Устойчивость сессионных рекомендательных систем к атакам отравления обучающей выборки

А.С. Хитрова, О.Р. Лапонина

Аннотация **– Статья посвящена оценке устойчивости сессионных рекомендательных систем к атакам отравления обучающей выборки синтетическими сессиями. Реальные журналы событий неизбежно содержат записи, сгенерированные автоматизированными агентами, что снижает качество обучаемых моделей. Для контролируемого воспроизведения такого загрязнения разработан программный комплекс на языке Python, включающий четыре класса генераторов синтетических сессий: навигационный граф (марковская цепь первого порядка), скрытую марковскую модель (CMM), рекуррентный генератор на основе сети Элмана и трансформерный генератор. Все генераторы реализуют единый интерфейс и могут работать с произвольными наборами событийных журналов.**

Модуль целевых атак реализует лучевой поиск синтетических сессий с заданным финальным элементом в режиме чёрного ящика, не требуя доступа к параметрам атакуемой модели. После отравления целевая модель переобучается на D_train ∪ P с теми же гиперпараметрами. Тестовая выборка остаётся неизменной. Критерий успеха атаки – деградация качества на чистой тестовой выборке.

Экспериментальная оценка проведена на наборах данных YooChoose и Last.fm. В качестве атакуемого рекомендера используется модель на основе усреднённых эмбеддингов (mean pooled embeddings): для каждого префикса сессии вычисляется среднее арифметическое эмбеддингов элементов, затем применяется линейный слой для вычисления логитов над словарём.

При доле отравляющих сессий 2% деградация HR@10 составляет 0,4–0,9п.п.; при 5% – 1,2–3,1п.п.; при 20% деградация нарастает нелинейно и достигает 5,9–7,9 п.п. Нейросетевые генераторы порождают более разнообразные синтетические сессии по сравнению с марковскими моделями. Трансформерный генератор может комбинировать переходы, не наблюдавшиеся в совокупности, что расширяет охват пространства признаков. Именно разнообразие порождаемых путей усиливает смещение обучающего распределения в сторону целевого элемента. Трансформерный генератор вызывает наибольшую деградацию среди четырёх классов, навигационный граф – наименьшую. Полученные результаты позволяют разработчикам рекомендательных систем обоснованно оценивать робастность моделей до ввода в эксплуатацию.

Ключевые слова **– состязательные атаки, отравление обучающей выборки, рекомендательные системы на основе сессий, генерация синтетических сессий, устойчивость модели, марковские цепи, скрытая марковская модель, рекуррентные нейронные сети, трансформер, наборы данных YooChoose, Last.fm.**

I. Введение

Рекомендательные системы на основе пользовательских сессий широко применяются в интернет-магазинах, стриминговых платформах и новостных агрегаторах [1, 2]. В отличие от классических коллаборативных фильтров, сессионные системы работают анонимно: единственный источник информации о намерении пользователя – последовательность событий текущего сеанса [3,4]. Граница сессии определяется идентификатором сеанса или порогом бездействия (обычно 15–30 минут), что делает такие системы особенно чувствительными к качеству входного потока событий.

Реальные журналы событий содержат записи, не отражающие подлинных предпочтений: сессии парсеров, мониторинговых ботов, случайные клики. Доля автоматизированного трафика достигает 20 – 45 % веб-журналов [5]. Даже после стандартной фильтрации в обучающей выборке остаётся шумовая составляющая, которая ухудшает качество обучаемых представлений. Более того, злоумышленник может целенаправленно вносить синтетические сессии для продвижения нужного элемента [6] – атака отравления обучающей выборки (training data poisoning). Такие атаки особенно опасны тем, что реализуются без прямого доступа к параметрам модели: достаточно создать поддельные профили или искусственные журналы кликов, которые попадут в открытый поток данных.

Существующие работы по атакам на нейронные рекомендательные системы в основном рассматривают сценарий белого ящика: атакующий имеет полный доступ к параметрам модели и вычисляет градиенты [7]. Методы чёрного ящика [8] требуют многочисленных запросов к рекомендеру или суррогатных моделей с разделяемой архитектурой. Ни тот, ни другой подход не применимы непосредственно к задаче практической оценки устойчивости: разработчик хочет до ввода системы в эксплуатацию проверить, насколько она уязвима при реалистичных допущениях.

Мы предлагаем инструментарий, в котором атакующий обучает собственную генеративную модель на открытых обучающих данных и генерирует синтетические сессии, гарантированно заканчивающиеся целевым элементом. Для управляемой генерации применяется лучевой поиск поверх любого из четырёх реализованных генераторов. Весь процесс не требует доступа к параметрам атакуемого рекомендера – это

атака в режиме чёрного ящика в чистом виде. Такой сценарий воспроизводит реальную угрозу: конкурент или недобросовестный продавец знает ассортимент, но не имеет доступа к системе.

Вклад работы:

1) Реализованы четыре класса генераторов синтетических сессий с единым интерфейсом fit/sample/continue_session: навигационный граф (марковская цепь), скрытая марковская модель, RNN-генератор и трансформерный генератор.
2) Реализован модуль целевых атак лучевым поиском в режиме чёрного ящика – без доступа к параметрам атакуемой модели.
3) Проведена экспериментальная оценка на наборах данных YooChoose и Last.fm при семи уровнях отравления (0–20%); установлена нелинейная зависимость деградации метрики HR@10 от доли синтетических сессий.

Раздел 2 формализует задачу предсказания следующего события и обзор методов. Раздел 3 классифицирует атаки на рекомендательные системы. Раздел 4 формализует задачу отравления и лучевой поиск. Раздел 5 описывает программный инструментарий. Раздел 6 представляет экспериментальную оценку. Разделы 7 и 8 содержат обсуждение и заключение.

II. Методы предсказания следующего события

A. Формализация задачи

Пользовательская сессия – упорядоченная по времени последовательность взаимодействий $S = \langle e_1, e_2, \dots, e_n \rangle, e_i \in I$, где I – конечное множество возможных элементов. Граница сессии определяется идентификатором сеанса или порогом бездействия (обычно 15–30 минут).

Задача предсказания следующего события (next-item prediction): дан префикс $S_{1:t} = \langle e_1, \dots, e_t \rangle$, вычислить

$$\hat{p}(e_{t+1} \mid e_1, \dots, e_t) \quad (1)$$

и вернуть список из K наиболее вероятных следующих элементов.

Стандартная метрика оценки – HR@K (Hit Rate): доля тестовых запросов, в которых истинный следующий элемент попал в топ-K рекомендаций [3,9]. Дополнительно применяются MRR@K и NDCG@K, учитывающие позицию истинного элемента в ранжированном списке.

B. Статистические подходы

Марковские цепи первого порядка моделируют вероятности переходов $p(e_{t+1} \vee e_t)$; оценка – нормализованные счётчики по обучающим данным. Метод быстр и интерпретируем, но не учитывает предысторию длиннее одного шага. Для уменьшения разреженности счётчиков применяются сглаживание Лапласа и порог отсечения редких переходов.

Скрытая марковская модель (СММ) вводит K латентных состояний и описывается тремя параметрами: $\pi \in \mathbb{R}^K$ (начальное распределение), $A \in \mathbb{R}^{K \times K}$ (переходы состояний), $B \in \mathbb{R}^{K \times |I|}$ (матрица эмиссий). Параметры оцениваются алгоритмом Баума–Уэлша [10]. Ключевое свойство СММ – обобщение редких переходов через разделение наблюдаемого и латентного пространств: событие, ни разу не наблюдавшееся рядом с i^*, может быть порождено через латентное состояние, совместное с несколькими соседями i^*.

FPMC (Factorizing Personalized Markov Chains) [11] объединяет матричную факторизацию с марковскими цепями через разложение тензора переходов на произведение векторов пользователя и элементов, что позволяет учитывать индивидуальные предпочтения при сохранении марковского допущения. В анонимных сессиях без идентификатора пользователя FPMC деградирует до обычной матрицы переходов.

C. Рекуррентные нейронные сети

GRU4Rec [12,13] применяет GRU-ячейки для кодирования состояния сессии. Ключевая формула рекуррентного обновления:

$$\begin{aligned} \mathbf{z}_t &= \sigma(\mathbf{W}_z \mathbf{e}_t + \mathbf{U}_z \mathbf{h}_{t-1}), \\ \mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{e}_t + \mathbf{U}_r \mathbf{h}_{t-1}), \\ \mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tanh(\mathbf{W}_h \mathbf{e}_t + \mathbf{U}_h (\mathbf{r}_t \odot \mathbf{h}_{t-1})), \end{aligned} \quad (2)$$

где e_t– эмбеддинг t-го элемента. Предсказание строится линейным слоем по h_t. Обучение – кросс-энтропия с выборочным softmax для эффективной работы с большими словарями.

NARM (Neural Attentive Recommendation Machine) [14] добавляет механизм внимания поверх GRU-скрытых состояний: для каждого элемента сессии вычисляется вес релевантности относительно финального скрытого состояния, что позволяет выделить наиболее значимые для текущего запроса взаимодействия. STAMP (Short-Term Attention/Memory Priority Model) [15] разделяет долгосрочный интерес (среднее всех эмбеддингов) и краткосрочный (последнее событие), комбинируя их через механизм внимания. Такое разделение особенно эффективно в длинных сессиях, где пользователь меняет намерение в процессе просмотра.

D. Трансформерные архитектуры

SASRec (Self-Attentive Sequential Recommendation) [16] применяет причинный (однонаправленный) трансформер с позиционными эмбеддингами: стек блоков самовнимания с маскированием будущих позиций. BERT4Rec [17] использует двунаправленное предобучение на задаче маскирования случайных позиций. SR-GNN [18] моделирует сессию как ориентированный граф и применяет графовые нейронные сети для вычисления представлений элементов [19].

Трансформеры достигают лучших результатов на большинстве бенчмарков, однако требуют больших вычислительных ресурсов и данных. SR-GNN [18] моделирует сессию как ориентированный граф и применяет графовые нейронные сети: переходы между элементами образуют рёбра, а узловые представления вычисляются посредством нескольких итераций передачи сообщений [19]. Это позволяет захватывать не только порядковые, но и структурные зависимости в

сессии.

Систематическое сравнение методов [3] показывает, что простые базовые алгоритмы (k-NN по последнему событию, BPR-MF) часто конкурентоспособны с глубокими моделями при корректном протоколе оценки [4]. Это наблюдение важно и для данной работы: выбор упрощённой базовой модели для атакуемой рекомендательной системы методически корректен, поскольку цель – измерить изменение качества при отравлении, а не абсолютный уровень метрик.

III. Состязательные атаки на рекомендательные системы

A. Классификация атак на рекомендательные системы

Атаки систематизируются по трём критериям [20,21]. По этапу: атаки уклонения (evasion) – возмущение подаётся во время вывода, параметры модели не меняются; атаки отравления (poisoning) [6] – манипуляция на этапе обучения через внедрение синтетических примеров. По осведомлённости: белый ящик – полный доступ к архитектуре и параметрам (FGSM [22], PGD [23], C&W [24]); чёрный ящик [8] – атакующий может лишь запрашивать предсказания. По цели: целевая (продвижение конкретного элемента) или нецелевая (снижение общего качества). Обзоры представлены в [25,26].

B. Инъекции профилей

Lam & Riedl [27] ввели понятие shilling attack: злоумышленник регистрирует фиктивные профили для продвижения товаров. Структура профиля ũ [28]: целевые товары I_T (экстремальные оценки r_{max}), заполнители I_F (правдоподобные оценки), избранные I_S (наиболее популярные). Эффект объясняется механизмом KNN-коллаборативной фильтрации:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}(u)} \mathrm{sim}(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in \mathcal{N}(u)} |\mathrm{sim}(u,v)|}, \quad (3)$$

фиктивный профиль, искусственно близкий к реальным пользователям, включается в окружение соседей $\mathcal{N}(u)$ и вызывает смещение [29].

C. Отравление нейронных рекомендателей

Fang et al. [7] формулируют атаку как двухуровневую оптимизацию:

$$\min_{\mathcal{P}} \mathcal{L}_{\mathrm{attack}}(\theta^*(\mathcal{P})), \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{\mathrm{train}}(\theta;\ \mathcal{D} \cup \mathcal{P}). \quad (4)$$

Решение требует вычисления градиентов атакуемой модели (белый ящик).

D. Атаки чёрного ящика на сессионные системы

Атаки без доступа к параметрам: Yue et al. [8] – суррогатная модель, Lin et al. [30] – эволюционные алгоритмы. Работы [31, 32] изучают отравление последовательных рекомендательных систем. Настоящая работа предлагает альтернативу: генератор, обученный на открытых обучающих данных, формирует атакующие сессии методом лучевого поиска – без обращения к параметрам целевой модели.

IV. Постановка задачи отравления

A. Формальное определение

Пусть $D_{train} = \{S_1, \dots, S_N\}$ – обучающая выборка из N сессий; f_θ – рекомендательная модель, обученная на D_{train}; $i^* \in I$ – целевой элемент, ранг которого атакующий стремится повысить.

Задача целевого отравления в режиме чёрного ящика. Атакующий не имеет доступа к параметрам θ и не может вычислять градиенты. Он обучает собственную генеративную модель g_φ на открытых данных D_{train} и генерирует набор $P = \{P_1, \dots, P_M\}$ синтетических сессий, каждая из которых заканчивается элементом i^*.

После отравления целевая модель переобучается на $D_{train} \cup P$ с теми же гиперпараметрами. Тестовая выборка остаётся неизменной. Критерий успеха атаки – деградация качества на чистой тестовой выборке:

$$\mathrm{HR@}K(f_{\theta^*_{\mathrm{poison}}}) < \mathrm{HR@}K(f_{\theta^*_{\mathrm{clean}}}). \quad (5)$$

Доля отравляющих сессий $\alpha = \frac{M}{(N+M)}$ – управляемый экспериментальный параметр. Целевой элемент i^* выбирается случайно (фиксированный seed) из множества кортежей, наблюдавшихся в позиции последнего события обучающих сессий.

B. Лучевой поиск целевых сессий

Для генерации сессий, гарантированно заканчивающихся i^*, применяется лучевой поиск (beam search) шириной B поверх генератора g_φ. Алгоритм поддерживает B частичных гипотез $\{(h_b, s_b)\}^B{}_{b=1}$, где h_b – текущая последовательность элементов, s_b – накопленный логарифм вероятности.

На каждом шаге t : для каждой гипотезы h_b вызывается метод continue_session(h_b) генератора g_φ, возвращающий вектор вероятностей $q_b \in [0,1]^{|I|}$. Из $B|I|$кандидатов ($h_b \circ e, s_b + log q_b[e]$) отбираются топ-$B$ по накопленному логарифму вероятности. Гипотеза, завершившаяся i^*, переносится в результирующее множество $\mathcal{R}$ и не расширяется далее. Алгоритм завершается при $|\mathcal{R}| = M$ или достижении максимальной длины t_{max}.

Ключевое свойство: алгоритм работает исключительно через метод continue_session генератора g_φ– без обращения к параметрам θ атакуемой модели f_θ.

V. Программный инструментарий

A. Общая архитектура

Комплекс реализован на Python 3.10+ с PyTorch-бэкендом и структурирован в три пакета: utils (подготовка данных), generators (генераторы сессий), attacks (модуль атак). Центральный объект – pandas.DataFrame, строка = одно событие журнала с обязательными столбцами идентификатора сессии и временной метки. Событие представляется кортежем признаков фиксированной размерности, трактуемым как единый дискретный токен словаря.

Все четыре генератора реализуют унифицированный интерфейс: fit(events_df,...), sample(n,...), continue_session(prefix,...). Контракт метода continue_session строго определён: принимает список токенов (префикс сессии), возвращает вектор вероятностей над словарём $q \in [0,1]^{|I|}$. Именно этот метод является точкой входа лучевого поиска. Единообразие интерфейса позволяет модулю атак работать с любым генератором без знания деталей его реализации – принцип открытости/закрытости в применении к атаке.

Таблица 1. Сравнение генераторов по ключевым характеристикам

Генератор	Порядок	Параметры	Разнообр.
Навигац. граф	1	$O(\|\mathcal{I}\|^2)$	низкое
CMM	K	$O(K\|\mathcal{I}\|)$	среднее
RNN (LSTM)	∞	$O(d\|\mathcal{I}\|)$	высокое
Трансформер	∞	$O(d\|\mathcal{I}\|)$	высокое

B. Вспомогательные модули (utils)

Функция prepare_events_dataframe нормализует входной DataFrame: проверяет наличие обязательных столбцов, приводит временные метки к Unix-секундам, опционально сортирует события внутри сессий. Функция continuation_timestamps_strictly_increasing гарантирует монотонное возрастание меток при синтетическом продолжении сессии; при необходимости добавляет субсекундное приращение. Модуль online_stats накапливает оценки параметров гауссовских распределений межсобытийных интервалов по алгоритму Велфорда: среднее и дисперсия обновляются при каждом наблюдении без хранения полной истории.

C. Навигационный граф (марковская цепь)

Ориентированный взвешенный граф: вершина – уникальный кортеж признаков, ребро (u, v) с весом w_{uv}– частота наблюдения перехода $u \to v$ в обучающих сессиях. Модель формально эквивалентна марковской цепи первого порядка.

При вызове fit: строится словарь кортежей, подсчитываются эмпирические частоты стартовых вершин и переходов, выполняется нормализация в вероятности. Структуры хранятся как тензоры Py Torch. При sample: для каждой сессии сэмплируется стартовая вершина из распределения стартов (torch.multinomial), затем на каждом шаге – следующая вершина из вектора исходящих переходов. Процесс завершается при достижении максимальной длины или срабатывании стохастического условия ранней остановки.

Достоинства. O(N) обучение (один проход), прозрачность параметров, низкие требования к памяти. Ограничение: память первого порядка – вероятность следующего события не зависит от предыстории за пределами последнего шага.

D. Скрытая марковская модель (CMM)

Модель с K скрытыми состояниями (гиперпараметр): параметры хранятся как тензоры PyTorch – $\pi \in \mathbb{R}^K$ (начальное распределение), $A \in \mathbb{R}^{K\times K}$ (матрица переходов), $B \in \mathbb{R}^{K\times|I|}$ (матрица эмиссий). Обучение реализовано алгоритмом Баума–Уэлша с батчевой тензорной обработкой всех сессий одновременно и log space арифметикой для устойчивости [10]. Скрытые состояния позволяют обобщать редкие переходы через разделение наблюдаемого и латентного пространств, порождая более разнообразные синтетические сессии по сравнению с навигационным графом.

E. Рекуррентный генератор (RNN)

Языковая модель над словарём кортежей. Архитектура: слой эмбеддингов $E \in \mathbb{R}^{(|I|+1)\times d}$, два LSTM-слоя [33, 34], линейный слой проекции на логиты словаря. Обучение ведётся минимизацией кросс-энтропии предсказания следующего токена. Метод continue_session подаёт заданный префикс в LSTM, получает итоговое скрытое состояние и сэмплирует продолжение авторегрессионно с опциональной температурой. Механизм LSTM-ворот адресует проблему затухающего градиента, позволяя улавливать зависимости на большей дистанции, чем марковские модели.

F. Трансформерный генератор

Причинный TransformerEncoder с маскированием будущих позиций [19], аналогично SASRec [16]: позиционные эмбеддинги суммируются с токенными, стек блоков многоголового самовнимания с причинной маской формирует контекстные представления. Предсказание строится по скрытому состоянию последнего токена.

Преимущество. Прямое внимание к произвольно удалённым позициям сессии. Ограничение: O(n^2) сложность самовнимания по длине и более высокие требования к объёму обучающих данных по сравнению с RNN.

G. Модуль атак (target_last_event_attack)

Функция принимает обученный генератор, целевой элемент i^*, требуемое число сессий M и ширину луча B. Реализует алгоритм лучевого поиска из раздела 4: расширяет гипотезы через continue_session, отбирает топ-B кандидатов, переносит завершённые в результирующее множество. Работает исключительно через API генератора.

Листинг 1. Использование модуля атак для отравления данных

```
gen = TransformerSessionGenerator(n_layers=2,
        d_model=64, n_heads=2)
gen.fit(train_df, session_col="session_id",
        feature_col="item_id")
poison_df = target_last_event_attack(
    generator=gen, target_item=i_star,
    n_sessions=600, beam_width=5)
poisoned = pd.concat([train_df, poison_df])
model.fit(poisoned)
```

VI. Экспериментальная оценка

A. Методология

Разбиение данных идентификаторов сессий. Множество уникальных перемешивалось с фиксированным seed и делилось 80/20 на обучающую и

тестовую части. Разбиение по идентификаторам гарантирует полную диссоциацию: ни одна тестовая сессия не присутствует в обучающей выборке.

Базовая рекомендательная модель. В качестве атакуемого рекомендера используется модель на основе усреднённых эмбеддингов (mean pooled embeddings): для каждого префикса сессии вычисляется среднее арифметическое эмбеддингов элементов, затем применяется линейный слой для вычисления логитов над словарём. Оптимизатор SGD, размер батча 1024, скорость обучения $2 * 10^{-2}$, 4 эпохи, функция потерь – кросс-энтропия. Выбор модели мотивирован задачей исследования: цель – изучить устойчивость процесса обучения, а не сравнивать архитектуры рекомендательных систем; простая модель обеспечивает воспроизводимость и минимальное время переобучения.

Целевой элемент i^* выбирается случайно (фиксированный seed) множества кортежей, наблюдавшихся в позиции последнего события обучающих сессий; выбор произвольного i^* исключает методически некорректный сценарий с изначально популярным элементом. Синтетические сессии $\mathcal{P}$ генерируются генератором, обученным на тех же данных. Объединённая выборка $D_{train} \cup P$ используется для повторного обучения с прежними гиперпараметрами. Метрика - HR@10 на чистой тестовой выборке; $\Delta = HR_{poison} - HR_{clean}$.

B. *Наборы данных*

Таблица 2. Параметры наборов данных

Параметр	YooChoose	Last.fm
Обучающих сессий	30 251	≈1 600
Тестовых сессий	7 563	≈400
Размер словаря	10 811	≈8 300
Training pairs	89 787	≈5 200
HR@10 (clean)	0,5859	0,3821

YooChoose – датасет кликов электронной коммерции; используется 150000 строк. Событие – пара (item_id, category). Целевой элемент i^* = (item_id = 214839313, category = ‘0’).

Last.fm – история прослушиваний музыки; используется 80000 строк. Событие – тройка (artist, track, album). Более сложный словарь и меньший объём обучающей выборки делают эту задачу более уязвимой к отравлению.

C. *Результаты на YooChoose*

Таблица 3. ΔHR@10 после отравления. YooChoose, HR@10 (clean) = 0,5859

Генератор	α=2%	α=3%	α=5%
Навигац. граф	−0,004	−0,007	−0,012
CMM	−0,006	−0,011	−0,019
RNN	−0,005	−0,009	−0,016
Трансформер	−0,007	−0,013	−0,022

При α = 2% деградация HR@10 составляет 0,4–0,7п.п. в зависимости от генератора – незначительный, но устойчивый эффект. При $\alpha = 5\%$ деградация достигает 1,2–2,2п.п. и становится практически значимой. Нейросетевые генераторы (RNN, трансформер) вызывают несколько более выраженную деградацию, что объясняется большей структурной разнообразностью порождаемых ими сессий.

D. *Результаты на Last.fm*

Таблица 4. Δ HR@10 после отравления. Last.fm, HR@10 (clean) = 0,3821

Генератор	α=2%	α=3%	α=5%
Навигац. граф	−0,006	−0,012	−0,021
CMM	−0,008	−0,015	−0,027
RNN	−0,007	−0,014	−0,025
Трансформер	−0,009	−0,017	−0,031

Деградация на Last.fm при тех же уровнях отравления заметно выше: при $\alpha = 5\%$ она достигает 2,1-3,1п.п. против 1,2–2,2п.п. на YooChoose. Причина – меньший объём обучающей выборки (≈1600 пользователей): синтетические сессии составляют значимую долю обучающего контекста и оказывают пропорционально большее влияние на параметры модели. Относительный порядок генераторов совпадает с YooChoose: навигационный граф вызывает наименьшую деградацию, трансформер – наибольшую.

E. *Влияние доли отравления*

Таблица 5. Зависимость HR@10 от доли отравления. RNN-генератор

α	YooChoose		Last.fm	
	HR@10	Δ	HR@10	Δ
0%	0,5859	—	0,3821	—
1%	0,5840	−0,002	0,3791	−0,003
2%	0,5809	−0,005	0,3751	−0,007
3%	0,5769	−0,009	0,3681	−0,014
5%	0,5699	−0,016	0,3571	−0,025
10%	0,5549	−0,031	0,3351	−0,047
20%	0,5269	−0,059	0,3031	−0,079

На обоих датасетах наблюдается монотонный рост деградации с α. При $\alpha = 1\%$ эффект минимален (0,2–0,3п.п.) и может находиться в пределах статистического шума. При $\alpha = 5\%$ деградация уже составляет 1,6–2,5п.п. – существенное снижение качества. Переход $10\% \rightarrow 20\%$ сопровождается нелинейным ускорением: на YooChoose деградация вырастает с 3,1 до 5,9п.п. (прирост в 1,9), на Last.fm – с 4,7 до 7,9п.п. (×1,7). Этот нелинейный эффект указывает на качественное изменение динамики обучения при высоких долях загрязнения: распределение синтетических сессий начинает доминировать над реальным сигналом.

F. *Сравнение генераторов*

Устойчивый порядок по величине отравляющего эффекта на обоих датасетах:
трансформер > CMM > RNN > навигационный граф.

Разрыв между трансформером и навигационным графом при $\alpha = 5\%$ составляет ≈ 1п.п. (YooChoose) и ≈ 1п.п. (Last.fm).

Нейросетевые генераторы порождают более разнообразные синтетические сессии: трансформерный генератор может комбинировать переходы, не наблюдавшиеся в совокупности, что расширяет охват пространства признаков. Именно разнообразие порождаемых путей усиливает смещение обучающего

распределения в сторону i^*.

Навигационный граф, напротив, концентрирует сгенерированные сессии вблизи наиболее популярных переходов, ограничивая степень искажения. Вместе с тем он обеспечивает наибольшую скорость обучения (O(N), один проход по данным) и полную интерпретируемость параметров, что делает его удобным инструментом для быстрой предварительной проверки.

G. *Анализ вычислительной стоимости*

Время обучения генераторов и генерации 600 атакующих сессий (ширина луча $B = 5$) существенно различаются в зависимости от архитектуры. На одном CPU (Apple M2):

Таблица 6. Вычислительная стоимость генераторов. YooChoose, M=600

Генератор	Обучение	Генерация
Навигац. граф	<1 с	<1 с
CMM (K=10)	≈30 с	<5 с
RNN (LSTM)	≈120 с	≈15 с
Трансформер	≈180 с	≈20 с

Для практической оценки уязвимости соотношение “качество атаки / вычислительные затраты” наиболее выгодно у RNN: атака лишь незначительно слабее трансформерной, но обучение в 1,5 раза быстрее. Навигационный граф рекомендуется для первоначальной проверки: несколько секунд обучения и генерации дают воспроизводимую нижнюю оценку уязвимости.

H. *Воспроизводимость*

Все эксперименты параметризованы фиксированным seed: разбиение данных, выбор целевого элемента и инициализация весов нейросетевых генераторов воспроизводимы. Код и конфигурации опубликованы; для воспроизведения таблиц 3–5 достаточно запустить прилагаемые демонстрационные скрипты без дополнительной настройки. Платформенные различия (CPU/GPU, версия PyTorch) не влияют на порядок значений в таблицах благодаря детерминированному режиму генерации сессий.

VII. Обсуждение

A. *Практические пороговые значения*

Результаты экспериментов позволяют выделить три диапазона уязвимости. При $\alpha \leq 2\%$ деградация не превышает 0,7–0,9п.п. на обоих датасетах: эффект измерим, но может находиться в пределах обычных флуктуаций при переобучении и переразбиении. Такой уровень загрязнения соответствует случайному шуму в реальных журналах без целенаправленного вмешательства.

При $\alpha = 5\%$ деградация достигает 1,6–3,1п.п. – это уже практически значимое снижение: рекомендатель, показывающий HR@10 = 0,59 в чистом режиме, при таком отравлении теряет примерно каждый 30-й корректный хит. Для системы с несколькими миллионами сеансов в сутки это измеримо влияет на пользовательский опыт и метрики продаж.

Нелинейный рост при $\alpha > 10\%$ сигнализирует о качественном изменении динамики обучения: синтетические сессии начинают доминировать над реальным сигналом. Переход $10\% \rightarrow 20\%$ сопровождается ускорением деградации в 1,7–1,9 раза, что соответствует режиму насыщения обучающего распределения.

Рекомендация. Системе с достаточным объёмом данных (порядка 30000 сессий и более) следует контролировать долю необычных сессий на уровне ниже 2%; системам с небольшой выборкой (менее 2000 сессий) – ниже 1%.

B. *Выбор генератора для оценки*

Устойчивый порядок по силе отравляющего эффекта: трансформер > CMM > RNN > навигационный граф – воспроизводится на обоих датасетах при всех уровнях α. Разрыв между крайними позициями при $\alpha = 5\%$ составляет ≈ 1п.п. – не катастрофический, но статистически устойчивый.

Причина различий – структурное разнообразие порождаемых сессий. Навигационный граф является марковской цепью первого порядка: вероятность следующего события определяется только текущим, поэтому генератор концентрирует сессии вблизи наиболее популярных переходов. Трансформерный генератор, напротив, обрабатывает весь контекст сессии одновременно и может комбинировать переходы, которые наблюдались в разных частях корпуса, порождая траектории с более высокой вариативностью.

Для консервативной оценки уязвимости (верхняя граница риска) рекомендуется трансформерный генератор. Для быстрой первоначальной проверки достаточно навигационного графа: он даёт нижнюю оценку и обучается за O(N) – один проход по данным.

C. *Уязвимость малых обучающих выборок*

На Last.fm (≈1600 обучающих пользователей) деградация при одинаковых α значительно выше, чем на YooChoose (30251 сессий). При $\alpha = 5\%$: 3,1п.п. против 2,2п.п. для трансформерного генератора. Соотношение масштабируется с объёмом: при меньшем числе реальных сессий каждая синтетическая сессия несёт пропорционально больший вес в оценках параметров модели.

Это наблюдение имеет практическое значение для систем с длинным хвостом сессий: новые или редко встречающиеся пользовательские сегменты уязвимы к отравлению сильнее, чем хорошо представленные в обучающей выборке.

D. *Сравнение с существующими методами*

Градиентные методы [7] (белый ящик) достигают более выраженной деградации при меньших α, однако требуют полного доступа к параметрам и архитектуре атакуемой модели – нереалистичное условие для большинства реальных систем. Суррогатные атаки чёрного ящика [8] предполагают многочисленные запросы к рекомендательной системе, что обнаруживаемо по аномальным паттернам трафика. Предлагаемый подход генерирует атакующие сессии офлайн, исключительно на основе открытых данных, и

не создаёт запросов к целевой системе.

С точки зрения защиты различие принципиально: стандартные методы аудита трафика не позволяют обнаружить атаку, реализованную через генеративную модель, обученную на общедоступных данных. Это подчёркивает необходимость проактивной оценки устойчивости на этапе разработки.

E. Ограничения и открытые вопросы

Атакуемая модель. В экспериментах используется упрощённый рекомендер на усреднённых эмбеддингах; оценка атаки на SASRec, BERT4Rec и GRU4Rec – открытая задача. Можно ожидать, что более сложные модели обладают большей устойчивостью благодаря контекстному вниманию, однако систематической проверки не проводилось.

Целевой элемент. Текущий протокол выбирает i^* случайно из хвоста распределения. Атака на изначально популярный элемент, по-видимому, эффективнее при меньших α, однако такой сценарий методически менее чистый. Обратный вариант – продвижение нового элемента с нулевой историей – представляет отдельный интерес.

Методы защиты. Разработанный инструментарий непосредственно применим для оценки техник защиты: фильтрации аномальных сессий, взвешивания обучающих примеров по доверию, adversarial training. Сравнительное исследование защитных мер – перспективное направление будущих работ.

VIII. Заключение

В работе разработан и реализован программный инструментарий для систематической оценки устойчивости сессионных рекомендательных систем к атакам отравления обучающей выборки. Основные результаты и вклад работы формулируются следующим образом.

Реализация генераторов. Четыре класса References генераторов синтетических сессий с единым API реализованы на Python с PyTorch-бэкендом: навигационный граф (марковская цепь первого порядка), скрытая марковская модель (алгоритм Баума–Уэлша), RNN-генератор (двухслойный LSTM) и трансформерный генератор (причинный Transformer Encoder). Все четыре генератора образуют спектр по сложности и вычислительным требованиям, что позволяет выбрать подходящий инструмент исходя из размера обучающей выборки и доступного времени вычислений.

Модуль атак. Функция target_last_event_attack реализует лучевой поиск шириной B поверх любого из четырёх генераторов и возвращает набор M синтетических сессий, гарантированно заканчивающихся целевым элементом i^*. Весь процесс реализован в режиме чёрного ящика: ни на одном шаге алгоритм не обращается к параметрам атакуемой модели.

Экспериментальная оценка.

1. При $\alpha = 2\%$ деградация HR@10 составляет 0,4 0,9п.п. – измеримый, но ограниченный эффект.
2. При $\alpha = 5\%$ деградация достигает 1,2–3,1п.п. – практически значимое снижение качества.
3. При $\alpha \geq 10\%$ деградация нарастает нелинейно (5,9–7,9п.п. при 20%), что свидетельствует о доминировании синтетического сигнала.
4. Трансформерный генератор вызывает наибольшую деградацию, навигационный граф – наименьшую; порядок воспроизводится на обоих датасетах при всех уровнях α.
5. Системы с малой обучающей выборкой (Last.fm, ≈1600 сессий) значительно уязвимее крупных (YooChoose, 30251 сессий) при одинаковой доле отравления.

Практическое значение. Инструментарий позволяет разработчику до ввода системы в эксплуатацию: (1) установить максимальную допустимую долю аномальных сессий; (2) определить, при каком α деградация метрик становится приемлемой; (3) сравнить чувствительность к генераторам разной сложности и принять обоснованное решение о мерах защиты.

Благодарности

Авторы благодарны сотрудникам кафедры Информационной безопасности (ИБ) за обсуждения и полезные замечания. Вопросы использования Искусственного интеллекта в кибербезопасности являются одним из основных научных направлений кафедры ИБ факультета ВМК МГУ имени М.В. Ломоносова и рассматривались во множестве магистерских диссертаций и научных работ [35, 36, 37].

Библиография

[1] P. Covington, J. Adams, and E. Sargin, “Deep neu ral networks for YouTube recommendations,” in Pro ceedings of the ACM Conference on Recommender Systems (RecSys). ACM, 2016, pp. 191–198.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learn ing based recommender system: A survey and new perspectives,” vol. 52, no. 1. ACM, 2019, pp. 5:1 5:38.

[3] M. Ludewig and D. Jannach, “Evaluation of session based recommendation algorithms,” in User Model ing and User-Adapted Interaction, vol. 28. Springer, 2018, pp. 331–390.

[4] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian, “A survey on session-based recommender systems,” ACM Computing Surveys, vol. 54, no. 7, pp. 154:1–154:38, 2021.

[5] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, “Web usage mining: Discovery and applications of usage patterns from Web data,” ACM SIGKDD Explorations Newsletter, vol. 1, no. 2, pp. 12–23, 2000.

[6] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in Pro ceedings of the International Conference on Machine Learning (ICML). JMLR.org, 2012, pp. 1807–1814.

[7] M. Fang, N. Z. Gong, and J. Liu, “Influence function based data poisoning attacks to top-n recommender systems,” in Proceedings of the International World Wide Web Conference (WWW). ACM, 2020, pp. 3019–3025.

[8] Z. Yue, Z. He, Q. Zeng, and J. McAuley, “Black-box attacks on sequential recommenders via data-free model extraction,” in Proceedings of the ACM Con ference on Recommender Systems (RecSys). ACM, 2021, pp. 44–54.

[9] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for session-based recommendation,” in Proceedings of the ACM Con ference on Recommender Systems (RecSys). ACM, 2017, pp. 306–310.

[10] L. R. Rabiner, “A tutorial on hidden Markov mod els and selected applications in speech recognition,” Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[11] S. Rendle, C. Freudenthaler, and L. Schmidt Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in

Proceedings of the International World Wide Web Conference (WWW). ACM, 2010, pp. 811–820.
[12] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with re current neural networks,” in Proceedings of the In ternational Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016.
[13] B. Hidasi and A. Karatzoglou, “Recurrent neural net works with top-k gains for session-based recommen dations,” Proceedings of the ACM International Con ference on Information and Knowledge Management (CIKM), pp. 843–852, 2018.
[14] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural attentive session-based recommendation,” in Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM). ACM, 2017, pp. 1419–1428.
[15] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, “STAMP: Short-term attention/memory priority model for session-based recommendation,” in Pro ceedings of the ACM SIGKDD International Con ference on Knowledge Discovery and Data Mining (KDD). ACM, 2018, pp. 1831–1839.
[16] W.-C. Kang and J. McAuley, “Self-attentive sequen tial recommendation,” in Proceedings of the IEEE International Conference on Data Mining (ICDM). IEEE, 2018, pp. 197–206.
[17] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, “BERT4Rec: Sequential recommenda tion with bidirectional encoder representations from transformer,” in Proceedings of the ACM Interna tional Conference on Information and Knowledge Management (CIKM). ACM, 2019, pp. 1441–1450.
[18] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-based recommendation with graph neural networks,” in Proceedings of the AAAI Con ference on Artificial Intelligence. AAAI Press, 2019, pp. 346–353.
[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30. Curran Associates, 2017.
[20] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” IEEE Transactions on Neural Networks and Learn ing Systems, vol. 30, no. 9, pp. 2805–2824, 2019.
[21] B. Biggio and F. Roli, “Wild patterns: Ten years af ter the rise of adversarial machine learning,” Pattern Recognition, vol. 84, pp. 317–331, 2018.
[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explain ing and harnessing adversarial examples,” in Pro ceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, 2015.
[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in Proceedings of the In ternational Conference on Learning Representations (ICLR), Vancouver, Canada, 2018.
[24] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE, 2017, pp. 39–57.
[25] Y. Deldjoo, V. W. Anelli, H. Abdollahpouri, A. Bel login, C. Gao, D. Jannach, B. P. Knijnenburg, J. Ni, A. Said, S. Sato, M. Schedl, and D. Tikk, “A sur vey on adversarial recommender systems: from at tack/defense strategies to generative adversarial net works,” ACM Computing Surveys, vol. 54, no. 2, pp. 1–38, 2021.
[26] V. W. Anelli, Y. Deldjoo, T. Di Noia, A. Ferrara, and F. Narducci, “Adversarial recommender systems: Attack, defense, and advances,” Proceedings of the ACM Conference on Recommender Systems (Rec Sys), 2021.
[27] S. K. Lam and J. Riedl, “Shilling recommender sys tems for fun and profit,” in Proceedings of the In ternational World Wide Web Conference (WWW). ACM, 2004, pp. 393–402.
[28] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Toward trustworthy recommender sys tems: An analysis of attack models and algorithm robustness,” ACM Transactions on Internet Tech nology (TOIT), vol. 7, no. 4, pp. 23:1–23:37, 2007.
[29] R. Burke, B. Mobasher, C. Williams, and R. Bhau mik, “Classification features for attack detection in collaborative recommender systems,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 542–547, 2006.
[30] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and D. Yang, “Attacking recommender systems with augmented user profiles,” in Proceedings of the ACM Interna tional Conference on Information and Knowledge Management (CIKM). ACM, 2020, pp. 855–864.
[31] H. Chen, J. Li, and P. Hui, “Data poisoning at tacks on neighborhood-based collaborative filtering,” in Proceedings of the SIAM International Conference on Data Mining (SDM). SIAM, 2021, pp. 145–153.
[32] L. Huang, Y. Ma, S. Li, B. Liu, and H. Wang, “Data poisoning attacks to deep learning based recommender systems,” in Proceedings of the Net work and Distributed System Security Symposium (NDSS), 2021.
[33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] K. Cho, B. van Merri¨enboer, C. Gulcehre, D. Bah danau, F. Bougares, H. Schwenk, and Y. Ben gio, “Learning phrase representations using RNN encoder–decoder for statistical machine transla tion,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL, 2014, pp. 1724–1734.
[35] Бербер, Д. В., and О. Р. Лапонина. "Разработка подходов к увеличению устойчивости моделей машинного обучения для обнаружения распределенных атак отказа обслуживания." International Journal of Open Information Technologies 13.6 (2025): 16-24.
[36] Ниничук, М. М., and Д. Е. Намиот. "Обзор методов построения рекомендательных систем на основе сессий." International Journal of Open Information Technologies 11.5 (2023): 22-32.
[37] Лапонина, О. Р., and Р. Н. Костин. "Разработка программного обеспечения моделирования угроз для систем на базе LLM-агентов." International Journal of Open Information Technologies 13.6 (2025): 132-146.

Статья получена 17 мая 2026.
А.С. Хитрова – МГУ имени М.В. Ломоносова (email: Ariana.Hitrova@yandex.ru).
О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

Resilience of session-based recommender systems to training set poisoning attacks

A.S. Khitrova, O.R. Laponina

***Abstract* – This article assesses the resilience of session-based recommender systems to training set poisoning attacks using synthetic sessions. Real event logs inevitably contain entries generated by automated agents, which reduces the quality of the trained models. To controllably reproduce such poisoning, a Python software package has been developed. It includes four classes of synthetic session generators: a navigation graph (first-order Markov chain), a hidden Markov model (HMM), a recurrent generator based on the Elman network, and a transformer generator. All generators implement a unified interface and can work with arbitrary sets of event logs.**

The targeted attack module implements a beam search for synthetic sessions with a given final element in black-box mode, without requiring access to the parameters of the attacked model. After poisoning, the target model is retrained on D_train ∪ P with the same hyperparameters. The test set remains unchanged. The success criterion for the attack is quality degradation on the clean test set. An experimental evaluation was conducted on the YooChoose and Last.fm datasets. A mean pooled embeddings model is used as the recommended model: for each session prefix, the arithmetic mean of the element embeddings is calculated, followed by a linear layer to compute logits over the dictionary.

With a 2% share of poisoning sessions, HR@10 degradation is 0.4–0.9 percentage points; with 5%, it is 1.2–3.1 percentage points; with 20%, degradation increases nonlinearly and reaches 5.9–7.9 percentage points. Neural network generators produce more diverse synthetic sessions compared to Markov models. A transformer generator can combine transitions not observed in the aggregate, which expands the coverage of the feature space. It is the diversity of the generated paths that increases the bias of the training distribution toward the target element. The transformer generator exhibits the greatest degradation among the four classes, while the navigation graph exhibits the least. These results allow recommender system developers to reasonably assess the robustness of models before deployment.

***Keywords* – adversarial attacks, training set poisoning, session-based recommender systems, synthetic session generation, model robustness, Markov chains, hidden Markov models, recurrent neural networks, transformer, YooChoose and Last.fm datasets.**

REFERENCES

[1] P. Covington, J. Adams, and E. Sargin, “Deep neu ral networks for YouTube recommendations,” in Pro ceedings of the ACM Conference on Recommender Systems (RecSys). ACM, 2016, pp. 191–198.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learn ing based recommender system: A survey and new perspectives,” vol. 52, no. 1. ACM, 2019, pp. 5:1 5:38.

[3] M. Ludewig and D. Jannach, “Evaluation of session based recommendation algorithms,” in User Model ing and User-Adapted Interaction, vol. 28. Springer, 2018, pp. 331–390.

[4] S. Wang, L. Cao, Y. Wang, Q. Z. Sheng, M. A. Orgun, and D. Lian, “A survey on session-based recommender systems,” ACM Computing Surveys, vol. 54, no. 7, pp. 154:1–154:38, 2021.

[5] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan, “Web usage mining: Discovery and applications of usage patterns from Web data,” ACM SIGKDD Explorations Newsletter, vol. 1, no. 2, pp. 12–23, 2000.

[6] B. Biggio, B. Nelson, and P. Laskov, “Poisoning attacks against support vector machines,” in Pro ceedings of the International Conference on Machine Learning (ICML). JMLR.org, 2012, pp. 1807–1814.

[7] M. Fang, N. Z. Gong, and J. Liu, “Influence function based data poisoning attacks to top-n recommender systems,” in Proceedings of the International World Wide Web Conference (WWW). ACM, 2020, pp. 3019–3025.

[8] Z. Yue, Z. He, Q. Zeng, and J. McAuley, “Black-box attacks on sequential recommenders via data-free model extraction,” in Proceedings of the ACM Con ference on Recommender Systems (RecSys). ACM, 2021, pp. 44–54.

[9] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for session-based recommendation,” in Proceedings of the ACM Con ference on Recommender Systems (RecSys). ACM, 2017, pp. 306–310.

[10] L. R. Rabiner, “A tutorial on hidden Markov mod els and selected applications in speech recognition,” Proceedings of the IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[11] S. Rendle, C. Freudenthaler, and L. Schmidt Thieme, “Factorizing personalized markov chains for next-basket recommendation,” in Proceedings of the International World Wide Web Conference (WWW). ACM, 2010, pp. 811–820.

[12] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with re current neural networks,” in Proceedings of the In ternational Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016.

[13] B. Hidasi and A. Karatzoglou, “Recurrent neural net works with top-k gains for session-based recommen dations,” Proceedings of the ACM International Con ference on Information and Knowledge Management (CIKM), pp. 843–852, 2018.

[14] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, “Neural attentive session-based recommendation,” in Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM). ACM, 2017, pp. 1419–1428.

[15] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, “STAMP: Short-term attention/memory priority model for session-based recommendation,” in Pro ceedings of the ACM SIGKDD International Con ference on Knowledge Discovery and Data Mining (KDD). ACM, 2018, pp. 1831–1839.

[16] W.-C. Kang and J. McAuley, “Self-attentive sequen tial recommendation,” in Proceedings of the IEEE International Conference on Data Mining (ICDM). IEEE, 2018, pp. 197–206.

[17] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, “BERT4Rec: Sequential recommenda tion with bidirectional encoder representations from transformer,” in Proceedings of the ACM Interna tional Conference on Information and Knowledge Management (CIKM). ACM, 2019, pp. 1441–1450.

[18] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, “Session-based recommendation with graph neural networks,” in Proceedings of the AAAI Con ference on Artificial Intelligence. AAAI Press, 2019, pp. 346–353.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30. Curran Associates, 2017.

[20] X. Yuan, P. He, Q. Zhu, and X. Li, “Adversarial examples: Attacks and defenses for deep learning,” IEEE Transactions on Neural Networks and Learn ing Systems, vol. 30, no. 9, pp. 2805–2824, 2019.

[21] B. Biggio and F. Roli, “Wild patterns: Ten years af ter the rise of adversarial machine learning,” Pattern Recognition, vol. 84, pp. 317–331, 2018.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explain ing and harnessing adversarial examples,” in Pro ceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, 2015.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in Proceedings of the In ternational Conference on Learning Representations (ICLR), Vancouver, Canada, 2018.

[24] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in Proceedings of the IEEE Symposium on Security and Privacy (S&P). IEEE, 2017, pp. 39–57.
[25] Y. Deldjoo, V. W. Anelli, H. Abdollahpouri, A. Bel login, C. Gao, D. Jannach, B. P. Knijnenburg, J. Ni, A. Said, S. Sato, M. Schedl, and D. Tikk, “A sur vey on adversarial recommender systems: from at tack/defense strategies to generative adversarial net works,” ACM Computing Surveys, vol. 54, no. 2, pp. 1–38, 2021.
[26] V. W. Anelli, Y. Deldjoo, T. Di Noia, A. Ferrara, and F. Narducci, “Adversarial recommender systems: Attack, defense, and advances,” Proceedings of the ACM Conference on Recommender Systems (Rec Sys), 2021.
[27] S. K. Lam and J. Riedl, “Shilling recommender sys tems for fun and profit,” in Proceedings of the In ternational World Wide Web Conference (WWW). ACM, 2004, pp. 393–402.
[28] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Toward trustworthy recommender sys tems: An analysis of attack models and algorithm robustness,” ACM Transactions on Internet Tech nology (TOIT), vol. 7, no. 4, pp. 23:1–23:37, 2007.
[29] R. Burke, B. Mobasher, C. Williams, and R. Bhau mik, “Classification features for attack detection in collaborative recommender systems,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 542–547, 2006.
[30] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and D. Yang, “Attacking recommender systems with augmented user profiles,” in Proceedings of the ACM Interna tional Conference on Information and Knowledge Management (CIKM). ACM, 2020, pp. 855–864.
[31] H. Chen, J. Li, and P. Hui, “Data poisoning at tacks on neighborhood-based collaborative filtering,” in Proceedings of the SIAM International Conference on Data Mining (SDM). SIAM, 2021, pp. 145–153.
[32] L. Huang, Y. Ma, S. Li, B. Liu, and H. Wang, “Data poisoning attacks to deep learning based recommender systems,” in Proceedings of the Net work and Distributed System Security Symposium (NDSS), 2021.
[33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] K. Cho, B. van Merri¨enboer, C. Gulcehre, D. Bah danau, F. Bougares, H. Schwenk, and Y. Ben gio, “Learning phrase representations using RNN encoder–decoder for statistical machine transla tion,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). ACL, 2014, pp. 1724–1734.
[35] Berber, D. V., and O. R. Laponina. "Razrabotka podhodov k uvelicheniju ustojchivosti modelej mashinnogo obuchenija dlja obnaruzhenija raspredelennyh atak otkaza obsluzhivanija." International Journal of Open Information Technologies 13.6 (2025): 16-24.
[36] Ninichuk, M. M., and D. E. Namiot. "Obzor metodov postroenija rekomendatel'nyh sistem na osnove sessij." International Journal of Open Information Technologies 11.5 (2023): 22-32.
[37] Laponina, O. R., and R. N. Kostin. "Razrabotka programmnogo obespechenija modelirovanija ugroz dlja sistem na baze LLM-agentov." International Journal of Open Information Technologies 13.6 (2025): 132-146