

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

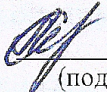
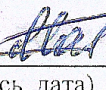
вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»
Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

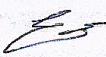
МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
на тему
РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА
ТУРИСТИЧЕСКИХ МАРШРУТОВ

Студент  31.05.2022 Сафронова Валерия Сергеевна
(подпись, дата) (ФИО полностью)
Руководитель  31.05.2022 М.П. Синев
(подпись, дата) (фамилия, инициалы)
Нормоконтролёр  31.05.2022 А.С. Бычков
(подпись, дата) (фамилия, инициалы)
Рецензент зав. кат. Программиров. Пенз. ГУ  02.06.2022 А.И. Мартышкин
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 01.06.2022 № 13)

Заведующий кафедрой  (подпись) М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от
14 июня 2022 № 2)

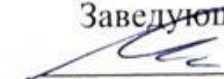
Секретарь ГЭК  (подпись) Е.И. Гурин
(фамилия, инициалы)

Пенза 2022

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
« 4 » сентября 20 20 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Сафронова Валерия Сергеевна гр. 20ВВм1 факультета ВТ

2. Тема работы «Рекомендательная система туристических маршрутов»

Тема утверждена приказом ПГУ №656/о от "21" сентября 2020 г.

3. Руководитель магистерской диссертации к.т.н., доцент Синев М.П.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: разработка рекомендательной системы для решения задачи прогнозирования туристических маршрутов на основе предпочтений пользователя.

Разрабатываемая система должна включать в себя алгоритм прогнозирования с применением корректирующих коэффициентов для решения проблемы информационного пузыря.

5. Перечень подлежащих разработке вопросов

1. Анализ предметной области и критериев классификации туристических маршрутов.
2. Выявление методов, необходимых для классификации туристических маршрутов и используемых для формирования списка прогнозов.
3. Разработка рекомендательной системы туристических маршрутов.
4. Анализ полученных результатов.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области и критериев классификации туристических маршрутов	14%	21.09.2020 22.12.2020	
2.	Анализ опыта построения рекомендательных систем	7%	23.12.2020 03.02.2021	
3.	Разработка рекомендательной системы туристических маршрутов	32%	04.02.2021 19.08.2021	
4.	Отладка рекомендательного модуля	8%	20.08.2021 08.10.2021	
5.	Обучение модели	17%	09.10.2021 21.01.2022	
6.	Анализ полученных результатов исследования	12%	22.01.2022 06.04.2022	
7.	Оформление выпускной работы	10%	07.04.2022 31.05.2022	

Дата выдачи задания "21" сентября 2020 г.

Руководитель магистерской диссертации

 21.09.2020
(подпись, дата)

Синев М.П.

Задание к исполнению принял студент

 21.09.2020
(подпись, дата)

Сафронова В.С.

Работу к защите допустить

Декан факультета


(подпись, дата)

Фионова Л.Р.

10.06.2022

РЕФЕРАТ

Пояснительная записка содержит 109 листов, 12 рисунков, 15 таблиц, 14 источников, 2 приложения.

РЕЛЕВАНТНЫЕ ПОДБОРКИ, ИНФОРМАЦИОННЫЙ ПУЗЫРЬ,
МАТРИЧНАЯ ФАКТОРИЗАЦИЯ, КОРРЕКТИРУЮЩИЕ
КОЭФФИЦИЕНТЫ, SVD

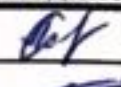
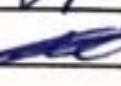
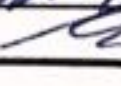
Объектом исследования является объект пользовательского интереса (туристические маршруты, события, локации).

Предметом исследования является рекомендательный сервис для решения задачи прогнозирования туристических маршрутов на основе предпочтений пользователя.

Целью исследования данной работы является разработка рекомендательной системы туристических маршрутов с целью сокращения временных затрат пользователя на поиск туристического маршрута, удовлетворяющего его требованиям, а также повышения востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее.

Научная новизна состоит в модификации существующих алгоритмов прогнозирования посредством применения корректирующих коэффициентов для решения проблемы информационного пузыря.

Результатом данной работы является рекомендательная система туристических маршрутов, обеспечивающая сокращение временных затрат пользователя на поиск туристического маршрута и повышение востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее.

					ПГУ 1.090401.08.001 ПЗ		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб.		Сафронова В.С.		31.05.20	Рекомендательная система туристических маршрутов Пояснительная записка	Лит.	Лист
Провер.		Синев М.П.		31.05.20			2
							109
Н. Коитр.		Бычков А.С.		31.05.20		ФВТ, каф. ВТ гр. 20ВВм1	
Утв.		Митрохин М.А.		31.05.20			

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ	6
1 Анализ предметной области	8
1.1 Анализ существующих туристических платформ	8
1.1.1 Booking.com	8
1.1.2 OneTwoTrip	9
1.1.3 Ostrovok.ru	10
1.2 Анализ существующих подходов составления рекомендаций	11
1.2.1 Матричная факторизация	13
1.2.2 Нейронные сети	15
1.2.3 Байесовские классификаторы	21
1.3 Опыт использования рекомендательных механизмов	24
1.4 Выводы по разделу	25
2 Проектирование рекомендательной системы	26
2.1 Исходные данные для построения системы рекомендательных сервисов	28
2.2 Описание выходных данных рекомендательной системы	36
2.3 Описание алгоритма прогнозирования	36
2.4 Выводы по разделу	44
3 Результаты работы рекомендательной системы	45
3.1 Описание API получения исходных данных	45
3.2 Описание API рекомендательной системы	75
3.3 Тестирование рекомендательной системы	78
3.4 Выводы по разделу	88

ЗАКЛЮЧЕНИЕ	89
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	90
ПРИЛОЖЕНИЕ А Листинги программы	92
ПРИЛОЖЕНИЕ В Презентация	100

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

API – (Application programming interface – «программный интерфейс приложения») – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

CMS (Content management system) – система управления содержимым

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи произвольных данных

JSON (JavaScript Object Notation) – Текстовый формат обмена данными

REST – (Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределенного приложения в сети

SVD (Singular value decomposition) – сингулярная декомпозиция значений

UML (Unified Modeling Language) – унифицированный язык моделирования

URI (Uniform Resource Identifier) – адрес запрашиваемого ресурса

ВВЕДЕНИЕ

Многочисленные онлайн-платформы электронной коммерции по предоставлению товаров или услуг нацелены на увеличение прибыли за счет повышения уровня востребованности целевой системы у потребителей. Для достижения данной цели подобные сервисы предлагают персонализированные подборки объектов интереса, которые отвечают заданным требованиям. Задача формирования релевантных списков решается путем создания рекомендательных систем, прогнозирующих предпочтения потребителей на основе коллективных паттернов поведения.

Информация, необходимая для составления списка рекомендаций, может быть получена явно, путем сбора оценок пользователей, или неявно, т.е. на основе его взаимодействия с системой. Кроме того, анализируются персональные данные потребителей, с последующей их классификацией по группам, для формирования прогнозов. В то же время данный подход имеет отрицательный аспект, заключающийся в однообразии подборок, наиболее подходящих под запрос, в следствии чего возникает проблема информационного пузыря, суть которой состоит в постоянно представляющейся одного и того же результата, в случае повторного обращения к системе с аналогичным запросом.

Данный вопрос свойственен в основном крупным информационным системам с постоянно пополняющейся базой объектов интереса и с неизменяемыми предпочтениями субъектов. Актуальность решения проблемы информационного пузыря обусловлено тем, что пользователь ограничивает себя определенного вида товарами или услугами, что в конечном итоге приводит к потере интереса к различным типам контента целевой платформы, а следовательно, к потере потенциальных клиентов.

Объект исследования: объект пользовательского интереса (туристические маршруты, события, локации).

Предмет исследования: рекомендательный сервис для решения задачи прогнозирования туристических маршрутов на основе предпочтений пользователя.

Цель исследования: разработка рекомендательной системы туристических маршрутов с целью сокращения временных затрат пользователя на поиск туристического маршрута, удовлетворяющего его требованиям, а также повышения востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее.

Задачи исследования:

- анализ критериев классификации туристических маршрутов,
- выявление методов, необходимых для классификации туристических маршрутов и используемых для формирования списка релевантных объектов интереса на основе предпочтений пользователя,
- разработка рекомендательной системы туристических маршрутов.

Актуальность (новизна) исследования: создание такого алгоритма ранжирования, позволяющего классифицировать объекты интереса на основе предпочтений пользователя и при этом решающего проблему информационного пузыря.

Практическая и научная значимость: модификация существующих алгоритмов прогнозирования посредством применения корректирующих коэффициентов для решения проблемы информационного пузыря.

Границы исследования: основной акцент ставится на изучении подходов составления рекомендаций и создании алгоритма, удовлетворяющего описанным ранее требованиям. В границы исследования не входят такие вопросы, как способы обмена данными в рекомендательной системе, а также способы хранения информации в базах данных.

1 Анализ предметной области

1.1 Анализ существующих туристических платформ

В настоящее время популярно использование туристических платформ для предоставления услуг бронирования, покупки не только туров, дорожных билетов и отелей, но и для ознакомления с перечнем наиболее известных достопримечательностей и заведений общественного питания в интересующей пользователя локации. Среди подобных сервисов наиболее востребованы являются такие, как:

- Booking.com,
- OneTwoTrip,
- Ostrovok.ru.

1.1.1 Booking.com

Booking.com – сервис, упрощающий процесс покупки авиабилетов, бронирования отелей и аренды автомобилей. Принцип работы системы основан на ранжировании объектов интереса в соответствии с запросом пользователя. При этом формирование результата происходит не только относительно явных оценок потребителей, но и на основании паттернов поведения пользователей в системе, а именно:

- общее количество просмотров деталей объекта интереса,
- общее число бронирований объекта интереса,
- число итоговых бронирований с учетом отмененных.

К преимуществам данной платформы можно отнести:

- удобный поиск необходимой информации,
- наличие рекомендательных подборок, основанных на наиболее популярных туристических направлениях,
- возможность обратной связи после получения услуги.

Отрицательными особенностями данного сервиса является отсутствие персонализированных подборок рекомендаций и механизма лексического определения близких по смыслу слов при поиске объектов интереса.

Интерфейс туристической платформы Booking.com представлен на рисунке 1.

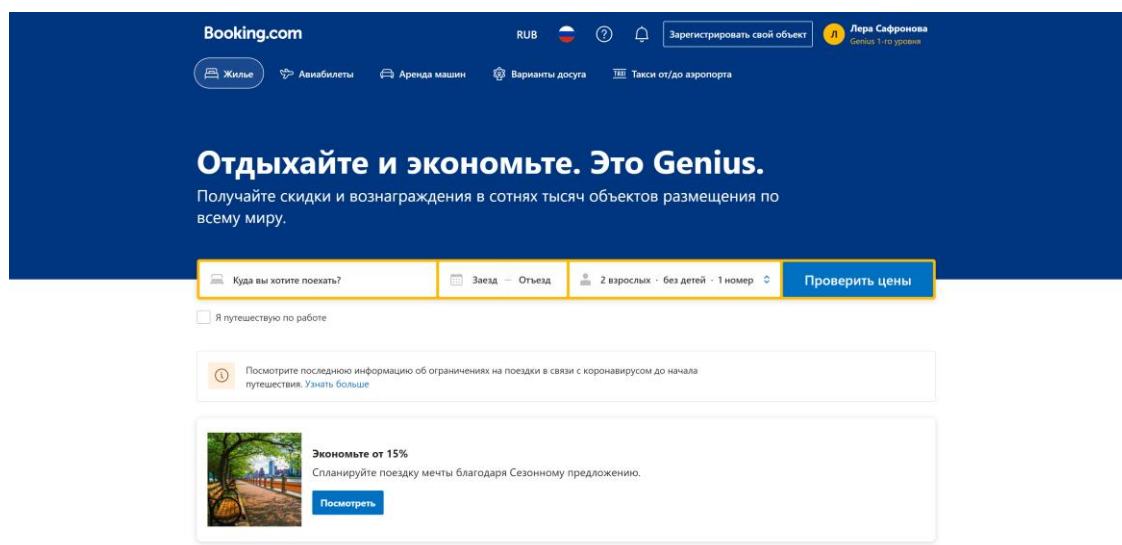


Рисунок 1 – Интерфейс платформы Booking.com

1.1.2 OneTwoTrip

OneTwoTrip – онлайн-сервис для предоставления возможности бронирования временного жилья, железно-дорожных, авиа, а также автобусных билетов. К положительным особенностям данной туристической системы относят:

- рейтинг рейсов, подходящих по запросу пользователя,
- агрегация пакетных предложений туристических маршрутов,
- релевантные подборки общественных мест, основанные на геолокации пользователя в текущий момент,
- рекомендательные списки локаций в рамках наиболее популярных направлений.

Существенным недостатком данной платформы является отсутствие рекомендательных прогнозов на основе предпочтений пользователя.

Интерфейс онлайн-сервиса бронирования билетов OneTwoTrip представлен на рисунке 2.

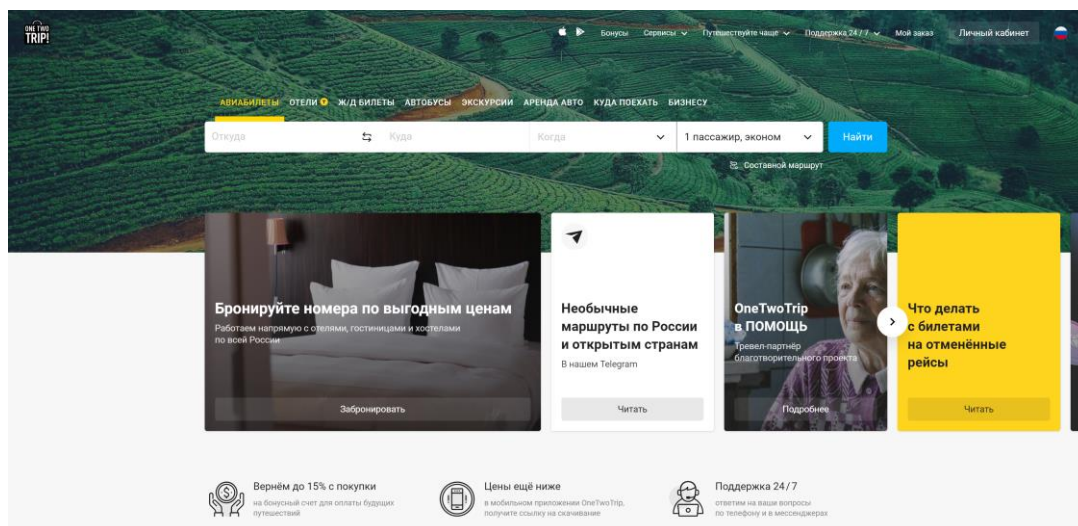


Рисунок 2 – Интерфейс платформы OneTwoTrip

1.1.3 Ostrovok.ru

Ostrovok.ru – российская система, нацеленная на быстрый поиск и онлайн бронирования отелей по всему миру. Данный сервис является классическим примером подобных платформ и имеет узкую направленность на подбор временного жилья. При этом ключевой особенностью Ostrovok.ru являются различные способы аренды: резервирование посредством предоплаты, бронирование без оплаты и полная оплата жилья. Дополнительной возможностью сервиса является наличие блога, содержащего полезную информацию о путешествиях.

К недостаткам системы можно отнести отсутствие каких-либо рекомендательных подборок для привлечения внимания пользователей к целевой системе, а так же дополнительных возможностей покупки проездных билетов и трансфера.

Пример интерфейса Ostrovok.ru представлен на рисунке 3.

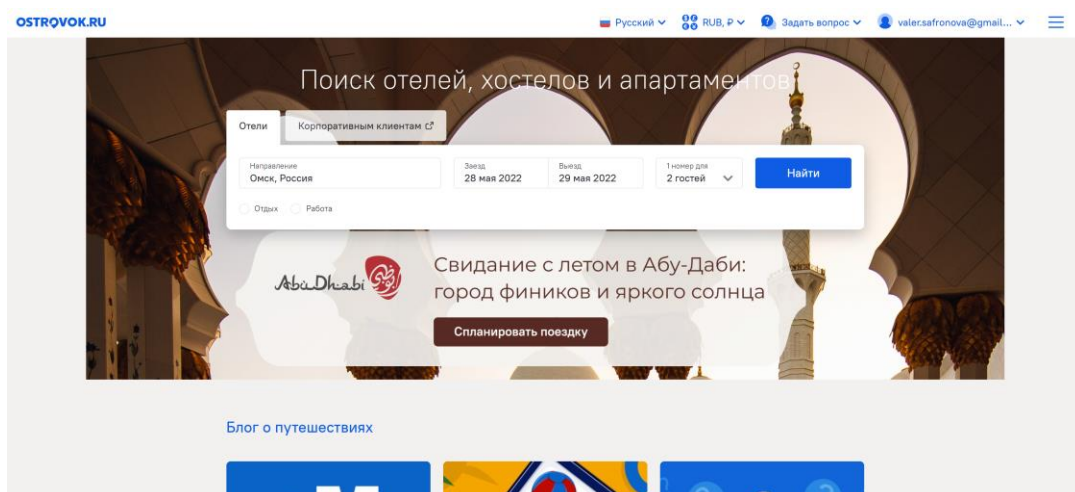


Рисунок 3 – Интерфейс платформы Ostrovok.ru

Проведя анализ существующих туристических платформ, была выявлена одна общая закономерность – отсутствие персонализированных подборок, которые могли бы не только упростить поиск интересующей информации, а соответственно, сократить временные затраты потребителя, но и повысить востребованность целевых платформ, а следовательно, увеличить потенциальную прибыль поставщика услуг.

1.2 Анализ существующих подходов составления рекомендаций

Современные подходы формирования прогнозов, представленные на рисунке 4, можно разделить на две базовые группы:

- 1) алгоритмы, основанные на данных, в которых выявляется уровень подобия объектов и субъектов системы, базируясь на имеющейся исходной информации,
- 2) алгоритмы, основанные на моделях, в которых создается, а затем обучается модель пользователя, объектов интереса и взаимосвязей между ними [1].

Алгоритмы первой группы в сравнении со второй являются наименее точными при формировании релевантных подборок, требуют накопления большого объема целевой информации, отсутствие которой приводит к невозможности составления списка рекомендаций [2, 3]. На основе этого, для выполнения данной работы были рассмотрены алгоритмы второй группы.

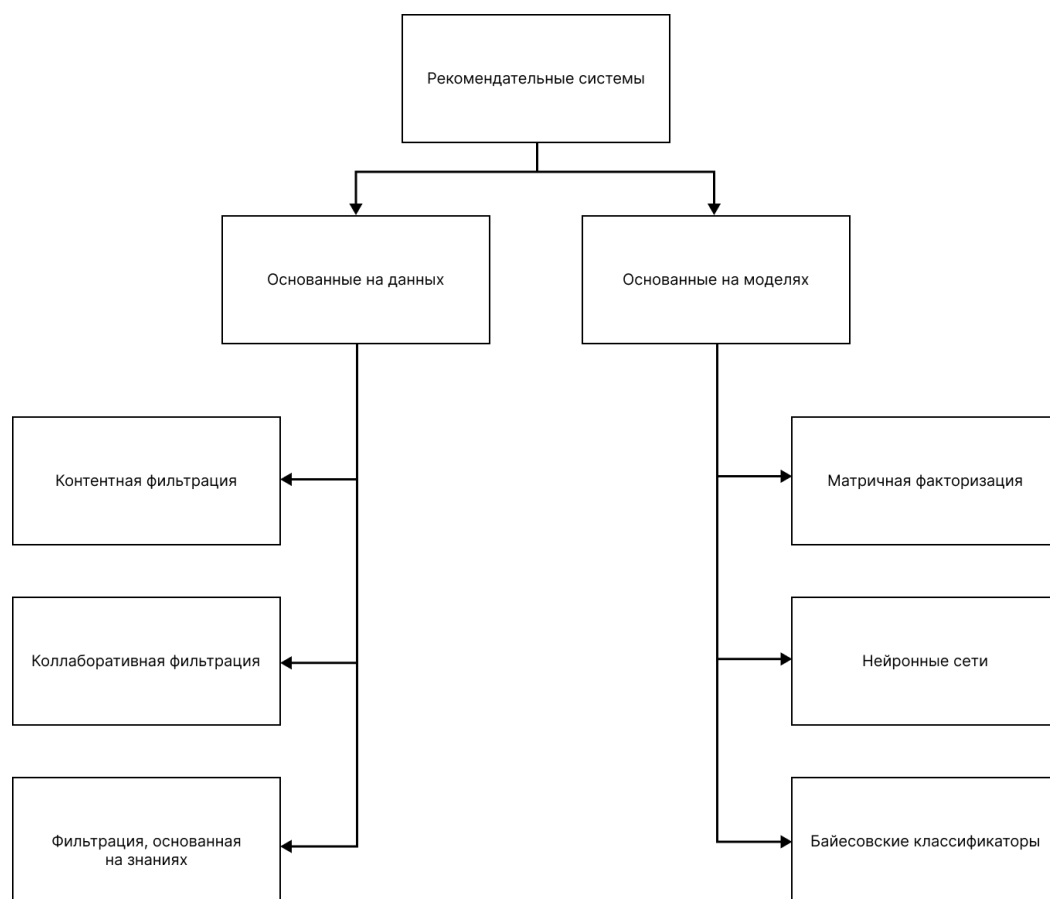


Рисунок 4 – Современные подходы формирования прогнозов

Подходы, основанные на моделях, при составлении рейтинговых прогнозов используют накопленную информацию для формирования модели и ее дальнейшего обучения. Модели в данном подходе базируются на коллективном поведении пользователей информационной системы, что приводит к созданию поведенческих паттернов, представленных в компактной форме. Результатом обучения модели является функция, позволяющая получить рекомендательный прогноз, исходя из уже имеющихся оценок. При поступлении новых данных модель модернизируется для повышения точности рекомендаций. При этом весь процесс использования модели разбивается на ресурсоемкую оптимизацию параметров и структуры модели (обучение) и не требовательный к ресурсам прогноз рейтингов в реальном времени [4].

Среди наиболее широко используемых методов формирования рекомендаций на основе моделей выделяют: алгоритмы матричной факторизации, нейронные сети и байесовские классификаторы.

1.2.1 Матричная факторизация

В последнее время модели матричной факторизации, являющиеся классом алгоритмов коллаборативной фильтрации, приобрели популярность в рекомендательных системах ввиду своей высокой точности и масштабируемости.

Методы редукции размерности, нацеленные на решение проблем с разреженностью в базах данных рекомендательных систем, так же основаны на матричной факторизации, поскольку данный подход наиболее полезен при обработке больших объемов данных. Данный механизм позволяет ускорить вычисления путем понижения размерности.

Модели факторизации матриц формируют содержательное описание объектов интереса пользователей в терминах латентных признаков, что частично решает проблему ограниченного покрытия.

Алгоритмы матричной факторизации работают путем разложения матрицы взаимодействия пользователя и объекта интереса A на произведение двух матриц меньшей размерности, а именно на отдельные двумерные массивы пользователей U и матрицы объектов интереса V . Разложение исходной матрицы происходит с применением алгоритма ранговой факторизации, схематично представленной на рисунке 5.

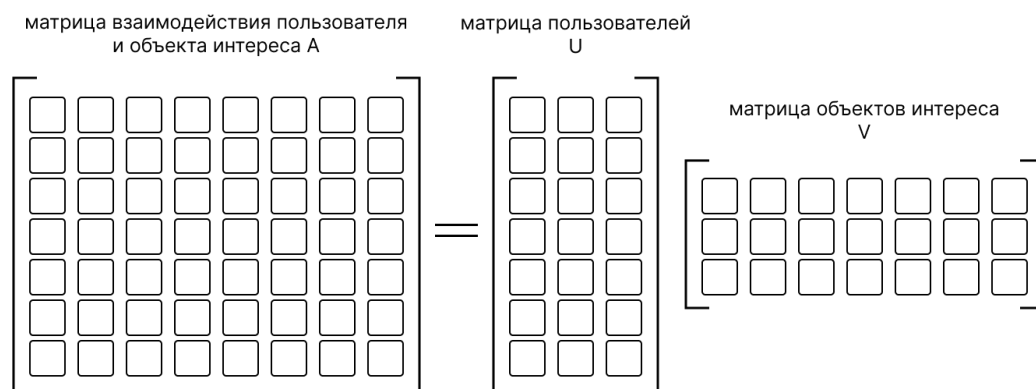


Рисунок 5 – Разложение матрицы рекомендаций

Релевантность определенного объекта интереса для конкретного пользователя рассчитывается как скалярное произведение полученных матриц.

Для повышения эффективности, а также уменьшения трудоемкости вычислений при большом наборе данных используется метод сингулярной декомпозиции значений (SVD) [5].

Методы SVD обеспечивают хорошие результаты прогнозирования за счет выявления латентных семантических факторов. Применение данного механизма в рекомендательной системе позволяет предсказать оценки, которые могут выставить пользователи конкретному товару или услуге.

Теорема сингулярного разложения значений подразумевает, что любую матрицу A , имеющую размеры $n \times m$ и ранг $r = \text{ran}(A) \leq \min\{n, m\}$, можно разложить на произведение трех матриц U , Σ и V^T :

$$A_{n \times m} = U_{n \times m} * \Sigma_{n \times m} * V_{n \times m}^T, \quad (1)$$

где $U \in \text{Mat}(n, r)$ – матрица пользователей, столбцы которой представляют собой ортогональный базис столбцов матрицы A ;

$V \in \text{Mat}(m, r)$ – матрица объектов интереса столбцы которой представляют собой ортогональный базис строк матрицы A ;

$\Sigma \in \text{Mat}(r, r)$ – диагональная матрица, элементы которой отсортированы по убыванию и не являются нулевыми.

Метод факторизации с урезанным набором данных базируется на таком преобразовании ранга исходной матрицы, что $d < r$. Тогда диагональная матрица будет состоять только из d (ранг матрицы) наибольших диагональных элементов: $\Sigma_d \in \text{Mat}(d, d)$. А матрицы объектов интереса и пользователей изменятся на $U \in \text{Mat}(n, d)$ $V \in \text{Mat}(m, d)$.

Соответственно, при вычислении релевантности товаров или услуг как скалярного произведения матриц объектов интереса и пользователей будет иметь вид:

$$\tilde{A}_{n \times m} = U_{n \times d} * \Sigma_{d \times d} * V_{d \times m}^T. \quad (2)$$

Выбор ранга матрицы d оказывает огромное влияние на точность прогнозов рекомендательной системы. Данное число должно быть, и в меру маленьким ввиду необходимости ускорения вычисления, и достаточно большим, чтобы в матрицу были включены важные взаимосвязи между объектами интереса и потребителями.

Исходя из этого, можно сделать вывод, что использование ограниченного набора наиболее схожих объектов и субъектов системы, а также замена значений элементов матрицы объектов интереса и пользователей меньшего ранга не только упрощает вычисления, но и повышает эффективность прогнозирования ввиду уменьшения влияния факторов переобучения модели.

К достоинствам использования моделей матричной факторизации в рекомендательных системах можно отнести:

- уменьшение сложности исходной задачи,
- обеспечение компактного представления данных взаимодействия между пользователями и элементами системы на основе поведенческих паттернов.

Несовершенство данного механизма заключается в погрешностях прогнозов, обусловленных наличием неявного взаимодействия пользователя с объектами интереса.

1.2.2 Нейронные сети

Помимо традиционных эвристик, которые основаны на механизме поиска информации, используются иные методы, базирующиеся на

контентной модели рекомендаций. Среди них можно выделить искусственные нейронные сети, которые являются прототипом нервной системы человека, состоящей из нейронов. Биологические нейроны – это чувствительные клетки, которые обрабатывают, хранят и передают информацию с помощью электрических и химических сигналов. На разветвленных отростках нейрона, называемыми дендритами, располагаются синапсы, являющиеся точками контакта. Изменение силы синаптических связей между нейронами под действием внешних раздражителей приводит к приобретению человеком новых навыков.

Искусственные нейронные сети имитируют биологический процесс обучения. Нейроны в данном случае – это отдельные узлы, являющиеся единицами вычислений, которые получают входные данные от других нейронов, производят обработку полученной информацией и передают ее остальным.

Аналогично изменению силы синаптических связей между биологическими нейронами, изменение веса в искусственных нейронных сетях приводит к тому, что вычислительная функция может быть изучена, а следовательно, приобретен новый навык. Внешним раздражителем в данном случае выступает массив данных, который необходимо обучить. Исходя из этого, искусственный нейрон можно представить как некоторую нелинейную функцию с единственным аргументом – линейной комбинации входных сигналов. Обучение модели состоит в постепенном изменении веса при неверном предсказании с текущим набором весов [6].

Существует множество архитектур, используемых для организации связи между узлами, начиная от простого однослойного персептрона и заканчивая сложными многослойными нейронными сетями. Выбранная архитектура является залогом эффективности работы нейронной сети.

Искусственная нейронная сеть состоит из нескольких слоев, которые можно разделить на три типа: входные, скрытые и выходные. Входные слои

реагируют на исходные данные, поступившие в сеть, скрытые блоки получают взвешенный выходной сигнал от входных блоков, выходные узлы, в свою очередь, реагируют на взвешенный выход из скрытых блоков и генерируют конечные выходные сигналы.

Наиболее распространенным подходом является использование базовой модели персептрона. Модель персептрона – это линейный классификатор, имеющий простой и эффективный алгоритм обучения. В данном случае используется два слоя узлов, где сигналы распространяются строго одним способом: от входа к выходу. Входные узлы, число которых равно количеству исходных данных, получают и передают один числовой атрибут выходному узлу, который в базовой модели персептрона является единственным узлом, выполняющим математическую функцию на своих входах.

Формальное описание искусственного нейрона представлено на рисунке 6, где x – входные данные, w – вес синаптической связи, S – текущее состояние нейрона, Y – выходные данные, вычисляемые функцией.

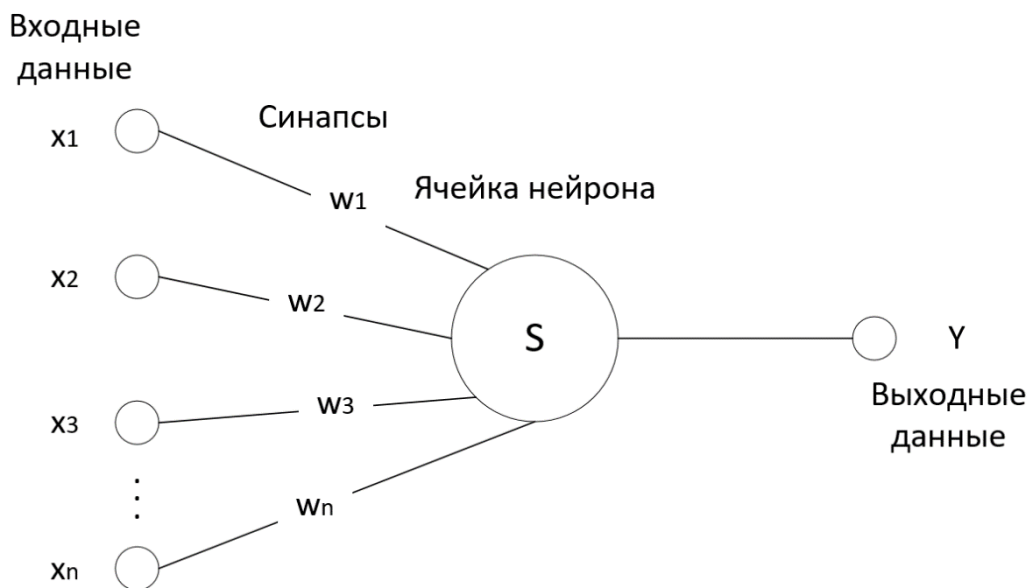


Рисунок 6 – Формальное описание искусственного нейрона в модели персептрона

Математический аппарат нейрона включает в себя:

- синапсы, характеризующиеся своим весом, осуществляют связь между нейронами, умножают входной сигнал на весовой коэффициент синапса,
- сумматор, аналог тела клетки нейрона. Выполняет сложение внешних входных сигналов, поступающих по синаптическим связям от других нейронов. Определяет уровень возбуждения нейрона,
- функция активации, определяет окончательные выходные сигналы.

Обучение в персептроне осуществляется путем изменения весов связей, соединяющих входные узлы с выходными узлами, когда предсказанная метка не соответствует истинной метке [7].

Пусть, $\bar{W}$ – веса для соединений n различных входов с выходным нейроном для записи данных размерности n :

$$\bar{W} = (w_1 \dots w_n) . \quad (3)$$

Кроме того, смещение n связано с функцией активации. Результат сложения $z_i \in \{-1, +1\}$ внешних входных сигналов $(x_i^1 \dots x_i^n)$ из i -записи данных $\bar{X}_i$ выглядит следующим образом:

$$z_i = \text{sign}\left\{ \sum_{j=1}^n w_j * x_i^j + n \right\} = \text{sign}(\bar{W} * \bar{X}_i). \quad (4)$$

Значение z_i представляет собой прогноз персептрона для входного параметра $\bar{X}_i$. Однако, для обучения модели необходимо произвести подбор весов таким образом, чтобы значение z_i было равно значению y_i активационной функции.

При предсказании $(z_i - y_i)$ с текущим набором весов результат может принимать любое из значений $-2, 0, +2$. Цель алгоритмов нейронной сети

состоит в том, чтобы узнать вектор весов $\bar{W}$ и смещение p , так, что z_i приближено к истинному результату y_i .

Алгоритм персептрона начинается с случайного вектора весов, затем механизм последовательно заменяет элементы входных данных для создания следующего прогноза. Затем веса обновляются, исходя из значения предсказания. Обновление вектора весов $\bar{W}$ на t -итерации выглядит следующим образом:

$$\bar{W}^{t+1} = \bar{W}^t + \eta * (y_i - z_i) * \bar{X}_i, \quad (5)$$

где η – скорость обучения.

Значение η прямо пропорционально быстрым темпам обучения и обратно пропорционально оптимальности результирующих прогнозов. В действительности значения η в начале обучения выбирается наибольшим, постепенно уменьшая, поскольку результат сложения входных данных на выбранных весах приближается к своим наиболее благоприятным значениям. Идея данного подхода состоит в том, что высокие темпы обучения полезны на ранних стадиях, однако могут привести к колебаниям между решениями на более поздних этапах.

Модель персептрона является базовой, однако она не всегда применима в сложных системах. В таком случае используются многослойные искусственные нейронные сети. Такие сети в своей архитектуре имеют скрытые слои помимо входных и выходных блоков. Сам скрытый слой, в свою очередь, может состоять из множества составных частей, в которых узлы связаны между собой. Многослойные нейронные сети не ограничиваются использованием линейных функций входных сигналов. Они могут использоваться в различных узлах скрытых слоев. В таком случае, вычисление сложения внешних сигналов выглядит следующим образом:

$$z_i = \sum_{j=1}^n w_j * \frac{1}{1 + e^{x_i^j}} + n. \quad (6)$$

В данном случае z_i не является прогнозируемым, если оно вычисляется в узлах скрытого слоя, так как эти данные используются в дальнейшем.

При использовании многослойных нейронных сетей возникает проблема неизвестности выходных данных в скрытых слоях, поскольку обучающие метки этих узлов не определены. Из чего появляется необходимость составления корректного алгоритма обновления веса узлов в случае несовпадения результата сложения внешних входных сигналов и значения активизирующей функции. Проблема решается с помощью алгоритма обратного распространения.

В прямой фазе данного алгоритма входные данные подаются в нейронную сеть, происходит вычисления по слоям, используя текущий набор весов. Прогнозируемый результат сравнивается с меткой обучающего экземпляра для проверки их схожести. Далее, в обратной фазе, определяется погрешность вычислений в скрытом слое как функция оценок ошибок и весов узлов в предыдущем слое. Затем она используется при вычислении погрешности относительно весов в узле и их обновления. Фактическое уравнение обновления весов в данном случае не очень отличается от базового персептрона.

Нейронные сети применяются в рекомендательных системах в двух вариациях:

- 1) механизм формирования рекомендаций,
- 2) промежуточный этап при построении моделей матричной факторизации.

В первом случае в качестве исходных данных предоставляются список объектов интереса и пользовательские оценки. Во время обучения модели вычисляемый результат сопоставляется с реальными предпочтениями, и в

конечном итоге формирует принцип, на основе которого осуществляется составление рейтинговых прогнозов.

Основу второго подхода составляет матричная факторизация, роль нейронной сети в данном случае второстепенна. На начальном этапе происходит разложение исходной матрицы на двумерные массивы меньшей размерности. Затем, полученные матрицы объединяются и подаются на вход в нейронную сеть. После чего, происходит стандартное обучение нейронной сети, целью которого является получение сложной нелинейной модели для факторизации матрицы.

На основе описанного механизма формирования рекомендаций при помощи нейронных сетей можно сделать вывод, что преимуществом их использования является высокий уровень эффективности, обусловленной работой даже в случае отказа некоторых фрагментов.

К недостаткам данного подхода можно отнести:

- сложность проектирования архитектуры сети,
- обучение сети в ряде случаев приводит к переобучению,
- продолжительные временные затраты на выполнение процесса обучения.

1.2.3 Байесовские классификаторы

Еще одним методом, базирующимся на контентной модели рекомендаций, являются байесовские классификаторы, представляющие собой вероятностную структуру, основанную на теореме Байеса и определения условной вероятности для решения задач классификации. К ним относят анализ данных и распознавание образов, требующих построения функции, которая присваивает метку класса экземплярам с собственным набором атрибутов. Модели байесовских классификаторов достаточно просты и эффективны при работе с большим массивом входных данных [8].

Как и нейронные сети, байесовский классификатор строится на основе обучения с учителем, когда вычисляемое значение сравнивается с меткой

обучающего экземпляра. Частным случаем данного метода является наивный байесовский классификатор. Ключевая особенность данного механизма заключается в том, что каждый параметр классифицируемых данных рассматривается независимо от других, т.е. наличие или отсутствие одного атрибута не влияет на существование остальных. Благодаря данной специфике разреженность входных данных существенно не повлияет на рекомендательные прогнозы. Однако на практике анализируемые параметры не являются полностью независимыми. В таком случае используются байесовские сети, которые включают в себя ориентированный ациклический граф для кодирования зависимостей между атрибутами и таблицей вероятности, связывающей узлы между собой. Вершины данного орграфа соответствуют атрибутам, а ребра связывают данные переменные. Совместное условное распределение использует операторы независимости и определяется произведением локальных условных распределений, заданных в каждом узле. Полученная графическая модель дает полное представление о взаимосвязях всех компонентов системы.

Байесовский классификатор представляет собой вероятностную модель, где атрибуты объектов и метка класса рассматриваются как дискретные величины. Он запоминает все признаки, а также вероятности того, что конкретный атрибут входит в конкретную классификацию (класс). Набор данных поступает в классификатор последовательно, после выполнения проверки наличия атрибутов в конкретном классе происходит обновление массива объектов интереса. Таким образом происходит обучение контентной модели.

Вероятностный классификатор предсказывает класс C_i с самой большой условной вероятностью P_i для заданного вектора признаков $x = (x_1 \dots x_n)$:

$$P(C_i | x_1 \& \dots \& x_n) = \frac{P(x_1 | C_i) * \dots * P(x_n | C_i) * P(C_i)}{P(x_1) * \dots * P(x_n)} . \quad (7)$$

Вероятность класса C_i находится на основе параметров $(x_1 \dots x_n)$, подразумевая, что признаки являются независимыми, и следовательно, приведенная выше вероятность пропорциональна:

$$P(C_i) \prod_{np} P(x_i | C_i). \quad (8)$$

Исходя из этого, простой байесовский классификатор можно рассматривать как функцию, которая каждому выходному значению модели присваивает метку класса, т.е. $y = C_i$ следующим образом:

$$y = \arg \max \prod_{np} P(x_i | C_i). \quad (9)$$

Из этого следует, что условная вероятность P_i , вычисленная для каждого класса C_i и признака x_i , присваивается к имеющему большую вероятность классу C_i . Результатом обучения контентной модели является список атрибутов вместе с условными вероятностями.

Несмотря на предположение, что признаки являются независимыми, байесовский классификатор обеспечивает высокую точность классификации.

Использование данного механизма в рекомендательных системах обусловлено преимуществом использования данного подхода, базирующегося на контентной модели рекомендаций, в том, что разреженность базы данных, а также наличие нерелевантных атрибутов игнорируются при вычислении вероятностной оценки.

К отрицательным сторонам можно отнести следующие аспекты:

- отсутствие в обучающем наборе данных некоторого признака приведет к его потере при формировании рекомендаций, поскольку, исходя из политики классификатора, анализируются только известные данные (явление известно под названием «нулевая частота»),

- значения спрогнозированных вероятностей не всегда являются достаточно точными,
- на практике допущение о полностью независимых признаках недопустимо.

1.3 Опыт использования рекомендательных механизмов

С появлением персонализации контента возникла проблема информационного пузыря, заключающаяся в предоставлении информации, базирующейся на таком поведении пользователя, как:

- поисковые запросы,
- просмотр характеристик объекта интереса,
- добавление объекта интереса в избранное,
- непосредственная оценка объекта интереса,
- покупка / бронирование объекта интереса.

Причем отслеживание пользовательских действий происходит не только внутри целевой платформы, но и вне ее. Сбор данной информации улучшает алгоритмы формирования релевантных подборок, с одной стороны, и делает их однообразными с другой [9, 10].

Говоря о системах, оперирующих новостными блогами, проблема информационного пузыря приводит к искажению восприятия действительности за счет просмотра только тех новостей, которые подходят пользователю по его убеждениям, тем самым ограничивая его круг зрения.

Применительно к туристическим платформам данная проблема отражается в ограничении целевой системой определенного типа маршрутами или местами, не добавляя объекты интереса другой категории. В результате чего снижается уровень востребованности у конкретной локации для текущего пользователя, а следовательно, понижается потенциальная прибыль поставщика услуг.

1.4 Выводы по разделу

В результате анализа существующих онлайн-сервисов организации путешествий и возможных механизмов прогнозирования, был сделан вывод о необходимости создания рекомендательной системы туристических маршрутов, которая позволяла бы ориентироваться на предпочтения пользователя, а также добавляла бы заведомо нерелевантные или не просмотренные ранее объекты интереса с целью решения последствий «пузыря фильтров».

2 Проектирование рекомендательной системы

Туристическая система решает задачу выработки рекомендаций туристических активностей на основе предпочтений пользователей. Прогнозирование базируется на модели предметной области, которая основана на данных об объектах интереса и информации о пользовательском поведении в системе, полученных от туристической платформы в определённом формате [11]. Схема обработки действий пользователей для составления прогнозов представлена на диаграмме кооперации (рисунок 7).

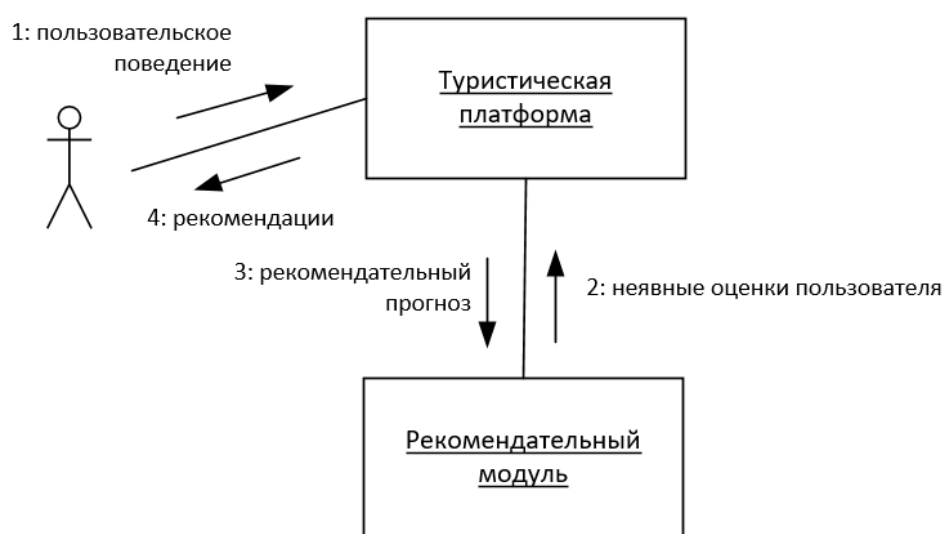


Рисунок 7 – UML диаграмма кооперации

Целевая туристическая платформа направлена на развитие внутреннего туризма, предоставляя пользователям обширный перечень маршрутов, актуальных событий, туристических пакетов и разнообразных локаций. Функционал системы позволяет осуществлять поиск, добавление в избранное, а также бронирование и покупку объектов интереса по следующим категориям:

- туры,
- места,
- рестораны,
- события,

- маршруты.

Объекты интереса, в зависимости от типа, имеют свои характеристики, позволяющие получить полную информацию о конкретной активности. Некоторые из них являются универсальными, т.е. не зависят от категории, в которую входит объект. К ним относят:

- наименование,
- описание,
- список изображений, включая картинку предварительного просмотра,
- местоположение.

Другие параметры характерны только для определенного типа. Объекты интереса, относящиеся к категории «туры», помимо общих характеристик, содержит информацию о параметрах тура, включающих в себя тариф, сезонность, возрастные ограничения, квоту на количество человек, длительность и рейтинг места проживания. Кроме того, данная активность предоставляет данные о программе тура, дополнительных условиях и перечне услуг, входящих в стоимость.

Активности категории «места» предоставляют данные о режиме работы, примерном времени проведения, преysкурant на доступные билеты, точную геолокацию, а также места, находящиеся рядом с данным объектом интереса.

Информация о ресторанах содержит тип и кулинарные направления, средний чек, режимы работы и среднее время пребывания в данном месте, а также заведения, располагающиеся неподалеку.

К данным категории «события» относят приблизительную длительность события, стоимость билетов и расписание сеансов.

Объекты интереса категории «маршруты» содержат параметры, характеризующие продолжительность маршрута, его тип и состав, стоимость посещения, точную геолокацию, а также количество заинтересованных путешественников.

Обобщая имеющуюся информацию о данных, хранящихся в системе и вариантах ее использования, наиболее подходящим для составления рекомендаций был выбран алгоритм матричной факторизации, а именно факторизация матрицы предпочтений клиентов при помощи SVD разложения, поскольку он обладает высокой эффективностью прогнозирования при большом наборе данных за счет понижения размерности, а также характеризуется компактным представлением информации о взаимодействии между пользователями и элементами системы на основе поведенческих паттернов.

2.1 Исходные данные для построения системы рекомендательных сервисов

Для построения рекомендательной системы необходимы данные, на основе которых будет осуществляться прогнозирование. Для формирования рекомендаций объектов интереса конкретного типа в туристической платформе запрашивается следующая информация:

- список туров (tours),
- список ресторанов (restaurants),
- список городов (cities),
- список маршрутов или пакеты (route),
- список мест (places),
- список событий (events),
- список подборок (collections),
- список тегов (tags).

Исходные данные можно представить в виде реляционной модели, состоящей из следующего набора связанных между собой отношений (таблиц):

- мест (Places),
- событий (Events),
- типов событий (Event_types),

- ресторанов (Restaurants),
- кулинарных направлений (Cuisines),
- партнеров (Partner_types),
- городов (Cities),
- регионов (Regions),
- тегов (Tags),
- туров (Tours),
- маршрутов (Routes),
- пользователей (Users).

Подробное описание атрибутов таблицы «Places» представлено в таблице 1.

Таблица 1 – Структура таблицы «Places»

Атрибут	Описание атрибута
id	идентификатор места
title	наименование места
address	адрес места
coordinates	координаты места
district_ids	идентификаторы районов
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
tags_ids	идентификаторы тегов
metro_ids	идентификаторы метро

Подробное описание атрибутов таблицы «Events» представлено в таблице 2.

Таблица 2 – Структура таблицы «Events»

Атрибут	Описание атрибута
id	идентификатор события
title	наименование события
place_ids	адреса места
tags_ids	идентификаторы тегов

Продолжение таблицы 2

Атрибут	Описание атрибута
city	идентификаторы города
images	список картинок данного места
image_preview	картинка предварительного просмотра места
image_main	основная картинка места
cg_announcement	рекламный слоган события
link_source	ссылка на событие
ticket_price	стоимость билета
duration	длительность события
event_types_ids	идентификаторы типов событий
sort	номер сортировки

Подробное описание атрибутов таблицы «Events_types» представлено в таблице 3.

Таблица 3 – Структура таблицы «Events_types»

Атрибут	Описание атрибута
id	идентификатор типа события
title	наименование типа события
images	список картинок данного типа события

Подробное описание атрибутов таблицы «Restaurants» представлено в таблице 4.

Таблица 4 – Структура таблицы «Restaurants»

Атрибут	Описание атрибута
id	идентификатор ресторана
title	наименование ресторана
address	адрес ресторана
avg_time_visit	режим работы
city	идентификатор города
coordinates	координаты ресторана
can_reserve	возможность бронирования
cuisines_ids	идентификаторы кулинарных направлений
images	список картинок данного ресторана
image_preview	картинка предварительного просмотра ресторана

Продолжение таблицы 4

Атрибут	Описание атрибута
image_main	основная картинка ресторана
images360	картинки меньшего размера
partner_type_id	идентификатор партнера
rest_id	идентификатор типа объекта интереса
screen	тип объекта интереса
short_description	описание ресторана
sort	номер сортировки
url	ссылка на объект интереса

Подробное описание атрибутов таблицы «Cuisines» представлено в таблице 5.

Таблица 5 – Структура таблицы «Cuisines»

Атрибут	Описание атрибута
id	идентификатор кулинарного направления
title	наименование кулинарного направления

Подробное описание атрибутов таблицы «Partner_types» представлено в таблице 6.

Таблица 6 – Структура таблицы «Partner_types»

Атрибут	Описание атрибута
id	идентификатор партнера
title	наименование партнера
images	список картинок данного партнера

Подробное описание атрибутов таблицы «Cities» представлено в таблице 7.

Таблица 7 – Структура таблицы «Cities»

Атрибут	Описание атрибута
id	идентификатор города
list_image	список картинок города
rating	рейтинг города
region	регион
short_description	описание города

Продолжение таблицы 7

Атрибут	Описание атрибута
title	наименование города

Подробное описание атрибутов таблицы «Regions» представлено в таблице 8.

Таблица 8 – Структура таблицы «Regions»

Атрибут	Описание атрибута
id	идентификатор региона
title	наименование региона

Подробное описание атрибутов таблицы «Tags» представлено в таблице 9.

Таблица 9 – Структура таблицы «Tags»

Атрибут	Описание атрибута
id	идентификатор тега
title	наименование тега
count	количество объектов интереса с данным тегом
images	список картинок данного тега

Подробное описание атрибутов таблицы «Tours» представлено в таблице 10.

Таблица 10 – Структура таблицы «Tours»

Атрибут	Описание атрибута
id	идентификатор тура
city	идентификаторы города
days	количество дней
images	все картинки данного тура
image_preview	картинка предварительного просмотра тура
image_main	основная картинка тура
main_group_count	ограничение на состав группы
nights	количество ночей
price	стоимость тура
screen	тип объекта интереса
season_end	конец сезона

Продолжение таблицы 10

Атрибут	Описание атрибута
season_start	начало сезона
short_description	описание тура
sort	номер сортировки
title	наименование тура
url	ссылка на объект интереса

Подробное описание атрибутов таблицы «Routes» представлено в таблице 11.

Таблица 11 – Структура таблицы «Routes»

Атрибут	Описание атрибута
id	идентификатор маршрута
activity_count	количество точек остановки
billing_product_id	платежный идентификатор
city	идентификаторы города
description	описание маршрута
image_preview	картинка предварительного просмотра маршрута
image_main	основная картинка маршрута
information_pages_ids	идентификаторы информационных блоков
packet_price	стоимость маршрута
route	характеристики маршрута
route_tags_ids	теги маршрута
rp_price_id	идентификатор получателя
service_count	количество услуг
short_description	краткое описание маршрута
sort	номер сортировки
title	наименование маршрута
time	продолжительность маршрута

Таблицы «Places», «Events» и «Routes» связаны с отношением «Tags» по атрибутам «tags_ids», «tags_ids» и «route_tags_ids» соответственно как один ко многим.

Отношения «Events» и «Event_types», «Events» и «Places» связаны по атрибутам «event_types_ids» и «place_ids» соответственно как один ко многим.

Таблицы «Tours», «Routes», «Restaurants» и «Events» связаны с отношением «Cities» по атрибуту «city» как один ко многим. Отношение «Cities», в свою очередь, связано с таблицей «Regions» по атрибуту «region» как один к одному.

Отношение «Restaurants» связано с таблицами «Cuisines» и «Partner_types» по атрибутам «cuisines_ids» и «partner_type_id» соответственно как один ко многим.

Взаимодействие между описанными выше отношениями, а также их атрибутами, представлено на рисунке 8.

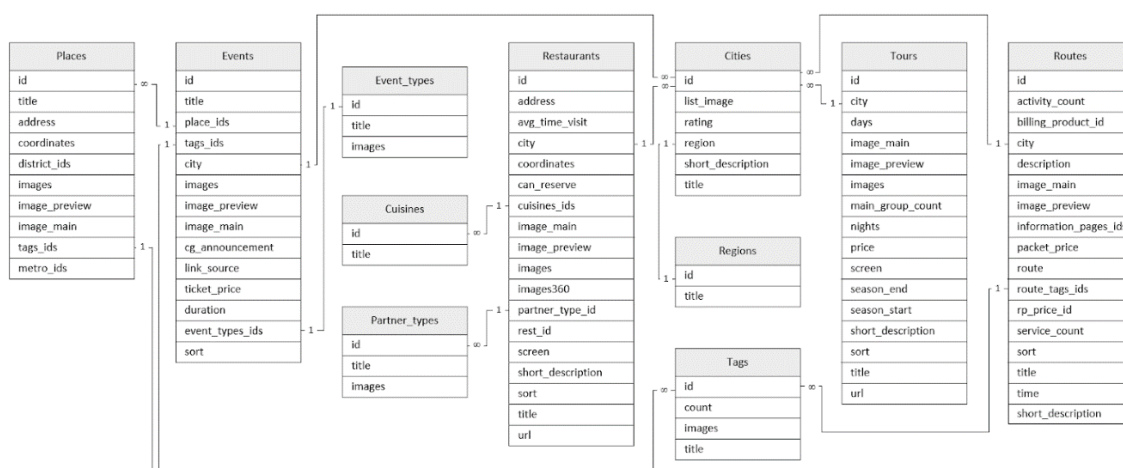


Рисунок 8 – Реляционная модель исходных данных

Отношение «Relationships» связано с таблицей «Users» и таблицами «Places», «Events», «Routes», «Tours» и «Restaurants» по атрибутам «uid» и «iid» соответственно как один ко одному.

Подробное описание атрибутов таблицы «Users» представлено в таблице 12.

Таблица 12 – Структура таблицы «Users»

Атрибут	Описание атрибута
id	идентификатор пользователя
username	идентификатор пользователя в битриксе

Продолжение таблицы 12

Атрибут	Описание атрибута
email	почта пользователя
salt	параметр, необходимый для преобразования пароля
hashed_password	пароль пользователя
bio	Биометрические характеристики пользователя
image	аватар пользователя

Подробное описание атрибутов таблицы «Relationships» представлено в таблице 13.

Таблица 13 – Структура таблицы «Relationships»

Атрибут	Описание атрибута
iid	идентификатор объекта интереса
uid	идентификатор пользователя
rating	тип взаимодействия (оценка)

Взаимосвязи между объектами и субъектом туристической платформы можно представить в виде реляционной модели, представленной на рисунке 9.

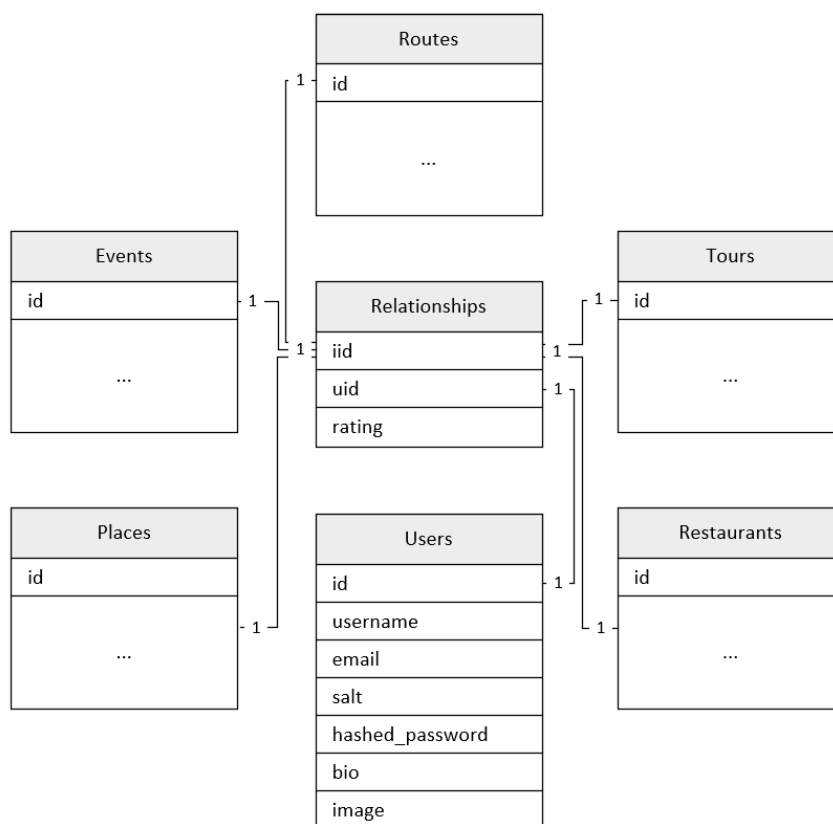


Рисунок 9 – Реляционная модель взаимосвязей между пользователями и объектами интереса

2.2 Описание выходных данных рекомендательной системы

Результатом формирования рекомендательных прогнозов должен являться список идентификаторов туристических маршрутов, отсортированных в порядке релевантности, а также дублирование номера страницы, заданного числа элементов и их общее количество.

2.3 Описание алгоритма прогнозирования

Рекомендательная система производит автоматизацию процесса обработки данных, составления рекомендаций и классификацию объектов интереса по заданным критериям.

Разработанная подсистема предусматривает возможность передачи со стороны сервиса различного типа параметров, с дальнейшей их обработкой. В данный набор характеристик может входить информация, которая не участвует при формировании рекомендательных прогнозов, однако идентифицирует текущий массив данных, необходимый для составления ответа от сервиса.

При проектировании системы рейтинговых прогнозов учитывались следующие кейсы:

- многомерная классификации данных информационной системы, в том числе обратной связи пользователей,
- отбор эталонных данных для наилучшего составления прогнозов,
- определение закономерностей в полученной информации,
- классификация объектов интереса,
- классификация групп пользователей,
- персонализированное прогнозирование.

Рекомендательный сервис предоставляет не только персонализированные рекомендации для конкретного пользователя, не подлежащие широкому распространению, но и общедоступные предложения [12]. В качестве публичных рекомендаций представлен список наиболее популярных продуктов, формируемый на основании количественных и

качественных характеристик взаимодействия пользователей с конкретным объектом интереса. При индивидуальном прогнозировании обязательным условием является авторизация потребителя в системе. Формирования рекомендаций в данном случае зависит от того, проводил ли субъект какие-то действия на портале или нет.

Рекомендательная система построена на основе алгоритмов, позволяющих решить задачу предложения объектов интереса по пользовательским предпочтениям, а также предусмотреть возможность включения в подборку заведомо нерелевантных туристических маршрутов с целью предотвращения возникновения проблемы информационного пузыря.

Прогнозирование персонализированных предложений осуществляется матричной факторизацией субъектов системы и объектов интереса, представленных в виде разреженной матрицы, где строки соответствуют потребителям, а столбцы элементам. Если между пользователем и объектом осуществлялось взаимодействие, то на месте пересечения данных компонентов в матрице, отображается соответствующая характеристика. В противном случае значение поле пересечения не определено.

Исходя из собранных данных, осуществляется построение персональной матрицы отношений пользователь-объект интереса. В эту матрицу входят схожие по предпочтениям потребители, а также оцененные ими товары.

Для выявления неопределенных значений используется модель матрицы факторизации, базирующейся на алгоритмах разложения исходного двумерного массива на матрицы меньшей размерности. Для целевой системы выбран метод сингулярной декомпозиции значений (SVD), который показывает оптимальный результат по времени обучения и предсказания за счет использования эталонного набора данных. Модель в данном случае строится по предпочтениям других пользователей в виде пользователь-продукт-оценка, при этом последняя может принимать следующие значения:

- SEARCH (1): пользователь произвел поиск объекта интереса,

- CLICK (2): пользователь перешел на страницу объекта интереса,
- DETAIL (3): пользователь просмотрел детализацию характеристик объекта интереса,
- FAVORITE (4): пользователь добавил объект интереса в избранное,
- BOOKING (5): пользователь забронировал / добавил в корзину объект интереса,
- BUY (6): пользователь осуществил покупку объект интереса.

При формировании рекомендаций необходимо учитывать не только непосредственные действия пользователя, но и дополнительные коэффициенты, формирующиеся на основании неявных процессов, происходящих в системе, поскольку они позволяют повысить уровень точности прогнозов. К подобным показателям относятся:

- отрицательные веса,
- коэффициент затухания.

Под отрицательными весами понимается массив данных, содержащий в себе информацию о степени заинтересованности пользователя в данном объекте интереса. Коэффициент, значение которого обратно пропорционально уровню востребованности текущего объекта интереса у пользователя, применяется на заключительном стадии расчета оценки релевантности (формула 10).

$$q(k) = y - d_k, \quad (10)$$

где k – текущая информация об объекте интереса относительно пользователя;
 y – список атрибутов вместе с условными вероятностями;
 d_k – текущее значение отрицательного веса данного объекта интереса.

Значение отрицательного веса может варьироваться от 0 до 1, базируясь на оценке пользователя. Исходя из описанных ранее возможных вариантов

взаимодействия, отрицательный вес может иметь одно из значений, представленных в таблице 14.

Таблица 14 – Шкала определения значения отрицательных весов

Значение отрицательного веса	Оценка пользователя
0 ($\approx 0,02$)	6 – покупка
0,18	5 – бронирование/добавление в корзину
0,36	4 – добавление в избранное
0,52	3 – просмотр детализации
0,68	2 – переход на страницу объекта интереса
0,84	1 – поиск
1	0 – отсутствие взаимодействия

Формирование массива отрицательных весов происходит на втором этапе цикла обратной связи и зависит от явных поведений пользователя, представленных ранее, и неявных процессов системы, например, от времени ожидания следующего действия субъекта относительно рекомендуемого объекта после перехода на его страницу.

Таким образом, продукты, имеющие высокий показатель отрицательного веса, оказываются в конце списка прогнозов или же вообще не попадают в него. Однако данный продукт, несмотря на высокий показатель отрицательного веса, может попасть в список рекомендаций, в случае превышения оценки релевантности над отрицательным коэффициентом, и даже находится на высоких позициях до тех пор, пока действия пользователя не снизят его востребованность до критического уровня.

Подобные рекомендации, имеющие высокий отрицательный вес и низкую оценку релевантности, всегда будут находиться в конце списка прогнозов. Исходя из этого, пользователю, повторно обратившемуся к системе с аналогичным запросом, будет предоставляться одна и та же подборка продуктов, наиболее соответствующих интересам субъекта. Во избежание подобных ситуаций применяется корректирующий механизм – коэффициент затухания, позволяющий вернуть объект интереса, имеющий высокий отрицательный вес, в выборку рекомендаций через определенный период

времени. Коэффициент затухания применяется в туристической платформе следующим образом: изменение массива отрицательных весов для данной подборки происходит спустя выполнения десять рекомендаций. При этом необходимо учитывать, что отрицательные веса не идентичны оценке пользователя, они лишь базируются на наибольшем из возможных типов взаимодействия. В дальнейшем происходит постепенное циклическое изменение значения отрицательного веса на основе показателя коэффициента затухания. В случае, если пользователь произвел действие, имеющее больший приоритет, относительно объекта интереса, происходит сброс коэффициента затухания, а отрицательный вес принимает соответствующее значение новой оценки.

Исходя из этого, применение коэффициента затухания вместе с отрицательными весами позволит решить проблему информационного пузыря рекомендаций, добавив в подборки, не просмотренные ранее или заведомо неактуальные объекты интереса.

Схема получения рекомендаций в туристической платформе представлена на рисунке 10.

Общий механизм вычисления рекомендаций на основе матричной факторизации и корректирующих коэффициентов был вынесен в отдельный модуль со следующим функционалом:

- вывод информации о начале построения модели,
- фиксация времени начала процесса,
- вывод информации о чтении данных из таблицы рейтингов,
- чтение данных (идентификатор пользователя и объекта интереса, оценка их взаимодействия) из таблицы рейтингов,
- проверка полученных данных на пустоту. В случае, если данные отсутствуют, вывод соответствующего сообщения и завершение вычислений,
- чтение списка возможных оценок,

- загрузка списка оценок и данных таблицы рейтингов в общий массив данных для построения модели,
- занесение параметров для обучения модели посредством SVD,
- вывод информации об обучении модели,
- обучение модели на полученном наборе данных,
- вычисление рекомендательных прогнозов,
- вычисление окончательного прогноза на основе коэффициента затухания,
- ранжирование рекомендательных прогнозов,
- вывод информации о записи результатов в базу данных,
- запись результатов в базу данных,
- вывод информации о завершении процесса построения рекомендаций и времени обучения.

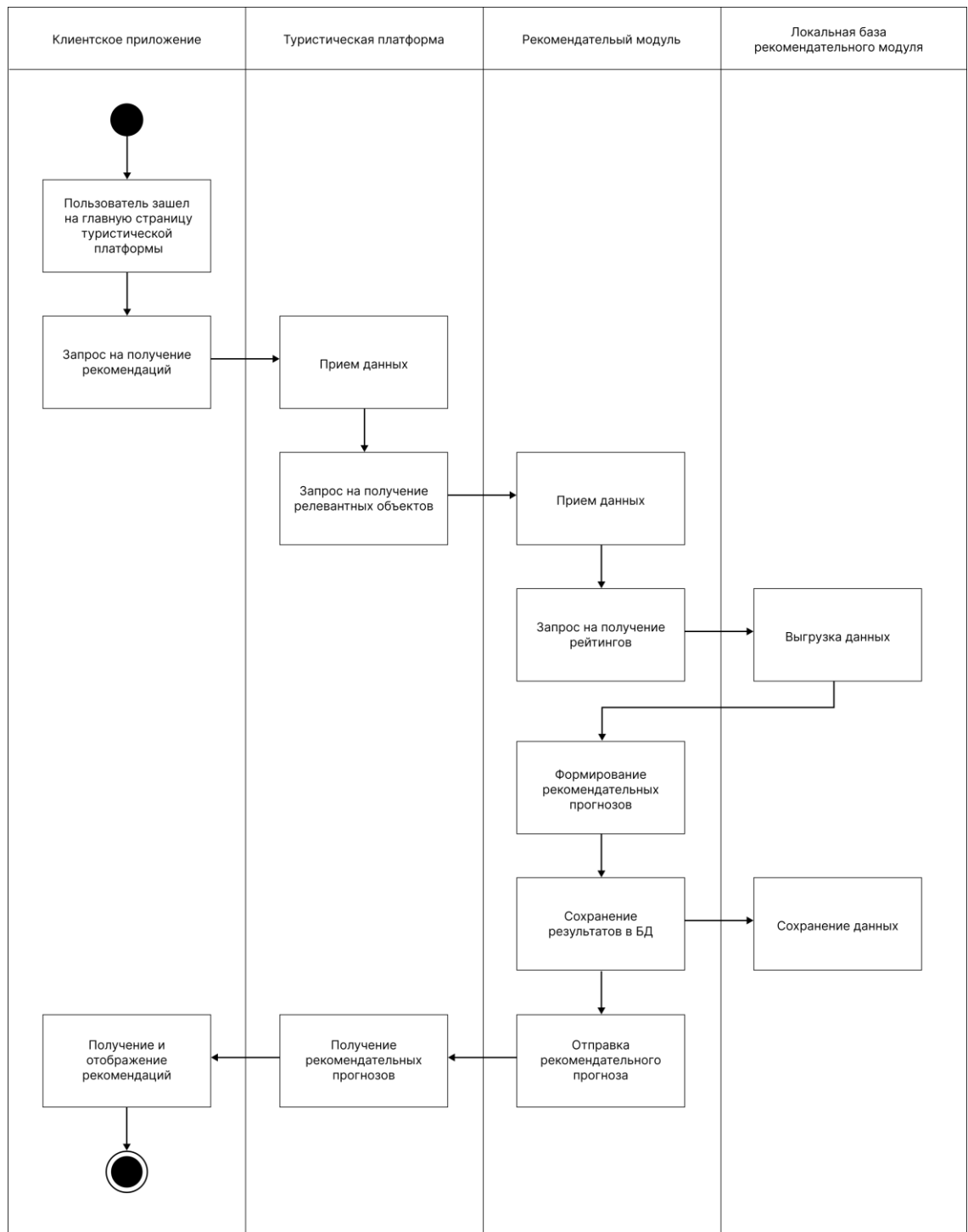


Рисунок 10 – UML диаграмма активности туристической платформы

Алгоритм работы матричной факторизации представлен на рисунке 11.

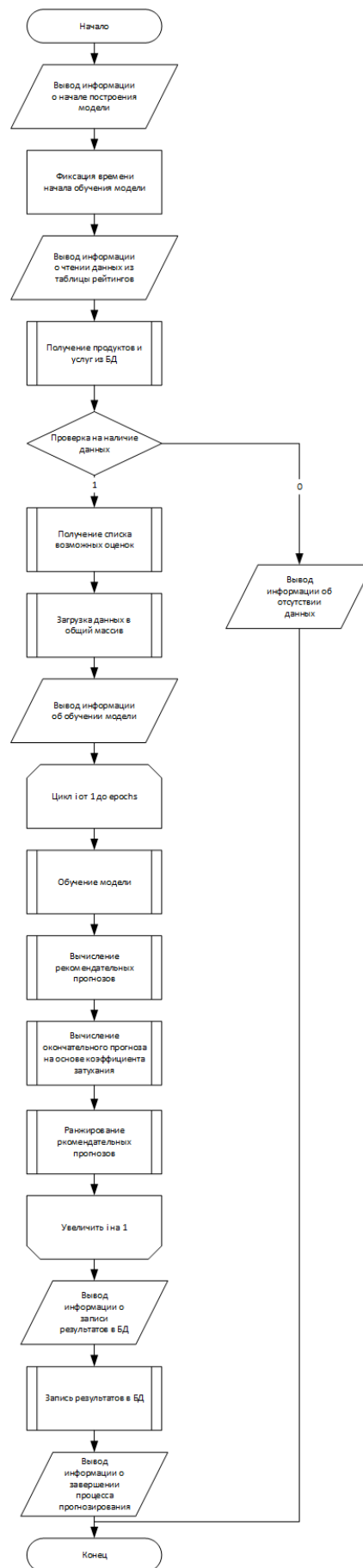


Рисунок 11 – Алгоритм работы матричной факторизации

2.4 Выводы по разделу

При проектировании рекомендательной системы туристических маршрутов обязательным является разработка алгоритма формирования релевантных прогнозов с использованием модели SVD матрицы факторизации, а также отрицательных весов и коэффициента затухания для решения проблемы «пузыря фильтров». Кроме того, при построении модели необходима информация о предпочтениях пользователей в виде пользователь-продукт-оценка.

3 Результаты работы рекомендательной системы

3.1 Описание API получения исходных данных

Механизмы прогнозирования в рекомендательном сервисе разработан с использованием фреймворка FastAPI, Python 3.8. Выбор данного инструмента обусловлен возможностью создания быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью. Для управления базами данных была выбрана такая СУБД как PostgreSQL, поскольку она предоставляет возможность работы с большими объемами данных и модификации под любую задачу [13].

Исходная информация предоставляется сторонним сервисом управления контентом CMS в формате JSON посредством использования REST API, которые предусматривают обязательный параметр пути (URI), включающий в себя идентификатор приложения [14].

Тело запроса состоит из полей, которые необходимы для получения информации о объекте интереса. При успешном получении данных возвращается код 200 и объект, описывающий характеристики интересующего элемента. В случае ошибки выдается код 400 и сообщение «Bad Request».

Запрос на получение списка туров. Для отображения информации о турах, а также дальнейшей ее обработки при прогнозировании, используется запрос на получение конкретного числа объектов интереса данного типа. В качестве входных параметров метод GET принимает атрибут локализации, номер страницы, а также количество необходимых элементов.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,
- общее количество элементов в списке туров,
- список городов,
- список туров.

Пример описания запроса на получение списка туров представлен ниже.

Метод: GET

URL: api/cmsapi/tours

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{  
  "language": "string",  
  "page": number,  
  "pageSize": number,  
}
```

Ответ:

200 OK

```
{  
  "page": number,  
  "pageSize": number,  
  "total": number,  
  "cities": [  
    {  
      "id": "string",  
      "title": "string",  
    },  
  ],  
  "items": [  
    {  
      "id": "string",  
      "title": "string",  
      "city": "string",  
      "days": number,  
      "image_detailed_page_main": [  

```

```

    {
      "avg": [
        {
          "a": number,
          "b": number,
          "g": number,
          "r": number,
        }
      ],
      "image": "string",
    }
  ],
  "image_explore_preview": [
    {
      "avg": [
        {
          "a": number,
          "b": number,
          "g": number,
          "r": number,
        }
      ],
      "image": "string",
    }
  ],
  "images": [
    {
      "avg": [
        {

```

```

        "a": number,
        "b": number,
        "g": number,
        "r": number,
    }
],
    "image": "string",
}
],
    "min_group_count": "string",
    "nights": number,
    "price": "string",
    "screen": "string",
    "season_end": "string",
    "season_start": "string",
    "short_description": "string",
    "sort": number,
    "url": "string",
}
]
}

```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка ресторанов. Для отображения информации о ресторанах используется запрос на получение конкретного числа объектов интереса данного типа. В качестве входных параметров метод GET принимает атрибут локализации, номер страницы, а также количество необходимых элементов.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,
- общее количество элементов в списке ресторанов,
- список партнёров,
- список городов,
- список ресторанов.

Пример описания запроса на получение списка ресторанов представлен ниже.

Метод: GET

URL: api/cmsapi/restaurants

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{  
  "language": "string",  
  "page": number,  
  "pageSize": number,  
}
```

Ответ:

200 OK

```
{  
  "page": number,  
  "pageSize": number,  
  "total": number,  
  "partner_types": [  
    {  
      "id": "string",  
      "title": "string",
```

```

        "images": [
            {
                "avg": [
                    {
                        "a": number,
                        "b": number,
                        "g": number,
                        "r": number,
                    }
                ],
                "image": "string"
            }
        ]
    },
],
"cities": [
    {
        "id": "string",
        "title": "string",
    }
],
"cuisines": [
    {
        "id": "string",
        "title": "string",
    }
],
"items": [
    {

```

```

    "id": "string",
    "title": "string",
    "address": "string",
    "avg_time_visit": number,
    "can_reserve": boolean,
    "city": "string",
    "coordinates": [
        {
            "lat": number,
            "lng": number,
        }
    ],
    "image_detailed_page_main": [
        {
            "avg": [
                {
                    "a": number,
                    "b": number,
                    "g": number,
                    "r": number,
                }
            ],
            "image": "string",
        }
    ],
    "image_explore_preview": [
        {
            "avg": [
                {

```

```

        "a": number,
        "b": number,
        "g": number,
        "r": number,
    }
],
    "image": "string",
}
],
"images": [
    {
        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ],
        "image": "string"
    }
],
"partner_types": "string",
"rest_id": "string",
"screen": "string",
"short_description": "string",
"sort": number,
"url": "string",
}

```

```
    ]  
  }  
Bad Request  
{ "errorMessage": "string" }
```

Запрос на получение списка городов. Для получения списка городов используется соответствующий запрос, входным параметром которого является атрибут локализации.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- список регионов,
- список городов.

Пример описания запроса на получение списка городов представлен ниже.

Метод: GET

URL: api/cmsapi/cities

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{  
  "language": "string",  
}
```

Ответ:

200 OK

```
{  
  "regions": [  
    {  
      "id": "string",  
      "title": "string",  
    },  
  ],  
}
```

```

    "items": [
      {
        "id": "string",
        "title": "string",
        "list_image": [
          {
            "avg": null,
            "image": "string"
          }
        ],
        "rating": number,
        "region": "string",
        "short_description": "string",
      }
    ]
  }
}

```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка маршрутов. Для получения списка пакетов используется соответствующий запрос, входным параметром которого является атрибут локализации, номер страницы, количество элементов на странице, а также аргументы фильтрации по тегам, поиска по названию и фильтрации по городам.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,
- общее количество элементов в списке маршрутов,
- список пакетов,

- список тегов,
- основной тег страницы,
- список городов.

Пример описания запроса на получение списка маршрутов представлен ниже.

Метод: GET

URL: api/cmsapi/routes

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{
  "language": "string",
  "page": number,
  "pageSize": number,
  "query[dictionary_data.route_tags][$in][]": "string",
  "query[dictionary_data.title][$regex]": "string",
  "query[dictionary_data.city][$in][]": "string"
}
```

Ответ:

200 OK

```
{
  "page": number,
  "pageSize": number,
  "total": number,
  "items": [
    {
      "activity_count": null,
      "billing_product_id": number,
      "city": "string",
      "description": "string",
```

```

    "id": "string",
    "image_detailed_page_main": [
      {
        "avg": [
          {
            "a": number,
            "b": number,
            "g": number,
            "r": number,
          }
        ],
        "image": "string"
      }
    ],
    "image_explore_preview": [
      {
        "avg": [
          {
            "a": number,
            "b": number,
            "g": number,
            "r": number,
          }
        ],
        "image": "string",
      }
    ],
    "images": [
      {

```

```

    "avg": [
      {
        "a": number,
        "b": number,
        "g": number,
        "r": number,
      }
    ],
    "image": "string",
  }
],
"packet_price": number,
"route": [
  {
    "address": "string",
    "coordinates": [
      {
        "lat": number,
        "lng": number,
      }
    ],
    "google_address": "string",
    "id": "string",
    "image_detailed_page_main": [
      {
        "avg": [
          {
            "a": number,
            "b": number,

```

```

        "g": number,
        "r": number,
    }
],
    "image": "string",
}
],
"image_explore_preview": [
    {
        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ],
        "image": "string",
    }
],
"images": [
    {
        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ]
    }
]

```

```

        ],
        "image": "string",
    }
],
    "partner_type_id": "string",
    "route_description": "string",
    "tags": ["string"],
    "title": "string",
    "type": "string"
}

],
"route_tags_id": ["string"],
"rp_price_id": number,
"service_count": null,
"short_description": "string",
"sort": number,
"tags_main_screen": ["string"],
"time ": "string",
"title": "string",
}

],
"route_tags": [
    {
        "id": "string",
        "title": "string",
    }
],
"tags": [
    {

```

```

        "id": "string",
        "title": "string",
    },
],
"tags_main_screen": [
    {
        "id": "string",
        "title": "string",
    }
],
"cities": [
    {
        "id": "string",
        "title": "string",
    }
]
}

```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка мест. Для отображения информации о местах используется запрос на получение конкретного числа объектов интереса данного типа. В качестве входных параметров метод GET принимает атрибут локализации, номер страницы, количество элементов на странице, а также аргументы фильтрации по тегам, поиска по названию и фильтрации по городам.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,

- количество элементов в списке мест,
- список мест,
- список районов,
- список тегов,
- список метро.

Пример описания запроса на получение списка мест представлен ниже.

Метод: GET

URL: api/cmsapi/places

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{
  "language": "string",
  "page": number,
  "pageSize": number,
  "query[dictionary_data.route_tags][$in][]": "string",
  "query[dictionary_data.title][$regex]": "string",
  "query[dictionary_data.city][$in][]": "string"
}
```

Ответ:

200 OK

```
{
  "page": number,
  "pageSize": number,
  "total": number,
  "items": [
    {
      "id": "string",
      "title": "string",
      "address": "string",
```

```
"coordinates": [  
  {  
    "lng": number,  
    "lat": number,  
  }  
,  
  "district_ids": ["string"],  
  "images": [  
    {  
      "avg": [  
        {  
          "a": number,  
          "b": number,  
          "g": number,  
          "r": number,  
        }  
      ],  
      "image": "string",  
    }  
,  
    "image_explore_preview": [  
      {  
        "avg": [  
          {  
            "a": number,  
            "b": number,  
            "g": number,  
            "r": number,  
          }  
        ]  
      }  
    ]  
  ]  
}
```

```

        ],
        "image": "string",
    }
],
"image_detailed_page_main": [
    {
        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ],
        "image": "string",
    }
],
"tags_ids": ["string"],
"metro_ids": ["string"],
}
],
"districts": [
    {
        "id": "string",
        "title": "string",
    }
],
"tags": [
    {

```

```

        "id": "string",
        "title": "string",
    },
],
"metro": [
    {
        "id": "string",
        "title": "string",
        "line": [
            {
                "title": "string",
                number: "string",
                "color": [
                    {
                        "r": number,
                        "g": number,
                        "b": number,
                        "a": number,
                    }
                ]
            }
        ]
    }
]
}
]
}

```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка событий. Информацию о событиях можно получить в ответ на соответствующий запрос, у которого входными

параметрами являются: атрибут локализации, номер страницы, количество элементов на странице, а также аргументы фильтрации по тегам, городам и датам проведения.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,
- общее количество элементов в списке событий,
- список событий,
- список мест проведения,
- список типов событий,
- список тегов,
- список городов.

Пример описания запроса на получение списка событий представлен ниже.

Метод: GET

URL: api/cmsapi/events

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{
  "language": "string",
  "page": number,
  "pageSize": number,
  "query[dictionary_data.route_tags][$in][]": "string",
  "query[dictionary_data.city][$in][]": "string",
  "query[dictionary_data.schedule.start][$gte]":
  "string",
  "query[dictionary_data.schedule.start][$lte]":
  "string"
```

```
}
```

Ответ:

200 OK

```
{
```

```
  "page": number,
```

```
  "pageSize": number,
```

```
  "total": number,
```

```
  "items": [
```

```
    {
```

```
      "id": "string",
```

```
      "title": "string",
```

```
      "place_ids": ["string"],
```

```
      "tags_ids": ["string"],
```

```
      "city": ["string"],
```

```
      "images": [
```

```
        {
```

```
          "avg": [
```

```
            {
```

```
              "a": number,
```

```
              "b": number,
```

```
              "g": number,
```

```
              "r": number,
```

```
            }
```

```
          ],
```

```
          "image": "string",
```

```
        }
```

```
    ],
```

```
    "image_explore_preview": [
```

```
      {
```

```

        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ],
        "image": "string",
    }
],
"image_detailed_page_main": [
    {
        "avg": [
            {
                "a": number,
                "b": number,
                "g": number,
                "r": number,
            }
        ],
        "image": "string",
    }
],
"cg_announcement": "string",
"link_source": "string",
"ticket_price": "string",
"duration": "string",
"event_types_ids": ["string"],

```

```

        "sort": "string"
    }
],
"places": [
    {
        "id": "string",
        "title": "string",
        "address": "string",
        "coordinates": [
            {
                "lng": number,
                "lat": number,
            }
        ],
        "images": [
            {
                "avg": [
                    {
                        "a": number,
                        "b": number,
                        "g": number,
                        "r": number,
                    }
                ],
                "image": "string",
            }
        ],
        "badge": "string",
    }
]

```

```

],
"event_types": [
  {
    "id": "string",
    "title": "string",
    "images": [
      {
        "avg": [
          {
            "a": number,
            "b": number,
            "g": number,
            "r": number,
          }
        ],
        "image": "string",
      }
    ]
  }
],
"tags": [
  {
    "id": "string",
    "title": "string",
  }
],
"cities": [
  {
    "id": "string",

```

```
        "title": "string"
      }
    ]
  }
}
```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка подборок. Список подборок можно получить в ответ на соответствующий запрос, у которого входными параметрами является атрибут локализации.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- список подборок,
- список тегов.

Пример описания запроса на получение списка подборок представлен ниже.

Метод: GET

URL: api/cmsapi/collections

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{
  "language": "string"
}
```

Ответ:

200 OK

```
{
  "items": [
    {
      "title": "string",
      "subtitle": "string",

```

```

    "is_collection": boolean,
    "description": "string",
    "full_url": "string",
    "screen": "string",
    "images": [
        {
            "avg": [
                {
                    "a": number,
                    "b": number,
                    "g": number,
                    "r": number,
                }
            ],
            "image": "string",
        }
    ],
    "rp_places_count": null,
    "rp_events_count": number,
    "rp_restaurants_count": null,
    "rp_tours_count": null,
    "half": boolean,
    "sort": number,
    "tags": ["string"],
}
],
"tags": [
    {
        "id": "string",

```

```
        "title": "string",
    }
]
}
```

Bad Request

```
{ "errorMessage": "string" }
```

Запрос на получение списка объектов интереса. Данный запрос возвращает общий список объектов интереса не зависимо от их типа. Входные параметрами являются: атрибут локализации, номер страниц, количество элементов страницы, аргументы фильтрации по городам и датам проведения.

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование количества элементов страницы,
- общее количество элементов в списке объектов интереса,
- список городов,
- список кулинарных направлений,
- список типов событий,
- список объектов интереса,
- список тегов.

Пример описания запроса на получение списка объектов интереса представлен ниже.

Метод: GET

URL: api/mainscreen

Заголовок запроса: x-app-token: 2374e3e3-7b13-4a08-a807-980e4b3802cd

Тело запроса:

```
{
    "language": "string",
    "page": number,
```

```

    "pageSize": number,
    "query[dictionary_data.city][$in][]": "string",
    "query[dictionary_data.schedule.start][$gte]":
        "string",
    "query[dictionary_data.schedule.start][$lte]":
        "string"
}

```

ОТВЕТ:

200 OK

```

{
    "page": number,
    "pageSize": number,
    "total": number,
    "cities": [
        {
            "id": "string",
            "title": "string",
        }
    ],
    "cuisines": [
        {
            "id": "string",
            "title": "string",
        }
    ],
    "event_types": [
        {
            "id": "string",
            "title": "string",
        }
    ]
}

```

```

    "images": [
      {
        "avg": [
          {
            "a": number,
            "b": number,
            "g": number,
            "r": number,
          }
        ],
        "image": "string",
      }
    ]
  },
],
"items": [
  {
    "billing_product_id": null,
    "city": ["string"],
    "cuisines_ids": ["string"],
    "id": "string",
    "packet_price": number,
    "purchase_addtional_info": "string",
    "rest_id": "string",
    "rp_events_count": number,
    "rp_places_count": number,
    "rp_price_id": number,
    "rp_restaurants_count": number,
    "rp_tours_count": number,
  }
]

```

```

        "screen": "string",
        "sort": number,
        "tags_ids": ["string"],
        "ticket_price": "string",
        "title": "string",
        "url": "string"
    }
],
"tags": [
    {
        "id": "string",
        "title": "string",
    }
]
}

```

Bad Request

```
{ "errorMessage": "string" }
```

3.2 Описание API рекомендательной системы

С заданной периодичностью сервис синхронизирует данные путем отправки в систему управления контентом (CMS) запросов на получение информации о событиях, маршрутах, турах и ресторанах, с последующим сохранением атрибутов, необходимых для построения модели и поддержки поисковых запросов.

Параметры запросов, получаемые сервисом рекомендаций, а также промежуточные результаты их обработки хранятся в оперативной памяти устройства, а данные, полученные в результате применения алгоритмов рекомендаций в реляционной базе данных системы управления контентом.

В качестве основных данных для построения модели выступают предпочтения субъектов в виде пользователь-продукт-оценка. Для этого в системе реализовано следующее API:

```
{  
    "uid": "string", // уникальный идентификатор пользователя  
    "iid": "string", // уникальный идентификатор продукта или услуги  
    "rating": enum // перечисление оценок  
}
```

Примеры заполнения отношений пользователь-продукт:

Метод: POST

URL: reco/api/rating

Тело запроса:

```
[{  
    "uid": "1",  
    "iid": "5e598dcbda6bb300122c582d",  
    "rating": "SEARCH"  
},  
{  
    "uid": "1",  
    "iid": "5e599723da6bb300122c5acf",  
    "rating": "FAVORITE"  
}]
```

Ответ:

200 OK

Данный запрос возвращает ошибку 422, в случае, когда в наборе два объекта с «uid» и «iid» идентичны.

Для получения рекомендаций по объектам интереса, сформированным на основе матричной факторизации, используются методы GET,

предусматривающие обязательный параметр пути (URI), включающий в себя идентификатор приложения.

Например, список релевантных туров можно получить в ответ на соответствующий запрос, у которого входными параметрами является атрибуты, представленные в таблице 15.

Таблица 15 – Параметры, необходимые для отображения персонализированных прогнозов

Название	Обязательность	По умолчанию	Тип	Описание	Пример
uid	Да		string	Идентификатор пользователя	uid=1
language	Нет	ru	string	Язык	language=en
city	Нет		string	Город	city=5bbe5
gte	Нет		string	Дата с	gte=2020-01-01T00:00:00+03:00
lte	Нет		string	Дата по	lte=2021-01-01T00:00:00+03:00
tags	Нет		string	Тэги	tags=5bbe&tags=912a
Page	Нет	1	number	Страница	page=2
size	Нет	10	number	Количество элементов на странице	size=20

В случае успешной обработки запроса, полученный ответ имеет следующие атрибуты:

- дублирование номера страницы,
- дублирование заданного количества элементов,
- общее количество элементов в списке ресторанов,
- список идентификаторов объектов интереса.

Пример описания запроса на получение списка туров представлен ниже.

Метод: POST

URL: reco/api/recommendation/tours

Тело запроса:

```
[{  
    "uid": "string",
```

```
    "language": "string",  
    "city": "string",  
    "gte": "string",  
    "lte": "string",  
    "tags": "string",  
    "page": number,  
    "size": number,  
  }  
}
```

Ответ:

200 OK

```
{  
  "page": number,  
  "pageSize": number,  
  "total": number,  
  "items": [  
    {  
      "id": "string"  
    }  
  ]  
}
```

Bad Request

```
{ "errorMessage": "string" }
```

Листинг рекомендательной системы туристических маршрутов представлен в приложении А.

3.3 Тестирование рекомендательной системы

При проектировании рекомендательной системы туристических маршрутов использовались исходные данные, предоставленные сервисом управления контентом платформы RUSSPASS в рамках хоздоговорной научно исследовательской работы. Модель матричной факторизации была построена,

а в дальнейшем обучена, на основе тестовой выборки, состоящей из тридцати записей. При этом текущая выборка содержала эталонный набор данных, с которым впоследствии и сравнивались результаты работы разрабатываемого механизма.

На завершающем этапе реализации задачи была выполнена конечная проверка работоспособности разработанного механизма с использованием контрольной выборки, состоящей из сорока пяти записей.

На основании полученных результатов была произведена проверка точности работы разработанного алгоритма с использованием метрики качества ранжирования, вычисление которой осуществляется по формуле:

$$accuracy_k = \frac{\sum_{i=1}^k (Relevant_i * Predication_i)}{\sum_{i=1}^k Predication_i} * 100, \quad (11)$$

где $Relevant_i$ – вероятностный показатель релевантности объекта интереса;

$Predication_i$ – предполагаемый показатель заинтересованности пользователя в объекте интереса.

По результатам проведенных вычислений было выявлено, что точность работы алгоритма примерно равна 83,81 процентам. Это позволяет сделать вывод о пригодности данного механизма прогнозирования для достижения поставленных целей.

Примеры туров, входящих в контрольную выборку, представлен ниже:

```
{
  "page": 1
  "pageSize": 2
  "total": 508
  "cities": [],
  "items": [
    {
      "id": "5fa430bfe791b200193891dc",
```

```

"title": "Привет, Горный Алтай! Макси",
"city": "",
"days": 8,
"image_detailed_page_main": [],
"image_explore_preview": [
    {
        "avg": null,
        "image": "/v1/file/6257d0f045cc11f9e82697b4",
    }
],
"min_group_count": "1-40",
"nights": 7,
"price": "48700",
"screen": "tour",
"season_end": "2022-09-17T21:00:00.000Z", "season_start":
"2022-06-11T21:00:00.000Z",
"short_description": "<p>Когда подъезжаешь к горе Пикет,
кажется, будто попал в кино: словно смотришь
заключительные минуты фильма «Печки-лавочки». Доля
правды в этом, конечно, есть. Именно тут, в родном селе
Сростки, Василий Шукшин снимал свою картину. Теперь на
Пикете стоит памятник режиссеру и актеру, а в местной
школе, где он некогда учился и даже успел поработать,
находится музей автора «Калины красной» и «Живет такой
парень».</p>",
"sort": 102,
"url": "/screens/event?id=5fa430bfe791b200193891dc"
},
{

```

```
"id": "61bb646814277dec6448c187",
"title": "Трансформация — лучший отдых с детьми. Два
города — миллион впечатлений",
"city": "",
"days": 6,
"image_detailed_page_main": [],
"image_explore_preview": [
  {
    "avg": null,
    "image": "v1/file/623983e5994c9997315b7648",
  }
],
"min_group_count": "1-4",
"nights": 5,
"price": "50000",
"screen": "tour",
"season_end": "2022-08-23T21:00:00.000Z", "season_start":
"2021-12-31T21:00:00.000Z",
"short_description": "<p>Проведите семейные каникулы
ярко, насыщенно, активно — отправьтесь в увлекательное
путешествие по двум российским столицам. Программа
специально разработана для тех, кому надоели стандартные
туристические маршруты. Вас ждут интерактивные
экскурсии и квесты по Петербургу и Москве — вы
познакомитесь с главными достопримечательностями
городов, посетите аквапарк, научитесь готовить шоколад и
продегустируете мороженое на высоте 327 метров.</p><p>В
стоимость тура входит: встреча на ж/д вокзале или в
аэропорту Санкт-Петербурга, проживание в гостиницах,
```

завтраки со второго дня тура, экскурсионное обслуживание, включая входные билеты в аквапарк «Питерлэнд», «Кидзанию», на смотровую площадку PANORAMA360 с дегустацией мороженого и автобус City Sightseeing, услуги гида-сопровождающего, трансферы по программе.</p>

```
"sort": 99,  
"url": "/screens/tour?id=61bb646814277dec6448c187"
```

```
}
```

```
]
```

```
}
```

Пример общего списка объектов интереса пользователя, не зависимо от их типа, представлен ниже:

```
{
```

```
  "page": 1,
```

```
  "pageSize": 45,
```

```
  "total": 302,
```

```
  "cities": [
```

```
    {
```

```
      "id": "5bbf0e5f729bd4000fe02d5e",
```

```
      "title": "Санкт-Петербург",
```

```
    }
```

```
  ],
```

```
  "cuisines": null,
```

```
  "event_types": [],
```

```
  "items": [
```

```
    {
```

```
      "city": [],
```

```
      "id": "62875a721bcce6664fa10d98",
```

```
      "screen": "tour",
```

```

"title": "Яркие выходные в Москве",
"image_explore_preview": [
  {
    "image": "v1/file/628e37b6d72a9ab024af013a",
    "avg": null
  }
],
"region": [
  "5d08e36dad3a9a001701b95b"
],
"nights": "2",
"days": "3",
"tags_ids": [
  "611a817f57f8c10019f8c8eb",
  "611a7cd057f8c10019f8c8d1",
  "611a7d6d57f8c10019f8c8d7",
  "611a842657f8c10019f8c8ff",
  "6123b4a1020892001197bb4d",
  "611a6d2257f8c10019f8c8aa",
  "611a84db57f8c10019f8c903",
  "611a852c57f8c10019f8c907"
],
"price": "16550",
"sort": 103
"url": "screens/tour?id=62875a721bcce6664fa10d9"
},
{
  "city": [],
  "id": "61e14c732d49e6f7fae29cf3",

```

```

    "screen": "tour",
    "title": "Сахалин, Итуруп",
    "image_explore_preview": [
        {
            "image": "v1/file/61e14bfe2d49e6f7fae29b53",
            "avg": null
        }
    ],
    "region": [
        "5d08e36fad3a9a001701b9c3"
    ],
    "nights": "7",
    "days": "8",
    "tags_ids": [
        "611a817f57f8c10019f8c8eb",
        "611a7cd057f8c10019f8c8d1",
        "611a827e57f8c10019f8c8f3",
        "611a7ea157f8c10019f8c8df",
        "611a85e957f8c10019f8c911",
        "611a842657f8c10019f8c8ff",
        "6123b4a1020892001197bb4d",
        "611a6d2257f8c10019f8c8aa"
    ],
    "price": "145000",
    "sort": 99
    "url": "/screens/tour?id=61e14c732d49e6f7fae29cf3"
}
],
"tags": [

```

```

    {
      "id": " 611a6d2257f8c10019f8c8aa",
      "title": " С детьми",
    },
    {
      "id": " 611a7cd057f8c10019f8c8d1",
      "title": "Обзорные",
    },
    {
      "id": " 611a815857f8c10019f8c8e9",
      "title": "Индивидуальные",
    },
    {
      "id": "611a85bf57f8c10019f8c90f",
      "title": "6 ночей",
    }
  ]
}

```

Результатом применения механизма SVD, полученный вследствие скалярного произведения элементов разложенной матрицы, является полный двумерный массив, состоящий из возможных оценок конкретного пользователя на данный продукт. Исходя из этого, полученная матрица анализируется на наличие наивысших оценок для конкретного пользователя, после чего эти данные записываются в общий массив прогнозов, пример которого представлен на рисунке 12.

	uid [PK] character varying (255)	iid [PK] character varying (255)	prediction double precision
1	0073d93f-1937-4769-856a-395ece1...	7d557046-f183-4edf-b0f9-701a600d...	0.464531984771238
2	0073d93f-1937-4769-856a-395ece1...	a82a44a6-dbb4-423f-ac08-65d7aa4...	0.230038522794868
3	0073d93f-1937-4769-856a-395ece1...	c32ad53b-b242-4e45-8c43-86fb7c1...	0.9496927120346776
4	0073d93f-1937-4769-856a-395ece1...	e7820871-a80b-4ab3-b455-29bdb9...	0.123500256281787
5	2e3bd807-0d56-4087-9838-704899...	61b4485b-3050-4624-a9cb-299316...	0.1143079064819834
6	2e3bd807-0d56-4087-9838-704899...	a82a44a6-dbb4-423f-ac08-65d7aa4...	0.9015127936426466
7	2e3bd807-0d56-4087-9838-704899...	c32ad53b-b242-4e45-8c43-86fb7c1...	0.302466889308357
8	2e3bd807-0d56-4087-9838-704899...	e7820871-a80b-4ab3-b455-29bdb9...	0.561668500932056

Рисунок 12 – Результат прогнозирования на основе матричной факторизации

В результате при запросе k рекомендаций для пользователя (uid), будут выбраны все k продукты или услуги (iid) с максимальным значением показателя заинтересованности (prediction).

Результатом формирования релевантных подборок для пользователя, просматривавшего туры в Сахалин и Москву, является список туров в Москву, Сахалин, а также парочка туров в Санкт-Петербург и в Хабаровский край. Сделано это для того, чтобы разнообразить список прогнозов элементами, заведомо неподходящими пользователю, тем самым решая вопрос информационного пузыря.

Пример получения списка рекомендаций для данного пользователя представлен ниже:

Тело запроса:

```
[{
  "uid": "1632226467667053696",
  "language": "ru",
  "page": 1,
  "size": 12,
}]
```

Ответ:

200 OK

```
{
  "page": 1,
  "pageSize": 12,
  "total": 302,
  "items": [
    {
      "id": "62875a721bcce6664fa10d98"
    },
    {
      "id": "6257d12a45cc11f9e82697fe"
    },
    {
      "id": "5e5994d3da6bb300122c5a4a"
    },
    {
      "id": "617146c153b83e0018a78db7"
    },
    {
      "id": "619776788db05de136ca705c"
    },
    {
      "id": "61bb646814277dec6448c187"
    },
    {
      "id": "61cc55622d49e6f7fa75897a"
    },
    {
      "id": "61e14c732d49e6f7fae29cf3"
    },
  ],
}
```

```

    {
      "id": "61e973c32d49e6f7fa0b1737"
    },
    {
      "id": "61f957152d49e6f7fa6842b5"
    },
    {
      "id": "6204d3faa70b798fc9dbe109"
    },
    {
      "id": "620bb8829480ec8db0635e04"
    }
  ]
}

```

Исходя из полученных результатов можно сделать вывод, что алгоритм прогнозирования, разработанный в рамках данного научного исследования, является достаточно точным, а также позволяет классифицировать объекты интереса на основе предпочтений пользователя и при этом решить проблему «пузыря фильтров».

3.4 Выводы по разделу

На завершающем этапе реализации механизма прогнозирования для рекомендательной системы туристических маршрутов были описаны API получения исходных данных и получения релевантных подборок. Кроме того была проведена проверка результатов исследования в целях установления работоспособности механизма составления рекомендательных прогнозов туристических маршрутов для достижения поставленных задач.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено исследование существующих туристических платформ и современных подходов к формированию релевантных подборок, а также была проанализирована проблема информационного пузыря. На основании проделанного анализа была разработана рекомендательная система туристических маршрутов, обеспечивающая сокращение временных затрат пользователя на поиск туристического маршрута и повышение востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее.

Основные положения и выводы данной диссертации базировались на анализе подходов к формированию рекомендательных подборок в рамках хоздоговорной научно-исследовательской работы «Исследование способов построения рекомендательной системы туристических маршрутов».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Jannach D. et al. Recommender Systems: An Introduction. – Cambridge University Press, 2010, p. 360.
2. Chen Y., Zhao X., de Rijke M. Top-N recommendation with high-dimensional side information via locality preserving projection //Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. – 2017. – С. 985-988.
3. Николенко С. Рекомендательные системы //СПб: Изд-во Центр Речевых Технологий. – 2012 – 53 с.
4. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions //IEEE transactions on knowledge and data engineering. – 2005. – Т. 17. – №. 6. – С. 734-749.
5. Gerbrands J. J. On the relationships between SVD, KLT and PCA //Pattern recognition. – 1981. – Т. 14. – №. 1-6. – С. 375-381.
6. Hidasi B. et al. Session-based recommendations with recurrent neural networks //arXiv preprint arXiv:1511.06939. – 2015.
7. Bottou L. Stochastic gradient descent tricks //Neural networks: Tricks of the trade. – Springer, Berlin, Heidelberg, 2012. – С. 421-436.
8. Звягин Л. С. Метод байесовских сетей и ключевые аспекты байесовского моделирования //Международная конференция по мягким вычислениям и измерениям. – Федеральное государственное автономное образовательное учреждение высшего образования Санкт-Петербургский государственный электротехнический университет ЛЭТИ им. ВИ Ульянова (Ленина), 2019. – Т. 1. – С. 30-34.
9. Aggarwal C. C. et al. Data mining: the textbook. – New York : springer, 2015. – Т. 1.

10. Ricci F., Rokach L., Shapira B. Introduction to recommender systems handbook //Recommender systems handbook. – Springer, Boston, MA, 2011. – С. 1-35.
11. Chapman P. et al. CRISP-DM 1.0: Step-by-step data mining guide //SPSS inc. – 2000. – Т. 9. – С. 13.
12. Marbán O. et al. Toward data mining engineering: A software engineering approach //Information systems. – 2009. – Т. 34. – №. 1. – С. 87-107.
13. Моргунов Е. П., Рогова Е. В., Лузанова П. В. PostgreSQL. Основы языка SQL //учеб. пособие/ЕП Моргунов. – 2018.
14. Горнаков С. Осваиваем популярные системы управления сайтом (CMS). – Litres, 2022 – 336 с.

ПРИЛОЖЕНИЕ А

Листинги программы

Файл reco_builder.py

```
import time

import pandas as pd
import surprise
from loguru import logger

from app.core.config import DATABASE_URL
from app.utils.sql_utils import pandas_df_to_sql_table
from app.utils.surprise_utils import compute_ranking_predictions

def build_collaborate_recommendation() -> None:
    logger.info("Start build collaborate model...")
    start_time = time.time()
    logger.info("Read rating_data table...")
    df = pd.read_sql_table('rating_data', DATABASE_URL.url, columns=['uid', 'iid', 'rating'])
    if df.empty:
        logger.info("Empty table rating_data. Return")
        return
    reader = surprise.Reader(rating_scale=(1.0, 5.0))
    dataset = surprise.Dataset.load_from_df(df, reader)
    svd = surprise.SVD(random_state=0, n_factors=200, n_epochs=30, verbose=True)
    logger.info("Train model...")
    svd.fit(dataset.build_full_trainset())
    logger.info("Compute predictions...")
    all_predictions = compute_ranking_predictions(svd, df, usercol='uid', itemcol='iid',
remove_seen=True)
    logger.info("Write collaborate model to db...")
    pandas_df_to_sql_table(all_predictions, DATABASE_URL.url, 'df_ranking_users_items')
    logger.info("Finish build collaborate model in {0} seconds".format(time.time() - start_time))
```

Файл rating.py

```
from typing import List

import orjson
import pandas as pd
from pydantic import BaseModel, validator

from app.models.schemas.rwschema import RWSchema

from enum import Enum

class RatingType(str, Enum):
    search = 'SEARCH'
    click = 'CLICK'
    detail = 'DETAIL'
    favorite = 'FAVORITE'
    booking = 'BOOKING'
    buy = 'BUY'

    @property
    def val_as_int(self):
        if self.value == self.search:
            return 1
        elif self.value == self.click:
            return 2
        elif self.value == self.detail:
            return 3
        elif self.value == self.favorite:
            return 4
        elif self.value == self.booking:
            return 5
        elif self.value == self.buy:
            return 6
```

```

    else:
        raise TypeError

def dump_rating_type_as_int(v, *, default):
    for rating_data in v:
        rating_data.update({'rating': rating_data['rating'].val_as_int})
    return orjson.dumps(v, default=default).decode()

class RatingDataInSave(RWSchema):
    uid: str
    iid: str
    rating: RatingType

class ListRatingDataInSave(BaseModel):
    __root__: List[RatingDataInSave]

    @validator('__root__', pre=True)
    def uniq_data(cls, v):
        df = pd.DataFrame(v)
        df_duplicated = df[df.duplicated(['uid', 'iid'])]
        if not df_duplicated.empty:
            raise ValueError('Together uid and iid must be unique')
        return v

class Config:
    json_dumps = dump_rating_type_as_int

```

Файл **users.py**

```

from typing import Optional

from pydantic import BaseModel, EmailStr, HttpUrl

from app.models.domain.users import User
from app.models.schemas.rwschema import RWSchema

```

```

class UserInLogin(RWSchema):
    email: EmailStr
    password: str

class UserInCreate(UserInLogin):
    username: str

class UserInUpdate(BaseModel):
    username: Optional[str] = None
    email: Optional[EmailStr] = None
    password: Optional[str] = None
    bio: Optional[str] = None
    image: Optional[HttpUrl] = None

```

```

class UserWithToken(User):
    token: str

```

```

class UserInResponse(RWSchema):
    user: UserWithToken

```

Файл rwschema.py

```

from app.models.domain.rwmodel import RWModel

```

```

class RWSchema(RWModel):
    class Config(RWModel.Config):
        orm_mode = True

```

Файл jwt_meta.py

```

from datetime import datetime

```

```

from pydantic import BaseModel

```

```
class JWTMeta(BaseModel):
    exp: datetime
    sub: str
```

```
class JWTUser(BaseModel):
    username: str
```

Файл `jwt.py`

```
from datetime import datetime, timedelta
from typing import Dict
```

```
import jwt
from pydantic import ValidationError
```

```
from app.models.domain.users import User
from app.models.schemas.jwt import JWTMeta, JWTUser
```

```
JWT_SUBJECT = "access"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 60 * 24 * 7 # one week
```

```
def create_jwt_token(
    *, jwt_content: Dict[str, str], secret_key: str, expires_delta: timedelta,
) -> str:
    to_encode = jwt_content.copy()
    expire = datetime.utcnow() + expires_delta
    to_encode.update(JWTMeta(exp=expire, sub=JWT_SUBJECT).dict())
    return jwt.encode(to_encode, secret_key, algorithm=ALGORITHM).decode()
```

```

def create_access_token_for_user(user: User, secret_key: str) -> str:
    return create_jwt_token(
        jwt_content=JWTUser(username=user.username).dict(),
        secret_key=secret_key,
        expires_delta=timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES),
    )

def get_username_from_token(token: str, secret_key: str) -> str:
    try:
        return JWTUser(**jwt.decode(token, secret_key, algorithms=[ALGORITHM])).username
    except jwt.PyJWTError as decode_error:
        raise ValueError("unable to decode JWT token") from decode_error
    except ValidationError as validation_error:
        raise ValueError("malformed payload in token") from validation_error

```

Файл cms.py

```

from collections import ChainMap
from typing import List

import orjson
from aioredis import Redis
from httpx import AsyncClient

async def get_cms_entities(_redis: Redis, ids: List[str], language: str, paging: Pageable):
    resp = {
        'page': paging.page,
        'pageSize': paging.size,
        'total': len(ids)
    }
    keys = [{"{0}_{1}".format(_id, language.lower()) for _id in ids}
    if not keys:
        resp['items'] = []
    return resp

```

```

data = [orjson.loads(c) for c in await _redis.mget(*keys) if c is not None]
data_keys = list(ChainMap(*data))
for key in data_keys:
    unzip = {item['id']: item for sublist in
              [item_list[key] for item_list in data if key in item_list and isinstance(item_list[key],
list)]
              for item in sublist if 'id' in item}
    resp[key] = list(unzip.values())
return resp

```

Файл sync_cms_tour.py

```

import re
from concurrent.futures.thread import ThreadPoolExecutor
from typing import List
from urllib.parse import urljoin

import requests
from loguru import logger
from redis import Redis
from sqlalchemy import create_engine
from sqlalchemy.sql import text

from app.core.config import DATABASE_URL, CMS_DETAILS_URL, CMS_HEADERS,
CMS_HOST, CMS_DEFAULT_LANGUAGE
from app.models.domain.cms import EntityType
from app.utils import redis_utils, sql_utils

CMS_TOUR_URL = urljoin(CMS_HOST, 'v1/screens/tours')
PAGE_SIZE = 100

REQUEST_THREAD_NUMBER = 20
TAG_RE = re.compile(r'<[^>]+>')

```

```
def _remove_tags(_text: str) -> str:
    return TAG_RE.sub("", _text)
```

```
def sync_tours() -> None:
```

```
    page = 1
    total = 0

    cms_languages = sql_utils.get_cms_languages(DATABASE_URL.url)
    logger.info("Start sync tours...")
    engine = create_engine(DATABASE_URL.url)
    with engine.connect() as con, redis_utils.connect() as r_con:
        while True:
            payload = {'language': CMS_DEFAULT_LANGUAGE, 'page': page, 'pageSize':
PAGE_SIZE}

            r = requests.get(CMS_TOUR_URL, params=payload, headers=CMS_HEADERS)
            data = r.json()

            if data['pageSize'] > 0 and 'items' in data:
                args = [(con, r_con, cms_languages, item['id'], ) for item in data['items']]
                with ThreadPoolExecutor(REQUEST_THREAD_NUMBER) as executor:
                    executor.map(lambda p: _processing_tour(*p), args)
                total += len(data['items'])
                if total >= data['total']:
                    break
    logger.info("End sync tours...")
```

ПРИЛОЖЕНИЕ В

Презентация

РЕКОМЕНДАТЕЛЬНАЯ СИСТЕМА ТУРИСТИЧЕСКИХ МАРШРУТОВ

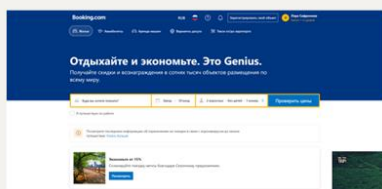
ВЫПОЛНИЛ СТУДЕНТ ГР.20ВВМ1
САФРОНОВА ВАЛЕРИЯ СЕРГЕЕВНА

НАУЧНЫЙ РУКОВОДИТЕЛЬ
СИНЕВ МИХАИЛ ПЕТРОВИЧ

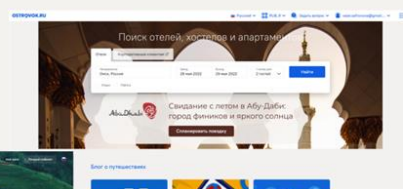
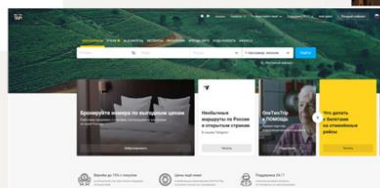
ПЕНЗА, 2022 г.

АНАЛОГИ ТУРИСТИЧЕСКИХ ПЛАТФОРМ

Booking.com

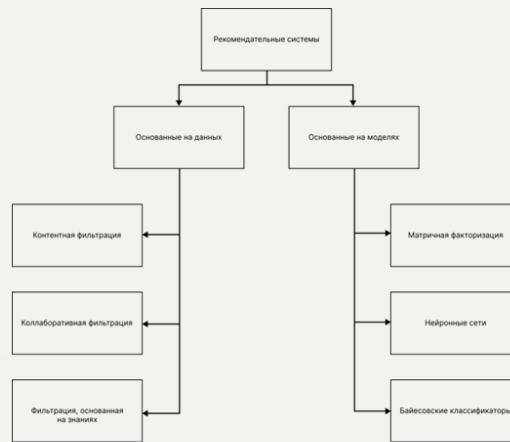


Ostrovok.ru

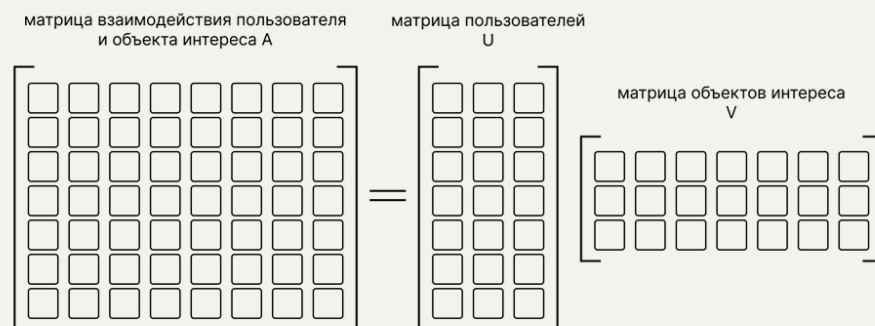


OneTwoTrip

СОВРЕМЕННЫЕ ПОДХОДЫ К ФОРМИРОВАНИЮ ПРОГНОЗОВ



МАТРИЧНАЯ ФАКТОРИЗАЦИЯ



НЕЙРОННЫЕ СЕТИ

Нейронные сети применяются в рекомендательных системах в двух вариациях:

1. механизм формирования рекомендаций;
2. промежуточный этап при построении моделей матричной факторизации.

В первом случае в качестве исходных данных предоставляются список объектов интереса и пользовательские оценки. Во время обучения модели вычисляемый результат сопоставляется с реальными предпочтениями, и в конечном итоге формирует принцип, на основе которого осуществляется составление рейтинговых прогнозов.

Основу второго подхода составляет матричная факторизация, роль нейронной сети в данном случае второстепенна. На начальном этапе происходит разложение исходной матрицы на двумерные массивы меньше размерности. Затем, полученные матрицы объединяются и подаются на вход в нейронную сеть. После чего, происходит стандартное обучение нейронной сети, целью которого является получение сложной нелинейной модели для факторизации матрицы.

БАЙЕСОВСКИЕ КЛАССИФИКАТОРЫ

Вероятностный классификатор предсказывает класс C_i с самой большой условной вероятностью P_i для заданного вектора признаков $x = (x_1, \dots, x_n)$:

$$P(C_i | x_1 \& \dots \& x_n) = \frac{P(x_1 | C_i) \cdot \dots \cdot P(x_n | C_i) \cdot P(C_i)}{P(x_1) \cdot \dots \cdot P(x_n)}$$

Вероятность класса C_i находится на основе параметров $(x_1, \dots, x_n)$, подразумевая, что признаки являются независимыми, и следовательно, приведенная выше вероятность пропорциональна:

$$P(C_i) \prod_{np} P(x_i | C_i)$$

Простой байесовский классификатор:

$$y = \arg \max \prod_{np} P(x_i | C_i)$$

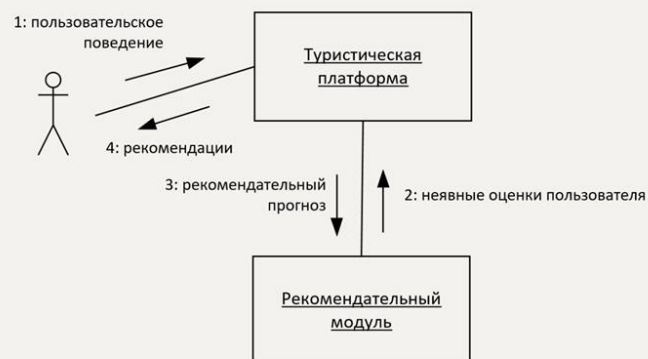
Результатом обучения контентной модели является список атрибутов вместе с условными вероятностями

ПРОБЛЕМА ИНФОРМАЦИОННОГО ПУЗЫРЯ

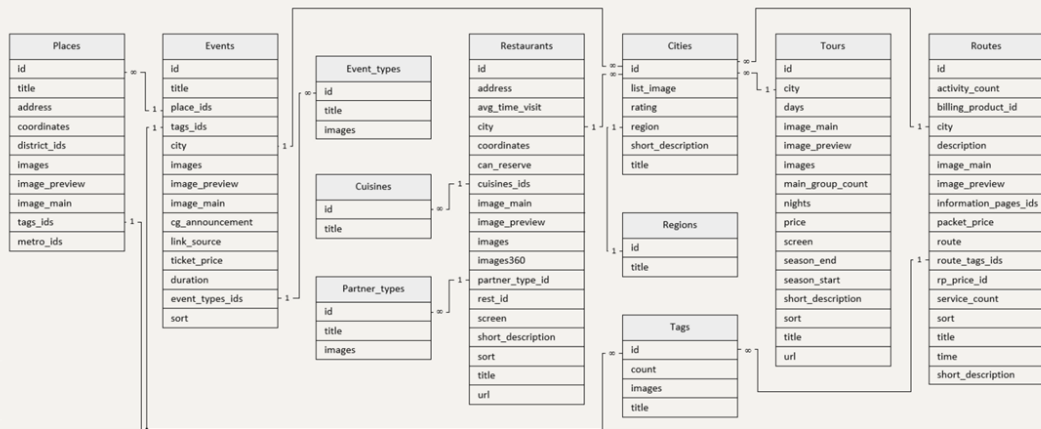
В новостных блогах проблема информационного пузыря приводит к искажению восприятия действительности за счет просмотра только тех новостей, которые подходят пользователю по его убеждениям, тем самым ограничивая его круг зрения

Применительно к туристическим платформам данная проблема отражается в ограничении целевой системой определенного типа маршрутами или местами, не добавляя объекты интереса другой категории. В результате чего снижается уровень востребованности у конкретной локации для текущего пользователя, а следовательно, понижается потенциальная прибыль поставщика услуг

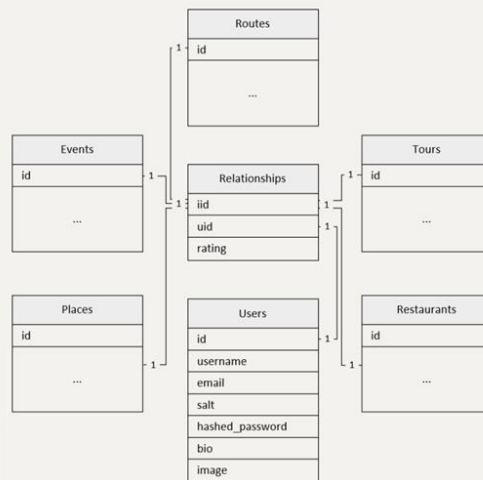
UML ДИАГРАММА КООПЕРАЦИИ



ИСХОДНЫЕ ДАННЫЕ



ИСХОДНЫЕ ДАННЫЕ



НЕЯВНЫЕ ОЦЕНКИ ПОЛЬЗОВАТЕЛЕЙ

Модель матрицы факторизации строится по предпочтениям других пользователей в виде пользователь-продукт-оценка

- SEARCH (1): пользователь произвел поиск объекта интереса;
- CLICK (2): пользователь перешел на страницу объекта интереса;
- DETAIL (3): пользователь просмотрел детализацию характеристик объекта интереса;
- FAVORITE (4): пользователь добавил объект интереса в избранное;
- BOOKING (5): пользователь забронировал / добавил в корзину объект интереса;
- BUY (6): пользователь осуществил покупку объект интереса.

КОРРЕКТИРУЮЩИЕ КОЭФФИЦИЕНТЫ

- отрицательные веса;
- коэффициент затухания.

Расчет оценки релевантности с учетом коэффициента затухания:

$$q(k) = y - d_k$$

k – текущая информация об объекте интереса относительно пользователя;
 y – список атрибутов вместе с условными вероятностями;
 d_k – текущее значение отрицательного веса данного объекта интереса

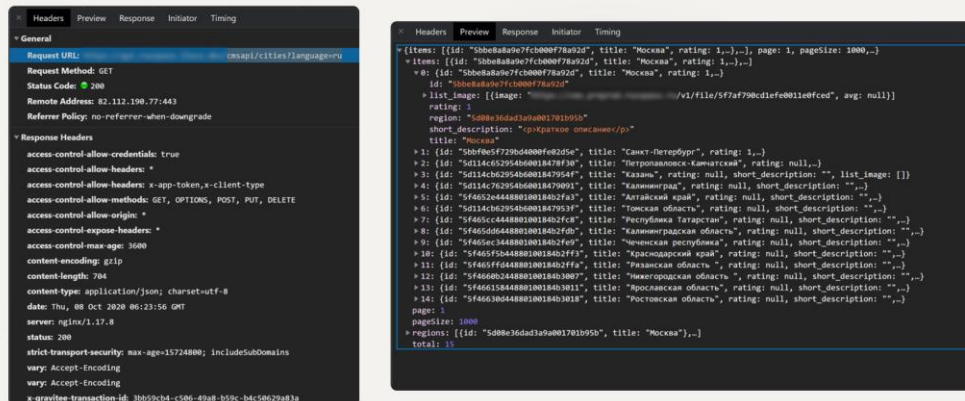
Значение отрицательного веса	Оценка пользователя
0 (0,02)	6 - покупка
0,18	5 - бронирование / доавление в корзину
0,36	4 - добавление в избранное
0,52	3 - просмотр детализации
0,68	2 - переход на страницу объекта интереса
0,84	1 - поиск
1	0 - отсутствие взаимосвязи

UML ДИАГРАММА АКТИВНОСТИ ТУРИСТИЧЕСКОЙ ПЛАТФОРМЫ

АЛГОРИТМ СОСТАВЛЕНИЯ РЕКОМЕНДАЦИЙ

```
graph TD
    Start([Начало]) --> Input1[/Ввод информации о новых пользователях/]
    Input1 --> Input2[/Ввод информации о новых товарах/]
    Input2 --> Input3[/Ввод информации о новых данных о предпочтениях/]
    Input3 --> GetDB[Получение товаров и услуг из БД]
    GetDB --> CheckData{Проверка наличия данных}
    CheckData -- да --> GetRec[Получение списка рекомендованных товаров]
    GetRec --> LoadArray[Загрузка данных в общий массив]
    LoadArray --> OutputUpdate[/Вывод информации об обновлении/]
    OutputUpdate --> Start
    CheckData -- нет --> OutputNone[/Вывод информации об отсутствии данных/]
    OutputNone --> LoopStart((1))
    LoopStart --> CalcPass[Считаем 1-ый проход]
    CalcPass --> TrainModel[Обучение модели]
    TrainModel --> ExecRec[Выполнение рекомендательного процесса]
    ExecRec --> ExecOpt[Выполнение оптимизационного процесса на основе конфигурации параметров]
    ExecOpt --> RecProcess[Рекомендуем рекомендательный процесс]
    RecProcess --> IncI[Увеличиваем i на 1]
    IncI --> OutputDB[Вывод информации о завершении работы в БД]
    OutputDB --> WriteRes[Запись результатов в БД]
    WriteRes --> OutputEnd[Вывод информации о завершении работы процесса]
    OutputEnd --> End([Конец])
    LoopStart --> LoopEnd((2))
    LoopEnd --> End
```

Запрос на получение списка городов:



13

Запрос на заполнение отношений
пользователь-продукт-оценка:

```
Метод: POST
URL: reco/api/rating
Тело запроса:
[[
  {
    "uid": "1",
    "id": "5e598dcdba6bb300122c582d",
    "rating": "SEARCH"
  },
  {
    "uid": "1",
    "id": "5e599723da6bb300122c5acf",
    "rating": "FAVORITE"
  }
]]
Ответ:
200 OK
```

Запрос на получение списка релевантных туров:

```
Method: POST
URL: reco/api/recommendation/tours
Teno aopoca:
[[
  {
    "uid": "string",
    "language": "string",
    "city": "string",
    "gte": "string",
    "lte": "string",
    "tags": "string",
    "page": number,
    "size": number,
  }
]]
Orbet:
  200 OK
{
  "page": number,
  "pageSize": number,
  "total": number,
  "items": [
    {
      "id": "string"
    }
  ]
}
Bad Request
{ "errorMessage": "string" }
```

РЕЗУЛЬТАТ РАБОТЫ СИСТЕМЫ

Метрика качества ранжирования:

$$accuracy_k = \frac{\sum_{i=1}^k (Relevant_i * Predication_i)}{\sum_{i=1}^k Predication_i} * 100$$

Relevant_i – вероятностный показатель релевантности объекта интереса;

Predication_i – предполагаемый показатель заинтересованности пользователя в объекте интереса

	uid [PK] character varying (255)	lid [PK] character varying (255)	prediction double precision
1	0073d93f-1937-4769-856a-395ece1...	7d557046-f183-4edf-b0f9-701a600d...	0.464531984771238
2	0073d93f-1937-4769-856a-395ece1...	a82a44a6-dbb4-423f-ac08-65d7aa4...	0.230038522794868
3	0073d93f-1937-4769-856a-395ece1...	c32ad53b-b242-4e45-8c43-86fb7c1...	0.9496927120346776
4	0073d93f-1937-4769-856a-395ece1...	e7820871-a80b-4ab3-b455-29bdb9...	0.123500256281787
5	2e3bd807-0d56-4087-9838-704899...	61b4485b-3050-4624-a9cb-299316...	0.1143079064819834
6	2e3bd807-0d56-4087-9838-704899...	a82a44a6-dbb4-423f-ac08-65d7aa4...	0.9015127936426466
7	2e3bd807-0d56-4087-9838-704899...	c32ad53b-b242-4e45-8c43-86fb7c1...	0.302466889308357
8	2e3bd807-0d56-4087-9838-704899...	e7820871-a80b-4ab3-b455-29bdb9...	0.561668500932056

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено исследование существующих туристических платформ и современных подходов к формированию релевантных подборок, а также была проанализирована проблема информационного пузыря

Была разработана рекомендательная система туристических маршрутов, обеспечивающая сокращение временных затрат пользователя на поиск туристического маршрута и повышение востребованности данных объектов за счет добавления в подборку совершенно новых или не просмотренных ранее

Основные положения и выводы данной диссертации базировались на анализе подходов к формированию рекомендательных подборок в рамках хозяйственной научно-исследовательской работы «Исследование способов построения рекомендательной системы туристических маршрутов».

СПАСИБО ЗА ВНИМАНИЕ!

