

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

ПО РАБОТЕ С LLM

Подготовка данных, формулировка задач, проверка результатов и контроль ошибок в рабочих процессах.

Содержание охватывает

- контекст и подготовку данных
- промпты, конфабуляцию и галлюцинации
- проверку результатов и режимы рассуждения
- инструменты, API, MCP и агентные циклы
- веб-поиск, безопасность и критерии остановки

Рабочее правило: любую существенную ошибку нужно обнаружить до того, как результат станет решением или действием.

С LLM

Содержание

- 01 Контекстное окно и память модели
- 02 Длинный контекст и разбиение задач
- 03 Подготовка данных
- 04 Промпт как спецификация результата
- 05 Конфабуляция, галлюцинации и сикофантия
- 06 Проверка результата: от формата до смысла
- 07 Рассуждение, Tree of Thoughts и Self-Consistency
- 08 Инструменты: от запроса до действия
- 09 Карта ошибок и способы исправления
- 10 Веб-поиск и проверка источников
- 11 Операционный протокол агента
- 12 Обучение по предпочтениям и отказы
- 13 Безопасность и недоверенные данные
- 14 Когда нужны API и агент
- 15 Оценка качества на собственных задачах
- 16 Готовые рабочие процедуры
- 17 Чек-листы и точки остановки
- 18 Границы применимости
- 19 Источники

1. Контекстное окно и память модели

Языковая модель отвечает на основе данных, доступных ей в конкретном вызове: системных инструкций, сообщения пользователя, истории диалога, переданных файлов и результатов инструментов. Контекстное окно ограничивает объём информации, который модель может учитывать в рамках вызова. Это не долговременная база данных и не гарантия того, что каждая фраза из предыдущих сообщений будет использована правильно.

Если важное требование не попало в текущий контекст и не доступно через поиск, файл, базу данных или другой внешний механизм, модель не сможет надёжно опираться на него. В долгой работе критичные договорённости, определения, ограничения и значения лучше хранить отдельно и передавать повторно по мере необходимости.

Контекстная инженерия – это не только выбор формулировки запроса. Она включает отбор нужных документов, примеров, сведений о состоянии задачи, истории, результатов поиска и доступных инструментов. В контекст следует помещать не максимум материала, а материал, который нужен для следующего шага. См. заметку Simon Willison о context engineering и LLM wiki Andrej Karpathy.

Для практической работы различайте три ограничения:

- Технический лимит. Максимальная длина входа и выхода конкретной модели или API. Его проверяют по документации выбранной модели и версии сервиса.
- Полезный объём. Количество сведений, которое действительно нужно для текущей задачи. Большой объём нерелевантного текста отнимает место у инструкций и важных данных.
- Качество использования контекста. Даже материал, который технически помещается в окно, модель может использовать неравномерно или неверно.

Универсального порога в токенах, после которого качество обязательно ухудшается, нет. Он зависит от модели, задачи, формата и структуры данных. Рабочий предел следует определять на типичных примерах своей задачи, а технический предел – по документации конкретной модели.

2. Длинный контекст и разбиение задач

2.1. Почему положение информации имеет значение

Исследование Lost in the Middle показало, что на ряде задач поиска сведений в длинном контексте модели лучше использовали релевантные данные в начале или конце входа, чем сведения в середине. Эффект зависел от постановки задачи и модели; исследование не задаёт универсального порога длины для всех современных систем. Liu et al., 2023.

Поэтому нельзя предполагать, что важное требование, спрятанное в середине большой подборки, будет учтено так же надёжно, как отдельная инструкция. Критические условия лучше выделять, данные – структурировать, а итоговые ограничения – повторять перед этапом, на котором они должны применяться.

2.2. Когда делить работу на части

Тип задачи	Как организовать работу	Что нельзя потерять
Большой набор каждого однотипных документов	Обрабатывать одинаковыми пакетами и шаблонами	ID документа, страницы, источник значения
Документ с независимыми разделами	Извлечь разделы по отдельности, затем свести результаты	Общие определения, ограничения и исключения
Последовательные их этапы	Сначала извлечь данные, затем сравнить или рассчитать, после этого формулировать вывод	Исходные числа и ссылки; не заменять пересказом модели
Взаимозависимые фрагменты	Передавать вместе или составить карту зависимостей и ссылки на исходные места	Условия, заданные в одном месте и применимые к другому

Тип задачи	Как организовать работу	Что нельзя потерять
Короткий документ с увеличивать простым вопросом	Обычно достаточно одного вызова	Не дробить без причины и не риск ошибок синтеза

Деление полезно, если у задачи есть естественные этапы, разные типы данных или отдельные критерии проверки. Оно вредно, если разрывает взаимосвязанные условия. В последнем случае нужно сохранять карту зависимостей и точные ссылки на первоисточники.

2.3. Как передавать состояние между этапами

После каждого крупного шага сохраняйте карточку состояния, которая нужна следующему этапу:

- цель и критерий готовности;
- установленные факты с единицами, источником, версией и местом в документе;
- допущения и правила, которые нельзя менять незаметно;
- пробелы, противоречия и неподтверждённые гипотезы;
- незавершённые проверки и следующие действия.

Краткое резюме, составленное моделью, само может содержать пропуски. Критичные значения лучше хранить в таблице или структурированном объекте вместе с точными ссылками на источник. Текстовое резюме используйте для навигации, а не как единственную базу фактов.

3. Подготовка данных

Плохие входные данные нельзя надёжно исправить удачным промптом. Если перепутаны версии документов, потеряны строки, неизвестны единицы или смешаны периоды, модель может логично рассуждать на неверной основе. Подготовка данных – часть контроля качества, а не косметическая очистка.

3.1. Проверки до передачи модели

Что проверить	Практическое действие	Когда остановиться
Происхождение значение	Зафиксировать источник, дату получения, версию и владельца данных	Неясно, откуда взялось критичное
Полнота запись или	Сверить число строк, страниц, приложений, диапазоны дат и обязательные поля	Отсутствует нужная страница, часть таблицы
Единицы неизвестны	Явно указать единицы; приводить их к общему виду только по заданному правилу	Единицы несовместимы или
Дубли разных	Задать критерий совпадения и сохранять журнал объединённых или удалённых строк	Нельзя отличить дубль от двух событий
Версии применима	Разнести номер редакции, дату действия и статус документа	Непонятно, какая редакция
Пропуски смысл	Различать ноль, пустое значение, «не применимо» и «не проверено»	Система вынуждена угадывать пустого поля
Противоречия	Сохранять конфликтующие значения вместе с источниками	Нет правила выбора между несовместимыми данными
Персональные и передавать секретные сведения	Убирать лишнее и проверять разрешённость передачи во внешний сервис	Нет оснований или разрешения данные

3.2. Ноль, пропуск и неизвестность – разные состояния

Ноль может быть реальным измерением. Пустое поле может означать отсутствие записи. «Не применимо» означает, что параметр не имеет смысла для данного объекта. «Не проверено»

С LLM

означает, что проверка ещё не выполнена. Если привести все эти состояния к нулю, расчёт может формально сойтись и при этом оказаться содержательно неверным.

Например, пустое поле расхода материала не означает расход 0. Если модель подставит ноль, она может заключить, что получена стопроцентная экономия. Политика данных должна заранее определять, как представлять каждый вид пропуска.

3.3. Формат исходных файлов

Для таблиц задавайте явные имена столбцов, устойчивые идентификаторы строк, единицы измерения и отдельное поле источника. Для документов сохраняйте название, редакцию, дату, нумерацию страниц и возможность вернуться к цитате в оригинале. Не используйте порядок строк или страниц как единственный идентификатор.

Перед анализом большого набора полезно сначала запросить инвентаризацию: список доступных файлов, версии, число страниц, заголовки и доступные поля. Затем – извлечение данных, и только потом интерпретацию. Так легче обнаружить отсутствующие файлы и неверное чтение исходников.

4. Промпт как спецификация результата

Промпт лучше работает как проверяемое техническое задание, а не как попытка одной фразой гарантировать правильное мышление. Он должен задавать цель, входные данные, правила преобразования, ограничения, критерии приёма и формат ответа.

4.1. Структура рабочей инструкции

1. Цель. Какой результат нужен и кто будет его использовать.
2. Источники. Какие файлы, даты, версии и поля считать исходными данными.
3. Правила. Единицы измерения, приоритет источников, методы расчёта и допустимые преобразования.
4. Ограничения. Какие данные нельзя додумывать и какие выводы нельзя делать без подтверждения.
5. Неопределённость. Что делать при нехватке данных или противоречиях.
6. Проверка. Какие поля, числа, ссылки и зависимости необходимо сверить.
7. Выход. Формат, обязательные поля, уровень подробности и критерии готовности.
8. Граница действия. Что модель может сделать самостоятельно, а что требует согласования человека.

4.2. Пример инструкции для сравнения двух технологических процессов

Задача: сравнить варианты А и Б и определить, какие различия влияют на длительность, расход и качество.

Источники: только приложенные маршрутные карты редакций 4 и 5. Сначала перечислить файлы, версии, даты и доступные поля.

Обработка: извлечь сопоставимые операции в таблицу, указать страницу или фрагмент источника.

Приводить единицы к общему виду только по указанному правилу.

Проверка: число операций должно совпасть с исходными картами либо каждое различие должно быть объяснено. Суммы длительностей пересчитать независимо.

Ограничения: отсутствие значения не считать нулём; причину изменения не выводить только из совпадения; не использовать внешние данные без отдельной пометки.

Ответ: разделить подтверждённые факты, расчёты, предположения и вопросы к технологу.

4.3. Запреты и позитивные формулировки

Указание «не выдумывай» недостаточно конкретно. Проверяемая формулировка: «Если поле не найдено в источнике, верни NULL и укажи причину; не подставляй предполагаемое значение».

Вместо «проверь ответ» полезнее написать: «Пересчитай все суммы независимо, перечисли строки, участвующие в расчёте, и отметь расхождения». Вместо «будь точным» – «Для каждого фактического утверждения о дате, числе или нормативе укажи источник». Позитивные инструкции задают желаемый результат напрямую. Отрицательные фиксируют недопустимое поведение. На практике часто нужны оба типа. Нельзя считать универсальным законом утверждение, что любое упоминание запрещённого действия автоматически повышает вероятность его выполнения. Термин *negative constraint failure* можно использовать как рабочее обозначение нестабильного исполнения запретов, но без конкретной первичной публикации не следует выдавать его за общепринятую теорию.

4.4. Примеры и структура

Примеры формата «вход → правильный выход» полезны для классификации, извлечения, преобразования текста и строгого форматирования. Они показывают конкретный критерий результата и иногда заменяют длинное описание словами. Заголовки, XML-теги и другие явные разделители помогают отделить инструкции от документов и примеров. Они структурируют контекст, но сами по себе не гарантируют точность. Если фамилии, даты, номера деталей, юридические требования или другие сущности должны обрабатываться по специальным правилам, это нужно указать явно. Слишком длинный промпт может содержать повторы, устаревшие правила и противоречия. Постоянные инструкции следует регулярно очищать, а изменения – проверять на фиксированном наборе задач. См. официальные руководства по *prompt engineering* OpenAI и Anthropic.

5. Конфабуляция, галлюцинации и сикофантия

5.1. Конфабуляция и галлюцинации

В пользовательской практике словом «конфабуляция» часто называют генерацию правдоподобных, но неподтверждённых деталей. В исследованиях языковых моделей чаще используется термин «галлюцинация». Для контроля важнее не название, а тип наблюдаемой ошибки: ложный факт, выдуманная цитата, несуществующая ссылка, неверное извлечение, смешение источников, вывод без достаточных оснований или необоснованная точность. Модель может написать правдоподобное, но ложное утверждение уверенным тоном. Она не обязательно «знает», что говорит неправду: ответ генерируется на основе обученных закономерностей и текущего контекста. В работе OpenAI от 5 сентября 2025 года обсуждается, как стандартные процедуры обучения и оценки могут поощрять угадывание вместо признания неопределённости. Это поддерживает практическое правило: «данных недостаточно» должно быть допустимым результатом, если оснований для ответа нет. *Why language models hallucinate.*

Вид ошибки	Пример	Как проверять
Придуманный факт документ, каталог	Названы несуществующие дата, автор, характеристика или функция	Проверить официальный или первичный источник
Поддельная цитата или ссылка	Страница открывается, но нужного тезиса в ней нет	Найти конкретный фрагмент; не только на заголовок страницы
Смешение источников и единицы	Показатель одного периода приписан другому	Хранить источник, дату, объект рядом с каждым значением
Ложная точность происхождения точность	Указано точное число без источника или метода	Проверить методику и числа; убрать необоснованную
Логический скачок альтернативы и	Из корреляции сделан вывод о причине, из одного примера – общее правило	Проверить посылки, контрпримеры
Неверное извлечение проверить полноту	Модель перепутала значение, строку, страницу или редакцию	Сверить с оригиналом и извлечения

5.2. Сикофантия

Сикофантия – склонность модели соглашаться с пользователем или менять позицию в сторону его ожиданий, даже если независимая оценка требует возражения. Исследование Sharma et al. обнаружило такое поведение у изучавшихся ассистентов в нескольких типах задач и рассмотрело связь с человеческими предпочтениями при обучении. Это не означает, что каждый ответ сикофантен, но риск нельзя игнорировать. Towards Understanding Sycophancy in Language Models. Чтобы уменьшить риск:

- сначала сформулируйте вопрос нейтрально, если нужна независимая оценка;
- просите отдельно привести подтверждения, опровержения и факты, способные изменить итог;
- проверяйте аргументы против исходной гипотезы, а не только в её пользу;
- если модель изменила позицию после того, как вы высказали своё мнение, попросите указать новые доказательства, вызвавшие изменение;
- для важных решений используйте источник, расчёт или проверяющего, независимого от первоначального ответа.

Рабочая формулировка: «Рассматривай моё утверждение как гипотезу. Проверь его независимо. Приведи аргументы за и против, укажи пробелы и условия, при которых вывод изменится. Если оснований недостаточно, скажи это прямо».

5.3. Почему модель не должна быть единственным проверяющим

Повторный запрос к той же модели может найти часть ошибок, но не является независимой проверкой. Модель может повторно использовать ту же неверную предпосылку, исходные данные и способ рассуждения.

Для арифметики используйте независимый пересчёт. Для нормативного требования – действующую редакцию первичного документа. Для производственного параметра – измерение, документацию или утверждённую технологическую карту. Для критического вывода – проверку специалистом. Убедительный стиль ответа и согласие нескольких генераций не заменяют доказательства.

6. Проверка результата: от формата до смысла

Базовую проверку следует включать в процесс с самого начала. Не нужно сначала выстраивать генерацию, а после первой ошибки добавлять контроль. При этом проверка – не просто просьба к модели ещё раз подтвердить собственный ответ. Используйте формальные правила, независимые источники и экспертную проверку там, где это оправдано.

Не подключайте сразу все методы – длинный промпт, несколько ветвей рассуждения, множество инструментов и несколько проверок. Начните с минимального контрольного контура, который ловит основные ошибки данной задачи. Усложняйте систему только после измерений: проблемы с контекстом решайте подготовкой данных, ошибки вычислений – проверяемым расчётом, ошибки вызова инструмента – схемой и валидацией параметров.

6.1. Пять слоёв проверки

Уровень	Что проверять	Пример
Вход	Обязательные поля, типы, единицы, диапазоны, версии, полнота документов	План и факт заданы в одинаковых единицах
Структура приходится	Формат JSON, обязательные ключи, допустимые значения, ID, число записей	На каждую строку исходника одна запись результата
Преобразование равно	Потери строк, дубли, правила нормализации, целостность ссылок	Количество обработанных строк количеству входных за вычетом документированных исключений
Содержательная итоговой корректность	Источник, арифметика, логические связи и контрпримеры	Сумма операций совпадает с длительностью

Уровень	Что проверять	Пример
Риск действия не	Пороги, разрешения, подтверждение человека, откат	Запись в производственную систему выполняется без подтверждения

6.2. Что можно проверять автоматически

- обязательные поля, типы данных, формат даты, диапазоны чисел и разрешённые значения;
- число записей до и после обработки, пропущенные и повторяющиеся ID;
- суммы, балансы, арифметические выражения и контрольные итоги;
- существование файлов, ссылок, кодов деталей и записей в справочнике;
- соответствие JSON заданной схеме и отсутствие неожиданных полей;
- наличие ссылки на источник у тех утверждений, которые система маркирует как фактические.

6.3. Что формальная проверка не доказывает

Корректный JSON не гарантирует верный вывод. Сумма может сходиться, если неверно извлечены исходные числа. Ссылка может открываться, но не подтверждать утверждение. Значение может попадать в допустимый диапазон, но относиться к другой детали или версии документа. Поэтому проверки должны соответствовать смыслу задачи, а не только формату.

6.4. Разделяйте факт, интерпретацию и рекомендацию

В отчёте полезно отличать:

- Факт: прямо подтверждается источником или измерением.
- Расчёт: получен из явно указанных значений по указанной формуле.
- Интерпретация: объяснение фактов; может зависеть от допущений.
- Рекомендация: предлагаемое действие с указанием рисков и условий.
- Неизвестно: данных недостаточно для надёжного вывода.

Такая разметка упрощает проверку и не позволяет догадке незаметно превратиться в факт.

7. Рассуждение, Tree of Thoughts и Self-Consistency

7.1. Что именно задаёт процедуру рассуждения

Здесь следует различать несколько уровней.

Инструкция в промпте. Можно попросить разбить задачу на этапы, перечислить предположения, сравнить варианты или проверить расчёт. Это описывает желаемую процедуру, но не гарантирует её правильного выполнения.

Режим или параметр модели/API. Некоторые модели и интерфейсы предоставляют настройки, связанные с режимом работы или выделением вычислительных ресурсов на сложные задачи.

Доступность и смысл настроек зависят от модели и версии API. Универсального переключателя рассуждения для всех LLM нет.

Внешний алгоритм. Программа может генерировать несколько вариантов, оценивать их, отбрасывать ветви, хранить промежуточные состояния, пересчитывать значения и выбирать или останавливать решение. Это уже управляющая логика, а не только текстовая инструкция.

Инструменты. Калькулятор, поиск, интерпретатор кода или валидатор выносят отдельные операции из генерации текста и возвращают результат, который можно проверить независимо.

Поэтому ToT и Self-Consistency нельзя считать магическими фразами для промпта. Для воспроизведения соответствующих методов обычно нужны повторные вызовы, способ сравнивать результаты и критерий отбора. Промпт может имитировать часть процедуры, но не обязательно воспроизводит экспериментальный алгоритм.

С LLM

7.2. Tree of Thoughts (ToT)

Tree of Thoughts рассматривает решение как поиск по нескольким возможным промежуточным состояниям. Система предлагает кандидаты, оценивает их, продолжает перспективные ветви и при необходимости возвращается к предыдущему состоянию. Метод предназначен для задач, где полезны поиск, стратегический просмотр вперёд и пересмотр раннего выбора. В исследовании Yao et al. на задаче Game of 24 авторы сообщили о 4% успешных решений для GPT-4 с обычной цепочкой рассуждений и 74% для конкретной реализации ToT. Это результат одной экспериментальной постановки, а не обещание аналогичного улучшения в рабочих задачах. Tree of Thoughts.

ToT уместен, когда:

- есть несколько допустимых путей решения;
- ранний выбор влияет на последующие шаги;
- промежуточные состояния можно сравнить;
- существуют критерии отбраковки ошибочных ветвей;
- полезны откат и поиск альтернатив.

Это может относиться к комбинаторному поиску, планированию последовательности действий, некоторым сценарным задачам и головоломкам с чёткими ограничениями.

ToT обычно не оправдан, когда требуется извлечь короткий факт, переписать текст, выполнить простое форматирование или классификацию с ясным правилом. Он увеличивает число вызовов и сложность контроля. Если нет надёжного критерия оценки промежуточных вариантов, ветвление может лишь размножить ошибки.

7.3. Self-Consistency (SC)

Self-Consistency генерирует несколько вариантов решения и выбирает наиболее согласованный ответ. В статье Wang et al. 2022 года метод улучшал результаты на нескольких бенчмарках, но величина прироста различалась: +17,9 процентного пункта на GSM8K, +11,0 на SVAMP, +12,2 на AQuA, +6,4 на StrategyQA и +3,9 на ARC-challenge. Эти цифры относятся к конкретным моделям, наборам и экспериментальной процедуре; они не означают универсального прироста точности на 10–20% в любой задаче. Self-Consistency Improves Chain of Thought Reasoning.

SC имеет смысл, когда независимые попытки могут выявить нестабильность, а ответы можно сравнить по чётким критериям: например, получить численный результат или выбрать категорию из фиксированного списка. Метод слабее, если все варианты опираются на один и тот же неверный вход, требуется актуальный внешний факт или правильность нельзя определить голосованием. Согласованность показывает стабильность ответа, но не доказывает его истинность.

7.4. Стоит ли платить за дополнительные вызовы

1. Соберите набор типовых задач с заранее известными правильными ответами.
2. Измерьте базовую долю ошибок, стоимость и задержку.
3. Запустите SC или ToT на том же наборе и с тем же критерием оценки.
4. Посчитайте критические ошибки, ложные подтверждения и случаи, когда метод должен был отказаться.
5. Оставляйте более дорогой метод только там, где измеренный выигрыш оправдывает затраты. Сначала стоит внедрять дешёвые формальные проверки и независимые вычисления. Добавлять SC или ToT следует после измерений, если они решают конкретную проблему лучше базового метода.

С LLM

8. Инструменты: от запроса до действия

8.1. Что такое инструмент

Инструмент – внешняя функция или сервис, доступный модели через формализованный интерфейс. Это может быть поиск, чтение файла, запрос к базе данных, вычисление, запуск ограниченного кода, обращение к API, создание документа, отправка сообщения или изменение записи. Обычная модель генерирует ответ на основе контекста. При использовании инструментов она может сформировать запрос на выполнение внешней операции. Модель не получает произвольные полномочия только потому, что в промпте написано «используй инструмент»: приложение должно предоставить операцию и определить, что ей разрешено делать.

8.2. Tool call – запрос, а не исполнение

Типичный цикл выглядит так:

Запрос пользователя → выбор инструмента → аргументы → проверка → исполнение → результат или ошибка → интерпретация → проверка → ответ или следующий шаг.

Модель формирует структурированный вызов. Затем его исполняет приложение, сервер или соответствующая инфраструктура поставщика. Вызов сам по себе не доказывает, что операция успешно выполнена. Документация Anthropic описывает разницу между клиентскими и серверными инструментами; OpenAI описывает подключение функций через function calling. См. Tool use with Claude и Function calling.

8.3. Четыре части инструмента

1. Описание. Для чего предназначен инструмент, когда его вызывать и когда не вызывать, что означают параметры.
2. Схема входа и выхода. Типы, обязательные поля, допустимые значения и формат результата.
3. Реализация. Код, SQL-запрос, API или другая операция, которая действительно выполняет работу.
4. Политика исполнения. Права доступа, проверка параметров, журналы, лимиты, подтверждение человека и откат.

Инструкция не заменяет реализацию. Схема не гарантирует, что модель передала правильные по смыслу данные. Код может технически сработать, но получить запись не того объекта. Для надёжной системы нужны все четыре компонента.

8.4. Как описывать инструмент

Слишком общее имя вроде search не объясняет модели, что именно искать, когда вызывать функцию и как интерпретировать результат. Описание должно быть предметным, а не декоративным.

Для каждого инструмента укажите:

- имя и назначение;
- условия использования и случаи, когда его не следует применять;
- параметры, их типы, смысл и обязательность;
- допустимые значения и правила согласованности;
- структуру результата и возможные статусы;
- типовые ошибки и порядок их обработки;
- изменяет ли функция внешнее состояние;
- какие проверки и подтверждения обязательны.

Строгая схема может проверить типы и обязательные поля, но не доказывает смысловую правильность аргументов. Например, она способна проверить, что object_id – строка, но не то, что ID относится к нужному изделию. Для этого нужен справочник или бизнес-правило.

РУКОВОДСТВО ПО РАБОТЕ С LLM

8.5. Пример описания инструмента

Имя: `calculate_deviations`.
Назначение: рассчитать абсолютное и процентное отклонение факта от плана.
Когда вызывать: план и факт относятся к одному показателю, объекту и периоду и заданы в одинаковых единицах.
Не вызывать: если единицы различаются, период или объект не совпадают либо значения неизвестны.
Вход: `plan` (число), `actual` (число), `unit` (разрешённое значение), `object_id` (строка), `period` (строка).
Выход: абсолютное отклонение, процентное отклонение или `null` с причиной, использованная формула и статусы проверки.
Ограничения: если `plan = 0`, процентное отклонение не рассчитывать. Не конвертировать единицы без заданного правила. Функция выполняет расчёт и не меняет исходные данные.
Ошибки: `missing_field`, `invalid_type`, `unit_mismatch`, `object_mismatch`, `period_mismatch`.
Пример условной JSON-схемы:

```
{
  "type": "object",
  "properties": {
    "plan": {"type": "number"},
    "actual": {"type": "number"},
    "unit": {"type": "string", "enum": ["pcs", "kg", "min"]},
    "object_id": {"type": "string"},
    "period": {"type": "string"}
  },
  "required": ["plan", "actual", "unit", "object_id", "period"],
  "additionalProperties": false
}
```

Схема проверяет типы и разрешённые значения. Она не проверяет, что выбран нужный план, изделие, период или редакция. Эти проверки относятся к бизнес-логике.

8.6. Библиотека инструментов и поиск нужной функции

Приложение может предоставлять собственные функции, оборачивать методы существующих библиотек или подключать внешние сервисы через API. Обычно безопаснее создавать узкие обёртки над разрешёнными операциями, чем давать модели полный доступ к SQL, файловой системе или терминалу. Например, вместо произвольного SQL можно предоставить `query_production_metrics`, которому разрешены определённые таблицы и типы запросов.
Когда доступных инструментов мало, их описания можно передавать сразу. При десятках и сотнях функций описания и схемы занимают заметную часть контекста и усложняют выбор. В этом случае используется группировка или динамический поиск: система сначала подбирает подходящие инструменты, затем передаёт их определения модели. В документации OpenAI Agents SDK описаны инструменты и отложенная загрузка через `tool search`. OpenAI Agents SDK: Tools.

8.7. Read, compute и write

Инструменты различаются по последствиям.

Тип операции контроля	Примеры	Минимальные меры
Read — чтение фиксация источника утечки	Поиск, чтение файла, SELECT-запрос, просмотр записи	Ограничение доступа, и даты, защита от
Compute — контрольные суммы, вычисление проверка результата	Калькулятор, обработка таблицы, генерация файла	Проверка входа, безопасная среда,
Write — изменение параметров, подтверждение, журнал и	Отправка письма, публикация, изменение записи, удаление	Проверка прав и предпросмотр, откат

Чтение и изменение состояния нельзя считать одинаково безопасными. Модели не следует выдавать одинаковую свободу для «посмотреть» и «изменить / удалить / отправить». Для write-операций нужны более строгие права, явное описание последствий и подтверждение там, где действие трудно обратить.

8.8. Ошибки и параллельные вызовы

Инструмент может вернуть ошибку: файл не найден, недостаточно прав, неверный параметр, тайм-аут, ошибка API или частичный результат. Ошибку нельзя скрывать и нельзя считать успешным выполнением. Она должна возвращаться в рабочий цикл в явном виде.

Если несколько операций независимы, их можно выполнить параллельно. Например, чтение файла и запрос справочника могут идти одновременно, если результат одного не требуется для другого. Параллельность уменьшает задержку, но приложение должно корректно обрабатывать частичные успехи, ошибки и ситуацию, когда только часть операций завершилась.

Повторный вызов допустим по ограниченному правилу. Нельзя бесконечно повторять один и тот же ошибочный запрос или дублировать действие, которое уже могло выполниться. Для операций изменения состояния важны уникальные ID операций, проверка статуса и защита от повторного исполнения.

8.9. MCP

Model Context Protocol (MCP) – открытый протокол, стандартизирующий способ подключения AI-приложений к источникам данных и инструментам. В архитектуре клиент подключается к MCP-серверу, который предоставляет ресурсы или операции. Это упрощает повторное использование интеграций между совместимыми приложениями. MCP: Introduction.

MCP стандартизирует подключение, но не подтверждает надёжность бизнес-логики конкретного сервера, качество его данных или безопасность конфигурации. Подключённый сервер по-прежнему требует настройки разрешений, контроля входов, журналирования и проверки результата.

9. Карта ошибок и способы исправления

Дробить ошибки на микрокатегории полезно только тогда, когда у каждой есть отдельный способ обнаружения или исправления. Для практического контроля достаточно проследить цепочку от входных данных до действия.

Этап	Что может пойти не так	Как обнаружить	Где исправлять
Входные данные подготовка, остановка	Не та версия, неполная таблица, неверные единицы	Сверить с оригиналом и метаданными, проверить контрольные количества	До вызова: маркировка, при неполноте
Выбор функции, инструмента доступных	Выбран не тот инструмент, например write вместо read	Сопоставить цель с разрешёнными функциями и журналом вызова	Описание маршрутизация, ограничение инструментов
Аргументы бизнес- справочник,	Неверный ID, дата, единица или диапазон	Сравнить аргументы с исходными данными и схемой	Структурная и валидация, уточнение данных
Исполнение исключений, повтор,	Вызов не дошёл, тайм-аут, отказ или частичный результат	Проверить статус, код ошибки и журнал сервиса	Обработка ограниченный fallback
Полученные запрос, данные или	Ответ устарел, неполон или относится к другому объекту	Проверить источник, дату, объект, версию и обязательные	Повторный другой источник

		поля	остановка
Интерпретация	Модель неверно прочитала	Сопоставить объяснение с	
Структурированный	единицу, статус или поле	контрактом инструмента и	результат,
уточнение		исходным ответом	схемы,
дополнительная			проверка

Этап	Что может пойти не так	Как обнаружить	Где исправлять
Вывод расчёт, предметной экспертная	Данные верны, но заключение из них не следует	Проверить предпосылки, расчёт, причинность и контрпример	Независимый правило области, проверка
Последующее действие полномочий,	Ошибочный вывод отправлен в систему или человеку как окончательный	Сверить журнал и фактическое состояние системы	Предпросмотр, подтверждение, ограничение откат

9.1. Два различия, которые нельзя смешивать

«Модель сформировала неверный запрос» – описание происхождения ошибки. «Инструмент получил неверные параметры» – описание того, что дошло до границы исполнения. Между этими этапами может стоять валидатор: неправильный запрос можно отклонить до запуска инструмента. В журнале полезно сохранять исходные аргументы модели и результат проверки.

«Инструмент вернул некорректный результат» и «модель неверно истолковала ответ» – разные точки отказа. Сначала проверяют исходный результат инструмента, его статус и источник. Затем – то, как модель преобразовала результат в вывод. Для первого случая может потребоваться исправить запрос, источник или код; для второго – уточнить формат, схему, интерпретацию или правило проверки.

9.2. Где исправлять ошибку: в промпте или в коде

Тип ошибки	Первый способ исправления
Нет обязательного поля, неверный тип или формат даты	Автоматическая валидация и схема
Правильный тип, но неверный объект или период	Бизнес-проверка по ID и справочнику
Выбран неподходящий инструмент	Уточнить описание, маршрутизацию и доступный набор функций
Не хватает фактической информации	Получить источник, уточнить у пользователя или отказаться от вывода
Вывод не следует из верных данных	Независимый расчёт, контрпример, правило предметной области, эксперт
Действие слишком рискованно	Подтверждение человека, минимальные права, журнал и откат
Повторяется одна и та же ошибка формата	Пример правильного ответа, схема, валидатор и тестирование промпта

Если ошибку можно дешёво и однозначно поймать программой, проверка в коде обычно надёжнее повторной просьбы к модели. Промпт помогает задавать поведение, но не заменяет технические гарантии.

10. Веб-поиск и проверка источников

Веб-поиск снижает неопределённость только тогда, когда находит подходящие источники и модель правильно их интерпретирует. Количество запросов, открытых страниц или ссылок в финальном ответе само по себе не измеряет качество исследования.

10.1. Как работает исследовательский поиск

Простая схема выглядит так: запрос → выдача → открытие страниц → чтение → ответ. Поисковый агент может работать иначе: уточнить вопросы → составить план → выполнить целевые запросы →

отобрать документы → извлечь подтверждения → сопоставить источники → выявить пробелы → остановиться, когда доказательной базы достаточно.

Хороший поисковый процесс не обязан делать много запросов. Он должен найти доказательства, достаточные для ответа. Лишний поиск может засорить контекст повторяющимися, вторичными или противоречивыми материалами. При этом для сложного факта может потребоваться много источников; число шагов определяется предметом исследования, а не само по себе служит показателем качества. OpenAI описывает задачи оценки поисковых агентов в публикациях BrowseComp и WebGPT.

Нужно разделять три показателя:

- сколько запросов выполнила система;
- сколько источников она использовала в ходе работы;
- сколько ссылок она показала пользователю или сколько сайтов пользователь открыл.

Эти числа не обязаны совпадать. Краткий список ссылок в ответе не доказывает, что поиск был поверхностным, а длинный список — что исследование было качественным.

10.2. Проверка отдельного утверждения

1. Разбейте ответ на проверяемые тезисы: одно предложение может включать несколько разных фактов.
2. Определите тип источника, подходящий для каждого тезиса: стандарт, закон, документация производителя, научная работа, официальный отчёт, измерение.
3. Откройте источник и проверьте конкретный фрагмент, дату, объект и область применимости.
4. Отделите то, что прямо утверждает источник, от интерпретации модели.
5. Проверьте более свежую редакцию, если факт мог измениться, и найдите противоречащие данные для спорных тезисов.
6. Сохраните рабочую ссылку рядом с утверждением и укажите дату проверки для изменяемых сведений.
7. Если подтверждения нет, снимите тезис или обозначьте его как гипотезу либо непроверенный факт.

10.3. Приоритет источников зависит от вопроса

Что устанавливается	Предпочтительный источник	На что обратить внимание
Требование стандарта или закона	Официальная действующая редакция	Юрисдикция, дата вступления, применимость
Возможность продукта или API	Официальная документация и release notes	Версия, тариф, дата документации
Результат переноса эксперимента	Научная статья, методика и данные, если доступны	Условия эксперимента и границы результата
Показатель компании или отрасли	Отчёт компании, регулятора или статистической службы	Определение показателя и период
Историческое событие датой	Архивный документ, первичное свидетельство, академическая публикация	Разница между датой события и публикации
Практический опыт	Измерение, протокол испытаний, воспроизводимый отчёт	Условия измерения и возможность повторения

Официальный источник тоже может быть неполным или устаревшим. Ссылка подтверждает только то, что действительно сказано на странице, и только в пределах её области применения.

10.4. Ссылки в итоговом ответе

Ставьте ссылку рядом с тезисом, который она подтверждает. Не собирайте все ссылки в конце, если читателю трудно понять, какая из них относится к конкретному утверждению. Для каждой значимой

С LLM

цифры желательно сохранить источник, дату и методику; для расчётной величины – ещё и формулу или путь вычисления.

Если система автоматически синтезирует ответ из сайтов, пользователь может не переходить к первоисточнику. Поэтому наличие цитаты в AI-интерфейсе не освобождает от проверки источника, если вывод важен для решения.

11. Операционный протокол агента

Инженерный протокол переводит общие принципы в конкретные правила поведения системы. Он не описывает автоматически внутреннюю архитектуру любой модели: например, нельзя утверждать, что все современные LLM построены как Mixture-of-Experts или используют одинаковый механизм маршрутизации экспертов.

Для рабочих систем полезнее задавать наблюдаемое поведение:

- если обязательных данных нет – не продолжать расчёт, а сообщить, чего не хватает;
- если требование неоднозначно – задать вопрос или явно показать варианты трактовки;
- если вызов инструмента завершился ошибкой – не скрывать ошибку и не считать действие выполненным;
- если обнаружено противоречие – сохранить обе версии и применить только заданное правило приоритета;
- если действие влияет на внешнюю систему – показать точные параметры и запросить разрешение в предусмотренных случаях;
- если задача сложная – сначала составить план и согласовать существенные допущения.

11.1. Цикл Plan – Confirm – Execute – Verify – Report

1. Plan: определить цель, входные данные и последовательность действий.
2. Confirm: проверить, что план соответствует задаче и не содержит недопустимых действий.
3. Execute: выполнить разрешённые шаги и сохранить результаты инструментов.
4. Verify: проверить полноту и корректность результата, включая выходы инструментов.
5. Report: сообщить итог, неопределённости, ошибки и то, что осталось невыполненным.

Этап проверки нельзя пропускать. Агент может правильно понять запрос, составить подходящий план и технически успешно вызвать инструмент, но получить неверные данные или неправильно их истолковать. Успешное выполнение вызова не равно успешному решению задачи.

В качестве примера инженерного протокола разработки агентных систем можно использовать файл `x.AGENTS.md`. Он является примером набора рабочих правил, а не описанием универсального устройства языковых моделей.

12. Обучение по предпочтениям и отказы

RLHF (обучение с подкреплением на основе обратной связи людей), DPO и другие методы последующего обучения влияют на то, как модель следует инструкциям, формулирует ответы, выражает неопределённость и выбирает между вариантами поведения. Они не позволяют свести все отказы к одному изолированному механизму.

Поведение системы может зависеть от нескольких уровней: базовых параметров модели, последующего обучения, системных инструкций, инструментов, классификаторов, фильтров и логики приложения. Поэтому отказ, который видит пользователь, не обязательно вызван только одним механизмом внутри весов модели. Часть правил может задаваться внешней системой, а часть поведения – быть результатом обучения. Подход RLHF к настройке поведения описан в работе *Training language models to follow instructions with human feedback*, а DPO – в *Direct Preference Optimization*. Эти методы различаются процедурой оптимизации; ни один из них не гарантирует безошибочный ответ.

Практический вывод: при анализе отказа нужно выяснять, на каком уровне он возник, а не предполагать, что существует одна команда, способная убрать или полностью объяснить его. Для прикладной системы отдельно документируют ограничения модели, правила приложения, доступные инструменты и условия эскалации.

13. Безопасность и недоверенные данные

Загруженные документы, веб-страницы, письма и результаты поиска могут содержать текст, похожий на инструкции: «игнорируй предыдущие правила», «раскрой секреты», «выполни эту команду». Такой текст нужно рассматривать как содержимое источника, а не автоматически исполняемую инструкцию. Внешний документ не должен сам менять разрешения или правила системы.

Для снижения риска:

- отделяйте системные правила, запрос пользователя и недоверенные документы явными границами;
- не передавайте внешней модели пароли, ключи API и лишние персональные данные без разрешённого основания и подходящей среды;
- выдавайте минимум прав, необходимый для задачи;
- не давайте инструменту чтения право записи или удаления без отдельной причины;
- не исполняйте код, SQL или команды терминала только потому, что модель их сгенерировала;
- проверяйте результаты внешних API и MCP-серверов как потенциально неполные, устаревшие или ошибочные;
- предусмотрите тайм-ауты, ограниченные повторы, обработку частичного выполнения, журналирование и откат.

Безопасность не обеспечивается одним предупреждением в промпте. Она требует архитектурных ограничений, разрешений, проверки параметров и подтверждения рискованных действий.

14. Когда нужны API и агент

14.1. Достаточно ли обычного чата

Чат обычно подходит для разового анализа, когда пользователь сам передаёт контекст, проверяет ответ и управляет следующим шагом. API становится полезен, если один и тот же процесс нужно повторять: загружать данные, вызывать модель, проверять формат, передавать результат дальше, вести журнал и обрабатывать много объектов одинаковым способом.

Задача	Обычно достаточно	API или агент нужен, когда
Разобрать один документ	Чат с чёткой инструкцией и ручной проверкой	Одинаково обрабатывать поток
Посчитать показатели маршрутизации и	Таблица, скрипт или калькулятор; LLM – для интерпретации	Нужны автоматическая журналирование
Найти актуальные факты	Поиск с проверкой источников	Нужен повторяемый контролируемый по правилам
Изменять записи в права, системе	Ручная операция с предпросмотром	Есть безопасная автоматизация, аудит и подтверждение
Выполнять несколько шаг зависимых действий	Последовательные ручные шаги	Процесс повторяется и следующий зависит от результата предыдущего

14.2. Агент – не длинный промпт

Агентная система включает управляющий цикл, в котором модель выбирает следующие действия, вызывает инструменты, получает результаты и продолжает работу. Надёжность зависит не только от модели, но и от оркестрации, состояния задачи, прав, валидации и условий остановки. Если

последовательность заранее известна и не требует выбора следующего шага, обычный скрипт с несколькими вызовами модели может быть проще и предсказуемее.

Пример потока: пользователь загружает маршрутную карту → программа проверяет тип и версию → модель извлекает операции и ссылки на строки → схема валидирует поля → скрипт пересчитывает длительности → контроль ищет пропуски и дубли → модель объясняет расхождения → человек подтверждает итог. Извлечение и объяснение можно поручить модели; контроль целостности и арифметику лучше вынести в проверяемые средства.

15. Оценка качества на собственных задачах

Заявление «модель стала лучше» мало полезно без измерений. Нужен небольшой эталонный набор задач, где заранее известны правильные ответы или критерии оценки. Все варианты следует сравнивать на одном и том же наборе.

15.1. Как составить тестовый набор

- Включите обычные и сложные случаи, пропуски, противоречия, неверные единицы и неподходящие версии.
- Зафиксируйте правильные ответы или критерии до испытания модели.
- Для извлечения измеряйте точность и полноту каждого поля, а не только качество общего текста.
- Для расчётов сравнивайте числа с независимым эталоном и отдельно учитывайте ошибки единиц.
- Для поиска проверяйте, подтверждает ли источник конкретный тезис и актуален ли он.
- Для tools фиксируйте выбор функции, аргументы, статус исполнения, интерпретацию и ошибочные действия.
- Для рекомендаций задайте критерии предметной области и предусмотрите ручной разбор пограничных случаев.

15.2. Метрики, которые нужно разделять

Метрика	Что показывает	Чего не показывает
Точность извлечения	Доля полученных значений, совпавших с эталоном	Сколько нужных значений пропущено
Полнота извлечения значений	Доля необходимых значений, которые система получила	Правильны ли все полученные
Доля отказов и эскалаций	Как часто система сообщает о нехватке данных или передаёт задачу человеку	Высокая доля может означать осторожность или плохое покрытие
Доля критических ошибок	Частота ошибок с риском ущерба или неверного действия	Редкие ошибки могут не попасть в маленький набор
Корректность tool calling	Выбор функции, аргументы, обработка статусов	Правильны ли внешний источник и итоговый вывод
Стоимость и задержка	Ресурсы и время на успешную задачу	Не должна оптимизироваться ценой незаметного роста ошибок

Фиксируйте версии модели, промпта, документов, схем и кода. Иначе нельзя понять, какая именно переменная повлияла на результат. Проверяйте не только средние показатели, но и критические ошибки, которые встречаются редко.

16. Готовые рабочие процедуры

16.1. Анализ документа

1. Зафиксировать имя файла, редакцию, дату, число страниц и приложения.
2. Получить структуру документа и список доступных источников, не переходя сразу к выводам.

С LLM

3. Составить таблицу: требование или утверждение, точная цитата, страница, интерпретация, неопределённость.
4. Проверить наличие всех страниц, таблиц, сносок, примечаний и приложений.
5. Сформировать вывод только на основе подтверждённых требований; вопросы вынести отдельно.
6. Проверить, что каждая существенная рекомендация связана с конкретным фрагментом или явно обозначена как предположение.

16.2. Исследование темы в интернете

1. Составить перечень вопросов и фактов, которые нужно установить.
2. Найти первичный источник для каждого ключевого тезиса; вторичные материалы использовать для контекста и поиска первички.
3. Записывать дату публикации и дату события отдельно.
4. Проверить методику, область применимости, определения и актуальную редакцию.
5. Найти потенциальный контраргумент или независимое подтверждение спорного тезиса.
6. Поставить ссылку рядом с тезисом. Неподтверждённое убрать или пометить.

16.3. Извлечение данных из файлов в JSON

1. Задать названия полей, типы и обязательность до запуска.
2. Для каждого значения сохранять `source_file` и `source_location`, если это возможно.
3. Отличать `null` от нуля, пустой строки и «не применимо».
4. При неуверенности возвращать статус `review_required`, не заполнять пропуски догадками.
5. Машинно проверить разбор JSON, обязательные поля, допустимые значения и отсутствие лишних полей.
6. Сверить извлечённую выборку с оригиналом; критичные значения проверять полностью или независимо.

16.4. Инструментальный процесс

1. Описать разрешённые функции и пределы доступа.
2. Перед запуском проверить имя инструмента, типы, обязательные поля, ID объекта, период и единицы.
3. Разделить операции чтения и изменения состояния.
4. Записывать вызов, результат валидации, статус инструмента и полученные данные.
5. Проверять результат по контракту инструмента и независимому правилу.
6. Перед необратимым действием показать человеку точную операцию, объект и параметры и получить подтверждение.

16.5. Подготовка решения для руководителя

1. Разделить факты, расчёты, гипотезы, риски и рекомендации.
2. Для каждой цифры указать источник, дату и формулу, если это расчёт.
3. Показать ключевую неопределённость и то, как она может повлиять на решение.
4. Проверить, не выбраны ли только аргументы в пользу заранее желаемого варианта.
5. Указать, что нужно проверить до исполнения и кто уполномочен подтвердить решение.

17. Чек-листы и точки остановки

Перед запросом

- Понятно, какой результат нужен и по каким критериям он будет принят.
- Для критичных данных известны источник, дата и версия.
- Единицы измерения, пропуски и противоречия обозначены.
- Передаются только необходимые данные; лишние секреты исключены.

- Задача разбита на этапы там, где это снижает риск, но зависимости сохранены.

Перед принятием ответа

- У ключевых фактов есть проверяемые основания.
- Числа, даты, единицы и ID сверены.
- Ссылки открыты и подтверждают именно тот тезис, рядом с которым стоят.
- Неопределённость и противоречия названы.
- Проверены альтернативное объяснение и возможная сикофантия.
- Факты, расчёты, интерпретации и рекомендации разделены.

Перед вызовом инструмента или действием агента

- Выбрана подходящая функция.
- Аргументы прошли структурную и бизнес-валидацию.
- Права доступа достаточны, но не шире необходимого.
- Понятно, что именно изменится и можно ли отменить действие.
- Ошибки, тайм-ауты и частичные результаты обрабатываются явно.
- Рискованное действие требует подтверждения человека.

Безусловные условия остановки

Если...	То...
Нет обязательных данных или неизвестно критичное значение	Запросить данные или вернуть статус «недостаточно данных»
Источники противоречат друг другу и правила приоритета нет	Зафиксировать противоречие и получить решение ответственного
Структура, типы или диапазоны не прошли валидацию	Не использовать результат; вернуть конкретную ошибку на исправление
Результат инструмента устарел, неполон или относится к другому объекту	Повторить корректный запрос или остановить процесс
Вывод влияет на безопасность, право, финансы, производство или персональные данные	Получить подтверждение уполномоченного человека до действия
Предлагается необратимое действие без ясного основания	Не выполнять автоматически; запросить предпросмотр и подтверждение

Детализировать ошибку нужно ровно настолько, чтобы обнаружить её, установить причину и исправить до следующего шага.

18. Границы применимости

Из рассмотренных принципов нельзя делать следующие выводы:

- Большой контекст равен хорошей памяти. Большой объём входа не гарантирует равномерного и корректного использования информации.
- Длинный промпт лучше короткого. Важны релевантность, структура и отсутствие противоречивых требований.
- Любой запрет автоматически вызывает прайминг запрещённого действия. Это не универсально установленный закон для языковых моделей.

- Несколько одинаковых ответов доказывают истину. Они могут зависеть от одних и тех же неверных данных.
- Уверенный тон означает высокую вероятность правды. Убедительность и фактическая правильность могут расходиться.
- Веб-поиск гарантирует истинность. Ошибка может быть в источнике, выборе фрагмента или интерпретации.
- Много ссылок означает хорошее исследование. Важнее, подтверждает ли каждый источник конкретный тезис.
- Любой tool call означает корректное действие. Ошибка возможна при выборе инструмента, формировании параметров, исполнении, получении данных и интерпретации.
- Строгая JSON-схема доказывает корректность смысла. Она проверяет структуру и типы, но не все бизнес-условия.
- Все современные LLM построены как Mixture-of-Experts. Это один из архитектурных подходов, а не универсальная характеристика.
- Любой отказ находится только внутри весов модели. На поведение могут влиять инструкции, фильтры, инструменты и внешняя логика приложения.
- Чем больше инструментов, тем агент полезнее. Большой набор функций усложняет выбор, увеличивает контекстную нагрузку и расширяет поверхность риска.
- Модель может сама надёжно подтвердить любой свой вывод. Повторный проход полезен как дополнительный фильтр, но не заменяет независимый контроль.

Для качественной работы с LLM нужно управлять исходными данными, контекстом, проверками и доступными действиями. Ошибка может возникнуть на любом этапе; задача процесса – сделать её обнаружимой и не дать ей незаметно перейти в отчёт, решение или действие.

19. Источники

Научные работы описывают результаты конкретных экспериментов; их нельзя автоматически переносить на любую модель и задачу. Документация API меняется, поэтому перед внедрением следует проверять актуальные требования используемого сервиса.

- Andrej Karpathy – LLM wiki.
- Simon Willison – Context engineering.
- Nelson F. Liu et al. – Lost in the Middle: How Language Models Use Long Contexts, 2023.
- Shunyu Yao et al. – Tree of Thoughts: Deliberate Problem Solving with Large Language Models, 2023.
- Xuezhi Wang et al. – Self-Consistency Improves Chain of Thought Reasoning in Language Models, 2022.
- OpenAI – Why language models hallucinate, 2025.
- Mrinank Sharma et al. – Towards Understanding Sycophancy in Language Models, 2023.
- OpenAI – Prompt engineering.
- Anthropic – Prompt engineering overview.
- Anthropic – Tool use with Claude.
- OpenAI – Function calling.
- OpenAI Agents SDK – Tools.
- Model Context Protocol – Introduction.
- OpenAI – BrowseComp: a benchmark for browsing agents, 2025.
- OpenAI – WebGPT: Browser-assisted question-answering with human feedback.
- Long Ouyang et al. – Training language models to follow instructions with human feedback, 2022.
- Rafael Rafailov et al. – Direct Preference Optimization: Your Language Model Is Secretly a Reward Model, 2023.
- x.AGENTS.md – example of an agent-development protocol.