Санкт-Петербургский Государственный Университет
Математико-механический факультет

Кафедра системного программирования

Илья Игоревич Келим

Система комбинированных рекомендаций эстетичных изображений с применением коллаборативной фильтрации

Бакалаврская работа

Научный руководитель:
доцент кафедры СП, к. т. н. Брыксин Т. А.

Рецензент:
Инженер-программист в компании Empath inc. Гурин В. С.

Консультант:
Основатель “graphica.ai” Брыксин М. А.

Санкт-Петербург
2020

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering

Ilya Kelim

Hybrid recommendation system for aesthetic images using collaborative filtering approach

Graduation Thesis

Scientific supervisor:
PhD Timofey Bryksin

Consultant :
Founder of "graphica.ai" Matvey Bryksin

Reviewer:
Software Developer in Empath inc. Vlad Gurin

Saint-Petersburg
2021

Оглавление

Введение

Объем информации, доступной в интернете, растет с каждым годом. В связи с этим все большую популярность набирают рекомендательные системы. Их основная задача — автоматически предлагать пользователям тот контент, который их заинтересует.

Рекомендательные системы полезны как пользователям, так и платформам. Автоматизация поиска и фильтрации контента позволяет пользователю тратить меньше времени на нерелевантную информацию и получать только самую интересную и актуальную. В идеальном случае рекомендательная система должна сразу выдавать человеку те элементы, которые он сам бы выбрал, посмотрев все предлагаемые варианты. Чем проще пользователю найти подходящий контент, тем больше удовольствия он получает от использования сервиса. Что, в свою очередь, повышает его конкурентоспособность и может положительно влиять на конверсию.

Подбор и рекомендации похожих материалов особенно актуальны для современных дизайнеров. Первый этап работы над любым проектом — это поиск и анализ других работ на схожие темы с целью вычленения и заимствования успешных элементов и приемов. Из-за этого дизайнеры вынуждены тратить много времени на просмотр и отсев нерелевантного контента. Поскольку даже в конкретной области дизайнерских изображений множество доступных элементов слишком велико для человеческого восприятия, остро стоит проблема необходимости рекомендательных систем.

Взаимодействие пользователя с рекомендательной системой в целом происходит по одному и тому же сценарию вне зависимости от ее внутреннего устройства. Пользователь смотрит на контент, предоставленный на платформе, и какими-то действиями показывает интерес к некоторым элементам. В ответ на это система выдает ему рекомендации.

Существует три основных подхода к созданию рекомендательных систем: основанный на содержании, то есть использующий информа-

цию исключительно о контенте, персонализированный, то есть использующий информацию исключительно о пользователях, и комбинированный. Комбинированный подход позволяет реализовать решение, сочетающее достоинства двух предыдущих подходов, и лишенное их недостатков, однако требует для своей работы значительно больше данных.

Постановка задачи

Цель этой работы — реализовать систему комбинированных рекомендаций эстетичных изображений, использующую коллаборативную фильтрацию. Для достижения этой цели поставлены следующие задачи.

- Исследовать подходы к реализации коллаборативной фильтрации.
- Разработать инфраструктуру поддержки пользователей.
- Создать модуль экспорта и обработки данных из системы аналитики.
- Спроектировать и разработать модуль коллаборативной фильтрации.
- Интегрировать модуль коллаборативной фильтрации с рекомендательной системой проекта graphica.ai.
- Провести апробацию в проекте graphica.ai.

1. Обзор

В данной главе рассмотрены подходы к реализации коллаборативной фильтрации, способы сбора данных для нее и пользовательской аналитики, варианты аутентификации пользователя в системе, а также обоснован выбор используемых сторонних сервисов и алгоритмов для достижения поставленной цели.

1.1. Коллаборативная фильтрация

Используемые данные

Для корректной работы системы коллаборативной фильтрации требуется информация о предпочтениях пользователя. Обычно эти данные предоставляются в формате оценок различных элементов пользователем по фиксированной шкале интереса. Некоторые сервисы не дают человеку возможность оценить контент напрямую. Тогда вместо этого приходится использовать неявные оценки, такие как сохранения и просмотры. В таком случае для применения коллаборативной фильтрации неявные оценки требуется конвертировать в явные [1]. Например, если в качестве неявной оценки используются сохранения, то отношение пользователя к элементу можно представить в виде дискретной шкалы от нуля до единицы, отмечая сохраненные элементы единицей, а не сохраненные — нулем [2].

Подход, основанный на соседстве

Подход, основанный на соседстве [3], предлагает предсказывать отношение пользователя к новому элементу, используя оценки этого элемента, поставленные другими пользователями, похожими на данного. Существует две разновидности этого подхода:

- основанный на пользователях [4];
- основанный на контенте [5].

Для первого варианта требуется построить матрицу отношения пользователей к контенту, где в каждой строке находится вектор оценок некоторого пользователя. Элементам, которые не были оценены, проставляется минимальное значение шкалы интереса. Для выдачи рекомендаций конкретному пользователю в построенной матрице находятся вектора, наиболее похожие на вектор исходного пользователя. Из этих векторов извлекаются положительно оцененные элементы и сортируются в порядке группового интереса, то есть чем больше положительных оценок получил элемент, тем выше он в выдаче. Отсортированный список элементов показывается пользователю.

Вариант, основанный на контенте, предполагает построение матрицы оценок элементов, где в каждой строке находится вектор, содержащий оценки, данные всеми пользователями этому элементу. Если элемент не оценен каким-то пользователем, отсутствующее значение заменяется минимальным по шкале интереса. Когда пользователь проявляет интерес к какому-то контенту, система ищет в построенной матрице элементы, вектора которых похожи на вектор исходного элемента, и выдает результат пользователю, сортируя по схожести.

Для определения похожести векторов чаще всего используются метрики сходства по Пирсону и косинусного сходства [3].

Подход, основанный на модели

Идея подхода, основанного на модели, заключается в использовании алгоритмов машинного обучения для выдачи рекомендаций. Сначала строится матрица, аналогичная используемой в подходе, основанном на соседстве. Затем на этой матрице происходит обучение модели, которая будет предсказывать отношение пользователя к контенту.

Для реализации этого подхода часто применяются алгоритмы кластеризации и понижения размерности [3, 5].

Идея алгоритмов кластеризации [6, 7] заключается в разбиении пользователей на кластеры, используя матрицу отношения пользователей к контенту. В качестве рекомендаций пользователю выдаются элементы, которые наиболее популярны у его кластера, за исключением тех,

которые он уже видел. Преимущества этого метода заключаются в более высокой скорости работы по сравнению с подходом, основанным на соседстве, так как вместо поиска ближайших соседей используется информация о кластерах, посчитанная заранее. Недостатками этого метода считаются необходимость регулярно пересчитывать кластеры, что может занимать значительное время при большом объеме данных, и медленная реакция на изменения, так как новые данные не оказывают влияния на работу системы, пока не будет произведен пересчет кластеров. Дополнительная сложность этого подхода заключается в том, что не всегда известны количество и размер кластеров, которые должен обнаружить алгоритм, поэтому может потребоваться тонкая настройка алгоритма под конкретную задачу.

Алгоритмы понижения размерности [8] предполагают разбиение матрицы отношения пользователей к товарам на произведение двух матриц значительно меньшей размерности, где первая матрица отвечает за хранение информации о пользователях, вторая — о контенте. Произведение столбца пользователя из первой матрицы на строку элемента из второй выдает отношение пользователя к элементу контента.

Снижение размерности позволяет существенно уменьшить время выдачи рекомендаций и сократить затраты на хранение данных. Недостатком этих алгоритмов является потеря интерпретируемости, поскольку новые матрицы уже не содержат в себе информации, понятной человеку. Дополнительно подход имеет схожие недостатки с алгоритмами кластеризации: медленная реакция на изменения и длительность подготовки к работе. Наиболее популярными подходами к снижению размерности в области коллаборативной фильтрации сейчас считаются сингулярное разложение [9] и матричная факторизация [10]. Исследования показывают, что эти алгоритмы понижения размерности позволяют достичь наилучших результатов по сравнению с другими подходами [11].

1.2. Сохранение пользовательских действий

Хранение в базе данных

Одним из распространенных способов сохранения информации о действиях пользователей является использование базы данных. Это может быть как база данных, уже используемая в проекте для других целей, так и отдельное решение, выделенное исключительно для хранения пользовательских действий. В качестве отдельного решения могут выступать облачные базы данных, самыми популярными представителями которых являются Hadoop[1] и Clickhouse[2], либо отдельная локальная база данных.

Этот подход позволяет получать доступ к собранным данным в реальном времени, гибко настраивать формат хранения и избежать накладных расходов на интеграцию со сторонними сервисами. Ключевым недостатком этого решения является необходимость выделения дополнительных ресурсов на сервера и базы данных проекта, чтобы они могли справляться с обработкой постоянного потока данных о действиях пользователей. Другим существенным недостатком является ограниченность этого подхода, так как он дает возможность только собирать и хранить данные, а для интерпретации данных, подсчета статистики и построения аналитики потребуется дополнительная работа.

Использование систем аналитики

Другим распространенным способом сохранения данных о пользовательских действиях является использование систем аналитики. Система аналитики — это сервис, предоставляющий возможность сохранения действий пользователей и подсчета статистики. Основные преимущества использования внешней системы аналитики — широкий спектр предоставляемых возможностей. Большинство систем аналитики позволяют не только собирать и хранить данные, но и предоставляют встроенные примитивы для построения аналитик и подсчета статисти-

[1] `hadoop.apache.org`

[2] `clickhouse.tech`

ки. Часто для сбора аналитики используются OLAP системы, позволяющие получать доступ к данным в реальном времени. К недостаткам можно отнести необходимость интеграции со внешней системой и необходимость выгружать данные для применения в рекомендательной системе.

Существует множество различных систем аналитики, одни из самых популярных — это Google Analytics[3] и Amplitude[4], хотя эти системы не позволяют пересчитывать аналитику в реальном времени, допуская задержку в несколько минут.

Достоинствами системы Google Analytics являются возможность выгрузки данных с помощью BigQuery[5] и наличие собственных библиотек analytics.js[6] и Firebase SDK[7] для интеграции с веб-сайтами и мобильными приложениями. К недостаткам можно отнести жесткие ограничения на формат собираемых данных и сложность настройки системы для сбора нетипичных событий.

Система Amplitude также предлагает свои библиотеки для интеграции с мобильными приложениями[8] и вебсайтами[9], предоставляет возможность выгрузки данных и практически не накладывает ограничений на формат хранимых данных, то есть дает возможность собирать информацию о любой пользовательской активности и строить по ней базовую аналитику. Недостатками этой системы являются отсутствие возможности выбрать, какие конкретно данные требуется выгрузить — можно указать только интересующий временной период, и необходимость конвертации данных из предоставляемого формата в удобный для работы.

[3] analytics.google.com
[4] amplitude.com
[5] cloud.google.com/bigquery
[6] developers.google.com/analytics/devguides/collection/analyticsjs
[7] firebase.google.com/docs/firestore/client/libraries
[8] developers.amplitude.com/docs/ios
[9] developers.amplitude.com/docs/javascript

1.3. Аутентификация

Локальная аутентификация

Для аутентификации пользователя в системе могут использоваться ресурсы серверов проекта. В контексте языка Python[10] и библиотеки Django[11] самыми популярными способами являются встроенные в Django возможности аутентификации и использование библиотеки Authlib[12].

Преимуществом локальной аутентификации является возможность полностью контролировать процесс, не полагаясь на другие сервисы. К недостаткам относятся необходимость обеспечения безопасности хранения паролей, потребность в дополнительных ресурсах, нужных механизму аутентификации, и необходимость отдельно поддерживать регистрацию с использованием других сервисов, таких как Google[13], Facebook[14], Apple[15] и других.

Использование систем аутентификации

Другая возможность поддержки возможности аутентификации — это использование сторонних сервисов. Преимущества этого подхода заключаются в том, что внешний сервис берет на себя ответственность за хранение паролей а также все накладные расходы на процесс аутентификации. Также многие системы аутентификации предоставляют возможность разрешить вход через другие известные сервисы. Недостатком этого подхода является необходимость интеграции со внешней системой.

Одними из самых популярных систем аутентификации, являются Amazon Cognito[16] и Firebase Authentication[17].

[10] `www.python.org`
[11] `www.djangoproject.com`
[12] `github.com/lepture/authlib`
[13] `google.com`
[14] `facebook.com`
[15] `apple.com`
[16] `aws.amazon.com/cognito/`
[17] `firebase.google.com`

Сервис Amazon Cognito предлагает свои библиотеки для интеграции[18] и предоставляет возможность входа через популярные сервисы и социальные сети. Недостатками этого сервиса являются сложность интеграции и настройки, редко обновляемая документация и ограничения на количество данных, хранимых о пользователях

Сервис Firebase Authentication так же предлагает свои библиотеки для интеграции и возможность входа через популярные социальные сети и сервисы. Так же это решение позволяет получить доступ к части информации, собранной компанией Google[19] о клиенте, например, страна проживания, имя и изображение профиля. Сервис Firebase Authentification прост в настройке и использовании.

1.4. Выводы

Для реализации коллаборативной фильтрации был выбран алгоритм понижения размерности, так точность и скорость выдачи рекомендаций в данном случае важнее оперативности обновления и минимизации затрачиваемых ресурсов.

В качестве решения для сбора данных о действиях пользователей был выбран сторонний сервис Amplitude, так как он позволяет собирать информацию о любой пользовательской активности, строить по ней аналитику, и дает возможность регулярно выгружать данные для рекомендательной системы. Так как для реализации коллаборативной фильтрации был выбран алгоритм понижения размерности, доступ к данным в реальном времени не требуется.

Для аутентификации была использована внешняя система Firebase Authentication, так как она предоставляет возможность аутентификации как с помощью адреса электронной почты и пароля, так и через сторонние сервисы, и дает возможность не заниматься обеспечением безопасности хранения паролей.

[18] aws.amazon.com/amplify/

[19] google.com

2. Архитектура и реализация

В данной главе рассматривается разработанная система рекомендаций эстетичных изображений и описывается ее архитектура и реализация.

2.1. Архитектура системы

В результате работы была реализована комбинированная система рекомендаций эстетичных изображений, предоставляющая следующие возможности:

- рекомендация изображений;
- рекомендация похожих пользователей;
- генерация случайной обновляющейся ленты;
- выдача персонализированных рекомендаций по изображению.

Основными компонентами системы являются четыре модуля.

- Модуль экспорта и обработки данных. Этот модуль отвечает за выгрузку событий из системы аналитики, построение локальной пользовательской аналитики и перевод данных в вид, подходящий для коллаборативной фильтрации.
- Модуль коллаборативной рекомендательной системы, отвечающий за поиск похожих пользователей и рекомендацию изображений.
- Модуль рекомендательной системы, основанной на содержании, отвечающий за выдачу рекомендаций по изображению. Этот модуль был разработан в рамках курсовой работы прошлого года.
- Модуль коллаборативной фильтрации, отвечающий за генерацию случайной обновляющейся ленты и выдачу персонализированных рекомендаций по изображению.

Проект graphica.ai реализован на языке Python с использованием библиотеки Django, поэтому все модули были также реализованы на языке Python. В качестве базы данных в проекте используется PostgreSQL[20]. Взаимодействие с сервером происходит на языке запросов graphql[21], обработка запросов на сервере реализована с помощью библиотеки graphene[22].

Человек может воспользоваться сервисом с помощью любого клиента, например, iOS мобильного приложения или веб-сайта. Чтобы получить доступ ко всем возможностям продукта, требуется пройти авторизацию. Эта возможность реализована с использованием платформы Firebase Authentication. Данные о действиях пользователей сохраняются в системе Amplitude. Раз в сутки модуль экспорта и обработки данных загружает новые данные и производит разбиение пользователей на кластеры, используя информацию об их действиях. Также ежедневно происходит подсчет аналитики для всех пользователей и кластеров. Обработанные данные передаются в модуль коллаборативной рекомендательной системы, которая позволяет пользователю получить список других, похожих на него, пользователей и список рекомендованных лично ему изображений. Модуль коллаборативной фильтрации использует рекомендательную систему на основе содержания и модуль коллаборативной рекомендательной системы, чтобы предоставить пользователю случайную обновляющуюся ленту и персонализированные рекомендации по изображению. Общая архитектура системы представлена на рисунке 1.

[20] `www.postgresql.org`

[21] `https://graphql.org/`

[22] `https://github.com/graphql-python/graphene`

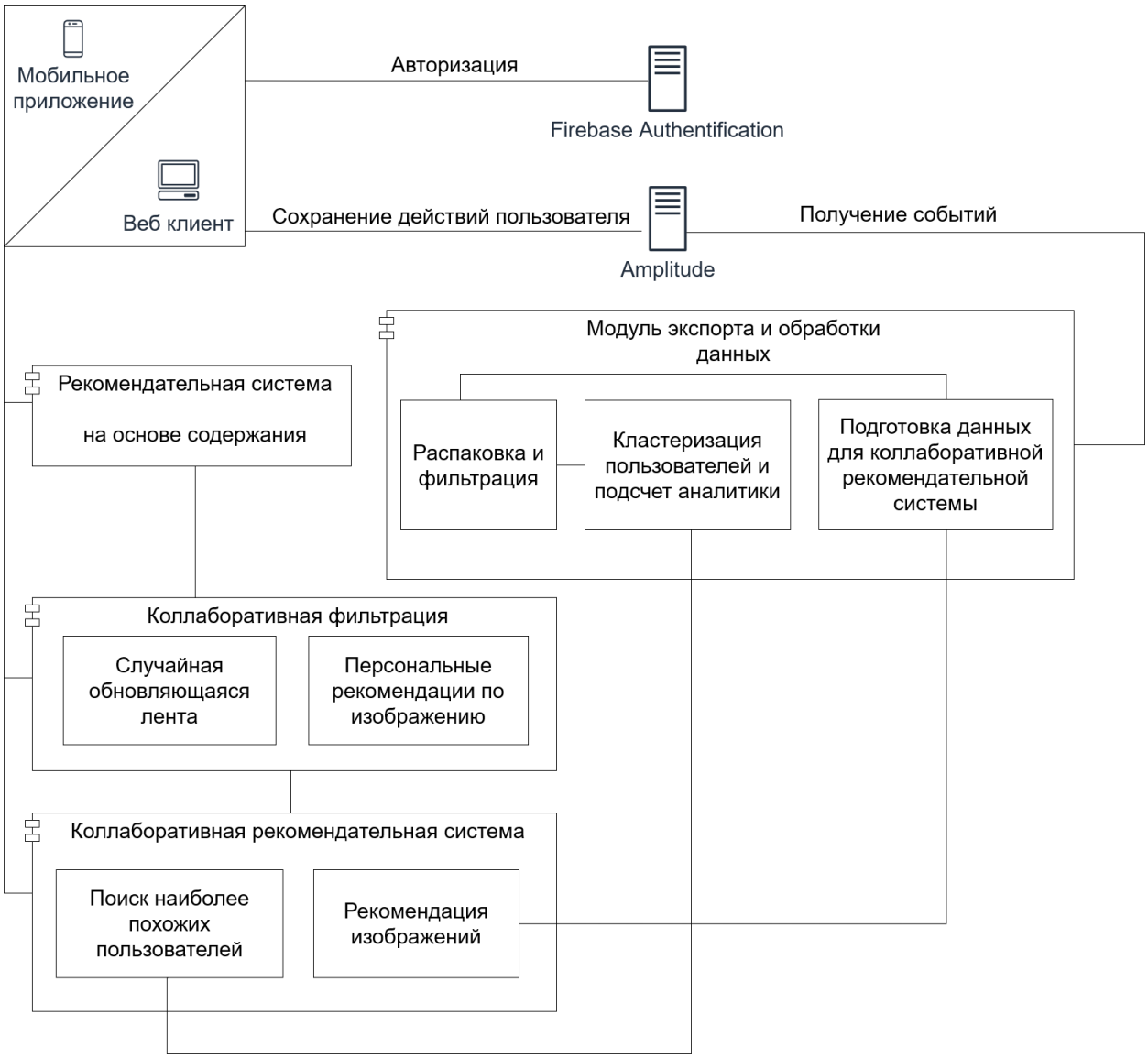

Рис. 1: Архитектура системы.

2.2. Инфраструктура поддержки пользователей

Для персонализации необходимо, чтобы система имела возможность различать людей, которые ей пользуются. Для достижения этой цели в существующую инфраструктуру проекта была добавлена модель пользователя.

Сущность пользователя

Для интеграции сущности пользователя в систему сначала был реализован класс User, позволяющий хранить для каждого пользователя

информацию, необходимую для работы сервиса:

- уникальный идентификатор объекта;
- адрес электронной почты, имя и описание;
- уникальный ключ firebaseId, позволяющий связать пользователя с информацией, хранимой о нем в системе Firebase Authentication;
- библиотеки, созданные пользователем, и изображения, сохраненные в них;
- изображения, добавленные пользователем;
- текущая подписка;
- кластер, к которому относится пользователь.

Для того, чтобы пользователи могли проявлять свое отношение к изображениям и сохранять те, которые им понравились, в систему была введена сущность библиотеки изображений. Для ее интеграции был реализован класс Library, устроенный таким образом:

- уникальный идентификатор объекта;
- название;
- сохраненные изображения.
- статус библиотеки — она может быть открыта для всех или только для создателя;
- ссылка на родительскую библиотеку;

С помощью библиотек Django и plsql[23] описанные выши классы были отражены в таблицы User и Library базы данных. Устройство этих классов продемонстрировано на рисунке 2.

[23] `www.postgresql.org/docs/13/app-psql.html`

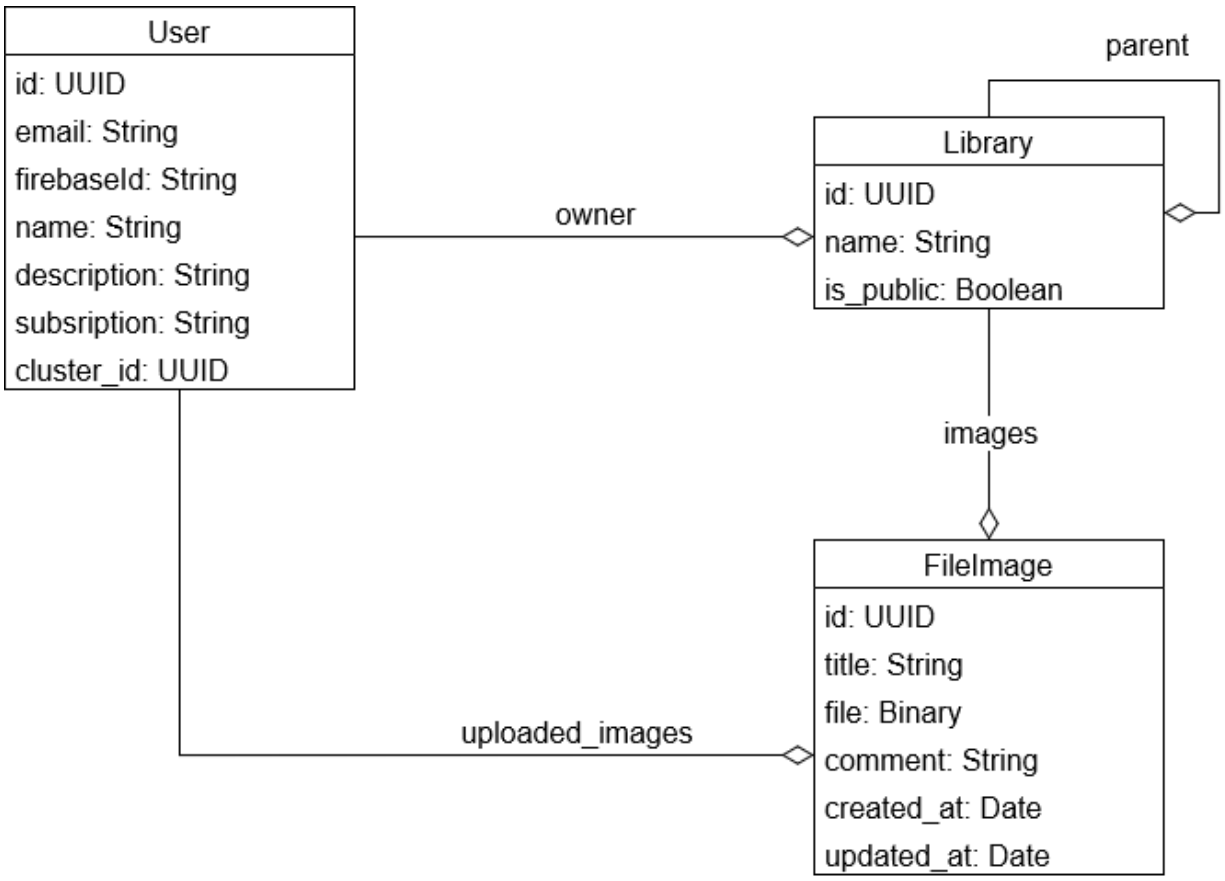

Рис. 2: Диаграмма классов для сущности пользователя.

Клиент может получить полную информацию о пользователе, предоставив JWT[24] токен, содержащий зашифрованный, уникальный для каждого пользователя, ключ firebaseId, полученный из системы Firebase Authentication в процессе аутентификации, или получить частичный доступ без ключа, позволяющий увидеть имя пользователя, публичные библиотеки и описание.

Аутентификация

Для аутентификации пользователей используется внешняя система Firebase Authentication, которая была интегрирована с существующей системой.

Процесс первичной регистрации пользователей в сервисе graphica.ai представлен на рисунке 3 и состоит из трех шагов:

[24] `jwt.io`

- Клиент отправляет адрес электронной почты и пароль клиента в Firebase Authentication.
- Firebase Authentication отправляет в ответ ключ, уникально идентифицирующий пользователя.
- Клиент отправляет на сервер проекта запрос на создание нового пользователя с полученным и ключом и дополнительной информацией, если она известна.

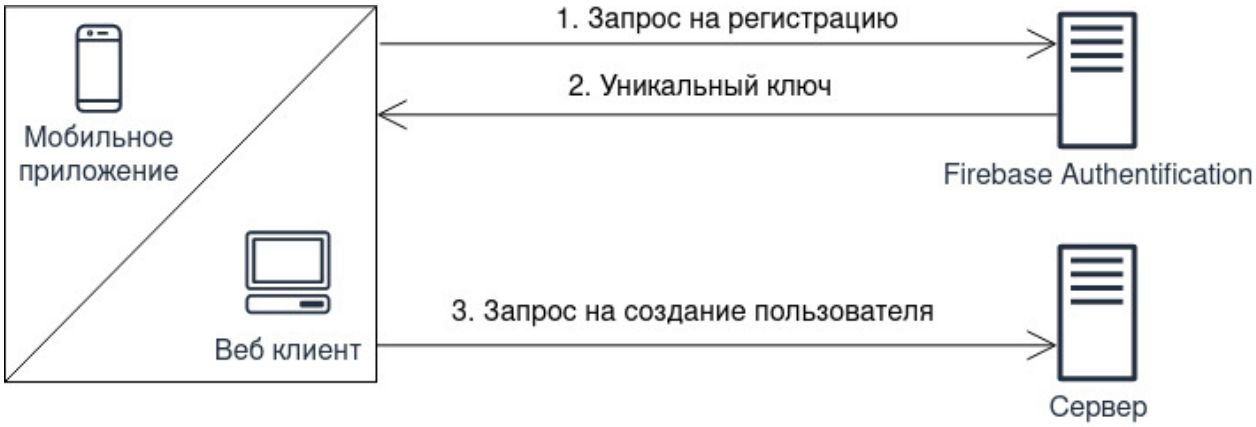

Рис. 3: Процесс регистрации нового пользователя.

При аутентификации клиент отправляет данные пользователя в Firebase Authentication, получая в ответ JWT[25] токен, который включается в каждый последующий запрос к серверу. Токен расшифровывается на сервере с использованием библиотеки firebase-admin[26], чтобы получить firebaseId и идентифицировать пользователя. В случае, если токен устарел, сервер возвращает ошибку с кодом 401, и приложение обновляет токен.

2.3. Система аналитики

В качестве системы сбора информации о действиях пользователей была использована внешняя система сбора аналитики Amplitude.

[25] `jwt.io`
[26] `https://github.com/firebase/firebase-admin-python`

Собираемые данные

В систему аналитики Amplitude сохраняется информация о двадцати различных действиях. Из них для коллаборативной фильтрации было отобрано шесть, позволяющих оценить отношение пользователя к изображению:

- просмотр изображения из ленты;
- просмотр изображения из рекомендаций;
- сохранение изображения в библиотеку с экрана изображения;
- сохранение изображения в библиотеку с экрана списка изображений;
- сохранение изображения в избранное с экрана изображения;
- сохранение изображения в избранное с экрана списка изображений;

2.4. Модуль экспорта и обработки данных

Для получения и обработки данных из системы аналитики был разработан модуль, содержащий шесть классов:

- Event — вспомогательный класс для хранения данных о запросе;
- Info — вспомогательный класс для хранения данных об аналитике;
- EventLoader — класс, отвечающий за загрузку и распаковку данных из Amplitude;
- EventParser — класс, отвечающий за перевод данных из json в Event;
- Handler — класс, предоставляющий интерфейс для работы с данными (загрузка, обновление, сохранение в формате DataFrame);

- Manager — класс, выступающий в качестве фасада для этого и других разработанных в рамках данной работы модулей;

Разработанный модуль предоставляет следующие возможности:

- загрузка и обновление данных о запросах;
- сохранение загруженных данных в формате DataFrame таблицы, содержащей только интересующую нас информацию о запросах;
- представление данных о предпочтениях пользователя в понятном человеку формате.

Экспорт данных

Система аналитики Amplitude предоставляет возможность выгрузить собранные данные, используя GET запрос с параметрами StartDate и EndDate, ограничивающими интересующий период времени. Также для этого запроса требуются два секретных ключа: API_Key и Secret_Key, индивидуальных для каждого проекта. В ответ на этот запрос отправляются данные обо всех событиях за определенный период. Каждое событие содержит в себе информацию о своем типе, идентификаторе пользователя, идентификаторе изображения, времени записи и прочее. Данные предоставляются в формате zip архива, содержащего сжатые json файлы в формате gzip, содержащие события. Загрузка событий происходит с помощью класса EventLoader. Полученные файлы распаковываются с помощью стандартных библиотек gzip[27] и zipfile[28]. Затем с помощью класса EventParser из полученных json файлов извлекаются события, выбранные для коллаборативной фильтрации и из каждого берется информация об идентификаторе пользователя, идентификаторе изображения и типе события. Полученные данные сохраняются в объект класса DataFrame[29], предоставляемый библиотекой pandas[30]. Полученный объект сохраняется в файл формата csv.

[27] github.com/python/cpython/blob/3.9/Lib/gzip.py

[28] github.com/python/cpython/blob/3.9/Lib/zipfile.py

[29] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html?highlight=dataframe#pandas.DataFrame

[30] pandas.pydata.org

Обновление данных производится раз в сутки. В качестве StartDate передается дата последнего сохраненного события, в качестве EndDate — текущая дата или дата начала сбора аналитики, если загрузка производится впервые. Если csv файл с событиями уже существует, он дополняется новой информацией.

Подсчет аналитики и кластеризация

В отличии от системы аналитики сервер обладает информацией об изображениях, с которыми взаимодействовали пользователи, поэтому именно на сервере существует возможность посчитать понятную человеку аналитику предпочтений пользователей. Для каждого пользователя считаются три метрики, показывающие его предпочтения:

- 3 любимых автора;
- 3 любимых категории;
- любимый источник изображений (сайт, с которого они были взяты).

Для подсчета этих метрик используется сохраненный csv файл. Для каждого пользователя выбираются изображения, которые он посмотрел или сохранил. Для каждого изображения известна информация о его авторе, источнике и категориях. В качестве результатов выбирается три автора, с изображениями которых пользователь взаимодействовал больше всего, и, аналогично, три категории и источник. Полученная аналитика сохраняется в базу данных в таблицу Analytics.

Для проверки собранных данных на предмет возможности делать рекомендации по ним был использован механизм кластеризации. Базовое предположение заключается в том, что если кластеризация отработает удовлетворительно на имеющихся данных, то выбранные данные несут в себе достаточно информации для использования подхода коллаборативной фильтрации. Для кластеризации использовалась реали-

зация алгоритма MeanShift[31] из библиотеки sklearn[32]. Этот алгоритм и реализация были выбраны, так как они уже используются в проекте для задач кластеризации и показывают хорошие результаты и производительность [12]. В качестве данных для кластеризации выступает матрица, содержащая в строках пользователей, в столбцах изображения, а на пересечении отношения. Эта матрица строится с помощью сохраненного csv файла с событиями. Отношения выставляются таким образом: если пользователь не взаимодействовал с изображением, отношение выставляется в ноль, если пользователь посмотрел изображение — в единицу, а если сохранил — в двойку.

Поскольку оценить степень похожести пользователей мы можем только по метрикам, используемым для аналитики, среднее этих метрик было посчитано для каждого кластера. Результат представлен на рисунке 5.

Как видно по рисункам 4 и 5, пользователи, собранные в один кластер, имеют схожие интересы в плане авторов и категорий. Например, все пользователи, попавшие в кластер 0, заинтересованы в категории “Typography Textures” и изображениях автора “Wayne Season”.

Полученные кластеры и информация о принадлежности пользователей к ним сохраняются в базу данных в таблицу Cluster для дальнейшего использования.

Для удобства администраторов сервиса graphica.ai с использованием библиотеки Django был разработан пользовательский интерфейс, позволяющий получить доступ к полной информации о пользователе и собранной аналитике. Так же предоставляется возможность изменения любых данных вручную. Результат представлен на рисунке 4.

[31] scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html?highlight=meanshift#sklearn.cluster.MeanShift

[32] sklearn.org

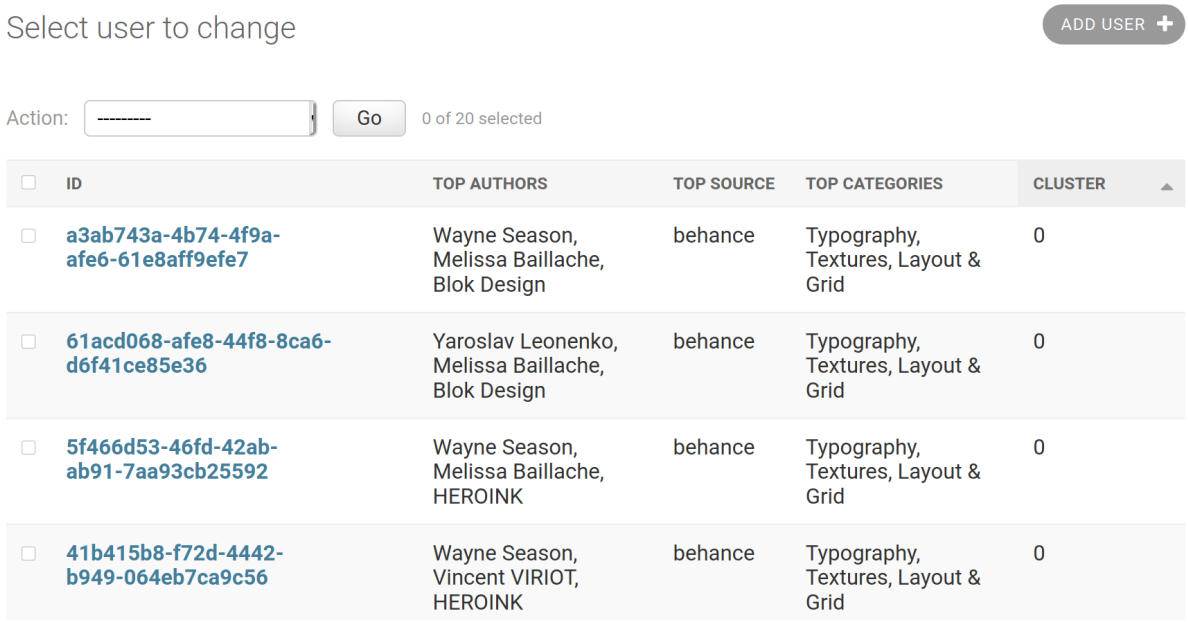

Рис. 4: Пользовательский интерфейс с аналитикой по пользователю.

NUMBER	AUTHORS	SOURCE	CATEGORIES
0	Vincent VIRIOT, Wayne Season, HEROINK	behance	Typography, Textures, Layout & Grid
1	József G Kiss, Greg Stewart, Aneta Lewandowska	behance	Colors, Identity Elements, Packaging
2	Wayne Season, Rron Berisha, moodley design	behance	Identity Elements, Books & Journals, Packaging

3 clusters

Рис. 5: Пример вычисления метрик для кластеров, посчитанных на пользователях с рисунка 4.

2.5. Коллаборативная рекомендательная система

Для выдачи персональных рекомендаций был разработан модуль коллаборативной рекомендательной системы. Основным классом выступает Recommender, а в качестве фасада используется класс Manager. Модуль предоставляет следующие возможности:

- обновление весов для модели машинного обучения;
- предсказание отношения пользователя к заданным изображениям;

- отсев уже просмотренных пользователем изображений;
- выдача персонализированных рекомендаций;
- выдача похожих пользователей;

Сравнение алгоритмов понижения размерности

Было проведено сравнение двух самых популярных алгоритмов понижения размерности: сингулярное разложение и матричная факторизация. Чтобы узнать, какой из них дает лучшие результаты в области дизайнерского контента, оба алгоритма были реализованы с использованием самой популярной библиотеки машинного обучения tensorflow[33].

Подбор оптимальных параметров проводился методом grid search на собранных ранее данных о предпочтениях пользователей, а результаты оценивались с помощью метрик MSE[34], RMSE [35] и MAE[36], позволяющих оценить отклонение выдаваемых алгоритмом результатов от правильных [13]. Оптимальные значения подбирались для всех параметров моделей, которые можно настраивать, то есть: оптимизатор, функция потери, количество итераций обучения и размер партии.

Подбор параметров проводился на данных о 400 взаимодействиях 15 пользователей с 200 изображениями. Подобранные оптимальные параметры для обоих алгоритмов представлены в таблице 2. Результаты, достигнутые алгоритмами за 10 запусков с оптимальными параметрами на собранных данных, представлены в таблице 1. Оба алгоритма оказались достаточно точны для данной задачи, однако алгоритм матричной факторизации показал более высокую точность по всем измеряемым метрикам.

[33] www.tensorflow.org

[34] www.probabilitycourse.com/chapter9/9_1_5_mean_squared_error_MSE.php

[35] towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

[36] www.sciencedirect.com/topics/engineering/mean-absolute-error

Алгоритм	MAE	MSE	RMSE
Сингулярное разложение	0,09 ±0,011	0,09 ±0,011	0,30 ±0,013
Матричная факторизация	0,085 ±0,01	0,082 ±0,015	0,28 ±0,013

Таблица 1: Результаты работы алгоритмов.

Алгоритм	Optimizer	Loss	Epochs	Batch size
Сингулярное разложение	Adam	MAE	100	1024
Матричная факторизация	adagrad	MSE	50	1024

Таблица 2: Оптимальные параметры алгоритмов.

Так же было проведено сравнение времени работы алгоритмов. Для этого каждый алгоритм был запущен 100 раз на одинаковых данных, выбранных случайно. Время работы алгоритмов оказалось идентичным до тысячной доли секунды. Для использования был выбран алгоритм матричной факторизации, так как он оказался лучше по всем измеряемым метрикам.

Рекомендация похожих пользователей

Вторая возможность коллаборативной рекомендательной системы — это выдача списка десяти похожих пользователей. Эта возможность реализована с помощью кластеризации, полученной после подсчета аналитики. Как было показано ранее, пользователи, попавшие в один кластер, имеют схожие предпочтения. Таким образом, в качестве похожих пользователей выдается список людей, попавших в тот же кластер, что и изначальный пользователь.

Клиент может получить доступ к этой функциональности, отправив graphql запрос, содержащий его токен аутентификации, получая в ответ список похожих пользователей.

Рекомендация изображений

На основе выбранного алгоритма была реализована система персонализированных рекомендации изображений, которая работает следующим образом:

- модель машинного обучения обучается на данных, полученных в результате работы модуля экспорта и обработки данных. Переобучение модели происходит раз в сутки после обновления данных о действиях пользователей;
- из сохраненных данных выделяется список картинок, известных модели;
- из списка выбираются изображения, которые пользователь еще не видел;
- с помощью модели предсказываются отношения пользователя к отобранным изображениям;
- 10 изображений с наибольшим рейтингом выдаются в качестве рекомендаций. Ограничение в 10 изображений выставлено для апробации. Так же реализована возможность выдавать список изображений любого размера.

Для доступа к этой функциональности клиенту предоставляется возможность послать graphql запрос, содержащий его токен аутентификации, возвращающий список рекомендованных изображений.

2.6. Коллаборативная фильтрация

На основе модуля коллаборативных рекомендаций и реализованного в рамках предыдущей курсовой работы модуля рекомендаций по содержанию [12], предоставляющего возможность выдавать список изображений, похожих на данное, был реализован модуль коллаборативной фильтрации.

Его основным классом является Manager. Модуль предоставляет следующие возможности:

- выдача случайной обновляющейся ленты;
- выдача персонализированных рекомендаций по изображению.

Случайная обновляющаяся лента

Модуль коллаборативной фильтрации предоставляет возможность генерации случайной обновляющейся ленты для каждого пользователя. Для ее создания применяется указанный ниже процесс:

- С помощью модуля коллаборативной рекомендательной системы генерируется список изображений, рекомендованных пользователю.
- Из них случайным образом выбирается одна.
- С помощью модуля рекомендаций по содержанию находится список картинок, похожих на выбранную.
- Изображения, которые пользователь уже видел, удаляются из списка с помощью модуля коллаборативной рекомендательной системы.
- Полученный список выдается пользователю.

Лента будет отличаться от пользователя к пользователю, так как стартовая картинка выбирается из тех, которые понравятся персонально им. Также, поскольку начальная картинка выбирается случайным образом, лента будет изменятся от запроса к запросу для каждого конкретного пользователя, и это создает возможность разнообразить контент, который видит пользователь. В случае, если модуль коллаборативных рекомендаций не обладает достаточной информацией, чтобы предсказать отношение пользователя к изображениями, в качестве начальной картинки выбирается случайное изображение из базы и выдается результат работы рекомендательной системы по содержанию. Таким образом обходится проблема холодного старта, и сервис может выдавать человеку случайную обновляющуюся ленту без персонализации на начальных этапах.

Для доступа к этой функциональности клиент может отправить graphql запрос, содержащий токен аутентификации. Так же реализо-

вана возможность пагинации, то есть выдачи списка по частям по мере прокрутки пользователем контента, для оптимизации скорости работы.

Примеры случайной обновляющейся ленты в интерфейсе мобильного приложения graphica.ai представлены на изображении 6.

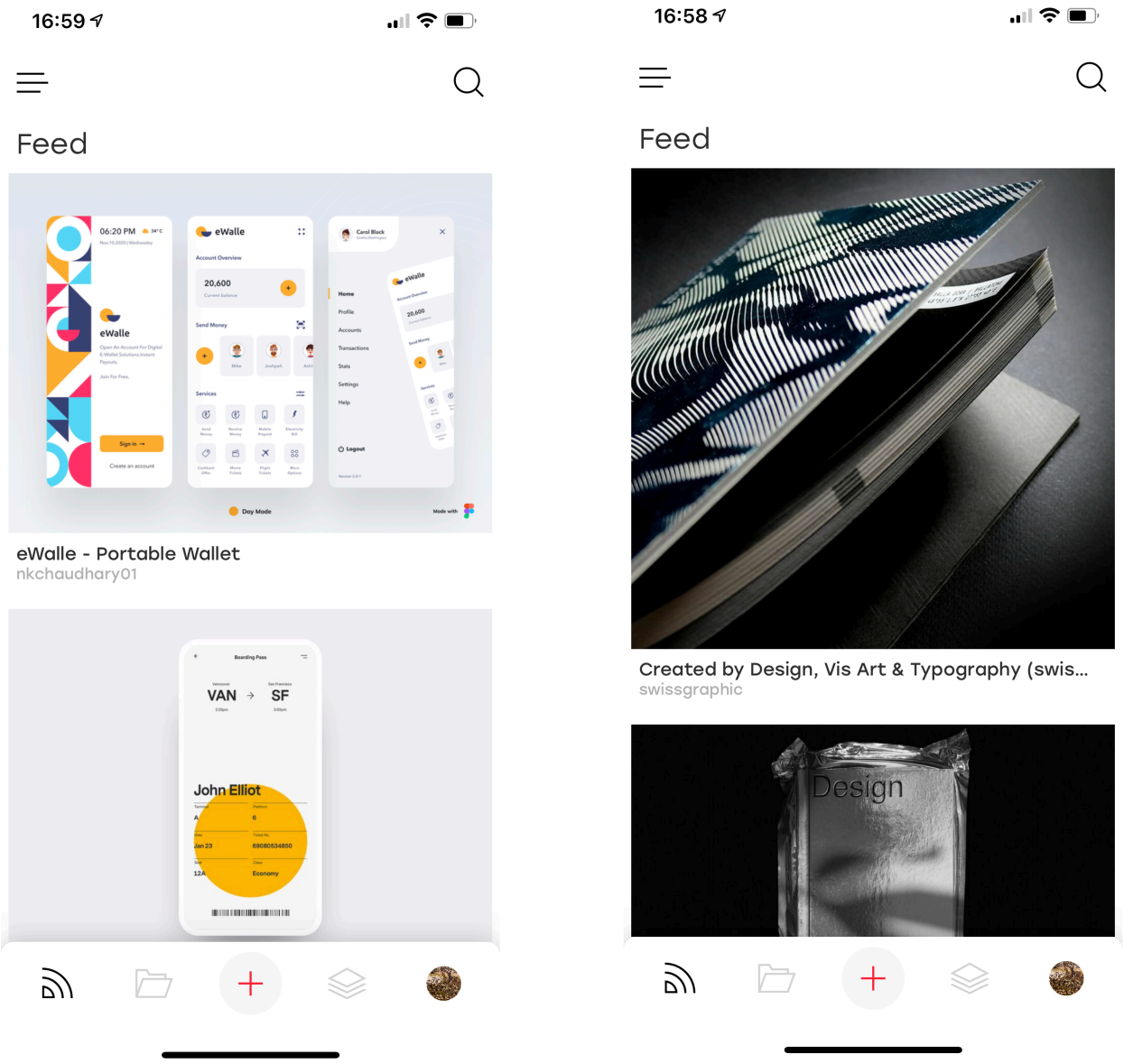

Рис. 6: Случайная обновляющаяся лента.

Персонализированные рекомендации по изображению

Вторая возможность модуля коллаборативной фильтрации — это выдача персонализированных рекомендаций по изображению. Эта возможность реализована следующим образом:

- С помощью модуля рекомендаций по содержанию для выбранного изображения находится список наиболее похожих.
- С помощью модуля коллаборативной рекомендательной системы из списка удаляются изображения, которые пользователь уже видел.
- С помощью модуля коллаборативной рекомендательной системы предсказывается отношение пользователей к картинкам из списка.
- Список сортируется по предполагаемому отношению.
- Для каждой картинки вычисляется новая метрика — сумма ее места в исходном списке и места в отсортированном.
- Список сортируется по посчитанной метрике.
- Итоговый отсортированный список выдается пользователю.

Благодаря сортировке по указанной метрике наверх будут выводиться картинки, которые не просто сильнее похожи на исходную, но и больше понравятся пользователю. В случае, если модуль коллаборативных рекомендаций не обладает достаточной информацией, чтобы предсказать отношение пользователя к изображениям, в качестве рекомендаций выдается отфильтрованный список изображений, похожих на исходное.

Клиент может получить доступ к этой функциональности с помощью graphql запроса, содержащего токен аутентификации. Аналогично случайной обновляющейся ленте, реализована пагинация.

Пример выдачи персонализированных рекомендаций в интерфейсе мобильного приложения graphica.ai представлен на изображении 7.

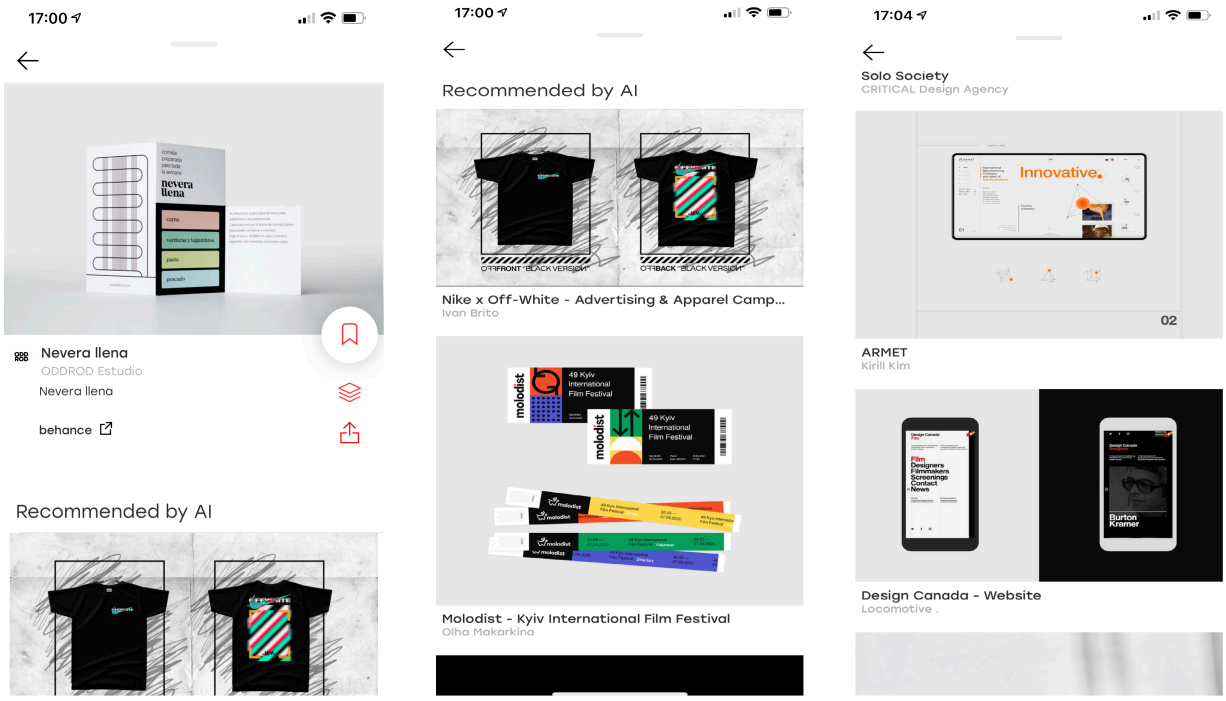

Рис. 7: Персонализированные рекомендации по изображению.

3. Апробация

Основной интересующей нас метрикой при оценке разработанной рекомендательной системы является уровень удовлетворения пользователей. Для оценки этой метрики было проведено тестирование и опрос 10 человек.

3.1. Методика проведения апробации

Для сравнения уровня удовлетворенности пользователей при использовании разных рекомендательных систем были опубликованы два приложения. Одно использовало старую рекомендательную систему проекта graphica.ai, выдающую рекомендации на основании содержания, второе — разработанную в рамках этой работы рекомендательную систему. В остальном приложения были полностью идентичны. Доступ пользователей к приложениям обеспечивался с помощью системы testflight[37], позволяющей людям устанавливать приложения, не опубликованные в App Store[38] — едином магазине приложений компании Apple, на устройства, работающие на операционной системе iOS. В рамках апробации людям предлагалось сначала воспользоваться первым приложением, чтобы оценить старую рекомендательную систему и дать сервису возможность собрать данные об их предпочтениях. Затем воспользоваться вторым приложением, чтобы оценить новую рекомендательную систему и предоставляемые ей возможности. В итоге, с каждым принимающим участие в апробации пользователем проводилось личное интервью, состоящее из трех вопросов.

- Какая рекомендательная система понравилась вам больше?
- Почему выбранная рекомендательная система понравилась вам больше?
- Какие из отобранных для вас изображений вам нравятся?

[37] `developer.apple.com/testflight/`

[38] `www.apple.com/app-store/`

Так как возможность персональной рекомендации изображений еще не встроена в приложение, изображения для последнего вопроса были выбраны на сервере с помощью модуля коллаборативной рекомендательной системы и предоставлены пользователям в виде списка ссылок.

3.2. Результаты апробации

Все 10 из 10 пользователей, прошедших опрос, указали, что новая рекомендательная система понравилась им больше. Самыми частыми указанными преимуществами новой рекомендательной системы являются:

- отсутствие в выдаче уже просмотренных изображений;
- более интересная лента;
- более интересные рекомендации по изображению.

Из предлагаемых 10 изображений пользователям понравилось не менее 5. Большей части пользователей понравились от шести до восьми предлагаемых изображений.

Результаты представлены на графиках 8, 9.

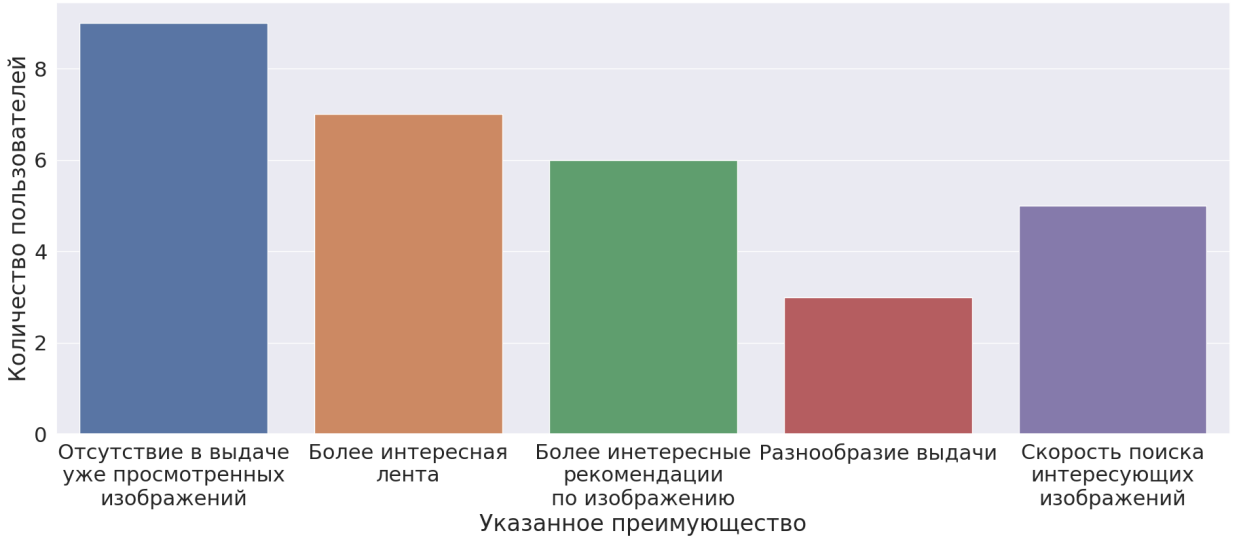

Рис. 8: Указанные преимущества выбранной рекомендательной системы.

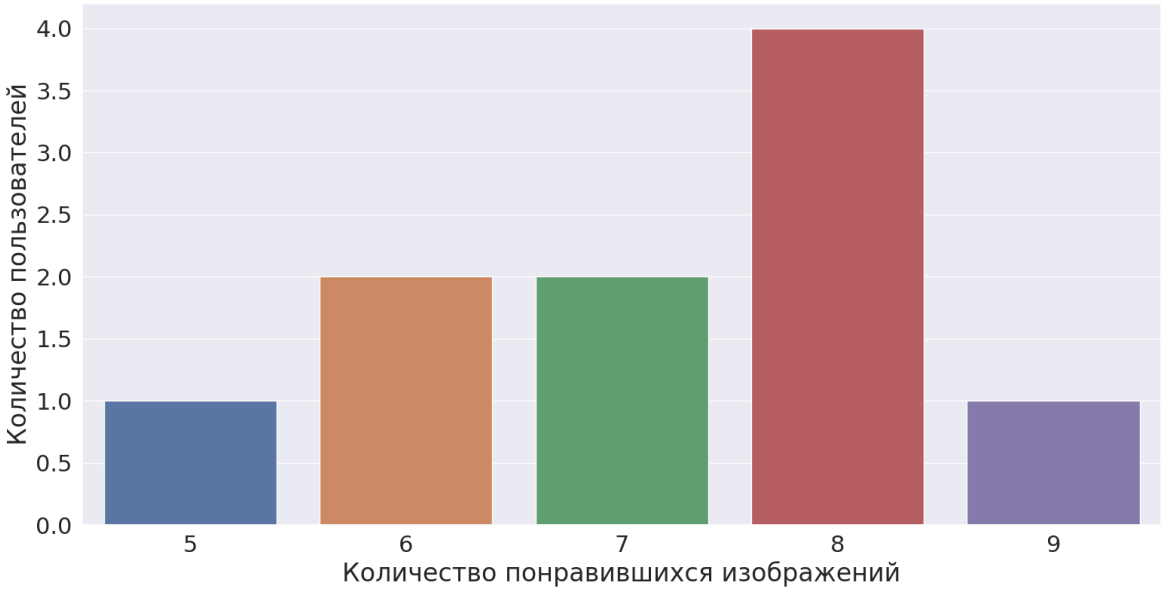

Рис. 9: Статистика выбора понравившихся изображений из персональных рекомендаций.

Так же было проведено сравнение скорости работы разработанной рекомендательной системы со старой рекомендательной системой. Для сравнения было отправлено 100 случайных запросов к каждой рекомендательной системе. Старая рекомендательная система в среднем обрабатывает запросы за 0.4 секунды. Новая рекомендательная система за 0.6.

Обработка запросов в среднем стала медленнее на 0.2 секунды. В контексте данной задачи разница не критична.

4. Заключение

В ходе данной работы была разработана система комбинированных рекомендаций по дизайнерскому контенту, позволяющая пользователям получать релевантные и персонализированные рекомендации.

В ходе работы были достигнуты следующие результаты:

- Проведен обзор подходов к реализации коллаборативной фильтрации, подходов к сбору действий пользователей и подходов к реализации механизма аутентификации.
- Реализована и интегрирована в сервис graphica.ai инфраструктура поддержки пользователей, включающая в себя сущность пользователя, библиотеки и возможность аутентификации.
- Реализован и интегрирован в сервис graphica.ai модуль получения и обработки данных из системы аналитики, использующий систему аналитики Amplitude.
- Разработан и интегрирован модуль коллаборативных рекомендаций, использующий алгоритм матричной факторизации, реализованный с помощью библиотеки tensorflow.
- Разработан модуль коллаборативной фильтрации, использующий модуль коллаборативных рекомендаций и предыдущую рекомендательную систему проекта graphica.ai.
- Модуль коллаборативной фильтрации интегрирован с рекомендательной системой проекта graphica.ai. Получившаяся система дает возможность персонализации рекомендаций и обрабатывает запросы на 0.2 секунды медленнее.
- Проведена апробация в проекте graphica.ai. Все 10 опрошенных пользователей предпочли разработанную в рамках данной работы рекомендательную систему старой.

Список литературы

[1] Yin Zheng, Cailiang Liu, Bangsheng Tang, and Hanning Zhou. Neural autoregressive collaborative filtering for implicit feedback. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, page 2–6, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, page 263–272, USA, 2008. IEEE Computer Society.

[3] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms, 2012.

[4] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, page 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[5] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery.

[6] Jon Herlocker Shilad Sen J. Ben Schafer, Dan Frankowski. Collaborative filtering recommender systems.

[7] S. Wei, N. Ye, S. Zhang, X. Huang, and J. Zhu. Collaborative filtering recommendation algorithm based on item clustering and global similarity. In *2012 Fifth International Conference on Business Intelligence and Financial Engineering*, pages 69–72, 2012.

[8] Daniel Billsus Michael. Learning collaborative information filters, 1998.

[9] Ralf Nikolaus. Learning the parts of objects using non-negative matrix factorization (nmf), 2007.

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[11] Xiaoyua Su. Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. In *Advances in Artificial Intelligence 2009*, pages 1–19, 2009.

[12] Келим Илья Игоревич. Архитектура и системы для комбинированных рекомендаций и поиска эстетичных изображений., 2020.

[13] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research*, 10(12), 2009.