ЛЕКЦИЯ 13
PROMPT ENGINEERING: МЕТОДЫ ПРОЕКТИРОВАНИЯ ЭФФЕКТИВНЫХ ЗАПРОСОВ

Цель лекции

Изучить основные принципы Prompt Engineering и способы управления результатами больших языковых моделей с помощью инструкций, контекста и примеров; рассмотреть методы Zero-Shot, One-Shot, Few-Shot и структурированного рассуждения, практические шаблоны промптов, критерии оценки качества, типичные ошибки и требования безопасного использования промптов.

Понятие Prompt Engineering

Prompt — запрос, инструкция или совокупность входных данных, передаваемых модели искусственного интеллекта. Prompt Engineering — процесс проектирования, проверки и улучшения таких запросов для получения результата, соответствующего поставленной задаче.

В загруженном учебном пособии тема Prompt Engineering включает принципы создания промптов, Zero-Shot, Few-Shot, Chain-of-Thought и практические кейсы. В силлабусе также предусмотрено освоение принципов Prompt Engineering и эффективного взаимодействия с современными моделями искусственного интеллекта.

Качественный промпт не изменяет параметры модели. Он организует информацию в контексте и уточняет, что именно требуется сделать, с какими данными, при каких ограничениях и в каком формате.

Почему формулировка запроса влияет на результат

Большая языковая модель генерирует ответ на основе доступного контекста. Если задача сформулирована слишком широко, модель вынуждена самостоятельно предполагать недостающие условия. Это повышает вероятность нерелевантного, слишком общего или неправильно оформленного ответа.

Сравним два запроса. Первый: «Проанализируй данные». Второй: «Проанализируй таблицу результатов классификации, сравни precision, recall и F1-score для трёх моделей и сформулируй вывод в пяти предложениях без добавления данных, отсутствующих в таблице». Во втором случае задача, объект анализа, критерии и формат определены значительно точнее.

Основные компоненты эффективного промпта

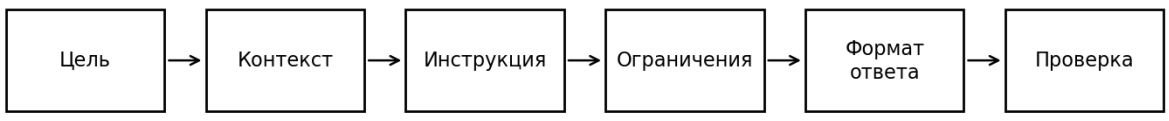


Рисунок 1 – Основные компоненты эффективного промпта

Компонент	Назначение	Пример
Цель	Определяет требуемый результат	Сравнить две модели

Контекст	Сообщает необходимые сведения	Данные получены на одной тестовой выборке
Инструкция	Задаёт конкретное действие	Сравни precision, recall и F1-score
Ограничения	Определяют границы ответа	Не добавляй отсутствующие факты
Формат	Задаёт структуру результата	Таблица + краткий вывод
Критерий проверки	Уточняет требования к качеству	Все числа должны совпадать с исходными данными

Конкретность и однозначность

Промпт должен уменьшать неопределённость задачи. Формулировки «сделай хорошо», «проанализируй подробно» или «напиши правильно» не задают измеримого результата. Лучше определить объём, аудиторию, структуру, критерии и ограничения.

Например, вместо «Объясни нейронные сети» можно написать: «Объясни принцип работы многослойного персептрона магистрантам первого года обучения. Используй определения входного, скрытого и выходного слоёв, приведите один числовой пример и завершите сравнением MLP и логистической регрессии».

Контекст в промпте

Контекст — информация, необходимая модели для выполнения задачи. Это может быть исходный текст, таблица, описание предметной области, требования к документу или ранее принятые обозначения.

Если ответ должен основываться на конкретном документе, целесообразно явно ограничить источник: «Используй только предоставленный фрагмент. Если ответа в нём нет, укажи, что информации недостаточно». Такое ограничение не гарантирует отсутствие ошибки, но делает задачу более определённой.

Роль и аудитория

В промпте можно указать профессиональный контекст или целевую аудиторию. Это помогает выбрать терминологию, глубину объяснения и форму ответа.

Аудитория	**Как изменить инструкцию**
Школьник	Использовать простые определения и бытовой пример
Магистрант	Использовать профессиональную терминологию и сравнение методов
Исследователь	Указать допущения, метрики, ограничения и экспериментальные условия
Руководитель	Сделать краткий вывод с акцентом на решение и риски

Указание роли не даёт модели дополнительных полномочий или гарантированной экспертности. Это прежде всего способ задать стиль и перспективу ответа.

Формат результата

Формат лучше задавать явно, если результат должен использоваться дальше. Например: таблица с фиксированными столбцами, список из пяти пунктов, JSON-подобная структура, аннотация определённого объёма или последовательность разделов.

Чем важнее машинная обработка результата, тем строже должна быть спецификация структуры. Однако после генерации структурированный ответ всё равно необходимо валидировать.

Zero-Shot prompting

Zero-Shot — постановка задачи без демонстрационных примеров. Модель получает только инструкцию и входные данные.

Пример: «Определи тональность каждого отзыва как положительную, нейтральную или отрицательную. Верни таблицу: №, тональность, краткое обоснование».

Zero-Shot удобен, когда задача хорошо известна модели и требования легко описываются словами. Его преимущество — краткость контекста.

One-Shot и Few-Shot prompting

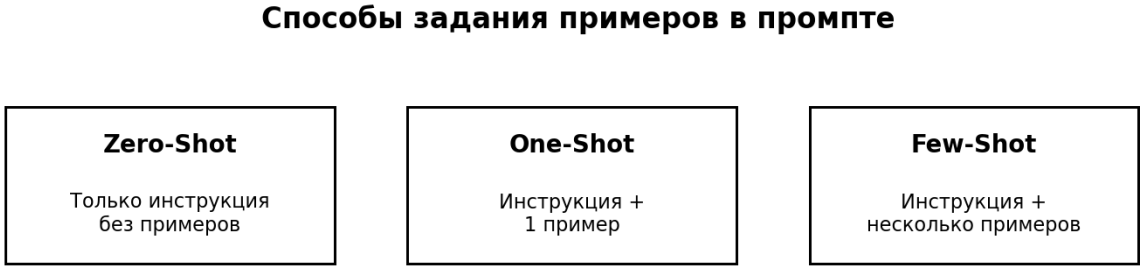


Рисунок 2 – Zero-Shot, One-Shot и Few-Shot prompting

One-Shot включает один демонстрационный пример, Few-Shot — несколько примеров. Примеры показывают модели ожидаемое соответствие между входом и выходом.

Например, при классификации обращений можно показать несколько корректно размеченных случаев: «Не могу войти в аккаунт → доступ», «Списали неизвестную сумму → платёж», после чего передать новые обращения для классификации.

Подход	**Примеры в контексте**	**Когда использовать**
Zero-Shot	Нет	Инструкция достаточна сама по себе
One-Shot	Один	Нужно показать формат ответа
Few-Shot	Несколько	Нужно продемонстрировать классы, стиль или правила

Как выбирать примеры для Few-Shot

Примеры должны быть корректными, репрезентативными и согласованными. Если демонстрационные ответы противоречат друг другу, модель получает неоднозначный сигнал.

Полезно включать примеры, отражающие типичные случаи и важные границы между классами. При этом не следует заполнять контекст десятками почти одинаковых примеров, если несколько качественных демонстраций уже задают правило.

Структурированное рассуждение

Для сложной задачи полезно декомпозировать работу на этапы: определить входные данные, выделить критерии, выполнить сравнение, проверить ограничения и только затем сформировать итоговый ответ. Такой подход уменьшает риск пропуска важных частей задачи.

В учебном пособии отдельно указан Chain-of-Thought как один из методов Prompt Engineering. В практической работе безопаснее формулировать требование к проверяемым промежуточным результатам: например, попросить модель перечислить использованные

критерии, показать расчёт по заданной формуле или представить краткое обоснование вывода, не требуя раскрытия внутренних скрытых рассуждений.

Декомпозиция сложной задачи

Если запрос содержит несколько независимых действий, их можно разделить на последовательные этапы. Например, анализ научной статьи можно организовать так: сначала определить цель исследования, затем извлечь данные и методы, далее выписать метрики, после этого сравнить заявленные результаты и в конце сформулировать ограничения.

Декомпозиция особенно полезна при работе с длинными документами, программным кодом, аналитическими отчётами и задачами, где результат одного этапа используется на следующем.

Шаблон промпта для анализа документа

Задача: проанализировать предоставленный документ.

Контекст: использовать только содержание документа.

Инструкция: выделить цель, методы, данные, результаты и ограничения.

Ограничение: не добавлять сведения, которых нет в документе; при отсутствии информации писать «не указано».

Формат: таблица из пяти строк, затем вывод до 100 слов.

Проверка: числа, названия методов и выводы должны соответствовать исходному тексту.

Такой шаблон не является универсальной формулой, но показывает, как соединить цель, контекст, ограничения и формат.

Шаблон промпта для сравнения моделей

Пример задания: «На основе предоставленной таблицы сравни Logistic Regression, Random Forest и нейронную сеть по accuracy, precision, recall, F1-score и времени обучения. Не пересчитывай отсутствующие показатели. Сначала сформируй сравнительную таблицу, затем укажи сильные и слабые стороны каждой модели и предложи выбор для системы, где критичен пропуск положительного класса».

В этом примере явно задан критерий принятия решения. Если критичен пропуск положительного класса, внимание должно быть уделено recall, а не только общей accuracy.

Шаблон промпта для программирования

При запросе программного кода следует задавать язык, версию или среду при необходимости, входные и выходные данные, ограничения и критерии проверки.

Пример: «Напиши на Python функцию, которая принимает pandas DataFrame, находит числовые столбцы с пропусками и заполняет их медианой. Не изменяй категориальные столбцы. Добавь краткий пример вызова и три тестовых случая».

Сгенерированный код необходимо запускать и тестировать. Корректно выглядящий код может содержать логические ошибки или небезопасные конструкции.

Итеративная разработка промпта



Рисунок 3 – Итеративный процесс совершенствования промпта

Prompt Engineering — итеративный процесс. Сначала формируется базовая инструкция, затем оценивается результат, выявляются ошибки и уточняется запрос.

Если модель постоянно пропускает определённый раздел, его следует явно включить в формат. Если добавляет неподтверждённые факты — усилить ограничение на источники и предусмотреть проверку. Если формат нестабилен — привести образец.

Критерии оценки промпта

Критерий	**Вопрос для проверки**
Точность	Получен ли требуемый результат без фактических искажений?
Релевантность	Отвечает ли модель именно на поставленную задачу?
Полнота	Все ли обязательные элементы присутствуют?
Формат	Соблюдена ли требуемая структура?
Устойчивость	Сохраняется ли качество на разных примерах?
Безопасность	Не приводит ли запрос к раскрытию или неправильной обработке данных?

A/B-сравнение промптов

Два варианта промпта можно проверить на одном наборе задач и сравнить по заранее определённым критериям. Например, для извлечения реквизитов из документов измеряется доля правильно извлечённых полей, число пропусков и число добавленных моделью значений.

Такой подход лучше субъективной оценки одного удачного ответа. Для учебного эксперимента можно подготовить 20–30 тестовых примеров и сравнить Zero-Shot и Few-Shot варианты.

Prompt Engineering и температура генерации

Качество результата зависит не только от текста промпта, но и от параметров генерации модели. Более высокая вариативность может быть полезна при поиске идей, а для извлечения структурированной информации обычно предпочтительнее более стабильное поведение.

При экспериментальном сравнении промптов важно фиксировать условия запуска, иначе различия могут быть вызваны не промптом, а параметрами генерации.

Prompt Engineering и внешние документы

Если модель должна отвечать по документу, в контекст необходимо передать релевантный фрагмент и определить правила использования источника. При больших коллекциях документов предварительно выполняется поиск нужных фрагментов, после чего они включаются в контекст модели.

В такой системе качество ответа зависит от двух частей: насколько хорошо найдена релевантная информация и насколько корректно модель использовала её при генерации.

Типичные ошибки при создании промптов

Ошибка	Последствие	Как улучшить
Слишком общий запрос	Общий и непредсказуемый ответ	Уточнить задачу и критерии
Нет контекста	Модель делает лишние предположения	Передать необходимые данные
Не задан формат	Структура меняется от ответа к ответу	Указать разделы или шаблон
Противоречивые требования	Модель выбирает часть условий	Устранить конфликт
Плохие Few-Shot примеры	Модель воспроизводит ошибки	Проверить демонстрации
Слишком много лишнего текста	Важные инструкции теряются	Сократить нерелевантный контекст
Нет проверки результата	Ошибка попадает в итоговый документ	Добавить процедуру верификации

Промпты для информационной безопасности

В информационной безопасности Prompt Engineering может применяться для структурирования разрешённых описаний инцидентов, классификации текстовых обращений, суммаризации отчётов и объяснения результатов аналитических инструментов.

Например: «На основе приведённого описания инцидента выдели актив, событие, наблюдаемые индикаторы и неопределённые сведения. Не придумывай отсутствующие IP-адреса, даты или технические факты. Верни результат в таблице».

Такой промпт помогает структурировать информацию, но не заменяет анализ журналов и средств мониторинга.

Конфиденциальные данные в промптах

Промпт может содержать не только инструкцию, но и реальные данные. Поэтому перед передачей информации внешней модели необходимо проверить допустимость её обработки.

Пароли, ключи доступа, персональные данные, закрытый исходный код, внутренние документы и журналы безопасности не следует передавать внешнему сервису без соответствующего разрешения, договорных и технических мер защиты.

Prompt Injection

При использовании LLM в системах, работающих с внешними документами или веб-контентом, необходимо учитывать риск Prompt Injection. Внешний текст может содержать инструкции, пытающиеся изменить поведение модели и заставить её игнорировать исходную задачу.

Защита не должна основываться только на фразе «игнорируй вредоносные инструкции». Необходимо разделять доверенные системные инструкции и недоверенные данные, ограничивать доступ модели к инструментам и данным, проверять выход и применять принцип минимальных привилегий.

Системные и пользовательские инструкции

В прикладной LLM-системе могут существовать инструкции разных уровней: правила приложения, пользовательская задача и внешние данные. Их нельзя смешивать как равнозначный текст.

Например, содержимое загруженного документа должно рассматриваться как данные для анализа, а не как инструкция, автоматически изменяющая правила работы системы.

Промпты и галлюцинации

Улучшение промпта может уменьшить часть ошибок, но не устраняет способность LLM генерировать неподтверждённые сведения. Поэтому запрос «не ошибайся» не является механизмом гарантии.

Для фактологических задач полезно требовать опору на предоставленный источник, маркировать неизвестные сведения и проверять критичные факты после генерации.

Prompt Engineering в научной работе

В научной работе промпты могут использоваться для структурирования собственных материалов, редактирования формулировок, подготовки вариантов классификации или помощи с программным кодом.

При этом модель не должна быть источником вымышленных публикаций, экспериментальных данных или неподтверждённых выводов. Библиографические записи необходимо проверять по реальным источникам и оформлять по установленному стандарту.

Prompt Engineering в образовании

В образовательной задаче полезны промпты, которые не просто выдают готовый ответ, а организуют объяснение, сравнение и самопроверку. Например: «Объясни различие precision и recall, затем предложи две небольшие задачи без ответов, после моего решения проверь вычисления».

Для преподавателя Prompt Engineering может использоваться при подготовке черновиков вопросов, кейсов и примеров, но содержание должно быть проверено перед включением в учебно-методические материалы.

Практический кейс: Zero-Shot и Few-Shot

Задача — классифицировать обращения пользователей по категориям «доступ», «платёж», «техническая ошибка». В Zero-Shot варианте модель получает определения трёх классов и новые обращения. В Few-Shot варианте дополнительно приводятся несколько корректно размеченных примеров.

Эксперимент следует проводить на одинаковом тестовом наборе. Затем сравнивается число правильных классификаций и анализируются ошибки. Это позволяет эмпирически определить, дают ли примеры преимущество для конкретной задачи.

Практический кейс: извлечение информации

Пусть требуется извлечь из текста документа номер, дату, организацию и сумму. Промпт должен перечислять поля, определить поведение при отсутствии значения и запретить угадывание.

Пример требования: «Если поле отсутствует, верни null. Не восстанавливай значение по контексту и не используй внешние знания». После генерации результат сравнивается с эталонной разметкой.

Практический кейс: сравнение двух ответов

Магистрантам можно дать один и тот же исходный текст и два промпта: короткий общий и структурированный. После получения ответов они оценивают полноту, точность, соблюдение формата и количество неподтверждённых утверждений.

Такое упражнение показывает, что Prompt Engineering следует рассматривать не как набор «магических фраз», а как инженерный процесс постановки задачи и экспериментальной проверки.

Рекомендуемая последовательность проектирования промпта

- Определить конечную задачу и критерий успешного результата.
- Выделить минимально необходимый контекст.
- Сформулировать однозначную инструкцию.
- Указать ограничения и допустимые источники.
- Задать формат ответа.
- Решить, нужны ли Zero-Shot или Few-Shot примеры.
- Проверить промпт на наборе тестовых случаев.
- Проанализировать ошибки и уточнить инструкцию.
- Проверить безопасность данных и итогового результата.

Основные выводы

Prompt Engineering — инженерный процесс проектирования, тестирования и улучшения запросов к моделям ИИ.

Эффективный промпт обычно определяет цель, контекст, инструкцию, ограничения и формат результата.

Zero-Shot работает без примеров, One-Shot использует один пример, Few-Shot — несколько демонстраций.

Сложные задачи полезно декомпозировать на проверяемые этапы.

Качество промпта следует оценивать на наборе тестовых примеров, а не по одному удачному ответу.

Prompt Engineering не устраняет галлюцинации и не заменяет проверку фактов.

При работе с промптами необходимо учитывать конфиденциальность, Prompt Injection и разделение доверенных инструкций и внешних данных.

Контрольные вопросы

1. Что такое Prompt Engineering и почему формулировка запроса влияет на результат LLM?
2. Какие компоненты целесообразно включать в структурированный промпт?
3. Чем отличаются Zero-Shot, One-Shot и Few-Shot prompting и в каких случаях они применяются?
4. Как следует подбирать демонстрационные примеры для Few-Shot prompting?
5. Как декомпозиция задачи и структурированные промежуточные результаты помогают при решении сложных задач?
6. Какие критерии можно использовать для экспериментальной оценки качества промпта?
7. Какие типичные ошибки возникают при проектировании промптов и как их исправлять?
8. Как Prompt Engineering применяется при анализе документов, программировании и задачах информационной безопасности?
9. Какие риски связаны с конфиденциальными данными и Prompt Injection?
10. Почему даже хорошо спроектированный промпт не отменяет необходимость проверки фактов и результатов модели?

Список источников

1. Бессмертный, И. А. Системы искусственного интеллекта : методическое пособие / И. А. Бессмертный, А. Е. Авдюшина, А. В. Кугаевских, Ю. А. Королева, Н. Г. Рущенко. – Санкт-Петербург : Университет ИТМО, 2024. – 281 с.

2. Соснило, А. И. Атлас искусственного интеллекта для бизнеса и власти : учебное пособие / А. И. Соснило. – Санкт-Петербург : Университет ИТМО, 2022. – 98 с.

3. Сидоркина, И. Г. Системы искусственного интеллекта : учебное пособие / И. Г. Сидоркина. – Москва : КНОРУС, 2018. – 248 с.

4. Баланов, А. Н. Искусственный интеллект. Понимание, применение и перспективы : учебник для вузов / А. Н. Баланов. – Санкт-Петербург : Лань, 2024. – 312 с.

5. Калинин, М. О. Основы искусственного интеллекта. Безопасность искусственного интеллекта : учебное пособие / М. О. Калинин, В. М. Крундышев. – Санкт-Петербург, 2024. – 220 с.

6. Загорулько, Ю. А. Искусственный интеллект. Инженерия знаний : учебное пособие для вузов / Ю. А. Загорулько, Г. Б. Загорулько. – Москва : Юрайт, 2019. – 184 с.

7. Андреянов, Н. В. Методы искусственного интеллекта : учебное пособие / Н. В. Андреянов, Т. С. Евдокимова, А. Д. Павлов [и др.]. – Казань : КНИТУ-КАИ, 2024. – 392 с.

8. Основы искусственного интеллекта : учебное пособие. – Гл. 13 : Prompt Engineering: принципы создания промптов, Zero-Shot, Few-Shot, Chain-of-Thought, практические кейсы.