ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2023, № 6, с. 150–165

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

УДК 004.89,004.048,004.67

НЕЙРОСЕТЕВЫЕ ПОДХОДЫ ДЛЯ РЕКОМЕНДАТЕЛЬНЫХ СИСТЕМ

© 2023 г. М. А. Жарова[a],*, В. И. Цурков[b],**

[a] *МФТИ, Долгопрудный, Россия*

[b] *ФИЦ ИУ РАН, Москва, Россия*

**e-mail: zharova.ma@phystech.edu*

***e-mail: v.tsurkov@frccsc.ru*

Поступила в редакцию 29.06.2023 г.
После доработки 02.07.2023 г.
Принята к публикации 31.07.2023 г.

Рекомендательные системы – это специальные алгоритмы, которые позволяют пользователям получать персонализированные рекомендации по интересующим их темам. Системы такого рода широко используются в различных областях, например, в электронной коммерции, провайдерских сервисах, социальных сетях и т.д. Наряду с классическими подходами в последние годы в рекомендательных системах стали также популярны нейронные сети, которые постепенно вытесняют традиционные методы коллаборативной фильтрации и контентбазированные алгоритмы. Однако нейросети требуют больших вычислительных ресурсов, в связи с чем часто возникает вопрос: оправданно ли будет увеличение качества и будет ли оно вообще? Проведено исследование нейросетевого подхода в рекомендательных системах – а именно трансформерной модели SASRec из Microsoft Recommenders – и ее сравнение с классическим алгоритмом – гибридной моделью LightFM. Для обучения и валидации применяются данные, взятые из приложения по поиску жилья. В качестве основной метрики для сравнения предлагается использовать HitRate. Результаты экспериментов помогут понять, какие алгоритмы обладают более высокой точностью предсказаний и рекомендаций. Также в качестве дополнительной части рассматривается кластеризация эмбеддингов пользователей и объектов.

DOI: 10.31857/S0002338823060124, **EDN:** FNCIVV

Введение. Простейшим праобразом рекомендательнй системы является приобретение некоторого предмета, когда к нему предлагают другие сопутствующие предметы. Эта модель обобщается. Имеем две группы объектов – пользователей и предметов. Рассматривается задача их взаимодействия, если данных много, то применяются компьютерные алгоритмы.

Развитие нейронных сетей не обошло область рекомендательных систем, в настоящее время существует уже достаточное число таких алгоритмов. Они могут обрабатывать большие объемы данных и выявлять более сложные зависимости между пользователями и объектами, что позволяет создавать более точные и персонализированные рекомендации. Но несмотря на преимущества нейросетевых методов, многие компании до сих пор продолжают использовать в своей работе только алгоритмы матричной факторизации. Это может быть связано с тем, что применение нейросетей более сложно и требовательно к вычислительным ресурсам, также они могут дольше работать по времени в зависимости от архитектуры и нуждаться в большем объеме данных для обучения. Все это может стать проблемами для малых компаний.

Таким образом, с одной стороны, стоит более затратное применение нейросетевых моделей, с другой – гипотетически более высокое качество выдаваемого результата. В работе предлагается исследовать нейросетевой подход в рекомендательных системах – а именно трансформерную модель SASRec из Microsoft Recommenders [1, 2] – и сравнить ее с классическим алгоритмом – гибридной моделью LightFM [3, 4]. Данные модели выбраны не случайно: LightFM является лидером среди представителей классических алгоритмов, превосходя по метрикам в равных условиях бустинги [5] и другие подходы матричной факторизации [6]; SASRec имеет архитектуру трансформер [7], которая также показала свое значительное превосходство в области CV [8, 9] и NLP [10].

С развитием технологий и доступности инструментов для реализации нейросетевых методов с каждым годом все большее число компаний могут позволить себе применять нейросети в качестве собственных решений задач рекомендаций. Обоснованно ли это по метрикам качества таких алгоритмов – предстоит выяснить.

1. Постановка задачи. Пусть имеется множество пользователей $U = \{u_j \mid j \in \overline{1, n_{users}}\}$, множество объектов $I = \{i_j \mid j \in \overline{1, n_{items}}\}$, а также информация про их взаимодействие (например, покупки, звонки или просмотры объявлений). Последнее представляется в виде матрицы $R = \|r_{ui}\|$, по столбцам которой обычно располагают объекты, а по строкам – пользователей. На пересечении ячеек i и j (т.е. для i-го пользователя и для j-го объекта) стоит число, обозначающее их взаимодействие: если i-й пользователь никогда не взаимодействовал с j-м объектом, то на месте соответствующей ячейки матрицы R будет стоять 0; а если касание в том или ином виде было – можно поставить в зависимости от его степени какое-либо натуральное число (например, если пользователь просмотрит объявление – поставим 1, если он сохранил его в закладки – поставим 2, а если купил – 3). Такую матрицу взаимодействий часто называют термином “коллаборативные данные”.

К этому можно добавить также метаданные – расширенную информацию про пользователей и объекты – например, дополнительные характеристики пользователей, такие, как возраст, пол, изначально указанные предпочтения, увлечения и т.д.; для объектов это может быть также категория, цвет, жанр и т.п. в зависимости от содержания.

Сама *задача рекомендаций* заключается в том, чтобы на основе списка (базы) имеющихся объектов составить для каждого конкретного пользователя персональный список объектов, с которыми он еще не взаимодействовал и которые могут быть потенциально ему интересны (можно отранжировать список рекомендаций по релевантности и в таком порядке показывать пользователю). В зависимости от имеющегося набора данных используются различные алгоритмы рекомендаций, для каждого типа существуют отдельные способы оценки качества их работы.

Таким образом, работа рекомендательной системы включает следующие этапы.

1. Сбор данных: можно собрать информацию о взаимодействии пользователей и объектов, таких, как история их предпочтений, покупок, оценок или просмотров (коллаборативные данные). Аналогично собирается информация об объектах и пользователях по отдельности – их характеристики, описание или жанры (метаданные).

2. Предобработка данных: собранные данные подвергаются предварительной обработке, включая очистку, преобразование и структурирование. Например, из коллаборативных данных необходимо удалить дубликаты, малоактивных пользователей, а метаданные привести к векторному виду. Произвольная текстовая информацию может быть обработана с использованием методов естественного языка для извлечения ключевых слов или характеристик.

3. Выбор подхода: в зависимости от типа имеющихся данных выбирается общий подход (тип) построения рекомендательной системы – коллаборативная фильтрация, контентные или гибридные методы. После определения группы модели, с помощью которой можно решить поставленную задачу, можно выбирать уже определенную архитектуру, например из списка state-of-the-art подходов или в соответствии с имеющимися ресурсами и масштабами данных.

4. Обучение модели и генерация рекомендаций: после обучения модели рекомендательная система генерирует персонализированные рекомендации для каждого пользователя. Они могут быть основаны на сравнении предпочтений пользователя с другими пользователями, анализе сходства предметов или применении других методов в зависимости от выбранного подхода (подробнее о каждом подходе будет рассказано в следующем разделе).

5. Оценка и обратная связь: как правило, сначала происходит проверка на валидационном наборе данных, на нем рассчитываются метрики качества работы системы, формируется первичное понимание возможности решения поставленной задачи таким способом. В дальнейшем возможно провести AB-тест на настоящих пользователях и оценить его статистическую значимость. Обратная связь пользователей также может применяться для улучшения системы путем адаптации к изменяющимся предпочтениям и обновлениям.

6. Обновление системы и мониторинг: при эксплуатации чаще всего требуется периодически переобучать модель – например, чтобы включить новые данные, улучшить веса и качество работы, учесть новейшие изменения в предпочтениях и потребностях пользователей.

Рис. 1. Типы рекомендательных систем

В совокупности, работа рекомендательной системы состоит из итеративного цикла сбора данных, анализа, обучения и предоставления персонализированных рекомендаций, а также улучшения их качества с течением времени.

2. Основные типы рекомендательных систем. Существует несколько типов рекомендательных систем, которые используют различные подходы и методы для предоставления персонализированных рекомендаций (рис. 1). Разберем основные из них.

Две большие группы, на которые делятся все рекомендательные системы, – персонализованные и неперсонализованные. Эти типы представляют различные способы построения рекомендательных систем и предоставления рекомендаций пользователям.

Неперсонализованные методы: рекомендации основаны на общих трендах, популярности или общей пользе предметов. Они не учитывают индивидуальные предпочтения и интересы конкретного пользователя. Неперсонализованные рекомендации могут быть полезны, когда у пользователя нет истории взаимодействия или для предоставления общих рекомендаций на базе популярных и актуальных трендов. Примерами использования таких подходов могут быть рекомендации в разделах “Лучшие продажи” или “Популярные товары”; для их реализации достаточно обычных статистических расчетов, без применения каких-либо алгоритмов машинного обучения.

Персонализованные подходы: рекомендации настраиваются в соответствии с индивидуальными предпочтениями, интересами и поведением конкретного пользователя. Они могут учитывать историю взаимодействия с предметами, а также контекстуальную информацию, – такую, как местоположение, пол, возраст пользователя или цвет, размер и другие характеристики для объектов. Персонализованные рекомендации стремятся предлагать для каждого отдельного пользователя наиболее релевантные и интересные именно ему предметы. Для построения таких рекомендаций используются методы машинного обучения – коллаборативная фильтрация, фильтрация контента и другие алгоритмы.

Персонализованные подходы имеют ряд преимуществ, например: повышение удовлетворенности пользователей, улучшение точности рекомендаций, возможность адаптироваться к изменяющимся предпочтениям пользователей и, как следствие, улучшение клиентского опыта, увеличение оборотов компании. Единственное – эти алгоритмы могут столкнуться с проблемой холодного старта, когда у нас недостаточно данных про конкретного пользователя для персонализированных рекомендаций (например, для новых клиентов или вследствие невозможности использования конфиденциальных данных). В таких случаях можно прибегнуть к неперсонализованному подходу.

Остановимся подробнее на персонализованных подходах к построению рекомендательных систем.

C o n t e n t-b a s e d m o d e l. Такие алгоритмы будут рекомендовать пользователю продукты, похожие на те, которые он выбирал ранее. Под похожестью имеются в виду такие сущности, как категория, тег, жанр и т.п., т.е. метаданные. Данные модели анализируют характеристики продуктов, составляют некоторый набор их критериев (жанры, содержание, ключевые слова, кате-

гории и т.д.) и далее предлагают пользователю то, что наиболее схоже по этим критериям с его последними интересами и предпочтениями.

Такие модели хорошо работают в рекомендациях фильмов, музыки, магазинах повседневных товаров. Единственный возможный минус, который здесь в целом можно учесть, – пользователь не попробует новые товары или услуги.

В качестве "меры похожести" часто используют следующие метрики:

1. Индекс Жаккара измеряет сходство между двумя объектами A и B как мощность множества пересечения, деленную на мощность множества объединения каких-либо характеристик объекта (мощность множества X обозначается как $|X|$). Применяется в основном для категориальных признаков (при необходимости может быть интерпретирован и на числовые):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

2. Косинусное расстояние. Оценка векторов по косинусному расстоянию между ними используется для оценки близости массивов с числами:

$$similarity = \cos\theta = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}.$$

Collaborative filtering. Коллаборативная фильтрация – это системы, в которых рекомендации пользователю рассчитываются на основе его оценок, а также оценок других пользователей. Применение этого подхода дает лучшее качество [11]; можно прогнозировать рейтинги по оценкам похожих пользователей или для похожих продуктов. Именно по этим признакам выделяют несколько подтипов таких моделей.

Итак, имеем множество пользователей, множество продуктов и список оценок некоторых пользователей по некоторым продуктам. Можно представить эти данные в виде "троек" (u, i, r_{ui}), где u обозначает пользователя, i – продукт, r_{ui} – рейтинг, который пользователь u поставил продукту i. Далее можно расположить их в виде матрицы, каждая строка которой соответствует пользователю, а столбец – продукту.

Наша задача – предсказать неизвестные элементы матрицы, т.е. рейтинги для продуктов, которые пользователь еще не видел. Далее можно отсортировать их для каждого пользователя и понять, какие продукты потенциально наиболее релевантны для него. Это и будет список рекомендаций.

Memory-based. Так как матрица собрана сразу и по пользователям, и продуктам, в первую очередь можно применить для нее следующие алгоритмы:

user2user (поиск соседей-пользователей по оценкам);

item2item (поиск схожих предметов на основе оценок пользователей).

В целом, они идентичны друг другу, первый заключается в следующем:

1) определить, насколько другие пользователи в базе данных похожи на данного пользователя (например, используя меры схожести, описанные в предыдущем параграфе);

2) по оценкам других пользователей предсказать, какую оценку даст данный пользователь данному продукту, учитывая с большим весом тех, которые больше похожи на исходного.

А второй можно описать так:

1) определить, насколько другие продукты в базе данных похожи на данный;

2) по оценкам других продуктов предсказать, какую оценку даст данный пользователь данному продукту, учитывая с большим весом те продукты, которые больше похожи на данный.

Таким образом, все сводится опять же к поиску похожих пользователей или объектов, но исходя из коллаборативной информации, а не метаданных. Эти подходы достаточно редко используются на практике в силу их трудозатратности. Необходимость вычислять попарные расстояния для всего множества, а затем их сортировка занимают довольно много времени.

Model-based. В силу недостатков предыдущего подхода на практике чаще используются алгоритмы из семейства model-based, эту группу называют просто "матричная факторизация". На данный момент они остаются самыми популярными в области рекомендательных систем

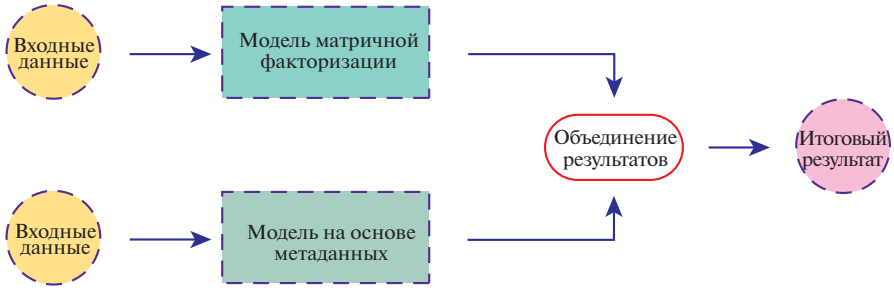

Рис. 2. Схема работы гибридного алгоритма

благодаря своей простоте и эффективности. Для memory-based схожесть между пользователями и схожесть между объектами рассчитывалась по отдельности. Но что если по той же матрице “пользователь-объект” обучить факторы, характеризующие и пользователей, и объекты одновременно?

Основная идея методов матричной факторизации заключается в разложении исходной матрицы пользователей и предметов на две более низкоранговые – так называемые матрицы факторов. Они представляют собой скрытые признаки пользователей и объектов, а их произведение дает оценки или предсказания рейтингов для всех возможных комбинаций пользователей и предметов. Матрицы факторов могут быть обучены с помощью различных методов оптимизации, градиентного спуска или сингулярного разложения.

Преимущества методов матричной факторизации:

способность обрабатывать разреженные данные: матрицы “пользователь–объект” практически всегда содержат большое количество нулей, так как, как правило, пользователи взаимодействуют лишь с небольшой частью объектов;

учет скрытых признаков (факторов): методы этой группы позволяют моделировать скрытые характеристики пользователей и предметов. Например, в рекомендательных системах для фильмов скрытые факторы могут представлять жанры, настроение, стиль и т.д. Это даст возможность системе делать более точные предсказания, основанные на глубоком понимании предпочтений пользователей.

Из недостатков: невозможность выставлять рекомендации для полностью новых пользователей, которые пока никак не провзаимодействовали ни с одним из объектов. Также в таких алгоритмах не предусмотрено использование метаинформации, которая тоже может быть полезна.

Гибридные модели. Наверное, не найдется такой сферы деятельности, в которой бы не возникло идеи собрать все лучшее воедино. Рекомендательные системы не стали исключением – группу моделей, которые сочетают в себе характеристики различных типов алгоритмов, назвали гибридными.

Работа гибридных рекомендательных систем основывается на интеграции различных методов и подходов для генерации рекомендаций. Самые распространенные типы комбинирования:

по отдельности реализовать коллаборативный и контентный алгоритмы, а затем объединить их результаты-рекомендации;

добавлять контентные правила в коллаборативный подход;

добавлять коллаборативные правила в контентный подход;

построить одну общую модель, включающую в себя правила обоих подходов.

Обычно эти варианты берут в качестве основы и дополняют по собственному желанию и согласно критериям сферы деятельности. Ниже приведен пример схемы работы гибридной рекомендательной системы на основе объединения результатов коллаборативного и контентного алгоритмов (рис. 2).

Единственный недостаток гибридных систем – сложность разработки и повышенные технические требования к ресурсам для обучения и инференса таких алгоритмов.

3. LightFM – это библиотека, разработанная для построения гибридных рекомендательных систем. Она предоставляет простой и эффективный способ создания и оценки моделей рекомендаций, объединяя коллаборативную фильтрацию и фильтрацию контента [3].

Преимущества LightFM содержат:

гибкость: LightFM позволяет интегрировать различные типы данных, включая информацию о взаимодействиях пользователей, контентные признаки предметов и контекстуальную информацию;

эффективность: библиотека предоставляет оптимизированные алгоритмы для обучения моделей рекомендаций на больших и разреженных данных;

поддержка различных метрик: LightFM дает возможность оценивать качество рекомендаций с помощью большого числа встроенных метрик, таких как точность, покрытие, разнообразие и др.;

простота использования: API LightFM дружественен к пользователю и обладает простым интерфейсом для построения и оценки моделей рекомендаций.

Рассмотрим подробнее поддерживаемые алгоритмы ранжирования данной библиотеки.

Bayesian Personalized Ranking (BPR). Основная идея заключается в выборке и попарном сравнении положительных и отрицательных объектов (под отрицательными можно понимать те, с которыми пользователь не взаимодействовал). Алгоритм в упрощенном виде можно представить следующим образом.

Случайным образом возьмем пользователя u и объект i, который ранее был выбран пользователем. В таком случае i будет считаться положительным.

Случайным образом возьмем объект j, который был выбран пользователем реже, чем i (в том числе, который пользователь никогда не выбирал). Тогда j будет считаться отрицательным.

Вычисляем оценку P_{ui} и P_{uj} пользователя u, а также положительного объекта i и отрицательного j соответственно.

Находим разницу между положительными и отрицательными оценками: $x_{uij} = P_{ui} - P_{uj}$.

Пропускаем эту разницу через сигмоиду и вычисляем веса для обновления всех параметров модели с помощью градиентного шага.

Weighted Approximate-Rank Pairwise (WARP). Концепция данного подхода схожа с предыдущим, за исключением случаев, когда происходит градиентный шаг.

В BPR градиентный шаг происходит каждый раз с разницей в качестве веса.

WARP совершает градиентный шаг только в случае неверного предсказания (т.е. когда оценка отрицательного объекта оказалась больше положительного). Если предсказание было верным, то продолжаем выбирать отрицательные объекты, пока не получим неверный прогноз или не достигнем некоторого порогового значения.

Для этих целей WARP предоставляет два гиперпараметра:

Margin определяет, насколько ошибочным должен быть прогноз для совершения градиентного шага;

Cutoff определяет, сколько раз можно выбирать отрицательные примеры, пытаясь получить неверное предсказание, прежде чем алгоритм остановится и перейдет к следующему пользователю.

На практике, вероятнее всего, WARP предпочтительнее для большинства рекомендательных систем, нежели BPR [12], в дальнейших экспериментах будем использовать его.

Также в реализации данной модели [4] можно варьировать большое число гиперпараметров: размерность скрытых факторов; максимальное количество положительных результатов для каждого обновления; максимальное количество отрицательных выборок, используемых при подборе WARP; можно указать номер положительного примера, который будет выбран из всего набора таковых для каждого пользователя; есть возможности для применения различных множителей регуляризации. Благодаря широкому выбору гиперпараметров и алгоритмов, возможности соединить все методы и данные воедино, LightFM находится в числе лучших классических моделей рекомендательных систем. Это подтвержают многочисленные сравнения алгоритма с бустингами и отдельными классическими реализациями алгоритмов рекомендаций [5, 6, 13].

4. Глубокое обучение и модели-трансформеры. В основном известно, что нейронные сети применяются в задачах компьютерного зрения, обработки текстов и аудио. Deep Learning часто превосходит классические методы Machine Learning (ML), поэтому сейчас многие компании внедряют их для работы с табличными данными, в том числе для задач рекомендаций. Несмотря на то, что в области рекомедательных систем было предложено много разнообразных и эффективных методов, нейросети пришли в эту область относительно недавно.

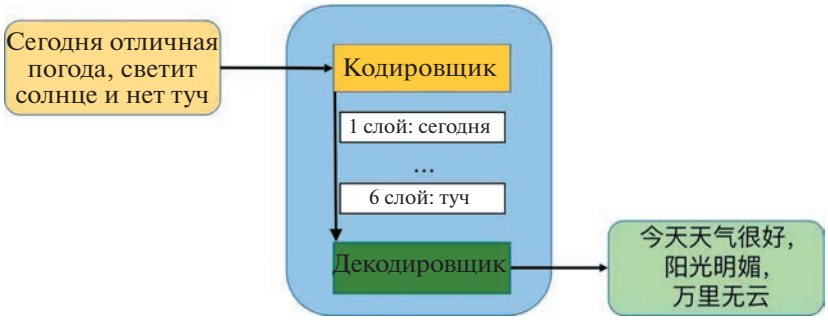

Рис. 3. Простейшая схема работы трансформерной модели

Архитектурные особенности таких моделей берут идеи из разных подходов. Преимущество использования нейронных сетей:

более высокое качество по сравнению со стандартными ML-моделями;

большая гибкость – в рамках одной модели можно получить ответы сразу на несколько вопросов, например: “Дабавит, ли пользователь товар в корзину?”, “Начнет, ли он оформление заказа с этим товаром?” или “Купит, ли он этот товар?”;

возможность применения различных типов данных – картинок и текста, а также алгоритмов обучения с подкреплением.

Обычно архитектура всех нейросетей в рекомендательных системах устроена следующим образом: для каждого пользователя и продукта создается вектор (случайный или некоторым образом кодирующий исходные данные), который обучается – меняется после каждого прохода через слои нейросети. Таким образом, после этого процесса получаются обученные эмбеддинги пользователей/продуктов. Здесь можно провести аналогию с матрицами, которые получаются после разложения в алгоритмах матричной факторизации. Каждое число в эмбеддинге тоже характеризует некоторый фактор пользователя или объекта, только выявляются они нейросетями, а не обычным разложением, что, конечно, дает более четкие зависимости. Далее можно делать с этими эмбеддингами (матрицей эмбеддингов) что угодно – перемножать, рассчитывать косинусное расстояние, корреляцию Пирсона, использовать как признаки для моделей второго уровня и т.д., находя таким образом схожести между пользователями, объектами и выдавая рекомендации.

Наряду с комбинацией нейросетевого и классического подходов рекомендательная система может быть полностью построена на основе нейросети. Рассмотрим принцип работы и архитектуру моделей-трансформеров. Первыми они применялись в задачах обработки естественного языка (рис. 3).

Кратко опишем их принцип работы следующим образом.

1. Исходные данные (текст/картинки/последовательность действий) переводятся в набор эмбеддингов при помощи энкодера (от англ. encoder), в зависимости от типа данных и типа разбиения их количество может быть разным (на на рис. 3, например, эмбеддинг сопоставляется каждому слову в предложении). Делается это алгоритмами кодирования и понижения размерности.

2. Происходит обучение нейросети, здесь главную роль играет слой внимания или self-attention [7]. Его суть состоит в том, что для каждого эмбеддинга он учитывает его контекст. В задачах обработки текста смысл этого очевиден – можно таким образом понять, насколько слово важно в предложении, с какими оно больше всего связано и т.д. Для рекомендательных алгоритмов, например, определяется связь действий одного пользователя и выявляются в них зависимости. Под “капотом” слоя внимания находятся математические вычисления – для каждого эмбеддинга рассчитывается несколько значений (query – q, key – k, value – v) путем перемножения этого эмбеддинга на соответствующие матрицы Q, K, V, которые в свою очередь получаются в процессе обучения. Затем они перемножаются друг с другом для разных элементов, нормируются на размерность векторов матрицы K (обозначается d_k) и прогоняются через softmax. Функция “внимания” определяется следующим образом:

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V.$$

Напомним, что здесь softmax – функция активации, которая применяется для выходных значений какого-либо слоя:

$$\mathrm{softmax}(w_1^{\mathrm{T}}x,\ldots,w_k^{\mathrm{T}}x) = \frac{\exp(w_k^{\mathrm{T}}x)}{\sum\limits_{i=1}^{k}\exp(w_i^{\mathrm{T}}x)},$$

где w_i – значение веса соответствующего нейрона в слое, x – вектор выходных значений.

По смыслу концепция query/key/value (ключ/значение/запрос) аналогична поисковым системам. Например, когда пользователь что-то ищет в Интернете, поисковая система сопоставит его запрос (текст в строке поиска) с набором ключей (название сайтов, описание и т.д.), связанных с кандидатами в своей базе данных, а затем представит наиболее подходящие ссылки на сайты (значения).

3. Далее запросы, ключи и значения линейно проецируются на матрицы K, Q, V. Для каждой из этих спроецированных версий запросов, ключей и значений параллельно рассчитывается функция внимания *Attention*: на вход поступают d_v-мерные выходные значения, затем они объединяются и снова проецируются (d_v – размерность векторов в матрице V). Данный механизм называют Multi-head attention, математически можно записать эту функцию следующим образом:

$$MultiHead(Q,K,V) = Concat(head_1,\ldots,head_h)W^Q,$$

где $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$, $W_i^Q \in \mathbb{R}^{d_{model}d_q}$, $W_i^K \in \mathbb{R}^{d_{model}d_k}$, $W_i^V \in \mathbb{R}^{d_{model}d_v}$, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ – матрицы проекций, h – количество параллельно вычисляемых слоев внимания, а *Concat* – операция "склеивания" справа-налево векторов в матрицу. Данный механизм позволяет модели совместно воспринимать информацию из разных подпространств представления в разных позициях. При одном слое внимания усреднение препятствует этому.

4. После слоя внимания можно вставить еще несколько различных слоев нейросети. В оригинале статьи – это два линейных преобразования со следующей функцией активациии ReLU между ними:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2,$$

т.е. проводится две свертки с размером ядра 1.

5. После всех проделанных манипуляций обученные эмбеддинги декодируются обратно в исходный тип данных, например в набор рекомендаций или переведенный текст.

Создание архитектуры трансформера стало своего рода одной из революционных точек в развитии глубокого обучения нейросетей, дав мощный толчок в первую очередь направлению языковых моделей [8]. На данный момент есть уже достаточно много результатов, показывающих их преимущества [8, 14]. После аналогичные модели появились в области компьютерного зрения [9, 10] и не обошли область рекомендательных систем [1].

5. SASRec (self-attentive sequential recommendation) – это алгоритм рекомендаций, основанный на архитектуре Transformer [1]. Он обрабатывает и моделирует последовательности взаимодействий пользователей с предметами для предсказания следующего наиболее вероятного предмета, с которым пользователь может взаимодействовать.

Основные шаги работы алгоритма.

1. Входные данные должны состоять из истории взаимодействий пользователей с предметами, представленных в виде последовательностей. Каждая последовательность – список предметов, с которыми пользователь взаимодействовал, отсортированный по времени.

2. Входные последовательности обрабатываются, чтобы сделать их одинаковой длины, например путем обрезания или дозаполнения. Также выполняется кодирование предметов и пользователей в числовые индексы.

3. SASRec – нейросетевая модель, поэтому для работы ей необходимы векторные представления для пользователей и предметов – эмбеддинги. В самом начале они инициализируются случайным образом и далее меняются в процессе обучения модели.

4. SASRec использует архитектуру Transformer, поэтому здесь приходится иметь дело со слоями внимания. Multi-head attention дает возможность модели обрабатывать последовательности

входных данных с учетом их взаимосвязей и зависимостей. Подобно словам в предложении, оно дает возможность модели сосредоточиться на важных аспектах последовательности действий пользователя и учитывать “контекстные” зависимости. А позиционно-сетевое представление добавляет информацию о месте элементов в последовательности, это важно для моделирования зависимостей между ними.

5. Модель обучается на “положительных” и “отрицательных” последовательностях: сначала для каждого элемента рассчитываются логиты, после чего методом WARP (описан подробно выше) оптимизируется функция потерь.

6. После обучения модель может использоваться для генерации рекомендаций. Для каждого пользователя прогнозируются вероятности взаимодействия с каждым предметом из общего набора, после чего они ранжируются и “верхние” позиции в первую очередь попадают в список рекомендаций.

Из преимуществ, доставшихся SASRec в наследство от трансформерной архитектуры, выделяется способность учитывать долгосрочные зависимости и контекстные особенности в последовательностях взаимодействий пользователей с объектами, что делает ее эффективной для задач последовательных рекомендаций.

6. Оценка эффективности моделей и метрики качества. В рекомендательных системах используются различные метрики для оценки и измерения качества предоставляемых рекомендаций. Рассмотрим некоторые из наиболее распространенных.

Точность (precision): измеряет долю релевантных рекомендаций среди всех. Она вычисляется как отношение числа релевантных рекомендаций к общему числу рекомендаций и показывает, насколько они полезны для пользователей.

Полнота (recall): измеряет долю релевантных рекомендаций, найденных среди всех действительно релевантных элементов. Она вычисляется как отношение числа релевантных рекомендаций к общему числу действительно релевантных элементов. Полнота показывает, насколько система способна найти все подходящие элементы.

F-мера (F-measure): объединяет точность и полноту в одну общую меру качества. F-мера вычисляется как гармоническое среднее между точностью и полнотой и позволяет учесть обе эти метрики одновременно.

Индекс покрытия (coverage): измеряет долю уникальных элементов, для которых были сделаны рекомендации, относительно общего числа уникальных элементов в системе. Индекс покрытия показывает, насколько система разнообразна и способна предложить рекомендации для широкого спектра элементов.

Ранговая корреляция (ranking correlation): оценивает степень соответствия порядка рекомендованных элементов и фактического порядка предпочтений пользователей. Некоторые из популярных ранговых корреляционных метрик включают коэффициенты Спирмена и Кендалла.

Среднее смещение ранжирования (mean average precision, MAP): учитывает ранжирование элементов в рекомендациях. Она измеряет среднее значение точности для каждого релевантного элемента в ранжированном списке рекомендаций. MAP позволяет учесть как точность, так и порядок элементов в рекомендациях.

Нормализованная взаимная информация (normalized discounted cumulative gain, NDCG): учитывает релевантность и ранжирование элементов в рекомендациях. Она присваивает более высокие значения более релевантным и лучше ранжированным элементам в рекомендациях.

Метрика Hit Rate (или Hit Ratio) измеряет долю рекомендаций, которые содержат хотя бы один релевантный элемент для пользователя. Она показывает, насколько успешно рекомендательная система предлагает пользователю хотя бы один элемент, который он считает релевантным.

Обычно выбор метрики напрямую связан со способом тестирования. Здесь существует два принципиально разных подхода.

1. Помещать для каждого пользователя несколько объектов в тестовую часть. Тогда фактически необходимо предсказать последние 20–30% элементов, которые понравятся пользователю (в зависимости от размера тестового набора данных). В качестве метрик здесь нет ограничений – можно использовать любые.

2. Помещать для каждого пользователя только один объект в тестовую часть. Тогда необходимо предсказать только один следующий интерес пользователя. Этот подход хорош тем, что большее количество данных можно оставить для обучения модели, но появляются ограничения в

используемых метриках – чаще всего будет полезен hit rate, NDCG, а вот recall и precision применить уже не получится.

7. Проведение эксперимента. 7.1. О п и с а н и е д а н н ы х. Обучать обе модели будем только на коллаборативных данных. Напомним, что это данные, основанные на истории взаимодействия пользователей с предметами; используюся в моделях коллаборативной фильтрации для рекомендаций объектов, схожих по скрытым факторам с теми, что конкретный пользователь уже просмотрел или оценил.

Такие данные часто представляются в виде матрицы, которая содержит информацию о том, какие пользователи взаимодействовали с какими объектами. Обычно их располагают по строкам и столбцам соответственно, а ячейки заполняются нулями или единицами – в зависимости от того, взаимодействовал пользователь с этим объектом или нет. Например, в случае рекомендации фильмов можно записать в нее, какие зрители и какие фильмы оценили или просмотрели.

Часто также применяется метод записи коллаборативных данных в виде простой последовательности “пользователь–объект”. Из недостатков – это занимает больше места, сложнее учитывать вес взаимодействий. Но зато появляется возможность использовать временные характеристики – какие взамодействия были раньше, а какие позже, что очень важно для нейросетевых моделей-трансформеров, так как их основное преимущество заключается в проведении анализа поведения пользователя во времени и генерация рекомендаций с учетом всех предыдущих зависимостей в поведении.

Перейдем к описанию данных, используемых в эксперименте. Их тематика – действия пользователей на сайте по поиску жилья, интерфейс системы выглядит следующим образом (рис. 4).

Пользователи могут задавать определенные фильтры системе – комнатность, этажность здания; аренда или продажа и др. В первую очередь показываются объявления, которые наиболее релевантны его предыдущим интересам и запросу. Для построения моделей нам понадобятся уникальные номера (ID) пользователей, объявлений, а также временная отметка. Промежуток, за который были выгружены данные – июль 2022 г. В итоге получилось 431 982 уникальных пользователя и 388 607 уникальных объектов, общее число исходных записей равно 82 463 346.

В коллаборативную матрицу можно записать следующие виды взаимодействий (признак *event_name*):

просмотр,

добавить в “избранное”,

показать контакты продавца,

сделать звонок,

завершить звонок.

Возможность учета веса события доступна только для модели LightFM, поэтому для более корректного сравнения заполним матрицу бинарными значениями 0 или 1, чтобы наилучшим образом соответствовать модели SASRec.

Для составления матрицы “пользователь–продукт” понадобятся только пары признаков ID пользователя – ID объекта в виде истории их взаимодействий (напомним, что свойства объектов в коллаборативной фильтрации напрямую не учитываются).

Закодируем пользователей и объявления натуральными числами, начиная с нуля, чтобы в итоговой матрице можно было провести биекцию между индексом строки или столбца и исходным ID. Для этого используем *category_encoder*.

По трем столбцам *user_id_enc*, *offer_id_enc* и искусственно созданному столбцу *action*, заполненному единицами для датасета (рис. 5), можно построить матрицу “пользователь–объект”, которая понадобится для обучения LightFM. Так как такие матрицы в большинстве случаев довольно разреженные (пользователи взаимодействуют далеко не со всеми объектами), для них был создан особый формат – *sparse* или *csr*, *matrix*. Они устроены таким образом, что нули в них не хранятся. Так как они составляют основную часть данных, то не занимают лишней памяти, в условиях разреженности это становится особенно актуально. Для SASRec же достаточно просто выделить столбцы *user_id_enc*, *offer_id_enc* в отдельный текстовый файл, отсортировав по *user_id_enc*, и по времени.

Перед составлением итогового набора данных в виде матрицы или последовательности с датасетом также были произведены следующие стандартные преобразования по очистке:

удалены дубликаты в связке “пользователь–объект”;

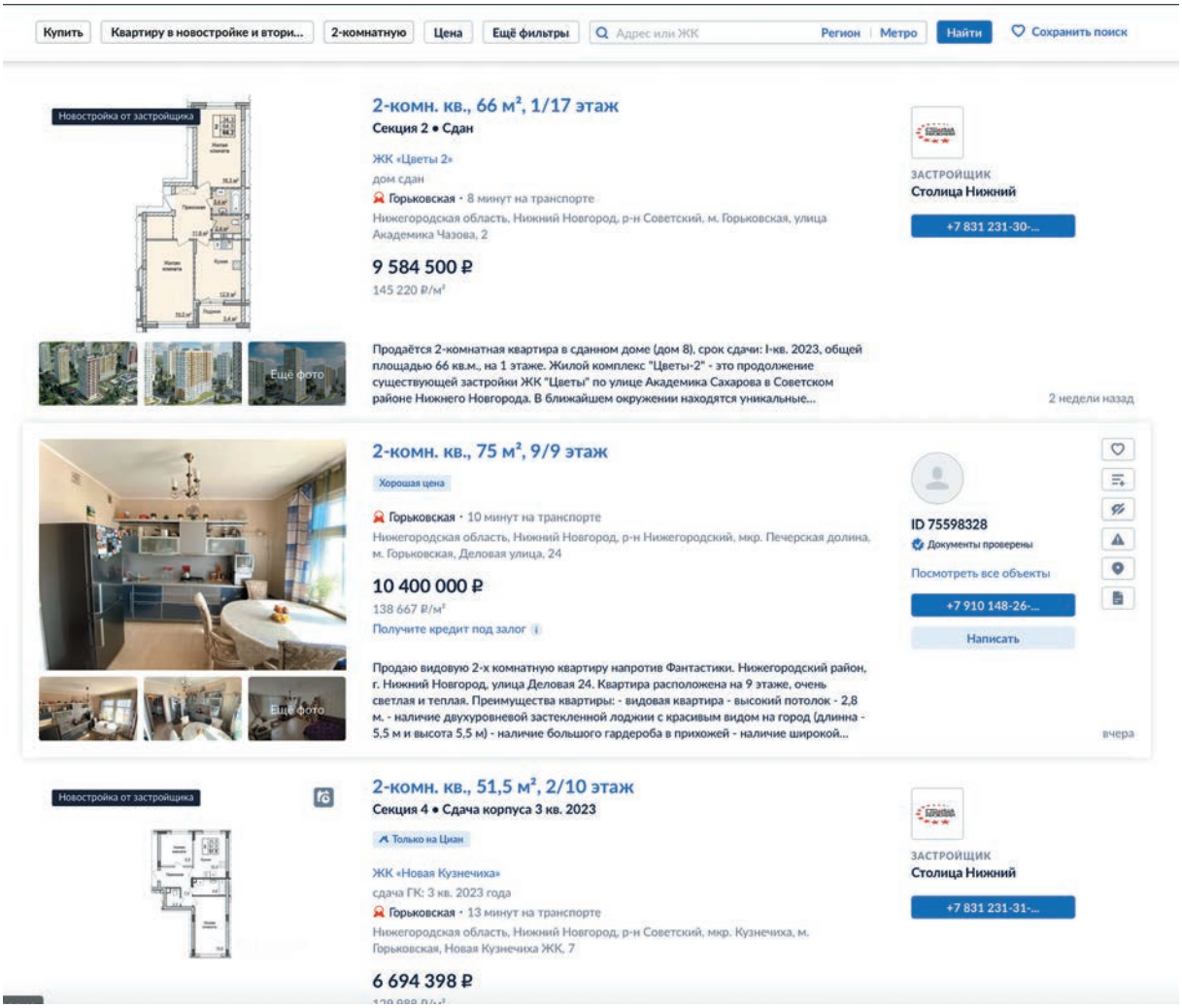

Рис. 4. Пример интерфейса сайта

	timestamp	user_id	offer_id	event_name	user_id_enc	offer_id_enc	event_name_enc
0	2022-07-29 00:00:02.095	82662998	275699899	OpenOfferScreen	0	0	1
1	2022-07-29 00:00:20.794	70430591	263076121	Add2Like	1	1	2
2	2022-07-29 00:00:21.326	93889189	254183210	OpenOfferScreen	2	2	1
3	2022-07-29 00:00:26.945	91525566	268686455	OpenOfferScreen	3	3	1
4	2022-07-29 00:00:27.162	78613959	275660175	OpenOfferScreen	4	4	1

Рис. 5. Исходные данные с закодированными ID пользователей и объектов

взяты только пользователи, у которых более пяти действий за указанный промежуток времени;

последовательность дополнительно отсортирована по коду пользователя, а затем по времени для нейросетевой модели.

После всех преобразований осталось 8 666 106 записей, из которых часть будет отдана на обучение моделей, часть – на оценку качества работы.

7.2. Выбор унифицированного подхода тестирования. Существует два кардинально отличающихся подхода проверки качества работы рекомендательных систем. Они связаны с различными способами прогнозирования поведения пользователя: предсказание некото-

рого процента следующих кликов или предсказание только одного следующего действия пользователя. Рассмотрим их подробнее и разберем преимущества и недостатки.

1. Предсказание процента кликов (Click-Through Rate, CTR): в таком подходе из общего числа данных обычно выделяют 20–30% для тестирования и валидации, остальное – для обучения.

Преимущества подхода:

более состоятельные оценки, можно попытаться оценить поведение пользователя не только в начале, но и в перспективе;

может быть использован для оценки эффективности алгоритмов по всем возможным метрикам.

Недостатки подхода:

может стать критичным в условиях ограниченности данных, так как значительная их часть пойдет на тестирование, а не на обучение, что может повлиять на качество модели в целом;

возможна предвзятость в пользу популярных элементов. Некоторые модели могут склоняться к рекомендации объектов с высоким средним рейтингом, что может привести к искажению оценки эффективности рекомендаций.

2. Предсказание следующего действия пользователя (Next Action Prediction): в таком подходе из общего числа данных выделяется по одному действию пользователя для тестирования и валидации, остальное – для обучения.

Преимущества подхода:

получаем больше данных для обучения модели, что позволяет улучшить предсказания; это полезно, когда рекомендательная система должна предлагать наиболее релевантные элементы, учитывая контекст и последовательность всего предыдущего (в особенности последних действий), а также в условиях ограниченности данных;

как следствие, более эффективен для задачи последовательной рекомендации, так как позволяет лучше учесть как краткосрочные зависимости, так и изменение предпочтений пользователя со временем.

Недостатки подхода:

само по себе качество тестирования будет менее информативным, особенно, если задача заключается в том, чтобы предсказать для пользователя последовательность из набора следующих объектов, а не только один;

ограниченность использования метрик – например, не получится воспользоваться оценками “at k”.

Выбор между этими двумя подходами зависит от конкретного контекста и целей рекомендательной системы. В нашем исследовании был выбран второй подход, так как данных по итогу получилось не очень много, и он более подходящий для моделей последовательных рекомендаций, в числе которых SASRec. В качестве главной метрики возьмем Hit Rate; в самой библиотеке уже реализован алгоритм такого тестирования, которым можно воспользоваться.

8. Обучение LightFM. Зафиксируем основные детали при обучении модели и результаты.

1. Обучащие данные: матрица взаимодействий “пользователь–объект” в формате *csr*, *matrix* на основе всех данных, за исключением двух последних действий для каждого пользователя. Тестовые и валидационные данные: для каждого пользователя взято по два последних клика.

2. Обучение происходило за одну итерацию в 150 эпох на заранее эмпирически подобранных оптимальных гиперпараметрах. Это достаточно точный процесс, значения указаны с большой точностью, так как именно они дают наилучшее качество модели (подробно указаны в табл. 1).

3. Оценка качества проводилась методом MAP, метрика HitRate составила 0.047 или 4.7% точности предсказания последующего клика пользователя.

4. Обсуждение адекватности полученной метрики: на достаточно больших наборах данных известно, что “хорошее” значение метрики *precision*@5 равняется 30%, на малых наборах для этой же модели на том же наборе данных получилось *precision*@5 = 20%. Если примерно экстраполировать HitRate на пять элементов, получается значение в этих пределах (если просто умножить на пять, получится 23.5%, т.е. даже присутствует эффект наличия более релевантных рекомендаций в начале списка). Это позволяет утверждать о правильности полученной метрики и корректно проведенном эксперименте.

Таблица 1. Гиперпараметры для обучения модели LightFM

Гиперпараметр	Значение на обучении
loss	warp
max_sampled	19
no_components	150
user_alpha	0.000012403932688782397
item_alpha	0.00007600815597574313
learning_rate	0.3702643399600358

Таблица 2. Гиперпараметры для обучения модели SASRec

Гиперпараметр	Значение на обучении
num_epoch	30 + 14
batch_size	200
RANDOM_SEED	100
lr	0.001 + 0.0001
maxlen	50
num_blocks	2
hidden_units	100
num_heads	1
dropout_rate	0.1
_emb	0.0
num_neg_test	20577

9. Обучение SASRec. Аналогично зафиксируем основные детали при обучении модели SASRec.

1. Обучащие данные: последовательность взаимодействий “пользователь–объект” в формате обычной записи пар в текстовом файле в необходимом порядке на основе всех данных, за исключением двух последних действий для каждого пользователя. Тестовые и валидационные данные: для каждого пользователя взято по два последних клика.

2. Обучение происходило за две итерации: 30 эпох на заранее эмпирически подобранных оптимальных гиперпараметрах (подробно указаны в табл. 2) с шагом $lr = 0.001$ и дополнительно 14 эпох с шагом $lr = 0.0001$ (так как обучение с 30+ эпохи с $lr = 0.001$ приводило к переобучению).

3. Оценка качества проводилась методом NAP, метрика HitRate составила 0.140 или 14.0% точности предсказания последующего клика пользователя.

4. Обоснование адекватности полученной метрики приведено в разд. 8, здесь же она составила почти в 3 раза больше. Отдельного тестирования методом CTR (Click Through Rate) для данной модели не проводилось, поэтому сложно утверждать, насколько релевантными могли бы быть рекомендации на последующих местах списка.

10. Результаты. Финальная метрика Hit Rate (число правильно предсказанных следующих действий пользователя) составила:

для LightFM 0.047,

для SASRec 0.140.

Нейросетевая модель показала по целевой метрике преимущество почти в 3 раза по качеству. При этом нужно учесть, что суммарное время обучения LightFM составило примерно 10 мин, для SASRec – 30 мин. – разница тоже в 3 раза. Можно утверждать, что на данном наборе данных достигнуто максимальное качество для обеих моделей, так как при превышении 150 эпох LightFM переставала показывать прирост в метриках. В силу строения модели это означает, что метод сошелся к оптимуму, дальнейшее увеличение качества невозможно. Для SASRec на исходном наборе данных также при дальнейшем обучении алгоритма метрики начинали падать, что свидетельствует о переобучении.

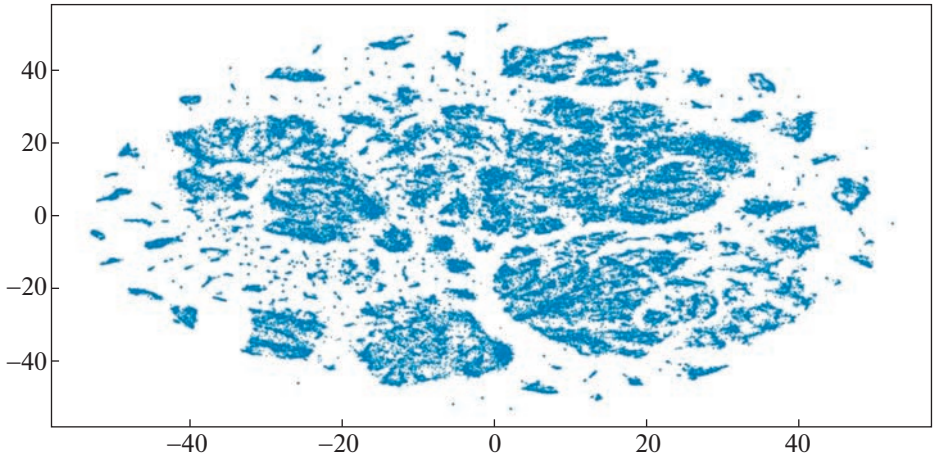

Рис. 6. Визуализация эмбеддингов объявлений после TSNE-разложения

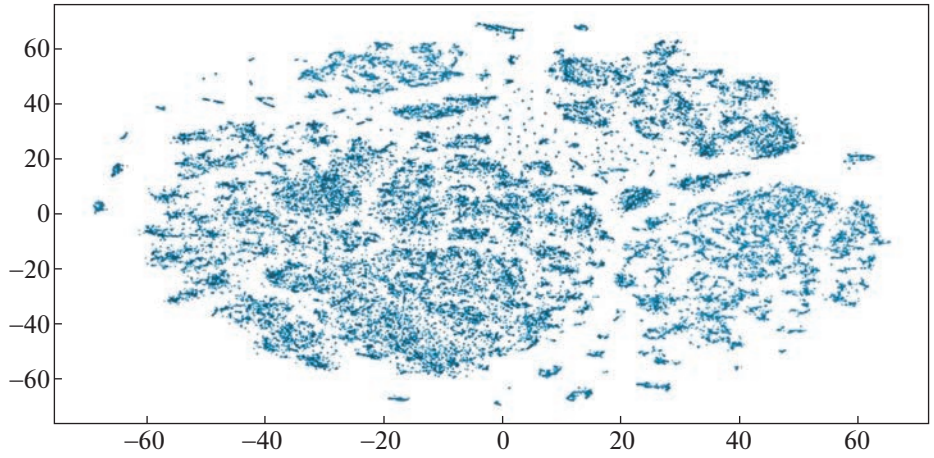

Рис. 7. Визуализация эмбеддингов пользователей после TSNE-разложения

В качестве дополнительного подтверждения корректной работы нейросетевой модели из нее были извлечены эмбеддинги пользователей и объявлений, исходная размерность векторов равнялась 100 (гиперпараметр модели, который был задан вручную – *hidden_units*). Далее размерность векторов была понижена при помощи TSNE-разложения до двух, получившаяся визуализация представлена на рис. 6 и 7.

На графиках каждая точка соответствует какому-либо одному пользователю или объявлению, прослеживается разделение на кластеры, что уже хорошо. Значит, модель уловила их скрытые факторы. Предположительно, в каждом кластере находятся близкие по интересам пользователи или близкие по содержанию объявления.

Проверим гипотезу на простом интерпретируемом признаке для пользователей – предпочтении в покупке или продаже. Для каждого обявления из исходного набора данных можно определить, относится оно к категории продажи или аренды. Далее разметим пользователей: если больше 70% их интересов (просмотров объявлений) относятся к категории объектов продажи, то присвоим им целевую переменную 1; если аналогичная ситуация справедлива для категории объектов аренды, то поставим в таргет 0. К счастью, “промежуточных” пользователей не возникло. За месяц все исключительно интересовались преимущественно либо арендой, либо продажей.

При визуализации разметки “продажи–аренды” на графике рис. 7 получилась следующая картинка (рис. 8): отмечены области, в которые попали пользователи, интересующиеся преимущественно арендой, остальная часть – продажами.

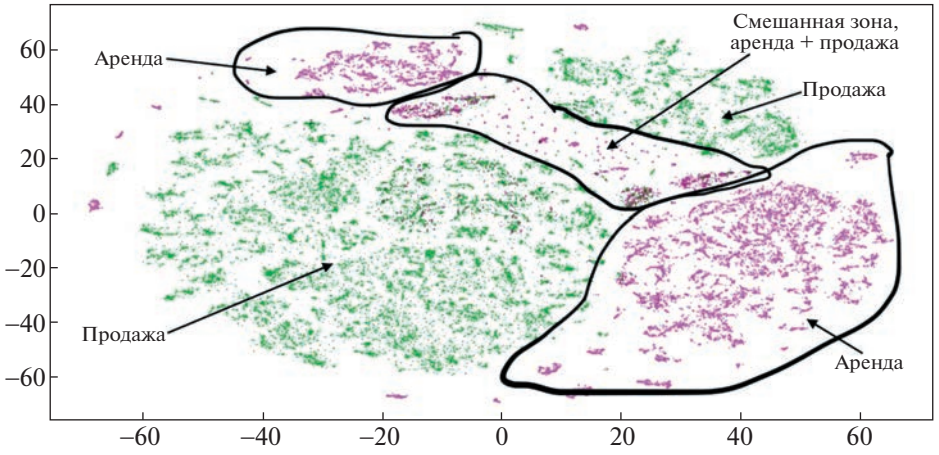

Рис. 8. Визуализация эмбеддингов пользователей по предпочтениям продажи и аренды

Видно, что разделения обеих групп проходят четко по границам некоторых кластеров, выделившихся в простанстве после TSNE-разложения. Это уже наглядно свидетельствует о хорошей кластеризации, что, в свою очередь, – об информативных эмбеддингах и корректно проведенном обучении.

Подтвердим визуальный результат численным – соберем датасет, в котором признаками будут выступать эмбеддинги, а целевой переменной – бинарный признак предпочтения в “аренде–продаже”. Обучим на этих размеченных данных классификатор CatBoost, метрики получились следующие:

precision 0.97,

recall 0.96,

roc-auc 0.99.

Классификация на датасете, составленном из эмбеддингов, показала высокие результаты, что подтверждает численно качество полученных эмбеддингов (и качественно обученную модель SASRec), а также предоставляет возможности для переиспользования эмбеддингов для решения других различных задач.

Заключение. Проведено детальное исследование двух моделей коллаборативной фильтрации для решения задачи рекомендаций: классическая LightFM и нейросетевая SASRec. Для этого был проведен сбор и подготовка аналогичных данных для обеих моделей, обучение и подбор гиперпараметров. В качестве дополнительного подтверждения качества работы нейросетевой модели собран датасет на основе эмбеддингов модели и обучен простой классификатор.

Проведен сравнительный анализ различных подходов к решению данной задачи и выбраны наилучшие алгоритмы-представители классического ML и глубокого обучения с использованием нейросетей. По результатам сравнения в абсолютно равных условиях нейросетевая модель SASRec показала метрики качества примерно в 3 раза выше, чем классическая LightFM. Обе модели были обучены с подобранными гиперпараметрами до максимально возможного качества на имеющемся наборе данных. Дополнительно качество работы нейросетевой модели показывает проведенный эксперимент по кластеризации эмбеддингов пользователей по предпочтениям “аренды–продажи”.

Результат может быть применен для улучшения качества работы рекомендательных алгоритмов в тех компаниях, где до сих пор используются классические подходы. На основе полученных метрик можно рассчитать потенциально возможную прибыль от продаж после введения нового алгоритма для конкретных случаев, чтобы понимать, будет ли внедрение нейросетевых моделей компенсировать затраты на оборудование и обслуживание таких систем.

СПИСОК ЛИТЕРАТУРЫ

1. *Kang W.-Ch., McAuley J.* Self-Attentive Sequential Recommendation // IEEE Intern. Conf. on Data Mining (ICDM). Singapore, 2018. P. 197–206.

2. Имплементация модели SASRec на Python // GitHub. Microsoft recommenders : webcite https://github.com/microsoft/recommenders/tree/main/recommenders/models/sasrec (accessed: 22.04.2023).
3. *Kula M.* Metadata Embeddings for User and Item Cold-start Recommendations // Proc. 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conf. on Recommender Systems (RecSys). Vienna, Austria, 2015. P. 14–21.
4. LightFM Documentation // LYST's Engineering Blog : webcite https://making.lyst.com/lightfm/docs/home.html (accessed: 20.04.2023).
5. *Sineva I.S., Denisov V.Y., Galinova V.D.* Building Recommender System for Media with High Content Update Rate // IEEE Intern. Conf. Quality Management, Transport and Information Security, Information Technologies. St. Petersburg, Russia, 2018. P. 385.
6. *Sudasinghe P. G.* Enhancing Book Recommendation with the use of Reviews : Master's Degree Programmes. Colombo, 2019. 54 p.
7. *Manotumruksa J., Yilmaz E.* Sequential-Based Adversarial Optimisation for Personalised Top-N Item Recommendation // Proc. 43rd Intern. ACM SIGIR Conf. on Research and Development in Information Retrieval. Xi'an, China, 2020. P. 2045–2048.
8. *Tenney I., Das D., Pavlick E.* BERT Rediscovers the Classical NLP Pipeline // Proc. 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy, 2019. P. 4593–4601.
9. *Bi J., Zhu Z., Meng Q.* Transformer in Computer Vision // IEEE Intern. Conf. on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI). Fuzhou, China, 2021. P. 178–188.
10. *Han K., Wang Y., Chen H., Chen X., Guo J., Liu Z., Tang Y., Xiao A., Xu C., Xu Y., Yang Z., Zhang Y., Tao D.* A Survey on Vision Transformer // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022. V. 45. № 1. P. 87–110.
11. *Schafer J.B., Frankowski D., Herlocker J., Sen Sh.* Collaborative Filtering Recommender Systems // The Adaptive Web: Methods and Strategies of Web Personalization. Berlin, Germany, 2007. P. 291–324.
12. *Rosenthal E.* Learning to Rank Sketchfab Models with LightFM https://www.ethanrosenthal.com/2016/11/07/implicit-mf-part-2/ (accessed: 20.04.2023).
13. *Patoulia A.A., Kiourtis A., Mavrogiorgou A., Kyriazis D.* A Comparative Study of Collaborative Filtering in Product Recommendation // Emerging Science Journal. 2023. V. 7. № 1. 15 p.
14. *Polignano M., de Gemmis M., Semeraro G.* Comparing Transformer-based NER approaches for analysing textual medical diagnoses // Proc. Working Notes of CLEF. Bucharest, Romania, 2021. V. 2936. P. 818–833.