ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 3 за 2025 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

И.С. Ведьманов Т.В. Сивакова, В.А. Судаков,

Интеллектуальные технологии для формирования рекомендаций по запросам на естественном языке

Рекомендуемая форма библиографической ссылки: Ведьманов И.С., Сивакова Т.В., Судаков В.А. Интеллектуальные технологии для формирования рекомендаций по запросам на естественном языке // Препринты ИПМ им. М.В.Келдыша. 2025. № 3. 12 с. EDN: ALJDCU
https://library.keldysh.ru/preprint.asp?id=2025-3

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Российской академии наук

И.С. Ведьманов, Т.В. Сивакова, В.А. Судаков

Интеллектуальные технологии для формирования рекомендаций по запросам на естественном языке

Москва — 2025

Ведьманов И.С., Сивакова Т.В., Судаков В.А.

Интеллектуальные технологии для формирования рекомендаций по запросам на естественном языке

В данной статье представлена разработка интеллектуальной системы для поиска и рекомендаций организаций в сфере услуг. Программное обеспечение позволяет пользователям вводить запросы на естественном языке, после чего происходит классификация запроса и выдача наиболее релевантных результатов из датасета Yandex Geo-Reviews. Описана архитектура программного обеспечения, включающая фронтенд на фреймворке React, бэкенд на языке Golang с использованием FastHTTP, а также ML-сервис на платформе FastAPI. Используются модели SBERT для анализа текстов и классификатор CatBoost для формирования множества классов предпочтений организаций. В статье также рассматриваются методы взаимодействия компонентов системы на концептуальном уровне и на уровне протоколов обмена данными, что позволяет другим разработчикам использовать предложенную методику для создания аналогичных программных решений.

Ключевые слова: машинное обучение, рекомендательные системы, каркасы разработки программного обеспечения, протоколы обмена данными.

Ilya Sergeevich Vedmanov, Tatiana Vladimirovna Sivakova, Vladimir Anatolievich Sudakov

Intelligent technologies for generating recommendations based on natural language queries

This article presents the development of an intelligent system for searching and recommending organizations in the service sector. The software allows users to enter queries in natural language, after which the query is classified and the most relevant results are issued from the Yandex Geo-Reviews dataset. The software architecture is described, including a frontend on the React framework, a backend in the Golang language using FastHTTP, and an ML service on the FastAPI platform. SBERT models are used for text analysis and the CatBoost classifier is used to form a set of preference classes for organizations. The article also discusses methods for interaction between system components at the conceptual level and at the level of data exchange protocols, which allows other developers to use the proposed methodology to create similar software solutions.

Key words: machine learning, recommender systems, software development frameworks, data exchange protocols.

Введение

Современные интеллектуальные системы поиска и рекомендаций играют важную роль в облегчении доступа пользователей к необходимой информации и услугам. В условиях растущего объёма данных и разнообразия пользовательских запросов возникает необходимость в использовании эффективных методов обработки естественного языка (Natural Language Processing, NLP) и точной классификации запросов для предоставления релевантных результатов [1]. Среди последних достижений в этой области выделяются модели-трансформеры, такие как Sentence-BERT (SBERT), и алгоритмы классификации, включая CatBoost, которые демонстрируют высокую точность и производительность в рекомендательных системах [2-4].

Использование трансформеров, включая SBERT, позволяет эффективно извлекать смысловую информацию из текста, что делает их важным инструментом для анализа данных и рекомендаций. Например, комбинация SBERT и других трансформеров, таких как RoBERTa, показала улучшение точности рекомендаций в задачах обработки естественного языка [2]. В то же время CatBoost, благодаря своему подходу к обработке категориальных данных, обеспечивает высокую точность классификации, что делает его предпочтительным выбором для задач ранжирования в больших данных [3].

Системы рекомендаций также активно развиваются в направлении гибридных моделей, которые объединяют контентный и коллаборативный подходы с использованием глубинных нейронных сетей и графовых методов. Такие подходы уже находят применение в различных областях, от электронной коммерции до здравоохранения, что подчёркивается в современных обзорах [5-7]. Кроме того, в эпоху больших данных ключевым становится использование технологий распределённой обработки, таких как Hadoop и Spark, что позволяет системам рекомендаций масштабироваться и сохранять производительность при работе с огромными объёмами информации [8].

Цель данной работы заключается в разработке системы, способной обрабатывать пользовательские запросы на естественном языке, классифицировать их по категориям и выдавать релевантные результаты из большого объёма географических отзывов. Предлагаемый подход сочетает возможности SBERT для представления текста и CatBoost для классификации, что позволяет построить систему, сочетающую высокую точность и производительность, и адаптировать её к современным требованиям индустрии [4, 5].

Архитектура системы

Разработанная система состоит из нескольких ключевых компонентов, обеспечивающих обработку запросов пользователя от ввода до отображения результатов. Основные компоненты системы включают фронтенд, бэкенд, ML-

сервис и хранилища данных (см. рис. 1). Ниже приведено детальное описание каждого компонента и показаны их взаимодействия.

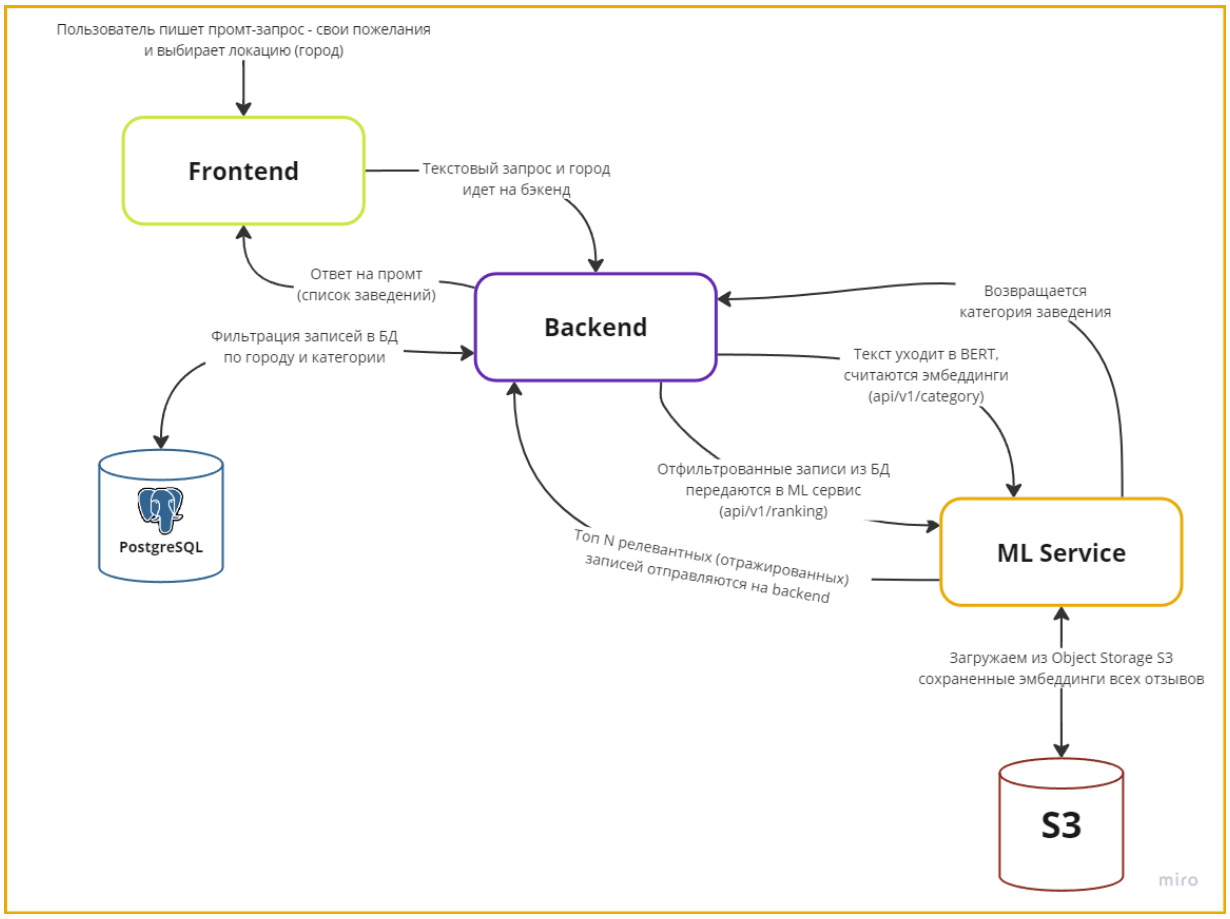

Рис. 1. Основные компоненты рекомендательной системы

Ввод данных пользователем. Пользователь взаимодействует с системой через веб-интерфейс. Он вводит текстовый запрос (например, «хочу поесть вкусной пиццы и выпить квасу») в текстовое поле и выбирает город из выпадающего списка. Фронтенд реализован с использованием современных веб-технологий (фреймворк React), обеспечивающих интерактивность и удобство использования. После ввода данных пользователем они отправляются на бэкенд посредством HTTP-запроса. Это позволяет обеспечить эффективную передачу информации между клиентом и сервером.

Обработка на бэкенде. Бэкенд, реализованный на языке Golang с использованием библиотеки FastHTTP, получает входящий запрос от фронтенда. Он извлекает текст запроса и выбранный город из полученных данных. Далее бэкенд отправляет текст запроса на ML-сервис через API endpoint /api/v1/category. Это позволяет использовать возможности машинного обучения для классификации запроса.

Классификация текста реализуется ML-сервисом. Для этого модель SBERT (Sentence-BERT) преобразует текст запроса в эмбеддинг — числовое

представление смыслового содержания текста. Это позволяет эффективно использовать текстовые данные для дальнейшей обработки.

Классификатор CatBoost на основе полученного эмбеддинга выполняет классификацию по множеству классов, предсказывая категорию заведения, соответствующую запросу пользователя. Общий вид функции потерь с регуляризацией:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} Loss(y_i, \hat{y}_i) + \lambda \cdot \Omega(w),$$

где y_{ij} — истинная метка класса, $\hat{y}_{ij}$ — предсказанная метка класса, Loss(·) — функция ошибки, $\Omega(w)$ — регуляризационный член (например, L2-норма), w – параметры модели, λ — коэффициент регуляризации. Предсказанная категория возвращается на бэкенд для дальнейшей обработки.

Фильтрация данных. Бэкенд использует предсказанную категорию и выбранный город для фильтрации записей в базе данных PostgreSQL, содержащей информацию о заведениях, их отзывах и категориях. Отфильтрованные записи передаются на ML-сервис через API endpoint /api/v1/ranking для дальнейшего ранжирования.

Ранжирование записей осуществляет ML-сервис. При запуске сервера предварительно загружаются эмбеддинги текстовых отзывов из хранилища S3 для быстрого доступа. По идентификаторам записей, полученных от бэкенда, извлекаются соответствующие эмбеддинги. Вычисляется косинусовое сходство между эмбеддингом пользовательского запроса и эмбеддингами записей, что позволяет определить степень релевантности:

$$S_c(A, B) = \frac{\sum_{j=1}^{n} A_j B_j}{\sqrt{\sum_{j=1}^{n} A_j^2} \cdot \sqrt{\sum_{j=1}^{n} B_j^2}},$$

где A, B – это n-мерные векторы признаков пользовательского запроса и организации соответственно, а A_j и B_j – это элементы соответствующих векторов. Далее проводится ранжирование: учитывается рейтинг заведения, и отбирается топ M наиболее релевантных записей. Формула взвешенного суммирования с использованием сходства, рейтингов и популяроности организаций:

$$S_i = \alpha \cdot S_c\left(E_q, E_i\right) + \beta \cdot R_i + \gamma \cdot P_i,$$

где S_i — итоговый скор записи i, E_q — эмбеддинг запроса, E_i — эмбеддинг отзыва, R_i — рейтинг заведения, P_i — показатель популярности заведения, а α, β, γ — веса показателей. В перспективе возможно применение более сложной системы ценностей пользователя в форме гибридной функции предпочтений или нечетких областей предпочтений [9].

Расчет рейтинга заведения R_i представляет собой нетривиальную задачу. Большинство сервисов агрегатов используют расчет среднего по всем отзывам. Но при наличии большого количества положительных отзывов несколько отрицательных отзывов слабо влияют на итоговую оценку. Альтернативным механизмом учета отрицательного рейтинга является подсчет числа отрицательных. И при превышении числом негативных оценок определённого порога θ организация получает нулевой рейтинг:

$$R_i = \begin{cases} 0, \text{если } \left|\{r_{ij} : r_{ij} \le 2, j = \overline{1,m}\}\right| > \theta \\ \sum_{j=1}^{m} r_{ij} / m, \text{иначе} \end{cases}.$$

Выбор корректного значения θ затруднителен. По этой причине вместо суммирования для подсчета R_i предложено использовать мультипликативную свертку вида:

$$R_i = \prod_{j=1}^{m} r_{ij}^{w_j},$$

где r_{ij} – оценка организации i пользователем j, w_j – (важность) пользователя j на сайте с отзывами. Таким образом, если оценка пользователя $r_{ij} \to 0$, то общий рейтинг $R_i \to 0$ для конечного m. Целесообразно также устранять смещение в оценках пользователей оптимистов/пессимистов:

$$r_{ij} \leftarrow r_{ij} - \min_j r_{ij} + 1.$$

Отображение результатов. Бэкенд получает ранжированный список записей и передает его на фронтенд. Фронтенд отображает пользователю список релевантных заведений с учетом предпочтений и выбранного города.

Эмбеддинги текстовых данных хранятся в хранилище S3 для обеспечения быстрого доступа и масштабируемости системы. Информация о заведениях, отзывы пользователей, категории и другая метаинформация хранятся в базе данных PostgreSQL. Это обеспечивает надежное и эффективное управление данными с возможностью выполнения сложных запросов и фильтрации.

Фронтенд разработан с использованием фреймворка React, обеспечивающего интерактивность и отзывчивость интерфейса. Бэкенд реализован на языке Golang с использованием библиотеки FastHTTP, что обеспечивает высокую производительность и возможность обработки большого количества запросов. ML-сервис построен на основе FastAPI, что позволяет создавать быстрые и масштабируемые API для взаимодействия с моделями машинного обучения. Для преобразования текста в эмбеддинги используются модели SBERT (largenluru) и CatBoost Classifier для выполнения классификации запросов. Эмбеддинги хранятся в базе данных S3, а основная информация о заведениях и отзывах — в PostgreSQL.

Взаимодействие компонентов системы

На концептуальном уровне система разделена на фронтенд, бэкенд и ML-сервис. Такой подход обеспечивает модульность и независимость компонентов, что облегчает их разработку, тестирование и масштабирование. Взаимодействие между компонентами осуществляется через RESTful API, что обеспечивает стандартизированный и легковесный способ обмена данными. REST использует не сохраняющий состояние клиент-серверный кэшируемый протокол связи, который почти всегда является протоколом HTTP. Его изначальной особенностью является работа с использованием простого HTTP для совершения вызовов между машинами вместо выбора сложных механизмов. Характерными элементами технологии RESTFul являются источники определенной информации, которые называются ресурсами. Каждый из них связан с глобальным идентификатором, например, URI в HTTP. К этим ресурсам обращаются компоненты сети (пользовательские агенты и серверы), которые взаимодействуют через стандартизированный протокол (например, HTTP) и обмениваются содержимым (представлениями) этих ресурсов. Для того чтобы взаимодействовать с ресурсом, приложение должно обладать как идентификатором ресурса, так и требуемым методом. Напротив, нет необходимости знать реализацию служб и конфигурацию системы, т. е. есть ли кэши, прокси, шлюзы, брандмауэры, туннели или что-либо еще между приложением и сервером, на котором размещены ресурсы. Однако приложение должно иметь возможность интерпретировать формат данных (представление), возвращаемый ресурсом, который часто является документом JSON или XML, хотя это также может быть изображение, простой текст или любой другой контент.

Взаимодействие компонентов системы основано на четырех принципах:

1. Идентификация ресурсов через URI. Ресурсы идентифицируются с помощью URI, которые предоставляют механизм обнаружения сервисов.
2. Единый интерфейс. Фиксированный набор из четырех операций создания, чтения, обновления и удаления отвечает за манипулирование ресурсами.
3. Самоописательные сообщения. Содержимое ресурсов может быть представлено в нескольких форматах (например, HTML, XML, простой текст, PDF или JPEG). Метаданные о ресурсе могут использоваться для обнаружения ошибок передачи и выполнения аутентификации или контроля доступа.
4. Взаимодействие с сохранением состояния через гиперссылки. Взаимодействие с сохранением состояния может быть достигнуто с помощью нескольких методов, например, переписывания URL, файлов cookie или скрытых полей форм.

Преимущества RESTFul:

- Веб-сервисы RESTFul достаточно простые, поскольку REST применяет множество существующих известных стандартов (HTTP, XML, URI

и MIME) и нуждается только в той инфраструктуре, которая уже стала обыденной.

- HTTP-клиенты и серверы совместимы со всеми языками программирования и операционными системами/аппаратными платформами, а HTTP-порт 80 по умолчанию обычно остается открытым в большинстве конфигураций брандмауэра.
- Для создания клиента RESTFul-сервиса требуется лишь небольшое усилие. Сервисы можно тестировать, используя всего лишь обычный веб-браузер, и разработка клиентского программного обеспечения становится излишней.
- REST позволяет обнаруживать веб-ресурсы без какого-либо обнаружения или репозитория реестра.

К недостаткам RESTFul можно отнести:

- Кодирование большого объема входных данных в URI ресурса невозможно, так как сервер либо отклоняет такие запросы, либо аварийно завершает работу.
- Также может быть сложно кодировать сложные структуры данных в URI, поскольку нет общепринятого механизма маршалинга. По сути, метод POST не страдает от таких ограничений.
- В отличие от веб-сервисов на основе SOAP, которые имеют стандартный словарь для описания интерфейса веб-сервиса через WSDL, веб-сервисы RESTFul в настоящее время не имеют такой грамматики. И потребитель сервиса, и производитель сервиса должны иметь внеполосное соглашение. Сервисы могут быть описаны с помощью языка описания веб-приложений (WADL). Это основанный на XML формат файла, который предоставляет машиночитаемое описание веб-сервисов REST. WADL пока не получил широкой поддержки.
- В то время как веб-сервисы на основе SOAP поддерживают стандартный словарь для определения интерфейса веб-сервиса с помощью WSDL, веб-сервисы RESTFul не обладают такой возможностью.

Несмотря на наличие некоторых недостатков у RESTFul, его достоинства перевешивают, в частности, ключевую роль играет ускорение разработки за счет простоты. Кроме того, недостатки отсутствия контроля спецификации частично устраняются путем написания тестов, покрывающих API.

Взаимодействие включает в себя следующие шаги:

1. Пользователь вводит запрос и выбирает город на фронтенде.
2. Фронтенд отправляет данные на бэкенд через HTTP-запрос.
3. Бэкенд обрабатывает запрос и отправляет текст на ML-сервис для классификации.
4. ML-сервис отвечает с предсказанной категорией.
5. Бэкенд фильтрует данные из базы и отправляет их на ML-сервис для ранжирования.

6. ML-сервис возвращает ранжированный список, который бэкенд передает на фронтенд.
7. Фронтенд отображает результаты пользователю.

Для преобразования пользовательских запросов в числовые представления используется модель SBERT (Sentence-BERT), которая эффективно кодирует смысловую информацию текста в высокоразмерные эмбеддинги. Эта модель обучена на большом объеме русскоязычных данных, что обеспечивает хорошую производительность для задач, связанных с естественным языком.

Для многоклассовой классификации категорий заведений применяется модель CatBoost Classifier. Эта модель обладает высокой точностью и способностью работать с различными типами данных, включая числовые и категориальные признаки. Модель обучена на эмбеддингах, полученных из SBERT, что позволяет эффективно классифицировать запросы пользователей.

Для определения релевантности заведений используется косинусовое сходство между эмбеддингами пользовательского запроса и эмбеддингами отзывов из базы данных. Это позволяет определить степень соответствия запроса и содержания отзывов, обеспечивая высокую точность рекомендаций. Дополнительно учитывается рейтинг заведений, что способствует представлению наиболее популярных и качественных вариантов.

Разработка и внедрение

Разработка системы осуществлялась с использованием следующих инструментов и технологий:

- Golang для разработки высокопроизводительного бэкенда.
- FastAPI для создания быстрого и масштабируемого ML-сервиса.
- PostgreSQL как надежной системы управления базами данных.
- БД S3 для хранения больших объемов данных, таких как эмбеддинги.
- CatBoost и SBERT для реализации моделей машинного обучения.
- Docker для контейнеризации компонентов системы и обеспечения их независимости при развертывании.

Этапы разработки

1. Анализ требований: Определение функциональных и нефункциональных требований к системе.

2. Проектирование архитектуры: Разработка общей структуры системы и взаимодействия компонентов.

3. Разработка фронтенда: создание пользовательского интерфейса с возможностью ввода запросов и отображения результатов.

4. Разработка бэкенда: реализация серверной части на Golang с использованием FastHTTP.

5. Создание ML-сервиса: реализация моделей SBERT и CatBoost в рамках сервиса на FastAPI.

6. Интеграция компонентов: обеспечение взаимодействия между фронтендом, бэкендом и ML-сервисом через API.

7. Тестирование: проведение модульного и интеграционного тестирования для обеспечения корректности работы системы.

8. Развертывание: использование контейнеризации с Docker для развертывания системы в производственной среде.

9. Мониторинг и оптимизация: непрерывный мониторинг производительности системы и внесение необходимых оптимизаций.

На рисунке 2 показан фрагмент интерфейса системы. Для введенного запроса и локации система рекомендует организацию с учетом рейтинга и релевантности отзывов.

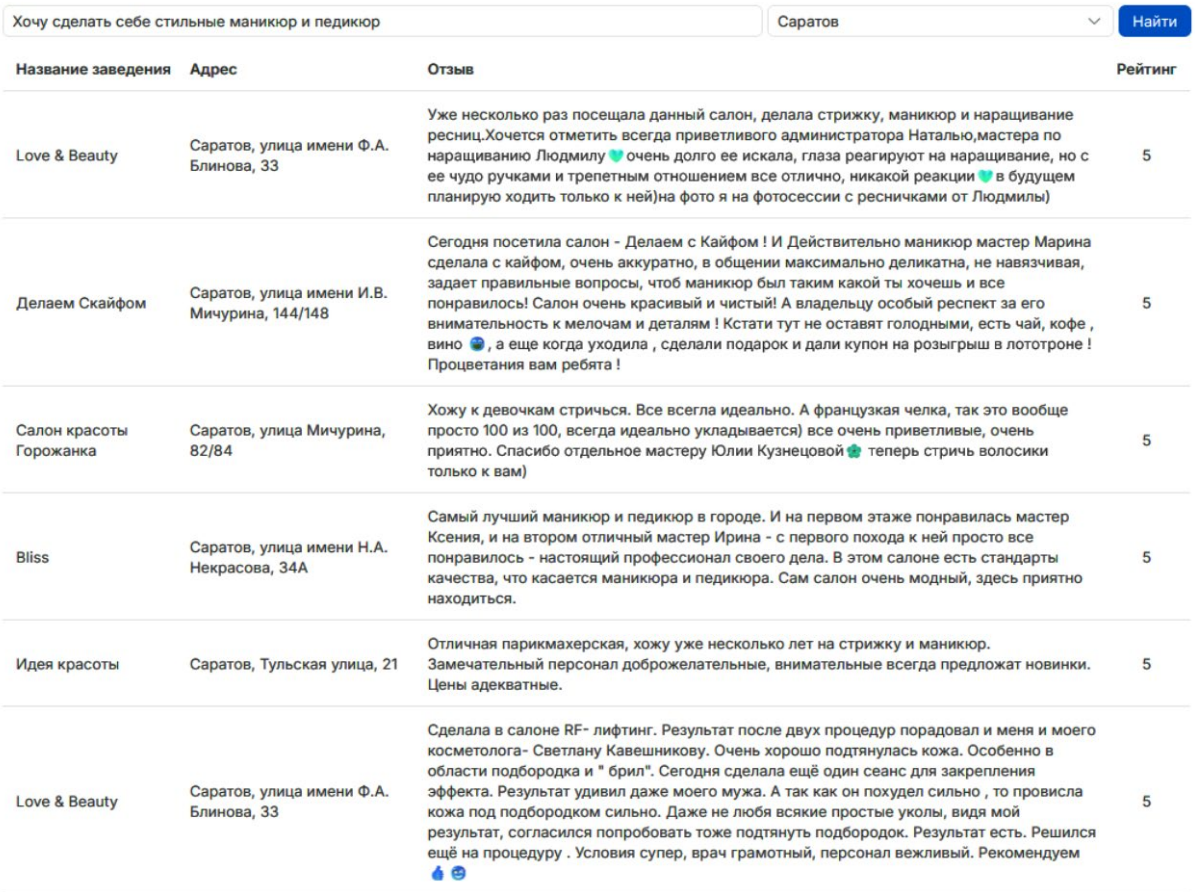

Рис. 2. Результат работы интеллектуальной рекомендательной системы

Заключение

Разработанная система демонстрирует эффективное применение современных методов машинного обучения для решения задач классификации и ранжирования в контексте поиска и рекомендаций организаций в сфере услуг. Описанная методика и архитектура могут служить основой для создания аналогичных решений в различных областях, требующих обработки естественного языка и предоставления персонализированных рекомендаций.

Созданное программное обеспечение обладает несколькими ключевыми преимуществами:

- Модульность. Разделение системы на независимые компоненты облегчает внесение изменений и масштабирование.
- Эффективность. Использование современных моделей машинного обучения обеспечивает высокую точность классификации и ранжирования.
- Гибкость. Стандартизированные API позволяют легко интегрировать систему с другими сервисами и расширять ее функциональность.
- Масштабируемость. Хранилище данных на основе S3 и PostgreSQL обеспечивает возможность обработки больших объемов данных.
- Для разработчиков. Четкая методика взаимодействия компонентов позволяет другим разработчикам быстро создавать аналогичные решения, адаптируя предложенную архитектуру под свои задачи.

Реализована архитектура системы, включая фронтенд, бэкенд и ML-сервис, а также использованы технологии и методы машинного обучения. Разработанная методика взаимодействия компонентов системы на концептуальном и протокольном уровнях предоставляет разработчикам четкие рекомендации для создания аналогичных программных решений. Проведенный анализ и описанные этапы разработки демонстрируют эффективность предложенного подхода в решении задач классификации запросов и предоставления релевантных рекомендаций, что имеет практическую значимость для улучшения пользовательского опыта в сфере поиска требуемых услуг.

Библиографический список

1. Баданина Н.Д., Судаков В.А. Модели машинного обучения для классификации отзывов о банках // Препринты ИПМ им. М.В. Келдыша. - 2021. - № 50. - С. 1-14. - DOI: 10.20948/prepr-2021-50. EDN: FYMDZL

2. Javaji S. R., Sarode K. Multi-BERT for Embeddings for Recommendation System // arXiv.org. 2023. URL: https://arxiv.org/abs/2308.13050 (дата обращения: 02.12.2024).

3. Prokhorenkova L., Gusev G., Vorobev A., Dorogush A. V., Gulin A. CatBoost: unbiased boosting with categorical features // NeurIPS Proceedings. 2018. URL: https://proceedings.neurips.cc/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf (дата обращения: 02.12.2024). EDN: NRYEVS

4. Ahanger, M. M., Wani, M. A., Palade, V. sBERT: Parameter-Efficient Transformer-Based Model for Scientific Literature Classification // Knowledge. 2024. Vol. 4, №3. P. 397-421. DOI: 10.3390/knowledge4030022 EDN: JAJDNQ

5. Raza, S., Rahman, M., Kamawal, S., Toroghi, A., Raval, A., Navah, F., Kazemeini, A. A Comprehensive Review of Recommender Systems: Transitioning from Theory to Practice // arXiv.org. 2024. URL: https://arxiv.org/abs/2407.13699 (дата обращения: 02.12.2024).

6. A Systematic Literature Review on AI-Based Recommendation Systems and Their Ethical Considerations // IEEE Journals & Magazine. 2024. URL: https://ieeexplore.ieee.org/document/10654261 (дата обращения: 02.12.2024).

7. Recent Developments in Recommender Systems: A Survey // arXiv.org. 2023. URL: https://arxiv.org/abs/2306.12680 (дата обращения: 02.12.2024).

8. A Survey on Modern Recommendation Systems based on Big Data // arXiv.org. 2022. URL: https://arxiv.org/abs/2206.02631 (дата обращения: 02.12.2024).

9. Баданина Н.Д., Зинченко А.А., Судаков В.А. Интегрирование модуля нечеткого логического вывода с помощью веб-технологий // Научный сервис в сети Интернет: труды XXV Всероссийской научной конференции (18-21 сентября 2023 г., онлайн). - М.: ИПМ им. М.В.Келдыша, 2023. - С. 62-73. DOI: 10.20948/abrau-2023-5 EDN: FLWBON

Оглавление