Описание эксплуатации программного обеспечения для электронно-вычислительных машин Arenadata Advanced RAG (ARAG)

Москва
2025

Содержание:

1 Термины и определения

Термин	Значение
ANSI SQL	Стандартный язык запросов к базам данных, разработанный Американским национальным стандартом (ANSI)
LLM-модели	Языковые модели, обученные на больших объемах текстовых данных

2 Сокращения и обозначения

Сокращение	Наименование
API	(Application Programming Interface) – набор классов, процедур, функций, структур или констант, которыми одна компьютерная программа может взаимодействовать с другой программой.
CPU	(англ. – Central Processing Unit) – центральный процессор
DDL	Data Definition Language
DML	Data Manipulation Language
IP	Маршрутизируемый протокол сетевого уровня стека TCP/IP.
MSSQL	Microsoft SQL Server
LDAP	(англ. – Lightweight Directory Access Protocol) – протокол прикладного уровня для доступа к службе каталогов
LLMs	Large language models
SSL	Secure Sockets Layer
SQL	(англ. – Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционных базах данных.
UI	User interface
БД	База данных
ГБ	Гигабайт
ОС	Операционная система
ПО	Программное обеспечение
СУБД	Система управления базами данных
ТБ	Терабайт

3 Введение

Arenadata Advanced RAG (ARAG) – это продвинутая платформа смыслового поиска (Retrieval Augmented Generation), которая помогает ИИ и людям находить точные ответы на специфичные для компании вопросы с учетом внутренних данных, регламентов и специфики отрасли.

ARAG сочетает различные стратегии векторного поиска, обеспечивая максимальную полноту и точность извлечения знаний. Модульная архитектура позволяет выборочно применять наиболее подходящие инструменты обработки информации для групп документов, отдельных источников данных или целых доменов знаний, а также выбирать наиболее подходящие для задачи ИИ-модели. Платформа спроектирована для интеграции в корпоративные среды и масштабируется в зависимости от бизнес-сценариев.

Основные функции продукта:

- Семантический поиск – находит текст, близкий по смыслу к запросу, даже если нет точного совпадения ключевых слов;
- Поиск релевантных документов – извлекает информацию из внешних источников;
- Фильтрация и ранжирование – отбирает наиболее подходящие фрагменты текста по релевантности;
- Поддержка разных форматов – работает с файлами разных форматов, имеет возможность интеграции с источниками данных по API;
- Контекстно-зависимые ответы – использует найденные документы для генерации точных и развернутых ответов;
- Обобщение информации – может суммировать длинные документы или несколько источников;
- Ссылки на источники – сохраняет мета-данные исходного документа;
- Мультиязычность – поддерживает поиск и генерацию на разных языках.

4 Интерфейсы взаимодействия

Техническая консоль позволяет выполнять как загрузку, так и поиск (см. рис.1).

При загрузке новых документов, они загружаются в указанный домен. При поиске - поиск будет осуществлен в рамках указанного домена. Если домен не указан, поиск осуществляется по домену, найденному с помощью алгоритма на основе ключевых слов.

При поиске на панели Pipeline Stages отображается техническая информация о всех метаданных о чанках.

адрес UI http://<host>:8501/

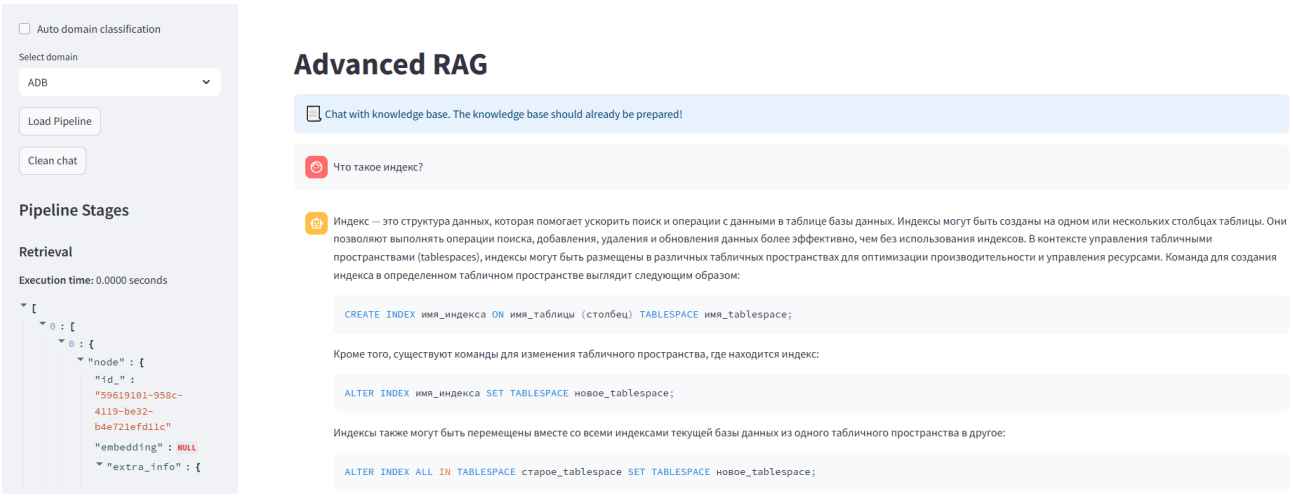

Рис.1 – Техническая консоль ARAG

5 API

Реализованы следующие endpoint:

GET /ingestion_run/{domain_name} - запускает процесс загрузки данных в векторную БД, извлекает словарь с описанием результата. Параметр: domain - Имя домена.

POST /get_llm_result/ - ответ от LLM.

POST /get_chunks_result/ - возвращает список чанков. Работает в двух режимах обработки результатов поиска чанков (debug, info). В режиме debug

возвращается содержимое чанка, его метаданные и score. В режиме info (режим по умолчанию) возвращается только содержимое чанка.

POST Get Search Result - возвращает отранжированный список чанков для поиска.

Classifier

POST /domains_load - удаляет существующие домены и загружает информацию о новых доменах в векторную базу данных.

GET /get_domains - извлекает информацию о доменах из векторной базы данных. Возвращает список доменов.

GET /get_domain/{domain_name} - извлекает указанную в параметре информацию о имени домена из векторной базы данных.

6 Загрузка документов

Поддерживаемый источники документов:

- Интеграция с одним из самых распространенных вики-решений;
- Локальная папка с файлами.

В параметре loader.selected_loader конфигурации указывается источник confluence_loader или simple_directory_reader.

Загрузка документов может быть осуществлена двумя способами:

- через UI
- curl

При загрузке новых документов указывается домен, в который они будут загружены.

7 Возможности загрузки spaces

- Поддержка конфигурации со списком пространств;
- Возможность указания списка пространств в конфигурационном файле. При наличии списка загружаются только указанные пространства;

- Массовая загрузка пространств;
- Возможность получения всех пространств через API-запрос;
- Итеративная загрузка всех доступных пространств и соответствующих страниц/данных.

Поддержка логики обновления:

- Идентификация обновленного документа:
    - список id страниц, измененных за последние сутки статей
    - список id страниц, вновь созданных за последние сутки статей
    - список id страниц, удаленных за последние сутки статей
- Поиск и удаление связанных чанков из векторной БД (маппятся по id статей id родительских чанков и соответственно id дочерних чанков. Получаем полный список всех чанков, по которым произошли изменения за последние сутки)
- Повторная обработка и загрузка документа (только для метода DenseX старые чанки удаляются, формируются новые чанки по id страниц).

Все ошибки загрузки фиксируются стандартным способом (через текущую систему логирования), без дополнительных механизмов обработки.

В случае падения Docker-контейнера или критической ошибки, после восстановления система не продолжает загрузку.

Восстановление выполняется вручную: оператор проверяет последнее успешно загруженное пространство и вносит нужный список в конфиг.

В случае ошибки при загрузке пространства или страницы, они пропускаются, и начинается обработка следующего доступного объекта.

Поддерживаются следующие параметры:

- confluence_url: URL-адрес пространства
- user_name: Имя пользователя

- api_token: токен API
- space_key: идентификатор пространства
- limit: ограничение на количество загружаемых страниц
- page_ids: Список идентификаторов страниц.

8 Настраиваемые параметры

Для управления конфигурацией программного продукта используется config manager. Поддерживаются как локальные эмбеддинг модели, так и эмбеддинг по API.

1. Конфигурируются следующие параметры векторной базы данных:
    - local_persist_directory: Путь к данным Qdrant для локального (несерверного) режима.
    - url: URL-адрес сервера базы данных, например, http://qdrant:6333
    - .collection_name: Имя коллекции.
    - batch_size: Размер пакета для загрузки векторов.
2. Конфигурирование используемых моделей:
    - model_name: Название модели.
    - base_url: Базовый URL-адрес сервера модели.
    - token: токен доступа для модели.
    - temperature: контролирует степень разнообразия реакций (от 0 до 1).
    - presence_penalty: штраф за использование одинаковых токенов.
    - max_tokens: максимальное количество токенов, которые будут возвращены.
    - timeout: время ожидания запроса в секундах.

3. Параметры Dense Embeddings:
    - selected_dense_embedding: указывается используемый бэкэнд. Должен соответствовать одному из ключей ниже dense_embeddings (например, local или text_embeddings_inference).
    - local: запускает модель локально в процессе приложения.
        - model_name: Имя модели для локальной загрузки, например, sentence-transformers/all-MiniLM-L6-v2.
        - device: Устройство для исполнения (cpu или cuda).
    - text_embeddings_inference: Использует удаленный сервер вывода на основе Hugging Face text-embeddings-inference.
        - model_name: Имя модели, настроенной на удаленном сервере, например, sentence-transformers/all-MiniLM-L6-v2.
        - base_url: URL-адрес работающего text-embeddings-inferenceсервера, например, http://host.docker.internal:8080.
        - timeout: время ожидания запроса в секундах.
        - embed_batch_size: Максимальное количество текстов для встраивания в один запрос.
4. Параметры Sparse Embeddings:

- model_name: Модель для разреженных вложений. В настоящее время доступно только Qdrant/BM25

5. Настраиваются следующие параметры Chunking:
    - llm: LLM, используемый для обработки фрагментации текста. Доступные варианты: llm или small_llm.
    - chunk_size: максимальный размер одного текстового фрагмента.
    - chunk_overlap: Перекрытие между фрагментами текста.

- selected_type: выбранный(е) тип(ы) интеллектуального разбиения на фрагменты, например, ["raptor", "densex"].
    - Если выбрано несколько типов (например, ["raptor", "densex"]), разбиение на фрагменты и извлечение будут выполняться с использованием этих методов.
    - Если выбран только один тип (например, ["raptor"]), разбиение на фрагменты и извлечение будут выполняться исключительно с использованием этого метода.

6. Параметры Raptor:
    - tree_depth: Глубина дерева кластеризации.
    - threshold: Порог вероятности принадлежности к кластеру. Параметр threshold в функции GMM_cluster контролирует, насколько «уверенно» точка должна принадлежать кластеру. Увеличение threshold приведет к тому, что точки с меньшей вероятностью будут отнесены к какому-либо кластеру, что может сократить общее количество кластеров.
    - max_length_in_cluster: Максимальная длина текста в кластере. Еще один параметр для управления количеством документов в кластере — на основе общей длины. Если кластер превышает эту длину, он разделяется.
    - context_window: Окно контекста. Если параметр max_length_in_cluster превышает максимальное окно контекста LLM, то суммирование кластера будет выполняться рекурсивно (разделяя кластер на размеры context_window и суммируя каждый полученный фрагмент) - в конечном итоге приводя к суммированию сумм.
    - num_workers: Количество потоков для обработки.

- summary_prompt: промт для кластера.

7. Параметры Densex:
    - num_workers: Количество потоков.
    - propositions_prompt: Промт для генерации предложений.

Пример конфигурации:

```
chunking:
llm: small_llm
chunk_size: 512
chunk_overlap: 50
selected_type: raptor
type:
  raptor:
    tree_depth: 3
    threshold: 0.1  #Параметр threshold в функции GMM_cluster контролирует,
    насколько "уверенно" точка должна принадлежать кластеру.
    Увеличение threshold приведет к тому, что точки с меньшей вероятностью будут отнесены
к какому-либо кластеру,
    что может уменьшить общее количество кластеров.
    max_length_in_cluster: 100000
    context_window: 30000
    num_workers: 10
    summary_prompt: >
      As a professional referee, write a concise and comprehensive summary of the provided
text in as much detail as possible.

  densex:
    num_workers: 5
    propositions_prompt: >
     #указывается промт
```

- Используемая модель указывается в конфигурационном файле, ее настройка - на стороне пользователя.

Retriever / reranker/ Экстракторы

8. Настраиваемые параметры reranker:
    - top_n: Количество результатов для ранжирования. Конечное количество релевантных документов.
    - rerank_model: Модель для переоценки. В настоящее время доступна только colbert-ir/colbertv2.0.

- score_threshold: Минимальный балл, который должен получить документ, чтобы считаться релевантным после переоценки. Документы с баллами ниже этого порога будут отброшены.

9. Настраиваемые параметры retriever:
    - similarity_top_k: контролирует конечное количество возвращаемых узлов.
    - sparse_top_k: показывает, сколько узлов будет извлечено из каждого плотного и разреженного запроса.
    - retriever_weights: вес для объединения плотного и разреженного поиска. (Если установлено значение 0, будет выполняться только разреженный поиск. Если установлено значение 1, будет выполняться только плотный векторный поиск).

Например, если установлено sparse_top_k=5, то это означает, что будет извлечено 5 узлов с помощью разреженных векторов и 5 узлов с помощью плотных векторов. Параметр similarity_top_k управляет окончательным числом возвращаемых узлов. В приведенной выше настройке получается 10 узлов. Алгоритм слияния применяется для ранжирования и упорядочивания узлов из разных векторных пространств (в данном случае слияние с относительной оценкой). Значение параметра similarity_top_k=2 означает, что возвращаются два верхних узла после слияния.

10. Настраиваемые параметры экстракторов:
    - llm: LLM, используемый для извлечения.
    - num_workers: Количество потоков.

SummaryExtractor:

- enabled: логическое значение, указывающее, включен ли SummaryExtractor.
- summaries: Список, указывающий, какие типы краткого содержания следует включить. Возможные значения: «self», «prev» и «next».
    - self: Включить краткое содержание текущего фрагмента.
    - prev: Включить краткое содержание предыдущего фрагмента.
    - next: Включить резюме следующего фрагмента. Использование 'prev' и 'next' добавляет контекст к резюме текущего фрагмента. ВАЖНО: параметр SummaryExtractor предназначен для использования с большими фрагментами, где включение резюме смежных фрагментов обеспечивает ценный контекст. Это гарантирует, что большой фрагмент будет понят в более широком контексте окружающей информации.
- prompt_template: Шаблон для резюмирования каждого фрагмента.

KeywordExtractor:

- enabled: логическое значение, указывающее, включен ли KeywordExtractor.
- prompt_template: Шаблон для извлечения ключевых слов.

11. Настраиваемые параметры для больших контекстов:

Этот раздел управляет процессом формирования окончательного ответа, используя подход древовидного резюмирования для обработки потенциально больших контекстов.

- context_window: определяет размер контекстного окна для LLM. Если общий размер извлеченных соответствующих документов превышает это значение, контекст разбивается на несколько фрагментов, каждый из которых не больше context_window. Затем каждый фрагмент обрабатывается LLM асинхронно. Полученные резюме из каждого фрагмента затем объединяются и возвращаются в LLM для генерации окончательного, всеобъемлющего ответа. Это позволяет системе обрабатывать контексты, превышающие максимальное контекстное окно LLM. Например, если context_window 50 000 токенов, а извлеченный контекст составляет 150 000 токенов, контекст будет разделен на три фрагмента, каждый из которых обрабатывается независимо перед объединением для окончательного ответа.
- num_output: Устанавливает максимальное количество токенов, разрешенных в сгенерированном ответе.
- prompt: Шаблон для генерации ответа.

9 Оценка качества ARAG

Для оценка качества системы существует подключаемый модуль RAGAS.

RAGAS (RAG Assessment) — это открытый Python-фреймворк, предназначенный для автоматической оценки качества RAG-систем. Он помогает измерить эффективность поиска на основе эталонного набора вопросов / ответов (Golden Questions) к загруженному дата-сету. Набор вопросов / ответов должен быть подготовлен заранее.

Пример эталонного датасета:

- user_input - вопрос от пользователя.
- reference - ожидаемый ответ на вопрос.

- reference_contexts - контекст - абзац или несколько абзацев на основе которых был сформирован ответ.

	user_input	reference	reference_contexts
1	"Какие таблицы входят в состав схемы arenadata_toolkit и для чего они предназначены?",	"В состав схемы arenadata_toolkit входят следующие таблицы \n\n1. daily_operation - содержит информацию об операциях VACUUM и ANALYZE, проводимых над таблицами базы данных автоматически по расписанию. Содержимое таблицы обновляется ежедневно при запуске соответствующих скриптов vacuum, analyze, vacuum_analyze_pg_catalog. Включает столбцы schema_name (название схемы), table_name (название таблицы), action (название операции ANALYZE или VACUUM), status (статус операции SCS - успешно, ERROR - с ошибкой), time (длительность операции в секундах), processed_dttm (дата и время операции).\n\n2. db_files_current - содержит актуальную на момент последнего запуска скрипта collect_table_stats информацию о файлах БД на всех сегментах кластера с привязкой к таблицам, индексам и другим объектам БД. Таблица перезаписывается при каждом запуске скрипта.\n\n3. db_files_history - хранит историю изменений файлов БД на всех сегментах кластера. Структура идентична db_files_current за исключением дополнительного поля collecttime. Содержимое обновляется ежедневно при каждом запуске скрипта collect_table_stats и не очищается.\n\n4. operation_exclude - содержит информацию о схемах БД, к	"Входящие в состав arenadata_toolkit таблицы описаны ниже. Таблица arenadata_toolkit.daily_operation содержит информацию об операциях VACUUM и ANALYZE, проводимых над таблицами базы данных автоматически по расписанию. Содержимое таблицы обновляется ежедневно при запуске соответствующих скриптов [vacuum, analyze, vacuum_analyze_pg_catalog](#script-purpose). Таблица arenadata_toolkit.db_files_current содержит актуальную на момент последнего запуска скрипта [collect_table_stats](#script-purpose) информацию о файлах БД на всех сегментах кластера с привязкой к таблицам, индексам и другим объектам БД (при возможности определения таких связей). Таблица перезаписывается при каждом запуске скрипта. Таблица arenadata_toolkit.db_files_history хранит историю изменений файлов БД на всех сегментах кластера с привязкой к таблицам, индексам и другим объектам БД (при возможности определения таких связей). Таблица может быть полезна для отслеживания динамики изменения размера БД. Структура таблицы полностью идентична arenadata_toolkit.db_files_current за исключением дополнительного поля collecttime, описанного ниже. Содержимое arenadata_toolkit.db_files_history обновляется ежедневно при

	user_input	reference	reference_contexts
		которым не требуется применять операции VACUUM и ANALYZE при запуске соответствующих скриптов vacuum и analyze.",	каждом запуске скрипта [collect_table_stats](#script-purpose) (путем копирования актуальных данных из arenadata_toolkit.db_files_current). Таблица не очищается. Таблица arenadata_toolkit.operation_exclude содержит информацию о схемах БД, к которым не требуется применять операции VACUUM и ANALYZE при запуске соответствующих скриптов [vacuum и analyze](#script-purpose)."
2	"Что представляет собой представление adb_skew_coefficients и как интерпретировать значение коэффициента skccoeff?",	"Представление arenadata_toolkit.adb_skew_coefficients отображает информацию о наличии \"перекосов\" (skew) в распределении данных, полученную путем расчета коэффициента вариации (Coefficient of Variation, CV). В отличие от взятого за основу представления Greenplum gp_skew_coefficients, adb_skew_coefficients учитывает физический размер файлов данных таблиц, а не количество их строк, что является более быстрым и точным подходом.\n\nКоэффициент skccoeff представляет собой коэффициент вариации распределения табличных данных между сегментами, рассчитанный как отношение среднеквадратичного отклонения file_size к среднему значению file_size по всем сегментам, где file_size — это размер файлов текущей таблицы на одном сегменте. Величина skccoeff определяет равномерность распределения данных между сегментами кластера и наличие \"перекосов\" при обработке запросов к таблицам. Чем ниже	"Представление arenadata_toolkit.adb_skew_coefficients отображает информацию о наличии перекосов (skew) в распределении данных, полученную путем расчета коэффициента вариации (Coefficient of Variation, CV). В отличие от взятого за основу представления Greenplum gp_skew_coefficients, adb_skew_coefficients учитывает физический размер файлов данных таблиц, а не количество их строк, что является более быстрым и точным подходом. | skccoeff | Коэффициент вариации (Coefficient of Variation, CV) распределения табличных данных между сегментами, рассчитанный как отношение среднеквадратичного отклонения <file_size> к среднему значению <file_size> по всем сегментам, где <file_size> — это размер файлов текущей таблицы на одном сегменте. Величина skccoeff определяет равномерность [распределения данных](../../concept/data-model/tables/distribution.html) между сегментами кластера и наличие перекосов (skew) при обработке запросов к таблицам.

	user_input	reference	reference_contexts
		показатель, тем лучше. При высоких значениях параметра рекомендуется пересмотреть политики распределения данных. Например, если по крайней мере один сегмент кластера хранит более 25% табличных данных по сравнению с другими сегментами, значение skccoeff будет более 10. Однако критичный уровень коэффициента определяется для каждого окружения индивидуально.",	Чем ниже показатель, тем лучше. При высоких значениях параметра рекомендуется пересмотреть политики распределения данных (см. [Рекомендации по равномерному распределению данных](../../concept/data-model/tables/distribution.html#recommend)). Например, если по крайней мере один сегмент кластера хранит более 25% табличных данных по сравнению с другими сегментами, значение skccoeff будет более 10. Однако критичный уровень коэффициента определяется для каждого окружения индивидуально |"
3	"Какие операции выполняет скрипт operation.py и какие аргументы можно использовать при его запуске?",	"Скрипт operation.py выполняет различные операции на уровне кластера ADB. Первый позиционный аргумент определяет имя запрашиваемой операции \n- vacuum_system_db - запускает операцию VACUUM FREEZE для системных БД postgres и template1\n- remove_orphaned_temp_schemas - во всех пользовательских БД удаляет временные схемы, имена которых начинаются на pg_temp_ и pg_toast_temp_\n- vacuum - запускает операцию VACUUM во всех пользовательских БД за исключением схем из таблицы operation_exclude\n- analyze - запускает операцию ANALYZE во всех пользовательских БД за исключением схем из таблицы operation_exclude\n- collect_table_stats - собирает статистику по файлам данных в разрезе сегментов\n- vacuum_analyze_pg_catalog - запускает операции VACUUM и ANALYZE для схемы pg_catalog во всех пользовательских	"*operation.py* — Python-скрипт, который выполняет различные операции на уровне кластера ADB. Тип операции определяется первым аргументом (например, vacuum, analyze и т.д.). | Первый позиционный аргумент в списке (без alias) | Имя запрашиваемой операции * vacuum_system_db * remove_orphaned_temp_schemas * vacuum * analyze * collect_table_stats * vacuum_analyze_pg_catalog Описание каждой операции приведено в таблице [Описание скриптов для обслуживания ADB](#script-purpose) | — | | --loglevel | Уровень логирования. Возможные значения * 2 — INFO; * 3 — WARNING; * 4 — ERROR; * 5 — CRITICAL. Информацию об уровнях логирования можно получить в [документации Python](https //docs.python.org/3/library/logging.html#logging-levels) | 2 | | --processes | Число процессов-обработчиков. Возможные значения

	user_input	reference	reference_contexts
		БД\n\nДоступные аргументы \n--loglevel - уровень логирования (2-INFO, 3-WARNING, 4-ERROR, 5-CRITICAL), по умолчанию 2\n--processes - число процессов-обработчиков от 1 до 10, по умолчанию 10\n--timelimit - период времени в минутах, по истечении которого работа скрипта останавливается\n--excludedb - список БД для исключения из обработки дополнительно к игнорируемым по умолчанию\n--vacuum-order - порядок применения команды VACUUM к таблицам (newest-first, newest-last или имя кастомной функции)",	1 — 10 | 10 | | --timelimit | Период времени, по истечении которого работа скрипта останавливается (в минутах) | — | | --excludedb | Список БД, которые необходимо исключить из обработки — в дополнение к тем, что игнорируются скриптом *operation.py* по умолчанию * template0 * template1 * postgres * gpperfmon | — | | --vacuum-order | Порядок, в котором команда VACUUM будет применяться к таблицам в пользовательских БД. Параметр может использоваться только при вызове операции [vacuum](#script-purpose). Возможные значения * newest-first — таблицы, по которым в pg_catalog.pg_stat_last_operation нет данных о проведении операции VACUUM, будут обрабатываться первыми. Для обработки используется функция arenadata_toolkit.adb_vacuum_strategy_newest_first. * newest-last — таблицы, по которым в pg_catalog.pg_stat_last_operation нет данных о проведении операции VACUUM, будут обрабатываться последними. Для обработки используется функция arenadata_toolkit.adb_vacuum_strategy_newest_last. * Имя кастомной функции в формате <schema_name>.<function_name>. Сигнатура кастомной функции должна быть идентична двум функциям, упомянутым выше, и возвращать список строк, каждая из которых содержит имя схемы и имя таблицы. Если указанная функция отсутствует в каких-либо БД, операция VACUUM к ним не применяется, а в логе сохраняется соответствующее

	user_input	reference	reference_contexts
			предупреждение. Дополнительную информацию можно получить в разделе [Пример использования кастомной функции для VACUUM](#example). | newest-first |"
4	"Какие требования предъявляются к кастомной функции для определения порядка выполнения операции VACUUM?",	"Кастомная функция для определения порядка выполнения операции VACUUM должна удовлетворять следующим условиям \n\n1. Наличие одного входного параметра типа text. При запуске скрипта operation.py в качестве этого параметра всегда передается значение 'VACUUM'. В теле функции он может не использоваться.\n\n2. Функция должна возвращать набор строк со столбцами table_schema (имя схемы БД) и table_name (имя таблицы БД). Это отсортированный перечень таблиц БД, к которым следует применить операцию VACUUM.\n\nВажные требования к размещению функции \n- Функцию необходимо добавить во все пользовательские БД, к которым будет применяться операция VACUUM, т.е. во все БД за исключением системных (template0, template1, postgres, gpperfmon) и тех, что опционально могут быть указаны в аргументе --excludedb скрипта operation.py.\n- Имя кастомной функции заполняется в формате <schema_name>.<function_name>, где <schema_name> — имя схемы, в которой создана функция; <function_name> — имя функции.\n- Передача аргументов в функцию не поддерживается.",	"Создайте необходимую функцию, удовлетворяющую следующим условиям * Наличие одного входного параметра типа text. При запуске скрипта *operation.py* в качестве этого параметра всегда передается значение VACUUM. В теле функции он может не использоваться. * Функция должна возвращать набор строк со столбцами table_schema (имя схемы БД) и table_name (имя таблицы БД). Это отсортированный перечень таблиц БД, к которым следует применить операцию VACUUM. | | |\n| --- | --- |\n| | ВНИМАНИЕ Функцию необходимо добавить во все пользовательские БД, к которым будет применяться операция VACUUM, т.е. во все БД за исключением системных (template0, template1, postgres, gpperfmon) и тех, что опционально могут быть указаны в аргументе [--excludedb](#arg) скрипта *operation.py*. | | | |\n| --- | --- |\n| | ВАЖНО Имя кастомной функции заполняется в формате <schema_name>.<function_name>, где <schema_name> — имя схемы, в которой создана функция; <function_name> — имя функции. Обратите внимание, что передача аргументов в функцию не поддерживается. |"

	user_input	reference	reference_contexts
5	"В какое время по расписанию выполняются основные скрипты обслуживания ADB и что делает каждый из них?",	"Основные скрипты обслуживания ADB выполняются по следующему расписанию (время в формате HH mm UTC) \n\n00 04, 08 04, 16 04 - operation.py → vacuum_analyze_pg_catalog Запускает операции VACUUM и ANALYZE для схемы pg_catalog во всех пользовательских БД. Записывает результат операции в таблицу arenadata_toolkit.daily_operation.\n\n01 00 - run_sql_to_gpssh.sh → gzip_pg_log.sql Для каждого сегмента кластера архивирует CSV-логи, которые хранятся в <data_directory>/pg_log и не изменялись более 3 дней.\n\n02 00 - operation.py → vacuum_system_db Запускает операцию VACUUM FREEZE для системных БД postgres и template1 (путем запуска vaccumdb -F).\n\n03 00 - operation.py → remove_orphaned_temp_schemas Во всех пользовательских БД удаляет более не используемые временные схемы, имена которых начинаются на pg_temp_ и pg_toast_temp_.\n\n06 00 - operation.py → vacuum Запускает операцию VACUUM во всех пользовательских БД за исключением схем, перечисленных в таблице arenadata_toolkit.operation_exclude. Записывает результат операции в таблицу arenadata_toolkit.daily_operation.\n\n11	0 | run_sql_to_gpssh.sh → gzip_pg_log.sql | Для каждого сегмента кластера архивирует CSV-логи, которые хранятся в *<data_directory>/pg_log* и не изменялись более 3 дней (где <data_directory> — директория с данными сегмента) | 01 00 | | 1 | operation.py → vacuum_system_db | Запускает операцию [VACUUM FREEZE](https //docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/ref_guide-sql_commands-VACUUM.html) для системных БД postgres и template1 (путем запуска vaccumdb -F) | 02 00 | | 2 | operation.py → remove_orphaned_temp_schemas | Во всех пользовательских БД удаляет более не используемые временные схемы, имена которых начинаются на pg_temp_ и pg_toast_temp_ | 03 00 | | 3 | operation.py → vacuum | * Запускает операцию [VACUUM](https //docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/ref_guide-sql_commands-VACUUM.html) во всех пользовательских БД за исключением схем, перечисленных в таблице [arenadata_toolkit.operation_exclude](#operation_exclude). * Записывает результат операции в таблицу [arenadata_toolkit.daily_operation](#daily_operation). | 06 00 | | 4 | operation.py → analyze | * Запускает операцию [ANALYZE](https //docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/ref_guide-sql_commands-ANALYZE.html) во всех

	user_input	reference	reference_contexts
		00 - operation.py → analyze Запускает операцию ANALYZE во всех пользовательских БД за исключением схем, перечисленных в таблице arenadata_toolkit.operation_exclude. Записывает результат операции в таблицу arenadata_toolkit.daily_operation.\n\n21 00 - operation.py → collect_table_stats Для всех пользовательских БД собирает актуальную статистику по файлам данных в разрезе сегментов, сохраняет информацию в таблице arenadata_toolkit.db_files_current и добавляет новые данные в таблицу arenadata_toolkit.db_files_history. ",	пользовательских БД за исключением схем, перечисленных в таблице [arenadata_toolkit.operation_exclude](#operation_exclude). * Записывает результат операции в таблицу [arenadata_toolkit.daily_operation](#daily_operation). | 11 00 | | 5 | operation.py → collect_table_stats | Для всех пользовательских БД 1. Собирает актуальную статистику по файлам данных в разрезе сегментов (размер, расположение и т.д.) и пытается определить связи файлов с таблицами, индексами и другими объектами БД. Для этого вызывает представления (view) схемы arenadata_toolkit, которые обращаются к системным таблицам gp_segment_configuration, pg_class, pg_namespace, pg_tablespace, pg_database. 2. Сохраняет полученную информацию в таблице [arenadata_toolkit.db_files_current](#db_files_current), предварительно очищая ее содержимое. Некоторые табличные столбцы могут остаться пустыми (NULL), если для файлов не определены соответствующие объекты БД на предыдущем шаге (например, oid, table_name и т.д.). 3. Добавляет новые данные в таблицу [arenadata_toolkit.db_files_history](#db_files_history) (копируя их из [arenadata_toolkit.db_files_current](#db_files_current)). | 21 00 | | 6 | operation.py → vacuum_analyze_pg_catalog | * Запускает операции

	user_input	reference	reference_contexts
			[VACUUM](https //docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/ref_guide-sql_commands-VACUUM.html) и [ANALYZE](https //docs.vmware.com/en/VMware-Greenplum/6/greenplum-database/ref_guide-sql_commands-ANALYZE.html) для схемы pg_catalog во всех пользовательских БД. * Записывает результат операции в таблицу [arenadata_toolkit.daily_operation](#daily_operation). | 00 04 08 04 16 04 |"
6	Когда запускается вакуум	Операция Vacuum запускается для схемы pg_catalog во всех пользовательских БД в 00:04, 08:04 и 16:04 , во всех пользовательских БД в 06:00	Описание скриптов для обслуживания ADB Время запуска в формате HH:mm (UTC) operation.py → vacuum_analyze_pg_catalog Запускает операции VACUUM и ANALYZE для схемы pg_catalog во всех пользовательских БД. 00:04 08:04 16:04
			operation.py → vacuum Запускает операцию VACUUM во всех пользовательских БД за исключением схем, перечисленных в таблице arenadata_toolkit.operation_exclude. Записывает результат операции в таблицу arenadata_toolkit.daily_operation. 06:00
7	какие коннекторы поддерживаются ADB	ADB поддерживает следующие коннекторы ADB to ADB Connector — это реализованный на основе foreign data wrapper и параллельных курсоров (parallel retrieve cursor)	Для интеграции ADB и Kafka можно использовать один из коннекторов: ADB to Kafka Connector — основан на протоколе PXF и механизме внешних (external) таблиц.

	user_input	reference	reference_contexts
		коннектор, обеспечивающий двустороннюю передачу данных между двумя кластерами ADB Tkhemali Connector 1.X - коннектор для отправки данных из ADB в ClickHouse ADB ClickHouse Connector - ADB ClickHouse Connector реализован на базе foreign data wrapper и foreign-таблиц. ADB to Kafka Connector — основан на протоколе PXF и механизме внешних (external) таблиц. Реализует загрузку данных из ADB в Kafka Kafka to ADB Connector — основан на механизме foreign data wrapper. Использует библиотеку librdkafka для взаимодействия с Kafka и реализует транзакционную загрузку данных из Kafka в ADB ADB to ADB Connector — это реализованный на основе foreign data wrapper и параллельных курсоров (parallel retrieve cursor) коннектор, обеспечивающий двустороннюю передачу данных между двумя кластерами ADB. ADB Spark Connector - Многофункциональный коннектор с поддержкой параллельных операций чтения/записи между Apache Spark и Arenadata DB.	Реализует загрузку данных из ADB в Kafka. Описание коннектора представлено в статьях: Kafka to ADB Connector — основан на механизме foreign data wrapper. Использует библиотеку librdkafka для взаимодействия с Kafka и реализует транзакционную загрузку данных из Kafka в ADB. ADB to ADB Connector — это реализованный на основе foreign data wrapper и параллельных курсоров (parallel retrieve cursor) коннектор, обеспечивающий двустороннюю передачу данных между двумя кластерами ADB. Обзор ADB ClickHouse Connector В отличие от предыдущей версии коннектора Tkhemali 1.X, основанной на механизме внешних (external) таблиц, ADB ClickHouse Connector реализован на базе foreign data wrapper и foreign-таблиц. Настройка Tkhemali Connector 1.X Для отправки данных из ADB в ClickHouse через Tkhemali Connector 1.X необходимо предварительно выполнить следующие шаги на стороне кластера ADB: Коннекторы ADB Spark Connector Многофункциональный коннектор с поддержкой параллельных операций чтения/записи между Apache Spark и Arenadata DB. ADB Kafka Connector Специализированный коннектор для интеграции Apache Kafka с Arenadata DB. ADB PXF Connector Фреймворк для параллельного и высокопроизводительного доступа к гетерогенным источникам данных из Arenadata DB при помощи встроенных

	user_input	reference	reference_contexts
			коннекторов. ADB ClickHouse Connector Fdw-коннектор для передачи данных в Arenadata QuickMarts или ClickHouse.
8	Какие минимальные требования к сегмент-хостам ADB	Системные требования, приведенные ниже, являются минимальными для Сегмент-хоста ADB. Физический сервер. CPU: от 16 ядер. RAM: от 32 ГБ. Диск: один или несколько физических неформатированных массивов RAID 10 (от 10 ГБ каждый).	Требования к оборудованию ВАЖНО Системные требования, приведенные ниже, являются минимальными. Целевой сайзинг необходимо рассчитывать исходя из требований вашей организации. ADB Сегмент-хосты Физический сервер. CPU: от 16 ядер. RAM: от 32 ГБ. Диск: один или несколько физических неформатированных массивов RAID 10 (от 10 ГБ каждый).
9	Можно ли установить ADB на Android	Android не указан в списке поддерживаемых операционных систем ADB ADB поддерживает следующие операционные системы: RHEL 7.9. CentOS 7.9. Ubuntu 22.04 (в ADB 6 начиная с версии 6.27.1.56). См. Требования для Astra Linux и Ubuntu ниже. AltLinux 8.4 SP (для Enterprise-версии ADB). РЕД ОС 7.3 Сертифицированная редакция (для Enterprise-версии ADB 6 начиная с 6.27.1.58). Astra Linux 1.7 "Орел" SE (для Enterprise-версии ADB 6 начиная с 6.25.2.52). См. Требования для Astra Linux и Ubuntu ниже. Начиная с версии 6.26.2.55 ADB 6 поддерживает только Astra Linux 1.7.5.	ADB поддерживает следующие операционные системы: RHEL 7.9. CentOS 7.9. Ubuntu 22.04 (в ADB 6 начиная с версии 6.27.1.56). См. Требования для Astra Linux и Ubuntu ниже. AltLinux 8.4 SP (для Enterprise-версии ADB). РЕД ОС 7.3 Сертифицированная редакция (для Enterprise-версии ADB 6 начиная с 6.27.1.58). Astra Linux 1.7 "Орел" SE (для Enterprise-версии ADB 6 начиная с 6.25.2.52). См. Требования для Astra Linux и Ubuntu ниже. Начиная с версии 6.26.2.55 ADB 6 поддерживает только Astra Linux 1.7.5.

Ключевые возможности RAGAS:

1. Оценка компонента Retrieval (поиска)

Проверяет, насколько хорошо система подбирает контекст для ответа:

- Context Relevance: Релевантность найденных документов запросу.
- Context Recall: Полнота извлечения ключевых фактов.
- Context Precision: Точность поиска (минимум «мусора»).

2. Оценка компонента Generation (генерации)

Анализирует качество ответов языковой модели:

- Faithfulness: Отсутствие выдуманных фактов (галлюцинаций).
- Answer Relevance: Соответствие ответа вопросу.
- Answer Correctness: Фактическая точность ответа.

3. Дополнительные метрики

- Answer Similarity: Семантическая близость к эталонному ответу.
- Aspect Critiques: Оценка по конкретным критериям (например, безопасность, токсичность).

Описание метрик

context_entity_recall

Context Entity Recall (CER) — это метрика, используемая для оценки способности модели корректно упоминать именованные сущности (NER) в контексте генерации текста (например, в диалоговых системах, QA или summarization).

Она измеряет, какая доля сущностей из эталонного ответа была корректно воспроизведена в сгенерированном тексте.

nv_context_relevance

Normalized Variant of Context Relevance — это метрика, оценивающая, насколько сгенерированный текст соответствует контексту исходного запроса или документа.

Метрика нормирована в диапазоне [0, 1], где:

- 1 — идеальное соответствие контексту,
- 0 — текст полностью нерелевантен.

nv_response_groundedness

Normalized Variant of Response Groundedness

NV_Response_Groundedness — это метрика, оценивающая, насколько сгенерированный ответ основан на фактических данных и подтверждается исходным контекстом.

1. Оценка на основе контекста

Метрика проверяет, содержит ли ответ:

- Прямые факты из контекста (даты, имена, цифры),

- Логические выводы. которые можно обосновать контекстом,

- Отсутствие противоречий с исходными данными.

2. Методы расчета

- BERT-based NLI (Natural Language Inference)

context_precision

Context Precision — это метрика, оценивающая, насколько релевантные фрагменты контекста система использует для формирования ответа. Она особенно полезна для RAG-систем (Retrieval-Augmented Generation) и вопросно-ответных систем, где важно:

1. Извлекать правильные фрагменты контекста,
2. Игнорировать нерелевантную информацию.

Диапазон значений:

- 1.0 — все извлеченные фрагменты контекста полезны для ответа,
- 0.0 — контекст полностью нерелевантен.

Как определяется релевантность:

1. Вручную: Эксперт помечает фрагменты как релевантные/нерелевантные.
2. Автоматически:

- NLI-модель (например, DeBERTa) проверяет, поддерживает ли фрагмент ответ,
- Семантическое сходство (например, косинусная близость эмбеддингов).

context_recall

Context Recall — это метрика, оценивающая полноту извлечения релевантной информации из доступного контекста для ответа на запрос. Она показывает, какую долю нужных данных система смогла найти

nv_accuracy

Метрика NV_Accuracy (Normalized Variant of Accuracy - NV_Accuracy — это нормированная метрика, оценивающая фактическую точность ответов генеративных моделей (чат-ботов, QA-систем) по сравнению с эталонными данными. Она измеряет долю верных утверждений в сгенерированном тексте.

Метрики анализа естественного языка

factual_correctness (mode=f1)

Factual Correctness (F1) — это комбинированная метрика, оценивающая точность и полноту фактологической корректности сгенерированного текста.

Эта метрика используется для определения степени соответствия сгенерированного ответа эталону. Показатель фактической корректности варьируется от 0 до 1, при этом более высокие значения указывают на более высокую эффективность. Для измерения соответствия между ответом и эталоном метрика сначала использует метод LLM для разбиения ответа и ссылки на утверждения, а затем использует вывод на естественном языке для определения фактического совпадения между ответом и ссылкой. Фактическое совпадение количественно оценивается с помощью точности, полноты и оценки F1, которую можно контролировать с помощью параметра mode.

Формула для расчета истинно положительного (TP), ложноположительного (FP) и ложноотрицательного (FN) результата выглядит следующим образом:

1. Истинно положительные (True Positives, TP)

Утверждения в ответе, которые корректны и подтверждены контекстом.

TP = Число верных утверждений, соответствующих эталону/контексту

2. Ложноположительные (False Positives, FP)

Утверждения в ответе, которые неверны или не подтверждены контекстом.

FP = Число ошибочных утверждений в ответе

3. Ложноотрицательные (False Negatives, FN)

Ключевые факты из эталона/контекста, которые пропущены в ответе.

FN = Число важных фактов в эталоне, отсутствующих в ответе

Precision (какая доля утверждений верна);

Precision = (Число подтверждённых утверждений) / (Общее число утверждений в ответе)

Recall (какая доля необходимых фактов упомянута):

Recall = (Число подтверждённых утверждений) / (Общее число ключевых фактов в эталоне)

semantic_similarity

Semantic Similarity (Семантическое сходство) — это метрика, измеряющая степень смысловой близости между двумя текстами на основе их контекстуального значения, а не точного лексического совпадения.

Semantic Similarity в RAGAS измеряет, насколько смыслово близок ответ RAG-системы к эталонному ответу. В отличие от точного совпадения (Exact Match), она оценивает идеи, а не слова.

Пример:

Эталон: "ИИ преобразует данные в полезную информацию"

Ответ ARAG: "Искусственный интеллект анализирует данные и извлекает знания"

Высокая семантическая схожесть (хотя слова разные).

Диапазон значений:

- 0 — тексты совершенно не связаны по смыслу,
- 1 — полное семантическое совпадение.

Традиционные метрики НЛП

rouge_score (mode=fmeasure)

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) F-measure - это комплексная метрика для оценки качества автоматически генерируемых текстов (рефератов, переводов и т.д.), объединяющая precision (точность) и recall (полноту) в единый показатель.

F1 = 2 (precision recall) / (precision + recall)

Где:

- Precision = Количество совпавших n-грамм / Общее количество n-грамм в сгенерированном тексте. Precision (точность) в ROUGE показывает, какая доля n-грамм (слов, биграмм и т. д.) из сгенерированного текста (например, автоматического реферата или перевода) действительно присутствует в эталонном тексте (оригинале или ручном реферате).
- Recall = Количество совпавших n-грамм / Общее количество n-грамм в эталонном тексте
- Precision отвечает на вопрос: "*Какой процент моего текста корректен?*"
- Recall отвечает на вопрос: "*Какой процент эталонного текста я воспроизвел?*"

bleu_score

Метрика BLEU (Bilingual Evaluation Understudy Score)

BLEU — это стандартная метрика для оценки качества машинного перевода и других задач генерации текста. Она измеряет n-граммное совпадение между сгенерированным текстом и эталонным (человеческим) переводом.

non_llm_string_similarity

Non-LLM String Similarity (Сходство строк без использования языковых моделей) - это метрика, которая оценивает степень схожести между двумя строками текста без использования сложных языковых моделей (как LLM). Она основана на простых алгоритмах сравнения строк и полезна в случаях, когда нужно быстро оценить похожесть текстов без глубокого семантического анализа.

Основные методы расчета

1. Расстояние Левенштейна - минимальное количество операций вставки, удаления и замены символов для превращения одной строки в другую

2. Коэффициент Жаккара - сравнение множеств слов в двух строках

3. Косинусное сходство на основе TF-IDF

4. Метрика Дамерау-Левенштейна (учитывает перестановки соседних символов)

Диапазон значений: от 0 (строки совершенно разные) до 1 (полное совпадение)

string_present

String Present — это бинарная метрика, которая определяет, содержится ли определенная строка (подстрока) в тексте. Она возвращает:

- 1 - если строка найдена,
- 0 - если строка отсутствует.

exact_match

Exact Match (Точное совпадение) — это строгая метрика, которая оценивает, полностью ли совпадает сгенерированный ответ с эталонным ответом. Она возвращает:

- 1, если ответы идентичны,
- 0, если есть любые расхождения.

Метрика чувствительна к:

- Регистру букв
- Пунктуации
- Порядку слов
- Формату ответа

Варианты проверки:

- Строгое сравнение (включая все символы)
- Нормализованное сравнение (игнорирование регистра/пунктуации)