Time interval modelling with DETPP approach

Работу выполнили: Некрасова Дарья, Линиченко Дарья, Лагерь Майя

1 Цель работы

Целью работы было сравнить качество предсказания следующего события для моделей, специализированных под задачу next-item prediction, и модели **DeTPP** (описана в работе [3]), изначально ориентированной на долгосрочное прогнозирование.
Дополнительно были исследованы следующие факторы:

1. влияние типа энкодера истории событий (сравнение **GRU** и **Transformer**; замена `RnnEncoder(GRU)` на `TransformerEncoder(SimpleTransformer)`);
2. влияние компоненты функции потерь, связанной с предсказанием присутствия события (**BCE**).

2 Бейзлайны

2.1 SASRec

Self-Attentive Sequential Recommendation (SASRec) был предложен в 2018 году в работе [2]. Как следует из названия, модель использует механизм самовнимания (self-attention), генерируя последовательность рекомендаций аналогично тому, как в задачах NLP генерируются последовательности слов.

Основные преимущества

- Возможность учитывать все предыдущие взаимодействия
- Одинаково хорошо работает как с плотными, так и с разреженными данными

- Блок self-attention в модели можно легко параллелизовать, в результате чего модель обучается на порядок быстрее RNN или CNN аналогов.

Архитектура

SASRec состоит из эмбеддинг-слоя с позиционным кодированием, стэка блоков самовнимания с остаточными связями и нормализацией, и выходного слоя на основе матричной факторизации.

Применение в исследовании

Мы использовали SASRec в качестве базовой модели благодаря её проверенной эффективности, балансу между точностью и скоростью, а также универсальности на различных типах данных. Наша реализация опиралась на реализацию из работы [1].

Метрики

В таблицах 2.1, 2.2 представлены 95% доверительные интервалы для метрик. Эксперименты проводились в сетапе global-time-split, last item на валидации и random item на тесте.

Метрика	@1	@5	@10	@20	@50	@100
NDCG	0.0098 ± 0.0007	0.0186 ± 0.0005	0.0227 ± 0.0009	0.0264 ± 0.0007	0.0327 ± 0.0006	0.0388 ± 0.0006
HR	0.0098 ± 0.0007	0.0272 ± 0.0005	0.0397 ± 0.0014	0.0537 ± 0.0011	0.0875 ± 0.0006	0.1245 ± 0.0016
MRR	0.0098 ± 0.0007	0.0158 ± 0.0006	0.0175 ± 0.0006	0.0185 ± 0.0006	0.0195 ± 0.0006	0.0200 ± 0.0006
COV	0.2402 ± 0.0022	0.5221 ± 0.0019	0.6464 ± 0.0037	0.7570 ± 0.0014	0.8695 ± 0.0022	0.9233 ± 0.0004

Таблица 2.1: SASRec, Amazon-beauty

Метрика	@1	@5	@10	@20	@50	@100
NDCG	0.0387 ± 0.0060	0.0854 ± 0.0081	0.1083 ± 0.0103	0.1295 ± 0.0096	0.1574 ± 0.0098	0.1757 ± 0.0126
HR	0.0387 ± 0.0060	0.1302 ± 0.0131	0.2013 ± 0.0178	0.2857 ± 0.0216	0.4265 ± 0.0211	0.5398 ± 0.0339
MRR	0.0387 ± 0.0060	0.0707 ± 0.0075	0.0800 ± 0.0085	0.0858 ± 0.0081	0.0903 ± 0.0081	0.0919 ± 0.0083
COV	0.1623 ± 0.0053	0.3749 ± 0.0013	0.4989 ± 0.0038	0.6257 ± 0.0042	0.7751 ± 0.0038	0.8765 ± 0.0041

Таблица 2.2: SASRec, Movielens-1M

2.2 T2ARec

Test-Time Alignment for Tracking User Interest Shifts in Sequential Recommendation был предложен в 2025 году в работе [5]. Как следует из названия, модель направлена на решение

ключевой проблемы динамических рекомендательных систем — сдвигов пользовательских интересов во время инференса. В отличие от традиционных подходов, T2ARec использует *Test-Time Training (TTT)*, позволяя модели адаптироваться к новым паттернам поведения непосредственно на тестовых данных без размеченных ответов, за счёт двух специально разработанных self-supervised модулей выравнивания.

Основные преимущества

- Способность отслеживать изменения пользовательских интересов в реальном времени.
- Явное моделирование временной динамики через выравнивание абсолютных временных интервалов с адаптивными шагами модели.
- Теоретически обоснованная устойчивость к сдвигам распределений благодаря механизму согласования скрытых состояний интересов (interest state alignment).
- Высокая эффективность при работе с длинными последовательностями за счёт использования структурных state space моделей (Mamba-2/SSD) как основы архитектуры.

Архитектура

Архитектурно T2ARec состоит из следующих компонентов:

1. **Эмбеддинг-слой** (Embedding Layer).
2. **T2A-Mamba блок**, построенный на основе Mamba-2 с динамически генерируемыми параметрами Δ, B, C.
3. **Align²-SSM блок**, реализующий два ключевых механизма:
   - *Time Interval Alignment Loss* для захвата временной динамики.
   - *Interest State Alignment Loss* для реконструкции и выравнивания состояния интереса.
4. **Выходной слой** для предсказания следующего айтема.

Применение в исследовании

Мы использовали T2ARec в качестве продвинутой baseline-модели благодаря её уникальной способности адаптироваться в онлайн-режиме, теоретической строгости и доказанному превосходству над современными методами на стандартных датасетах, особенно в сценариях с выраженными сдвигами интересов.

3 DeTPP

В данной работе использовалась архитектура **DeTPP** (detection-based temporal point process), в которой модель на каждом шаге по истории событий строит контекст и далее *одновременно* предсказывает набор из K будущих событий в пределах заданного горизонта. Данная архитектура была предложена в работе [3].

3.1 Формат данных и обозначения

Пусть наблюдаемая последовательность событий для одного объекта имеет вид

$$\{(t_i, y_i)\}_{i=1}^{L},$$

где t_i — время события, $y_i \in \{1, \dots, C\}$ — категориальная метка (item/класс), L — длина истории. На каждом шаге i модель получает контекст по префиксу $\{(t_j, y_j)\}_{j \le i}$ и предсказывает K будущих событий.

3.2 Энкодер истории событий

Сначала каждое событие кодируется эмбеддингами:

- метка y_i проходит через embedding-слой: $e(y_i) \in \mathbb{R}^{d_e}$;
- время (или приращение времени) t_i подаётся как числовой признак (возможна нормализация/клиппинг по `max_time_delta`).

Полученный вектор признаков события обозначим x_i.

Далее последовательность $\{x_i\}_{i=1}^{L}$ кодируется в контекстные представления $\{h_i\}_{i=1}^{L}$ с помощью одного из вариантов энкодера:

1. **RNN-энкодер** (GRU): $h_i = \mathrm{GRU}(x_i, h_{i-1})$;

2. **Transformer-энкодер**: $h_{1:L} = \text{Transformer}(x_{1:L})$ (каузальная маска, позиционное/временное кодирование).

3.3 Декодер: K кандидатов будущих событий

На каждом шаге i из контекста h_i модель генерирует K "слотов" будущих событий. Для каждого слота $k \in \{1, \ldots, K\}$ предсказываются:

- логит присутствия события $s_{i,k}$ (presence logit);
- прогноз времени будущего события $\widehat{\Delta t}_{i,k}$ (или $\hat{t}_{i,k}$ в выбранной параметризации);
- логиты по классам $\ell_{i,k} \in \mathbb{R}^C$ для метки $\hat{y}_{i,k}$.

Это можно записать как

$$(s_{i,k},\, \widehat{\Delta t}_{i,k},\, \ell_{i,k}) = g_k(h_i),$$

где $g_k(\cdot)$ реализуется общей "головой" (head) с K выходами (в коде это `ConditionalHead` с параметром $k =$ `detection_k`).

3.4 Функция потерь

Модель одновременно предсказывает K будущих событий $\{\hat{y}_i\}_{i=1}^{K}$ в заданном горизонте H. Пусть $\{y_i\}_{i=1}^{T}$ — множество истинных событий в этом горизонте, где T может меняться от примера к примеру.

3.4.1 Парная функция потерь (Pairwise Loss)

Для одного предсказанного события $\hat{y}_i$ и истинного события $y_j = (t_j, l_j)$ парная потеря определяется как negative log-likelihood:

$$\mathcal{L}_{\text{pair}}(y_j, \hat{y}_i) = |t_j - \hat{t}_i| - \log \hat{p}_i(l_j),$$

где:

- $\hat{t}_i$ — предсказанное время события,
- $\hat{p}_i(l)$ — предсказанное распределение вероятностей по классам (softmax).

3.4.2 Потеря на присутствие события (BCE Loss)

Для каждого предсказанного слота i модель также предсказывает вероятность $\hat{\sigma}_i$ того, что в этом слоте есть событие. Пусть σ — некоторое сопоставление (биекция) между истинными событиями и подмножеством слотов. Тогда:

$$\mathcal{L}_{\text{BCE}}(\sigma, \hat{y}) = -\sum_{i \in \sigma} \log \hat{\sigma}_i \; - \sum_{i \notin \sigma} \log(1 - \hat{\sigma}_i),$$

где:

- первый член — для слотов, сопоставленных с истинными событиями,
- второй член — для слотов, оставшихся несопоставленными (пустые).

3.4.3 Matching Loss

Оптимальное сопоставление σ^* находится минимизацией суммарной стоимости:

$$\mathcal{L}_{\text{matching}}(y, \hat{y}) = \min_{\sigma \in \mathcal{A}} \left[\sum_{i=1}^{T} \mathcal{L}_{\text{pair}}(y_i, \hat{y}_{\sigma(i)}) + \mathcal{L}_{\text{BCE}}(\sigma, \hat{y}) \right],$$

где $\mathcal{A}$ — множество всех возможных сопоставлений. Оптимальное σ^* находится с помощью **Hungarian algorithm**.

3.4.4 Дополнительный loss для следующего события

Чтобы улучшить предсказание next item, для первого слота ($i = 1$) добавляется дополнительный loss:

$$\mathcal{L}_{\text{next}}(y_1, \hat{y}_1) = |t_1 - \hat{t}_1| - \log \hat{p}_1(l_1).$$

3.4.5 Итоговая функция потерь

Полный loss для обучения модели DEF:

$$\mathcal{L}_{\text{DEF}}(y, \hat{y}) = \mathcal{L}_{\text{matching}}(y, \hat{y}) \; + \; \lambda \cdot \mathcal{L}_{\text{next}}(y_1, \hat{y}_1),$$

Авторы статьи брали $\lambda = 4$ во всех экспериментах.

3.4.6 Взвешивание компонентов в matching cost

При вычислении стоимости для Hungarian matching используется следующая взвешенная формула (в коде авторов):

$$\text{cost}(i,j) = w_t \cdot |t_j - \hat{t}_i| \;+\; w_y \cdot \big[-\log \hat{p}_i(l_j)\big] \;+\; w_p \cdot \big[-\log \hat{\sigma}_i\big],$$

где веса выбираются эмпирически.

3.4.7 Калибровка порога присутствия на инференсе

На инференсе для отбора предсказанных событий используется динамический порог τ_i для каждого слота i. Порог обновляется во время обучения:

- отслеживается частота сопоставления слота i с истинными событиями,
- τ_i устанавливается как квантиль распределения $\hat{\sigma}_i$, соответствующий этой частоте (используется streaming-алгоритм).

Событие сохраняется, если $\hat{\sigma}_i > \tau_i$, после чего предсказания сортируются по времени.

Таким образом, **DEF использует единую matching-based функцию потерь**, которая:

- динамически сопоставляет предсказания с истинными событиями,
- учитывает вероятность присутствия события через $\mathcal{L}_{\text{BCE}}$,
- дополнительно усиливает качество предсказания следующего события через $\mathcal{L}_{\text{next}}$.

Это позволяет модели достигать высокой точности и разнообразия предсказаний на длинных горизонтах.

3.5 Итоговый пайплайн

1. Embedder кодирует события (t_i, y_i) в x_i.
2. SeqEncoder (GRU или Transformer) строит контекст h_i.
3. Head предсказывает K кандидатов будущих событий: $\hat{\sigma}_k$ (presence), $\hat{t}_k$ (time), $\hat{p}_k(l)$ (label).
4. Matching сопоставляет кандидатов с истинными событиями в горизонте.

5. Считается loss как сумма $\mathcal{L}_{\text{matching}}$ и $\mathcal{L}_{\text{next}}$ с каким-то коэффициентом.

6. На инференсе применяется порог по presence, сортировка по времени и получение конечного прогноза.

За основу мы брали реализацию авторов алгоритма, находящуюся в репозитории [4].

4 Стандартный сценарий next-item

Сравнение производилось на Amazon Beauty и ML. В таблице 4.1 представлены результаты экспериментов по стандартным метрикам

Таблица 4.1: Результаты экспериментов (NDCG ↑ / HR ↑ / MRR ↑) (@10)

Model	ML			Amazon		
	NDCG	HR	MRR	NDCG	HR	MRR
SASRec	0.1083 ±0.0103	0.2013 ±0.0178	0.0800 ±0.0085	**0.0227** ±0.0009	**0.0397** ±0.0014	**0.0175** ±0.0006
T2ARec	**0.1316** ±0.0034	**0.2079** ±0.0104	**0.0936** ±0.0021	0.0079 ±0.0024	0.0121 ±0.0011	0.0082 ±0.0010
DeTPP	-	-	-	0.0071 ±0.0029	0.0131 ±0.0055	0.0091 ±0.0082

Примечание: Жирным выделены лучшие результаты.

5 Улучшения DeTPP

5.1 Трансформер вместо GRU

В исходной конфигурации DeTPP в качестве энкодера истории может использоваться рекуррентная сеть (например, GRU), которая последовательно агрегирует информацию о прошлых событиях и передаёт итоговый контекст в K голов предсказания. Такой подход естественен для последовательных данных, однако имеет известные ограничения: рекуррентная модель вынуждена “сжимать” всю историю в один скрытый вектор, из-за чего при длинных последовательностях ухудшается учёт дальних зависимостей и взаимодействий событий на разных временных масштабах.

В качестве альтернативы мы рассматриваем энкодер на основе Transformer. Механизм self-attention позволяет каждому событию напрямую учитывать другие события в истории, что потенциально лучше моделирует:

1. долгосрочные зависимости (например, повторяемые паттерны и сезонность),
2. неоднородную важность событий (attention выступает как адаптивное взвешивание контекста),
3. взаимодействие между типом события и временем (контекстно-зависимые шаблоны).

С точки зрения архитектуры DeTPP замена GRU на Transformer означает, что вместо рекуррентного обновления скрытого состояния h_t мы строим контекстные представления $\{z_1, \ldots, z_n\}$ для событий истории и используем агрегированное представление (например, z_n или pooling по $\{z_i\}$) как общий контекст для всех K голов. При этом остальная часть модели не меняется: K голов по-прежнему предсказывают кандидатов будущих событий параллельно, а обучение выполняется с помощью matching loss на горизонте.

Экспериментально замена GRU на Transformer приводит к улучшению качества предсказаний на горизонте: в наших запусках конфигурация с Transformer даёт существенно больший T-mAP и меньший OTD по сравнению с GRU, что согласуется с гипотезой о более эффективном извлечении зависимостей в последовательности при помощи self-attention.

5.2 BCE поправка

В исходном алгоритме DeTPP учитывается поправка $\mathcal{L}_{\text{BCE}}$, которая по задумке позволяет обучать модель правильно предсказывать, в каких из K слотов действительно произойдёт событие. (штрафует, если слот сопоставлен с истинным событием, но его вероятность $\hat{\sigma}_i$ низка, или если слот остался пустым, но модель предсказала в нём высокую $\hat{\sigma}_i$). Мы посмотрели, что будет, если не учитывать эту поправку.

6 Сравнение гиперпараметров DeTPP

Сравнение производилось на Amazon от easyTPP. В таблице 6.1 представлены результаты экспериментов по стандартным метрикам

Таблица 6.1: Результаты экспериментов для Amazon (OTD ↓ / T-mAP ↑)

Model	OTD ↓	T-mAP ↑
DeTPP (~~BCE~~, GRU)	9.85	1.64%
DeTPP (BCE, GRU)	9.53	1.00%
DeTPP (~~BCE~~, Transformer)	**8.55**	**12.15%**

Примечание: OTD ↓ означает "чем меньше, тем лучше T-mAP ↑ означает "чем больше, тем лучше".

Вывод. На Amazon лучшее качество по обеим метрикам показывает конфигурация DeTPP (Transformer, без BCE): она даёт минимальный OTD (8.55↓) и максимальный T-mAP (12.15%↑). Для GRU добавление BCE/presence улучшает OTD (9.85 → 9.53), но заметно снижает T-mAP (1.64% → 1.00%), что указывает на компромисс: presence помогает лучше согласовать число/распределение событий, но может ухудшать точность ранжирования типов событий на горизонте.

Список литературы

[1] Danil Gusak, Anna Volodkevich, Anton Klenitskiy, Alexey Vasilev и Evgeny Frolov. “Time to Split: Exploring Data Splitting Strategies for Offline Evaluation of Sequential Recommenders”. В: *Proceedings of the 19th ACM Conference on Recommender Systems*. 2025. DOI: 10.1145/3705328.3748164.

[2] Wang-Cheng Kang и Julian McAuley. *Self-Attentive Sequential Recommendation*. 2018. arXiv: 1808.09781 [cs.IR]. URL: https://arxiv.org/abs/1808.09781.

[3] Ivan Karpukhin и Andrey Savchenko. *Detecting the Future: All-at-Once Event Sequence Forecasting with Horizon Matching*. 2025. arXiv: 2503.19889 [cs.LG]. URL: https://arxiv.org/abs/2503.19889.

[4] Ivan Karpukhin, Foma Shipilov и Andrey Savchenko. “HoTPP Benchmark: Are We Good at the Long Horizon Events Forecasting?” В: *arXiv preprint arXiv:2406.14341* (2024). URL: https://arxiv.org/abs/2406.14341.

[5] Changshuo Zhang, Xiao Zhang, Teng Shi, Jun Xu и Ji-Rong Wen. *Test-Time Alignment for Tracking User Interest Shifts in Sequential Recommendation*. 2025. arXiv: 2504.01489 [cs.IR]. URL: https://arxiv.org/abs/2504.01489.