BERT4CTR: An Efficient Framework to Combine Pre-trained Language Model with Non-textual Features for CTR Prediction

Dong Wang, Kavé Salamatian, Yunqing Xia, Weiwei Deng, Qi Zhang

BERT4CTR: An Efficient Framework to Combine Pre-trained Language Model with Non-textual Features for CTR Prediction

Dong Wang
STCA, Microsoft Corporation
Beijing, China
donwa@microsoft.com

Kavé Salamatian
University of Savoie
Annecy, France
Kave.salamatian@univ-smb.fr

Yunqing Xia
STCA, Microsoft Corporation
Beijing, China
yxia@microsoft.com

Weiwei Deng
STCA, Microsoft Corporation
Beijing, China
dedeng@microsoft.com

Qi Zhang
STCA, Microsoft Corporation
Beijing, China
zhang.qi@microsoft.com

ABSTRACT

Although deep pre-trained language models have shown promising benefit in a large set of industrial scenarios, including Click-Through-Rate (CTR) prediction, how to integrate pre-trained language models that handle only textual signals into a prediction pipeline with non-textual features is challenging.

Up to now, two directions have been explored to integrate multi-modal inputs in fine-tuning of pre-trained language models. One consists of fusing the outcome of language models and non-textual features through an aggregation layer, resulting into ensemble framework, where the cross-information between textual and non-textual inputs are learned only in the aggregation layer. The second one consists of splitting and transforming non-textual features into fine-grained tokens that are fed, along with textual tokens, directly into the transformer layers of language models. However, by adding additional tokens, this approach increases the complexity of the learning and inference.

We propose in this paper, a novel framework, BERT4CTR, that addresses these limitations. The new framework leverages Uni-Attention mechanism to benefit from the interactions between non-textual and textual features, while maintaining low training and inference time-costs, through a dimensionality reduction. We demonstrate through comprehensive experiments on both public and commercial data that BERT4CTR outperforms significantly the state-of-the-art approaches to handle multi-modal inputs and is applicable to CTR prediction. In comparison with ensemble framework, BERT4CTR brings more than 0.4% AUC gain on both tested data sets with only 7% increase on latency.

CCS CONCEPTS

• **Information systems** → **Online advertising**; *Recommender systems*; *Language models.*

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599780

KEYWORDS

Non-textual features; Multi-modal inputs; Pre-trained language model; CTR prediction; Uni-Attention

ACM Reference Format:
Dong Wang, Kavé Salamatian, Yunqing Xia, Weiwei Deng, and Qi Zhang. 2023. BERT4CTR: An Efficient Framework to Combine Pre-trained Language Model with Non-textual Features for CTR 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, 12 pages. https://doi.org/10.1145/3580305.3599780

1 INTRODUCTION

Machine learning has frequently to deal with multi-modal inputs that are mixing numerical, ordinal, categorical, and textual data. This is especially the case for the Click-Through-Rate (CTR) prediction, *i.e.*, predicting the likelihood that a candidate ad shown after an entered query on a search engine, will be clicked based on the semantic relevance between the query and the candidate ad description that is textual, and the user's attributes such as user's ID, user's gender, user's category, *etc.*, which are non-textual. Pre-trained language models like BERT [4] and RoBERTa [16], which can make use of the semantic relationship between words in natural language texts, have shown to be beneficial for improving the accuracy of CTR prediction. However, combining pre-trained language models that only handle textual features, with numerous non-textual features into the CTR prediction pipeline, is still challenging. Pre-trained language models have already been used in classical CTR prediction, through adding the final score [32], or intermediate embedding after fine-tuning with textual signals [18], leading to cascading workflow. However, such frameworks cannot leverage, in the fine-tuning of language models, the cross-information between textual and non-textual signals. In this paper, we confirm that learning such cross-information can improve the accuracy of CTR prediction, and the goal of this paper is to design an efficient framework doing such information fusion at the first stage of the fine-tuning process. It is noteworthy that although CTR prediction is the main application in this paper, the approach developed here can be extended to a large set of applications that have to deal with multi-modal inputs in pre-trained language models.

Up to now, two directions have been explored for integrating multi-modal inputs in the fine-tuning of pre-trained language models. In the first approach, called here as "Shallow Interaction", the

language model with textual input is treated as a separated and specific network, and the outcome of this network (final output score or [CLS] pooling layer) is fused into the other network dealing with non-textual inputs through an aggregation layer. This approach has been adopted in [2][21], resulting into ensemble learning framework. In [30] an in-depth analysis of this approach is presented. In this approach, interaction between textual and non-textual features happens only in the last aggregation layer. As a consequence, cross-information between textual and non-textual inputs are not exploited enough for fine-tuning the language model. In the second class of approach, non-textual features are directly fed as the inputs of transformer layers in the language model. This makes possible to leverage the non-textual inputs at the beginning stage of model learning. Such an approach is at the core of VideoBERT [27], VL-BERT [26] and NumBERT [34], where non-textual signals, such as images or numbers, are split into fine-grained fragments (*e.g.*, regions-of-interest in images or digits) each of which is transformed as a new token and combined with textual tokens. However, there might be hundreds of non-textual features for CTR prediction, and these long additional inputs complicate the computations and make the time-costs in learning and inference of model intractable.

Given the limitations of these two approaches, we introduce in this paper a simple and light framework, named *BERT4CTR*, to handle multi-modal inputs mixing textual and non-textual features in pre-trained language models. Our approach is based on a *Uni-Attention* mechanism that combines semantic information coming from textual features, with cross-information between textual and non-textual features. We further apply a dimensionality reduction operation, in order to decrease the time-costs in both learning and inference. Besides, a two-steps joint-training is introduced to fine-tune the model and further improve the accuracy of prediction. The proposed approach scales well with the expected growing of the number of non-textual features, for improving CTR prediction.

Through empirical evaluation on both commercial and public data, we show that BERT4CTR significantly improves the CTR prediction, in comparison with the state-of-the-art approaches that combine textual and non-textual features, while keeping low latency both in training and inference. In particular, our results indicate that increasing the number of non-textual inputs can enhance the advantages of BERT4CTR, *e.g.*, on the public data set with 57 non-textual features, BERT4CTR shows a significant gain of 0.7% for the Area Under the ROC Curve (AUC) along with a decrease in training cost of 30%, and a decrease in inference cost of 29%, compared with NumBERT. On the commercial data set with 90 non-textual features, BERT4CTR provides an AUC gain of 0.6%, and a decrease in training cost of 64%, and in inference cost of 52%.

In section 2, we present the related work. The section 3 introduces the design of BERT4CTR. The evaluation is presented in section 4. Finally, we provide concluding remarks.

2 RELATED WORK

This section presents the related works on handling multi-modal inputs which combine non-textual features with pre-trained language models, and its application to CTR prediction.

2.1 Multi-modal Inputs Handling

The issue of handling multi-modal inputs that are mixing textual and non-textual input, and integrating semantic insights coming from pre-trained language models like BERT [4] has been already investigated in the literature VideoBERT [27], VL-BERT [26], NumBERT [34] and CTR-BERT [21]. The approach followed in these works consists of splitting the non-textual signals into fine-grained fragments, each of which is transformed as a new token and combined with textual tokens as the inputs of transformer layers. However, the addition of tokens representing the non-textual features complicates the language model structure and can make the learning and inference phases too costly for model updating and online serving.

2.2 Models for CTR Prediction

CTR prediction is one of the major practical applications of deep learning. Clicks made on advertisements or candidates shown along with search results, or web content presentation, are the main source of revenue for a large set of web actors. In this context, models are always used to select high quality advertisements or candidates to present according to the web contents, *e.g.*, in sponsored search engines [11] or personal recommendation systems [24], which should both achieve low-latency and high-accuracy. For example, the CTR prediction model in Baidu.com uses a deep neural network, called *Phoenix Nest*, fed with a handcrafted set of features extracted from the user, the query, and advertisement properties [5]. Google Ads is using the "Follow The Regularized Leader" (FTRL) model to predict CTR [19], while Google play is using a Wide & Deep model described in [3]. In [23] the "Product based Neural Network" (PNN) model is introduced to capture interactive patterns between features. This PNN model is extended in [7] to DeepFM model that emphasizes the interactions between low- and high-order feature. Microsoft Bing.com has adopted a Neural Network boosted with GBDT ensemble model [14] for ads CTR prediction, which is the commercial scenario we are considering through this paper. The features used in these CTR prediction models can be grouped into two categories: the raw texts from user, query and ad, and the other being the non-textual features including the attributes of users and items such as gender, age, UserId, AdId, *etc.* and the outputs generated from sub-models, such as LR model [14][19], pre-trained language model [18][32], *etc.*.

2.3 Application of Pre-trained Language Models in CTR Prediction

Recent work has shown the abilities of pre-trained language models to extract deep relationship in a sentence pair [4][16][13][28][15], that are useful for augmenting the semantic features of query and recommendation pair in CTR prediction [18][32][12][8][33]. Generally, the pre-trained language models are trained against the real click data, targeting directly the prediction of click/non-click labels. Thereafter, the score from the final layer [32][12], or the embedding from the intermediate layer [18][8] of these fine-tuned language models is used as an additional NLP input feature in the CTR prediction model. For example, Microsoft Bing Ads uses the embedding from the hidden layer of TwinBERT model as a semantic feature [18] while Meituan.com and JD.com use the output score

of BERT [32][12]. Besides that cascading framework, some works consider the fusion between the outputs of language models and non-textual features through an aggregation layer, resulting into ensemble learning frameworks (called "Shallow Interaction" in this paper), such as BST [2] and CTR-BERT [21], and [30] has done an in-depth analysis on the Shallow Interaction frameworks, where the cross-information between textual and non-textual inputs are not dug enough to fine tune the language models.

3 DESCRIPTION OF BERT4CTR

3.1 Problem Statement

CTR prediction models are using multi-modal inputs, mixing different textual features, denoted as $\mathcal{T} = \{t_1, t_2, ..., t_N\}$, like searching query, titles and URLs of potential ads to show, and non-textual features, denoted as $C = \{c_1, c_2, ..., c_M\}$, of different type, *e.g.*, dense features such as historical CTR of the query, last click time of the user, *etc.*, and sparse features, like ID, category of user, *etc.*.

The learning process of pre-trained language models, with textual features alone to fine-tune over click data for CTR prediction, can be formalized as calibrating a network that approximates the conditional probability of all outcome alternatives, click or non-click for CTR application, given the textual contexts of query and candidates:

$$P_{click} = P(click = 1|\mathcal{T}) \tag{1}$$

As stated above, the non-textual features are crucial in CTR prediction and they should not be ignored. When non-textual features are added, the conditional probability becomes:

$$P_{click} = P(click = 1|\mathcal{T}, C) \tag{2}$$

The goal in this paper is to design an efficient network structure that can generate scores approximating the distribution P_{click} in Equation 2, while maintaining acceptable training and inference time-costs for industrial application.

3.2 Model Design

We describe here the evolution of our proposed network structure, BERT4CTR, by beginning with the NumBERT framework and gradually adding new components to it.

3.2.1 NumBERT Description. NumBERT [34], is the widely-used systematic approach to integrate textual and numerical features in pre-trained language models. Pre-trained language models like BERT, along with a large class of neural networks, are using attention layers, that enhance over time some part of the input to enable the training process to concentrate on learning them. In each attention layer, a feed-forward network and a residual network are used to control the Vanishing/Exploding gradient issue [9]. NumBERT uses a similar structure, with several layers of bidirectional self-attention. The core idea in NumBERT is to replace all numerical instances with their scientific notation representations, *i.e.*, the number 35 is replaced by "35 [EXP] 1", where [EXP] is a new token that is added to the vocabulary. These transformed non-textual inputs are thereafter considered as normal texts and are fed to the language model. For the CTR prediction application, several transformed non-textual inputs might be concatenated using separator token [SEP], to distinguish one numerical feature from other, generating a long string of text which is appended to the end of $< query, ad >$ textual input, and is used for the fine-tuning of language model on click data. Figure 1 depicts an example of transformation from original non-textual features to transformed inputs in NumBERT.

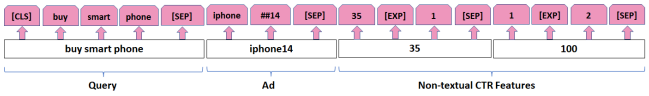

Figure 1: Example of transformed and concatenated non-textual and textual inputs for NumBERT

While NumBERT approach enables the language model to understand the numbers in the non-textual signals, the model still misses two crucial issues. First, the contextual relationship between textual features and non-textual ones might not be obvious. For example, the numerical features such as the historical CTR of user, the ID of user *etc.*, are less correlated with the semantics of the $< query, ad >$ texts. The second issue is related to the fact that the positions of these transformed tokens from non-textual features do not bear semantic meanings as normal texts, *i.e.*, the numerical features in CTR prediction models are always independent of each other. These two limitations indicate that sharing the same attention weights and mechanisms for textual features and the transformed non-textual ones is not optimal. Thus, simply using NumBERT to integrate non-textual inputs cannot improve the performance of learning objectives well, as will be shown later in Section 4.

3.2.2 Uni-Attention. To address these two issues, we have improved the architecture of NumBERT. We are using the same bidirectional self-attention mechanism as in NumBERT with inputs only from textual tokens. However, for non-textual part, a new type of attention mechanism is introduced, called *Uni-Attention*. It is still a Query-Key-Value (QKV) attention function [1], where the Query is coming only from non-textual tokens, while the Key and Value are coming from textual tokens in the same layer, *i.e.*, in the calculation of uni-attention on each token in non-textual part, one input is the matrix projected from the value of that token itself, and the other input is the matrix projected from values of all tokens in textual part. In the uni-attention mechanism, the non-textual components have no positional-embedding, which avoids the issue on positional semantics described above. Moreover, this hybrid framework allows the tokens in textual part to dig deep for semantic relationship between each other by the aid of pre-trained attention weights, while grasping the cross-information between textual and non-textual ones in parallel.

Feed-forward and residual networks are also used on each uni-attention output to control the Vanishing/Exploding gradient issue. We show in Figure 2 the "Uni-Attention" design. In the last attention layer, all uni-attention outputs from transformed non-textual inputs are gathered as a single hidden layer, which is concatenated with the [CLS] pooling layer from textual part. Thereafter, the concatenated layer is fed to a MultiLayer Perception (MLP) that will finally predict the probability of click/non-click. We will show in Section 4 that the proposed design improves strongly the final AUC for both commercial and public data, compared with simple NumBERT.

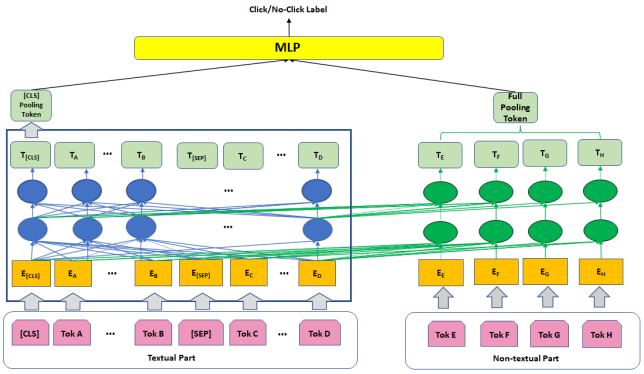

Figure 2: Framework of Uni-Attention

3.2.3 Dimensionality Reduction. The number of non-textual features used in industry for CTR prediction models can be very large, *e.g.*, Microsoft Bing Ads uses 90 numerical features transformed each into 4 tokens (accounting for the [EXP] and [SEP] tokens). This large size of inputs impacts negatively the learning cost and the prediction latency in CTR prediction.

One way to solve this issue is to apply a dimensionality reduction on the non-textual features. Such approaches have already been explored in several previous works like [3][7]. Following these works, our approach consists of representing each non-textual feature in C as a N-dimensional point in space. The resulting $N \times |C|$ space is then mapped to a K-dimensional embedding ($K \ll N \times |C|$) through a fully connected network that is fed, along with the embedding from textual tokens, to the layer calculating the uni-attentions.

The mapping to the initial N-dimensional space depends on the non-textual features being dense, *e.g.*, the length of the query, the historical value of CTR *etc.*, or being sparse, *e.g.*, user's gender, query's category *etc.*. For sparse features, we use an embedding table that defines for each given value the corresponding N-dimensional embedding. Dense features, on the other hand, are first normalized using a max-min normalization and thereafter expanded into a 101-dimensional one-hot vectors with 0.01 buckets, used as index in an embedding table containing the N-dimensional embeddings. We show in Figure 3 the embedding of non-textual features used in BERT4CTR.

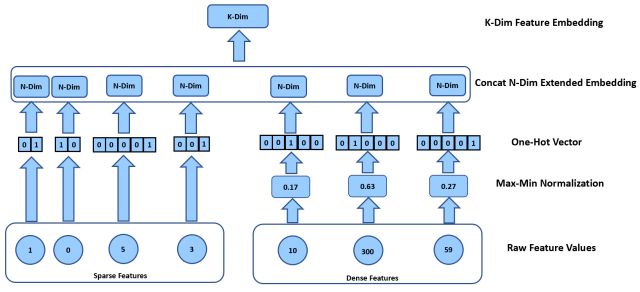

Figure 3: Dimensionality Reduction on embedding of non-texual features in BERT4CTR

Similarly to NumBERT, the attention alignment score in textual part is calculated as a dot-product form where the dimensions of Query and Key are the same. However, after dimensionality reduction, there is no guarantee that the dimensions of embedding in the textual and non-textual parts are equivalent.

Considering the flexibility of our model, we use for non-textual part, an additive attention, known as Bahdanau attention [1], that consists of a feed-forward network with a single hidden layer, to calculate the attention alignment score. The formula for deriving the attention alignment score between Query and Key is as follows:

$$f_{att}(Q, K) = v_a^T tanh(W_a[Q; K]) \tag{3}$$

where $[Q; K]$ is the concatenation of Query and Key, and v_a and W_a are learned attention parameters. It is shown in [29] that additive and dot-product attentions are equivalent, while additive one does not require Query and Key with same embedding dimensions.

In Section 4, we will show, in Table 3 and Table 4, that the dimensionality reduction operation proposed here can hold more than 90% of the best AUC achieved with uni-attention while substantially reducing the time-costs of training and inference.

3.2.4 Two-steps Joint-training. The calibration of BERT4CTR consists of jointly training both textual and non-textual features. It is shown in [30] that a two-steps training can significantly improve the accuracy of prediction of such joint-training framework, and inspired with this, BERT4CTR is trained in two-steps too. In the first step, called *warm-up step*, we pre-train the standard language model with only textual features using a Mask Language Model (MLM) task, and then we fine-tune this model on the same textual data with click label. The non-textual part, with dimensionality reduction, is also pre-trained using a MultiLayer Perceptron (MLP) that predicts the click probability using non-textual features alone. This pre-training phase will calibrate the parameters of the dimensionality reduction using a cross entropy loss function. The second step of training, called *joint-training step*, is initialized with the pre-trained textual and non-textual weights, and continues the training of the whole network of BERT4CTR, by mixing textual and non-textual inputs, with a small learning rate.

We demonstrate this two-steps joint-training in Figure 4. The results in Section 4 will show that the two-steps joint-training provides significant AUC gain on both commercial and public data sets.

4 EXPERIMENTS AND EVALUATIONS

In this section, we evaluate BERT4CTR over two data sets: one, called commercial data set, is from Microsoft Bing Ads, and the other one, called public data set, is from KDD CUP 2012. We will first describe the experimental settings of the data sets, the pre-trained language models, the baselines, the evaluation metrics and the environments used. We then compare four incrementally completed versions of the proposed framework, followed by the introductions in Section 3, *i.e.*, NumBERT alone, with uni-attention added, with dimensionality reduction for non-textual feature embedding added, and with the two-steps joint-training. This incremental addition will show the improvements coming from each individual component.

We will also compare BERT4CTR with three current state-of-the-art frameworks handling multi-modal inputs in CTR prediction.

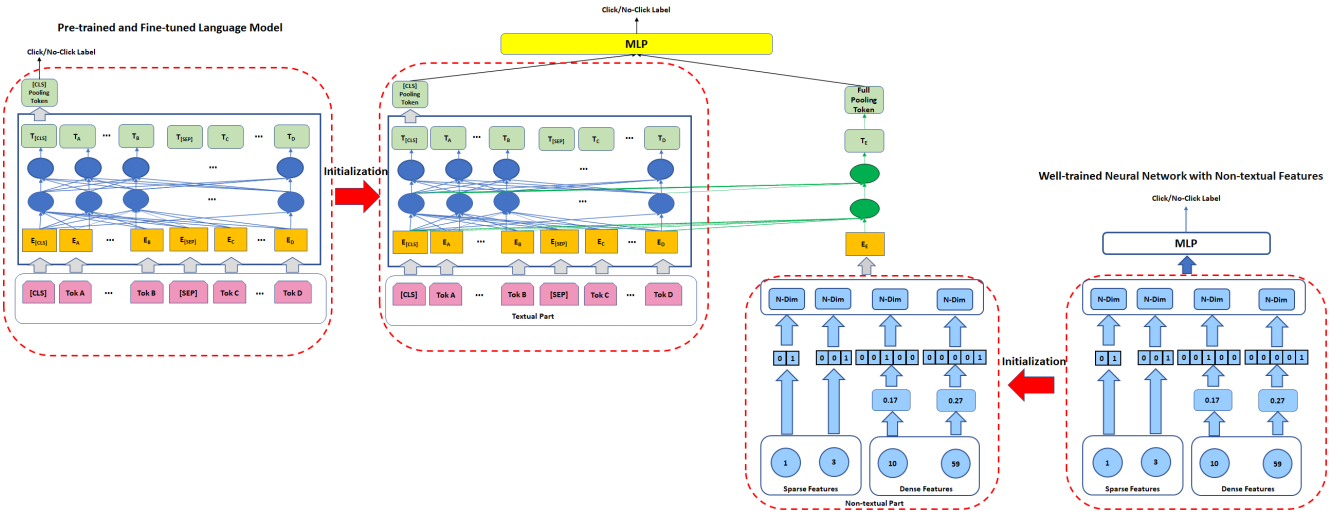

Figure 4: Two-steps Joint-training in BERT4CTR

These comparisons will provide evidences that BERT4CTR is an efficient framework to combine pre-trained language model with non-textual features for CTR prediction.

4.1 Experimental Settings

4.1.1 Data Sets. We use the following two data sets for evaluation. Moreover, to evaluate the robustness of our work, the experiments on different data sets are also based on different pre-trained language models.

Microsoft Bing Ads Data Set: Microsoft Bing Ads is a commercial system used by Microsoft to select the ads presented to users after a searching request. This data set consists of 190 million $< query, ad >$ pairs with click labels, which are randomly sampled from Bing Ads logs obtained in April 2022. The samples in the first three weeks of April are used as training set, and the remaining as validation set. Similarly to [30], we use the text of the query, and ad title concatenated with ad display URL as two textual features. In addition, a set of 90 non-textual features are also available in this data set, which can be categorized as: (1) dense features representing continuous numerical values, such as historical value of CTR per user, number of historical impressions per ad *etc.*; (2) sparse features representing discrete values, such as the user's gender, searching query's category *etc.*; (3) ads position, a special feature in CTR prediction. As in [14] and [30], the displayed position of an ad is assumed to be independent of other features, *i.e.*, for calculating the likelihood of a click, we assume that the displayed position and the quality of an ad are independent.

KDD CUP 2012 Data Set[1]: We are also using in our experiments a public data set coming from KDD CUP 2012 Track 2 [22][31]. This data set contains 235 million $< query, ad >$ pairs with click label, sampled from the logs of Tencent search engine Soso.com. This data set contains 57 non-textual features which can also be classified into dense and sparse features, along with the position of ads.

However, differently from Bing Ads, in this data set, there is no time information, meaning that it is not possible to split the training and validation data based on time. Thus, we have generated the training and validation set by randomly selecting 1/11 of samples as validation data and the remaining as training data.

4.1.2 Pre-trained Language Model Settings. The textual part of Bing Ads data set is initialized over the RoBERTa-Large model with 24 layers (abbreviated as RoBERTa-24) created by Facebook [16]. Similarly to [4][30],the pre-training of the RoBERTa-24 model is done using the popular Mask Language Model (MLM) task.

For the textual part in KDD CUP 2012 data set, a BERT-Base model with 12 layers (abbreviated as BERT-12) [4] is pre-trained with MLM task and then used as initial model for further experiments.

To enable reproducibility, we present all the details of the experiments run on the KDD CUP 2012 data, including the data preprocessing steps, hyperparameters settings and pseudocodes, in the appendix.

4.1.3 Baseline Setups. We compare here BERT4CTR with three state-of-the-art frameworks handling pre-trained language model and non-textual features for CTR prediction.

The first baseline framework, called *Cascading Framework*, is a traditional way to introduce pre-trained language models in CTR prediction. It consists of injecting the outcome (final score or intermediate embedding) of a language model fine-tuned on the textual inputs alone as a new input feature, along with the non-textual features, for the CTR prediction. Here, we first fine-tune one language model (RoBERTa-24 or BERT-12) with only $< query, ad >$ textual pairs, and then feed the predicted score of this fine-tuned language model as a new feature into a downstream CTR prediction model. To show the generality of our work, we choose three different CTR prediction models: (1) Wide & Deep [3], introduced by Google, which combines a shallow linear model with a deep neural network; (2) DeepFM [7], an improved version of Wide & Deep, which replaces the linear model with a Factorization-Machine (FM);

[1]https://www.kaggle.com/c/kddcup2012-track2

(3) NN boosted GBDT [14], used for Microsoft Bing Ads, that consists of a Neural Network (NN) boosted with a Gradient Boosting Decision Tree (GBDT) ensemble model.

The second baseline is called *Shallow Interaction Framework*, which is also widely used in practice [2][32][21]. It consists of fusing the non-textual embedding layer and the last layer of pre-trained language model, *e.g.*, the [CLS] pooling layer, through an aggregation layer. We use two variants of this approach: the first one, called *Shallow Interaction-1 Layer*, connects the language model and the non-textual embedding layer directly through a MultiLayer Perception (MLP). The second one, called *Shallow Interaction-N Layers*, uses the same number of feed-forward network (FFN) and residual network layers stacked above the non-textual embedding layer, as the ones used in the language model, followed by a MLP. The second variant provides a more fair comparison, as the depths of network in textual and non-textual part are the same as BERT4CTR.

The third baseline is the NumBERT framework [34] described in Section 3.2.1.

4.1.4 Evaluation Metrics. We use in our evaluations, the Area Under the ROC Curve (AUC) [6] and Relative Information Gain (RIG) [20], as two crucial metrics to evaluate the performance of a predictive model. Besides the measurements on the whole validation data (called as *ALL Slice*), we also focus on the infrequent $< query, ad >$ pairs (called *Tail Slice*), which could lead to cold starting problem in CTR prediction. As reported in [7], 0.1% improvement on AUC or RIG can be seen as a significant gain for industrial use. We also use t-test results with $\alpha = 0.05$ to compare the performances of different models, *i.e.*, a difference between two AUCs (or RIGs) with t-value larger than 3 can be considered as significant [17].

Besides the AUC and RIG, we also use, as two additional performance metrics, the average, median, 90th percentile and 95th percentile of the time-costs (milliseconds per sample), both for training and inference. These two metrics are important for CTR prediction in practice. First, the CTR prediction models must adapt to user's interest drift, through frequent updates, *e.g.*, the CTR model is refreshed weekly in Microsoft Bing Ads. Therefore, the training time should be less than the refreshing interval. Second, the online serving latency is directly related to the time-cost in inference, and should be as low as possible. It is noteworthy that for different frameworks, the calculations of time-cost are also different. In terms of cascading framework, the training/inference time-cost is the sum of the training/inference time-cost of the language model and the one of downstream CTR prediction model. For both Shallow Interaction and BERT4CTR which need two-steps joint-training, the training time-costs are calculated as the sum of time taken in warm-up step and in joint-training step. While the time-cost in training of NumBERT is the time only taken in pre-training and fine-tuning. The time-costs in inference of all these three no-cascading frameworks are measured as the time taken in single prediction by language models.

4.1.5 Environments. All model evaluations are implemented using TensorFlow and running on NVIDIA V100 GPUs with 32 GB memory. The maximum lengths of sequence for $< query, ad >$ are set as 64, and the batch sizes are set as 10 on both RoBERTa-24 and BERT-12 model. It is noteworthy that the maximum length of sequence for NumBERT is set to $64+4\times|C|$, where $|C|$ is the number of non-textual features used, since each non-textual feature can be split into 4 tokens in NumBERT, as stated in Section 3.2.1. To account the random variation, each experiment for time-cost is repeated for twenty times to obtain metrics. Without explicit statement, all AUCs and RIGs shown in this section are obtained at the best step during training.

4.2 Performance of Components in BERT4CTR

In this section, we evaluate the improvement on model performance coming from each individual component of BERT4CTR described in Section 3.

4.2.1 NumBERT's Performance. We present in Table 1 the performance of NumBERT for CTR prediction over the two data sets used in this paper. Here, two baseline models can be used for comparison. The first one, called *TextOnly*, uses the pre-trained language model fine-tuned with only $< query, ad >$ textual input and without any non-textual features. The second one is the Shallow Interaction-1 Layer described above. We show in the table along with absolute value of AUC and RIG for each model, the difference of metrics between two models with t-values, *e.g.*, $\Delta\ AUC_{M3-M1}$, the AUC difference between Model 3 (NumBERT) and Model 1 (TextOnly).

One can observe, from Table 1, that NumBERT has been able to benefit from non-textual features. It brings, when compared with the model without non-textual features, 2.7% AUC improvement over Bing Ads data and 6.8% AUC improvement on KDD CUP 2012 data. However, compared with the Shallow Interaction model, NumBERT does not provide benefits neither on AUC nor on RIG, and shows worse performance. This means that even if NumBERT allows textual and non-textual features to interact through complex bidirectional self-attention with multi-layers, it is not efficient in learning the cross-information between multi-modal signals.

4.2.2 Uni-Attention's Performance. We follow up with evaluation of the improvements coming from uni-attention architecture. We show in Table 2 the performance achieved by NumBERT compared with *NumBERT + Uni-Attention*, *i.e.*, transformed non-textual features (as depicted in Figure 1) are fed to uni-attention architecture, as shown in Figure 2.

Table 2 shows that the uni-attention architecture can bring significant gains both in terms of AUC and RIG, compared with the NumBERT model without uni-attention, over both data sets. For example, the uni-attention architecture can bring additional 0.3% AUC gain and 0.5% RIG gain on Tail Slice of Bing Ads data. These gains are even more obvious for KDD CUP 2012 data, where the AUC gain is 0.5% and the RIG improves by 0.6% over Tail Slice. All these changes are statistically significant with t-values larger than 70.

4.2.3 Dimensionality Reduction's Performance. Here, we evaluate the impact of dimensionality reduction of non-textual features, shown in Figure 3, that is made mandatory because of the large number of non-textual inputs in industrial CTR prediction models.

The performances of dimensionality reduction on the two data sets are shown in Table 3, where *NumBERT + Uni-Attention + Dimensionality Reduction* is the NumBERT model with uni-attention framework, as in Figure 2, that is completed with a dimensionality

Dateset	Slice	Model 1		Model 2		Model 3		$\Delta\ AUC_{M2-M1}$		$\Delta\ RIG_{M2-M1}$		$\Delta\ AUC_{M3-M1}$		$\Delta\ RIG_{M3-M1}$		$\Delta\ AUC_{M3-M2}$		$\Delta\ RIG_{M3-M2}$	
		TextOnly		Shallow Interaction - 1 Layer		NumBERT													
		AUC	RIG	AUC	RIG	AUC	RIG	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T
Bing Ads	ALL	0.8691	0.4987	0.8968	0.5360	0.8961	0.5348	0.0277	187.59	0.0373	191.72	0.0270	185.07	0.0361	188.95	-0.0007	2.97	-0.0012	3.73
	Tail	0.7703	0.4382	0.8084	0.4726	0.8078	0.4719	0.0381	181.50	0.0344	183.06	0.0375	178.81	0.0337	180.34	-0.0006	2.39	-0.0007	2.96
KDD CUP	ALL	0.7591	0.3917	0.8286	0.4842	0.8273	0.4827	0.0695	135.68	0.0925	191.72	0.0682	132.17	0.0910	187.66	-0.0013	5.11	-0.0015	5.62
	Tail	0.6757	0.2768	0.7537	0.3739	0.7521	0.3724	0.0780	142.24	0.0971	185.87	0.0764	136.09	0.0956	182.23	-0.0016	5.15	-0.0015	4.74

Table 1: AUC and RIG performance of NumBERT on two data sets

Dataset	Slice	Model 1		Model 2		$\Delta\ AUC_{M2-M1}$		$\Delta\ RIG_{M2-M1}$	
		NumBERT		NumBERT + Uni-Attention					
		AUC	RIG	AUC	RIG	Diff	T	Diff	T
Bing Ads	ALL	0.8961	0.5348	0.8988	0.5397	0.0027	70.06	0.0049	73.91
	Tail	0.8078	0.4719	0.8111	0.4772	0.0033	76.08	0.0053	75.08
KDD CUP	ALL	0.8273	0.4827	0.8311	0.4875	0.0038	82.13	0.0048	80.70
	Tail	0.7521	0.3724	0.7569	0.3780	0.0048	94.14	0.0056	85.11

Table 2: AUC and RIG performance of Uni-Attention on two data sets

reduction operation in non-textual part, as shown in Figure 3. Table 3 reports that AUC and RIG for both alternative models are close on the two data sets. Besides, no one of the performance differences is statistically significant, *i.e.*, the performance equality hypothesis cannot be refuted.

Besides the accuracy of prediction, the time-costs in training and inference of these two models are also evaluated in Table 4. One can observe that dimensionality reduction reduces strongly the time-cost, up to 45% of training cost and 24% of inference cost on KDD CUP 2012 data, with 57 non-textual features, and up to 68% in training and 43% in inference on Bing Ads data with 90 non-textual features. This means that dimensionality reduction does not entail a significant performance reduction while reducing obviously the time-costs.

Dataset	Slice	Model 1		Model 2		$\Delta\ AUC_{M2-M1}$		$\Delta\ RIG_{M2-M1}$	
		NumBERT + Uni-Attention		NumBERT + Uni-Attention + Dimensionality Reduction					
		AUC	RIG	AUC	RIG	Diff	T	Diff	T
Bing Ads	ALL	0.8988	0.5397	0.8980	0.5393	-0.0008	2.48	-0.0004	1.97
	Tail	0.8111	0.4772	0.8104	0.4766	-0.0007	2.77	-0.0006	2.46
KDD CUP	ALL	0.8311	0.4875	0.8306	0.4869	-0.0005	1.86	-0.0006	2.71
	Tail	0.7569	0.3780	0.7563	0.3774	-0.0006	2.15	-0.0006	1.93

Table 3: AUC and RIG performance of Dimensionality Reduction on two data sets

4.2.4 Two-steps Joint-training's Performance. The last component to be evaluated is the two-steps joint-training described in Section 3.2.4. For this purpose, we compare three initialization approaches for the textual and non-textual parts: (1) Pre-trained but not fine-tuned language model for textual part + Random weights in non-textual part (abbreviated as *No Fine-tuned + Randomly Initialized* in Table 5); (2) Fine-tuned weights in textual part + Random weights in non-textual part (abbreviated as *Fine-tuned + Randomly initialized* in Table 5), where the weights in the textual part are initialized using the language model pre-trained and fine-tuned on our $< query, ads >$ textual pairs, and random initial weights are used in non-textual part; (3) Two-steps joint-training where both weights in textual part and non-textual part are initialized with the weights trained in advance as described in Section 3.2.4. This last setting is the one used for the BERT4CTR model introduced in this paper.

Table 5 shows the AUC/RIG performance of these three settings on both data sets. From this table, one can observe that two-steps joint-training brings significant gain for both data sets. On Bing Ads data, the AUC gain is more than 0.3% , and more than 0.4% over KDD CUP 2012 data. All these gains are shown by the t-tests to be significant.

4.2.5 Aggregated Training Loss. We show in Figure 5 the training loss evaluations for all the alternative models. The aggregated log-loss is derived after training each million samples, and the trends are reported in Figure 5 for the first training epoch over Bing Ads data set.

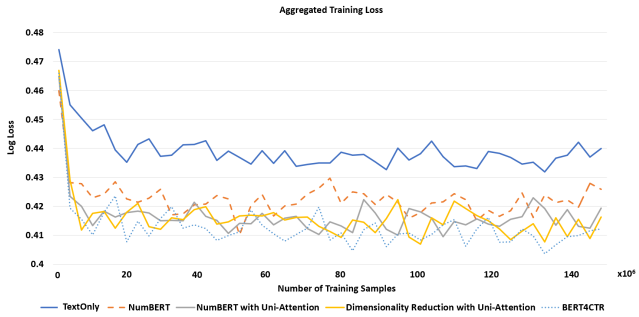

Figure 5: Curves of aggregated training loss on Bing Ads data

The figure leads to four observations. First, the training loss of the model without non-textual features (*i.e.*, TextOnly model) is higher than the ones of the other alternative models, indicating that non-textual features are important for CTR prediction. Second, training loss for NumBERT with uni-attention is below the one of NumBERT, which provides another evidence that uni-attention architecture improves CTR prediction. Third, the training loss curves for NumBERT with uni-attention are close, with and without dimensionality reduction. This means that dimensionality reduction does not compromise the accuracy of prediction much while reducing the time-costs of training and inference. Finally, training loss for BERT4CTR is the lowest one , showing clearly that the two-steps joint-training improves the performance of CTR prediction. These observations are consistent with the ones obtained based on AUC and RIG metrics.

4.3 Comparison of BERT4CTR with Other Multi-modal Frameworks

In this part, we compare BERT4CTR performances with the three alternative frameworks, cascading framework, Shallow Interaction framework and NumBERT, that can handle multi-modal inputs for CTR prediction.

In Table 6, the AUC and RIG performances are shown for all possible alternatives. Three major observations can be extracted from this table. First, cross-information learning between textual

(a) Training Cost

Model	Average		Median		90th percentile		95th percentile	
	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP
NumBERT + Uni-Attention	54.05	11.87	53.76	11.78	54.64	11.92	54.95	11.95
NumBERT + Uni-Attention + Dimensionality Reduction	17.32	6.54	17.16	6.49	17.59	6.61	17.97	6.72

(b) Inference Cost

Model	Average		Median		90th percentile		95th percentile	
	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP
NumBERT + Uni-Attention	12.34	4.43	12.29	4.37	12.50	4.49	12.69	4.58
NumBERT + Uni-Attention + Dimensionality Reduction	7.05	3.36	7.03	3.32	7.19	3.42	7.31	3.51

Table 4: Time-cost performance (ms/sample) of Dimensionality Reduction on two data sets

Dataset	Slice	Model 1		Model 2		Model 3		$\Delta\ AUC_{M3-M1}$		$\Delta\ RIG_{M3-M1}$		$\Delta\ AUC_{M3-M2}$		$\Delta\ RIG_{M3-M2}$	
		No Fine-tuned + Randomly Initialized		Fine-tuned + Randomly Initialized		Two-steps Joint-training (BERT4CTR)									
		AUC	RIG	AUC	RIG	AUC	RIG	Diff	T	Diff	T	Diff	T	Diff	T
Bing Ads	ALL	0.8980	0.5393	0.8985	0.5396	0.9014	0.5413	0.0034	31.37	0.0020	32.13	0.0029	29.71	0.0017	28.74
	Tail	0.8104	0.4766	0.8106	0.4769	0.8136	0.4801	0.0032	42.53	0.0035	41.86	0.0030	32.41	0.0032	39.42
KDD CUP	ALL	0.8306	0.4869	0.8315	0.4881	0.8347	0.4903	0.0041	50.62	0.0034	41.48	0.0032	32.25	0.0022	29.59
	Tail	0.7563	0.3774	0.7571	0.3783	0.7618	0.3821	0.0055	52.14	0.0047	47.95	0.0047	50.73	0.0038	44.56

Table 5: AUC and RIG performance of Two-steps Joint-training on two data sets

(a) AUC Performance

Dataset	Slice	Cascading Framework			Shallow Interaction Framework		NumBERT Framework	BERT4CTR Framework	$\Delta\ AUC_{M7-M1}$		$\Delta\ AUC_{M7-M2}$		$\Delta\ AUC_{M7-M3}$		$\Delta\ AUC_{M7-M4}$		$\Delta\ AUC_{M7-M5}$		$\Delta\ AUC_{M7-M6}$	
		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7												
		Wide & Deep	DeepFM	NN + GBDT	Shallow Interaction - 1 Layer	Shallow Interaction - N Layers	NumBERT	BERT4CTR												
		AUC	AUC	AUC	AUC	AUC	AUC	AUC	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T
Bing Ads	ALL	0.8932	0.8961	0.8956	0.8968	0.8976	0.8961	0.9014	0.0082	73.42	0.0053	52.64	0.0058	55.37	0.0046	53.47	0.0038	47.15	0.0053	60.37
	Tail	0.8043	0.8078	0.8070	0.8084	0.8097	0.8078	0.8136	0.0093	72.68	0.0058	60.13	0.0066	61.32	0.0052	52.76	0.0039	40.89	0.0058	67.51
KDD CUP	ALL	0.8223	0.8278	0.8269	0.8286	0.8304	0.8273	0.8347	0.0124	95.73	0.0069	47.05	0.0078	69.98	0.0061	74.33	0.0043	51.58	0.0074	69.90
	Tail	0.7471	0.7531	0.7526	0.7537	0.7567	0.7521	0.7618	0.0147	106.42	0.0087	65.09	0.0092	71.48	0.0081	74.68	0.0051	60.46	0.0097	79.34

(b) RIG Performance

Dataset	Slice	Cascading Framework			Shallow Interaction Framework		NumBERT Framework	BERT4CTR Framework	$\Delta\ RIG_{M7-M1}$		$\Delta\ RIG_{M7-M2}$		$\Delta\ RIG_{M7-M3}$		$\Delta\ RIG_{M7-M4}$		$\Delta\ RIG_{M7-M5}$		$\Delta\ RIG_{M7-M6}$	
		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7												
		Wide & Deep	DeepFM	NN + GBDT	Shallow Interaction - 1 Layer	Shallow Interaction - N Layers	NumBERT	BERT4CTR												
		RIG	RIG	RIG	RIG	RIG	RIG	RIG	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T	Diff	T
Bing Ads	ALL	0.5321	0.5356	0.5349	0.5360	0.5372	0.5348	0.5413	0.0092	78.46	0.0057	54.89	0.0064	61.73	0.0053	55.82	0.0041	51.34	0.0065	68.89
	Tail	0.4702	0.4734	0.4723	0.4726	0.4738	0.4719	0.4801	0.0099	80.43	0.0067	57.67	0.0078	68.34	0.0075	65.19	0.0063	50.38	0.0082	75.33
KDD CUP	ALL	0.4794	0.4829	0.4822	0.4842	0.4853	0.4827	0.4903	0.0109	100.25	0.0074	71.26	0.0081	80.16	0.0061	75.82	0.0050	56.63	0.0076	67.79
	Tail	0.3689	0.3726	0.3719	0.3739	0.3754	0.3724	0.3821	0.0132	107.94	0.0095	91.30	0.0102	96.59	0.0082	76.54	0.0067	61.09	0.0097	81.33

Table 6: AUC and RIG performance of BERT4CTR compared with representative models in use on two data sets

and non-textual features during fine-tuning phase can improve the accuracy of prediction significantly, *e.g.*, BERT4CTR brings more than 0.5% AUC gain on both Bing Ads data and KDD CUP 2012 data, compared with all cascading methods (Wide & Deep, DeepFM and NN+GBDT). Second, although increasing the depth of network in non-textual part improves the accuracy of CTR prediction, deep uni-attentions between textual features and non-textual features still bring considerable improvement for CTR prediction, *e.g.*, BERT4CTR can bring 0.4% AUC gain on both Bing Ads data and KDD CUP 2012 data, compared with Shallow Interaction-N Layers model, showing the pure benefits brought by the uni-attention architecture. Finally, among all seven alternative models in Table 6, BERT4CTR shows the highest AUC on both data sets, which gives evidence that the design presented in Section 3 is an effective way to learn the cross-information between multi-modal inputs for CTR prediction.

The time-costs of training and inference for the alternative models are shown in Table 7. We observe, first, that BERT4CTR does not significantly increase the training time compared with Shallow Interaction. In detail, the training time of BERT4CTR only increases by 7% compared with Shallow Interaction-N Layers and by 14% compared with Shallow Interaction-1 Layer that have been widely used in industry [30][2]. For example, in Microsoft Bing Ads, Shallow Interaction-1 Layer framework has been used to refresh a RoBERTa-24 model. The training takes 5 days in one cycle. According to Table 7, BERT4CTR will take 5.8 days on the same settings, that is still less than the weekly re-calibration deadline.

The inference delay of BERT4CTR is close to cascading, and Shallow Interaction framework, and much less than NumBERT, *e.g.*, BERT4CTR can reduce inference delay by 52% (resp., 29%) on Bing Ads data (resp., KDD CUP 2012 data), compared with NumBERT.

(a) Training Cost

Framework	Model	Average		Median		90th Percentile		95th Percentile	
		Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP
Shallow Interaction Framework (Two-steps)	Shallow Interaction - 1 Layer	19.72	7.69	19.66	7.67	19.76	7.72	19.87	7.75
	Shallow Interaction - N Layers	20.54	8.24	20.42	8.22	20.66	8.26	20.75	8.33
NumBERT Framework	NumBERT	60.50	12.36	60.11	12.29	60.61	12.45	60.89	12.58
BERT4CTR Framework (Two-steps)	BERT4CTR	22.06	8.69	21.88	8.64	22.32	8.77	22.45	8.85

(b) Inference Cost

Framework	Model	Average		Median		90th Percentile		95th Percentile	
		Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP	Bing Ads	KDD CUP
Cascading Framework	Wide & Deep	6.09	2.79	6.06	2.76	6.19	2.88	6.30	2.98
	DeepFM	6.18	2.81	6.14	2.78	6.27	2.91	6.39	3.00
	NN+GBDT	6.14	2.81	6.12	2.77	6.23	2.90	6.35	3.01
Shallow Interaction Framework	Shallow Interaction - 1 Layer	6.05	2.82	6.01	2.78	6.13	2.87	6.29	2.96
	Shallow Interaction - N Layers	6.57	3.08	6.52	3.04	6.66	3.13	6.75	3.22
NumBERT Framework	NumBERT	14.73	4.72	14.70	4.68	14.83	4.81	14.88	4.92
BERT4CTR Framework	BERT4CTR	7.05	3.36	7.03	3.32	7.19	3.42	7.31	3.51

Table 7: Time-cost performance (ms/sample) of BERT4CTR compared with representative models in use on two data sets

The results from Table 6 and Table 7 give strong evidences that BERT4CTR can achieve both high accuracy and low training and inference delay for CTR prediction.

4.4 Online Performance

The BERT4CTR model with 6-Layers has already been passed the online A/B testing over 1% of the whole traffic of Microsoft Bing Ads recommender system, where the control flight is based on BERT-6 with Shallow Interaction. During the online A/B testing in a one-month period, the treatment flight brought a 1.4% gain on click yields, compared to the control flight, with only 0.5% increase on display yields. It is noteworthy that 1% increase on click yields and display yields can be seen as a significant change in practice.

5 DISCUSSION

Although, we used in this paper the CTR prediction with numerical features as our main application scenario, the framework of BERT4CTR is applicable to other settings mixing textual and non-textual features. For example, one can extract the representative embedding from images through VGGNet [25], ResNet [9] *etc.* to replace the token E_E in Figure 4 and calculate the uni-attentions.

Besides, Knowledge-Distillation approach [10], where a light model handling textual and non-textual inputs, with uni-attention and dimensionality reduction, is learned under the supervision of predicted scores from a well-trained BERT4CTR model with deep layers, can be adapted here.

6 CONCLUSION

In this paper, we designed an efficient framework for CTR prediction that combines pre-trained language model with non-textual features. We started from NumBERT, the classical technique to integrate textual and numerical features in pre-trained language model, and introduced three gradual improvements, uni-attention, dimensionality reduction and two-steps joint-training, resulting into a novel framework, BERT4CTR. Comprehensive experiments on both commercial data and public data showed that BERT4CTR can achieve significant prediction accuracy gains while keeping low training and inference time-costs. It therefore provides a promising solution for CTR prediction, and more largely for any applications needing multi-modal input features in real world.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[3] 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 *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

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

[5] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. 2019. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2509–2517.

[6] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[8] Weiwei Guo, Xiaowei Liu, Sida Wang, Huiji Gao, Ananth Sankar, Zimeng Yang, Qi Guo, Liang Zhang, Bo Long, Bee-Chung Chen, et al. 2020. Detext: A deep text ranking framework with bert. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2509–2516.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[11] Bernard J Jansen and Tracy Mullen. 2008. Sponsored search: an overview of the concept, history, and technology. *International Journal of Electronic Business* 6, 2 (2008), 114–131.

[12] Yunjiang Jiang, Yue Shang, Ziyang Liu, Hongwei Shen, Yun Xiao, Wei Xiong, Sulong Xu, et al. 2020. BERT2DNN: BERT Distillation with Massive Unlabeled

Data for Online E-Commerce Search. *arXiv preprint arXiv:2010.10442* (2020).

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[14] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 689–698.

[15] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504* (2019).

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[17] Edward H Livingston. 2004. Who was student and why do we care so much about his t-test? 1. *Journal of Surgical Research* 118, 1 (2004), 58–65.

[18] Wenhao Lu, Jian Jiao, and Ruofei Zhang. 2020. TwinBERT: Distilling Knowledge to Twin-Structured Compressed BERT Models for Large-Scale Retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2645–2652.

[19] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, 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.

[20] Andrew W Moore. 2001. Information gain. *School of Computer Science, Carnegie Mellon University, http://www. cs. cmu. edu/˜ awm/tutorials* (2001).

[21] Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*. 1–7.

[22] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 695–704.

[23] 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.

[24] Lalita Sharma and Anju Gera. 2013. A survey of recommendation system: Research challenges. *International Journal of Engineering Trends and Technology (IJETT)* 4, 5 (2013), 1989–1992.

[25] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[26] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2019. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530* (2019).

[27] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7464–7473.

[28] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[30] Dong Wang, Shaoguang Yan, Yunqing Xia, Kavé Salamatian, Weiwei Deng, and Qi Zhang. 2022. Learning Supplementary NLP Features for CTR Prediction in Sponsored Search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4010–4020.

[31] Fang Wang, Warawut Suphamitmongkol, and Bo Wang. 2013. Advertisement click-through rate prediction using multiple criteria linear programming regression model. *Procedia Computer Science* 17 (2013), 803–811.

[32] Zhe Wang, Rundong Shi, Shijie Li, and Peng Yan. 2020. GBDT and BERT: a Hybrid Solution for Recognizing Citation Intent. *Studies* 55 (2020), 12c2a39230188.

[33] Puxuan Yu, Hongliang Fei, and Ping Li. 2021. Cross-lingual Language Model Pretraining for Retrieval. In *Proceedings of the Web Conference 2021*. 1029–1039.

[34] Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020. Do language embeddings capture scales? *arXiv preprint arXiv:2010.05345* (2020).

A APPENDIX

For the purpose of reproducibility, we provide the details of the experiments on the public KDD CUP 2012 data set, including the data description, the settings on textual and non-textual part, and the pseudocode of uni-attention.

A.1 Data Details

The data set contains 235 million search ads impressions sampled from session logs of Tencent search engine Soso.com. Each sample in this data set contains five components:

- Query text: which is a list of anonymous tokens hashed from natural language;
- Ad text: which includes ad title and ad display URL and are also lists of anonymous tokens;
- CTR prediction features: there are 56 CTR prediction features including sparse features such as UserID, AdID, user's gender *etc.*, and dense features such as historical CTRs, number of impressions per Ad/Query/User *etc.*;
- Position feature: a special feature which indicates the impressed position of this ad;
- Click label: where 1 means this ad has been clicked and 0 is not clicked.

The 56 non-textual CTR features used in BERT4CTR are generated as below:

- ID features: each raw ID attribute (such as AdID, QueryID, UserID, Gender, Age *etc.*) appearing in training data is mapped to an ordinal number. To consider the situation where new IDs not appearing in the training data, appear later, we used robust training approach, *i.e.*, we randomly removed 5% of IDs in training data and set them as "Missing", which is mapped to a special number. We also set, during inference, the new IDs in validation data, as "Missing" to make the model work well;
- Historical features: we calculated the historical CTRs and number of impressions from different perspectives, including the Ad-level, User-level, Query-level, Gender-level and Age-level *etc.*;
- Length features: these length features include Query Length, AdTitle Length, AdDesription Length and Keyword Length *etc.*;
- Semantic features: we calculated TF × IDF value for each token (such as Query tokens, AdTitle tokens, AdDescription tokens *etc.*) provided in training data, as a type of semantic feature.

As there is no time information in this data set, it is impossible to use the impression time to split the data into the training and validation data. Alternatively, we randomly selected 1/11 of samples as validation data and the remaining as training data. Table 8 summaries the statistics of training data and validation data.

	Impressions	Clicks	CTR
Training Data	216,038,149	7,550,609	0.0349
Validation Data	19,544,730	667,024	0.0341

Table 8: Statistics for KDD CUP 2012 data set

A.2 Settings on Textual Part

We choose BERT-12 model as initial model, where query and ad title concatenated with ad display URL are used as two input sentences. For the purpose of privacy protection, each textual token in KDD CUP 2012 data set is anonymized into one hash ID and therefore, it is difficult to map the hash ID from data to vocabulary ID of pre-trained language models. To solve this issue, we treat these anonymous tokens as new words, where we calculate the TF × IDF for each one of the anonymous token and choose the top 300 thousand of them to build a new vocabulary. The masking rate is 15% for MLM task and Adam optimizer with learning rate of 1×10^{-4} is used in both pre-training and fine-tuning phase. After two epochs of pre-training and four epochs of fine-tuning with this new vocabulary, the AUC of BERT model can achieve 75.91% on ALL slice. This shows the effectiveness of our approach.

A.3 Settings on Non-textual Part

As described in Section 3, the processing methods for sparse features and dense ones are different. For sparse features, we extend each input to a 32-dimensional embedding. While for dense features, we normalize them with max-min normalization at first and then expand each normalized value into 101-dimensional one-hot vectors based on 0.01 buckets. Afterward, the 32-dimensional embedding is generated by looking up the embedding table. Totally a 1792-dimensional embedding is generated by concatenating all of 56 32-dimensional sub-embeddings (excluding the embedding from position feature) and then a 512-dimensional hidden layer is followed as the final embedding layer to calculate uni-attention.

For uni-attention, we set the dimension of hidden layer in Equation 3, denoted as d_a, as 64. The resulting network to calculate the attention alignment score between Query and Key in uni-attention (based on BERT-12) is depicted in Figure 6.

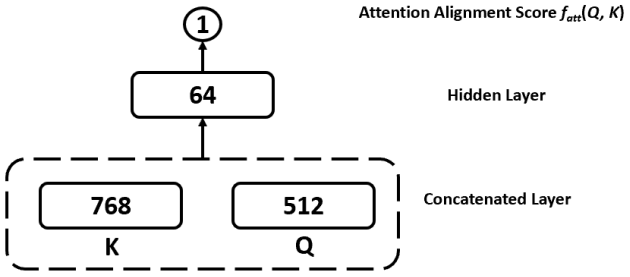

Figure 6: The network to calculate the attention alignment score between Query and Key in uni-attention

A.4 Implementation of Uni-Attention

We also provide the details on the implementation of uni-attention, which is the core of BERT4CTR. In one layer of BERT4CTR, the output from the non-textual token after dimensionality reduction can be denoted as $X \in \mathbb{R}^{d_X \times 1}$, where d_X is the dimension of output for this non-textual token (512 in our experiment). Besides, the outputs from textual tokens can be denoted as $Y \in \mathbb{R}^{d_Y \times l_Y}$, where

d_Y is the dimension of output for each textual token (768 in our experiment) and l_Y is the maximum sequence length of textual input (64 in our experiment). We also use the same Attention-Mask mechanism as the one used in textual part, for the calculation of uni-attention, where $Mask \in [0,1]^{l_Y}$.

The pseudocode for uni-attention based on additive attention is shown in Algorithm 1, in which W_q, b_q, W_k, b_k, W_v, b_v, W_a, b_a, W_i and b_i are trainable parameters.

Algorithm 1 $U \leftarrow UniAttention(X, Y, Mask)$

Input: $X \in \mathbb{R}^{d_X \times 1}$, $Y \in \mathbb{R}^{d_Y \times l_Y}$, $Mask \in [0,1]^{l_Y}$
Output: $U \in \mathbb{R}^{d_X \times 1}$

function UniAttention(X, Y, $Mask$)
  $Q \leftarrow W_q X + b_q \mathbb{1}^T$ ▷ $W_q \in \mathbb{R}^{d_X \times d_X}$, $b_q \in \mathbb{R}^{d_X}$
  $K \leftarrow W_k Y + b_k \mathbb{1}^T$ ▷ $W_k \in \mathbb{R}^{d_Y \times d_Y}$, $b_k \in \mathbb{R}^{d_Y}$
  $V \leftarrow W_v Y + b_v \mathbb{1}^T$ ▷ $W_v \in \mathbb{R}^{d_X \times d_Y}$, $b_v \in \mathbb{R}^{d_X}$
  $Repeat(Q, l_Y)$ ▷ $Q \in \mathbb{R}^{d_X \times l_Y}$
  $M \leftarrow Concat(Q, K)$ ▷ $M \in \mathbb{R}^{(d_X + d_Y) \times l_Y}$
  $H \leftarrow Tanh(W_a M + b_a \mathbb{1}^T)$ ▷ $H \in \mathbb{R}^{d_a \times l_Y}$
  $S \leftarrow W_i H + b_i \mathbb{1}^T$ ▷ $S \in \mathbb{R}^{1 \times l_Y}$
  $\forall S_i \in S, if\ \neg Mask[i],\ then\ S_i \leftarrow -\infty$
  $U \leftarrow V\ Softmax(S^T)$ ▷ $U \in \mathbb{R}^{d_X \times 1}$
  return U
end function