МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«Южный федеральный университет»

Институт математики, механики
и компьютерных наук им. И. И. Воровича

Соколов Михаил Игоревич

СОЗДАНИЕ МЕТА-РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ДЛЯ ИНТЕРНЕТ-МАГАЗИНА НА ОСНОВЕ СУЩЕСТВУЮЩИХ МОДЕЛЕЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по направлению подготовки
02.04.02 – Фундаментальная Информатика и информационные технологии,
направленность программы
«Разработка мобильных приложений и компьютерных игр»

Научный руководитель –
доц., к. т. н. Чердынцева Марина Игорьевна

Рецензент –
к. ф.-м. н. Штейнберг Роман Борисович

Допущено к защите:
руководитель
образовательной программы _____ Демяненко Я. М.

Ростов-на-Дону – 2022

Оглавление

Постановка задачи

Цель работы — создание мета-рекомендательной системы, способной адаптироваться и давать рекомендации для любого интернет-магазина в области повышения качества обслуживания покупателей и эффективности продаж. Для достижения цели был сформирован набор рабочих задач:

1) Провести анализ полученных из интернет-магазинов треков активности пользователей и описания продаваемых товаров.
2) Выполнить выбор методов рекомендации, а также подбор типов моделей-рекомендаторов.
3) Сформировать набор представлений данных для создания и обучения моделей.
4) Выполнить обучение моделей и поиск оптимальных гиперпараметров.
5) Добавить возможности автоматического пересоздания представлений данных и моделей и переобучения моделей в связи с изменившимися данными и возможности использования оперативной истории без переобучения моделей.
6) Осуществить интеграцию в одну из существующих платформ интернет-магазинов.

Для решения задач (по согласованию с заказчиком) был выбран язык программирования Python v 3.8.10 с пакетами numpy v 1.19.5, scikit-learn v 1.0.0, scipy v 1.7.1, implicit v 0.4.4, pandas v 1.3.5 и TensorFlow v 2.4.2. Для ускорения обучения определенных нейросетевых моделей с использованием GPU также использовался NVidia CUDA Toolkit v 10.1.

Введение

Дальнейшее повышение эффективности работы субъектов электронной коммерции связано с расширением списков клиентов, получающих доступ к упомянутым субъектам, а также с персонализацией систем онлайн-маркетинга. По оценкам McKinsey [1] 35% выручки Amazon или 75% Netflix приходится именно на рекомендованные товары и процент этот, вероятно, будет расти. В работе создан прототип универсальной рекомендательной системы (мета-рекомендательной системы) на базе собранной информации интернет-магазинов.

Задача рекомендательной системы — проинформировать пользователя о товаре, который ему может быть наиболее интересен в данный момент времени. Клиент получает информацию, а сервис (на котором работает интернет-магазин и установлена рекомендательная система) зарабатывает на предоставлении качественных услуг. Услуги — это не обязательно прямые продажи предлагаемого товара. Магазин также может зарабатывать на комиссионных или просто увеличивать лояльность пользователей, которая потом выливается в рекламные и иные доходы.

В зависимости от модели бизнеса рекомендации могут быть его основой, как, например, у компании TripAdvisor, а могут быть просто удобным дополнительным сервисом (как, например, в каком-нибудь интернет-магазине одежды), призванным улучшить взаимодействие клиента с сервисом (англ. Customer Experience) и сделать навигацию по каталогу более удобной.

В данной работе предполагается использование рекомендательной системы как сервиса в дополнение к основной платформе(интернет-магазин), в качестве предмета рекомендаций используются товары магазина.

В работе проведено исследование данных, полученных от интернет-магазина. Выявлено, что имеются данные только о неявной обратной связи, и, соответственно изучение теоретической базы проводилось с целью определения моделей машинного обучения, применимых для обработки неявной обратной связи. Создан автоматический пайплайн обработки данных для

получения датасетов для обучения моделей. Произведен выбор метрик для оценок моделей.

Реализован унифицированный интерфейс модели машинного обучения, далее реализованы 7 моделей машинного обучения. Произведена настройка оптимальных гиперпараметров для унификации моделей и реализован пайплайн автоматического обучения/переобучения моделей.

В целях повышения качества рекомендаций создан автоматический ансамбль, который, тестируя модели при обучении, выбирает наилучшие по выбранной метрике модели и, используя авторский алгоритм, формирует финальные рекомендации. Произведено сравнение показателей моделей и ансамбля.

Эта работа может быть использована в предприятиях электронной коммерции (интернет-магазинах) для быстрого и легкого создания и обновления автоматической рекомендательной системы и соответственно, имеет высокую практическую значимость.

Результаты данной работы апробированы на конференциях "Математика. Компьютер. Образование. 2022"[2] и СИТО 2022 [3].

1. Входные данные. Анализ исходных данных

Любая работа, связанная с машинным обучением обязательно начинается с анализа данных(англ. EDA — Explorative Data Analysis).

Заказчиком были предоставлены экземпляры файлов, выгружаемых из интернет-магазинов. Важно отметить, что все такие выгрузки являются стандартизированными. Один из файлов содержал описание событий происходящих в интернет-магазине, в дальнейшем — данные о событиях. Данный файл был представлен в формате csv. Второй файл был представлен в формате json и содержал описания товаров, а также фильтров товаров, представленных в магазине.

Файл с данными о событиях содержал 44 типа полей, однако некоторые из них не заполнены во всем датасете. Далее следует список полей и указаны гипотезы для каждого поля. Незаполненные поля, а также поля не несущие полезной информации(сервисные хеши и т.д), опущены.

1) datetime. Поле времени начала события.
2) userip. Поле, в котором указывается IPv4-адрес, с которого осуществлялся доступ.
3) userid. Цифробуквенный уникальный ID пользователя в системе, основной способ идентификации пользователя.
4) useragent. Фрагмент HTML-header, указывающий на используемый браузер.
5) eventtype. Тип произошедшего события:
    а) ProductView - события просмотра товара пользователем.
    б) AddToCart - события добавления пользователем товара в корзину.
    в) FacetSelection - событие выбора пользователем фильтра показа товаров.
    г) Search - событие поиска пользователем товара по названию.
    д) AutoComplete - событие автодополнения запроса пользователя.
    е) ClickOnSearchResult - событие нажатия на результат поисковой выдачи.

6) usersearchphrase. Запрос пользователя. Используется в типах событий AutoComplete, Search.
7) correctedsearchphrase. Исправленный запрос пользователя. Используется в типе событий Search.
8) facetname. Имя фильтра по которому производился поиск. Используется в типе событий Facet Selection.
9) facetvalue. Значение фильтра. Используется в типе событий Facet Selection.
10) productid. Внутренний уникальный ID продукта.

Рассмотрим подробнее файл,содержащий описание товаров. Вследствие предполагаемой универсальности системы, а также большого количества полей, являющимися пустыми, приведем описание только используемых универсальных полей, и обобщенное описание остальных полей. В данном файле интерес представляют два поля: Items и Facets. Рассмотрим их подробнее.

Поле Items:

1) CatalogID. Уникальный числовой ID продукта. Совпадает со значением поля productid в событиях типа ProductView.
2) Name. Имя товара.
3) OrdersCount. Количество заказанных товаров.
4) Типы и значения фильтров, отвечающих товару.

Поле Facets:

1) FacetName. Имя фильтра.
2) TotalHits. Количество раз, которое этот фильтр был выбран.
3) Values. Список возможных значений, которые может принимать фильтр. Может иметь как категориальный тип, так и числовой тип.

После рассмотрения списка полей разумно рассмотреть несколько гипотез, которые вытекают из выбранных типов полей:

1) Поле userip может использоваться для учета страны пользователя в рекомендациях.
2) Поле usersearchphrase может быть использовано для лингвистического анализа запроса, что, возможно, позволит улучшить рекомендации.
3) Поле facetvalue может быть использовано при рекомендации, отфиль-

тровывая рекомендации по соответствующему значению фильтров.

4) Поле Name может быть использовано для поиска похожих по названию товаров.
5) Поле OrdersCount может быть использовано для фильтрации товаров по популярности.
6) Поле TotalHits может быть использовано для определения самых часто выбираемых фильтров.

Особо следует отметить, что эти стандартизированные выгрузки данных не содержат в себе информации об оценках пользователями купленных товаров, то есть в наличии только неявная обратная связь. Подробнее рассмотрим отличия явной и неявной обратной связи.

1.1. Явная и неявная обратная связь

Рекомендательные системы используют два основных типа обратной связи от пользователей. Это явная обратная связь(англ. explicit feedback) и неявная обратная связь(англ. implicit feedback). Приведем примеры каждой из обратных связей [4].

Явная обратная связь - это оценки контента пользователями. Такими оценками могут быть, например, количество звезд, числовые оценки по некоторым шкалам, и даже простое нравится/не нравится, использующееся, допустим, видеохостингом YouTube [4].

Неявная обратная связь - это любые записанные действия пользователя при взаимодействии с контентом. Просмотры товаров, добавление товара в корзину, покупка товара, история покупок, - все это является одним из вариантов неявной обратной связи.

Эти два типа связей значительно отличаются друг от друга по ряду характеристик:

1) Количество данных обратной связи. Для явной обратной связи количество собранных данных всегда значительно меньше, чем количество данных для неявной обратной связи. Кроме того, не всегда в системах вообще существуют встроенные механизмы сбора данных явной

обратной связи и/или возможность их создания.

2) Негативные оценки. При явной обратной связи всегда существует явная негативная обратная связь(или возможность ее установления). Так, допустим, недовольный товаром человек может поставить в оценке одну звезду из пяти возможных. При неявной обратной связи же практически невозможно установить негативный feedback, любая такая связь сама по себе будет позитивной. Так, допустим, отсутствие записанного взаимодействия пользователя и конкретной части контента может быть как в силу негативного отношения пользователя к данной части контента, так и в силу отсутствия взаимодействия этих пользователей вообще.
3) Наличие шума в данных. Когда мы проводим сбор неявной обратной связи, невозможно достоверно установить их предпочтения и истинные мотивы совершаемых ими действий. Так, например, данные о покупке (или просмотре/добавлении в корзину) пользователем продукта далеко не всегда являются следствием хорошего отношения пользователя к продукту. Существует вероятность, что продукт был приобретен в качестве подарка, или что позднее пользователь был разочарован при использовании продукта.
4) Отличие в смысле полученных данных. Так, при явной обратной связи числовое значение(количество покупок, оценка качества) означает предпочтение одному продукту другого, в то время как при неявной обратной связи это означает лишь частоту взаимодействия. Тем не менее, хотя более высокая частота взаимодействия не всегда отражает предпочтение одного продукта другому, повторяющееся действие с более высокой вероятностью показывает предпочтения пользователя.
5) Учет дополнительных признаков. При неявной обратной связи необходимо учитывать также доступность каждого конкретного продукта в момент времени, повторяемые покупки и т.д. При явной обратной связи такое не требуется.

По причинам, которые будут описаны в следующем разделе, в работе используется только неявная обратная связь.

1.2. Принятые решения об использовании

После проведенного анализа данных было принято решение использовать лишь следующие поля:

Из файла событий:

1) datetime.
2) userid.
3) productid.
4) eventtype. Было принято решение для простоты использовать исключительно типы событий ProductView и AddToCart, так как они несут наибольшее количество полезной информации для рекомендательной системы.

Из файла товаров:

1) CatalogID.
2) Типы и значения фильтров, отвечающих товару.

Такой подход позволяет значительно сократить количество используемой ОЗУ и памяти для хранения данных.

Для обработки данных был создан специализированный пайплайн. Он состоит из 6 стадий обработки данных, которые можно представить в виде графа, показанного на рисунке 1. Пайплайн был выполнен с использованием пакета DVC(Data Version Control) v2.5.4.

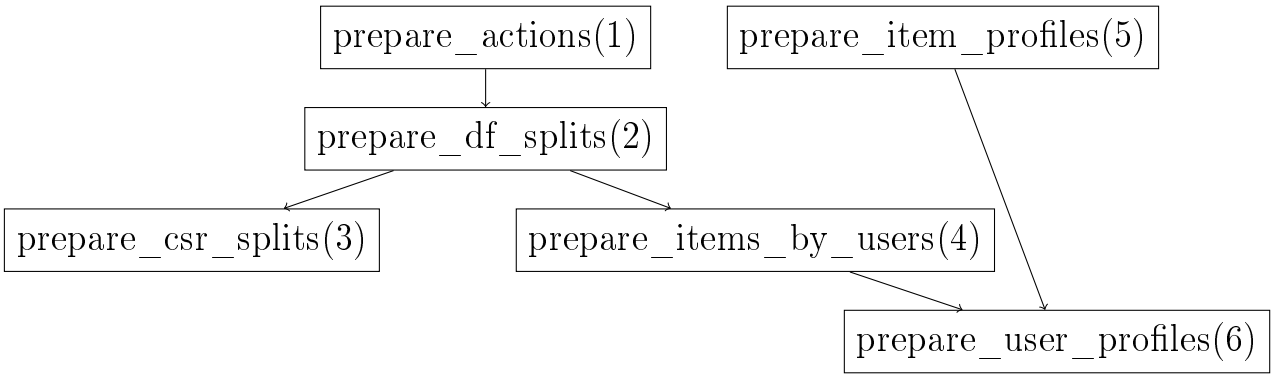

Рис. 1. Граф пайплайна обработки данных

На каждом этапе формируется одно из представлений данных. Кратко опишем их:

1) Стадия prepare_actions. На ней формируется временное представление — датафрейм (англ. $pandas.DataFrame$), содержащий в себе записи о всех событиях типа ProductView и AddToCart.
2) Стадия prepare_df_splits. На ней формируется основное представление данных — два датафрейма — соответственно разбиения train и test. Разбиение train содержит в себе все события, произошедшие до определенной даты включительно, разбиение test - все события, которые произошли после определенной даты. Важно, что test содержит в себе события только тех пользователей, которые входят во множество train.
3) Стадия prepare_csr_splits. На основе результатов предыдущей стадии (prepare_df_splits) формируются два основных представления данных CSR (csr_train и csr_test) — разреженные матрицы , созданные с использованием пакета scipy $scipy.sparse.csr\_matrix$, размерностью $M \times N$, где M - количество пользователей в системе, а N - количество продуктов в системе. Данные матрицы содержат все взаимодействия пользователей со всеми товарами.

   Следует вспомнить, что рассматриваются события двух видов - AddToCart и ProductView. Так как эти события не являются равнозначными, то было принято решение ввести повышающий коэффициент для событий AddToCart, что позволяет одно событие AddToCart рассматривать как одно или более событий ProductView, причем конкретный коэффициент задается в конфигурационном файле системы. Путем экспериментов был подобран коэффициент равный 10.
4) Стадия prepare_items_by_users. На этой стадии создается словарь взаимодействия пользователей и товаров. Он используется для контроля обучения моделей и в стадии prepare_user_profiles.
5) Стадия prepare_item_profiles. На этой стадии формируется представление данных ItemProfileStorage - создается таблица ($pandas.DataFrame$), содержащая профили всех товаров, построенная на извлеченных из json файла фильтров показа и их значений. Строки таблицы - товары, столбцы - характеристики товаров. В основном количестве признаки

являются категориальными, но в процессе подготовки представления данных преобразуются в бинарные.

6) Стадия prepare_user_profiles. На этой стадии формируется представление данных UserProfileStorage - таблица($pandas.DataFrame$), содержащая профили всех пользователей, построенная на извлеченных из json файла фильтров показа и их значений и представлении данных Item Profile Storage. Каждый профиль пользователя представляет собой алгебраическую сумму профилей всех товаров, с которыми он взаимодействовал (точно так же, как в CSR - с повышающими коэффициентами для событий AddToCart). Это основа для собственного подхода, объединяющего принципы Content based подхода и коллаборативной фильтрации.

Основные представления данных, которые будут использованы в дальнейшем - CSR, ItemProfileStorage, UserProfileStorage.

2. Теоретические основы рекомендательных систем

2.1. Анализ поставленной задачи

Как уже говорилось, поставленная задача - создание максимально универсальной рекомендательной системы, подходящей для любого интернет-магазина при условии стандартизированной выгрузки данных. Предварительно стоит отметить, что есть две категории товаров:

- Повторяемые товары. Это те товары-расходники, которые люди покупают часто. К таким относятся, например, продукты питания, бритвенные станки, предметы бытовой химии.
- Неповторяемые товары. Это такие товары, которые редко приобретают повторно. Примеры таких товаров - электроника, бытовая техника, ювелирные украшения.

Опишем нашу рекомендательную систему [4]:

1) Предмет рекомендации. Для данного проекта предметом рекомендации может являться только товар. Ситуация с применением системы для

магазинов, использующих услуги, не рассматривалась. В силу предполагаемой максимальной универсальности системы она предполагает одинаковое отношение как к товарам из повторяемой группы, так и к товарам из неповторяемой группы.

2) Цель рекомендации. Здесь цель рекомендации - информирование пользователя о товарах, которые могут ему подойти, и в конечном счете - покупка пользователем дополнительных товаров.
3) Контекст рекомендации. На момент получения рекомендации предполагается, что пользователь смотрит товары и/или находится в корзине.
4) Источники рекомендации. Здесь это как общая аудитория магазина, так и схожие по интересам пользователи, в зависимости от подхода.
5) Степень персонализации рекомендации. Здесь в зависимости от подхода используются разные степени персонализации.
6) Алгоритм рекомендации. Об этом будет написано ниже.

2.2. Обзор подходов к созданию рекомендательных моделей

Было установлено, что существует три основных подхода к созданию моделей для решения следующих задач — модели на основе популярности(англ. Summary-based), модели на основе содержимого(англ. Content-based) и модели на основе коллаборативной фильтрации(англ. Collaborative filtering). В подходе коллаборативной фильтрации также выделяют отдельный субподход - на основе факторизации матриц(англ. Matrix Factorization). Опишем отличительные особенности этих подходов.

Подход на основе популярности. Данный подход является наиболее простым и, тем не менее, достаточно эффективным. Он основан на неперсонализированной оценке популярности каждого товара, и соответственно, рекомендации пользователю самых популярных товаров.

Подход на основе содержимого. Данный подход основан на описании товара. Для товаров создаются признаковые описания, представляющие

собой векторы категориальных признаков. В рамках подхода рекомендуются товары, которые наиболее похожи на популярные у пользователя продукты.

Коллаборативная фильтрация. Данный класс систем начал активно развиваться в 90-е годы. В рамках подхода рекомендации генерируются на основании интересов других похожих пользователей, таким образом являясь результатом «коллаборации» множества пользователей. Отсюда и происходит название метода. Этот метод уже является персонализированным, то есть рекомендации подбираются персонально под каждого пользователя.

Одними из главных субподходов для коллаборативной фильтрации являются подходы на основе факторизации матриц. В основе таких субподходов лежит матрица товар-клиент - разреженная матрица взаимодействия оценок товаров и пользователей. Каждое значение суть мера заинтересованности пользователя в товаре.

Одна из типичных ситуаций для рекомендательной системы — проблема холодного старта. Данная проблема заключается в том, что при условии добавления нового пользователя/товара система не имеет информации о взаимодействии с ним, и он не будет отображаться в неперсонализированных рекомендациях, потому что им не хватает популярности, чтобы войти в статистику продаж, и он не будет появляться в персонализированных рекомендациях, потому что система не знает, как эти элементы связаны с другими. В нашей системе будет использоваться механизм паддинга рекомендаций, если что система, столкнувшаяся с проблемой холодного старта, не сможет предоставить необходимое количество рекомендаций. Кроме того, предположительно, система будет пересоздаваться и переобучаться каждые 168 часов, что должно далее нивелировать проблему холодного старта. Наконец, модели, использующие подход на основе содержимого, не испытывают проблем с отсутствием информации о взаимодействии пользователь-товар.

2.3. Метрики оценки моделей

Следующим шагом в создании мета-рекомендательной системы будет выбор метрик, способных оценить качество работы системы.

Введем общие обозначения для метрик:

- N — количество пользователей.
- K — количество товаров.
- $rel(i)$ — бинарный признак, показывающий, является ли товар на позиции i релевантным.

Оценка моделей производилась по набору следующих метрик:

1) Метрика NAP@K(Normalized Average Precision at K) является одной из типичных метрик для измерения качества рекомендательной системы [4]. Метрика сравнивает N рекомендаций системы и N наиболее релевантных товаров для пользователя, учитывая их позицию.

$$NAP@K = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{K}\sum_{i=1}^{K} rel(i) \tag{1}$$

2) Метрика NDCG@K(Normalized Distributed Continued Gain at K) также применяется достаточно часто. Она является модификацией Cumulative Gain at K с учетом порядка элементов.

$$nDCG@K = \frac{1}{N}\ sumj = 1^{N}\sum i = 1K\frac{2^{rel(i)}}{log_2(k+1)} \tag{2}$$

3) Метрика IOR@K(Intersection Over Recommended at K). Данная метрика предлагается как аналог NAP, с учетом того что порядок представления рекомендаций не имеет значения. Действительно, для пользователя нет особой важности в том, в каком порядке ему представляются рекомендации.

$$IOU@K\frac{1}{N}\sum_{j=1}^{N}/frac(\sum_{i=1}^{K} rel(i))K \tag{3}$$

4) Метрика HUR@K(Happy Users Ratio at K). Данная метрика была предложена заказчиком как методика измерения эффективности системы. Идея данной метрики заключается в том, наша рекомендация считается удачной, если множества рекомендаций нашей модели и множество товаров, с которыми пользователь взаимодействовал, имеют хотя бы одно пересечение. Она имеет интересное практическое значение - здесь K - количество рекомендаций, которые интернет-магазин может поместить в ленту.

$$HUR@K = \frac{1}{N}\sum_{j=1}^{N} max(1, \sum_{i=1}^{K} rel(i)) \tag{4}$$

3. Модели, используемые в системе

3.1. Общее описание моделей

В системе созданы 7 моделей машинного обучения, одна из которых является baseline. Перечислим их по порядку: TopN, ALS, BPR, AE, IP, UIP, DRN. Выход каждой модели - список товаров, которые необходимо рекомендовать пользователю.

Отметим, что процесс генерации и обучения моделей также является полностью автоматизированным, что позволяет переобучать модели без участия ml-инженеров. Кроме того, все модели имеют унифицированный интерфейс, что в перспективе позволит добавлять другие модели для улучшения качества рекомендаций. К сожалению, часть моделей(все модели кроме ALS, BPR и TopN) жестко привязываются к данным, что делает процесс дообучения на новых данных невозможным, вследствие чего используется процесс полного переобучения.

3.2. Модель TopN

Модель TopN является одной из наиболее частых и простых в как в составлении, так и в использовании моделей и является представителем подхода на основе популярности [5]. Путем парсинга представления данных CSR производится сортировка товаров по количеству взаимодействий, и любому пользователю предлагается N самых популярных товаров, где N задается необходимостями системы. Данная модель используется в качестве baseline для всех остальных моделей, а также для дополнения списка рекомендаций, в случае если другие модели не смогли дать необходимое их число.

3.3. Модель ALS

Модель ALS(англ. Alternating Least Squares) [5] — одна из классических моделей рекомендательных систем. Данная модель принадлежит к классу моделей коллаборативной фильтрации.
Введем отношения, необходимые для работы данной модели. Примем общее количество пользователей на данный момент времени в системе как n, общее количество товаров на данный момент времени в системе как m. Запишем все взаимодействия пользователей и продуктов в матрицу $R_{n \times m}$, где $r_{u,i}$ - количество раз которое пользователь u взаимодействовал(в нашем случае просматривал или клал в корзину) товар i. Результирующая матрица является разреженной, так как на практике эффективно невозможна ситуация, когда каждый пользователь взаимодействует с каждым товаром, что отражается нулевыми значениями в матрице. Кроме того, как предложено в статье [6], введем T - множество $(u, i) : r_{u,i} > 0$ и $\hat{r}_{ui}$ - предсказанное взаимодействие пользователя с системой.

Процесс факторизации матрицы позволяет представить матрицу R как $R \approx PQ^T$, где $P \in \mathbb{R}^{n \times k}$ - матрица признаков пользователя, а $Q \in \mathbb{R}^{m \times k}$ - матрица признаков товаров, причем k - заранее определенная константа. Соответственно $p_u \in \mathbb{R}^k$ это строка u в матрице P, а $q_i \in \mathbb{R}^k$ это строка i в

матрице Q.

Формула для предсказания $\hat{r}_{ui}$:

$$\hat{r}_{ui} = p_u^T q_i \tag{5}$$

В статье [5] предложено ввести переменные s_{ui}, которые можно принять как меру предпочтения пользователя u товару i. Для удобства использования бинаризуем их. Тогда

$$pr_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} \tag{6}$$

Тем не менее сразу использовать эти значения pr_{ui} нерационально. Авторы статьи [5] предлагают введение дополнительных переменных c_{ui} таких, что $c_{ui} = 1 + \alpha * r_{ui}$. Эти переменные отвечают за меру нашей уверенности в значении pr_{ui}. Всегда будет существовать ненулевое предпочтение пользователя u товару i, и при этом, чем больше количество взаимодействий пользователя и товара зафиксировано, тем более высокое значение примет c_{ui}. Параметр α здесь повышающий коэффициент, который предлагается принять равным 40.

Введем ценовую функцию:

$$g(P, Q) = \sum_{u,i \in \mathrm{T}} (c_{ui}(\hat{r}_{ui} - r_{ui})^2 + \lambda_P \parallel p_u \parallel^2 + \lambda_Q \parallel q_i \parallel^2) \tag{7}$$

В силу того что будет применена оптимизация путем использования метода сопряженных градиентов, значения c_{ui} и r_{ui} не будут константными в позициях $(u, i) \notin \mathbb{T}$. Обозначим такие исходные значения как c_0, r_0.

Особенность примененной здесь модели заключается в том, что для ускорения процесса вычислений применяется взвешенная гребневая регрессия, которую можно обозначить как WRR — Взвешенная гребневая регрессия (англ. Weighted Ridge Regression) [7].

Приведем один из алгоритмов, часто применяющихся для прибли-

женного расчета WRR-метода сопряженных градиентов с необходимой точностью ϵ [8].

Input: $A, M \in \mathbb{R}^{d \times d}, b, w_0 \in \mathbb{R}^{d \times 1}, E \in \mathbb{N}$
Output: $w_0 \in \mathbb{R}^d$
$w \leftarrow w_0, r \leftarrow b - Aw, z \leftarrow M^{-1}r, p \leftarrow z$
for $k \leftarrow 1...E$ **do**
  if $\| r \| < \epsilon$ **then**
    return w
  else
    $\gamma \leftarrow r^T z$ $\alpha \leftarrow \gamma/(p^T Ap)$ $x \leftarrow x + \alpha r$ $r \leftarrow r - \alpha Ap$
    $z \leftarrow M^{-1}r$ $\beta \leftarrow \gamma/(r^T Az)$ $p \leftarrow z + \beta p$
  end
end

Algorithm 1: Предусловленный метод решения системы $Aw = b$ методом сопряженных градиентов.

Расчет (u, i) происходит по следующим формулам:

$$\hat{r}_{ui} = p_u^T q_i, \tag{8}$$

$$p_u = s_u \sum_{j \in \mathbb{J}} w_j, \tag{9}$$

$$s_u = (n_u + 1)^{\frac{1}{2}} \tag{10}$$

При этом ценовая функция остается такой же, с тем отличием, что P может быть выражена через W и обучающее множество. Ценовая функция является невыпуклой, поэтому используется не точная, но примерная минимизация по методу IALS.

Пусть $W \in \mathbb{R}^{m \times k}$ -матрица векторов w_j. Введем $B \in R^{n \times m}$ такую, что $B_{ui} = r_{ui} s_u$. Исходя из этого, верно следующее выражение: $P = BW$, где B - это разреженная матрица.

Во время оптимизации используется три комплекта параметров - P, Q, W. Процесс оптимизации представлен листингом 2, где WRR - алгоритм 1.

```
for e = 1...I do
    Q − step: for i ∈ 𝕀 do
        q_i ← WRR(P, r̄_i, c̄_i, n_i λ_Q I)
    end
    P − step: for u ∈ 𝕌 do
        p_u ← WRR(Q, r_u, c_u, n_u λ_P I)
    end
    W − step: for k ∈ 𝕂 do
        q_i ← WRR(B, p̄_k, 1, λ_W I)
    end
    for u ∈ 𝕌 do
        p_u ← p_u = s_u Σ_{j∈𝕀≋} w_j
    end
end
```

Algorithm 2: Процесс оптимизации алгоритма ALS с помощью WRR.

Рассмотрим шаги алгоритма подробнее. Первый шаг оптимизирует Q в уравнении $R \approx PQ^T$. Здесь $\bar{r}_i \in \mathbb{R}^{N\times 1}$ и $r_u \in \mathbb{R}^{M\times 1}$ - столбец i и ряд u матрицы R. Векторы $\bar{}_i \in \mathbb{R}^{N\times 1}$ и $\bar{}_u \in \mathbb{R}^{M\times 1}$ соответственно составлены из значений $c_u i$. В качестве WRR может быть использован любой алгоритм расчета взвешенной гребневой регрессии, причем $\bar{r}_i$, $\bar{c}_i$ не имеют необходимости полного пересчета каждый раз, вследствие того, что изменяется лишь малая их часть.

Второй шаг очень похож на первый, но оптимизирует лишь P.

Третий шаг отличен в том, что он оптимизирует только W из $P \approx BW$, причем он направлен на лучшую аппроксимацию матрицы P, а не R. Здесь $\bar{p}_k \in \mathbb{R}^{N\times 1}$ — это k-й столбец матрицы P, веса взвешенной гребневой регрессии константно равны 1.

Описанный алгоритм реализован в библиотеке implicit v.0.4.4, которая используется в данном проекте. Библиотека implicit требует представление данных CSR.

3.4. Модель BPR

Модель BPR(англ. Bayesian Personalized Ranking) также принадлежит к классу моделей коллаборативной фильтрации.

Обозначим множество всех пользователей как U, множество всех товаров как I. Доступные данные по неявной обратной связи — $S = U \times I$. Введем функции предпочтения пользователя: $>_u \in I^2$, которая отвечает следующим условиям:

- $\forall i, j : i \neq j \rightarrow i >_u j \vee j >_u i$
- $\forall i, j : i >_u j \wedge j >_u i \rightarrow i = j$
- $\forall i, j, k : i >_u j \wedge j >_u k \rightarrow i >_u k$

Также, статья [9] предлагает ввести следующие обозначения.

- $I_u+ : \{i \in I \wedge (u, i) \in S\}$
- $U_i+ : \{u \in U \wedge (u, i) \in S\}$

Как уже говорилось, зачастую данные неявной обратной связи крайне разреженны. Кроме того, в неявной обратной связи присутствует только положительная обратная связь — остальные данные не заполнены. Стоит еще раз отметить, что при использовании неявной обратной связи — незаполненные данные на самом деле являются смесью отрицательных данных и незаполненных. Обычно такие данные просто отбрасываются, но в нашем случае они будут использованы.

Типичный подход для рекомендательных систем — расчет такого значения x_{ui}, которое отражает степень предпочтения пользователя к данному товару. Тренировочным данным $(u, i) \in S$ присваивается позитивная метка, а любым другим комбинациям $(u, i) \notin S$ — негативная. Это означает, что модель натренирована предсказывать 1 для предметов принадлежащих S, и 0 - в противном случае. Однако в данном подходе есть проблема — все элементы, которые модель должна ранжировать в дальнейшем, представлены для обучения модели как негативная обратная связь. Соответственно, достаточно чувствительная модель будет иметь плохую эффективность - ведь она будет предсказывать только 0.

Применяется другой подход — используем в качестве тренировочного

множества пары товаров, причем оптимизация происходит в зависимости от корректности ранжирования пары, в отличие от использования одиночных "верных" товаров [9]. То есть задача — насколько это возможно, получить из S отношения $>_u$ для каждого пользователя. Так, если $(u, i) \in S$ — то можно считать что пользователь u предпочитает i. Формализуем: создадим тренировочный датасет $D_S : U \times I \times I$ такой, что:

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\} \tag{11}$$

То есть мы создаем пары товаров для пользователей такие, что пользователь u предпочитает товар i товару j. Данный подход имеет два преимущества:

- Датасет D состоит из позитивных и негативных пар и незаполненных значений. Незаполненные значения между двумя товарами, с которыми не взаимодействовал пользователь, это те значения, которые требуется предсказать. Соответственно, если рассмотреть попарно, то тренировочный и тестовый датасеты разделены.
- При этом тренировочные данные являются подмножеством D.

Рассмотрим алгоритм работы BPR [9]. Формулировка Байесовской задачи для нахождения персонализированного ранжирования для товара $i \in I$ заключается в максимизации следующей условной вероятности:

$$p(\theta | >_u) \sim p(>_u | \theta) p(\theta). \tag{12}$$

Здесь θ — вектор параметров модели, а $>_u$ — это действительные предпочтения пользователя u, причем предполагается, что все пользователи действуют независимо друг от друга. Кроме того, предполагается что порядок каждой пары товаров i, j является независимым от порядков других

пар. Отсюда, функцию $p(>_u|\theta)$ можно переписать как:

$$\prod_{u\in U} p(>_u|\theta) = \prod_{(u,i,j)\in U\times I\times I} p(i >_u j|\theta)^{\sigma((u,i,j)\in D_S)}\cdot$$
$$\cdot (1-p(i>_u j|\theta))^{\sigma((u,i,j)\notin D_S)} \quad (13)$$

Причем

$$\sigma(b) = \begin{cases} 1 & if\ b == true \\ 0 & otherwise \end{cases} \quad (14)$$

Однако, используя тот факт, что схема попарного ранжирования всеобща и антисимметрична, можно упростить данную формулу до:

$$prod_{u\in U} p(>_u|\theta) = \prod_{(u,i,j)\in D_S} p(i >_u j|\theta) \quad (15)$$

Чтобы гарантировать исполнение свойств всеобщности и антисимметричности, введем индивидуальную вероятность того, что пользователь u предпочтет товар i товару j как $p(i >_u j) = \sigma(\hat{x}_{uij}(\theta))$, где $\sigma(x) = \frac{1}{1+e^{-x}}$. Здесь $\hat{x}_{uij}(\theta)$ — функция от вектора θ, которая и описывает отношения пользователя u и товаров i, j. Таким образом, мы делегируем моделирование отношений внутреннему классу модели (здесь использована факторизация матриц, описанная в параграфе про ALS).

Для того чтобы закончить моделирование по Байесу, введем понятие "Общей плотности"$p(\theta)$, которая является нормальным распределением с математическим ожиданием = 0 и матрицей ковариации $\sum_\theta$: $p(\theta) = N(0, \sum_\theta)$. Представим $\sum_\theta$ как $\lambda_\theta I$. Тогда сформулируем Байесовский кри-

терий оптимизации BPR-Opt:

$$\begin{aligned} BPR-Opt = ln(p(\theta)| >_u) = \\ ln(p(>_u |\theta)p(\theta) = \\ ln \prod_{(u,i,j)\in D_S} \sigma_{x_{\hat{uij}}} p(\theta) = \\ \sum_{(u,i,j\in D_S)} ln\ \sigma(\hat{x}_{uij}) + ln\ p(\theta) = \\ \sum_{(u,i,j\in D_S)} ln\ \sigma(\hat{x}_{uij}) - \lambda_\theta \parallel \theta \parallel^2 \end{aligned} \tag{16}$$

Рассмотрим сам алгоритм обучения LearnBPR. Это специальный алгоритм, основанный на стохастическом градиентном спуске и динамической выборке триплетов для тренировки.

Найдем градиент BPR-Opt.

$$\frac{\partial BPR-Opt}{\partial\theta} = \sum_{(u,i,j\in D_S)} \frac{\partial}{\partial\theta} ln\ \sigma(\hat{x}_{uij}) - \lambda_\theta \frac{\partial}{\partial\theta} \parallel \theta \parallel^2 \propto$$

$$\sum_{(u,i,j\in D_S)} \frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \frac{\partial}{\partial\theta} \hat{x}_{uij} - \lambda_\theta\theta \tag{17}$$

Соответственно, можно записать и общий алгоритм LearnBPR.

Input: D_S, θ
Output: θ
initialize θ
while *not convergence* **do**
  take (u, i, j) из D_S
  $\theta \leftarrow \theta + \alpha(\frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \frac{\partial}{\partial\theta} \hat{x}_{uij} - \lambda_\theta\theta)$
end
return θ

Algorithm 3: Общий алгоритм LearnBPR.

Особо следует отметить выбор данных для тренировки. Если выби-

рать данные триплетов, принадлежащих к одному пользователю или товару, то сходимость даже стохастического градиентного спуска будет медленной, так как для пары (u, i) существует много j, для которых выполняется что $(u, i, j) \in D_S$. В статье [9] предложено выбирать триплеты случайно и равномерно. С таким подходом шанс выбрать одинаковые пары "пользователь-товар" достаточно мал, и поэтому алгоритм сходится значительно быстрее.

3.5. Модель AE

Модель AE(AutoEncoder) принадлежит к классу моделей коллаборативной фильтрации, а также является представителем semi-supervised learning [10].

Автоэнкодер представляет собой нейронную сеть, состоящую из двух частей — энкодер(англ. encoder) и декодер(англ. decoder). Эти части соответственно выполняют преобразования $encode(x) : R^n \to R^d$ и $decode(x) : R^d \to R^n$, где n — исходная размерность данных, а d — целевая размерность. Цель этих преобразований: получить представление данных исходной размерности n в размерности d такое, чтобы минимизировать отклонение исходных данных от полученных: $x - decode(encode(x)) - > min$.

В данной модели и энкодер и декодер — это feed-forward сети с полносвязными слоями $I = f(W * x + b)$, где f — некоторая нелинейная активационная функция, причем архитектура декодера такая же, как и архитектура энкодера.

Рассмотрим принцип работы систем. Forward pass — на вход подается вектор рейтингов пользователя $x \in R^M$, где M — количество продуктов. На выходе мы получаем вектор $x \in R^M$, но с одним значительным отличием — на входе вектор был разреженным, на выходе все значения пользователя заполнены.

Особо стоит отметить процесс выбора функций активации и loss-функции. Для функции активации классическим решением было бы использование RELU(Rectified Linear Unit). Однако, вследствие характера об-

рабатываемых данных (неявная обратная связь от пользователей), нежелательна потеря отрицательных значений после активации, что делает нерациональным его использование. Поэтому в качестве функции используется ELU (Exponential Linear Unit). Также был проведен эксперимент с использованием SELU(Scaled Exponental Linear Unit). Эксперименты проводились на трех размерностях бутылочного горлышка(англ.bottleneck) — 128, 256, 512. Результаты внесены в таблицу.

Приведем формулы всех трех функций активации.

$$ELU(x) = \begin{cases} x & x > 0 \\ exp(x) - 1 & x \leq 0 \end{cases} \tag{18}$$

$$RELU(x) = max(0, x) \tag{19}$$

$$SELU(x) = scale * (max(0, x) + min(0, \alpha * (exp(x) - 1))) \tag{20}$$

Применение таких функций дало результаты, представленные в таблице 1.

Таблица 1. Результаты использования различных функций активации

Размерность	Функция	DCG@30	HUR@5	IOU@30	NAP@30
128	ELU	0.06878	0.24956	0.04602	0.02510
	SELU	0.05771	0.22301	0.03988	0.02014
	RELU	0.06313	0.23894	0.03917	0.02453
256	ELU	0.07552	0.25841	0.04696	0.02894
	SELU	0.07021	0.24779	0.04566	0.02497
	RELU	0.06343	0.24248	0.03988	0.02427
512	ELU	0.07842	0.25664	0.04773	0.03082
	SELU	0.07398	0.26018	0.04678	0.02780
	RELU	0.06468	0.24071	0.03947	0.02510

Для достижения максимальной универсальности системы было принято решение использовать размер бутылочного горлышка равный 256 и функцию активации elu.

В подобных задачах [10] часто используют MMSE(Masked Mean Squared Error) (loss оценивается только по не нулевым позициям входного вектора,

позволяя сети сколько угодно сильно "ошибаться" по тем позициям, где стояли нули) в качестве loss-функции.

$$MMSE = \frac{m_i * (r_i - y_i)^2}{\sum_{i=0}^{i=n} m_i} \tag{21}$$

Однако в процессе экспериментов выяснилось, что результаты нейронной сети с такой конфигурацией значительно хуже. Тогда было принято решение использовать обычный MSE(Mean Squared Error), имеющий формулу:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(r_i - y_i)^2 \tag{22}$$

MSE не может занулить все наши рекомендации просто потому, что автоэнкодер не может дать 100% точность после разжатия данных.

Модель реализована с использованием пакета TensorFlow 2.4.2. Использование программного пакета CUDA позволяет производить обучение с использованием графических ускорителей. Обучение модели происходит на протяжении 500 эпох, с использованием batch_size = 256.Время обучение даже с использованием графического ускорителя достаточно велико — около 30 минут. Архитектура модели представлена на рисунке 2.

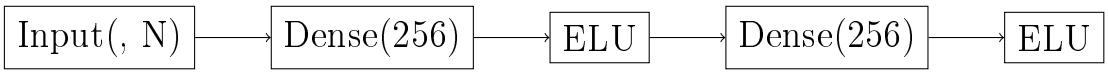

Рис. 2. Архитектура модели AE

3.6. Модель IP

Модель IP(Item Profiles) является представителем подхода на основе содержимого [4].

Создается представление данных IP, в котором каждый товар представляется как вектор размерности n, где n — общее количество бинаризованных фильтров. То есть существует метрическое пространство $\mathbb{B}^n$ с набором векторов $x \in \mathbb{B}^n$. Задача - для $\forall q \in \mathbb{R}^n$ найти N ближайших точек к q.

Построенные данные используются для обучения модели, основанной на алгоритме нахождения ближайших соседей, который реализован в пакете Sklearn как Sklearn.NearestNeighbours. Рассмотрим его подробнее.

Алгоритм Sklearn.NearestNeighbours автоматически выбирает один из трех алгоритмов — BallTree, KDtree и brute [11]. Кратко рассмотрим эти алгоритмы:

1) KDTree [12] — бинарное дерево, где каждый узел — точка с размерностью n. Можно представить, что каждая точка является гиперплоскостью, разделяющей пространство на два подпространства. Направление гиперплоскости выбирается следующим образом: каждая точка ассоциируется с одной из n осей, а гиперплоскость выбирается ортогонально данной оси;
2) BallTree — бинарное дерево, где каждый узел определяет гиперсферу с размерностью n, которая содержит в себе все точки;
3) brute — наиболее простой brute-force алгоритм, находящий все возможные расстояния точек друг от друга.

Необходимо выбрать метрику расстояния, которая будет использоваться в алгоритме Sklearn.NearestNeighbours. Примем, что все метрики находят расстояние между двумя одномерными тензорами $v1$ и $v2$ размерности n. Рассмотрим несколько возможных вариантов метрик :

1) Косинусная метрика(Cosine). Данная метрика является достаточно универсальной. Формула:
$$\frac{\sum_{i=1}^{n} v1_i v2_i}{\sum_{i=1}^{n} \sqrt{v1_i^2}\sqrt{v2_i^2}} \tag{23}$$
2) Жаккардова метрика(Jaccard). Данная метрика используется для бинаризованных классов. Введем дополнительные обозначения:
   - M_{00} — общее количество индексов i , когда $v1_i = v2_i = 0$.
   - M_{01} — общее количество индексов i , когда $v1_i = 0, v2_i = 1$.
   - M_{10} — общее количество индексов i , когда $v1_i = 1, v2_i = 0$.
   - M_{11} — общее количество индексов i , когда $v1_i = v2_i = 1$.

Формула:

$$\frac{M_{01} + M_{10}}{M_{01} + M_{10} + M_{11}} \tag{24}$$

3) Метрика Dice. Данная метрика используется для бинаризованных классов. Используя обозначения из предыдущего пункта, формулу можно записать следующим образом:

$$\frac{M_{01} + M_{10}}{2 * M_{11} + M_{01} + M_{10}} \tag{25}$$

4) Евклидова метрика. Это наиболее естественная функция расстояния, возникающая в геометрии, отражающая интуитивные свойства расстояния между точками. Формула метрики:

$$\sqrt{\sum_{i=1}^{n}(v1_n - v2_n)^2} \tag{26}$$

5) Манхэттенская метрика. Формула данной метрики очень проста:

$$\sqrt{\sum_{i=1}^{n}|v1_n - v2_n|} \tag{27}$$

6) Метрика Чебышева. Формула:

$$\sqrt{\max_{i=1}^{n}|v1_n - v2_n|} \tag{28}$$

Таблица 2. Результаты использования различных метрик расстояния

Метрика	DCG@30	HUR@5	IOU@30	NAP@30
Cosine	0.03754	0.12851	0.02505	0.01080
Jaccard	0.03101	0.11245	0.02155	0.00779
Dice	0.03101	0.11245	0.02155	0.00779
Eucledian	0.03459	0.11245	0.02302	0.00972
Manhattan	0.03613	0.12249	0.02377	0.01029
Chebyshev	0.03200	0.10843	0.02113	0.00897

Как можно заметить из данных таблицы 2, наилучшие показатели модель получила при работе с косинусной метрикой, которая, является и наиболее универсальной. Так как универсальность крайне важна для выполняемой задачи, то данная метрика и будет использоваться в дальнейшем.

В случае рекомендаций товаров на основе истории взаимодействий пользователя применяется алгоритм 4. Здесь N — количество необходимых рекомендаций, u — пользователь, для которого даются рекомендации, k — максимальное число похожих на данный товар товаров, D_u — набор товаров, с которыми взаимодействовал пользователь.

Input: N, D_u, z
Output: Набор товаров
result = {}
i = 0
sort D_u by popularity
while *len(result)* $< N$ **do**
    result $\leftarrow$ result $\cup$ kNN($D_u[i]$)
end
return result

Algorithm 4: Процесс генерации рекомендаций с помощью IP.

Если алгоритм не позволяет найти необходимое количество товаров ($len(result) < N$), что происходит, например, в ситуации когда пользователь малоактивен и/или начал пользоваться сайтом недавно, результат дополняется результатами модели TopN до необходимого количества рекомендаций. Правило брать не более чем k похожих на текущий товара выведено экспериментально, в противном случае рекомендация получается маловариативной, и как соответствие, зачастую худшей по качеству. Число k=3 выведено экспериментальным путем.

3.7. Модель UIP

Модель UIP(User Item Profiles) является гибридной моделью, сочетающей в себе черты моделей с подходами на основе содержимого и моделей коллаборативной фильтрации.

Аналогично предыдущей модели, построенные в представлении данных UIP профили пользователей используются как обучающий датасет для модели sklearn.NearestNeighbours [11]. Данная модель позволяет находить пользователей, похожих на данного, и, соответственно, рекомендовать пользователю товары, являющиеся популярными у похожих пользователей. Используется представление данных UIP. Применяется алгоритм, приведенный в листинге 5:

Input: N, D_u, z
Output: Набор товаров
result = {}
i = 0
sort D_u by popularity
while *len(result)* $< N$ **do**
    result $\leftarrow$ result $\cup$ kNN($D_u[i]$)
end
return result

Algorithm 5: Алгоритм генерации рекомендаций с помощью UIP

Аналогично, правило брать не более чем 5 товаров у похожего пользователя выведено экспериментально.

3.8. Модель DRN

Данная модель примечательна тем, что она так же, как и модель UIP является гибридной на базе моделей с подходами на основе содержимого и моделей коллаборативной фильтрации [13]. Также она примечательна и своей конструкцией — это сиамская нейронная сеть, имеющая структуру,

показанную на рисунке 3. Задача этой сети — обучиться отличать вещи, которые могут понравиться пользователю. Необычен и процесс обучения — на вход данной модели подаются тройки вида (профиль пользователя, профиль понравившейся ему вещи, профиль не понравившейся ему вещи).

Снова рассмотрим представление IP, которое суть метрическое пространство $\mathbb{B}^n$ с набором векторов $x \in \mathbb{B}^n$. Обращаясь же к представлению данных UIP, вспомним, что профиль каждого пользователя представляет собой алгебраическую сумму профилей товаров, с которыми он взаимодействовал, то есть это пространство — метрическое пространство $\mathbb{R}^n$ с набором векторов $x \in \mathbb{R}^n$.

На вход данной модели необходимо подать тройку(пользователь, товар, с которым он взаимодействовал, товар, с которым он не взаимодействовал). Здесь так же используется аксиома, что если человек взаимодействовал с товаром, то данный товар ему понравился. При этом, несмотря на то, что в данной системе размерности профилей пользователей и товаров идентичны, это не обязательно должно быть так. Так, допустим, в случае добавления в систему механизмов отслеживания поведения каждого пользователя, представляется возможным добавить в профиль пользователя новые данные и/или заменить профиль пользователя.

Приведем граф модели — рисунок 3. Граф использует переменную emb_dim. Значение данной переменной экспериментально подобрано как 32.

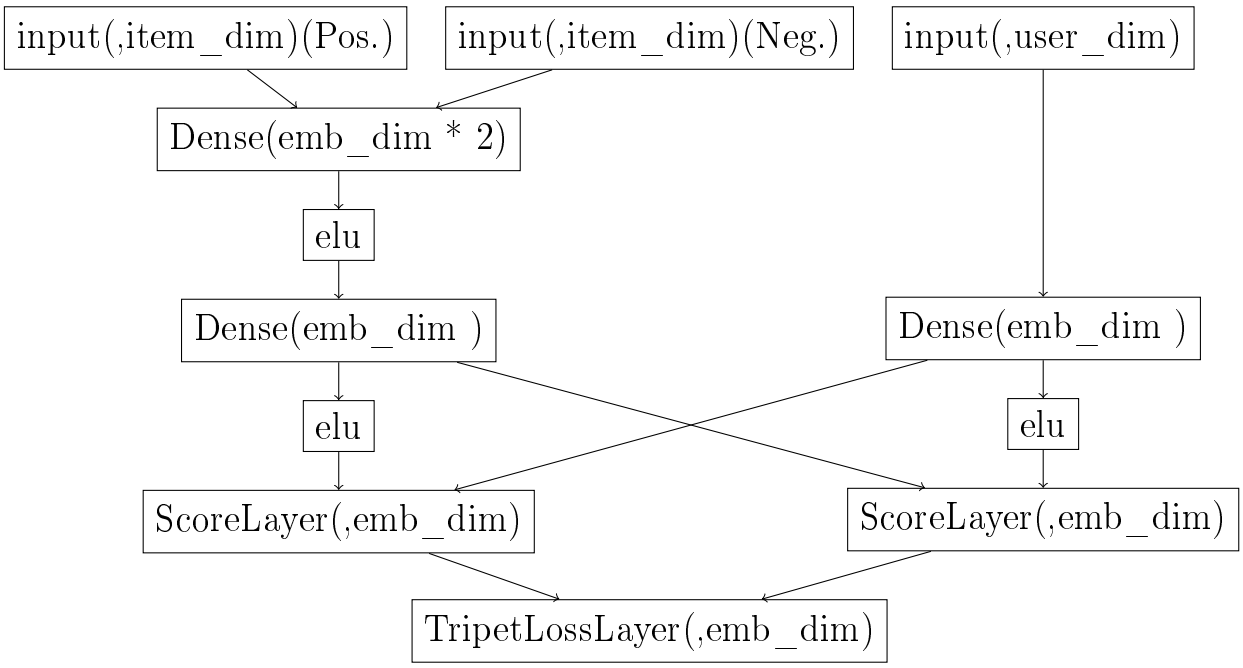

Рис. 3. Архитектура модели DRN

Конкретизируем значение данного графа. Данная модель использует две подмодели — для создания эмбеддингов(англ. Embedding) товаров (полносвязный слой(англ. Dense Layer) размерности 64, eLU, полносвязный слой размерности 32, ReLU) и пользователей (полносвязный слой размерности 32, eLU).

Стоит отметить, что слои TripletLossLayer и ScoreLayer реализованы специально для этой модели. Рассмотрим их подробнее.

Score layer — слой, который определяет близость эмбеддинга пользователя к эмбеддингу товара. Очевидно, что задача сети — максимизировать близость(минимизировать расстояние) между эмбеддингом пользователя и эмбеддингом товара, с которым он взаимодействовал, и, соответственно, максимизировать расстояние между эмбеддингом пользователя и эмбеддингом товара, с которым он не взаимодействовал.

Triplet loss layer — слой, который определяет насколько хорошо сеть обучилась. Он использует следующую функцию:

$$Loss = 1.0 - \frac{1}{1 + e^{-(p\_score - n\_score)}} \tag{29}$$

Здесь p_score — выход Score layer, отвечающего за товары, с которыми

пользователь взаимодействовал, n_score - выход Score layer, отвечающего за товары, с которыми пользователь не взаимодействовал.

Из-за сложности и специализированных слоев данная модель имеет низкую скорость обучения и не показывает высоких результатов.

3.9. Сравнительный анализ моделей

Для сравнительного анализа моделей была рассчитаны метрики для всех моделей на тестовом датасете, содержащем данные за 2 недели, и сведены в единую таблицу 3. Основополагающей метрикой здесь является метрика HUR@5. Как мы видим, наилучший результат показали модели AE и UIP.

Таблица 3. Сравнительные характеристики моделей

Модель	DCG@30	HUR@5	IOU@30	NAP@30
TopN	0.02464	0.11504	0.02029	0.00535
ALS	0.02949	0.10442	0.01994	0.00861
BPR	0.01492	0.05664	0.01298	0.00306
AE	0.07552	0.25841	0.04696	0.02894
IP	0.03754	0.12851	0.02505	0.01080
UIP	0.05629	0.26707	0.0207	0.02035
DRN	0.00861	0.03363	0.0027	0.00401

4. Создание ансамбля моделей

Для улучшения результатов был создан автоматический ансамбль, который выбирает лучшую модель на основе коллаборативной фильтрации и лучшую модель на основе содержимого(в данном случае единственную — IP). Вследствие того, что в ансамбле присутствуют модели, сходные по характеристикам, а в результате обучения метрики немного меняются в зависимости от данных, этот подход позволяет потенциально всегда выбирать наилучшие модели для заданных магазинов. Введем $Prods$ — множество всех товаров, доступных в интернет магазине — и рассмотрим алгоритм формирования финальных рекомендаций(листинг 6).

Сравним метрики, полученные таким способом, с уже имеющимися и занесем данные в таблицу 4.

Таблица 4. Сравнительные характеристики моделей и ансамбля

Модель	DCG@30	HUR@5	IOU@30	NAP@30
TopN	0.02464	0.11504	0.02029	0.00535
ALS	0.02949	0.10442	0.01994	0.00861
BPR	0.01492	0.05664	0.01298	0.00306
AE	0.07552	0.25841	0.04696	0.02894
IP	0.03754	0.12851	0.02505	0.01080
UIP	0.05629	0.26707	0.0207	0.02035
DRN	0.00861	0.03363	0.0027	0.00401
Ансамбль	0.07159	0.34184	0.03376	0.02385

Следует отметить, что модели AE и UIP имеют одни из лучших метрик, а итоговый ансамбль опережает даже их. Таким образом, при использовании предложенного кастомного ансамбля моделей мы получаем порядка 30 %(по выбранной главной метрике - HUR@5) лучшие результаты без проигрыша во времени выполнения, что говорит о целесообразности применения ансамбля моделей по сравнению с отдельными моделями.

Input:

CB_Recs - рекомендации модели на основе содержимого. K или менее элементов $\in Prods$;

CF_Recs - рекомендации модели коллаборативной фильтрации. K или менее элементов $\in Prods$;

TP_Recs - рекомендации TopK модели. K элементов $\in Prods$;

K - количество рекомендаций.

Output: Out_Recs - финальные рекомендации. K элементов $\in Prods$

```
T ← CB_Recs ∩ CF_Recs;
Out_Recs ← T;
i,j ← 0, sw ← True;
while len(Out_Recs) < K do
    if sw then
        if CB_Recs[i] ∉ Out_Recs and len(CB_Recs) < i then
            Out_Recs.append(CB_Recs[i]);
        end
        i++;
    else
        if CF_Recs[j] ∉ Out_Recs and len(CF_Recs) < j then
            Out_Recs.append(CF_Recs[j]);
        end
        j++;
    end
    sw ← ŝw
end
if len(Out_Recs) < K then
    Out_Recs.join(TP_Recs[K - len(Out_Recs):])
end
```

Algorithm 6: Алгоритм формирования финальных рекомендаций.

Заключение

Резюмируя вышеизложенное, в данной работе создан прототип универсальной рекомендательной системы с адаптивным ансамблем моделей, способной к полностью автоматическому переформированию представлений данных, пересозданию и переобучению моделей, приспособленной для любого интернет-магазина при условии предоставления им стандартизированной выгрузки данных. На данный момент производится A/B тестирование на серверах интернет-магазинов для определения промышленных показателей эффективности системы. Результаты данной работы апробированы на конференциях "Математика. Компьютер. Образование. 2022"[2] и СИТО 2022 [3]. Ссылка на видеопрезентацию МКО-2022 и репозиторий находятся в Приложении 1.

Для достижения цели работы был решен ряд задач, включающих исследование данных, выгруженных с серверов магазина; определение на этой основе теоретической базы применимых моделей машинного обучения для встраивания в рекомендательную систему; создание автоматического пайплайна обработки данных для получения датасетов для обучения моделей.

Далее, был реализован унифицированный интерфейс модели машинного обучения, а также произведен выбор метрик для оценки моделей. Были реализованы семь моделей машинного обучения, а также пайплайн их автоматического пересоздания и переобучения.

Был сформирован набор моделей:

1) модель-baseline, рекомендует N самых популярных товаров(TopN);
2) автоэнкодер(AE);
3) модель на основе матрицы взаимодействия пользователей и товаров(ALS);
4) модель на основе матрицы взаимодействия пользователей и товаров и ранжирования вероятности взаимодействия пользователя и товаров(BPR);
5) модель на основе нахождения похожих на популярные у данного пользователя товаров(IP);
6) модель на основе нахождения похожих на пользователя пользователей

и рекомендации популярных у тех пользователей товаров(UIP);

7) модель с гибридным подходом, основана на сиамской нейросети, которая учится предсказывать вероятность того, что пользователю a понравится товар b. Может принимать на вход не только историю взаимодействия пользователей и товаров, но и специфическую информацию для каждой из сущностей(DRN).

В целях повышения качества рекомендаций реализован автоматический ансамбль на основе авторского алгоритма формирования рекомендаций. По выбранной метрике, ансамбль показывает результат примерно на 30 % лучше отдельных моделей.

Таким образом, разработанная мета-рекомендательная система выполняет заявленные функции и может быть внедрена в деятельность субъектов электронной коммерции.

Литература

1. Официальный сайт McKinsey. — URL: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (дата обр. 20.05.2022)
2. *Соколов М.И., Чердынцева М.И., Прокопенко А.В.* Об одном подходе к созданию мета-рекомендательной системы / XXIX международная научная конференция "МАТЕМАТИКА. КОМПЬЮТЕР. ОБРАЗОВАНИЕ"(24 – 28.01.2022 г.) Материалы конференции. — URL: http://www.mce.su/rus/archive/abstracts/mce29/sect101361/doc399806/ (дата обр. 20.05.2022)
3. *Соколов М.И., Чердынцева М.И.* Прототип мета-рекомендательной системы для интернет-магазинов / XXIX научная конференция "СИТО 2022"(21 – 23.04.2022 г.) Материалы конференции. — URL: https://inftech.uginfo.sfedu.ru/sites/default/files/ProgrammSITO2022.pdf/ (дата обр. 20.05.2022)
4. *Фальк, К.* Рекомендательные системы на практике / пер. с англ. Д. М. Павлова. – М.: ДМК Пресс, 2020. – 448 с.
5. *Hu, Y., Koren, Y. , Volinsky, C.* Collaborative Filtering for Implicit Feedback Datasets. // Proceedings — IEEE International Conference on Data Mining, ICDM 2008. 263-272.
6. *Takács, G., Tikk, D.* Alternating least squares for personalized ranking. — 2012. — URL: https://www.researchgate.net/publication/254464370_Alternating_least_squares_for_personalized_ranking (дата обр. 2.05.2022)
7. Documentation of "implicit" library — https://benfred.github.io/implicit/ (дата обр. 10.05.2022)
8. *Takács, G., Pilászy, I., Tikk, D.* Applications of the conjugate gradient method for implicit feedback collaborative filtering.// Proceedings — 5th ACM conference on Recommender systems, 2011. Association for Computing Machinery, New York, NY, USA, 297–300.
9. *Rendle, S., Freudenthaler, C., Ganther Z., Schmidt-Thieme, L.* BPR:

Bayesian Personalized Ranking from Implicit Feedback. — 2012. — URL:https://arxiv.org/pdf/1205.2618.pdf (дата обр. 15.05.2022)

10. *Kuchaiev, O., Ginsburg, B.*, Training Deep AutoEncoders for Collaborative Filtering. — 2017. — URL: https://arxiv.org/abs/1708.01715 (дата обр. 14.05.2022)
11. Documentation of "scikit-learn" library, Nearest Neighbour module. URL: https://scikit-learn.org/stable/modules/neighbors.html (дата обр. 11.05.2022)
12. *Maneewongvatana, S., Mount, D. M.*, Analysis of approximate nearest neighbor searching with clustered point sets, — 1999 — URL: https://arxiv.org/abs/cs/9901013 (дата обр. 10.04.2022)
13. *Elkahky A., Song, Y., He, X.*, A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. // Proceedings — 24th International Conference on World Wide Web, 2015. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 278–288. — URL: https://doi.org/10.1145/2736277.2741667 (дата обр. 20.05.2022)

Приложения

Приложение 1. Ссылки

Ссылка на репозиторий с кодом проекта:
https://gitlab.com/sfedu_master/recommender_system

Ссылка на видеопрезентацию проекта на конференции МКО-2022:
https://drive.google.com/file/d/1MRJck_qHvx7F4EjwNzrFtUY8x-wwdHqV/view?usp=sharing