Сравнительный анализ современных алгоритмов генерирования рекомендаций на основе сессий, применительно к потоковому сценарию использования (Streaming Session-based Recommendation)

Д.Р. Якупов

***Аннотация* — Системы персонализированных рекомендаций активно применяются во многих областях современной жизни (электронная коммерция, банковская сфера, коммуникации, развлечения и т.д.) и имеют огромное значение для бизнеса и потребителей. Отдельным классом данных систем являются системы рекомендаций на основе сессий, ключевой особенностью которых является генерирование рекомендаций с учетом последних действий пользователя в системе (его текущей сессии), анализ которых позволяет выявить текущие намерения и интересы пользователя. Особенно актуальным является применение систем рекомендаций на основе сессий в потоковом сценарии использования (*Streaming Session-based Recommender Systems*), например, на платформах развлекательного контента, маркетплейсах и т.д. Отличительной особенностью потокового сценария является непрерывный, высокообъемный и высокоскоростной характер поступления новых данных, которые необходимо обрабатывать в режиме реального времени. В настоящей работе проведен сравнительный анализ современных алгоритмов систем рекомендаций на основе сессий для потокового сценария использования: *Streaming Session-based Recommendation Machine, Global Attributed Graph Neural Network, Multi Global Information Assisted Streaming Session-Based Recommendation System*, выделены общие принципы построения данных систем, их основные различия, рассмотрены преимущества и недостатки. На основе исследования и анализа данных систем разработаны базовые (типовые) рекомендации по построению архитектуры, алгоритмам и сценарию работы рекомендательных систем на основе сессий в зависимости от внешних условий.**

***Ключевые слова*— Рекомендательные системы на основе сессий для потокового сценария использования, Streaming Session-based Recommendation.**

I. Введение

Бурное развитие интернет-технологий предоставило пользователям доступ к огромному количеству онлайн-контента, к новостям, развлечениям, товарам, услугам и т.д. При таком разнообразии предлагаемых товаров и количестве новой информации, поступающей ежесекундно, пользователи все чаще отдают предпочтение тем компаниям, которые предоставляют им качественные и своевременные рекомендации.

Например, согласно отчету McKinsey [1]:

1) 71% потребителей ожидают от компаний персонализированного взаимодействия, и 76% - расстраиваются, когда этого не происходит;

2) компании, преуспевающие в персонализации, получают на 40% больше дохода от этой деятельности, чем средние игроки.

В целях решения задачи генерирования для пользователей своевременных и качественных рекомендаций в настоящее время проводятся активные исследования в области разработки и внедрения рекомендательных систем.

В общем случае система рекомендаций представляет из себя вспомогательную систему, которая помогает пользователям находить информацию, продукты или услуги посредством обобщения и анализа предпочтений других пользователей и характерных особенностей текущего пользователя [2].

Отдельным классом данных систем являются системы рекомендаций на основе сессий (*Session-based Recommender System, SRS*) [3], ключевой особенностью которых является генерирование рекомендаций с учетом последних действий пользователя в системе (его текущей сессии), анализ которых позволяет выявить текущие намерения и интересы пользователя.

Особенно актуальным является применение систем рекомендаций на основе сессий в потоковом сценарии использования (*Streaming Session-based Recommender Systems, SSRS*), например, на платформах развлекательного контента, маркетплейсах и т.д. Отличительной особенностью потокового сценария является непрерывный, высокообъемный и высокоскоростной характер поступления новых данных, которые необходимо обрабатывать в режиме реального времени.

В настоящей работе проведено исследование и сравнительный анализ трех современных алгоритмов систем рекомендаций на основе сессий, предлагаемых различными исследователями для применения в потоковом сценарии:

- SSRM [4] (Streaming Session-based Recommendation Machine) - впервые представлен в 2019 году на 25-й

Статья получена 5 мая 2023. Статья подготовлена в рамках работы над магистерской диссертацией на факультете ВМК МГУ имени М.В. Ломоносова.

Д.Р. Якупов – МГУ имени М.В. Ломоносова (email: dyakupov1@gmail.com).

международной конференции «SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19);

• GAG [5] (Global Attributed Graph Neural Network) - впервые представлен в 2020 году на 43-й международной конференции «ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)»;

• MGIA SSRS [6] (Multi Global Information Assisted Streaming Session-Based Recommendation System») - впервые представлен в 2022 году в журнале «IEEE Transactions on Knowledge and Data Engineering».

На основе выполненного исследования разработаны общие (базовые) рекомендации по построению архитектуры, алгоритмам и сценарию работы рекомендательных систем на основе сессий для потокового сценария использования.

Структура настоящей работы построена следующим образом: в разделах II, III, IV – выполнено подробное исследование архитектуры, алгоритмов и сценариев работы SSRM, GAG и MGIA SSRS соответственно, в разделе V – изложены результаты их сравнительного анализа, в разделе VI – представлены базовые (общие) рекомендацию по построению (выбору) архитектуры, алгоритмам и сценарию работы рекомендательных на основе сессий для потокового сценария использования.

II. STREAMING SESSION-BASED RECOMMENDATION MACHINE (SSRM)

Согласно открытым источникам [[4], [5], [6], [7]] алгоритм SSRM был первой в мире работой посвященной одновременному решению проблемы генерации рекомендаций на основе сессий и работы рекомендательной системы в потоковом сценарии использования.

Алгоритм SSRM предназначен для решения задач по рекомендации следующего взаимодействия в текущей сессии. В качестве входных данных SSRM используется комбинация данных, известных о текущей сессии, и исторических данных.

В SSRM рассматривается работа с:

• упорядоченными, персонализированными сессиями, с одним типом доступных действий [3];

• короткими, средними и длинными сессиями [3] – за исключением аномальных сессий, включающих только 1 или более 20 взаимодействий;

• структура сессионных данных – многоуровневая [3].

Авторы SSRM предлагают следующие, представленные в таб. 1, решения проблем, характерных для рекомендательных систем на основе сессий и потокового сценария использования.

Таб. 1. Проблемы SSRS и предлагаемые решения

№	Проблема	Предлагаемое решение
1	Наличие «шума» в сессии, создаваемого нерелевантными взаимодействиями пользователя с элементами (т.е. взаимодействиями, не отражающими его реальные текущие намерения и интересы), вызванными «человеческим фактором»: механической (или в связи с рассеянностью/утомлением) ошибкой при выборе элемента, любопытством и т.д.	Снижение влияния на рекомендации «шума», создаваемого нерелевантными взаимодействиями, посредством использования индикатора важности (*attention signal*) для пользователя конкретных элементов, просмотренных им в текущей сессии. Задачей индикатора важности является уловить основные (фундаментальные) предпочтения и интересы пользователя на основе исторических данных: – о его прошлых взаимодействиях с элементами; – совокупности взаимодействий близких (схожих с ним) других пользователей, характеризующих их коллаборативное (распространенное) поведение и отражающее популярные интересы и предпочтения в их сообществе.
2	**Особенности потокового режима:**	
2.1	Большой объем поступающих новых данных – требует мощных аппаратных ресурсов, но на практике ресурсы всегда ограничены, в связи с чем, непрактично, а порой невозможно, сохранять в памяти все поступающие данные.	Использовать подход *online learning* для обучения рекомендательной модели, который включает в себя два этапа: 1) предварительно обучить модель на основе имеющихся исторических данных; 2) производить обновление модели (без полного переобучения модели) на основе выборки наиболее информативных данных из комбинации: – новых поступающих данных; – выборки из исторических данных, предварительно отобранных и сохраненных в резервуаре. Под резервуаром понимается хранилище, в котором по определенным правилам сохраняются и исключается отдельные ранее завершенные (исторические) сессии. Таким образом: 1) исключается необходимость хранить все поступающие данные; 2) увеличивается скорость обновления модели, так как для ее обновления обрабатываются не все исторические и вновь поступающие данные, а только наиболее информативная их часть.
2.2	Высокая скорость поступления новых данных – требует высокой скорости обработки данных и обновления рекомендательной модели, модель должна обновляться и реагировать мгновенно, чтобы улавливать текущие намерения и интересы пользователя.	

A. Архитектура SSRM

Архитектура SSRM представлена на рис. 1.

Сценарий работы SSRM включает в себя два

параллельно выполняющиеся процесса:

1) Генерирование рекомендаций - выполняется предварительно обученной рекомендательной моделью. Перед первым запуском рекомендательной модели SSRM обучение модели производится на основе исторических данных. В дальнейшем параметры рекомендательной модели периодически корректируются в соответствии с параметрами, получаемыми в процессе дообучения модели.

2) Обновление рекомендательной модели – выполняется посредством дообучения рекомендательной модели (с текущим набором параметров) на основе специально отобранной выборки наиболее информативных сессии (C^{sample}). Период обновления задается администратором системы.

На вход SSRM поступает поток новых сессий C^{new}, в котором каждая сессия s_i включает в себя упорядоченную по времени последовательность просмотренных соответствующим пользователем u к текущему моменту времени t элементов $s_i = \{u, [v_1, v_2, \dots, v_t]\}$.

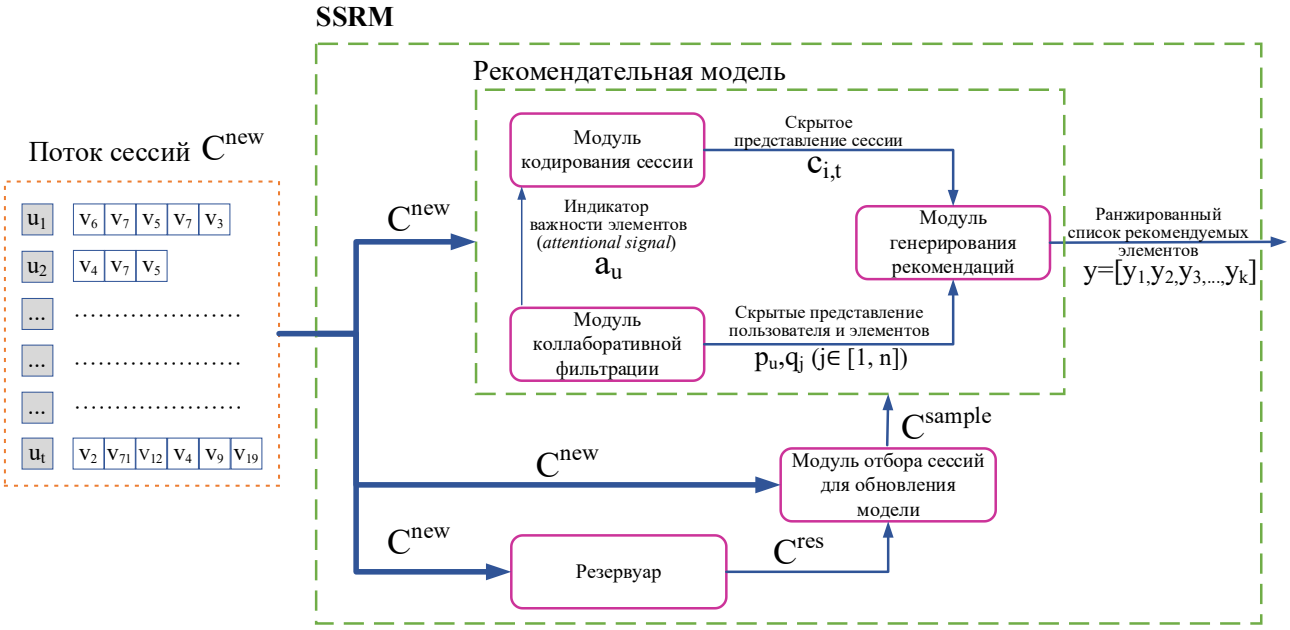

Рис. 1. Архитектура SSRM

На выходе SSRM выдает ранжированный список элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u. Так как, на практике пользователи просматривают только несколько первых из рекомендуемых им элементов, список рекомендаций ограничивается первыми k элементами, где число k задается администратором системы.

B. Рекомендательная модель SSRM

Рекомендательную модель SSRM (названную авторами *MF-based Attentive Session Recommender*) условно можно разделить три компонента:

1) Модуль кодирования сессий (*session feature encoder*) – компонент, отвечающий за кодирование последовательности взаимодействий пользователя с элементами в скрытое представление сессии;

2) Модуль коллаборативной фильтрации (*collaborative filtering*) – компонент, отвечающий хранение и обновление скрытых представлений пользователей и элементов (создаваемых посредством использования метода матричной факторизации) и генерацию индикаторов важности для модуля кодирования сессий;

3) Модуль генерирования рекомендаций (*predictions*) – компонент, отвечающий за вычисление рейтинга элементов и выдачу на основе него рекомендаций пользователю.

Внутренняя архитектура данных компонентов и взаимосвязи между ними представлены на рис. 2.

1) Модуль кодирования сессий

На вход модуля кодирования сессий в непрерывном режиме поступают текущие сессии s_i из потока сессий пользователей. На выходе модуль выдает сформированные векторы скрытых представлений сессий $c_{i,t}$. Вектор скрытого представления сессии представляет собой закодированную информацию о последовательности взаимодействий пользователя с элементами в рамках сессии с учетом оценки важности данных элементов для данного пользователя (соответствия их его долгосрочным интересам) на основе его прошлых исторических сессий.

Алгоритм обработки модулем поступающих сессий включает в себя следующие этапы:

а) Кодирование идентификаторов элементов сессии

Идентификаторы элементов v^{ID} каждой сессии $s_i = \{u^{ID}, [v_1^{ID}, v_2^{ID}, \dots, v_t^{ID}]$, поступающей на вход модуля кодирования, первоначально кодируются в одномерные вектора:

$$x_j = OneHotEncoding\left(v_j^{ID}\right)$$

где:

- $j \in [1, t]$ – порядковый индекс элемента в сессии;
- v_j^{ID} – идентификатор элемента j.

Для кодирования элементов используется кодирование *One Hot Encoding* (кодирование «1 из N»).

Сессия может содержать повторяющиеся элементы, т.е. возможно $v_a^{ID} = v_b^{ID}$, когда $a, b < t$, $a \neq b$. В таком случае $x_a = x_b$.

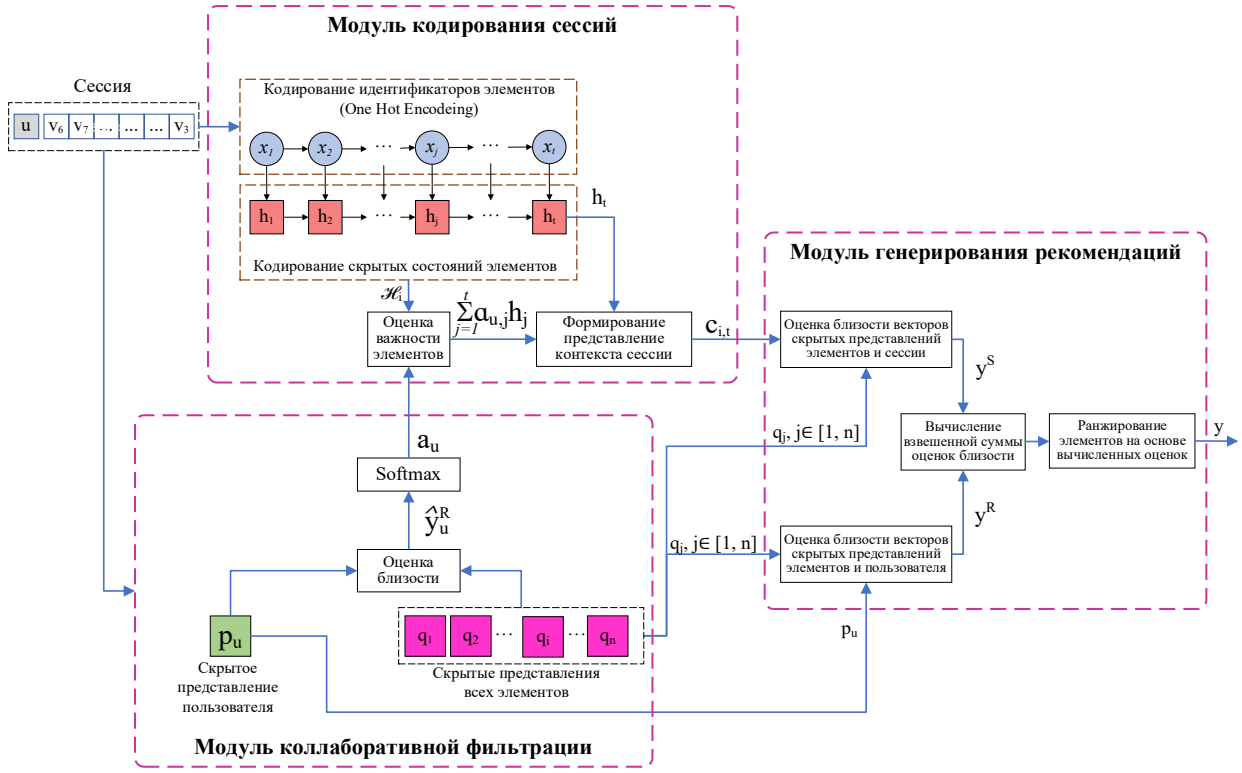

Рис. 2. Внутренняя архитектура рекомендательной модели SSRM

а) Кодирование скрытых состояний элементов

Элементы закодированной сессия $s_i^{ohe} = \{u, [x_1, x_2, \dots, x_t]\}$ кодируется в множество низкоразмерных скрытых состояний элементов $\mathcal{H}_i = [h_1, h_2, \dots, h_t]$ посредством использования рекуррентной нейронной сети с одним слоем управляемых рекуррентных блоков (*Gated Recurrent Unit*, далее - УРБ).

Для получения скрытых состояний элементов на вход УРБ последовательно передаются закодированные идентификаторы элементов сессии x_j $(1 \leq j \leq t)$, на выходе УРБ выдает скрытое состояние соответствующего элемента h_j. При этом, начиная со второго элемента x_2 на вход УРБ также рекуррентно передается вычисленное на предыдущем шаге скрытое состояния предыдущего элемента сессии h_{j-1} (рис. 3).

Внутреннее устройство УРБ представлено на рис. 4.

На j-шаге скрытое состояние j-го элемента вычисляется по формуле:

$$h_j = (1 - z_j)h_{j-1} + z_j\tilde{h}_j$$

где:

– h_{j-1} – скрытое состояние предыдущего (обработанного) элемента сессии;

– $\tilde{h}_j$ – промежуточное скрытое состояние текущего элемента сессии;

– z_j – вентиль обновления (*update gate*) информации.

Вентиль обновления z_t регулирует объем информации, которую необходимо «забыть» из скрытого состояния предыдущего элемента h_{j-1}, и объем информации, которую необходимо «запомнить» из промежуточного скрытого состояния элемента $\tilde{h}_j$.

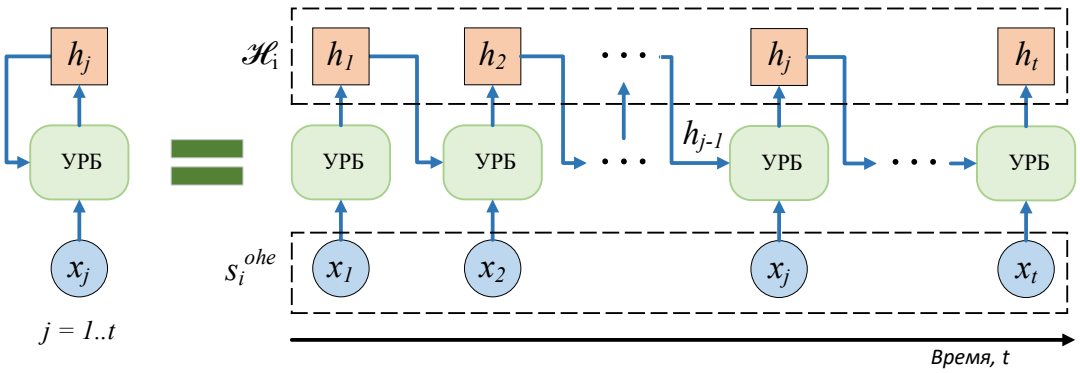

Рис. 3. Кодирование элементов сессии в их скрытые состояния

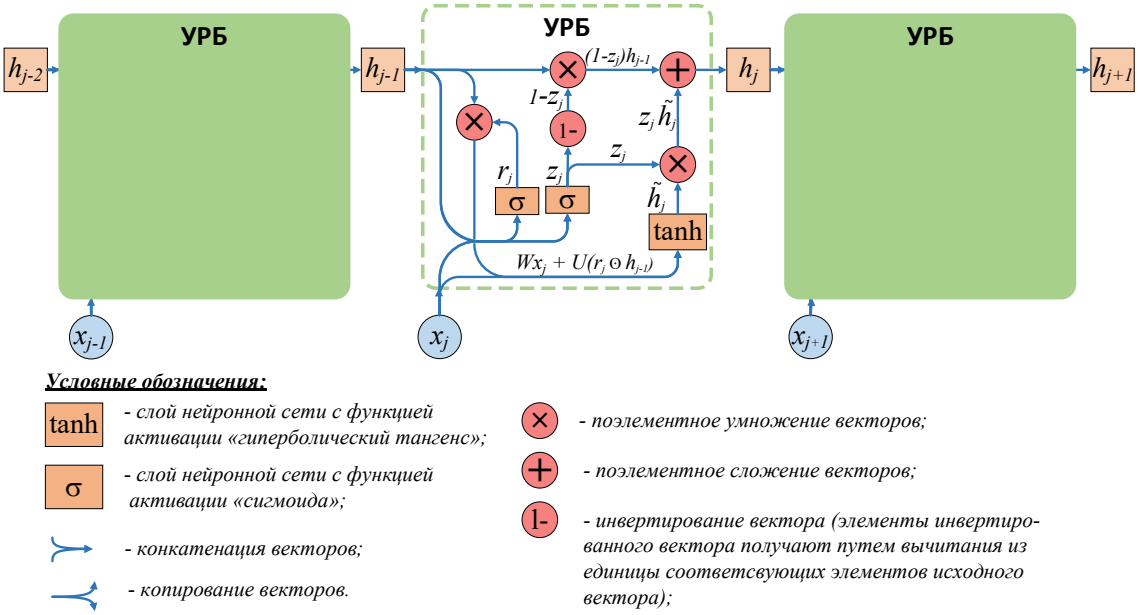

Рис. 4. Внутреннее устройство управляемого рекуррентного блока

Вентиль обновления на *j*-шаге вычисляется по формуле:

$$z_j = \sigma\left(W_z x_j + U_z h_{j-1}\right)$$

где W_z и U_z – матрицы весовых коэффициентов, σ – сигмоидальная функция.

Промежуточное скрытое состояние текущего элемента сессии на *j*-шаге вычисляется по формуле:

$$\tilde{h}_j = \tanh\left(Wx_j + U\left(r_j \odot h_{j-1}\right)\right)$$

где: *W* и *U* – матрицы весовых коэффициентов, tanh – функция гиперболического тангенса, r_j – вентиль сброса (*reset gate*), $\odot$ - произведение Адамара (покомпонентное умножение векторов).

Вентиль сброса r_j регулирует объем информации из скрытого состояния предыдущего элемента сессии h_{j-1}, который необходимо сохранить для вычисления промежуточного скрытого состояния текущего элемента сессии $\tilde{h}_j$.

Вентиль сброса на *j*-шаге вычисляется по формуле:

$$r_j = \sigma\left(W_r x_j + U_r h_{j-1}\right)$$

где W_r и U_r – матрицы весовых коэффициентов, σ – сигмоидальная функция.

В связи с рекуррентным вычислением скрытых состояний элементов, каждое последующее скрытое состояние элемента фактически консолидирует в себе информацию обо всех предшествующих ему элементах.

Таким образом, скрытое состояние последнего элемента сессии h_t консолидирует в себе информацию обо всех произошедших в рамках сессии взаимодействиях пользователя с элементами. Фактически h_t содержит в себе сжатое (закодированное) описание действий пользователя в рамках сессии, которое может использоваться при оценке его краткосрочных интересов и текущих намерений.

b) Оценивание важности для пользователя конкретных элементов сессии

После вычисления скрытых состояний всех элементов с учетом индикатора важности элементов a_u, полученного из модуля коллаборативной фильтрации в модуле кодирования сессии производиться оценка важности для пользователя конкретных элементов, просмотренных им в текущей сессии.

Индикатор важности элементов a_u представляет собой одномерный вектор, элементами которого являются весовые коэффициенты, характеризующие важность соответствующих им элементов сессии.

Алгоритм вычисления индикатора a_u будет рассмотрен далее в рамках модуля коллаборативной фильтрации.

Оценивание важности элементов производится посредством вычисления взвешенной суммы скрытых состояний элементов сессии:

$$\sum_{j=1}^{t} a_{u,j} h_j$$

По своему смыслу данная взвешенная сумма характеризует степень влияния, которую окажет каждый элемент из текущей сессии пользователя на формирование контекста сессии $\mathbf{c}_t$, который в дальнейшем будет использоваться для генерации списка рекомендаций.

c) Формирование представления контекста сессии

Посредством конкатенации вычисленной оценки важности элементов сессии и скрытого состояния h_t последнего просмотренного пользователем элемента в сессии модуль кодирования сессии формирует вектор скрытого представления $\mathbf{c}_{i,t}$ текущей сессии s_i:

$$\mathbf{c}_{i,t} = \left[\sum_{j=1}^{t} a_{u,j} h_j ; h_t \right]$$

Вычисленный вектор скрытого представления $\mathbf{c}_{i,t}$ дальше передается на вход модуля генерирования рекомендаций.

2) Модуль коллаборативной фильтрации

Модуль коллаборативной фильтрации хранит и обновляет исторические данные о взаимодействиях пользователей с элементами. Данная информация кодируется в форме матрицы $R = \left(r_{i,j}\right)_{m \times n}$, где m – количество пользователей, n – количество элементов, $r_{i,j} \in \{0,1\}$ – индикатор, отражающий наличие взаимодействия пользователя i с элементом j (0 – взаимодействия не было, 1 – взаимодействие было).

На основе исторических данных, закодированных в матрице R, посредством использования матричной факторизации модуль коллаборативной фильтрации формирует векторы скрытых представлений пользователей $p_u \in \mathbb{R}^{1 \times D}$ и элементов $q_j \in \mathbb{R}^{1 \times D}$, где D – размерность вектора, u -индекс пользователя, j – индекс элемента.

На вход модуля коллаборативной фильтрации в непрерывном режиме поступают текущие сессии s_i из потока сессий пользователей. На выходе модуль выдает:

1) в модуль кодирования сессий - индикаторы важности (*attentional signals*) a_u;

2) в модуль генерирования рекомендаций – скрытое представление пользователя p_u текущей сессии s_i и скрытые представления всех элементов $q_j, j \in [1, n]$.

Вычисление индикатора важности (*attentional signal*) a_u для пользователя u конкретных элементов, просмотренных им в текущей сессии, производится следующим образом:

1) Каждому элементу сессии $s_i = \{u, [v_1, v_2, \dots v_j, .., v_t]\}$ сопоставляется соответствующее скрытое представление q_j, а пользователю сессии u - скрытое представление p_u, полученные посредством использования матричной факторизации матрицы R;

2) Для каждого элемента сессии оценка близости векторов p_u и q_j посредством вычисления их скалярного произведения:

$$\hat{y}^R_{u,j} = \langle p_u, q_j \rangle = p_u q_j^T$$

Величина $\hat{y}^R_{u,j}$ характеризует насколько хорошо p_u соответствует q_j, т.е. насколько пользователю u нравится элемент j.

3) К полученному вектору $\hat{y}^R_u = [\hat{y}^R_{u,1}, \hat{y}^R_{u,2}, \dots, \hat{y}^R_{u,t}]$ применяется функция *softmax*:

$$a_u = softmax(\hat{y}^R_u)$$

Вычисленный вектор a_u передается в модуль кодирования сессии.

3) Модуль генерирования рекомендаций

На вход модуля генерирования рекомендаций поступает:

1) из модуля кодирования сессий - вектор скрытого представления $\mathbf{c}_{i,t}$ текущей сессии s_i;

2) из модуля коллаборативной фильтрации - скрытое представление пользователя p_u текущей сессии s_i и скрытые представления всех элементов $q_j, j \in [1, n]$.

На выходе модуль выдает ранжированный список из k элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u.

Алгоритм работы модуля генерирования рекомендаций:

1) для каждого из скрытых представлений элементов $q_j, j \in [1, n]$ производится оценка его близости к скрытому представлению $\mathbf{c}_{i,t}$ текущей сессии:

$$\hat{y}^S_{i,j} = q_t B c_i^T$$

где $B \in \mathbb{R}^{D \times H}$ – матрица трансформации размерности $D \times H$, D – размерность каждого из векторов скрытых представлений элементов

$q_j, j \in [1, n]$, H – размерность вектора скрытого представления сессии $\mathbf{c}_{i,t}$.

2) для каждого из скрытых представлений элементов $q_j, j \in [1, n]$ производится оценка его близости к скрытому представлению пользователя p_u текущей сессии:

$$\hat{y}_{i,j}^R = p_u q_j^T$$

3) для каждого элемента вычисляется его рейтинг посредством вычисления взвешенной суммы его оценок близости к скрытым представлениям сессии и пользователя:

$$\hat{y}_{i,j} = w\hat{y}_{i,j}^R + (1 - w)\hat{y}_{i,j}^S$$

где w – весовой коэффициент, определяющий степень влияния оценок $\hat{y}_{i,j}^R$ и $\hat{y}_{i,j}^S$ на итоговую оценку элемента.

4) на основе вычисленных рейтингов элементов выполняется их ранжирование, и первые k элементов с максимальным рейтингом выдаются пользователю в качестве рекомендации для следующего взаимодействия.

C. *Обучение рекомендательной модели*

Обучение рекомендательной модели SSRM выполняется в два этапа:

1) на первом этапе (до первого запуска SSRM в режиме генерации рекомендаций) производится предварительное обучение модели на основе имеющихся исторических данных;

2) на втором этапе (после запуска SSRM в режиме генерации рекомендаций) производится периодическое дообучение модели на основе специально отобранной выборки наиболее информативных сессии (C^{sample}).

Обучение рекомендательной модели SSRM рассматривается как решение задачи классификации, где в качестве функции потерь используется функция перекрестной энтропии (*cross-entropy*):

$$L(r_u, \hat{y}_u) = -\sum_{i=1}^{n} r_{u,i} \log(\hat{y}_{u,i})$$

где:

– i – индекс сессии;

– u – индекс пользователя;

– r_u – истинное распределение вероятностей выбора элементов для пользователя u;

– $\hat{y}_u$ – прогнозируемое распределение вероятностей выбора элементов для пользователя u.

Для повышения размера обучающей выборки используется метод увеличения данных (*data augmentation technique*) [8], суть которого заключается в том, что все начальные подпоследовательности взаимодействий с элементами исходных сессий дополнительно рассматриваются и используются при обучении рекомендательной модели в качестве отдельных сессий (рис. 5).

Алгоритм обучения: Truncated Backpropagation-Through-Time (*TBPTT*).

Метод оптимизации: ADAM (*adaptive moment estimation*) [9].

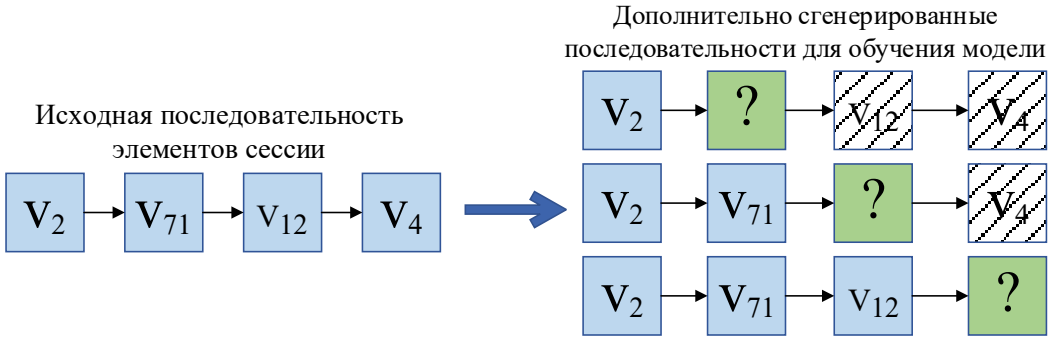

Рис. 5. Процесс генерации дополнительных данных для обучения модели

D. *Дообучение рекомендательной модели*

Дообучение (обновление) рекомендательной модели (*online updating*) позволяет избежать полного переобучения модели (на всех данных) каждый раз при поступлении новых данных.

Обновление рекомендательной модели производится на периодической основе, размер временного периода W, отводящегося на процесс дообучение модели (определения новых параметров модели) и ее обновления, является фиксированным и задается администратором системы.

Для дообучения рекомендательной модели авторы SSRM предлагают использовать выборку наиболее информативных данных из комбинации:

– новых поступающих данных;

– выборки из исторических данных, предварительно отобранных и сохраненных в резервуаре.

Необходимость повторно использовать исторические данные для дообучения модели связана с тем, что если производить дообучение модели только на основе новых данных, то модель быстро «забудет» (утратит)

информацию о прошлых взаимодействиях пользователей.

Использование для дообучения модели двухэтапной выборки данных направлено на снижение вычислительной нагрузки на дообучение модели и соответственно является мерой, призванной решить проблему перегрузки системы, характерную для потокового сценария использования. Проблема перегрузки связана с ограниченностью вычислительных ресурсов, что приводит к невозможности обработать большой поток поступающих данных.

1) Отбор данных для дообучения рекомендательной модели

Как упоминалось выше отбор данных для обновления рекомендательной модели производится в два этапа:

1) отбор из потока новых сессий C^{new} подмножества сессий и сохранение их в резервуаре;

2) отбор сессий из множества $C^{new} \cup C^{res}$, где C^{res} – сессии, хранящиеся в резервуаре.

Множество сессий C^{new} служит источником информации для изучения моделью текущих (краткосрочных) интересов пользователей, а C^{res} источником информации для изучения долгосрочных предпочтений и интересов пользователей.

2) Отбор сессий для сохранения в резервуаре

Для отбора сессий, сохраняемых в резервуаре, используется алгоритм случайной выборки с помощью резервуара (*Random Sampling with a Reservoir*) [10].

Суть алгоритма заключается в следующем:

1) пусть:

- $C^{new} = [s_1, s_2, \dots, s_i, \dots]$ – хронологически упорядоченный поток поступающих новых сессий, где индекс i указывает на номер сессии в хронологическом порядке;
- $n = |C^{res}|$ - размер резервуара C^{res}, т.е. максимальное количество сессий, которые могут быть сохранены в резервуаре;

2) тогда:

- при $i \leq n$ – вероятность включения сессии s_i в резервуар равна 1;
- при $i > n$ – вероятность включения сессии s_i в резервуар равна $\frac{n}{i}$, при этом, одновременно с включением в резервуар новой сессии из него случайным образом (с равномерно распределенной вероятностью) выбирается и исключается одна из хранящихся в нем прошлых сессий.

3) Отбор сессий из множества $C^{new} \cup C^{res}$

Для отбора сессий из множества $C^{new} \cup C^{res}$ используется алгоритм, названный авторами SSRM стратегией активной выборки (*Active Sampling Strategy*).

Суть алгоритма заключается в следующем:

1) Для каждой сессии $s_i = \{u, [x_1, x_2, \dots, x_t]\}$ из множества $C^{new} \cup C^{res}$ рассчитывается ее оценка (*prediction score*) по формуле:

$$r_{s_i} = \frac{\sum_{k=1}^{t} r_{u,k}}{t}$$

где:

– $r_{u,k}$ – оценка способности текущей рекомендательной модели для пользователя u предсказать элемент k (для снижения вычислительной нагрузки рекомендательная модель используется только в части матричной факторизации $r_{u,k} = p_u q_k^T$);

– t – количество элементов в сессии s_i.

Получение сессией высокой оценки означает, что рекомендательная модель хорошо обучена для генерирования рекомендаций для таких сессий, и тем меньший вклад внесет эта сессия в корректировку модели при ее обновлении. И наоборот, сессии с низкими оценками являются более информативными для корректировки рекомендательной модели.

2) Сессии ранжируются на основе рассчитанных оценок в убывающем порядке, и для каждой сессии вычисляется ее весовой коэффициент:

$$w_{s_i} = \exp\left(\frac{rank_{s_i}}{|C \cup C^{new}|}\right)$$

где $rank_{s_i}$ – порядковый номер (ранг) сессии в ранжированном списке.

Для снижения вычислительной нагрузки предполагается, что ранги сессий не нужно пересчитывать в процессе обновления рекомендательной модели.

3) Для каждой сессии рассчитывается вероятность ее выбора для обновления рекомендательной модели:

$$p(s_i) = \frac{w_{s_i}}{\sum_{s_i \in C^{res} \cup C^{new}} w_{s_i}}$$

4) Алгоритм дообучения рекомендательной модели

Формальное описание алгоритма дообучения модели:

1) Входные данные алгоритма:

– M_t – рекомендательная модель в момент времени t;

– $C^{res} = \{s_1, s_2, \dots, s_{|C^{res}|}\}$ – текущий резервуар;

– $s_i = \{u, [x_1, x_2, \dots x_{|s_i|}]\}$ – новая сессия, поступившая в систему SSRM;

– W – период времени, отводящегося на дообучения модели.

2) Выходные данные:

– M_{t+W} – обновленная модель в момент времени $t + W$.

3) Алгоритм:

1: *для каждого* $x_t \in \{C^{res} \cup s_i\}$:

2: вычислить $r_{u,t}$;

3: *конец цикла*;

4: вычислить $p(s_i)$ для всех сессий из множества $\{C^{res} \cup s_i\}$;
5: *пока* $W > 0$:
6: случайно (в соответствии с рассчитанными $p(s_i)$) выбрать подмножество сессий S^{sample} из множества $\{C^{res} \cup s_i\}$;
7: обновить модель M_t методом batch Stochastic Gradient Descent;
8: *конец цикла*;
9: вернуть обновленную модель M_{t+W}.

III. GLOBAL ATTRIBUTED GRAPH NEURAL NETWORK (GAG)

Алгоритм GAG предназначен для решения задач по рекомендации следующего взаимодействия в текущей сессии. В качестве входных данных GAG используется комбинация данных, известных о текущей сессии, и исторических данных.

В GAG рассматривается работа с:

- упорядоченными, персонализированными сессиями, с одним типом доступных действий [3];
- короткими, средними и длинными сессиями [3] – за исключением аномальных сессий, включающих только 1 или более 20 взаимодействий;
- структура сессионных данных – многоуровневая [3].

Авторы GAG предлагают следующие, представленные в таб. 2, решения проблем, связанных с ограничениями SSRM.

Таб. 2. Ограничения SSRM и предлагаемые решения

№	Ограничения SSRM	Предлагаемое решение
1	В составе рекомендательной модели SSRM используется метод матричной факторизации, который не способен уловить сложные зависимости между пользователями и элементами.	Использовать графовую нейронную сеть для консолидирования и кодирования информации о долгосрочных и краткосрочных предпочтениях пользователя в его скрытое представление.
2	В SSRM отбор наиболее информативных сессий для обновления рекомендательной модели предложено производить с помощью алгоритма *Active Sampling Strategy*, который основан на вычислении ранга сессии с использованием метода матричной факторизации. Узким местом данного алгоритма является вычисление ранга сессии на основе скрытых представлений элементов, т.е. предполагается, что все элементы новых сессий, поступающих в модель, ранее уже изучались моделью и для них имеются скрытые представления. В реальности сессия может содержать новые элементы, для которых в модели еще нет скрытых представлений.	1) В выборку сессий для обновления модели сразу включать новые сессии, которые содержат пользователей или элементы, которые ранее не изучались моделью; 2) Для остальных сессий в качестве метрики информативности использовать расстояние Вассерштейна между списком рекомендаций для сессии, генерируемым моделью, и реальным взаимодействием пользователя. Низкое значение метрики означает, что рекомендательная модель хорошо обучена для генерирования рекомендаций для таких сессий, и тем меньший вклад внесет эта сессия в корректировку модели при ее обновлении. И наоборот, чем выше значение метрики, тем более информативной является сессия для корректировки рекомендательной модели.

A. *Архитектура GAG*

Архитектура GAG представлена на рис. 6.

Сценарий работы GAG аналогичен сценарию работы SSRM и также включает в себя два параллельно выполняющиеся процесса:

1. Генерирование рекомендаций - выполняется предварительно обученной рекомендательной моделью. Перед первым запуском рекомендательной модели GAG обучение модели производится на основе исторических данных. В дальнейшем параметры рекомендательной модели периодически корректируются в соответствии с параметрами, получаемыми в процессе обновления модели.

2. Обновление рекомендательной модели – выполняется посредством дообучения рекомендательной модели (с текущим набором параметров) на основе специально отобранной выборки наиболее информативных сессии (C^{sample}). Период обновления задается администратором системы.

На вход GAG поступает поток новых сессий C^{new}, в котором каждая сессия s_i включает упорядоченную по времени последовательность просмотренных соответствующим пользователем u к текущему моменту времени t элементов: $s_i = \{u, [v_1, v_2, \dots, v_t]\}$.

На выходе GAG выдает ранжированный список элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u. Так как, на практике пользователи просматривают только несколько первых из рекомендуемых им элементов, список рекомендаций ограничивается первыми k элементами, где число k задается администратором системы.

B. *Рекомендательная модель GAG*

Рекомендательную модель GAG условно можно разделить два компонента:

1) Модуль кодирования – компонент, отвечающий за кодирование информации из поступающих сессий в скрытое представление соответствующего пользователя сессии;

2) Модуль генерирования рекомендаций – компонент, отвечающий за вычисление рейтинга элементов и выдачу на основе него рекомендаций пользователю.

1) *Модуль кодирования*

На вход модуля кодирования сессий в непрерывном режиме поступают текущие сессии s_i из потока сессий пользователей. На выходе модуль выдает обновленное скрытое представление пользователя сессии u^{upd}.

Внутренняя архитектура модуля кодирования представлена на рис. 7.

Алгоритм обработки модулем кодирования поступающих сессий включает в себя следующие этапы:

a) Кодирование идентификаторов пользователя и элементов сессии

Каждая сессия $s_i = \{u^{ID}, [v_1^{ID}, v_2^{ID}, \dots, v_t^{ID}]\}$, поступающая на вход модуля кодирования, первоначально анализируется на предмет того, кодировались ли ранее идентификаторы пользователя и элементов, входящих в сессию, в их скрытые представления. Для ранее кодировавшихся идентификаторов далее используются их текущие скрытые представления.

Для ранее не кодировавшихся идентификаторов u^{ID} и элементов v^{ID} производится их первичное кодирование в одномерные вектора вещественных чисел:

$$x_j = Embed_v(v_j^{ID})$$
$$u = Embed_u(u^{ID})$$

где:

- $j \in [1, t]$ – порядковый индекс элемента в сессии;
- v_j^{ID} – идентификатор элемента j;
- u^{ID} – идентификатор пользователя;
- $Embed_v$ и $Embed_u$ – функции кодирования идентификаторов элементов и пользователя соответственно.

Авторы GAG прямо не указывают какие функции кодирования они используют.

Сессия может содержать повторяющиеся элементы, т.е. возможно $v_a^{ID} = v_b^{ID}$, когда $a, b < t$, $a \neq b$. В таком случае $x_a = x_b$.

b) Построение графа сессии

После кодирования идентификаторов, каждая сессия $s_i = \{u, [x_{1_1}, x_2, \dots, x_t]\}$ представляется в форме взвешенного ориентированного графа (рис. 7), в

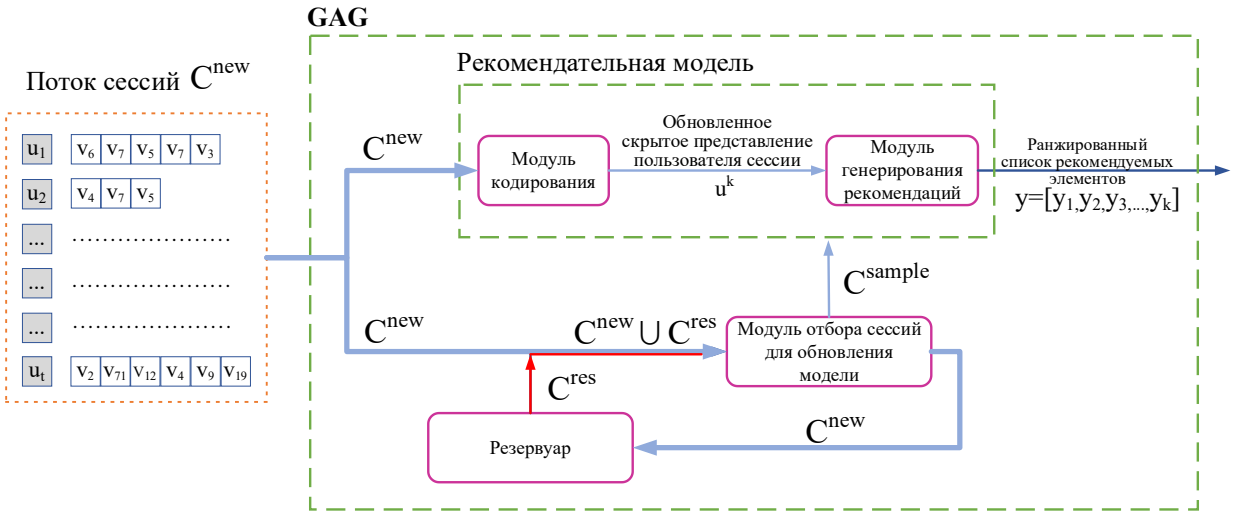

Рис. 6. Архитектура GAG

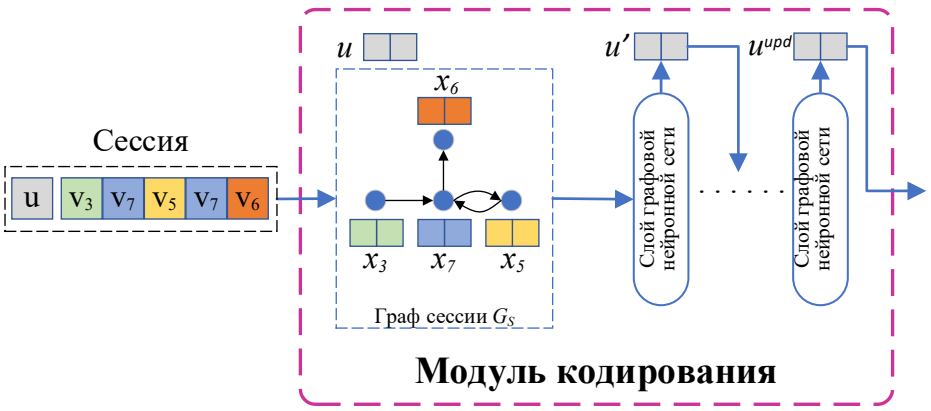

Рис. 7. Внутренняя архитектура модуля кодирования

котором вершинами являются элементы сессии, а ребрами – направления переходов пользователя между элементами. В зависимости от направления ребра, начальную вершину называют отправителем (*sender*), а конечную вершину - получателем (*receiver*).

Вершинам, ребрам и самому графу присваиваются атрибуты:

- вершинам – скрытые представления элементов сессии;
- ребрам – весовые коэффициенты, характеризующие частоту перехода пользователя между соответствующими элементами сессии;
- графу – скрытое представление пользователя сессии (также называемый глобальным атрибутом графа – *global attribute*).

Формально граф сессии можно описать следующим образом:

$$G_s = (u, V_s, E_s), G_s \in \mathcal{G}$$

где:

- G_{s_i} – граф сессии s_i;
- $\mathcal{G}$ – множество графов всех сессий;
- u – скрытое представление пользователя сессии s_i;
- V_{s_i} – множество всех вершин графа (скрытых представлений элементов сессии);
- E_{s_i} – множество всех направленных ребер графа, т.е. троичных кортежей $(w_{(n-1),n}, v_{n-1}, v_n)$, где $w_{(n-1),n}$ – вес ребра, направленного от элемента v_{n-1} к элементу v_n.

c) Выполнение свертки графа в глобальный атрибут

Свертка графа G_{s_i} в глобальный атрибут выполняется посредством использования графовой нейронной сети. Вычисления выполняются последовательно и для каждого слоя графой нейронной сети включают в себя следующие этапы:

1) Вектора скрытых представлений элементов сессии конкатенируются с текущим скрытым представлением пользователя (глобальным атрибутом графа), рис. 8.

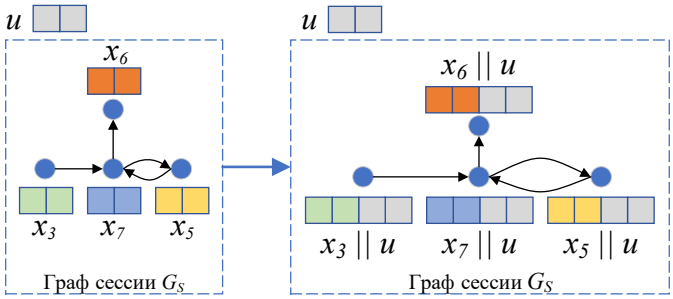

Рис. 8. Конкатенация векторов скрытых представлений элементов и пользователя сессии

2) Для каждого ребра графа вычисляется значение функции обновления $\mathbf{e_k}'$, которое в дальнейшем используется для пересчета значений вершин графа (скрытых представлений элементов сессии).

В модели GAG при вычислении свертки графа веса ребер в процессе вычислений не обновляются, в связи с тем, что они представлены в форме скалярных величин, а не в плотной векторной форме.

Так как, граф сессии является ориентированным, для учета различий, связанных с порядком перехода пользователя между элементами, значение функции обновления $\mathbf{e_k}'$ вычисляется в двух направлениях, т.е. отдельно для вершины-отправителя, из которой исходит ребро, и отдельно для вершины-получателя, в которую входит ребро.

Функции обновления $\mathbf{e_k}'$ вычисляются следующим образом:

$$\mathbf{e_{k,in}}' = \phi_{in}^{e}(\mathbf{e_k}, \mathbf{v_{r_k}}, \mathbf{v_{s_k}}, \mathbf{u}) = w_k \cdot \mathrm{MLP}_1(\mathbf{v_{r_k}} || \mathbf{u})$$

$$\mathbf{e_{k,out}}' = \phi_{out}^{e}(\mathbf{e_k}, \mathbf{v_{r_k}}, \mathbf{v_{s_k}}, \mathbf{u}) = w_k \cdot \mathrm{MLP}_2(\mathbf{v_{s_k}} || \mathbf{u})$$

где:

- $\mathbf{e_{k,in}}'$ – значение функции обновления для вершины-получателя $\mathbf{v_r}$, вычисляемое для ребра, исходящего из вершины-отправителя $\mathbf{v_s}$, вычисляемое на k – слое графовой нейронной сети;
- $\mathbf{e_{k,out}}'$ – значение функции обновления для вершины-отправителя $\mathbf{v_s}$, вычисляемое для ребра, входящего в вершину-получателя $\mathbf{v_r}$, вычисляемое на k – слое графовой нейронной сети;
- w_k – скалярная форма $\mathbf{e_k}$;
- $||$ – символ конкатенации векторов;
- MLP _ многослойный персептрон.

3) Для каждой вершины, в отдельности для входящих и исходящих ребер, вычисляется нормализованная сумма значений функций обновлений:

$$\mathbf{v_{i,in}}' = \sum_{j \in \{\mathbf{v_{s_j}} = \mathbf{v_{r_i}}\}} \frac{\mathbf{e_{j,out}}'}{\sqrt{N_{in}(i) N_{out}(j)}}$$

$$\mathbf{v_{i,out}}' = \sum_{j \in \{\mathbf{v_{r_j}} = \mathbf{v_{s_i}}\}} \frac{\mathbf{e_{j,in}}'}{\sqrt{N_{out}(i) N_{in}(j)}}$$

где:

- **i** – индекс вершины;
- $\mathbf{v_{i,in}}'$ – нормализованная сумма значений функций обновлений, вычисленных для ребер входящих в вершину **i**;
- $\mathbf{v_{i,out}}'$ – нормализованная сумма значений функций обновлений, вычисленных для ребер, исходящих из вершины **i**;
- $N_{out}(i)$ – исходящая степень вершины j;
- $N_{in}(j)$ – входящая степень вершины j.

4) Для каждой вершины вычисляется новое значение скрытого представления элемента:

$$\mathbf{v_i}' = \mathrm{MLP}(\mathbf{v_{i,in}}' || \mathbf{v_{i,out}}')$$

Таким образом, скрытое представление элемента сессии, фактически консолидирует в себе информацию:

- о самом элементе;

- об элементах, соседствующих с ним в сессии, т.е. элементах, просмотренных пользователем до и после него в сессии;
- частоте взаимодействий с этим элементом (переходах пользователя к этому элементу и от него);
- пользователе (глобальном атрибуте).

5) Обновление глобального атрибута (скрытого представления пользователя сессии) на основе всех вершин, ребер и текущего значения самого глобального атрибута. Для выделения долгосрочных и краткосрочных предпочтений пользователя в сессии (т.е. учета их в вычислении глобального атрибута с различными уровнями приоритета), объединение всех скрытых представлений элементов сессии выполняется посредством применения механизма самоконтроля (*self-attention*) на последнем элементе текущей сессии v_t:

$$\mathbf{u}' = \phi^u(V', \mathbf{u}) = \text{Self-Att}(v_t', v_i', \mathbf{u}) + \mathbf{u}$$

где $v_i \in V', i = 1, 2, 3 \dots t$ – новые вычисленные скрытые представления элементов.

Механизм самоконтроля реализуется следующим образом:

$$\alpha_i = \text{MLP}(\mathbf{v_t'}||\mathbf{v_i'}||\mathbf{u})$$

$$\mathbf{u_{sg}} = \sum_{i=1}^{n} \alpha_i \, \mathbf{v_i'}$$

где многослойный персептрон (MLP) используется для определения весовых коэффициентов для объединения скрытых представлений элементов сессии, последнего элемента в сессии и глобального атрибута.

6) На завершающем шаге в целях обогащения глобального атрибута дополнительной информацией о пользователе глобальный атрибут суммируется со скрытым представлением пользователя сессии:

$$\mathbf{u}' = \mathbf{u_{sg}} + \mathbf{u}$$

Описанные выше вычисления повторяются на каждом слое графовой нейронной сути. Вычисленный на последнем слое глобальный атрибут $\mathbf{u}'$ передается в модуль генерирования рекомендаций.

2) *Модуль генерирования рекомендаций*

На вход модуля генерирования рекомендаций поступает вычисленный глобальный атрибут $\mathbf{u}'$ текущей сессии и множество скрытых представлений всех элементов $\mathbf{X}$.

На выходе модуль выдает ранжированный список из k элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u.

Алгоритм работы модуля генерирования рекомендаций:

1) для каждого скрытого представления элемента из множества X вычисляется оценка близости его к глобальному атрибуту и формируется вектор оценок $\hat{z} \in \mathbb{R}^n$, где n – общее количество элементов:

$$\hat{\mathbf{z}} = \mathbf{u}'^{\mathbf{T}}\mathbf{X}$$

2) затем к полученному вектору оценок применяется функция *softmax,* на основании которой элементы ранжируются и первые k элементов с максимальным рейтингом выдаются пользователю в качестве рекомендации для следующего взаимодействия:

$$\hat{y} = softmax(\hat{\mathbf{z}})$$

C. *Обучение рекомендательной модели*

В части этапов обучения и используемой функции потерь обучение рекомендательной модели GAG одинаково с SSRM.

При обучении GAG (в отличии от SSRM) использование метода увеличения данных (*data augmentation technique*) не упоминается.

Алгоритм обучения: Backpropagation.

Метод оптимизации: ADAM (*adaptive moment estimation*) [9].

D. *Дообучение рекомендательной модели*

Дообучение (обновление) рекомендательной модели GAG выполняется аналогично SSRM. Далее будут рассмотрены только различия в алгоритмах дообучения GAG и SSRM.

Отличия алгоритма дообучения GAG от SSRM:

1) Из потока новых сессий C^{new} в выборку сессий C^{sample} для обновления модели сразу включаются сессии, которые содержат новых пользователей или элементы, которые ранее не изучались моделью;

2) Для остальных сессий, т.е. сессий из множества $C^{new} \cup C^{res} - C^{sample}$ процедура отбора информативных сессий построена на вычислении расстояния Вассерштейна между списком рекомендаций для сессии, генерируемым текущий моделью, и реальным взаимодействием пользователя.

3) Обновление резервуара на основе C^{new} производится после (а не до, как в SSRM) обновления рекомендательной модели. Сам алгоритм обновления резервуара (отбора сессий из C^{new}) – идентичен SSRM.

1) *Отбор сессий из множества $C^{new} \cup C^{res}$*

Отбор сессий из множества $C^{new} \cup C^{res}$ производится следующим образом:

1) Каждая сессия из множества C^{new} первоначально анализируется на предмет наличия в ней нового пользователя или элементов (т.е. таких, которые ранее не изучались рекомендательной моделью и для которых в модели отсутствуют соответствующие скрытые представления). Если сессия содержит нового пользователя или элемент, то она сразу включается в итоговую выборку сессий C^{sample} для обновления рекомендательной модели.

2) Для каждой сессии $s_i = \{u, [x_1, x_2, \dots, x_t]\}$ из множества $C^{new} \cup C^{res} - C^{sample}$ рассчитывается расстояние Вассерштейна между списком рекомендаций для сессии $\hat{y}$, генерируемым текущий моделью, и

реальным взаимодействием y (*one-hot* вектор) пользователя:

$$d_i(y,\hat{y}) = \inf_{\gamma \in \Pi(y,\hat{y})} \mathrm{E}_{(x,y)\sim\gamma} = [\|x-y\|]$$

Низкое значение метрики означает, что рекомендательная модель хорошо обучена для генерирования рекомендаций для таких сессий, и тем меньший вклад внесет эта сессия в корректировку модели при ее обновлении. И наоборот, чем выше значение метрики, тем более информативной является сессия для корректировки рекомендательной модели.

3) Для каждой сессии s_i из множества $C^{new} \cup C^{res} - C^{sample}$ рассчитывается вероятность включения ее в итоговую выборку:

$$p(s_i) = \frac{d_{s_i}}{\sum_{s_j \in C^{new} \cup C^{res} - C^{sample}} d_{s_j}}$$

4) В соответствии с вычисленными $p(s_i)$ из множества $C^{new} \cup C^{res} - C^{sample}$ случайным образом выбираются сессии и включаются в итоговую выборку сессий C^{sample} для обновления рекомендательной модели.

2) Алгоритм дообучения рекомендательной модели

Формальное описание алгоритма дообучения модели:

1) Входные данные алгоритма:

– t – текущий момент времени;

– M_t – рекомендательная модель в момент времени t;

– C^{res} – текущее множество сессий, хранящихся в резервуаре;

– C^{new} – множество новых сессий, поступивших на вход GAG.

2) Выходные данные:

– M' – обновленная рекомендательная модель;

– $C^{upd.res}$ – обновленный резервуар.

3) Алгоритм:

1: инициализировать пустое множество C^{sample};
2: ***если*** последняя эпоха завершена:
3: ***для каждой*** $s_i \in C^{new}$:
4: ***если*** s_i включает новых пользователя или элементы:
5: включить s_i в C^{sample};
6: ***конец если***;
7: ***конец цикла***;
8: ***для каждой*** $s_i \in C^{new} \cup C^{res} - C^{sample}$:
9: вычислить d_i;
10: ***конец цикла***;
11: вычислить $p(s_i)$;
12: в соответствии с $p(s_i)$ из $C^{new} \cup C^{res} - C^{sample}$ случайным образом выбрать s_i и включить их в C^{sample};
13: ***конец если***;
14: на основе C^{sample} обновить модель M_t до модели M';
15: ***для каждой*** $s_i \in C^{new}$:
16: обновить резервуар (в соответствии с алгоритмом *Random Sampling with a Reservoir*).
17: обновить t;
18: ***конец цикла***.

IV. MULTI GLOBAL INFORMATION ASSISTED STREAMING SESSION-BASED RECOMMENDATION SYSTEM (MGIA SSRS)

Алгоритм MGIA SSRS предназначен для решения задач по рекомендации следующего взаимодействия в текущей сессии. В качестве входных данных MGIA SSRS используется комбинация данных, известных о текущей сессии, и исторических данных.

В MGIA SSRS рассматривается работа с:

- упорядоченными, персонализированными сессиями, с одним типом доступных действий [3];
- короткими, средними и длинными сессиями [3] – за исключением аномальных сессий, включающих только 1 или более 20 взаимодействий;
- структура сессионных данных – многоуровневая [3].

Авторы MGIA SSRS предлагают следующие, представленные в таб. 3, решения проблем, связанных с ограничениями GAG.

Таб. 3. Ограничения GAG и решения, предлагаемые авторами MGIA SSRS

№	Ограничения GAG	Предлагаемое решение
1	В модели GAG не учтена возможность использовать для рекомендаций дополнительную полезную информацию, которую можно получить в результате совместного анализа сессий других пользователей, а именно информацию о общих закономерностях перехода между элементами, характерных для большинства пользователей. Учет данной информации позволит снизить влияние на рекомендации «шума», создаваемого нерелевантными взаимодействиями пользователя с элементами.	Построить и поддерживать обновление взвешенного ориентированного графа (называемого авторами глобальным), консолидирующего в себе информацию о взаимодействиях пользователей из всех сессий. Вершинами графа являются элементы, ребрами – направления переходов пользователей между элементами, каждому ребру присваивается вес, характеризующий частоту перехода пользователей между соответствующими элементами. В таком графе соседние элементы, связанные ребрами с более высокими весами с большей вероятностью будут похожи друг на друга. Анализ такой информации позволит помочь отличить элементы, связанные с реальным

		интересом пользователя, от нерелевантных взаимодействий.
2	Модель GAG не учитывает различия в важности различных признаков в скрытом представлении пользователя, что снижает эффективность учета его долгосрочных предпочтений.	Использовать механизм внимания для сбора информации о важности различных признаков в скрытом представлении пользователя.

A. Архитектура MGIA SSRS

Архитектура MGIA SSRS представлена на рис. 9.

Сценарий работы MGIA SSRS аналогичен сценарию работы SSRM и GAG, и также включает в себя два параллельно выполняющиеся процесса:

1. Генерирование рекомендаций - выполняется предварительно обученной рекомендательной моделью. Перед первым запуском рекомендательной модели MGIA SSRS обучение модели производится на основе исторических данных. В дальнейшем параметры рекомендательной модели периодически корректируются в соответствии с параметрами, получаемыми в процессе обновления модели.

2. Обновление рекомендательной модели – выполняется посредством дообучения рекомендательной модели (с текущим набором параметров) на основе специально отобранной выборки наиболее информативных сессии (C^{sample}). Период обновления задается администратором системы.

На вход MGIA SSRS поступает поток новых сессий C^{new}, в котором каждая сессия s_i включает упорядоченную по времени последовательность просмотренных соответствующим пользователем u к текущему моменту времени t элементов: $s_i = \{u, [v_1, v_2, \dots, v_t]\}$.

На выходе MGIA SSRS выдает ранжированный список элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u. Так как, на практике пользователи просматривают только несколько первых из рекомендуемых им элементов, список рекомендаций ограничивается первыми k элементами, где число k задается администратором системы.

B. Рекомендательная модель MGIA SSRS

Рекомендательную модель MGIA SSRS условно можно разделить три компонента:

1) Модуль кодирования элементов – компонент, отвечающий за кодирование информации из поступающих сессий в скрытые представления элементов;

2) Модуль кодирования сессий – компонент, отвечающий за свертку скрытых представлений элементов сессии в скрытое представление сессии.

3) Модуль генерирования рекомендаций – компонент, отвечающий за вычисление рейтинга элементов и выдачу на основе него рекомендаций пользователю.

1) Модуль кодирования элементов

На вход модуля кодирования элементов в непрерывном режиме поступают текущие сессии s_i из потока сессий пользователей. На выходе модуль выдает обновленное множество $\mathbf{H}^{upd}$скрытых представлений всех элементов.

Внутренняя архитектура модуля кодирования элементов представлена на рис. 10.

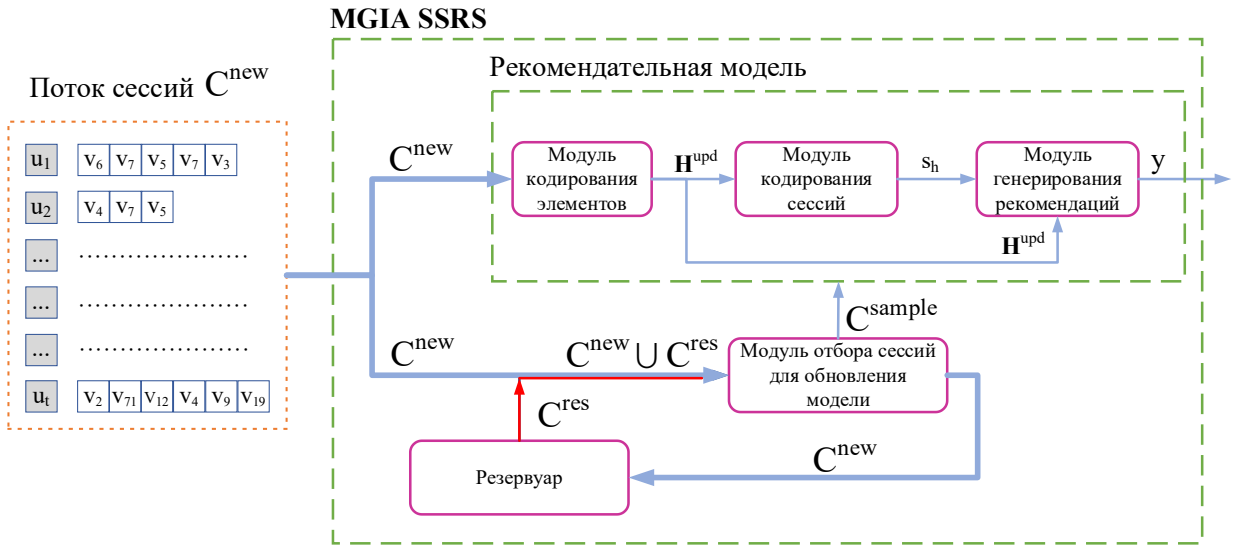

Рис. 9. Архитектура MGIA SSRS

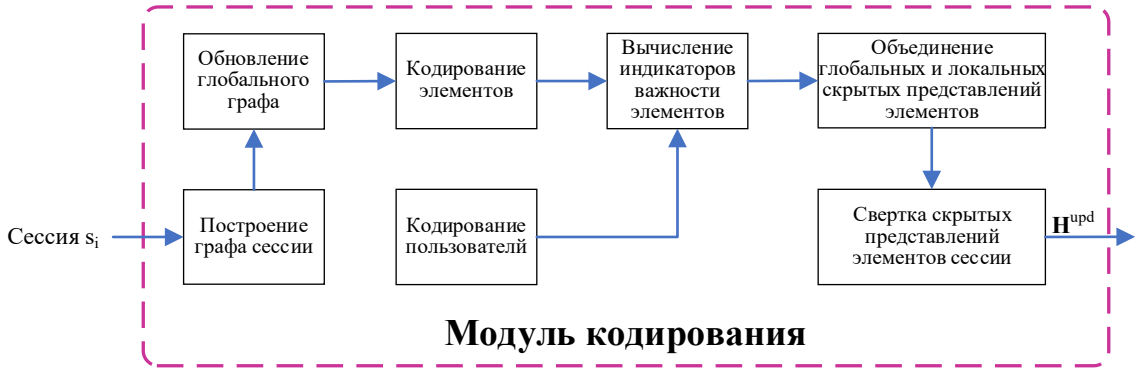

Рис. 10. Внутренняя архитектура модуля кодирования элементов рекомендательной модели MGIA SSRS

Алгоритм обработки модулем кодирования поступающих сессий включает в себя следующие этапы:

a) Построение графа сессии

Построение графа сессии в MGIA SSRS идентично построению графа сессии в GAG.

Каждая новая поступающая сессия $s_i = \{u, [x_{1_1}, x_2, \dots, x_t]\}$ представляется в форме взвешенного ориентированного графа, в котором вершинами являются элементы сессии, а ребрами – направления переходов пользователя между элементами. Каждому ребру присваивается весовой коэффициент, характеризующие частоту перехода пользователя между соответствующими элементами сессии.

Формально граф сессии описывается следующим образом:

$$G_{s_i} = (u, V_{s_i}, E_{s_i}), G_{s_i} \in \mathcal{G}$$

где:

- G_{s_i} – граф сессии s_i;
- $\mathcal{G}$ – множество графов всех сессий;
- u – пользователь сессии s_i;
- V_{s_i} – множество всех вершин графа (элементов сессии);
- E_{s_i} – множество всех направленных ребер графа, т.е. троичных кортежей $(w_{(n-1),n}, v_{n-1}, v_n)$, где $w_{(n-1),n}$ – вес ребра, направленного от элемента v_{n-1} к элементу v_n.

b) Построение глобального графа

Глобальный граф перехода между элементами строится посредством объединения в единый граф всех ранее построенных графов сессий (рис. 11). При поступлении в модуль кодирования новой сессии она также включается в глобальный граф.

Формально глобальный граф описывается следующим образом:

$$G^g = (V^g, E^g, W^g)$$

где:

- V^g – объединенное множество вершин всех графов из множества $\mathcal{G}$;
- $E^g_{i,j}$ – объединенное множество ребер всех графов из множества $\mathcal{G}$, где $1 \le i, j \le n$ – индекс ребра, исходящего из вершины i и в входящего в вершину j;
- W^g – множество весов всех ребер $E^g_{i,j}$.

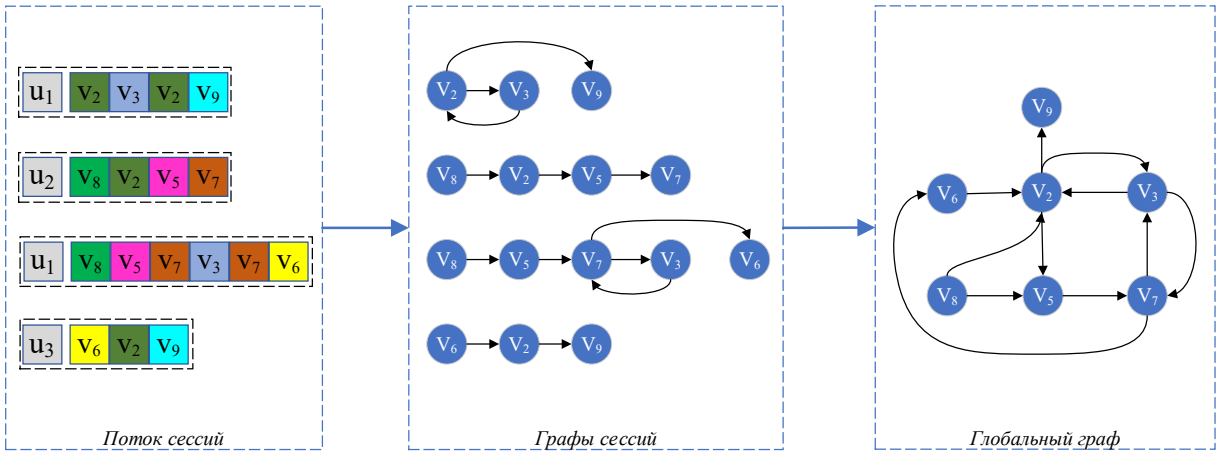

Рис. 11. Процесс построения глобального графа

Вес $w_{i,j} \in W^g$ ребра $E^g_{i,j}$ рассчитывается как сумма весов всех соответствующих ему ребер из множества графов всех сессий $\mathcal{G}$:

$$w^g_{i,j} = \sum_{G \in \mathcal{G}} w_{G,ij}$$

Далее, в целях нормализации, вес каждого ребра $w_{i,j}$ делится на максимальное значение веса, встречающееся в графе:

$$\widehat{w}_{i,j}^{g} = w_{i,j}^{g} / \max_{i,j \in V^{g}} \left(\sum_{G \in \mathcal{G}} w_{G,ij} \right)$$

где $\widehat{w}_{i,j}^{g}$ – нормализованный вес ребра.

c) Кодирование элементов

Каждый элемент в рекомендательной модели MGIA SSRS имеет два скрытых представления. Одно из представлений называется глобальным и вычисляется на основе обработки глобального графа, включающего взаимодействия всех пользователей, второе – локальным и вычисляется на основе взаимодействий пользователя в рамках сессии.

Первоначальные значения глобальных и локальных скрытых представлений элементов получаются путем кодирования их идентификаторов одномерными векторами случайных вещественных чисел.

Дальнейшее вычисление значений глобальных скрытых представлений элементов выполняется с использованием однослойной *Gated Graph Neural Network* на вход которой подается глобальный граф.

Формально, вычисление глобального скрытого представления элемента $\mathbf{v}_i^t$ на *t*-шаге в глобальном графе G^g описывается следующим образом:

$$a_i^t = \mathbf{A}_i [\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_m^{t-1}]^T \mathbf{H} + \mathbf{b}$$
$$\mathbf{z}_i^t = \sigma(\mathbf{W}_z \mathbf{a}_i^t + \mathbf{U}_z \mathbf{v}_i^{t-1})$$
$$\mathbf{r}_i^t = \sigma(\mathbf{W}_r \mathbf{a}_i^t + \mathbf{U}_r \mathbf{v}_i^{t-1})$$
$$\tilde{\mathbf{v}}_i^t = tanh(\mathbf{W}_o \mathbf{a}_i^t + \mathbf{U}_o (\mathbf{r}_i^t \odot \mathbf{v}_i^{t-1}))$$
$$\mathbf{v}_i^t = (1 - \mathbf{z}_i^t) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_i^t \odot \tilde{\mathbf{v}}_i^t$$

где:

- $\mathbf{H} \in \mathbb{R}^{d \times 2d}$ – матрица весовых коэффициентов, где *d* – размерность векторов;
- $\mathbf{z_i}$ – вентиль обновления (*update gate*) информации;
- $\mathbf{r_i}$ – вентиль сброса (*reset gate*);
- $\mathbf{v}_i^t$ – вектор скрытого представления узла v_i в момент времени *t*;
- $[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_m^{t-1}]$ – список векторов скрытых представлений элементов глобального графа элементов в момент времени *t*-1;
- σ – функция «сигмоида»;
- $\odot$ - поэлементное умножение векторов;
- $\mathbf{A} \in \mathbb{R}^{m \times 2m}$ – матрица, являющаяся конкатенацией двух матриц смежности $\mathbf{A}^{(out)}$ и $\mathbf{A}^{(in)}$, содержащих в себе веса исходящих и входящих ребер соответственно.

Поступление в модуль кодирования новой сессии и, соответственно, обновление глобального графа, влечет за собой обновление (пересчет) всех глобальных представлений элементов.

d) Кодирование пользователей

Каждый пользователь в рекомендательной модели MGIA SSRS имеет скрытое представление, называемое его глобальным скрытым представлением.

Первоначальные значения глобальных скрытых представлений пользователей получаются путем кодирования их идентификаторов одномерными векторами случайных вещественных чисел.

В дальнейшем глобальные скрытые представления пользователей обновляются в процессе обучения рекомендательной модели.

e) Вычисление индикаторов важности элементов

Для каждого элемента сессии, на основе его глобального представления и глобального представления пользователя сессии, вычисляется индикатор важности, характеризующий соответствие элемента предпочтениям пользователя сессии:

$$\alpha_{ui} = \frac{exp\ (LeakyReLU([\mathbf{W}_\alpha \mathbf{h}_i^\alpha + \mathbf{U}_\alpha \mathbf{u}^g]) * \mathbf{v}_\alpha^T)}{\sum_{k \in N_i} exp\ (LeakyReLU([\mathbf{W}_\alpha \mathbf{h}_i + \mathbf{U}_\alpha \mathbf{u}]) * \mathbf{v}_\alpha^T)}$$

где:

- α_{ui} – индикатор важности элемента *i* для пользователя *u;*
- $LeakyReLU$ – функция Leaky ReLU;
- $\mathbf{h}_i^\alpha$ – глобальное скрытое представление элемента *i*;
- $\mathbf{u}^g$ – глобальное скрытое представление пользователя *u*;
- $\mathbf{W}_\alpha$ и $\mathbf{U}_\alpha$ – матрицы весов для преобразования глобальных скрытых представлений пользователя и элемента в одно векторное пространство.

f) Объединение глобальных и локальных скрытых представлений элементов

Для каждого элемента сессии с учетом его индикатора важности для пользователя выполняется объединение глобального и локального скрытого представлений элемента:

$$\hat{\mathbf{h}}_i = \mathbf{h}_i + \alpha_{ui} * \mathbf{h}_i^g$$

где:

- $\hat{\mathbf{h}}_i$ – объединенное скрытое представление элемента *i*;
- $\mathbf{h}_i$ – локальное скрытое представление элемента *i*;
- α_{ui} – индикатор важности элемента *i* для пользователя *u;*
- $\mathbf{h}_i^g$ – глобальное скрытое представление элемента *i*.

Объединенные скрытые представления элементов передаются на вход модифицированной версии однослойной сверточной графовой сети (названной авторами *Bi-directed Attentional Graph Convolutional Network, BA-GCN*).

g) Свертка скрытых представлений элементов сессии

В модели MGIA SSRS свертка выполняется аналогично свертке в модели GAG:

1) Для каждой вершины v_k определяют два множества ее соседних вершин $S_{k,in}$ и $S_{k,out}$. В множество $S_{k,in}$ входят вершины $v_{k,in}$, которые имеют общие с вершиной v_k ребра, входящие в вершину v_k. В множество $S_{k,out}$ входят вершины $v_{k,out}$, которые имеют общие с вершиной v_k ребра, исходящие из вершины v_k.

2) Для каждого ребра графа вычисляется значение функции $\mathbf{e_k}$, которое в дальнейшем используется для пересчета значений вершин графа (скрытых представлений элементов сессии).

Так же, как и в модели GAG в модели MGIA SSRS при вычислении свертки графа веса ребер в процессе вычислений не обновляются.

Значение функции $\mathbf{e_k}$ вычисляется в двух направлениях, т.е. отдельно для вершины-отправителя, из которой исходит ребро, и отдельно для вершины-получателя, в которую входит ребро.

Значения функции $\mathbf{e_k}$ вычисляются следующим образом:

$$\mathbf{e_{k,in}} = \phi_{in}^{e}(w_e, v_{k,in}, v_{k,out}, u) = w_e * v_{k,in}$$

$$\mathbf{e_{k,out}} = \phi_{out}^{e}(w_e, v_{k,in}, v_{k,out}, u) = w_e * v_{k,out}$$

где:

- $\mathbf{e_{k,in}}$ – значение функции $\mathbf{e_k}$ для ребра, входящего в вершину v_k;
- $\mathbf{e_{k,out}}$ – значение функции $\mathbf{e_k}$ для ребра, исходящего из вершины v_k;
- w_e – вес соответствующего ребра.

3) После завершения расчета значений функции $\mathbf{e_k}$ для всех ребер графа, на основе вычисленных значений выполняется обновление скрытых представлений элементов.

Вычисления для каждого элемента производятся в две итерации: на первой производятся вычисления скрытого представления на основе обработки входящих ребер $\hat{\mathbf{h}}_{j,in}$, на второй – на основе обработки исходящих ребер $\hat{\mathbf{h}}_{j,out}$.

Для повышения чувствительности модели к различиям в важности для элемента различных его соседей вычисления производятся с использованием механизма внимания.

Далее, для примера, приводятся формулы расчета для обработки входящих ребер (для исходящих ребер вычисления производятся аналогично):

$$\mathbf{query}_i = \mathbf{W}_{query}\mathbf{h}_i + \mathbf{b}_{query}$$

$$\mathbf{key}_j = \mathbf{W}_{key}\mathbf{h}_j + \mathbf{b}_{key}$$

$$\beta_{i,j} = \frac{exp\left(ReLU\left(\frac{\mathbf{query}_i^T \mathbf{key}_j}{\sqrt{d}}\right)\right)}{\sum_{u \in N(i)} exp\left(ReLU\left(\frac{\mathbf{query}_i^T \mathbf{key}_j}{\sqrt{d}}\right)\right)}$$

$$\hat{\mathbf{h}}_{i,in} = \sum_{j \in N(i)} \beta_{ij} \odot \mathbf{h}_j$$

где:

- $\mathbf{h}_i$ и $\mathbf{h}_j$ – скрытые представления элементов i и j (до обновления);
- d – размерность скрытого слоя;
- $N(i)$ – множество соседей i-го элемента;
- $\hat{\mathbf{h}}_{i,in}$ – обновленное скрытое представление i-го элемента (на основе обработки входящих ребер);
- $\odot$ - поэлементное умножение.

4) После завершение вычисления $\hat{\mathbf{h}}_{i,in}$ и $\hat{\mathbf{h}}_{i,out}$ выполняется, с использованием механизма внимания, объединение их со скрытыми представлениями пользователей.

Вычисления для каждого элемента производятся в две итерации: на первой производится обработка $\hat{\mathbf{h}}_{i,in}$, на второй – $\hat{\mathbf{h}}_{i,out}$

Далее, для примера, приводятся формулы для обработки $\hat{\mathbf{h}}_{i,in}$ (для обработки $\hat{\mathbf{h}}_{i,out}$ вычисления производятся аналогично):

$$\gamma_{u,i} = \frac{exp\left(LeakyReLU\left(\hat{\mathbf{h}}_i^T \boldsymbol{u}\right)\right)}{\sum_{i \in V_S} exp\left(LeakyReLU\left(\hat{\mathbf{h}}_i^T \boldsymbol{u}\right)\right)}$$

$$\hat{\mathbf{h}}_{i,in}^{\gamma} = \hat{\mathbf{h}}_{i,in} + \gamma_{ui} * \mathbf{u}$$

где:

- $\hat{\mathbf{h}}_{i,in}$ – обновленное скрытое представление i-го элемента (на основе обработки входящих ребер);
- $\boldsymbol{u}$ – скрытое представление пользователя сессии;
- γ_{ui} – весовой коэффициент, характеризующий важность элемента i и пользователя u.

5) После завершения вычисления $\hat{\mathbf{h}}_{i,in}^{\gamma}$ и $\hat{\mathbf{h}}_{i,out}^{\gamma}$ выполняется их объединение в единое обновленное скрытое представление элемента:

$$\mathbf{out}_i = MLP(\hat{\mathbf{h}}_{i,in}^{\gamma} || \hat{\mathbf{h}}_{i,out}^{\gamma})$$

где:

- $\mathbf{out}_i$ – обновленное скрытое представление i-го элемента;
- MLP – многослойный персептрон.

Таким образом, обновленное скрытое представление элемента фактически консолидирует в себе информацию о глобальном и локальном скрытом представлении элемента, и глобальном представлении пользователя.

2) Модуль кодирования сессии

В модули кодирования сессии производится объединение всех скрытых представлений (локальных и глобальных) элементов в сессии и скрытого представления пользователя сессии в скрытое представление сессии с использованием механизма самоконтроля:

$$\delta_i = MLP_1(\mathbf{h}_l || \mathbf{h}_i || \mathbf{h}_i^g || \boldsymbol{u})$$

$$\mathbf{s} = \sum_{i=1}^{l} \delta_i(\mathbf{h_i} + \mathbf{h}_i^g)$$

$$\mathbf{s}_h = MLP_2(\mathbf{h_l}||\boldsymbol{s})$$

где:

- $\mathbf{h_l}$ – скрытое представление последнего элемента в сессии;
- $\mathbf{h_i}$ – скрытое представление i-го элемента сессии;
- $\mathbf{h}_i^g$ – глобальное скрытое представление i-го элемента сессии;
- $\boldsymbol{u}$ – скрытое представление пользователя сессии;
- MLP – многослойный персептрон.

Вычисленное скрытое представление сессии $\mathbf{s}_h$ далее передается в модуль генерации рекомендаций.

3) Модуль генерирования рекомендаций

На вход модуля генерирования рекомендаций поступает вычисленное скрытое представление сессии $\mathbf{s}_h$ и множество скрытых представлений всех элементов $\mathbf{H}$.

На выходе модуль выдает ранжированный список из k элементов $y = [v_1, v_2, \dots, v_k]$ для рекомендации соответствующему пользователю u.

Алгоритм работы модуля генерирования рекомендаций:

1) для каждого скрытого представления элемента из множества $\mathbf{H} \in \mathbb{R}^{d \times n}$ вычисляется оценка близости его к скрытому представлению сессии и формируется вектор оценок $\hat{z} \in \mathbb{R}^n$, где n – общее количество элементов:

$$\hat{z} = \mathbf{s}_h^T \mathbf{H}$$

2) затем к полученному вектору оценок применяется функция *softmax,* на основании которой элементы ранжируются и первые k элементов с максимальным рейтингом выдаются пользователю в качестве рекомендации для следующего взаимодействия:

$$\hat{y} = softmax(\hat{\mathbf{z}})$$

$$Rec = topk(\hat{y})$$

C. Обучение рекомендательной модели

В части этапов обучения и используемой функции потерь обучение рекомендательной модели MGIA SSRS одинаково с моделями GAG и SSRM.

При обучении MGIA SSRS (в отличии от SSRM) использование метода увеличения данных (*data augmentation technique*) не упоминается.

Алгоритм обучения: Backpropagation.

Метод оптимизации: ADAM (*adaptive moment estimation*) [9].

Дообучение (обновление) рекомендательной модели MGIA SSRS выполняется полностью аналогично GAG.

V. Сравнительный анализ SSRM, GAG и MGIA SSRS

A. Архитектура

Исследование алгоритмов SSRM, GAG и MGIA SSRS показало, что они имеют довольно схожую архитектуру, при проектировании которой закладывались следующие общие принципы:

1) Разделить процесс обучения рекомендательной модели на два этапа: offline и online обучение. Offline обучение производить предварительно (перед запуском рекомендательной модели) на базе имеющихся исторических данных. Online обучение производить параллельно с работой рекомендательной модели на основе комбинации непрерывно поступающих новых и исторических данных.

2) Учитывая непрерывный и высокообъемный характер поступления данных, сохранять для online обучения не весь объем данных, а только их часть. Для сохранения данных использовать временное хранилище (резервуар) небольшого объема.

3) Учитывая непрерывный и высокоскоростной характер поступления данных, использовать для online обучения не весь объем новых и исторических (сохраненных в резервуаре) данных, а только их часть. В качестве критерия для отбора данных использовать их информативность для рекомендательной модели, где под информативностью понимается низкая способность рекомендательной модели генерировать точные рекомендации на основе таких данных.

Основные различия в архитектуре алгоритмов SSRM, GAG и MGIA SSRS заключаются в архитектуре и алгоритмах работы рекомендательной модели:

1) в SSRM для кодирования скрытых представлений используется управляемые рекуррентные блоки (*Gated Recurrent Unit*), для учета исторических данных - индикатора важности (*attention signal*) вычисляемый посредством использования матричной факторизации (*Matrix Factorization*);

2) в GAG для кодирования скрытых представлений используется графовая сверточная сеть (*Graph Convolutional Networks*);

3) в MGIA SSRS для кодирования скрытых представлений используется графовая нейронная сеть с вентильными узлами (*Gated Graph Neuron Network*).

B. Рекомендательные модели

Рекомендательные модели SSRM, GAG и MGIA SSRS построены по схожему принципу, условно каждую из этих моделей можно разделить на два блока:

1) в первом блоке с использованием различных технологий производится кодирование информации:

- о новых и исторических сессиях пользователя в вектор вещественных чисел (скрытое представление), характеризующего пользователя (в GAG) или сессию (в SSRM и MGIA SSRS);

• об элементах в множество векторов вещественных чисел (скрытые представления), в котором каждый из векторов, характеризует соответствующий ему элемент.

2) во втором блоке производится оценка близости (сравнение сходства), полученных в первом блоке векторов вещественных чисел пользователя (сессии) с каждым из векторов элементов. По результатам оценки для рекомендации пользователю выбираются k-наиболее близких элементов.

Технологии кодирования подробно рассмотрены в соответствующих разделах, посвященных рекомендательным моделям SSRM, GAG и MGIA SSRS.

C. Алгоритмы отбора данных для сохранения в резервуаре

В SSRM, GAG и MGIA SSRS используются одинаковый алгоритм отбора данных для сохранения в резервуаре: *Random Sampling with a Reservoir* [10], который обеспечивает случайную выборку данных из всего потока данных.

D. Алгоритмы отбора информативных данных для online обучения

В SSRM, GAG и MGIA SSRS используются разные алгоритмы отбора наиболее информативных данных для online обучения рекомендательной модели:

1) в SSRM используется алгоритм Active Sampling Strategy, который основан на вычислении ранга сессии с использованием метода матричной факторизации;

2) в GAG и MGIA SSRS используется алгоритм, основанный на вычислении расстояния Вассерштейна между генерируемым моделью списком рекомендаций и реальным взаимодействием пользователя.

Узким местом алгоритма SSRM является предположение о том, что все элементы новых сессий, поступающих в модель, ранее уже изучались моделью и для них имеются скрытые представления. В реальности сессия может содержать новые элементы, для которых в модели еще нет скрытых представлений.

В отличие от SSRM, в GAG и MGIA SSRS в выборку информативных сессий для обновления модели сразу включаются новые сессии, которые содержат пользователей или элементы, которые ранее не изучались моделью.

E. Эффективность рекомендаций

Сравнение эффективности рекомендаций SSRM, GAG и MGIA SSRS произведено на основе результатов, представленных авторами в своих статьях.

В связи с тем, что только авторы GAG предоставили открытый доступ к своим исходным кодам, в рамках настоящей работы проверить воспроизводимость результатов и провести опытное сравнение эффективности алгоритмов на других датасетах не представляется возможным.

1) Датасеты

Датасеты (наборы данных), использованные для тестирования эффективности алгоритмов SSRM, GAG и MGIA SSRS, представлены в таб. 4.

Таб. 4. Датасеты

№	Датасет	SSRM	GAG	MGIA SSRS
1	Lastfm	+	+	+
2	Gowalla	+	+	+
3	Tmall	-	-	+

Предобработка датасетов производилась следующим образом:

1) Датасет Lastfm:

• в качестве рекомендуемых элементов из датасета было отобрано 10 000 самых популярных исполнителей;

• в качестве сессии рассматривались действия пользователя, совершенные в течение 8 часов;

• из итогового датасета исключены аномальные сессии длительностью менее 2 или более 20 взаимодействий.

Размер итогового датасета $D_{\text{Lastfm}}^{processed}$: **298 919** сессий.

2) Датасет Gowalla:

• в качестве рекомендуемых элементов из датасета было отобрано 30 000 самых популярных мест посещения;

• в качестве сессии рассматривались отметки пользователя о посещенных местах в течение одного дня;

• из итогового датасета исключены аномальные сессии длительностью менее 2 или более 20 взаимодействий.

Размер итогового датасета $D_{\text{Gowalla}}^{processed}$:

– у авторов SSRM и GAG: **198 608** сессий;

– у авторов MGIA SSRS: **435 722** сессий.

Авторы MGIA SSRS отметили, что они произвели предобработку датасета по тем же правилам, что и авторы GAG, но при этом количество сессий в итоговом датасете получилось значительно больше, чем в работе GAG (435 722 против 198 608). Также авторы MGIA SSRS отметили, что если исключать сессии длительностью менее 3 и более 20 взаимодействий, то они получают равное с авторами GAG количество сессий, но результаты работы GAG на таком датасете ухудшаются.

3) Датасет Tmall:

• в качестве рекомендуемых элементов из датасета было отобрано 10 000 самых популярных исполнителей;

• в качестве сессии рассматривалась данные по взаимодействиям пользователя на платформе Tmall в течение 10 дней;

• из итогового датасета исключены сессии длительностью менее 2 или более 20 взаимодействий.

Размер итогового датасета $D_{\text{Tmall}}^{processed}$: **962 384** или **809 993** сессии (в статье указаны противоречивые сведения: в текстовой части статьи указано 962 384 сессии, а в таблице №2: 809 993 сессии).

2) Алгоритм эмуляция потокового сценария

В SSRM, GAG и MGIA SSRS использовался одинаковый алгоритм для эмуляции потокового сценария (рис. 12), который включал в себя следующие шаги:

1) Все сессии в датасете (после его предобработки) $D^{processed}$ упорядочиваются в хронологическом порядке в соответствии со временем их завершения;

2) Датасет $D^{processed}$ разделяется на две части: тренировочное множество D^{train} и множество кандидатов $D^{candidate}$.

В D^{train} включают первые 60% сессий из упорядоченного датасета $D^{processed}$, а в $D^{candidate}$ – оставшиеся 40%.

Множество D^{train} рассматривается в качестве набора исторических данных и используется для предварительного offline обучения рекомендательной модели, а множество $D^{candidate}$ – для моделирования поступающего потока новых сессий.

3) Множество $D^{candidate}$ разделяется на 5 равных по размеру множеств $D_i^{test}, i \in [1,5]$, разделение производится в соответствии с хронологией завершения сессий.

4) Десять процентов (10%) множества D^{train} и множество D_1^{test} используются в качестве валидационного множества $D^{validation}$ для подбора оптимальных гиперпараметров модели.

5) В экспериментах все тестовые множества, предшествующие текущему, используются для online обучения модели, т.е. если в текущий момент модель генерирует предсказания для множества D_i^{test}, то для ее online обучения используются $\forall D_j^{test} \in D^{candidate}$, где $1 \leq j < i$.

6) Временное окно для обновления модели (время между поступления в модель множеств D^{test} задается фиксированным и одинаковым для всех тестируемых моделей.

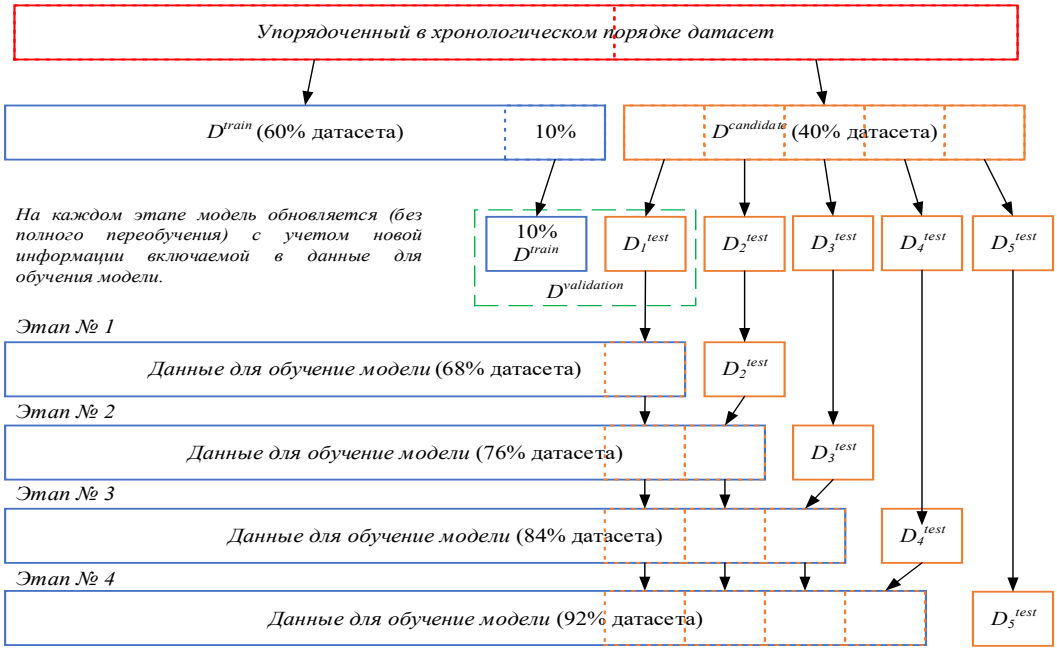

Рис. 12. Алгоритм эмуляции потокового сценария

3) Метрики

Для оценки эффективности рекомендаций SSRM, GAG и MGIA SSRS использовались метрики Recall@K и MRR@K (*Mean Reciprocal Rank*), где K – количество рекомендуемых пользователю элементов.

4) Результаты оценки эффективности алгоритмов

Достигнутые результаты эффективности алгоритмов SSRM, GAG и MGIA SSRS приведены на рис. 13-16.

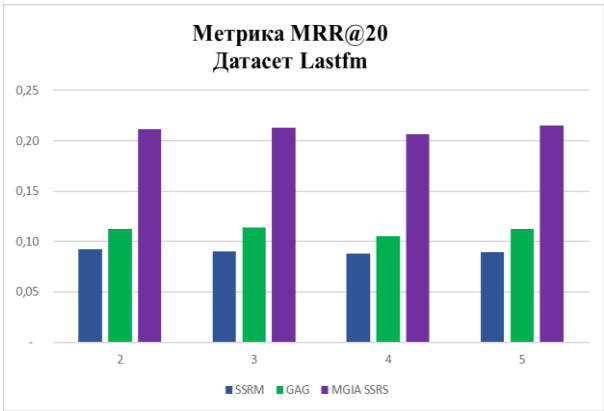

Рис. 13. Эффективность алгоритмов

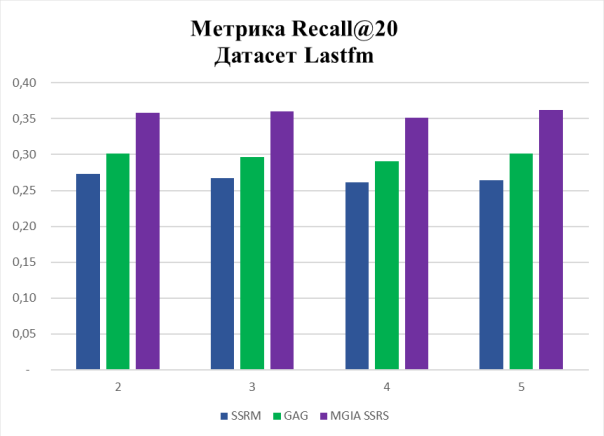

Рис. 14. Эффективность алгоритмов

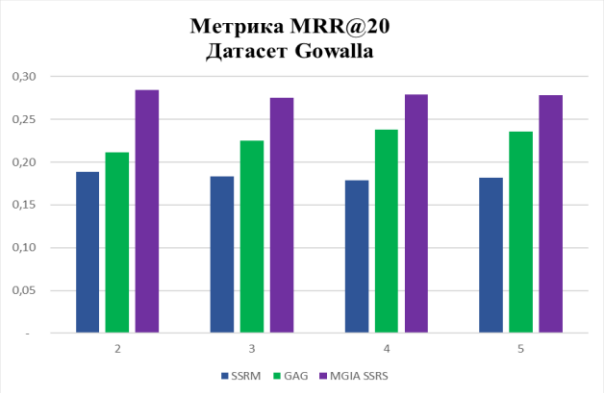

Рис. 15. Эффективность алгоритмов

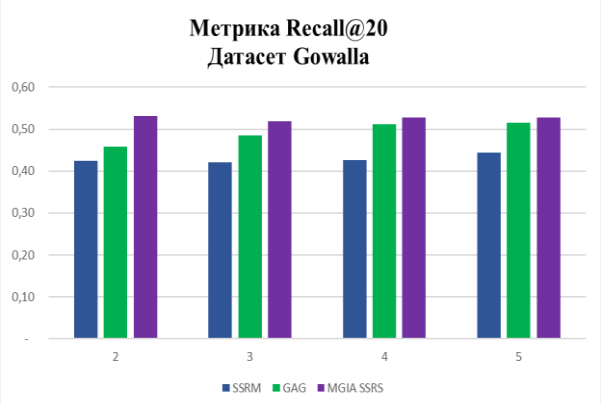

Рис. 16. Эффективность алгоритмов

F. Выводы по результатам сравнительного анализа SSRM, GAG, MGIA SSRS

Обобщенные результаты сравнения SSRM, GAG и MGIA SSRS приведены в таб. 5.

Также по результатам проведенного анализа необходимо отметить общую проблему, ранее уже обозначенную в научном сообществе еще в 2019 году [11], но по-прежнему сохраняющую свою актуальность, а именно невозможность воспроизвести результаты, представленные авторами работ в своих статьях. Из трех рассмотренных работ, только авторы GAG выложили в открытый доступ исходный код реализации своей модели. В связи с чем, отсутствует возможность произвести дополнительное экспериментальное исследование и сравнение результатов на других датасетах, с использованием других метрик и т.д.

VI. Общие (базовые) рекомендации по построению (выбору) архитектуры, алгоритмам и сценарию работы рекомендательных систем на основе сессий для потокового сценария использования

Исследование и анализ передовых алгоритмов рекомендаций на основе сессий для потокового сценария использования SSRM, GAG, MGIA SSRS позволяет разработать конкретные общие (базовые) рекомендации по построению архитектуры, алгоритмам и сценарию работы рекомендательных систем на основе сессий в зависимости от конкретных внешних условий.

В рамках настоящей работы в качестве внешних условий рассматриваются характеристики внешнего потока данных, поступающего на вход рекомендательной системы, а именно непрерывного, высокоскоростного и высокообъемного потока данных.

A. Рекомендации по сценарию работы рекомендательной системы

В сценарии работы рекомендательной системы рекомендуется предусмотреть два параллельно выполняющихся процесса:

1. Процесс генерирования рекомендаций текущей рекомендательной моделью;
2. Процесс обновления рекомендательной модели.

Разделение данных процессов позволяет производить непрерывную обработку поступающего потока данных и генерацию рекомендаций текущей рекомендательной моделью, и, одновременно, на регулярной основе производить обновление рекомендательной модели в целях учета ей динамически происходящих изменений интересов пользователей и предлагаемого в системе контента.

Перед первым запуском рекомендательной системы рекомендательную модель необходимо обучить на основе имеющихся исторических данных.

Обновление (дообучение) рекомендательной модели рекомендуется производить на основе специально отобранной выборки из новых и исторических данных.

Использование выборки данных позволяет не расходовать ограниченные аппаратные ресурсы для

сохранения всего объема новых и исторических данных, и увеличить скорость обработки данных и, соответственно, скорость обновления рекомендательной модели.

B. Рекомендации по типовой общей архитектуре решения

Общую архитектуру рекомендательной системы рекомендуется разделять на два функциональных блока (рис. 17):

1) Блок генерирования рекомендаций – данный блок отвечает за поддержку работы и обновление рекомендательной модели системы.

На вход данного блока поступает:

− непрерывно – поток новых сессий пользователей;

− с заданной периодичностью – выборка информативных сессий для обновления модели.

Периодичность обновления модели для конкретной рекомендательной системы рекомендуется подбирать опытным путем, исходя из поиска баланса между актуальностью рекомендательной модели и вычислительной нагрузкой на систему.

На выходе данного блока (для каждой сессии из потока): список персонализированных рекомендаций для пользователя сессии.

Таб.5. Обобщенные результаты сравнения SSRM, GAG, MGIA SSRS

№	Характеристика	Алгоритм		
		SSRM	**GAG**	**MGIA SSRS**
1	Год выпуска статьи	2019	2020	2022
2	Решаемая задача	Рекомендация следующего взаимодействия в текущей сессии		
3	Входные данные	Комбинация данных, известных о текущей сессии, и исторических данных		
4	Классификация рассматриваемых сессий	• упорядоченные, персонализированные сессии, с одним типом доступных действий; • короткие, средние и длинные сессий – за исключением аномальных сессий, включающих только 1 или более 20 взаимодействий; • структура сессионных данных – многоуровневая.		
5	**Архитектурные особенности**			
5.1	Этапы обучения рекомендательной модели	2 этапа (offline и online)		
5.2	Принцип хранения исторических данных	Выборочное сохранение данных во временном хранилище (резервуаре)		
5.3	Алгоритм отбора данных для сохранения в резервуаре	Random Sampling with a Reservoir		
5.4	Принцип отбора данных для online обучения модели	Отбор наиболее информативных сессий из новых и исторических данных		
5.5	Алгоритм отбора данных для online обучения	Active Sampling Strategy (основан на вычислении ранга сессии с использованием метода матричной факторизации)	Расстояние Вассерштейна между генерируемым моделью списком рекомендаций и реальным взаимодействием пользователя	
5.6	Основные алгоритмы, используемые в рекомендательной модели	• *Gated Recurrent Unit* • *Matrix Factorization*	*Graph Convolutional Networks*	*Gated Graph Neuron Network*
6	**Значения ключевых параметров и условий, при которых авторы проводили обучение и оценку эффективности своих алгоритмов**			
6.1	Функция потерь	перекрестная энтропия (*cross-entropy*)		
6.2	Использование метода увеличения обучающих данных	использовался (*data augmentation technique*)	не указано	не указано
6.3	Использование метода dropout (для избегания переобучения модели)	использовался (dropout rate: 0,25)	не указано	не указано
6.4	Алгоритм обучения	Truncated Backpropagation-Through-Time: • Time steps: 19; • Batch size: 512.	Backpropagation: • Batch size: 100.	Backpropagation
6.5	Метод оптимизации	ADAM (начальная скорость обучения 0,001)	ADAM (начальная скорость обучения 0,003)	ADAM (начальная скорость обучения 0,002)
6.6	Датасеты, использованные авторами для оценки модели	•*Lastfm;* •*Gowalla.*	•*Lastfm;* •*Gowalla.*	•*Lastfm;* •*Gowalla;* •*Tmall.*

6.7	Предобработка датасетов	Производилась		
6.8	Размер датасета после предобработки	• $D^{processed}_{\text{Lastfm}}$: 298 919 сессий. • $D^{processed}_{\text{Gowalla}}$: 198 608 сессий.	• $D^{processed}_{\text{Lastfm}}$: 298 919 сессий. • $D^{processed}_{\text{Gowalla}}$: 198 608 сессий.	• $D^{processed}_{\text{Lastfm}}$: 298 919 сессий. • $D^{processed}_{\text{Gowalla}}$: 435 722 сессий. • $D^{processed}_{\text{Tmall}}$: 962 384 или 809 993 сессии (в статье указаны противоречивые сведения)
6.9	Алгоритм эмуляция потокового сценария	Авторами SSRM, GAG и MGIA SSRS использовался одинаковый алгоритм для эмуляции потокового сценария		
6.10	Временной интервал между поступлением блоков новых данных (в online обучении)	$\frac{\lvert C^{new}\rvert}{2}$ итераций для каждого множества $C^{new} \cup C^{res}$ - задан в итерациях (а не в единицах времени), чтобы исключить влияние конфигурации оборудования на скорость вычислений		
6.11	Используемые метрики	• Recall@K; • MRR@K.		
6.12	Алгоритмы рекомендаций, с которыми авторы производили сравнение своего алгоритма	• POP; • S-POP; • BPR-MF (модифицированный); • HRNN; • GRU4Rec; • NARM.	• POP; • S-POP; • BPR-MF; (модифицированный); • GRU4Rec; • NARM; • FGNN; • SSRM.	• POP; • S-POP; • BPR-MF; • GRU4Rec; • NARM; • SR-GNN; • FGNN; • SSRM; • GAG.
6.13	Язык, использованный авторами для программной реализации своей модели	Python (в том числе библиотека Theano)	Python	Python (в том числе библиотеки PyTorch, PyTorch-geometric)
6.14	Аппаратные ресурсы, использованные авторами при тестировании своей модели	• Видеокарта: NVidia GTX 1080 Ti GPU; • другие характеристики оборудования не указаны.	не указаны	двухпроцессорный сервер: • Процессор (2 шт.): Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz; • ПЗУ: 256 GB; • Видеокарта: Nvidia GeForce RTX 2080 Ti.
7	**Значение настраиваемых параметров алгоритмов, при которых авторы достигли их максимальной эффективности**			
7.1	Размер резервуара	не указан	$\frac{\lvert D^{processed}\rvert}{100}$	$\frac{\lvert D^{processed}\rvert}{100}$
7.2	Количество слоев для MLP	не предусмотрен	1	1
7.3	Размер вектора скрытого представления	• пользователя: 50; • элемента: 50.	• пользователя: 200; • элемента: 200.	• пользователя: 200; • элемента: 200.

7.4	Весовой коэффициент w, определяющий степень влияния модуля коллаборативной фильтрации на рейтинг элемента	• Lastfm: 0,4; • Gowalla: 0,7.	не предусмотрен

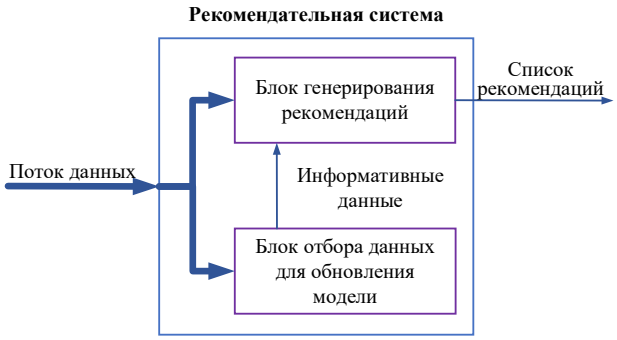

Рис. 17. Рекомендация по типовой общей архитектуре решения

2) Блок отбора данных для обновления модели – данный блок отвечает за отбор данных для обновления рекомендательной модели.

C. Рекомендации по архитектуре и алгоритмам рекомендательной модели

Результаты сравнения эффективности рекомендаций SSRM, GAG и MGIA SSRS показывают, что эффективность рекомендаций растет с увеличением числа факторов (информационных блоков), учитываемых в рекомендательной модели.

В этой связи при построении архитектуры рекомендательной модели рекомендуется исходить из возможности учета максимального количества факторов (информационных блоков), учитываемых рекомендательной моделью с учетом имеющихся ресурсных ограничений эксплуатирующей систему организации.

Соответственно, при выборе алгоритмов для рекомендательной модели необходимо ориентироваться на использование алгоритмов (композиции алгоритмов) позволяющих корректно учесть и закодировать в единое представление все блоки информации, которые планировалось учесть в рекомендательной модели.

Например, как показали авторы GAG и MGIA SSRS, хорошим выбором может быть использование подходов на основе графовых нейронных сетей, которые позволяют учесть сложные зависимости между пользователями и элементами.

D. Рекомендации по архитектуре и алгоритму работы блока отбора данных для обновления модели

Архитектуру блока отбора данных для обновления рекомендательной модели рекомендуется разделять на два функциональных модуля (рис. 18):

1) Модуль отбора сессий – данный модуль отвечает за отбор сессий для обновления рекомендательной модели.

На вход модуля отбора сессий поступает комбинация исторических данных, сохраненных в резервуаре, и поток новых данных. На выходе модуль выдает данные, отобранные для обновления модели.

В качестве критерия для отбора данных рекомендуется использовать информативность сессий, где под информативностью сессий понимается низкая способность текущей рекомендательной модели генерировать точные рекомендации для таких сессий.

В выборку информативных сессий рекомендуется сразу включать сессии, которые содержат новых пользователей или элементы, которые ранее не изучались моделью.

2) Резервуар – данный модуль отвечает за хранение и обновление выборки исторических данных.

На вход резервуара поступает поток новых данных (после прохождения его через модуль отбора сессий), на выходе модуль выдает выборку исторических данных.

Для отбора (из потока новых данных) сессий, сохраняемых в резервуаре, рекомендуется использовать алгоритм *Random Sampling with a Reservoir* [10], который обеспечивает случайную выборку из всего потока данных.

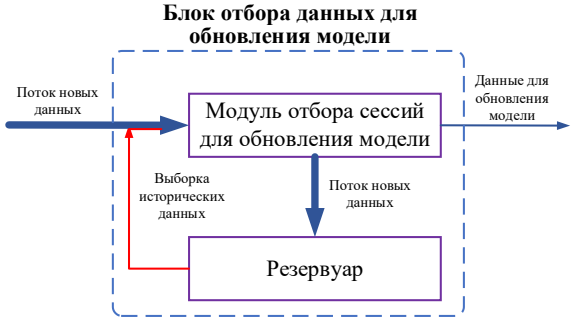

Рис. 18. Рекомендация по типовой архитектуре блока отбора данных для обновления рекомендательной модели

Благодарности

Работы в области рекомендательных систем для сессий были инспирированы работами В.П. Куприяновского по торговым системам [12, 13]. Другие материалы по данной тематике см. в работах [3, 14].

Библиография

[1] Nidhi Arora, Daniel Ensslen, Lars Fiedler, Wei Wei Liu, Kelsey Robinson, Eli Stein, Gustavo Schüler. “The value of getting personalization right - or wrong - is multiplying”. https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying#/. Дата обращения: 10.04.2023.

[2] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, Jae Kyeong Kim. A Literature Review and Classification of Recommender Systems on Academic Journals. // Journal of Intelligence and Information Systems. 2011.

[3] Якупов Д., Намиот Д. “Рекомендательные системы на основе сессий – модели и задачи”. В журнале INJOIT, издательство Лаборатория ОИТ факультета ВМК МГУ им. М.В. Ломоносова (Москва), 2022 г., том 10, № 7, с. 128-155.

[4] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung, “Streaming session-based recommendation,” in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1569–1577.

[5] R. Qiu, H. Yin, Z. Huang, and T. Chen, “Gag: Global attributed graph neural network for streaming session-based recommendation,” in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 669–678.

[6] Z. Yin, K. Han, P. Wang and H. Hu, "Multi Global Information Assisted Streaming Session-Based Recommendation System," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3199373.

[7] S. Latifi, D. Jannach. “Streaming Session-Based Recommendation: When Graph Neural Networks meet the Neighborhood”. In Sixteenth

ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3523227.3548485.

[8] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. DLRS (2016).

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[10] Jeffrey S Vitter. Random sampling with a reservoir. ACM Trans. Math. Software (1985).

[11] Maurizio Ferrari Dacrema, Paolo Cremonesi, and DietmarJannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark. ACM, New York, NY, USA

[12] Куприяновский, В. П., et al. "Розничная торговля в цифровой экономике." International Journal of Open Information Technologies 4.7 (2016): 1-12.

[13] Куприяновская, Ю. В., et al. "Умный контейнер, умный порт, BIM, Интернет Вещей и блокчейн в цифровой системе мировой торговли." International Journal of Open Information Technologies 6.3 (2018): 49-94.

[14] Ninichuk, Marina, and Dmitry Namiot. "Survey On Methods For Building Session-Based Recommender Systems." International Journal of Open Information Technologies 11.5 (2023): 22-32.

Comparative analysis of modern algorithms for generating recommendations based on sessions, in relation to the streaming usage scenario (Streaming Session-based Recommendation)

Dmitry Yakupov

Abstract — **Recommendation systems are actively used in many areas of modern life (e-commerce, banking, communications, entertainment, etc.) and are of great importance for businesses and consumers. A separate class of these systems are session-based recommendation systems, the key feature of which is the generation of recommendations based on the user's recent actions in the system (his current session), the analysis of which allows to identify the current intentions and interests of the user. Especially relevant is the use of session-based recommendation systems in a streaming usage scenario (Streaming Session-based Recommender Systems), for example, on entertainment content platforms, marketplaces, etc. A distinctive feature of the streaming scenario is the continuous, high-volume and high-speed nature of the receipt of new data that needs to be processed in real time. In this paper, a comparative analysis of modern algorithms of session-based recommendation systems for a streaming usage scenario is carried out: *Streaming Session-based Recommendation Machine, Global Attributed Graph Neural Network, Multi Global Information Assisted Streaming Session-Based Recommendation System*, the general principles of building these systems, their main differences are highlighted, advantages and disadvantages are considered. Based on the research and analysis of these systems, basic (typical) recommendations for the construction of architecture, algorithms and the scenario of the work of recommendation systems based on sessions, depending on external conditions, have been developed.**

Keywords— Streaming Session-based Recommendation.

REFERENCES

[1] Nidhi Arora, Daniel Ensslen, Lars Fiedler, Wei Wei Liu, Kelsey Robinson, Eli Stein, Gustavo Schüler. "The value of getting personalization right - or wrong - is multiplying". https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying#/. Retrieved: 10.04.2023.

[2] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, Jae Kyeong Kim. A Literature Review and Classification of Recommender Systems on Academic Journals. // Journal of Intelligence and Information Systems. 2011.

[3] Dmitry Yakupov, Dmitry Namiot. Session Based Recommender Systems – Models and Tasks. // International Journal of Open Information Technologies ISSN: 2307-8162 vol. 10, no. 7, 2022, p. 128-155.

[4] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung, "Streaming session-based recommendation," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1569–1577.

[5] R. Qiu, H. Yin, Z. Huang, and T. Chen, "Gag: Global attributed graph neural network for streaming session-based recommendation," in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 669–678.

[6] Z. Yin, K. Han, P. Wang and H. Hu, "Multi Global Information Assisted Streaming Session-Based Recommendation System," in IEEE Transactions on Knowledge and Data Engineering, 2022, doi: 10.1109/TKDE.2022.3199373.

[7] S. Latifi, D. Jannach. "Streaming Session-Based Recommendation: When Graph Neural Networks meet the Neighborhood". In Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3523227.3548485.

[8] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. DLRS (2016).

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[10] Jeffrey S Vitter. Random sampling with a reservoir. ACM Trans. Math. Software (1985).

[11] Maurizio Ferrari Dacrema, Paolo Cremonesi, and DietmarJannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark. ACM, New York, NY, USA

[12] Kuprijanovskij, V. P., et al. "Roznichnaja torgovlja v cifrovoj jekonomike." International Journal of Open Information Technologies 4.7 (2016): 1-12.

[13] Kuprijanovskaja, Ju. V., et al. "Umnyj kontejner, umnyj port, BIM, Internet Veshhej i blokchejn v cifrovoj sisteme mirovoj torgovli." International Journal of Open Information Technologies 6.3 (2018): 49-94.

[14] Ninichuk, Marina, and Dmitry Namiot. "Survey On Methods For Building Session-Based Recommender Systems." International Journal of Open Information Technologies 11.5 (2023): 22-32.