В рамках проекта была разработана нейросетевая система, предназначенная для обработки текстов, генерации вопросов по их содержанию и создания заголовков для текстовых фрагментов. Основой системы стала платформа Ollama, которая позволила локально запускать модель. Для повышения эффективности был создан собственный LoRA-адаптер, обеспечивающий тонкую настройку базовой модели. Также был разработан контроллер, автоматизирующий процесс обработки текста и генерации вопросов с вариантами ответов. В этом отчете описывается, как создавалась система, как работает технология LoRA, какие данные использовались для обучения, как взаимодействуют компоненты для генерации заголовков и вопросов, а также какие ресурсы потребовались для реализации проекта.

### Анализ языковых моделей

Перед началом разработки системы проводился анализ различных языковых моделей, чтобы выбрать наиболее подходящую для задач обработки текста, генерации вопросов и создания заголовков. Среди рассмотренных кандидатов были Mistral, Gemini, LLaMA, Grok и другие популярные модели, каждая из которых обладает своими сильными и слабыми сторонами. Основными критериями выбора стали точность генерации, способность понимать контекст, а также требования к вычислительным ресурсам, поскольку проект предполагал локальное использование на ограниченном оборудовании.

### Оценка моделей

Модель Gemini, разработанная Google, известна своей универсальностью и высокой производительностью в задачах обработки естественного языка. Однако её интеграция в локальную среду оказалась сложной из-за зависимости от облачных сервисов, а требования к ресурсам для работы с большими версиями модели превышали доступные возможности оборудования. В предварительных тестах Gemini показала хорошие результаты в генерации текста, но её способность создавать точные вопросы по заданному контексту уступала другим моделям, особенно в условиях ограниченного объема обучающих данных.

LLaMA, созданная Meta AI, привлекла внимание благодаря своей эффективности и компактности. Эта модель оптимизирована для исследовательских задач и демонстрирует отличные результаты в задачах понимания текста. Однако в тестах на генерацию вопросов LLaMA часто выдавала слишком общие или недостаточно связанные с контекстом результаты. Её настройка также требовала больше времени и ресурсов для адаптации через LoRA по сравнению с другими кандидатами, что сделало её менее предпочтительной для быстрого старта проекта.

Grok, разработанный xAI, позиционируется как модель для генерации человекоподобных ответов с акцентом на диалоговые системы. Она показала неплохие результаты в создании связного текста, но её способность к генерации вопросов оказалась ограниченной из-за недостаточной ориентации на аналитические задачи. Кроме того, Grok требовал значительных вычислительных ресурсов для обработки длинных текстовых фрагментов, что противоречило цели минимизации затрат.

Среди других моделей рассматривались также варианты вроде Qwen от Alibaba, которая хорошо справляется с многоязычными задачами, но уступает в точности при работе с русскоязычными текстами, такими как SberQuAD. Более крупные модели, такие как GPT-3 или его аналоги, были исключены сразу из-за их высоких требований к памяти и невозможности локального запуска без значительных вложений в оборудование.

Mistral, разработанная французским стартапом Mistral AI, представляет собой модель, которая сочетает в себе высокую производительность и удобство интеграции в локальные среды. Mistral была специально оптимизирована для работы с ограниченными вычислительными ресурсами, что делает её идеальным выбором для проектов, где важна минимизация затрат на оборудование. В предварительных тестах Mistral показала отличные результаты как в генерации текста, так и в создании точных вопросов по заданному контексту. Её способность адаптироваться к различным задачам без необходимости значительных изменений в архитектуре или настройках делает её особенно привлекательной для быстрого старта проектов.

Одним из ключевых преимуществ Mistral является её гибкость и возможность настройки под конкретные нужды пользователя. В отличие от других моделей, Mistral не требует сложных процедур адаптации и может быть быстро интегрирована в существующие системы. Это позволяет значительно сократить время разработки и ускорить выход на рынок новых продуктов.

В целом, Mistral представляет собой мощный и универсальный инструмент, который сочетает в себе лучшие качества других моделей, таких как Gemini, LLaMA и Grok, при этом оставаясь доступной и удобной в использовании. Её высокая производительность, гибкость и возможность локального запуска делают её идеальным выбором для широкого спектра приложений, от образовательных платформ до корпоративных решений.

### Выбор модели Mistral-Nemo

Семейство моделей Mistral включает несколько вариантов, и перед окончательным выбором Mistral-Nemo были рассмотрены другие доступные версии, такие как Mistral-7B и Mixtral-8x7B, чтобы убедиться, что выбранная модель максимально соответствует задачам проекта. Каждая из них имеет свои особенности, которые влияют на производительность, требования к ресурсам и удобство использования в контексте генерации вопросов и заголовков.

Mistral-7B — это компактная модель с 7 миллиардами параметров, которая славится своей эффективностью и скоростью работы. Она требует меньше памяти (около 14 ГБ в 16-битном формате и 7-8 ГБ в 8-битном), что делает её привлекательной для локального запуска. В тестах на SberQuAD Mistral-7B показала хорошие результаты в генерации текста, но её способность к пониманию длинных контекстов оказалась ограниченной из-за меньшего размера окна контекста (4096 токенов). Это приводило к тому, что при обработке текстовых фрагментов по 1000 символов модель иногда теряла связь между частями текста, выдавая менее точные вопросы. Кроме того, её производительность в задачах, требующих глубокого анализа, уступала более крупным моделям.

Mixtral-8x7B, напротив, представляет собой смесь экспертов (MoE) с общей сложностью 56 миллиардов параметров, из которых активно используется около 14 миллиардов за раз. Эта модель демонстрировала выдающееся качество генерации и глубокое понимание контекста, что делало её потенциально идеальной для сложных аналитических задач. Однако её требования к памяти были значительно выше: даже в 8-битном формате она занимала около 20-22 ГБ, что оставляло мало места для дополнительных вычислений на моей видеокарте. Кроме того, интеграция Mixtral с LoRA и запуск через Ollama требовали более сложной настройки, а время обработки каждого запроса увеличивалось из-за архитектуры MoE, что замедляло работу системы.

Mistral-Nemo, с её 12 миллиардами параметров, заняла промежуточное положение. В 8-битном формате она требовала около 12 ГБ памяти, что оставляло достаточно пространства для LoRA-адаптеров и работы контроллера, укладываясь в общий лимит 18 ГБ. Её окно контекста, которое может доходить до 128000 токенов позволяло обрабатывать длинные текстовые фрагменты без потери связности, что было критически важно для генерации вопросов по текстам из SberQuAD. В тестах Mistral-Nemo показала высокую точность: например, для отрывка о Древнем Египте она стабильно генерировала вопросы вроде «Где впервые упоминается строение человеческого тела?» с правильными ответами, тогда как Mistral-7B иногда упрощала их до «Что изучали в Египте?», а Mixtral-8x7B, хоть и давала точные результаты, работала медленнее.

### Настройка уровня квантования

При настройке системы важным этапом стал выбор уровня квантования модели, который влияет на баланс между потреблением памяти и качеством генерируемых ответов. Квантование позволяет уменьшить размер модели, переводя её веса из полного формата с плавающей точкой (обычно 16 или 32 бита) в более компактные представления, такие как 8 бит или 4 бит. Это снижает требования к памяти и ускоряет вычисления, но может повлиять на точность работы модели. В моем случае рассматривались три варианта: полное представление (16 бит), 8-битное квантование и 4-битное квантование, каждый из которых тестировался на задаче генерации вопросов по SberQuAD.

Полное представление с 16-битной точностью обеспечивало максимальное качество генерации, так как сохраняло все детали весов модели. Однако это требовало в два раза больший объем памяти по сравнению с 8-битной версией. Такой подход был бы оправдан для серверных систем с большим объемом памяти, но для локального запуска оказался непрактичным.

4-битное квантование, напротив, значительно сокращало потребление памяти — до примерно 6-8 ГБ для модели Mistral-Nemo. Это делало её чрезвычайно компактной и подходящей для устройств с минимальными ресурсами. Однако тесты показали заметное снижение качества: вопросы становились менее точными, а в некоторых случаях модель теряла способность корректно связывать их с контекстом текста. Например, для отрывка о Древнем Египте 4-битная версия могла сгенерировать вопрос вроде «Что изучали в Египте?» вместо более конкретного «Где впервые упоминается строение человеческого тела?», что снижало полезность системы для образовательных целей, также эта модель могла сбиваться в формате вывода, что мешало последующей обработке запроса.

8-битное квантование оказалось золотой серединой. Оно уменьшало размер модели до примерно 12 ГБ. При этом качество генерации оставалось близким к 16-битной версии: вопросы сохраняли точность и контекстную связь, а потери в производительности были минимальными. В тестах на SberQuAD 8-битная версия Mistral-Nemo стабильно выдавала релевантные вопросы с правильными вариантами ответов, что соответствовало целям проекта. Таким образом, было выбрано 8-битное квантование как оптимальное решение, обеспечивающее лучшее соотношение между потреблением памяти и качеством даваемых ответов, что позволило эффективно использовать доступное оборудование без значительных компромиссов в функциональности.

### Технология LoRA

LoRA (Low-Rank Adaptation) — это метод, который позволяет эффективно адаптировать большие языковые модели к новым задачам, не переобучая их полностью. Вместо изменения всех весов модели, которые могут занимать гигабайты памяти, LoRA добавляет небольшие матрицы низкого ранга к существующим слоям. Оригинальные веса остаются неизменными, а корректировки вносятся через произведение двух матриц, что значительно снижает объем вычислений. Такой подход экономит время и память: если полное обучение модели может занять недели и требует мощных серверов, то настройка LoRA-адаптера занимает лишь часы и может быть выполнена на одной видеокарте.

Преимущество LoRA в том, что адаптеры можно легко заменять или комбинировать, адаптируя систему под разные сценарии использования без значительных затрат ресурсов.

При разработке системы одним из ключевых решений было определение метода адаптации модели к задачам проекта. Среди доступных технологий рассматривались два основных типа адаптеров: LoRA (Low-Rank Adaptation) и QLoRA (Quantized Low-Rank Adaptation). Оба подхода предназначены для эффективной тонкой настройки больших языковых моделей, но они различаются по принципам работы и применимости в зависимости от условий задачи.

LoRA работает за счет уменьшения объема памяти, необходимого для хранения изменений весов модели. Вместо полного обновления весовой матрицы (ΔW) LoRA использует низкоранговое приближение, представляя ΔW как произведение двух небольших матриц. Это значительно сокращает количество параметров, которые нужно обучать, что делает процесс быстрым и менее требовательным к ресурсам. В результате LoRA обеспечивает высокую скорость тонкой настройки, сохраняя при этом производительность, близкую к традиционным методам полного переобучения. В моем проекте это означало, что Mistral-Nemo мог быть быстро адаптирован к генерации вопросов и заголовков, используя синтетические данные и SberQuAD, без необходимости длительного обучения на мощных серверах.

QLoRA, в свою очередь, развивает идеи LoRA, добавляя квантование весов адаптеров. Это означает, что матрицы, используемые в LoRA, дополнительно сжимаются до более низкой точности (например, с 8 бит до 4 бит), что еще больше снижает объем памяти и требования к хранению. QLoRA сохраняет эффективность LoRA в плане производительности, но делает её ещё более экономичной с точки зрения памяти, что особенно полезно в условиях крайне ограниченных ресурсов. Однако этот дополнительный шаг квантования может слегка замедлить процесс обучения из-за необходимости обработки сжатых данных, хотя разница в скорости обычно невелика.

Выбор между LoRA и QLoRA зависел от приоритетов проекта. Если бы основной задачей была максимальная экономия памяти, QLoRA могла бы стать предпочтительным вариантом, так как она позволяет работать с ещё меньшим объемом ресурсов, что идеально для устройств с ограниченной вычислительной мощностью. Однако в моем случае скорость тонкой настройки была важнее, чем экстремальная минимизация памяти.

Кроме того, LoRA предлагала более простую интеграцию с платформой Ollama и быстрее адаптировалась к задаче генерации вопросов и заголовков. В тестах на SberQuAD LoRA показала стабильные результаты, выдавая точные и контекстно связанные вопросы, что было критически важно для проекта. QLoRA, хотя и могла бы сэкономить ещё больше памяти, не давала значительного прироста в качестве генерации, а её настройка требовала дополнительных усилий для управления квантованием, что могло усложнить процесс разработки. Таким образом, был выбран LoRA, так как он обеспечивал оптимальный баланс между скоростью обучения, простотой реализации и достаточной эффективностью в использовании ресурсов, что идеально соответствовало условиям проекта и доступному оборудованию.

### Обучение LoRA-адаптера

Для обучения LoRA-адаптера использовался датасет SberQuAD, доступный на Hugging Face. Это крупный набор данных на русском языке, созданный для задач построения вопросно-ответных систем на основе текстов из русской Википедии. Датасет содержит около 50 000 вопросов, каждый из которых связан с определенным текстовым отрывком и сопровождается ответами, указывающими конкретные фрагменты текста. Он разделен на обучающую, валидационную и тестовую выборки, что позволило эффективно использовать его для тренировки и проверки качества модели. Например, в датасете можно найти текстовый контекст вроде «Первые упоминания о строении человеческого тела встречаются в Древнем Египте» и связанный с ним вопрос «Где встречаются первые упоминания о строении человеческого тела?» с правильным ответом «в Древнем Египте».

В проекте SberQuAD был адаптирован для задачи генерации вопросов: текстовые отрывки использовались как входные данные, а вопросы служили целевыми выходами. Обучающая выборка из 45 000 примеров позволила LoRA-адаптеру научиться создавать релевантные и контекстно связанные вопросы, а валидационная выборка из 5 000 примеров использовалась для оценки качества. Кроме того, были добавлены синтетические данные с вариантами ответов — правильными и неправильными, — чтобы модель могла генерировать вопросы с выбором ответа, что сделало систему более интерактивной и полезной для тестирования знаний.

### Взаимодействие компонентов системы

Система объединяет несколько нейросетевых компонентов, которые работают вместе для обработки текста, создания заголовков и генерации вопросов. Это позволяет не только проверять понимание текста через вопросы, но и улучшать его восприятие за счет кратких и точных заголовков. Для генерации заголовков использовалась отдельная нейросеть, также основанная на архитектуре, схожей с Mistral-Nemo, с собственным LoRA-адаптером, который был обучен на парах «текст — заголовок». Процесс выглядит следующим образом: текст разбивается на фрагменты, каждый фрагмент отправляется в модель, которая выделяет ключевые идеи и преобразует их в заголовок, например, «История изучения человеческого тела» для текста об анатомии. Полученные заголовки сохраняются в базе данных вместе с соответствующими фрагментами.

Генерация вопросов осуществляется основной моделью Mistral-Nemo с LoRA-адаптером, обученным на SberQuAD. Контроллер отправляет текстовый фрагмент в модель с запросом создать вопросы с вариантами ответов, где один из них правильный. Модель возвращает структурированный ответ, который затем сохраняется в базу данных. Например, для текста о Древнем Египте может быть сгенерирован вопрос «Где встречаются первые упоминания о строении человеческого тела?» с вариантами «в Древнем Египте» (правильно) и «в Древней Греции» (неправильно). Оба процесса — генерация заголовков и вопросов — интегрированы через контроллер, который сначала получает заголовок для фрагмента, а затем передает тот же фрагмент для создания вопросов. Это обеспечивает связность данных: каждый текстовый отрывок в базе сопровождается заголовком и списком вопросов с ответами.

Взаимодействие компонентов построено так, чтобы минимизировать нагрузку на вычислительные ресурсы. Обе модели используют одну базовую архитектуру Mistral-Nemo, но с разными LoRA-адаптерами, что делает систему модульной и позволяет легко адаптировать ее к новым задачам. Контроллер управляет очередностью обработки, обеспечивая плавную работу даже при больших объемах текста.

### Ресурсы и эффективность

Для реализации проекта потребовалось значительно меньше ресурсов, чем для тренировки полноценной языковой модели с нуля. Использование LoRA позволило обойтись без многодневного обучения на кластерах GPU, что сделало процесс быстрым и доступным. Само обучение LoRA-адаптеров длилось значительно меньшее количество времени, по сравнению с тем, сколько его потребовалось бы для тренировки полноценной языковой модели, которое могло бы занять недели. Общий объем памяти для работы системы, включая базовую модель и адаптеры, составил около 18 ГБ, что делает ее пригодной для локального использования на мощных персональных компьютерах.

### Автоматизация процесса обработки текста

Для автоматизации работы с текстом был создан контроллер. Он разбивает исходный текст на фрагменты по 1000 символов, сохраняет их в базе данных и отправляет в нейросеть через API платформы Ollama. Модель обрабатывает каждый фрагмент и возвращает список вопросов с вариантами ответов, один из которых правильный. Такой подход позволяет систематизировать процесс и делает систему удобной для использования в образовательных или аналитических целях. Запуск модели осуществлялся локально с использованием Ollama, что обеспечило гибкость и независимость от внешних серверов.

Для удобства пользователей был реализован веб-интерфейс, который позволяет взаимодействовать с существующими текстами, вопросами и добавлять новые. Интерфейс предоставляет возможность загружать текстовые фрагменты, просматривать сгенерированные вопросы и заголовки, а также редактировать и добавлять новые данные. Это значительно упрощает процесс работы с системой и делает её доступной для широкого круга пользователей, включая образовательные учреждения и аналитические компании.

### Заключение

Разработанная нейросетевая система представляет собой продуманное и эффективное решение для обработки текстов, генерации вопросов, что делает её полезным инструментом для образовательных и аналитических целей. Основой успеха проекта стал тщательный выбор модели Mistral-Nemo, который обеспечил оптимальный баланс между производительностью и требованиями к ресурсам. Её преимущества — большое окно контекста, точность генерации и совместимость с локальной платформой Ollama — позволили создать систему, способную работать на ограниченном оборудовании без ущерба качеству. Сравнительный анализ альтернативных моделей, таких как Gemini, LLaMA или Mixtral, показал, что ни одна из них не могла предложить столь же удачного сочетания характеристик для поставленных задач.

Применение технологии LoRA стало ключевым фактором, обеспечившим экономичность и гибкость системы. Этот метод позволил адаптировать мощную базовую модель к специфическим задачам проекта, минимизируя вычислительные затраты и сохраняя высокое качество результатов. Выбор 8-битного квантования дополнительно оптимизировал использование ресурсов, доказав, что даже при сжатии модели можно достичь точности, сравнимой с полным представлением. Интеграция с датасетом SberQuAD и синтетическими данными обеспечила системе способность генерировать релевантные и контекстно связанные вопросы, что особенно ценно для русскоязычных приложений.

Автоматизация через контроллер сделала процесс обработки текста систематичным и удобным, а модульная архитектура системы открывает возможности для дальнейшего развития. Например, добавление поддержки других языков, улучшение качества заголовков или расширение функционала для анализа тональности текста — всё это достижимо без кардинальных изменений в структуре. Низкие требования к ресурсам (18 ГБ памяти) и локальный запуск через Ollama делают систему доступной для использования на персональных компьютерах, что расширяет её потенциальную аудиторию.