Большие языковые модели (LLM) и их применение в анализе нормативных документов

КОРОТЧЕНКО В.В., к.т.н., МЕЗЕНЦЕВ Я.С.

Аннотация. В статье рассматриваются принципы работы больших языковых моделей и их применение при анализе нормативных документов. Описаны архитектурные основы LLM, особенности reasoning-моделей, понятие токенов и ограничения контекстного окна. Также обсуждаются подходы к работе с большими документами и практические аспекты использования LLM через чат-интерфейсы и API.

Ключевые слова: большие языковые модели, LLM, искусственный интеллект, анализ документов.

ЧТО ТАКОЕ LLM И ЗАЧЕМ ОНИ НУЖНЫ

Большие языковые модели (large language model, LLM) — это системы ИИ, обученные на гигантских объёмах текстов и способные понимать и генерировать текст на естественном языке. По сути, они имитируют человеческую работу с языком, опираясь на знания из миллионов страниц информации. Современные модели хорошо держат контекст диалога, отвечают связно и по существу. Благодаря этому LLM решают широкий спектр текстовых задач: пишут письма, обзоры и отчёты, переводят, суммируют длинные документы, отвечают на вопросы по тексту и выполняют творческие задания [1].

Зачем они нужны? LLM сильно упрощают работу с большими объёмами информации. В бизнесе и управлении они автоматизируют рутину: ускоряют подготовку отчётов и аналитики, берут на себя черновую работу с документами. Модель может за секунды прочитать объёмный приказ или положение, выдать краткое резюме с ключевыми требованиями, ответить на конкретный вопрос по документу или сгенерировать черновик по заданным условиям. Таким образом, LLM превращаются в интеллектуальных ассистентов, которые быстро генерируют тексты, анализируют документы и извлекают нужные сведения [2]. Особенно полезны они при работе с нормативными актами. При этом LLM — это инструмент, а не волшебная палочка. Модель ускоряет обработку текстов и подсказывает, где именно в документе содержится нужная информация, но решения и ответственность всегда остаются за человеком.

КАК УСТРОЕНЫ ЯЗЫКОВЫЕ МОДЕЛИ

В основе работы LLM лежит прогнозирование следующего фрагмента текста по контексту. Модель обучена предсказывать следующее слово (или его часть) исключительно на основе всех предыдущих слов [2]. Она не знает правил языка в традиционном смысле и не «читает мысли», работает только со статистическими шаблонами, выявленными из гигантского массива текстов во время обучения.

Что значит «обучение на больших данных»? Это как если бы модель прочитала почти весь доступный цифровой архив: книги, статьи, законы, сайты. В процессе она самостоятельно выявила закономерности: какие слова чаще стоят рядом, как строятся предложения, какие факты обычно упоминаются вместе. Ей не задавали жёсткие правила — ей показали миллиарды примеров, и она научилась, что за словом X с наибольшей вероятностью следует слово Y [1, 2]. Например, увидев «договаривающиеся стороны обязаны…», модель почти наверняка продолжит про исполнение обязательств или условия — просто потому, что тысячи раз встречала похожие формулировки в нормативных актах. Поэтому LLM генерирует текст не по логическим правилам, а по вероятности: выбирает то продолжение, которое чаще всего встречалось в похожем контексте. Отсюда и точность в типовых ситуациях (она видела тысячи аналогичных документов), и возможные ошибки в нетипичных: тогда модель просто «угадывает» по ближайшим аналогам. Важно: LLM не понимает текст по-человечески, у неё нет собственного смысла или мнения. Но благодаря миллиардам параметров она убедительно имитирует понимание: удерживает длинный контекст, связно продолжает мысль, перескажет документ, ответит на вопросы и выделит факты — всё это потому, что в обучающих данных были миллионы подобных примеров, и она усвоила шаблоны их решения [1]. Практический вывод: модель блестяща в стандартных задачах —

продолжить текст, написать в привычном стиле, сделать резюме. Но в совершенно новых ситуациях или при необходимости строгой логики она может выдать лишь наиболее вероятный, а не обязательно правильный ответ [3].

Что такое *reasoning*-модели и зачем они нужны

В развитии ИИ появилось направление reasoning-LLM — модели с усиленной способностью к логическому, поэтапному мышлению. В отличие от обычных LLM, которые сразу выдают наиболее вероятный ответ, reasoning-модели умеют «думать» шаг за шагом, как человек. Это достигается специальным обучением: модель поощряли разбивать задачу на части, делать промежуточные выводы и только потом давать финальный ответ [3]. В итоге она работает почти как эксперт: понимает условие → декомпозирует задачу → последовательно решает подзадачи → проверяет себя → выдаёт ответ. Во многих моделях эти рассуждения можно явно увидеть через специальный режим или промпт.

Ключевое отличие — умение выстраивать явные логические цепочки. Обычная LLM обладает огромными знаниями и генерирует связный текст, но при сложной логике часто ошибается, пропуская скрытые условия. Reasoning-модели специально обучены тратить больше «умственных усилий» на трудные вопросы. Например, OpenAI в серии o1 сделала акцент на том, чтобы модель «думала подольше», пробовала разные пути и исправляла ошибки до ответа — почти как человек [3].

Для работы с нормативными и техническими документами это особенно ценно. Сложные тексты полны условий и взаимосвязей вида «если А, то В, иначе С». Обычная модель может пропустить важное условие, а reasoning-модель склонна пройтись по каждому пункту. Например, при сверке отчёта с приказом она последовательно проверит выполнение каждого требования вместо общей оценки «вроде бы нормально». Такие модели лучше выявляют несоответствия, проверяют сложные условия и разбирают причинно-следственные связи.

Почему не использовать их повсеместно? Они заметно медленнее и дороже, поскольку тратят много шагов на размышления. Для простых задач (короткое резюме, переформулировка) обычной модели достаточно — она быстрее и дешевле. Но когда нужно глубоко разобрать хитросплетения нормативных актов и минимизировать риск упустить нюанс на странице 57, reasoning-LLM оправдывает затраты. Даже с ними окончательную проверку всё равно делает человек, но вероятность грубых пропусков существенно ниже. В приведенной таблице показан пример работы модели на условной задаче.

ЧТО ТАКОЕ ТОКЕНЫ — И ПОЧЕМУ ЭТО ВАЖНО

Работая с LLM, вы сразу столкнётесь с понятием «токен» — это минимальная единица текста, на которую модель разбивает всё, что получает. Токен может быть целым коротким словом («дом»), частью длинного слова («закон» + «одатель» + «ство» в слове «законодательство») или даже знаком препинания. В среднем один токен — это 3–4 символа русского текста и около 4-х символов английского [4].

Русский текст обычно даёт больше токенов, чем английский той же длины, потому что длинные слова чаще дробятся. Числа разбиваются блоками: «2025» → «20» + «25», а не одним токеном. Изображения сначала делятся на патчи (визуальные токены), а надписи на них дополнительно проходят через OCR и превращаются в обычные текстовые токены.

Зачем это знать? Потому что у каждой модели есть жёсткий лимит «контекстного окна» — максимальное количество токенов, которое она может обработать за один раз. Например: GPT-3.5 (2022 год) — около 4 000 токенов, многие современные модели — 32–128 тысяч, модели с наибольшим контекстным окном — до 1 миллиона токенов.

Таблица

Шаги работы	Действие в примере
Запрос пользователя	*«Проверь, соответствует ли проект отчёта приказу № 1»*
Декомпозиция задачи	Выделить требования приказа: А (расчёт потерь), Б (резервирование источников), В (прогноз нагрузки)
Пошаговые рассуждения	Сравнить каждый пункт приказа с отчётом: А → найден в разделе 3; Б → отсутствует; В → найден в приложении 2
Промежуточные выводы и проверка	Зафиксировать результат по каждому требованию: А — выполнено, Б — не выполнено, В — выполнено
Финальный ответ	*Итог: «Отчёт в целом соответствует приказу № 1, но отсутствует раздел о резервных источниках»*

Одна страница А4 (1500–2000 символов) на русском — это примерно 800–1000 токенов. Значит, 100 страниц отчёта ≈ 80–100 тысяч токенов. Даже топовые модели не всегда вмещают такой объём целиком. Отсюда практические выводы:

Большие документы приходится резать на части и обрабатывать поочерёдно. Зная примерное соотношение «1 страница ≈ 900–1000 токенов», вы сразу можете прикинуть, сколько запросов понадобится и уместится ли нужный фрагмент в одно обращение к модели.

Токен — это та «оперативная память» модели: сколько токенов поместилось, столько она и «видит» одновременно. Всё остальное нужно подавать частями.

ЧТО ТАКОЕ API И ЗАЧЕМ ОН НУЖЕН

При работе с LLM есть два главных способа взаимодействия: обычный чат-интерфейс (ChatGPT, Grok, Claude и т.п.) и API.

API (Application Programming Interface) — это программный интерфейс, который позволяет вашей программе напрямую обращаться к языковой модели. Проще говоря, вместо того чтобы вручную копировать-вставить текст в окошко чата, вы пишете код (или скрипт), который сам отправляет запросы модели и получает ответы в удобном для дальнейшей обработки виде [5].

Когда использовать:

- обычный чат — для разовых вопросов, небольших текстов, когда не хочется писать код. Просто, быстро, не требует программирования. Но неудобно при повторяющихся операциях и невозможно встроить в свои процессы;
- API — когда нужно автоматизировать и масштабировать. Например, вы пишете программу, которая берёт папку с сотней PDF документов, по очереди отправляет каждый модели через API и автоматически сохраняет полученные сводки или результаты проверки на соответствие нормативам. Всё работает без вашего участия после запуска.

Именно через API LLM перестаёт быть просто «умным чатом» и превращается в настоящий производственный инструмент: его можно встроить в корпоративный портал, документооборот, чат-бот на сайте или любую внутреннюю систему. Пользователь увидит красивый интерфейс, а «под капотом» будет вызов API модели.

Вывод прост: если вы разово спрашиваете что-то у модели, то достаточно веб-чата. Если же предстоит регулярно обрабатывать большие объёмы документов или встраивать ИИ в рабочие процессы — без API не обойтись. В этом случае понадобится либо разработчик, либо базовые навыки написания скриптов, зато результат — полностью автоматизированный анализ десятков и сотен документов по единой логике.

КАКИЕ БЫВАЮТ LLM-МОДЕЛИ

Моделей LLM сейчас много, и они различаются по нескольким ключевым признакам.

1. По разработчику: зарубежные (OpenAI GPT-5.1, Google Gemini, Grock xAI и др.) и российские/локальные (GigaChat от Сбера, дообученные версии открытых моделей).
2. По способу развёртывания: облачные (работают на серверах вендора) и локальные (устанавливаются на ваших мощностях).

Облачные модели:

- Плюсы: начать можно сразу, не нужна собственная IT-инфраструктура, высокое качество, быстрый отклик.
- Минусы: платно за токены или подписка, данные уходят на чужие серверы, при большом объёме данных становится дорого, некоторые сервисы недоступны из России.

Локальные модели:

- Плюсы: полная конфиденциальность (ни один документ не покидает вашу IT-инфраструктуру), после развёртывания бесплатное использование, нет зависимости от интернета и иностранных ресурсов.

ChatGPT	Gemini	DeepSeek	GigaChat	YandexGPT
OpenAI, США	Google DeepMind, США/Великобритания	DeepSeek, Китай	СБЕР, Россия	Яндекс, Россия

- Минусы: нужна мощная IT-инфраструктура (серверы, видеокарты), сложнее настройка и поддержка, доступные для локального запуска модели обычно меньше по объёму данных (7–70 млрд параметров против сотен миллиардов у топовых облачных), поэтому часто уступают в «уме» и точности.

Что важно учитывать при выборе:

- Стоимость: при разовых задачах дешевле облако, при сотнях запросов в день часто выгоднее своя модель.
- Размер контекстного окна: от 2–4k токенов у простых моделей до 1m у самых продвинутых. От этого зависит, сколько страниц можно проанализировать за один запрос.
- Качество русского языка: зарубежные топовые модели уже работают с русским очень хорошо, но русскоязычные или дообученные локальные модели иногда точнее понимают специфику нормативных правовых актов и юридических формулировок.
- Приватность: если документы содержат коммерческую тайну, персональные данные или государственную тайну — выбор почти всегда в пользу локального решения.

Вывод для практики: если критична максимальная точность и данные можно отправлять вовне — выбор за лучшими облачными моделями. Если важна конфиденциальность и/или объёмы большие и регулярные — разумнее развернуть локальную или отечественную облачную модель в закрытом контуре. В большинстве случаев оптимальный путь — протестировать 2–3 варианта на своих реальных документах и выбрать тот, который даёт лучшее соотношение точности, скорости и стоимости именно для ваших задач.

МОЖЕТ ЛИ LLM АНАЛИЗИРОВАТЬ БОЛЬШИЕ ДОКУМЕНТЫ (500–1000 СТР. С ТАБЛИЦАМИ И ИЗОБРАЖЕНИЯМИ)

Это очень актуальный вопрос: обычные нормативные документы, отчёты, схемы могут иметь сотни страниц, таблицы, графики. **Способна ли модель справиться с таким объёмом?** Короткий ответ: *да, но с определённой стратегией и ограничениями*.

Прежде всего, LLM работает главным образом с текстом. Всё, что является текстовым содержимым документа, модель может воспринять. Если в PDF есть разделы, параграфы, списки — при конвертации в текст (обычно перед использованием LLM PDF преобразуют в текстовый формат) модель прочитает и проанализирует их. **Таблицы** также можно представить в текстовом виде (например, как CSV или просто как строки, где значения разделены пробелами). Модель в принципе может понять таблицу, особенно если её попросить (например: «Перед тобой таблица, вот её колонки и строки…»). Однако есть нюанс: модель **не гарантирует идеального сохранения структуры**. Она читает таблицу как набор строк, и может перепутать столбцы, если её явно не направить. Тем не менее, **извлечь информацию из таблицы модель может** — например, найти в таблице нужное значение, сравнить показатели. В режиме анализа нормативных документов LLM сможет, скажем, выделить из таблицы перечень показателей или отметить, какие строки соответствуют таким-то критериям (при корректной формулировке запроса).

Объём 500–1000 страниц явно превышает контекстные возможности любой модели на сегодня. Значит, модель **не сможет взять и залпом «проглотить» тысячу страниц**. Реалистичный подход — **разделить документ на части** и обрабатывать поэтапно. Например, документ на 1000 страниц можно разбить по главам или по 50 страниц, и анализировать постепенно, задавая модели серию вопросов: сначала по первой части, потом по второй и т.д., а затем агрегировать результаты. Это, конечно, усложняет процесс: фактически вы строите над моделью некий сценарий. Однако другого пути нет, поскольку ограничение на токены непреодолимо напрямую.

Какие типы информации может извлекать модель из большого документа? Практически любые, которые вы сумеете сформулировать в запросе. Например:

— **Извлечение ключевых положений.** Модель может прочитать раздел и перечислить основные тезисы (идеально для подготовки краткого резюме большой главы).

— **Поиск упоминаний.** Можно спросить: «Какие меры поддержки упоминаются в документе и в каком контексте?» — модель найдёт и перечислит.

— **Свод требований.** Для нормативного акта — «Перечисли все требования, которые документ предъявляет к проектам теплоснабжения». Модель пробежится по тексту и вытащит пункты с «должен», «обязан», «необходимо».

— **Структурирование.** Модель может разделить текст на логические блоки, если попросить: «Определи основные разделы и их краткое содержание». Это полезно, чтобы получить «карту» документа.

— **Сравнение двух документов.** Это сложнее, но возможно: нужно поочерёдно показать модели фрагменты первого и второго документа и задать вопрос о взаимосоответствии. Например, дать перечень требований из документа А, затем дать текст документа В (или тоже список его положений) — и попросить сопоставить. Модель при правильной организации ответа укажет, какие требования А нашли отражение в В, а какие нет.

Может ли LLM сравнить документы, например, отчёт и требования приказа? Да, и это один из самых востребованных сценариев — **проверка соответствия**. Однако, как отмечалось, оба документа целиком модель разом не удержит в памяти, значит нужно стратегически подойти: либо заранее извлечь требования приказа и разделы отчёта (например, отдельными списками), либо сравнивать раздел за разделом. Вручную человек бы тоже не читал два талмуда параллельно — он бы брал требование и искал его отражение. Точно так же можно организовать и с моделью. Если всё сделать правильно, модель способна пройтись по списку требований и для каждого указать, где в отчёте оно выполнено или что не учтено.

Как загружаются документы в модель? Без технических деталей опишем логику: PDF сначала конвертируется в текст (это подготовительный шаг, обычно вне самой модели, с помощью утилит OCR если PDF сканированный, либо просто вычленения текста). Затем этот текст **делится на куски** разумного размера (например, по одному параграфу, или ~800 токенов, чтобы не превысить контекст). Дальше есть варианты: либо **непосредственно скормить кусок модели и попросить рассказать что-то про него**, либо **использовать более сложный подход — поиск по документу**. Последний работает так: сначала загружаем в специальную систему все куски (индексируем), а потом модель сама выбирает релевантные при ответе на вопрос (это называется Retrieval-Augmented Generation). Главное — понимание, что **большой документ целиком модель не «понимает» мгновенно, она работает с фрагментами**. Поэтому при анализе, например, 1000-страничного отчёта о теплоснабжении может быть организован цикл: раздел 1 → вопрос к модели → ответ, раздел 2 → вопрос → ответ, и так далее, с последующей компоновкой итогов.

Как правильно формулируется задача для модели? Чтобы модель поняла, что от неё хотят, запрос (промпт) должен быть **предельно чётким и конкретным**. Например, недостаточно сказать: «Проанализируй документ и дай выводы». Такой расплывчатый запрос приведёт к неопределённому ответу — модель сама будет гадать, что считать «выводами». Лучше явно указать, *что именно* нужно: *«Прочитай текст раздела 5 и кратко перечисли цели и основные мероприятия, указанные в этом разделе»* — понятная и конкретная задача. Или: *«Сравни требования приказа № XX (список требований приведён) с содержанием отчёта (далее приведены тезисы отчёта) и перечисли, какие требования приказа не отражены в отчёте»*. В таком запросе модель видит явную цель (найти несоответствия) и необходимые данные (требования и тезисы отчёта). Важно также **ограничить область внимания** модели, если документ очень большой. Например: «Ниже текст приложения 3, проанализируй только его».

Простыми словами, при постановке задачи модели на анализ документов нужно:

— Чётко указать, *что ищем* или *что нужно сделать* (найти, сравнить, перечислить, суммировать…).

— Указать контекст: либо сам текст включить, либо сказать, что он «далее по тексту». Модель ведь не имеет «доступа» к вашим файлам напрямую — она оперирует только тем, что получит в сообщении.

— По возможности, разбить сложную задачу на несколько простых. Вместо одного вопроса: «Есть ли в схеме X всё, что требуется приказом Y?» лучше задать несколько: «Какие ключевые требования приказа Y?» → (получить список) → «Для каждого из этих требований укажи, упоминается ли оно в схеме X и в каком разделе».

Такой пошаговый подход даст более надёжный результат.

Резюмируя: LLM может анализировать даже очень большие документы, просто это делается итеративно и с тщательной постановкой вопросов. Модель извлекает текстовую информацию — как явную (факты, цифры в тексте), так и структурную (заголовки, связи между разделами) — и помогает ответить на практические вопросы по документу. Но от пользователя (или разработчика) требуется организовать процесс: загрузить текст частями, правильно *спросить* и потом собрать мозаику ответов.

Современные модели способны **принимать на вход значительно бóльшие объёмы текста и даже изображения**. Это меняет подход к работе с большими документами (500–1000 страниц) и упрощает многие операции.

Большее «контекстное окно». В отличие от ранних версий, у современных моделей контекстное окно в сотни тысяч токенов (что эквивалентно десяткам или даже сотням страниц текста). Это значит, что **можно сразу загрузить в модель крупный раздел или несколько приложений целиком**, не разбивая документ на слишком мелкие фрагменты.

Поддержка таблиц и изображений. Современные модели умеют обрабатывать не только текст, но и **распознавать содержимое изображений и таблиц**, если они представлены в виде встроенных в сообщение файлов. Это позволяет:

— извлекать данные из таблиц без предварительного преобразования в текст;

— понимать схематические иллюстрации (например, упрощённые чертежи теплоснабжения);

— давать комментарии к графикам и диаграммам, если они корректно загружены.

Стратегии работы с полными документами.

- **Загрузка крупных разделов:** при анализе отчёта достаточно выделить отдельные логические части (раздел «Требования», «Безопасность», «Экономика») и передать их модели по одному сообщению. Модель удержит весь раздел.
- **Целостный обзор:** можно попросить модель сразу дать общее резюме документа, а затем уточнять детали по конкретным пунктам.

Гибридные подходы. Несмотря на расширенное окно, для очень объёмных документов (больше 1000 страниц) всё ещё полезно:

- Делить документ на крупные блоки по темам;
- Формулировать вопросы к каждому блоку;
- Автоматически объединять ответы в единый отчёт.

Какие задачи возможны «на одном заходе»

- **Автоматическая выжимка:** «Перечисли основные рекомендации по энергосбережению» по всему документу.
- **Полная проверка:** «Найди все пункты, где упоминается “надёжность системы” и укажи страницу и формулировку».
- **Мультидокументный анализ:** «Сравни положения трёх разных нормативных актов о теплоснабжении и выдели общие требования».

Особенности больших контекстов:

- **Меньше потеря информации.** Большой фрагмент текста остаётся «в памяти» модели сразу, снижая риск упустить важный нюанс.
- **Более точные сводки.** Модель может строить ссылки внутри текста («на странице 45 говорится…»), если ей указать сохранять нумерацию или разметку.
- **Скорость и стоимость.** Несмотря на расширенные возможности, такие запросы всё ещё дороже и могут идти медленнее, чем у моделей с узким контекстом.

Использование reasoning-моделей при анализе нормативных документов

Рассмотрим, какие задачи при работе с нормативной документацией особенно выигрывают от «рассуждающей» модели (и примеры таких задач), а когда это лишнее.

Примеры задач, где полезно умение рассуждать:

— **сопоставление требований разных документов.** Например, есть приказ, устанавливающий требования, и отчёт или проект, который должен им соответствовать. Задача: выяснить, **какие требования из приказа учтены в отчёте, а какие нет**. На человеческом языке — «проверка соответствия». Для модели это не тривиально: нужно взять каждое требование и найти, где (и как) оно отражено в тексте отчёта. Обычная модель может ответить в общем («в целом отчёт соответствует» или перечислить несколько пунктов, но не все). *Reasoning*-модель будет методично проверять пункт за пунктом. По сути, она выполнит **многошаговое рассуждение**: «Требование 1 — есть/нет? Требование 2 — есть/нет?» и так далее, и составит вывод. Такой пошаговый подход ближе к тому, как работал бы человек-эксперт, и уменьшает шанс пропустить детали.

— **Выявление нарушений, несоответствий, пропусков.** Предположим, есть текст регламента и нужно проверить, нет ли в нём **противоречий** или **расхождений с вышестоящими документами**. Это задача, где модель должна держать в уме несколько условий сразу. *Reasoning*-LLM способна «покрутить» разные части текста, сопоставить их. Например, если в пункте 5 написано одно требование, а в пункте 9 — другое, и они конфликтуют, модель с развитым reasoning может заметить: *«В пункте 5 говорится А, однако в пункте 9 — Б, это выглядит как противоречие»*. Обычная модель, не умея специально анализировать логику, может не обратить на это внимания (она же читает линейно и не гарантировано сопоставит два далёких положения). Аналогично — пропуски: reasoning-модель может заметить, что в списке условий не хватает какого-то очевидного шага (например, все пункты 1, 2, 3 есть, а 4 отсутствует — значит, возможно, выпало требование). Здесь требуется немного *«понимания контекста»*, что выходит за рамки простого продолжения текста — то, что и дают reasoning-модели.

— **Построение сложной логики выполнения условий.** Некоторые нормативные документы — особенно технические регламенты, стандарты — имеют структуру «дерево условий» (if-then). Скажем, «если источник тепла > 100 Гкал, то нужны такие-то мероприятия; если < 100 — другие». Чтобы проверить, соответствует ли конкретный проект нормам, модель должна **разобрать такие логические конструкции и применить к данным проекта**. Это как мини-вычисление или мини-логический вывод внутри текста. Reasoning-модели более приспособлены к таким задачам: они могут внутренне выполнить последовательность шагов (вплоть до решения простых вычислений или логических задач). Таким образом, когда нужно проанализировать нормативный текст на логическом уровне (не просто извлечь фразу, а понять условие и вывести следствие), лучше использовать модель с поддержкой рассуждений.

Когда *reasoning* бывает избыточен? Если задача простая и не требует многошагового анализа, то специальную «думающую» модель можно и не привлекать. Например, **извлечь список определений** из документа или **суммировать содержание главы**

вполне по силам и обычной модели. Более того, *reasoning*-модели обычно работают медленнее: они могут генерировать развёрнутые рассуждения (иногда даже в явном виде, если не отключить) и тратить на это время. Это лишнее, если вы уверены, что задача прямолинейная. Также они **дороже,** если использовать API, где оплата идёт за токены, так как внутренние рассуждения модели учитываются во входном количестве токенов запроса. Поэтому имеет смысл выбирать инструмент по сложности задачи: *«брить окорок — берём топор, брить человека — бритву»*. Для тривиальных выжимок из текста лучше быстрые и дешёвые модели, для комплексной аналитики — продвинутые, несмотря на стоимость.

Ещё один нюанс: чтобы *reasoning*-модель проявила себя, **нужна правильно поставленная инструкция**. Если даже самую умную модель спросить туманно («ну что, есть там проблемы?»), она не станет самовольно расписывать логику. Ей желательно прямо указать: «проанализируй пошагово», «приведи рассуждения для каждого пункта». Иначе вы не получите преимущества её возможностей. А избыточная болтливость reasoning-моделей тоже может мешать — они склонны иногда «думать вслух», что не всегда нужно в ответе. Применение таких моделей тоже требует навыка: *спросить так, чтобы модель выдала именно структурированный анализ*. Но освоив это, вы сможете решать с их помощью очень сложные задачи — например, автоматизированно проверять, соответствует ли сложный проект множеству регламентов сразу, с объяснениями, где именно несоответствие.

Подытожим: reasoning-модели полезны там, где нужна многослойная логика и анализ, например сопоставление документов, проверка сложных условий, поиск противоречий. Они дают более надёжный и «интеллектуальный» результат, но работают медленнее и стоят дороже. Поэтому применять их стоит осознанно — на действительно трудных случаях, где простая модель может ошибиться или упростить ответ. В остальных ситуациях можно использовать обычные LLM для экономии времени и средств.

ЧТО ТАКОЕ PROMPT ENGINEERING — И ПОЧЕМУ ЭТО ВАЖНО

Работая с LLM, очень быстро убеждаешься: **то, как сформулируешь запрос, напрямую влияет на результат**. *Prompt Engineering* — это искусство и наука составления эффективных запросов (prompt'ов) для модели. Проще говоря, **модель не «угадывает» ваши намерения — она точно следует инструкциям, которые удаётся уловить из текста вашего запроса**. Если инструкция размыта или неполная, модель ответит неопределённо или не о том, что нужно. Если запрос чёткий и продуманный — шанс получить нужный результат гораздо выше [6].

Почему так происходит? LLM генерирует ответ, **предсказывая следующие токены на основе входного текста (промпта)** [6]. Промпт включает и ваш вопрос, и контекст, и, возможно, примеры. **Качество и структура этого промпта напрямую влияют на итог** — фактически, формулировка запроса задаёт *рамки*, в которых модель ищет продолжение. Неясный запрос = модель не уверена, что именно от неё хотят, и может «попробовать по-разному». Чёткий запрос = модель видит конкретную задачу и держится её.

В контексте нормативных документов **правильная постановка задач — ключ к полезности LLM**. Документы бывают сложные, многоаспектные; если спросить абы как, модель может уцепиться не за то или утонуть в деталях. Поэтому навык *prompt engineering* ценен: он позволяет действительно «приручить» модель, направить её мощность на решение ваших конкретных вопросов. Хорошо составленный промпт превращает LLM в эффективного ассистента-аналитика, тогда как без инструкций она — просто болтун, который может и не попасть в точку.

ВЫВОДЫ И ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ LLM В РАБОТЕ С НОРМАТИВНОЙ ДОКУМЕНТАЦИЕЙ

Современные большие языковые модели уже сейчас способны существенно облегчить работу с нормативными документами. Они успешно применимы для целого ряда задач, с которыми ежедневно сталкиваются специалисты по документации и планированию. Перечислим основные возможности, которые доступны «уже сегодня».

- **Анализ текстов приказов, правил, схем, положений.** LLM может быстро пробежать по документу и выдать сжатое **резюме**: о чём документ, какие разделы включает, какие основные тезисы. Это ускоряет ознакомление с новыми нормативными актами — вместо часа чтения получить за минуту конспект, а при необходимости углубиться в отдельные места.
- **Извлечение требований и ключевых пунктов.** Модель способна выделить из длинного текста именно **требования («shall/должен») или критерии**, что очень ценно при составлении списков обязательных условий. Например, из ГОСТа на 80 страниц она может вытащить все «предприятие должно обеспечить…» и представить их списком — по сути, готовый чек-лист для проверки.
- **Поиск несоответствий и пропусков.** Если дать модели два текста — например, новый проект

положения и старые требования — она поможет **найти расхождения**. LLM укажет, какие пункты из требований не отражены в проекте, или наоборот, что лишнего появилось. Это ускоряет экспертизу проектов на соответствие базовым документам.

- **Сравнение версий документа.** Очень типовая задача: вышла новая редакция закона, нужно понять, чем отличается от старой. Модель может сопоставить два текста и **найти изменения**: какие фразы изменились, что добавлено или удалено. Человек может пропустить мелкую правку, а ИИ дотошно отметит каждое отличие (конечно, при правильной постановке задачи).
- **Подготовка сводок и справок.** На основе пачки документов модель может **сформировать обзор** для руководства. Например: «Перечисли основные изменения нормативной базы по теплоснабжению за последний год» — модель, получив тексты изменений, выдаст аккуратный перечень. Или: «Подготовь краткую справку по такому-то вопросу на основании нескольких источников» — LLM суммирует информацию, укажет ключевые моменты, и у вас готов проект справки, который останется чуть откорректировать.

Все эти задачи **можно решать уже сейчас** с помощью LLM, экономя время и силы сотрудников. Однако очень важно подчеркнуть: LLM — это не волшебная кнопка «сделать всё идеально». За каждым успешным применением стоит правильно организованная работа с моделью.

Несколько реалистичных тезисов о LLM:

— **модель требует постановки задачи.** Без явного вопроса она не даст полезного ответа. Нужно формулировать запросы так, как вы формулировали бы задачу живому эксперту. Хорошая LLM — послушный исполнитель, но не инициативный аналитик;

— **LLM не работает сама по себе.** Её нужно интегрировать в процесс. Например, если у вас есть система электронного документооборота, модель можно подключить к ней через API, чтобы она помогала с анализом входящих документов. Но она не «включится» магически без ваших действий. Зачастую требуется участие ИТ-специалистов, чтобы внедрить LLM в рабочий цикл — особенно если речь про автоматизацию. В ручном же режиме всё равно нужен пользователь, который будет задавать вопросы и «скармливать» тексты;

— **Качество результата зависит от качества промпта и структуры документов.** Мы уже говорили про prompt engineering — без него никуда. Кроме того, **структурированные документы легче анализировать**. Если документ хорошо разбит на главы, пункты пронумерованы и названы понятно, модель легче сориентируется и выдаст точный ответ. Если текст «сплошной» и хаотичный, ИИ тоже может запутаться или дать сумбурный ответ. Он ведь отражает то, что видит. Поэтому иногда перед использованием LLM имеет смысл чуть подготовить текст: хотя бы разделить явные части, удалить явный мусор (например, сканы печатей или бессмысленные повторы). Правильно подготовленный ввод + чёткий запрос = качественный вывод;

Также необходимо иметь в виду перспективу: **возможности LLM постоянно расширяются**. То, что вчера казалось фантастикой (проанализировать мгновенно 100 документов и выдать связанный отчёт), завтра может стать реальностью с выходом новых моделей и инструментов. Уже появляются специализированные решения под нормативку. Поэтому организациям, работающим с большими документами, стоит присмотреться к LLM — не для того, чтобы слепо довериться, а чтобы **встроить их как вспомогательный инструмент**. В перспективном планировании применения ИИ можно ожидать:

— **Сокращение времени обзора нормативной базы.** Вместо того чтобы вручную читать десятки документов, специалист сможет получать от ИИ краткие выжимки, тратя время только на проверку важных моментов.

— **Подготовку черновиков отчётов и резюме.** LLM может автоматически готовить черновые версии обзоров, сопоставительных таблиц, презентаций по документам. Руководителю или эксперту останется только внести правки и утвердить. Это **ускорит подготовку материалов** для принятия решений.

— **Анализ больших массивов для планирования.** Например, при разработке схем теплоснабжения на десятилетие вперёд нужно учесть множество нормативов, стандартов, прогнозов. ИИ может помочь собрать и структурировать всю эту информацию, указать, где есть противоречия или риски несоответствий. Это не снимает с экспертов функции принятия решений, но делает подготовительный этап гораздо быстрее.

Однако, повторимся: LLM — не панацея. Он склонен ошибаться (см. далее про галлюцинации), требует контроля. Но при грамотном применении это **мощный ускоритель и упрощитель** рутинных операций с текстами. Организации, которые научатся правильно его использовать, смогут выигрывать во времени и эффективности. В случае нормативных документов это значит более быстрое приведение проектов в соответствие требованиям, более своевременное выявление проблем, более качественная подготовка обоснований и справок.

ОГРАНИЧЕНИЯ И РИСКИ ПРИ ИСПОЛЬЗОВАНИИ LLM

Наконец, нельзя не упомянуть об ограничениях и рисках, связанных с применением больших языковых моделей в работе с документами. Несмотря на все их возможности, относиться к результатам нужно с осторожностью и понимать, где модель может подвести.

Главный риск — «галлюцинации» модели. Под этим термином в области ИИ понимают ситуации, когда модель с уверенностью **выдаёт ответ, не опирающийся на реальные данные**, фактически выдумывает его [1]. Причём делает это убедительно: в стиле документа, с деталями, но эти детали могут быть неправильными или вообще отсутствовать в тексте. Например, задаёте вопрос: «В документе есть требования к резервным источникам?» — если модель «решит», что по контексту должны быть, она может ответить: «Да, предусмотрено, что необходимо иметь резервный источник тепла 20 Гкал…» — даже если в тексте этого нет. Она не пытается солгать, просто статистически ей кажется, что *«резервный источник 20 Гкал»* звучит правдоподобно, ведь она где-то такое видела. Для пользователя опасность в том, что ответ звучит авторитетно, а проверять лень — так и проскакивает **вымысел под видом факта** [7].

В нормативной работе это критично: нельзя полагаться на непроверенные факты. Поэтому **все важные выводы LLM нуждаются в верификации**. Особенно если модель ссылается на текст («в пункте 5.3 говорится то-то») — стоит открыть документ и убедиться, что так и есть. Кстати, хороший приём: просить модель **давать ссылки на пункт или цитату из документа** при ответе. Тогда видно первоисточник, и можно сразу сверить. Если модель ссылается на несуществующий пункт — тревожный сигнал, значит, генерирует от себя. Например, в одном испытании ~21 % ответов модели при суммировании регуляторных документов содержали неточности или упущения [8]. То есть 1 из 5 выводов мог быть ошибочным. Без проверки это неприемлемо — представьте, 20 % пунктов в отчёте для руководства неверны.

Необходимость перепроверять критичные выводы — железное правило. LLM можно доверять черновую работу, но не финальное решение. Как минимум, важные пункты надо вычитать, а лучше — иметь специалиста, который пробежит оригинал документа по диагонали и убедится, что модель ничего не нафантазировала сверх. Пока технологии не достигли 100-процентной надёжности, **человек остаётся в цикле принятия решения**. Модель — советник, ускоритель, но не заменитель экспертного суждения.

Ещё одно ограничение — зависимость от качества запроса (промпта). Мы говорили о prompt engineering: если запрос сформулирован нечётко, модель может ответить неправильно или не о том. Это риск: пользователь может получить ответ и не осознать, что сам неправильно спросил, и сделать ложные выводы. Поэтому нужно обучать персонал правильной постановке вопросов к LLM. Фактически, появляется новая компетенция — умение общаться с ИИ, как раньше умение задавать грамотные запросы в поисковике. Без этого можно получить мусор на выходе и винить модель, хотя виноват «кривой» промпт. Осознание этого риска — уже полдела: пользователь должен быть настороже, если ответ модели явно не соответствует вопросу, переспросить или уточнить, а не принимать первое, что она выдала.

Отсутствие прозрачности мышления модели (эффект «чёрного ящика»). Это философское, но и практическое ограничение. Когда эксперт читает документ и делает вывод, он может обосновать: «Вот пункт 7, в нём сказано то-то, поэтому я решил, что…». У модели же внутри нейросети происходят сложнейшие вычисления, и она не может толком объяснить, *почему* выдала именно такой ответ (если только мы явно не заставим её расписать рассуждения). Вы видите только результат. В итоге **сложно проследить логику модели**. Например, она решила, что документ не соответствует требованиям, а вы не сразу поймёте, на каком основании — придётся самому разбираться. Это ограничение: **мы не можем полностью доверять модели, потому что не видим аргументацию**, и иногда не можем даже понять, где она ошиблась. В традиционных алгоритмах можно отладить шаги, а тут — нет, всё скрыто в весах нейросети. Поэтому при конфликте: модель говорит одно, эксперт считает другое — всегда надо склоняться к эксперту (или, по крайней мере, внимательно перепроверять первоисточники).

Практический вывод: LLM требует дисциплины контроля и валидации. Хорошо встроить в процесс двойную проверку: модель даёт результат — человек его просматривает и оценивает перед использованием. В некоторых случаях можно использовать саму модель для проверки: например, задать ей отдельный вопрос «на основании какого фрагмента текста вы сделали этот вывод?» — если она может процитировать соответствующий абзац, доверия больше. Если начинает путаться — повод не доверять данному выводу.

Наконец, стоит упомянуть **этические и правовые риски**, хотя они менее технические. Модель может содержать предвзятости из обучающих данных, может нечаянно сгенерировать нежелательный контент (например, грубость или нечто не соответствующее политике компании). В контексте нормативных документов это вряд ли проявится, но помнить нуж-

но: LLM **не всегда политически корректна или юридически аккуратна**. Если попросить «проанализируй закон и дай совет, как его обойти» — модель может нафантазировать сомнительный совет. Это уже вопрос использования: доверять ли такие задачи ИИ.

Подводя итог по рискам: **использование LLM похоже на работу стажёра-ассистента**. Он очень быстрый, много знает, но **может ошибаться и даже выдумывать, если не знает ответа**, вместо того чтобы сказать «не знаю». Нужен **контроль опытного специалиста**, который проверит и поправит. Нужны **чёткие инструкции**, чтобы не было недопонимания. И нужно понимание, что ответственность остаётся на человеке — ИИ-инструмент её не несёт. Если придерживаться этих принципов, риски можно минимизировать. Как отмечают исследования, при совместной работе «человек + LLM» продуктивность сильно возрастает, но **только при условии, что человек остаётся внимательным и критичным к работе модели** [8].

P.S. Большие языковые модели — мощный новый инструмент для тех, кто работает с текстами, в том числе с нормативными и техническими документами. При правильном подходе они способны взять на себя значительную часть рутины: чтение, обобщение, сопоставление. Это открывает возможности экономить время, быстро получать инсайты из гор документов и даже находить в них то, что человек мог просмотреть. Однако LLM — именно инструмент, а не чудо-мозг. Его эффективность зависит от умения пользующегося. Для руководителей и специалистов, далёких от ИТ, это вызов и возможность одновременно: нужно немного разобраться в принципах работы ИИ, научиться чётко ставить задачи — но освоив это, вы получаете в команду **неутомимого помощника**, который в любое время дня и ночи прочитает за вас тысячу страниц и набросает ответ. Будем использовать его с умом и осторожностью — тогда выгоды перевесят риски.

СПИСОК ЛИТЕРАТУРЫ

1. *What Are Large Language Models (LLMs)? | IBM.*
   https://www.ibm.com/think/topics/large-language-models
2. *Language Models Explained.*
   https://www.altexsoft.com/blog/language-models-gpt/
3. *Reasoning-LLM: архитектура и обзор передовых моделей / Хабр.*
   https://habr.com/ru/companies/selectel/articles/892600/
4. *What are tokens and how to count them? | OpenAI Help Center.*
   https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them
5. *API — что это такое: простыми словами об интерфейсах и интеграции по API / Skillbox Media.*
   https://skillbox.ru/media/code/chto_takoe_api/
6. *Основные приёмы Prompt Engineering — Systems analysis wiki.*
   https://systems-analysis.ru/wiki/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B8%D1%91%D0%BC%D1%8B_Prompt_Engineering
7. *Legal Tech: 5 Use Cases for Large Language Models in Legal Departments | ACC Docket.*
   https://docket.acc.com/legal-tech-5-use-cases-large-language-models-legal-departments
8. *LLMs for Regulatory Compliance Document Processing.*
   https://www.rohan-paul.com/p/llms-for-regulatory-compliance-document