Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Прикладной Математики и Информатики
Кафедра алгоритмов и технологий программирования

Направление подготовки / специальность: 09.04.01 Информатика и вычислительная техника
Направленность (профиль) подготовки: Анализ данных и разработка информационных систем

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ, ОСНОВАННЫЕ НА ГРАФОВЫХ НЕЙРОННЫХ СЕТЯХ

(магистерская диссертация)

Студент:
Минеев Никита Русланович

_____
(подпись студента)

Научный руководитель:
Киселёв Дмитрий Андреевич,

_____
(подпись научного руководителя)

Консультант *(при наличии)***:**
_____
_____
(подпись консультанта)

Москва 2023

Аннотация

Работа посвящена исследованию графовых методов решения задачи рекомендации онлайн. Задача рекомендации сегодня возникает во многих сферах человеческой жизни и потому качественное её решение крайне востребовано. Решение задачи рекомендации в онлайн постановке с применением инструментария обучения с подкреплением обладает большим количеством преимуществ перед классической постановкой, в их числе обучение модели во время эксплуатации, гибкая функция награды, позволяющая во время обучения учитывать дополнительные целевые метрики и так далее. А использование графовых нейронных сетей для извлечения информации из истории взаимодействий пользователей с товарами, имеющей графовую структуру, позволяет основанным на них рекоммендерам одновременно эксплуатировать два мощных подхода к решению задачи рекомендации, что положительно отражается на качестве.

В рамках работы был проведен анализ существующего графового метода решения задачи рекомендации онлайн --- Graph Convolutional Q-Network, найдены его недостатки, в том числе, дискретный учет динамики графа взаимодействий пользователей и товаров, который потенциально может приводить к потере важной информации, и, соответственно, снижать качество решения задачи. Была предпринята попытка устранения данного недостатка, путем внедрения, вместо используемого в GCQN для извлечения графовой информации фреймворка Graph Convolutional Recurrent Network, более совершенного, способного учитывать динамику графа непрерывно, фреймворка Temporal Graph Network. В результате была получена новая усовершенствованная модель Temporal Graph Q-Network.

Temporal Graph Q-Network обладает всеми преимуществами рекомендательных систем, основанных на графовых нейронных сетях, при этом учитывающая динамику графа взаимодействий пользователей и товаров непрерывно.

В результате численного эксперимента, было показано, что предложенная модель превосходит в метрике качества Average Reward в решении сессионной задачи рекомендации онлайн, как графовые аналоги, так и некоторые стандартные для данной задачи модели, что говорит о состоятельности метода и возможности его успешного применения на практике.

Содержание

Обозначения и сокращения

MLP — многослойный перцептрон [5]

RNN — рекуррентная нейронная сеть [6]

LSTM — долгая-краткосрочная память [7]

GRU — управляемый рекуррентный блок. [8]

Глава 1

Введение

Актуальность. Задача рекомендации возникает во многих сферах человеческой жизни, от комерции и развлечений[27], до образования[25] и медицины[26], соответственно, высокого качества решение данной задачи позволяет улучшить моральное и физическое состояние людей, повысить их интелектуальное развитие и образование, углубить проффессиональные навыки, что в масштабах, например, государства приведет экономическому росту, национальной самообеспеченности и общественному развитию.

Рекомендательные системы - это особый тип систем фильтрации информации, целью которых является решение задачи рекомендации путем прогнозирования предпочтений пользователя в отношении того или иного товара/услуги[24]. В основе прогнозирования лежат различные способы обработки истории взаимодействий пользователей с товарами, порождающие различные подходы к решению задачи, из которых наиболее результативными являются контентый и коллаборативная фильтрация. Контентный подход предполагает, что пользователю понравится товар, если он похож на понравившиеся данному пользователю товары ранее[28]. Коллаборативная фильтрация рассматривает процессы взаимодействий шире, включая, для прогнозирования предпочтений пользвателя, информацию о других похожих пользователях, предполагая, основываясь на схожести, что понравившиеся им товары понравятся и целевому пользователю[29]. Так как система взаимодействий пользователей и товаров имеет графовую структуру, для извлечение информации возможно привлечение популярного сегодня и быстро развивающего в

последние годы инструментария графовых нейронных сетей. Это позволяет рекомендательным системам эксплуатировать одновременно оба наиболее успешных подхода к решению задачи рекомендации, что выражается в повышении качества решения[30].

Для решения задачи рекомендации необходимо обучить модели успешно справляться с динамично развивающимися интересами и предпочтениями пользователей. Помимо этого рекомендательные системы способны непосредственно на данную динамику влиять, что приводит к изменениям самих таргетов для обучения и делает задачу в разы труднее в случае решения её в стандартной постановке обучения с учителем и приводит к решению в форме обучения с подкреплением. Обучение с подкреплением в случае задачи рекомендации имеет ряд преимуществ, например, возможность непрерывного обучения моделей в процессе их эксплуатации, оптимизация различных дополнительных целевых функций с помощью изменения функции награды и так далее[31].

Цели и задачи работы. В данной работе проведен анализ существующего графового метода для решения сессионной задачи рекомендации онлайн — Graph Convolutional Q-Network (GCQN)[32], основанного на Graph Convolutional Recurrent Network (GCRN)[11] фреймворке, найдены недостатки, в том числе, дискретный учет динамики графа взаимодействий пользователей и товаров, что может приводить к потере информации и снижению качества решения задачи рекомендации.

Целью работы является разработка нового метода, устраняюшего недостатки предыдущего, добавление непрерывного учета динамики графа взаимодействий пользователей и товаров, с помощью внедрения вместо, лежащего в основе предыдущего метода, Graph Convolutional Recurrent Network более совершенного Temporal Graph Network (TGN)[12] и проверка изменения качества решения сессионной задачи рекомендации онлайн. Для достжения поставленной цели необходимо решить следующие задачи:

1. Построить модель, основанную на TGN, для решения задачи рекомендации онлайн.

2. Провести численные эксперименты

3. Провести анализ результатов экспериментов, сравнить качество решения задачи рекомендации онлайн модели GCQN и предложенной модели.

Глава 2

Основные понятия и определения

В данной главе определяются все необходимые понятия, описываются необходимые концепции, используемые повсеместно в работе и требующиеся для её полноценного понимания.

2.1 Общая постановка задачи рекомендаций и методы решения

Постановка задачи рекомендаций. Задано множество пользователей $U = \{u_1, u_2, \ldots, u_m\}$ и множество товаров $I = \{i_1, i_2, \ldots, i_n\}$, и существует целевая функция $r : U \times I \times \mathcal{T} \to R$, выражающая результат взаимодействия пользователей с товарами, где R — линейно упорядоченное множество возможных результатов взаимодействия, $\mathcal{T}$ — время.[24]

Для краткости записи, введем следующие обозначения:
$r_{ui}^t = r(u, i, t) \in R$,
$r^t = r : \{t\} \times U \times I \to R$, где t — константа,
$r_{ui} = r : U \times I \to R$ — отображение не зависит от $t \in \mathcal{T}$.

Задача рекомендации K товаров: По известной до момента времени T $\in \mathcal{T}$ истории взаимодействий пользователей с товарами $\{u, i, t, r_{ui}^t\}_{u \in U,\, i \in I}^{t<T}$, необходимо

1. для момента времени T построить аппроксимацию целевой функции $\hat{r}^T : U \times I \to R$;

2. для каждого пользователя отранжировать товары по убыванию аппроксимированного результата взаимодействия в момент времени T, взять первые K товаров в качестве рекоммендованных.

Для простоты, здесь и далее будем предполагать K = 1, а $R = \{0, 1\}$, где результат взаимодейтсвия 1 будет интерпретироваться как положительный (понравилось, лайк, клик, покупка, итд), а 0 как отрицательный (не понравилось, дизлайк, отказ от клика/покупки итд). Все приведенные в работе рассуждения несложно обобщаются на случай большего K и большей мощности множества R, тем не менее оставим данное обобщение для будущих исследований.

Для решения задачи рекомендации применяют два основных подхода: контентный и коллаборативная фильтрация.

Контентный подход решения задачи рекомендации. В контентном подходе, пользователю рекоммендуются товары, похожие на понравившиеся ему ранее.

Пусть для каждого товара $i \in I$ имеется его признаковое описание $h_i \in \mathbb{R}^n$. Тогда для решения задачи рекомендации задаются функция аггрегации последовательности признаковых описаний товаров произвольной длины l, $agg : (\mathbb{R}^n)^l \to \mathbb{R}$ и, в пространстве признаковых описаний товаров, метрика $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$.
На первом шаге с помощью функции аггрегации строится профиль пользователя как аггрегация признаковых описаний товаров, с которыми данный пользователь провзаимодействовал $h_u = agg(\{h_i\})$.
На следующем шаге в качестве рекомендации пользователю выбирается товар, признаковое описание которого к профилю пользователя оказалось ближайшим: $i_{rec} = \arg\min_i d(h_u, h_i)$.[34]

Основное преимущество контентного подхода в поддержке холодного старта для товаров и несложной интерпретируемости, объяснимости

рекомендаций. *Недостаток* — оступствие новизны: рекомендации основываются на похожих товарах.

Метод коллаборативной фильтрации для решения задачи рекомендации. В случае коллаборативной фильтрации, пользователю рекоммендуются товары, понравившиеся похожим на данного пользователя пользователям.

Пусть для каждого товара $i \in I$ и пользователя $u \in U$ имеется признаковое описание, соответственно, $h_i \in \mathbb{R}^n$ и $h_u \in \mathbb{R}^n$. Тогда для решения задачи рекомендации определяется понятие окрестности пользователя $S(u)$ в пространстве U, а также задаются функция скоринга $score : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, преобразующая пару признаковых описаний пользователя и товара в действительный число — оценку, характеризующую привлекательность пользователю данного товара, и функция аггрегации последовательности оценок длины l, $agg : \mathbb{R}^l \to \mathbb{R}$.
На первом шаге с помощью функций скоринга и аггрегации оценок вычисляются приближенные рейтинги пользователя для каждого товара $\hat{r}_{ui} = agg_{v \in S(u)}(score(h_v, h_i))$.
На следующем шаге в качестве рекомендации пользователю выбирается товар, для которого приближенный рейтинг пользователя максимальный $i_{rec} = \arg\max_i \hat{r}_{ui}$. [24]

Основное преимущество коллаборативной фильтрации в более высоком качестве рекомендаций, но проблему холодного старта необходимо решать отдельно.

Графовая структура задачи рекомендации.
Граф — двойка, состоящая из множества вершин V и множества ребер $E \subseteq V \times V$ — пар элементов множества вершин: $G =< V, E >$.
Нейронные сети — метод машинного обучения, позволяющий извлекать информацию из сложноструктурированных данных, подобным человеческому мозгу образом.
Графовые нейронные сети (GNN) — тип нейронных сетей, позволя-

ющих извлекать информацию из графов.[35]

В основе графовых нейронных сетей лежит парадигма **обмена сообщениями (message passing)[36]**:
На нулевой итерации для каждой вершины графа $v \in V$ инициализируется соответствующее векторное представление $x_v^0 \in \mathbb{R}^n$.
На каждой последующей итерации $t+1$:

1. с помощью функции сообщений msg для каждого ребра графа $e = (u, v) \in E$ вычисляется сообщение: $m_e^{t+1} = msg(x_u^t, x_v^t, w_e^t)$

2. для каждой вершины графа $v \in V$ вычисляется обновленное векторное представление с помощью функции аггрегации сообщений от ребер agg, инцидентных данной вершине, и функции обновления upd: $x_v^{t+1} = upd(x_v^t, agg(\{m_e^{t+1} : (u, v) \in E\}))$

Таким образом, по окончании k-итерационной процедуры векторные представления вершин содержат информацию о своих соседях до kго порядка. Часто, итерации называют слоями, а саму графовую нейронную сеть, как следствие, k-слойной.

В силу того, что процесс взаимодействия пользователей и товаров имеет графовую структуру, а именно, представим в виде двудольного графа, который будем называть **графом взаимодействий пользователей и товаров**, где вершины — пользователи и товары, а ребра — взаимодействия: $G =< U \cup I, r >$, для решения задачи рекомендации возможно применение графовых нейронных сетей следующим образом:

1. Необходимо с помощью графовой нейронной сети для графа взаимодействий пользователей и товаров вычислить векторные представления вершин, соответственно, векторные представления пользователей и товаров: h_u и h_i.

2. С помощью скоринговой функции $score$ вычислить приближенные рейтинги пользователя для каждого товара $\hat{r}_{ui} = score(h_v, h_i)$.

3. в качестве рекомендации пользователю выбирается товар, для которого приближенный рейтинг пользователя максимальный $i_{rec} =$

$\arg\max_i \hat{r}_{ui}$

Таким образом, вычисленные с помощью графовых нейронных сетей на первом шаге, векторные представления пользователей, благодаря парадигме обмена сообщений, лежащей в основе GNN, будут содержать информацию как о товарах, с которыми провзаимодействовал данный пользователь, так и о других пользователях, которые также провзаимодействовали с данными товарами. То же и для товаров, их векторные представления будут содержать информацию как о пользователях, которые провзаимодействовали с этим товаром, так и о других товарах, с которыми данные пользователи также провзаимодействовали.

Соответственно, *основное преимущество* решений задачи рекомендации, основанных на графовых нейронных сетях, является применение одновременно обоих подходов, контентного и коллаборативной фильтрации, благодаря лежащей в основе графовых нейронных сетей парадигме обмена сообщений.

Важной особенностью графа взаимодействий пользователей и товаров является зависимость его структуры от времени. Между пользователями и товарами возникают новые взаимодействия, количество пользователей и товарой может расти и убывать, что в структуре графа выражается в виде добавления/удаления ребер и вершин. Исходя из этого, для достижения выского качества решения задачи рекомендаций необходимо не только уметь извлекать информацию из графовой стркутуры, но и, для случая динамического графа, учитывать её динамику.

2.2 Обучение представлений статических графов

В случае графов, структура которых статична, для вычисления векторных представлений вершин используют статичные графовые нейронные сети [1, 44, 3, 10]. В их основе лежит описанная в предыдущей секции парадигма обмена сообщений[36], а различия, в основном, заключены в используемых функциях сообщений, агрегации и обновления. Рассмотрим статические графовые нейронные сети, используемые в данной ра-

боте.

Graph Convolutional Network [1]. Для аггрегации информации от соседей вершины и итеративного обновления векторных представлений использует спектральное разложение Лапласиана графа:

$$m_{vu} = d_{vv}^{-\frac{1}{2}} a_{vu} d_{uu}^{-\frac{1}{2}} h_u^l,$$

$$h_v^{l+1} = \sigma\left(W^l \sum_{u \in N(v)} m_{vu}\right),$$

где $\sigma(\cdot)$ — нелинейная функция активации, например, relu, W^l — обучаемая матрица преобразования для слоя l, a_{vu} — вес смежности ($a_{vv} = 1$) и $d_{vv} = \sum_k a_{vk}$

GraphSAGE [3]. Авторами было предложено обобщение стандартных функций аггрегации в графовых сверточных сетях. В качестве аггрегатора можно использовать, например, mean/sum/max - pooling'и или рекуррентную нейронную сеть, вроде LSTM. Так же, в GraphSAGE аггрегация происходит не по всем соседям вершины, а по их случайному подмножеству фиксированного размера

$$m_{vu} = h_u^l,$$

$$n_v^l = Aggregator_l(\{h_u^l, \forall u \in N(v)\})$$

$$h_v^{l+1} = \sigma\left(W^l \cdot [h_v^l || n_v^l]\right),$$

где $Aggregator_l$ функция аггрегации на l-ом слое, $\sigma(\cdot)$ нелинейная функция активации, W^l Обучаемая матрица преобразования и $||$ операция конкатенации.

Graph Attention Netowrk[44]. Делается предположение, что влияние каждого соседа может быть не одинаковым и информацию каждого соседа необходимо взвешивать. В таком случае, отличным решением является механизм внимания, где оценка внимания для каждого соседа бу-

дет характеризовать количество полезной информации, которое данный сосед может передать в векторное представление целевой вершины

$$m_{vu} = h_u^l,$$

$$n_v^l = \sum_{u \in N(v))} \alpha_{vu} h_u^l$$

$$\alpha_{vu} = \frac{\exp(Att(h_v^l, h_u^l))}{\sum_{k \in N(v)} \exp(Att(h_v^l, h_k^l))}$$

$$h_v^{l+1} = \sigma\left(W^l n_v^l\right),$$

где $Att(\cdot)$ функция внимания, обычно в качестве функции $Att(\cdot)$ берут $LeakyReLU(a^T[W^l h_v^l || W^l h_u^l])$, $\sigma(\cdot)$ нелинейная функция активации, W^l Обучаемая матрица преобразования, a^T Обучаемый вектор и $||$ операция конкатенации.

2.3 Обучение представлений динамических графов

Динамический граф — граф, структура которого зависит от времени: $G(t) =< V(t), E(t) >$,($t \in \mathcal{T}$ — время), то есть граф, в котором могут добавляться/удаляться вершины и ребра в различные моменты времени. Тогда для извлечения информации из графа такого рода, построения векторных представлений вершин, учитывающих динамику изменений, требуется определить способ представления данного графа и, в соответствии ему, фреймворк для извлечения информации.[12]

Существует два основных способа представления динамических графов: динамический граф дискретного времени (Discrete-time dynamic graphs, DTDG) и динамический граф непрерывного времени (Continious-time dynamic graphs, CTDG). Рассмотрим каждый из способов представления и соответствующие фреймворки для извлечения информации и построения векторных представлений вершин подробнее.

Динамический граф дискретного времени. Пусть задана последовательность моментов времени $\{t_0, \ldots, t_n\}$. Динамический граф дискретного времени представляет динамический граф в виде последовательности статичных графов, называемых снэпшотами, полученных с помощью фиксации динамического графа в соответствующий момент времени $G(t) = \{G^{t_0}, \ldots, G^{t_n}\}, t_n \leqslant t$.

Для построения векторных представлений вершин динамического графа дискретного времени, учитывающих динамику, чаще всего используют различные вариации Graph Convolutional Recurrent Network (GCRN)-фреймворка[11].

Работу GCRN можно разбить на два основных этапа:

1. Для каждой вершины i на каждом снэпшоте G^τ вычисляем статичное графовое векторное представление h_i^τ вершины i, путем применения некоторой статичной графовой нейронной сети.

2. Аггрегируя последовательность статичных векторных представлений $\{h_i^{t_0}, \ldots, h_i^{t_n}\}$, с помощью некоторой функции аггрегации $agg : (\mathbb{R}^d)^{n+1} \to \mathbb{R}^d$, получают итоговое векторное представление h_i вершины i, учитывающее динамику графа.

В качестве аггрегатора используются mean/max-пулинги, свертки [17, 19, 18], но чаще, как рассматривается и в данной работе, используют в качестве аггрегатора рекурентную нейронную сеть[11, 13, 16, 15], например, GRU, LSTM.

Основным недостатком представления динамического графа в виде динамического графа дискретного времени и использование соответствующих фреймворков для построения векторных представлений вершин данного графа, является возможная потеря информации при дискретизации времени и получении снэпшотов, что приводит к отсутствию прямого учета порядка и времени появления/удаления ребер и вершин.

Динамический граф непрерывного времени. Пусть $x(t)$ событие на динамическом графе G в момент времени t. Событие $x(t)$ может быть двух типов: 1) $x(t) = x_{vu}(t)$ — добавление/удаление ребра между вершинами v и u, 2) $x(t) = x_v(t)$ — добавление/удаление вершины v. Тогда динамический граф непрерывного времени представляет динамический граф в виде последовательности событий $G(t) = \{x(t_1), x(t_2), \ldots, x(t_n)\}, t_n \leqslant t$.[12]

Для построения векторных представлений вершин динамического графа непрерывного времени, Rossi et al. представили Temporal Graph Network (TGN) фреймворк[12], обобщающий предыдущие модели для решения данной задачи: JODIE[22], DyRep[20], TGAT[21].

TGN фреймворк состоит из пяти модулей: Memory, Message Function, Message Aggregator, Memory Updater, Embedding.

1. Memory модуль содержит векторное представление каждой вершины, называемое, вектором памяти, которое обновляется каждый раз, когда на графе происходит событие, в котором данная вершина участвует, что позволяет в векторе памяти хранить долговременную аггрегированную информацию об истории событий с данной вершиной.

   Тем-не-менее, такое представление может приводить к проблеме «застаревания памяти»[23], для решения которой TGN имеет embedding-модуль.

2. Message Function — функция, которая для каждой вершины участвующей в событии вычисляет вектор сообщения, используемый для обновления вектора памяти.

3. Message Aggregator — функция, аггрегирующая несколько сообщений в одно. Так как для более эффективного обучения модели события могут быть объединены в батчи, возможна ситуация, когда в один батч попали несколько событий, в которых участвует одна и та же вершина, соответственно, для этой вершины получится

несколько векторов сообщений, которые с помощью данного модуля аггрегируются в одно сообщений.

4. Memory Updater — функция, обновляет вектор памяти вершины на основании полученного от нового события, в котором участвует данная вершина, вектора сообщения.

5. Embedding модуль вычисляет итоговые векторные представления для каждой вершины, учитывающие динамику графа.

Основное преимущество представления динамического графа в виде динамического графа непрерывного времени и использование соответствующих фреймворков для построения векторных представлений вершин данного графа, является устранение недостатка динамического графа дискретного времени в возможной потери информации при дискретизации времени, что приводит к более высокому качеству решения различных задач, где необходима обработка динамических графов.

Глава 3

Графовые модели для решения задачи рекомендаций онлайн

Для решения задачи рекомендации необходимо обучить модели успешно справляться с динамично развивающимися интересами и предпочтениями пользователей. Помимо этого рекомендательные системы способны непосредственно на данную динамику влиять, что приводит к изменениям самих таргетов для обучения и делает задачу в разы труднее в случае решения её в стандартной постановке обучения с учителем и приводит к решению в форме обучения с подкреплением.

Преимуществом решения задачи рекомендации в форме обучения с подкреплением является, например, возможность обучения моделей в процессе их эксплуатации, в то же время к недостаткам можно отнести сложность построения хорошей искусственной среды для обучения моделей.

3.1 Постановка задачи рекомендаций онлайн

Постановка сессионной задачи рекомендации онлайн. Пусть имеется множество пользователей $U = \{u_1, u_2, \ldots, u_m\}$, множество товаров $I = \{i_1, i_2, \ldots, i_n\}$ и существует функция $r : U \times I \times \mathcal{T} \to \{0, 1\}$,

выражающая результат взаимодействия пользователей с товарами так, что $r(u, i, t) = 0$ интерпретируется как «пользователь u в момент времени t не удовлетворен товаром i» (в то же время $r(u, i, t) = 1$ интерпретируется как «пользователь u в момент времени t удовлетворен товаром i»), где $\mathcal{T}$ — время[31].
В случае сессионной постановки задачи рекомендации[32], когда $\mathcal{T} = \{0, 1, \ldots, T-1\}$, агент, решающий задачу рекомендации, взаимодействует с пользователем в течении T временных шагов. Для каждого момента времени t:

1. агент наблюдает состояние s_t пользователя u, которое представляется в виде последовательности товаров, с которыми провзаимодейтсвовал пользователь u до момента времени t: $s_t = \{a_0, a_1, \ldots, a_{t-1}\}$

2. на основании своей политики $\pi : I^t \to I$ агент выбирает действие $a_t \in I$, то есть следующий рекомендуемый пользователю товар.

3. Далее, в момент времени $t + 1$, в соответствии с выбранным действием a_t, агент получает награду $reward_t = r_t = r^t_{ui} = r(u, i, t)$ от пользователя u и наблюдает новое состояние $s_{t+1} = \{a_0, a_1, \ldots, a_t\}$ пользователя u.

Для оценки качества решения задачи онлайн рекомендации в сессионной постановке будем использовать **среднюю награду за сессию (Average Reward)**: $AverageReward = \frac{1}{T}\sum_t r_t$.

Таким образом, **сессионная задача рекомендации онлайн** формулируется следующим образом:
Пусть дано множество пользователей для обучения $U_{train} \subseteq U$. Необходимо по взаимодейтсвиям с пользователями из U_{train} построить политику π, которая для любого нового пользователя $u \in U_{test} \subseteq U$ максимизирует среднюю награду за сессию $AverageReward \to \max$.

Для решения задачи будем использовать **метод Q-обучения[37]**:
Выразим политику через Q-функцию $\pi(s) = \arg\max Q^*(s, a)$. Тогда для

решения задачи, необходимо построить аппроксимацию оптимальной Q-функции с помощью модели машинного обучения $Q^*(s,a) \approx Q_\theta(s,a)$, где θ – параметры модели.

3.2 Graph Convolutional Q-Network

В 2020 году была предложена графовая модель для решения сессионной задачи рекомендации онлайн. Векторное представление состояния пользователя вычисляется с помощью модели, основанной на GCRN-фреймворке, а именно состояние пользователя u, представляемое в виде последовательности товаров, с которыми данный пользователь провзаимодействовал, преобразуется в последовательность графовых векторных представлений, полученных с помощью Graph Attention Netowork (GAT), примененной к графу взаимодействий пользователей и товаров, после чего данная последовательность аггрегируется с помощью рекуррентной нейронной сети Gated Recurrent Network (GRU), вычисляя таким образом векторное представление текущего состояния пользователя, учитывающее динамику графа взаимодействий пользователей и товаров. Далее рассматривается архитектура модели подробнее.

Слой векторных представлений. Отобразим каждого пользователя u и товар i в векторное пространство размерности d, таким образом получим, соответственно, векторные представления для пользователя $e_u \in \mathbb{R}^d$ и товара $e_i \in \mathbb{R}^d$. Данные векторные представления будем рассматривать как признаки вершин в графе, они будут случайно проициализированы в начале процесса обучения и обновляться вместе с другими параметрами модели.

GCN-слой. Состояние $s_t = \{a_0, a_1, \ldots, a_{t-1}\}$ пользователя u представим в виде динамического графа дискретного времени $G = \{G^0, G^1, \ldots, G^{t-1}\}$, где G^τ — снэпшот графа взаимодействий пользователей и товаров, взятый в момент времени τ. Тогда для вычисления векторного представления h_{s_t} состояния пользователя u, учитывающего структуру графа взаимодействий пользователей и товаров и его динамику, используем GCRN

фреймворк, а векторные представления действий h_a вычислим, применив статичную графовую нейронную сеть к последнему снэпшоту G^{t-1}.

На первом этапе GCRN к каждому снэпшоту применяется статичная графовая нейронная сеть, в случае GCQN применяется GAT. Для каждого товара i найдем его векторное графовое представление $x_i \in \mathbb{R}^d$:

$$x_i = relu(W_{fc}[e_i||e_{N(i)}] + b_{fc}),$$

где $W_{fc} \in \mathbb{R}^{d \times 2d}$ и $b_{fc} \in \mathbb{R}^d$ обучаемые веса и смещения полносвязного слоя, e_i векторное представление товара i, $||$ операция конкатенации, и $e_{N(i)} \in \mathbb{R}^d$ векторное представление соседей товара i:

$$e_{N(i)} = \sum_{w \in N(i)} \alpha_{iw} e_w,$$

где $N(i)$ множество соседей первого порядка товара i в соответствующем снэпшоте графа взаимодействий пользователей и товаров G^τ, e_w векторное представление пользователя w, α_{iw} оценка внимания (attention score), определяющая количество информации, которое будет передано векторному представлению соседей товара i от соседа w и вычисляющаяся следующим образом:

$$\alpha_{iw} = \frac{\exp(w_a^T \tanh(W_a[e_i||e_w]))}{\sum_{v \in N(i)} \exp(w_a^T \tanh(W_a[e_i||e_v]))},$$

где $W_a \in \mathbb{R}^{d \times 2d}$ и $w_a \in \mathbb{R}^d$ обучаемые веса механизма внимания, $\cdot^T$ операция транспонирования.

Таким образом, для каждого товара $a_\tau \in s_t$, берется соответствующий снэпшот графа G_τ, для которого применяется процедура описанная выше и вычисляется векторное графовое представление x_{a_τ} товара a_τ, а вся последовательность целиком $s_t = \{a_0, a_1, \ldots, a_{t-1}\}$ преобразуется в последовательность $x_{s_t} = \{x_{a_0}, x_{a_1}, \ldots, x_{a_{t-1}}\}$.

В качестве векторного представления h_{a_i} действия $a_i \in I$ используется векторное графовое представление $x_i \in \mathbb{R}^d$ товара i, полученное

применением описанноей выше процедуры к последнему снэпшоту G^t.

GRU-слой. На втором этапе GCRN, к последовательности графовых векторных представлений применяют операцию аггрегации, в случае GCQN применяют GRU, для учета динамики графа.

Последовательность $x_{s_t} = \{x_{a_0}, x_{a_1}, \ldots, x_{a_{t-1}}\}$ преобразуют в последовательность $\{h_0, h_1, \ldots h_{t-1}\}$, где $h_j \in \mathbb{R}^d$ скрытое состояние GRU на шаге j: $h_j = GRU(h_{j-1}, x_{a_j})$. Последовательность скрытых состояний, в свою очередь, аггрегируется с помощью механизма внимания, таким образом получаем векторное представление h_{s_t} состояния s_t:

$$h_{s_t} = \sum_{j=0}^{t-1} \beta_j h_j,$$

где β_j оценки внимания, определяющия количество информации, которое будет передано векторному представлению состояния h_{s_t} от скрытого состояния h_j и вычисляющиеся следующим образом:

$$\beta_j = \frac{\exp(w_{sa}^T \tanh(W_{sa} h_j))}{\sum_{l=0}^{t-1} \exp(w_{sa}^T \tanh(W_{sa} h_l))},$$

где $W_{sa} \in \mathbb{R}^{d \times d}$ и $w_a \in \mathbb{R}^d$ обучаемые веса механизма внимания.

MLP-слой. Для вычисления итогового приближенного значения Q-функции используем многослойный перцептрон (MLP), который принимает на вход конкатенацию векторных представлений состояния h_{s_t} и действия h_{a_i} и возвращает приближенное значение $Q_\theta(s_t, a_i) \approx Q^*(s_t, a_i)$.

Благодаря тому, что в основе GCQN лежит GCRN фреймворк, модель обладает всеми *преимуществами* решений задачи рекомендации, основанных на графовых нейронных сетях. Но в модели существует *недостаток*, внесенный GCRN, а именно, дискретный учет динамики графа взаимодействий пользователей и товаров, из-за чего, во время разбиения графа на снэпшоты, возможна потеря важной информации, что приводит, в конечном итоге, к потере качества решения задачи рекомендации.

В предлагаемой далее модели предпринята попытка данный недостаток устранить.

3.3 Предлагаемая модель

Как было заключено в предыдущей секции, в основе модели GCQN для решения сессионной задачи рекомендации онлайн, лежит графовая модель GCRN, учитывающая динамику графа взаимодействий пользователей и товаров дискретно, что потенциально может приводить к потере информации во время дискретизации и снижению качества решения задачи рекомендации. Одним из возможных способов устранить данный недостаток является внедрение в модель TGN фреймворка, учитывающего динамику графа непрерывно, вместо GCRN. Таким образом была получена новая усовершенствованная модель — TGQN. Далее рассмотрим архитектуру модели подробнее.

Слой векторных представлений. Отобразим каждого пользователя u и товар i в векторное пространство размерности d, таким образом получим, соответственно, векторные представления для пользователя $e_u \in \mathbb{R}^d$ и товара $e_i \in \mathbb{R}^d$. Данные векторные представления будем рассматривать как признаки вершин в графе, они будут случайно проициализированы в начале процесса обучения и обновляться вместе с другими параметрами модели.

TGN-слой. Состояние $s_t = \{a_0, a_1, \ldots, a_{t-1}\}$ пользователя u представим в виде динамического графа непрерывного времени, то есть в виде последовательности событий $G = \{x_{u_0,i_0}(t_0), x_{u_1,i_1}(t_1), \ldots, x_{u_n,i_n}(t_n)\}$, где $x_{u_k,i_k}(t_k)$ — событие взаимодействия в момент времени t_k между пользователем u_k, возможно отличным от пользователя u, для состояния s_t которого мы ищем векторное представление, и товаром i_k. Тогда для вычисления векторных представлений h_{s_t} состояния пользователя u и действий h_a, учитывающих структуру графа взаимодействий пользователей и товаров и его динамику, используем TGN фреймворк.

TGN фреймворк состоит из пяти модулей: Memory, Message Function, Message Aggregator, Memory Updater, Embedding. Рассмотрим работу каждого модуля по отдельности.

Memory-модуль. Для каждой вершины v графа взаимодействий пользователей и товаров в момент времени t memory-модуль содержит вектор памяти $s_v(t)$. Вектор памяти обновляется после каждого события на динамическом графе, таким образом, сохраняя в сжатом виде историю о событиях, в которых участвовала данная вершина v. В начале обучения все векторы памяти инициализируются нулями.

Message Function. Функция сообщений msg вычисляет сообщение для каждой вершины u и i, участвующей в событии $x_{u,i}(t)$, как конкатенацию их признаков $e_u \in \mathbb{R}^d$ и $e_i \in \mathbb{R}^d$ и векторного представления разности текущего момента времени t и момента времени t^- последнего события, в котором участвовала соответствующая вершина

$$m_u(t) = [e_u||e_i||\varphi(t - t_u^-)],$$

$$m_i(t) = [e_u||e_i||\varphi(t - t_i^-)],$$

где $||$ — операция конкатенации, $\varphi(\cdot)$ — функция, преобразующая разность моментов времени в вектор.

Message Aggregator. Так как в одном батче событий одна и та же вершина v может встречаться несколько раз, соответственно, для неё будет несколько сообщений, которые необходимо аггрегировать с помощью agg функции. В качестве agg функции будем использовать max-pooling по времени, то есть просто вытаскивать самое последнее сообщение

$$\hat{m}_v(t) = \max_t(m_v(t_1), \ldots, m_v(t_k))$$

Memory Updater. Данный модуль обновляет вектор памяти для вершины v, учавствующей в событии, на основе её аггрегированного сообщения $\hat{m}_v(t)$ и старого вектора памяти $s_v(t^-)$. В качестве memory updater'а

будем использовать Gated Recurrent Network (GRU)

$$s_v(t) = GRU(\hat{m}_v(t), s_v(t^-))$$

Embedding-модуль. Для решения проблемы «застаревания памяти» (memory staleness problem) $s_v(t)$ используется дополнительный модуль, вычисляющий итоговое векторное представление $h_v(t) \in \mathbb{R}^d$ вершины графа v. В качестве embedding-модуля используем двуслойный Temporal Graph Attention: на вход каждый слой принимает представление $h_v^{l-1}(t)$ вершины v с предыдущего слоя, текущий момент времени t, представления соседей вершины v $\{h_1^{l-1}(t), \ldots, h_N^{l-1}(t)\}$ вместе с моментами времени $t_1, \ldots, t_N$

$$h_v^0(t) = [s_v(t)||e_v]$$

$$C^l(t) = [h_1^{l-1}(t)||\varphi(t - t_1), \ldots, h_N^{l-1}(t)||\varphi(t - t_N)]$$

$$K^l(t) = V^l(t) = C^l(t)$$

$$q^l(t) = h_v^{l-1}||\varphi(0)$$

$$\hat{h}_i^l(t) = MultiHeadAttention^l(q^l(t), K^l(t), V^l(t))$$

$$h_v^l(t) = MLP^l(h_v^{l-1}(t)||\hat{h}_v^l(t))$$

Здесь $\varphi(\cdot)$ — функция, преобразующая разность моментов времени в вектор, $||$ — операция конкатенации, MLP — многослойный перцептрон.

Таким образом, выходные векторы $h_v^2(t) = h_v(t) \in \mathbb{R}^d$ с последнего слоя Temporal Graph Attention, примененного к графу взаимодействий пользователей и товаров, непрерывно учитывающие структуру графа и его динамику, будем использовать в качестве векторных представлений состояния $s_t = h_u(t)$ пользователя u и, соответственно, действий $h_{a_i} = h_i$ товара i.

MLP-слой. Для вычисления итогового приближенного значения Q-функции используем многослойный перцептрон (MLP), который принимает на вход конкатенацию векторных представлений состояния h_{s_t} и действия h_{a_i} и возвращает приближенное значение $Q_\theta(s_t, a_i) \approx Q^*(s_t, a_i)$.

Благодаря тому, что в основе TGQN лежит TGN фреймворк, модель обладает всеми *преимуществами* решений задачи рекомендации, основанных на графовых нейронных сетях. К тому же, модель *устраняет недостаток* предыдущего аналога GCQN, учитывая динамику графа взаимодействий пользователей и товаров непрерывно, не разбивая граф на снэпшоты, а обрабатывая непрерывный поток событий, что решает проблему потенциальной потери информации при дискретизации и на практике должно привести к приросту качества. Далее будет рассмотрен вычислительный эксперимент, и на основании проведенного сравнения и анализа сделанное выше предположение будет эмпирически проверено.

Глава 4

Вычислительный эксперимент

Для проверки предположения о том, что замена дискретного учета динамики графа взаимодействий пользователей и товаров на непрерывный приводит к росту качества решения задачи рекомендации, а также общей состоятельности предложенного метода относительно некоторых стандартных алгоритмов, был поставлен вычислительный эксперимент. В рамках эксперимента решалась сессионная задача рекомендации онлайн, с длиной сессии $T = 20$ шагов, то есть агент, решающий задачу рекомендации, взаимодействует с пользователем в течении 20 шагов. Метрика качества данной задачи — AverageReward

Для постановки и проведения эксперимента были проделаны следующие действия:

1. Произведена программная реализация рассмотренных графовых, а так же некоторых стандартных моделей.
2. Построена среда, имитирующая поведение пользователей и определена процедура обучения и тестирования моделей.
3. Модели были обучены согласно процедуре обучения.
4. Модели были протестированы, согласно процедуре тестирования, собраны метрики качества для проведения сравнительного анализа.

По окончании эксперимента был проведен сравнительных анализ, сделаны выводы.

4.1 Среда и процедура

Среда. Для проведения вычислительных экспериментов была простроена среда, имитирующая поведение реальных пользователей. В её основу были положены публичные датасеты: MovieLens, Goodreads, Steam. Каждый датасет представляет из себя последовательность взаимодействий пользователей с товарами, соответствующий результат взаимодействия и момент времени. Все результаты взаимодействий были приведены к значениям 0, 1, где 0 — отрицательное взаимодействие и 1 — положительное. Так как длина сессии T = 20 шагов, из датасетов были удалены пользователи имеющие менее 20 товаров с положительным взаимодействием.

MovieLens[39]. Датасет содержит оценки пользователей фильмам, собранные с апреля 2000 по февраль 2003. Оценки были приведены к 0, 1 следующим образом: 0 — оценка < 4, 1 — оценка $\geqslant 4$. Количество пользователей: 5041, количество товаров: 3458.

Goodreads[41]. Оценки пользователей книгам, собранные с Февраля 2007 по Ноябрь 2017. Оценки были приведены к 0, 1 слудующим образом: 0 — оценка $<$ 4, 1: оценка $>=$ 4. Количество пользователей: 5717, количество товаров: 1500

Steam[40]. Отзывы пользователей на игры, собранные с Октября 2010 по Декабрь 2022. В качестве оценки использовался параметр is_recommended: рекомендует ли пользователь данную игру, 0 — не рекомендует, 1 — рекомендует. Количество пользователей: 7008, Количество товаров: 2132

Поскольку в датасетах присутствуют не все возможные взаимодействия, доопределим результаты отсутствующих взаимодействий как отрицательные. Очевидно, что данное положение для некоторых товаров

будет ошибочно, поэтому в предположении, что таких товаров относительно немного, решим эту проблему следующим образом: во время обучения на каждом шаге сессии действие, то есть следующий рекомендуемый пользователю товар, выбирается не из всего множества товаров, а из подмножества, состоящего из объединения множества товаров для которых в датасете имеется реальный результат взаимодействия с данным пользователем и 1000 случайно выбранных товаров, для которых результат взаимодействия был доопределен как отрицательный. Таким образом, если ложно-отрицательно-доопределенных товаров небольшое относительно истинно-отрицательно-доопределенных товаров количество, они редко будут попадать в подмножество, из которого рекоммендер выбирает положительный товар, соответственно, связанная с данным недостатком исскуственной среды ошибка рекоммендера не будет вносить большого смещения в результат обучения.

Процедура. Для каждого датасета пять ряз выполняем следующую процедуру:

1. Разбиваем случайным образом с сидом, отличным от предыдущих, множество пользователей датасета U на тренировочное U_{train}, содержащее 80% пользователей, и тестовое U_{test}, содержащее 20% пользователей.

2. Обучаем модели на тренировочном множестве пользователей в соответствии с алгоритмом

   Вход: U_{train};

   Выход: обученная модель;

   для $session = 1, \ldots, N$
   - Сэмплируем пользователя u из U_{train};
   - Инициализируем состояние случайным товаром $s_0 = \{i_c\}$;
   - **для** $t = 0, \ldots, T-1$
     - выбираем действие a_t: с вероятностью ϵ случайный товар, иначе товар с наибольшим значением $Q_\theta(s_t, a)$;
     - $reward_{t+1} = r(u, a_t, t)$;
     - $s_{t+1} = s_t \cup a_t$;

8: обновляем веса θ модели, используя DQNLoss;

3. Тестируем модели на тестовом множестве пользователей в соответствии с алгоритмом

Вход: U_{test};

Выход: метрика качества AverageReward;

1: $AverageReward := 0$;
2: **для** $u \in U_{test}$
3: Инициализируем состояние случайным товаром $s_0 = \{i_c\}$
4: **для** $t = 0, \ldots, T-1$
5: выбираем действие a_t: товар с наибольшим значением $Q_\theta(s_t, a)$;
6: $reward_{t+1} = r(u, a_t, t)$;
7: $s_{t+1} = s_t \cup a_t$;
8: $AverageReward+ = reward_{t+1}/T/|U_{test}|$;

Собранные во время тестирования значения метрики качества усредняем по количеству выполнений процедуры, получаем итоговое значение метрики качества.

4.2 Бейзлайн-модели

В дополнении к аналогичной предложенной модели GCQN, сравнения также были проведены с некоторыми стандартными алгоритмами решения задачи рекомендации.

Random. Алгоритм на каждом шаге выбирает случайное действие независимо от состояния пользователя.

SVDQ[42]. SVD обученный с помощью Q-обучения в соответствии приведенной выше процедуре, для решения сессионной задачи рекомендации онлайн. Векторные представления состояния и действий вычисляются через обучаемый слои векторных представлений.

GRUQ[43]. GRU4Rec обученный с помощью Q-обучения в соответствии приведенной выше процедуре, для решения сессионной задачи

	MovieLens	Goodreads	Steam
Random	0.077	0.024	0.037
SVDQ	0.155	0.071	0.104
GRUQ	0.283	0.108	0.133
LSTMQ	0.273	0.117	0.155
GCQN	0.300	0.122	0.179
TGQN	**0.359**	**0.134**	**0.232**

Таблица 1. Значение метрики AverageReward для сессионной задачи рекомендации.

рекомендации онлайн. Вычисляет векторное представление состояния пользователя с помощью аггрегации послеодовательности товаров, с которыми пользователь провзаимодействовал, используя в качестве аггрегатора GRU модель. Векторыне представления действий вычисляются через обучаемый слой векторных представлений.

LSTMQ. Модель аналогична GRUQ, но использует LSTM для аггрегации последовательности товаров, с которыми провзаимодействовал пользователь.

4.3 Результаты и анализ

Программную реализацию моделей и процедур обучения, тестирования можно найти в github-репозитории работы[9].

Значения метрики качества сессионной задачи рекомендации AverageReward приведены в Таблице 1. Как видно, предложенная модель, TGQN, превосходит аналог, GCQN, на 10-30%, что подтверждает предположение, о возможной потере информации во время дискретизации динамического графа. Предложенная модель также превосходит стандартные модели решения данной задачи, что говорит о состоятельности метода и возможности его применения на практике.

Глава 5

Заключение

Работа посвящена исследованию графовых методов решения задачи рекомендации онлайн. Задача рекомендации сегодня возникает во многих сферах человеческой жизни и потому качественное её решение крайне востребовано. Решение задачи рекомендации в онлайн постановке с применением инструментария обучения с подкреплением обладает большим количеством преимуществ перед классической постановкой, в их числе обучение модели во время эксплуатации, гибкая функция награды, позволяющая во время обучения учитывать дополнительные целевые метрики и так далее. А использование графовых нейронных сетей для извлечения информации из истории взаимодействий пользователей с товарами, имеющей графовую структуру, позволяет основанным на них рекоммендерам одновременно эксплуатировать два мощных подхода к решению задачи рекомендации, что положительно отражается на качестве. В рамках работы был проведен анализ существующего графового метода решения задачи рекомендации онлайн — Graph Convolutional Q-Network, найдены его недостатки, в том числе, дискретный учет динамики графа взаимодействий пользователей и товаров, который потенциально может приводить к потере важной информации, и, соответственно, снижать качество решения задачи. Была предпринята попытка устранения данного недостатка, путем внедрения, вместо используемого в GCQN для извлечения графовой информации фреймворка Graph Convolutional Recurrent Network, более совершенного, способного учитывать динамику графа непрерывно, фреймворка Temporal Graph Network. В результате была получена новая усовершенствованная модель Temporal Graph Q-

Network.

Temporal Graph Q-Network обладает всеми преимуществами рекомендательных систем, основанных на графовых нейронных сетях, при этом учитывающая динамику графа взаимодействий пользователей и товаров непрерывно.

В результате численного эксперимента, было показано, что предложенная модель превосходит в метрике качества Average Reward в решении сессионной задачи рекомендации онлайн, как графовые аналоги, так и некоторые стандартные для данной задачи модели, что говорит о состоятельности метода и возможности его успешного применения на практике.

Литература

[1] N., Kipf T. Semi-Supervised Classification with Graph Convolutional Networks. — 2017.

[2] P., Veli˘ckovi´c. Graph attention networks. — 2017.

[3] L., Hamilton W. Inductive representation learning on large graphs / Hamilton W. L., Ying R., Leskovec J. // Proceedings of IC on NIPS'17. — 2017. — Pp. 1025–1035.

[4] Neural message passing for quantum chemistry / Justin Gilmer, Samuel S Schoenholz, Patrick F Riley et al. // International conference on machine learning / PMLR. — 2017. — Pp. 1263–1272.

[5] Haykin, Simon. Neural networks: a comprehensive foundation / Simon Haykin. — Prentice Hall PTR, 1994.

[6] Marhon, Sajid A. Recurrent Neural Networks / Sajid A. Marhon, Christopher J. F. Cameron, Stefan C. Kremer // Handbook on Neural Information Processing / Ed. by Monica Bianchini, Marco Maggini, Lakhmi C. Jain. — Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. — Pp. 29–65.

[7] Hochreiter, Sepp. Long short-term memory / Sepp Hochreiter, J¨urgen Schmidhuber // Neural computation. — 1997. — Vol. 9, no. 8. — Pp. 1735–1780.

[8] Learning phrase representations using RNN encoder-decoder for statistical machine translation / Kyunghyun Cho, Bart Van Merri¨enboer, Caglar Gulcehre et al. // arXiv preprint arXiv:1406.1078. — 2014.

[9] Mineev, Nikita, Recommender systems based on graph neural netowrks, (2023), GitHub repository, https://github.com/nmineev/mipt-masters-project

[10] J., You. Graphrnn: Generating realistic graphs with deep auto-regressive models / You J., et al. // ICML'18 / PMLR. — 2018. — Pp. 5708–5717.

[11] Y., Seo. Structured Sequence Modeling with Graph Convolutional Recurrent Networks. — 2016.

[12] E., Rossi. Temporal Graph Networks for Deep Learning on Dynamic Graphs. — 2020.

[13] F., Manessi. Dynamic graph convolutional networks / Manessi F., Rozza A., Manzo M. // Pattern Recognition. — 2020. — Vol. 97. — P. 107000.

[14] J., Chen. GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction. — 2018.

[15] L., Zhao. T-gcn: A temporal graph convolutional network for traffic prediction / Zhao L., et al. // IEEE ITSS. — 2019. — Vol. 21, no. 9. — Pp. 3848–3858.

[16] Y., Li. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. — 2018.

[17] B., Yu. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting / Yu B., Yin H., Zhu Z. // Proceedings of IJCAI-18. — International Joint Conferences on Artificial Intelligence Organization, 2018. — Pp. 3634–3640.

[18] Fathy, A. TemporalGAT: Attention-Based Dynamic Graph Representation Learning / A Fathy, et al. // Advances in Knowledge Discovery and Data Mining. — Cham: Springer, 2020. — Pp. 413–423.

[19] Z., Liu. Motif-Preserving Dynamic Attributed Network Embedding / Liu Z., et al. // Proceedings of the Web Conference 2021. — WWW '21. — Association for Computing Machinery, 2021. — P. 1629–1638.

[20] R., Trivedi. DyRep: Learning Representations over Dynamic Graphs / Trivedi R., et al. // ICLR (Poster). — LA: OpenReview.net, 2019.

[21] D., Xu. Inductive Representation Learning on Temporal Graphs. — 2020.

[22] S., Kumar. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. — 2019.

[23] Representation Learning for Dynamic Graphs: A Survey. / Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain et al. // J. Mach. Learn. Res. — 2020. — Vol. 21, no. 70. — Pp. 1–73.

[24] Y., Peng. A Survey on Modern Recommendation System based on Big Data. — 2022.

[25] A. C., Rivera. Recommendation Systems in Education: A Systematic Mapping Study . — 2018.

[26] T., Tran. Recommender systems in the healthcare domain: state-of-the-art and research issues. — 2021.

[27] E., Zangerle. Evaluating Recommender Systems: Survey and Framework. — 2022

[28] F.Ricci, L.Rokach, B.Shapira. Recommender Systems Handbook. Springer. 2015. https://shuaizhang.tech/posts/2019/08/blog-post-2

[29] Rui Chen et al. A survey of collaborative ltering-based recommender systems: from traditional methods to hybrid methods based on social networks. 2018.

[30] S., Wu. Graph Neural Networks in Recommender Systems: A Survey. — 2022

[31] X., Chen. A Survey of Deep Reinforcement Learning in Recommender Systems: A Systematic Review and Future Directions. — 2021

[32] Y., Lei. Reinforcement Learning based Recommendation with Graph Convolutional Q-network. — 2020

[33] H., Ko. A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. — 2022

[34] P. Lops, M. d. Gemmis, and G. Semeraro, “Content-based recommender systems: State of the art and trends,” Recommender systems handbook, pp. 73–105, 2011.

[35] C., Gao. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. — 2023

[36] Neural message passing for quantum chemistry / Justin Gilmer, Samuel S Schoenholz, Patrick F Riley et al. // International conference on machine learning / PMLR. — 2017. — Pp. 1263–1272.

[37] C., Watkins. 1992. Q-learning. Machine learning 8, 3-4 (1992), 279–292.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. nature 518, 7540 (2015), 529–533

[39] GroupLens Research. MovieLens 1M Dataset. — https://grouplens.org/datasets/movielens/1m/. — 2023. — June.

[40] Google LLC, Kaggle. [Weekly] 15M+ Game Recommendations on Steam. – https://www.kaggle.com/datasets/antonkozyriev/game-recommendations-on-steam. – 2023. – June.

[41] M., Wan. Fine-grained spoiler detection from large-scale review corpora. – https://cseweb.ucsd.edu/ jmcauley/datasets.html. – 2019

[42] Y., Zhang. An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond. — 2022

[43] B., Hidasi. Session-based Recommendations with Recurrent Neural Networks. — 2015

[44] P., Veli˘ckovi´c. Graph attention networks. — 2017