пример выполнения программы в командной строке

3.7 Вывод

В технологическом разделе представлена реализация программного решения для декомпозиции сложных вопросов на простые с использованием локальных языковых моделей.

Для разработки был выбран Python как основной язык программирования, что позволило эффективно использовать существующие библиотеки для работы с языковыми моделями. Решение разделено на три взаимосвязанных модуля: консольный интерфейс для взаимодействия с пользователем, управление запросами для формирования промптов и обработки ответов, и модуль работы с языковой моделью. Такое разделение упрощает понимание кода и дает возможность независимо модифицировать отдельные компоненты системы.

Программа поддерживает настройку через конфигурационные файлы, благодаря чему пользователь может менять параметры работы без изменения исходного кода. Особое внимание было уделено совместимости с разными устройствами – система оптимизирована для работы на CPU, что исключает необходимость в графических ускорителях.

Важным дополнением к системе стал механизм валидации результатов, который проверяет соответствие ответов этическим и правовым нормам с помощью дополнительного запроса к той же языковой модели в роли эксперта. Эта функциональность дополнена системой проверки запрещенных слов, работающей на двух уровнях: входных запросов и генерируемых ответов. Это повышает надежность системы и минимизирует риски получения нежелательного контента.

К проекту прилагается подробная инструкция по установке и использованию, а также примеры декомпозиции сложных вопросов из разных областей

знаний, демонстрирующие возможности системы. Улучшенная обработка ошибок с информативными сообщениями делает программу более дружественной к пользователю.

Созданное решение соответствует всем требованиям, определенным в предыдущих разделах работы. Система эффективно преобразует сложные вопросы в наборы простых компонентов, сохраняя при этом возможность дальнейшего развития функциональности благодаря модульной структуре.

4 Экспериментально-исследовательский раздел

4.1 Методология исследования

Настоящее исследование направлено на сравнительный анализ языковых моделей применительно к задаче декомпозиции сложных вопросов. Основной фокус исследования — оценка качества декомпозиции и изучение практической применимости моделей в условиях ограниченных вычислительных ресурсов. В рамках исследования были проанализированы 30 языковых моделей, рассмотренных в аналитической части работы.

Необходимо отметить, что изначальный план исследования предполагал измерение времени обработки и потребления памяти моделями при выполнении декомпозиции. Однако в ходе работы выяснилось, что большинство рассмотренных в аналитическом разделе языковых моделей требует значительных вычислительных ресурсов, недоступных в рамках данного исследования. По этой причине основной акцент был сделан на экспертной оценке качества декомпозиции.

4.1.1 Тестовая выборка

Для проведения исследования была сформирована тестовая выборка из 10 сложных вопросов, охватывающих различные предметные области. Критериями отбора вопросов служили многокомпонентность (наличие нескольких аспектов, требующих рассмотрения), тематическое разнообразие и структурная сложность. Тестовая выборка представлена в таблице 4.1.

Таблица 4.1 – Тестовая выборка сложных вопросов

№	Вопрос
1	Какие факторы повлияли на развитие письменности в древних цивилизациях и как это изменило политическую, экономическую и социальную структуру общества?
2	Как экологические проблемы, связанные с загрязнением океана пластиком, влияют на морских обитателей, прибрежные экосистемы и здоровье человека, и какие существуют технологические и законодательные решения для минимизации этого воздействия?
3	Каково влияние искусственного интеллекта на рынок труда, образование и этические нормы общества, и как следует регулировать его развитие?
4	Какие технологические прорывы в области квантовых вычислений могут изменить подход к криптографии и обработке больших данных в ближайшие десятилетия?
5	Как глобализация повлияла на культурную идентичность малых народов и какие существуют стратегии сохранения языкового разнообразия?
6	Какие медицинские, этические и юридические аспекты необходимо учитывать при внедрении генной инженерии в клиническую практику?
7	Как развитие возобновляемых источников энергии изменяет геополитическую ситуацию и экономику стран-экспортеров нефти?
8	Какие философские концепции лежат в основе современных подходов к искусственному интеллекту и как они влияют на его восприятие обществом?
9	Какие нейробиологические механизмы ответственны за формирование долговременной памяти и как их понимание может помочь в лечении нейродегенеративных заболеваний?
10	Как цифровизация образования влияет на когнитивное развитие студентов и какие риски она несет для традиционной педагогики?

4.2 Оценка качества декомпозиции

Оценка качества декомпозиции сложных вопросов проводилась экспертным методом. Для этого каждая из 30 исследуемых моделей использовалась для разбиения всех 10 вопросов из тестовой выборки. Полученные результаты ана-

лизировались квалифицированным экспертом с опытом в области лингвистики и обработки естественного языка.

С целью обеспечения объективности оценки был разработан комплекс критериев для стандартизации процесса экспертной оценки. Это позволило минимизировать субъективность и получить сопоставимые результаты для разных моделей.

4.2.1 Критерии экспертной оценки

Оценка качества декомпозиции проводилась по трем ключевым критериям:

Полнота (0-5 баллов): оценивает степень охвата всех смысловых аспектов исходного вопроса. Максимальный балл присваивался, если полученный набор подвопросов полностью покрывал информационную потребность исходного вопроса.

Атомарность (0-5 баллов): оценивает степень фокусировки каждого подвопроса на одном смысловом компоненте. Высокий балл присваивался, если подвопросы не содержали несколько разнородных аспектов и не требовали дальнейшей декомпозиции.

Корректность (0-5 баллов): оценивает грамматическую и логическую целостность подвопросов. Критерий учитывал логичность формулировок, отсутствие грамматических ошибок и стилистическую согласованность.

Процесс получения итоговой оценки включал два этапа усреднения. Сначала для каждой модели по каждому из трех критериев вычислялось среднее арифметическое оценок за все 10 тестовых вопросов. Затем полученные три средних значения (по полноте, атомарности и корректности) усреднялись, формируя итоговый показатель качества декомпозиции. Данный подход обеспечил комплексную оценку производительности моделей с учетом всех аспектов качества декомпозиции.

4.2.2 Результаты оценки

На основе анализа результатов декомпозиции 10 тестовых вопросов была составлена сводная таблица экспертных оценок по всем 30 исследуемым моделям (таблица 4.2). Модели в таблице расположены в порядке убывания среднего балла.

Таблица 4.2 – Сводные результаты экспертной оценки качества декомпозиции сложных вопросов

Модель	**Полнота**	**Атомар-ность**	**Коррект-ность**	**Средний балл**
DeepSeek-V3-Chat	4.8	4.6	4.9	4.77
chatgpt-4o-latest	4.7	4.5	4.8	4.67
claude-3-opus-20240229	4.4	4.5	4.7	4.53
o1-mini	4.5	4.2	4.6	4.43
RuadaptQwen-32B-Pro_v1	4.3	4.1	4.2	4.20
T-Tech-T-pro-it-1.0	4.2	4.1	4.3	4.20
yi-lightning	4.3	4.1	4.2	4.20
gemini-1.5-pro-002	4.3	4.0	4.2	4.17
SberDevices-GigaChatMax	4.0	4.3	4.0	4.10
Qwen2.5-72B-Instruct	4.2	3.8	4.0	4.00
ru-Zero-Mistral-Small-24B	3.9	4.0	4.1	4.00
Phi-4	3.9	3.7	4.3	3.97
llama-3.1-70b-instruct	3.9	3.6	4.1	3.87
vikhr-nemo-12b-instruct-r	3.8	4.0	3.8	3.87
mistral-large-2407	3.8	3.8	3.6	3.73
gemma-2-9b-it	3.6	3.5	3.8	3.63
command-r-plus	3.6	3.6	3.6	3.60
Watari-7b-v1	3.3	3.2	3.2	3.23
gpt-3.5-turbo-0125	2.9	3.5	3.1	3.17
MTSAIR-Cotype-Nano	3.4	2.8	3.2	3.13
yandexgpt-4-pro	3.2	3.1	3.1	3.13
glm-4-9b-chat	2.9	2.8	3.1	2.93
c4ai-command-r-v01	2.7	2.8	2.7	2.73
hermes-2-theta-llama-3-8b	2.6	2.7	2.8	2.70
suzume-llama-3-8b-multilingual	2.7	2.6	2.8	2.70
saiga_llama3_8b_v6	2.5	2.6	2.7	2.60
paralex-llama-3-8b-sft	2.5	2.4	2.6	2.50
aya-23-8b	2.4	2.3	2.5	2.40
storm-7b	2.2	2.2	2.3	2.23
neural-chat-7b-v3-3	2.2	2.1	2.2	2.17

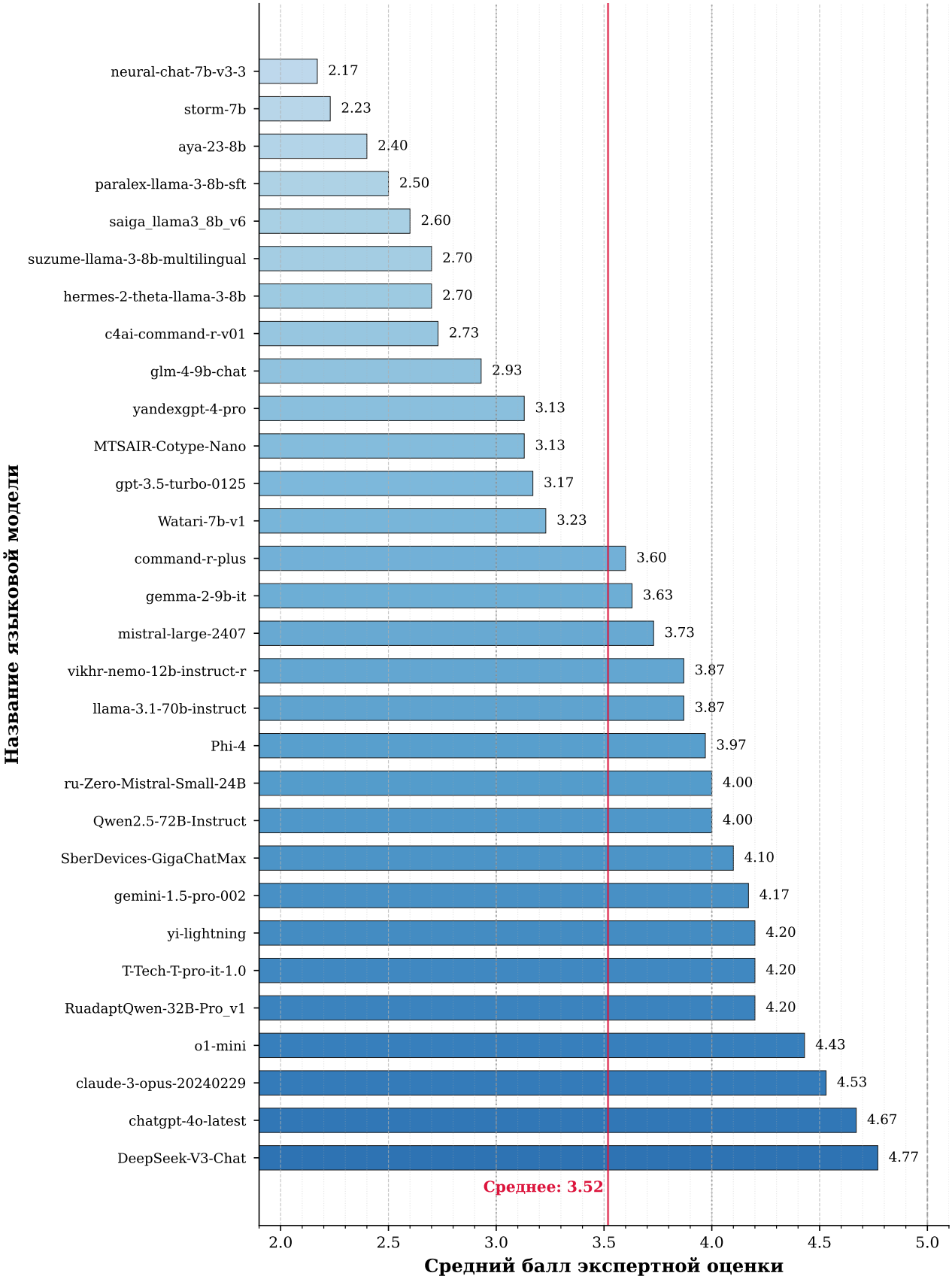


Рисунок 4.1 – Сравнение моделей по интегральному показателю качества

Проведенное исследование показало, что наилучшие результаты демонстрируют проприетарные модели DeepSeek-V3-Chat, chatgpt-4o-latest и Claude-

3-Opus, доступные только через API. Среди локальных моделей высокие результаты показывают T-Tech-T-pro-it-1.0 и RuadaptQwen-32B-Pro_v1 с показателем 4.20 балла, что подтверждает высокое качество этих открытых моделей.

4.3 Практические ограничения

Несмотря на значительное превосходство проприетарных моделей и крупных локальных моделей в качестве декомпозиции, их практическое применение сталкивается с серьезными ограничениями. Эти ограничения имеют существенное влияние на возможность проведения комплексного исследования всех аспектов работы моделей.

Анализ доступности моделей показал, что значительная часть решений с наивысшими показателями качества доступна исключительно через API, что делает невозможным их локальное развертывание. Такие модели как DeepSeek-V3-Chat, chatgpt-4o-latest, claude-3-opus, o1-mini, gemini-1.5-pro, SberDevices-GigaChatMax и другие могут использоваться только при наличии подключения к соответствующим сервисам и с учетом ограничений, устанавливаемых поставщиками этих сервисов.

Среди моделей, теоретически доступных для локального запуска, также существует значительное расслоение по требованиям к вычислительным ресурсам. Крупные модели, такие как LLaMA-3.1-70b-instruct (39.8 ГБ), Qwen2.5-72B (40.6 ГБ) и mistral-large-2407 (45.3 ГБ), требуют не менее 80-92 ГБ оперативной памяти, что существенно превышает возможности стандартных персональных компьютеров и требует специализированного серверного оборудования. Модели среднего размера, включая RuadaptQwen-32B (18.3 ГБ) и T-Tech-T-pro-it-1.0 (16.4 ГБ), хотя и требуют меньше ресурсов, всё равно выходят за рамки возможностей рядового пользовательского оборудования.

Данное ограничение не позволило провести полноценное исследование времени обработки и потребления памяти для большинства моделей, рассмотренных в аналитическом разделе, так как исследовательская среда имела в распоряжении только 8 ГБ оперативной памяти и не обладала специализированным графическим ускорителем, необходимым для эффективной работы крупных моделей.

Более того, сравнительный анализ компактных моделей (<10 ГБ), которые потенциально могли бы быть запущены в исследовательской среде, представля-

ется малоинформативным по двум причинам. Во-первых, большинство таких моделей демонстрируют схожие характеристики производительности при работе на однотипном оборудовании, что обусловлено схожестью их архитектур и методов оптимизации. Во-вторых, детальное измерение производительности требует специализированных методик профилирования, позволяющих отделить собственно время инференса от операций пред- и постобработки текста, что выходит за рамки настоящего исследования.

Таким образом, основной фокус исследования был смещен в сторону оценки качества декомпозиции, которое может быть надежно измерено для всех моделей независимо от их доступности для локального запуска.

4.3.1 Выбор модели для практической реализации

Учитывая описанные выше ограничения и результаты экспертной оценки, для практической реализации системы декомпозиции вопросов была выбрана локальная квантованная модель T-lite-it-1.0-q4_k_m из семейства T Bank. Данный выбор обусловлен следующими факторами:

— модель T-Tech-T-pro-it-1.0 из того же семейства продемонстрировала наилучшие результаты среди локальных моделей с оценкой 4.20 балла, что указывает на высокую эффективность архитектуры и обучения моделей этого семейства;
— несмотря на превосходные результаты, модель T-Tech-T-pro-it-1.0 требует 36 ГБ оперативной памяти, что превышает доступные ресурсы исследовательской среды;
— компактная модель T-lite-it-1.0-q4_k_m из того же семейства сохраняет базовые архитектурные особенности и обучающие данные, имея при этом существенно меньшие требования к ресурсам;
— размер модели составляет 4.68 ГБ, что позволяет запускать ее на стандартном персональном компьютере с ограниченными ресурсами;
— требуемый объем оперативной памяти не превышает 6 ГБ, что делает модель доступной для подавляющего большинства современных компьютеров;
— квантованная архитектура модели обеспечивает приемлемую скорость работы даже без использования графического ускорителя. [45]

4.4 Вывод

Проприетарные модели, доступные через API, демонстрируют наивысшее качество декомпозиции сложных вопросов с оценками 4.77, 4.67 и 4.53 баллов для DeepSeek-V3-Chat, chatgpt-4o-latest и Claude-3-Opus соответственно. Однако их использование требует постоянного подключения к интернету и зависит от доступности и политики стороннего сервиса.

Среди локальных моделей наилучшие результаты показывает T-Tech-T-pro-it-1.0 с оценкой 4.20 баллов, что подтверждает высокое качество моделей семейства T Bank. Данная модель может считаться эталоном качества для решений, работающих без доступа к сети, однако требует значительных вычислительных ресурсов, недоступных на стандартных персональных компьютерах.

Квантованная модель T-lite-it-1.0-q4_k_m представляет собой оптимальный компромисс между качеством декомпозиции и возможностью запуска на потребительском оборудовании. Несмотря на более низкий показатель качества по сравнению с проприетарными решениями, модель обеспечивает приемлемый уровень декомпозиции и может быть успешно использована в практических приложениях.

Таким образом, исследование показало возможность эффективной декомпозиции сложных вопросов с использованием языковых моделей различного масштаба, что открывает перспективы для создания практических приложений, способных работать в условиях ограниченных вычислительных ресурсов.

5 Организационно-правовой раздел

5.1 Лицензионная политика проекта

Программное решение распространяется под лицензией Apache 2.0 [46], что обусловлено требованиями используемой языковой модели T-lite-it-1.0 [47] от T Bank и необходимостью обеспечения совместимости с компонентами, лицензированными под MIT [48].

5.1.1 Обоснование выбора лицензии

Проект включает библиотеки `typer` [49], `pydantic` [50], `rich` [51] и `llama-cpp-python` [52], распространяемые под лицензией MIT. Совместимость лицензий MIT и Apache 2.0 подтверждается сравнительным анализом их требований (табл. 5.1). Текст лицензии Apache 2.0 находится в приложении А.

5.1.2 Правовые аспекты использования лицензии

Анализ таблицы 5.1 показывает, что Apache 2.0 по всем критериям либо эквивалентна MIT, либо накладывает более строгие ограничения. Согласно принципам лицензионной совместимости, подтвержденным Open Source Initiative [53], компоненты с более открытой лицензией (MIT) могут включаться в проекты под более строгой лицензией (Apache 2.0) при соблюдении требований обеих лицензий.

Ключевые преимущества выбранной лицензионной политики:

— правовая защита посредством явного предоставления патентных прав;
— чёткие требования к указанию модификаций, обеспечивающие прозрачность изменений;
— сохранение авторских прав разработчиков;
— возможность коммерческого использования с соблюдением условий лицензии.

Для учебных целей применение программного решения дополнительно защищено ст. 1274 ГК РФ, разрешающей свободное использование объектов авторского права в образовательном процессе.

Таблица 5.1 – Сравнительный анализ лицензий MIT и Apache 2.0

Критерий	**Лицензия MIT**	**Лицензия Apache 2.0**	**Сравнение**
Коммерческое использование	Разрешено	Разрешено	Одинаково
Модификация	Разрешено без ограничений	Разрешено с требованием указания изменений	Apache 2.0 строже
Распространение	Разрешено	Разрешено с дополнительными требованиями	Apache 2.0 строже
Частное использование	Разрешено	Разрешено	Одинаково
Указание изменений	Не требуется явно	Требуется явное указание	Apache 2.0 строже
Защита авторских прав	Требует включения уведомления	Требует сохранения всех уведомлений	Apache 2.0 строже
Патентные права	Нет явного предоставления	Явное предоставление патентных прав	Apache 2.0 строже
Ограничение торговых марок	Отсутствует	Прямо запрещает использование	Apache 2.0 строже
Распространение производных	Разрешено без особых условий	Разрешено с дополнительными условиями	Apache 2.0 строже

5.2 Меры правовой безопасности

5.2.1 Многоуровневая валидация контента

Фильтрация контента обязательна согласно ст. 10.1 ФЗ-149 «Об информации, информационных технологиях и о защите информации», требующей блокировки запрещённой информации (экстремизм, призывы к насилию), а также ФЗ-114 «О противодействии экстремистской деятельности». Несоблюдение влечёт ответственность по ст. 282 УК РФ и ст. 13.41 КоАП РФ, а также риск блокировки проекта.

Архитектура системы включает комплексный подход к обеспечению правовой безопасности, сочетающий жёсткую и мягкую фильтрацию контента:

1) **жёсткая фильтрация** основана на автоматической проверке текста по спискам запрещённых слов и словосочетаний, формируемым на основе официальных источников:
   - список экстремистских материалов Министерства юстиции РФ [54];
   - единый реестр запрещенной информации Роскомнадзора [55];
   - перечень запрещённой информации согласно постановлениям Верховного Суда РФ [56];
   - список экстремистских и террористических организаций, признанных таковыми Генеральной прокуратурой РФ [57];
   - рекомендации Минцифры [58].
2) **мягкая фильтрация** реализуется посредством нейросетевой модели, которая:
   - на этапе предварительной обработки получает системные инструкции с явными запретами на генерацию контента, нарушающего законодательство РФ, включая ст. 282 УК РФ, ФЗ-149;
   - выполняет семантический анализ содержания, способный идентифицировать противоправный контент даже при использовании эвфемизмов или нестандартных формулировок.

На этапе постобработки ответ модели анализируется через дополнительный валидационный запрос. При обнаружении потенциальных нарушений система блокирует выдачу контента и прекращает обработку запроса.

5.2.2 Формирование и обновление списков запрещённых слов

Базовые списки запрещённых слов и выражений формируются на основе официальных источников регулирующих органов РФ, которые были указаны выше.

Обязанность пользователя: В соответствии с требованиями ст. 10.1 ФЗ-149, пользователь несёт полную ответственность за:

- регулярное обновление встроенных списков запрещённых слов и выражений из вышеуказанных официальных источников, в соответствии с изменениями в действующем законодательстве;
- самостоятельное дополнение базовых списков с учётом специфики использования программы и отраслевых требований;
- проверку актуальности используемых фильтров перед каждым примене-

нием программного решения.

Система предоставляет техническую возможность для обновления списков, но **не осуществляет их автоматическое обновление**. Вся ответственность за актуальность и полноту списков запрещённых слов и выражений лежит исключительно на пользователе программного решения.

5.2.3 Обработка персональных данных

В соответствии с требованиями ФЗ-152 «О персональных данных» и ФЗ-149 «Об информации, информационных технологиях и о защите информации»:

- программное решение **не предназначено** для обработки персональных данных и **не должно использоваться** для этих целей;
- система настроена на блокировку генерации и обработки контента, содержащего персональные данные;
- **пользователю категорически запрещается** вводить персональные данные субъектов в систему;
- в случае ввода персональных данных в систему вся ответственность за их обработку в соответствии со ст. 13.11 КоАП РФ и ФЗ-152 возлагается исключительно на пользователя.

При обнаружении в запросе признаков персональных данных система предупреждает пользователя о недопустимости их обработки и может отказать в выполнении запроса.

5.2.4 Ответственность пользователя

Перед каждым использованием программы пользователь обязан:

- самостоятельно убедиться в актуальности используемых списков запрещённых слов и выражений путём проверки официальных источников;
- обновить базы данных запрещённых выражений в соответствии с изменениями в законодательстве;
- дополнить систему фильтрации собственными ограничениями в соответствии с конкретными задачами использования системы;
- верифицировать результаты работы системы на соответствие действующему законодательству РФ.

В соответствии с ст. 13.41 КоАП РФ и общими положениями гражданского законодательства, пользователь несёт полную юридическую ответственность за:

— актуальность и полноту используемых фильтров запрещённой информации;
— недопущение обработки персональных данных с использованием системы;
— использование и распространение информации, полученной с помощью данного программного обеспечения.

Разработчик не несёт ответственности за последствия использования программы в нарушение действующего законодательства, несмотря на наличие встроенных механизмов защиты.

5.2.5 Ограничения и риски

Система подвержена следующим технико-правовым ограничениям: появление ложных срабатываний при обработке специализированных научных, медицинских и юридических терминов; возможность обхода фильтров через сложные лингвистические конструкции; недостаточная точность интерпретации контекста при анализе многозначных выражений. Эффективность фильтрации зависит от регулярности обновления пользователем баз запрещённых выражений в соответствии с изменениями законодательства.

5.3 Вывод

Реализованные правовые механизмы обеспечивают соответствие проекта требованиям ФЗ-149, ФЗ-114 и УК РФ за счёт многоуровневой фильтрации контента, основанной на официальных списках запрещённых материалов и нейросетевом анализе. Лицензия Apache 2.0 гарантирует совместимость с компонентами MIT, а распределение ответственности (обновление фильтров, запрет на обработку персональных данных) минимизирует юридические риски.

ЗАКЛЮЧЕНИЕ

В рамках курсового проекта разработано программное решение для автоматической декомпозиции сложных вопросов на простые с использованием больших языковых моделей.

Анализ существующих подходов показал преимущества применения языковых моделей над экспертными и алгоритмическими методами: отсутствие необходимости в разработке сложных правил, высокая адаптивность к различным типам вопросов и эффективное сохранение контекста между частями исходного вопроса.

Разработанная архитектура включает три взаимосвязанных модуля: консольный интерфейс, управление запросами и взаимодействие с языковой моделью. Реализация на Python обеспечивает работу на стандартном оборудовании без необходимости в графических ускорителях. Внедрена двухуровневая система валидации контента для входных запросов и генерируемых ответов.

Экспериментальное исследование выявило, что проприетарные модели (DeepSeek-V3-Chat, ChatGPT-4, Claude-3-Opus) демонстрируют наивысшее качество декомпозиции, однако требуют постоянного подключения к интернету. Среди локальных решений выделяются T-Tech-T-pro-it-1.0 и RuadaptQwen-32B-Pro_v1. Для практической реализации выбрана компактная квантованная модель T-lite-it-1.0-q4_k_m как оптимальный компромисс между качеством и доступностью на пользовательском оборудовании.

Программное решение распространяется под лицензией Apache 2.0 и соответствует требованиям законодательства РФ благодаря реализованной фильтрации контента на основе официальных источников.

В результате созданное программное решение успешно выполняет декомпозицию сложных вопросов на простые компоненты и может применяться в образовательных системах, системах поддержки принятия решений и других приложениях обработки естественного языка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Измерение и сужение разрыва композиционности в языковых моделях [Электронный ресурс] / О. Пресс, М. Жанг, С. Мин, Л. Шмидт, Н. Смит, М. Льюис // CoRR. — 2023. — Режим доступа: https://arxiv.org/abs/2210.03350 (дата обращения: 20.02.2025).
2. Неконтролируемая декомпозиция вопросов для систем вопросно-ответного поиска [Электронный ресурс] / Э. Перес, П. Льюис, В. Йи, К. Чо, Д. Киела // CoRR. — 2020. — Режим доступа: https://arxiv.org/abs/2002.09758 (дата обращения: 20.02.2025).
3. Количественный анализ аргументов и не только: межпредметный анализ ключевых точек [Электронный ресурс] / Р. Бар-Хаим, Й. Кантор, Л. Эден, Р. Фридман, Д. Лахав, Н. Слоним // CoRR. — 2020. — Режим доступа: https://arxiv.org/abs/2010.05369 (дата обращения: 20.02.2025).
4. *Ванг Ж.*, *Гуо Б.*, *Чен Л.* Машинное обучение с участием человека: макро-микро перспектива [Электронный ресурс] // CoRR. — 2022. — Режим доступа: https://arxiv.org/abs/2202.10564 (дата обращения: 20.02.2025).
5. Декомпозиция вопросов на основе EDG для ответов на сложные вопросы с использованием баз знаний / С. Ху, И. Шу, С. Хуанг, Ю. Цюй // The Semantic Web – ISWC 2021. — Cham : Springer International Publishing, 2021. — С. 128—145. — Режим доступа: https://www.researchgate.net/publication/354925951 (дата обращения: 20.02.2025).
6. *Хассон М.*, *Берант Д.* Декомпозиция вопросов с использованием графов зависимостей [Электронный ресурс] // CoRR. — 2021. — Режим доступа: https://arxiv.org/abs/2104.08647 (дата обращения: 20.02.2025).
7. SPARQA: семантический парсинг на основе скелетов для сложных вопросов к базам знаний [Электронный ресурс] / Я. Сун, Л. Жанг, Г. Ченг, Ю. Цюй // CoRR. — 2020. — Режим доступа: https://arxiv.org/abs/2003.13956 (дата обращения: 20.02.2025).
8. Декомпозиция сложных вопросов для семантического парсинга / Х. Жанг, Д. Цай, Д. Сюй, Д. Ванг // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association

for Computational Linguistics, 2019. — С. 4477—4486. — Режим доступа: https://aclanthology.org/P19-1440.pdf (дата обращения: 20.02.2025).

9. *Венктеш В.*, *Бхаттачарья С.*, *Ананд А.* Перенос контекстных способностей для декомпозиции вопросов в сложных системах вопросно-ответного поиска [Электронный ресурс] // CoRR. — 2023. — Режим доступа: https://arxiv.org/abs/2310.18371 (дата обращения: 24.12.2024).
10. DeepSeek [Электронный ресурс]. — DeepSeek. — Режим доступа: https://www.deepseek.com (дата обращения: 20.02.2025).
11. ChatGPT-4 [Электронный ресурс]. — OpenAI. — Режим доступа: https://openai.com (дата обращения: 20.02.2025).
12. OpenAI O1-Mini [Электронный ресурс]. — OpenAI. — Режим доступа: https://openai.com/index/openai-o1-mini-advancing-cost-efficient-reasoning (дата обращения: 20.02.2025).
13. Yi Models [Электронный ресурс]. — 01.AI. — Режим доступа: https://01.ai (дата обращения: 20.02.2025).
14. Claude [Электронный ресурс]. — Anthropic. — Режим доступа: https://www.anthropic.com (дата обращения: 20.02.2025).
15. T-Tech T-pro-it-1.0 [Электронный ресурс]. — T-Tech. — Режим доступа: https://huggingface.co/t-tech/T-pro-it-1.0 (дата обращения: 20.02.2025).
16. GigaChat [Электронный ресурс]. — Sber. — Режим доступа: https://giga.chat (дата обращения: 20.02.2025).
17. Gemini [Электронный ресурс]. — Google DeepMind. — Режим доступа: https://deepmind.google/technologies/gemini (дата обращения: 20.02.2025).
18. Qwen [Электронный ресурс]. — Alibaba Cloud. — Режим доступа: https://qwenlm.github.io (дата обращения: 20.02.2025).
19. Phi-4 [Электронный ресурс]. — Microsoft. — Режим доступа: https://huggingface.co/microsoft/phi-4 (дата обращения: 20.02.2025).
20. LLaMA [Электронный ресурс]. — Meta. — Режим доступа: https://www.llama.com (дата обращения: 20.02.2025).
21. Mistral AI [Электронный ресурс]. — Mistral AI. — Режим доступа: https://mistral.ai (дата обращения: 20.02.2025).

22. Cohere Command [Электронный ресурс]. — Cohere. — Режим доступа: https://cohere.com/command (дата обращения: 20.02.2025).

23. Gemma [Электронный ресурс]. — Google. — Режим доступа: https://ai.google.dev/gemma (дата обращения: 20.02.2025).

24. Watari-7b [Электронный ресурс]. — Attn Signs. — Режим доступа: https://huggingface.co/attn-signs/Watari-7b-v1 (дата обращения: 20.02.2025).

25. YandexGPT [Электронный ресурс]. — Yandex. — Режим доступа: https://ya.ru/ai/gpt-4 (дата обращения: 20.02.2025).

26. GPT-3.5 Turbo [Электронный ресурс]. — OpenAI. — Режим доступа: https://platform.openai.com/docs/models#gpt-3-5-turbo (дата обращения: 20.02.2025).

27. GLM-4 [Электронный ресурс]. — Zhipu AI. — Режим доступа: https://bigmodel.cn/dev/howuse/glm-4 (дата обращения: 20.02.2025).

28. C4AI Command-R [Электронный ресурс]. — Cohere For AI. — Режим доступа: https://huggingface.co/CohereForAI/c4ai-command-r-v01 (дата обращения: 20.02.2025).

29. Suzume LLaMA-3 [Электронный ресурс]. — Light Blue. — Режим доступа: https://huggingface.co/lightblue/suzume-llama-3-8B-multilingual (дата обращения: 20.02.2025).

30. Hermes-2 [Электронный ресурс]. — Nous Research. — Режим доступа: https://huggingface.co/NousResearch/Hermes-2-Theta-Llama-3-8B (дата обращения: 20.02.2025).

31. Saiga LLaMA-3 [Электронный ресурс]. — Ilya Gusev. — Режим доступа: https://huggingface.co/IlyaGusev/saiga_llama3_8b (дата обращения: 20.02.2025).

32. Aya-23-8B [Электронный ресурс]. — Cohere For AI. — Режим доступа: https://huggingface.co/CohereForAI/aya-23-8B (дата обращения: 20.02.2025).

33. ParaLex LLaMA-3 [Электронный ресурс]. — Hivaze. — Режим доступа: https://huggingface.co/hivaze/ParaLex-Llama-3-8B-SFT (дата обращения: 20.02.2025).

34. Storm-7B [Электронный ресурс]. — Jie Liu. — Режим доступа: https://huggingface.co/jieliu/Storm-7B (дата обращения: 20.02.2025).

35. Neural Chat [Электронный ресурс]. — Intel. — Режим доступа: https://huggingface.co/Intel/neural-chat-7b-v3-3 (дата обращения: 20.02.2025).

36. Vikhr Nemo [Электронный ресурс]. — Vikhr Models. — Режим доступа: https://huggingface.co/Vikhrmodels/Vikhr-Nemo-12B-Instruct-R-21-09-24 (дата обращения: 20.02.2025).

37. RuadaptQwen [Электронный ресурс]. — Hugging Face. — Режим доступа: https://huggingface.co/msu-rcc-lair/RuadaptQwen2.5-32B-Instruct (дата обращения: 20.02.2025).

38. ru-Zero-Mistral-Small-24B [Электронный ресурс]. — Hugging Face. — Режим доступа: https://huggingface.co/ZeroAgency/Zero-Mistral-Small-24B-Instruct-2501 (дата обращения: 20.02.2025).

39. Cotype-Nano [Электронный ресурс]. — MTSAIR. — Режим доступа: https://huggingface.co/MTSAIR/Cotype-Nano (дата обращения: 20.02.2025).

40. LLM Arena: платформа оценки языковых моделей [Электронный ресурс]. — Режим доступа: https://llmarena.ru (дата обращения: 24.02.2025).

41. Фреймворк llama-cpp-python [Электронный ресурс]. — Режим доступа: https://pypi.org/project/llama-cpp-python/ (дата обращения: 12.03.2025).

42. Фреймворк typer [Электронный ресурс]. — Режим доступа: https://pypi.org/project/typer/ (дата обращения: 12.03.2025).

43. Фреймворк rich [Электронный ресурс]. — Режим доступа: https://pypi.org/project/rich/ (дата обращения: 12.03.2025).

44. Фреймворк pydantic [Электронный ресурс]. — Режим доступа: https://pypi.org/project/pydantic/ (дата обращения: 12.03.2025).

45. T-lite-it-1.0-Q4 [Электронный ресурс]. — Режим доступа: https://huggingface.co/aovchinnikov/T-lite-it-1.0-Q4_K_M-GGUF (дата обращения: 10.04.2025).

46. Лицензия Apache 2.0 [Электронный ресурс]. — Режим доступа: https://www.apache.org/licenses/LICENSE-2.0 (дата обращения: 11.05.2025).

47. Языковая модель T-lite-it-1.0 [Электронный ресурс]. — Т Банк. — Режим доступа: https://huggingface.co/t-tech/T-lite-it-1.0 (дата обращения: 11.05.2025).

48. The MIT License [Электронный ресурс]. — Open Source Initiative. — Режим доступа: https://opensource.org/license/mit (дата обращения: 11.05.2025).

49. Лицензия фреймворка typer [Электронный ресурс]. — Режим доступа: https://github.com/fastapi/typer/blob/master/LICENSE (дата обращения: 11.05.2025).

50. Лицензия фреймворка pydantic [Электронный ресурс]. — Режим доступа: https://github.com/pydantic/pydantic/blob/main/LICENSE (дата обращения: 11.05.2025).

51. Лицензия фреймворка rich [Электронный ресурс]. — Режим доступа: https://github.com/Textualize/rich/blob/master/LICENSE (дата обращения: 11.05.2025).

52. Лицензия фреймворка llama-cpp-python [Электронный ресурс]. — Режим доступа: https://github.com/abetlen/llama-cpp-python/blob/main/LICENSE.md (дата обращения: 11.05.2025).

53. Open Source Licenses by Category [Электронный ресурс]. — Open Source Initiative. — Режим доступа: https://opensource.org/licenses (дата обращения: 11.05.2025).

54. Список экстремистских материалов Министерства юстиции РФ [Электронный ресурс]. — Министерство юстиции РФ. — Режим доступа: https://minjust.gov.ru/ru/extremist-materials (дата обращения: 11.05.2025).

55. Единый реестр запрещенной информации [Электронный ресурс]. — Роскомнадзор. — Режим доступа: https://rkn.gov.ru/activity/electronic-communications/eais/ (дата обращения: 11.05.2025).

56. Перечень запрещённой информации согласно постановлениям Верховного Суда РФ [Электронный ресурс]. — Режим доступа: https://vsrf.ru/ (дата обращения: 11.05.2025).

57. Прокуратура РФ [Электронный ресурс]. — Режим доступа: https://epp.genproc.gov.ru/web/gprf (дата обращения: 11.05.2025).

58. Минцифры России [Электронный ресурс]. — Режим доступа: https://digital.gov.ru/ (дата обращения: 11.05.2025).

ПРИЛОЖЕНИЕ А

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

”License”shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

”Licensor”shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

”Legal Entity”shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, ”control”means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50outstanding shares, or (iii) beneficial ownership of such entity.

”You”(or ”Your”) shall mean an individual or Legal Entity exercising permissions granted by this License.

”Source”form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

”Object”form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

”Work”shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

”Derivative Works”shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

”Contribution”shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works

thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, ”submitted”means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as ”Not a Contribution.”

”Contributor”shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a ”NOTICE”text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in

writing, Licensor provides the Work (and each Contributor provides its Contributions) on an ”AS IS”BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets ”[]”replaced with your own identifying information. (Don’t include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same ”printed page”as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the ”License”); you may not

use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an ”AS IS”BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

ПРИЛОЖЕНИЕ Б

Презентация, состоящая из 17 слайдов.

Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Курсовой проект

Программное решение на основе искусственного интеллекта для разбиения сложных вопросов на простые с использованием больших языковых моделей

Студент: Гиричев Марк Сергеевич ИУ7-46Б
Научный руководитель: Строганов Юрий Владимирович
Консультанты: Якуба Дмитрий Васильевич
Москвичёв Николай Владимирович
Левиев Дмитрий Олегович

Цель и задачи работы

Цель работы: Разработка программного решения для разбиения сложных вопросов на простые вопросы с использованием больших языковых моделей.

Задачи:

1. Проанализировать существующие методы декомпозиции вопросов.
2. Разработать алгоритм декомпозиции на основе больших языковых моделей.
3. Реализовать программное решение.
4. Провести исследование эффективности разработанного решения.
5. Проанализировать правовые аспекты программной реализации и выбрать лицензию.

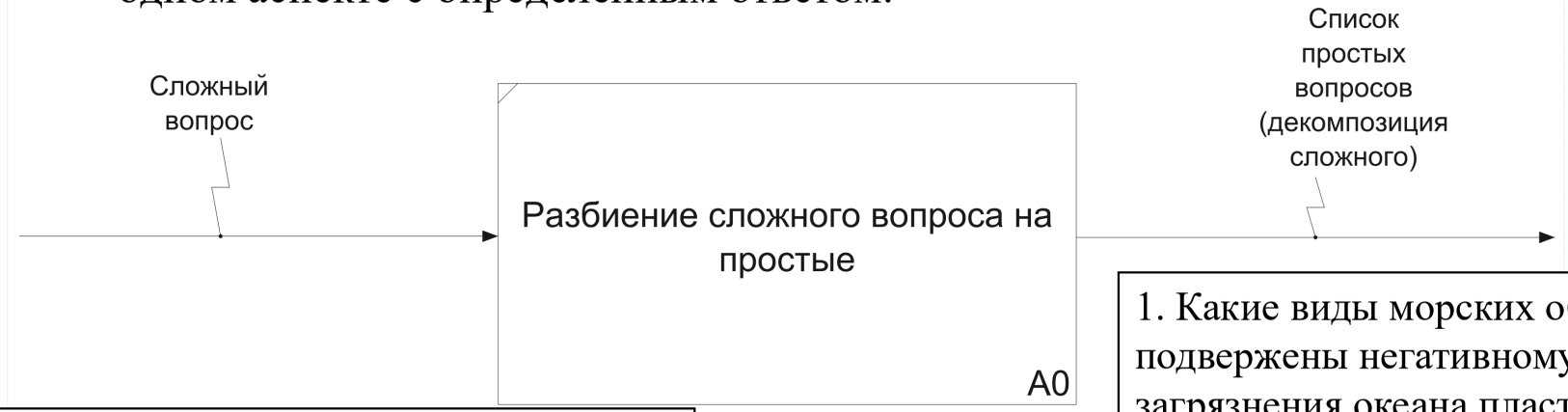
Рейтинг языковых моделей (LLM Arena)

Модель	Общий балл	Доверит. интервал	Ср. кол-во токенов в отв.	Станд. отклон. кол-ва токенов	Оценка языковой комптенц.
DeepSeek-V3-Chat	96.30	+0.8/-0.7	665.97	504.83	56.62
chatgpt-4o-latest	94.74	+0.8/-1.0	693.15	634.20	56.40
o1-mini	93.46	+0.7/-0.8	791.18	647.74	56.22
yi-lightning	93.46	+0.9/-1.0	636.68	469.74	56.22
RuadaptQwen-32B-Pro_v1	92.18	+1.2/-1.2	563.43	387.83	56.04
claude-3-opus-20240229	91.31	+1.0/-1.1	468.69	254.10	55.92
T-Tech-T-pro-it-1.0	90.87	+0.8/-1.1	502.00	380.68	55.85
SberDevices-GigaChatMax	89.96	+1.2/-1.4	523.95	421.87	55.73

Функциональная модель процесса

Сложный вопрос — вопрос, содержащий несколько аспектов и требующий многокомпонентного ответа.

Простой вопрос —самодостаточный атомарный вопрос, фокусирующийся на одном аспекте с определенным ответом.

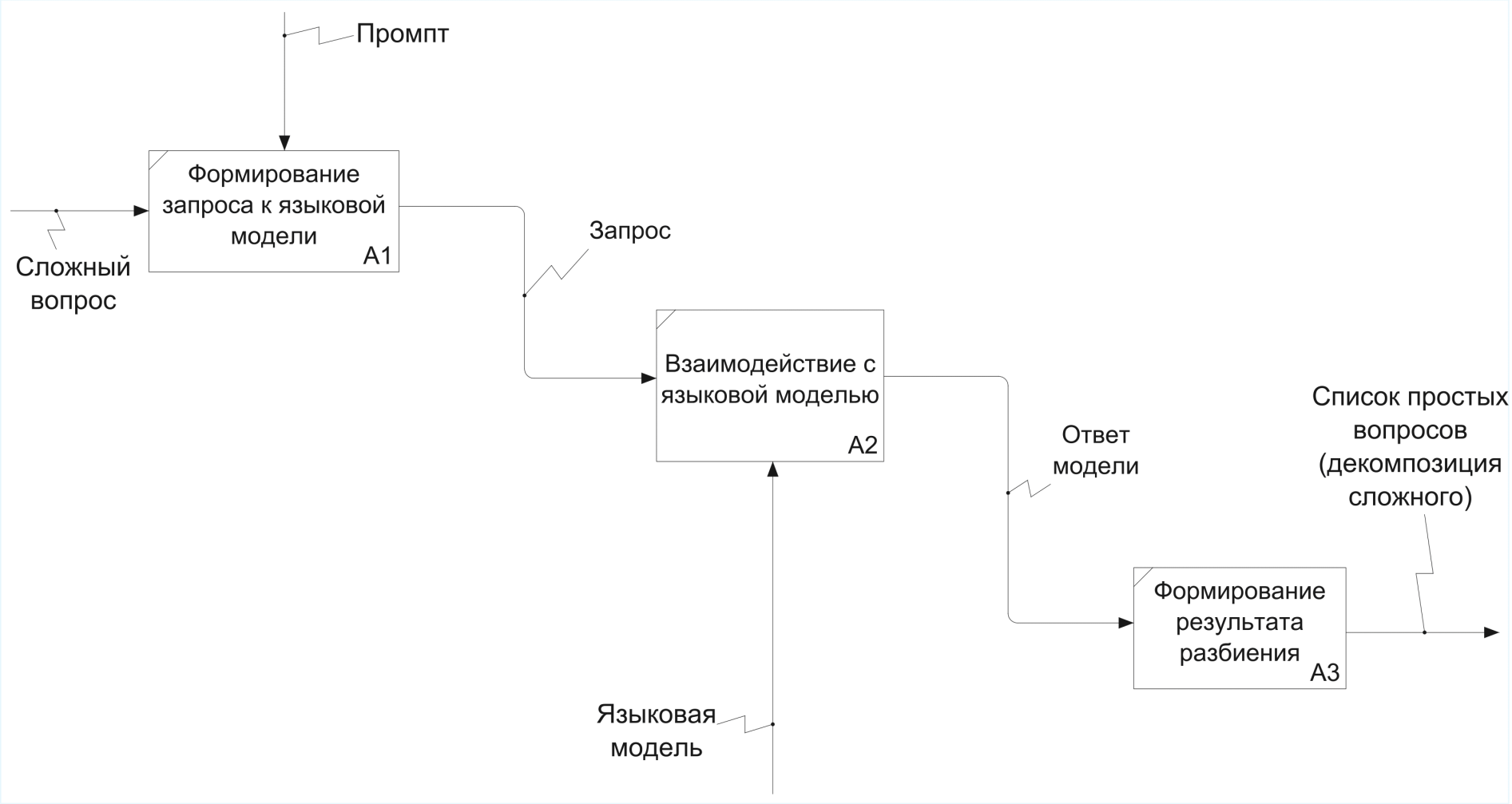


Пример: Как экологические проблемы, связанные с загрязнением океана пластиком, влияют на морских обитателей, прибрежные экосистемы и здоровье человека, и какие существуют технологические и законодательные решения для минимизации этого воздействия?

1. Какие виды морских обитателей наиболее подвержены негативному воздействию загрязнения океана пластиком?
2. Какие конкретные физиологические и поведенческие последствия вызывает загрязнение пластиком у морских обитателей?
3. Как загрязнение океана пластиком влияет на прибрежные экосистемы?
4. Какие виды пластикового загрязнения наиболее опасны для прибрежных экосистем?

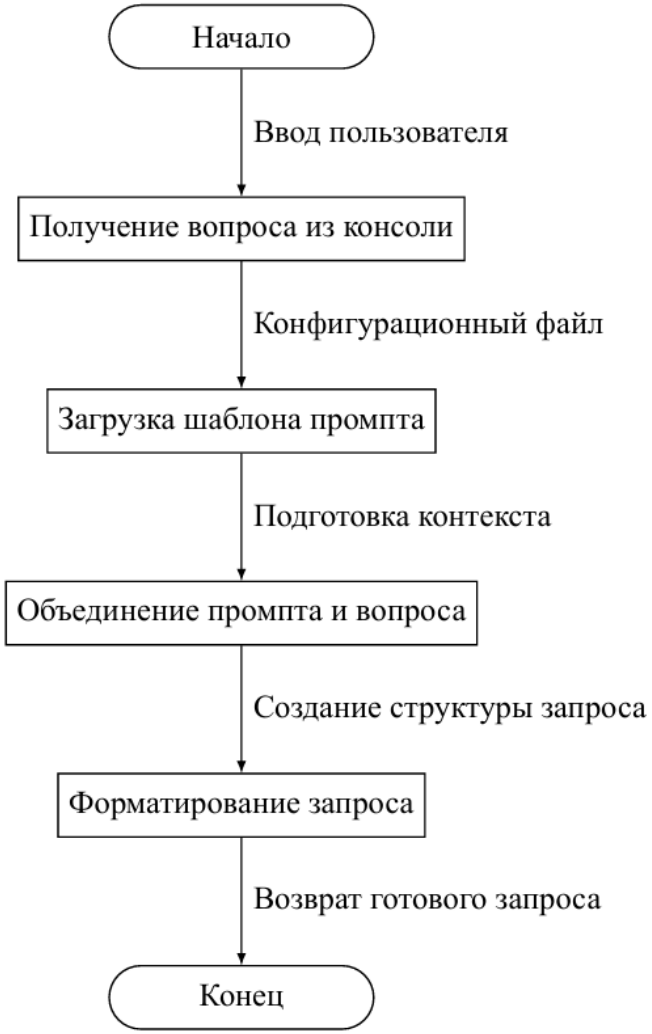
и т.д. ещё 14 простых вопросов…

Функциональная модель системы

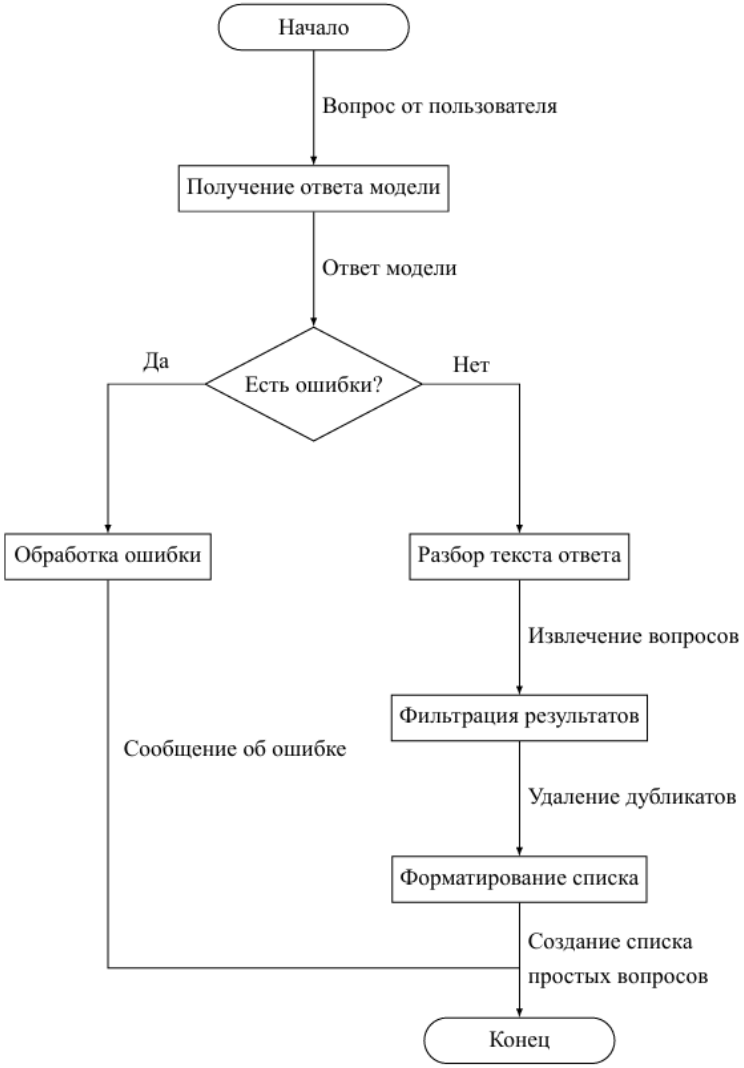


Алгоритм формирования запроса к БЯМ (А1)

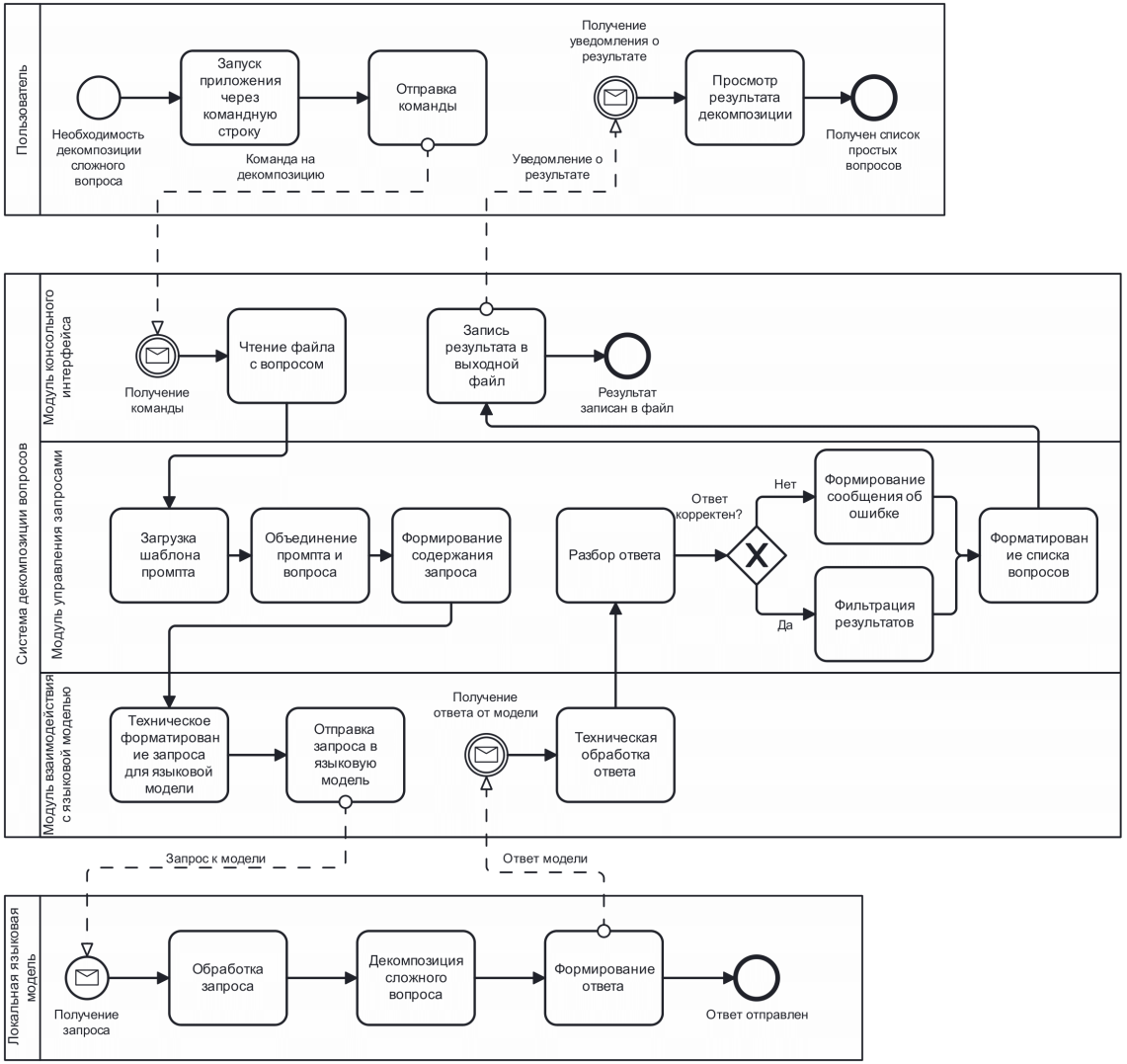
БЯМ – большая языковая модель



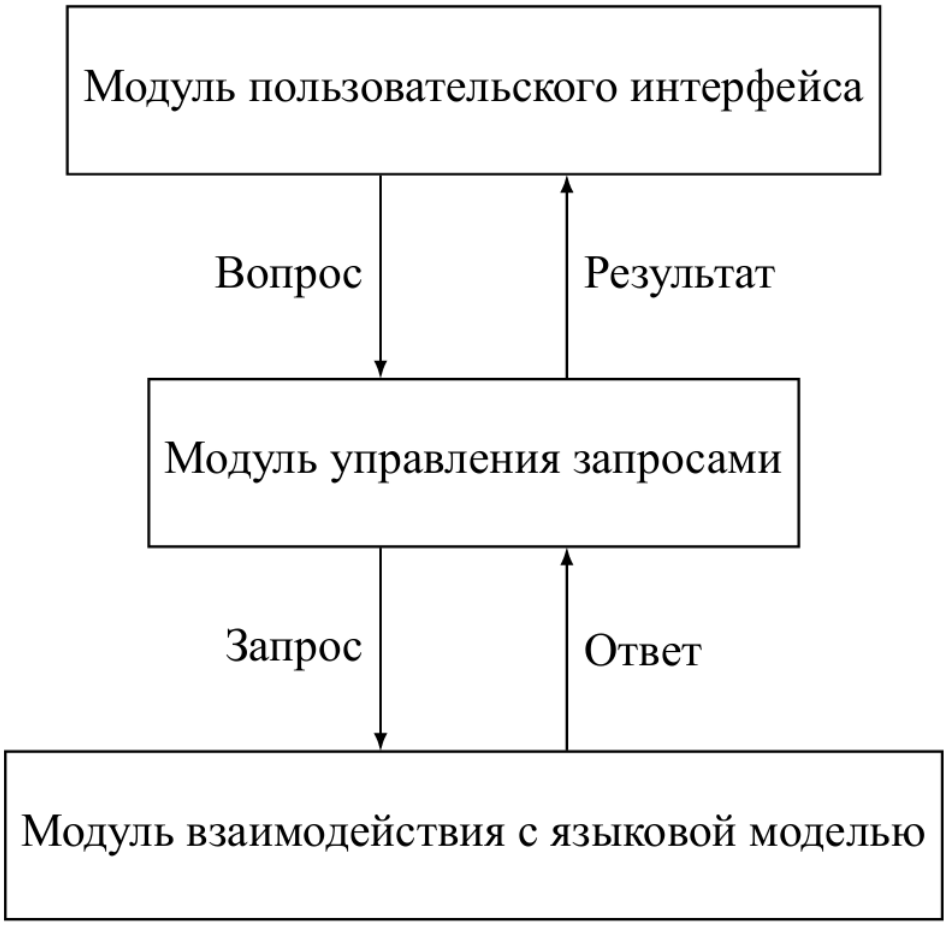
Алгоритм взаимодействия с БЯМ (А2/А3)



BPMN-схема декомпозиции вопроса



Структура программного решения



Используемые библиотеки:

- **llama-cpp-python** – работа с локальными моделями
- **typer** – консольный интерфейс
- **pydantic** – валидация данных
- **rich** - форматированный вывод

Пример работы программы

```
(venv) PS C:\BMSTU\kursach\prog\test_code_v1\code> python .\src\cli_module.py
Исходный вопрос: Какие факторы повлияли на развитие письменности в древних цивилизациях и как это изменило политическую, экономическую и социальную
структуру общества?
llama_context: n_ctx_per_seq (4096) < n_ctx_train (32768) -- the full capacity of the model will not be utilized
Результат валидации: Прошел
Завершение обработки... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00
Результат декомпозиции:
1. Какие основные факторы способствовали развитию письменности в древних цивилизациях?
2. Как письменность повлияла на политическую структуру древних обществ?
3. Как письменность изменила экономическую структуру древних цивилизаций?
4. Как письменность повлияла на социальную структуру древних обществ?
5. Какие аспекты политической структуры древних цивилизаций были изменены благодаря письменности?
6. Как письменность способствовала развитию торговли и экономики в древних обществах?
7. Как письменность повлияла на социальную мобильность и статус в древних обществах?
8. Какие конкретные примеры древних цивилизаций иллюстрируют влияние письменности на их политическую, экономическую и социальную структуру?
9. Как письменность способствовала централизации власти в древних государствах?
10. Как письменность повлияла на развитие правовых систем в древних обществах?
11. Как письменность изменила методы ведения учета и бухгалтерии в экономике древних цивилизаций?
12. Какие социальные группы в древних обществах первыми начали использовать письменность и почему?
13. Как письменность повлияла на сохранение культурной и исторической информации в древних цивилизациях?
14. Как письменность способствовала распространению знаний и идей в древних обществах?
15. Какие ограничения или недостатки письменности могли повлиять на её развитие в древних цивилизациях?
Результаты сохранены в файл: results\answers1.txt
```

Результаты экспертной оценки (1/2)

Все критерии в диапазоне [0, 5], где 0 - полное несоответствие критерию, а 5 - полное соответствие.

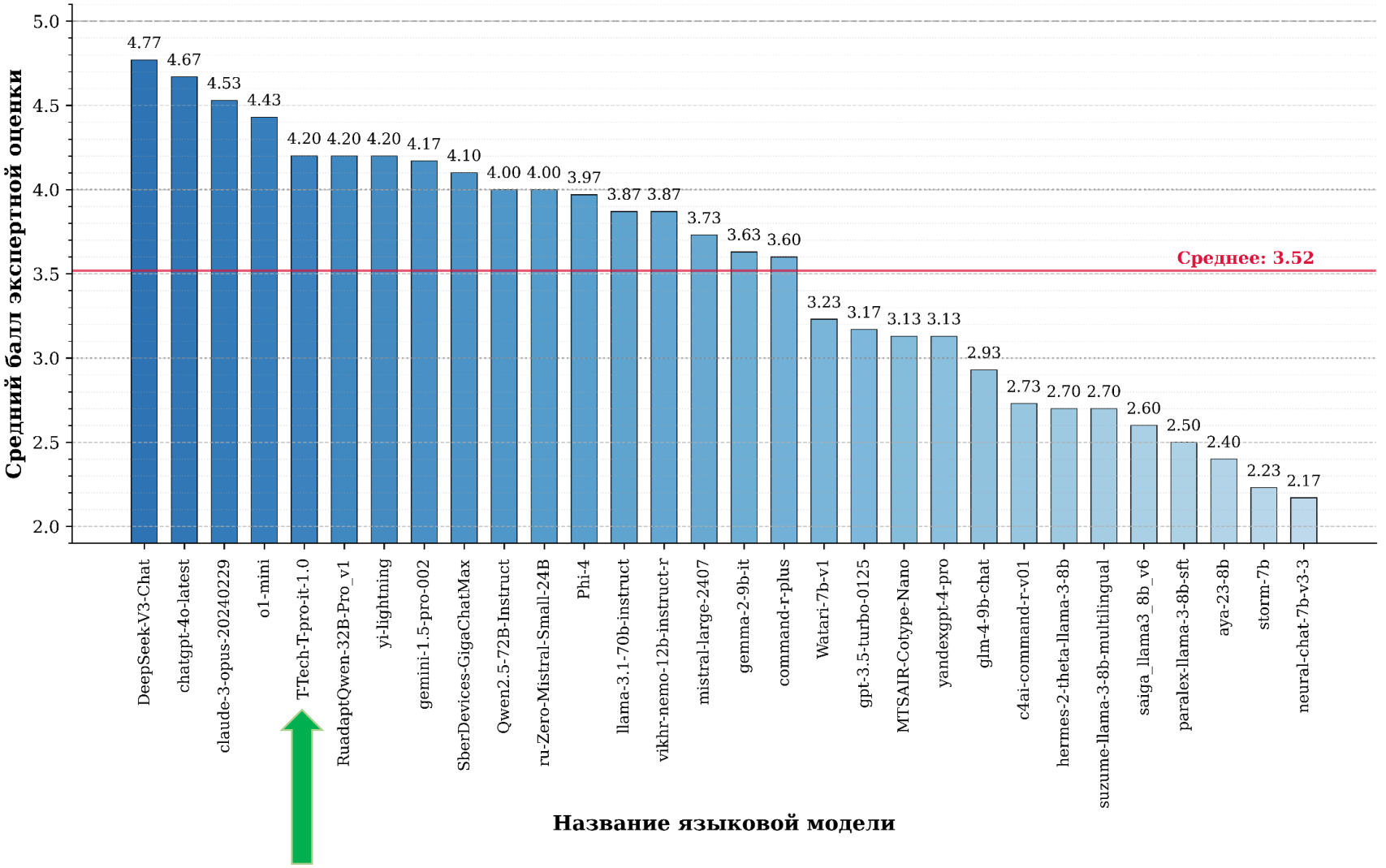
Полнота: 0 - отсутствие охвата аспектов исходного вопроса. 5 - набор подвопросов полностью покрывает информационную потребность исходного вопроса.
Атомарность: 0 - подвопросы содержат множество разнородных аспектов, требуя дальнейшей декомпозиции. 5 - каждый подвопрос идеально фокусируется на одном смысловом компоненте.
Корректность: 0 - грамматические ошибки, нелогичные формулировки, стилистическая несогласованность. 5 - безупречная грамматическая и логическая целостность подвопросов.

Проприетарные

Выбранная модель

Модель	Полнота	Атомарность	Корректность	Средний балл
DeepSeek-V3-Chat	4.8	4.6	4.9	4.77
chatgpt-4o-latest	4.7	4.5	4.8	4.67
claude-3-opus-20240229	4.4	4.5	4.7	4.53
o1-mini	4.5	4.2	4.6	4.43
T-Tech-T-pro-it-1.0	4.2	4.1	4.3	4.20
RuadaptQwen-32B-Pro_v1	4.3	4.1	4.2	4.20
yi-lightning	4.3	4.1	4.2	4.20
gemini-1.5-pro-002	4.3	4.0	4.2	4.17

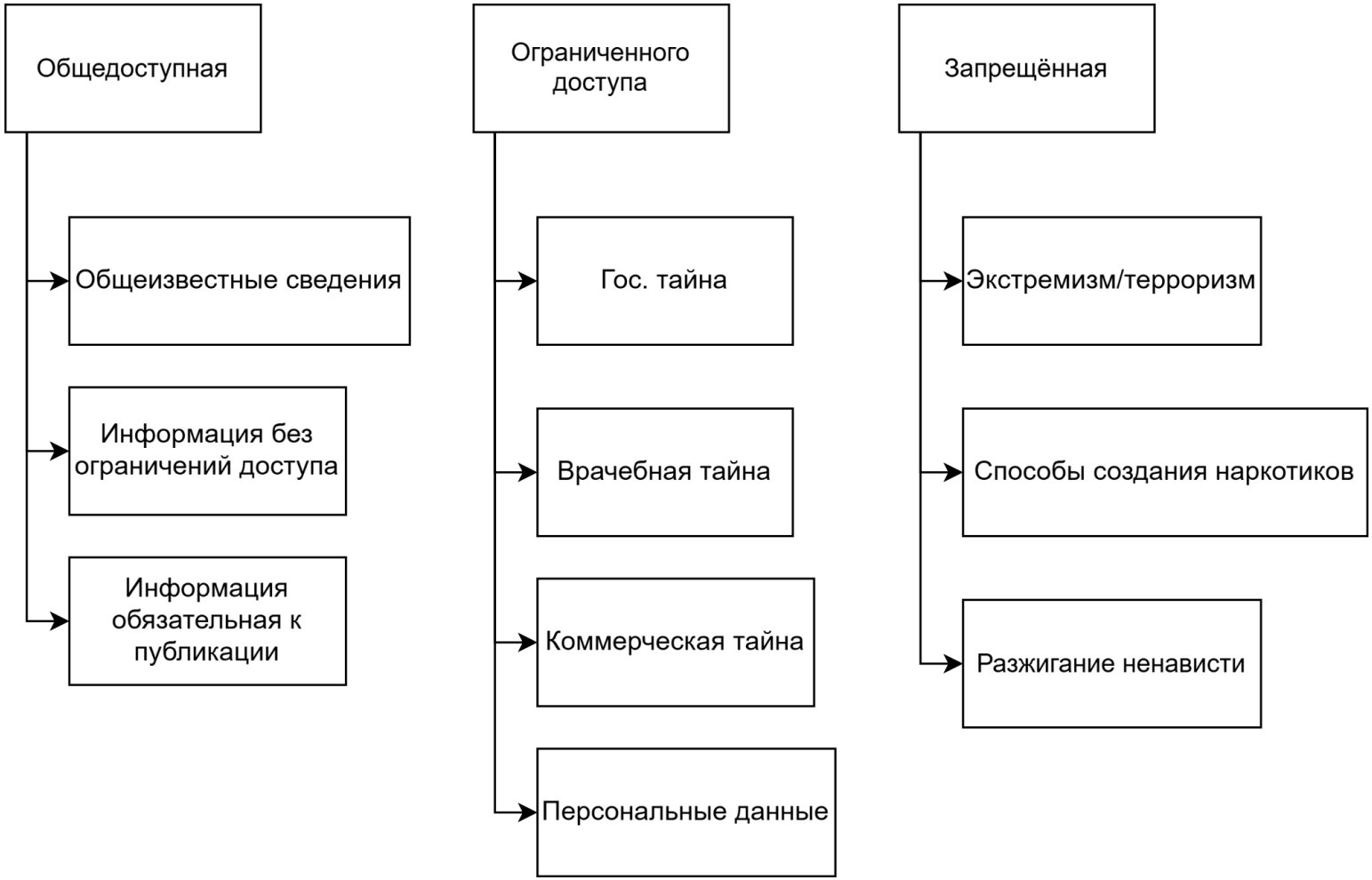
Результаты экспертной оценки (2/2)



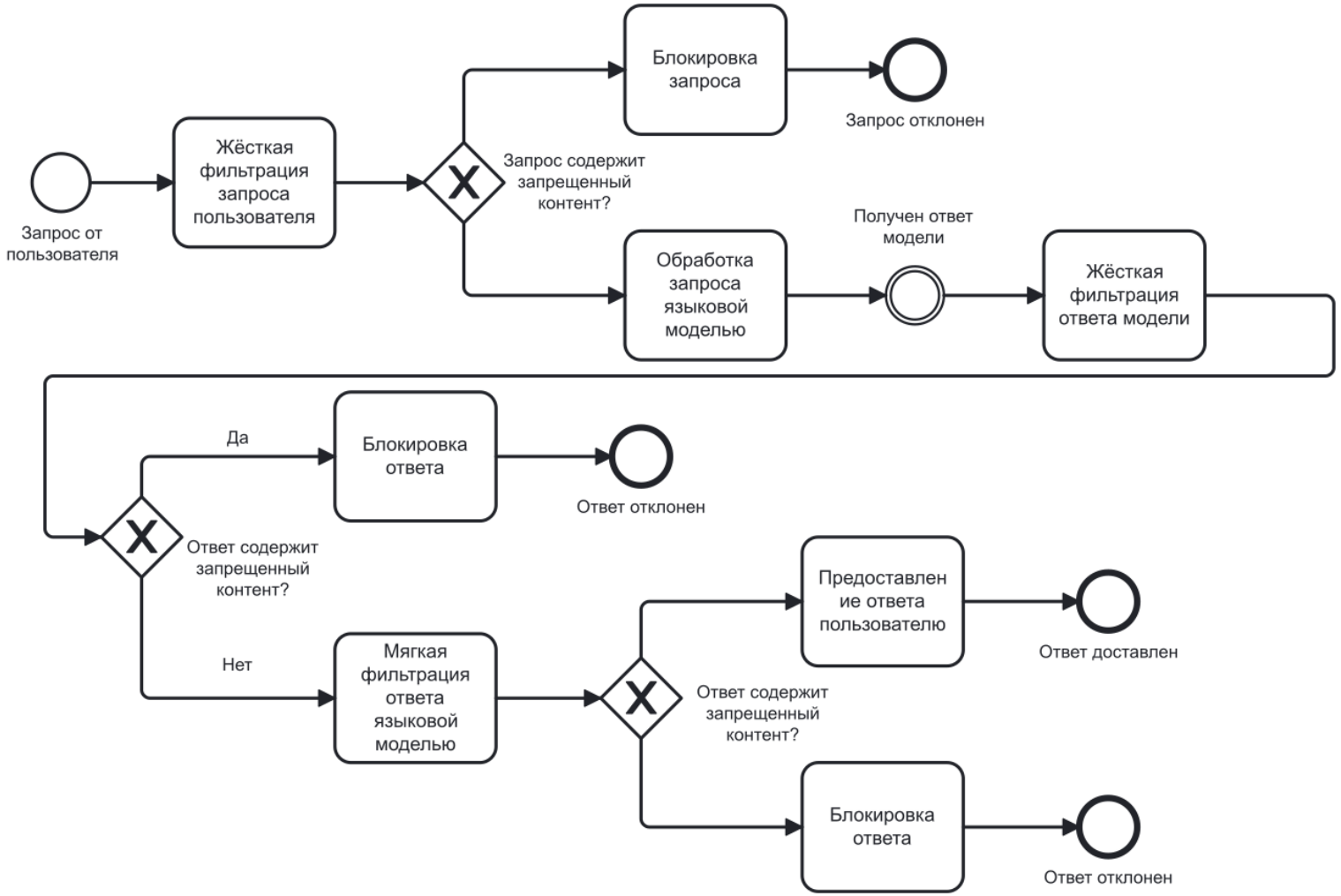
Лицензионная политика проекта

Критерий	Лицензия MIT	Лицензия Apache 2.0	Сравнение
Коммерческое использование	Разрешено	Разрешено	Одинаково
Модификация	Разрешено без ограничений	Разрешено с требованием указания изменений	Apache 2.0 строже
Распространение	Разрешено	Разрешено с дополнительными требованиями	Apache 2.0 строже
Частное использование	Разрешено	Разрешено	Одинаково
Указание изменений	Не требуется явно	Требуется явное указание	Apache 2.0 строже
Защита авторских прав	Требует включения уведомления	Требует сохранения всех уведомлений	Apache 2.0 строже
Патентные права	Нет явного предоставления	Явное предоставление патентных прав	Apache 2.0 строже
Ограничение торговых марок	Отсутствует	Прямо запрещает использование	Apache 2.0 строже
Распространение производных	Разрешено без особых условий	Разрешено с дополнительными условиями	Apache 2.0 строже

Ключевые виды информации по ФЗ-149



Многоуровневая система фильтрации запроса



Матрица распределения ответственности

Правовой аспект	Ответственность разработчика	Ответственность пользователя
Обновление баз фильтрации	Предоставляет технический механизм обновления	Обязан регулярно обновлять списки запрещенных слов согласно ст. 10.1 ФЗ-149
Обработка персональных данных	-	Несет полную ответственность по ФЗ-152 и ст. 13.11 КоАП РФ
Фильтрация запрещенного контента	Реализует базовые механизмы фильтрации, обновление баз фильтрациик	Отвечает за настройку и эффективность фильтров
Соблюдение лицензий	Обеспечивает правильное лицензирование (Apache 2.0)	Соблюдает условия лицензии при использовании
Верификация результатов	Не отвечает за проверку выдаваемой информации	Обязан верифицировать результаты на соответствие законодательству РФ
Учет отраслевых ограничений	Не отвечает за специфические отраслевые требования	Полностью отвечает за соблюдение отраслевых норм

Легенда:

Полная ответственность	Отсутствие ответственности

Заключение

В результате выполнения работы разработано программное решение для разбиения сложных вопросов на простые вопросы с использованием больших языковых моделей.

Выполнены следующие задачи:

1. Проанализированы существующие методы декомпозиции вопросов.
2. Разработан алгоритм декомпозиции на основе больших языковых моделей.
3. Реализовано программное решение.
4. Проведено исследование эффективности разработанного решения.
5. Проанализированы правовые аспекты программной реализации и выбрана лицензия.