СИГНАЛ

Модуль 3

Контекст и знания

Конспект-опора для обсуждения

КОРОТКАЯ РАМКА

Суть модуля	Качество ответа модели определяется не только “силой” модели и не только формулировкой промпта. Ключевой объект управления - контекст: что именно попало в рабочее окно модели, в каком статусе, с какими приоритетами и насколько чисто это отделено от шума.

Модуль: 3

Тема: контекст, знания, память модели, контекст-инжиниринг

Формат: опорный конспект для обсуждения после лекции

1. Главный вопрос лекции

Главный вопрос	**Как управлять контекстом так, чтобы модель работала предсказуемо: не тонула в лишних данных, не цеплялась за старые числа, не исполняла чужие инструкции и не выдавала уверенный ответ из плохого контекстного пакета?**

Почему это вопрос, а не просто тема

• Если задача простая, промпта часто хватает. Если задача рабочая и длительная, модель начинает зависеть от контекстного пакета сильнее, чем кажется.

• Проблема редко выглядит как “модель ничего не знает”. Чаще проблема в другом: данных слишком много, они противоречат друг другу, не размечены, устарели или попали в чат случайно.

• Главный навык пользователя - не писать красивые промпты, а собирать для модели рабочий стол: нужные источники, правила, ограничения, актуальные данные и понятные метки.

2. Ключевые тезисы

1. У модели нет одной простой “памяти”

Почему важно: Есть знания на весах модели, текущее контекстное окно, проектная память, файлы, коннекторы и инструменты. Это разные механизмы, и они ведут себя по-разному.

Где работает / где ограничение: Работает как объяснительная рамка для любой задачи с AI. Не работает, если воспринимать “память” как человеческое воспоминание.

На чем держится: Разбор весов модели, истории переписки, системных инструкций, файлов и коннекторов.

2. Контекстное окно - зона нашего влияния

Почему важно: Мы не управляем тем, на чем модель обучалась, но управляем тем, что кладем в текущую работу: промпт, файлы, примеры, историю, инструменты, правила.

Где работает / где ограничение: Работает в продуктовых, исследовательских, редакторских и агентских задачах. Не спасает, если данные плохие или неактуальные.

На чем держится: Аналогия с рабочим столом и оперативной памятью: чем чище стол, тем легче работать.

3. Больше контекста не значит лучше

Почему важно: Лишние документы, шум, старые версии и противоречивые цифры могут ухудшить ответ сильнее, чем отсутствие части данных.

Где работает / где ограничение: Работает при анализе отзывов, документов, метрик, кода, переписок. Неочевидно в маленьких задачах, где все помещается в голове человека.

На чем держится: Примеры распыления, сбивания, противоречий и эксперимента “Полдник”.

4. Хороший контекст должен быть размечен

Почему важно: Модели нужно говорить, что является актуальным фактом, мнением, гипотезой, архивом, черновиком, ограничением или источником истины.

Где работает / где ограничение: Работает при долгих проектах и корпусах знаний. Не является гарантией: модель все равно может ошибиться, но меньше путается.

На чем держится: Паспорт контекста, метки источников, правила приоритетов и обновления.

5. Контекст нужно регулярно чистить и переносить

Почему важно: Длинная переписка деградирует: автоматическая компакция может потерять важное. Лучше делать осознанный handoff и начинать новую сессию в логических точках.

Где работает / где ограничение: Работает для разработки, исследований, подготовки материалов, агентных процессов. Не заменяет проверку по источникам.

На чем держится: Разбор compaction/compact, длинных чатов, новых сессий и распределения ролей по чатам.

3. Ключевые идеи для обсуждения

Идея 1. “Память модели” - это несколько разных механизмов

Суть: В лекции разводятся знания на весах модели и данные, которые попадают в текущий запрос. Весовая “память” похожа на опыт человека, прочитавшего много кейсов, но это не таблица фактов. Контекстное окно - то, с чем модель работает прямо сейчас.

Почему это важно: Без этого легко ждать от модели стабильной человеческой памяти и злиться, когда она “забыла” то, что обсуждалось раньше.

Пример / кейс из лекции: Пример с ТРИЗ и кружкой из эпоксидной смолы показывает: модель может уверенно отвечать благодаря обученным связям, но это не значит, что она достает факт из надежного хранилища.

Ограничение / риск / возражение: Аналогия упрощает реальную механику. В разных сервисах поверх модели добавлены свои механизмы памяти, проектов, файлов и поиска.

Вопрос для обсуждения	В каких ваших задачах вы сейчас путаете “модель должна это знать” и “я должен положить это в контекст”?

Идея 2. Контекстное окно - рабочий стол модели

Суть: В контекст попадают системные инструкции, доступные инструменты, примеры, история переписки, прикрепленные файлы и данные из коннекторов. Модель не “вспоминает” это как человек: каждый раз она получает пакет данных и на его основе строит ответ.

Почему это важно: Это переводит разговор из магии в управление: можно думать, что должно лежать на столе, а что надо убрать.

Пример / кейс из лекции: Новый чат похож на чистый стол. Чем дольше переписка и чем больше файлов, тем больше на столе лишнего: старые решения, случайные числа, мусорные фрагменты, нерелевантные документы.

Ограничение / риск / возражение: Чистый стол не гарантирует правильного ответа, но грязный стол почти гарантирует лишние риски на сложной задаче.

Вопрос для обсуждения	Какие три объекта контекста должны лежать на столе модели в вашей типовой рабочей задаче?

Идея 3. Один промпт - лотерея; контекстный пакет - управление вероятностью

Суть: Сильный промпт помогает, но рабочая задача требует достаточных данных: цели, ограничений, источников, формата результата, правил проверки. Контекстный пакет меняет

качество ответа даже при той же формулировке задачи.

Почему это важно: Это снимает ожидание, что можно решить сложную работу одной красивой инструкцией.

Пример / кейс из лекции: В кейсе “Полдник” модели выбирали 5 метрик из 20 для дашборда. При простом промпте качество было слабым или модель отказывалась; при правильном подборе документов обе модели дошли до 5 из 5.

Ограничение / риск / возражение: Один эксперимент не доказывает универсальное правило для всех моделей и задач. Он полезен как демонстрация механизма: структура данных влияет на результат.

Вопрос для обсуждения	Где в вашей работе стоит сначала спросить модель, какие документы ей нужны, вместо того чтобы сразу сваливать все файлы?

Идея 4. Хаотичные данные могут быть хуже отсутствия данных

Суть: Лишний контент распыляет внимание модели. Нерелевантные фрагменты, старые цифры, HTML-мусор, противоречивые версии документов и случайные мнения попадают в один рабочий пакет и начинают конкурировать за влияние.

Почему это важно: Многие используют загрузку файлов как универсальное лекарство: “скину все, пусть разберется”. На сложных задачах это может ухудшить ответ.

Пример / кейс из лекции: В лекции отдельно разбирались распыление, сбивание и противоречие. Также приводился пример с тысячами отзывов: лучше кластеризовать частями, а не просить модель целиком “понять все”.

Ограничение / риск / возражение: Иногда широкий контекст действительно нужен для исследования. Но тогда его надо дробить, фильтровать и проверять промежуточные выводы.

Вопрос для обсуждения	Какие данные в ваших документах выглядят как полезный контекст, но на самом деле являются шумом для решения?

Идея 5. Метки источников - минимальная защита от путаницы

Суть: Документы и фрагменты нужно размечать: актуальные данные, архив, мнение, гипотеза, ограничение, черновик, источник истины. Модель не всегда сама понимает статус текста и может принять старый или чужой фрагмент за команду или факт.

Почему это важно: Рабочие корпуса быстро стареют. Через месяц два файла могут противоречить друг другу, а модель не поймет, какой из них важнее.

Пример / кейс из лекции: Паспорт контекста: кто написал документ, когда, насколько он актуален, какие данные приоритетны, как обновлять. Отдельная метка “мнение” могла бы ослабить влияние мнения руководителя продаж из кейса прошлого модуля.

Ограничение / риск / возражение: Метки не панацея. Они не отменяют проверку источника и не гарантируют, что модель не ошибется.

Вопрос для обсуждения	Какая минимальная схема меток нужна вашему рабочему корпусу: дата, статус, владелец, приоритет, источник истины?

Идея 6. Компакция и handoff - не техническая мелочь, а часть процесса

Суть: Когда контекст переполняется, сервисы сжимают его автоматически. В этот момент часть деталей может потеряться или исказиться. Осознанный handoff лучше: зафиксировать решения, источники, открытые вопросы и текущее состояние задачи.

Почему это важно: Если этого не делать, модель может уверенно продолжать работу с обрывочным остатком старого контекста.

Пример / кейс из лекции: В Claude Code и похожих инструментах compact виден явно. В облачных чатах он скрыт, но длинные переписки тоже со временем становятся менее надежными.

Ограничение / риск / возражение: Summary тоже может ошибаться. Для важных задач нужно просить ссылки на файлы, точные цитаты, список решений и список того, что не переносится.

Вопрос для обсуждения	Где в вашем процессе должна быть логическая точка “закрыли сессию, сделали handoff, начали новый чат”?

Идея 7. Разные роли лучше держать в разных контекстах

Суть: Когда в одной задаче нужны разные режимы мышления - исследователь, аналитик, редактор, критик, директор, разработчик - их можно разводить по чатам или агентам. Так у каждого контекста будет своя задача и меньше мусора.

Почему это важно: Один общий чат быстро превращается в смесь исследований, решений, черновиков, споров и технических деталей.

Пример / кейс из лекции: В лекции упоминалось распределение по чатам/агентам: отдельно исследование, отдельно график, отдельно валидация решения с другой роли.

Ограничение / риск / возражение: Слишком много агентов создают новую проблему: координация, стоимость токенов, потеря общего контекста и иллюзия виртуальной компании.

Вопрос для обсуждения	Какие роли в вашей работе стоит развести, а какие лучше оставить в одном контексте?

Идея 8. Экзокортекс полезен только при дисциплине корпуса

Суть: Долгий проект может жить в папке, Obsidian, базе знаний или проектном хранилище. Но сам факт наличия файлов не делает систему умной: нужны структура, архив, правила доступа, понятные названия и запрет на чтение всего подряд.

Почему это важно: Агенты и модели быстрее человека создают и читают файлы. Без гигиены корпус начинает отравлять сам себя.

Пример / кейс из лекции: В лекции показан личный пример с Obsidian, дневными заметками, профилями, фокусом, командой, подкастом, агентами и архивом. Это работает, но агенты все равно путаются без правил.

Ограничение / риск / возражение: Такой подход требует времени, безопасности данных и понимания, что нельзя складывать в корпус чувствительную информацию без отдельной политики.

Вопрос для обсуждения	Какой самый маленький рабочий корпус можно собрать за один день, чтобы модель стала полезнее уже завтра?

4. Что здесь новое или спорное

• Контекст-инжиниринг оказывается важнее промпт-инжиниринга в рабочих задачах: красивый запрос без нужного пакета данных не выдерживает сложной задачи.

• Большое контекстное окно - не бесплатное расширение памяти. После определенного объема качество может падать из-за распыления, шума и противоречий.

• Память и проекты помогают, но одновременно создают риск: модель может подтянуть старый, случайный или плохо размеченный фрагмент и встроить его в ответ.

• Иногда лучше не давать модели все файлы, а сначала попросить ее выбрать, какие источники ей нужны для решения.

• Работа с AI становится похожа на менеджмент людей: онбординг, общий контекст, правила, источники истины, роли, архив и регулярная синхронизация.

• Виртуальные “команды агентов” звучат привлекательно, но упираются в стоимость, координацию и ограничение контекста. Сама идея не магическая: каждому агенту все равно нужен чистый и достаточный контекст.

5. Вопросы для группы

• Где в вашей текущей работе с AI главная проблема: нехватка контекста, загрязненный контекст или отсутствие разметки источников?

• Какие данные модель должна считать источником истины, а какие - мнением, гипотезой, архивом или черновиком?

• Как вы поймете, что пора завершать чат и начинать новую сессию с handoff-summary?

• Какая задача у вас чаще всего ломается из-за того, что вы даете модели все документы сразу?

• Какие вопросы модель должна задать вам перед тем, как решать вашу типовую рабочую задачу?

• Что в вашем рабочем корпусе нельзя давать модели по соображениям безопасности, персональных данных или коммерческой чувствительности?

6. Что попробовать на практике

Эксперимент 1. Один и тот же запрос в трех контекстах

Что сделать: Возьмите рабочую задачу: выбрать метрики, подготовить решение, собрать план, сделать анализ. Прогоните ее в трех режимах: только промпт; все файлы разом; сначала список доступных файлов, затем только те, которые модель попросила.

Какой результат считать сигналом: третий режим дает более точный, проверяемый и менее расплывчатый результат, а модель лучше объясняет, на какие источники опиралась.

Эксперимент 2. Паспорт контекста для 3-5 документов

Что сделать: Выберите небольшой набор рабочих документов и добавьте к каждому мини-паспорт: владелец, дата актуальности, статус, приоритет, источник истины или справочный материал.

Какой результат считать сигналом: модель перестает смешивать архивные данные с текущими и начинает ссылаться на правильные документы при объяснении решения.

Эксперимент 3. Handoff после длинной сессии

Что сделать: В конце длинной переписки попросите модель подготовить handoff: цель задачи, принятые решения, актуальные факты, источники, открытые вопросы, что не переносить. Начните новый чат только с этим handoff и нужными файлами.

Какой результат считать сигналом: новый чат быстрее входит в задачу, меньше спорит с предыдущими решениями и не тащит старый мусор из переписки.

Эксперимент 4. Анализ отзывов через сжатие по слоям

Что сделать: Не загружайте все отзывы одним массивом. Разбейте их на части, попросите выделить кластеры, затем сведите кластеры в отдельной сессии и попросите другую модель или новый чат проверить структуру.

Какой результат считать сигналом: появляются устойчивые категории, понятные примеры и меньше случайных обобщений по одному яркому отзыву.

7. Открытые вопросы

• Как измерять качество контекстного пакета до того, как модель уже ошиблась?

• Где проходит граница между полезной долгосрочной памятью и контекстным загрязнением?

• Какие типы данных нужно хранить в проектной базе, а какие безопаснее держать вне доступа модели?

• Можно ли стандартизировать паспорт контекста для команды так же, как стандартизируют PRD, RFC или дизайн-доки?

• Когда RAG, MCP и коннекторы действительно нужны, а когда достаточно хорошо названной папки с markdown-файлами?

• Как проверять, что автоматический compact или summary не выкинул критичную информацию?

8. Мини-шпаргалка: как готовить контекст

• Сформулировать задачу и критерий хорошего результата.

• Назвать источники, которые могут понадобиться, но не грузить все подряд.

• Попросить модель выбрать нужные документы, если корпус больше нескольких файлов.

• Разметить каждый источник: актуальное, архив, мнение, гипотеза, черновик, ограничение.

• Указать приоритеты: чему верить при противоречиях.

• Просить модель показывать, откуда она взяла ключевые данные.

• Разбивать длинные задачи на сессии и делать handoff в логических точках.

• Не хранить в корпусе чувствительные данные без отдельной политики доступа.

Фраза, которую стоит запомнить	**Модель ошибается не только когда ей не хватает данных. Она ошибается и тогда, когда данных много, но мы не сказали, какие из них важны, актуальны и обязательны для решения.**

Источник: транскрипт модуля 3 “Контекст и знания”. Конспект собран как опора для обсуждения, а не как дословная расшифровка.