Dense Representation Learning and Retrieval for Tabular Data Prediction

Lei Zheng
zhenglei2016@sjtu.edu.cn
Shanghai Jiao Tong University

Ning Li
lining01@sjtu.edu.cn
Shanghai Jiao Tong University

Xianyu Chen
xianyunin@sjtu.edu.cn
Shanghai Jiao Tong University

Quan Gan
quagan@amazon.com
Amazon

Weinan Zhang*
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University

ABSTRACT

Data science is concerned with mining data patterns from a database, which is assembled by tabular data. As the routine of machine learning, most of the previous work mining the tabular data's pattern based on a single instance. However, they neglect the similar tabular data instances that could help make the label prediction of the target data instance. Recently, some retrieval-based methods for tabular data label prediction have been proposed, which, however, treat the data as sparse vectors to perform the retrieval, which fails to make use of the semantic information of the tabular data. To address such a problem, in this paper, we propose a novel framework of dense retrieval on tabular data (DERT) to support flexible data representation learning and effective label prediction on tabular data. DERT consists of two major components: (i) the encoder that makes the tabular data as embeddings, which could be trained by flexible neural networks and auxiliary loss functions; (ii) the retrieval and prediction component, which makes use of similar rows in the table to make label prediction of the target row. We test DERT on two tasks based on five real-world datasets and experimental results show that DERT achieves consistent improvements over the state-of-the-art and various baselines.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Deep learning, tabular data, retrieval, representation learning

ACM Reference Format:
Lei Zheng, Ning Li, Xianyu Chen, Quan Gan, and Weinan Zhang. 2023. Dense Representation Learning and Retrieval for Tabular Data Prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599305

*The corresponding author W. Zhang is partially supported by NSFC (62177033).

KDD '23, August 06–10, 2023, Long Beach, CA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599305

1 INTRODUCTION

An important domain of data science involves discovering the underlying data patterns from databases, where the data is usually organized as tables. Such patterns are useful for statistics, prediction, and decision-making [3, 43], with various applications including click-through rate (CTR) prediction [32], recommender systems [5], fraud detection [44], anomaly detection [33], etc.

Most modern data science methods working on tabular data utilize machine learning techniques. Basically, the machine learning model is trained to make predictions of a column for each row based on the values of the row's other columns. Such a framework includes various methods ranging from tree models, linear models, Bayesian networks [4, 11, 15, 24], to more recent deep learning models that account for complex feature interactions and inductive biases [2, 9, 13, 29, 39, 46]. However, this framework only focuses on exploring feature interaction patterns within *one* data instance with an i.i.d.(Independent and identically distributed) assumption of the tabular data, ignoring the potential complex dependencies *across data instances*.

To address the shortcoming of the framework with i.i.d. assumptions, several attempts have been made to mine patterns across multiple instances. To predict for a single row, instead of only accounting for the row's own features, these methods take in the features of some other rows as well to further influence the output. The instances are either randomly chosen [23, 42] or based on some similarity metric defined on the row's raw input features [35–37].

However, such a sparse retrieval method suffers from some shortcomings. First, the retrieval method may not be learned from the data, e.g., cosine, TFIDF, BM25. Second, it is common that the mismatch of the different discrete features is equally regarded in such sparse retrieval methods, which results in the ignorance of possible semantic matching. We justify this by considering an example of three data instances with a categorical column City, with values New York, Shanghai, and Boston, respectively. Metrics like BM25 would treat mismatching elements *equally*, i.e. the difference between Shanghai and New York is the same as that between Boston and New York. However, Boston should have a smaller semantic distance to New York than Shanghai since they are in the same country. Such semantic differences can only be learned from the data. Finally, the raw data itself is not sufficiently discriminative to make the label prediction so as to be used in the retrieval process.

In this paper, we perform a study on the dense retrieval methods on tabular data for label prediction tasks. First, we investigate whether the dense vectors can be a more effective medium to perform data retrieval than the sparse vectors. Intuitively, the

representation vector learned by a supervised deep model for label prediction is a naturally more informative indexer than the raw sparse-feature data. Thus the close neighbor vectors in such a latent space would be more helpful in making the label prediction of the target instance. Second, as the dense vector of each data instance can be learned by gradient, we explore whether an effective auxiliary learning objective would help the dense retrieval yield higher prediction performance. Particularly, based on a retrieved neighbor set, we leverage a supervised contrastive learning loss, which shortens the distance between the target instance and the neighbor one with the same label, and enlarges the distance between the target instance and the neighbor one with different labels (in the binary classification task). Such an auxiliary loss is proved to be effective in tuning the dense vector of the data instances and yields a further performance gain in dense vector retrieval enhanced label prediction. Third, at the stage of aggregating the retrieved instance vectors, previous methods simply leverage an attention aggregation over the embedding vectors of the retrieved instance, which usually drops the patterns of feature interactions and ignores feature-label interactions. To this end, we design a label-aware interaction layer, which explicitly explores feature interactions and feature-label interactions of each retrieved instance before taking the aggregation. Such a layer design is proved to be effective in learning the representation of the retrieved instance and thus boosts the performance of label prediction. The proposed framework is named DERT, which is short for dense retrieval on tabular data.

To sum up, the main technical contributions of this paper are threefold.

- We propose DERT, a novel framework that learns the dense representation vectors of the tabular data instances and performs retrieval-enhanced deep learning for the target data label prediction. To our knowledge, DERT is the first work on dense retrieval for tabular data prediction.
- Taking advantage of the learnable dense vectors, we incorporate a novel supervised contrastive learning in the retrieval batch as an auxiliary loss in DERT, which demonstrates the effectiveness of representation learning on the tabular data in DERT.
- In the final predictor, We design a novel label-aware interaction layer, which manages to perform fine-grained feature interactions and feature-label interaction pattern mining for further improving the label prediction performance of DERT.

We conduct extensive experiments on five real-world tabular datasets, where DERT achieves consistent performance gain over the state-of-the-art methods for tabular data label prediction with the tasks of CTR prediction and top-N ranking.

We claim that DERT opens a new direction of data science on tabular data, which is significant. First, dense vector based retrieval-enhanced deep learning yields a new dimension of model capacity, i.e., the capacity from the retrieved data instances and their labels. Such a framework makes use of single-instance representation learning on tabular data. Second, with the tabular data encoder, DERT can be seamlessly extended to retrieve over other types of data, such as text and images. Finally, it is notable that DERT could inspire a new paradigm of tabular data index and retrieval in database applications since dense retrieval may inspire data science on fuzzy search on multiple tables beside a single table.

Notation	Description
$\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{retrieval}}$	Training set, test set, retrieval set
$\mathbb{D}_{\cdot}$	The embeddings of the data instances in the corresponding set
X_z, x_z	The raw feature and the embedding of z-th sample
X_t, x_t	The raw feature and the embedding of target sample
E, ω	The encoder and its parameters
R, ϕ	The retriever and its parameters
f, θ	The predictor and its parameters
d	The embedding vector dimension
S	The set of retrieved data instances
$\mathbb{S}$	The set of retrieved data instance embeddings
k	The size of retrieval set

Table 1: Notations and corresponding descriptions.

The remaining part of this paper is organized as follows. We first present the problem formulation of DERT in Section 2. In Section 3, the overall DERT framework and the detailed model instantiation are discussed. We report the experimental results in Section 4. Then we discuss the related work in Section 5. Finally, we conclude this paper in Section 6.

2 PROBLEM FORMULATION

This paper focuses on predicting a single column for a table,which is a common task in data science with various applications. We follow the task formulation of RIM [36] which is the first model applied retrieval-based method on tabular data.

In recent years, many sequential models in tabular data are proposed. These models usually aim to capture the sequential pattern of a user's behavior. For this purpose, the common way is to use some architectures like RNN, Transformer, and other attention mechanisms to leverage the most recent consecutive data [35, 52, 53]. The other way is to retrieve relevant data from history [36, 37] and feed the retrieved instances and the target data instance into a predictor.

In such a setting, the entire dataset could be split into three disjoint parts as $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{retrieval}}$, representing the subset allowed for supervision, the subset for evaluation, and the subset allowed as alternative help rather than direct supervision respectively.

In order to use tabular data's deep representation to help data instance (X_z, y_z) find its neighbors. The framework has three coupled components: an *encoder* that maps each tabular data sample X_z into a dense vector x_z, a *retriever* that uses the target sample as the query and retrieves the neighbor samples from $\mathcal{D}_{\text{retrieval}}$, and a *predictor* that makes the label prediction given the retrieved set S_z and x_z.

Different from the representation learning in other fields such as NLP [21, 25], where the data encoder is usually trained with unsupervised learning methods, e.g., self-supervised learning, for tabular data, as the final task is the label prediction, we can leverage the label-discriminative representation learned by a supervised model to build the encoder and then use an unsupervised method to boost the encoder further to learn a better dense vector representation. And we use the same encoder to encode target instance and retrieval dataset. Generally, the training objective of the encoder E_ω parameterized by ω is written as

$$\omega^* = \arg\min_{\omega} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(X_z, y_z) \in \mathcal{D}_{\text{train}}} \mathcal{L}_{\text{encoder}}(E_\omega(X_z), y_z). \quad (1)$$

We could use the E_{ω^*} to get dense vector representation x_z for every tabular data sample X_z in $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{retrieval}}$. Then for each sample X_z we can obtain the neighbor set S_z using the retriever R_ϕ as

$$S_z = R_\phi(x_z, \mathcal{D}_{\text{retrieval}}), \tag{2}$$

where the retrieverR_ϕ is written as parameterized by ϕ. Generally, R_ϕ could be trained using with typical learning-to-rank (LTR) formulation [30], and it may also be implemented with some non-learning methods.

With the data sample X_z and the retrieved set S_z, the predictor makes the final label prediction, denoted as $\hat{y}_z = f_{\theta^*}(x_z, S_z)$. Thus, the optimization objective of the prediction model f_θ can be written as

$$\theta^* = \arg\min_\theta \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x_z, y_z) \in \mathcal{D}_{\text{train}}} \mathcal{L}_{\text{prediction}}(f_\theta(x_z, S_z), y_z). \tag{3}$$

3 THE DERT MODEL

In this section, we present the details of dense retrieval on tabular data (DERT). As mentioned in Section 2, there are three major components in DERT, namely, encoder, retriever, and predictor. The encoder E_ω encodes each data sample in $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{test}}$, $\mathcal{D}_{\text{retrieval}}$. The retriever R_ϕ fetches the relevant samples from $\mathcal{D}_{\text{retrieval}}$. The predictor f_θ makes use of the target sample features X_z and the retrieved set S_z to make the label prediction.

We will first take a brief overview of the whole DERT framework and then discuss the three components separately. Finally, we will perform an analysis of the time complexity of DERT.

3.1 Framework Overview

Figure 1 illustrates the overview of DERT framework and its difference from previous work. The goal of the DERT framework is to index all the data in $\mathcal{D}_{\text{retrieval}}$ in low-dimensional and continuous spaces such that the retriever could make use of the semantic similarity between data samples to find the top-k relevant samples in the retrieval pool in order to help the predictor make the label prediction for any target instance X_t.

First, we train the encoder E for each target instance X_t and $\mathcal{D}_{\text{retrieval}}$. Although in principle the encoder could be implemented by any parametric model such as neural network and we could use independent encoders for samples in $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{retrieval}}$. Particularly, we use an identical encoder to encode all samples from the training set and retrieval set. Then we build a dense vector index on the $\mathcal{D}_{\text{retrieval}}$ for the retriever. For implementation, we use the offline deployment of FAISS (Facebook AI Similarity Search) [20], which is an open-source library for efficient similarity search and clustering of dense vectors.

At run-time, given a target instance X_t, we use the encoder E to project the target instance from sparse format X_t to the d-dimensional real-valued vectors x_t. The target instance x_t is then used to retrieve top-k instances S_t from $\mathcal{D}_{\text{retrieval}}$. After obtaining the dense vectors of the target instance and the retrieved ones, we feed x_t and S_t into the predictor $f_\theta(x_t, S_t)$. In predictor, there is an aggregation layer using an attention mechanism to aggregate the embeddings in S_t to a constant length dense vector. In addition, the label of instances in S_t is also essential, we both aggregate embeddings and labels in S_t. After the aggregation layer, we concatenate the output vector and the target x_t and then input the vector to a multi-layer perceptron (MLP) to generate the output label prediction.

3.2 Encoder

The encoder E_ω maps a data instance X to a d-dimensional real-valued vector. The encoder is shared among the retrieval pool $\mathcal{D}_{\text{retrieval}}$, the training set $\mathcal{D}_{\text{train}}$ and the test set $\mathcal{D}_{\text{set}}$.

We can use any neural network to encode the data instances as long as it can handle discrete values. In this work, we use RIM [36] as our encoder since it is the state-of-the-art deep model on tabular data prediction. In our experiments, we encode the categorical values into embedding vectors, and pass the concatenation into an MLP. Namely, consider a data instance $X_z = \{c_i^z\}_{i=1}^F$, where c_i^z is the categorical value on the i-th column of data instance X_z. We encode each c_i^z with a column-specific learnable embedding matrix e_i and pass it into an MLP:

$$x_z = E_\omega(X_z) = MLP(\|_{i=1}^F e_i(c_i^z)), \tag{4}$$

where the encoder's parameters ω consist of the embedding matrices e_i and the MLP's weights.

For training the encoder, we use both supervised learning and unsupervised learning methods, as described in detail below.

Supervised learning loss. We use the data in $\mathcal{D}_{\text{train}}$ to train the encoder. The loss function form follows the original tabular prediction task, e.g. cross entropy for binary classification:

$$\mathcal{L}_{\text{label}} = -\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{\mathcal{D}_{\text{train}}} (y_z \log \bar{y}_z + (1 - y_z) \log(1 - \bar{y}_z)), \tag{5}$$

$$\bar{y}_z = \text{sigmoid}(w^\top x_z + b), \tag{6}$$

where y_z is the label of the target data instance z, and w and b are learnable parameters. If the original tabular prediction task is regression, we could change the supervised learning loss to mean squared error.

Note that although we are training the encoder with supervision from the training set labels, we do not directly use the encoder to make predictions. Instead, the encoder is a mere intermediate step that produces dense embeddings for further use in the retriever R and downstream predictor f_θ.

Contrastive learning loss. The second task is the contrastive learning task, which provides an auxiliary loss to the supervised learning loss. It is generally difficult to judge whether the two sentences or images are in the same group. In contrast, tabular data (X_z, y) is composed of categorical features and numerical features which are normally discretized into categorical features for the unified processing by neural networks [7, 12]. The feature columns of tabular data could be features such as "Age", "Sex", "City", and etc. It is very easy for computers to identify two samples of tabular data by categorical features.[16, 45, 48] So the important thing for contrastive learning is to distinguish positive and negative pairs from the *similar samples* in the dataset. Previous contrastive learning methods on tabular data overlook this problem [1]. As such, the existing negatives sampling method from a random batch is not suitable for contrastive learning.

We propose a *retrieval-based negative sampling* method for tabular data.

Given a sample $X_z = \{c_i^z\}_{i=1}^F$ in $\mathcal{D}_{\text{train}}$, we retrieve k-most relevant instances in D_{train} according to the relevance value defined as

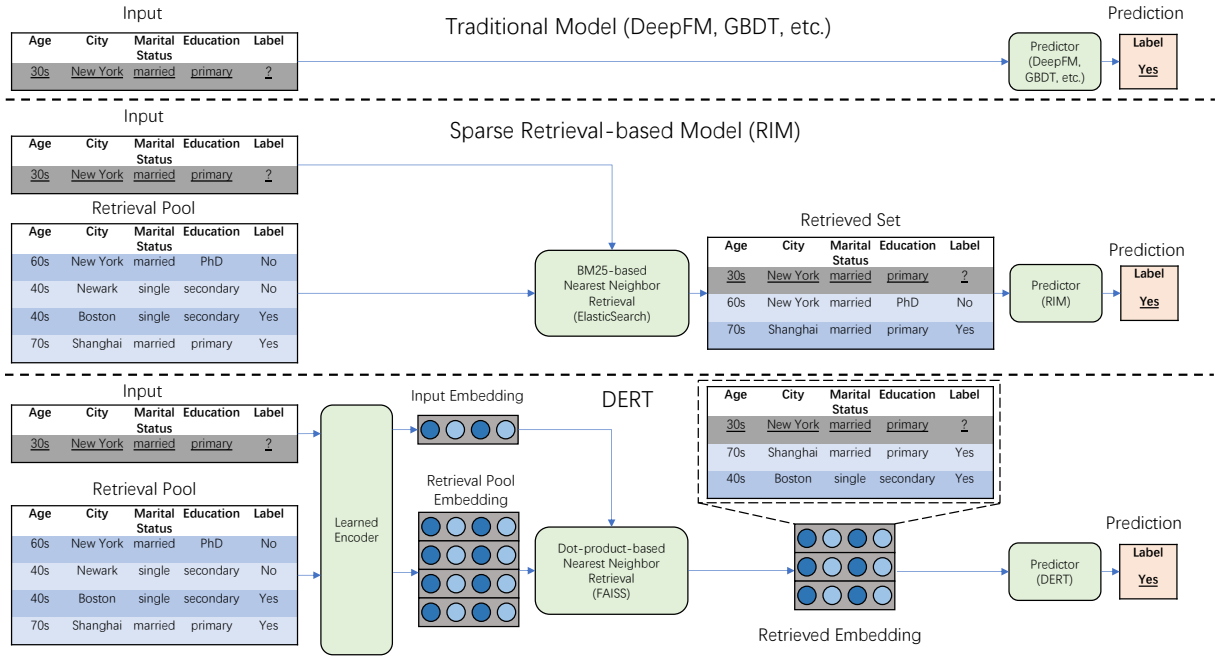

Figure 1: An overview of DERT framework and its comparisons to traditional models and a previous retrieval-based model. The model in the top subfigure is the traditional model. They only take the sparse feature vector as the input and output the prediction result. The middle subfigure illustrates the diagram of the sparse retrieval model, e.g., RIM [36]. RIM makes use of the sparse data in two ways. The common way is to use the feature as the input of the prediction model. The other way is to use data as the retrieval set. RIM uses the target instance as a query and performs the sparse retrieval method BM25 to find relevant instances in the retrieval set. Our model DERT is illustrated the bottom subfigure, where the dataset is split in the same way as RIM. The key difference lies in that we adopt a learnable encoder to obtain the dense vector of each data instance and use the vector of the target instance to find the relevant data instances in the retrieval set.

below:

$$\text{Relevance}(X_z, X_j) = \sum_{i=1}^{F} \text{IDF}(c_i^z) \cdot \mathbf{1}(x_i^z = x_i^j), \tag{7}$$

$$\text{IDF}(x_i^z) = \log \frac{N - N(c_i^z) + 0.5}{N(c_i^z) + 0.5}, \tag{8}$$

where $\mathbf{1}(\cdot)$ is the indicator function, N is the number of data instances in $\mathcal{D}_{\text{train}}$ and $N(x_f^z)$ is the number of data instances that have feature value x_f^z. This follows the practice of RIM [36], where it is shown to be equivalent to BM25 [41].

Let $\hat{S}_z$ be the set of instances retrieved from above, we now compare the labels of X_z and $\hat{S}_z$. For binary classification tasks, we partition $\hat{S}_z$ into a *positive set* $\hat{S}_z^+$ and a *negative set* $\hat{S}_z^-$, with the former containing the instances with the same label as X_z and the latter containing those with different labels. Additionally, we follow SimCSE [8] to perform dropout on x_z to generate x_z' so that we always have one (artificial) instance with the same label. Thus, the contrastive loss can be defined as

$$\mathcal{L}_{\text{contra}} = -\log \sum_{|\mathcal{D}_{\text{train}}|} \frac{e^{\text{dist}(x_z, x_z')/\tau} + \sum_{X_j^+ \in \hat{S}_z^+} e^{\text{dist}(x_z, x_j^+))/\tau}}{\epsilon + \sum_{X_j^- \in \hat{S}_z^-} e^{\text{dist}(x_z, x_j^-)/\tau}}, \tag{9}$$

where the distance function $\text{dist}(\cdot, \cdot)$ can be defined as the L2 distance of two dense vectors, ϵ is a constant value in the denominator, τ is a hyperparameter, and $x_\cdot^\cdot = E_\omega(X_\cdot^\cdot)$ as we recall from Eq. 4.

Moreover, the contrastive learning loss $\mathcal{L}_{\text{contra}}$ does not rely on other parts of DERT, and it can be applied to any tabular data related neural networks, such as DeepFM [13], DCN [46], and RIM [36] to enhance these models with a better tabular data representation and achieve improved performance for their downstream tasks.

To sum up, the learning objective of the encoder of DERT is a linear combination of the supervised learning loss and the contrastive learning loss with a tradeoff hyperparameter λ as

$$\mathcal{L}_{\text{encoder}} = \mathcal{L}_{\text{label}} + \lambda \mathcal{L}_{\text{contra}}. \tag{10}$$

3.3 Retriever

Give the insight that similar tabular data instances could help make the label prediction of the target instance, the goal of the retriever R is to find the most relevant samples of target instance X_t in the retrieval set $\mathcal{D}_{\text{retrieval}}$. For using dense representation to find the most relevant samples, we first use the encoder E_ω to embed $\mathcal{D}_{\text{retrieval}}$ as

$$\mathbb{D}_{\text{retrieval}} = \{E_\omega(X) : X \in \mathcal{D}_{\text{retrieval}}\}. \tag{11}$$

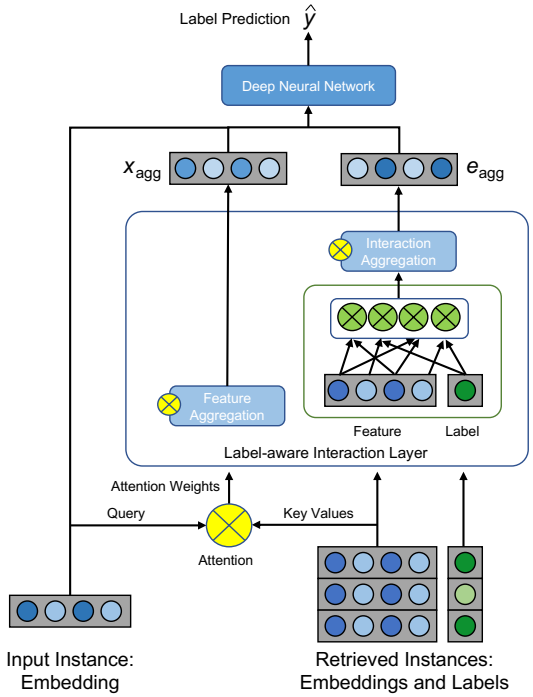

Figure 2: The architecture of predictor in DERT.

Then given the data instance to predict X_t, we encode it to a dense embeddings $x_t = E_\omega(X_t)$, and retrieve $\mathbb{S}_t \subseteq \mathbb{D}_{\text{retrieval}}$, the set of top-$k$ relevant instances sorted by the dot product between their dense embeddings:

$$\text{sim}(x_t, x_i) = x_t^\top x_i, x_i \in \mathbb{D}_{\text{retrieval}}. \tag{12}$$

We additionally retrieve the ground truth labels of the retrieved subset $\mathbb{Y}_t = \{y_z : X_z \in \mathbb{S}_t\}$.

This retrieval procedure is known as Maximum Inner Product Search (MIPS), which has highly-efficient sub-linear approximate solutions, e.g. FAISS [20].

3.4 Predictor

The predictor takes in the embedding x_t of the target instance X_t, the set of embeddings $\mathbb{S}_t$ as well as their labels $\mathbb{Y}_t$ yielded by the retriever R, and produces the prediction $\hat{y}_t$.

Attention weight computation. Before feeding x_t, $\mathbb{S}_t$ and $\mathbb{Y}_t$ to the final prediction layer, we aggregate the $\mathbb{S}_t$ to a single vector with the same size of x_t. We use attention mechanism to do so. Let $\mathbb{S}_t = \{x_1, x_2, ..., x_k\}$ and $\mathbb{Y}_t = \{y_1, y_2, ..., y_k\}$. The attention weight is defined as

$$\alpha_i = \frac{e^{x_i^\top \boldsymbol{W} x_t}}{\sum_{j=1}^k e^{x_j^\top \boldsymbol{W} x_t}}, \tag{13}$$

where $\boldsymbol{W}^{d\times d}$ is the attention layer parameter matrix.

Label-aware interaction layer. RIM's predictor aggregates the retrieved instance set $\mathbb{S}_t$ as well as $\mathbb{Y}_t$ separately. This does not allow pairwise interaction between the features and labels within each individual instance in $\mathbb{S}_t$, which we claimed and verified to be important. To this end, we leverage an interaction layer which transfers each x_i and its label y_i to a d-dimensional vector $e_i \in \mathbb{R}^d$ that enables explicit feature-feature and feature-label interactions, denoted as

$$e_i = \text{interaction}(x_i, y_i). \tag{14}$$

We instantiate interaction$(\cdot,\cdot)$ with PNN [38], one of the earliest explicit feature interaction models with product operators. Finally, the aggregated embedding e_{agg} can be defined as

$$e_{\text{agg}} = \sum_{i=1}^k \alpha_i e_i. \tag{15}$$

Output layer. Besides the label-aware interaction layer, we also apply the attention coefficient to S_z, and the aggregated retrieved data representation is calculated as

$$x_{\text{agg}} = \sum_{i=1}^k \alpha_i x_i, \tag{16}$$

We then feed the concatenation of x_z, x_{agg} and e_{agg} into an MLP to produce the final output as

$$\hat{y}_z = \rho(\text{MLP}([x_z, x_{\text{agg}}, e_{\text{agg}}])). \tag{17}$$

The $\rho(\cdot)$ is a non-linear layer with sigmiod function when the task is binary prediction task and the $\rho(\cdot)$ is a linear function when the task is regression task. The loss function is different for different tasks. We use cross-entropy loss for binary prediction and ranking task. For regression task, we use mean square error loss.

3.5 Time Complexity During Inference

The difference of DERT from other traditional tabular data prediction models is the usage of encoder E_ω and retriever R. We note that once the encoder E_ω is learned, the embeddings of the retrieval pool $\mathbb{D}_{\text{retrieval}}$ can be done offline and reused thereafter. Therefore, here we just analysis the dense retrieval operation in DERT. As discussed in Section 3.3, we use FAISS to implement the retriever. FAISS has an optimal time complexity of $O(\log\log|\mathcal{D}_{\text{retrieval}}|)$ [20], giving approximate solutions to MIPS problems that are usually good enough.

4 EXPERIMENTS

In this section, we evaluate DERT[1] on two types of tasks: click-through-rate (CTR) prediction (which is binary classification) and top-N ranking. Additionally, we conduct several ablation studies to validate the components of our model. The specific research questions to study from the experiments are as follows.

- **RQ1** Does DERT beat all baselines and does the dense retrieval outperform sparse retrieval in tabular data?
- **RQ2** How do the proposed modules, i.e., contrastive learning, label-aware interaction, and retrieval method, have an impact on the prediction?
- **RQ3** How do the hyperparameters (λ and k) influence the performance of DERT?
- **RQ4** Is DERT efficient enough in the inference stage?

[1]The experiment code is available at https://anonymous.4open.science/r/DERT-1C7B.

4.1 Datasets

We conduct experiments for CTR prediction on three large-scale datasets, i.e., Tmall[2], Alipay[3], Taobao[4]. For top-N ranking, we use two widely-used public recommendation datasets, i.e., MovieLens[5], LastFM[6]. For MovieLens, we designate the instances with rating greater than or equal to 4 as positive examples, and the rest as negative examples. The detailed information of the five datasets is summarized in Table 2.

Datasets	# Instances	# Fields	Task
Taobao	100,150,807	4	CTR Prediction
Tmall	54,925,331	9	CTR Prediction
Alipay	35,179,371	6	CTR Prediction
Movielens-1M	1,000,209	7	Top-N Ranking
LastFM	18,993,371	5	Top-N Ranking

Table 2: Dataset statistics.

We split each dataset following the protocols in the previous work RIM [36], where the oldest data instances are grouped into the retrieval set, the most recent data instances form the test set, while the intermediate data instances form the train set. Although we are aware of other possible settings of data splitting, such as stacking, due to the page limit we focus on the above one and leave the empirical study in other settings in future work.

4.2 Evaluation Metrics

For CTR prediction, we use the evaluation metrics including area under the ROC curve (AUC) and negative log-likelihood (LogLoss). For top-N ranking, we use hit ratio (HR@N), normalized discounted cumulative gains (NDCG@N), and mean reciprocal rank (MRR). Significance test on each metric between the first and second performed methods is conducted, with * marked for positive test results.

4.3 Compared Methods

On CTR prediction, we compare DERT with 8 strong baselines which can be divided into three categories. The first category contains traditional tabular models which could not use the retrieval set. GBDT [3] is a widely used tree model and DeepFM [13] is a common feature-interaction-based deep model. For fair comparison, we allow these models to train on the retrieval set as well, in addition to the training set. The second category contains end-to-end sequential CTR models. DIN [53] and DIEN [52] are examples that are attention-based and recurrent. The third category contains retrieval-based models including UBR [37] and RIM [36] which only retrieves data instances based on raw input features directly. In addition, FATE [47] is a tabular data representation learning model that takes in a random minibatch of data instances as whole input and outputs the prediction of all its elements simultaneously, allowing interactions between the samples in the minibatch. This can be viewed as leveraging a random retrieval method to make prediction.

On top-N ranking, we compare DERT with 6 baselines. FPMC [40] and TransRec [14] are factorization-based models. Other baselines NARM [26], GRU4Rec [17], SASRec[22], and RIM [36] are recently proposed neural network models.

4.4 Overall Performance (RQ1)

For CTR prediction, The overall performance comparison is provided in Table 3, from which one can have the following observations. (i) DERT consistently outperforms all the baselines on CTR prediction. Specifically, compared to the best baseline RIM, DERT achieves the improved AUC by 0.98%, 0.68% and 1.01% on three datasets on Taobao, Tmall, and Alipay, respectively. This demonstrates the effectiveness of dense retrieval that manages to fetch semantically relevant neighbor instances. (ii) The retrieval-based methods DERT and RIM are superior to the traditional methods and sequential CTR models. The result indicates that the retrieval methods are able to make use of the long-term history data, which helps the prediction model work better. We show the results on top-N ranking in Table 4. One can observe that DERT achieves significant improvements over these baselines in almost all the metrics on the two datasets. That shows dense retrieval works well in top-N ranking tasks.

4.5 Ablation Study (RQ2)

4.5.1 Contrastive Learning. In this section, we discuss the impact of adding contrastive loss in encoder training. We first verify the effectiveness on the encoder's learned representations by reporting the AUC of the encoder's own prediction (i.e. $\bar{y}_z$ in Eq. 6) versus the ground truth. Table 6 shows the result where the performance without contrastive learning (named **No CL** in the table) was consistently lower, showing that contrastive learning indeed made encoder's representation better. Table 5 also shows the end-to-end impact of the contrastive loss on the dense retrieval. From the comparison between the first row and the third row of each dataset, we could conclude that the contrastive loss could help the downstream prediction model work better as well.

4.5.2 Label-aware Interaction Layer. We further study the usage of the label-aware interaction layer in DERT. The result can be found in Table 5 by comparing the first row and the second row in each dataset. The result of DERT with label interaction and DERT without label interaction both use the label information of the retrieved set. The difference is that we remove the interaction of label and feature interaction in the ablation experiment which is the way of previous retrieval-based methods (e.g., RIM). After retrieval, the previous retrieval-based methods only use an attention mechanism to aggregate the feature embeddings and labels respectively, which ignore the interaction between features and the label. From Table 5, we could find that the label-aware interaction layer improves the prediction consistently, which indicates that it is important to explore the fine-grained feature-label interaction patterns in each retrieved instance before aggregating them.

4.5.3 Retrieval Mechanism in Contrastive Learning. As mentioned in Section 3.2, contrastive learning in our encoder involves retrieving relevant data instances using BM25 to generate the positive and negative sets in Eq. 9. Here we compare it against retrieving the

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=53
[4]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[5]https://grouplens.org/datasets/movielens/1m/
[6]http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html

Models	Taobao			Tmall			Alipay		
	LogLoss	AUC	Rel. Impr.	LogLoss	AUC	Rel. Impr.	LogLoss	AUC	Rel. Impr.
GBDT	0.6797	0.6134	40.97%	0.5103	0.8319	10.59%	0.9062	0.6747	19.86%
DeepFM	0.6497	0.6710	28.87%	0.4695	0.8581	7.21%	0.6271	0.6971	16.01%
FATE	0.6497	0.6762	27.88%	0.4737	0.8553	7.56%	0.6199	0.7356	9.94%
DIN	0.6086	0.7433	16.33%	0.4292	0.8796	4.59%	0.6044	0.7647	5.75%
DIEN	0.6084	0.7506	15.20%	0.4445	0.8838	4.10%	0.6454	0.7502	7.80%
SIM	0.5795	0.7825	10.50%	0.4520	0.8857	3.87%	0.6089	0.7600	6.41%
UBR	0.5432	0.8169	5.85%	0.4368	0.8975	2.51%	0.5747	0.7952	1.70%
RIM	0.4644	0.8563	0.98%	0.3804	0.9138	0.68%	0.5615	0.8006	1.01%
DERT	**0.4486***	**0.8647***	-	**0.3585***	**0.9200***	-	**0.5319***	**0.8087***	-

Table 3: Performance comparison of CTR prediction task baselines. GBDT and DeepFM are the traditional methods. Others are sequential modeling methods. For fair comparison, traditional models are trained on both the retrieval set and the training set. The best results are in bold fonts while the second best results are underlined. "Rel. Impr." means the relative AUC improvement of DERT against each baseline. Improvements are statistically significant with $p < 0.01$.

Datasets	Metric	FPMC	TransRec	NARM	GRU4Rec	SASRec	RIM	DERT
ML-1M	HR@1	0.0261	0.0275	0.0337	0.0369	0.0392	0.0645	**0.0747***
	HR@5	0.1334	0.1375	0.1418	0.1395	0.1588	0.2515	**0.2540***
	HR@10	0.2577	0.2659	0.2631	0.2624	0.2709	0.4014	**0.4035***
	NDCG@5	0.0788	0.0808	0.0866	0.0872	0.0981	0.1577	**0.1634***
	NDCG@10	0.1184	0.1217	0.1254	0.1265	0.1341	0.2059	**0.2117***
	MRR	0.1041	0.1078	0.1113	0.1135	0.1193	0.1704	**0.1774**
LastFM	HR@1	0.0148	0.0563	0.0423	0.0658	0.0584	0.0915	**0.1488***
	HR@5	0.0733	0.1725	0.1394	0.1785	0.1729	0.3468	**0.3742***
	HR@10	0.1531	0.2628	0.2227	0.2581	0.2499	**0.5780***	0.5597
	NDCG@5	0.0432	0.1148	0.0916	0.1229	0.1163	0.2165	**0.2620***
	NDCG@10	0.0685	0.1441	0.1185	0.1486	0.1409	0.2911	**0.3217***
	MRR	0.0694	0.1303	0.1083	0.1362	0.1289	0.2210	**0.2694***

Table 4: Performance comparison of sequential top-N ranking task in terms of HR@N, NDCG@N, and MRR. The best results are in bold values while the second best results are marked with underline. Significant improvements are judged with $p < 0.01$.

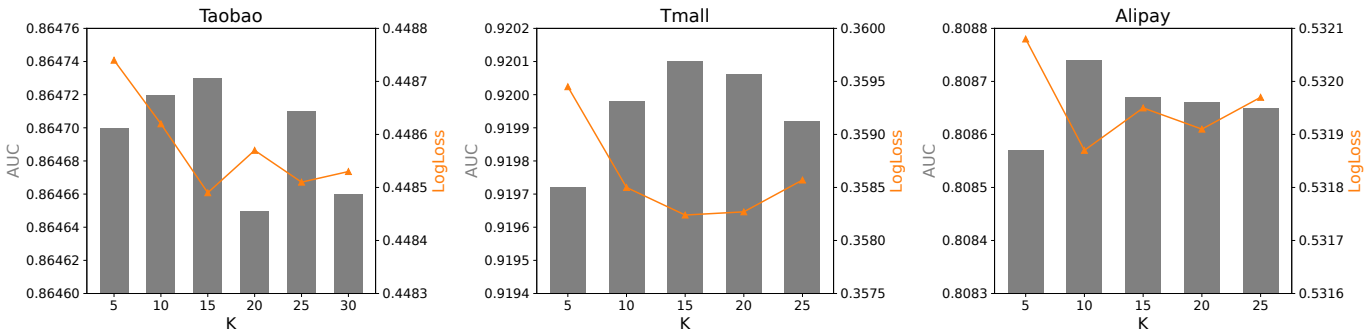

Figure 3: Performance of DERT w.r.t. different retrieval sizes k.

data instances uniformly at random. From Table 6, we observe that the relevance retrieval method incorporated in contrastive learning yields better encoder prediction AUC, hence better encoder representations, than random retrieval.

4.6 Hyperparameter Study (RQ3)

Here we study two hyperparameters of DERT, namely, the hyperparameter λ in Eq. (10) for tuning the weight of contrastive loss when training the encoder, and the hyperparameter k for the number of retrieved instances in DERT.

Dataset	Contrastive loss	Label Interaction	LogLoss	AUC
Taobao	×	×	0.4491	0.8643
	×	✓	0.4490	0.8644
	✓	×	0.4488	0.8646
	✓	✓	**0.4486**	**0.8647**
Tmall	×	×	0.3641	0.9173
	×	✓	0.3586	0.9199
	✓	×	0.3628	0.9185
	✓	✓	**0.3585**	**0.9200**
Alipay	×	×	0.5346	0.8065
	×	✓	0.5336	0.8081
	✓	×	0.5335	0.8078
	✓	✓	**0.5319**	**0.8087**

Table 5: Ablation Study on contrastive learning and label-aware interaction layer.

Dataset	No CL		Random Retrieval		Relevance Retrieval	
	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC
Taobao	0.4644	0.8563	0.4567	0.8580	**0.4528**	**0.8583**
Tmall	0.3804	0.9138	0.3721	0.9146	**0.3678**	**0.9148**
Alipay	0.5615	0.8006	0.5376	0.8012	**0.5363**	**0.8022**

Table 6: Encoder's prediction performance with and without contrastive learning (CL) and relevance-based retrieval.

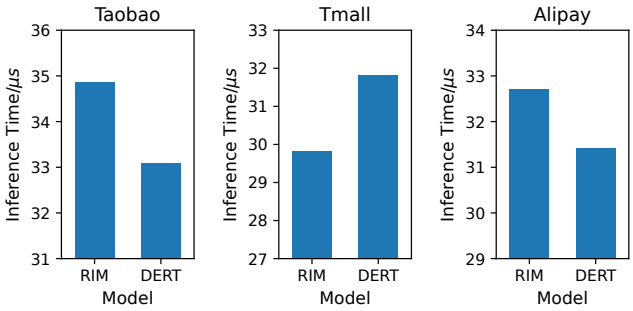

Figure 4: Predictor inference time of DERT and RIM.

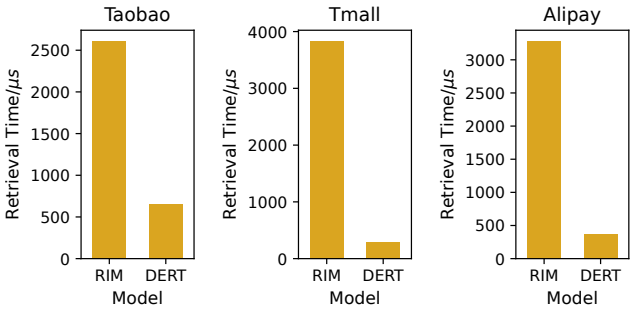

Figure 5: Retrieval time of DERT and RIM.

From Figure 6, we can observe that when λ is large, contrastive loss has a negative impact on the performance because the loss is not related label directly and it affects the label prediction loss. If we tune λ to a proper value, it can help the encoder achieve better performance. Furthermore, we can find that generally the better encoder (with higher AUC) will help DERT get a better result (higher AUC), which further verifies our intuition that if we use a more effective encoder, we will get better retrieval results in DERT.

From Figure 3, we could find that the AUC bars of the three datasets are similar. If k is too small, the retrieved set could not carry sufficient information for the prediction model, while if k is too large, the retrieved set would include too much noise which may harm the performance.

4.7 Study of Efficiency (RQ4)

Tabular data prediction models are expected to be deployed in online services, such as advertising platforms, fraud user detection systems, and recommender systems. Hence, it is important for the model to be efficient in the inference stage. We compare the running time of retrieval and prediction processes between dense retrieval (DERT) and sparse retrieval (RIM). In our experiment, both predictors took roughly the same time during inference, while for the retrieval process, dense retrieval is substantially faster than sparse retrieval. Furthermore, we note that the scale of y-axis in predictor inference time (Figure 4) is orders of magnitude smaller than that in retrieval time (Figure 5), implying that DERT also optimized a major bottleneck of retrieval-based models.

5 RELATED WORK

5.1 Deep Learning on Tabular Data

Tabular data is an alternative but major data form to vision, audio, and language, which is widely used in various application fields including web services, banking, government, education, etc. The cell of the tabular data is an either discrete or numeric value. Generally, as deep neural networks may not deal with such a hybrid form of data, it is common to discretize the numeric values [12], then each row of the table can be regarded as a multi-field categorical data [51]. As such, a majority of deep learning models on tabular data prediction focus on exploring the feature interactions within one instance, i.e., in a row of the tabular data [13, 14, 28, 39]. Wide&Deep network [5] builds fully connected (FC) layers based on the feature embeddings of the fields to implicitly mine the feature interactions. CCPM [29] leverages the convolution operators to explore the local feature interaction patterns based on a list of feature embeddings. DeepFM [13] extends from factorization machine (FM) [40] and Wide&Deep network to build a two-branch architecture to perform explicit and implicit feature interaction learning simultaneously. Fi-GNN [27] regards each data instance as a fully connected graph and leverage graph neural network (GNN) to explore the feature interactions. Despite numerous models proposed in this direction, the model capacity may be limited as the input of the model is a single instance, which requires the model to encode all the knowledge in the parameters.

5.2 Retrieval-based Prediction Model

Retrieval-based methods for prediction on tabular data generally originate from sequential recommendation tasks. It is natural to leverage recurrent neural network (RNN) to build the user profile representation so as to make recommendation predictions. However, when the user's historical behavior sequence is much longer, it is infeasible that RNN architectures could memorize and utilize the

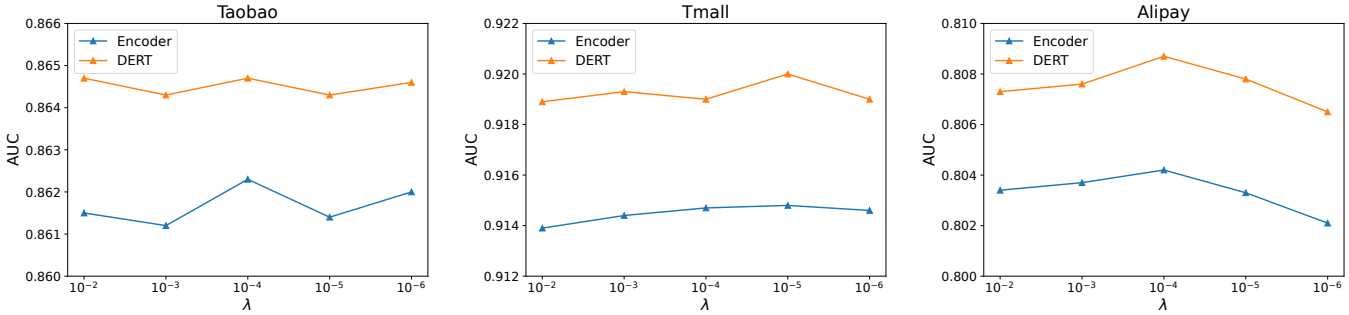

Figure 6: The performance of DERT and the base model (acting as the encoder of DERT) w.r.t. the hyperparameter λ.

very early behavior patterns [34]. As such, retrieval methods play as a practical implementation for replacing the long recurrence and attention operations. UBR [37] builds a search index of the user historical sequence data and uses reinforcement learning to formulate the search query based on current prediction data. The retrieved data is then aggregated via an attention operation and fed into an MLP to make the label prediction. SIM [35] implements both hard search based on item category information and soft search based on item embedding vectors over the backbone model DIEN [52]. RIM [36] further extends the study scenario from the sequential recommendation scenarios to the generic tabular data prediction scenario. The query is the target row data while the retrieved data instances are the relevant rows in the table with BM25 [41] as the predefined relevance function between two rows.

Retrieval-enhanced machine learning has been positioned as an important perspective of the frontier of machine learning research [50]. Retrieval-based methods for prediction tasks have been well explored in natural language processing and computer vision [31]. kNN-LM [22], as its name claims, combines k nearest neighbor and neural language model to perform generalization via memorization of the data. Besides, there is emerging research on reinforcement learning tasks [10].

To our knowledge, for tabular data prediction, there has been no previous work that performs data retrieval based on dense vectors of the data, which is the main focus of our paper.

5.3 Representation Learning of Tabular Data

Compared to the sequential structure of natural language and the matrix structure of images, tabular data is much different. Each row of the table corresponds to a data instance and each column of the table corresponds to a field with a predefined data schema. There is relatively few work on learning the representation of tabular data.

As the cell data of the table has a strong underlying correlation between each other, it is natural to build graph structures for the tabular data and leverage graph neural networks (GNNs) to model the representation of the data. GRAPE [49] builds a data-field bipartite graph, where the two sides of nodes are data instance (i.e., the row of the table) and field (i.e., the column of the table) while the edges are labeled with the specific category of the cell. As such, GRAPE formulates the feature imputation task as edge-level prediction while formulating the label prediction task as node-level prediction, both of which can be solved with GNNs. Compared to GRAPE, FATE [47] builds a different form of a bipartite graph, where one side is the data instances while the other side is the specific feature categories. With such a setting, FATE can perform inductive learning and extrapolate in the representation space.

TabTransformer [18] learns the contextual embeddings based on the original embedding of each cell data of the table row using column embedding, which is followed by an MLP to make label prediction of the tabular data. TABBIE [19] learns the representation of cells, rows, and columns of the table with a scheme of cell corruption & identification. With text as part of the data, TaPas [16] and TaBERT [48] leverage the BERT model with masking recovery training objective to pre-train the representation of tabular data and text. TURL [6] is a table structure-aware Transformer model that works on the tabular of text cells.

Note that most representation learning models that map the tabular data instance to a dense vector can be leveraged as the encoder implementation of DERT.

6 CONCLUSION

In this paper, we propose DERT, the first framework that learns dense vector representation of tabular data instances and based on it performs dense retrieval to improve tabular data prediction performance. Compared to previous sparse retrieved-based methods on tabular data, the dense retrieval method in DERT makes use of semantic information rather than just term matching to better distinguish positive and negative cases from top retrieved candidates. In all experiments, DERT achieves superior performance over strong baselines, which indicates that DERT can make better use of data from the retrieval set than previous sparse retrieval.

With the dense vector representation learning for tabular data, DERT opens a new direction of data science. In the long run, tabular data prediction tasks would leverage external data sources, not only the sources with the same format. Retrieval with dense representation can help tabular prediction models make use of any embedding data. DERT also bridges the gap between tabular data and other types of data such as text, images, and graphs. If we treat the dataset as a table assembled by many rows of tabular data, the tabular data prediction task can be regarded as a task based on one table in a database. The dense retrieval method could help the model work across multiple tables. DERT is potentially capable of enhancing these end-to-end tabular data prediction models.

ACKNOWLEDGMENTS

Thanks to the anonymous reviewers for improving this work.

REFERENCES

[1] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. 2021. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147* (2021).

[2] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[3] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. 1996. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering* 8, 6 (1996), 866–883.

[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *1st DLRS workshop*. 7–10.

[6] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[7] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. 2021. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 322–331.

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*. Association for Computational Linguistics (ACL), 6894–6910.

[9] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.

[10] Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, et al. 2022. Retrieval-augmented reinforcement learning. In *International Conference on Machine Learning*. PMLR, 7740–7765.

[11] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress.

[12] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An embedding learning framework for numerical features in ctr prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2910–2918.

[13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *IJCAI* (2017).

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[15] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*. 1–9.

[16] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4320–4333.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

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

[19] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3446–3456.

[20] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[21] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[22] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations*.

[23] Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Thomas Rainforth, and Yarin Gal. 2021. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems* 34 (2021), 28742–28756.

[24] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past performance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 768–776.

[25] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[26] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[27] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*.

[28] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. *KDD* (2020).

[29] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A convolutional click prediction model. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 1743–1746.

[30] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[31] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. 2022. Retrieval augmented classification for long-tail visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6959–6969.

[32] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1222–1230.

[33] Salima Omar, Asri Ngadi, and Hamid H Jebur. 2013. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications* 79, 2 (2013).

[34] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2671–2679.

[35] Pi Qi, Xiaoqiang Zhu, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[36] Jiarui Qin, Weinan Zhang, Rong Su, Zhirong Liu, Weiwen Liu, Ruiming Tang, Xiuqiang He, and Yong Yu. 2021. Retrieval & Interaction Machine for Tabular Data Prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1379–1389.

[37] Jiarui Qin, W. Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Y. Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[38] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[39] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *TOIS* 37, 1 (2018), 1–35.

[40] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[41] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[42] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. 2021. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342* (2021).

[43] Stéphane Tufféry. 2011. *Data mining and statistics for decision making*. John Wiley & Sons.

[44] Dejan Varmedja, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, and Andras Anderla. 2019. Credit card fraud detection-machine learning methods. In *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*. IEEE, 1–5.

[45] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: table convolutional network for web table interpretation. In *Proceedings of the Web Conference 2021*. 4020–4032.

[46] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *ADKDD*. 1–7.

[47] Qitian Wu, Chenxiao Yang, and Junchi Yan. 2021. Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach. In *Advances in Neural Information Processing Systems: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*.

[48] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 8413–8426.

[49] Jiaxuan You, Xiaobai Ma, Daisy Yi Ding, Mykel J. Kochenderfer, and Jure Leskovec. 2020. Handling Missing Data with Graph Representation Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

[50] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-Enhanced Machine Learning. *arXiv preprint arXiv:2205.01230* (2022).

[51] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data: A Case Study on User Response Prediction. *ECIR* (2016).

[52] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.

[53] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*.