ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

На правах рукописи

Рябцев Антон Борисович

РАЗВИТИЕ МЕТОДА ДИНАМИЧНОГО ФОРМИРОВАНИЯ ГРУПП ОБЪЕКТОВ ПО ПРИНЦИПУ ИДЕНТИЧНОСТИ ДЛЯ БОЛЬШИХ ДАННЫХ

Специальность 2.3.8 —
«Информатика и информационные процессы (технические науки)»

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель:
д.т.н., профессор
Дулин Сергей Константинович

Москва — 2025

Оглавление

Стр.

Введение

В последние десятилетия наблюдается стремительный рост объёмов обрабатываемых данных, что сопровождается усложнением задач их интеграции, сопоставления и анализа. На фоне развития технологий больших данных, распределённых вычислений и методов искусственного интеллекта одной из ключевых задач становится эффективное объединение идентичных или схожих объектов в группы. Корректная группировка позволяет не только повысить качество аналитических систем, но и существенно оптимизировать процессы принятия решений в бизнесе, логистике, здравоохранении и других отраслях.

Формирование групп идентичных объектов особенно усложняется в условиях высокой динамичности данных. Постоянные изменения в характеристиках объектов, появление новых и удаление уже имеющихся требуют от систем быстрой адаптации и переоценки связей между элементами. При этом классические методы кластеризации, такие как K-means, требуют заранее заданного числа кластеров и поэтому не могут быть непосредственно применены к задаче динамичного формирования групп. Методы на основе плотности, такие как DBSCAN, также имеют ограничения по масштабируемости и чувствительности к выбору параметров. Это приводит к необходимости выбора метода, который способен эффективно обрабатывать данные с заранее неизвестной структурой и числом групп, и его развитие для успешного применения в динамичных системах. Одним из активно развивающихся направлений является исследование методов повышения структурной согласованности в динамических графах. Современные обзоры показывают, что для эффективного обнаружения сообществ в меняющихся сетях необходимы подходы, способные учитывать временные аспекты, разрывы связей и слияния сообществ. Тем не менее, подавляющее большинство существующих решений ориентировано на слабую динамику или требуют значительных ресурсов для регулярного пересчёта кластерной структуры. Помимо задач структурного группирования, критически важной становится проблема быстрого и эффективного расчёта признаков для оценки идентичности объектов. Во многих прикладных задачах такие признаки строятся на основе агрегатов событийных данных, извлекаемых с помощью аналитических SQL-запросов. Однако выполнение сложных аналитических запросов в условиях больших данных зачастую становится узким местом. Современные исследования демонстрируют, что применение методов машинного обучения в оптимизаторах SQL-запросов позволяет добить-

ся существенного повышения их производительности, хотя внедрение подобных решений сопряжено с рядом технологических и практических трудностей.

Таким образом, представленное исследование находится на пересечении нескольких направлений — теории структурной интероперабельности, динамической кластеризации и оптимизации аналитических вычислений. В работе предлагается развитие метода динамичного формирования групп объектов по принципу идентичности с учётом ограничений реального времени, высокой изменчивости системы и необходимости поддержания высокой согласованности групп.

Целью данной работы является развитие метода динамичного формирования групп объектов по принципу идентичности в условиях больших и изменяющихся данных, обеспечивающего высокую структурную согласованность групп при учёте необходимости ускоренного вычисления признаков идентичности.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Анализ существующих методов группировки объектов и выявление их ограничений при работе с большими динамическими данными.
2. Исследование возможностей ускорения аналитических вычислений, необходимых для построения признаков объектов, с применением методов машинного обучения для оптимизации выполнения SQL-запросов.
3. Разработка алгоритма формирования групп объектов, устойчивого к ошибкам в определении идентичности и способного поддерживать согласованную структуру в условиях высокой динамики данных.
4. Создание методики оценки качества сформированных групп с учётом требований к однородности и полноте разбиения.

Таким образом, работа направлена на комплексное решение задачи динамичного и структурно согласованного формирования групп идентичных объектов с учётом особенностей выполнения вычислений в больших динамических системах.

Научная новизна:

1. Впервые проведён комплексный анализ применения методов машинного обучения для оптимизации выполнения аналитических SQL-запросов в условиях изменяющихся данных и высоких нагрузок, что позволило выявить ограничения по их практической применимости в реальных системах обработки больших данных.

2. Обоснован выбор сочетания бинарной классификации для оценки парной схожести объектов и алгоритма распространения меток (Label Propagation Algorithm) для формирования групп как наиболее эффективной комбинации в задачах динамичного объединения семантически идентичных объектов при ограниченных вычислительных ресурсах.
3. Предложен модифицированный двухэтапный алгоритм кластеризации на основе LPA с калибровкой пороговых параметров на основе допустимой доли ошибочных связей (5% и 20%), что позволяет повысить структурную согласованность получаемых групп даже при наличии неполной или шумной информации.
4. Доказана возможность эффективной адаптации предложенного метода динамичной кластеризации для распределённых вычислительных систем, что обеспечивает его масштабируемость и применимость к обработке больших объёмов данных.
5. Разработана комплексная методика количественной оценки качества группировки объектов, включающая показатели однородности и полноты, адаптированные для анализа результатов динамичной кластеризации в условиях больших данных.

Научная и практическая значимость. Теоретическая значимость работы заключается в развитии подходов к динамичному формированию групп объектов по принципу идентичности в условиях больших и изменяющихся данных. В работе обоснован выбор комбинации бинарной классификации и алгоритма распространения меток (LPA) как эффективного решения для задач динамической кластеризации при ограниченных вычислительных ресурсах. Также предложена модификация процесса кластеризации, обеспечивающая повышение структурной согласованности групп за счёт калибровки пороговых параметров на основе анализа допустимой доли ошибок. Разработанная методика оценки качества группировки объектов, учитывающая показатели однородности и полноты, расширяет инструментарий анализа динамических кластеризаций и может быть использована в других задачах обработки больших данных.

Практическая значимость работы заключается в возможности применения разработанных методов и алгоритмов для широкого круга прикладных задач, связанных с обработкой больших динамических данных. К таким задачам относятся: автоматизированное управление ассортиментом товаров, интеллектуальная

агрегация предложений в маркетплейсах, анализ и сопоставление записей в базах данных, системы мониторинга состояния объектов, управление цифровыми двойниками, задачи интеграции данных из разнородных источников, очистка данных от дубликатов и повышение их качества. Разработанный подход адаптирован для распределённых вычислительных систем, что обеспечивает его масштабируемость и позволяет эффективно работать с большими объёмами данных. Проведённые эксперименты подтверждают практическую эффективность предложенных решений на реальных задачах, демонстрируя высокое качество группировки объектов и устойчивость методов к изменениям данных.

Основные положения, выносимые на защиту:

1. Проанализирован модифицированный подход на основе машинного обучения для оптимизации аналитических SQL-запросов в СУБД:
    (a) Выявлена сложность адаптации моделей машинного обучения к изменяющимся данным и нагрузкам в условиях реального времени.
    (b) Выявлены высокие накладные расходы на поддержание актуальности моделей при динамической смене структуры данных.
2. Предложен практико-ориентированный метод повышения качества данных через идентификацию семантически идентичных объектов:
    (a) Предложена комбинированная архитектура, сочетающая модель бинарной классификации для оценки степени схожести объектов и алгоритм распространения меток (LPA) для последующего формирования групп.
    (b) Обоснована возможность адаптации разработанного подхода для распределённых вычислительных систем с использованием парадигмы MapReduce.
    (c) Доказана практическая эффективность предложенного метода на задаче поиска идентичных товарных предложений в условиях реальных маркетплейсов.
3. Предложен модифицированный двухэтапный алгоритм кластеризации на основе LPA:

   (a) Обоснован выбор алгоритма LPA как оптимального решения для задачи формирования групп семантически идентичных объектов.
   (b) Предложена оригинальная модификация LPA с калибровкой параметров на основе анализа доли ошибочных связей между объектами (использованы пороговые значения 5% и 20%).
   (c) Экспериментально подтверждена высокая эффективность предложенного метода при решении практических задач управления ассортиментом в условиях больших динамичных данных.
4. Предложена комплексная методика оценки качества группировки объектов:
   (a) Предложен метод расчёта однородности кластеров как меры внутреннего качества группировки.
   (b) Предложен метод расчёта полноты группировки как меры соответствия найденных групп истинной структуре данных.

Степень достоверности и апробация результатов. Достоверность результатов обеспечивается обширным анализом работ в области исследования, описанием проведённых экспериментов, их воспроизводимостью, апробацией результатов на практике. Основные результаты диссертации докладывались на следующих конференциях: Интеллектуализация обработки информации (ИОИ-2022), Москва, 2022; 66-я Всероссийская научная конференция МФТИ, Долгопрудный, 2024; Интеллектуализация обработки информации (ИОИ-2024), Гродно, 2024; Интеллектуальные информационные технологии для индустрии, федеральная территория ”Сириус” , 2025. Данная работа выполнена в рамках государственного задания номер 103-00001-25-02.

Публикации. Материалы диссертации опубликованы в 8 печатных работах, из них 3 в журналах из списка ВАК и индексируемых в WoS, Scopus.

Личный вклад. Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованные работы. Подготовка к публикации полученных результатов проводилась совместно с соавторами, причём вклад диссертанта был определяющим. Все представленные в диссертации результаты получены лично автором.

Объем и структура работы. Диссертация состоит из введения, пяти глав и заключения. Полный объём диссертации составляет 129 страниц с 33 рисунками и 4 таблицами.

Глава 1. Теоретические основы интероперабельности и структурной согласованности

1.1 Идентичность и интероперабельность

В условиях возрастающей сложности и распределённости современных информационных систем критически важным становится не только их взаимодействие, но и способность точно идентифицировать и соотносить объекты и компоненты. Проблема идентичности — корректного распознавания и сопоставления одинаковых объектов в различных системах — выходит на первый план. Эта проблема имеет глубокие философские корни: Готфрид Лейбниц в XVII веке сформулировал фундаментальные принципы идентичности, согласно которым (1) идентичные объекты неразличимы по своим свойствам (закон неразличимости идентичного), и (2) объекты, неразличимые по всем свойствам, являются идентичными (закон тождества неразличимых). Однако применительно к современным информационным системам эти строгие философские принципы требуют практической адаптации: конкретные критерии идентичности определяются контекстом и целями задачи, а полная неразличимость всех свойств часто недостижима.

При формировании групп идентичных объектов возникает фундаментальное требование: каждый объект в группе должен быть идентичен каждому (свойство попарной идентичности). Это создаёт структуру, аналогичную клике в теории графов, где все вершины попарно соединены. В отличие от более слабых структур типа связных компонент (где существует путь между любыми вершинами, но не обязательно прямое соединение), такая организация гарантирует абсолютную согласованность объектов в группе. Для верификации корректности таких структур особенно продуктивным оказывается применение методов анализа интероперабельности — способности различных систем, компонентов или организаций эффективно обмениваться данными, интерпретировать их и использовать для согласованных действий. Данный подход основан на принципиальной связи между понятиями идентичности и интероперабельности: если идентичность определяет требования к объектам (их неразличимость по заданным критериям), то интероперабельность обеспечивает механизмы проверки выполнения этих требований в условиях реального взаимодействия. Первоначально термин «интероперабельность» (interoperability) возник в военной сфере в середине XX века для обозначения совместной работы вооружённых сил разных стран. Впоследствии это понятие было адаптировано для информационных систем и стан-

дартов информационных технологий. Согласно определению, приведённому в ГОСТ Р 55062-2012 [1], интероперабельность — это «способность двух или более систем или компонентов обмениваться информацией и использовать эту информацию». В научной литературе и практике существует несколько точек зрения на интероперабельность:

- Функциональная интероперабельность — способность систем выполнять совместные операции, независимо от их внутренней реализации.
- Данные и семантика — акцент на том, чтобы данные не только передавались, но и правильно интерпретировались принимающей стороной.
- Процессная интероперабельность — способность бизнес-процессов разных организаций эффективно взаимодействовать на основе согласованных данных.

Таким образом, интероперабельность охватывает не только технический аспект передачи информации, но и вопросы её понимания, осмысленной обработки и координации действий.

Для более точного описания интероперабельности принято выделять уровни интероперабельности [2] (рис. 1.1):

- Организационная интероперабельность связана с координацией бизнес-процессов, политик и процедур между различными организациями или подразделениями, что требует согласованности на уровне целей и процессов.
- Семантическая интероперабельность направлена на обеспечение единого понимания смысла данных всеми участниками взаимодействия. Это требует согласования понятийных моделей, онтологий и схем данных.
- Техническая интероперабельность предполагает возможность обмена данными между системами на уровне сетевых протоколов, форматов сообщений и интерфейсов. Основное внимание здесь уделяется стандартизации протоколов и совместимости программно-аппаратных средств.
- Оценивая потенциальную возможность установления интероперабельности той или иной степени в структуре взаимосвязанных элементов, можно говорить о структурной интероперабельности [3]. Структурная интероперабельность фокусируется на согласованности связей между элементами системы. Здесь важны не столько сами передаваемые данные, сколько способ организации элементов и их взаимосвязей. Структурная интероперабельность

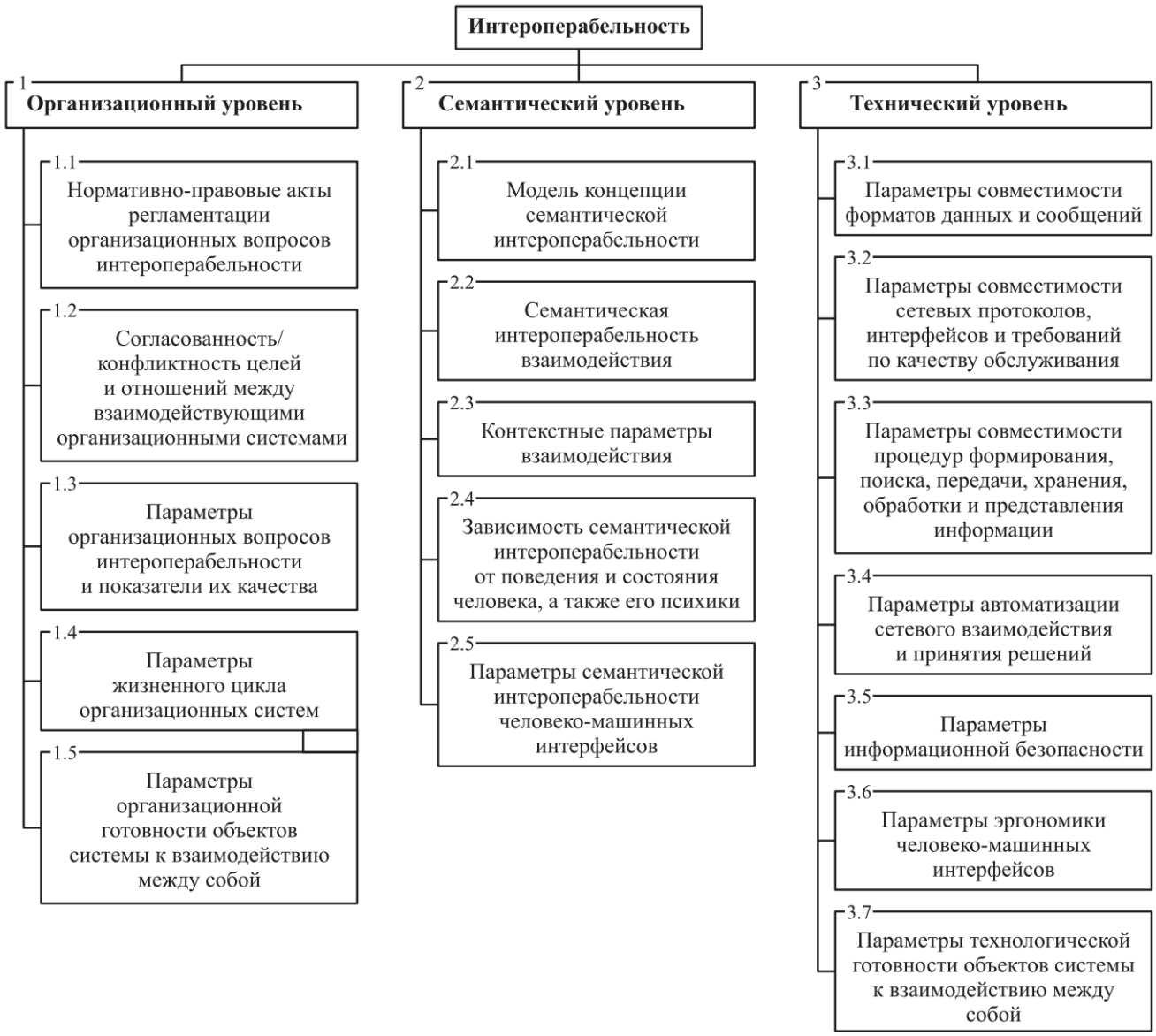


Рисунок 1.1 — Общая структура интероперабельности в соответствии с ГОСТ Р 55062-2012

рабельность определяет, насколько устойчиво и последовательно устроено взаимодействие между группами объектов.

Особое внимание в современных исследованиях уделяется структурной интероперабельности в динамически меняющихся системах, где структура данных и их связи могут изменяться с течением времени. В таких условиях проблема заключается не только в обеспечении первоначальной согласованности, но и в её поддержании при постоянных изменениях состояния системы.

Интероперабельность играет критическую роль в самых разных приложениях: от интеграции медицинских информационных систем и государственных реестров до работы маркетплейсов, банковских систем и промышленных

интернет-платформ. Например, в сфере e-commerce высокая степень интероперабельности между продавцами, маркетплейсами и логистическими компаниями обеспечивает эффективную обработку заказов, актуальность информации о товарах и своевременное выполнение поставок.

Таким образом, понятие интероперабельности стало фундаментальной концепцией для построения современных информационных систем и требует комплексного изучения с учётом технических, семантических, организационных и структурных аспектов. В рамках данного исследования особое внимание уделяется структурной интероперабельности, как наиболее тесно связанной с задачами динамичного формирования согласованных групп объектов.

1.2 Структурная интероперабельность: роль взаимосвязей между объектами

Структурная интероперабельность представляет собой один из наиболее фундаментальных аспектов общей интероперабельности, так как она напрямую связана с организацией связей между отдельными элементами системы. Если техническая интероперабельность обеспечивает возможность передачи данных, а семантическая — их интерпретацию, то структурная интероперабельность отвечает за целостность и согласованность взаимодействия элементов в рамках сложной системы. Основной задачей структурной интероперабельности является обеспечение устойчивости связей между объектами, что особенно критично в условиях динамики данных. В любой сложной системе объекты редко существуют изолированно: они связаны между собой отношениями различной природы — от семантической схожести и сопоставимости до прямых взаимодействий или принадлежности к одним и тем же категориям. Поэтому целостность структуры этих связей оказывает прямое влияние на функциональность и надёжность всей системы. Ключевые принципы структурной интероперабельности включают:

- Связность элементов: объекты, между которыми установлены связи, должны образовывать связные группы, в которых информация может свободно распространяться.
- Однородность связей внутри групп: связи между объектами внутри одной группы должны обладать определённой степенью схожести или согласованности.

– Отделённость групп: объекты, принадлежащие различным группам, должны иметь минимальное количество связей друг с другом для поддержания чёткости границ групп.

В контексте больших данных и систем с высокой динамикой, проблема структурной интероперабельности усложняется рядом факторов:

– Появление новых объектов и исчезновение старых.
– Изменение характеристик объектов и связей между ними.
– Ошибки в определении связей, вызванные неполнотой или неточностью данных.

Все эти явления приводят к так называемым ассонансным структурам, в которых идеальная согласованность нарушена: внутри предполагаемых групп могут существовать ошибочные связи, а между группами могут появляться слабые связи, затрудняющие корректную идентификацию общностей.

В научной литературе структурная интероперабельность изучается как в рамках общих проблем управления качеством данных [4], так и в специализированных направлениях, таких как связь записей (Record Linkage) [5], разрешение сущностей (Entity Resolution) [6] и обнаружение сообществ в графах (Community Detection) [7]. Общим в этих направлениях является стремление выявить устойчивые, внутренне согласованные группы объектов в условиях неполной или изменяющейся информации.

При этом важно учитывать, что в задачах динамичного формирования групп объектов структурная интероперабельность должна быть не только достигнута на момент построения группировки, но и поддерживаться в процессе эволюции системы. Это требует разработки методов, которые могут адаптивно пересчитывать структуру групп при изменении связей между объектами без полного пересмотра всей структуры.

Таким образом, роль взаимосвязей между объектами становится центральной в обеспечении структурной интероперабельности. Качественная организация этих связей позволяет не только повысить точность и надёжность обработки данных, но и значительно упростить последующие этапы интеграции, анализа и принятия решений в сложных информационных системах.

1.3 Понятие структурной согласованности

Структурная согласованность является ключевым понятием при рассмотрении задач, связанных с формированием и поддержанием устойчивых групп объектов в информационных системах. Если структурная интероперабельность отвечает за возможность взаимодействия между элементами системы, то структурная согласованность отражает качество организации этих взаимодействий. В самом общем виде структурная согласованность определяется как степень соответствия фактической структуры связей между объектами некоторому идеальному или целевому состоянию структуры, которое характеризуется максимальной внутренней согласованностью групп и минимальными нарушениями между ними. Идеальная структура для задач группировки объектов соответствует следующим требованиям:

- внутри каждой группы объекты имеют плотные и устойчивые положительные связи, подтверждающие их принадлежность к одной общности;
- между различными группами отсутствуют (или минимальны) связи, что обеспечивает чёткое разделение общностей и предотвращает их смешивание;
- структура устойчива к локальным ошибкам в определении связей и способна сохранять общую согласованность при небольших изменениях данных.

В литературе подобная идеальная структура описывается понятием консонансной структуры — структуры, в которой внутри каждой группы элементы связаны положительно, а между группами связи отсутствуют или являются отрицательными [4; 8]. В реальных условиях данные редко соответствуют этому идеалу, и фактическая структура, как правило, имеет ассонансный характер: существуют ошибочные связи между группами и недостающие связи внутри групп.

Ключевыми факторами, влияющими на степень структурной согласованности, являются:

- точность определения связей между объектами (например, правильная оценка идентичности или схожести);
- шум в данных — наличие ошибок, пропусков или несовершенных измерений;
- изменчивость данных — добавление, удаление или изменение характеристик объектов и связей в динамике времени.

Оценка структурной согласованности осуществляется с использованием различных метрик качества кластеризации, таких как однородность, полнота, индекс Рэнда с поправкой на случайность, оценка Фаулкса-Мэллоуза и других показателей [9—13]. Эти метрики позволяют количественно определить, насколько хорошо фактическая структура групп соответствует ожидаемому разбиению.

В контексте задач динамичного формирования групп объектов задача повышения структурной согласованности заключается не только в построении изначально качественной структуры, но и в её корректной адаптации при изменении данных. Особенно важными становятся методы, позволяющие оперативно обнаруживать и устранять нарушения согласованности — например, разрывы внутри групп или появление нежелательных связей между различными группами. Таким образом, понятие структурной согласованности связывает между собой вопросы точности определения связей между объектами и организацию их в устойчивые группы, обеспечивая основу для эффективной работы сложных информационных систем в условиях больших динамичных данных.

1.4 Критерии и методы оценки согласованности структуры

Оценка степени структурной согласованности является важной частью задач группировки объектов в информационных системах. Корректная оценка качества групп позволяет судить о надёжности сформированной структуры, выявлять ошибки и принимать решения о необходимости её пересмотра или корректировки. Для количественного анализа структурной согласованности разработан ряд критериев, основанных как на внутренних свойствах групп, так и на сопоставлении полученного разбиения с эталонным. Основные критерии оценки согласованности структуры включают:

- Однородность (Homogeneity) отражает степень, в которой все элементы внутри одной группы принадлежат к одному истинному классу. Группа считается идеально однородной, если все её элементы действительно идентичны или схожи по целевому признаку. Однородность важна для оценки внутреннего качества группировки.
- Полнота (Completeness) показывает, насколько полно все объекты, принадлежащие одному истинному классу, были собраны в одну группу. Высокая полнота означает, что практически нет ”разброса” идентичных объектов по разным группам.

– V-мера (V-measure) — гармоническое среднее между однородностью и полнотой. Этот показатель позволяет одновременно учитывать стремление к высокой внутренней чистоте групп и к полноте охвата объектов.
– Индекс Рэнда с поправкой на случайность (Adjusted Rand Index, ARI) измеряет степень совпадения между истинным разбиением объектов и полученным разбиением, учитывая вероятность случайных совпадений.
– Оценка Фаулкса-Мэллоуза (Fowlkes–Mallows Index, FMI) также учитывает пары объектов и оценивает, насколько пара объектов, помещённая в одну группу в одном разбиении, помещена в ту же группу в другом разбиении.

Каждая из этих метрик имеет свою область применимости и особенности интерпретации. Например, однородность может быть высокой даже при низкой полноте (если группы содержат очень мало объектов), а высокая полнота может сопровождаться низкой однородностью в случае объединения слишком разных объектов.

Методы расчёта критериев основаны на анализе распределения объектов по группам, числа пар объектов, сгруппированных вместе или раздельно в истинном и полученном разбиениях, сопоставления структуры связей между объектами с ожидаемым паттерном связности.

Для динамичных систем особую важность приобретает оценка стабильности структурной согласованности во времени. Это требует не только анализа качества разбиения в фиксированный момент, но и отслеживания изменений согласованности при поступлении новых данных или обновлении характеристик объектов.

В литературе также рассматриваются подходы к восстановлению или улучшению структурной согласованности, включая методы перепроверки сомнительных связей, перестройки локальных участков графа или пересчёта кластерной структуры с использованием дополнительных источников данных [7; 14]. Таким образом, критерии и методы оценки согласованности структуры обеспечивают необходимую основу для формального анализа качества группировки объектов и позволяют диагностировать текущую структуру.

1.5 Постановка задачи динамичного формирования согласованных групп объектов

В условиях постоянно изменяющихся данных классическая задача формирования групп идентичных объектов усложняется динамичностью самой системы. Появление новых объектов, удаление существующих, изменение характеристик — всё это требует от алгоритмов группировки не только способности строить качественные группы на начальном этапе, но и механизмов поддержания согласованности структуры на протяжении жизненного цикла системы.

Постановка задачи заключается в следующем. Имеется множество объектов:

$$I_t = \{i_1, i_2, \ldots, i_{N_t}\},$$

существующее на момент времени t, где каждый объект характеризуется множеством признаков:

$$D_t = \{d_{i1}, d_{i2}, \ldots, d_{iN_t}\}.$$

Требуется для каждого момента времени t построить разбиение объектов на группы:

$$C_t = \{c_1, c_2, \ldots, c_{n_t}\},$$

такое, что:

- внутри каждой группы c_k находятся объекты, идентичные или практически идентичные по заданной мере семантической близости;
- между различными группами отсутствуют связи, свидетельствующие об идентичности объектов;
- структура групп устойчива к локальным ошибкам в данных и может быть адаптивно обновлена при изменении характеристик объектов или поступлении новых данных.

Основные особенности задачи:

- **Отсутствие априорного знания о числе групп.** Количество групп n_t заранее неизвестно и должно определяться автоматически на основе структуры данных.
- **Динамика системы.** Набор объектов и их характеристики изменяются во времени, что требует переоценки группировок без полного пересчёта всех данных.

- **Необходимость высокой вычислительной эффективности.** Методы должны обеспечивать приемлемое время обработки при больших объёмах данных.
- **Учёт ошибок и шумов.** Ошибочные связи между объектами и неточности характеристик неизбежны и должны учитываться в моделях.
- **Поддержание высокой структурной согласованности.** Критерии однородности и полноты должны оставаться на высоком уровне в процессе эволюции данных.

Формальная цель заключается в построении алгоритма, который, используя модель оценки семантической близости объектов и алгоритмы кластеризации на графах, обеспечивает:

- высокую однородность внутри групп;
- высокую полноту объединения идентичных объектов;
- масштабируемость обработки больших данных;
- устойчивость к изменениям структуры системы.

В рамках данной работы в качестве решения рассматривается подход, основанный на сочетании:

- бинарной классификации пар объектов для оценки степени их идентичности;
- алгоритма распространения меток (Label Propagation Algorithm, LPA) для эффективного формирования групп в графе бинарных связей;
- модифицированных процедур калибровки точности для повышения качества кластеризации в условиях наличия ошибок в данных.

Таким образом, сформулированная задача требует интеграции методов машинного обучения, графовых алгоритмов и динамического управления структурой данных для достижения высокой степени интероперабельности и согласованности групп в условиях работы с большими динамическими данными.

Выводы к первой главе

В данной главе рассмотрены теоретические положения, лежащие в основе задачи динамичного формирования групп объектов по принципу идентичности. Было проанализировано понятие интероперабельности в широком смысле, а также выделено ключевое направление — структурная интероперабельность,

как наиболее релевантная категория задач по организации взаимодействий между объектами в условиях больших данных.

Показано, что важнейшей характеристикой структуры системы является её согласованность — способность формировать устойчивые, однородные группы с минимальным количеством ошибок и шумов. Структурная согласованность рассматривается как необходимое условие эффективного функционирования распределённых и динамичных систем, где объекты постоянно изменяются, добавляются или удаляются.

Были подробно рассмотрены основные критерии и методы оценки качества группировок: однородность, полнота, V-мера и другие. Эти меры качества позволяют формально оценить степень близости текущей структуры к идеальной.

В завершение была сформулирована задача динамичного формирования согласованных групп: для множества объектов с изменяющимися характеристиками требуется построить устойчивую кластерную структуру, способную адаптироваться к изменениям и обеспечивать высокое качество группировки.

Рассмотренные в первой главе теоретические положения служат основой для последующего анализа методов оптимизации вычислений. Результаты главы опубликованы в работе [15].

Глава 2. Определения схожести объектов и проблемы вычисления признаков в задаче динамичного формирования групп

2.1 Задача оценки схожести и идентичности

Одним из основных этапов в задаче динамичного формирования групп объектов по принципу идентичности является построение механизма оценки схожести между объектами. Эта задача возникает в различных прикладных областях: от объединения товарных предложений в маркетплейсах до слияния дублирующихся записей в реестрах и базах данных, сопоставления событий в логах, идентификации сущностей в текстах и других. На интуитивном уровне под идентичностью понимается факт того, что два объекта описывают одну и ту же сущность в предметной области. В информационных системах идентичность не всегда может быть установлена напрямую — чаще всего приходится делать вывод о ней косвенно, на основании набора признаков, которые объекты имеют. Поэтому первым этапом становится построение численной меры схожести, позволяющей определить, насколько близки два объекта друг к другу. Эта мера, по сути, является приближённой оценкой вероятности их идентичности.

В общем случае представим описания каждого элемента в виде набора из q признаков-атрибутов: $d_i = (p_1^i, \cdots, p_q^i)$. Сравнивая любые два элемента этого множества на основе этих признаков, можно оценить их сходство. Для двух элементов j и k, сходство которых устанавливается на основе m числовых признаков $\{p_q\}$, функция сходства может выглядеть так [3; 4]:

$$F(j,k) = 1 - \frac{1}{m}\sum_{z=1}^{q} w_z \frac{|p_z^j - p_z^k|}{\max\limits_{j,k}\left|p_z^j - p_z^k\right|},$$

где $0 \le w_z \le 1$ – вес z-го признака, а $\max\limits_{j,k}\left|p_z^j - p_z^k\right|$ – диапазон значений z-го признака. Функция F принимает значения из $[0,1]$ так, что $F = 1$ – это абсолютное сходство элементов j и k, а $F = 0$ – абсолютное различие. На основе значений функции $F(j,k)$ строится матрица схожести:

$$M \in \mathbb{R}^{N\times N}, \quad M_{jk} = F(j,k),$$

где N — общее число объектов. Эта матрица симметрична, но в большинстве прикладных случаев — разреженная: схожесть рассчитывается лишь для пар, про-

шедших предварительный отбор (например, по граничным значениям категориальных признаков или предварительным эвристикам). Это позволяет существенно снизить вычислительные затраты на построение матрицы. Для принятия решения о том, считать ли пару объектов идентичной, используется порог $\tau \in [0,1]$. Если

$$F(j,k) \geq \tau,$$

то объекты j и k считаются идентичными, и между ними устанавливается связь. Этот шаг преобразует матрицу схожести в граф, где вершины — объекты, а рёбра отражают идентичные (по мнению модели) пары. Выбор значения τ имеет ключевое значение. Слишком высокий порог ведёт к высокой точности, но низкой полноте — идентичные объекты могут остаться несвязанными. Слишком низкий — к высокой полноте, но с ростом ложных объединений. Порог подбирается в процессе оптимизации ключевой меры качества (Precision или Recall) с использованием размеченной выборки.

На практике возможны различные ошибки, связанные с погрешностями оценки схожести:

- **Ложноположительные ошибки (false positives)** — объединение неидентичных объектов в одну группу. Это приводит к снижению однородности кластеров.
- **Ложноотрицательные ошибки (false negatives)** — нераспознание идентичных объектов как пары. Это снижает полноту группировки.

Частота и распределение таких ошибок оказывают существенное влияние на последующую кластеризацию и её согласованность.

Возможны альтернативные подходы к построению функции $F(j,k)$, в том числе:

- **Обучаемые модели** — логистическая регрессия, градиентный бустинг, нейросети, обучаемые на размеченных парах объектов;
- **Эвристические правила** — использование бизнес-логики или ручных правил;
- **Метрики векторов признаков** — косинусная мера, расстояние Минковского и другие.

Выбор конкретной функции зависит от доступных данных, требований к скорости, интерпретируемости и возможности масштабирования.

Таким образом, задача оценки схожести между объектами является основополагающей в рамках более широкой задачи формирования согласованных групп. Именно на этом этапе формируются связи, которые в дальнейшем будут преобразованы в кластеры с использованием графовых алгоритмов. Качество, устойчивость и масштабируемость меры схожести в значительной степени определяют успешность всей системы группировки.

2.2 Методы вычисления схожести по признакам

Оценка схожести между объектами лежит в основе всей процедуры их группировки по принципу идентичности. От того, насколько корректно построена функция сходства, зависит точность формирования кластеров, устойчивость групп к шумам, а также способность алгоритма адаптироваться к изменениям данных. Поскольку объекты в реальных системах, как правило, описываются набором признаков различной природы, задача сводится к построению меры близости в признаковом пространстве. В этом разделе рассматриваются существующие подходы к определению схожести между объектами по признакам, даётся их классификация, сравнительный анализ, а также обсуждаются их применимость и ограничения в условиях динамичного формирования групп.

Метрики и функции расстояния — методы данной группы базируются на представлении объектов в виде векторов признаков и применении к ним стандартных метрик. Они особенно распространены при работе с числовыми или бинарными признаками. Примеры наиболее часто используемых метрик:

– **Евклидово расстояние**:

$$d(i,j) = \sqrt{\sum_{k=1}^{m}(p_{ik} - p_{jk})^2},$$

используется при равнозначных признаках и одинаковом масштабе.

– **Манхэттенское расстояние**:

$$d(i,j) = \sum_{k=1}^{m}|p_{ik} - p_{jk}|,$$

более устойчиво к выбросам, чем евклидово.

– **Косинусное сходство**:

$$S(i,j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{\|\mathbf{d}_i\| \cdot \|\mathbf{d}_j\|},$$

применяется, когда важна форма распределения признаков, а не абсолютные значения.

– **Jaccard-индекс** — особенно эффективен для бинарных признаков и множества категорий.

– **Hamming-дистанция** — используется при сравнивании битовых строк и категориальных признаков с фиксированными наборами значений.

Метрики просты, интерпретируемы и легко реализуемы. Однако они предполагают наличие нормализованных, числовых признаков и равнозначность их вклада. При нарушении этих предпосылок результат может быть неадекватным.

Во многих прикладных задачах используются заранее заданные правила, отражающие специфику предметной области. Например, в системах товарного сопоставления может применяться правило: ”если название, производитель и объём совпадают, а цена отличается не более чем на 10%, объекты считаются идентичными”. Такие подходы легко реализуются, обладают высокой интерпретируемостью, позволяют жёстко контролировать ошибки. Недостатки эвристик: плохо адаптируются к изменяющимся данным, плохо масштабируются — с ростом числа признаков экспоненциально растёт сложность логики, низкая полнота — многие идентичные объекты могут не попадать под правила. Эвристики особенно уязвимы в условиях шумных и неполных данных, а также в задачах, где трудно формализовать правила без привлечения статистики.

Современные подходы всё чаще используют обучение моделей на размеченных парах объектов. Каждый объект описывается набором признаков, а модель обучается различать пары: *match* (идентичны) и *non-match* (различны). Примеры таких моделей:

– логистическая регрессия — интерпретируема, хорошо работает на линейно разделимых данных;

– градиентный бустинг (XGBoost, CatBoost) — способен учитывать нелинейные зависимости и эффективно работать с категориальными признаками;

– нейронные сети — позволяют обрабатывать векторные представления объектов, тексты и сложные зависимости;
– двухбашенные архитектуры — обучаются на парных сравнениях и возвращают меру схожести.

Преимущества обучаемых моделей: высокая точность и адаптивность, возможность автоматического определения важности признаков, устойчивость к шумам при достаточном объёме данных. Недостатки: необходимость размеченной обучающей выборки, переобучение на частные случаи, сложность интерпретации (особенно у нейросетей), высокая стоимость поддержки в условиях изменяющихся данных. Особенно критично это в системах, где структура и объёмы данных меняются ежедневно: модель быстро теряет актуальность, а переобучение требует автоматизации и вычислительных ресурсов.

На практике часто используются комбинации перечисленных подходов [14; 16; 17]: эвристика как фильтр кандидатов, модель для ранжирования и отбора N наиболее перспективных кандидатов (при необходимости), обученная модель бинарной классификации. Такие методы позволяют добиться высокого качества при разумных затратах, особенно если грамотно разделены этапы отбора кандидатов и проверки отобранных кандидатов.

2.3 Роль аналитических запросов в формировании признаков

В предыдущих разделах рассматривались методы оценки схожести между объектами, основанные на признаковом описании. Однако для большинства реальных задач критическим этапом является не только выбор функции схожести, но и построение самих признаков. Во многих случаях признаки не являются непосредственно хранимыми в исходных данных, а формируются путём агрегации, фильтрации и объединения событий, связанных с объектом. Это делает задачу формирования признаков тесно связанной с выполнением аналитических запросов, в первую очередь — SQL-запросов к хранилищам событий и логов.

Рассмотрим ряд примеров признаков, часто используемых в системах, где требуется сопоставление или группировка объектов:

– **Частотные признаки**: общее число событий, связанных с объектом (например, количество покупок товара, количество просмотров, количество запросов к API).

- **Временные признаки**: дата первого и последнего события, среднее время между событиями, сезонность активности.
- **Агрегаты по параметрам**: средняя цена покупок, медианное значение определённого параметра, количество уникальных пользователей, взаимодействовавших с объектом.
- **Поведенческие профили**: доля событий определённого типа, стабильность активности.
- **Групповые признаки**: статистики по отношению к другим объектам (например, насколько цена объекта выбивается из среднего по категории).

Получение каждого такого признака требует выполнения агрегатных операций по подмножеству большого объёма данных. Как правило, события хранятся в распределённых хранилищах, что делает выполнение даже одного аналитического запроса ресурсоёмким. В динамично меняющейся системе, где объекты и события появляются или обновляются постоянно, возникает необходимость регулярного пересчёта признаков.

Главная особенность динамичной среды заключается в том, что признаки объектов не являются статичными. Изменения в событиях (например, новый заказ, обновление карточки, регистрация пользователя) могут повлиять на признаки нескольких объектов. Это создаёт несколько проблем:

- **Частота пересчёта**: признаки должны быть пересчитаны при каждом значимом изменении данных — в идеале в реальном времени или с минимальной задержкой.
- **Объём данных**: даже один запрос на пересчёт признаков для одного объекта может требовать чтения и агрегации миллионов строк событий.
- **Нагрузка на систему**: массовый пересчёт признаков по многим объектам создаёт значительную нагрузку на базу данных или аналитическое хранилище.
- **Задержка в обновлении модели**: если признаки обновляются медленно, модель начинает работать с устаревшими данными, что снижает точность оценки схожести.

В совокупности эти факторы приводят к тому, что именно процесс формирования признаков становится бутылочным горлышком в задаче динамичного формирования групп. Даже при наличии качественной модели оценки схожести, своевременное получение входных данных для неё оказывается затруднено. Существуют

различные направления, позволяющие частично смягчить вычислительные затраты:

- использование материализованных представлений и предварительных агрегаций (но они требуют дополнительного места и усложняют актуализацию);
- кэширование признаков с частичным обновлением;
- ограничение глубины истории для агрегаций;
- переход от SQL-кластеров к специализированным системам потоковой аналитики (например, Apache Flink, Apache Druid).

Однако каждое из решений связано с компромиссами по точности, сложности поддержки или затратам на инфраструктуру. Кроме того, в некоторых системах (например, построенных на классических СУБД) невозможно полностью уйти от SQL-запросов. Это делает задачу оптимизации и ускорения аналитических запросов принципиально важной.

Выводы ко второй главе

Указанные выше проблемы являются предметом рассмотрения в следующей главе, где описаны исследованные методы ускорения аналитических запросов и сделаны выводы об их применимости в контексте динамичного формирования групп идентичных объектов. Несмотря на то, что не удалось получить универсально применимое решение, анализ существующих подходов позволил точнее сформулировать требования к архитектуре системы и уточнить границы применимости предложенного метода.

Глава 3. Ускорение выполнения аналитических SQL-запросов как элемент преобразования данных

Цель данной главы — исследовать возможности повышения эффективности выполнения аналитических запросов, применяемых для вычисления признаков. Рассматриваются существующие подходы к оптимизации запросов в централизованных строчных СУБД и предлагается их доработка для применения к массово-параллельным колоночным СУБД. Оцениваются их пределы применимости, а также предлагаются практические методики, основанные на машинном обучении и профилировании нагрузки. Полученные результаты позволяют обосновать архитектурные ограничения последующих этапов работы и определить классы задач, в которых методы динамичного формирования групп объектов могут быть реализованы эффективно. В рамках настоящего исследования эксперименты проводились как на СУБД с открытым исходным кодом (PostgreSQL), так и на промышленной проприетарной СУБД (GaussDB), что позволило оценить применимость подходов в условиях, приближенных к реальной эксплуатации.

3.1 Анализ существующих подходов к оптимизации запросов

3.1.1 Традиционные методы

На рисунке 3.1 представлен традиционный [18] метод, базирующийся на расчёте стоимостных оценок. Для формирования оптимального плана выполнения запроса оптимизатор анализирует множество возможных вариантов соединения, применяя, в частности, алгоритмы динамического программирования. Опираясь на предварительно вычисленные оценки кардинальности, стоимостная модель выбирает наиболее эффективный вариант среди семантически равнозначных планов. В идеальном случае, при абсолютно точных оценках кардинальности и

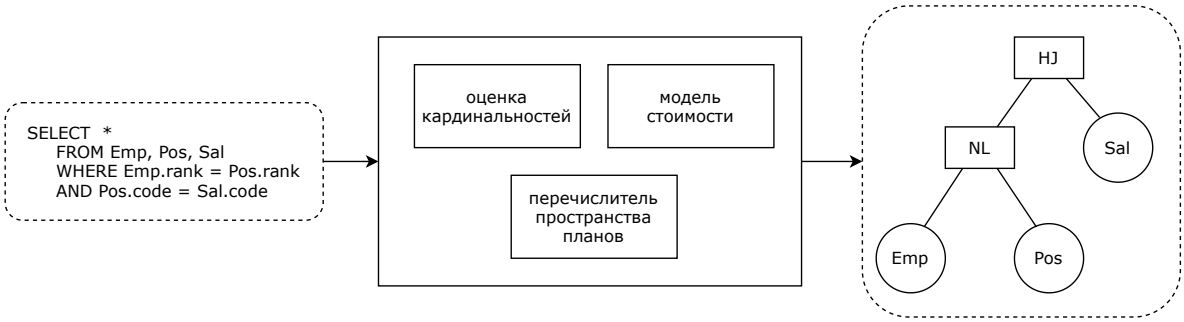


Рисунок 3.1 — Архитектура традиционного оптимизатора запросов.

корректной стоимостной модели, данная архитектура обеспечивает выбор наи-

лучшего плана запроса. Однако на практике вычисление кардинальности часто основывается на упрощённых допущениях, таких как равномерное распределение данных и отсутствие зависимостей между атрибутами. В реальных базах данных эти предположения обычно не выполняются, что может приводить к неоптимальным, а в некоторых случаях и к крайне неэффективным планам выполнения запросов. Компонент оценки кардинальности (CardEst) является ключевым элементом в процессе оптимизации запросов [19]. Его основная задача заключается в прогнозировании количества строк для всех промежуточных результатов запроса, что позволяет оптимизатору выбирать наиболее эффективные операции соединения. Качество итогового плана выполнения запроса напрямую зависит от точности оценок, предоставляемых CardEst. CardEst представляет собой важнейший компонент СУБД, что объясняет пристальный интерес к этой технологии со стороны исследователей [19; 20] и корпоративных разработчиков [21—23]. Современные СУБД с открытым исходным кодом и коммерческие СУБД в основном используют один из двух традиционных методов CardEst: гистограммы [24—29] в PostgreSQL [30] и SQL Server [31], сэмплирование [32—36] в MySQL и MariaDB [37].

В научных публикациях CardEst чаще всего рассматривается как статистическая проблема. Рассмотрим таблицу T, содержащую k атрибутов $A = A_1, A_2, \ldots, A_k$. Здесь T может представлять как самостоятельную реляционную таблицу, так и промежуточный результат соединения нескольких таблиц. В данном исследовании предлагается исходить из допущения, что любой атрибут A_i (где $1 \leq i \leq k$) принадлежит к одному из двух типов: категориальный (его значения допускают целочисленное кодирование) или непрерывный, с областью допустимых значений D_i. Любой запрос выбора Q к таблице T может быть формализован в стандартном виде: $Q = \{A_1 \in R_1 \wedge A_2 \in R_2 \wedge \cdots \wedge A_n \in R_n\}$, где $R_i \subseteq D_i$ определяет ограничивающее условие запроса для атрибута A_i. Без ограничения общности $R_i = D_i$, если Q не накладывает ограничений на A_i. Обозначим через $Card(T, Q)$ истинную кардинальность - точное число записей в T, удовлетворяющих всем условиям Q. Суть задачи CardEst заключается в максимально точном приближении значения $Card(T, Q)$ без фактического выполнения запроса Q к таблице T.

Гистограммы. В системах, использующих для оценки кардинальности подход на основе гистограмм, мощность базовых таблиц оценивается с помощью гистограмм (квантильная статистика), наиболее распространённых значений с их частотами и кардинальности областей определения (количества различных значений). Эта статистика по атрибутам вычисляется заранее с использованием выборки таблицы. Для сложных предикатов, где нельзя применять гистограммы, система прибегает к специальным методам, которые теоретически не обоснованы. Чтобы объединить конъюнктивные предикаты для одной и той же таблицы, СУБД просто предполагает независимость и перемножает селективности отдельных оценок.

Полученные размеры объединений двух таблиц оцениваются по формуле

$$|T_1 \bowtie_{x=y} T_2| = \frac{|T_1||T_2|}{\max(\text{dom}(x), \text{dom}(y))}, \tag{3.1}$$

где T_1 и T_2 - произвольные таблицы, а $\text{dom}(x)$ - мощность области определения атрибута x, то есть количество различных значений x.

Сэмплирование. В системах, использующих для оценки кардинальности подход на основе сэмплирования, мощность базовых таблиц оценивается с помощью подвыборок. Для набора данных также называемого «популяцией», и запроса с неизвестным числовым результатом, который мы хотим оценить, сначала случайным образом выбирается небольшое количество элементов из набора данных. Затем по выборке вычисляется несколько статистических данных, таких как среднее по выборке и дисперсия. Наконец, эта статистика используется для оценки значения результата запроса и определения границ точности оценки.

Задача оптимизатора (планировщика) — построить наилучший план выполнения. Один и тот же SQL-запрос может быть выполнен множеством различных способов, гарантируя при этом идентичные результаты. Когда вычислительные затраты остаются приемлемыми, оптимизатор запросов анализирует все возможные варианты выполнения, выбирая среди них наиболее эффективный по времени. На первом этапе оптимизатор генерирует планы сканирования для каждого отдельного отношения (таблицы), участвующего в запросе. Доступные варианты планов определяются наличием индексов в соответствующих отношениях. Поскольку последовательное сканирование (SeqScan) возможно для

любой таблицы, этот вариант всегда включается в множество рассматриваемых планов. Предположим, для отношения создан индекс и запрос содержит ограничение **<отношение.атрибут ОПЕРАТОР константа>**. Если окажется, что **<отношение.атрибут>** совпадает с ключом индекса-B-дерева и **<ОПЕРАТОР>** — один из операторов, входящих в класс операторов индекса, создаётся ещё один план, с использованием индекса-B-дерева для чтения отношения. Если находятся другие индексы, ключи которых соответствуют ограничениям запроса, могут добавиться и другие планы. Для индексов генерируются планы сканирования (IndexScan) в тех случаях, когда их порядок сортировки совпадает с условием ORDER BY (при его наличии) или когда этот порядок может быть полезен для последующего соединения слиянием (Merge Join/MJ).

Когда запрос предполагает соединение нескольких отношений, после формирования всех возможных планов сканирования для отдельных таблиц система переходит к анализу стратегий соединения. Доступны три основных метода:

- Соединение вложенными циклами (Nested Loop/NL) предполагает многократное сканирование правого отношения для каждой строки левого отношения. Несмотря на простоту реализации, данный подход может оказаться ресурсоёмким. Однако его эффективность значительно повышается, если для правого отношения доступно сканирование по индексу, когда значения из левого отношения выступают в качестве ключей поиска.
- Соединение слиянием (Merge Join/MJ) требует предварительной сортировки обоих отношений по атрибутам соединения. После этого происходит параллельное сканирование обоих отношений с объединением соответствующих строк. Этот метод привлекателен однократным сканированием каждого отношения. Необходимый порядок сортировки может быть достигнут либо явной операцией сортировки, либо использованием индекса по ключу соединения.
- Соединение по хешу (Hash Join/HJ) начинается с построения хеш-таблицы по атрибутам соединения правого отношения. Затем левое отношение сканируется, и для каждой строки вычисляется хеш-ключ для поиска соответствий в подготовленной хеш-таблице. Этот метод особенно эффективен при работе с большими объёмами данных.

Когда количество таблиц в запросе не превышает заданного порога, система применяет стратегию практически полного перебора для нахождения оптималь-

ного порядка соединений. В процессе планирования особый приоритет отдаётся соединениям между таблицами, для которых явно указаны условия соединения в предложении **WHERE** (в форме ограничений типа **<таблица1.атрибут1 = таблица2.атрибут2>**). Соединения между таблицами без явных условий в **WHERE** рассматриваются лишь в крайнем случае - когда для некоторой таблицы отсутствуют какие-либо условия соединения с другими таблицами. Для каждой допустимой пары таблиц планировщик анализирует все возможные варианты выполнения соединения, выбирая среди них наиболее эффективный согласно внутренним оценкам стоимости. Такой подход обеспечивает нахождение качественного плана выполнения даже при ограниченном переборе вариантов. Когда число таблиц в запросе становится больше установленного предела, порядок соединений определяется с помощью эвристик на основе генетического алгоритма. Остальные этапы построения плана остаются без изменений. Итоговое дерево плана включает узлы сканирования (индексного или полного для исходных таблиц) и при необходимости узлы соединений трёх типов: вложенными циклами, слиянием или хешированием. Оптимизатор при оценке планов принимает во внимание вычислительные ресурсы, необходимые для выполнения запроса.

Стоимость — это внутренний числовой показатель, представляющий расчетное использование ресурсов для плана. Стоимость зависит от запроса в среде оптимизатора. Для оценки стоимости оптимизатор учитывает следующие факторы:

1. Системные ресурсы, в том числе расчетный ввод-вывод, ЦП и память.
2. Расчетное количество возвращённых строк (количество элементов).
3. Размер исходных наборов данных.
4. Распределение данных.
5. Структуры доступа.

Стоимость соединения представляет собой комбинацию индивидуальных затрат на доступ двух соединяемых наборов строк, а также стоимость операции соединения.

3.1.2 Учёт особенностей GaussDB

GaussDB — это распределенная база данных, использующая архитектуру массово-параллельной обработки и поддерживающая колоночное хранение данных. Из-за колоночного хранения стандартные типы чтения данных

(SeqScan и IndexScan) здесь не применимы, их заменяет колоночное считывание (CStoreScan), а соединение слиянием становится в принципе невозможным. Основное назначение — выполнение аналитических запросов, подразумевающее соединение большого числа крупных таблиц. Поэтому NestLoop практически всегда является неоптимальным выбором, и его просто отключают. Из всех стратегий соединения таблиц остаётся только одна — HashJoin.

Также распределённость данных делает невозможным производить соединение таблиц сразу. Изначально база данных распределена по какому-то ключу. Если соединение таблиц происходит по ключам, отличным от ключа партицирования, требуется перераспределить данные, что делается одним из двух способов:

1. Broadcast — пересылает данные выбранной таблицы по всем дата-нодам (эффективно для маленьких таблиц).
2. Redistribute — пересылает данные выбранной таблицы по конкретным дата-нодам, что требует дополнительных вычислений, но работает быстрее первого для больших таблиц.

Если в условии соединения **<where табл1.атр1=табл2.атр2>** ни **атр1**, ни **атр2** не являются ключом изначального распределения данных, то нужно либо произвести Broadcast любой из двух таблиц, либо Redistribute обеих. Если же данные распределены по одному из ключей соединения, то вариант с двумя Redistribute заменяется на Redistribute второй таблицы.

3.2 Методы повышения эффективности выполнения

В последние годы исследователи в области баз данных активно изучают применение нейросетевых подходов для модернизации оптимизаторов запросов. Основное направление этих работ связано с заменой стандартных компонентов оптимизатора обученными моделями. В частности, такие системы как DQ [38] и ReJOIN [39] применяют методы обучения с подкреплением, комбинируя их с классическими стоимостными функциями, что позволяет автоматически выявлять эффективные стратегии поиска и анализировать пространство вариантов соединения. Результаты исследований демонстрируют, что подобные обученные стратегии могут превосходить традиционные эвристические подходы при использовании той же стоимостной функции. Однако следует отметить, что помимо основной стоимостной модели, данные системы продолжают использовать эври-

стические методы для оценки кардинальности, выбора физических операторов и определения оптимальных индексов.

Другие подходы демонстрируют, как можно использовать машинное обучение для получения более точных оценок кардинальности. Однако ни один из них не демонстрирует, что их улучшенные оценки количества элементов на самом деле приводят к лучшим планам выполнения запросов. Относительно легко улучшить среднюю ошибку оценок количества элементов, но гораздо сложнее улучшить оценки для случаев, которые фактически улучшают планы запросов.

Важно отметить, что в отличие от оптимизации порядка соединений, выбор конкретных алгоритмов соединения (таких как хеш-соединение или сортировка слиянием) и определение оптимальных индексов не могут основываться исключительно на оценках кардинальности. Исследование SkinnerDB [40] продемонстрировало потенциал применения обучения с подкреплением для адаптивных стратегий обработки запросов, однако такая методика требует специализированной системы выполнения запросов, способной к динамической адаптации.

Исследование, представленное в работе ”Neo: A Learned Query Optimizer” [41], демонстрирует, что данный интеллектуальный оптимизатор запросов достигает сравнимой или превосходящей производительности относительно ведущих коммерческих решений (Oracle и Microsoft). Система Neo функционирует следующим образом:

1. Использует набор правил преобразования запросов для сохранения семантической корректности.
2. Автоматически обучается принимать решения относительно оптимального порядка соединений, а также выбора алгоритмов соединения и подходящих индексов.

Для оптимизации этих решений Neo применяет как метод обучения с учителем, так и механизм обратной связи. Данный подход позволяет системе адаптироваться к конкретному экземпляру базы данных, принимая решения на основе фактических показателей времени выполнения запросов.

В данной работе исследовалась применимость многих из этих подходов к массово-параллельной СУБД. Опробованы как методы, основанные на предсказании кардинальности, так и на основе предсказания стоимостей.

3.2.1 Подходы к улучшению оценки кардинальности

Современные подходы к оценке кардинальности, использующие машинное обучение, направлены на построение моделей, непосредственно прогнозирующих значение $Card(T, Q)$ для произвольного запроса Q. Наиболее продвинутые реализации данной концепции применяют сложные алгоритмы, включая глубокие нейронные сети и ансамбли на основе градиентного бустинга, что позволяет существенно повысить точность оценок.

В отличие от запросо-зависимых методов, подходы, основанные на моделировании данных, не требуют анализа конкретных запросов. В данной парадигме каждый кортеж таблицы T рассматривается как точка в пространстве признаков, соответствующая совместному распределению $P(T(A)) = P(T(A_1, A_2, ..., A_n))$. Для заданного запроса Q с условиями $A_i \in R_i$ вероятность $P(T(Q)) = P(T(A_1 \in R_1, ..., A_n \in R_n))$ позволяет выразить кардинальность как $Card(Q, T) = P(T(Q)) \cdot |T|$. Таким образом, ключевая задача сводится к точной аппроксимации функции плотности $P(T(A))$ для целевой таблицы.

1. Адаптивный оптимизатор запросов AQO [42] предлагает модификацию стандартной формулы расчета селективности, вводя индивидуальные коэффициенты для каждого простого условия. Эти коэффициенты подбираются с помощью машинного обучения (метод k-ближайших соседей) таким образом, чтобы вычисленная селективность максимально соответствовала реальной селективности, наблюдаемой AQO при предыдущих выполнениях запросов. Для этого система сохраняет два типа данных: селективность условий, предсказанную штатным планировщиком, и фактическую селективность, полученную после выполнения запросов. Особенность AQO заключается в анализе условий с точностью до константных значений. Такой подход снижает сложность задачи обучения, при этом в большинстве случаев не приводит к потере информации - хотя AQO не учитывает конкретные значения констант, он работает с наблюдаемой селективностью условий. Исключение составляют случаи, когда планировщик использует фиксированные оценки по умолчанию, например:
   - для условий вида ”выражение1 = выражение2” всегда применяется селективность 0.005.

– для условий ”выражение1 > выражение2”используется значение 1/3.

Среди всех рассмотренных в работе подходов AQO является наиболее простым как в реализации, так и в концептуальном плане. Более того, этот метод уже интегрирован в PostgreSQL и может быть активирован через настройки системы. Однако проведенные тесты для OLAP-систем показали, что при использовании исключительно hash-join (который доминирует в аналитических запросах как наиболее эффективный метод) применение AQO не дает значимого ускорения выполнения запросов.

2. $Naru$ (Neural Relation Understanding) [22] – современный оценщик кардинальности, который полностью фиксирует корреляции между всеми столбцами одной таблицы, используя глубокую авторегрессионную модель. Имея таблицу T, авторегрессионная модель θ принимает кортеж $\mathbf{x} \in T$ в качестве входных данных и предсказывает условные распределения вероятностей, $\{p_\theta(X_i|\mathbf{x}_{<i})\}$, каждое из которых является одномерным распределением по i-ому столбцу (условие на всех предыдущих значениях столбца $\mathbf{x}$. Вероятность входного кортежа затем прогнозируется как $p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(X_i = \mathbf{x}_i|\mathbf{x}_{<i})$. Любая глубокая авторегрессивная архитектура, например, $Transformer$, может создать экземпляр этой структуры.

   $Naru$ является подходом на основе обучения без учителя. Он не требует выполнения запросов для сбора обучающих данных, ему нужны только сами данные конкретной БД. Хоть этот подход и имеет исходный код в открытом доступе, в отличии от многих других, результаты статьи являются скорее теоретическими, нежели практическими. Подход применим только к оценке кардинальности для запросов, использующих всего 1 таблицу, что делается достаточно качественно и традиционными методами. Вклад этой статьи заключается скорее в отработке подхода на основе авторегрессионных моделей к данному роду задач для дальнейшего расширения на случай нескольких таблиц.

3. Метод *NeuroCard* [43] представляет собой многотабличное расширение подхода *Naru*, основанное на глубокой авторегрессионной модели. В его основе лежит декомпозиция совместной функции плотности вероятности (ФПВ) по цепному правилу: $P_T(A) = P_T(A_1) \cdot$

$\prod_{i=2}^{k} P_T(A_i|A_1,...,A_{i-1})$, где каждая условная ФПВ параметризуется 4-слойной нейронной сетью. Для совместного обучения таблиц используется единый автоэнкодер с маскированием. Оценка вероятности запроса Q выполняется методом прогрессивной выборки из области определения запроса.

Важное ограничение оригинального *NeuroCard* - поддержка только древовидных схем соединения. В экспериментах данного исследования с циклическими схемами применялось разбиение на набор древовидных подграфов с построением отдельной модели для каждого. Хотя метод демонстрирует высокую точность, его практическое применение сталкивается с проблемами:

- Экспоненциальный рост числа подграфов с увеличением таблиц.
- Высокие вычислительные затраты на инференс нейросетевой модели.

Эти ограничения делают текущую реализацию *NeuroCard* непригодной для интеграции в промышленные СУБД, несмотря на теоретическую точность подхода.

4. Метод *Pessimistic CardEst* [23] реализует принципиально иную парадигму оценки кардинальности, ориентированную не на точное предсказание, а на построение гарантированной верхней оценки, всегда превышающей фактическое значение. В отличие от традиционных подходов, где минимизируется ошибка предсказания, здесь оптимизируется строгость верхней границы. Теоретическая основа метода базируется на аппарате условной энтропии для вывода оценок и использовании гиперграфов для представления соединений, что позволяет оптимизировать вычислительную сложность. Авторы формализовали пространство компромиссов между точностью оценок и вычислительными затратами, обеспечивая теоретическую основу для адаптивного управления этим балансом.

   Однако практическое применение метода сталкивается с существенными ограничениями: отсутствием поддержки циклических схем соединения и необходимостью линейного объема дополнительной памяти в худшем случае. Эти факторы, наряду с вычислительной сложностью, пока пре-

пятствуют интеграции подхода в промышленные СУБД, несмотря на его теоретическую строгость и математическую элегантность.

5. Метод $FLAT$ [44], основанный на усовершенствованных сетях факторизации-разделения-суммы-произведения (FSPN), предлагает адаптивное разложение совместной плотности вероятности $P_T(A)$ с учетом степени зависимости между атрибутами. В отличие от классических SPN, в FSPN вводится дополнительный узел факторизации, который разделяет $P_T(A)$ на произведение $P_T(W) \cdot P_T(H|W)$, где W и H представляют соответственно слабо и сильно коррелированные атрибуты. При этом $P_T(W)$ моделируется традиционным SPN-подходом, а для $P_T(H|W)$ применяется итеративное разбиение на подпространства до достижения локальной независимости H от W, после чего многомерная ФПВ $P_T(H)$ аппроксимируется непосредственно с помощью многолистового узла. Как и в SPN, структура FSPN и вычисление вероятности запроса выполняются рекурсивно - в нисходящем и восходящем порядке соответственно. Ключевое преимущество подхода заключается в кластеризации данных по степени взаимной зависимости, что позволяет использовать более простые и быстрые модели машинного обучения, так как оставшиеся после факторизации зависимости хорошо описываются кусочно-линейными функциями. Кроме того, метод обеспечивает быстрое обновление структуры при изменении данных.

   $FLAT$ представляет собой наиболее перспективное направление, сочетая более изощренный подход, чем AQO, с уже доказанной практической применимостью (метод внедрен в одну коммерческую СУБД). Однако отсутствие открытой реализации и значительная сложность самостоятельной разработки системы на момент проведения исследования не позволили включить экспериментальную проверку этого подхода в данную работу.

3.2.2 Подходы на основе замены традиционной функции стоимости на нейросетевую

Экспериментальные данные демонстрируют отсутствие прямой зависимости между точностью оценок кардинальности и производительностью запросов. Как показано на рисунке 3.2, даже значительное улучшение точности оценок не

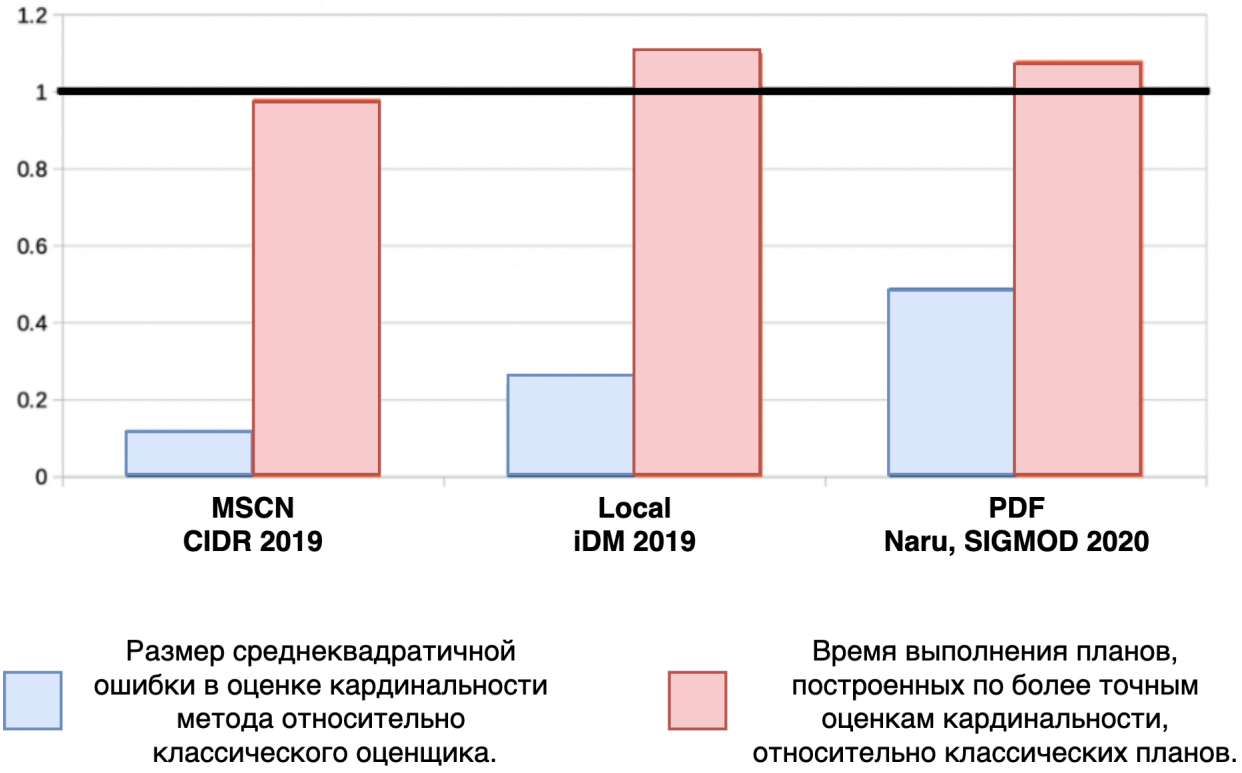


Рисунок 3.2 — Сопоставление точности оценок кардинальности разными методами и времени выполнения запросов.

гарантирует соответствующего ускорения выполнения запросов. В случае с подходом $MSCN$ (multi-set convolutional network) [45] десятикратное повышение точности оценок кардинальности привело лишь к незначительному (в пределах нескольких процентов) росту производительности. Более того, два других исследованных метода, обеспечившие 3-х и 2-х кратное улучшение точности соответственно, фактически продемонстрировали увеличение времени выполнения запросов.

Этот парадокс объясняется несколькими факторами:

– Стоимостная модель оптимизатора, являясь по сути эвристической, вносит существенный вклад в конечный результат.
– Различия в реализации стоимостных функций между разными СУБД.
– Наличие дополнительных факторов, влияющих на производительность (например, выбор физических операторов).

Таким образом, при разработке методов оценки кардинальности следует учитывать, что повышение точности само по себе не является достаточным условием для улучшения производительности запросов. Необходим комплексный под-

ход, учитывающий особенности конкретной стоимостной модели и других компонентов оптимизатора запросов.

Осознание рисков, связанных с подходом, основанным исключительно на повышении точности предсказания кардинальности, побудило исследовательское сообщество, включая автора данного исследования, рассмотреть альтернативные методы оптимизации запросов. Новое направление предполагает непосредственную оптимизацию планов выполнения, минуя этап оценки кардинальности. В этом контексте особую популярность приобрели методы обучения с подкреплением (Reinforcement Learning, RL), что объясняется естественным соответствием между процессом построения плана запроса и структурой RL-задач. Фундаментальное сходство заключается в последовательном характере обоих процессов: при построении плана запроса оптимизатор постепенно формирует оптимальную последовательность соединений, аналогично тому, как RL-агент выстраивает оптимальную стратегию действий. Это соответствие позволяет органично интегрировать RL в традиционные оптимизаторы, заменяя стандартные алгоритмы перечисления на RL-стратегии. Такой подход обладает двумя существенными преимуществами: во-первых, он сохраняет совместимость с классической архитектурой оптимизаторов, а во-вторых, использует иерархическую структуру задачи для значительного сокращения вычислительных затрат на обучение по сравнению с методами обучения с учителем. Для более глубокого понимания этой аналогии рассмотрим классический алгоритм динамического программирования ”снизу вверх”для определения порядка соединений. Согласно принципу оптимальности, алгоритм последовательно строит оптимальные подпланы, начиная с соединений двух таблиц и постепенно увеличивая их размер. Традиционно для хранения промежуточных результатов используется справочная таблица, однако этот подход сталкивается с проблемой экспоненциального роста вычислительной сложности. Методы RL предлагают решение этой проблемы, интерпретируя справочную таблицу как обучаемую модель, которая аккумулирует знания о качестве различных подпланов и позволяет предсказывать эффективность потенциальных решений без необходимости явного хранения всей статистики.

Подход из статьи «Learning to Optimize Join Queries With Deep Reinforcement Learning» [38] (DQN – **D**eep **Q**-**N**etwork) — глубокая нейронная сеть, приближающая Q-функцию, — переосмысливает процесс оптимизации запросов как зада-

чу предсказания. Анализируя стоимость ранее построенных подпланов, система определяет, какое следующее действие с наибольшей вероятностью приведёт к оптимальному решению. В основе подхода лежит Q-learning, который устанавливает зависимость между выбором конкретного соединения таблиц и его ожидаемой эффективностью. Эта зависимость выводится на основе исторических данных о том, как подобные соединения влияли на итоговую производительность всего плана запроса. По сути, алгоритм учится предсказывать ценность каждого возможного соединения, опираясь на накопленный опыт выполнения запросов, что позволяет избегать полного перебора всех вариантов.

1. **Пример.**
   Авторы этой статьи сосредоточились на классической проблеме поиска плана запроса, состоящего из бинарных операторов соединения и унарных выборок, проекций и методов доступа. В качестве примера они использовали следующую базу данных из трех отношений, обозначающих заработную плату сотрудников:

   $$Emp(id,name,rank)\ \ Pos(rank,title,code)\ \ Sal(code,amount)$$

   Рассмотрим следующий запрос:

   ```
   SELECT *
     FROM Emp, Pos, Sal
    WHERE Emp.rank = Pos.rank
     AND Pos.code = Sal.code
   ```

   Существуют различные варианты выполнения данного запроса. В качестве примера можно выполнить запрос как $Emp \bowtie (Sal \bowtie Pos)$, или как $Sal \bowtie (Emp \bowtie Pos)$.
2. **Обучение с подкреплением.**
   Принцип оптимальности Беллмана не только лежит в основе оптимизации реляционных запросов, но и имеет глубокую связь с марковскими процессами принятия решений (МППР) — классом стохастических процессов, применяемых для формализации разнообразных задач, начиная от поиска пути до планирования расписаний. В рамках модели МППР агент осуществляет последовательный выбор действий, направленный

на достижение определённой целевой функции, такой как увеличение производительности или улучшение точности. Принимаемые решения зависят от текущего состояния системы и, как правило, приводят к переходу в новое состояние. Марковский характер процесса означает, что будущая эволюция системы полностью определяется её текущим состоянием. Математически МППР задаётся кортежем из пяти элементов:

$$\langle S, A, P(s,a), R(s,a), s_0 \rangle$$

Здесь S представляет множество возможных состояний системы, A — множество доступных агенту действий, $s' \sim P(s,a)$ определяет вероятностное распределение новых состояний при заданных текущем состоянии и выбранном действии, а s_0 задаёт распределение начальных состояний. Функция $R(s,a)$ определяет величину вознаграждения за выполнение действия a в состоянии s, служащую мерой эффективности агента. Основная задача МППР заключается в поиске оптимальной стратегии $\pi : S \to A$, то есть функции, сопоставляющей состояния действиям и максимизирующей ожидаемое суммарное вознаграждение:

$$_{\pi} \quad \mathbf{E}\left[\sum_{t=0}^{T-1} R(s_t, a_t)\right]$$

при условии, что $s_{t+1} = P(s_t, a_t),\ a_t = \pi(s_t)$

Как и в случае динамического программирования в комбинаторных задачах, большинство МППР сложно решить точно. Стоит обратить внимание, что жадное решение, жадно максимизирующее вознаграждение на каждом шаге, может оказаться неоптимальным в долгосрочной перспективе. Как правило, аналитические решения таких проблем плохо масштабируются во временном горизонте.

3. **Марковская модель перечисления.**
   Рассмотрим стандартное перечисление соединений «снизу вверх», а затем установим связь с марковским процессом принятия решений. Каждый запрос на соединение можно описать как граф запросов, где ребра обозначают условия соединения между таблицами, а вершины обозначают таблицы. Любая реализация оптимизатора соединений динамического программирования должна отслеживать свой прогресс: что уже было сделано в конкретном подплане (какие отношения уже были объединены и какие варианты осталось «присоединить» к рассматриваемому подплану). Формализм графа запросов позволяет представить это состояние. Граф запроса G — это неориентированный граф, в котором каждое отношение R является вершиной, а каждый предикат соединения ρ определяет ребро между вершинами. Пусть κ_G обозначает количество компонент связности G. Принятие решения о соединении двух подпланов соответствует выбору двух вершин, соединённых ребром, и объединению их в одну вершину. Пусть $G = (V, E)$— граф запроса. Применение соединения $c = (v_i, v_j)$ к графу G определяет новый граф со следующими свойствами: (1) v_i и v_j удаляются из V, (2) новая вершина $(v_i + v_j)$ добавляется в V и (3) ребра $(v_i + v_j)$ являются объединением ребер, инцидентных v_i и v_j. Каждое соединение уменьшает количество вершин на 1. Каждый план можно описать как последовательность таких соединений $c_1 \circ c_2 ... \circ c_T$ до тех пор, пока $|V| = \kappa_G$. Приведенное выше описание включает в себя еще одну эвристику System R: «избегание декартовых произведений».

Можно ослабить эту эвристику, просто добавив ребра к G в начале алгоритма, чтобы убедиться, что он полностью связан.

Рассмотрим пример, в котором исходный граф запросов содержит вершины (Emp, Pos, Sal). Предположим, что первое выполняемое соединение имеет вид $c_1 = (Emp, Pos)$. В результате этого соединения структура графа изменяется, и теперь он состоит из вершин $(Emp + Pos, Sal)$. Далее применяется единственное оставшееся возможное соединение, что приводит к образованию финальной вершины $Sal + (Emp + Pos)$. Данная вершина соответствует плану соединения, который можно записать как $Sal \bowtie (Emp \bowtie Pos)$. Таким образом, последовательное применение операций соединения преобразует исходный граф запросов в итоговый план выполнения.

Задача оптимизации соединения состоит в том, чтобы найти наилучшую возможную последовательность соединений, т.е. наилучший план запроса. Эту модель можно расширить, чтобы она также учитывала выбор физического оператора. Множество допустимых соединений можно классифицировать по типам, например, соединение может быть представлено как $c = (v_i, v_j, HashJoin)$ или $c = (v_i, v_j, NestLoop)$. Также имеется стоимостная модель $J(c) \rightarrow R+$, представляющая собой функцию для вычисления стоимости выполнения определённого соединения.

Даны граф запроса G и модель стоимости J. Требуется найти последовательность соединений $c_1 \circ c_2 \ldots \circ c_T$, которая преобразует граф до состояния с $|V| = \kappa_G$, минимизируя общую стоимость:

$$\min_{c_1,\ldots,c_T} \quad \sum_{i=1}^{T} J(c_i)$$

$$\text{при условии, что } \; G_{i+1} = c(G_i)$$

Эта формулировка задачи явным образом задаёт марковский процесс принятия решений (МППР), несмотря на то, что здесь рассматривается минимизация, а не максимизация. В данном случае G описывает состояние системы, c соответствует действию, а процедура объединения вершин определяет переход между состояниями $P(G,c)$. Функция возна-

граждения выражается как отрицательная стоимость $-J$. Итогом МППР является функция, которая для заданного графа запроса определяет оптимальное следующее соединение.

4. **Стоимость соединения с учётом "последствий".**
   Чтобы представить, как обучение с подкреплением даёт нам новый взгляд на эту классическую проблему оптимизации базы данных, сначала рассмотрим жадное решение. Наивным решением является независимая оптимизация каждого c_i. Алгоритм работает следующим образом: (1) начать с графа запроса, (2) найти соединение с наименьшей стоимостью, (3) обновить граф запроса и повторять, пока не останется только одна вершина.
   Жадный алгоритм, конечно, не учитывает, как локальные решения могут повлиять на будущие затраты. Для иллюстрации рассмотрим наш пример запроса со следующими простыми затратами:

$$J(EP) = 100, \quad J(SP) = 90, \quad J((EP)S) = 10, \quad J((SP)E) = 50$$

   Жадное решение будет стоить 140 (поскольку оно не учитывает стоимости будущих соединений), в то время как оптимальное решение имеет стоимость 110.
   Жадное решение неоптимально, потому что на каждом шаге оно не учитывает долгосрочную цену своего действия. Иногда приходится жертвовать краткосрочной выгодой ради экономии в совокупности. Рассмотрим задачу оптимизации для конкретного графа запроса G:

$$V(G) = \min_{c_1,\ldots,c_T} \quad \sum_{i=1}^{T} J(c_i) \tag{3.2}$$

   В классическом подходе к динамическому программированию эта функция называется функцией ценности. Отмечено, что оптимальное поведение на всем горизонте решений предполагает оптимальное поведение в каждый момент времени, что лежит в основе идеи динамического программирования.

В зависимости от текущего соединения можно записать её в следующем виде:

$$V(G) = \min_{c} Q(G,c)$$

$$Q(G,c) = J(c) + V(G')$$

что приводит к следующему рекурсивному определению Q-функции (функции остаточной стоимости):

$$Q(G,c) = J(c) + \min_{c'} Q(G',c') \tag{3.3}$$

Интуитивно Q-функция отражает общую стоимость выполнения соединений при условии, что на каждом последующем шаге выбирается оптимальное действие. Если Q-функция известна, то задача считается решённой, поскольку для построения оптимальной последовательности достаточно на каждом шаге выбирать соединение, минимизирующее значение Q: $\min_{c'} Q(G',c')$. Таким образом, локальная оптимизация Q-функции гарантирует глобально оптимальный план соединений.

Рассмотрим модифицированную версию жадного алгоритма, работающую следующим образом: (1) исходным состоянием является граф запроса, (2) на каждом шаге выбирается соединение с минимальным значением Q-функции, (3) граф обновляется, и процесс повторяется. Такой алгоритм сохраняет вычислительную сложность $O(|V|^3)$, но при этом становится доказуемо оптимальным. Ключевая идея заключается в использовании глубокого обучения с подкреплением для аппроксимации глобальной Q-функции, которая обобщается на все возможные графы запросов в базе данных. Это позволяет получить полиномиальный алгоритм построения оптимального плана выполнения запроса, что является существенным улучшением по сравнению с традиционными методами.

5. **Применение обучения с подкреплением к задаче данного исследования.**

Алгоритмы Q-обучения представляют собой важный класс методов обучения с подкреплением, которые позволяют приближать Q-функцию на основе эмпирических данных. Возникает вопрос: что, если бы мы могли обучать модель предсказывать совокупную будущую стоимость, используя признаки (G, c) и ограниченный набор наблюдений? На практи-

ке данные для обучения могут быть получены из записанных последовательностей принятия решений, каждая из которых представляет собой кортеж $(G, c, J(c), G')$, где G — исходный граф запроса, c — выполненное соединение, $J(c)$ — его фактическая стоимость, а G' — преобразованный граф после выполнения соединения. Такие последовательности легко извлекаются из готовых планов выполнения запросов, что делает возможным обучение модели на реальных данных без необходимости явного моделирования среды.

Предположим, что у нас имеется параметризованная модель Q_θ, аппроксимирующая истинную Q-функцию:

$$Q_\theta(f_G, f_c) \approx Q(G, c)$$

Здесь f_G — это вектор признаков, кодирующий структуру графа запроса, а f_c — вектор признаков, описывающий конкретное соединение. Параметры модели θ инициализируются случайным образом и затем уточняются в процессе обучения. Для каждого обучающего примера i можно вычислить целевую метку y_i, которая представляет собой оценку Q-значения:

$$y_i = J(c) + \min_{c'} Q_\theta(G',c')$$

Эти метки $\{y_i\}$ затем используются для обучения модели методом регрессии, где цель состоит в минимизации расхождения между предсказанными $Q_\theta(f_G, f_c)$ и целевыми значениями y_i. Такой подход позволяет итеративно улучшать оценку Q-функции на основе наблюдаемых данных. Если бы Q была истинной Q-функцией, то имела бы место следующая рекуррентная формула:

$$Q(G,c) = J(c) + \min_{c'} Q_\theta(G', c')$$

Итак, процесс обучения, или Q-обучение, определяет функцию потерь на каждой итерации:

$$L(Q) = \sum_i \|y_i - Q_\theta(G,c)\|_2^2$$

Затем параметры Q-функции можно оптимизировать с помощью градиентного спуска до сходимости.

Обучение с подкреплением дает два ключевых преимущества: (1) стоимость поиска для одного запроса по сравнению с традиционной оптимизацией запросов радикально снижается, поскольку алгоритм имеет временную сложность жадного поиска, и (2) параметризованная модель может потенциально обучаться через запросы, которые имеют похожие, но не идентичные подпланы. Это связано с тем, что сходство между подпланами определяется признаковыми описаниями графа запроса и соединений, f_G и f_c; таким образом, если они разработаны достаточно выразительным образом, то нейронную сеть можно обучить экстраполировать оценки Q-функции на всю рабочую нагрузку.

Выбор Q-обучения вместо других алгоритмов обучения с подкреплением сделан не просто так. Во-первых, Q-обучение позволяет использовать оптимальные подструктуры во время обучения и значительно сократить объем необходимых данных. Во-вторых, по сравнению с обучением на основе стратегий, Q-обучение выводит оценку для каждого соединения, которое появляется в любом подплане, а не просто выбирает лучшее соединение. В-третьих, модель позволяет выбирать k лучших соединений на каждом шаге, а не просто получать лучший план.

6. **Эффективная генерация обучающей выборки.**
   Обучающая выборка генерируются автоматически в результате работы традиционного алгоритма планирования. Для каждого решения о соединении, которое принимает оптимизатор, можно получить оценку стоимости соединения, рассчитанную на основе оценок кардинальности. Предположим, мы запускаем традиционный алгоритм кустистого динамического программирования для оптимизации k-путевого соединения. Мы получаем не только окончательный план, но и оптимальный план для каждого отдельного подплана, перечисляемого на этом пути. Каждый запрос генерирует оптимальный план запроса для всех входящих в него подпланов, а также запоминает неоптимальные подпланы, которые не были достроены до конца. Это означает, что один запрос генерирует большое количество обучающих примеров. На рис. 3.3 показано, как принцип оптимальности помогает собрать обучающую выборку.

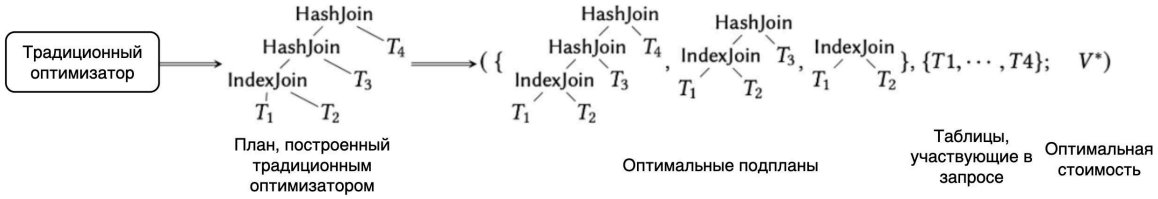


Рисунок 3.3 — Используя принцип оптимальности, из одного плана, созданного собственным оптимизатором, извлекаются три обучающих примера. Эти примеры имеют одни и те же долгосрочные затраты и отношения для соединения (т.е. принятие этих локальных решений в конечном итоге приводит к соединению в одну связную компоненту $\{T1, ...,T4\}$ с оптимальной совокупной стоимостью V^*).

7. **Инженерия ”признаков”.**

   Краткая мотивация того, как стоит думать о признаковом описании данных в такой задаче, заключается в следующем. Признаковые описания должны быть достаточно богатыми, чтобы они отражали всю необходимую информацию для прогнозирования стоимости принятого решения в купе с последующими затратами. Для этого необходимо знать, что требует SQL-запрос в целом, таблицы в левой части предлагаемого соединения и таблицы в правой части предлагаемого соединения. Также требуется знание того, как предикаты одной таблицы влияют на кардинальность по обе стороны соединения.

   – **”Участвующие” отношения.**

     Общая интуиция состоит в том, чтобы использовать имя каждого столбца в качестве признака, потому что оно определяет распределение этого столбца. Первым шагом является создание набора признаков, чтобы описать какие атрибуты участвуют в запросе и в конкретном соединении. Пусть A будет набором всех атрибутов в базе данных (например, $\{$Emp.id, Pos.rank, ..., Sal.code, Sal.amount$\}$). Каждое отношение rel (включая промежуточные результаты построения дерева соединений) имеет набор атрибутов, $A_{rel} \subseteq A$. Аналогично, каждый граф запросов G может быть представлен своими атрибутами $A_G \subseteq A$. Соединение можно описать как пару отношений (L, R), где для каждого из них существуют видимые атрибуты

A_L и A_R. Все наборы атрибутов – A_G, A_L и A_R – кодируются бинарным способом: единица означает наличие соответствующего атрибута, а ноль – его отсутствие. Применяя операцию $\oplus$ для объединения векторов, формируются признаки графа запроса $f_G = A_G$ и признаки соединения $f_c = A_L \oplus A_R$. В итоге, полный вектор признаков для пары (G, c) представляет собой конкатенацию $f_G \oplus f_c$. Наглядный пример такого представления можно увидеть на рисунке 3.4, где показано признаковое описание для демонстрационного запроса.

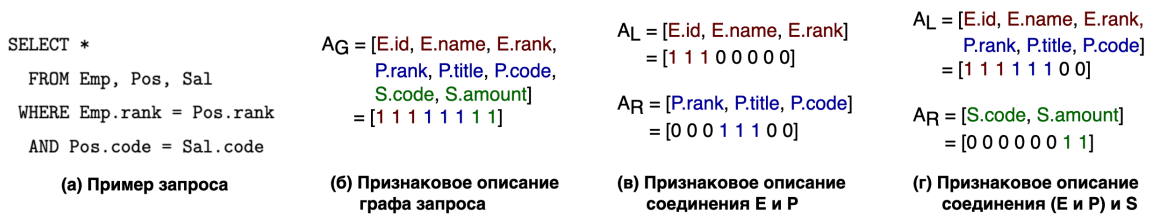


Рисунок 3.4 — Запрос и соответствующие ему признаковое описание. Бинарные векторы кодируют атрибуты в графе запроса (A_G), левой части соединения (A_L) и правой части (A_R). Такое кодирование позволяет описать как граф запроса, так и конкретное соединение. Показаны промежуточное соединение и финальное соединение. Пример запроса охватывает все отношения в схеме, поэтому $A_G = A$.

– **Предикаты.**
Предикаты могут изменить указанное распределение, то есть (col, selectivity) отличается от (col, TRUE). Чтобы обрабатывать предикаты в запросе, нужно учитывать их в признаковом описании. Можно использовать статистику таблиц, присутствующую в большинстве СУБД. Для каждого предиката σ в запросе можно получить селективность δ_σ, которая оценивает долю кортежей, остающихся после применения предиката. Чтобы учесть предикаты в признаковом описании, можно масштабировать слот в f_G, которому соответствует отношение и атрибут с предикатом, на δ_r. Например, если оценка селективности для фильтра Emp.id > 200 оценивается как 0.2, то слот Emp.id в f_G будет изменен на 0.2. Рисунок 3.5а наглядно иллюстрирует это масштабирование.

– **Операторы соединения.**
  Также важно закодировать тип оператора соединения. Это просто: мы добавляем еще один бинарный вектор, который указывает из фиксированного набора операторов используемый тип соединения (рисунок 3.5б).

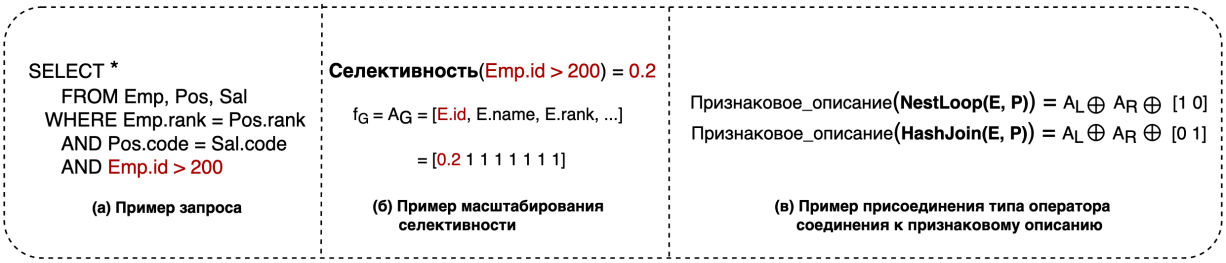


Рисунок 3.5 — Адаптация признакового описания для работы с предикатами и операторами соединения. Базовая структура признаков расширяется: слева добавляются предикаты, справа — физические операторы. В случае выбора между NestLoop и HashJoin, к признакам соединения присоединяется двумерный бинарный вектор, указывающий тип оператора.

8. **Обучение модели.**
   В методе DQN применяется многослойный перцептрон (MLP) для приближенного вычисления Q-функции. На вход нейросети подается вектор признаков (G, c), сформированный как $f_G \oplus f_c$. Экспериментальным путем установлено, что оптимальная производительность при ограниченном времени обучения (менее 10 минут) достигается при использовании двухслойной архитектуры. Обучение модели осуществляется посредством классического стохастического градиентного спуска (SGD).
9. **Дообучение модели на времени выполнения запроса.** Так как модель обучалась на оценках стоимости, которые, как было описано выше, далеки от истинных стоимостей, она не сможет превзойти классический оптимизатор. До текущего момента всё, что делалось моделью – это так называемый imitation learning (обучение повторению). На данном этапе система теоретически может показывать качество сравнимое с качеством традиционного оптимизатора, но никак не может превосходить его. Чтобы продвинуться дальше в сторону улучшения качества, нужно дать модели возможность узнать не оценки, а фактические стоимости тех или иных действий. Стандартным решением этого вопроса является дообу-

чение на данных, где целевым значением вместо оценок стоимости является время выполнения. Такое решение на первый взгляд логично и естественно: наша основная цель – уменьшить время выполнения запросов. Но на деле время, затраченное как на весь запрос, так и на отдельные его части, есть очень непостоянный показатель, зависящий от общей загруженности системы. Причём эти показатели нельзя даже назвать зашумлёнными, что было бы приемлемо для обучения модели. Колебания времени выполнения одного и того же запроса могут быть многократными. Поскольку на практике сбор данных и обучение модели не должно останавливать работу СУБД, эти вещи происходят одновременно со штатным функционированием СУБД. Таким образом, использование времени выполнения запросов в качестве целевых значений для обучения модели невозможно. Предложенное в данной работе решение описано в разделе ”Архитектура рабочего процесса” секции 3.3.2.

10. **Использование обученной модели.**
    После обучения становится доступна параметризованная оценка Q-функции $Q_\theta(f_G, f_c)$. Для её применения нужно просто вернуться к стандартному алгоритму, как в жадном методе, но вместо использования локальных стоимостей соединений теперь используется выученная Q-функция: (1) начинаем с графа запроса, (2) кодируем каждое соединение (возможное действие), (3) выбираем соединение с наименьшим оценочным значением Q (т.е. выходом нейронной сети), (4) обновляем подплан запроса и повторяем. Этот алгоритм имеет временную сложность жадного перечисления. Отличие лишь в том, что стоимостная модель жадного алгоритма здесь заменяется на нейронную сеть.

Методология работы системы «Neo: A Learned Query Optimizer» (**Ne**ural **O**ptimizer) [41] представляет собой обученный оптимизатор запросов, демонстрирующий сравнимую или превосходящую эффективность относительно ведущих коммерческих решений (Oracle и Microsoft) при работе на их собственных движках обработки запросов. Данная система обучается принимать решения как о последовательности соединений, так и о выборе оптимальных операторов. Процесс оптимизации реализуется через обучение с учителем, дополненное механизмом обратной связи для непрерывного совершенствования, что позволяет адаптиро-

ваться к конкретной инстанции базы данных и учитывать реальное время выполнения при формировании решений.

Архитектура Neo устраняет четкое разделение между ключевыми компонентами классического оптимизатора запросов: оценкой селективности, расчетом стоимости и алгоритмом перебора планов. В отличие от традиционных подходов, система не вычисляет кардинальность напрямую и не использует заданную вручную стоимостную модель. Вместо этого Neo интегрирует эти функции в единую нейросетевую модель оценки стоимости, которая анализирует подплан и предсказывает оптимальное время выполнения для любого плана запроса, включающего данный подплан. На основе предсказаний этой нейросети Neo осуществляет направленный поиск в пространстве возможных планов выполнения для конкретного запроса. По мере обнаружения более эффективных планов, нейросеть постепенно совершенствуется, что позволяет фокусировать поиск на наиболее перспективных вариантах. Этот итеративный процесс создает положительную обратную связь: улучшенные предсказания стоимости ведут к нахождению более оптимальных планов, что в свою очередь способствует дальнейшему обучению и уточнению нейросетевой модели. Цикл продолжается до стабилизации стратегии принятия решений оптимизатора.

Авторы статьи Neo ставили перед собой задачу по преодолению нескольких ключевых проблем. Во-первых, для автоматического улавливания интуитивно понятных паттернов в планах запросов с древовидной структурой авторы статьи разработали модель глубокой нейронной сети, использующую древовидные свёртки. Во-вторых, чтобы обеспечить понимание нейросетью семантики имеющейся базы данных, они разработали векторное представление характеристики, которая автоматически представляет семантику предикатов запроса с использованием данных из имеющейся базы данных. Они интегрировали эти подходы в комплексную систему обучения, способную строить планы выполнения запросов.

1. **Обзор структуры обучения.**

   Далее мы обсудим архитектуру системы Neo, изображённую на рисунке 3.6, и общую стратегию обучения. Neo работает в два этапа: начальный этап, на котором собирается информация о работе классического оптимизатора, и этап выполнения, на котором в работу вступает нейросетевая модель.

   – **Сбор знаний.**

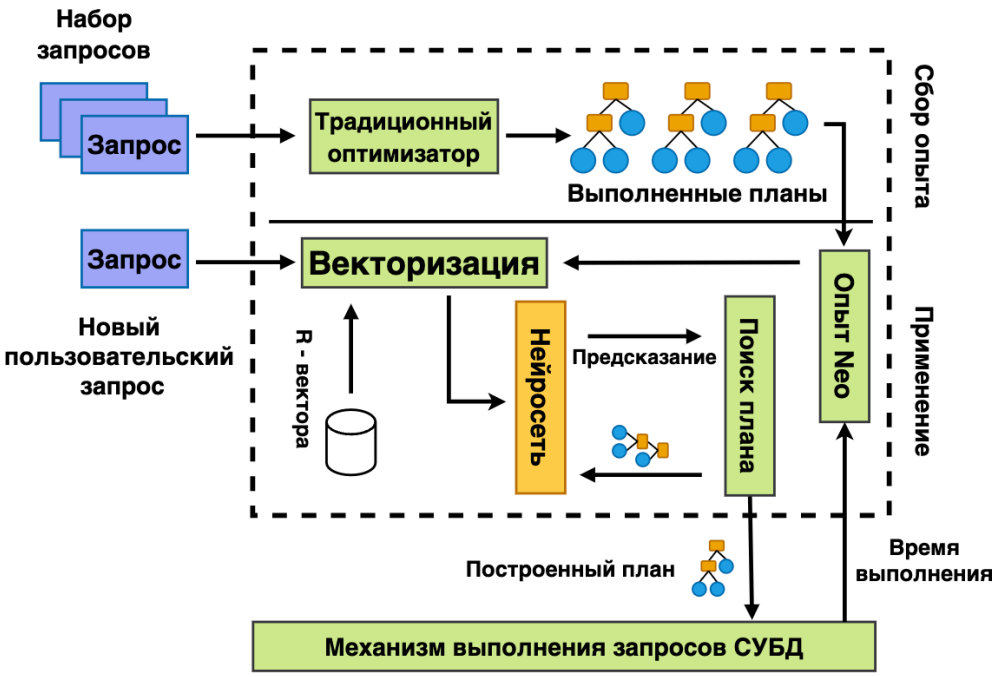


Рисунок 3.6 — Архитектура системы Neo.

На первом этапе, именуемом «Экспертиза», Neo получает опыт от традиционного оптимизатора запросов. Neo нуждается в наборе пользовательских запросов, представляющих общую рабочую нагрузку пользователя и возможности базового движка (т.е. реализующие репрезентативный набор операторов). Кроме того, предполагается, что у Neo есть доступ к традиционному оптимизатору. Neo использует этот оптимизатор только для создания планов выполнения запросов (QEP) для каждого запроса в наборе. Эти QEP вместе с их временем выполнения добавляются в базу знаний Neo (набор пар план/время), которые используются в качестве отправной точки на этапе обучения модели.

– **Построение модели.**
На основе накопленного опыта Neo строит первоначальную модель. Модель представляет собой глубокую нейронную сеть, предназначенную для прогнозирования конечного времени выполнения запроса по имеющемуся подплану или полному плану (частный случай). Нейросеть обучается, используя собранные заранее данные, в режиме обучения с учителем. Этот процесс включает преобразование каждого собранного запроса в признаковое описание. Эти признаки содержат информацию о запросе

(например, граф соединения) и информацию о текущем состоянии плана (например, порядок соединения). Neo может работать с рядом различных признаков, начиная от простых бинарных кодировок и заканчивая более сложными эмбедингами. Сеть стоимости Neo использует древовидные свертки для обработки QEP с древовидной структурой.

– **Поиск плана.**

После преобразования запроса в закодированное представление система Neo применяет нейросетевую модель для исследования пространства возможных планов выполнения запросов (QEP), включающего различные варианты порядка соединений и типов операторов. Модель находит оптимальный план, минимизирующий ожидаемое время выполнения, которое оценивается с помощью предсказания нейросети. Учитывая, что полный перебор всех возможных планов для заданного запроса вычислительно неосуществим, Neo реализует стратегию поиска по принципу «сначала наилучший вариант», используя нейросеть для эффективного исследования пространства решений. Сформированный Neo план содержит детализированную информацию о последовательности соединений, применяемых операторах (таких как хеш-соединение, соединение слиянием или вложенными циклами) и методах доступа к данным (например, сканирование индексов или полное сканирование таблиц). Этот план передаётся в нижележащий механизм выполнения, который обрабатывает его, добавляя при необходимости семантически корректные преобразования (вроде включения операций сортировки) и исполняя итоговый план. Такой подход гарантирует как эффективность, так и корректность генерируемых планов выполнения запросов.

– **Переобучение модели.**

По мере того, как Neo просматривает больше запросов, модель улучшается и адаптируется к базе данных пользователя. Это достигается за счет использования нового опыта. В частности, когда QEP выбран для определённого запроса, он отправляется в

базовый механизм выполнения, который обрабатывает запрос и возвращает результат пользователю. Кроме того, Neo записывает время выполнения QEP, добавляя пару план/время к своему опыту. Затем Neo переобучает модель на основе этого опыта, итеративно улучшая свои оценки.

2. **Инженерия признаковых описаний.**
   Признаковые описания объектов – это как раз то, с чем работает нейросеть. Именно их она принимает на вход и из них пытается извлечь знания. Поэтому от качества признаковых описаний зависит качество модели.
   - **Обозначения.**
     Для запроса q мы определяем набор базовых отношений, используемых в q, как $R(q)$. Подплан выполнения P для запроса q (обозначается $Q(P) = q$) это лес деревьев, представляющий план выполнения, который находится в процессе построения (или построен – частный случай). Каждый внутренний (нелистовой) узел дерева является оператором соединения $\bowtie_i \in J$, где J — набор возможных операторов соединения (например, с помощью хеширования $\bowtie_H$, слиянием $\bowtie_M$, циклом $\bowtie_L$), и каждый листовой узел является чтением таблицы, чтением индекса или невыбранным способом чтения отношения $r \in R(q)$, обозначаемыми $T(r)$, $I(r)$ и $U(r)$ соответственно. Например, частичный план выполнения запроса можно обозначить как

$$[(T(D) \bowtie_H T(A)) \bowtie_L I(C)], \quad [U(B)] \tag{3.4}$$

     Здесь тип чтения для B не указан, и не выбрано соединение для связи B с остальной частью плана. В плане также указан способ чтения таблиц D и A, которые соединяются хешом, результат которого затем будет объединён с использованием циклического соединения с C.
     Полный план выполнения — это план из одного дерева и без листьев с невыбранным способом чтения; все решения о том, как план должен быть выполнен, были приняты. Мы говорим, что один план P_i является подпланом другого плана P_j (обозначим $P_i \subset P_j$), если P_j может быть построен из P_i путем (1) выбора

способа чтения для листа с до сих пор неопределённым способом чтения или (2) объединением поддеревьев в P_i оператором соединения.

Построение полного плана выполнения можно рассматривать как марковский процесс принятия решений (MDP). Начальное состояние MDP — частичный план, в котором каждое сканирование не определено и соединения отсутствуют. Каждое действие включает либо (1) объединение двух подпланов с помощью оператора соединения, либо (2) преобразование неуказанного сканирования в определённый тип сканирования. Более формально каждое действие преобразует текущий план P_i в любой план P_j такой, что $P_i \subset P_j$. Цена за каждое действие равна нулю, за исключением последнего действия, которое имеет цену, равную времени выполнения построенного плана.

– **Кодировка информации о запросе.**

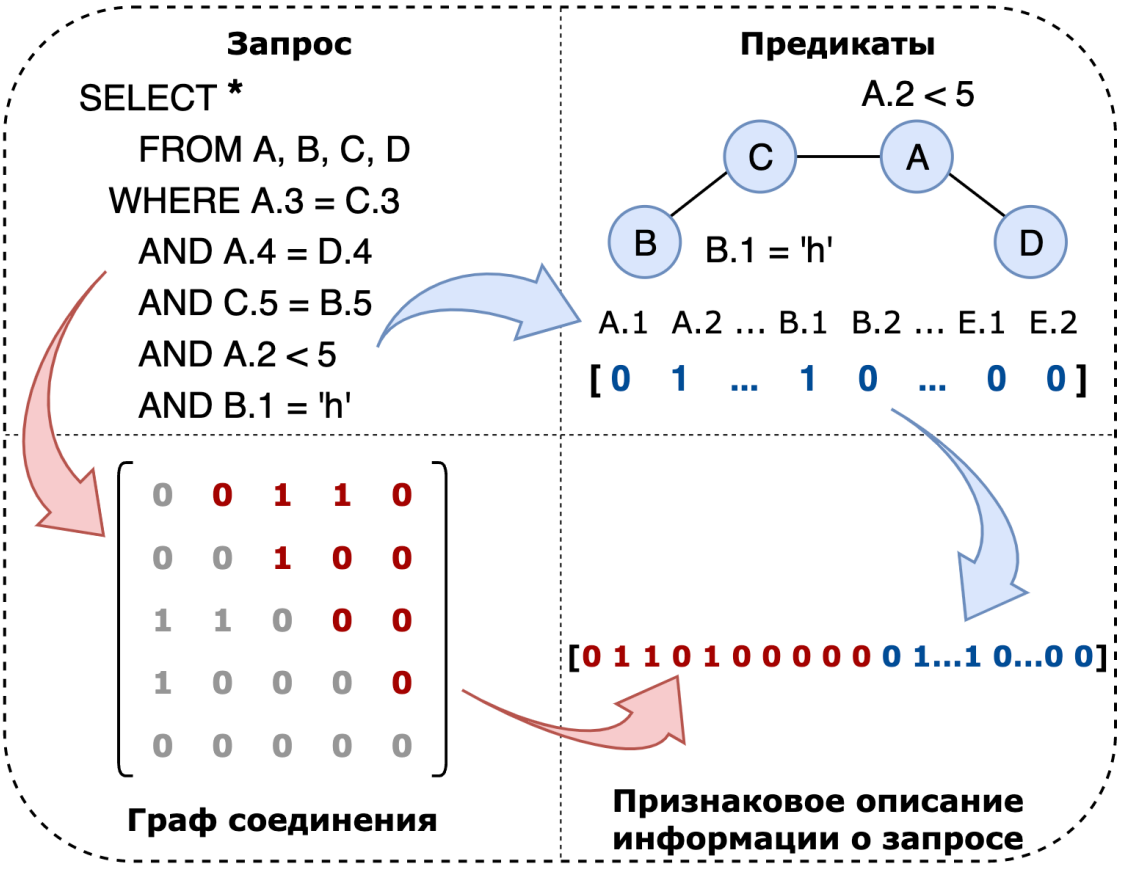


Рисунок 3.7 — Кодировка информации о запросе.

Представление зависимой от запроса, но независимой от плана, информации состоит из двух компонентов. Первый компонент кодирует граф соединения запроса как матрицу смежности, например, на рисунке 3.7 в графе соединения единица в первой строке третьего столбца указывает на наличие предиката соединения между отношениями A и C. Строка и столбец, связанные с отношением E, остаются незаполненными, так как E не фигурирует в данном запросе. Для упрощения модели предполагается, что любые два отношения могут быть связаны максимум одним внешним ключом. Тем не менее, предложенное представление допускает обобщение на случай нескольких внешних ключей путём замены значения «1» на индекс конкретного ключа. Более того, учитывая симметричность матрицы соединений, для формирования векторного представления запроса достаточно учитывать только элементы верхней треугольной части (выделенной красным), что сокращает объём хранимых данных без потери информации.

Второй компонент кодировки запроса — это вектор предикатов столбцов. Neo поддерживает три варианта с различными уровнями требований к предварительным вычислениям:

(a) Бинарная кодировка (наличие предиката): показывает, какие атрибуты участвуют в запросе. Длина вектора — это количество атрибутов во всех таблицах базы данных. Например, на рисунке 3.7 в блоке ”Предикаты” показан закодированный таким образом вектор с 1, установленными в позициях для атрибутов $A.2$ и $B.1$, поскольку оба атрибута используются как часть предиката. Предикаты соединения здесь не рассматриваются. Нейросеть знает только, присутствует ли атрибут в предикате или нет. Такая кодировка может быть построена без какого-либо доступа к базе данных.

(b) Гистограммное кодирование (селективность предиката): расширение бинарной кодировки, которое заме-

няет «0» или «1» предсказанной селективностью этого предиката (например, $A.2$ может быть 0.2, если мы предсказываем селективность 20%). Селективности предоставляются традиционным оптимизатором.

(c) R-Vector (семантика предиката): наиболее продвинутое кодирование с использованием языковой модели машинного обучения. На основе word2vec, модели обработки естественного языка, каждая запись в векторе предиката столбца заменяется вектором, содержащим семантическую информацию, связанную с предикатом. Это кодирование требует построения модели на основе данных в базе данных и является самым дорогим вариантом.

Более сложные схемы кодирования расширяют возможности модели, позволяя ей выявлять и анализировать нетривиальные зависимости в данных. Впрочем, даже упрощённые методы кодирования не становятся непреодолимым препятствием для изучения сложных взаимосвязей. Так, хотя гистограммное кодирование явно не отражает корреляции между таблицами, нейросетевая модель способна самостоятельно выявлять такие закономерности в процессе обучения, адаптируя оценки кардинальности на основе наблюдений за временем выполнения запросов. Однако использование R-Vector в Neo даёт существенное преимущество, предлагая семантически обогащённое представление предикатов запроса, что упрощает процесс оптимизации и повышает точность прогнозирования.

– **Кодировка информации о плане.**

В дополнение к кодировке запроса также требуется представление частичного или полного плана выполнения запроса. В то время как в остальных работах древовидная структура каждого частичного плана выполнения была преобразована в вектор, кодирование, предложенное авторами этой статьи, сохраняет древовидную структуру планов выполнения. Каждый узел частичного плана выполнения кодируется в вектор фиксированной длины,

формируя векторное дерево (рис. 3.8). Количество векторов соответствует числу узлов и может варьироваться, как и структура дерева (левосторонняя или ветвистая), сохраняя при этом единый формат векторного представления. Векторное представле-

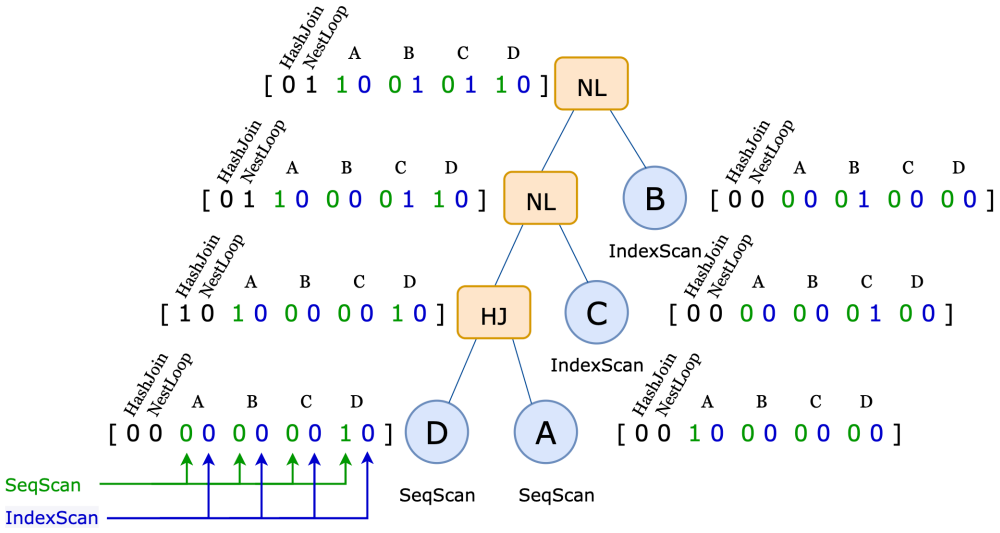


Рисунок 3.8 — Формат описания плана выполнения.

ние формируется путём отображения каждого узла в вектор размерности $|J| + 2|R|$, где $|J|$ — число возможных типов соединений, а $|R|$ — количество отношений. Первые $|J|$ компонент вектора определяют тип соединения (например, корневой узел на рисунке 3.8 кодирует циклическое соединение). Следующие $2|R|$ элементов описывают задействованные отношения и методы их сканирования (полное, индексное или неопределённое). Для листовых узлов в подвекторе отношений устанавливается ровно одна единица, за исключением случая неопределённого сканирования, когда отмечаются оба варианта (индексное и полное). Во внутренних узлах соответствующие значения получаются объединением характеристик дочерних узлов. Например, нижний узел циклического соединения на рисунке 3.8 содержит единицы для полного сканирования таблиц A и D и индексного сканирования таблицы C.

Важно отметить, что данное представление может включать несколько независимых частичных планов выполнения (с несколькими корневыми узлами), которые ещё не соединены

между собой. Например, для кодирования частичного плана из уравнения 3.4, корневой узел $U(B)$ будет представлен вектором $[0000110000]$. Эти кодировки предназначены исключительно для формирования векторного представления планов выполнения, которое будет использоваться нейросетевой моделью Neo, описанной в последующих разделах.

3. **Функция потерь при обучении нейросети.**
   Исследуем архитектуру нейронной сети, обученной предсказывать оптимальное время выполнения для частичного плана P_i - минимальное время, достижимое при любом полном плане P_f, содержащем P_i. Так как истинный оптимальный план выполнения заранее неизвестен, в рамках данного исследования в качестве целевого значения используется наилучшее зафиксированное время выполнения запроса на момент обучения.
   Обозначим через E набор полных планов выполнения запросов $P_f \in E$ с известным временем выполнения $L(P_f)$. Модель M обучается приближать минимальную стоимость для всех подпланов $P_i \subseteq P_f$ из обучающей выборки:

$$M(P_i) \approx \min\{C(P_f) \mid P_i \subseteq P_f,\ P_f \in E\} \tag{3.5}$$

где $C(P_f)$ представляет стоимость полного плана выполнения. Система позволяет настраивать целевую функцию стоимости для адаптации поведения Neo под конкретные требования. Система поддерживает различные стратегии оптимизации через настраиваемую функцию стоимости. Для минимизации абсолютного времени выполнения используется простая метрика: $C(P_f) = L(P_f)$. Альтернативно, для гарантии улучшения относительно базового плана можно определить относительную стоимость:

$$C(P_f) = \frac{L(P_f)}{Base(P_f)}, \tag{3.6}$$

где $Base(P_f)$ - наилучшее известное время выполнения для данного плана. Независимо от выбора метрики, Neo последовательно минимизирует

стоимость в процессе работы. Обучение модели основано на минимизации квадратичной ошибки:

$$(M(P_i) - \min\{C(P_f) \mid P_i \subseteq P_f, P_f \in E\})^2 \tag{3.7}$$

4. **Архитектура нейросети.**
   Архитектура нейросети Neo показана на рисунке 3.9. Она была разработана для создания индуктивного смещения, подходящего для оптимизации запросов: структура самой нейронной сети разработана так, чтобы отражать интуитивное понимание того, что делает планы запросов быстрыми или медленными. Люди, изучающие планы запросов, учатся распознавать неоптимальные или хорошие планы путем сопоставления с образцом: соединение слиянием поверх хеш-соединения с общим ключом соединения, вероятно, вызывает избыточную сортировку или хеширование; циклическое соединение поверх двух хеш-соединений, вероятно, очень чувствительно к ошибкам оценки количества элементов; хеш-соединение с использованием таблицы фактов в качестве отношения «построения», вероятно, приводит к выбросам; серия соединений слиянием, не требующих повторной сортировки, скорее всего, будет работать хорошо и т.д. Ключевой тезис состоит в том, что все эти шаблоны можно распознать, анализируя поддеревья плана выполнения запроса. Архитектура модели Neo, по сути, представляет собой большой банк этих шаблонов, которые изучаются автоматически, из самих данных, с использованием метода, называемого древовидной сверткой. Как пока-

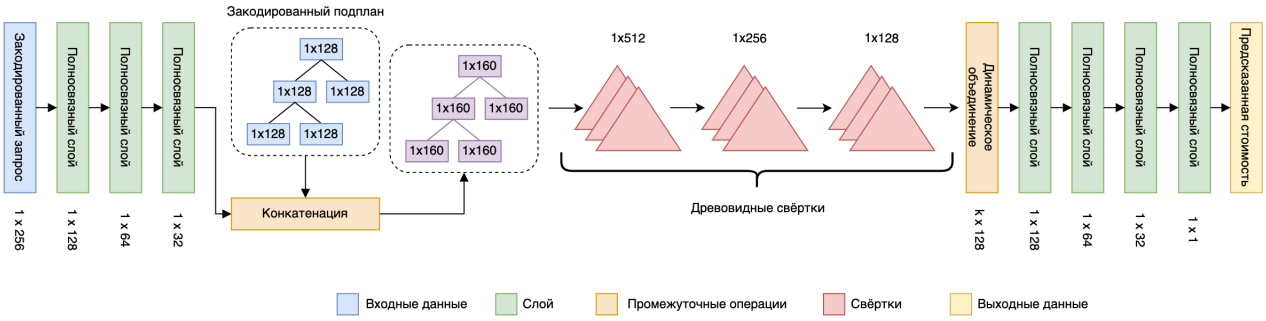


Рисунок 3.9 — Архитектура нейросети.

зано на рисунке 3.9, когда модель оценивает частичный план запроса, кодирование на уровне запроса проходит через несколько полносвязных слоёв, размер каждого из которых уменьшается. Выходной вектор тре-

тьего полносвязного слоя комбинируется с векторным представлением каждого узла дерева плана (одинаковый вектор контекста добавляется ко всем узлам). Данный подход, известный как ”пространственная репликация позволяет объединять признаки фиксированной размерности (кодировка запроса) с признаками переменной размерности (кодировка плана). Обогащенные векторы узлов затем обрабатываются через последовательность слоев древовидной свертки, сохраняющих древовидную структуру данных на выходе. После этого применяется операция динамического объединения, сглаживающая древовидную структуру в один вектор. Несколько дополнительных полносвязных слоёв используются для преобразования этого вектора в одно значение, используемое в качестве прогноза модели для введённого плана.

5. **Древовидные свёртки.**

   Свёрточные нейронные сети принимают входные тензоры с фиксированной структурой, такой как вектор или изображение. Для Neo признаковые описания планов выполнения структурированы как узлы в дереве (например, на рисунке 3.8). Таким образом, древовидные свёртки – это адаптация традиционной свёртки изображений для данных с древовидной структурой.

   Древовидные свёртки является естественным подходом для Neo. Подобно преобразованию свёртки для изображений, свёртка дерева перемещает набор общих фильтров по каждой части дерева плана. Интуитивно понятно, что эти фильтры могут охватывать широкий спектр локальных отношений между родительскими вершинами и их потомками. Например, фильтры могут искать хеш-соединения поверх соединений слиянием или объединение двух отношений при наличии определённого предиката. Выход этих фильтров обеспечивает сигналы, используемые конечными слоями нейросети; выходные данные фильтра могут обозначать соответствующие факторы, такие как сортировка дочерних элементов оператора соединения (предполагая объединение слиянием), или фильтр может оценить, будет ли правостороннее отношение соединения иметь низкую кардинальность (предполагая, что индекс может быть полезен).

   Каждый узел дерева-плана имеет ровно два дочерних узла, поэтому фильтр содержит три весовых вектора: e_p, e_l, e_r. Фильтр применяется к

каждому локальному участку из вектора узла x_p и его дочерних векторов x_l и x_r (которые заменяются на $\vec{0}$ для листьев), создавая новый вектор узла x'_p:

$$x' = \sigma(e_p \odot x_p + e_l \odot x_l + e_r \odot x_r) \tag{3.8}$$

Здесь $\sigma(\cdot)$ — нелинейное преобразование (например, ReLU), $\odot$ — скалярное произведение, а x'_p — выход фильтра. Каждый фильтр агрегирует данные из локального окружения узла дерева. Единый фильтр последовательно обрабатывает все узлы дерева плана выполнения, что обеспечивает его применимость к планам любой величины. Применение набора фильтров к дереву трансформирует его в новое дерево с идентичной структурой, но с узлами, представленными векторами потенциально иной размерности. На практике используется множество (сотни) таких фильтров.

Поскольку результатом свёртки дерева является другое дерево, несколько слоев фильтров свёртки дерева могут быть «наложены друг на друга». Первый слой фильтров свёртки дерева будет иметь доступ к расширенному дереву плана выполнения (т.е. каждый фильтр будет перемещаться по каждому родительскому/левому дочернему/правому дочернему треугольнику расширенного дерева). Количество информации, воспринимаемое конкретным фильтром, называется рецептивным полем фильтра. Второй слой фильтров будет применяться к выходным данным первого, и, таким образом, каждый фильтр этого второго слоя будет видеть информацию, полученную от узла n в исходном расширенном дереве, дочерних элементов n и дочерних для дочерних n: таким образом, каждый слой свёртки дерева имеет большее рецептивное поле, чем у предыдущего. В результате первый слой свёртки дерева изучает простые признаки (например, распознавание объединения слиянием поверх соединения слиянием), тогда как последний слой свертки дерева изучает сложные признаки (например, распознавание левосторонней цепочки соединений слиянием).

6. **Поиск плана с помощью нейросети.**

   Нейросеть предсказывает стоимость плана выполнения запроса, но не даёт сам план выполнения напрямую. Поэтому для создания планов необходимо объединить нейросеть с методом поиска.

Используя обученную модель и входящий запрос q, Neo выполняет поиск в пространстве планов для данного запроса. В некотором смысле этот поиск отражает процесс поиска, используемый традиционными оптимизаторами баз данных, при этом обученная нейросеть берет на себя роль функции стоимости. Однако, в отличие от традиционных систем, нейросеть предсказывает не стоимость подплана, а минимальное возможное время выполнения полного (финального) плана, включающего данный подплан. Эта разница позволяет выполнить поиск по принципу «сначала наилучшее», чтобы найти план выполнения с низкой ожидаемой стоимостью. По сути, это сводится к многократному изучению кандидата с наилучшей предполагаемой стоимостью до тех пор, пока не возникнет условие остановки.

Поиск для запроса q стартует с создания пустой min-кучи, упорядоченной по нейросетевым оценкам стоимости частичных планов. Изначально в кучу помещается частичный план с неопределёнными типами сканирования всех отношений из $R(q)$. Например, при $R(q) = \{A, B, C, D\}$ инициализация задаётся как

$$P_0 = \Big([U(A)], [U(B)], [U(C)], [U(D)]\Big) \tag{3.9}$$

На каждой итерации из вершины кучи извлекается подплан P_i, после чего генерируются его потомки $Children(P_i)$: каждый оценивается нейросетью и добавляется в кучу. Потомки соответствуют всем планам, которые могут быть сформированы путём уточнения сканирования в P_i или объединения двух его поддеревьев оператором соединения. Формально $Children(P_i)$ пусто для завершённого плана, а иначе представляет доступные действия в данном состоянии MDP. Следующая итерация исследует новый наиболее перспективный план из кучи, причём каждая операция поиска выполняется за $O(\log n)$, где n — текущий размер кучи.

Хотя выполнение можно завершить при обнаружении полного плана, алгоритм легко модифицируется для продолжения поиска улучшенных вариантов до истечения заданного времени. В этой версии Neo исследует наиболее перспективные элементы из кучи до достижения временного ограничения, после чего возвращает оптимальный обнаруженный пол-

ный план. Это позволяет пользователю управлять балансом между длительностью планирования и временем выполнения запроса. Для различных запросов могут устанавливаться индивидуальные временные лимиты согласно требованиям. При срабатывании таймера до формирования полного плана Neo переключается в экстренный режим: жадно обрабатывает наиболее перспективные потомки последнего анализированного подплана до получения завершённого плана.

3.3 Реализация проведённых исследований

В качестве набора запросов использовались запросы к базе данных IMDB – Join Order Benchmark (JOB) [46]. Это набор так называемых ”запросов реального мира”. Такие наборы ценятся больше, чем наборы синтетических запросов, так как есть мнение, что результаты, полученные на синтетических запросах, могут значимо отличаться от результатов на реальных запросах.

Пример запроса.

```
SELECT MIN(t.title) AS movie_title
FROM keyword AS k,
     movie_info AS mi,
     movie_keyword AS mk,
     title AS t
WHERE k.keyword LIKE '%sequel%'
  AND mi.info IN ('Bulgaria')
  AND t.production_year > 2010
  AND t.id = mi.movie_id
  AND t.id = mk.movie_id
  AND mk.movie_id = mi.movie_id
  AND k.id = mk.keyword_id;
```

Количество таблиц в запросах варьируются от 3 до 17. Всего в наборе чуть больше 100 запросов.

3.3.1 Машинное обучение для оценки кардинальности

При сравнении методов оценки кардинальности сравнивались два показателя:

1. Точность полученных оценок – сравнивались предсказанные моделями значения с фактическими значениями кардинальности. Фактические зна-

чения доступны после выполнения плана – это размеры промежуточных результатов, получаемые автоматически, без дополнительных вычислений.

2. Время выполнения запросов из JOB IMDB по планам, построенным классическим оптимизатором с использованием его собственных оценок кардинальности, и по планам классического оптимизатора, построенным при использованием оценок кардинальности, предсказанных моделью.

Первый показатель – способ оценить качество модели в техническом плане. Второй же даёт возможность понять, помогают ли более точные оценки кардинальности построить более быстрые планы, и если да, то насколько быстрее выполняется выбранный набор запросов.

NARU (Neural Relation Understanding) и NeuroCard (Neural Network for Cardinality Estimation)

Подход Naru [22] применим только для оценки кардинальности предикатов на одном отношении, чего не достаточно для решения задачи оптимизации. Подход был выбран для проверки эффективности метода. Обучение модели на CPU длилось 8 часов, что слишком долго для внедрения метода в продукт. При этом результаты были положительными и соответствовали результатам, заявленным авторами оригинальной статьи.

Подход NeuroCard [43] развивает концепцию NARU [22], добавляя возможность оценки кардинальности промежуточных результатов соединений. Время обучения данной модели, как и предполагалось, превышает аналогичный показатель NARU. Модель обучается без учителя, анализируя фактические данные в базе данных. При незначительных изменениях данных она сохраняет эффективность, однако для предоставления гарантий время обучения должно быть существенно меньше среднего периода значимых изменений данных. Точные количественные оценки сознательно опущены из-за отсутствия экспериментальных данных, однако очевидно, что обучение, занимающее более нескольких часов (например, 8 часов для базы IMDB, существенно уступающей по масштабам реальным корпоративным системам), представляет собой непрактичное решение.

Учитывая данный опыт, было решено изучить подходы, уже интегрированные в open-source СУБД. В качестве примера рассматривается Adaptive Query Optimizer (AQO) [42], успешно реализованный в PostgreSQL.

PostgreSQL AQO (Adaptive Query Optimizer)

В дополнение к JOB IMDB, разработчики AQO [42] использовали синтетический набор запросов TPC-DS. В экспериментах запросы классифицировались по времени выполнения: ускорение быстрых запросов оказалось невозможным, тогда как для медленных удалось достичь двукратного улучшения. Тестирование проводилось в PostgreSQL со стандартным планировщиком, поддерживающим хеш-соединения и соединения вложенными циклами.

В настоящем исследовании применялась колоночная OLAP-СУБД с массовым параллелизмом. В подобных системах соединения вложенными циклами отключены по практическим причинам (раздел 3.1.2). При такой конфигурации AQO не внес изменений в планы JOB IMDB, что породило гипотезу: метод способен лишь заменять «неоптимальные» NestLoop на HashJoin. Данная гипотеза в работе не верифицировалась.

3.3.2 Глубокое обучение для аппроксимации функции стоимости

Различия в архитектуре централизованных СУБД и массово-параллельных колоночных СУБД необходимо учитывать при модификации рассмотренных методов аппроксимации функции стоимости. Например, вместо нескольких методов чтения таблиц, используемых в централизованных СУБД, в рассмотренной колоночной СУБД используется только 1 метод. Для соединения таблиц доступен только метод HashJoin. При этом обычное дерево-план в массово-параллельных СУБД дополняется также операторами, отвечающими за решение вопроса распределённости используемых в запросе таблиц – методы BroadCast и Redistribute. Использовать предложенные в статьях, взятых за основу, признаковые описания как есть не получится. В качестве решения данной проблемы были выбраны следующие модификации:

1. Набор доступных операторов соединения был заменён. Вместо [HashJoin, NestLoop] в данной работе использовался только HashJoin, причём чтобы получить признаковые описания он был объединён с операторами перераспределения данных как показано на рисунке 3.10.
2. Так как вместо нескольких способов чтения таблиц в работе был доступен только один, соответствующим образом изменилась и кодировка.

Одной из сложностей таких подходов является сбор обучающей выборки. Требования к нему, как и всегда, предъявляются стандартные – разнообразие,

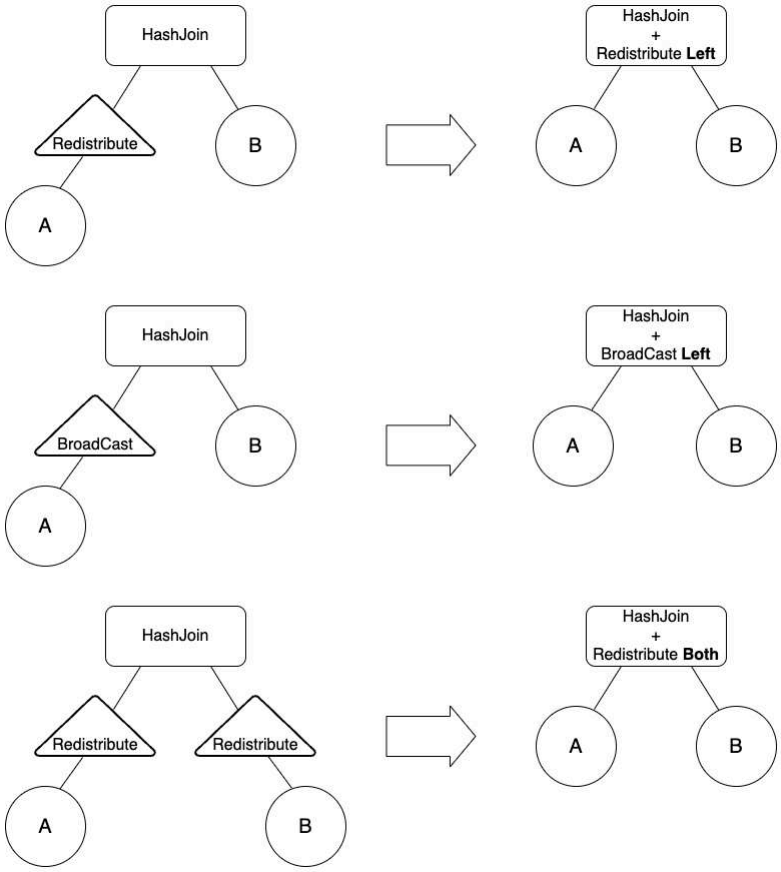


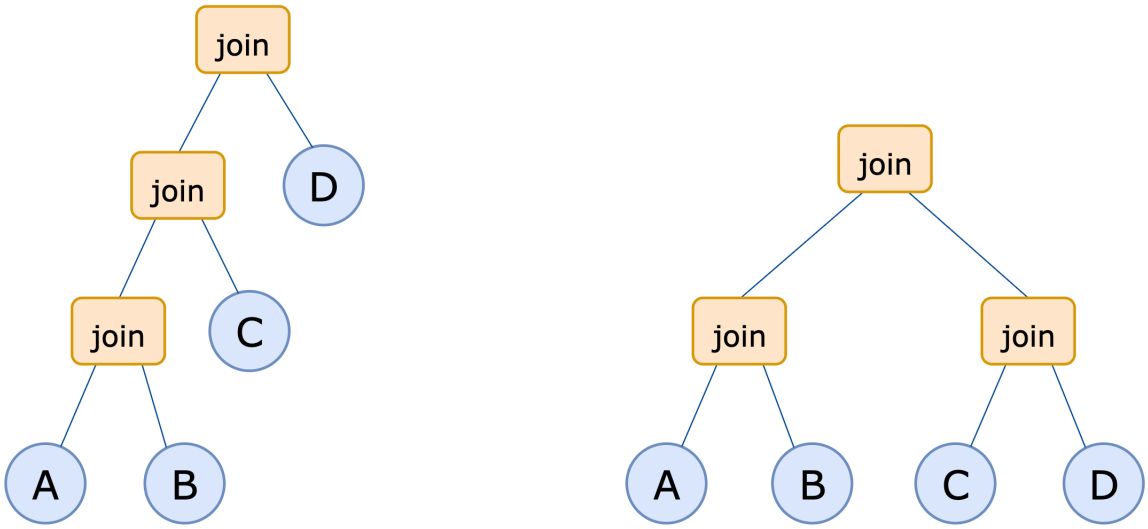
Рисунок 3.10 — Трансформация плана выполнения запроса для создания признаковых описаний.

чистота, достаточный размер. Данными в рассмотренной задаче являются пары [план; величина, характеризующая стоимость плана]. Если использовать в качестве величины, характеризующей стоимость, например, время выполнения плана, то для сбора обучающей выборки придётся выполнять запросы. Выполнение запроса требует времени, а с учётом необходимости собрать большую выборку данных – времени требуется неприемлемо много. Альтернативой этому служит подход, не требующий выполнения запросов. Традиционный оптимизатор вычисляет оценки стоимостей за доли секунды, вне зависимости от размера плана. И хотя оценки стоимости не позволят превзойти традиционный оптимизатор, можно собрать достаточно большую и разнообразную обучающую выборку за куда более короткое время и научить модель работать не хуже, чем традиционный оптимизатор. Так как нейросеть выступает в роли выделителя признаков, она может выучить основные зависимости и из таких данных. Последний слой нейросети представляет собой линейную регрессию, отображающую признаки плана в его стоимость. Обычно достаточно дообучить на более качественных данных толь-

ко последний слой. Таким образом удаётся обучить модель в два этапа. Подходы из разобранных статей используют в качестве величин, характеризующих фактическую стоимость плана, время выполнения. Практическое применение времени выполнения запроса невозможно из-за влияния внешних факторов: например, при разной текущей загрузке системы длительность исполнения идентичного плана может существенно варьироваться. Для решения этой проблемы в настоящем исследовании предложено использовать вместо времени фактическую стоимость плана — значение, рассчитываемое стандартной функцией стоимости оптимизатора, но с подстановкой реальных кардинальностей (а не их оценок), которые становятся доступны после выполнения плана. Данный подход обеспечивает получение объективного показателя временных затрат на выполнение плана, инвариантного к внешним условиям.

DQN (Deep Q-Network) подход

Признаковое описание данных, представленное в статье DQN [38], не позволяет строить ветвистые планы (рисунок 3.11). Это можно доказать простым



(а) Левый-глубокий план **(б) Ветвистый план**

Рисунок 3.11 — Примеры левого-глубокого и ветвистого планов.

примером. Одно из основных требований к признаковым описаниям при подходе Q-learning – разные состояния (state) системы должны иметь различные признаковые описания. В векторе признаков DQN можно выделить три части: левая кодирует финальное состояние (терминальное, когда все таблицы соединены), средняя

кодирует текущее состояние (state) – все таблицы, которые соединены на данный момент (для первого шага – одна из двух таблиц первого соединения), и правая кодирует действие (action) – таблицы, которые присоединяются на этом шаге к имеющейся цепочке соединений, закодированных в предыдущей части. Модель на основе этого вектора предсказывает минимальную остаточную стоимость, то есть стоимость текущего действия плюс сумму стоимостей всех последующих действий при условии что они будут оптимальны. Допустим в запросе есть 4 таблицы и условия соединения таковы, что план выполнения запроса может быть ветвистым, например, таким: $(A \bowtie B) \bowtie (C \bowtie D)$. Такой план подразумевает, что в процессе его построения в какой-то момент времени текущим состоянием (state) было наличие одной пары соединённых таблиц – например, $(A \bowtie B)$. На этом шаге действием (action) обязательно было соединение двух других таблиц – $(C \bowtie D)$. Так как модель предсказывает стоимость последующих действий начиная с текущего, она должна знать, что таблицы A и B уже соединены. Также одна из таблиц, соединяемых на текущем шаге, должна быть закодирована в среднем векторе. Получается, что в среднем векторе на этом шаге будут закодированы три таблицы – A, B и либо C, либо D – в зависимости от того, какая таблица в соединенияя будет слева, а какая справа. Но этот вектор будет совпадать с вектором состояния, в котором таблицы A, B и C или D уже соединены, что недопустимо. Таким образом, предложенный в статье DQN способ кодирования состояний и действий (state и action) позволяет строить только левые-глубокие деревья. Такие деревья образует по сути отдельный класс деревьев. Сравнение левых-глубоких планов с ветвистыми планами в части эффективности до сих пор ни кем не проводилось. Столь явное ограничение метода DQN вызывает опасения – а может ли вообще левый-глубокий план быть лучше ветвистого? Можно ли найти для абсолютного большинства запросов левый-глубокий план, который был бы эффективнее субоптимального ветвистого плана, построенного традиционным оптимизатором? Чтобы приблизиться к ответам на эти вопросы, в рамках данной работы был проведён эксперимент: для запросов IMDB JOB с малым количеством таблиц (5 и менее) был произведён полный перебор планов. Для каждого запроса определялся как лучший план (оптимальный), так и лучший левый-глубокий план. Результаты сравнивались между собой, а также с планами, построенными традиционным оптимизатором. Эксперимент показал следующее:

1. Среди оптимальных планов нет ни одного левого-глубокого.

2. В 80% запросов лучший левый-глубокий план проигрывает плану, построенному традиционным оптимизатором.

С одной стороны, эти два факта ставят под сомнение целесообразность использования метода DQN, с другой стороны – эксперимент проводился только для запросов с малым количеством таблиц. Такие запросы хорошо оптимизируются и традиционным оптимизатором. В больших же запросах ошибки в оценках накапливаются с ростом глубины дерева, что приводит к планам, далёким от оптимальных. Несмотря на результаты эксперимента, работа с методом DQN была продолжена.

Модель представляет собой двухслойный перцептрон (полносвязную нейронную сеть), принимающий на вход текущий запрос и частично построенный подплан. Обучение модели направлено на прогнозирование Q-функции, которая оценивает не мгновенную стоимость, а долгосрочные последствия включения конкретного соединения в подплан. Следовательно, здесь необходима информация о том, как выглядит финальное состояние системы (какие таблицы должны присутствовать в финальном плане). Поэтому используется кодировка на уровне запроса – кодировка всего запроса, так же как и кодировка текущего подплана. Кодировка на уровне запроса, Q, кодирует все атрибуты и их оценки селективностей. То есть для заданной схемы базы данных с таблицами $T_1, \ldots, T_n$, имеющими атрибуты $A_{i_1}, \ldots, A_{i_m}$, кодировка на уровне запроса для атрибута A_{i_j} будет следующей:

- 0 если T_i не участвует в запросе
- предсказанная селективность атрибута A_{i_j}, если A_{i_j} имеет предикат (фильтр)
- 1 если T_i участвует в запросе, но для A_{i_j} в запросе нет предиката

Например, для схемы базы данных:

- Emp(id, name, rank)
- Pos(rank, title, code)
- Sal(code, amount)

и запроса:

```
SELECT *
  FROM Emp, Pos
 WHERE Emp.rank = Pos.rank
  AND Emp.id > 200
```

кодировкой на уровне запроса будет вектор $[0.2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0]$, где 0.2 – селективность предиката $Emp.id > 200$.

Помимо кодировки на уровне запроса здесь требуется кодировка текущего соединения (action). Кодировка соединения состоит из кодировки левого поддерева, L, правого поддерева (всегда лист), R, и способа соединения, J. Левое поддерево кодируется 1 и 0 для каждого атрибута в схеме базы данных в зависимости от того, относится ли он к таблице, являющейся частью этого поддерева. Рассмотрим запрос:

```
SELECT *
  FROM Emp, Pos, Sal
 WHERE Emp.rank = Pos.rank
  AND Pos.code = Sal.code
  AND Emp.id > 200
```

и предположим, что на текущем шаге построения плана таблицы Emp и Pos уже соединены, а действием является присоединение к ним таблицы Sal (рисунок 3.12).

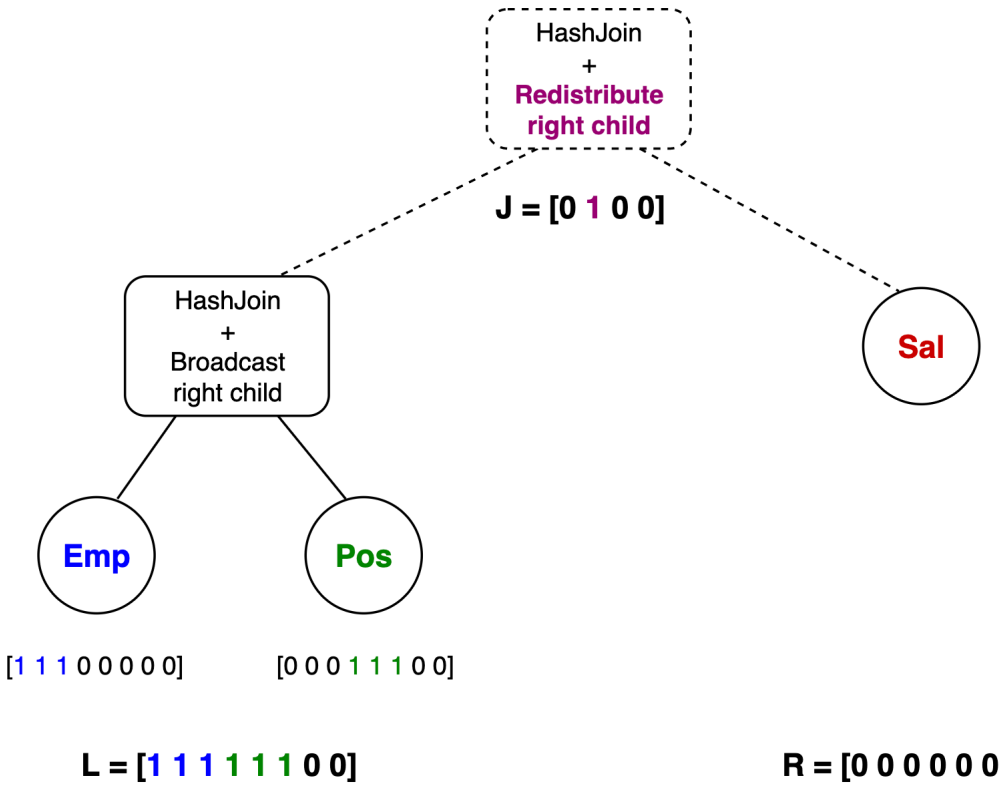


Рисунок 3.12 — Пример создания векторного представления для текущего состояния и действия.

В этом случае кодировками Q, L и R будут вектора:

– $Q = [0.2 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1]$
– $L = [\;\; 1 \;\;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 1 \;\; 0 \;\; 0]$
– $R = [\;\; 0 \;\;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 1]$

Полной кодировкой запроса и текущего соединения будет конкатенация векторов Q, L, R и вектора, кодирующего тип соединения J. Так как в данной работе использовался только один способ соединения – HashJoin, но были дополнительные операторы перераспределения данных, вектор, кодирующий способ соединения, по сути, кодировал способ перераспределения данных в дочерних узлах. Вектор J мог иметь следующие виды:

– $J = [0 \;\;\; 0 \;\;\; 0 \;\;\; 0]$, если перераспределение данных дочерних узлов не требуется
– $J = [1 \;\;\; 0 \;\;\; 0 \;\;\; 0]$, если для левого дочернего узла применяется оператор Redistribute
– $J = [0 \;\;\; 1 \;\;\; 0 \;\;\; 0]$, если для правого дочернего узла применяется оператор Redistribute
– $J = [1 \;\;\; 1 \;\;\; 0 \;\;\; 0]$, если для двух дочерних узлов применяется оператор Redistribute
– $J = [0 \;\;\; 0 \;\;\; 1 \;\;\; 0]$, если для левого дочернего узла применяется оператор Broadcast
– $J = [0 \;\;\; 0 \;\;\; 0 \;\;\; 1]$, если для правого дочернего узла применяется оператор Broadcast

Архитектура нейросети представлена на рисунке 3.13. $\{q_i, l_i, r_i\}$ – первый слой нейросети, соединяющийся со скрытым слоем из 32 узлов h_i, которые в свою очередь соединены с выходным слоем O размерности 1. В качестве функции активации использовалась кусочно постоянная функция ReLU. $\{q_i, l_i, r_i\}$ кодируют кардинальности и текущий подплан, а скрытый слой комбинирует информацию, выходящую из первого слоя, чтобы создать промежуточные величины, включающие в себя различные взвешенные комбинации атрибутов. Несмотря на простоту, такая архитектура может моделировать сложные взаимосвязи между атрибутами.

Вместо использования стандартного подхода обучения с учителем, здесь применяется Q-learning, один из подходов обучения с подкреплением, которые тренирует модель аппроксимировать Q-функцию – функцию, которая вычисляет суммарную стоимость следующих действий по трём вещам: описание запро-

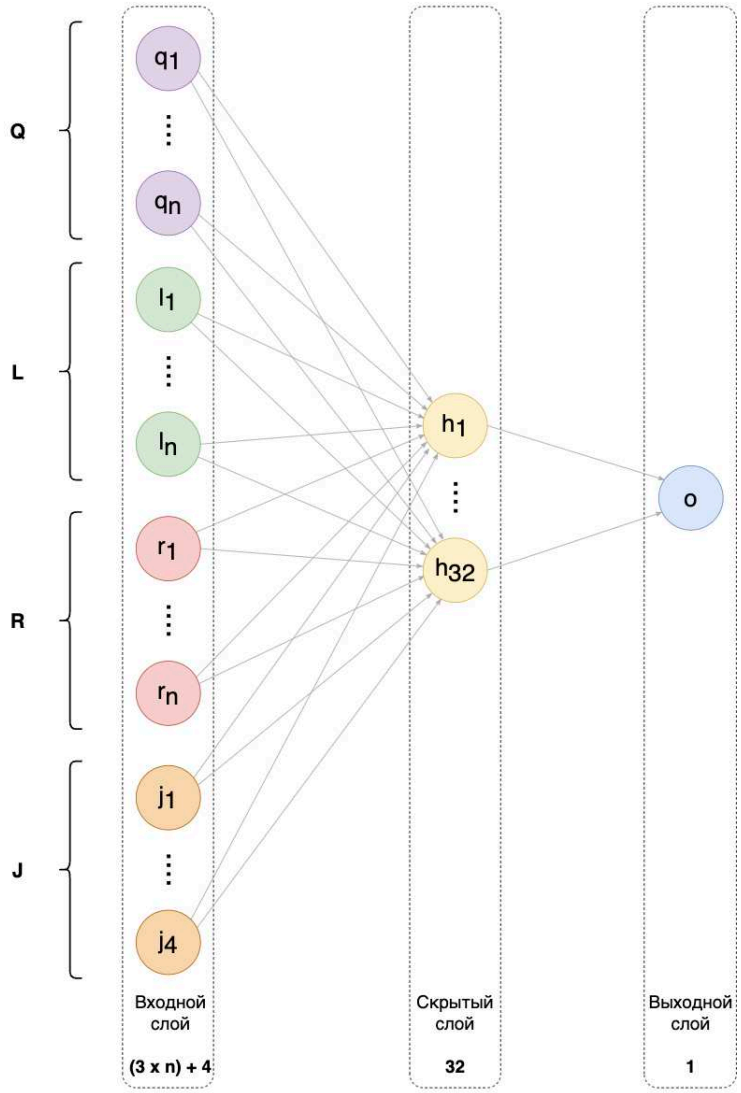


Рисунок 3.13 — Архитектура нейросети, использовавшаяся в данной работе в экспериментах с подходом DQN.

са (вектор Q), подплан, построенный на текущий момент (вектор L) и действие, предпринимаемое на текущем шаге (вектора R и J). В процессе построения плана с использованием Q-функции, можно получить оценку суммарной стоимости всех последующих соединений для каждого доступного на данном шаге действия, выбрать самое дешёвое и продолжать действовать жадно пока план не будет построен полностью. Каждый тренировочный экземпляр – это кортеж ($state$, $action$, $cost(action)$, $state'$), где

- $state$ – граф текущего состояния (вектора Q и L)
- $action$ – присоединение новой таблицы к имеющемуся подплану (вектора R и J)
- $cost(action)$ – стоимость этого присоединения
- $state'$ – новое состояние после выполнения присоединения $action$

Нейросеть – это параметризованная модель для аппроксимации Q-функции. Обозначим модель $\tilde{Q}_\theta$, $\tilde{Q}_\theta(f_{state},\ f_{action}) \approx \tilde{Q}(state,\ action)$, где f_{state} – это вектор, кодирующий состояние, а f_{action} – это вектор, кодирующий действие. θ – параметры модели, которые инициализируются методом Кайминга [47]. Для каждого кортежа из обучающей выборки можно вычислить целевое значение по формуле:

$$y_i = cost(action) + \min_{action'} \tilde{Q}_\theta(state',\ action') \tag{3.10}$$

где $action'$ пробегает все возможные в состоянии $state'$ действия (рисунок 3.14).

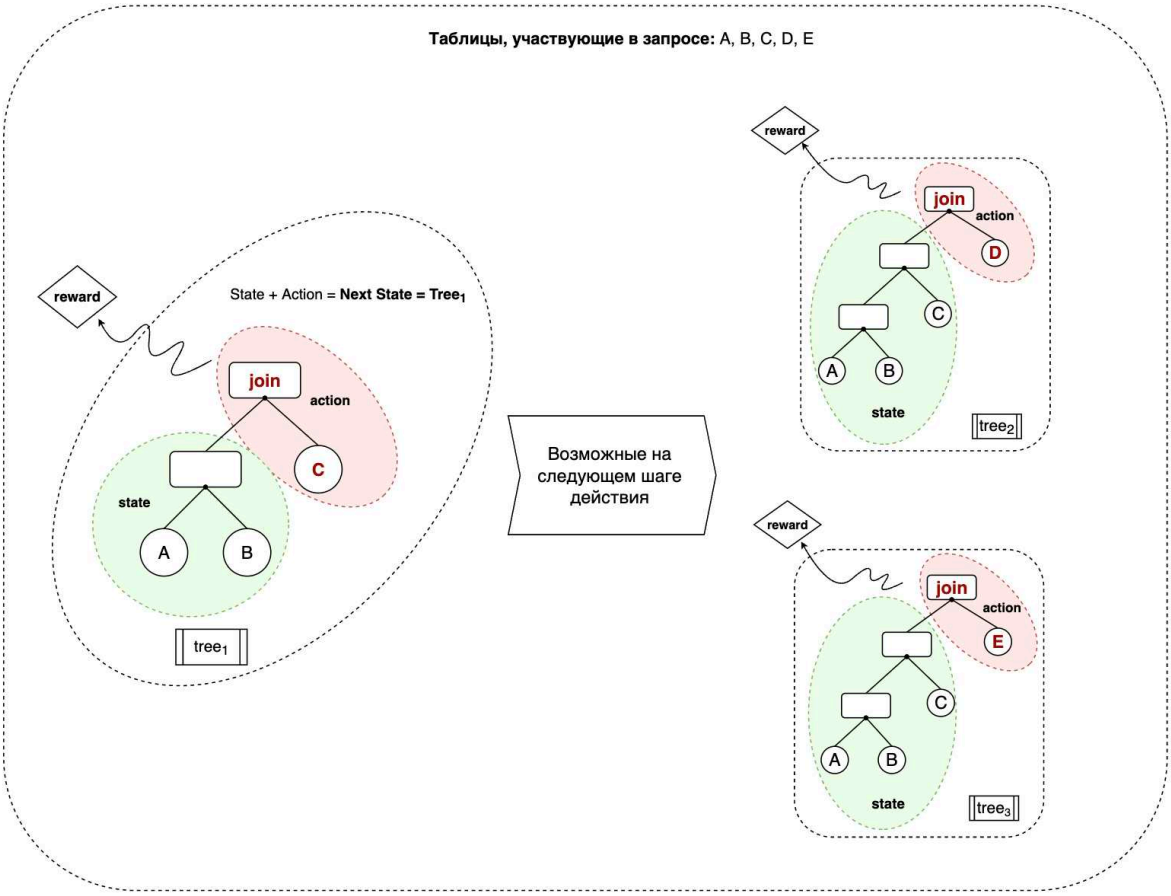


Рисунок 3.14 — Пример определения терминов state, action, reward и next state в дереве (подплане).

$\{y_i\}$ могут быть использованы как целевые значения в задаче регрессии. Если бы $\tilde{Q}$ была истинной Q-функцией, выполнялось бы следующее равенство:

$$\tilde{Q}(state,\ action) = cost(action) + \min_{action'} \tilde{Q}_\theta(state',\ action') \tag{3.11}$$

Функция ошибки вычисляется по формуле:

$$L(\tilde{Q}) = \frac{1}{N}\sum_{i=1}^{N}\Big(y_i - \tilde{Q}_\theta(state, action)\Big)^2 \tag{3.12}$$

Параметры модели $\tilde{Q}_\theta$, аппроксимирующей Q-функцию, могут быть оптимизированы с помощью градиентных методов [48].

Чтобы повысить стабильность тренировочного процесса и, следовательно, его скорость, использовались две нейросети. Первая нейросеть тренировалась предсказывать $\{y_i\}$, являющиеся суммой стоимости текущего действия и оценкой минимальной суммарной стоимости всех последующих действий, требующихся для построения полного плана. Второе слагаемое можно получить, взяв предсказания второй нейросети на соответствующих входных данных. Вторая нейросеть в точности повторяет архитектуру первой и отличается лишь тем, что её параметры заморожены и обновляются лишь раз в K шагов посредством копирования новых весов первой нейросети во вторую. Обозначим вторую нейросеть как $\tilde{Q}_{\theta^-}$, где θ^- означает, что веса модели не обновляются. K – гиперпараметр, который стоит подбирать исходя из поведения тренировочного процесса в конкретной ситуации. В данной работе вторая нейросеть обновляла веса раз в 1000 шагов.

После обучения на оценках стоимостей, модель дообучалась на фактических значениях стоимостей. Для этого первый слой замораживался и изменялись веса только второго слоя.

В исходной конфигурации нейросеть оказалась неспособной к обучению. Для решения проблемы были предложены три модификации:

1. Добавление Layer Normalization [49].
2. Входной вектор данных дополняется числом, равным разности между общим количеством таблиц в запросе и текущим количеством таблиц в подплане. Хотя исходный входной вектор содержит всю информацию для определения оставшихся шагов построения плана, явное предоставление этого значения существенно упрощает задачу обучения для нейросети.
3. Double Q-Learning [50] – вместо минимума по всем возможным действиям в уравнении (3.10) здесь с помощью первой нейросети (обновляющейся на каждом шаге) выбирается самое дешёвое действие, а дальше с помощью второй нейросети (обновляющейся раз в K шагов) вычисляет-

ся оценка Q-функции для этого конкретного действия. Вместо формулы

$$\tilde{Q}_\theta(state,\ action) = cost(action) + \min_{action'} \tilde{Q}_{\theta^-}(state',\ action') \quad (3.13)$$

в Double Q-Learning исползуется формула

$$\tilde{Q}_\theta(state,\ action) = cost(action) + \tilde{Q}_{\theta^-}(state',\ {}_{action'}\tilde{Q}_\theta(state',\ action')) \quad (3.14)$$

После внесения трёх указанных модификаций процесс обучения стал сходиться. Однако наилучший результат для набора запросов IMDB JOB уступал традиционному оптимизатору: совокупное время выполнения запросов оказалось на 43% больше (таблица 1). Одним из факторов, объясняющих этот результат, могут быть ограничения на структуру планов, формируемых данным подходом.

	Время выполнения 100 запросов.	
	Планы классического оптимизатора.	**Планы обученного оптимизатора.**
Версия из оригинальной статьи.	**1.5 часа**	**7 часов**
Версия с рядом модификаций.		**2.15 часа**

Таблица 1 — Результаты сравнения подхода DQN с классическим оптимизатором запросов.

NEO (Neural Optimizer) подход

В отличие от подхода DQN, метод Neo способен формировать ветвистые планы выполнения запросов. В данном подходе не используется концепция $action$ - вместо этого алгоритм работает исключительно с понятием $state$, оценивая потенциал каждого подплана независимо от конкретных действий на каждом шаге. Модель предсказывает не остаточную стоимость, а минимально возможную стоимость полного плана, который может быть получен из текущего подплана. На рисунке 3.15 представлена визуализация одного шага алгоритма Neo по построению плана выполнения запроса. Векторные представления запросов и подпланов соответствуют оригинальной работе с учётом внесённых модификаций, аналогичных изменениям, применённым в подходе DQN.

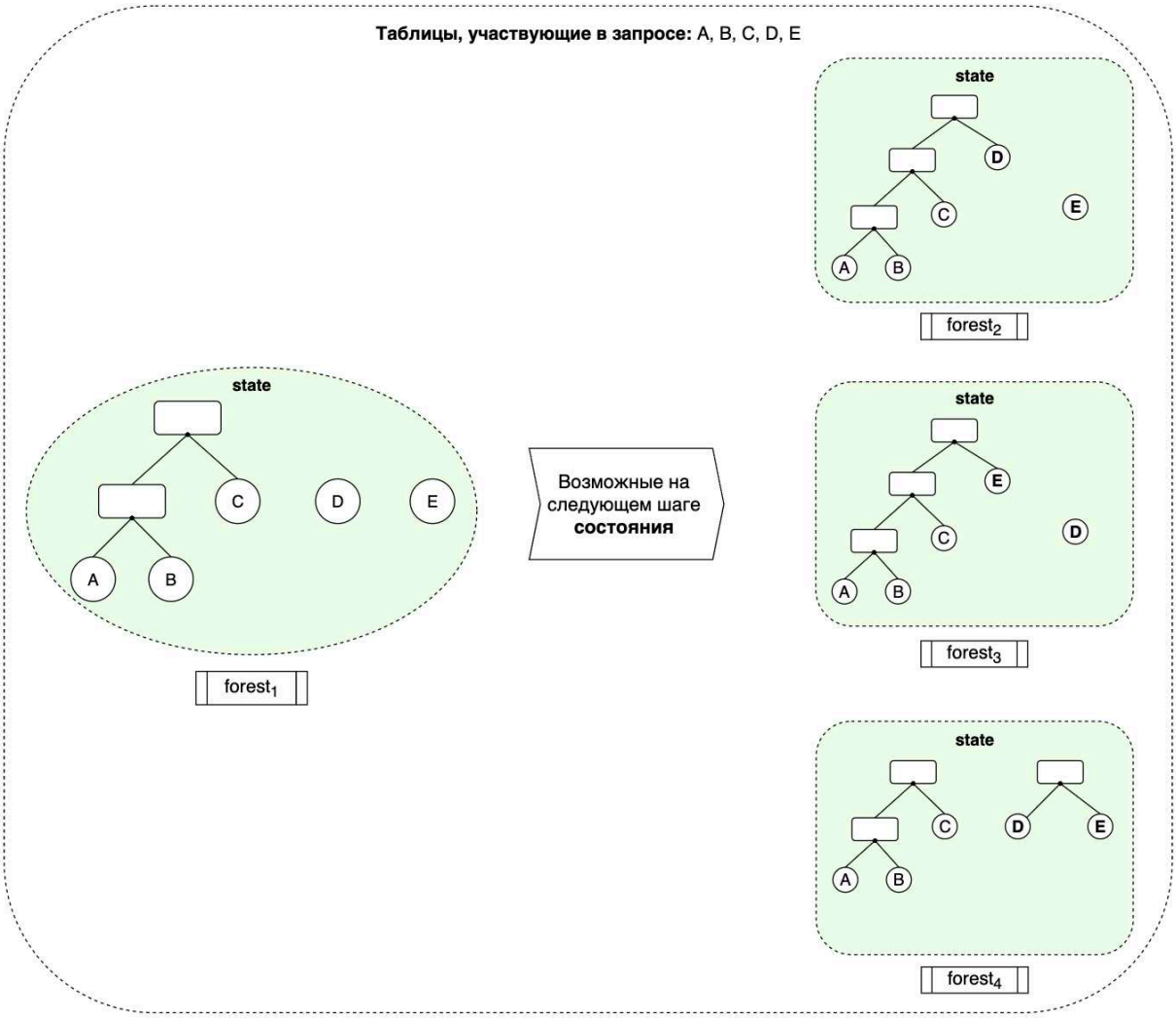


Рисунок 3.15 — Пример множества возможных состояний в подходе Neo.

Архитектура рабочего процесса. Оригинальный подход (рисунок 3.16(а)) обладает существенными ограничениями. Во-первых, модель требует значительного объема ”опыта”для достижения приемлемого качества - экспериментально установлено, что необходимо не менее 500 уникальных планов для каждого запроса. Это означает необходимость выполнения сотен неоптимальных планов на реальной СУБД, что приводит к двум проблемам: (1) неприемлемо большие временные затраты и (2) чрезмерная нагрузка на рабочую базу данных. Во-вторых, подготовка R-векторов требует значительных вычислительных ресурсов.

На рисунке 3.16(б) представлена модифицированная архитектура Neo, решающая эти проблемы. Ключевые изменения включают:

– Замену реального выполнения планов на процедуру EXPLAIN, которая мгновенно вычисляет оценку стоимости плана с использованием стандартного оценщика кардинальности.

– Отказ от использования R-векторов.

Данная модификация позволяет за секунды получить необходимый объем тренировочных данных без нагрузки на рабочую СУБД.

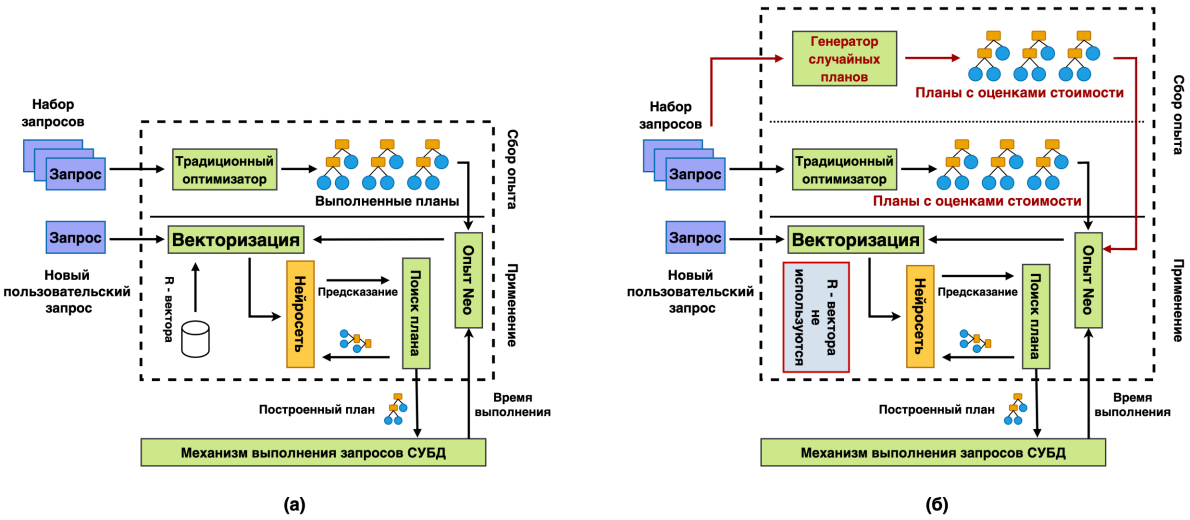


Рисунок 3.16 —
(а) Дизайн системы Neo в оригинале.
(б) Модифицированный дизайн системы Neo – модификации подсвечены бордовым цветом.

Использование архитектуры Transformer. Поскольку входные данные представляют собой набор деревьев (в простейшем случае - листья), а выходное значение - одно число, возникает необходимость агрегации множества векторов в единое представление. В оригинальной архитектуре для этого применялось динамическое объединение (Max Pooling или Average Pooling) [51] - стандартный, но ограниченный подход, широко используемый в компьютерном зрении, но приводящий к потере информации.

В данной работе для решения этой проблемы предложено использовать механизм Self-Attention из архитектуры Transformer [52], который вычисляет взвешенную сумму векторов, где веса определяются самими векторами. Этот подход:

– Позволяет сохранить больше информации при агрегации.
– Компенсирует отсутствие R-векторов в модифицированной архитектуре.
– Обеспечивает более богатое представление данных.

Соответствующая модификация архитектуры показана на рисунке 3.17.

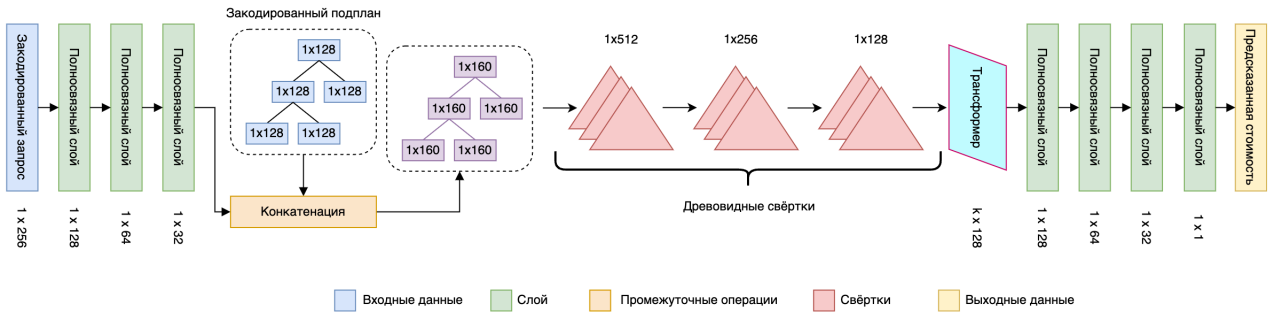


Рисунок 3.17 — Модификация архитектуры нейросети Neo.

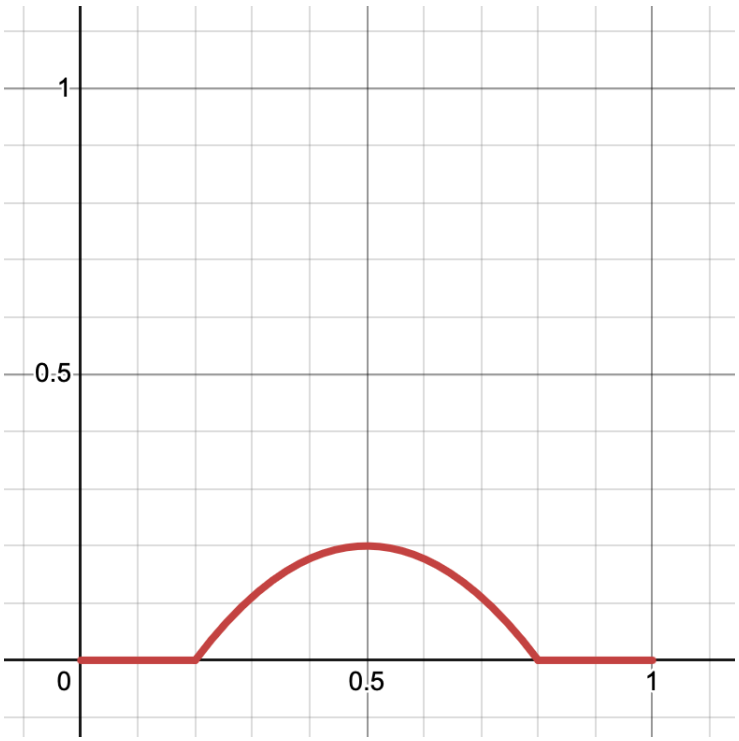


Рисунок 3.18 — Зависимость максимальной величины шума от x.

Аугментация данных. У использовавшейся СУБД есть одна особенность – сбор статистик по данным при запуске СУБД обладает элементом стохастики. Из-за этого селективности предикатов флуктуируют от запуска СУБД к запуску. При этом модель очень чувствительна к этим изменениям. Чтобы снизить чувствительность модели и таким образом повысить её надёжность, в работе была предложена своего рода аугментация данных. Её суть заключалась в том, чтобы при формировании батча добавлять к селективностям шум. Масштаб шума зависил от значения селективности: чем ближе селективность к 0.5, тем больше возможный шум, причём на отрезках $[0, 0.2]$ и $[0.8, 1]$ шум не добавлялся – селективности из этих отрезков имеют важное значение. Они показывают, что после применения фильтра таблица либо станет значимо меньше, либо её размер всё ещё будет близок к исходному. Добавление шума в таких случаях может навредить. На отрезке

$[0.2, 0.8]$ максимальный размер шума ограничен функцией $y = 0.2 - \frac{20}{9}(x - 0.5)^2$ – параболой с максимумом равным 0.2 и достигающимся при $x = 0.5$, и ветвями, пересекающими ось абсцисс в точках 0.2 и 0.8 (рисунок 3.18). Такое решение сделало обученную модель устойчивой к флуктуациям селективностей – при перезапуске СУБД одна и та же модель показывала себя одинаково.

Получение оценок неопределённости предсказаний. Хотя новый метод формирования обучающей выборки обладает существенными преимуществами, он не обеспечивает полного покрытия пространства возможных планов и не гарантирует отсутствия шума в обучающих данных. Эти ограничения могут привести к ошибочным предсказаниям нейронной сети, что особенно критично в рассматриваемой задаче. Для снижения рисков предлагается расширить модель, добавив оценку неопределённости предсказания стоимости наряду с самим предсказанием. Подпланы, для которых оценка неопределённости превышает установленный пороговый уровень, должны автоматически исключаться из рассмотрения, даже если их предсказанная стоимость выглядит привлекательно. Такой подход позволяет компенсировать фундаментальные ограничения метода формирования обучающих данных и минимизировать вероятность выбора субоптимальных планов выполнения запросов.

Неопределённость, в свою очередь, делится на алеаторическую и эпистемическую [53]. Их суть хорошо проиллюстрирована на рисунке 3.19. Эпистемическая неопределённость описывает то, чего модель не знает, потому что данных для обучения нет в некоторой части области определения. Эпистемическая неопределённость отражает недостаточность данных и знаний о моделируемой системе. Она уменьшается по мере увеличения полноты обучающей выборки, но сохраняется в областях с недостаточным количеством обучающих примеров. В отличие от неё, алеаторическая неопределённость обусловлена фундаментальной стохастичностью наблюдаемых процессов и не может быть устранена даже при увеличении объёма данных. В случае ошибок измерений такая неопределённость называется гомоскедастической, демонстрируя постоянный уровень вариативности. Когда неопределённость варьируется в зависимости от входных параметров, она классифицируется как гетероскедастическая. Байесовский статистический подход предоставляет механизм для интеграции априорных знаний с наблюдаемыми данными при построении выводов. Его принципиальная особен-

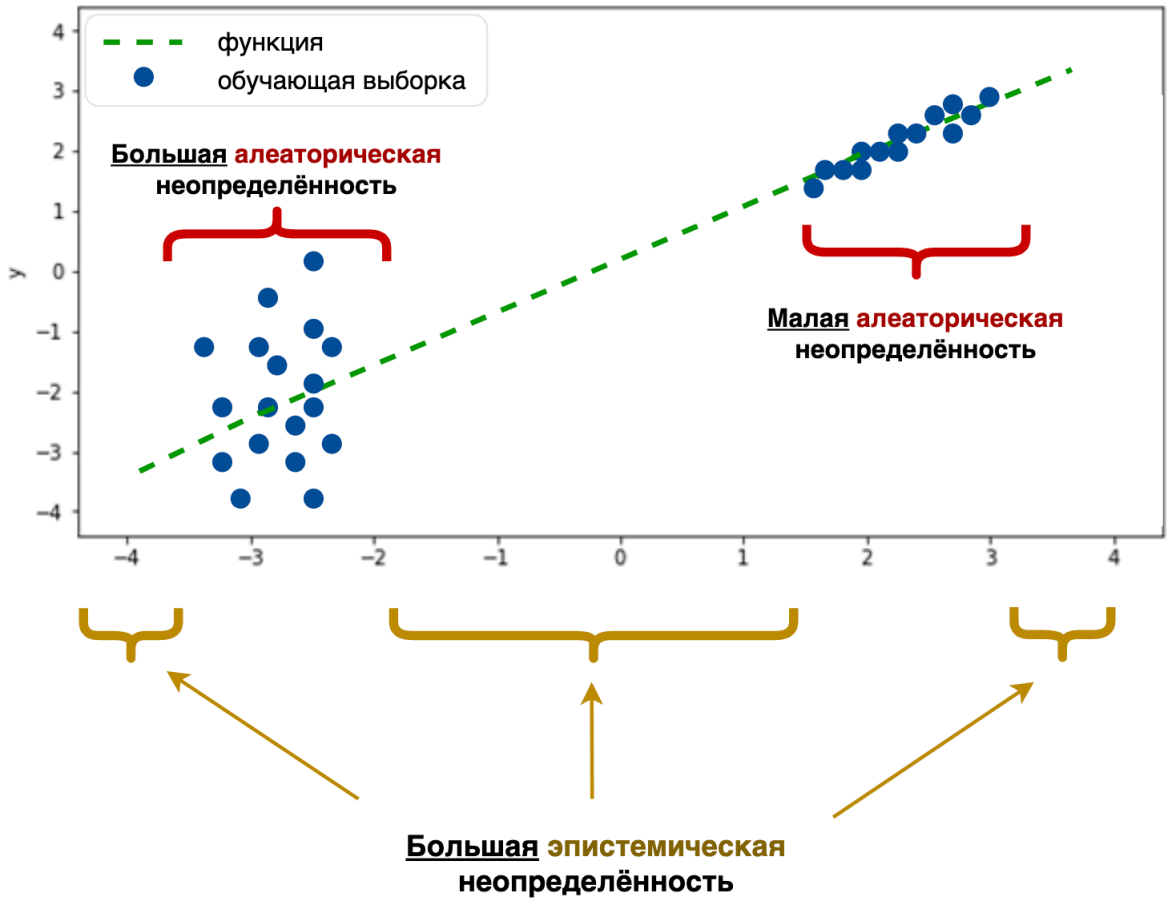


Рисунок 3.19 — Демонстрация различных видов неопределённости в контексте линейной регрессии.

ность заключается в рассмотрении параметров модели как вероятностных распределений, а не фиксированных значений. Такой подход открывает возможность оценки неопределённости через анализ распределения весов модели, что принципиально невозможно в классических детерминированных моделях.

– **Получение оценок эпистемической неопределённости.**
  Ансамблирование представляет собой эффективный метод, основанный на обучении множества моделей или их копий на различных подмножествах данных, с последующим объединением их предсказаний в единое распределение. Однако данный подход требует значительных вычислительных ресурсов, что привело к разработке альтернативной методики - использования dropout [54] в качестве байесовской аппроксимации ансамбля моделей [55]. Dropout, изначально предложенный как метод регуляризации, случайным образом обнуляет веса сети согласно распределению Бернулли во время обучения. Как показано в работе [55], нейронная сеть с dropout-слоями перед каждым полносвязным слоем математиче-

ски эквивалентна байесовской аппроксимации гауссовского процесса. В этом подходе каждое подмножество активных нейронов определяет уникальную конфигурацию сети. Процесс обучения можно рассматривать как совместное обучение 2^m различных моделей, где m - количество нейронов в сети, причем для каждого батча обучается случайно выбранная конфигурация. Для практического применения предлагается выполнять множественные прогоны модели (порядка нескольких сотен) с включенным dropout-механизмом на этапе предсказания. Среднее значение полученных предсказаний служит точечной оценкой, а их дисперсия - мерой эпистемической неопределенности модели. Этот подход позволяет оценить надежность предсказаний без необходимости обучения полноценного ансамбля моделей.

– **Получение оценок алеаторической неопределённости.**
Эпистемическая неопределённость характеризует ограничения самой модели, тогда как алеаторическая неопределённость отражает внутреннюю стохастичность данных и принципиальную невозможность полного объяснения наблюдаемых явлений. Алеаторическая неопределённость подразделяется на два типа: гомоскедастическую, остающуюся постоянной для всех входных данных, и гетероскедастическую, варьирующуюся в зависимости от характеристик входных данных. Гетероскедастическая неопределённость, будучи функцией от входных параметров, может быть непосредственно смоделирована и предсказана нейронной сетью как часть её выходных данных. В отличие от неё, гомоскедастическая неопределённость оценивается как глобальный параметр модели, значение которого определяется спецификой решаемой задачи и остаётся неизменным для всех входных данных. Это различие имеет принципиальное значение при проектировании архитектур нейронных сетей для оценки неопределённости предсказаний. Изучение гетероскедастической неопределённости выполняется путём замены функции потерь среднеквадратичной ошибки на следующую:

$$L = \sum_{i=1}^{N} \frac{(y_i - \hat{y}_i)^2}{2N\sigma(x_i)^2} + \frac{1}{2}\log\sigma(x_i)^2 \tag{3.15}$$

Модель предсказывает как среднее значение $\hat{y}$, так и дисперсию σ^2. В случае значительной невязки между предсказанными и фактическими значениями модель увеличивает оценку дисперсии, отражая низкую уверенность в своих предсказаниях. Однако логарифмическая составляющая в функции потерь играет критическую роль, ограничивая чрезмерный рост дисперсии и предотвращая её стремление к бесконечности. Этот механизм обеспечивает баланс между чувствительностью к ошибкам и численной устойчивостью модели. Формально это выражается в том, что при больших невязках логарифмический член доминирует, смягчая влияние квадратичного члена на итоговое значение дисперсии.

Эксперименты показали, что в данной задаче алеаторическая неопределённость всегда пренебрежимо мала, а эпистемическая неопределённость достаточно часто оказывается высока. Это обусловлено сложностью покрытия обучающей выборкой всей области определения (всего пространства планов выполнения запросов). 500 случайных планов для запросов с количеством таблиц более 8 – это крайне малая доля всех возможных планов. Поэтому при использовании модели во время построения плана, в качестве кандидатов ей часто предоставляются подпланы, не похожие на подпланы из обучающей выборки. Этот подплан может быть как ”хорошим” так и ”плохим”. В целях снижения риска выбора неудачного плана имеет смысл не рассматривать кандидатов с высоким уровнем эпистемической неопределённости модели. Перед заменой традиционного оптимизатора ”умным” можно итеративно подобрать порог для значения неопределённости, по которому будет приниматься решение об исключении кандидата из рассмотрения. Таким образом можно обезопасить систему от ошибок при выборе плана.

Анализ результатов подхода Neo Аналогично результатам, полученным для метода DQN, первоначальная версия подхода Neo из оригинальной работы показала худшую производительность по сравнению с традиционным оптимизатором. Эксперименты с Neo проводились в двух различных режимах работы СУБД: централизованном табличном режиме и массово-параллельном колоночном режиме. В централизованном табличном режиме модифицированная версия Neo продемонстрировала значительное улучшение, построив планы выполнения запросов, которые в совокупности выполнялись на 40% быстрее планов, сгенерированных традиционным оптимизатором. Однако в массово-параллельном колоночном ре-

жиме результаты оказались противоположными - планы, построенные ”умным” оптимизатором, выполнялись на 30% медленнее (таблица 2).

	Время выполнения 100 запросов.			
	Табличный режим СУБД		**Колоночный режим СУБД**	
	Традиционный оптимизатор	**Обученный оптимизатор**	**Традиционный оптимизатор**	**Обученный оптимизатор**
Версия из оригинальной статьи.	**2.5 часа**	**5 часов**	**1.5 часа**	**3 часа**
Версия с рядом модификаций.		**1.5 часа**		**2 часа**

Таблица 2 — Результаты сравнения подхода Neo с классическим оптимизатором запросов

Выводы к третьей главе

Результаты, полученные в ходе проведённого исследования, опубликованы в работах [56—58] и позволяют сделать ряд обобщающих выводов о возможностях и ограничениях применения современных методов ускорения аналитических SQL-запросов в контексте задач динамичного формирования групп объектов.

Во-первых, необходимо отметить, что среди протестированных подходов лишь один — модель *Neo*, представляющая собой обучаемую функцию стоимости запросов, — продемонстрировала устойчивое ускорение выполнения запросов на заранее определённом тестовом наборе. Использование модели позволяло избежать неоптимальных планов, формируемых традиционными эвристическими оптимизаторами, особенно в случаях с высокой сложностью соединений и нетривиальной структурой условий. При этом прочие исследованные методы (включая модели оценки кардинальности, стратегии замены отдельных компонентов оптимизатора и адаптивные механизмы типа AQO) либо показали крайне ограниченное улучшение, либо оказались неработоспособными в условиях исследуемой архитектуры.

Однако даже относительно успешный случай применения *Neo* требует критического переосмысления с позиции прикладной пригодности. Все полученные улучшения справедливы только при условии фиксированного состава данных и стабильного запроса. При этом:

– сама модель обучается на выборке запросов, сформированной для конкретного состояния базы данных;

– любые изменения в данных (обновление, добавление, удаление) могут приводить к рассогласованию между моделью и реальной системой;
– для поддержания точности модели необходимо её постоянное дообучение или периодическое переобучение на новых данных.

В случае статичной системы такой подход может быть оправдан, однако в задаче динамичного формирования групп объектов, где обновления происходят регулярно, объекты появляются и исчезают, а признаки пересчитываются многократно, возникает необходимость либо в непрерывном онлайн-обучении, либо в построении механизма оценки степени устаревания модели и её релевантности в текущий момент времени. Последнее представляет собой отдельную, принципиально нерешённую задачу, требующую ресурсов, экспериментальной валидации и архитектурной поддержки. Отсутствие универсального решения для отслеживания деградации моделей и автоматического триггера на переобучение делает подобные системы уязвимыми и ненадёжными в условиях высокой изменчивости данных.

Таким образом, даже несмотря на теоретические преимущества методов машинного обучения для оптимизации SQL-запросов, их применение в условиях динамичной среды оказывается сопряжено с избыточными накладными расходами и неопределённостями. Иными словами, затраты на внедрение и поддержку подобных подходов в большинстве случаев превышают потенциальный выигрыш в производительности.

Исходя из этого, в рамках данной работы был сделан принципиальный вывод о невозможности универсального применения исследованных методов ускорения аналитических запросов в задачах, где признаки объектов формируются посредством сложных агрегаций. Подобные задачи, к сожалению, остаются в зоне ограниченной оптимизируемости.

Тем не менее, важно подчеркнуть, что существует обширный класс задач, в которых признаки объектов либо доступны непосредственно в исходных данных, либо могут быть получены простыми трансформациями без необходимости обращения к агрегатным операциям. Именно на такие задачи ориентировано дальнейшее развитие предлагаемого метода динамичного формирования групп объектов по принципу идентичности. Отказ от попытки универсального ускорения запросов позволил сосредоточиться на более реалистичных сценариях, где возможна

высокочастотная переоценка схожести объектов и последующая кластеризация с приемлемыми вычислительными затратами.

Следующая часть диссертации посвящена архитектуре, алгоритмам и экспериментальному обоснованию метода динамичного формирования групп, разработанного с учётом этих ограничений.

Глава 4. Методы динамичного формирования групп объектов по принципу идентичности

4.1 Структурная и семантическая согласованность как основа интероперабельности

Формирование групп объектов по принципу идентичности в динамичной информационной системе требует формального описания связей между объектами. Такие связи отражают вероятность того, что два объекта представляют одну и ту же сущность. Модель, описывающая эти связи, служит основой для последующего применения алгоритмов кластеризации и анализа структурной согласованности. В данной главе под «структурой» понимается графовая модель, задающая схему потенциальной идентичности между объектами.

Рассмотрим множество объектов:

$$I = \{i_1, i_2, \ldots, i_N\},$$

где каждый объект $i \in I$ описывается вектором признаков $\mathbf{d}_i = (p_{i1}, p_{i2}, \ldots, p_{iq})$.

На множестве пар объектов вводится функция семантической близости $F(i,j) \in [0,1]$, которая отражает степень их идентичности. Эта функция может быть как вручную заданным расстоянием в признаковом пространстве, так и обучаемой моделью. Далее фиксируется порог τ:

$$F(i,j) \geq \tau \quad \Rightarrow \quad \text{объекты } i \text{ и } j \text{ считаются идентичными.}$$

На основе значений $F(i,j)$ строится граф связей:

$$G = (V, E, w),$$

где:

- $V = I$ — множество вершин, соответствующих объектам;
- $E \subseteq \{(i,j) \mid F(i,j) \geq \tau\}$ — множество рёбер между “схожими” объектами;
- $w(i,j) = F(i,j)$ — вес ребра, отражающий степень схожести.

Такой граф, называемый структурой связей объектов, является входными данными для задачи группировки.

Поскольку граф строится на основе оценки схожести, он может содержать ошибки:

– ложноположительные связи — рёбра между объектами, не являющимися идентичными;
– ложноотрицательные ”связи” — отсутствие рёбер между действительно идентичными объектами.

Структурная согласованность основывается на анализе связей между объектами, где каждая связь имеет конкретное значение или название, при этом сами свойства объектов не учитываются [4]. Такой подход предполагает рассмотрение всей системы как однородного набора взаимосвязанных элементов. Классификация или разделение этой системы на подмножества производится исключительно на основе изучения структуры связей между объектами. Безусловно, подобный метод имеет существенное ограничение, поскольку не учитывает содержательную сторону объектов. Однако важно понимать, что структурная согласованность служит лишь дополнением к семантической согласованности, которая обеспечивает содержательную непротиворечивость системы и напрямую зависит от специфики рассматриваемой предметной области.

Для теоретического анализа удобно ввести идеальную структуру — консонансную [59; 60]. Структура является консонансной, если все тройки объектов в ней находятся в консонансном состоянии. Вводится консонансная функция

$$F_k(i_1, i_2, i_3) = r_{1,2} \wedge (r_{2,3} = r_{1,3}) \vee (\neg r_{12}) \wedge (r_{2,3} \neq r_{1,3})$$

принимающая значения из множества $\{0, 1\}$:

– если $F_k(i_1, i_2, i_3) = 1$, то тройка объектов находится в консонансом состоянии;
– если $F_k(i_1, i_2, i_3) = 0$, то тройка объектов находится в диссонансном состоянии.

В реальных условиях мы имеем ассонансную структуру — граф, содержащий как консонансные тройки, так и диссонансные.

В отличие от структурной согласованности, которая фокусируется исключительно на формальных связях между элементами, семантическая согласованность требует содержательного соответствия между критериями группировки и реальными свойствами объектов. Если структурный подход отвечает на вопрос ”образуют ли объекты связные группы по заданному формальному правилу? то

семантический — ”действительно ли сгруппированные объекты идентичны по существу?”. Ключевая проблема возникает, когда формальные критерии группировки расходятся с содержательной идентичностью объектов. Например, при кластеризации текстовых документов по векторным представлениям структурная согласованность может быть достигнута за счет близости в векторном пространстве, но семантическая согласованность при этом нарушится, если документы с разным смыслом окажутся в одном кластере из-за случайного совпадения терминов. Аналогично, при группировке изображений по формальным признакам (гистограммам цветов) в один кластер могут попасть совершенно разные по содержанию изображения, имеющие схожие цветовые распределения.

Интероперабельность — это способность системы взаимодействовать между компонентами, данными и источниками. В задаче группировки объектов интероперабельность достигается, когда система находится в консонансном состоянии и является семантически согласованной. Только сочетание обоих видов согласованности позволяет создавать интероперабельные системы, где группы объектов сохраняют как формальную целостность, так и смысловую идентичность.

Важно подчеркнуть, что сама по себе структура — это лишь описание связей, полученных на основе текущих данных. Такие свойства, как:

- устойчивость к ошибкам в связях,
- возможность масштабной обработки,
- адаптация к изменению данных,

— относятся не к структуре, а к алгоритмам, которые её обрабатывают. Алгоритм должен быть способен корректно выделять группы даже при наличии умеренного количества ошибок, эффективно масштабироваться на большие графы и поддерживать инкрементальную переработку при изменении состава объектов. Эти аспекты рассматриваются далее в разделе 4.2.

В данном разделе была представлена формальная модель структуры связей между объектами, основанная на графе, порождённом функцией сходства. Такая структура служит основой для следующего этапа — кластеризации объектов в группы по принципу идентичности. Были рассмотрены различия между идеальной (консонансной) и реальной (ассонансной) структурами, возникающими из-за ошибок в оценке схожести. Также была уточнена роль структуры в обеспечении согласованного, интероперабельного представления данных в системе. Переход к графовой модели позволяет учитывать как признаки объектов, так и взаимосвязи

между ними, что особенно важно в условиях динамически изменяющихся данных. Граф схожести объектов выступает как обобщённая, масштабируемая форма представления предварительных связей, пригодная для дальнейшей обработки и кластеризации.

4.2 Сравнительный анализ методов группировки по идентичности

Очевидным методом [61] для объединения товаров в группы является использование принципа транзитивности (рис. 4.1).

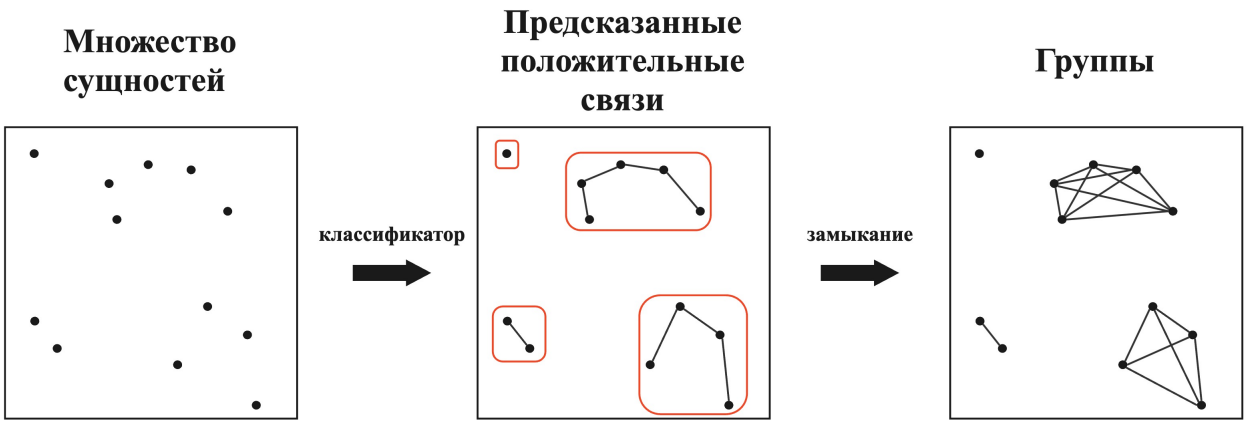


Рисунок 4.1 — Подход на основе транзитивного замыкания.

Суть подхода заключается в том, чтобы добавлять новый товар в уже существующую группу, если он похож хотя бы на один из товаров этой группы. Хотя этот метод достаточно прост в реализации, он вызывает серьёзные проблемы, влияя на структуру групп. Вместо чётко выраженных групп товаров (объекты одной группы имеют связи друг с другом и не имеют связей с объектами других групп), мы часто получаем группы, связанные между собой редкими рёбрами (рис. 4.2). Эти связи некорректны и возникают при ошибках процедуры сравнения двух товаров. Из-за них транзитивный подход приводит к рассогласованию внутри групп.

Задача формирования групп объектов по принципу идентичности в графе сводится к задаче выделения сообществ (community detection). В классическом виде сообщество в графе — это подмножество вершин, обладающее высокой плотностью внутренних связей и относительно малым числом рёбер, ведущих наружу. При этом отсутствует строгое универсальное определение сообщества: на практике оно задаётся алгоритмически — как результат работы метода.

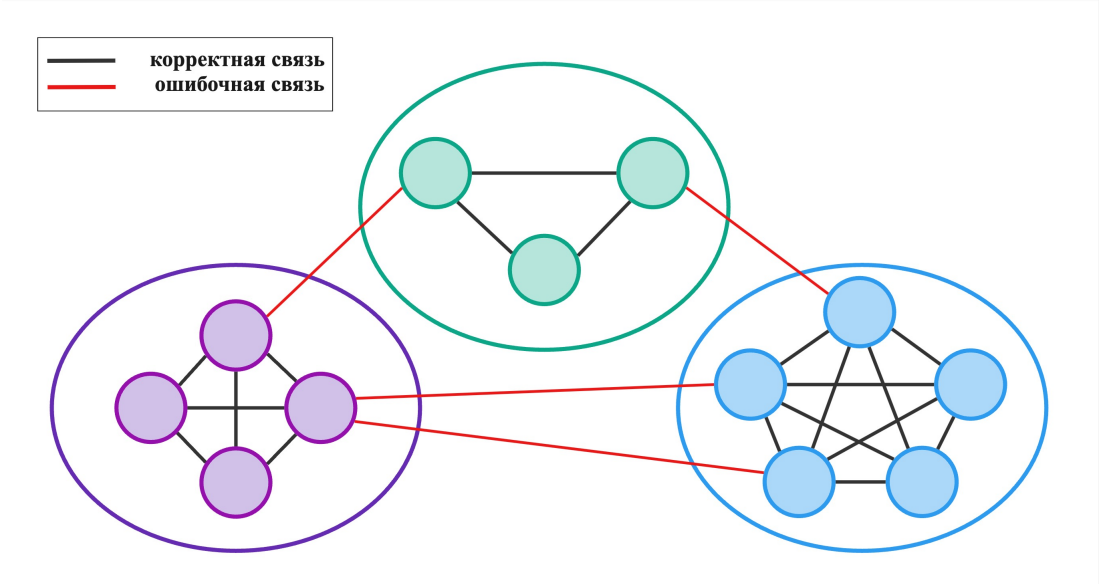


Рисунок 4.2 — Три группы идентичных товаров, ошибочно соединённых между собой малым числом рёбер.

В данном разделе рассматриваются основные подходы к выделению сообществ в графах, проанализированы их достоинства и недостатки с точки зрения специфики рассматриваемой задачи: работа с разреженными графами, высокая динамика структуры, наличие ошибочных связей, отсутствие априорного знания о числе групп.

1. **Методы на основе edge betweenness (Girvan–Newman)** [62]
   Основаны на поэтапном удалении рёбер с наибольшей центральностью (межкластерной нагрузкой). Обеспечивают иерархическое разбиение.
   Плюсы: не требует задавать число кластеров (можно выбрать оптимальное по принципу максимизации модулярности.
   Минусы: низкая устойчивость к ошибочным связям; дорогие пересчёты после каждого удаления; низкая масштабируемость (временная сложность $O(V \cdot E^2)$ или $O(V^3)$).
2. **Модулярностные методы (Modularity-based)** [63]
   Например, Louvain. Стремятся максимизировать модулярность — функционал, сравнивающий плотность внутренних рёбер с ожидаемой при случайной структуре.
   Плюсы: не требует задавать число кластеров; относительно высокая устойчивость к ошибочным связям (благодаря глобальности оптимизации); высокая скорость (временная сложность $O(E)$) (Louvain).
   Минусы: эффект «порога разрешения» — не выявляют малые сообще-

ства; локальные максимумы (нет чётко выраженного глобального максимума); чувствительны к инициализации.

3. **Label Propagation Algorithm (LPA)** [64]
   Алгоритм, в котором каждая вершина принимает метку, наиболее распространённую среди её соседей. Повторяется до сходимости.
   Плюсы: не требует задавать число кластеров; масштабируем; высокая скорость (временная сложность $O(|V| + |E|)$ на 1 итерацию).
   Минусы: нестабильность результатов (зависимость от порядка обновления); возможен «захват» всей связной компоненты одной меткой (но это можно нивелировать за счёт обеспечения низкой доли ошибочных рёбер в графе).
4. **Infomap** [65]
   Использует модель случайного блуждания с минимизацией длины описания маршрута. Оптимизирует map equation.
   Плюсы: высокая точность на графах с ярко выраженными сообществами и относительная устойчивость к ошибочным связям (учёт вероятностей переходов снижает влияние ошибок).
   Минусы: чувствителен к структуре графа (в структурах без явных сообществ может быть неэффективен); относительно низкая масштабируемость (временная сложность $O(ElogV)$).
5. **Walktrap** [66]
   Кластеризация на основе расстояния между распределениями случайных блужданий фиксированной длины. Строит иерархию путём агломерации.
   Плюсы: не требует задавать число кластеров (можно выбрать оптимальное по принципу максимизации модулярности); относительная устойчивость к ошибочным связям (помогает усреднение по случайным блужданиям).
   Минусы: чувствителен к длине блуждания и требует предварительной настройки числа шагов; требует полный граф; низкая масштабируемость (временная сложность $O(V^2logV)$).
6. **Методы на основе спектральной теории** [67]
   Включают разложение Лапласиана или матрицы смежности. Часто применяются как предварительный шаг к кластеризации.

Минусы: чувствительность к числовым погрешностям; сложность интерпретации; низкая масштабируемость (временная сложность от $O(k \cdot E)$ до $O(V^3)$ в зависимости от числа компонент k и разреженности графа).

Таблица 3 — Сравнение алгоритмов кластеризации по ключевым критериям

Метод	**Не требует числа кластеров**	**Масштабируемость**
Girvan–Newman	$\pm$	$-$
Louvain	$+$	$+$
Label Propagation (LPA)	$+$	$+$
Infomap	$+$	$-$
Walktrap	$\pm$	$-$
Спектральные методы	$-$	$-$

На основании анализа можно заключить, что **Label Propagation Algorithm (LPA)** и **Louvain** обладают наиболее подходящими характеристиками для решения задачи динамичного формирования групп идентичных объектов: они не требуют указания числа кластеров, работают локально и масштабируются на большие графы. Кроме того, LPA может быть модифицирован для учёта весов рёбер, а также, как будет показано дальше, может быть эффективно реализован в парадигме MapReduce [68], что делает его особенно подходящим для решения задачи в условиях больших данных.

4.3 Алгоритмы поиска групп

На основе анализа, проведённого в предыдущем разделе, в качестве базового метода кластеризации в данной работе выбран алгоритм распространения меток (Label Propagation Algorithm, LPA). Он обеспечивает масштабируемость, не требует задания числа кластеров и легко адаптируется к графовой структуре входных данных.

Тем не менее, при применении LPA на графах, очищенных от слабых связей (для повышения точности), возникает проблема снижения полноты: часть объектов оказывается изолированной и не попадает в формируемые кластеры. Это критично для задач, где важно максимальное покрытие объектов, в том числе слабо связанных. Для решения данной проблемы предлагается модифицированный двухэтапный подход к кластеризации.

Алгоритм LPA реализует следующий процесс. На начальном этапе каждая вершина графа получает уникальную метку. Затем выполняется итератив-

ный процесс: на каждом шаге вершина обновляет свою метку, выбирая наиболее распространённую метку среди соседних вершин. В случае наличия нескольких меток с одинаковой максимальной частотой, выбор осуществляется случайным образом. Итерации продолжаются до достижения сходимости решения либо до исчерпания заданного максимального числа итераций.

Для формального описания алгоритма (алг. 1) необходимо ввести следующие понятия. Матрица связности A представляет собой симметричную квадратную матрицу, отражающую наличие или отсутствие связей между объектами. Диагональная матрица степеней D содержит на диагонали степени вершин графа, где степень вершины определяется количеством её связей с другими вершинами: $D = \text{diag}(d_1, d_2, \ldots, d_N)$.

Матрица перехода вычисляется как $P = D^{-1}A$. Матрица меток C имеет размерность $N \times n$, где N - количество объектов (вершин графа), а n - число уникальных меток на начальной итерации алгоритма (максимальное количество групп). В рассматриваемом случае принимается $N = n$, что соответствует исходному состоянию, когда каждая вершина графа имеет уникальную метку. Таким образом, без потери общности можно записать:

$$C(0) = \text{diag}(1, 1, \ldots, 1),$$

$$C(t+1) = PC(t).$$

При этом новые метки вершин определяются как $\arg\max_{label} C$.

А л г о р и т м [64].

Вход: G – граф, C_0 – начальные метки.

Выход: $\hat{C}$ – финальные метки.

Ш а г 1. Составить матрицу связности A для графа G.

Ш а г 2. Вычислить $d_i = \sum_j A_{ij}$.

Ш а г 3. Рассчитать $D = \text{diag}(d_1, d_2, \ldots, d_N)$.

Ш а г 4. Вычислить $P = D^{-1}A$.

Ш а г 5. Инициализировать $C(t = 0) = C(0)$.

Ш а г 6. Пока $C(t)$ не сошлось:

$$C(t+1) = PC(t),$$
$$t = t + 1.$$

Ш а г 7. $\hat{C} = C(t)$.

Алгоритм обладает высокой степенью распараллеливания и эффективно реализуется в распределённых вычислительных системах, что обеспечивает его практическую применимость для обработки крупномасштабных графов. Это достигается благодаря локальному характеру вычислений - обновление метки каждой вершины зависит только от её непосредственных соседей. Пример реализации алгоритма в парадигме MapReduce, демонстрирующий его масштабируемость, представлен на рис. 4.3.

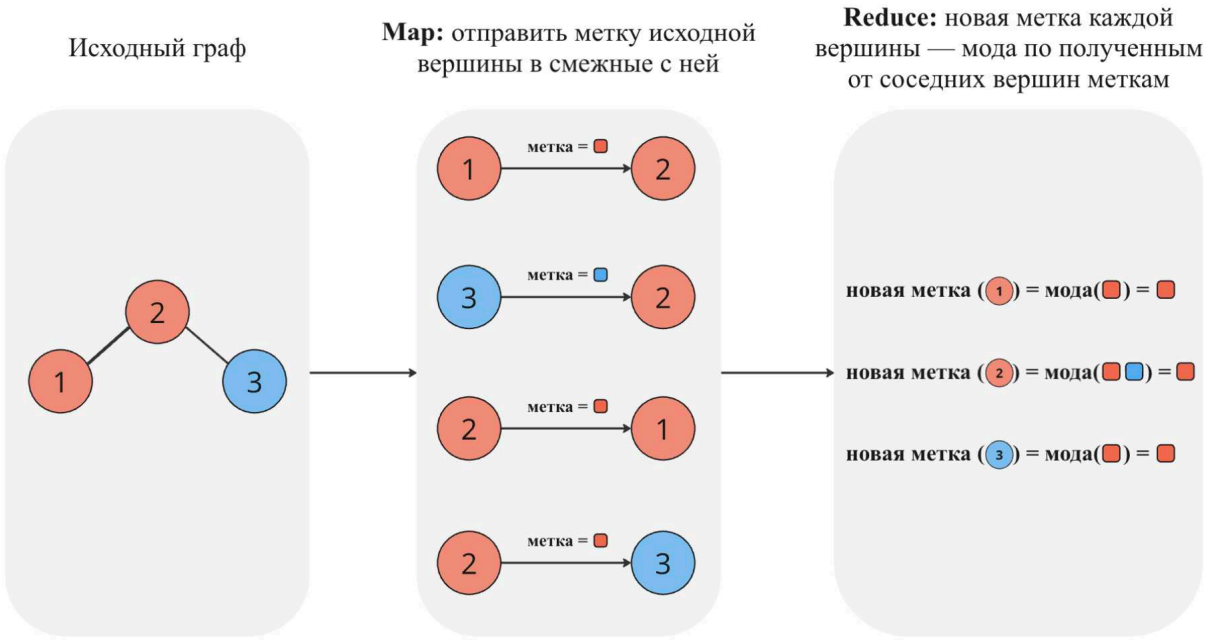


Рисунок 4.3 — Одна итерация LPA в парадигме MapReduce: на первом шаге каждая вершина посылает всем соседям свою метку, на втором шаге каждая вершина меняет свою метку на моду от всех полученных на первом шаге меток.

В рамках данной работы была предложена модификация алгоритма. Предложенная схема состоит из двух логически разделённых этапов:

1. **Первый этап: построение кластеров на сильных связях.** Из исходного графа $G = (V, E, w)$ удаляются все рёбра, для которых значение функции сходства $F(i,j) < \tau_{strong}$, где τ_{strong} — порог, соответствующий высокой степени сходства. На полученном подграфе G_{strong} выполняется стандартный алгоритм LPA. Это позволяет сформировать начальные устойчивые группы с высокой однородностью.
2. **Второй этап: доагрегация и включение слабо связанных объектов.** После построения начального разбиения объекты, не попавшие в класте-

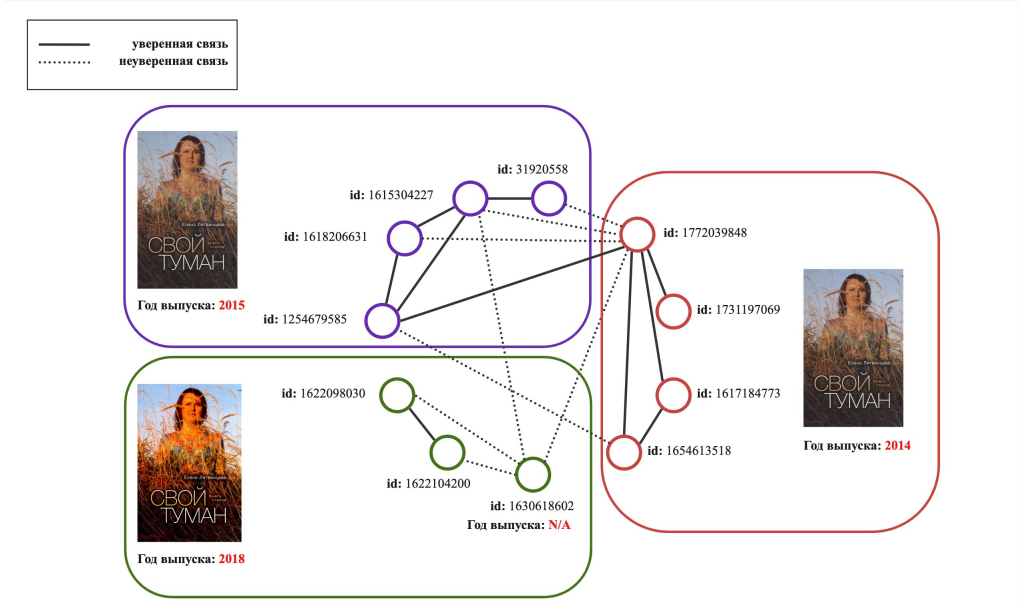


Рисунок 4.4 — Пример работы двухстадийного алгоритма LPA. LPA(95%) упустил бы один из зелёных товаров, что негативно сказалось бы на полноте. LPA(80%) сгруппировал бы все товары, что негативно сказалось бы на однородности.

ры (в силу недостатка сильных связей), анализируются на предмет возможного присоединения к существующим группам. Для этого рассматриваются рёбра с весами в диапазоне $[\tau_{weak}, \tau_{strong}]$. Каждый неподключённый объект присоединяется к той группе, с которой он связан наибольшим общим весом, при условии, что такая связь превышает τ_{weak}.

Таким образом, первый этап обеспечивает высокий уровень однородности кластеров за счёт строгого фильтра, а второй — *полноту* за счёт мягкого дообъединения. Пример работы представлен на рис. 4.4. На первом этапе были образованы следующие группы:

1. Фиолетовые: 1254679585, 1618206631, 1615304227, 31920558.
2. Коричневые: 1772039848, 1731197069, 1617184773, 1654613518.
3. Зелёные: 1622098030, 1622104200.

Эта разбивка корректна с точки зрения однородности: группы отличаются годом выпуска книги, и в каждой группе оказались книги с одинаковыми обложками, названиями и годом выпуска. На втором же этапе книга с id 1630618602 добавилась в зелёную группу, так как с ней у данного товара было больше связей, чем с другими группами (2 связи против 1 с каждой другой группой). У данной книги не указан год выпуска, но оттенок обложки совпадает с оттенком обложки книг 2018

года выпуска, поэтому с учётом доступной информации добавление этой книги в зелёную группу можно считать корректным. Получается, что для этой компоненты связности из 11 товаров транзитивный подход на графе с уверенными рёбрами сгруппировал бы фиолетовые с коричневыми, а на графе с неуверенными рёбрами сгруппировал бы вообще все товары в одну группу. LPA (80%) сделал бы то же, что транзитивный подход на графе с уверенными рёбрами. LPA (95%) справился бы хорошо с точки зрения однородности, но не идеально с точки зрения полноты. 2-stage LPA привёл к наилучшему результату.

Для выбора порогов τ_{strong} и τ_{weak} использовалась размеченная выборка, состоящая из идентичных и неидентичных пар объектов. Основные параметры настройки:

- τ_{strong} выбирается так, чтобы среди пар объектов со значением функции сходства $F(i,j) \geq \tau_{strong}$ доля ложноположительных примеров (false positive) не превышала 5%;
- τ_{weak} выбирается так, чтобы среди пар объектов со значением функции сходства $\tau_{weak} \leq F(i,j) \leq \tau_{strong}$ доля ложноположительных примеров (false negative) не превышала 20%.

Такое разделение диапазона весов рёбер позволяет отделить высоконадежные связи (используемые для построения начальных кластеров) от потенциально полезных, но более шумных, применяемых только для дообъединения.

Модифицированный алгоритм сохраняет ключевые достоинства LPA, при этом даёт более высокие показатели полноты при том же уровне однородности. Его характеристики:

- **Локальность:** оба этапа требуют только информации о соседях вершины;
- **Масштабируемость:** реализация возможна в распределённой среде;
- **Адаптивность:** добавление новых объектов возможно без полного пересчёта;
- **Устойчивость:** итоговая структура слабо чувствительна к отдельным ошибочным рёбрам.

Выводы к четвёртой главе

Предложенная двухэтапная модификация алгоритма распространения меток позволяет добиться высокой полноты кластеризации при сохранении точности. Такой подход особенно актуален в задачах, где исходный граф идентично-

сти зашумлён, а качество модели предсказания идентичности не идеально. Метод применим для обработки больших объёмов данных и интеграции в динамические системы, требующие переоценки связей в реальном времени.

Глава 5. Экспериментальные исследования и анализ результатов

5.1 Методика оценки качества решения

Подходы сравнивались на собранном вручную тестовом экземпляре данных в задаче поиска идентичных товарных предложений. Была выбрана одна категория товаров, состоящая из около 20 тыс. карточек товаров. Из них случайным образом было выбрано 3 тыс. карточек, которые были разбиты на группы вручную. В результате получилось 704 группы, распределение их размеров представлено на рис. 5.1.

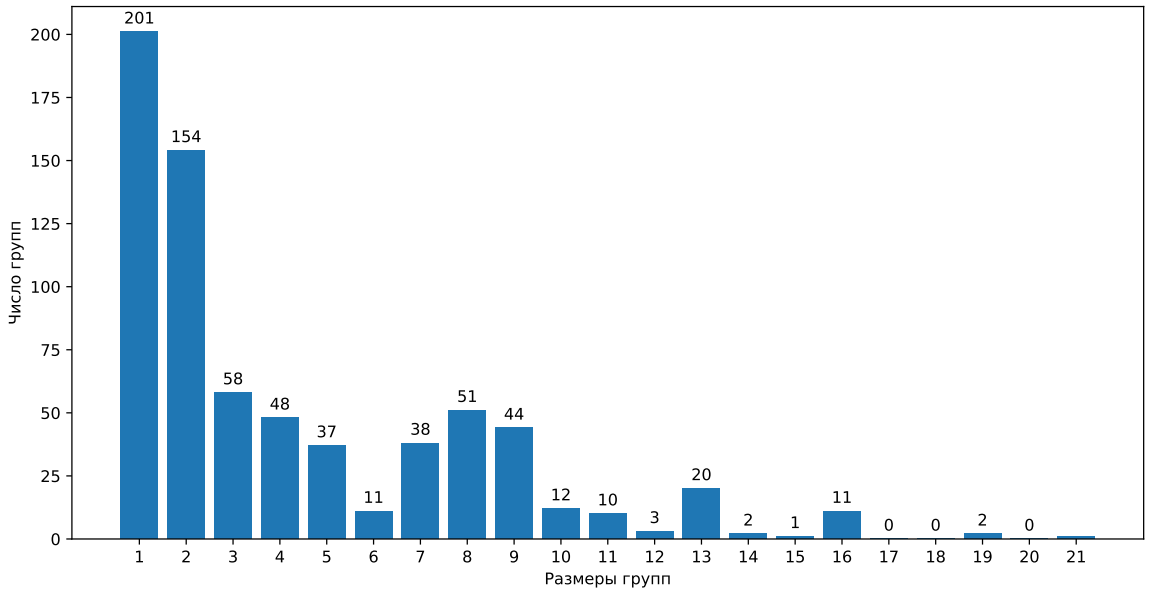


Рисунок 5.1 — Демонстрация различных видов неопределённости в контексте линейной регрессии.

Из постановки задачи у нас есть N карточек товаров, для которых существует $C = \{c_1, c_2, \cdots, c_n\}$ – истинное разбиение на группы идентичных товаров. Пусть мы нашли некоторое свое разбиение $K = \{k_1, k_2, \cdots, k_m\}$. Тогда можно составить матрицу A, элементы $\{a_{c_i k_j}\}$ которой – это количество карточек товаров из истинной группы c_i и определенные нами в группу k_j. Данная матрица используется для вычисления различных численных оценок качества подходов.

– 1. Однородность [9] – схожесть объектов внутри группы:

$$h = \begin{cases} 1, & \text{если } H(C, K) = 0, \\ 1 - \frac{H(C|K)}{H(C)} & \text{иначе,} \end{cases}$$

где

$$H\left(C|K\right) = -\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{a_{c_i k_j}}{N}\log\frac{a_{c_i k_j}}{\sum\limits_{i=1}^{n} a_{c_i k_j}},$$

$$H(C) = -\sum_{i=1}^{n}\frac{\sum\limits_{j=1}^{m} a_{c_i k_j}}{n}\log\frac{\sum\limits_{j=1}^{m} a_{c_i k_j}}{n}.$$

– 2. Полнота [9] – доля идентичных объектов, объединенных в одну группу:

$$c = \begin{cases} 1, & \text{если } H\left(K, C\right) = 0, \\ 1 - \frac{H(K|C)}{H(K)} & \text{иначе,} \end{cases}$$

где

$$H\left(K|C\right) = -\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{a_{c_i k_j}}{N}\log\frac{a_{c_i k_j}}{\sum\limits_{j=1}^{m} a_{c_i k_j}},$$

$$H(K) = -\sum_{j=1}^{m}\frac{\sum\limits_{i=1}^{n} a_{c_i k_j}}{n}\log\frac{\sum\limits_{i=1}^{n} a_{c_i k_j}}{n}.$$

– 3. V-мера [9] – гармоническое среднее между однородностью и полнотой:

$$V_\beta = \frac{(1+\beta)\ h\, c}{\beta\, h + c},$$

где β – параметр: если β больше 1, полнота имеет больший вес в расчетах, если β меньше 1, больший вес имеет однородность.

Второй класс методов оценки качества кластеризации основан на комбинаторном подходе, который исследует количество пар объектов, сгруппированных одинаково в истинном и найденном в качестве решения задачи разбиениях. Другими словами, каждая пара объектов может быть либо 1) сгруппирована вместе в обоих разбиениях (N_{11}), 2) сгруппирована отдельно в обоих разбиениях (N_{00}), 3) сгруппирована вместе в найденном, но не в истинном разбиении (N_{01}), 4) сгруппированы вместе в истинном, но не в найденном разбиении (N_{10}). На основе этих четырех значений можно рассчитать следующие меры качества.

– 4. Оценка Фаулкса-Мэллоуза [10] – геометрическое среднее между однородностью и полнотой (в комбинаторном смысле):

$$FM_{score} = \sqrt{\frac{N_{11}}{N_{11}+N_{01}} \cdot \frac{N_{11}}{N_{11}+N_{10}}} = \frac{N_{11}}{\sqrt{(N_{11}+N_{01}) \cdot (N_{11}+N_{10})}}$$

– 5. Индекс Рэнда с поправкой на случайность [11; 12] – при рассмотрении всех пар объектов – это сумма двух долей: 1) доля пар, попавших в одну группу при условии, что они и должны были попасть в одну группу в силу идентичности, 2) доля пар, не попавших в одну группу при условии, что они и не должны были попасть в одну группу. Поправка на случайность позволяет нивелировать эффект от высокой вероятности получить большое число пар второго типа даже при случайном разбиении:

$$ARI = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\binom{a_{c_i k_j}}{2} - \left[\sum_{i=1}^{n}\binom{x_i}{2}\sum_{j=1}^{m}\binom{y_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_{i=1}^{n}\binom{x_i}{2} + \sum_{j=1}^{m}\binom{y_j}{2}\right] - \left[\sum_{i=1}^{n}\binom{x_i}{2}\sum_{j=1}^{m}\binom{y_j}{2}\right] / \binom{n}{2}},$$

где

$$x_i = \sum_{j=1}^{m} a_{c_i k_j} \ ,$$

$$y_j = \sum_{i=1}^{n} a_{c_i k_j} \ .$$

5.2 Полученные результаты

В данном разделе приведены результаты сравнения работы следующих алгоритмов:

– транзитивное замыкание связных компонент;
– LPA и его модификация, предложенная автором данного исследования;
– Louvain;
– DIANA [69], не являющийся графовым методом кластеризации, но рассмотренный для полноты картины.

Как уже было отмечено выше, алгоритм распространения меток работает не с непрерывными характеристиками (расстояниями), а с бинарными (есть связь /

нет связи). Последние же получаются из первых путем сравнения расстояния с некоторым пороговым значением. Выбор порогового значения позволяет регулировать соотношение числа одних и других, отдавая приоритет либо полноте решения, либо точности. В рассматриваемой задаче исходно доступно несколько графов, соответствующих разным значениям точности решения (80 и 95%).

Результаты экспериментов представлены в таблице. Как уже было сказано выше, все рассмотренные методы в качестве исходных данных используют граф. Если транзитивный подход и LPA могут работать и с разреженным графом, то DIANA требует наличия связей каждого объекта с каждым. Поэтому для разных методов использовались разные исходные графы. Всего таких графов было три.

1. Граф идентичных объектов с точностью 95% – это граф, полученный классическим методом в два этапа: отбор кандидатов, на котором можно потерять пару идентичных объектов в силу не идеальности эвристики, и непосредственно измерение расстояний для отобранных пар-кандидатов с последующим сравнением с пороговым значением. Пороговое значение подобрано таким образом, чтобы доля ошибок составляла 5%. Этот граф использовался как исходные данные для транзитивного метода, а также для LPA (95%) и Louvain (95%).
2. Граф идентичных объектов с точностью 80% – аналогично первому графу, только доля ошибок здесь составляет 20%. В этом графе больше ребер, благодаря чему возможная полнота решения задачи поиска идентичных товаров у него выше, а однородность ниже. Данный граф использовался в эксперименте LPA (80%) и Louvain (80%).
3. Полный граф, каждому ребру которого соответствует значение функции близости между парой товаров. Этот граф был необходим для экспериментов с методом DIANA.

LPA (80%) работает с точки зрения однородности ожидаемо хуже, чем LPA (95%), так как разница заключается в разном количестве рёбер между группами различающихся товаров: слишком большая доля таких рёбер приводит к получению разными группами товаров одинаковой метки, что нарушает однородность, хотя в то же время повышает полноту. 2-stage LPA берёт лучшее от обоих, обеспечивая более высокую полноту при незначительных потерях в однородности. Diana1 благодаря дивизивной природе даёт решение с наивысшей однородностью, однако полнота проигрывает методу LPA. Для удобного сравнения методов

Таблица 4 — Результаты сравнения методов объединения объектов в группы

Метод	Однородность	Полнота	V-мера	Оценка Фаулкса-Мэллоуза	Индекс Рэнда с поправкой на случайность
Транзитивный подход (95%)	0.685	0.942	0.793	0.720	0.696
LPA (95%)	0.992	0.916	0.952	0.872	0.863
LPA (80%)	0.744	**0.973**	0.843	0.755	0.731
2-стадийный LPA	0.986	0.945	**0.965**	**0.882**	**0.871**
Louvain (95%)	0.993	0.902	0.945	0.868	0.855
Louvain (80%)	0.783	0.965	0.865	0.773	0.768
$DIANA_1$	**0.995**	0.879	0.933	0.840	0.819
$DIANA_2$	0.991	0.810	0.891	0.774	0.756

в таблице также приведена V-мера, позволяющая комплексно оценивать и однородность, и полноту. По V-мере именно 2-stage LPA является лидером среди всех рассмотренных методов. Учитывая отсутствие повышенных требований к графу в виде необходимости наличия рёбер между каждой парой товаров, LPA также является более удобным на практике. Более того, LPA хорошо ложится на модель распределённых вычислений MapReduce, что позволяет ему иметь эффективную реализацию, решающую задачу с сотнями миллионов объектов за несколько часов. Интересным результатом является снижение однородности LPA (80%) в сравнении с точностью исходных данных с 80% до 74.4%, и рост однородности LPA (95%) соответственно с 95% до 99.2%. Это объясняется сутью алгоритма — если ошибочных рёбер между разными группами товаров мало (например, 5% от всех рёбер), LPA «уберёт» часть из них. Если же их достаточно много (например, 20%), LPA не справится с задачей разделения и оставит рёбра, а после объединения нескольких групп разных товаров добавятся дополнительные ошибочные рёбра между остальными товарами разных групп, что повысит долю таких рёбер. Louvain (95%) показывает однородность 99.3%, что практически совпадает с результатом LPA (95%) и объясняется тем, что оба алгоритма работают на одном и том же графе, построенном по высокому порогу. Полнота у Louvain (95%) немного ниже — 90.2% против 91.6% у LPA (95%). Это связано с особенностями самого метода: он принимает решение о присоединении объекта к кластеру, оценивая, приведёт ли это к росту модулярности. Если добавление объекта в какой-либо из соседних кластеров не даёт выигрыша по целевой функции, объект может остаться изолированным. В отличие от LPA, который даже при слабом, но локально пре-

обладающем сигнале передаёт метку, Louvain действует более консервативно и избегает включения объектов с неочевидной принадлежностью. Это и объясняет умеренное снижение полноты при практически той же однородности. В варианте Louvain (80%), где граф содержит больше рёбер, в том числе ошибочных, наблюдается закономерное снижение однородности до 78.3% и рост полноты до 96.5%. Увеличение количества слабых связей в графе приводит к тому, что Louvain начинает объединять области, которые ранее были разделены. Поскольку алгоритм полностью опирается на структуру графа — то есть на набор и конфигурацию рёбер между объектами — и не учитывает степени достоверности этих связей, его устойчивость к шуму оказывается ограниченной. Это проявляется в том, что даже относительно слабые связи, если они соединяют ранее раздельные компоненты, могут повлиять на агрегацию кластеров и привести к их частичному слиянию. Тем не менее Louvain (80%) лучше справляется с ростом доли ошибочных связей, чем LPA (80%), что согласуется с теорией. Алгоритм DIANA на обоих вариантах демонстрирует очень высокую однородность (99.5% и 99.1%), что отражает аккуратный характер дивизивной кластеризации. Он начинает с полной группы и пошагово делит её, ориентируясь на уменьшение средних внутригрупповых расстояний, разбивая изначальный кластер на всё более однородные подгруппы. Такой подход хорошо работает в части избежания включения заведомо разнородных объектов. Однако это приводит к заметным потерям полноты: 87.9% у DIANA_1 и 81.0% у DIANA_2. Различие между ними объясняется способом вычисления меры схожести: оба алгоритма работают на одном и том же графе, но оценивают расстояние между объектами по-разному, что влияет на последовательность разбиений и на то, какие группы оказываются разделёнными. DIANA, в отличие от графовых методов, не использует структуру связей напрямую — она ориентируется на глобальные признаки различия между объектами, что делает её чувствительной к выбору метрики. Кроме того, алгоритм требует матрицу всех попарных расстояний, что ограничивает его применимость задачами умеренного масштаба и делает его мало пригодным для работы с большими графами.

5.3 Методика оценки качества группировки объектов в реальных задачах

В условиях ежедневной обработки сотен миллионов объектов, с миллионами обновлений, добавлений и удалений, критически важно постоянно контролировать качество работы системы динамического формирования групп идентич-

ных объектов. Бизнес-логика напрямую зависит от точности этих групп: неверная группировка ведет к искажению аналитики, некорректной работе и финансовым потерям. Однако практический мониторинг качества на таких масштабах и при такой динамике упирается в фундаментальные ограничения измерения ключевых мер качества – полноты и однородности.

Главная проблема – невозможность прямого измерения полноты. Эта мера качества требует сравнения полученной группировки с полной эталонной разметкой всех объектов. Но сама необходимость автоматической группировки возникает именно потому, что создание такого эталона для всего массива данных принципиально нереализуемо из-за его объема и постоянных изменений. Объекты не статичны: ежедневные обновления характеристик могут сделать любую, даже гипотетически созданную эталонную разметку, устаревшей. Без полного и актуального ”ground truth” оценка того, какая доля всех существующих групп идентичных объектов была найдена системой, превращается в неразрешимую задачу.

Однородность является более доступной мерой качества, так как теоретически ее можно оценить внутри уже сформированных групп, не требуя глобального эталона. Однако и здесь возникают практические барьеры, делающие точную оценку крайне затруднительной. Чтобы подтвердить, что все объекты внутри группы действительно идентичны, необходимо проверить каждую возможную пару объектов в этой группе на соответствие критериям идентичности. Для группы размером k это означает проверку $k \cdot (k-1)/2$ пар. Для относительно небольшой группы из 10 объектов это уже 45 пар. В реальности группы могут достигать размеров в десятки или даже сотни объектов, а количество групп исчисляется миллионами. Даже если проверять не все группы, а лишь репрезентативную выборку, объем работы по ручной или экспертной верификации внутри каждой выбранной группы становится неподъемным для ежедневного контроля. Необходимость проверки огромного числа пар внутри групп делает попытку точного измерения однородности на всем массиве данных или даже на значительной выборке практически неосуществимой задачей в условиях ограниченных ресурсов и требований к оперативности.

Таким образом, стандартные подходы к оценке качества группировки – через прямой расчёт однородности и полноты – наталкиваются на непреодолимые практические препятствия в контексте больших, динамически обновляющихся данных. Возникает острая потребность в разработке альтернативных, практиче-

ски реализуемых и оперативных методик контроля качества, которые могли бы давать значимую оценку работы системы ежедневно, не требуя невозможного – полной эталонной разметки или полной проверки миллионов пар объектов внутри групп.

5.3.1 Приближённая оценка однородности

Стандартным подходом по оценке какого-либо параметра в больших данных является расчёт этого параметра по выборке, сделанной из этих данных. Семплировать случайные группы и проверять их однородность не получится – здесь мы сталкиваемся с необходимостью проверки всевозможных пар объектов внутри каждой выбранной группы, а число пар квадратично по числу объектов в ней. Это обстоятельство подталкивает к необходимости семплировать не группы, а сами пары. Простейшим подходом является формирование в каждой группе всевозможных пар, объединение их в общее множество пар и семплирование из него. Но оценки, полученные таким методом, могут плохо отражать качество пользовательского опыта. Пользователь работает не с парами объектов, а с самими объектами, поэтому вероятность встретить неоднородную группу скорее пропорциональна числу объектов в этой группе, а не числу пар объектов. Анализ распределения размеров групп в системе (рис. 5.2) выявляет ключевую особенность, определяющую методику оценки: хотя подавляющее большинство групп являются малыми (размером 2-10 объектов), именно редкие крупные группы (размером >10 объектов) содержат значительную долю всех возможных пар объектов в системе. Во-первых, проверка всех пар объектов является вычислительно неоправданной, особенно для крупных групп, так как количество пар растёт квадратично. Во-вторых, простое равномерное семплирование пар по всей системе приведёт к чрезмерному представительству пар из крупных групп в итоговой выборке, что не соответствует реальной пользовательской нагрузке, где преобладает взаимодействие с малыми и средними группами.

Предлагаемый метод двухэтапного семплирования решает эту проблему. На первом этапе из каждой группы извлекается $k-1$ пар, что обеспечивает линейную, а не квадратичную зависимость числа примеров от размера группы. Такой подход сохраняет баланс: малые группы, которых много, вносят небольшое количество пар каждая, но в совокупности составляют значительную часть выборки; крупные группы, которых мало, вносят больше пар на группу, но их общий вклад остается

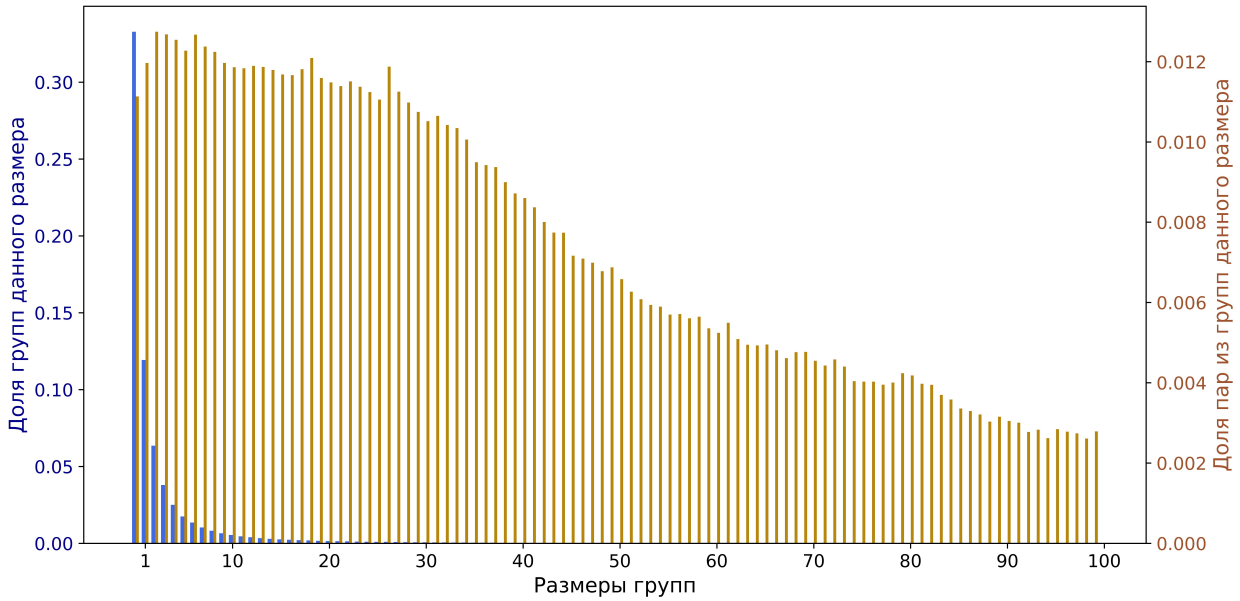


Рисунок 5.2 — Распределение числа групп по размерам (синий) и числа пар объектов в этих группах (бежевый). График ограничен по оси X числом 100.

пропорциональным их встречаемости в системе. Второй этап – семплирование из этого предварительно сбалансированного множества – позволяет получить итоговую выборку, где вероятность включения пары соответствует вероятности встречи пользователя с подобной парой в реальной работе с системой.

Важно отметить, что выбор именно $k - 1$ пар на первом этапе является оптимальным компромиссом между полнотой охвата и вычислительной эффективностью. Это количество достаточно для выявления грубых нарушений однородности (если группа неоднородна, высока вероятность обнаружить это даже при таком семплировании), но при этом не создает непомерной вычислительной нагрузки даже для крупных групп. Применение более агрессивного семплирования (например, фиксированного количества пар на группу) привело бы к потере чувствительности к особенностям крупных групп, в то время как менее интенсивное семплирование не обеспечило бы достаточной репрезентативности для малых групп.

5.3.2 Оценка покрытия как альтернатива полноте

Оценка полноты системы представляет собой методологически сложную задачу, существенно превосходящую по нетривиальности даже проблему измерения однородности. В практических реалиях больших данных полнота может быть операционализирована через концепцию покрытия (coverage), определяемого как

доля объектов системы, для которых алгоритмом идентифицирована хотя бы одна содержательно значимая группа (т.е. группа размером ≥ 2). Ключевое преимущество данной метрики — её полная автоматизируемость и независимость от ручной верификации, поскольку факт принадлежности объекта к какой-либо группе фиксируется на этапе выполнения алгоритма. Фундаментальная ценность покрытия заключается в его бизнес-релевантности: наличие у объекта хотя бы одного идентичного партнёра в группе создаёт основу для информационного обмена, взаимодополняющего анализа данных и синергетических эффектов. Однако интерпретация этой метрики требует учёта принципиального ограничения: не для всех объектов в системе существуют идентичные экземпляры. Следовательно, теоретический максимум покрытия заведомо меньше 100%, а стремление алгоритма к абсолютному значению является методологической ошибкой.

Для оценки максимально достижимого покрытия (МДП) предлагается следующий научно обоснованный подход:

1. Формирование репрезентативной выборки из объектов, не включённых алгоритмом в группы.
2. Экспертная верификация силами асессоров, задачей которых является ручной поиск потенциально идентичных объектов во всей системе.
3. Расчёт доли принципиально уникальных объектов в выборке.
4. Экстраполяция на генеральную совокупность.

Важно подчеркнуть, что МДП является динамическим параметром системы. Изменение состава данных (появление новых объектов, изменение характеристик) требует регулярного оценивания.

5.4 Примеры работы алгоритмов и примеры использования полученных групп

Теоретические выкладки и алгоритмические решения, представленные в предыдущих разделах, находят своё непосредственное применение в реальных системах электронной коммерции. В данном разделе представлена демонстрация работоспособности описанного подхода, иллюстрирующая ключевые сценарии использования:

– Устранение дублирования товаров в выдаче поисковых результатов.
– Формирование объединенных карточек товаров.

– Плашка ”Есть быстрее” , ”Есть дешевле” и ”Самое выгодное предложение”.
– Предложение такого же товара в разделе ”Избранное” , если добавленный ранее товар закончился на складе.

Представленные примеры показывают, как результаты работы подхода на реальных данных (рис. 5.3, рис. 5.4), так и бизнес-ценность получаемых результатов – улучшение пользовательского опыта и ключевых метрик платформы.

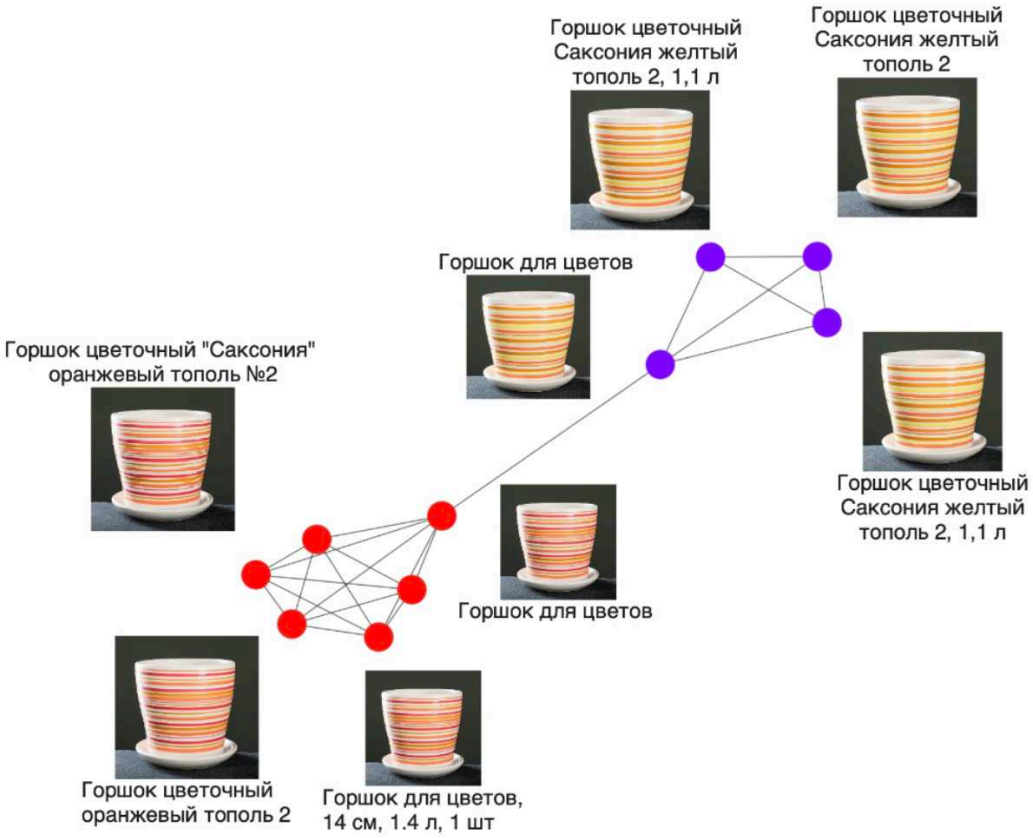


Рисунок 5.3 — Пример работы двухстадийного алгоритма LPA на реальных данных. Цветочные горшки разных оттенков попали в разные группы.

На рисунке 5.5 представлен пример с предложениями выбранного товара от других продавцов. У каждого предложения своя цена и сроки доставки, а также разный рейтинг продавца. Покупатель может выбрать любое предложение, понимая, что в любом случае получит нужный товар.

Похожий пример можно увидеть на рисунке 5.6. Здесь помимо блока с предложениями других продавцов покупатель видит подсказку, что этот же товар можно купить по более выгодной цене у другого продавца. Кликнув на неё, покупатель попадёт на страницу самого выгодного предложения. В случае отсутствия

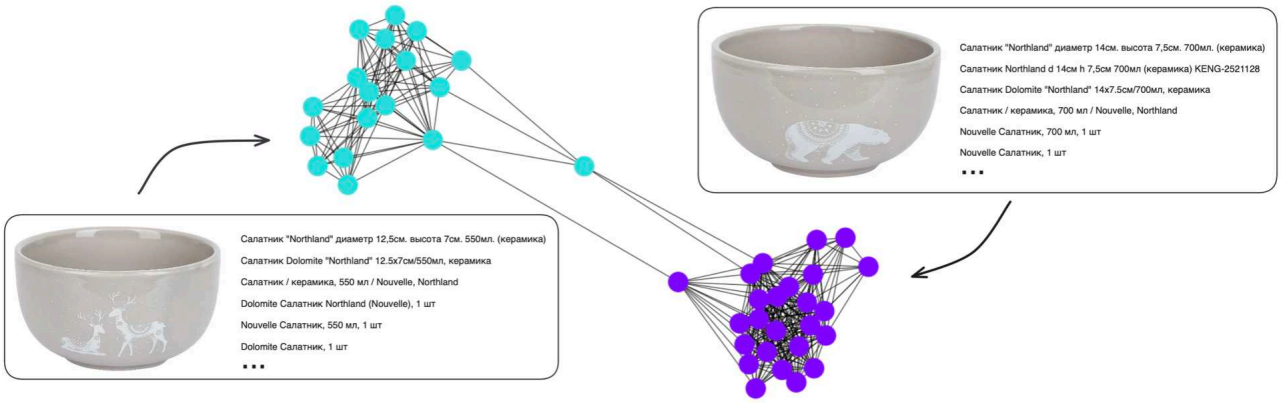


Рисунок 5.4 — Пример работы двухстадийного алгоритма LPA на реальных данных. Тарелки для салата с разными узорами попали в разные группы.

системы поиска групп идентичных товаров покупателю пришлось бы искать более дешёвые предложения вручную.

На рисунке 5.7 представлен пример товарного предложения с подсказкой о том, что это достаточно выгодное предложение на маркетплейсе. Это возможно благодаря системе формирования групп идентичных товаров. Система сформировала группу с этим товаром, благодаря чему стало понятно, что другие такие же товары, то есть товары в сформированной группе, стоят дороже.

Помимо прочего, наличие групп идентичных товаров позволяет делать подсказку о том, что у другого продавца можно купить такой же товар с более быстрой доставкой (рис. 5.8). Это полезно для тех покупателей, которым важна скорость доставки.

И наконец, на рисунке 5.9 показан пример ещё одной бизнес-механики. Покупатель может добавить товар в избранное, чтобы купить его позже. К моменту, когда покупатель будет готов совершить покупку, помеченное им товарное предложение может закончиться на складе. Наличие групп идентичных товаров позволяет показать ему такой же товар от другого продавца рядом с помеченным им товарным предложением в разделе ”Избранное”. Это помогает пользователю совершить покупку без дополнительных действий.

Также есть ряд других сценариев использования групп идентичных товаров, которые сложно проиллюстрировать рисунками, но которые оказывают не меньшее влияние на эффективность маркетплейса.

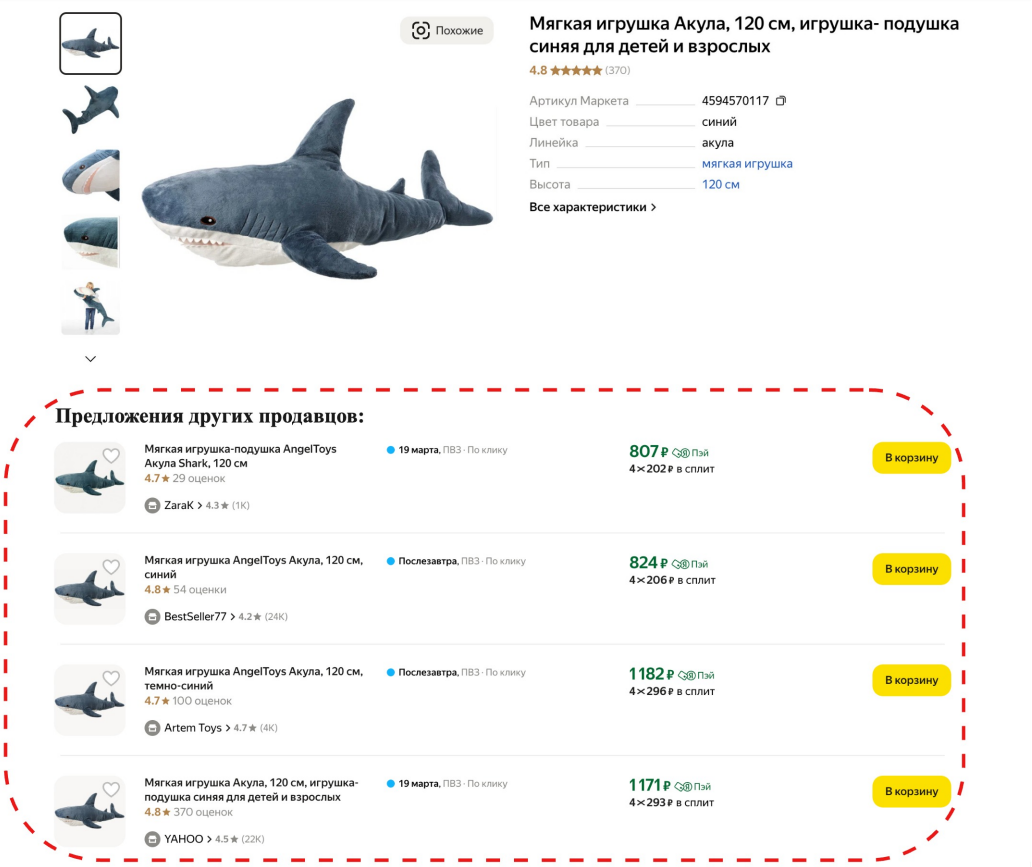


Рисунок 5.5 — Блок с предложениями других продавцов.

Выводы к пятой главе

В ходе экспериментальных исследований были проанализированы различные методы кластеризации для задачи поиска идентичных товарных предложений. Сравнение алгоритмов, включая транзитивное замыкание, LPA, его модификацию, Louvain и DIANA, показало, что двухстадийный LPA демонстрирует наилучшие результаты по комплексной мере качества — V-мера, сочетая высокую однородность и полноту. Этот метод также оказался наиболее практичным, так как не требует полного графа связей и эффективно работает в распределённых системах. Louvain показал схожие с LPA результаты по однородности, но немного уступил в полноте из-за своей консервативности. DIANA, несмотря на высокую однородность, значительно проигрывает в полноте и требует больших вычислительных ресурсов, что ограничивает его применение в крупномасштабных задачах.

Для оценки качества группировки в реальных условиях были предложены методики, учитывающие ограничения больших данных. Приближённая оценка однородности через двухэтапное семплирование пар объектов позволила сба-

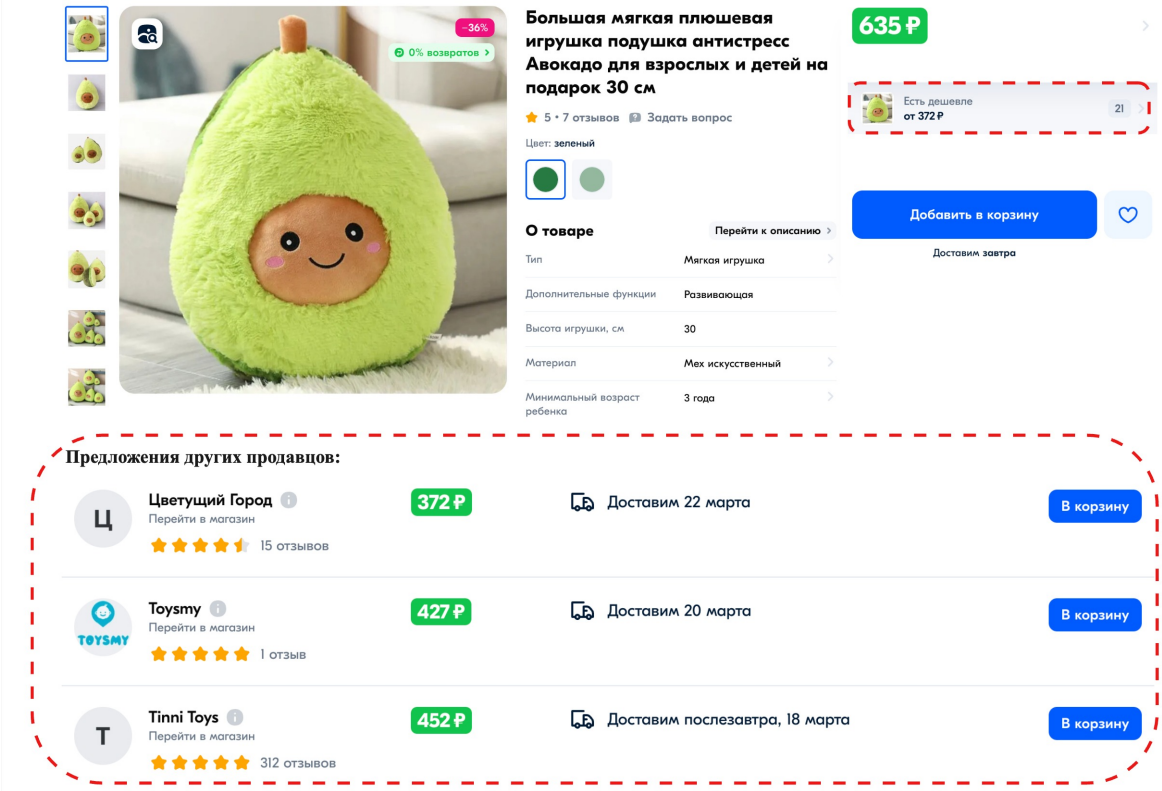


Рисунок 5.6 — Блок с предложениями других продавцов и подсказка ”Есть дешевле”.

лансировать точность и вычислительную эффективность. Вместо полноты была введена мера качества ”покрытие” . Эти подходы позволяют контролировать качество системы без необходимости полной эталонной разметки, что критически важно в условиях динамически обновляющихся данных.

Практическое применение алгоритмов продемонстрировало их ценность для электронной коммерции. Формирование групп идентичных товаров улучшает пользовательский опыт, устраняя дублирование в поисковой выдаче, предлагая альтернативные варианты покупки и обеспечивая подсказки о более выгодных или быстрых предложениях. Это не только повышает удобство для покупателей, но и способствует увеличению ключевых метрик платформы. Таким образом, предложенные методы и алгоритмы доказали свою эффективность как в теоретическом, так и в практическом аспектах, обеспечивая решение актуальных задач в условиях больших и динамичных данных.

Результаты данной главы опубликованы в работах [70—72].

Рисунок 5.7 — Подсказка о том, что данное товарное предложение выбранного товара выгоднее других.

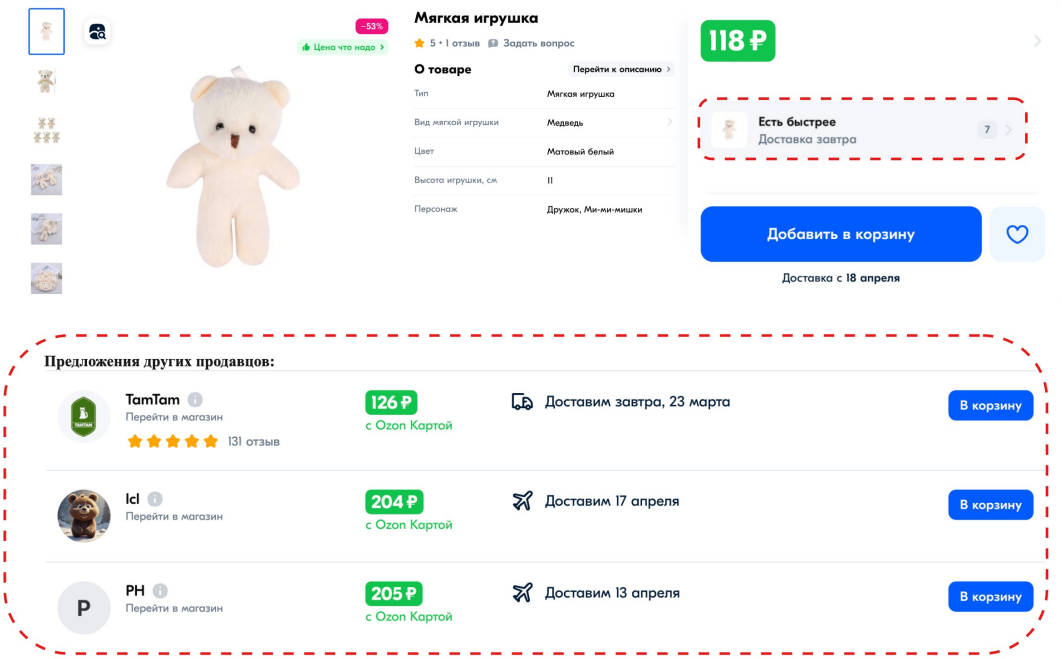


Рисунок 5.8 — Подсказка о том, что этот же товар можно купить у другого продавца с более быстрой доставкой.

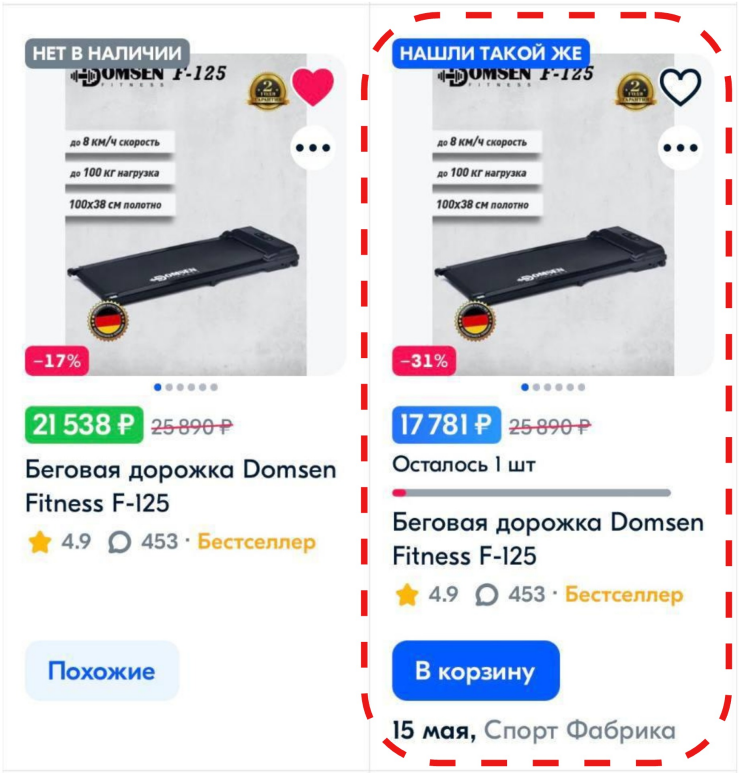


Рисунок 5.9 — Блок ”Нашли такой же” товар от другого продавца в случае, когда исходное товарное предложение закончилось на складе.

Заключение

Основные результаты работы заключаются в следующем.

1. Изучены подходы к оптимизации аналитических SQL-запросов с использованием методов машинного обучения, который включает модели предсказания кардинальности и нейросетевые аппроксиматоры функции стоимости выполнения. Показано, что при условии значительных доработок предложенных в литературе методов, с помощью некоторых из них можно достичь ускорения выполнения сложных запросов в лабораторных условиях. Однако на практике в условиях динамично изменяющихся данных возникают проблемы устойчивости моделей к изменениям и высоких накладных расходов на их переобучение в условиях реального времени.
2. Разработан метод идентификации семантически идентичных объектов, основанный на комбинации бинарной классификации пар объектов и алгоритма кластеризации на графах. Произведён теоретический анализ нескольких алгоритмов выделения сообществ, а также сравнение качества алгоритмов по ключевым мерам качества (однородность и полнота) на реальных данных. Алгоритм распространения меток (LPA) выбран в качестве наиболее подходящего по ряду характеристик, а также по качеству работы. Предложенная архитектура адаптирована для распределённых систем с использованием парадигмы MapReduce, что обеспечивает масштабируемость и практическую применимость подхода в широком спектре приложений.
3. Разработан модифицированный двухэтапный алгоритм кластеризации на основе LPA, в котором реализована калибровка порогов идентичности с учётом допустимого уровня ошибок (5% и 20%). Показано, что данный алгоритм позволяет повысить полноту кластеризации без потери однородности, обеспечивая структурную согласованность групп. Предложенный метод апробирован на задаче поиска идентичных товаров на маркетплейсе. Предоставлены результаты работы системы.
4. Предложена и апробирована методика оценки качества группировки объектов, включающая показатели однородности и полноты, а также дополнительный показатель покрытия как альтернативу полноте в условиях отсутствия эталонной разметки. Эта методика адаптирована для анали-

за результатов динамической кластеризации в системах обработки больших данных и продемонстрировала свою пригодность при проведении экспериментальных исследований.

Список литературы

1. GOST R 55062-2012. Information Technology (IT). Industrial Automation Systems and Their Integration. Interoperability. Basic Provisions. — 2014. — Standartinform.
2. *Creps R.*, *Polzer H.*, *Yanosy J.* Systems, Capabilities, Operations, Programs, and Enterprises (SCOPE). Model for Interoperability Assessment : тех. отч. / Network-Centric Operations Industry Consortium. — 2008. — С. 154.
3. *Rosenberg I.*, *Dulin S.*, *Dulina N.* Modeling the Structure of Interoperability by Means of Structural Consistency // Computer Science and its Applications. — 2023. — Т. 17. — С. 57—65.
4. *Dulin S.* Introduction to the Theory of Structural Coherence. — Moscow : Computing Center of the Russian Academy of Sciences, 2005. — С. 135.
5. *Dunn H. L.* Record linkage // American Journal of Public Health and the Nations Health. — 1946. — Т. 36, № 12. — С. 1412—1416.
6. *Fellegi I. P.*, *Sunter A. B.* A theory for record linkage // Journal of the American statistical association. — 1969. — Т. 64, № 328. — С. 1183—1210.
7. *Fortunato S.* Community detection in graphs // Physics reports. — 2010. — Т. 486, № 3—5. — С. 75—174.
8. *Harary F.* On the notion of balance of a signed graph. // Michigan Mathematical Journal. — 1953. — Т. 2, № 2. — С. 143—146.
9. *Rosenberg A.*, *Hirschberg J.* V-Measure: A Conditional Entropy-based External Cluster Evaluation Measure. — 2007.
10. *Fowkles E.*, *Mallows C.* A Method for Comparing Two Hierarchical Clusterings // Journal of the American Statistical Association. — 1983. — Т. 78. — С. 553—569.
11. *Rand W.* Objective Criteria for the Evaluation of Clustering Methods // Journal of the American Statistical Association. — 1971. — Т. 66. — С. 846—850.
12. *Hubert L.*, *Arabie P.* Comparing Partitions // Journal of Classification. — 1985. — Т. 2. — С. 193—218.
13. *Manning C. D.* An introduction to information retrieval. — 2009.

14. Blocking and filtering techniques for entity resolution: A survey / G. Papadakis [и др.] // ACM Computing Surveys (CSUR). — 2020. — Т. 53, № 2. — С. 1—42.

15. *Дулин С.*, *Рябцев А.* Алгоритм улучшения согласованности структурной интероперабельности // Надёжность. — 2024. — Т. 24, № 2. — С. 8—16.

16. *Miao Z.*, *Li Y.*, *Wang X.* Rotom: A Meta-learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond // Proc. Intern. Conf. on Management of Data. — Xi'an, 2021. — С. 1303—1316.

17. Deep Learning for Blocking in Entity Matching: a Design Space Exploration / S. Thirumuruganathan, H. Li, N. Tang [и др.] // Proc. VLDB Endowment. — 2021. — Т. 14. — С. 2459—2472.

18. *Silberschatz A.*, *Korth H. F.*, *Sudarshan S.* Database system concepts. Т. 5. — McGraw-Hill New York, 2002.

19. *Dittrich J.* Architecture and implementation of database systems.

20. *Harmouch H.*, *Naumann F.* Cardinality estimation: An experimental survey // Proceedings of the VLDB Endowment. — 2017. — Т. 11, № 4. — С. 499—512.

21. *Han Y. e. a.* Cardinality Estimation in DBMS: A Comprehensive Benchmark Evaluation // arXiv preprint arXiv:2109.05877. — 2021.

22. Deep Unsupervised Cardinality Estimation / Z. Yang [и др.] // Proceedings of the VLDB Endowment. — 2019. — Т. 13, № 3.

23. *Cai W.*, *Balazinska M.*, *Suciu D.* Pessimistic cardinality estimation: Tighter upper bounds for intermediate join cardinalities // Proceedings of the 2019 International Conference on Management of Data. — 2019. — С. 18—35.

24. *Bruno N.*, *Chaudhuri S.*, *Gravano L.* STHoles: A multidimensional workload-aware histogram // Proceedings of the 2001 ACM SIGMOD international conference on Management of data. — 2001. — С. 211—222.

25. *Deshpande A.*, *Garofalakis M.*, *Rastogi R.* Independence is good: Dependency-based histogram synopses for high-dimensional data // ACM SIGMOD Record. — 2001. — Т. 30, № 2. — С. 199—210.

26. *Gunopulos D. e. a.* Selectivity estimators for multidimensional range queries over real attributes // the VLDB Journal. — 2005. — Т. 14, № 2. — С. 137—154.

27. *Muralikrishna M.*, *DeWitt D. J.* Equi-depth multidimensional histograms // Proceedings of the 1988 ACM SIGMOD international conference on Management of data. — 1988. — С. 28—36.

28. *Selinger P. G. e. a.* Access path selection in a relational database management system // Readings in Artificial Intelligence and Databases. — Elsevier, 1989. — С. 511—522.

29. *Wang H.*, *Sevcik K. C.* A multi-dimensional histogram for selectivity estimation and fast approximate query answering // Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research. — 2003. — С. 328—342.

30. *Documentation P.* 12. 2020. Chapter 70.1. Row Estimation Examples. — 2020.

31. *Lopes P.*, *Guyer C.*, *Gene M.* Sql docs: cardinality estimation (SQL Server). — 2019.

32. *Heimel M.*, *Kiefer M.*, *Markl V.* Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. — 2015. — С. 1477—1492.

33. *Kiefer M. e. a.* Estimating join selectivities using bandwidth-optimized kernel density models // Proceedings of the VLDB Endowment. — 2017. — Т. 10, № 13. — С. 2085—2096.

34. *Leis V. e. a.* Cardinality Estimation Done Right: Index-Based Join Sampling. // Cidr. — 2017.

35. *Li F. e. a.* Wander join: Online aggregation via random walks // Proceedings of the 2016 International Conference on Management of Data. — 2016. — С. 615—629.

36. *Zhao Z. e. a.* Random sampling over joins revisited // Proceedings of the 2018 International Conference on Management of Data. — 2018. — С. 1525—1539.

37. *Documentation M. S.* Statistics for optimizing queries: InnoDB persistent statistics. — 2020.

38. *Krishnan S. e. a.* Learning to optimize join queries with deep reinforcement learning // arXiv preprint arXiv:1808.03196. — 2018.

39. *Marcus R.*, *Papaemmanouil O.* Deep reinforcement learning for join order enumeration // Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. — 2018. — С. 1—4.

40. *Trummer I. e. a.* Skinnerdb: Regret-bounded query evaluation via reinforcement learning // ACM Transactions on Database Systems (TODS). — 2021. — Т. 46, № 3. — С. 1—45.

41. *Marcus R. a. a.* Neo: A Learned Query Optimizer // Proceedings of the VLDB Endowment. — 2021. — Т. 12, № 11.

42. *Ivanov O.*, *Bartunov S.* Adaptive query optimization in PostgreSQL // PGCon 2017 Conference, Ottawa, Canada. — 2017.

43. *Yang Z. e. a.* NeuroCard: one cardinality estimator for all tables // Proceedings of the VLDB Endowment. — 2020. — Т. 14, № 1. — С. 61—73.

44. *Zhu R. e. a.* FLAT: fast, lightweight and accurate method for cardinality estimation // Proceedings of the VLDB Endowment. — 2021. — Т. 14, № 9. — С. 1489—1502.

45. *Woltmann L. e. a.* Cardinality estimation with local deep learning models // Proceedings of the second international workshop on exploiting artificial intelligence techniques for data management. — 2019. — С. 1—8.

46. *Leis V. e. a.* How good are query optimizers, really? // Proceedings of the VLDB Endowment. — 2015. — Т. 9, № 3. — С. 204—215.

47. *He K. e. a.* Delving deep into rectifiers: Surpassing human-level performance on imagenet classification // Proceedings of the IEEE international conference on computer vision. — 2015. — С. 1026—1034.

48. *Gasnikov A.* Modern numerical optimization methods. Universal Gradient Descent Method // e-print. arXiv:1711.00394. — 2018.

49. *Xu J. e. a.* Understanding and improving layer normalization // Advances in Neural Information Processing Systems. — 2019. — Т. 32.

50. *Hasselt H.* Double Q-learning // Advances in neural information processing systems. — 2010. — Т. 23.

51. *Cui Y. e. a.* Kernel pooling for convolutional neural networks // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2017. — С. 2921—2930.

52. *Vaswani A. e. a.* Attention is all you need // Advances in neural information processing systems. — 2017. — Т. 30.

53. *Hüllermeier E.*, *Waegeman W.* Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods // Machine Learning. — 2021. — Т. 110, № 3. — С. 457—506.

54. *Srivastava N. e. a.* Dropout: a simple way to prevent neural networks from overfitting //. Т. 15. — JMLR. org, 2014. — С. 1929—1958.

55. *Gal Y.*, *Ghahramani Z.* Dropout as a bayesian approximation: Representing model uncertainty in deep learning // international conference on machine learning. — PMLR. 2016. — С. 1050—1059.

56. *Дулин С.*, *Рябцев А.* Анализ подходов к оптимизации запросов в аналитических СУБД // Образовательные ресурсы и технологии. — 2023. — № 3. — С. 73—80.

57. *Дулин С.*, *Рябцев А.* Оценка планов выполнения SQL запросов для решения транспортных задач // Сетевой научно-методический журнал «Наука и технологии железных дорог», АО «НИИАС». — 2023. — Т. 7, № 1. — С. 38—43.

58. *Рябцев А.*, *Дулин С.* Интеллектуализация анализа выполнения запросов в колоночной СУБД // Тезисы докладов 14-й международной конференции ”Интеллектуализация обработки информации”. — Москва, 2022. — С. 103—105.

59. *Дулин С.* Исследование сетей с диссонансами // Известия АН СССР. Техническая кибернетика. — 1982. — № 5. — С. 74—85.

60. *Дулин С.* Введение в диссонансную логику // Вычислительные машины и искусственный интеллект. — 1982. — Т. 1, № 4. — С. 291—299.

61. *Baas J.*, *Dastani M.*, *Feelders J.* Exploiting Transitivity for Entity Matching // The Semantic Web: ESWC Satellite Events: Virtual Event. Revised Selected Papers 18. — Cham : Springer International Publishing, 2021. — С. 109—114.

62. *Girvan M.*, *Newman M. E.* Community structure in social and biological networks // Proceedings of the national academy of sciences. — 2002. — Т. 99, № 12. — С. 7821—7826.

63. Fast unfolding of communities in large networks / V. D. Blondel [и др.] // Journal of statistical mechanics: theory and experiment. — 2008. — Т. 2008, № 10. — P10008.

64. *Zhu X.*, *Zoubin G.* Learning from Labeled and Unlabeled Data with Label Propagation : тех. отч. / Carnegie Mellon University. — 2002. — CMU-CALD-02—107. — URL: https://mlg.eng.cam.ac.uk/zoubin/papers/CMU-CALD-02-107.pdf.

65. *Rosvall M.*, *Bergstrom C. T.* Maps of random walks on complex networks reveal community structure // Proceedings of the national academy of sciences. — 2008. — Т. 105, № 4. — С. 1118—1123.

66. *Pons P.*, *Latapy M.* Computing communities in large networks using random walks // Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20. — Springer. 2005. — С. 284—293.

67. *Shi J.*, *Malik J.* Normalized cuts and image segmentation // IEEE Transactions on pattern analysis and machine intelligence. — 2000. — Т. 22, № 8. — С. 888—905.

68. Mapreduce: simplified data processing on large clusters. / J. Dean, S. Ghemawat [и др.] // osdi. Т. 4. — USA. 2004. — С. 5.

69. *Kaufman L.*, *Rousseeuw P. J.* Finding groups in data: an introduction to cluster analysis. — John Wiley & Sons, 2009.

70. *Антипов И.*, *Дулин С.*, *А.Б Р.* Формирование групп идентичных объектов // Известия РАН. Теория и системы управления. — 2025. — № 3. — С. 113—120.

71. *Рябцев А.*, *Дулин С.* Повышение структурной согласованности в задаче поиска групп идентичных объекто // Тезисы докладов 15-й международной конференции ”Интеллектуализация обработки информации”. — Гродно, 2024. — С. 33—35.

72. *Рябцев А.*, *Дулин С.* Подход к повышению согласованности структурной интероперабельности // Тезисы докладов 66-ой Всероссийской научной конференция МФТИ. — Долгопрудный, 2024. — С. 244—247.

Список рисунков

Список таблиц