Advanced-промптинг, каким образом добиваться от моделей нужного результата

Марат Зимнуров

ЦИФРОВЫЕ ДНИ ИННОПОЛИСА
DIGITAL INNOPOLIS DAYS

Марат Зимнуров

Руководитель группы разработки ООО «Авито Тех»»

Разрабатываем HR Tech: приём, отпуска, согласия, бенефиты, увольнения, удержания, мерит, комплайенс и так далее...

Преподаватель курса по AI в ИТМО

Преподаватель на кафедре ИТиЦЭ в ИГХТУ: функциональное программирование, фундаментальная информатика, технологии программирования, тестирование программного обеспечения

Ревьювер практик управления проектами ИТМО х ВШЭ

Автор научных работ, образовательных и open- source-проектов, спикер профессиональных конференций HolyJS и CodeFest.

Исследую функциональное программирование, LLM и формальные способы объяснить машине, что правдоподобный код — еще не обязательно рабочий.

Ex-worker: Microsoft, Adjust, Roadmunk, Xsolla, Контур, EPAM, Akvelon

Написал детскую книгу, опубликовал музыкальный альбом, веду блоги и стримы, разрабатываю казуальные игры и визуальные новеллы



ОТКРЫТЫЙ ПОРТАЛ ИНЖЕНЕРНЫХ ЗНАНИЙ

Разобраться, вырасти и сделать

Курсы, статьи, видео и интерактивные разборы по разработке, архитектуре, AI, карьере и коммуникации. Без регистрации и пейволла.

● Что здесь есть | ○ Куда расти | ○ Доступ и продукты

УЧЕБНАЯ КАРТА

Выберите ступень

7 ступеней · 70 треков · 1075 глав · 1149 материалов · 18 видео

Верхний ярус карты: семь ступеней, за каждой треки и главы. Порядок не обязателен — откройте ту, что про вас сегодня.

Вся карта: 1152 узла →

01 **Первые шаги** — Как учиться, как устроен компьютер, как делают софт. — 7 треков · 116 глав →

02 **Фундамент** — Математика, алгоритмы, структуры данных, ОС, парадигмы. — 10 треков · 161 глава →

03 **Языки и парадигмы** — Восемь языков с нуля до продакшена. — 13 треков · 206 глав →

04 **Проектирование систем** — Паттерны, стили, границы контекстов, компромиссы. — 5 треков · 71 глава →

05 **Данные, ML и ИИ** — Данные, базы, ML, нейросети, ИИ-инженерия. — 10 треков · 161 глава →

06 **Инфраструктура, инструменты и качество** — CI/CD, облака, тестирование, редакторы, безопасность. — 16 треков · 230 глав →

07 **Продукт, процесс и люди** — Продукт, проекты, Scrum, системный анализ, UX. — 9 треков · 130 глав →

ВАШ КОНТЕКСТ

Продолжить

Вы остановились здесь

Продвинутые техники: chain-of-thought, self-consistency,...

65% прочитано

Новое · 04.09.2026

Своя поверхность: как написать тот же язык по-...

6 мин чтения

Опишите задачу своими словами

Локальный поиск предложит курс и покажет, почему.

КРОМЕ КАТАЛОГА

Карта знаний целиком → | Книги портала: 2 → | Пространство логических ошибок: 67 приёмов с разбором → | Все теги списком →

LLM и их токены

#мантра



Это новое электричество и Интернет

Тогда человек зачем?

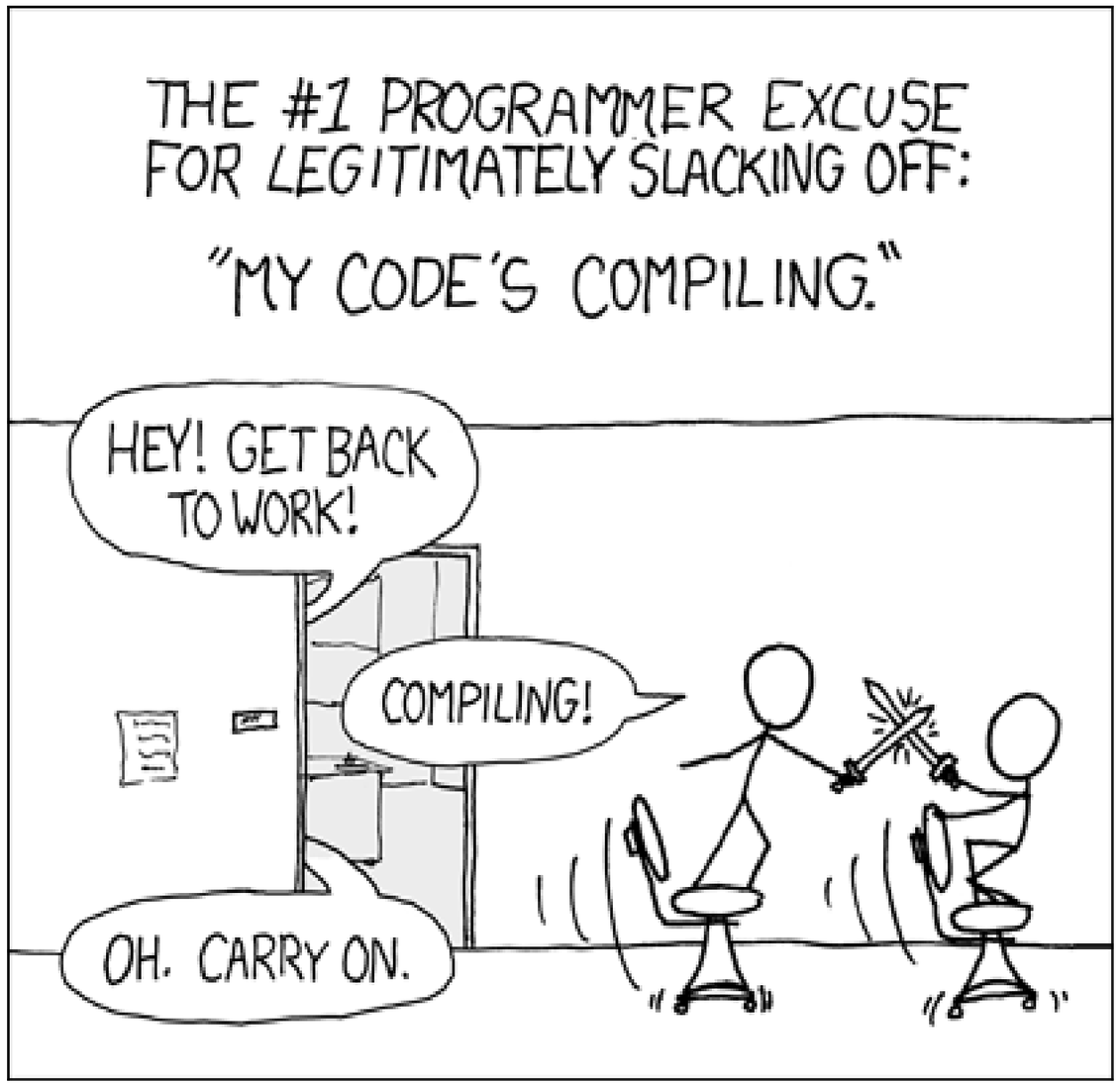
Чтобы помогало, а не шарахнуло

Нейрослоп приходит на code review, в чатах, документации



2007 Компилируется

2026 Агенты работают



Начнём с человеческой коммуникации

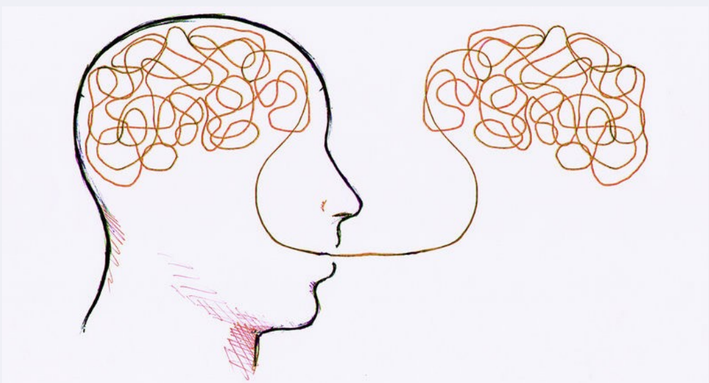
?

Отчёт опять сломан. Ты же senior. Сделай нормально!!1

Типищно

Короче, мы сделали новую версию отчётов, вчера пока параллельно работал над задачей для новой версии нашей системы отчётов. Ну и потом чё? Наш Лёха выкатил новый PR. А у нас там конфликты, мы с ним короче созвонились, а потом представляешь? Пока мы фиксили наши падающие тесты всё пошло не по плану и оказалось что контрпримеры невалидные, типа тест подобрать не могнули. Я теряюсь в догадках что с этим делать





Начнём с человеческой коммуникации

!

Отчёт опять сломан. Ты же senior. Сделай нормально!!1

S

Чужой пользователь видит отчёт

T

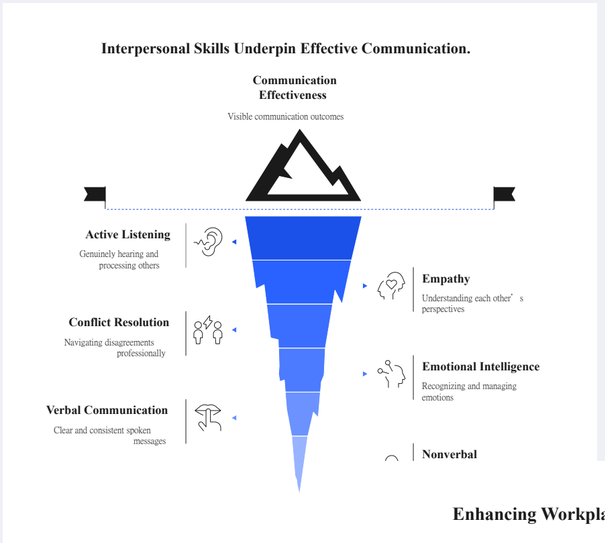
Нужно выдать доступ только незаблокированному сотруднику

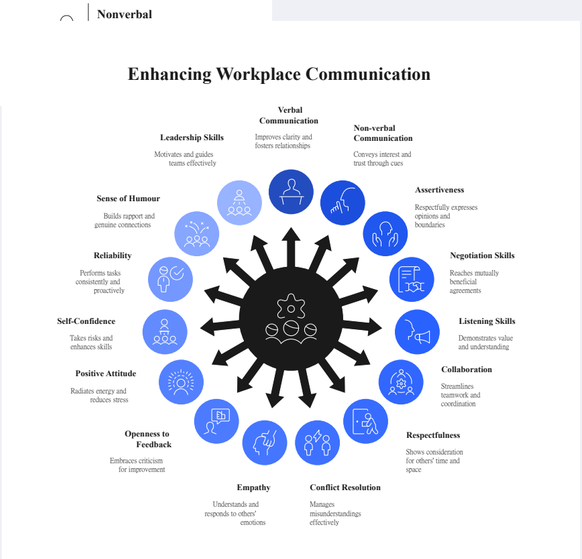
A

Прогнал несколько тестов, 4 шт

R

75% тестов проходят





А как коммуницировать с LLM?

#мантра

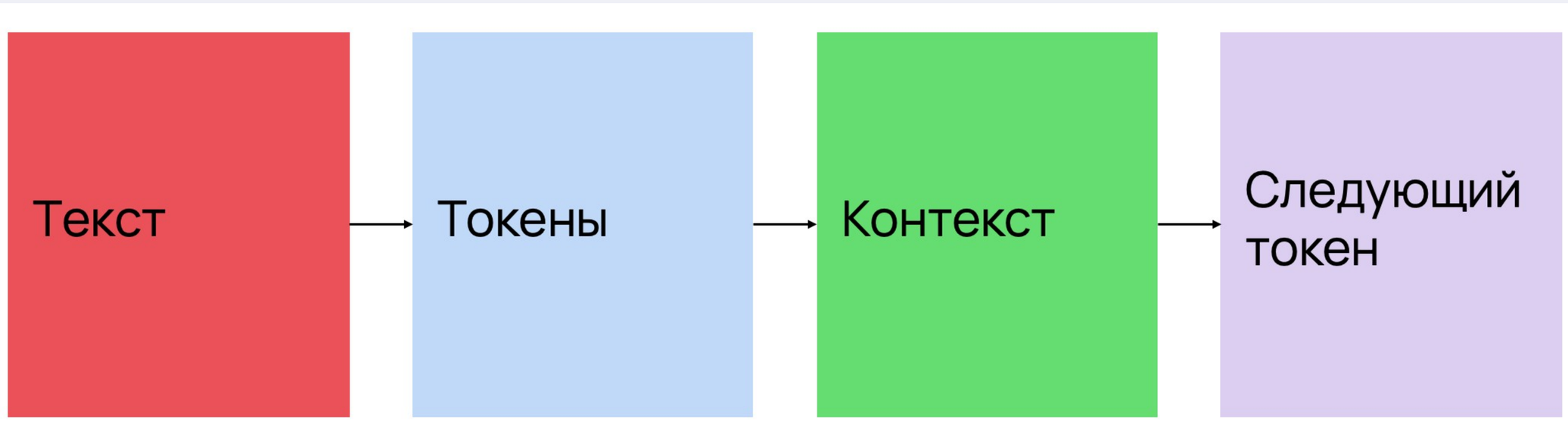
Шаблонно!

Примеры нейрослопа с судоязыком

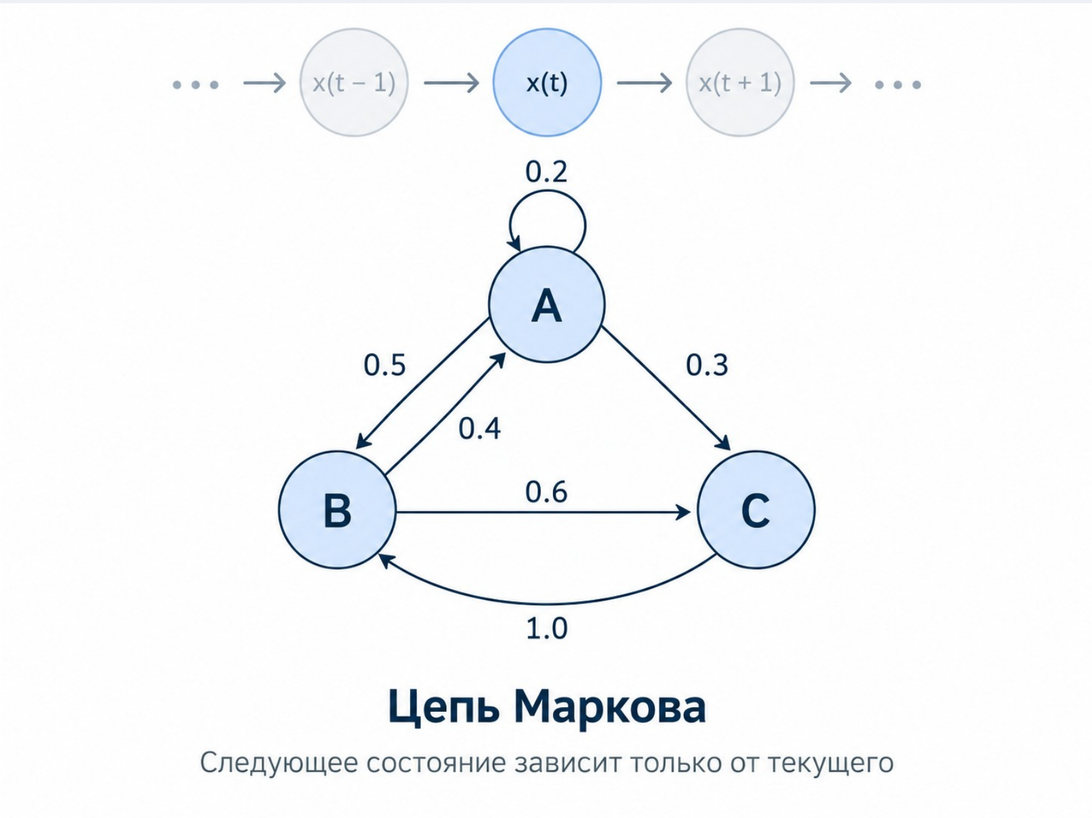
Топо-фикс….

Сторож тотальности… Какой вход запускали?

Как LLM понимает текст?



Цепи Маркова?



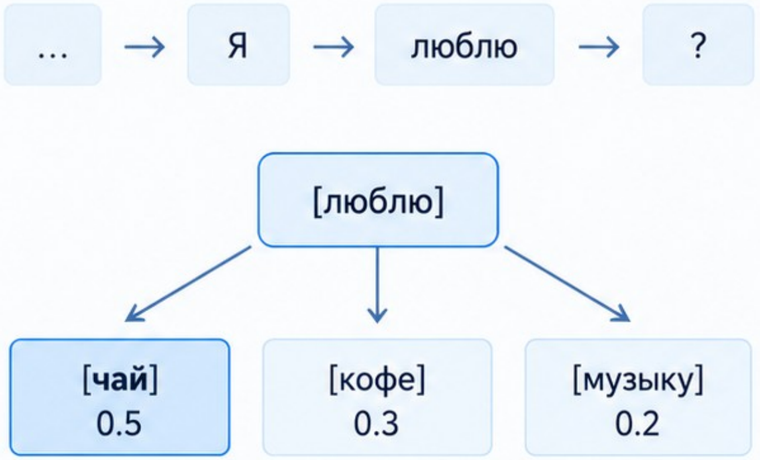
T9: цепи Маркова vs LLM

Оба предсказывают следующее слово, но делают это по-разному

T9 на цепях Маркова

Текущий текст: «Я люблю»

Смотрим в основном на текущее состояние / последнее слово



Решение опирается в основном на локальные переходы между словами.

T9 на LLM

Контекст: «Сегодня холодно, и я люблю»

Модель смотрит на весь контекст и предсказывает следующий токен.

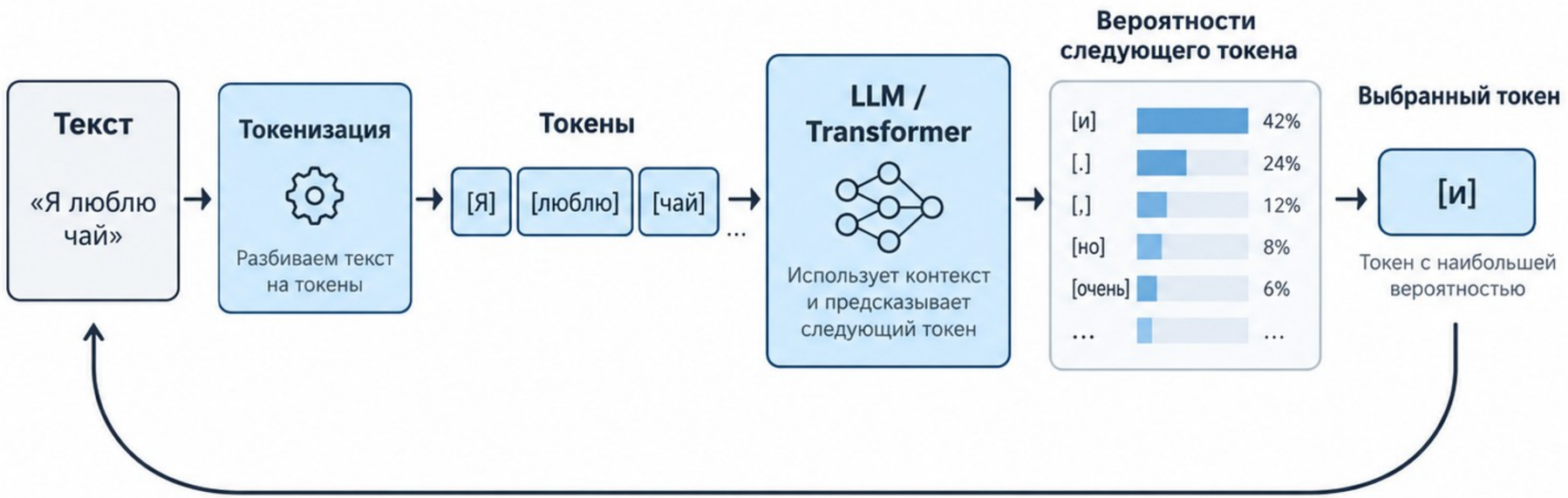


Решение использует весь контекст, а не только последнее слово.

Идея: цепи Маркова чаще смотрят на локальный переход, а LLM — на более широкий контекст.

LLM и токены

Модель предсказывает следующий токен по предыдущим токенам

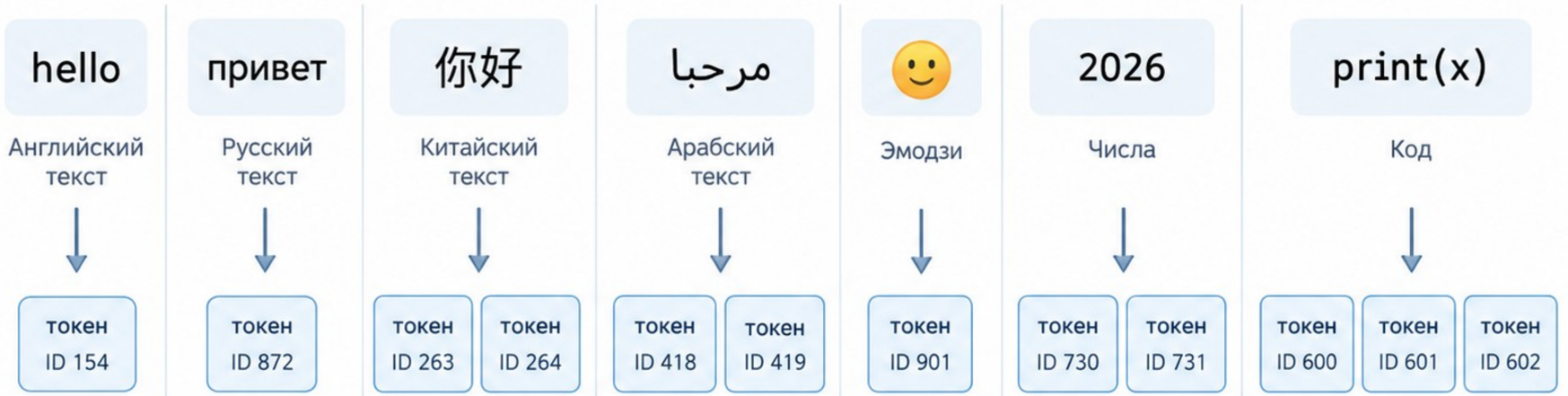


Добавляем токен к контексту и повторяем процесс

Теперь контекст: «Я люблю чай и ...»

Токен — это кусочек текста

Не зависит от языка: слова, части слов, символы, числа, эмодзи, код



Один токен ≠ одно слово

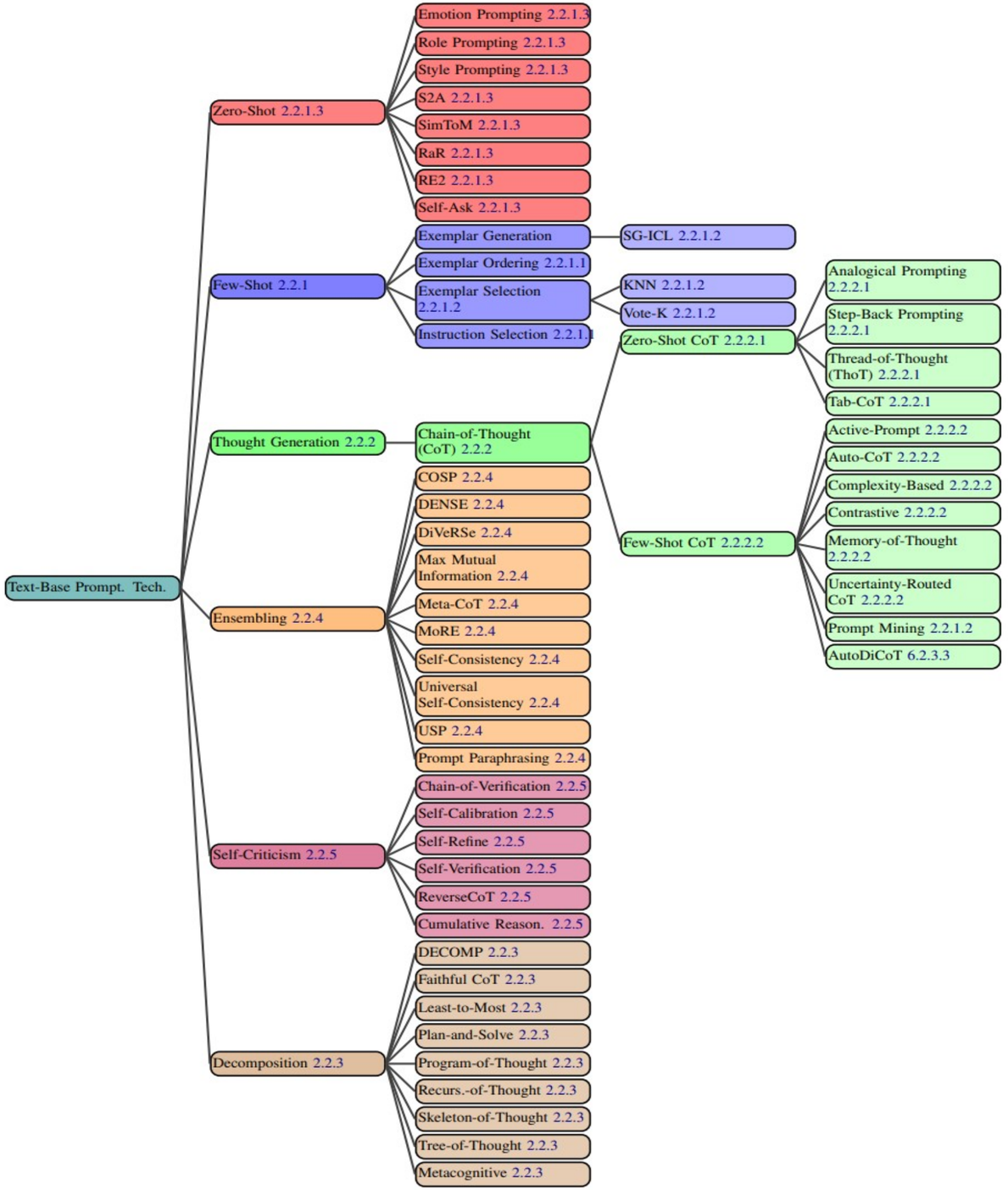
Слово может состоять из нескольких токенов,
а токен может быть словом, частью слова, символом, числом и т. д.

Зачем учиться промптить

LLM не читает мысли: она продолжает текст по вероятности

Плохой	**«Сделай нормально!!1»**	Модель угадывает задачу и часто отвечает общими словами.
Хороший	**роль + контекст + задача + формат**	Модель получает рамки и понимает, какой результат нужен.
Продвинутый	**шаги + проверка + инструменты**	Модель не просто отвечает, а планирует, проверяет и уточняет.

Промптинг — это не «магические слова», а способ ясно поставить задачу машине.


Text-Base Prompt. Tech.
Zero-Shot 2.2.1.3
Few-Shot 2.2.1
Thought Generation 2.2.2
Ensembling 2.2.4
Self-Criticism 2.2.5
Decomposition 2.2.3
Emotion Prompting 2.2.1.3
Role Prompting 2.2.1.3
Style Prompting 2.2.1.3
S2A 2.2.1.3
SimToM 2.2.1.3
RaR 2.2.1.3
RE2 2.2.1.3
Self-Ask 2.2.1.3
Exemplar Generation
Exemplar Ordering 2.2.1.1
Exemplar Selection 2.2.1.2
Instruction Selection 2.2.1.4
SG-ICL 2.2.1.2
KNN 2.2.1.2
Vote-K 2.2.1.2
Chain-of-Thought (CoT) 2.2.2
COSP 2.2.4
DENSE 2.2.4
DiVeRSe 2.2.4
Max Mutual Information 2.2.4
Meta-CoT 2.2.4
MoRE 2.2.4
Self-Consistency 2.2.4
Universal Self-Consistency 2.2.4
USP 2.2.4
Prompt Paraphrasing 2.2.4
Chain-of-Verification 2.2.5
Self-Calibration 2.2.5
Self-Refine 2.2.5
Self-Verification 2.2.5
ReverseCoT 2.2.5
Cumulative Reason. 2.2.5
DECOMP 2.2.3
Faithful CoT 2.2.3
Least-to-Most 2.2.3
Plan-and-Solve 2.2.3
Program-of-Thought 2.2.3
Recurs.-of-Thought 2.2.3
Skeleton-of-Thought 2.2.3
Tree-of-Thought 2.2.3
Metacognitive 2.2.3
Zero-Shot CoT 2.2.2.1
Few-Shot CoT 2.2.2.2
Analogical Prompting 2.2.2.1
Step-Back Prompting 2.2.2.1
Thread-of-Thought (ThoT) 2.2.2.1
Tab-CoT 2.2.2.1
Active-Prompt 2.2.2.2
Auto-CoT 2.2.2.2
Complexity-Based 2.2.2.2
Contrastive 2.2.2.2
Memory-of-Thought 2.2.2.2
Uncertainty-Routed CoT 2.2.2.2
Prompt Mining 2.2.1.2
AutoDiCoT 6.2.3.3

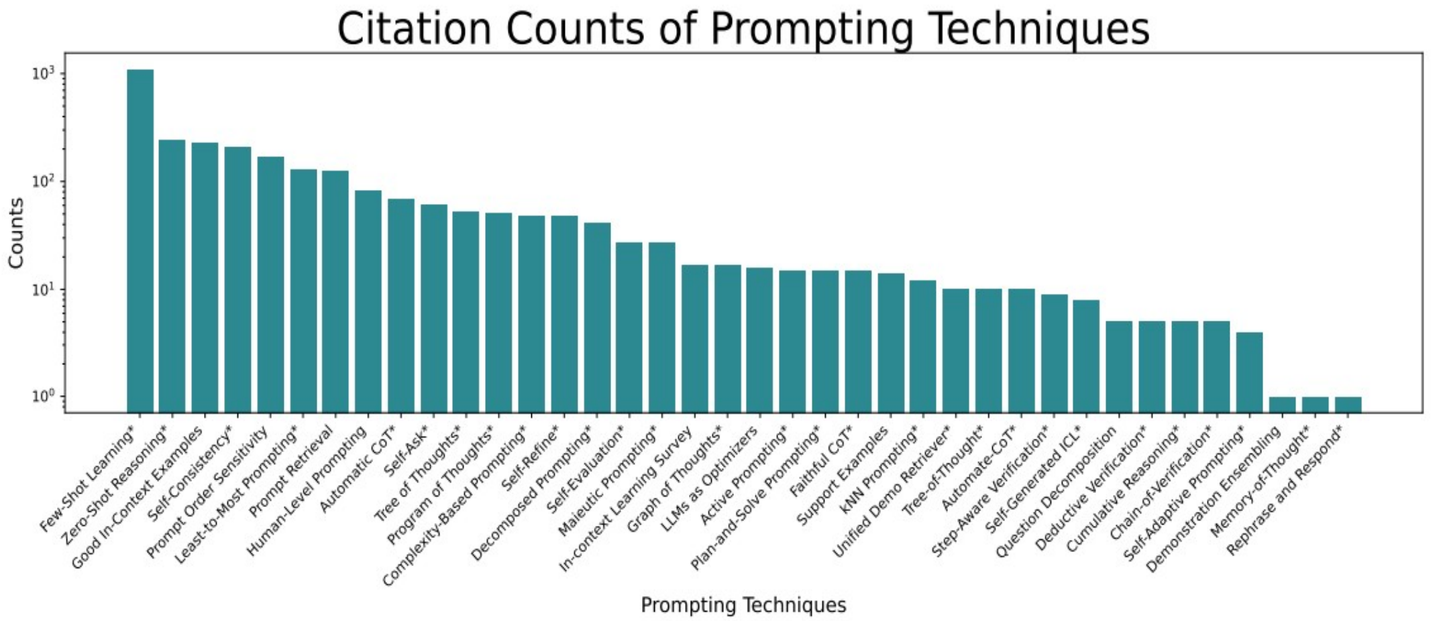


Figure 2.11: Citation Counts of Prompting Techniques. The top 25 papers in our dataset, measured by how often they are cited by other papers in our dataset. Most papers here are prompting techniques*, and the remaining papers contains prompting advice.

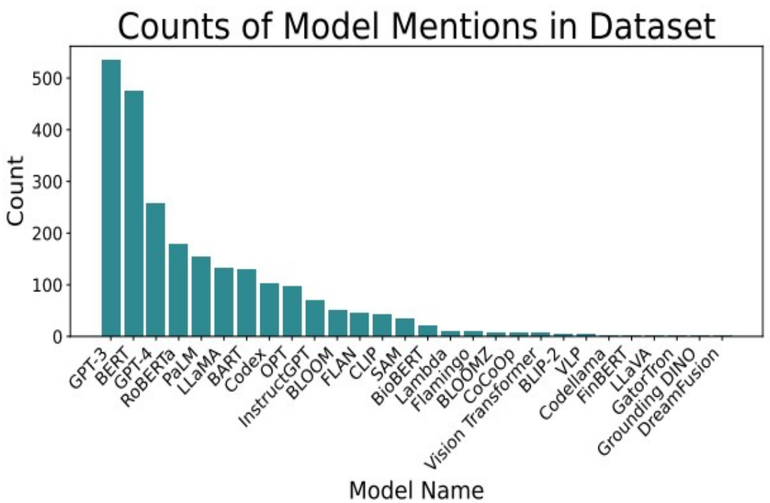


Figure 2.9: Citation Counts of GenAI Models

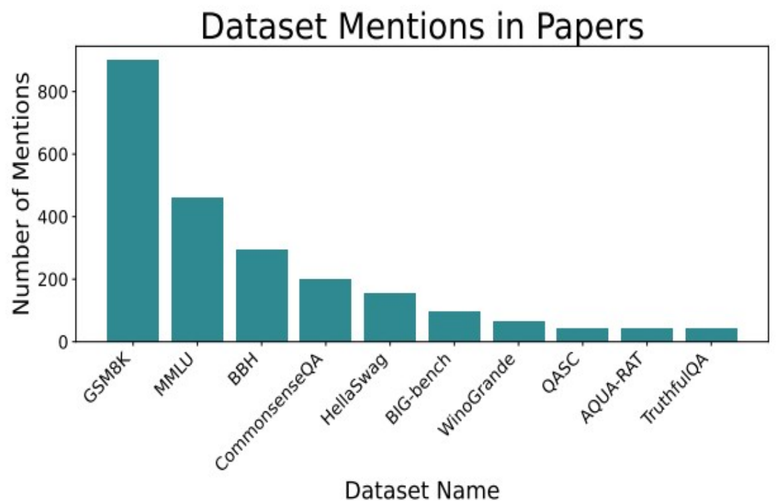
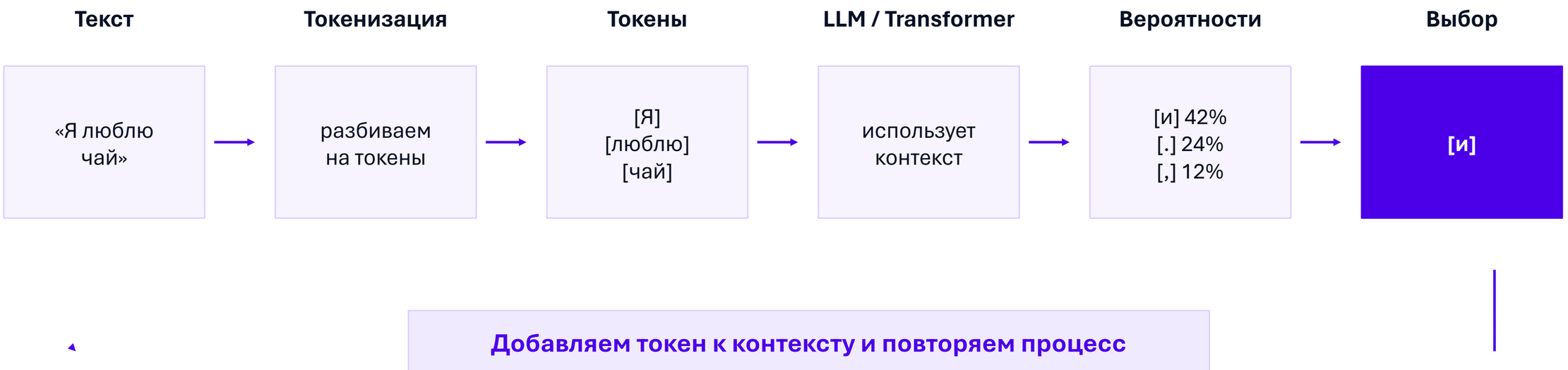


Figure 2.10: Citation Counts of Datasets

Как LLM работает с текстом

Модель предсказывает следующий токен по предыдущим токенам



LLM генерирует текст по одному токену за раз, каждый раз используя весь предыдущий контекст.

Zero-shot и Few-shot

База: либо просим сразу, либо показываем примеры

Zero-shot

Попытка решить задачу за раз

Когда задача простая: классификация, краткий перевод, форматирование.

Пример: «Суммируй текст в 3 пункта».

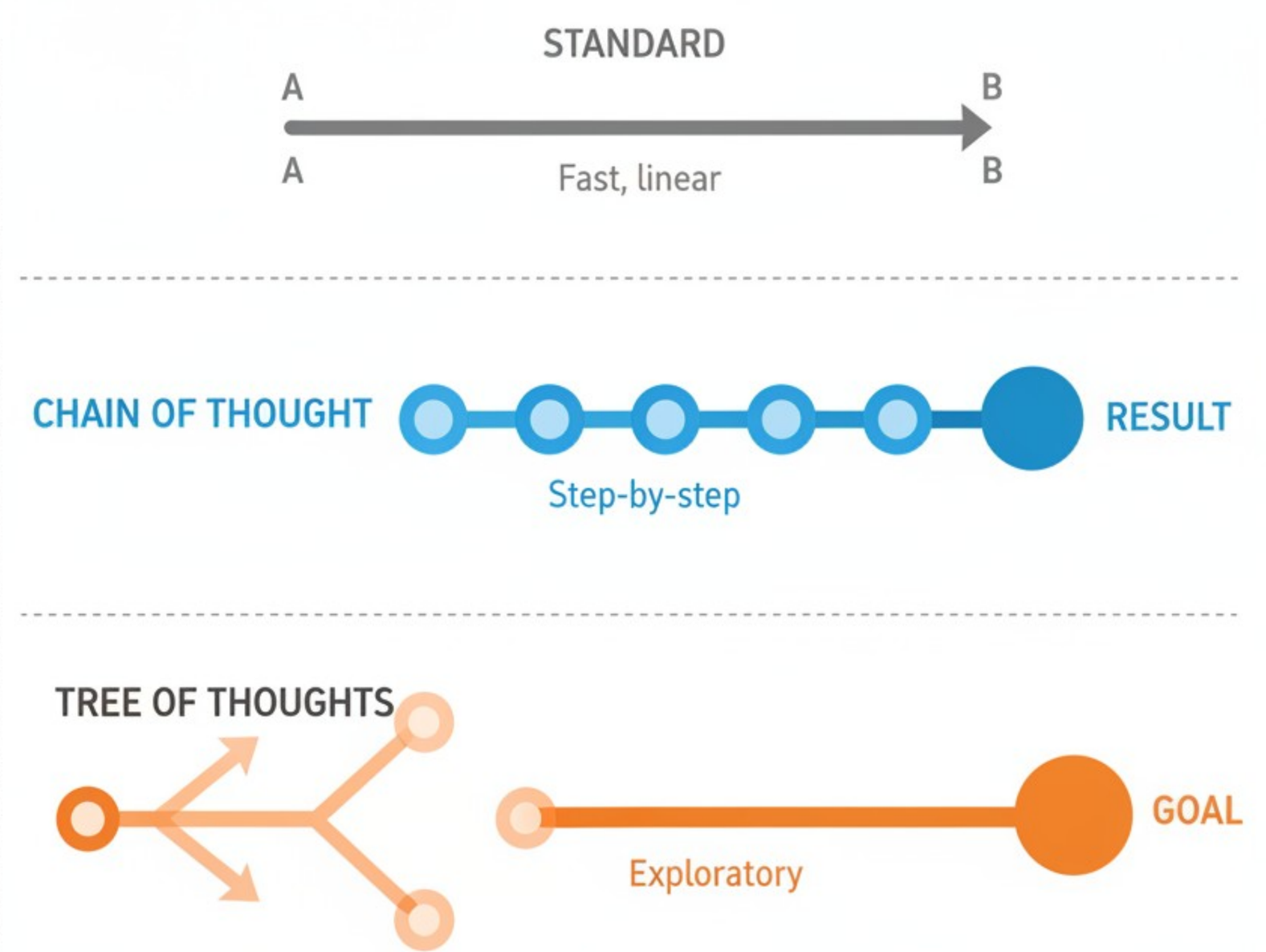
Few-shot

Zero-shot + несколько примеров

Когда важен стиль, формат или нестандартное правило.

Пример: покажи 2–3 варианта и ожидаемые ответы.

Few-shot — это самый простой способ объяснить модели, «как выглядит правильный ответ».


STANDARD
A
B
A
B
Fast, linear
CHAIN OF THOUGHT
RESULT
Step-by-step
TREE OF THOUGHTS
GOAL
Exploratory

Chain-of-Thought: просим идти шагами

Для задач с логикой полезно попросить план, проверки и краткие шаги

Идея

Модель получает больше «времени подумать»: сначала разбирает условие, потом отвечает. В реальной работе лучше просить краткий план и проверяемые шаги, а не длинные скрытые рассуждения.

Шаблон

Сначала перечисли важные условия. Затем реши задачу. В конце дай короткий ответ и проверь, нет ли ловушки.

Мини-задача

У меня 5 мокрых рубашек. Одна рубашка сохнет на солнце 4 часа. Сколько времени потребуется, чтобы высушить все 5?

Что важно проверить: процесс последовательный или параллельный? Хватает ли места на верёвке?

CoT помогает не прыгать к первому ответу, а проверить условия задачи.

Chain of Verification

Черновик → вопросы к черновику → проверка → финальный ответ

1 **Draft**

Напиши черновик решения.

→ **2** **Verification**

Сформулируй 3 вопроса, которые могут найти ошибку.

→ **3** **Check**

Ответь на вопросы и проверь синтаксис/факты.

→ **4** **Final**

Верни исправленный финальный ответ.

Пример задачи

Написать команду FFmpeg для создания GIF из видео с водяным знаком.

Вопросы проверки: правильный порядок input-файлов? корректный filter_complex? указан ли palettegen/paletteuse для качества?

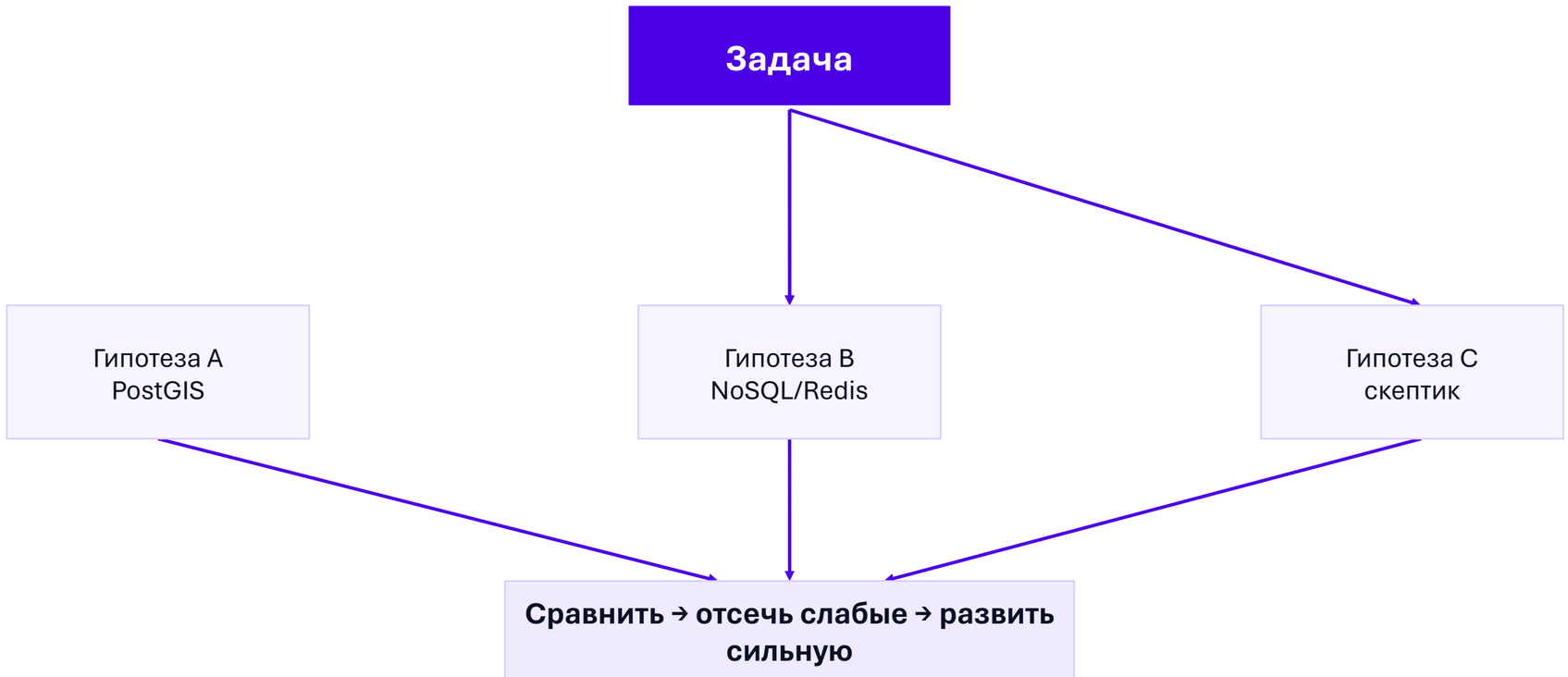
CoV полезен там, где ошибка дорогая: команды, код, факты, инструкции.

Tree of Thoughts

Не один путь решения, а несколько веток с критикой

Когда нужен

Архитектура систем, стратегические решения, brainstorming, сюжетные линии — там, где нет одного очевидного ответа.



Промпт: представь трёх экспертов, пусть предложат варианты, покритикуют друг друга и придут к компромиссу.

ReAct: Reason + Act

Модель думает, действует инструментом, смотрит результат и повторяет



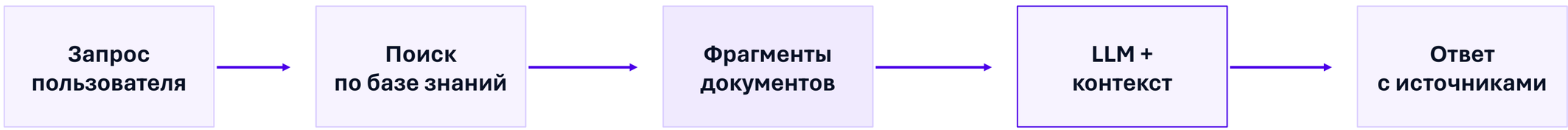
Цикл: пока нет надёжного ответа, модель может снова выбрать действие.

ReAct нужен, когда ответ зависит от внешнего действия: поиск, калькулятор, база данных, запуск теста.

RAG: Retrieval-Augmented Generation

Сначала достаём внешние знания, потом просим модель ответить по ним

Без RAG модель отвечает только из того, что «помнит». С RAG она получает нужные документы в промпт.



Важно: RAG не гарантирует правду. Надо проверять качество поиска, релевантность фрагментов и источники.

Пример: «ответь по школьному регламенту» → сначала найти регламент → затем сформировать ответ.

Как выбрать технику

Чем сложнее задача — тем больше структуры, проверки и контекста

Техника	Аналогия	Для чего	Точность	Latency & Cost
Zero-shot	Интуиция	Классификация, простые переводы, форматирование данных.	★★	низкие
Few-shot	Пример	Стиль, формат, правила, которые проще показать, чем описать.	★★★	низкие/средние
CoT / CoV	Решение + проверка	Логика, математика, команды, дебаг, факты.	★★★★	средние
ToT / ReAct / RAG	Команда экспертов	Архитектура, внешние знания, инструменты, сложные решения.	★★★★★	высокие

Правило: сначала самый простой промпт. Усложняем только если качество не хватает.

Анатомия хорошего промпта: RCTF

:3

Когда промпт структурирован, модель меньше додумывает и больше следует задаче.

Role

Кем должна быть модель?
Например: Senior QA, архитектор, редактор.

Context

Что уже известно?
Проект, ограничения, вводные, аудитория.

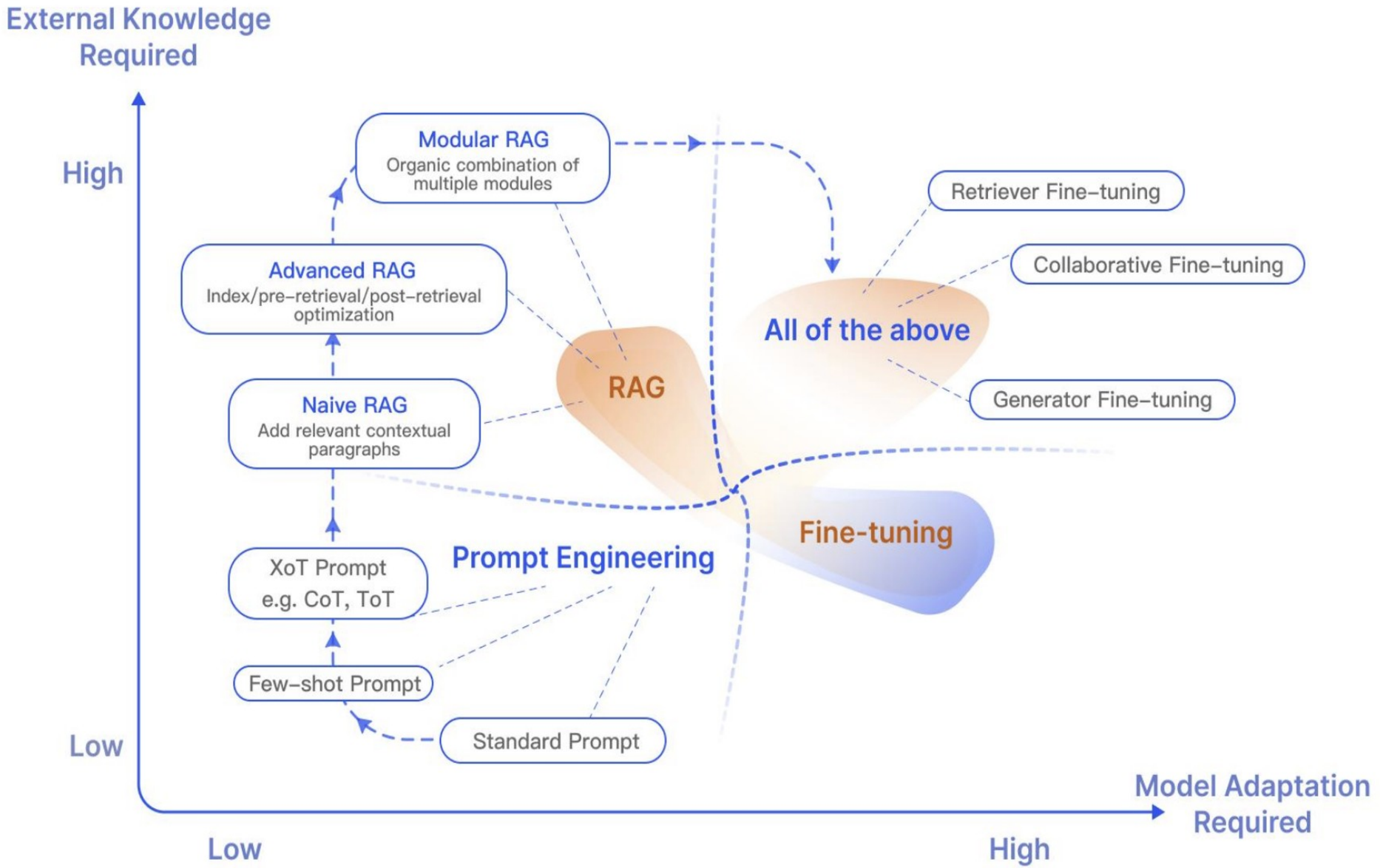
Task

Что нужно сделать?
Одно действие, ясный результат.

Format

Как вернуть ответ?
JSON, таблица, список, код, критерии.





A typical case of RAG

Практика: превращаем хаос в промпт

Берём живую формулировку и делаем из неё управляемую задачу

Плохой запрос

?

Задание : перепишите плохой запрос в RCTF и добавьте один шаг проверки.

Чеклист хорошего промпта

Перед отправкой проверь пять вещей

Есть проверка

Как понять что ответ хороший?

Роль
С какой экспертизой?

Контекст
Какие вводные важны?

Задача
Что именно надо сделать?

Формат
Как выглядит ответ?



Если промпт не проходит чеклист, модель будет угадывать.

Техники

Zero-shot, a few shot, CoV, CoT, ToT, ReAct, RAG, RCTF

СПАСИБО
ЗА ВНИМАНИЕ!

ЦИФРОВЫЕ ДНИ ИННОПОЛИСА
DIGITAL INNOPOLIS DAYS

Марат Зимнуров

Руководитель группы разработки ООО «Авито Тех»

ЦИФРОВЫЕ ДНИ
ИННОПОЛИСА
DIGITAL INNOPOLIS DAYS