Self-supervised Representation Learning Across Sequential and Tabular Features Using Transformers

Rajat Agarwal, Anand Muralidhar, Agniva Som and Hemant Kowshik
{agrajat,anandmur,agnivsom,hkowshik}@amazon.com

Amazon Ads

Abstract

Machine learning models used for predictive modeling tasks spanning across personalization, recommender systems, ad response prediction, fraud detection etc. typically require a variety of tabular as well as sequential activity features about the user. For tasks like click-through or conversion (purchase) rate prediction where labeled data is available at scale, popular methods use deep sequence models (sometimes pre-trained) to encode sequential inputs, followed by concatenation with tabular features and optimization of a supervised training objective. For tasks like bot and fraud detection, where labeled data is sparse and incomplete, the typical approach is to use self-supervision to learn user embeddings from their historical activity sequence. However, these models are not equipped to handle tabular input features during self-supervised learning. In this paper, we propose a novel Transformer architecture that can jointly learn embeddings on both sequential and tabular input features. Our model learns self-supervised user embeddings using masked token prediction objective on a rich variety of features without relying on any labeled data. We demonstrate that user embeddings generated by the proposed technique are able to successfully encode information from a combination of sequential and tabular features, improving AUC-ROC for linear separability for a downstream task label by 5% over embeddings generated using sequential features only. We also benchmark the efficacy of the embeddings on the bot detection task for a large-scale digital advertising program, where the proposed model improves recall over known bots by 10% over the sequential only baseline at the same False Positive Rate (FPR).

1 Introduction

As deep learning techniques advanced, machine learning models used for predictive modeling tasks across personalization, recommender systems, ad response prediction, bot and fraud detection etc. started using a large variety of sequence inputs (like users' page view history, search query sequences etc.) alongside traditionally used tabular features. Modeling activity sequences reduces reliance on engineering complex features by directly operating on granular event level data, while tabular features help model domain specific feature aggregates and features that are stable in time (like user logged-in status, account tier etc). For tasks like click-through rate prediction, where large amounts of labeled data is present, sequential input features are typically encoded using a deep sequence model like an LSTM [1] or a Transformer [2], and the encoded representation [3] is concatenated with tabular features before being fed into a fully connected network for supervised training. More recent techniques also use a separate unsupervised pre-training step for the sequence encoder, instead of just learning the model end-to-end with the concatenated features. However, for domains like fraud detection, where large scale and accurate ground truth data is unavailable, supervised modeling and

Table Representation Learning Workshop at NeurIPS 2022, New Orleans, USA.

fine-tuning based techniques fall short and risk introducing bias contained in the labels. Hence in such settings, models rely more on self-supervised or unsupervised techniques, motivating us to explore self-supervised representation learning techniques that model not just sequential or time-series data but can jointly model tabular features and sequential inputs during the self-supervised embedding learning phase.

While most of the prior work in self-supervised learning focuses on modeling language, speech, vision, time-series or tabular features alone using transformer models as the backbone, joint self-supervised modeling of sequential and tabular features with transformers has four unique challenges. First, the model needs to be able to take as input, a variety of different types of feature variables, which can be either categorical, real valued, natural language or a time series. Second, each input feature needs to contain information about the feature column as transformers treat all inputs as a bag. Third, alongside the column information to differentiate between features, the sequence or time-series inputs also need to contain information about the position of the token in the input sequence. A common positional encoding cannot be used across the sequence and tabular inputs as it would require pre-defining a position order for the tabular inputs, which maybe sub-optimal. Finally, to generate the output embedding, a unique task needs to be defined to aggregate contextualized output representations from the transformer across individual tokens, since the concept of next-sentence prediction (which is used in BERT [4] for natural language processing) is not defined for joint sequential and tabular inputs, and mean-pooling [5] of the contextualized output representations across all inputs is not an option since it would assign disproportionate weight to tokens in long sequence lengths as compared to tabular features.

In this paper, we present a novel Transformer architecture to address the above mentioned challenges with joint self-supervised modeling of tabular and sequential features. The proposed transformer model architecture can accept both tabular features and sequences as inputs where individual tabular features as well as features describing the sequence can be categorical, real valued or in natural language. We differentiate the feature columns by the usage of column embeddings at the time of input, alongside the usage of positional encodings to preserve the sequential information. The model uses masked token prediction (also known as masked language modeling or MLM in the literature) as the self-supervised training objective. We supplement masked token prediction with an auxiliary self-supervised task that uses contrastive learning to generate the final output embedding. We demonstrate the effectiveness of the architecture by learning self-supervised user embeddings for a large-scale digital advertising service based on their ad click activity sequences, and multiple tabular features about the user and benchmarking the efficacy of the learnt embeddings based on linear separability over a downstream label (purchases) as well as on the bot detection task for the ad service.

The paper is structured as follows: Section 2 describes the related work, Section 3 outlines the modeling framework and the self-supervised training objective. This is followed by description of experiments and results in Section 4 and we conclude in Section 5.

2 Related Work

Deep Sequence Modeling Deep learning techniques have demonstrated state-of-the-art results in sequence models and have alleviated the need of engineering complex features for sequential inputs. While earlier deep learning models used LSTM [1] to model sequences, in recent years, Transformer [2] models have achieved state-of-the-art results primarily because of their use of self-attention, as against the inherent sequential nature of LSTMs. Since the self-attention mechanism leads to inputs being treated as bag instead of ordered sequences, transformer models additionally require positional encodings [2, 6] at the time of input.

Self-supervised learning Pre-training of models using self-supervised learning has achieved state-of-the-art results across multiple domains. The core idea of self-supervision is to learn representations (embeddings) of the inputs using a deep learning model, trained on a proxy task constructed from unlabeled data which is usually available in large volumes. Pre-training the model on proxy tasks allows it to capture various important characteristics and features about the input data in an embedding, which can be fine-tuned with labeled data downstream in supervised settings, or used for clustering or in a linear classifier in few-shot settings. Earliest form of self-supervision was seen in denoising autoencoders [7], which used a simple fully connected network to reconstruct tabular inputs perturbed

with noise. In natural language processing, next word prediction using deep autoregressive techniques [8] is popularly used to pre-train models on an unsupervised input corpus. In computer vision, convolutional networks were pre-trained to predict masked patches [11, 12] or different augmentations [13] of input images. More recently, Transformer models have been successfully applied for self-supervised learning in natural language (using next-token prediction in GPT [9] or masked token prediction in BERT [4]), computer vision (using masked image patch prediction [14]) as well as tabular data (using masked token prediction or replaced token detection in TabTransformer [15] and TabNet [10]) domains. While there are some initial works on self-supervised representation learning for multi-modal inputs [16, 17], self-supervision across sequential and tabular data, to the best of our knowledge, remains a relatively under-explored area.

Digital advertising Self-supervised learning has been applied for click-through rate (CTR) prediction by pre-training a large-transformer model using masked language modeling objective [18], treating all input features as a single input string. For conversion rate (CVR) prediction, next event prediction has been used to pre-train deep sequence models on users' historic ad activity, representations from which are concatenated with the tabular features during supervised fine-tuning [20]. For ad fraud and robot detection, attempts to model users' ad activity history using supervised proxy labels are known to give biased models due to imprecise labels. More recent techniques in ad fraud detection eliminate the dependency on proxy labels by using the self-supervised objective of next event prediction on users' ad activity history to learn user embeddings [20]. However, none of the above methods learn jointly across sequential and tabular inputs explicitly during the self-supervised learning phase.

3 Modeling Framework

In this section, we describe the enhancements in the input encoding strategy to accommodate tabular and sequential inputs in Transformers and outline the self-supervised training objective to generate user embeddings.

3.1 Encoding Input Features

Each user entity is defined by a set of tabular features and the click activity sequence over a specified aggregation time window. The click activity sequence is a multi-dimensional time series of clicks made by the user, where each click in the sequence is defined by multiple features. Each of the tabular and sequential features can be either categorical, real valued or in natural language. Formally,

$$U = [T_1, T_2, ..., T_p, X_1, X_2, ..., X_i, ..., X_n] \quad (1)$$

$$X_i = [F_1(i); F_2(i); ...; F_k(i)] \quad (2)$$

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

We do not model the input as a single string as defined in [18, 19], since it inherently removes any ordinal information contained in numerical inputs and forces the network to model basic arithmetic operations over strings, and is not a natural way of modeling different features. Instead, each categorical and natural language feature in tabular as well as the sequential input is mapped to an embedding before being provided as input to the Transformer based encoder network. Categorical features are mapped from their one-hot representation to a dense embedding using a linear transform. Natural language features are represented as a bag-of-words by summing up embeddings of the constituent words [21]. These embedding functions are trained end-to-end with the model objective. All numerical feature inputs are log scaled and concatenated at the time of input. Formally, let $E(x)$ be the embedding function for individual features:

$$E(x) = \begin{cases} W.x & \text{if x is a categorical feature} \\ ln(x) & \text{if x is a numerical feature} \\ \sum_{i=1}^{d} x_i & \text{if x is a natural language input of } d \text{ words with embedding } x_i \end{cases} \quad (3)$$

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

the sequence,

$$R(X_i) = Concat(E(F_1(i)), ..., E(F_k(i))) \tag{4}$$

To embed the tabular inputs meaningfully, we follow the technique used in DLRM [22], where the features are split into two categories - numerical and non-numerical. All non-numerical features are embedded separately using the embedding function E as described above, and a single embedding is constructed for all numerical tabular features using a two layer feed forward network (FFN). 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.

$$R(T_i) = E(T_i) \text{ if } i <= m \tag{5}$$

$$R(T_{m+1,p}) = FFN(Concat(E(T_{m+1}), ..., E(T_p))) \tag{6}$$

Hence, the feature input representation I_f to the transformer becomes,

$$I_f = [R(T_1), R(T_2), ..., R(T_m), R(T_{m+1,p}), R(X_1), ..., R(X_n)] \tag{7}$$

3.2 Positional and Column Embeddings

Since transformers treat inputs as a bag, we need to enrich the input feature representations with information about the feature column and additionally for sequential inputs, information about the position of the event in the sequence. We cannot use positional encodings described in [2] for tabular features as that would require us to define an arbitrary ordering over the tabular features, which may be sub-optimal. Hence, we use learnable column embeddings as described in TabTransformers [15] for each input feature column in I_f. All sequential time-steps share the same column embedding while all tabular inputs in I_f get different column embeddings. However, for encoding positional information within the sequence inputs, all time-steps in the sequential input get different learnable positional encodings while all tabular inputs share a single learnable positional encoding. The final input to the transformer is as follows:

$$\begin{aligned} I = &[P(0) + C(1) + R(T_1), \\ &..... \\ &P(0) + C(m) + R(T_m), \\ &P(0) + C(m+1) + R(T_{m+1}, p), \\ &P(1) + C(0) + R(X_1), \\ &..... \\ &P(n) + C(0) + R(X_n)] \end{aligned} \tag{8}$$

where $P(i)$ and $C(i)$ refer to the positional and column embeddings respectively.

3.3 Self-Supervised Training

3.3.1 Masked Token Prediction

Similar to masked language modeling (MLM) used in BERT, we use masked token prediction as the self-supervised training objective. We randomly mask a fraction of the time-steps of the sequential input and a similar fraction of the tabular features for every user and train the transformer to reconstruct the masked input. 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. For the prediction, we assume all features of the masked time-step are independent given the input data.

$$p(X_i|mask(I), \theta) = p(F_1(i), F_2(i), ..., F_k(i)|mask(I), \theta) = \prod_{j=1}^{k} p(F_j(i)|mask(I), \theta) \tag{9}$$

where θ are the parameters of the model. For numerical tabular inputs, only a subset of the features are masked and multiple output heads are used for prediction of the masked values. To compute the prediction loss over masked features, we use mean-square error and cross entropy loss for numerical and low-cardinality categorical features respectively. Since computing the full probability distribution

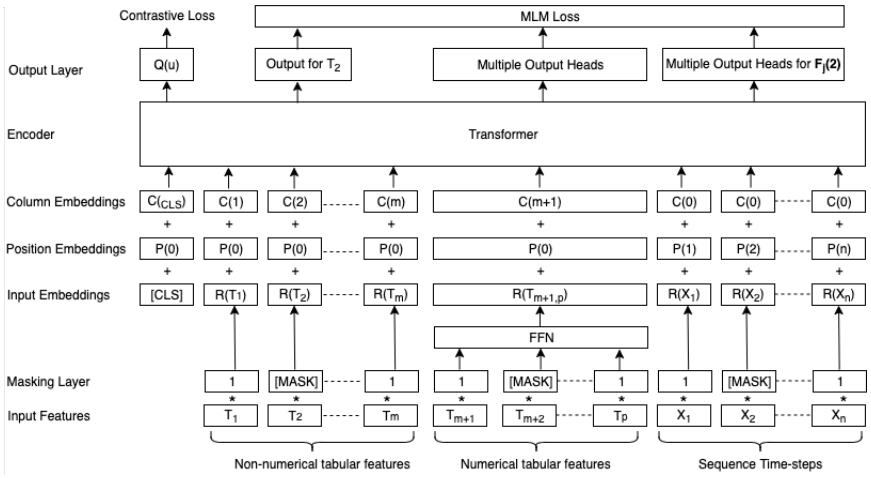

Figure 1: Model Architecture

using softmax for high cardinality and natural language features is expensive, contrastive predictive coding [11, 12] is used, which classifies the ground truth value of the masked feature against a set of randomly chosen negative examples directly in the embedding space. The dot product between the predicted embedding and the target embedding (ground truth or negative samples) represents the logits, using which the cross entropy is computed. We do not apply weighing in the loss terms for different features.

3.3.2 Extracting Final Embedding

Transformers generate contextualized representations for the individual input tokens and require additional steps to generate a single embedding representing the entire input. Similar to BERT, we append a special $[CLS]$ token as another tabular feature for every user and the output corresponding to the $[CLS]$ token is considered as the final user embedding. We train this by adding a contrastive loss term [13] 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.

$$J_{contrastive} = -\log \frac{e^{Q(U_{i,m_1})^T Q(U_{i,m_2})}}{e^{Q(U_{i,m_1})^T Q(U_{i,m_2})} + \sum_l e^{Q(U_{i,m_1})^T Q(U_{l,m_1})}} \tag{10}$$

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.

The overall proposed architecture is illustrated in Figure 1.

4 Experiments

4.1 Downstream Tasks

We train the model over a month of user interaction data for a large-scale advertising program to learn user embeddings and evaluate the performance of the learnt representations on two downstream tasks - first, where accurate labels are known for training a classifier and another where no task specific fine-tuning is possible due to lack of labels. User account level features like logged-in status, account tier, email domain, account creation date, numerical aggregates like total distinct devices etc. are used as tabular inputs and the click activity sequence is defined by features like time since last click, search query, placement, product attribute etc.

4.1.1 Linear Separability in Classification

One way to analyze the quality of unsupervised or self-supervised embeddings is to evaluate their performance on a downstream supervised classification task. In this experiment, we benchmark the user embeddings on the user conversion (purchase) prediction task based on linear separability. We train a linear binary classifier on the learnt users embeddings to predict if the user makes a purchase, and evaluate the efficacy based on AUC-ROC. Higher AUC-ROC implies that the embeddings have better linear separation and few-shot capability with respect to the downstream supervised label.

4.1.2 Click bot detection

Due to absence of accurate ground truth labels for robotic users, supervised techniques fall short in bot and fraud detection scenarios. 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, self-supervised user embeddings are clustered into groups using k-means and clusters of users based on these heuristics are marked as robotic.

We calibrate the heuristics to achieve a fixed False Positive Rate (FPR), which refers to the fraction of genuine human traffic invalidated 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. For a fixed operating point FPR, we compute the robotic recall for the algorithms over a highly precise set of bot traffic events identified using domain knowledge and manual investigations.

4.2 Hyperparameters

For sequential input, we model we truncate the sequence at latest 100 events. The Transformer uses a stack of 4 encoder blocks with 8 attention heads each and a hidden state dimension of 256. The hidden state is mapped to 128 dimensions using a linear transform at the output layer. 20% of sequential as well tabular inputs are masked during pre-training. The loss is computed and optimized over a mini-batch size of 4096 using Adam [23] optimizer with a learning rate of 2e-4 for 3 epochs. The model trains on over 50 million examples, on 8 NVIDIA V100 GPUs with synchronous weight updates for 18 hours using TensorFlow.

4.3 Baseline

We do not build a sequence embedding + tabular feature concatenation baseline as concatenation based approaches do not work when tabular features are of high cardinality - the tabular input size explodes due to the one-hot representation for the high cardinality feature. Embedding such high cardinality features without using self-supervision is challenging in cases where fine-tuning is not performed, as in the case of robot detection task. We also do not show ablations with original Transformer positional embeddings since they require us to define an ordering over the tabular features. Since number of such orderings are combinatorially large, defining an arbitrary ordering would be sub-optimal. Removing positional encodings would collapse the activity sequence into a bag and removing column embeddings would remove distinction among different tabular features.

Instead, we benchmark the proposed transformer model (Sequence + Tab MLM) with joint sequential and tabular inputs against transformers trained on sequential input only to verify if tabular features influence the final user embedding. For the sequential input only baselines, we use a model trained using next token prediction objective (Sequence GPT) and another model trained using masked token prediction (Sequence MLM). For Sequence GPT, output representation from the last time-step is used as the user embedding. We do not benchmark the model against next token prediction applied on sequential and tabular features because it would need a pre-defined ordering over the tabular features to make the autoregressive predictions. For Sequence MLM, the final user embedding is the output representation corresponding to the $[CLS]$ token, trained using the auxiliary contrastive loss term as described in the previous section.

The zero-shot ability of the model is demonstrated by evaluating on linear separability only (as against full fine-tuning) and robot detection task where embeddings are directly clustered to identify bot clusters.

4.4 Results

Table 1 shows the relative AUC-ROC for the linear classifier trained on the learnt embeddings for the conversion prediction task and the robotic recall with models calibrated at a fixed FPR.

Table 1: Relative Performance Comparision

Model	AUC	Robotic Recall (%) at Fixed FPR
Sequence MLM	x	y (undisclosed, relative comparisons only)
Sequence GPT	0.99 x	1.07 y
Sequence + Tab MLM	1.05 x	1.10 y

4.5 Discussion

Results in Table 1 are inconclusive in establishing if masked token prediction works better than next token prediction for sequence only models, and we point the readers to sequence modeling literature which explores this topic in depth [24, 25]. We conclude that Sequence + Tab MLM technique is successfully able to model tabular as well as sequential features, leading to better linear separability based on conversions and higher robotic recall at the same FPR when compared to sequence only MLM model. AUC improvement demonstrates that tabular features were successfully encoded in the learnt embedding. Since activity sequence features capture behavioral patterns and are key towards distinguishing between bot and human users, improvement in bot recall demonstrates that sequential input continues to remain encoded in the user embedding in the proposed Sequence + Tab MLM architecture.

5 Conclusion and Future Work

We presented that masked token prediction using Transformers can be extended to learn self-supervised representations across sequential and tabular features, further positioning Transformers as the unified backbone model that can be applied not only on individual modalities but across modalities. We also showed that the framework can be used to learn self-supervised user embeddings based on a variety of features, which can be applied to multiple downstream tasks.

In future work, we plan to extend the framework to support modeling of extremely long sequence lengths, so that more granular event data can be used to model user behavior. Since our current tabular features are relatively simple, we plan to experiment with more complex feature aggregates and support pre-trained embeddings as inputs. As with the recent trends, we also plan to experiment with significantly larger transformer models to further improve the quality of the learnt representations.

References

[1] Hochreiter, Sepp, and Jürgen Schmidhuber. Long short-term memory. Neural computation 9, no. 8 (1997): 1735-1780.

[2] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998-6008. 2017.

[3] Gligorijevic, Djordje, Jelena Gligorijevic, and Aaron Flores. Time-Aware Prospective Modeling of Users for Online Display Advertising. arXiv preprint arXiv:1911.05100 (2019).

[4] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[5] Reimers, Nils, and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019).

[6] Liu, Xuanqing, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In International Conference on Machine Learning, pp. 6327-6335. PMLR, 2020.

[7] Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pp. 1096-1103. 2008.

[8] Dai, Andrew M., and Quoc V. Le. Semi-supervised sequence learning. In Advances in neural information processing systems, pp. 3079-3087. 2015.

[9] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf (2018).

[10] Arik, Sercan Ö., and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 8, pp. 6679-6687. 2021.

[11] Oord, Aaron van den, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018).

[12] Hénaff, Olivier J., Ali Razavi, Carl Doersch, S. M. Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. arXiv preprint arXiv:1905.09272 (2019).

[13] Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pp. 1597-1607. PMLR, 2020.

[14] He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021).

[15] Huang, Xin, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020).

[16] Liang, Paul Pu, Yiwei Lyu, Xiang Fan, Shengtong Mo, Dani Yogatama, Louis-Philippe Morency, and Ruslan Salakhutdinov. HighMMT: Towards Modality and Task Generalization for High-Modality Representation Learning. arXiv preprint arXiv:2203.01311 (2022).

[17] Yu, Jiahui, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive Captioners are Image-Text Foundation Models. arXiv preprint arXiv:2205.01917 (2022).

[18] Muhamed, Aashiq, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In NeurIPS Efficient Natural Language and Speech Processing Workshop. 2021.

[19] Yin, Pengcheng, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. arXiv preprint arXiv:2005.08314 (2020).

[20] Liao, Yiping. On the Effectiveness of Self-supervised Pre-training for Modeling User Behavior Sequences. In AdKDD, 2020.

[21] Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics 5 (2017): 135-146.

[22] Naumov, Maxim, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang et al. Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091 (2019).

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

[24] Ethayarajh, Kawin. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. arXiv preprint arXiv:1909.00512 (2019).

[25] Topal, M. Onat, Anil Bas, and Imke van Heerden. Exploring transformers in natural language generation: Gpt, bert, and xlnet. arXiv preprint arXiv:2102.08036 (2021).

[26] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283. 2016.