ПРОЕКТИРОВАНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ СЕМАНТИЧЕСКИХ РЕКОМЕНДАЦИЙ КНИГ

Жданова С.И., ст. преподаватель,
Путилин Н.И., студент,
БГТУ им. В.Г. Шухова, г. Белгород, Россия

Аннотация. В статье описано проектирование интеллектуальной системы управления персональной библиотекой с функцией семантических рекомендаций. Технической основой выступает база данных PostgreSQL (pgvector) и локальные нейросетевые модели Ollama, что обеспечивает полную конфиденциальность пользовательских данных. Разработанные функциональные модели и алгоритм векторного поиска по косинусному сходству закладывают надежный фундамент для программной реализации приложения.

Ключевые слова: интеллектуальная система, семантические рекомендации, векторные эмбеддинги, локальные LLM, PostgreSQL, pgvector, косинусное сходство, IDEF0.

Создание любой современной интеллектуальной системы неизбежно сталкивается с проблемой четкого описания внутренних связей. Особенно остро этот вопрос встает на тех участках, где классические реляционные базы данных пересекаются с нейросетевыми алгоритмами. Чтобы избежать архитектурного хаоса на этапе написания кода, потребовалось применить строгий методологический подход [1]. Базовым инструментом формализации здесь выступил стандарт IDEF0 [3], который отлично подходит для демонстрации программного комплекса как единого конвейера переработки информации. Если рассмотреть начальный этап проектирования, то разработанная контекстная диаграмма верхнего уровня (A0) (Рис. 1) сразу задает жесткие рамки проекта. На вход системы подаются совершенно

разнородные данные. С одной стороны, это стандартизированные учетные записи. С другой -абсолютно неструктурированный контент, будь то тяжелые файлы форматов ePub и PDF или просто случайно выделенные пользователем куски текста. Проходя через алгоритмы системы, весь этот сырой материал должен превратиться в упорядоченную цифровую библиотеку, где к каждому изданию привязан свой интеллектуальный помощник.

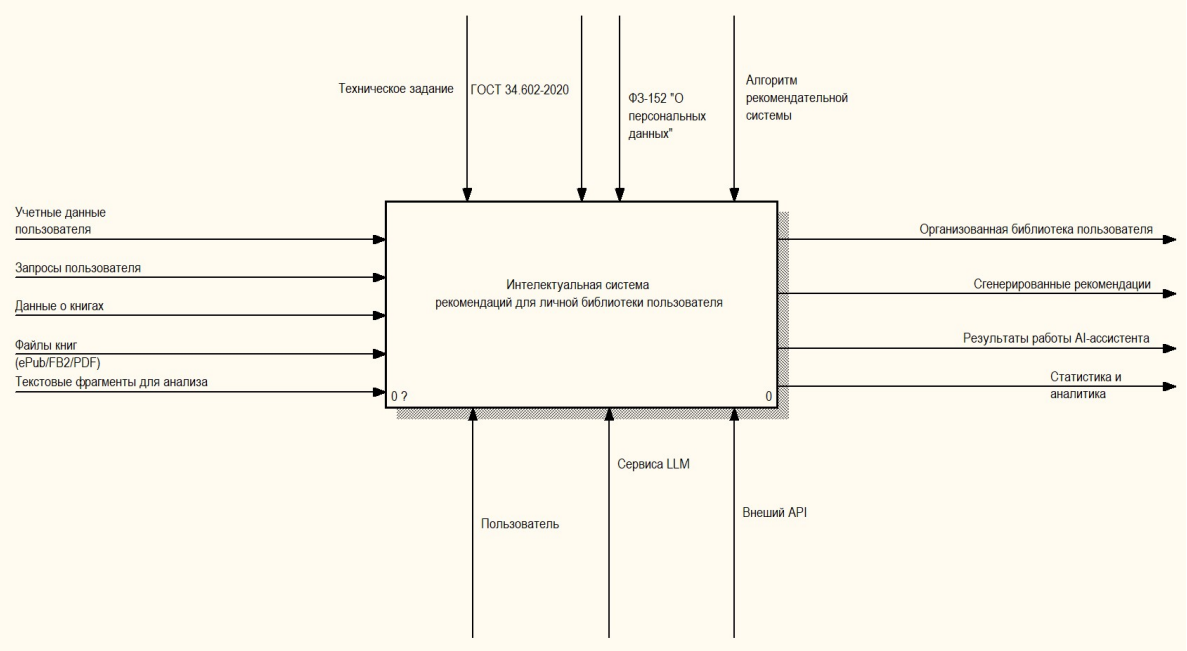

Рис. 1. Контекстная диаграмма IDEF0

Если сравнивать нашу архитектуру с привычными электронными каталогами, то главное отличие кроется в самой сути управляющих воздействий. Зачастую нормативные акты воспринимаются при разработке как фоновая формальность. Однако в нашей модели жесткие требования ФЗ-152 «О персональных данных» [5] наравне с алгоритмами семантического поиска работают как активные регуляторы. Иными словами, абсолютно любая операция, начиная от загрузки обложки книги и заканчивая генерацией ответа нейросетью, изначально зажата в тиски требований информационной безопасности и математической логики. Что касается механизмов реализации, то здесь упор сделан на полный отказ от облачных решений. В качестве основного ресурса выступает локально развернутая среда Ollama с большими языковыми моделями. Подобная реализация позволяет вообще не зависеть от

платных или нестабильных внешних API.

Когда началась детальная декомпозиция внутренних процессов, стало очевидно, что систему нужно разбить на четыре независимые зоны ответственности. Сюда вошли подсистема управления доступом, модуль ведения каталога, интерфейс читалки и блок интеллектуальных сервисов. Последний модуль оказался самым тяжелым с алгоритмической точки зрения. Ему приходится непрерывно собирать телеметрию со всех соседних подсистем, чтобы максимально точно сформировать вектор читательских предпочтений. Сама логика работы с запросами к нейросети выстроена на принципах строгой асинхронности. Допустим, читатель выделяет абзац текста, чтобы получить его краткий пересказ. В этот момент интерфейс приложения не имеет права зависать. Программа инициирует параллельное выполнение сразу нескольких задач: пока локальная LLM пытается сгенерировать осмысленный ответ, сервер в фоновом режиме пишет логи транзакций, а на экране клиента крутится плавная анимация. Именно такой вариант архитектуры страхует приложение от возникновения зависаний и так называемых «мертвых зон» в интерфейсе, многократно повышая общую отказоустойчивость готового продукта.

Когда теоретическое моделирование переходит в стадию технической реализации, первой серьезной проблемой становится организация хранения информации. Нам потребовалась гибридная структура базы данных. Нужно было заставить классические реляционные таблицы эффективно работать с многомерными векторами. В качестве технологического фундамента мы взяли СУБД PostgreSQL, интегрировав в нее расширение pgvector. Подобное решение позволило опустить математический аппарат искусственного интеллекта прямо на уровень хранения данных. В итоге расчет косинусного расстояния выполняется практически с той же скоростью, что и рутинный SQL-запрос.

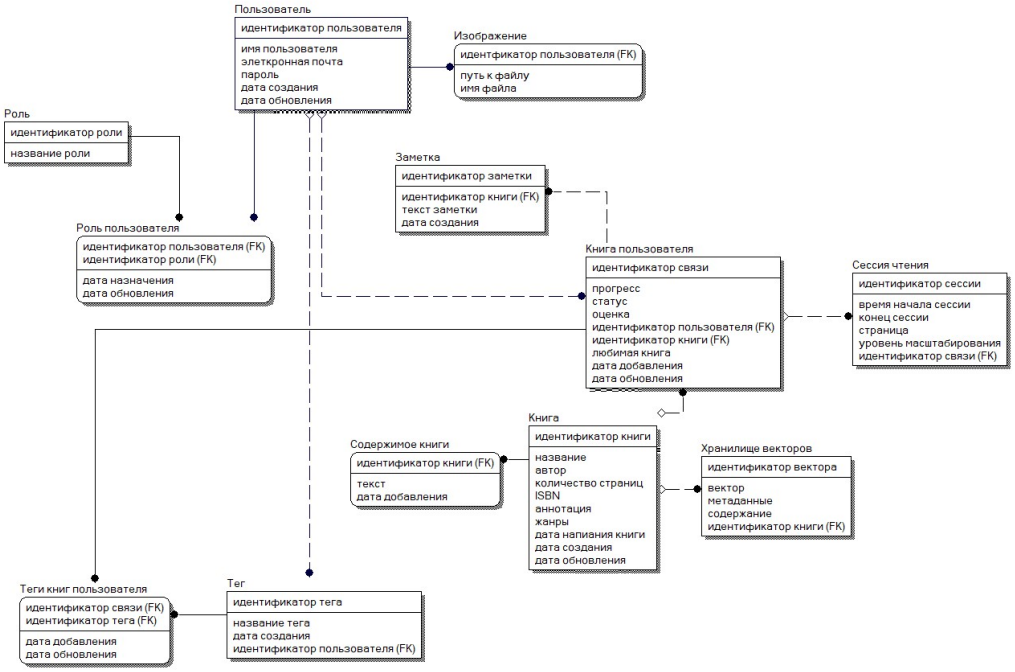

Рис. 2. Логическая схема базы данных

Если посмотреть на логическую схему (Рис. 2), можно заметить, что она опирается на принцип вертикального шардирования. Говоря простым языком, мы жестко разделили краткие библиографические описания и тяжелые текстовые массивы литературных произведений. Этот архитектурный прием отлично снимает нагрузку с шины данных. Пока читатель просто просматривает каталог библиотеки, система не пытается вытянуть из базы мегабайты текста, подгружая контент исключительно при запуске режима чтения. Центральное место в спроектированной схеме занимает таблица «Хранилище векторов». В ней лежат эмбеддинги размерностью 768, жестко привязанные к уникальным идентификаторам контента. По сути, эта таблица выступает в роли глобального семантического индекса. Имея под рукой такой индекс, нейросети больше не требуется заниматься полным ресурсоемким перебором базы для поиска смысловых совпадений.

В плане программного кода мы взяли за основу проверенную трехуровневую архитектуру, концептуально дополнив её изолированным слоем искусственного интеллекта. За клиентскую часть отвечает библиотека React.js, поскольку нам была критически важна отзывчивость интерфейса и возможность локального управления сложными визуальными состояниями.

Серверный бэкенд написан на Java с использованием фреймворка Spring Boot. В нашем проекте он играет роль интеллектуального диспетчера, который непрерывно маршрутизирует потоки данных между фронтендом, базой PostgreSQL и нейросетевым ядром.

Наверное, главной особенностью предложенного архитектурного решения стал принципиальный отказ от облачных вычислений. Мы полностью локализовали мощности искусственного интеллекта через платформу Ollama. Под капотом экосистемы развернуты сразу две специализированные модели. Первая, Gemma-Embedding [2], функционирует в фоновом режиме и занята исключительно векторизацией загружаемых текстов. Вторая модель, Llama 3.1, отвечает за генеративные задачи: делает краткие выжимки, переводит фрагменты и объясняет сложные термины. Запуск AI-компонентов внутри закрытого контура нашего сервера решает одну из самых острых проблем современных сервисов -утечку пользовательских данных. История запросов, приватные заметки и тексты авторских книг вообще не уходят на сторонние API. Это дает отличный компромисс между высокой мощностью приложения и железным соблюдением конфиденциальности.

На финальной стадии проектирования фокус сместился на то, чтобы связать сложную математику алгоритмов с понятным пользовательским интерфейсом. За основу визуальной концепции мы взяли гайдлайны Material Design 3, но при этом сильно разбавили их приемами современного скевоморфизма. Практика показывает, что бесконечные плоские списки быстро вызывают у читателя когнитивную усталость. Поэтому было решено использовать старую добрую метафору реального книжного шкафа (Рис. 3).

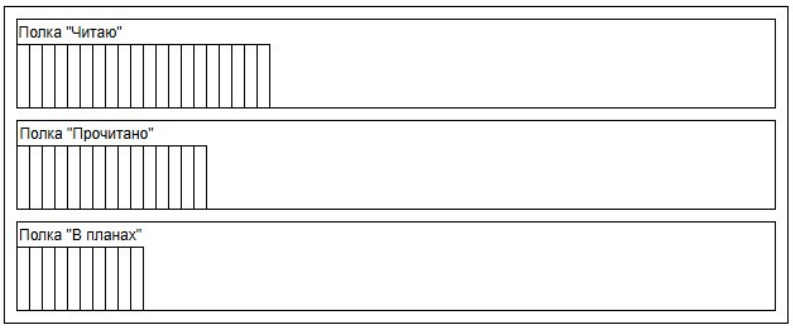

Рис. 3. Главный экран системы

Главной особенностью UX-дизайна (Рис. 3) стал отказ от классических кнопок в пользу механики Drag-and-Drop. Мы постарались перевести рутинное управление библиотекой на уровень интуитивных физических жестов. Человек просто берет и перетаскивает виртуальный томик на нужную полку, а интерфейс тут же откликается реалистичной анимацией наклона и сменой теней. Чтобы визуально уравновесить современные технологии и классическую культуру чтения, в типографике мы сочетали строгую антикву Playfair Display для заголовков с хорошо читаемым гротеском Inter для основного массива текста. Сам процесс погружения в книгу протекает в режиме нулевого вмешательства: все нейросетевые инструменты спрятаны и вызываются только при выделении конкретного абзаца, чтобы лишний раз не дергать фокус внимания пользователя.

Если говорить об интеллектуальном ядре системы, то здесь главную роль играет нестандартный алгоритм семантических рекомендаций [4]. Большинство популярных сервисов сегодня идут по пути наименьшего сопротивления, применяя коллаборативную фильтрацию -то есть просто анализируют оценки толпы. Наш проект опирается исключительно на векторный анализ самого текста. Именно это решение помогает обойти извечную проблему «холодного старта». Программе не нужны тысячи лайков от других людей: загрузив всего один файл, система сканирует его смысловой отпечаток и сразу выдает адекватную рекомендацию.

В процессе выбора математического аппарата для рекомендательной модели мы остановились на метрике косинусного сходства (Cosine Similarity).

Почему не Евклидово расстояние? Дело в том, что оно слишком чувствительно к объему текста. А вот косинусная метрика смотрит только на направление векторов в многомерном пространстве, игнорируя разницу в длине аннотаций. Математически расчет степени сходства между вектором загруженной книги A и вектором читательских интересов B выглядит следующим образом:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

Такая формула вытаскивает на поверхность очень глубокие тематические пересечения, которым абсолютно не важны стандартные магазинные ярлыки жанров. Если прогнать предварительно сжатые аннотации через модель Gemma-Embedding с размерностью в 768 параметров, мы получаем высочайшую точность рекомендаций, при этом сохраняя минимальную нагрузку на вычислительные мощности сервера.

Подводя черту под этапом проектирования, можно сказать, что нам удалось собрать довольно нетипичную архитектуру. В ней жесткий инженерный формализм методологии IDEF неплохо ужился с вероятностными методами обработки естественного языка. Связка из реляционно-векторной базы PostgreSQL и закрытой инфраструктуры Ollama доказала, что мощные интеллектуальные сервисы вполне могут быть одновременно быстрыми и абсолютно приватными. Разработанные алгоритмы семантического поиска вместе с продуманным интерфейсом полностью закрывают проектную стадию работы, формируя прочный технический фундамент для дальнейшего написания программного кода и тестирования системы.

Литература

1. ГОСТ Р 7.0.100–2018. Библиографическая запись. Библиографическое описание. Общие требования и правила составления: нац. стандарт Рос. Федерации. М.: Стандартинформ, 2018. 124 с.

2. Жданова С.И. Цифровая трансформация в формировании образовательных траекторий личности // Наукоемкие технологии и инновации

(XXV научные чтения): сб. докл. Междунар. науч.-практ. конф. Белгород: БГТУ им. В.Г. Шухова, 2023. С. 713-716.

3. Марка Д.А. Методология структурного анализа и проектирования: SADT (Structured Analysis & Design Technique) / Д.А. Марка, К.Л. МакГоуэн. М.: Мета-технология, 1993. 240 с.

4. О персональных данных: Федер. закон Рос. Федерации от 27 июля 2006 г. N 152-ФЗ // Собрание законодательства Рос. Федерации. 2006. N 31 (ч. 1). Ст. 3451.

5. Солонин Е.Б. Интеллектуальные технологии поиска и анализа данных [Электронный ресурс] / Урал. федер. ун-т. Екатеринбург: УрФУ, 2015. URL: https://study.urfu.ru/Aid/Publication/13334/1/Solonin.pdf (дата обращения: 03.05.2026).