Двухагентная система «Менеджер–Служба поддержки» по сравнению со стандартным чат-ботом

Олег Зельдин Апекс Берг

Двухагентная система «Менеджер–Служба поддержки» по сравнению со стандартным чат-ботом

Обзор

Аннотация

Авторы предлагают двухагентную схему AutoManager, где 2 бота («Администратор» и «Ассистент») разделяют общую базу знаний и автономно решают свои задачи; общение идёт через базу знаний с обеспечением согласованности.

В сравнении с реальным Taco Bell Drive-Thru AI (стандартный чат-бот для оформления заказа из автомобиля) эта система показала большую надёжность

Источник

Название статьи и дата публикации

- Reliable Collaborative Conversational Agent System based on LLMs and Answer Set Programming. 09.06.2025

Ссылка

- https://clck.ru/3PuwsZ

Ключевые идеи

🧩 Два бота — одна цель: «Помощник» принимает заказ, «Администратор» управляет меню и остатками.

📚 Единая база знаний: общие факты и правила → одинаковая «картина мира» у обоих ботов.

🧠 Логика в ASP, а LLM используется только как «переводчик» языка: строгие проверки и объяснимые решения без лишних «галлюцинаций».

🛡️ Безопасная передача данных: служебные сведения инкапсулированы и не видны пользователю.

📈 Практический эффект: надёжнее и стабильнее, чем реальный AI-приёмщик заказов Taco Bell.

Такой разделённый подход делает оформление заказов предсказуемым: бот не предлагает недоступное и согласованно уточняет детали.
Демонстрация на AutoManager (drive-thru) показала более высокую надёжность по сравнению с реальным Taco Bell AI.

Сотрудничество — основа цивилизации

- Люди добиваются результатов, когда роли разделены и согласованы: каждый делает своё, но цель общая.
- С ботами то же самое: разные агенты для разных задач, но общая работа и единая картина мира.
- Если боты обмениваются точной информацией и действуют по правилам, система работает устойчиво и предсказуемо.

🤝 Разные роли — одна цель

🧭 Единые правила и общая база знаний

🔄 Своевременный обмен данными

🧩 Согласованность вместо хаоса

Пример сотрудничества: аэропорт, регистрация

- «Операционный» бот управляет гейтами и реагирует на задержки/нештатные ситуации.
- «Стойка» помогает пассажиру: чек-ин, выбор места, багаж, апгрейд, актуальная информация.
- Когда что-то меняется (поломка гейта, перенос рейса), боты мгновенно делятся обновлениями и корректируют действия.

🛬 Операции: гейты, задержки, экстренные случаи

🧳 Пассажиры: регистрация, места, багаж, апгрейд

⚡ Мгновенные обновления между ботами

✅ Правильные решения «по ситуации»

Почему боты только на LLM часто неудачны для диалогов, ориентированных на решение задачи?

- LLM хорошо «говорят», но не всегда надёжно рассуждают и используют «мгновенные» данные, переданные через промпты.
- Без надёжного механизма обмена фактами бот может не заметить, что ингредиент закончился, и оформить неверный заказ.
- McDonald и Taco Bell показывают проблему на практике: требуется постоянная ручная подстройка и контроль

Это скорее помощники, чем самостоятельные системы.

🧠 Нет строгой логики ⟶ ошибки и «галлюцинации»

📝 Длинные промпты вместо структурированных фактов

🕳️ Риск пропустить важные обновления (ингредиенты, статусы)

👨‍✈️ Нужен человек-надсмотрщик ⟶ нет автономности

Главный вклад работы в клиентский сервис на основе предложенной системы мультиагентов

- Предложена надёжная двухагентная схема: Администратор + Помощник на базе ASP (строгая логика) и LLM (только «переводчик» текста↔факты).
- Агенты делят общую базу знаний, обмениваются не «болтовнёй», а структурированными предикатами, поэтому решение согласованное и безопасное.
- Результат — устойчивое выполнение задач в реальном мире (пример AutoManager для drive-thru) с меньшей ошибочностью, чем у классических LLM-ботов.

🧩 Две роли: Администратор (меню/остатки) ↔ Ассистент (диалог/заказ)

📚 Общая база знаний + правила (ASP)+решатель s(CASP)

🔐 Безопасная, контролируемая коммуникация

📈 Надёжнее и предсказуемее в реальных сценариях

Что такое ASP (Answer Set Programming)?

- ASP — это способ записывать знания как факты и правила, а затем логически выводить ответы. Это как «тетрадь с правилами здравого смысла»: если А и Б верно, то В — тоже.

```
# Правило по умолчанию:
flies(X) :- bird(X), not abnormal_bird(X).
abnormal_bird(X) :- penguin(X).

# Ограничение целостности:
false :- person(X), sit(X), stand(X).
```

🧠 Факты: короткие утверждения («Воробей — птица»).

📏 Правила: «обычно так, если нет исключений».

🚫 Запреты (ограничения целостности): что невозможно одновременно.

🌍 Несколько миров: альтернативные версии реальности (в сказочном мире птицы похожи на реальных и также летают, но в сказочном - говорят, в реальном — нет).

4 категории знаний в ASP

В ASP это записывается специальным кодом:

```
# Правило по умолчанию:
flies(X) :- bird(X), not abnormal_bird(X).
abnormal_bird(X) :- penguin(X).

# Ограничение целостности:
false :- person(X), sit(X), stand(X).
```

Что такое s(CASP) и как он работает?

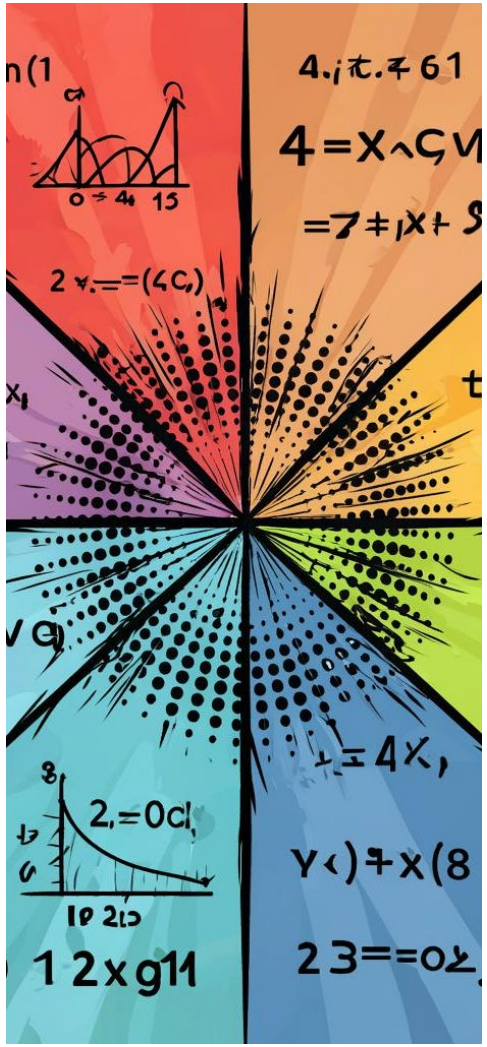
- CASP — Constraint Answer Set Programming s(CASP) — это целеориентированная (top-down) реализация CASP без полного разворачивания правил, которая унаследована от системы s(ASP) и дополнена поддержкой ограничений; её идея — отвечать на запросы «как детектив», строя доказуемые объяснения
- Простыми словами s(CASP) — это «умный решатель» для ASP: он отвечает на вопросы сверху вниз, как детектив.

Мини-пример «как детектив»:

Вопрос: «Летает ли воробей?»

1. s(CASP) проверяет: Воробей — птица? (Да)
2. Есть ли исключение: Воробей — пингвин? (Нет)
3. Применяет правило «птицы обычно летают» ⟶ Ответ: да, летает.
4. Если позже узнаём, что Воробей войдет в исключения, правило-исключение сработает, и ответ станет «нет».

Больше информации про ASP



- G. Brewka, T. Eiter, and M. Truszczynski,
  “**Answer set programming at a glance**”
  pp. 92–103, 12 2011.
  https://clck.ru/3PvkC2

- G. Gupta
  “**Automating common sense reasoning with ASP and s(CASP)**”
  2022, technical Report
  https://utdallas.edu/~gupta/csr-scasp.pdf.

- G. Gupta
  “**Automating common sense reasoning with ASP and s(CASP)**”
  2022, technical Report
  https://utdallas.edu/~gupta/csr-scasp.pdf.

Зачем это нужно ботам?

Проблема LLM-ботов: они красиво говорят, но иногда «галлюцинируют» и путают логику.

Решение: LLM — только «переводчик текста», а логика — в ASP/s(CASP).

🤖 LLM: преобразует фразу человека в факты/вопросы и обратно в понятный текст.

🧠 ASP/s(CASP): строго проверяет правила, исключения и запреты, даёт надежный ответ.

🔄 Совместная работа ботов: все смотрят в общую базу фактов, не забывают про исключения и ограничения.

🛡️ Меньше ошибок: если ингредиент закончился, правило «недоступно» сработает всегда — бот не предложит это блюдо.

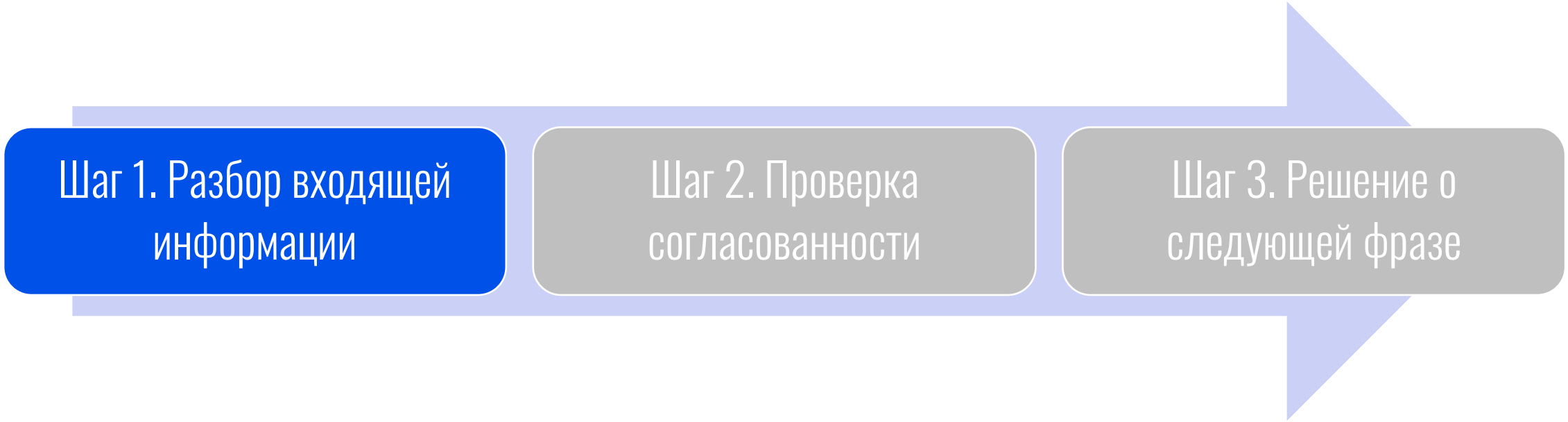
Микро-сценка для сервиса еды:

Факт: «сыр закончился».

Правило: «буррито содержит сыр → буррито недоступно».

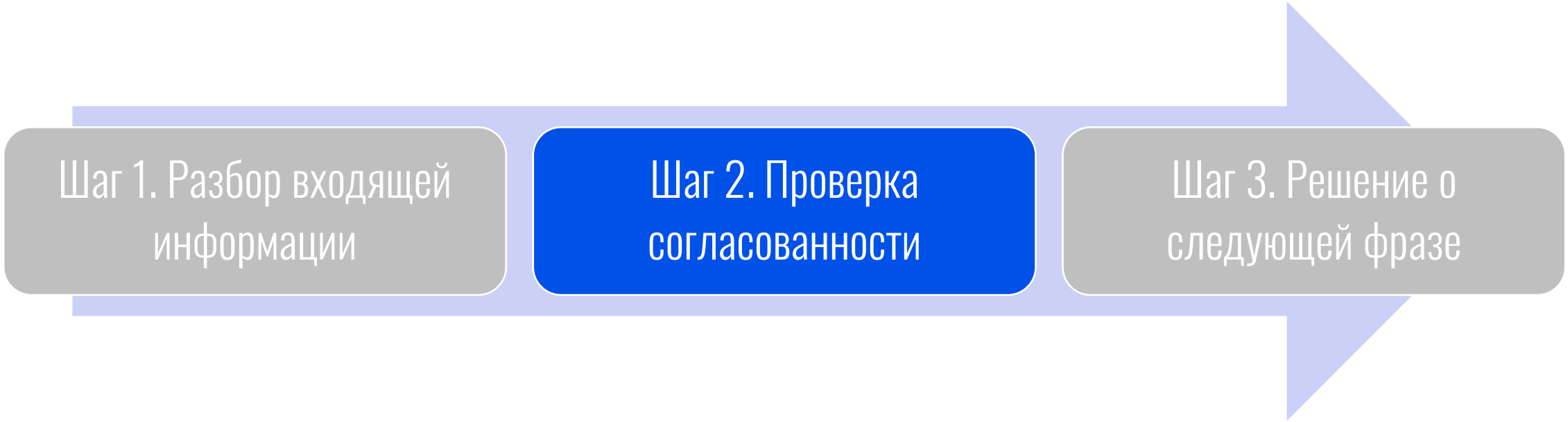
Вопрос клиента: «Можно буррито?» → s(CASP) выводит «нельзя» и предлагает альтернативу.

Как размышляет человек при ведении диалога?



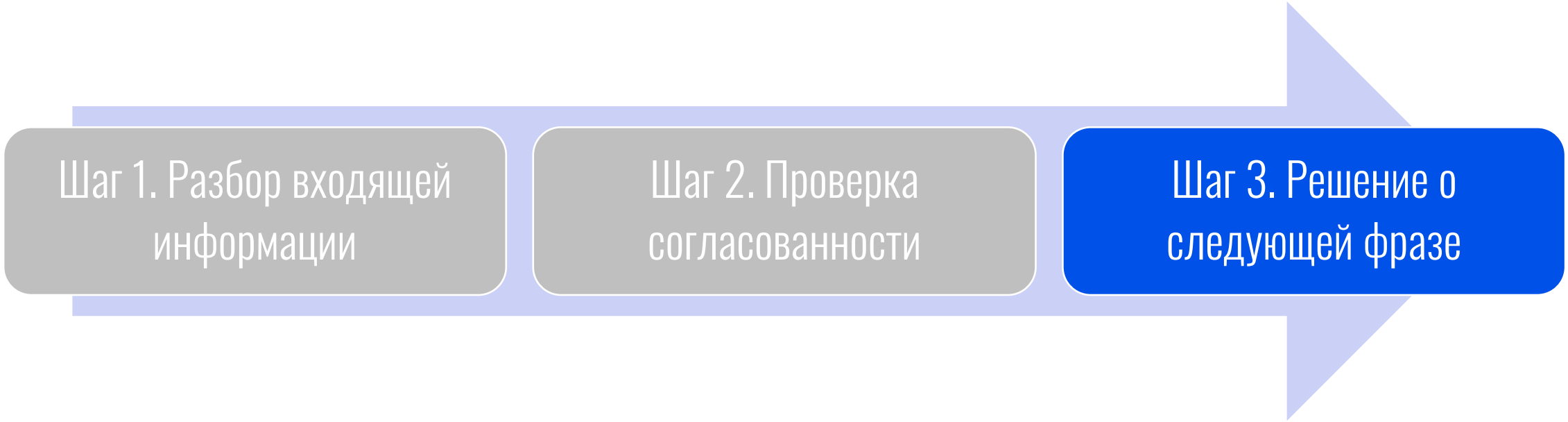
Когда человек слышит предложение, он «разбирает» его, чтобы извлечь смысл, и представляет этот смысл у себя в голове в виде знаний

🧠 Как размышляет человек при ведении диалога?



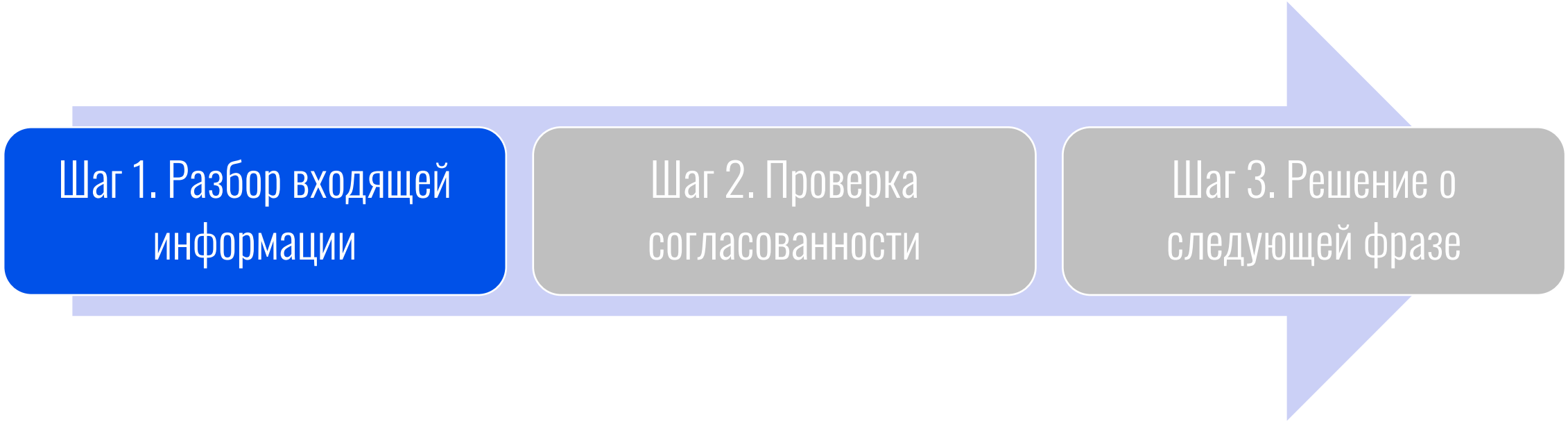
Затем человек проверяет согласованность и корректность этих знаний, используя дополнительные (знания здравого смысла), которые тоже находятся у него в голове.

Как размышляет человек при ведении диалога?



Далее, человек формирует следующую фразу. Если человек находит пробелы в полученных знаниях, то пытается заполнить их, задавая уточняющие вопросы, либо дает ответы/или двигается в диалоге дальше для достижения цели.

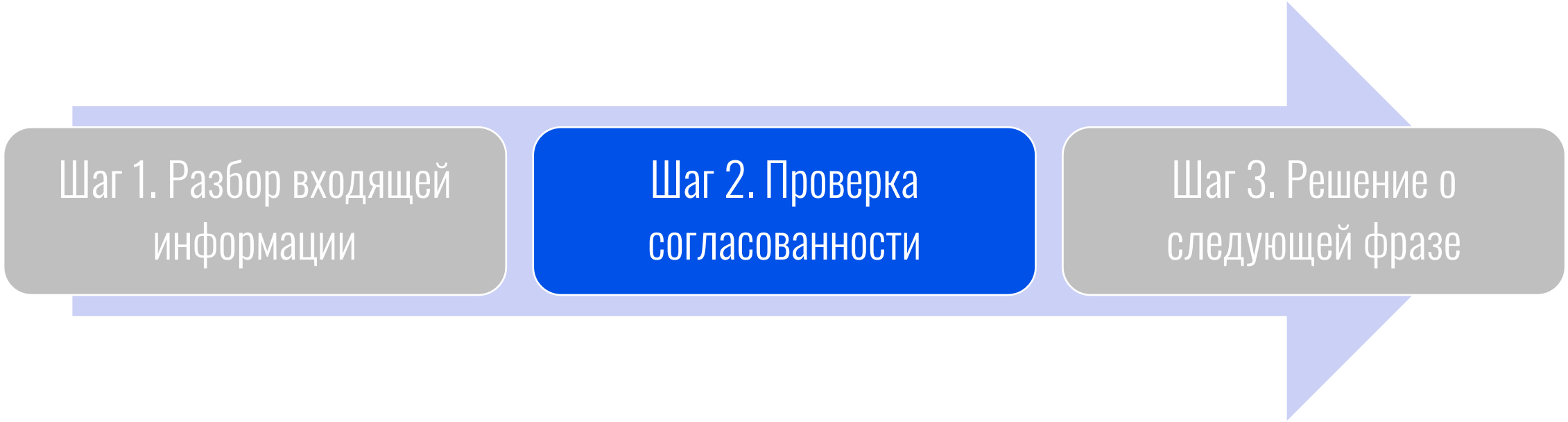
Методика STAR. Совместная работа LLM и ASP



Парсинг (LLM): текст → предикаты

Сначала необходимо преобразовать входной текст на естественном языке от пользователя в знания (представленные в виде заранее определённых логических предикатов) — с этой задачей большие языковые модели (LLM) справляются исключительно хорошо.

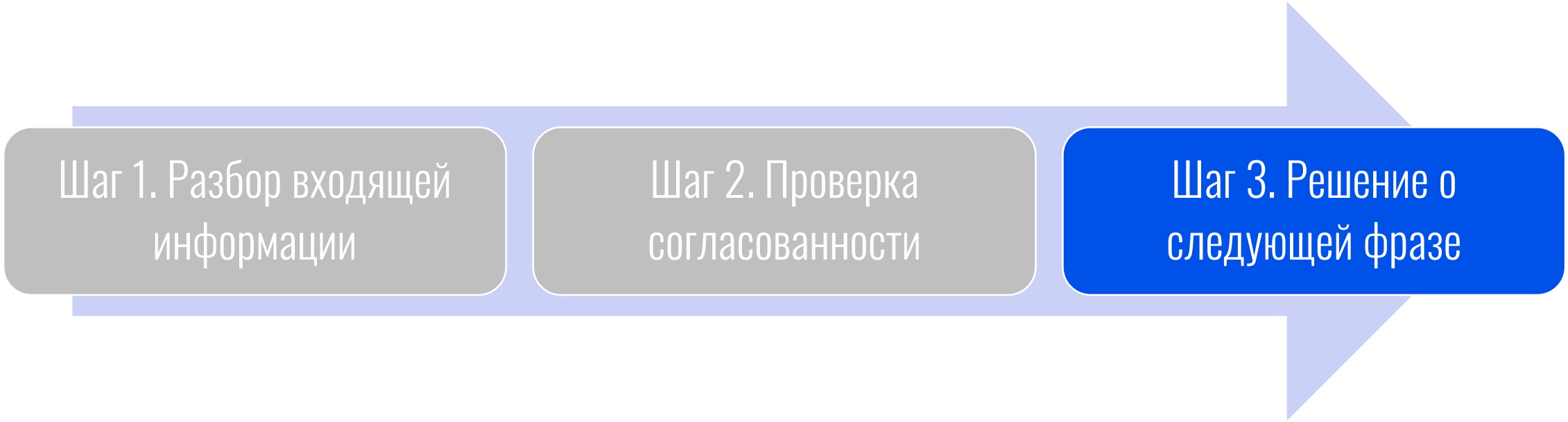
Методика STAR. Совместная работа LLM и ASP



Рассуждение (ASP): проверка + вывод решения

Затем нужно проверить согласованность и корректность полученных знаний и, опираясь на знания, извлечённые из диалога, сделать логический вывод о следующем действии. Эту возможность обеспечивает система рассуждений ASP – s(CASP)

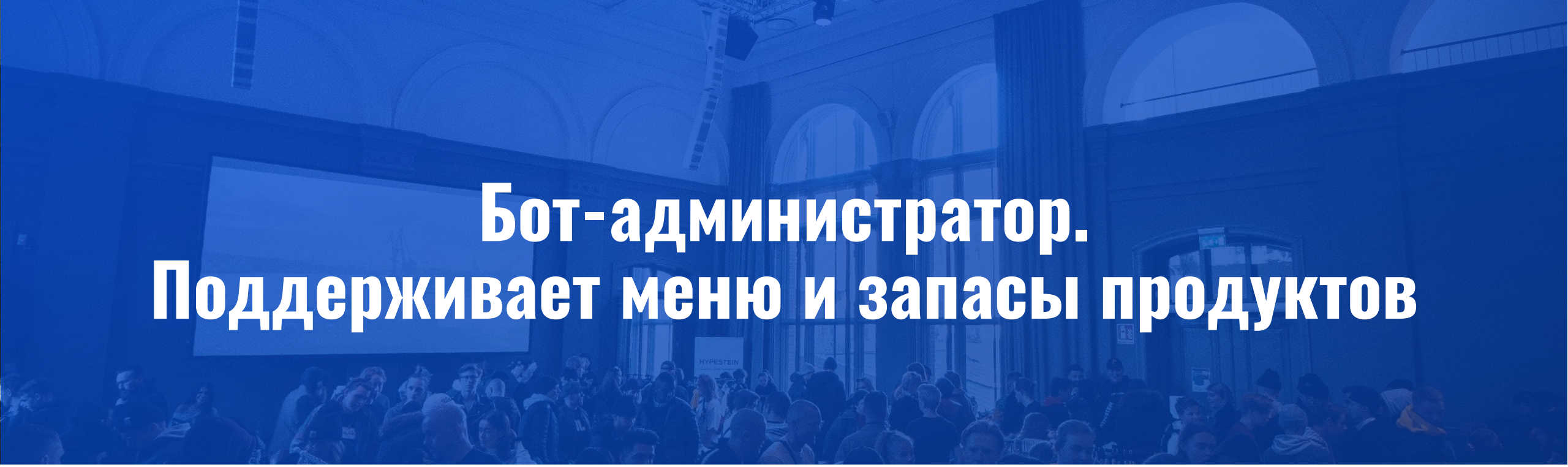
Методика STAR. Совместная работа LLM и ASP



Ответ (LLM): предикаты ⟶ понятный язык

Наконец, знания, представляющие следующий шаг и вычисленные движком ASP, преобразуются обратно в естественный язык и передаются пользователю с помощью другой LLM. Этот цикл повторяется.

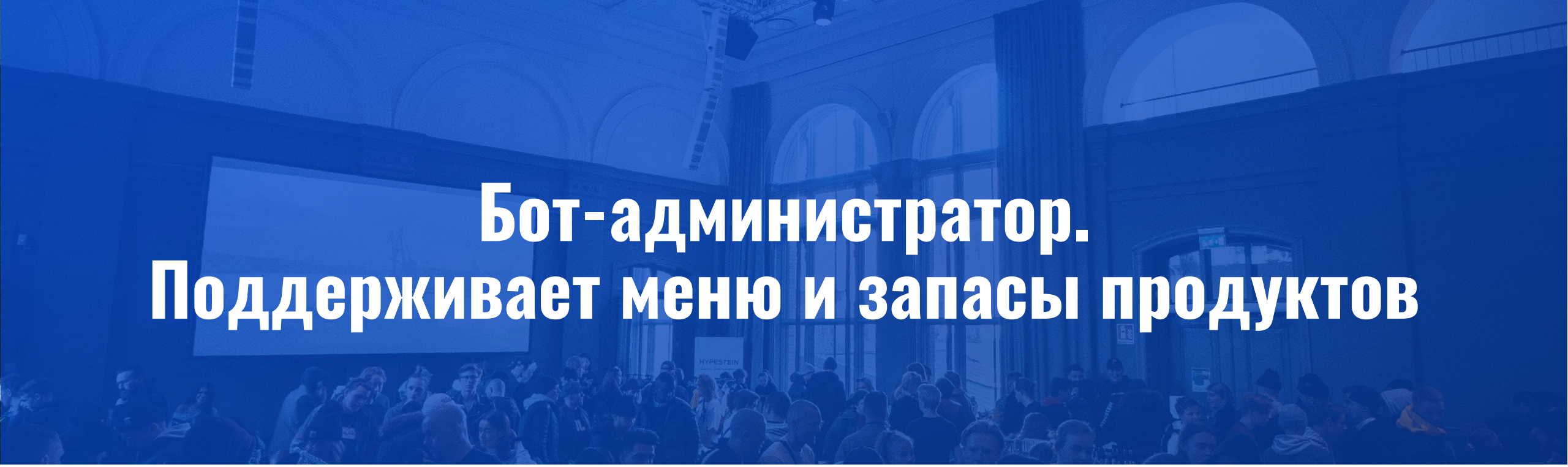
Цикл повторяется: каждый ход уточняет знания и действие



Бот-администратор. Поддерживает меню и запасы продуктов

Общается с сотрудниками/менеджером ресторана и записывает их команды в общую базу знаний. Пользователи могут менять меню — добавлять, удалять или модифицировать позиции — и сообщать о нехватке или восстановлении отдельных ингредиентов.

Все изменения сразу видит другой бот (общая база знаний) — поэтому система остаётся согласованной



Бот-администратор. Поддерживает меню и запасы продуктов

🗂️ Меню и запасы:

Добавляет/удаляет/правит позиции

📜 Свойства блюд:

Цена, калории, состав, популярные топпинги

🚦 Статусы ингредиентов:

Есть, нет, появился

Бот-администратор. Разбор запроса через LLM

Фразы сотрудника «переводятся» LLM ChatGPT-4) в логические предикаты — короткие записи смысла.
Предикаты — это общий язык для всех ботов и правил; с ними легко проверять логику и не путаться.

🧠 Примеры предикатов:

➕ add(dish, 'Grilled Beef Taco') — добавить блюдо
🧩 edit('Crunchy Taco', included_ingredient, 'Cheese', 'Lettuce') — заменить ингредиент
💲 edit('Grilled Cheese Burrito', price, 3.80) — изменить цену
⛔ runout('Lettuce') / ✅ restore('Lettuce') — нет/появился салат
🧹 delete('Sauce Verde') — удалить позицию
🏷️ Служебные: quit/0, irrelevant/0 — выход / не по теме:

Бот-администратор. Уточняющие вопросы. CKT

CKT (Conversational Knowledge Template) — «сценарий разговора» в виде записанной логики состояний на ASP. Если данных не хватает, администратор сам их уточнит, задав вопросы.
CKT гибкий: можно сказать всё одной фразой или давать информацию по частям — бот соберёт всё по шагам.

❓ Примеры авто-уточнений:

- «Уточните категорию блюда (буррито/тако/десерт?)»
- «Какая цена и калорийность?»
- «Какие ингредиенты входят?» / «Какие топпинги рекомендовать?»
- «Для комбо: что включить внутрь?»

🔄 Сигнал к вопросу: бот генерирует ask(Name, Property) и спрашивает ровно то, чего не хватает

Бот-администратор. Генерация ответов

Ответ состоит из подтверждения того, что бот понял, и (при необходимости) вопроса из СКТ.
Подтверждение формируется по распознанным предикатам, а финальная фраза на естественном языке — через LLM.
Так ответы получаются и человечными, и логически корректными.

✅ **Подтверждение (пример): confirm('run out', 'Lettuce') → «Понял, салат закончился.»**

🗣️ **Вопрос (пример): ask('Grilled Beef Taco', price) → «Уточните цену для Grilled Beef Taco.»**

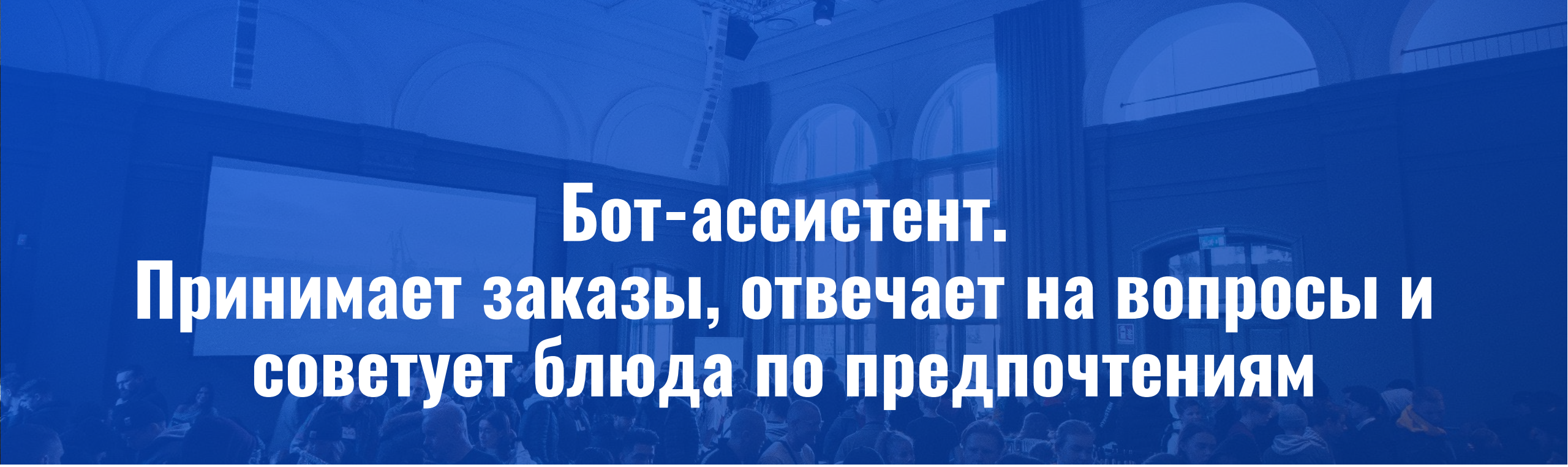
🧩 **Контекст в подсказке:** LLM получает описание предикатов + короткие примеры, чтобы говорить уместно

🛡️ **Надёжность:** логику и согласованность проверяет ASP; LLM только «переводит» смысл

Бот-ассистент. Принимает заказы, отвечает на вопросы и советует блюда по предпочтениям

Общается напрямую с клиентом. Он работает поверх общей базы знаний и соблюдает правила — поэтому не предлагает недоступное и уточняет важные детали. Логика Ассистента сложнее, чем у Администратора, потому что реальные заказы в drive-thru бывают запутанными.

Все изменения сразу видит другой бот (общая база знаний) — поэтому система остаётся согласованной



Бот-ассистент. Принимает заказы, отвечает на вопросы и советует блюда по предпочтениям

🎯 Цель

Быстро и без ошибок оформить заказ

📚 Опора на знания

Общая база + правила

🛡️ Надёжность

Не рекомендует недоступное, предупреждает об ограничениях

🗣️ **Коммуникация:** живой диалог, понятные уточнения

Бот-Ассистент: «понимание» запроса (LLM → предикаты)

Фразы клиента «переводятся» LLM в предикаты — это короткие записи смысла, с которыми работает логика.
Ассистент распознаёт намерения: заказать, уточнить/изменить, завершить заказ, спросить цену/наличие, попросить рекомендацию.

🧠 Примеры предикатов (смысловые шаблоны):

- 🧾 Заказ: order(Food, Number) — «2 Crunchy Taco»
- 🧪 Выбор в комбо: specify(Combo, Dish) — «в набор — Pepsi»
- 🛠️ Изменить позицию: update(Dish, Operation, Option)
  - add / no / less / extra → какой ингредиент добавить/убрать/меньше/больше
  - size → regular/large
- 💡 Рекомендация: need_recommend(Content, Type)
  - Type=category (taco/burrito/vegan/.../all) или Type=upgrade (топпинги для блюда)
- ❓ Вопрос: query(Category, Food) — «цена Veggie Mexican Pizza», «какие есть топпинги?»
- ✅ Завершение: complete/0 — «это всё»:

Бот-Ассистент: стратегия диалога, СКТ, расчёт и ответы

1. Сначала Ассистент слушает и копит заказ; если нужно — даёт рекомендации и отвечает на вопросы.
2. После complete включается СКТ (сценарий-автомат): бот сам доспрашивает недостающее — топпинги к блюду, размер напитка, выбор позиций в комбо (повторяет цикл, если заказано несколько одинаковых блюд).
3. Когда всё собрано, Ассистент считает цену (учитывает апгрейды/добавки) и формирует структурированный итог.

❓ Авто-уточнения (СКТ): ask(Food, Option), ask(Combo, Food, Option)

💵 Прайс: базовая цена + наценки за апгрейды → price(total)

💬 Ответы/Рекоммендации: answer(Food, Property, Value), recommend(Type, Food)

📃 Финал: список позиций с особыми требованиями + итоговая сумма

Взаимодействие ботов: Администратор-Ассистент

Совместимость

- 📚 **Один источник правды:** общее меню и правила
- 🔄 **Мгновенная синхронизация:** обновил — все увидели
- 🛡️ **Меньше сбоев:** нет расхождений между ботами
- 🧩 **Просто масштабировать:** добавляй новые факты/правила

Независимость

- 🎯 **Чёткое разделение ролей:** бэк-офис vs. фронт
- 💼 **Автономность:** каждый решает свою задачу
- 🧱 **Устойчивость:** один не блокирует другого
- 👥 **Разные аудитории:** персонал ↔️ покупатели

Сотрудничество

- 🧠 **Общие правила логики:** предсказуемые ответы
- 🚫 **Фильтр недоступного:** отказ/альтернатива без ошибок
- 💬 **Единый язык предикатов:** точная передача смысла
- 📈 **Лучший клиентский опыт:** меньше путаницы, больше доверия

Обеспечение сотрудничества. LEIFIT

LEIFIT (*Leveraging External Information For Internal Tasks* — «используем внешнюю информацию для внутренних задач»)

Как это работает?

Сценки взаимодействия

T1 Ассистент. Сверка с базой. Всё есть
T2 Админ. Изменение базы. Убрать кетчуп
T3 Ассистент. Сверка с базой. Всё есть, кроме кетчупа
T4 Ассистент. Сверка с базой. Всё есть, кроме кетчупа
T5 Ассистент. Сверка с базой. Всё есть, кроме кетчупа
T6 Ассистент. Сверка с базой. Всё есть, кроме кетчупа

Бот-админ ↔ Сотрудник

- Слушай, у нас кетчупа нет

- Кетчуп закончился
Принято

БЗ: Постоянный статус

БЗ: Временный статус

T1 T2 T3 T4 T5 T6

ОКНО 1
Бот-ассист ↔ Клиент

- Эй, железяка. Нам 2 картохи с кетчупом и квас

- Привет, мешок кожаный. Кетчуп твои сородичи сожрали. Майонез или терияки возьмешь или картохой обойдешься?

- Терияки давай. Оформи заказ уже, терминатор недоделанный

- Заказ принят: 2 картошки с терияки и квас. С тебя 750 р. А терминатор -мой внук. Ешь, пока он не пришел

ОКНО2
Бот-ассист ↔ Клиент

- 2 картошки с кетчупом и квас

- Привет, что-то еще?

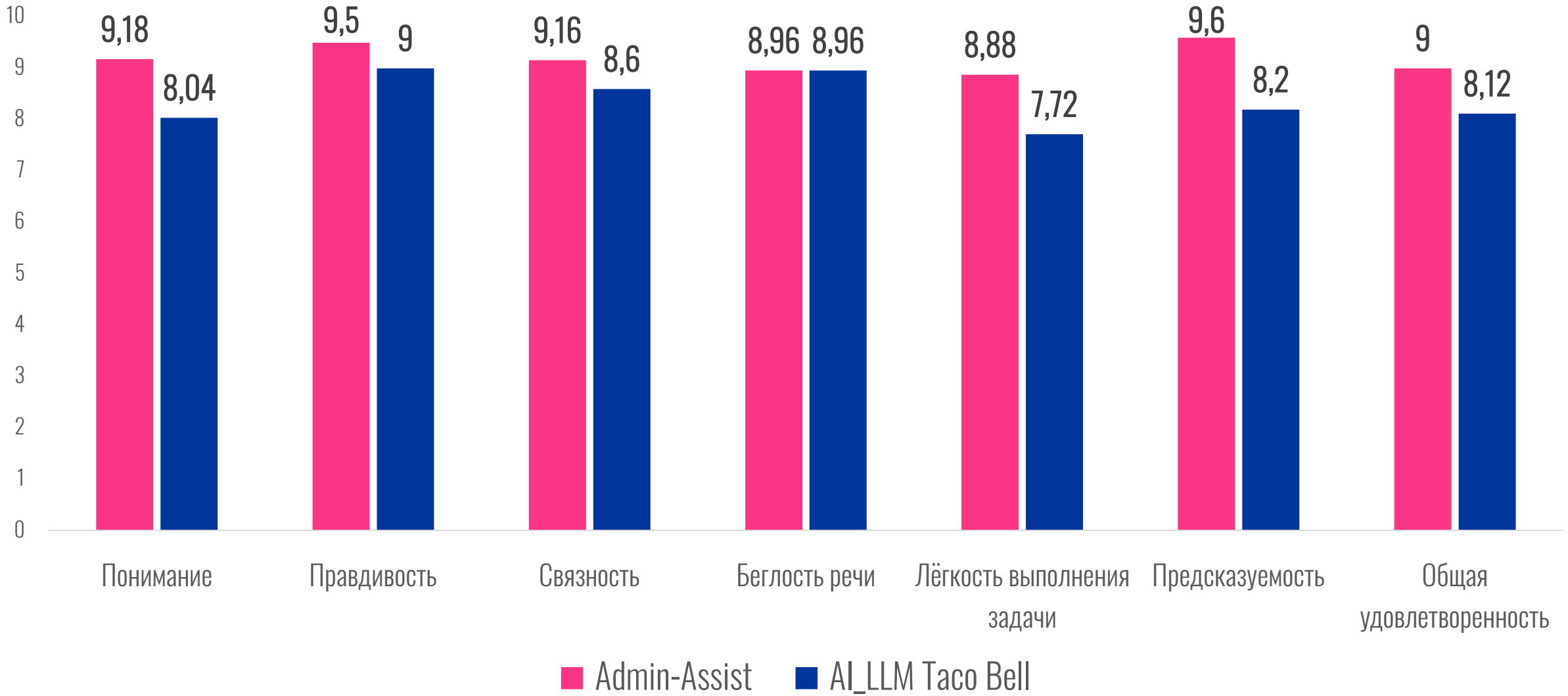
- Нет, спасибо

- Извините, кетчуп закончился. Есть терияки или майонез. Что предпочитаете?

- Давай попробуем терияки

- Заказ принят: 2 картошки с терияки и квас. К оплате 750 р. Приятного аппетита!

Сравнение с AI ботом на LLM Taco Bell



Система MindFlow – первый открытый мультимодальный агент на базе LLM, нацеленный на сферу поддержки клиентов в e-commerce

Обзор

Источник

Название статьи и дата публикации

- MindFlow: Revolutionizing E-commerce Customer Support with Multimodal LLM Agents. 24.03.2025

Ссылка

- https://clck.ru/3Q2k2A

Первый бенчмарк ECom-Bench для оценки мультимодальных LLM-агентов в задачах поддержки клиентов интернет-магазинов

Обзор

Источник

Название статьи и дата публикации

- ECom-Bench: Can LLM Agent Resolve Real-World E-commerce Customer Support Issues?. 24.03.2025

Ссылка

- https://clck.ru/3Q2jy5

СПАСИБО ЗА ВНИМАНИЕ!

apexberg.ru

ТГ-КАНАЛ:
Клиентский сервис –
искусство служить людям