BEHAV-E! You are Not Just a Number to Us, but an R^{2048} Embedding

Juan Manuel Rodriguez*
Aalborg University
Aalborg, Denmark
jmro@cs.aau.dk

Antonela Tommasel*
CONICET-UNCPBA
Tandil, Argentina
Johannes Kepler University
Linz, Austria
antonela.tommasel@isistan.unicen.edu.ar

Abstract

Universal user representations hold the promise of reducing the need for task-specific modelling in personalized systems, enabling a single embedding to support a wide range of predictive tasks such as churn prediction, product propensity, and category-level recommendations. In this paper, we introduce BEHAV-E (*Behavioural Embedding via Hybrid Action Variational Encoder*), a self-supervised architecture that learns rich, 2048-dimensional user embeddings from multi-event behavioural data. BEHAV-E integrates both temporal and semantic signals by combining kernel density estimation (KDE) of user activity timelines, product and category embeddings via shared embedding bags, and an LSTM-based autoencoder for modelling search query semantics. We evaluate our approach within the *RecSys Universal Behavioural modelling Challenge*. Our results demonstrate that BEHAV-E effectively captures complex, multi-modal user behaviour in a compact and transferable embedding format. This paper provides an overview of the approach we used as team DArgk[1] for the *ACM RecSys Challenge 2025*. Our submission achieved the 13^{th} rank and sum of scores of 4.5713 (borda count 483) in the competition academia-track final results. We release our source code at: https://github.com/knife982000/RecSys2025Challenge

Keywords

user modelling, user universal encoding, user variational encoder

ACM Reference Format:
Juan Manuel Rodriguez and Antonela Tommasel. 2025. BEHAV-E! You are Not Just a Number to Us, but an R^{2048} Embedding. In *RecSys Challenge 2025 (RecSysChallenge '25), September 22, 2025, Prague, Czech Republic.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3758126.3758130

1 Introduction

User modelling is essential for large-scale personalized services, allowing businesses to better understand user behaviour and improve predictive tasks, such as churn prediction, recommendations, and user profiling [1]. However, most existing approaches are task-specific, relying on supervised training for individual downstream applications like click-through rate prediction or personalized recommendations. A key challenge in this context is the diversity of user characteristics, which often results in multi-modal and long-tailed data distributions. These patterns can significantly reduce model performance and robustness, particularly in large, heterogeneous user populations [9, 10]. Furthermore, each predictive task typically requires its own dataset, training pipeline, and embedding space. While effective in isolated scenarios, this approach is not scalable and incurs in high engineering and computational costs, limiting generalization to unseen tasks [1, 2].

To address these limitations, recent work has focused on learning universal user representations, compact encoding of user behaviour designed to be free of task-specific biases [1]. These pre-trained representations can be applied across a wide range of downstream tasks, requiring only lightweight models for adaptation instead of retraining full task-specific pipelines [9]. Universal representations offer key advantages, including lower computational costs, improved generalization, and the ability to support diverse predictive tasks within a unified framework [9].

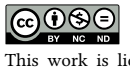
In this work, we introduce BEHAV-E (*Behavioural Embedding via Hybrid Action Variational Encoder*), a self-supervised architecture for universal user modelling. BEHAV-E combines session-level autoencoding with contrastive learning to capture both temporal dynamics and semantic relationships in user behaviour. A variational encoder further enhances the robustness and expressiveness of the learned representations. This hybrid design allows BEHAV-E to generate compact 2048-dimensional embeddings that effectively summarize user histories for diverse downstream tasks.

To understand and evaluate the properties of the generated embeddings, we performed a series of embedding space analyses. These include visualizing the representation space with dimensionality reduction, assessing cluster homogeneity to determine whether users with similar behavioural patterns are grouped together, and analysing correlations between embedding dimensions and user activity attributes. These analyses revealed that BEHAV-E embeddings capture both temporal dynamics and activity levels, structuring the space in a way that reflects key aspects of user behaviour.

2 Problem formulation

The *ACM RecSys Challenge 2025*[2], organized by *Synerise*, a deep-tech company specializing in AI and big data, focused on learning user representations that generalize across multiple predictive tasks. Using real-world interaction logs, the goal is to generate *Universal Behavioural Profiles*, compact embeddings that capture essential

*Both authors contributed equally to this research.
[1] DArgk stands for the authors' base countries: Denmark and Argentina.



RecSysChallenge '25, Prague, Czech Republic

ACM ISBN 979-8-4007-2099-4/25/09
https://doi.org/10.1145/3758126.3758130

[2] https://recsys.synerise.com/

Table 1: Distribution of number of actions logged per User

Actions	Mean	Min	25%	50%	75%	Max
Buys	2.15 ± 3.75	1.00	1.00	1.00	2.00	644.00
Adds	3.90 ± 12.82	0.00	0.00	1.00	3.00	1, 597.00
Removes	1.65 ± 8.00	0.00	0.00	0.00	1.00	1, 145.00
Searches	6.23 ± 25.83	0.00	0.00	0.00	3.00	3, 384.00
Visits	50.14 ± 145.04	0.00	0.00	17.00	49.00	29, 676.00

behavioural patterns. These embeddings are then evaluated by training standardized neural models on top of them, with performance measured in terms of accuracy, novelty, and diversity across three open and three hidden downstream tasks: • **Churn Prediction**. Predict whether an active user will make no purchases in the next 14 days. • **Product Propensity**. Predict which products a user is likely to purchase within 14 days. • **Category Propensity**. Predict the categories in which a user will make purchases within 14 days. • **Conversion (Hidden 1)**. Predict whether a relevant client will make a purchase during the evaluation period. • **New Product Propensity (Hidden 2)**. Predict user preferences for the 20 most popular items from the training period that appear only in the evaluation period. • **Price Propensity (Hidden 3)**. Predict which of 100 predefined price buckets a user will purchase from.

2.1 Data

The data provided by the RecSys Challenge consists of anonymized real-world user interaction logs encompassing various activities and product metadata. It includes six event types: product buy, add to cart, remove from cart, page visit, search query, and product properties. The first three events capture transactional behaviour, each with a user ID, timestamp, and product ID. Page visit events record a user ID, timestamp, and URL identifier but do not specify the content of the visited page (e.g., which products were displayed). Search query events store user-issued queries as compressed textual embeddings. Additionally, product metadata includes a category ID, price bucket, and embeddings of the product name.

Table 1 summarizes the distribution of logged actions for the one million users in the competition data. The data shows a steep decline from general activities like visits ($\mu = 50.14$) to high-intent actions such as purchases ($\mu = 2.15$), suggesting that while users often interact with the platform, only a small subset completes transactions. Large standard deviations and extreme maximum values across all action types indicate a highly skewed distribution, likely driven by a small number of exceptionally active users.

2.2 Evaluation

The primary performance metric is AUROC. For category and product propensity, evaluation uses a weighted average of AUROC, novelty, and diversity. To compute diversity, each prediction is transformed using an element-wise sigmoid and then L1-normalized. The entropy of the resulting distribution serves as the diversity measure for that prediction. The model's overall diversity score is then calculated as the average entropy across all predictions.

Novelty is derived from the popularity of the top-k predicted items. For each prediction, popularity is computed as the weighted sum of the popularities of the top-k targets, normalized so that a score of 1 indicates full confidence in the k most popular items. The model's overall popularity score is the average across all predictions. Due to data sparsity, raw popularity values are typically close to zero, resulting in novelty values near 1. To enhance sensitivity to small variations, raw popularity scores are raised to the 100th power before computing novelty as $1 - popularity$.

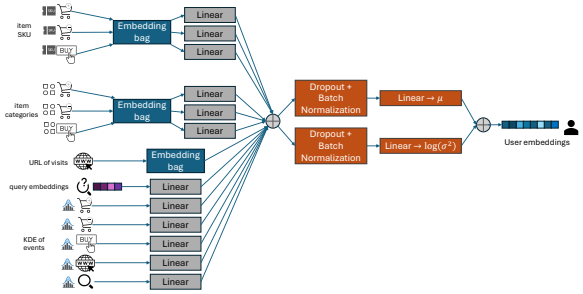


Figure 1: BEHAV-E architecture

2.3 User embedding analysis

To analyse whether the resulting user embeddings capture meaningful behavioural patterns, we examined how distances in the embedding space correlated with user similarity based on the categories of the items bought, added, or removed from the cart. Category similarities were derived from embeddings obtained via matrix factorization on user-category purchase data. We also visualized the 2D embedding projection, highlighting the number of user interactions and the timing of their activity.

3 BEHAV-E: Generating user embeddings

Figure 1 presents the architecture of our approach for generating user representations, named BEHAV-E. This model produces embeddings that integrate both temporal and semantic aspects of user behaviour. While variational autoencoders have shown success in modelling user behaviour [6], BEHAV-E departs from the traditional autoencoder design; it is designed solely as an encoder and is not trained to reconstruct user activity. VAEs provide a probabilistic framework that naturally captures the uncertainty and variability in sequential data, which is crucial when modelling noisy or sparse user behaviour sequences common in recommendation tasks. By encoding data into distributions rather than fixed points, VAEs allow robust generalization [13]. The inputs capture multiple dimensions of user interactions; SKUs and categories encode lists of products and their associated categories for specific actions, and URL visits are embedded directly, while query embeddings summarize user searches. Kernel Density Estimates (KDEs) provide vectorized representations of the temporal distribution of actions. Using a KDE representation allows our model to avoid using recurrent neural networks that tend to have problems learning short sequences [15], which is the common case. Moreover, they might have limitations for very large sequences of events, an infrequent but real case, requiring truncation and risking vanishing gradient.

Preprocessing. For each event type, we represent its temporal distribution using KDE with a Gaussian kernel [12]. Timestamps are scaled to the $[0, 1]$ interval, where 0 corresponds to the earliest event across all users and 1 the latest. This process generates, for each user and activity type, a list of event times $T = \{x_1, x_2, \ldots, x_n\}$.

To generate the KDE profile vector, we defined eleven equidistant reference points across the $[0, 1]$ interval to ensure equal contribution from all temporal segments. While increasing the number of points could improve discretization, it would also raise model

complexity. Given the sparsity of most user activities, we found that eleven points provided an effective trade-off. The KDE value at each reference point r_j is computed as $\hat{f}_h(r_j) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{r_j - x_i}{h}\right)$, where $K(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}u^2\right)$ denotes the kernel function, $h > 0$ is the bandwidth controlling the kernel's smoothness ($h = 0.1$ in our implementation), and x_i are the observed data points. We use the *Gaussian kernel* for its smoothness and desirable statistical properties [7].

Each user is thus represented by five KDE vectors, one for each activity type (add, remove, buy, visit, search). For product-related events (add, remove, buy), these representations were further enriched with the corresponding lists of SKUs and their associated categories. Similarly, URL visits are encoded as lists of visited URLs. For search queries, the provided quantized embeddings were converted into 64-dimensional floating-point embeddings using an LSTM-based autoencoder trained on product names, achieving over 99.9% reconstruction accuracy[3]. To aggregate multiple search queries per user, we computed the element-wise mode, yielding a compact representation of their search behaviour.

For SKUs, categories, and URLs, we used embedding bags that compute the mean embedding across each user's list of items. Formally, for a user's set of items (e.g., $I = \{i_1, i_2, \dots i_k\}$), $Emb_t(I) = \frac{1}{k}\sum_{j=1}^{k} E_{i_j}$ where $E \in \mathbb{R}^{N\times d}$ is a trainable weight matrix, where N is the number of unique items and d the embedding size. To capture semantic differences between operations on the same item (e.g., adding vs. removing a product), we applied operation-specific linear transformations. For example, the intermediate embedding for SKU_{add} is computed as $iter\_SKU_{add} = W_{add}Emb_{sku}(SKU_{add}) + b_{add}$, while for SKU_{rm} it is $iter\_SKU_{rm} = W_{rm}Emb_{sku}(SKU_{rm}) + b_{rm}$. This strategy allows BEHAV-E to share a single embedding matrix for memory efficiency while producing distinct representations for each operation. The only exception is URLs, which require no such transformation since they are only associated with visits.

Finally, the KDE vectors and query embeddings (already in $\mathbb{R}$) were passed through linear layers. The resulting intermediate vectors were concatenated into a single feature vector, which defines a sampling distribution $\mathcal{N}(\mu, \sigma^2)$. To compute μ and $\log(\sigma^2)$, the concatenated vector was passed through two parallel pipelines of dropout, batch normalization, and linear layers. The final user embedding $z \sim \mathcal{N}(\mu, \sigma^2)$ was sampled using the reparameterization trick [3]. Dropout and sampling were intended to introduce noise to improve robustness and mitigate overfitting.

Training. To train BEHAV-E, we used a contrastive learning strategy with two randomly initialized model instances, $M_1(U)$ and $M_2(U)$, referred to as Dual-BEHAV-E. For each training batch, a set of n users is randomly sampled and encoded by both models, producing two behaviour embedding matrices Y_1 and Y_2 in $\mathbb{R}^{n\times e}$, where e is the embedding size. As the dataset lacks explicit embedding targets, we optimized the model using the InfoNCE loss [11], which encourages alignment between the embeddings from $M_1(U)$ and $M_2(U)$. The InfoNCE loss is defined as:

$$\mathcal{L}_{\text{InfoNCE}}(Y_1, Y_2) = \text{CrossEntropy}\left(\text{Softmax}\left(Y_1 Y_2^\top\right),\ I\right) \quad (1)$$

where, Softmax was applied row-wise to the similarity matrix $Y_1 Y_2^\top$, and I denotes the identity matrix, enforcing maximum similarity between corresponding user embeddings. To balance contributions from both models, we symmetrize the loss:

$$\mathcal{L}(Y_1, Y_2) = \frac{\mathcal{L}_{\text{InfoNCE}}(Y_1, Y_2) + \mathcal{L}_{\text{InfoNCE}}(Y_2, Y_1)}{2} \quad (2)$$

We trained two versions of Dual-BEHAV-E with identical model configurations but different training strategies. Each model uses embedding bags with dense layer outputs of 128 dimensions, and a final layer producing 512-dimensional user embeddings. Optimization was performed with AdamW (learning rate $1e-3$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 0.01)[4].

For training, one model used a batch size of 1,000 with Exponential Moving Average (EMA) smoothing (0.999 decay) to smooth parameter updates [5]. The second employed a larger batch size of 4,000 and a 1Cycle learning rate schedule [8], which cyclically adjusted the learning rate for improved convergence. Larger batches are generally preferred for contrastive learning as they create a harder InfoNCE objective, though GPU memory constraints limited the maximum batch size. These two versions were designed to explore complementary optimization strategies and enhance representation diversity for ensembling.

Inference. We used the trained Dual-BEHAV-E and concatenated their resulting embeddings. While the two embeddings are expected to be broadly similar, they are not identical; concatenating them effectively acts as an ensemble of complementary representations. During inference, each BEHAV-E returns the expected value $\mathbb{E}(z) = \mu$ instead of sampling $z \sim \mathcal{N}(\mu, \sigma^2)$, with disabled dropout. Since we trained two Dual-BEHAV-E models, this results in four BEHAV-E instances. Each one produces a $\mathbb{R}^{512}$ embedding, meaning each Dual-BEHAV-E outputs a $\mathbb{R}^{1024}$ vector. Concatenating both Dual-BEHAV-E outputs yields a final $\mathbb{R}^{2048}$ embedding for each user.

Table 2: Task performance for submitted embeddings

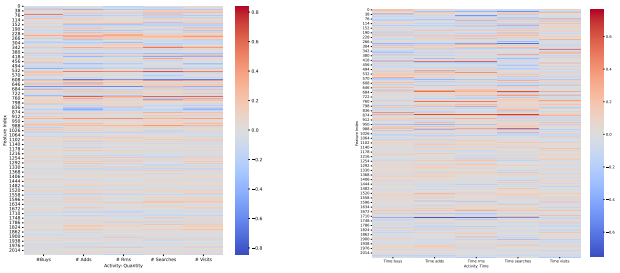
Configuration	Churn	Category	Product	Hidden 1	Hidden 2	Hidden 3
Simple	0.711	0.783	0.762	0.734	0.739	0.785
1Cycle	0.719	0.787	0.765	0.737	0.718	0.784
1Cycle+4kB	0.709	0.781	0.773	0.734	0.745	0.783
EMA	0.713	0.787	0.768	0.734	0.733	0.785
Final	**0.719**	**0.792**	**0.787**	0.739	**0.746**	0.788
Final+KL	0.701	0.775	0.750	0.719	0.704	0.774
Final+Reg	0.716	0.787	0.778	0.738	0.743	0.785
Full mix	0.719	0.791	0.783	**0.741**	0.729	**0.790**
Ens. 1Cycle+4kB	0.713	0.781	0.777	0.740	0.718	0.783

4 Experimental Analysis

Table 2 summarizes the results of our submissions. *Simple* is the baseline model, trained with a batch size of 1,000 and AdamW. *1Cycle* builds on this by incorporating a 1Cycle learning rate schedule, while *1Cycle+4kB* further increases the batch size to 4,000. *EMA* follows *Simple* but applies *EMA* during training. We then evaluated ensemble-based models. *Final* represents our main submission, combining *1Cycle+4kB* and *EMA* via embedding concatenation. *Final+KL* extends this by adding a KL-divergence term to the loss, as in a standard VAE [4]. *Final+Reg* introduces a regularization term that penalizes embedding dimensions exceeding one, inspired by Zhao et al. [14]. *Full Mix* ensembles all four base models but averages embeddings within each Dual-BEHAV-E, producing 512-dimensional embeddings. Finally, *Ens. 1Cycle+4kB* is an ensemble of

[3]Treating each vector element as a classification task over 256 quantization classes.

[4]Following PyTorch's default settings: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html



(a) Number of actions (b) Average time of the actions

Figure 2: Spearman correlations between embedding features and s' distribution (0-1023: *1Cycle+4kB* 1024-2047: *EMA*)

four Dual-BEHAV-E models trained with *1Cycle+4kB* and a reduced embedding size (128 dimensions).

Among the non-ensemble models, *1Cycle+4kB* and *EMA* performed best, motivating their inclusion in our final submission. Experiments with additional regularization techniques and alternative ensembling strategies did not improve generalization. All models were trained solely on the one million relevant users, as earlier tests on the full dataset yielded no performance gains but substantially increased training time. In prior dataset versions, we also explored transformer-based and recurrent architectures; however, these struggled to generate meaningful embeddings, likely due to the limited number of logged activities per user.

4.1 User embedding analysis

Figure 2 illustrates the correlation between embedding features and two user activity metrics: total number of actions (Figure 2a) and the average time of those actions within the logged period (Figure 2b), where higher values indicate activity concentrated toward the end of the period. The first 1,024 features, from the *1Cycle+4kB* model, show consistently stronger absolute correlations than those from *EMA*, suggesting that *1Cycle+4kB* may be slightly more effective at capturing these behavioural patterns.

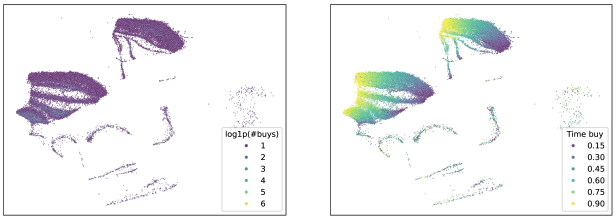
To evaluate how well embeddings captured user interests, we computed the Spearman correlation between pairs[5] of user embedding dot products and the semantic similarity of their category interaction sets (buy, add, and remove). Category set similarities were derived using a Matrix Factorization model trained on user purchase data to learn category embeddings, then the similarity between two users' sets was computed through bipartite matching of their category embeddings. For the *Full* model, correlations[6] were modest: 0.0446 (buy), 0.1340 (add), and 0.1367 (remove). The *1Cycle+4kB* model achieved consistently higher correlations of 0.0650, 0.1666, and 0.1374, indicating stronger alignment with semantic interest similarity. In contrast, the *EMA* variant showed near-zero or slightly negative correlations: −0.0251 (buy), −0.0090 (add), and 0.0725 (remove). These findings suggest that *1Cycle+4kB* embeddings better capture user interests in a linear sense than *EMA*.

To explore the structure of the learned representations, we projected the 2048-dimensional user embeddings into 2D (Figure 3), colouring points by behavioural attributes for each action type[7].

[5]Due to the computational complexity, we sample a million pairs.

[6]All correlations are statistically significant with $p - value < 3e - 7$ within a 5σ confidence interval.

[7]The visualizations for the other action types can be found in the companion repository.



(a) Number of actions (b) Average time of the actions

Figure 3: User embeddings vs "buy" actions distribution

Across all actions, embeddings exhibited consistent structural patterns. Main clusters displayed smooth temporal gradients, suggesting that BEHAV-E encodes the timing of user interactions as a continuous feature, as users with early actions clustered together, while those with later actions occupied distinct regions. Activity levels further shaped the embedding space, with high-activity users concentrated in dense cluster cores and low-activity users dispersed along the periphery, often forming smaller, isolated groups that likely correspond to sparse or atypical behavioural histories. Each action type also exhibited distinct characteristics:

- **Buy**. Two main clusters exhibited smooth temporal gradients from early to late purchases. High-activity users concentrated in dense core regions, while low-activity users appeared in smaller peripheral clusters.
- **Add**. Temporal gradients were evident, with a distinctive subcluster indicating a unique behavioural pattern. High-activity users formed dense cores, while low-activity users were dispersed across multiple clusters.
- **Remove**. Clusters were sparse, with few high-activity users and a predominance of peripheral low-activity groups. Temporal organization persisted, though high-volume removers were rare and tightly localized.
- **Search**. Richer signals were evident, with high-activity users more widely distributed and peripheral clusters appearing less fragmented. Temporal dynamics remained well preserved.
- **Visit**. Clusters were primarily shaped by activity level, with clear temporal gradients from early to late visits within each cluster.

These findings highlight BEHAV-E's ability to capture temporal patterns and activity sparsity across diverse user behaviours, structuring the embedding space to reflect intent, engagement, and behavioural diversity. This organization is well-suited to support downstream tasks requiring fine-grained temporal and semantic understanding of user behaviour.

5 Conclusions

In this paper, we introduced BEHAV-E, an encoder for generating general-purpose user embeddings from activity logs. Our results demonstrated that BEHAV-E effectively captures user behaviour despite activity sparsity, producing embeddings that support a wide range of downstream models without task-specific information. Embedding analyses further showed that the model encodes both temporal dynamics and user preferences.

Acknowledgments

We want to acknowledge CLAAUDIA from Aalborg University for providing the infrastructure to train the models.

References

[1] Jie Gu, Feng Wang, Qinghui Sun, Zhiquan Ye, Xiaoxiao Xu, Jingmin Chen, and Jun Zhang. 2021. Exploiting behavioral consistence for universal user representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4063–4071.

[2] Clark Mingxuan Ju, Leonardo Neves, Bhuvesh Kumar, Liam Collins, Tong Zhao, Yuwei Qiu, Qing Dou, Yang Zhou, Sohail Nizam, Rengim Ozturk, et al. 2025. Learning Universal User Representations Leveraging Cross-domain User Intent at Snapchat. *arXiv preprint arXiv:2504.21838* (2025).

[3] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

[4] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

[5] Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. 2024. Exponential Moving Average of Weights in Deep Learning: Dynamics and Benefits. *Transactions on Machine Learning Research* (2024). https://openreview.net/forum?id=2M9CUnYnBA

[6] Qianzhen Rao, Yang Liu, Weike Pan, and Zhong Ming. 2023. BVAE: Behavior-aware Variational Autoencoder for Multi-Behavior Multi-Task Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 625–636. doi:10.1145/3604915.3608781

[7] Bernard W Silverman. 2018. *Density estimation for statistics and data analysis*. Routledge.

[8] Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006. SPIE, 369–386.

[9] Qinghui Sun, Jie Gu, XiaoXiao Xu, Renjun Xu, Ke Liu, Bei Yang, Hong Liu, and Huan Xu. 2022. Learning interest-oriented universal user representation via self-supervision. In *Proceedings of the 30th ACM International Conference on Multimedia*. 7270–7278.

[10] Xiaoyu Tan, Yongxin Deng, Chao Qu, Siqiao Xue, Xiaoming Shi, James Zhang, and Xihe Qiu. 2023. Adaptive learning on user segmentation: Universal to specific representation via bipartite neural interaction. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 205–211.

[11] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 [cs.LG] https://arxiv.org/abs/1807.03748

[12] Jihu Wang, Yuliang Shi, Han Yu, Kun Zhang, Xinjun Wang, Zhongmin Yan, and Hui Li. 2023. Temporal Density-aware Sequential Recommendation Networks with Contrastive Learning. *Expert Systems with Applications* 211 (2023), 118563. doi:10.1016/j.eswa.2022.118563

[13] Yu Wang, Hengrui Zhang, Zhiwei Liu, Liangwei Yang, and Philip S. Yu. 2022. ContrastVAE: Contrastive Variational AutoEncoder for Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 2056–2066. doi:10.1145/3511808.3557268

[14] Mingjun Zhao, Liyao Jiang, Yakun Yu, Xinmin Wang, Yi Yuan, Zheng Wei, and Di Niu. 2024. DimReg: Embedding Dimension Search via Regularization for Recommender Systems. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. 562–570. doi:10.1137/1.9781611978032.65

[15] Xingrui Zhuo, Shengsheng Qian, Jun Hu, Fuxin Dai, Kangyi Lin, and Gongqing Wu. 2024. Multi-Hop Multi-View Memory Transformer for Session-Based Recommendation. *ACM Trans. Inf. Syst.* 42, 6, Article 144 (July 2024), 28 pages. doi:10.1145/3663760