Learning Self-supervised User Representations Using Contextualized Mixture of Experts in Transformers

Surajit Chakrabarty, Rajat Agarwal, and Agniva Som

Amazon Ads, Bangalore, India
{chakrsur, agrajat, agnivsom}@amazon.com

Abstract. Robotic traffic is an endemic problem in digital advertising, often driven by a large number of fake users indulging in advertising fraud. Temporal sequences of user ad activity contain rich information about user intention while interacting with digital ads, and can be effectively modeled to segregate robotic users with abnormal browsing patterns from regular human users. Sequence models on user ad activity trail trained with generative pre-training produce self-supervised user embeddings that work well on the downstream task of robotic user detection. However, they fall short on robot detection for low-and-slow attacks with very short user sequence lengths, i.e., low activity robotic users with a small number of ad traffic events. As sophisticated bot traffic gravitates toward complex *modus operandi* at a fast pace and exploits gaps in detection systems, it opens up a critical requirement to build advanced user models that go beyond modeling activity sequences. This problem is circumvented by a variation of TabTransformer networks [1], which simultaneously encode user behavioral information from a mix of sequential data (for long activity sequences), and from tabular and numerical user/ads metadata (for short sequences). Despite the overall improvement in detection with TabTransformers [1], there are pockets of under-represented traffic slices where model performance is sub-optimal due to biased allocation of weights between sequential and tabular features to optimize for high volume slices. To that end, we propose a novel sparse Mixture of Experts with TabTransformers as component experts, where the sparse gating function follows a new context-aware routing mechanism comprising of local-global experts. We demonstrate that our proposed model helps to uniformly improve detection and to de-bias vanilla TabTransformer networks with respect to user sequence length, with a maximum gain of 33% over the vanilla TabTransformer model achieved on short activity sequences.

Keywords: Self-supervision · TabTransformers · Mixture of experts · Sparse gating · Contextualized gates · User sequence modeling · Robot detection

1 Introduction

Sponsored Search or Sponsored Ads or Promoted Listings refer to digital performance advertising programs that enable advertisers to bid on search keywords for prime real estate and hence procure higher reach to the enormous and immensely diverse visitor population on popular e-commerce web pages and apps. The high revenue potential from sponsored advertising incentivizes bad actors to commit ad fraud and to unfairly profit from genuine advertisers' revenue share. Fraudsters drive robotic traffic to these ad programs through automated software programs to accomplish deceitful motives like competitor budget exhaustion, boosting of product search ranking etc., which often result in targeted attacks on chosen advertisers. Robotic ad traffic usually originates from malicious user accounts or devices that visit sponsored advertising pages and click on ads with the ulterior motive to defraud advertisers. Naturally, robotic user detection (and mitigation) is an important hygiene function for any sponsored advertising program to root out unwanted and inimical ad traffic, and to protect bona fide advertiser interests.

One major challenge in developing Machine Learning (ML) models to detect malicious user accounts and devices in sponsored ads is that no ground truth labels are available for model training. In order to circumvent the human/robot labeling challenge, an alternate but constructive path forward is to build modern deep neural networks following the unsupervised or self-supervised training regime. Time-ordered sequences of browsing activity can be used for embedding user behavior and user interaction (view or click) with ads. Self-supervision is a useful tool for this modeling problem, which allows to encode user preferences without any labels via prediction of user action at the next time step. Self-supervision on user activity sequences using gated recurrent units (GRUs) [4], long short-term memory networks (LSTMs) [9] and Transformers [28] have shown promise in tasks related to conversion and click prediction [17, 30], web search recommendations [27] as well as fraud and threat detection [18, 29]. We adopt a similar strategy in robotic user detection where ad activity sequence models are trained with self-supervision to generate user embeddings, which are then segregated into human and robotic users by simple downstream tasks.

Empirical evidence suggests that regular sequence models perform admirably in robot detection when user activity sequences are sufficiently long, but falls short on identifying low activity robotic users with very short sequence lengths (e.g. users with only 2 clicks in a given time period). Nonetheless, sophisticated bot traffic is adversarial in nature and have evolved to carry out targeted attacks from a large and distributed set of low activity users (*aka* low-and-slow bots). In order to improve detection of low-and-slow bots, we need to learn user embeddings going beyond the usual user ad activity trail, and enrich information from a variety of rich user metadata and features associated with the ad event. These event level features (e.g. IP address, User Agent, frequency of user clicks in last N days, number of IPs observed from the user, user membership status) can supplement the model with the required intelligence to push the user embeddings of *similar* short ad sequence users toward a common region of the

vector subspace. Sequence encoders like LSTMs and Transformers cannot inherently model a complex combination of sequential, natural language, numerical and tabular features simultaneously with self-supervision. So [1] follows the extension [10] of the original Transformer encoder architecture to develop a version of TabTransformer network that models tabular features along with sequential ones.

In this paper, we rely on the TabTransformer architecture [1, 10] as a building block, using positional and column embeddings for sequential and tabular/numerical inputs respectively, along with output embedding generation using a contrastive learning objective. We propose an enhancement to this architecture by using a mixture of experts (MoE) to attend to each sequence in TabTransformer [6, 7, 25], in order to de-bias the network and prevent from unfairly focusing on user sequences (tokens) with larger representation in the ad traffic data. While mixtures of experts in Transformer have been already explored in Switch-Transformer [7, 22], the major contributions of this paper:

- The design of local-global experts using a routing strategy to fairly apportion parameters and model capacity to sequence tokens from under-represented user slices.
- Introduction of dedicated experts for tabular features, with a pool of experts dedicated to sequential features to mitigate learning bias on the tabular features.

Our model architecture intrinsically accounts for all input feature classes, and has been developed to comprehensively counter adversarial robotic users on sponsored advertising, with a special focus on low-and-slow bots. Experiments show that the proposed contextualized Mixture of Transformer Experts outperforms the TabTransformer baseline by 4.5% considering all traffic slices, and by 32.7% specifically on the slice corresponding to the low and slow bot category.

The rest of the paper is structured as follows. Section 2 discusses related work in gating networks and existing research on user behavioral modeling. We incrementally describe modeling enhancements to sequence only models with our proposed contextualized mixture of Transformers to produce more informative user embeddings in Section 3. Section 4 demonstrates significant performance improvements in the downstream task of robotic user detection brought about by the proposed model, while we conclude in Section 5 with future plans on model enhancement.

2 Related Work

Behavioral modeling of online user activity via sequences of user actions is a powerful and well-established technique, and has been widely used in the context of supervised [19, 21, 32] (with complete or limited labels) as well as self-supervised [1,2,18,27,29,33] learning in a variety of applications like click prediction, search recommendation engine, personalization and fraud/threat detection. Recent research advancements in deep learning (DL) have increasingly magnified

the complexity and the performance of models trained on sequential input data, as first demonstrated in the natural language processing domain. In nearly every application taking sequential inputs, state-of-the-art performance is achieved by Transformer networks [5, 28].

Training Mixtures of Experts (MoE) [6, 11, 25] has been an effective methodology for scaling model capacity with a small computation overhead. In recent years, MoE layers [7] have been incorporated in standard Transformer [28] architecture to get the benefit of power law scaling with model size, data set size and computational budget. Although there have been many studies of gating mechanisms used to choose experts in MoE [3], sparse Mixture-of-Experts (SMoE) models have emerged as the most popular choice for recent applications. With nearly constant computational overhead, SMoE models achieve better performance than dense models on various tasks, including machine translation [15], image classification [23] and speech recognition [14].

Routing mechanism plays an important role in SMoE, as model performance is guided mainly by token-to-expert assignment. Recent studies explored various token assignment algorithms to improve SMoE training [8]. Most gating networks choose one of two broad categories–token choice and expert choice routing. In token choice routing, the gating network uses Top-k gate [7] or DSelect-k gate [8] that is optimized to route a token to a specialized expert. In expert choice routing [31], experts select top tokens to achieve optimal load balancing across experts in a MoE system. Many studies in recent years focus on how to design the token-expert assignment algorithm; some formulate SMoE routing as a linear assignment problem [16] while others prefer to use hashing [24]. In this work, inspired by graph sparsification methods [26], we present a mixture of specialized experts that uses a modified contextual design based on each entity for selecting experts.

3 Model Overview

In this section, we describe how unsupervised user embeddings are generated by different variations of the activity sequence model, to be later utilized to segregate human and non-human ad traffic. As the model complexity gradually increases, so does the richness of the user embeddings, as we are able to encode more information related to user behavior and intent. While the model formulation holds for any ad traffic event type (click, page view or purchase), we focus on ad click activity sequence in the remainder of this paper for reader clarity.

3.1 TabTransformer Model

Traditionally, sequential features are encoded by the Transformer encoder while tabular features are concatenated directly with the encoded vector, following which the concatenation is fed to a fully connected network for fine-tuning or full-scale training with supervised labels [19]. Because of the lack of trustworthy

labels, this modeling technique does not work in the bot user detection domain, where we require end-to-end unsupervised training of both feature classes jointly. The TabTransformer [10] model variation presented in [1] is an appropriate choice, since it allows for joint learning of embeddings with both sequential and tabular input features. In the following sections, we describe the exact model training methodology using the variation of TabTransformer networks described in [1].

Encoding Input Features In this problem, user click feature space comprises of tabular and sequential features spanning over categorical, real valued and natural language data classes. We define any user ad click activity with the vector U:

$$U = [T_1, \ldots, T_p, X_1, \ldots, X_n]$$
$$X_i = [F_1(i); \ldots; F_k(i)]$$

where each user U is defined by p tabular features [T_1,...,T_p] and a sequence of n ad clicks [X_1,...,X_n], where each click event X_i is defined by k features [F_1, F_2,...,F_k].

We encode the user features following equations 1 and 2, where categorical features are mapped from their one-hot representation to a dense embedding using a linear transformation. Natural language features (like search query) are represented as a bag-of-words by summing up embeddings of the constituent words. All the numerical features are log-normalized and concatenated to the input. Let M be be the embedding function for individual features:

$$M = \begin{cases} W.x & \text{if categorical feature} \\ ln(x) & \text{if numerical feature} \\ \sum_{i=1}^{d} x_i & \text{if natural language input of } d \text{ words with embedding } x_i \end{cases} \quad (1)$$

For each event in the sequential input, embeddings of the individual features are concatenated to generate the input embedding for the event. $R(X_i)$ is the embedding of each input event X_i,

$$R(X_i) = Concat(M(F_1(i)), ..., M(F_k(i))) \quad (2)$$

Similar to DLRM [20], non-numerical features are embedded separately using the embedding function M as described above, and a single embedding is constructed for all numerical tabular features using a feed forward network. Let [T_1,...,T_m] be the list of non-numerical tabular features and [T_{m+1},...,T_p] be the list of numerical tabular features. We obtain $m + 1$ input embeddings (represented by R) for tabular features.

$$R(T_i) = M(T_i) \text{ if } i <= m$$
$$R(T_{m+1,p}) = FFN\big(Concat(M(T_{m+1}), ..., M(T_p))\big)$$

Hence, the final input representation I_f to the Transformer becomes

$$I_f = \big[R(T_1), R(T_2), ..., R(T_m), R(T_{m+1,p}), R(X_1), ..., R(X_n)\big]$$

We cannot use positional embeddings as described in [28] for tabular features, as it would require us to define an arbitrary and likely to be sub-optimal ordering over tabular features. So we use learnable column embeddings as in [10] for each input feature column.

TabTransformer Model Architecture We have used masked token prediction similar to masked-language modeling (MLM) in the self-supervised learning paradigm. A random masking strategy has been used across tabular and sequential features. Figure 1 illustrates the TabTransformer architecture used to encode user ad activity information into the user embeddings. To generate user embeddings, we append a [CLS] token as another tabular input feature, and the output corresponding to this token is considered as the summarized user representation. To learn this token representation, we add a contrastive loss [12] to the mask prediction loss, where embeddings of two different masked samples of the same user are considered as positive examples and embeddings for different users are considered as negative examples. Further training details are described in Appendix A.2.

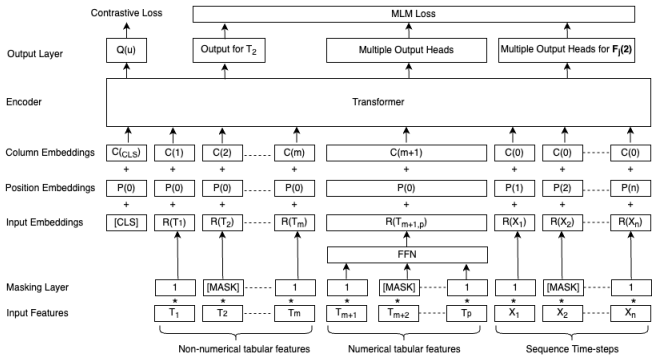


Fig. 1: TabTransformer for learning user embeddings

3.2 TabTransformer with Contextual MoE

Similar to Switch Transformer [7], we have included an MoE layer in the TabTransformer encoder. The MoE layer helps in performing conditional computation of tokens with the help of a routing network. It increases flexibility in the model by routing tokens to a different number of specialized experts, thus capturing varied user behavioral patterns in the same model. We show that routing

of tokens to experts with respect to chosen traffic slice volume indicators is an effective way to uplift model performance on under-represented traffic slices in the input data. Our context-aware MoE with TabTransformers is illustrated in Figure 2.

Contextual MoE Architecture The contextual MOE architecture consists of a router which redirects tokens to any k of $[E_i]_{i=1}^{N}$ experts. The router variables W_{r1}, W_{r2} produce logits which are normalized via softmax distribution over N experts. Since restricting tokens to some experts is sub-optimal and also leads to a load balancing issue across the experts, we use a bias distribution strategy in routing tokens to experts by contextualizing on a feature Z as follows:

- When Z is a categorical feature with b categories, we add bias to gate logits such that token favours any one of the global experts or the local expert mapped to the specific category. Global experts are $[E_i]_{i=1}^{g}$, which attends to all tokens irrespective of the feature category and rest of the experts $[E_i]_{i=g+1}^{N}$ are uniformly mapped across the categories.
- When Z is a numerical feature, we bucketize and split its domain into a set of b disjoint intervals. The newly created bins $[B_t]_{t=1}^{b}$ clearly have an ordinal nature. We add bias in the logits such that the token favors routing to any one of the global experts $[E_i]_{i=1}^{g}$ or to local experts in a window span of length w around the expert mapped to that category.

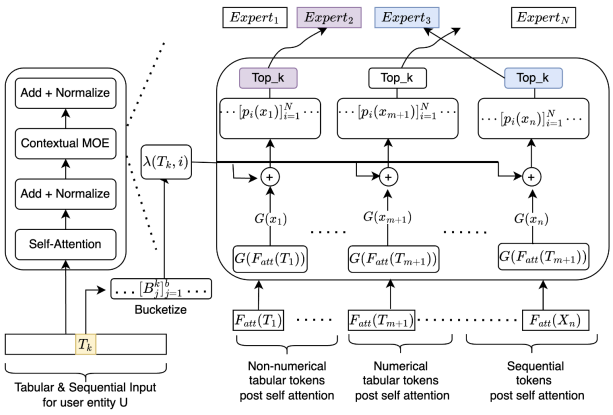
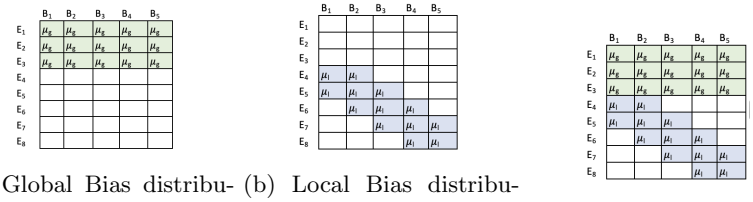


Fig. 2: Contextual MoE with TabTransformer. We have considered $Z = T_k$ a numerical tabular feature in the architecture

Figure 3 explains how the experts are chosen for each category while contextualizing for a real-valued feature Z. For any general feature class, let x



(a) Global Bias distribution (b) Local Bias distribution (c) Bias distribution

Fig. 3: Consider Z to be a real-valued feature representing the number of clicks by a user, which is partitioned into $b = 5$ intervals. For simplicity, assume that we have $[E_i]_{i=1}^{8}$ experts , out of which $[E_i]_{i=1}^{3}$ are global experts and the rest are local experts. Sharing across local experts are done with a window span of $w = 1$.

be any token from user entity U. Then the intermediate routing logit $G(x) = FFN_{W_{r2}}(FFN_{W_{r1}}(x, Relu))$, and the probability that token x is routed to expert E_i is represented as

$$p_i(x) = \frac{\exp^{[G(x)+\lambda(Z,i)+\varepsilon_i]}}{\sum_{j=1}^{N} \exp^{[G(x)+\lambda(Z,j)+\varepsilon_j]}}$$

where ε_l is a random Unif$(0, 1)$ noise and $\lambda(Z, j)$ is the bias distribution function

$$\lambda(Z,j) = \begin{cases} \mu_g,\ j \leq g \\ \mu_l,\ j > g \text{ and } j \in [c-w, c+w] \text{ where } Z \in B_c \\ 0, \text{ otherwise} \end{cases}$$

Finally, we wish to apply a top-k gate on the routing probabilities for directing the token x to the *best* k experts. If τ is the set of selected top-k indices, then the output for x is the linearly weighted combination, $H(x) = \sum_{i\in\tau} p_i(x)E_i(x)$.

4 Experiments

We train the model on over a month of user click interaction data for a large-scale e-commerce advertising program, with user account level features like logged-in status, account tier, email domain, account creation date, numerical aggregates like total distinct devices etc. as tabular inputs and the click activity sequence is defined by features like time since last click, search query, placement, product attribute etc. We generate 128-dimensional user embeddings from each model with an unsupervised training objective. While labeling individual samples accurately may not be possible, multiple domain-knowledge based heuristics can be applied to reliably evaluate if a given group of users are robotic. Hence, user embeddings are clustered into groups using k-means and clusters of users based on these heuristics are marked as robotic.

4.1 Metrics

We calibrate the heuristics to achieve a fixed False Positive Rate (FPR), which refers to the fraction of genuine human traffic detected by the algorithm. Since we do not have ground truth labels, FPR is approximated by using purchasing users as a proxy for the distribution of human labels. The fraction of purchasing clicks that were marked as robotic is computed as FPR. To compare model performance at fixed operating point FPR, we use two key metrics: a) detection rate (DR) depicting the fraction of total traffic that was marked as robotic b) Robotic Signal (RS) coverages, which represent of a highly precise set of robotic events identified using domain knowledge.The full set of RS signals used can be found in Appendix A.1. At a fixed FPR, a better performing model will have a higher detection rate and coverage over robotic signals.

4.2 Models

We train the following models to demonstrate the benefits of the proposed contextual MoE model:

- **TabTransformer** Trained on both sequence and tabular inputs, as described in Section 3.1
- **MoE** TabTransformer model augmented with vanilla Mixture-of-Experts
- **CMoE** Contextualized Mixture-of-Experts model as described in Section 3.2. We also perform an ablation study for CMoE by considering different bias strategies across global and local experts, by setting $\mu_g = u_l$ and $\mu_g < \mu_l$.
- **CMoE-Tab** We introduce another variant of CMoE where we dedicate a single expert to all the tabular features and route the sequential tokens through the remaining pool of experts, with $\mu_g = \mu_l$.

4.3 Hyperparameters

In our experiments we have used 2 encoder blocks, each with 12 attention heads and hidden state size of 256. The hidden state is mapped to 128 dimensions using a linear transform at the output layer. 20% of the sequential and tabular inputs are masked while training. For all the MoE setups we have used $N = 8$ experts and chosen $k = 1$ for each token. For a fair comparison, we considered different model architectures such that the number of floating point operations (FLOPs) per token are the same as the base TabTransformer model. The experts in each encoder increase the number of model parameters independent of the amount of computation, improving performance on the bot user detection task. It is demonstrated in [7] that a computationally matched dense Transformer is outpaced by its SwitchTransformer counterpart, and hence we exclude related experiments in this paper. User click volumes follow a power law distribution, due to which there is a marked discrepancy in training data proportions for different click volume buckets, leading to lower quality embeddings for under-represented

click buckets. We have chosen the MoE context variable Z to represent number of user clicks in the training period. We have trained all the models in a distributed data parallel method with 32 NVIDIA V100 GPUs using TensorFlow. The loss is computed and optimized over a global-batch size of 4096 using Adam [13] optimizer with a learning rate of $2e-04$. For the CMoE ablation, we used $\mu_g = \mu_l = 0.5$ and $\mu_g(0.2) < \mu_l(0.8)$ respectively.

4.4 Results

We present the performance of different models relative to the TabTransformer baseline at a fixed FPR. Table 1 presents the relative Detection Rate for various models, sliced across different user click volume (low, medium and high) buckets and Table 2 demonstrates the corresponding relative robotic signal coverage metrics.

Table 1: Detection Rate Improvement Relative to TabTransformer

Models	Overall	Low	Medium	High
MoE	2.20%	20.24%	17.17%	-1.62%
CMoE $\mu_g = \mu_l$	2.35%	18.45%	12.65%	-0.57%
CMoE $\mu_g < \mu_l$	4.32%	30.95%	18.37%	0.00%
CMoE-Tab	4.47%	32.74%	15.66%	0.43%

Table 2: Robotic Signal Coverage Relative to TabTransformer

Models	RS1	RS2	RS3	RS4
MoE	-0.13%	10.40%	2.37%	7.83%
CMoE $\mu_g = \mu_l$	0.18%	12.29%	2.75%	8.67%
CMoE $\mu_g < \mu_l$	0.25%	14.08%	2.47%	10.69%
CMoE-Tab	0.25%	16.32%	2.87%	11.76%

We observe that vanilla Mixture of Experts in TabTransformer dedicates experts more towards low and medium click buckets, leading to a 1.62% performance drop in the high click bucket, and the proposed usage of combined global and local experts with higher bias towards local experts helps improve performance across all sequence lengths. Further, having a dedicated expert for tabular features boosts performance on the low click slice, while improving the overall robotic coverage. Our proposed CMoE and CMoE-Tab models help in increasing recall towards known bots (as captured by RS metrics), whose attacks span multiple ad traffic dimensions such as user sessions, User Agents, Keywords.

5 Conclusion and Future Work

We introduced a novel contextualization framework for MoE with TabTransformer that demonstrates superlative performance in robot detection and is more effective at catching low-and-slow bots than vanilla TabTransformer models. Our experiments reveal that this model is uniformly able to increase detection across ad traffic slices owing to the user click volume bucket contextualization and the bias distribution strategy across local and global experts.

In the future, we plan to experiment with expanding the framework to longer sequence lengths and larger model sizes to further improve the embedding quality. Since the user embedding learning framework is self-supervised, we will also explore the usefulness of MoEs for multi-task training in advertising.

References

1. Agarwal, R., Muralidhar, A., Som, A., Kowshik, H.: Self-supervised representation learning across sequential and tabular features using transformers. In: NeurIPS 2022 First Table Representation Workshop (2022)
2. Chen, H., Dou, Z., Zhu, Y., Cao, Z., Cheng, X., Wen, J.R.: Enhancing User Behavior Sequence Modeling by Generative Tasks for Session Search. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management. ACM (Oct 2022)
3. Chen, Z., Deng, Y., Wu, Y., Gu, Q., Li, Y.: Towards understanding the mixture-of-experts layer in deep learning. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022)
4. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (Oct 2014)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)
6. Eigen, D., Ranzato, M., Sutskever, I.: Learning factored representations in a deep mixture of experts. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings (2014)
7. Fedus, W., Zoph, B., Shazeer, N.: Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. The Journal of Machine Learning Research **23**(1), 5232–5270 (2022)
8. Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., Chi, E.: Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. Advances in Neural Information Processing Systems **34**, 29335–29347 (2021)

9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
10. Huang, X., Khetan, A., Cvitkovic, M., Karnin, Z.: TabTransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)
11. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural computation **3**(1), 79–87 (1991)
12. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning (2021)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
14. Kumatani, K., Gmyr, R., Salinas, F.C., Liu, L., Zuo, W., Patel, D., Sun, E., Shi, Y.: Building a great multi-lingual teacher with sparsely-gated mixture of experts for speech recognition (2022)
15. Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., Chen, Z.: GShard: Scaling giant models with conditional computation and automatic sharding. In: International Conference on Learning Representations (2021)
16. Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., Zettlemoyer, L.: Base layers: Simplifying training of large, sparse models. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 6265–6274. PMLR (18–24 Jul 2021)
17. Liao, Y.: On the effectiveness of self-supervised pre-training for modeling user behavior sequences. AdKDD (2020)
18. Liu, C., Gao, Y., Sun, L., Feng, J., Yang, H., Ao, X.: User behavior pre-training for online fraud detection. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. p. 3357–3365. KDD '22, Association for Computing Machinery, New York, NY, USA (2022)
19. Muhamed, A., Keivanloo, I., Perera, S., Mracek, J., Xu, Y., Cui, Q., Rajagopalan, S., Zeng, B.: CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In: Efficient Natural Language and Speech Processing NeurIPS 2021 Workshop (2021)
20. Naumov, M., Mudigere, D., Shi, H.M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C., Azzolini, A.G., Dzhulgakov, D., Mallevich, A., Cherniavskii, I., Lu, Y., Krishnamoorthi, R., Yu, A., Kondratenko, V., Pereira, S., Chen, X., Chen, W., Rao, V., Jia, B., Xiong, L., Smelyanskiy, M.: Deep learning recommendation model for personalization and recommendation systems. CoRR **abs/1906.00091** (2019)
21. Oak, R., Du, M., Yan, D., Takawale, H., Amit, I.: Malware Detection on Highly Imbalanced Data through Sequence Modeling. In: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security. p. 37–48. AISec'19, Association for Computing Machinery, New York, NY, USA (2019)
22. Qin, Z., Cheng, Y., Zhao, Z., Chen, Z., Metzler, D., Qin, J.: Multitask mixture of sequential experts for user activity streams. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 3083–3091. KDD '20, Association for Computing Machinery, New York, NY, USA (2020)
23. Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., Houlsby, N.: Scaling vision with sparse mixture of experts. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 8583–8595. Curran Associates, Inc. (2021)

24. Roller, S., Sukhbaatar, S., Szlam, A., Weston, J.E.: Hash layers for large sparse models. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021)
25. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q.V., Hinton, G.E., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017)
26. Spielman, D.A., Teng, S.H.: Spectral sparsification of graphs. SIAM Journal on Computing **40**(4), 981–1025 (2011)
27. Tan, Y.K., Xu, X., Liu, Y.: Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. p. 17–22. DLRS 2016, Association for Computing Machinery, New York, NY, USA (2016)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
29. Yuan, S., Zheng, P., Wu, X., Xiang, Y.: Wikipedia vandal early detection: From user behavior to user embedding. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 832–846. Springer International Publishing, Cham (2017)
30. Zhang, Y., Dai, H., Xu, C., Feng, J., Wang, T., Bian, J., Wang, B., Liu, T.Y.: Sequential click prediction for sponsored search with recurrent neural networks. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. p. 1369–1375. AAAI'14, AAAI Press (2014)
31. Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A.M., Le, Q.V., Laudon, J., et al.: Mixture-of-experts with expert choice routing. Advances in Neural Information Processing Systems **35**, 7103–7114 (2022)
32. Zhu, Y., Xi, D., Song, B., Zhuang, F., Chen, S., Gu, X., He, Q.: Modeling Users' Behavior Sequences with Hierarchical Explainable Network for cross-domain Fraud Detection. In: Proceedings of The Web Conference 2020. p. 928–938. WWW '20, Association for Computing Machinery, New York, NY, USA (2020)
33. Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., Cai, D.: What to Do Next: Modeling User Behaviors by Time-LSTM. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. p. 3602–3608. IJCAI'17, AAAI Press (2017)

A Appendix

A.1 Robotic Signal Definitions

To calculate recall on known bot traffic, we use the following set of very confident heuristics in the form of robot signals (RS).

- **RS1** : Sessions with abnormally large number of clicks in a single hour
- **RS2** : User Agents with large number of clicks and unusually low purchase rates
- **RS3** : Keywords under attack by robotic traffic
- **RS4** : Transport layer signatures with large number of clicks and unusually low purchase rates

A.2 Additional modeling details on TabTransformers

For masked sequential inputs, all features for a time-step are masked and predicted using multiple output heads over the shared representation at the output layer of the Transformer.

$$p(X_i|mask(I), \theta) = \prod_{j=1}^{k} p(F_j(i)|mask(I), \theta) \tag{3}$$

where θ are the parameters of the mode and I is defined as,

$$\begin{aligned} I = &[P(0) + C(1) + R(T_1), \\ &..... \\ &P(0) + C(m) + R(T_m), \\ &P(0) + C(m+1) + R(T_{m+1}, p), \\ &P(1) + C(0) + R(X_1), \\ &..... \\ &P(n) + C(0) + R(X_n)] \end{aligned} \tag{4}$$

To compute the prediction loss over masked features, we use mean-square error and cross entropy loss for numerical and low-cardinal categorical features respectively. For high cardinal and natural language features we used contrastive loss [12], where the negative samples are chosen randomly.

$$J_{contrastive} = -\log \frac{e^{Q(U_{i,m_1})^T Q(U_{i,m_2})}}{e^{Q(U_{i,m_1})^T Q(U_{i,m_2})} + \sum_l e^{Q(U_{i,m_1})^T Q(U_{l,m_1})}} \tag{5}$$

where Q represents the output embedding corresponding to the $[CLS]$ token, U_{i,m_1} and U_{i,m_2} represent two differently masked inputs for a user i and $\{l\}$ represents the set of users forming the negative samples.