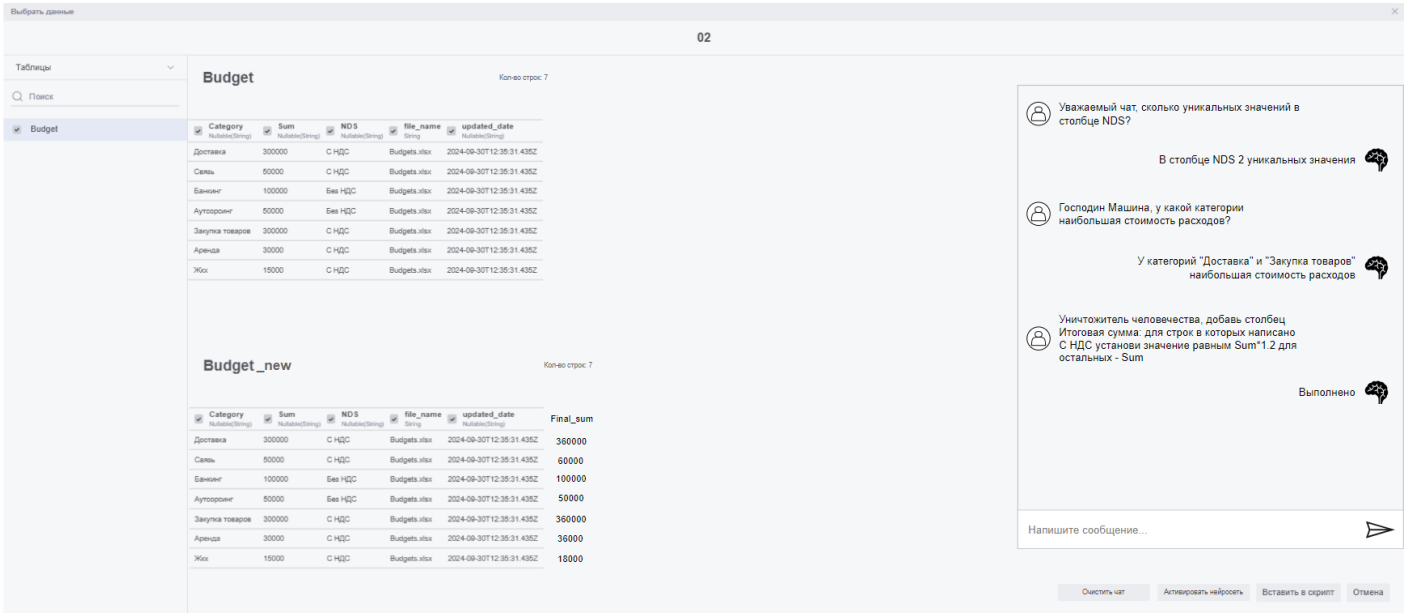
Нейроконнекторы

Цель

Создание инструмента для **Data Discovery** в секции подключения, обеспечивающего удобное для пользователя изучение набора данных и формирование скрипта загрузки без необходимости использовать SQL.

Концепт для интерфейса



- Кнопка для активации/деактивации ИИ модели.
- При активации появляется редактируемая таблица под исходной и разблокируется чат с моделью справа от таблиц. При неактивной модели писать в чат невозможно, редактируемая таблица скрыта.
- Исходная таблица блокируется для изменений на время работы модели.
- Одновременно редактировать можно только одну таблицу, но вопросы остаются по всем таблицам в подключении (в чате могут быть запросы к нескольким таблицам последовательно, но не параллельно).
- В чате можно обмениваться сообщениями с моделью. Если направлено задание на изменение данных, то модель отправляет код для редактирования временной таблицы и даёт ответ вида "Выполнено"/"Невозможно выполнить". Если модели задан вопрос, она даёт полноценный текстовый ответ на основании рассматриваемых данных.

- При деактивации модели (нажатием на кнопку) все предложенные ИИ изменения отменяются, однако чат остаётся и его можно возобновить кнопкой для активации модели. Переписка в чате сохраняется для каждого отдельного подключения.
- Должна быть кнопка, позволяющая очистить чат перед выходом.
- При активной ИИ модели кнопка "Вставить в скрипт" добавляет код для редактируемой нейросетью таблицы, а не для исходной.

Концепт ML

Языковая модель, способная обрабатывать запросы от пользователя и давать ответ либо преобразовывать их в код на языке ClickHouse. Должна уметь распознавать и выполнять следующие типы запросов:

- Поиск в данных: по запросу от пользователя модель должна уметь находить необходимые столбцы, строки и значения, сравнивать их с заданными, запоминать введённые условия, находить максимум, минимум, среднее и т.д. Пример: "Напиши, есть ли в столбце Cost значения больше 20000?".
- Анализ использования: модель должна уметь определять контекст использования данных из источника, назначение отдельных столбцов, область применения (задачи, которые можно решать с помощью данных). Пример: "Объясни, для чего тут столбец Month_num, если есть столбец Month?" или "Достаточно ли этих данных, чтобы проанализировать динамику расходов за 2024 год?".
- Работа с типами данных: модель должна уметь определять и подставлять подходящий тип данных столбца. Пример: "Выбери наиболее подходящий тип данных для столбца Cost" или "Ты неправильно определил тип данных столбца Cost. Смени его на вещественный".
- Обработка данных: модель должна уметь писать код для трансформации таблицы, изменения имеющихся столбцов, создания новых на основании запроса (в т.ч. с использованием агрегирующих функций). Пример: "Добавь столбец Profit как результат выражения (Price - Cost) * Count - 5000" или "Транспонируй текущую таблицу".
- Фильтрация и ограничение: модель должна уметь оставлять в наборе данных только те столбцы, строки и значения, которые удовлетворяют запросу от пользователя. Пример: "Удали все значения больше 20000" или "Оставь только первые 5000 строк".

Задание на разработку ML

При обработке запросов модель должна уметь определять столбцы и таблицы, даже если пользователь совершает ошибки в написании их названий либо пишет их на другом языке.

>

> При обучении модели необходимо учитывать, что организация работы с данными происходит на ClickHouse, поэтому особенно важно обрабатывать особенности этого языка запросов, такие как использование сортировки для индексации и специфичность наименований типов данных.

> **ВАЖНО!** Нейросеть не работает с готовым SQL-кодом или таблицей (не создаётся секции ALTER и запросов на изменение). Её задача - написать код для формирования модели данных как результата выборки из БД ClickHouse по шаблону, подставляя в него фрагменты, позволяющие решить поставленную пользователем задачу.

Вход модели:

Таблицы со всеми столбцами и значениями в них + промпт от пользователя. Задачи от пользователя отправляются последовательно через чат, в то время как исходные таблицы неизменны.

Необходимо учитывать, что число значений в таблице может превышать десятки и сотни миллионов при том, что ограничивать или сэмплировать выборку нельзя (поскольку результат работы модели будет использован в итоговом скрипте), но для демонстрационного варианта отображаются только первые 10 строк.

Модель должна определить решаемую задачу в зависимости от запроса пользователя. Если пользователь неудовлетворен результатом работы модели, то нейросеть должна пересмотреть решаемую задачу.

Поиск в данных ("Что и где находится?")

Не возвращает SQL-кода ни для таблицы-примера, ни для добавления в скрипт загрузки.

Модель должна по запросу от пользователя определить зону поиска в данных (локализовать до столбца, набора столбцов, интервала строк и т.д.) после чего выполнить сам запрос на поиск и написать ответ в чате. Решаемые в рамках данной задачи запросы *могут* включать в себя:

- Поиск конкретного значения (числового или текстового).
  Запрос вида: "Есть ли среди Компаний AVA?".
  Возвращает ответ вида: "{Количество} {Значение} есть/отсутствует в {Зона поиска}" – "AVA есть среди компаний".
- Вывод всех уникальных значений в столбце. При первом запросе модель должна выдавать 10 самых часто встречающихся результатов с припиской "и ещё n", где n: (число уникальных значений)-10. При повторном запросе от пользователя выдаёт все значения через запятую (даже если их 100500).
- Поиск/подсчёт значений из диапазона/в диапазоне (набор строк, элемент строки, диапазон числовых значений).

Запрос вида: "У скольких работников из таблицы зарплата больше 100 000?".
Возвращает ответ вида: "{Количество} {Значение} есть/отсутствует в {Зона поиска}" – "У 28 работников зарплата больше 100 000".

- Поиск агрегированного значения из числа следующих: минимум, максимум, среднее, сумма, разность, произведение, частное, статистические функции (мода, медиана, дисперсия, ско).
  Возвращает предупреждение о больших затратах времени на выполнение операции вида "Этот запрос может выполняться длительное время. Напишите "да", если вы уверены, что хотите получить результат. Иначе можете продолжить чат".
  Запрос вида: "Найди проект с наибольшими затратами".
  Если пользователь напишет "Да", то модель должна выдать ответ вида: "{Запрашиваемое агрегированное значение} в {Зона поиска} – {Результат работы модели}" – "Проект с наибольшими затратами – AVA". Иначе продолжает отвечать на другие запросы.
- Другие запросы на поиск в данных.

Вид ответа может меняться в зависимости от сути вопроса пользователя. Модель должна определять последовательность логики запроса (когда пользователя интересует показатель, а когда - его аналитический разрез). Например, ответом на вопрос: "В каком городе наибольшее число жителей?" должно быть название города, а результатом поиска по запросу: "Найди наибольшее число жителей среди представленных городов" – число жителей.

Анализ использования ("Зачем и где применяется?")

Не возвращает SQL-кода ни для таблицы-примера, ни для добавления в скрипт загрузки.

Модель должна по запросу от пользователя определить и описать в ответе в чате контекст применения набора данных: решаемую задачу, экономическое применение, назначение столбцов.

> Для решения этой задачи необходимо анализировать не только метаданные (названия таблиц и столбцов), но и содержимое этих таблиц.

- Решаемая задача – ответ на вопросы вида: "Что я могу сделать с этими данными?".
  Модель должна определять спектр задач по типу: "Вы можете использовать эти данные для определения финансовых показателей компании за 2024 год".
- Экономический контекст – ответ на вопросы вида: "Где используются эти данные?".
  Модель должна определять сферу применения по типу: "Предположительно, эти данные используются в сфере продаж".
- Назначение столбцов – детализация решаемой задачи в области применения отдельных столбцов. Ответ на вопросы вида: "Зачем мне нужен этот столбец?".
  Модель должна определять назначение столбцов по типу: "Это вспомогательный столбец, который нужен для организации корректной сортировки".
- Другие запросы на выявление области применения данных.

Работа с типами данных

Типы данных

Модель должна уметь:

- Определять текущий тип данных указанного пользователем столбца. Эта информация передаётся вместе с метаданными, поэтому для решения этой задачи необходимо лишь передать в ответе имеющееся значение.
- Определять наиболее подходящий тип данных. Причём самый простой из подходящих (не выбирать словари или LowCardinality). Для решения этой задачи модель должна обрабатывать имеющиеся в столбце данные и на их основании предлагать нужный тип.
- Менять тип данных. Менять можно только в том случае, если значения столбца можно хранить в конечном типе данных. Модель должна уметь переводить данные как в конкретный тип ("Измени тип данных столбца Доход на вещественный"), так и в подходящий ("Измени тип данных столбца Расход на наиболее подходящий"). Измененный тип данных попадает в итоговый скрипт загрузки.

Обработка данных

Модель должна уметь изменять имеющиеся столбцы, создавать новые, сортировать по указанным столбцам.

- Изменение столбцов – модель записывает выражение на языке ClickHouse для текстового запроса от пользователя на редактирование существующего столбца. В качестве выражения может выступать комбинация арифметических и агрегирующих операций над столбцами, числами и текстом. Результат – вместо передачи названия существующего столбца в скрипт загрузки передаётся это выражение.
  Пример: "Сделай так, чтобы в столбце Сумма результат выводился в миллионах, а не в 1000".
  Результат (в Select): toFloat32("Sum")/1000.
- Создание столбцов – модель записывает выражение на языке ClickHouse для текстового запроса от пользователя на формирование нового столбца. В качестве выражения может выступать комбинация арифметических и агрегирующих операций над столбцами, числами и текстом. Результат – после списка с названиями существующих столбцов добавляется это выражение.
  Пример: "Создай столбец Дельта как разность между доходами и расходами".
  Результат (в Create): "Delta" Int32.
  Результат (в Select): toInt32("Income")-toInt32("Cost").
- Переименование столбцов – в секции CREATE вместо существующего имени столбца модель добавляет введенное пользователем.
- Сортировка столбцов – модель добавляет секцию ORDER BY согласно запросу от пользователя.

Фильтрация и ограничение

Модель должна уметь конструировать запросы для секций WHERE, LIMIT и OFFSET:

- Ограничение по условию – модель формирует секцию WHERE: интерпретирует текст от пользователя как выражение, содержащее оператор сравнения, принадлежности или шаблон для столбцов, чисел и текста.
  Пример: "Оставь только те значения в столбце Доход, которые больше 500 000"
  Результат (в Where): toInt32("Income")>500000
- Ограничение по числу значений:
  - "Оставь только первые N значений" – модель добавляет секцию LIMIT N.
  - "Оставь последние N значений" – модель добавляет секцию OFFSET N.
  - "Оставь значения, начиная с N и заканчивая M" – модель добавляет секции LIMIT N и OFFSET M.

> Если пользователь пытается ввести запрос, не имеющий отношения к формированию скрипта загрузки или изучению данных, то модель должна выдать ответ вида: "Я могу помочь только с формированием скрипта загрузки или изучением данных.
> Пожалуйста, уточните Ваш запрос или составьте новый таким образом, чтобы он имел отношение к представленным данным!"

Шаблон кода для скрипта загрузки (выход модели)

```
Table "Название таблицы"

Create @@@ -- шаблонная секция, подставляются из модели только названия столбцов и их типы данных
CREATE TABLE IF NOT EXISTS "Название таблицы"(
  "Название столбца 1" type, -- тип данных, если допускается null, то Nullable(type) // ЗАДАЧА НА РАБОТУ С ТИПАМИ ДАННЫХ
  "Название столбца 2" type, -- название столбца может быть изменено моделью
  ...
  "Название столбца n" type
) ENGINE = MergeTree ()

ORDER BY
  tuple ()
@@@

Delete @@@ -- ещё одна шаблонная секция
  ALTER TABLE "Название таблицы" DELETE WHERE 1=1
@@@
```

```
Source "Название подключения"

Read @@@ -- основная секция для обработки моделью
SELECT
  "Название столбца 1", -- перенос столбцов из из источника
  "Название столбца 2",
  ...
  "Название столбца k",
  Выражение для изменения столбца 1, -- ЗАДАЧА НА ИЗМЕНЕНИЕ СТОЛБЦОВ
  Выражение для изменения столбца 2,
  ...
  Выражение для изменения столбца m,
  Выражение для нового столбца 1 as "Название нового столбца 1", -- ЗАДАЧА НА СОЗДАНИЕ НОВЫХ СТОЛБЦОВ
  Выражение для нового столбца 2 as "Название нового столбца 2",
  ...
  Выражение для нового столбца n as "Название нового столбца n"
FROM
  "Название таблицы"
WHERE
  toType("Название столбца k") Условие 1 -- для каждого столбца, используемого в WHERE, у которого изменился тип данных, необходимо указывать тип из Create (например, для типа Int32 будет toInt32 и т.д.) // ЗАДАЧА НА ОГРАНИЧЕНИЕ
  ...
ORDER BY
  "Название столбца k" тип сортировки -- ЗАДАЧА НА СОРТИРОВКУ
  ...
LIMIT a
OFFSET b -- ЗАДАЧА НА ОГРАНИЧЕНИЕ

Optimize @@@ -- шаблонная секция на удаление дубликатов
OPTIMIZE TABLE "Название таблицы" DEDUPLICATE
@@@
```

Пример выхода модели для скрипта загрузки

```
Table "Costs"

Create @@@
```

```
CREATE TABLE IF NOT EXISTS "Costs" (
  "Date" Date,
  "Sum" Int32,
  "Product_name" Nullable (String),
  "Category" Nullable (String),
  "file_name" Nullable (String),
  "new_object" Int32
) ENGINE = MergeTree ()

ORDER BY
  tuple ()
@@@

Delete @@@
ALTER TABLE "Costs" DELETE WHERE 1=1
@@@

Source "01"

Read @@@
SELECT
  "Date",
  "Sum",
  "Product_name",
  "Category",
  "file_name",
  toInt32("Sum")+666
FROM
  "Costs"
WHERE
  toInt32("Sum")>5000
ORDER BY
  "Product_name" desc
LIMIT 10
OFFSET 5
@@@

Optimize @@@
OPTIMIZE TABLE "Costs" DEDUPLICATE
@@@
```

Выход модели в секции подключения:

- Ответ языковой модели в чате (либо "Выполнено/Не выполнено", если поступил запрос на обработку данных).
- Код для изменения выборки из 10 строк для предпросмотра (СОСТЫКОВАТЬСЯ С БЭКОМ!)

"Память" модели

Для каждого подключения нейросеть хранит собственную "Память", включающую в себя по меньшей мере историю переписки с пользователем и код для изменения выборки из 10 строк для предпросмотра. Кроме того, существует один из трёх вариантов (обсуждаемых с заказчиком), в зависимости от которого в памяти модели может храниться:

☐ Ничего – модель обращается напрямую к базе ClickHouse. Этот вариант возможен в том случае, если в секции подключения останется только редактор скрипта, а инструмент Data Discovery будет выделен в отдельную задачу.

☐ Кэшированную таблицу объёмом до 1 млн строк

☐ Всю таблицу целиком

Задание на разработку для фронт- и бэкенда

Расположение элементов (см. **Концепт**):

- Список доступных таблиц (вместе с поисковой строкой), исходная таблица и кнопки "Вставить в скрипт" и "Отмена" остаются на своих местах за тем исключением, что ширина исходной таблицы ограничивается в 70% от свободного места в окне для таблицы. При достижении этой границы снизу должен появляться ползунок для прокрутки.
- Подпись "Кол-во строк" перемещается к названию таблицы в левую часть окна.
- Вся правая часть окна до его границы будет занята чатом с нейросетью. Расстояние между чатом и таблицей должно составлять хотя бы 5% от ширины окна.
- Под исходной таблицей будет располагаться редактируемая (созданная нейросетью) таблица, равная по высоте исходной (ограничение на отображение первых 10 строк).
- Кнопки "Очистить чат" и "Активировать нейросеть" располагаются рядом с кнопками "Вставить в скрипт" и "Отмена".

Передача данных:

☐

Вариант 1 – передача всех данных

При нажатии на кнопку "Активировать модель" запускается **асинхронная** передача данных из источника в модель ИИ: сначала отправляется набор метаданных (название таблицы и столбцов, типы данных, количество строк), потом – сами значения из таблицы. Поскольку размер таблицы может быть достаточно велик, подобная операция может занимать длительное время.

Сценарий взаимодействия с чатом во время загрузки данных:

- Началась загрузка. Если спустя 2 секунды загрузка не завершается, то в чате появляется сообщение: "Инициализирована загрузка данных в модель. Вы можете писать запросы в чат до окончания загрузки, но обработка содержимого столбцов таблицы будет недоступна".
- Пользователь деактивировал модель ДО окончания загрузки (
- Выход из окна выбора данных нажатием на кнопку "Отмена" или крестик считается деактивацией модели).

  ☐ Вариант 1. Для данного подключения модель сохраняет скачанные данные и брейкпоинт в последней скачанной строке. После повторной активации модели скачивание данных возобновляется с точки остановки, а в чате моментально выдаётся сообщение: "Возобновлена загрузка данных в модель. Вы можете писать запросы в чат до окончания загрузки, но обработка содержимого столбцов таблицы будет недоступна".

  ☐ Вариант 2. Выдаётся окно для подтверждения деактивации модели с предупреждением: "В случае деактивации в следующий раз процесс загрузки придётся начинать сначала!". При подтверждении модель деактивируется, передача данных прекращается, скачанные данные удаляются.
- Пользователь написал запрос на работу с данными (не метаданными) ДО окончания загрузки. Модель выдаёт ответ: "Для обработки Вашего запроса необходимо скачать данные из таблицы. Пожалуйста, дождитесь окончания загрузки!"
- Пользователь деактивировал модель ПОСЛЕ окончания загрузки.

  ☐ Вариант 1. Для данного подключения модель сохраняет скачанные данные в кэшированной таблице. Плюс: После повторной активации не выдаётся никаких сообщений, пользователь может писать запросы. Минус: необходимо хранить скачанные данные.

  ☐ Вариант 2. Выдаётся окно для подтверждения деактивации модели с предупреждением: "В случае деактивации в следующий раз процесс загрузки придётся начинать сначала!". При подтверждении модель деактивируется, скачанные данные удаляются. Плюс: не нужно дополнительно хранить огромные массивы данных. Минус: затраты времени на повторное скачивание данных.

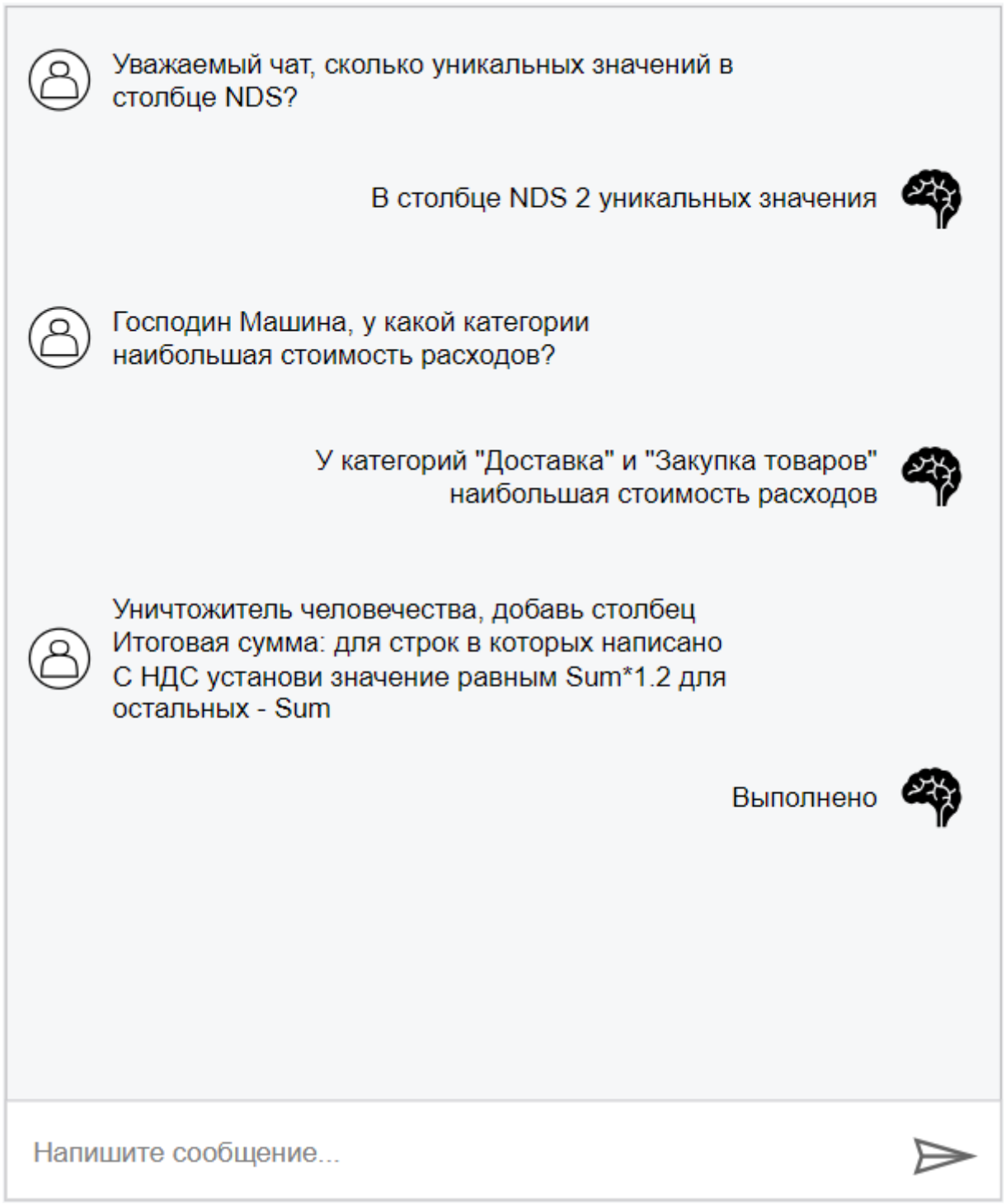
Вариант 2 – Передача выборочной совокупности

При нажатии на кнопку "Активировать модель" запускается сэмплирование (или ограничение с помощью Limit - НЕ РЕКОМЕНДУЕТСЯ!) исходной таблицы до 1 000 000 строк. Данная операция не занимает длительного времени, переданная таблица кэшируется.

> Данный вариант покрывает 95% задач, решаемых новой моделью. Невозможно будет обрабатывать некоторые поисковые запросы, например, вывод точного числа уникальных значений, или поиск одного конкретного значения, не попавшего в сэмплированную выборку. При возможности пренебречь данными задачами вариант 2 является наиболее предпочтительным.

Чат с моделью

Выглядит как переписка. Работает как переписка. В общем, это переписка. С нейросетью.



Особенности:

- В данном чате нет необходимости удалять или изменять сообщения, поскольку каждый запрос чат обрабатывает отдельно и может выдавать различные ответы (например, при анализе использования). Сами сообщения в чате можно выделять и копировать содержимое, а потом вставлять в запрос.
- На каждый запрос модель выдаёт только один ответ. Давать ссылку на ответ нейросети кнопкой "переслать" нельзя – модель запоминает собственные ответы и может отвечать на контекстные вопросы к предыдущим обсуждениям в рамках одного подключения.

- В текстовой строке ввода запроса нельзя прикреплять фото, видео, документы и т.п.
- Пока не нажата кнопка "Активировать нейросеть" строка для ввода сообщения недоступна пользователю (однако история переписки сохраняется).

Кнопки и обработки

Параллельные обработки

> При активированной нейросети водить изменения в исходную таблицу вручную ЗАПРЕЩАЕТСЯ – все поля становятся некликабельными до окончания работы с моделью.

> Вносить вручную изменения пользователь может только в текущую редактируемую таблицу, которая находится под исходной и появляется в момент активации нейросети.

Кнопка "Активировать нейросеть"

При нажатии на кнопку разблокируется чат и появляется редактируемая таблица, а сама кнопка меняет название на "Деактивировать нейросеть". При деактивации нейросети появляется диалоговое сообщение вида: "При деактивации результат обработки таблицы нейросетью не будет добавлен в скрипт загрузки. Если Вы хотите использовать обработанную таблицу, нажмите на кнопку "Вставить в скрипт" при активной модели" с вариантами ответа "Деактивировать" и "Вернуться".

После деактивации блокируется чат и исчезает редактируемая таблица. История переписки сохраняется.

Кнопка "Вставить в скрипт"

При активной нейросети нажатие на эту кнопку передаёт в скрипт загрузки SQL-код, сформированный моделью нейросети.

Кнопка "Очистить чат"

При нажатии на кнопку появляется диалоговое сообщение вида: "Подтвердите, что Вы хотите очистить чат с моделью" с вариантами ответа "Да/Нет". При подтверждении удаляются все сообщения из чата с моделью и отправляется сигнал об очистке памяти модели для текущего подключения.

Данная кнопка доступна как при активированной, так и при деактивированной модели.

Редактируемая таблица

Под исходной таблицей при активированной модели располагается редактируемая таблица. Взаимодействие с ней должно быть полностью аналогично исходной таблице: можно отмечать используемые колонки вручную, менять тип данных (в будущем). Все вносимые вручную изменения передаются в формируемый нейросетью SQL-код для создания подключения. В редактируемую таблицу могут добавляться новые столбцы в зависимости от работы модели. Для редактируемой таблицы также подписывается кол-во строк.

Revision #3
Created 25 November 2024 13:10:21 by Артём
Updated 27 November 2024 12:39:17 by Артём