Learning Supplementary NLP Features for CTR Prediction in Sponsored Search

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

Shaoguang Yan
STCA, Microsoft Corporation
Beijing, China
shaoyan@microsoft.com

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

Kavé Salamatian
University of Savoie, France
& Tallinn University of Technology, Estonia
Kave.salamatian@univ-smb.fr

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

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

ABSTRACT

In sponsored search engines, pre-trained language models have shown promising performance improvements on Click-Through-Rate (CTR) prediction. A widely used approach for utilizing pre-trained language models in CTR prediction consists of fine-tuning the language models with click labels and early stopping on peak value of the obtained Area Under the ROC Curve (AUC). Thereafter the output of these fine-tuned models, *i.e.*, the final score or intermediate embedding generated by language model, is used as a new Natural Language Processing (NLP) feature into CTR prediction baseline. This cascade approach avoids complicating the CTR prediction baseline, while keeping flexibility and agility. However, we show in this work that calibrating separately the language model based on the peak single model AUC does not always yield NLP features that give the best performance in CTR prediction model ultimately. Our analysis reveals that the misalignment is due to overlap and redundancy between the new NLP features and the existing features in CTR prediction baseline. In other words, the NLP features can improve CTR prediction better if such overlap can be reduced.

For this purpose, we introduce a simple and general joint-training framework for fine-tuning of language models, combined with the already existing features in CTR prediction baseline, to extract supplementary knowledge for NLP feature. Moreover, we develop an efficient Supplementary Knowledge Distillation (SuKD) that transfers the supplementary knowledge learned by a heavy language model to a light and serviceable model. Comprehensive experiments on both public data and commercial data presented in this work demonstrate that the new NLP features resulting from the joint-training framework can outperform significantly the ones from the independent fine-tuning based on click labels. we also show that the light model distilled with SuKD can provide obvious AUC improvement in CTR prediction over the traditional feature-based knowledge distillation.

KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539064

CCS CONCEPTS

• **Information systems** → **Sponsored search advertising**; **Language models**.

KEYWORDS

Pre-trained Language Model; CTR Prediction; Sponsored Search Engines; Supplementary Knowledge; Joint Training; Knowledge Distillation

ACM Reference Format:
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 (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539064

1 INTRODUCTION

Pre-trained language models, such as BERT [4] and RoBERTa [23], have been successfully used to improve the Click-Through-Rate (CTR) prediction in Sponsored Search Engines[24][28][15]. The traditional way used in industry for utilizing pre-trained language models into the CTR prediction pipeline is to use the output of these models, *e.g.*, the final score or the intermediate embedding, as a new NLP feature fed into the CTR prediction model. In this case, the pre-trained models are generally fine-tuned by augmenting official models with click labels and early stopped based on peak value of obtained Area Under the ROC Curve (AUC) in fine-tuning. The final performance of a new NLP feature can be evaluated through the additional AUC or Relative Information Gain (RIG) of the new CTR prediction model with this feature over the baseline without it (referred to as *end-to-end gain*) [25][5][2][21][16]. This cascade approach avoids complicating the CTR prediction model with the addition of new signals and ensures learning flexibility and iteration agility. Nonetheless, we show in this work that this approach can not always yield NLP feature with the best performance for CTR prediction. By accounting the correlation between the NLP feature and the existing features used in CTR prediction baseline (the one without this NLP feature), we observe this misalignment coming

from information overlap and redundancy between the fine-tuning of language model and the training of CTR prediction model with existing features, which also inspires us to improve the NLP features in CTR prediction by reducing such overlap.

To address the misalignment described above, we introduce a simple and general joint-training framework for the fine-tuning of pre-trained language models, combined with the existing features from CTR prediction baseline (referred to as *auxiliary features*). In this framework, a node-wise summed input for interaction layers is applied to make the language model learn the additional and less redundant information beyond these auxiliary features on CTR prediction task (thereafter referred to as *supplementary knowledge*), resulting into NLP features that can improve the performance of final CTR prediction model significantly.

To make the supplementary NLP feature more serviceable, we also design a *Supplementary Knowledge Distillation* (*SuKD*) component, which integrates our joint-training framework into the traditional feature-based distillation [12][30], to transfer the supplementary knowledge learned by a heavy language model (acting as teacher) to a light model (acting as student) efficiently. In SuKD, the student model can not only learn the supplementary knowledge supervised by the teacher model, but also keep the joint-training framework to learn the supplementary knowledge by itself simultaneously.

We summarize the contributions of this paper below:

- We show that fine-tuning separately the pre-trained language model with the peak single model AUC does not yield features that provide ultimately the best CTR prediction performances. Our analysis reveals that this is largely due to the contribution overlap between the new NLP features and the existing features used for CTR prediction baseline.
- We introduce, up to our knowledge for the first time, the supplementary knowledge in the building of NLP features for CTR prediction in sponsored search engines. A joint-training framework, which combines the fine-tuning of language models with the existing features from CTR prediction baseline, is presented to assist the language models with supplementary knowledge learning. Compared with separately fine-tuning task supervised with click labels, this framework considers avoiding information overlap between existing features and the new NLP features, which is beneficial to improve the performance of new NLP features in CTR prediction model. Furthermore, we develop an efficient framework for knowledge distillation SuKD to transfer the supplementary knowledge from a heavy teacher model to a light student model.
- Comprehensive experiments based on both commercial and public data show that the joint-training framework can bring significant improvement on end-to-end gain compared with independent fine-tuning with click labels. Moreover, we show that the gain obtained by our approach, while maintaining lower complexity and high flexibility, is close to the gain that could be obtained by integrating the network of language model into CTR prediction baseline directly. On both of the two data sets, we show that using the SuKD, light BERT model with only 3 distilled transformer layers can generate NLP features with almost twice end-to-end gain compared with traditional feature-based distillation. Besides, we show that SuKD can even improve the performance of student models with networks different from the teacher.

We describe below the organization of this paper. In Section 2, we present the related work. In Section 3, we give the details of the joint-training framework and the SuKD. In Section 4, we present our evaluation results on both commercial and public data to show the advantages of our approach and, we develop some conclusions in the last section.

2 RELATED WORK

In this section, we present a summary of related work on CTR prediction models, pre-trained language models, knowledge distillation frameworks and joint-training.

2.1 CTR Prediction Model in Sponsored Search Engines

CTR prediction models used for the choosing of advertisements to present in sponsored search engines should ensure low-latency in addition to high-accuracy. For example, the CTR prediction model in Baidu.com uses a deep neural network, called *Phoenix Nest*, with a handcrafted set of features coming from the user, query, and advertisement properties [5]. Google Ads has developed a "Follow The Regularized Leader" (FTRL) model to predict CTR and is presented in [25]. Google play uses a Wide & Deep model described in [2]. Inspired by the Wide & Deep model, [7] introduces DeepFM model to emphasize both low- and high-order feature interactions. Microsoft Bing.com adopts a Neural Network boosted with GBDT ensemble model [21] for ads CTR prediction. This is the industrial scenario we are using through this paper. The features of these CTR prediction models can be grouped into two categories: one is the raw text/ID from user, query and ad, and the other is the output generated from sub-models, such as LR model [21][25], pre-trained language model [24][34], *etc.*

2.2 Pre-trained Language Model and Knowledge Distillation

Recent work has shown the abilities of pre-trained language models to extract deep relationship in a sentence pair [4][23][19][32][22], that are useful for augmenting the semantic features of query and recommendation pair in sponsored search engines [24][34][15][8][35]. Generally, one trains the pre-trained language models against the real click data, targeting directly the prediction of click/non-click labels. Thereafter, one can use the score from the final layer [34][15], or the embedding from the intermediate layer [24][8], of these language models 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 [24] while Meituan.com and JD.com use only the output score of BERT [34][15].

Although pre-trained language models can provide strong semantic insights, their deployment in an online environment is complex and costly. [12][1] propose Knowledge Distillation approaches (KD)

to help in alleviating this issue. These approaches can be categorized into two groups: response-based distillation where a student model tries to mimic the final output score of the teacher model [12][1][37], and feature-based distillation which supervises the student network training with both the output score and the outputs of intermediate layers in the teacher model [30][11][18][14][27].

In this work, we extract more efficient NLP features in terms of final advertisement CTR prediction. This is achieved by introducing supplementary knowledge in fine-tuning of pre-trained language model to reduce redundancy and overlap with existing features in baseline. We also develop a KD method to reduce the complexity of the heavy language models for use in online sponsored search engines.

2.3 Joint-Training

Joint-training consists of training together multiple networks with different learning objectives in order to benefit from the synergy of the learning [31][6]. Joint-training has been widely used for online advertisement recommendation. In [2] a wide linear model and a deep neural network are trained together to both memorize and generalize a recommendation systems. [20] designs a jointly-trained network during the cold start recommendation phase. In our work, we have applied the idea from joint-training by using a node-wise summed input for the interaction layers to make pre-trained language model and the auxiliary features learn the supplementary knowledge from each other efficiently.

3 SUPPLEMENTARY LEARNING THROUGH JOINT-TRAINING

CTR prediction for advertisements is a core component of sponsored search systems. For this purpose, we need to adopt simple and low-latency models that are usable in online platforms, while still achieving high CTR prediction accuracy. In CTR prediction models, both numerical features (which can characterize the user historical behaviors) and semantic features (which can represent the relationship between query and impressed ads) are used as inputs. BERT, RoBERTa and other pre-trained language models that can extract deep semantic relationship between sentence pair, are to be considered as efficient tools to generate relevant NLP features from query and ad text that can be exploited in CTR prediction.

We can make a theoretical analysis for the impact of adding NLP features to a CTR prediction baseline with auxiliary features, where the end-to-end RIG [10] is used as the evaluation metric for presentation convenience. Supposing a CTR prediction system using $n-1$ features $X^{n-1} = \{x_1, x_2, ..., x_{n-1}\}$ (the auxiliary features used in CTR prediction baseline in our case), and augmented CTR prediction system with an additional feature x_n (the new NLP feature in our case), the end-to-end RIG gain of feature x_n can be calculated as :

$$\Delta RIG = \left(1 - \frac{L^n_{Predict}}{L_{emp}}\right) - \left(1 - \frac{L^{n-1}_{Predict}}{L_{emp}}\right) = \frac{L^{n-1}_{Predict} - L^n_{Predict}}{L_{emp}} \tag{1}$$

where $L^{n-1}_{Predict}$ is the average log-loss of CTR prediction baseline model with a feature set $X^{n-1} = \{x_1, x_2, \ldots, x_{n-1}\}$, while $L^n_{Predict}$ is the average log-loss of augmented CTR prediction model with additional feature x_n. L_{Emp} is the average log-loss of a naive model that predicts CTR with the average empirical CTR, which can be considered as a constant in our study. Therefore, to improve ΔRIG, $L^{n-1}_{Predict} - L^n_{Predict}$ should be maximized. Assuming that we have N samples and $P^{n-1}_i = P(click = 1|X^{n-1})$ and $P^n_i = P(click = 1|X^{n-1}, x_n)$ are respectively the predicted CTR for sample i using $n-1$ features without x_n and using n features including x_n, and y_i is the click label of sample i, we can rewrite the log-loss difference as:

$$\begin{aligned} & L^{n-1}_{Predict} - L^n_{Predict} \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[y_i \log P^n_i + (1-y_i)\log(1-P^n_i) - y_i \log P^{n-1}_i - (1-y_i)\log(1-P^{n-1}_i)\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \frac{P^n_i}{P_i{}^{n-1}} + (1-y_i)\log\frac{1-P^n_i}{1-P^{n-1}_i}\right] \end{aligned} \tag{2}$$

By applying the Bayes formula, we have finally :

$$\begin{aligned} & L^{n-1}_{Predict} - L^n_{Predict} \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \frac{P(x_n|X^{n-1}, click=1)}{P(x_n|X^{n-1})} + (1-y_i)\log\frac{P(x_n|X^{n-1}, click=0)}{P(x_n|X^{n-1})}\right] \end{aligned} \tag{3}$$

While the log-loss function, L_{ft}, of fine-tuning of the pre-trained language model with click labels, is given by:

$$L_{ft} = H_{click} - \frac{1}{N}\sum_{i=1}^{N}\left[y_i \log \frac{P(x_n|click=1)}{P(x_n)} + (1-y_i)\log\frac{P(x_n|click=0)}{P(x_n)}\right] \tag{4}$$

where H_{click} is the entropy of click variable:

$$H_{click} = -P(click=1)\log P(click=1) - P(click=0)\log P(click=0) \tag{5}$$

and is not dependent on the feature x_n. Under the condition that x_n and X^{n-1} are independent, it is clear that minimizing L_{ft} can maximize the value $L^{n-1}_{Predict} - L^n_{Predict}$ and ultimately ΔRIG. However, when x_n and X^{n-1} are dependent, this is not anymore the case. In the extreme case that x_n and X^{n-1} are directly related, *i.e.*, $x_n = f(X^{n-1})$, we have $P(x_n|X^{n-1}) = P(x_n|X^{n-1}, click) = 1$, yielding $\Delta RIG = 0$. So this is important to ensure that x_n and X^{n-1} are not overlapping, *i.e.*, $\frac{P(x_n|X^{n-1},click)}{P(x_n|X^{n-1})}$ is as large as possible, rather than only trying to maximize $\frac{P(x_n|click)}{P(x_n)}$.

We will illustrate the impact of misalignment in Section 4.2.1 where we show that the NLP feature generated by the fine-tuned model with peak single model AUC does not give the best ultimate performance on CTR prediction. This motivates us to consider how to reduce such overlap, and to design a joint-training framework to extract additional and less redundant semantic knowledge to existing features in the baseline.

We show in Figure 1 the proposed joint-training architecture. There are two components: the left one is only using auxiliary features, *i.e.*, features already used by the CTR prediction baseline. Among these features, the position of showing an ad is a special one that is independent of the others. Following [21], we separate this position feature from the other auxiliary inputs, by connecting it to a separate hidden node H_p, while the others are connected to another set of nodes, making a low dimensional hidden layer H_a that is called *auxiliary layer*. This ensures that the position feature is not interacting with others through the model learning, as the position an ad is displayed and its quality should be independent factors of the final CTR.

The right part of architecture deals with NLP features. It gets as input the pair <query, ad> and the main part is a standard language

model, *e.g.*, coming from BERT or RoBERTa, with a [CLS] pooling layer to provide the embedding. This pooling layer is mapped into a low-dimensional hidden layer H_c (called *supplementary layer*) that has the same dimension as the auxiliary layer H_a. The two hidden layers, H_a and H_c are node-wise summed and the outcome is sent to a sequence of interaction layers H_s. The node-wise summation provides the fusion mechanism of auxiliary features with NLP ones, *i.e.*, each node in the input of H_s is the summation of the general abstraction of auxiliary features through the layer H_a and the deep embedding coming from the language model. We will evaluate this architecture in Section 4.2.2 and show that it can improve the end-to-end gain, compared with the simple concatenation of the auxiliary layer H_a and the supplementary layer H_c.

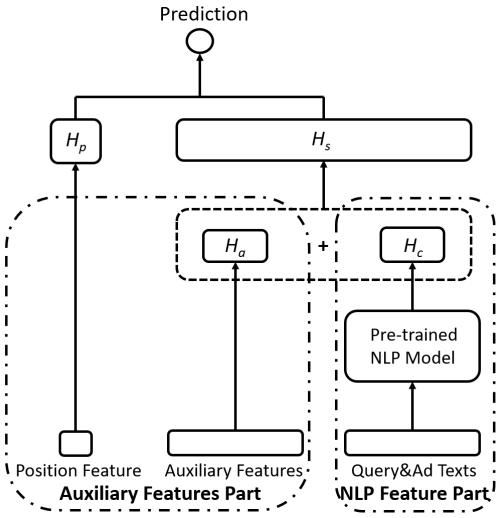


Figure 1: Architecture of joint-training with NLP information fusion

It is noteworthy that the set of layers in H_s (see Figure 1) can follow different type of architectures, such as Deep Neural Network [36], FM[29], ResNet[9] *etc.* depending on the application need. In our experiments, we use a single hidden layer with Sigmoid activation for our commercial data (coming from Bing), while for public dataset two hidden layers with ReLU activation are used.

The pre-trained language models are generally complex and have a large depth, *e.g.*, RoBERTa-Large is a 24-layers transformer model, while the neural network of auxiliary features is typically shallow. The training of such a wide and deep framework is difficult, when the weights in the network are initialized to random values [13]. To solve this issue, we split the learning into two steps. In the first initial training step, we fine-tune the pre-trained language model alone, and train the remaining network with only auxiliary features respectively. The targets for both two learnings are directly the click labels. In the second step, we use the calibrated weights in the first step, as initial values for the two parts of the network in Figure 1. We then continue the learning combining both auxiliary and NLP features and update all weights simultaneously through a set of iterations. By doing this, in the second step we implement a joint-training. We show the performance of these two steps joint-training in Section 4.2.2. Theoretically, the loss function of our joint-training framework can be derived as:

$$L_{jt} = H_{click} - \frac{1}{N}\sum_{i=1}^{N}\Big[y_i \log \frac{P(x_n|X^{n-1}, click=1)P(X^{n-1}|click=1)}{P(x_n|X^{n-1})P(X^{n-1})} + (1-y_i)\log \frac{P(x_n|X^{n-1}, click=0)P(X^{n-1}|click=0)}{P(x_n|X^{n-1})P(X^{n-1})}\Big] \tag{6}$$

As the joint-training focuses on the learning of NLP feature x_n, $P(X^{n-1}|click)$ and $P(X^{n-1})$ can be considered as constants in Equation 6. Therefore, minimizing L_{jt} involves maximizing $\frac{P(x_n|X^{n-1},click)}{P(x_n|X^{n-1})}$ as in Equation 3 and ultimately maximizing ΔRIG.

Another issue is relative to the complexity of heavy language model that makes it too costly to use for online applications. One solution proposed in the literature to deal with this is the distillation approach to transfer the knowledge learned by a heavy teacher model to a light student model, which could be run online and matched the delay constraints. This inspires us to propose the Supplementary Knowledge Distillation (SuKD) architecture shown in Figure 2.

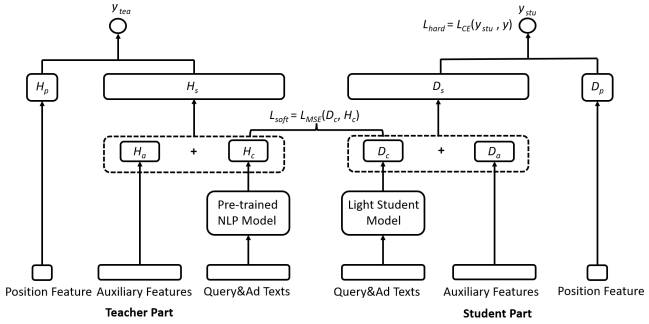


Figure 2: Framework of Supplementary Knowledge Distillation

On the left side of the figure is the teacher model, which is the fully joint-trained model integrating auxiliary and NLP features described earlier. This model provides supervision information, in the form of embedding coming from layers H_c, to the student model on the right side, that can be a model mimicking the teacher model structure but with less depth or lower dimension of hidden layers (as shown in Figure 2) or any other type of light network. The light student can learn the supplementary knowledge from the teacher with a distillation loss function, in our case the Mean Square Error (MSE) between the embedding resulting from H_c in the teacher model and the embedding resulting from D_c in the student model. Inspired by [12], we also use the ensemble of soft and hard losses in distillation, *i.e.*, besides considering the outputs from H_c of the teacher model that results in soft MSE loss, we also consider the hard loss that is evaluated from the click prediction at the last stage of the joint-training framework described above. The final loss of distillation is evaluated as a weighted sum of hard and soft losses. We use the multitask weighting auto-tuning method introduced in [17], that can learn to balance these weights optimally during training, to select the weights (w_1, w_2) used for the final loss. This yields the following loss function for SuKD:

$$L_{distill} = e^{-w_1} L_{MSE}(D_c, H_c) + \log w_1 + e^{-w_2} L_{CE}(y_{stu}, y) + \log w_2 \tag{7}$$

where $L_{MSE}(D_c, H_c)$ is the MSE loss of approximating the embedding from teacher H_c with the one from student D_c, and $L_{CE}(y_{stu}, y)$

is the cross entropy loss between the final CTR prediction y_{stu} of the student model integrating both NLP and auxiliary features and the true click labels y of training sample. The weights w_1 and w_2 are trainable during distillation.

In online deployment, only the NLP feature part in student side is served to generate the embedding output from D_c, which represents the supplementary semantic knowledge learned by student and is fed into downstream CTR prediction model as new feature.

4 EXPERIMENTS

In this section, we evaluate the joint-training framework and SuKD architecture described in the previous section on two distinct data sets: one commercial, and one public, each with different CTR prediction baselines. This will give evidences that our proposed joint-training framework is general and can be used for augmenting various CTR prediction models with supplementary NLP features.

4.1 Experimental Settings

4.1.1 Bing Ads Data. Microsoft Bing Ads is a commercial environment that implements ad sponsored search. We have used this platform to gather a data set consisting of 230 million <query, ad> pairs with click labels, which are randomly sampled from Bing Ads logs obtained in Feb. 2019. Among these samples, we use the ones in the first three weeks as training set and the ones in the last week as validation set for CTR prediction. Each sample contains five types of information:

- **Query text**: A short text entered by the users in Bing.com search box.
- **Ad text**: The text of advertisement presented to the user along with the query, including ad title and ad display URL.
- **CTR prediction features**: A set of 290 numerical features which have been used in the Bing Ads CTR prediction baseline. These features are grouped into 4 categories: (1) query-related features such as length of the query, historical value of CTR of the query, *etc.*; (2) ad-related features such as historical value of CTR of the ad, history of displaying of the ad *etc.*; (3) user-related features which describe the user's current and historical behaviors, *e.g.*, the last time an ad was clicked, the total number of ads clicked in the last month, *etc.*; (4) cross-features, *e.g.*, the matched pair *(Query, AdTitle)* indicating the similarity between a query and an ad title [21].
- **Position feature**: As suggested by [21], the displayed position of the ad is a special feature that is independent of other features, with the consideration that the displayed position and the ad quality can affect the likelihood of a click separately.
- **Click label**: an indicator of the ad being clicked or not.

The CTR prediction model currently used in Bing Ads is described in [21] with the above described features. This is a Neural Network (NN) boosted with a Gradient Boosting Decision Tree (GBDT) ensemble model. The GBDT is initialized with a prediction score coming from a fully trained NN. In other terms, the GBDT tries to predict the residual error between the ground truth and a NN's output. This NN boosted with GBDT model is also used as our baseline model on Bing Ads data.

We build the joint-training architecture following Figure 1. In H_s, we use a single 30-dimensional layer with Sigmoid activation function. Adding to the separated node for position feature H_p, the total dimension of hidden layer in this network is 31. Both layers H_c and H_a are also set to be 30-dimensional, *i.e.*, we generate a 30-dimensional vector to represent NLP feature used for CTR prediction.

The RoBERTa-Large model with 24 layers (abbreviated as RoBERTa-24) released by Facebook [23] is used as the pre-trained language model in teacher training. Query and ad title concatenated with ad display URL are given as inputs to language model. We fine-tune the RoBERTa-24 model using the approach described in [4], *i.e.,* we first pre-train the standard RoBERTa-24 model on our training data, using the Mask Language Model (MLM) task without click labels, and then continue fine-tuning the model on the same data, using the Classification (CLS) task with click labels. For this training, the batch size is set to 256 and the RoBERTa-24 model is fine-tuned for 4 epochs with learning rate 2e-5. In parallel, the 290 features described above are introduced as auxiliary features from baseline, yielding a neural network NN_{init} with 291-dimensional input (*i.e.*, 290 auxiliary features and one position feature) and a 31-dimensional hidden layer (30 hidden nodes plus the one for position feature), which is fully trained using the click labels with 20 iterations. For the joint-training phase, we initialize the weights in the auxiliary features component with the weights trained in NN_{init} and the weights of the NLP feature part are initialized with the weights coming from the fine-tuned RoBERTa-24 model. We continue training all weights in the joint-training framework with two epochs and a small learning rate of 5e-6. Finally, the 30-dimensional embedding generated from H_c which follows the RoBERTa-24 [CLS] pooling layer is used as the new NLP feature for CTR prediction model. We also calibrate over light student models using the SuKD architecture, based on this RoBERTa-24 teacher model.

4.1.2 KDD Cup 2012 CTR prediction dataset for search ads. [1] In addition to Bing Ads data, we also evaluate our approach on a public dataset, the KDD Cup 2012 data [26][33]. This dataset contains 235 million pairs of query and ad sampled from the logs at Tencent search engine Soso.com. Each sample in this dataset is likewise the Bing Ads data, *i.e.*, it contains five components: query text, ad text, CTR prediction features, position feature and click labels.

For this dataset, we use BERT-Base with 12 layers (abbreviated as BERT-12) [4] as the pre-trained language model that gets as input the pair of query and ad title concatenated with ad display URL. In order to show the generality and effectiveness of the proposed method, we follow the work in [26] and choose two representative models as the CTR prediction baselines: Wide & Deep, and DeepFM.

- **Wide & Deep** is a popular CTR prediction model introduced by Google [2] which combines a shallow linear model with a deep neural network.
- **DeepFM** is an improved version based on Wide & Deep, which replaces the linear model in the wide component with a Factorization-Machine (FM) [7].

Similarly to what described on the Bing Ads dataset, we pre-train the standard BERT-12 model on KDD Cup 2012 dataset first using

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

MLM task and after fine-tune with click labels. We also use the SuKD approach with a light BERT-3 model as student model in the distillation phase.

For the sake of reproducibility, We provide all the details of the experiments run on the KDD cup 2012 data, including data preprocessing and all hyperparameter settings, in the appendix.

4.1.3 Evaluation Metrics. We are using, as most papers and researches in the space of CTR prediction, the AUC [3] and the RIG [10] to evaluate the end-to-end performance of CTR prediction. In order to account for infrequent (*Query, Ad*) couples which account for 42.6% of samples in Bing Ads, we evaluate the AUC and the RIG both on the whole dataset (called *ALL Slice*) and also on these infrequent couples (called *Tail Slice*), representing the tail behavior. The metrics on tail slice indicate the performance of CTR prediction model, and in particular the improvement provided by NLP features, for cold queries.

For knowledge distillation, we introduce the *Conversion Ratio* (*CR*), which measures the share of the gain in the teacher model that is transferred to the student model.

Without explicit statement, all results shown in this section are obtained during the best step in training process. The baseline is the CTR prediction model without new NLP features, and all improvements of AUC and RIG are relative to the baseline.

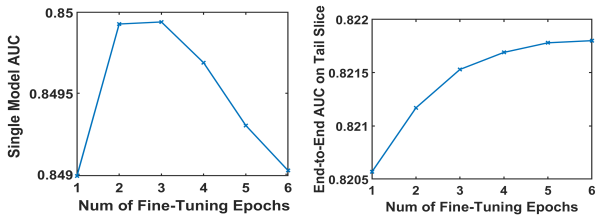
4.2 Results

In this section, we present the results obtained by using the experimental settings described earlier on the two datasets. It is noteworthy that in an industrial setting, on Bing Ads for example, even an AUC gain of 0.02% on ALL slice and of 0.05% on tail slice are statistically significant, and are moreover relevant from the business perspectives. For the KDD Cup 2012 data, the potential for improvement is larger than the Bing Ads, as the AUC/RIG of KDD Cup 2012 baseline are relatively weak. Using [26] as reference, 1% improvement in AUC of KDD Cup 2012 data can be seen as significant gain.

4.2.1 Inconsistency between Single-Model AUC and Global AUC. We present in Figure 3 the evolution during the fine-tuning epochs of the AUC obtained on the tail slice of Bing Ads dataset. We compare the single-model AUC for the RoBERTa-24 model in fine-tuning, along with the end-to-end AUC obtained over the full CTR prediction system, including the score from the RoBERTa-24 model fine-tuned with the same epochs. As can be seen in Figure 3, in the first three epochs, both the two increase along with each other, while after the third epoch the single-model AUC begins to drop while the end-to-end AUC is still climbing. This indicates the language model achieving the best single-model AUC is not the most favorable one for the end-to-end AUC.

This is in accordance with the theoretical analysis given before, which shows that single-model optimization yields global optimization only if the generated NLP feature is independent of the auxiliary ones. Thus, the observed inconsistency is likely resulting from the overlap between NLP features and previous features coming from the baseline. We investigate this in Section 3.

4.2.2 Contribution of Joint-Training. In this part, we first show the advantages of joint-training framework as described in Section 3.



(a) Single-Model AUC (b) End-to-End AUC on Tail Slice

Figure 3: Comparison of the trends of single-AUC and end-to-end AUC on tail slice as a function of the number of fine-tuning epochs on Bing Ads dataset

For this purpose, we compare three different designs: 1) node-wise summation in a fusion layer vs. concatenation of auxiliary layer and supplementary layer; 2) training all weights vs. fixing weights in auxiliary features part; 3) two-steps initialization vs. random initialization for all weights.

We show in Table 1 the comparison of results obtained over Bing Ads data. The table shows that the node-wise summation achieves a supplemental 0.06% end-to-end AUC gain on tail slice compared with the concatenation of auxiliary and supplementary layers. A second element is that the comparison between the first and the third lines indicates that making all weights trainable yields an additional 0.14% end-to-end AUC gain on the tail slice, compared with fixing the weights in the auxiliary features part. Finally, the two-steps initialization can provide an additional 0.15% end-to-end AUC gain compared to the weights random initialization at the beginning of joint-training.

In Table 2, we present a comparison of the end-to-end AUC/RIG gain obtained with separately fine-tuning the language model and the joint-training with auxiliary CTR prediction features over our two datasets. In order to make a fair comparison, as the NLP feature from joint-training is a 30-dimensional vector, we also reduce the dimension of RoBERTa/BERT [CLS] pooling layer to 30 using a full-connection layer before the final output and present the results of end-to-end gain after separate fine-tuning of the language model, where we use the score only, or the 30-dimensional embedding, or both the score & the embedding from fine-tuning as three references.

On both the two datasets, the end-to-end AUC and RIG gains of three types of NLP feature from separate fine-tuning of language model are very similar. However, the joint-training of NLP and auxiliary features yields an end-to-end AUC/RIG gain that is 70% (resp., 80%) higher than the separate fine-tuning, both on ALL slice and tail slice on Bing Ads data (resp., KDD Cup 2012 data). The consistency of the improvement from joint-training over different datasets and different CTR prediction baselines, gives strong indication on the pertinence of our proposed joint-training approach.

We also evaluate the ideal situation where the network of language model is integrated directly into the CTR prediction baseline by increasing the dimension of input space, which can reduce the overlap between NLP feature and auxiliary features globally but needs much more training efforts. We show in Table 3 the comparison of the supplementary learning approach with such global

Framework			ALL Slice		Tail Slice	
Input Design for H_s	Trainable weights	Initialization	ΔAUC	ΔRIG	ΔAUC	ΔRIG
Node-wise summation	All weights	Two-steps	0.09%	0.32%	0.47%	2.53%
Concatenation of H_a, H_c	All weights	Two-steps	0.08%	0.31%	0.41%	2.22%
Node-wise summation	Only NLP feature part	Two-steps	0.06%	0.22%	0.33%	1.70%
Node-wise summation	All weights	Random initialization for auxiliary features part	0.05%	0.20%	0.32%	1.70%

Table 1: Comparison between end-to-end gains from different framework designs on Bing Ads Data

Dataset	CTR model	NLP Features	ALL Slice		Tail Slice	
			ΔAUC	ΔRIG	ΔAUC	ΔRIG
Bing Ads	NN+GBDT	FT Score	0.05%	0.19%	0.27%	1.50%
		FT Embedding	0.05%	0.20%	0.27%	1.43%
		FT Score & Embedding	0.05%	0.21%	0.28%	1.53%
		JT Embedding	0.09%	0.32%	0.47%	2.53%
KDD Cup 2012	Wide & Deep	FT Score	1.67%	1.61%	2.63%	2.17%
		FT Embedding	1.66%	1.65%	2.64%	2.31%
		FT Score & Embedding	1.68%	1.71%	2.66%	2.36%
		JT Embedding	3.14%	2.80%	4.35%	3.54%
	DeepFM	FT Score	1.67%	1.73%	2.62%	1.93%
		FT Embedding	1.67%	1.85%	2.64%	2.16%
		FT Score & Embedding	1.68%	1.91%	2.65%	2.24%
		JT Embedding	3.16%	2.61%	4.01%	3.37%

Table 2: Comparison of end-to-end gains obtained by adding NLP features generated from joint-training (JT) and separate fine-tuning (FT) on two datasets

learning on both datasets. The integrated model used for KDD Cup 2012 data is built by concatenating the 30-dimensional embedding from BERT-12 model with the embedding layer of Wide & Deep model (resp., DeepFM model) to generate a new embedding layer and all parameters in these two sub-models are optimized simultaneously in training. On the Bing Ads data, we replace the NN in NN+GBDT with our joint-training network (including both auxiliary features part and NLP feature part).

Dataset	CTR model	Technology	ALL Slice		Tail Slice	
			ΔAUC	ΔRIG	ΔAUC	ΔRIG
Bing Ads	NN+GBDT	Integration	0.10%	0.34%	0.50%	2.70%
		JT	0.09%	0.32%	0.47%	2.53%
KDD Cup 2012	Wide&Deep	Integration	3.30%	2.94%	4.41%	3.76%
		JT	3.14%	2.80%	4.35%	3.54%
	DeepFM	Integration	3.44%	2.78%	4.53%	3.53%
		JT	3.16%	2.61%	4.01%	3.37%

Table 3: Comparison between end-to-end gains of integrated model and NLP embedding features generated by joint-training on two datasets

From Table 3, we can observe that the end-to-end gain brought by the supplementary joint-training approach is close to the ideal gain expected from the integrated model, without the huge associated learning burden. This observation is valid over different CTR prediction baselines. For DeepFM baseline applied to KDD Cup 2012 data, the NLP features from joint-training holds 92% of the ideal end-to-end AUC gain, and this proportion increases even for Wide & Deep baseline as 95% of gain is held. On Bing Ads data, the joint learning approach maintains 90% of ideal end-to-end AUC gain. This validates the fact that the joint-learning approach is relevant, as it maintains more than 90% of the gain while having a much lower and tractable learning complexity. As an example, the integration of DeepFM and BERT-12 model in Table 3 should take 9 days on 8 × V100 GPUs to achieve the best end-to-end AUC while our simple joint-training framework can take only 5 days in the same baseline settings, which is more suitable for the frequent re-calibration in CTR prediction.

4.2.3 Contribution of SuKD. As described before, despite the significant end-to-end AUC gain coming from the pre-trained language model, its usage in practice is costly. As an example, the RoBERTa-24 model has 355M parameters. In order to manage the complexity and to make possible an online usage of the language models, we proposed in Section 3 a knowledge distillation approach, named SuKD, that can generate a light student model learning from the heavy teacher model. For this purpose we extend the joint-training to the knowledge distillation, and we show here the results obtained from SuKD on the two datasets.

We show in Table 4 a performance comparison of three distillation approaches based on a simple BERT-3 student model. The three distillation approaches are response-based distillation where the soft target of student model is the predicting score of teacher model, feature-based distillation using a soft loss function that is the MSE between the two 30-dimensional embedding generated from student and teacher model, and SuKD that combines the joint-training and feature-based distillation into a single consistent approach as described in Section 3. All approaches use the ensemble of soft and hard losses in distillation. As shown in Equation 7 the weights of soft loss and hard loss are trainable, and we use auto-tuning for them. We use the RoBERTa-24 (resp., BERT-12) as teacher model on Bing Ads data (resp., KDD Cup 2012 data). The performances are evaluated over both datasets and to make the comparison fair, for all the three distillation frameworks a 30-dimensional vector is fed into the CTR prediction model.

As shown in Table 4, we can see that both the SuKD and the feature-based distillation yield a high conversion ratio, *e.g.*, for Bing Ads data, the BERT-3 student model generated from SuKD (resp., feature-based distillation) maintains on tail slice 80.85% (resp. 77.78%) of the end-to-end gain obtained from teacher, that is better than the response-based distillation, with 66.67%. Additionally, SuKD benefits successfully from the advantage of joint-training. Overall, for the same student model complexity, *i.e.*, BERT-3, and the same NLP feature dimension, *i.e.,* 30-dimensional embedding,

Dataset	CTR model	KD Technology	Teacher				Student				CR of Student from Teacher			
			ALL Slice		Tail Slice		ALL Slice		Tail Slice		ALL Slice		Tail Slice	
			ΔAUC	ΔRIG	ΔAUC	ΔRIG	ΔAUC	ΔRIG	ΔAUC	ΔRIG	AUC CR	RIG CR	AUC CR	RIG CR
Bing Ads	NN+GBDT	SuKD	0.09%	0.32%	0.47%	2.53%	0.07%	0.25%	0.38%	2.05%	77.77%	78.13%	80.85%	81.03%
		Response-based KD	0.05%	0.19%	0.27%	1.50%	0.03%	0.13%	0.18%	0.95%	60.00%	68.42%	66.67%	63.33%
		Feature-based KD	0.05%	0.19%	0.27%	1.50%	0.04%	0.15%	0.21%	1.17%	80.00%	78.95%	77.78%	78.00%
KDD Cup 2012	Wide & Deep	SuKD	3.14%	2.80%	4.35%	3.54%	2.47%	2.22%	3.62%	2.84%	78.66%	79.29%	83.22%	80.23%
		Response-based KD	1.67%	1.61%	2.63%	2.17%	1.05%	1.07%	1.66%	1.34%	62.87%	66.46%	63.12%	61.75%
		Feature-based KD	1.67%	1.61%	2.63%	2.17%	1.31%	1.25%	2.16%	1.72%	78.44%	77.64%	82.13%	79.26%
	DeepFM	SuKD	3.16%	2.61%	4.01%	3.37%	2.62%	2.06%	3.35%	2.72%	82.91%	78.93%	83.54%	80.71%
		Response-based KD	1.67%	1.73%	2.62%	1.93%	0.98%	1.06%	1.60%	1.14%	58.68%	61.27%	61.07%	59.07%
		Feature-based KD	1.67%	1.73%	2.62%	1.93%	1.35%	1.36%	2.05%	1.56%	80.84%	78.61%	78.24%	80.83%

Table 4: Comparison between end-to-end gains of NLP embedding features generated by response-based distillation, feature-based distillation and SuKD on two datasets

SuKD can achieve additional gain on both ALL and tail slices. On Bing Ads data, there is 0.17% end-to-end AUC gain on tail slice compared to feature-based distillation. On KDD Cup 2012 data, there is 1.46% AUC gain on tail slice for Wide & Deep baseline and 1.30% gain on tail slice for DeepFM baseline. The comparison over different datasets and different CTR prediction baselines proves the effectiveness and universality of SuKD in the improvement of end-to-end gain on student model.

As the conversion ratio of SuKD and feature-based distillation are very close, one might consider other way to combine joint-training and knowledge distillation, *i.e.*, to do the feature-based distillation from a teacher model during a separate fine-tuning of the language model and then jointly train the student model with auxiliary CTR prediction features. In Table 5, we compare the latter solution with SuKD over Bing Ads data and show that SuKD method is giving the better performance. It is notable that the best end-to-end AUC (resp., RIG) gain on the tail slice of the alternative solution is 0.34% (resp., 1.79%), and it yields a loss of 0.04% (resp., 0.26%) compared with SuKD. This loss happens probably because the joint-training in teacher side can make the language model learn supplementary knowledge more efficiently than in student side.

KD Technology	ALL Slice		Tail Slice	
	Δ AUC	Δ RIG	Δ AUC	Δ RIG
SuKD	0.07%	0.25%	0.38%	2.05%
Feature-based KD and Joint-Training Student Model	0.06%	0.22%	0.34%	1.79%

Table 5: Comparison between end-to-end gains of two ways generating student NLP feature on Bing Ads data

Besides BERT-3, we also tried more complex BERT models with more transformer layers in the student side to investigate if the conversion ratio improves. Table 6 presents the end-to-end AUC gains and conversion ratios on tail slice of Bing Ads data with BERT-3, BERT-4, BERT-6 and BERT-12. As showed, more complex student model can yield higher conversion ratio. It is notable that BERT-6 with SuKD achieves more than 93% end-to-end AUC gain from teacher which has 4 times more transformer layers, meaning that SuKD is an efficient technique to transfer the end-to-end supplementary knowledge from teacher to student.

SuKD is a general knowledge distillation approach, where the student model is not limited to be similar with the teacher model. We therefore tried a wide and sparse DNN model as student, where the inputs are hundreds of millions of one-hot encoding features, that can be beneficial to represent the cross characteristics between the

Language model	ΔAUC on Tail Slice	CR of Student on Tail Slice
Teacher RoBERTa-24	0.47%	–
SuKD BERT-3	0.38%	80.85%
SuKD BERT-4	0.40%	85.11%
SuKD BERT-6	0.44%	93.62%
SuKD BERT-12	0.45%	95.74%

Table 6: Comparison on end-to-end AUC gains and conversion ratios on tail slice between different number of BERT layers in SuKD on Bing Ads data

query and ad texts, *e.g., Matched(Query,AdTitle)* might give multiple one-hot features indicating matched word between query and ad. All one-hot encoding features are mapped into a 400-dimensional embedding layer, and two full-connected hidden layers with respectively 200 and 30-dimensions are following before a final output node.

We compare in Table 7 the obtained end-to-end gains of two 30-dimensional NLP features generated from the last hidden layer of the DNN model with two different distillation techniques: response-based distillation from RoBERTa-24 model with separate fine-tuning alone, and SuKD from RoBERTa-24 model with joint-training. The table shows that SuKD applied over the DNN model can bring an additional 0.10% end-to-end AUC gain on tail slice compared with response-based distillation. This result is in line with the conclusion drawn from BERT-3, showing the generality of our conclusions.

KD Technology	ALL Slice		Tail Slice	
	ΔAUC	ΔRIG	ΔAUC	ΔRIG
SuKD	0.05%	0.18%	0.25%	1.34%
Response-based KD	0.03%	0.10%	0.15%	0.80%

Table 7: Comparison of end-to-end gains between DNN features generated by two KD technologies on Bing Ads data

4.3 Online Performance

The CTR prediction model with new NLP feature has deployed in an online A/B testing. In this testing, we scheduled two flights (*i.e.*, control and treatment) with randomly sampled traffic and similar configuration settings except the CTR prediction models. For the control flight we used the baseline CTR prediction model and for the treatment one we used the new CTR prediction model with supplementary NLP features.

We deployed the distilled BERT-12 model with SuKD from RoBERTa-24 model in Bing.com Ads system to generate the 30-dimensional supplementary NLP feature for online CTR prediction. Twenty V100 GPUs were allocated for online application. The resulting

system achieved a 6.0 ms latency per <query, ad impression> pair, which is tolerable.

We observed during the online A/B testing for one month, the treatment flight brings 2% gain on click yields with only 0.6% increase on display yields, compared with the control flight, where 1% fluctuations on click yields and display yields can be seen as significant changes. This provides furthermore evidence for the interest of the method in a real deployment.

5 CONCLUSION

In this paper, we have focused on learning supplementary knowledge for end-to-end CTR prediction model using NLP features. We introduced a novel joint-training framework which can boost effectively the contribution of the NLP features in CTR prediction. We further proposed a distillation approach for the deep models that transfers efficiently the supplementary knowledge from a heavy teacher model to light student models. We evaluated the proposed approaches using a set of comprehensive experiments, and showed that the proposed joint-training technology and supplementary knowledge distillation framework are achieving performance gains in both commercial and public data, and therefore provide promising solutions for real world deployment.

ACKNOWLEDGMENTS

Research for this publication was supported by the EU Horizon2020 project MariCybERA (agreement No 952360).

REFERENCES

[1] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 535–541.

[2] 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.

[3] Corinna Cortes and Mehryar Mohri. 2004. AUC optimization vs. error rate minimization. *Advances in neural information processing systems* 16, 16 (2004), 313–320.

[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] Tian Gao, Jun Du, Li-Rong Dai, and Chin-Hui Lee. 2015. Joint training of front-end and back-end deep neural networks for robust speech recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4375–4379.

[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] 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.

[11] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. 2019. A comprehensive overhaul of feature distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1921–1930.

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

[13] Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *arXiv preprint arXiv:1801.06146* (2018), 194.

[14] Zehao Huang and Naiyan Wang. 2017. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219* (2017).

[15] 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).

[16] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. DeepGBM: A deep learning framework distilled by GBDT for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 384–394.

[17] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

[18] Jangho Kim, SeoungUK Park, and Nojun Kwak. 2018. Paraphrasing complex network: Network compression via factor transfer. *arXiv preprint arXiv:1802.04977* (2018).

[19] 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).

[20] Tingting Liang, Congying Xia, Yuyu Yin, and Philip S Yu. 2020. Joint Training Capsule Network for Cold Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1769–1772.

[21] 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.

[22] 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).

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

[24] 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.

[25] 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.

[26] 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.

[27] Nikolaos Passalis and Anastasios Tefas. 2018. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 268–284.

[28] Manish Patel. 2019. TinySearch–Semantics based Search Engine using Bert Embeddings. *arXiv preprint arXiv:1908.02451* (2019).

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

[30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* (2014).

[31] Hagen Soltau, George Saon, and Tara N Sainath. 2014. Joint training of convolutional and non-convolutional neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5572–5576.

[32] 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).

[33] 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.

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

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

[36] Weijie Zhao, Jingyuan Zhang, Deping Xie, Yulei Qian, Ronglai Jia, and Ping Li. 2019. Aibox: Ctr prediction model training on a single node. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 319–328.

[37] Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. 2018. Rocket launching: A universal and efficient framework for training well-performing light net. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

A APPENDIX

To ease the reproducibility of the work, we are providing here some details about the experiments on the public KDD Cup 2012 dataset, including the data description, the baseline settings and the language model settings.

A.1 Data Details

The dataset contains 235 million search ads impressions sampled from session logs of the Tencent search engine Soso.com. Each sample in this dataset is similar to Bing Ads data and contains five components:

- Query text: a list of tokens hashed from the natural language;
- Ad text: including ad title and ad display URL which are also a list of tokens;
- CTR prediction features: containing 57 CTR features used for prediction including sparse features such as UserID, AdID, user's gender etc., and numerical features such as historical CTRs, impressions per Ad/Advertiser/Query/User, user's age, and depth of session etc.;
- Position feature: indicating the position where the ad is shown;
- Click label: 1 means the ad has been clicked and 0 otherwise.

Differently from Bing Ads data, all textual features in KDD Cup 2012 dataset are anonymous, and it is difficult to map the hash ID from data to vocabulary ID of pre-trained language models. We solve this issue, by treating these anonymous tokens as new words and building the map from these hash IDs to new extended IDs following existing vocabulary of pre-trained language models.

As there is no time information in this dataset, it is impossible to split the training data and validation data according to impression time. We use a simple way to generate the training and validation datasets, by randomly choosing 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 dataset

A.2 Baseline Settings

We first introduce the generation of embedding layer for both Wide & Deep and DeepFM. As described in A.1, there are sparse features, textual features and numerical features in each training sample and for different types of feature, the processing methods are different. For sparse features, we extent each input to a 8-dimensional embedding respectively. In terms of textual features such as query text, ad title and ad display URL, we first extent each word into a 8-dimensional word-embedding, and then use Average-Pooling to get a sentence-level embedding. For numerical features, we normalize them with max-min normalization at first and then concatenate these into embedding layer directly.

- **Wide & Deep**: The wide component is a Logistic Regression that takes one-hot vectors as inputs and the deep component has embedding layer for each input and two following dense hidden layers with ReLU activation. The wide component and deep component are combined using a weighted sum of their output logits as the final prediction score.
- **DeepFM**: In FM component, we generate the first-order features from the input features directly and generate the second-order features by crossing the embedding layer described above. In deep component, we use two ReLU layers stacked over the embedding layer.

A.3 Language Model Settings

The teacher model of fine-tuning is initialized by BERT-12 model. The query and the concatenation of the ad title with its display URL are used as two input sentences.

As mentioned in Section 3, the training process consists of two phases: at first training separately the initial weights for auxiliary and NLP features, and in the second phase, these two parts are jointly trained. For auxiliary features, we follow the network architecture in [26], from which the embedding layer described in A.2 is borrowed, and stack two dense hidden layers with ReLU activation after the embedding layer. These layers have each a dimension of respectively 30 and 10, besides one separate position node. We train this network independently over 10 epochs with a learning rate 5e-4 to get the initial weights of the auxiliary features part. For initialization of NLP feature part, the standard BERT-12 model is pre-trained on the training data with MLM task, where the masking rate is 15%, and then is fine-tuned with click labels over the same training data. The hyperparameters are set to the same values in both pre-training and fine-tuning. The training batch size is 256, learning rate is 2e-5 with linear decay, and the epochs number is set to 4.

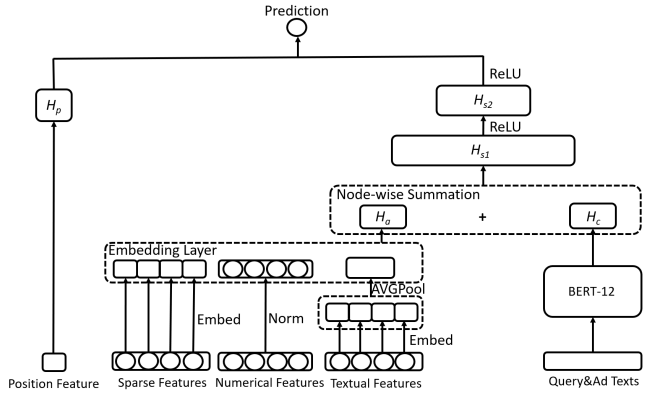


Figure 4: Framework of Joint-Training on KDD Cup 2012 Data

For joint-training, the [CLS] pooling output from the language model and the embedding layer from auxiliary features are mapped to 30-dimensional layers H_c and H_a respectively, and then node-wise summed, yielding the input of the interaction layers. We set a small learning rate value of 1e-5 with linear decay and only train for two epochs to avoid over-fitting. Figure 4 depicts the framework we used for joint-training of the KDD Cup 2012 data.

In distillation phase, we use light BERT-3 model as student model. The model is initialized with the weights from the bottom three layers of the jointly-trained BERT-12 model coming from the teacher model. The learning rate is set as 1e-5 with linear decay and the number of distillation epochs is 5.