Тихомирова Анна Николаевна

кандидат технических наук, доцент, НИЯУ МИФИ,

Россия, г. Москва

Ворожейкин Дмитрий Сергеевич

студент НИЯУ МИФИ,

Россия, г. Москва

Минеев Артем Сергеевич

студент НИЯУ МИФИ,

Россия, г. Москва

РАЗРАБОТКА И ОЦЕНКА МЕТОДА ГРАФОВОГО ПРЕДСТАВЛЕНИЯ СЛАБОСТРУКТУРИРОВАННЫХ ДАННЫХ ДЛЯ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ LLM-АНАЛИТИКИ

АННОТАЦИЯ

В статье рассматривается метод повышения эффективности вопросно-ответной аналитики слабоструктурированных JSON-данных за счет интеграции больших языковых моделей и графовой базы данных. Актуальность работы связана с ограничениями прямой передачи больших массивов документов в контекст LLM: ростом стоимости обработки, нестабильностью ответов, затруднением агрегаций и риском смешения сущностей. Предложен подход, при котором исходные JSON-документы предварительно преобразуются в графовую модель ArangoDB, а пользовательский вопрос переводится языковой моделью в запрос AQL. После выполнения запроса в базе данных LLM используется для формирования итогового текстового ответа на основе уже отфильтрованных и агрегированных результатов. Экспериментальная проверка выполнена на наборе из 130 JSON-файлов и 10 аналитических вопросов. Полученные результаты показывают, что графовый подход снизил стоимость LLM-запросов примерно в 9,4 раза и дал более точные ответы в 7 из 10 проверочных вопросов при отсутствии случаев, где прямой подход оказался лучше.

Ключевые слова: большие языковые модели, LLM, Data Science, графовые базы данных, ArangoDB, AQL, Graph-RAG, JSON, слабоструктурированные данные, интеллектуальный анализ данных.

Введение

Развитие больших языковых моделей привело к распространению систем, способных отвечать на вопросы по пользовательским данным на естественном языке. Такие системы применяются для анализа документов, отчетов, протоколов, транскриптов и других источников, которые часто имеют слабоструктурированную природу. Несмотря на высокое качество генерации текста, прямое использование LLM для аналитических задач сталкивается с рядом ограничений. Если модель получает на вход большой набор документов, она вынуждена обрабатывать значительный объем контекста, что увеличивает стоимость запроса и снижает воспроизводимость результата [1; 7].

Особенно заметны эти ограничения в задачах, где требуется не пересказ отдельных документов, а вычисление агрегированных показателей: средних значений, рейтингов, долей, группировок, динамики и сравнений между сущностями. В таких случаях языковая модель должна одновременно извлечь данные, сопоставить однотипные сущности, выполнить вычисления и сформулировать вывод. При прямом подходе возрастает вероятность арифметических ошибок, неполного учета документов и смешения ролей сущностей.

Одним из способов повышения надежности является перенос части аналитической нагрузки из языковой модели в специализированное хранилище данных. В данной работе рассматривается графовая база данных ArangoDB как промежуточный слой между массивом JSON-документов и LLM. Графовая модель позволяет явно представить сущности и связи между ними, а AQL-запросы обеспечивают детерминированное выполнение фильтрации, группировки и агрегации.

Предлагаемый подход относится к направлению Data Science, поскольку объединяет подготовку данных, моделирование структуры набора, автоматизированное выполнение аналитических запросов и оценку качества результатов. При этом LLM используется не как единственный вычислительный механизм, а как интерфейс между человеком и формальной структурой данных.

Постановка задачи

Целью работы является разработка и экспериментальная оценка подхода к интеллектуальному анализу слабоструктурированных JSON-данных, при котором LLM используется не для прямого чтения всего массива документов, а для генерации запросов к графовой базе данных и последующей интерпретации результатов.

Для достижения цели были поставлены следующие задачи: разработать схему преобразования JSON-документов в графовую структуру; реализовать механизм сопоставления сущностей и связей; организовать перевод вопросов на естественном языке в AQL-запросы; выполнить сравнительный эксперимент с прямой LLM-обработкой документов; оценить подход по стоимости, точности ответов и устойчивости при аналитических запросах.

Объектом исследования являются слабоструктурированные данные в формате JSON. Предметом исследования выступает метод организации таких данных в графовой базе для последующего вопросно-ответного анализа с использованием больших языковых моделей.

Научно-практическая значимость работы состоит в том, что предложенный способ может применяться для наборов документов, где данные уже частично структурированы, но не приведены к единой аналитической модели. Такие наборы часто возникают после автоматического извлечения информации из текстов, аудиозаписей, отчетов, анкет или протоколов.

Анализ существующих подходов к LLM-аналитике

Наиболее простой способ построить вопросно-ответную систему по набору документов заключается в передаче релевантных файлов или их фрагментов

непосредственно в контекст языковой модели. Такой подход удобен при малом объеме данных и вопросах, ориентированных на поиск отдельных фактов. Однако при росте числа документов увеличивается размер контекста, а вместе с ним стоимость обработки и вероятность потери важных деталей.

Классический Retrieval-Augmented Generation решает часть этой проблемы за счет поиска релевантных фрагментов перед генерацией ответа. Но в задачах аналитики этого бывает недостаточно. Если вопрос требует вычислить среднее значение по всем объектам, найти долю нулевых оценок или сравнить группы сущностей, необходимо не только найти фрагменты, но и выполнить формальные операции над данными [2].

Графовый подход расширяет идею RAG за счет явного представления сущностей и связей. Вместо набора независимых текстовых фрагментов система получает структуру, где документы, участники, категории, показатели и отношения между ними могут быть обработаны запросным языком. Это особенно важно для слабоструктурированных JSON-данных [3; 6].

Архитектура разработанной системы

Разработанный проект представляет собой приложение, объединяющее Flask, Celery, ArangoDB и LLM-сервис. Пользователь загружает исходные файлы, после чего система выполняет предварительную обработку, строит единую JSON-структуру, создает коллекции и графовые связи, формирует схему базы данных и использует ее при генерации запросов [4; 5].

Основной поток обработки включает несколько этапов. На первом этапе исходные файлы валидируются и приводятся к единому виду. На втором этапе выполняется построение схемы: выделяются основные документы, вершины и ребра графа. На третьем этапе данные загружаются в ArangoDB. На четвертом этапе пользовательский вопрос передается в LLM вместе со схемой базы, после чего модель формирует один или несколько read-only AQL-запросов.

В реализованной модели исходные документы представлены как встречи, а отдельные сущности, извлеченные из метаданных, представлены вершинами графа. Например, выделяются коллекции встреч, руководителей, сотрудников и

точек. Между ними создаются ребра, отражающие участие сущностей в конкретных документах.

Важным элементом системы является журналирование LLM-вызовов. Для каждого запроса фиксируются модель, тип операции, число входных и выходных токенов, длительность и расчетная стоимость. Это позволило провести не только качественное, но и количественное сравнение подходов.

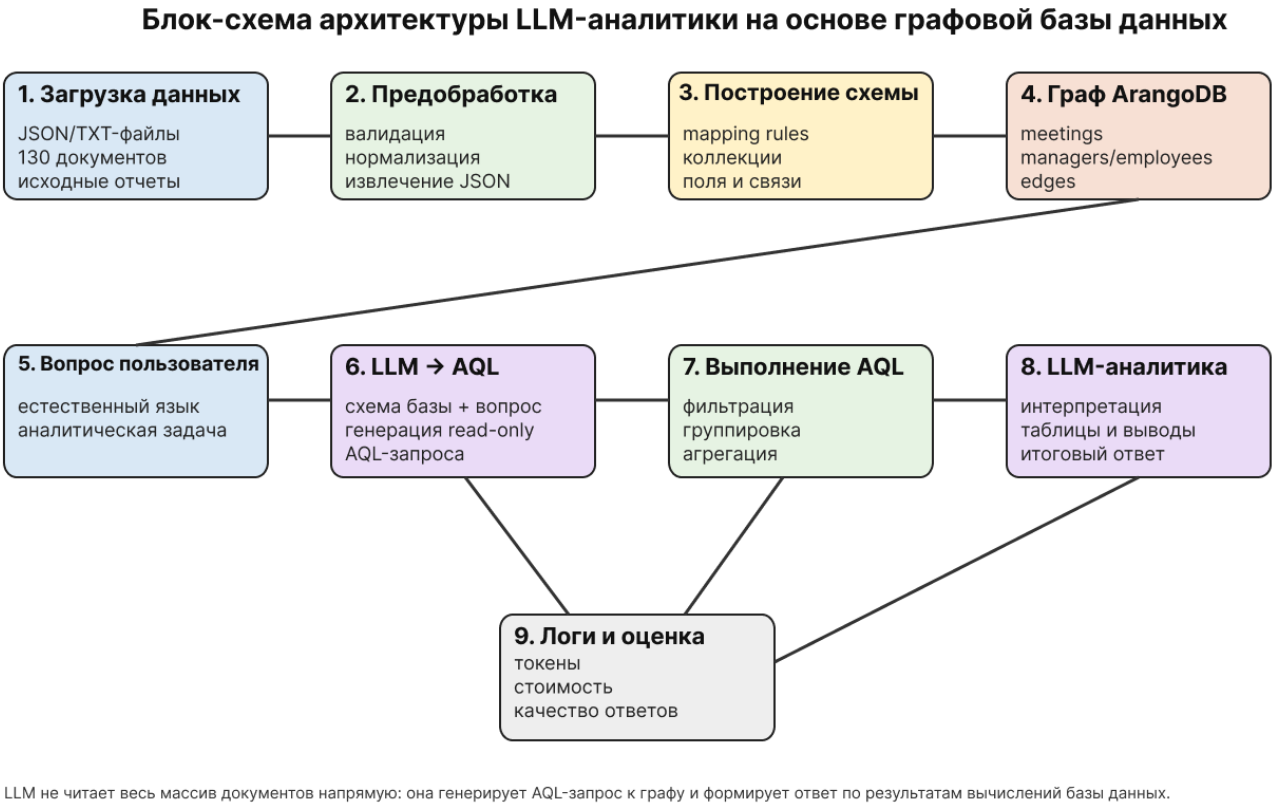

Рисунок 1. Блок-схема работы системы интеллектуального анализа данных

Компоненты архитектуры

Архитектуру системы можно представить как последовательность взаимосвязанных компонентов. Загрузка файлов отвечает за прием исходных документов и формирование набора данных. Предобработка выполняет

извлечение полезного JSON, удаление служебных оболочек, проверку структуры и нормализацию полей. Компонент построения схемы определяет основную коллекцию, вершины, ребра и правила переноса атрибутов.

ArangoDB выступает как аналитическое ядро системы. В базе хранятся документы, вершины и ребра, а AQL используется для выполнения точных операций фильтрации, группировки и агрегации. LLM-генератор AQL преобразует вопрос пользователя на естественном языке в формальный запрос к базе. LLM-аналитик получает результат запроса и превращает его в связный ответ, таблицу или аналитический вывод [4; 5].

Отдельный компонент логирования фиксирует токены, стоимость, статус выполнения и тип операции. Это важно для воспроизводимой оценки: система позволяет сравнивать подходы не только по качеству текста, но и по экономическим параметрам LLM-аналитики.

Таблица 1.

Компоненты архитектуры системы интеллектуального анализа данных

Компонент	Назначение

Загрузка и хранение файлов	Прием исходных JSON/TXT-документов.
Предобработка	Извлечение JSON, нормализация и проверка структуры.
Построение графовой схемы	Определение коллекций, вершин, ребер и правил маппинга.
ArangoDB	Хранение графа и выполнение AQL-запросов.
LLM-генератор AQL	Преобразование вопроса в read-only AQL-запрос.
LLM-аналитик	Формирование итогового ответа по результатам запроса.
Логирование	Фиксация токенов, стоимости и качества ответов.

Графовая модель данных

Ключевой особенностью разработанного подхода является перевод набора слабоструктурированных JSON-документов в графовую модель. В качестве основной коллекции используются документы, соответствующие отдельным объектам анализа. Для экспериментального набора такими объектами являются записи встреч. Метаданные и показатели хранятся как поля основного документа, а повторяющиеся сущности выносятся в отдельные коллекции вершин [6].

Такой способ организации данных позволяет избежать дублирования и неоднозначности. Например, один руководитель может встречаться во многих документах. При прямой обработке LLM должна самостоятельно понять, что разные упоминания относятся к одной сущности, а также отличить руководителя от участника. В графовой базе эта информация фиксируется явно.

Графовая модель удобна и для расширения. Если в новых данных появляются дополнительные типы объектов, например подразделения, категории, темы, продукты или временные периоды, они могут быть добавлены

как новые коллекции вершин и ребер без полного пересмотра логики вопросно-ответного интерфейса.

Механизм обработки пользовательского вопроса

Вопрос пользователя поступает в систему на естественном языке. На этом этапе LLM получает не весь набор исходных файлов, а описание схемы базы данных и сам вопрос. Такая постановка задачи существенно уменьшает объем входного контекста. Модель должна не пересказывать документы, а выбрать нужные коллекции, поля и связи, после чего сформировать AQL-запрос.

Сформированный запрос дополнительно ограничивается требованием read-only. Это необходимо для предотвращения изменения данных при ошибочной генерации. После выполнения запроса система получает структурированный результат: таблицу, список объектов, агрегированные значения или набор записей. Только после этого LLM используется повторно как компонент генерации итогового ответа.

Такой двухэтапный механизм разделяет вычисление и интерпретацию. База данных отвечает за точность выборки и расчетов, а языковая модель отвечает за понятное объяснение результата. Это особенно важно для задач Data Science, где ценность имеет не только текст ответа, но и корректность исходных вычислений.

Методика эксперимента

Для проверки эффективности подхода был использован набор из 130 JSON-файлов. Каждый файл содержал данные, полученные из материалов встреч: метаданные, показатели по блокам, оценки чек-листа и связанные сущности. На данном наборе были сформулированы 10 аналитических вопросов, ориентированных на вычисления, группировки, сравнения и поиск закономерностей.

Сравнивались два подхода. Первый подход заключался в прямой передаче данных в LLM и получении ответа без предварительного построения графовой базы. Второй подход использовал графовую базу ArangoDB: LLM генерировала

AQL-запрос, база выполняла вычисления, а модель формировала итоговый ответ на основе результата запроса.

Оценка проводилась по трем критериям: корректность ответа относительно фактических данных, способность выполнять аналитические операции и стоимость обработки, выраженная через количество входных и выходных токенов. Особое внимание уделялось вопросам, где необходимо различать роли сущностей и корректно считать средние значения.

Таблица 2.

Сравнение прямой LLM-обработки и графового подхода

Метрика	Прямая LLM-обработка	Графовая база + LLM
Количество JSON-файлов	130	130
Количество аналитических вопросов	10	10
Input tokens	249 740	7 012
Output tokens	6 467	4 627
Стоимость 10 вопросов	$0.8462	$0.0904
Относительная стоимость	100%	около 10,7%
Лучшие ответы	0 из 10	7 из 10
Примерно одинаковые ответы	3 из 10	3 из 10

Результаты эксперимента

Эксперимент показал значительное снижение объема входного контекста при использовании графовой базы данных. Прямая обработка потребовала 249 740 входных токенов, тогда как графовый подход потребовал 7 012 входных токенов. Таким образом, основная экономия достигается за счет того, что в LLM передается не весь массив документов, а схема базы, пользовательский вопрос и компактный результат AQL-запроса.

Стоимость обработки 10 вопросов при прямом подходе составила 0,8462 доллара, а при графовом подходе 0,0904 доллара. Следовательно, графовый вариант оказался примерно в 9,4 раза дешевле. При этом снижение стоимости не сопровождалось ухудшением качества: в 7 из 10 вопросов графовая база дала более точный ответ, в 3 случаях результаты были сопоставимы, а случаев превосходства прямого подхода выявлено не было.

Наиболее заметное преимущество графовой модели проявилось в вопросах, требующих агрегации по сущностям. Например, при определении руководителей с наибольшим и наименьшим средним баллом прямой подход смешивал роли участников и давал приблизительные оценки. Графовый подход использовал явные коллекции и связи, поэтому корректно группировал данные по нужному типу сущности и возвращал точные значения.

Также графовая модель оказалась устойчивее в вопросах о слабых и сильных блоках чек-листа. За счет выполнения расчетов в базе данных система возвращала средние значения и доли нулевых оценок по всем документам, а LLM использовалась уже для интерпретации результата. Это разделение вычислительной и генеративной частей уменьшило вероятность арифметических ошибок.

Обсуждение полученных результатов

Полученные результаты позволяют рассматривать предложенный подход как разновидность Graph-RAG, адаптированную для слабоструктурированных JSON-данных. В отличие от классического RAG, где основной задачей является поиск релевантных фрагментов текста, в данной системе ключевую роль играет

предварительное структурирование данных и выполнение аналитических запросов к графу.

Преимущество графовой модели состоит в том, что она явно фиксирует типы сущностей и отношения между ними. Это особенно важно для наборов данных, где одни и те же имена, роли или объекты могут встречаться в разных контекстах. Если такие данные передаются в LLM напрямую, модель вынуждена самостоятельно восстанавливать структуру связей из текста.

В контексте Data Science такой подход можно рассматривать как совмещение семантического интерфейса и детерминированного аналитического ядра. Пользователь формулирует вопрос естественным языком, но итоговые численные значения формируются не генеративной моделью, а запросом к базе данных. Это повышает объяснимость результата.

В то же время подход имеет ограничения. Качество результата зависит от корректности построения схемы, полноты маппинга и точности генерации AQL-запросов. Ошибка на этапе извлечения сущностей или построения ребер может повлиять на дальнейшие ответы.

Практическая значимость и направления развития

Практическая значимость предложенного решения заключается в возможности создавать аналитические системы поверх массивов документов без ручного проектирования классической реляционной схемы для каждого нового набора. Графовая база хорошо подходит для ситуаций, где данные имеют множество пересекающихся сущностей и связей, а вопросы пользователей заранее неизвестны.

В дальнейшем систему можно расширить несколькими способами. Во-первых, целесообразно добавить автоматическое сравнение сгенерированного AQL-запроса с набором эталонных шаблонов. Во-вторых, можно использовать гибридный подход, объединяющий графовую базу и векторный поиск: граф будет отвечать за структурные связи и агрегаты, а векторный поиск — за извлечение релевантных текстовых фрагментов.

Еще одно направление связано с автоматическим построением схемы. В текущей реализации используются правила маппинга, которые позволяют выделять основную коллекцию, вершины и ребра. Для универсального применения системы важно развивать алгоритмы, способные предлагать такую схему автоматически.

Формализация предложенного подхода

Предложенный метод можно представить как последовательность преобразований исходного набора документов. Пусть D обозначает множество слабоструктурированных JSON-документов, каждый из которых содержит набор полей, вложенных объектов и массивов. На первом этапе выполняется функция нормализации N(D), приводящая документы к единому внутреннему представлению. На втором этапе функция отображения M выделяет основные объекты, сущности и связи, после чего формируется граф G = (V, E), где V — множество вершин, а E — множество ребер между ними.

Вершины графа соответствуют устойчивым сущностям предметной области: документам, участникам, категориям, объектам или иным повторяющимся элементам данных. Ребра отражают отношения между сущностями: участие, принадлежность, связь с документом, роль или контекст. Таким образом, вложенная структура JSON переносится в модель, где аналитический запрос может использовать как атрибуты документов, так и связи между сущностями [6; 10].

Пользовательский вопрос q на естественном языке преобразуется языковой моделью в запрос a к базе данных: LLM(q, S) -> a, где S — описание схемы графовой базы. Запрос a выполняется средствами ArangoDB, и результат R = AQL(a, G) передается во второй LLM-вызов для формирования текстового ответа. Важным свойством является то, что вычислительная часть ответа зависит не от вероятностной генерации модели, а от результата формального запроса к графу [4; 6].

В отличие от прямого подхода LLM(D, q), где модель получает значительный объем исходных документов, в предложенной схеме входной

контекст ограничивается схемой, вопросом и результатом запроса. Это уменьшает стоимость обработки и снижает вероятность того, что модель пропустит часть документов или выполнит некорректную агрегацию.

Алгоритм работы системы

Алгоритм работы системы начинается с загрузки файлов пользователем. Каждый файл проходит проверку расширения, чтение содержимого и попытку извлечения JSON-объекта. Если документ содержит служебную оболочку внешней системы, такая оболочка удаляется, а в дальнейшую обработку передается только полезная предметная часть. Это позволяет не смешивать транспортные метаданные с аналитическими данными.

Далее выполняется построение шаблона структуры данных. Система анализирует повторяющиеся поля, типы значений и вложенные объекты. На основе правил маппинга определяется основная коллекция и набор дополнительных коллекций вершин. Для экспериментального набора основной коллекцией выступают документы встреч, а отдельными вершинами являются руководители, сотрудники и точки. Ребра связывают каждую встречу с соответствующими сущностями.

После загрузки графа в ArangoDB формируется описание схемы, используемое при генерации запросов. Схема содержит сведения о коллекциях, полях, типах связей и допустимых направлениях обхода графа. Когда пользователь задает вопрос, LLM получает эту схему и формирует AQL-запрос. Запрос проверяется на read-only характер, после чего выполняется в базе.

Если результат запроса пустой или содержит ошибку, система может использовать fallback-логику: сгенерировать более простой запрос, ослабить фильтры или вернуть диагностическую информацию. Это важно для практического использования, поскольку естественный язык допускает неоднозначные формулировки, а LLM может выбрать слишком строгие условия фильтрации.

На финальном этапе результат запроса передается LLM-аналитику. В отличие от первого вызова, здесь модель не решает задачу поиска данных. Она

получает уже подготовленную выборку и отвечает за связное изложение результата: формирование таблицы, ранжирование, краткое объяснение и выводы. Такой подход делает архитектуру более контролируемой.

Пример аналитического запроса

Типичный вопрос к системе может быть сформулирован следующим образом: «Какие сущности показывают наибольший и наименьший средний балл по всем блокам?» При прямой обработке LLM должна самостоятельно найти все документы, определить нужный тип сущности, извлечь числовые значения, выполнить усреднение и отсортировать результат. Ошибка возможна на любом из этих этапов.

В графовом подходе задача раскладывается на формальные операции. Сначала выбирается коллекция документов, затем через ребра определяется нужная связанная сущность, после чего выполняется группировка и вычисление среднего значения. AQL-запрос может использовать COLLECT для группировки, AVG для вычисления среднего и SORT для ранжирования. В результате база возвращает компактную таблицу, которую легко проверить.

Принципиальное отличие состоит в том, что языковая модель не «угадывает» итоговое число. Она выбирает способ обращения к данным. Само число получается в результате выполнения запроса к базе. Поэтому при повторном запуске одного и того же запроса результат остается воспроизводимым, что особенно важно для научной и прикладной аналитики.

Аналогично обрабатываются вопросы о худших блоках, доле нулевых оценок, динамике по датам, сравнении групп и поиске отклонений. Для каждого из таких вопросов графовая структура дает возможность явно задать, какие сущности участвуют в расчете и какие связи необходимо учитывать.

Сравнение с векторным RAG

Векторный RAG является распространенным способом работы LLM с внешними данными. Он строит эмбеддинги фрагментов документов, ищет наиболее близкие фрагменты к вопросу пользователя и передает их в контекст модели. Такой подход хорошо подходит для смыслового поиска, поиска цитат,

обобщения текстов и ответов на вопросы, где достаточно нескольких релевантных фрагментов [2].

Однако аналитические вопросы имеют иную природу. Если требуется посчитать среднее значение по всем документам, выбрать минимум и максимум или вычислить процент, то поиск наиболее похожих фрагментов может привести к неполной выборке. Даже если найденные фрагменты релевантны, они не обязательно покрывают весь набор данных. Следовательно, ответ может быть убедительно сформулирован, но статистически неполон.

Графовый подход решает эту проблему за счет явного хранения всех объектов и связей. Запрос к графовой базе может пройти по всему набору документов, применить фильтры и агрегировать значения без ограничения на число извлеченных фрагментов. Поэтому Graph-RAG в данной задаче следует рассматривать не как замену векторному RAG во всех случаях, а как более подходящий вариант для структурно-аналитических вопросов.

На практике наиболее перспективным является гибридный подход. Графовая база может использоваться для точных расчетов и связей между сущностями, а векторный поиск — для извлечения текстовых пояснений, цитат или фрагментов транскриптов. Такая комбинация позволит объединить полноту формальных запросов с гибкостью семантического поиска.

Требования к качеству данных и маппинга

Эффективность графовой аналитики зависит от качества исходной структуры данных. Если документы содержат противоречивые поля, разные варианты написания имен или пропуски в ключевых атрибутах, система должна учитывать эти особенности. Поэтому этап предобработки является не технической формальностью, а важной частью Data Science-пайплайна[8; 9].

Для корректной работы необходимо решать задачу нормализации сущностей. Например, одно и то же имя может быть записано с сокращениями, опечатками или разным порядком слов. Если такие варианты не объединить, граф будет содержать несколько вершин для одной реальной сущности. В

проекте эта проблема частично решается через механизм сопоставления имен и каноникализации сущностей.

Не менее важны правила маппинга. Они определяют, какие поля остаются атрибутами основного документа, какие становятся отдельными вершинами, а какие превращаются в ребра. Неправильный выбор может привести к усложнению запросов или потере аналитической выразительности. Поэтому для новых типов данных целесообразно проводить предварительный анализ структуры и проверять несколько вариантов схемы.

Качество маппинга можно оценивать через тестовые вопросы. Если система стабильно отвечает на вопросы о группировках, связях и агрегатах, значит выбранная модель достаточно хорошо отражает структуру данных. Если же возникают регулярные ошибки, это указывает не только на проблему LLM, но и на возможную недостаточность схемы.

Ограничения эксперимента

Несмотря на положительные результаты, эксперимент имеет ряд ограничений. Во-первых, проверка проводилась на конкретном наборе из 130 JSON-файлов. Этот набор достаточно показателен для демонстрации преимуществ графовой модели, но для обобщения результатов необходимо провести эксперименты на других типах документов и предметных областях.

Во-вторых, оценка качества ответов включала экспертное сравнение результатов. Такой способ подходит для первичной проверки, однако в дальнейшем желательно дополнить его формальными метриками: точностью числовых значений, полнотой выборки, долей корректно распознанных сущностей и устойчивостью ответа при повторном запуске.

В-третьих, качество графового подхода зависит от корректности генерации AQL. Хотя запросы выполняются детерминированно, сам выбор запроса остается задачей LLM. Поэтому необходимо развивать механизмы проверки запросов, ограничения допустимых операций и автоматической регенерации при ошибках.

Наконец, сравнение стоимости зависит от выбранной модели, тарифов и способа учета токенов. Тем не менее относительное снижение числа входных токенов является устойчивым эффектом, поскольку графовый подход принципиально уменьшает объем данных, передаваемых в контекст модели.

Заключение

В работе предложен и экспериментально проверен подход к анализу слабоструктурированных JSON-данных, основанный на интеграции больших языковых моделей и графовой базы данных ArangoDB. Подход разделяет задачи между компонентами системы: база данных выполняет точные операции выборки, фильтрации и агрегации, а LLM отвечает за перевод вопроса в запрос и представление результата на естественном языке.

Эксперимент на 130 JSON-файлах и 10 аналитических вопросах показал, что использование графового слоя позволяет существенно снизить стоимость LLM-аналитики и повысить точность ответов. Стоимость обработки снизилась примерно в 9,4 раза, а качество ответов оказалось выше в большинстве проверочных вопросов.

Дальнейшее развитие работы может быть связано с расширением набора тестовых данных, сравнением с векторным RAG, автоматической оценкой корректности AQL-запросов и созданием универсального механизма построения графовых схем для разных типов JSON-документов.

Список литературы:

1. Vaswani A., Shazeer N., Parmar N. et al. Attention Is All You Need // Advances in Neural Information Processing Systems. 2017.
2. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. 2020.
3. Edge D., Trinh H., Cheng N. et al. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130. 2024.
4. ArangoDB Documentation. AQL Documentation [Электронный ресурс]. URL: https://docs.arangodb.com/3.13/aql/ (дата обращения: 31.05.2026).
5. ArangoDB Documentation. Graphs in AQL [Электронный ресурс]. URL: https://docs.arangodb.com/3.12/aql/graphs/ (дата обращения: 31.05.2026).
6. Hogan A., Blomqvist E., Cochez M. et al. Knowledge Graphs // ACM Computing Surveys. 2021. Vol. 54, No. 4. P. 1-37.
7. Brown T.B., Mann B., Ryder N. et al. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems. 2020.
8. Chen W., Xie Y., Yang Y. et al. A Survey on Large Language Models for Data Management. arXiv preprint. 2023.
9. Gaur M., Faldu K., Sheth A. Semantics of the Black-Box: Can Knowledge Graphs Help Make Deep Learning Systems More Interpretable and Explainable? // IEEE Internet Computing. 2021.
10. Angles R., Arenas M. et al. G-CORE: A Core for Future Graph Query Languages // SIGMOD. 2018.