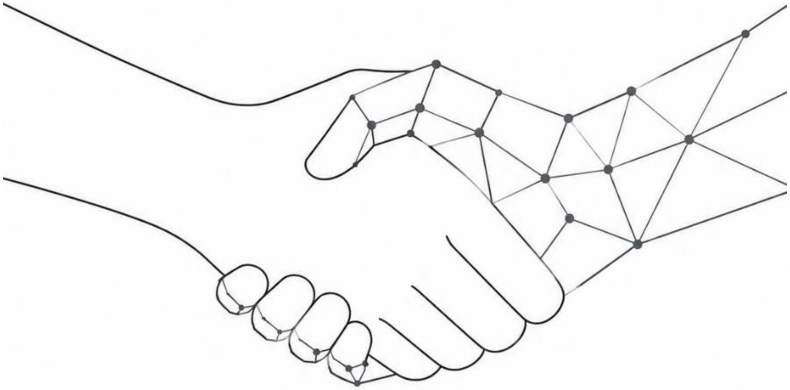

Алекс Промтов

Промтология: Как сделать нейросеть своим заместителем

https://litres.ru/73913759
SelfPub; 2026

Аннотация

Вы уже умеете задавать нейросети вопросы. Но вы всё ещё копируете ответы вручную и тратите часы на рутину. Эта книга — про автоматизацию: цепочки промтов, агентов, RAG по вашим документам, обучение моделей под себя, интеграции с API и контроль качества без вашего участия.

Вы построите конвейер, где нейросеть думает, действует, проверяет себя и масштабируется в команде. Продолжение бестселлера «Промтология. Искусство задавать вопросы». Сделайте ИИ своим заместителем.

Алекс Промтов
Промтология: Как сделать нейросеть своим заместителем

«От вопросов — к системе. Делегируйте, автоматизируйте, управляйте»

Вступление

От молотка к заводскому конвейеру

(почему одного хорошего промта больше недостаточно)

Помните, как всё начиналось?

Вы впервые открыли чат с нейросетью. Робко напечатали что-то вроде *«напиши стих про кота»*. Получили что-то забавное. Потом — *«помоги с заголовком для статьи»*. Потом — *«сделай таблицу»*. И вдруг обнаружили, что за несколько месяцев нейросеть стала вашим незаменимым помощником. Вы спрашиваете, она отвечает. Вы снова спрашиваете, она снова отвечает.

Это было круто. Это сэкономило вам сотни часов.

Но давайте честно посмотрим на то, как вы работаете с нейросетью *сейчас*.

Скорее всего, это выглядит так: вы сидите перед экраном, формулируете промт, копируете ответ, вставляете в документ или таблицу, переходите к следующей задаче, снова формулируете промт, снова копируете… И так по кругу, раз за разом, день за днём.

Вы всё ещё делаете **80% работы руками**. Нейросеть — лишь «генератор идей», но не полноценный сотрудник. Вы — тот самый руководитель, который нанял гениального ассистента, а потом бегает за ним с блокнотом и переписывает его слова от руки, потому что «так привычнее».

Первая книга «Промтология. Искусство задавать вопросы нейросетям» научила вас правильно говорить с нейросетью. Роль, контекст, ограничения, примеры — вы освоили базу. Теперь вы получаете хорошие ответы с первого раза.

Но база — это только молоток. А можно построить завод.

Эта книга — про то, чтобы перестать быть секретарём для нейросети. Про то, как превратить её из «генератора ответов» в **заместителя**, который делает всё сам: берёт данные, обрабатывает, проверяет, передаёт дальше, — пока вы пьёте кофе.

Моё собственное «прозрение» (и почему вторая книга — не просто добавленная глава)

После выхода первой «Промтологии» я думал, что достиг потолка. Я умел писать промты, которые давали идеальный результат с первого раза. Я мог заставить ChatGPT, YandexGPT или Claude делать всё, что угодно: от написания юридических писем до генерации кода. Мои читатели и подписчики восхищались: «Алексей, ты волшебник!»

Но внутри я чувствовал фальшь. Потому что каждый день я тратил **часы на рутину**. Например, мне нужно было проанализировать 200 отзывов на свой телеграм-канал. Я написал идеальный промт, нейросеть выдала анализ… одного отзыва. Потом я вручную скопировал следующий отзыв, отправил, получил анализ, скопировал… Через 40 минут такой работы у меня заболела спина, а прогресс был на 20 отзывах.

Я чувствовал себя идиотом. У меня есть супер-инструмент, а я использую его как дорогую печатную машинку.

Тогда я задал себе вопрос: **а что, если нейросеть может делать всю эту работу сама? Без моего копирования, вставки, запуска? Что, если я просто дам ей задачу «проанализируй 200 отзывов и выгрузи отчёт», и она это сделает, пока я пью кофе?**

Понадобилось два месяца экспериментов, куча ошибок и бессонных ночей, но я это сделал.

Я собрал конвейер из промтов, подключил нейросеть к

Google Sheets через несколько строчек скрипта, добавил автоматическую проверку качества. Итог: вместо 40 минут на 20 отзывов — 2 минуты на 200 отзывов. Нейросеть работала, а я наблюдал.

В тот вечер я понял: **первая книга учила вас говорить. Вторая — построить завод, где нейросеть работает на вас 24/7.**

Для кого эта книга (и что нужно знать перед прочтением)

Эта книга **НЕ для новичков**. Если вы никогда не писали промтов или только мельком видели нейросеть — откройте сначала первую книгу **«Промтология. Искусство задавать вопросы нейросетям»**. Там база: роли, контекст, семь смертных грехов, форматы ответов. Без этого вы не поймёте вторую часть.

Эта книга **ДЛЯ ТЕХ, КТО УЖЕ**:

Регулярно использует нейросети в работе (текст, код, аналитика, дизайн).

Чувствует, что много времени тратит на «копирование-вставку».

Хочет автоматизировать повторяющиеся задачи.

Управляет командой и хочет внедрить ИИ-ассистентов без хаоса.

Готов потратить время на настройку, чтобы потом экономить его в разы.

Техническая подготовка: минимальная. Я не требую знания программирования, но в главах про API и интеграции будут примеры кода (Python, JavaScript). Их можно пропустить без потери смысла, но для максимальной пользы рекомендую хотя бы понимать логику «переменных», «условий» и «циклов». Если вы никогда в жизни не видели строчку if x > 5: — не страшно, объясню на пальцах.

Что вы найдёте в этой книге (и что не найдёте)

ВЫ НАЙДЁТЕ:

Цепочки промтов (пайплайны) — как строить многошаговые процессы, где выход одного промта идёт на вход другому. Без вашего участия.

Промт как функцию — создание шаблонов с переменными, которые можно применять к тысячам разных входных данных.

Нейросетевых агентов — как запустить виртуального сотрудника, который сам планирует, ищет информацию, делает выводы, пишет отчёты.

RAG (база знаний) — подключение ваших внутренних документов к нейросети. Создание чат-бота по вашей продуктовой документации за час.

Fine-tuning — обучение модели вашему стилю и терминологии. Сделать нейросеть «копией себя» в письмах или постах.

Интеграции с API и без кода (Zapier, Make, Google Apps Script) — как заставить нейросеть работать в ваших экосистемах (таблицы, почта, мессенджеры).

Контроль качества в автоматическом режиме — чтобы нейросеть не галлюцинировала, когда вы спите.

Безопасность и масштабирование — как не слить данные, управлять доступом, логировать действия агентов.

10 готовых сквозных пайплайнов в приложении — копируй, вставляй, запускай.

ВЫ НЕ НАЙДЁТЕ:

Повторения основ первой книги (роли, контекст, грехи). Я предполагаю, что вы их знаете.

Углублённого машинного обучения или математики. Никакого градиентного спуска.

Обещаний «купи книгу — и нейросеть будет делать всё за вас». Нет, настройка потребует усилий. Но эти усилия окупаются стократно.

Как читать эту книгу, чтобы не утонуть в сложности

Книга построена по нарастающей: от простых цепочек промтов до сложных агентов и интеграций.

Режим «Листатель» (хочешь быстрых результатов):

Прочитай вступление и Главу 1 (цепочки промтов) — это 80% пользы для новичка в автоматизации.

Главу 2 (промт как функция) — даёт супер-силу над сотнями задач.

Пролистай Главу 5 (fine-tuning) и Главу 6 (интеграции), возьми готовые скрипты из приложения.

Режим «Инженер» (хочешь всё настроить под себя):

Читай последовательно, выполняя примеры.

В главе 3 (агенты) удели особое внимание «ограничениям и рискам» — агенты могут сильно навредить, если их не контролировать.

В главе 4 (RAG) потрать время на установку локального AnythingLLM — это лучший способ быстро получить своего чат-бота.

Режим «Команда» (ты руководитель):

Прочитай Главу 8 (безопасность и масштабирование) в первую очередь.

Затем создай прототип одного пайплайна из приложения и покажи команде.

После успеха — масштабируй.

Моя просьба к вам (прежде чем вы перевернёте страницу)

Эта книга — не просто набор техник. Это **смена мышления**.

Перестаньте думать о нейросети как о «болталке», которая отвечает на вопросы. Начните думать о ней как о **движке**, который можно встроить в любую повторяющуюся задачу. Каждый раз, когда вы ловите себя на том, что в пятый раз пишете похожий промт, копируете ответ, вставляете в отчёт — останавливайтесь. И спрашивайте: **«А можно ли это автоматизировать?»**.

Я написал эту книгу, чтобы вы перестали быть рабами промтов и стали их **архитекторами**.

Давайте строить конвейеры.

Глава 1

Цепочки промтов (промт-пайплайны)

Как заставить нейросеть работать без вашего участия

Давайте прямо сейчас проведём маленький эксперимент.

Откройте вашу любимую нейросеть. Напишите любой содержательный промт — например, *«напиши план контент-стратегии для маленького кафе»*. Получите ответ. Оцените его.

Теперь скажите честно: сколько времени вы потратили на копирование и вставку следующих действий?

Вы скопировали результат в рабочий документ.

Потом поняли, что не хватает таблицы с конкурентами, и написали **второй** промт: *«а теперь таблицу с тремя конкурентами по ценам и меню»*.

Получили таблицу, скопировали её в тот же документ.

Потом захотели список тем для постов в соцсетях — **третий** промт.

Потом перечитали первый ответ и заметили, что нейросеть забыла про десерты, — уточняющий **четвёртый** промт.

К концу часа вы имеете хороший, детальный план, но вы — **человек-оркестр**, который дирижировал каждым шагом.

А что, если бы нейросеть сама, без ваших копирований и дополнительных запросов, могла пройти все эти шаги? Сначала спросить вас о типе кафе, затем собрать данные о конкурентах, потом предложить рубрики, и всё это сложить в один готовый документ?

Это называется **цепочка промтов (prompt chain)** или **пайплайн (pipeline)**.

В этой главе мы разберём, как строить такие цепочки. Сначала самые простые, где вы всё ещё участвуете (но уже меньше). Потом — полностью автоматические, где вы только запускаете, а нейросеть делает всё сама.

1.1. Почему один промт — это всегда компромисс

В первой книге я учил вас писать «идеальный промт»: роль, контекст, инструкция, ограничения, примеры. Такой промт действительно решает одну конкретную задачу качественно. Но у него есть фундаментальное ограничение: **он делает только одно действие**.

Ни один, даже самый гениальный промт, не может одновременно:

Собрать данные из интернета (или из вашей таблицы).

Проанализировать их по трём разным методикам.

Сравнить результаты между собой.

Написать итоговый отчёт в нужном формате.

И проверить, нет ли галлюцинаций.

Даже если вы попытаетесь впихнуть всё это в один промт, нейросеть запутается. Она начнёт смешивать анализ с форматом, забудет часть требований, а в итоге выдаст нечто усреднённое и неглубокое.

Каждый промт — как один бригадир, который умеет делать одну операцию: копать, или мешать раствор, или класть кирпичи. Но чтобы построить дом, нужно скоординировать работу многих бригадиров.

Цепочка промтов — это конвейер, на котором каждая нейросеть (или один и тот же ИИ, но в несколько заходов) выполняет свою небольшую задачу, а результат передаётся

дальше.

1.2. Простейшая цепочка: «вопрос → уточнение → финал»

Начнём с самого простого — с цепочки, где вы всё ещё активный участник, но уже делите задачу на этапы.

Задача: нужно написать пост для LinkedIn о важности нейросетей в маркетинге.

Один промт (плохо): *«Напиши пост для LinkedIn о важности нейросетей в маркетинге»* — получим общий, безличный текст.

Цепочка из 3 промтов (хорошо):

Промт 1 (генерация идей):

«Ты — маркетолог-аналитик. Перечисли 5 ключевых преимуществ нейросетей в маркетинге для малого бизнеса. Каждое преимущество — коротко (5–7 слов). Формат: маркированный список.»

Промт 2 (выбор угла и заголовка на основе Промта 1):

*«Основываясь на этом списке из 5 преимуществ, выбери одно, которое лучше всего подходит для короткого по-

ста (400 знаков) в LinkedIn. Целевая аудитория — владельцы бизнеса, у которых нет времени на технические детали. Придумай заголовок-вопрос, который вовлекает. Выдай: заголовок и выбранное преимущество с кратким обоснованием (1 предложение).»*

Промт 3 (написание текста на основе выбора):

«Теперь напиши пост в LinkedIn. Заголовок: [из ответа Промта 2]. Основная мысль: [выбранное преимущество]. Структура: заголовок, 2 предложения расшифровки, один пример из жизни (выдуманный, но правдоподобный), призыв к комментариям. Тон — экспертный, но без пафоса. Длина 400–500 знаков. В конце добавь 3 хештега.»

Что произошло? Вы **разложили творческую задачу на три примитивные**. Нейросеть каждый раз фокусировалась на одном: сначала перечислить, потом выбрать, потом написать. Качество конечного поста будет выше, чем если бы вы просили всё сразу. И вы не тратили время на правки, потому что каждый шаг контролировали.

Это режим «ручное управление цепочкой» — вы запускаете промт, получаете ответ, копируете важную часть в следующий промт. Уже лучше, чем один супер-промт, но ещё не автоматизация.

1.3. Автоматическая цепочка: когда выход одного промта идёт на вход другому

Теперь шаг к настоящей автоматизации. Мы уберём «копирование-вставку». Вместо этого научим нейросеть **запоминать вывод предыдущего шага** и использовать его как часть контекста.

Способ 1: **Использовать диалог**

В одном окне чата (например, ChatGPT) система помнит всю историю. Вы пишете:

Вы: «Перечисли 5 преимуществ нейросетей в маркетинге. Список.»

Нейросеть: «1. Персонализация… 2. Экономия времени…»

Вы: «Теперь на основе этого списка выбери одно самое сильное преимущество для поста в LinkedIn и напиши заголовок.»

Нейросеть: (использует предыдущий список)

Вы: «Теперь напиши пост, используя выбранное преимущество и заголовок.»

Нейросеть помнит контекст, и вам не нужно копировать. Но **вы всё ещё нажимаете Enter каждый раз**. Это полу-

автомат.

Способ 2: **Параметризация и подстановка** (уже ближе к программному подходу)

Если вы работаете через API или в инструментах типа LangChain, вы можете записать цепочку так:

text

Промт_1: "Перечисли 5 преимуществ ИИ в маркетинге. Выдай как JSON-массив."

Промт_2: "Из предыдущего ответа возьми первый элемент массива. Напиши заголовок для LinkedIn на его основе."

Промт_3: "Используя заголовок из промта_2, напиши пост, где раскрывается это преимущество."

Инструмент автоматически подаёт выход Промта_1 на вход Промта_2 и так далее. Вы запускаете **один раз —** получаете готовый пост.

Простейший способ сделать такую автоматизацию без программирования — использовать **Zapier** (связка ChatGPT + Google Docs) или **Make**. Но даже в Excel можно настроить: ячейка с промтом подтягивает значение из другой ячейки.

Для нас, обычных пользователей, важно понять идею: **мы строим конвейер, а не пишем гигантский промт**.

1.4. Паттерны цепочек: последовательная, ветвя-

щаяся, циклическая

Существуют типовые архитектуры цепочек. Я опишу три самые полезные.

Паттерн 1: Последовательная цепочка (linear chain)

Это просто A → B → C → D. Выход первого — вход второго.

Пример:

Собрать данные (вымышленные) о продажах за месяц.

Найти в этих данных три главные тенденции.

Написать отчёт о тенденциях.

Перевести отчёт на английский.

Всё идёт одно за другим. Самый надёжный паттерн.

Паттерн 2: Ветвящаяся цепочка (branching)

В какой-то точке цепочка делится на две или более параллельных, а потом результаты объединяются.

Пример:

Нужно написать статью на основе опроса.

Вход: сырые ответы на опрос.

Ветка А: извлечь статистику (проценты, средние).

Ветка Б: выделить 3 самые яркие цитаты.

Ветка В: определить 2 главные проблемы аудитории.

Объединение: написать статью, где есть статистика, цитаты и проблемы.

Промт для объединения: *«Напиши статью, используя: статистику — [из ветки А], цитаты — [из Б], проблемы — [из В]».*

Ветвление позволяет обрабатывать разные аспекты независимо.

Паттерн 3: Циклическая / итерационная цепочка (loop / iterative)

Промт выполняется несколько раз, причём каждый раз использует результат предыдущего, пока не будет достигнуто условие.

Пример: генерация текста с саморедактированием.

Написать первый черновик.

Оценить черновик по шкале от 0 до 10 (критерии: ясность, полнота, стиль). Если оценка ниже 8 — переписать, учитывая слабые места.

Повторять до тех пор, пока оценка не станет ≥8, или не пройдёт 5 итераций.

Такой цикл можно сделать вручную, но для автоматизации нужно программирование (или специальные агенты). Однако понимание цикла полезно: вы можете сами повторять промт «улучши предыдущий ответ».

1.5. Реальный пример: автоматический анализ 200 отзывов (мой случай из жизни)

Вернёмся к истории, с которой началось вступление. Разберу по шагам пайплайн, который я построил для анализа отзывов подписчиков. Этот пайплайн до сих пор работает каждую неделю.

Исходные данные:

200 отзывов в Google Sheets (колонка A: текст отзыва, колонка B: дата, колонка C: оценка 1–5).

Что нужно на выходе:

Отчёт в Google Docs с разбивкой:

Общая статистика (средняя оценка, распределение по звёздам).

5 самых частых позитивных тем.

5 самых частых негативных тем.

3 рекомендации для улучшения.

Как я это сделал без кода (почти):

Шаг 0. Подготовил шаблон в Google Apps Script (JavaScript). Скрипт проходит по каждой строке, берёт текст отзыва и отправляет его в API ChatGPT (за это нужно платить, но копейки). Но можно и бесплатно, если делать вручную: скопировать 200 отзывов в текстовый файл.

Шаг 1 (один промт для всего списка, если влезает в контекст).

Современные нейросети имеют контекст до 128К токенов

— 200 коротких отзывов (средний 100 символов) влезают легко. Поэтому я сделал **один промт**, который обрабатывает ВСЕ отзывы разом. Но это частный случай — если бы отзывов было 2000, пришлось бы разбивать.

Вот тот самый промт (сокращённая версия):

*«Ты — аналитик по работе с клиентами. У меня есть 200 отзывов о моём телеграм-канале (каждый отзыв — одно предложение). Проанализируй их и выдай отчёт в формате маркированного списка по разделам:

Общая статистика: количество отзывов, средняя оценка (если оценка не указана — не придумывай), распределение по тону (позитивные, нейтральные, негативные).

5 самых частых позитивных тем (слово или короткая фраза).

5 самых частых негативных тем.

3 рекомендации для улучшения канала (конкретные, основанные на негативных темах).

Вот отзывы: [вставить 200 отзывов]»*

Нейросеть выдала идеальный отчёт за 20 секунд.

Но если бы отзывов было больше контекста, я бы сделал **циклическую цепочку**:

Разбил отзывы на блоки по 50.

Для каждого блока выполнил промт, извлекающий темы и статистику.

Потом объединил результаты блоков в финальном промте («сведи данные из 4 предварительных анализов в итоговый

отчёт»).

Именно такой пайплайн я и реализовал в скрипте. **Один запуск — и готово**.

1.6. Инструменты для построения цепочек (без кода и с минимум кода)

Вам не обязательно учиться программировать. Вот три способа собирать цепочки уже сегодня.

Ручная цепочка в одном диалоге (любой чат).

Плюс: бесплатно, просто. Минус: нужно нажимать Enter между шагами.

Рекомендую для начала: так вы прочувствуете логику.

Zapier / Make (бывший Integromat).

Создаёте сценарный редактор: «Когда появляется новая строка в Google Sheets → отправить текст в ChatGPT → ответ записать в соседнюю ячейку». Визуально, как конструктор.

Платно (но недорого для малых объёмов).

Google Apps Script (чуть-чуть кода).

Я написал простую функцию, которая проходит по стро-

кам таблицы, вызывает UrlFetchApp к OpenAI API, возвращает результат. Код из 20 строк.

Локальные инструменты: LangChain, Flowise, Dify.

Это для энтузиастов. Позволяют строить цепочки любой сложности, подключать базы знаний, агентов.

Мой совет: начните с ручного режима. Освоитесь — попробуйте Zapier для одной повторяющейся задачи. Как только поймёте, что экономите не часы, а дни, — тогда вкладывайте время в настройку автоматической цепочки через API.

1.7. Ошибки при построении цепочек (чтобы не наступать на мои грабли)

Ошибся ли я, когда строил свой первый пайплайн? О да. Вот три самые обидные.

Ошибка 1: Слишком мелкие шаги

Я разбивал задачу до уровня «напиши слово», потом «напиши пробел», потом «напиши следующее слово». Цепочка из 50 промтов, которая выполнялась 15 минут. Результат — тот же, что и двумя промтами. **Оптимальный размер шага**: шаг должен делать что-то нетривиальное, но не больше одной-двух логических операций.

Ошибка 2: Не передавать контекст между шагами

Если вы забыли скопировать важный вывод из предыдущего шага, следующий промт начинает с чистого листа. **Всегда проверяйте**, что нужная информация попала в следующий промт.

Ошибка 3: Отсутствие проверки качества на стыках

После Промта 2 у вас получился промежуточный результат, который уже содержит ошибку. Промт 3 её только усугубит. **Добавляйте в цепочку контрольные шаги**: например, после Промта 2 попросите нейросеть: *«Оцени, насколько этот список соответствует критериям. Если оценка ниже 8, перепиши»*.

Резюме главы 1

Цепочка промтов — это способ разбить сложную задачу на последовательность простых, где выход одного шага — вход другого.

Она лучше одного супер-промта, потому что каждый шаг простой, понятный и контролируемый.

Цепочки бывают **последовательные**, **ветвящиеся** и **циклические**.

Строить их можно вручную (диалог), с помощью кон-

структоров (Zapier, Make) или через простой код (Google Apps Script).

Главное правило: не мельчить, не терять контекст, добавлять контроль качества на стыках.

Вы уже умеете писать отличные одиночные промты. Теперь вы знаете, как соединять их в цепочки. Это первый шаг к тому, чтобы нейросеть стала вашим заместителем.

В следующей главе мы пойдём дальше: **«Промт как функция» —** превратим промты в шаблоны с переменными и научимся применять их к тысячам разных входных данных.

Глава 2
Промт как функция
Один шаблон — тысячи результатов

В прошлой главе мы научились соединять промты в цепочки. Но у этого подхода есть неудобство: каждый промт в цепочке обычно написан под конкретные данные. Сегодня вы анализируете отзывы на кафе, завтра — на фитнес-клуб. Приходится переписывать промты заново.

Что если я скажу, что один и тот же промт можно использовать сотни раз, просто подставляя разные входные данные? Как формулу в Excel: вы пишете =СУММ(A1:A10) один раз, а потом меняете диапазон или копируете в другую

ячейку.

Промт как функция — это именно такой подход. Вы создаёте шаблон промта с «дырками» (переменными), а потом подаёте на вход разные значения. Нейросеть каждый раз выполняет одну и ту же операцию, но с новыми данными.

В этой главе мы разберём:

Что такое «промт-функция» и чем она отличается от обычного промта.

Как создавать переменные и подставлять их.

Промты с условиями (if-else) и циклами (для массовой обработки).

Реальные примеры: генерация персонализированных писем для 500 клиентов, массовое извлечение сущностей из текстов, превращение таблиц в единый формат.

Инструменты для работы с промтами-функциями (от Google Sheets до API).

К концу главы вы сможете превратить любой свой промт в многоразовый «движок», который экономит не минуты, а часы.

2.1. Обычный промт vs промт-функция: в чём разница

Обычный промт — это как письменное указание для конкретного случая.

«Напиши приветственное письмо для нового клиента

Ивана Иванова, который купил подписку "Базовую". Упомяни, что у нас есть бесплатный вебинар по средам. Подпись: Алексей.»

Он работает один раз. Чтобы написать письмо другому клиенту, вы переписываете имя, тариф, возможно, другие детали.

Промт-функция — это шаблон с переменными.

«Напиши приветственное письмо для нового клиента [ИМЯ], который купил подписку "[ТАРИФ]". Упомяни, что у нас есть бесплатный вебинар по средам. Подпись: Алексей.»

Теперь вы можете подставить [ИМЯ] = "Мария Петрова", [ТАРИФ] = "Премиум" — и нейросеть сгенерирует письмо для Марии. Поменяете переменные — получите письмо для следующего клиента.

Ключевые отличия:

Характеристика	Обычный промт	Промт-функция
Наличие переменных	Нет	Есть (в квадратных скобках или другом синтаксисе)
Переиспользование		

Низкое
Высокое (один шаблон на тысячи запусков)

Автоматизация
Ручной ввод данных
Подстановка из таблицы или скрипта

Сложность написания
Просто
Чуть сложнее (нужно предусмотреть варианты)

По сути, промт-функция — это превращение вашего навыка промтинга в «программу», которую может исполнить нейросеть многократно.

2.2. Синтаксис переменных: как обозначать и подставлять

Нейросети понимают переменные в промте, если вы их явно отметите. Самый надёжный способ — **использовать квадратные скобки** или **двойные фигурные скобки**

Конец ознакомительного фрагмента.