Masked-field Pre-training for User Intent Prediction

Peng Wang
AI Labs, Didi Chuxing
wangpengeric@didiglobal.com

Jiang Xu *
AI Labs, Didi Chuxing
xujiang@didiglobal.com

Chunyi Liu *
AI Labs, Didi Chuxing
liuchunyi@didiglobal.com

Hao Feng
AI Labs, Didi Chuxing
fengfenghao@didiglobal.com

Zang Li
Tencent
gavinzli@tencent.com

Jieping Ye
AI Labs, Didi Chuxing
yejieping@didiglobal.com

ABSTRACT

For many applications, predicting the users' intents can help the system provide the solutions or recommendations to the users. It improves the user experience, and brings economic benefits. The main challenge of user intent prediction is that we lack enough labeled data for training, and some intents (labels) are sparse in the training set. This is a general problem for many real-world prediction tasks. To overcome data sparsity, we propose a masked-field pre-training framework. In pre-training, we exploit massive unlabeled data to learn useful feature interaction patterns. We do this by masking partial field features, and learning to predict them from other unmasked features. We then finetune the pre-trained model for the target intent prediction task. This framework can be used to train various deep models. In the intent prediction task, each intent is only relevant to partial features. To tackle this problem, we propose a Field-Independent Transformer network. This network generates separate representation for each field, and aggregates the relevant field representations with attention mechanism for each intent. We test our method on intent prediction datasets in customer service scenarios as well as several public datasets. The results show that the masked-field pre-training framework significantly improves the prediction precision for deep models. And the Field-Independent Transformer network trained with the masked-field pre-training framework outperforms the state-of-the-art methods in the user intent prediction.

CCS CONCEPTS

• **Information systems** → *World Wide Web.*

KEYWORDS

User intent prediction, Pre-trained models, Transfer learning, Supervised learning

ACM Reference Format:
Peng Wang, Jiang Xu *, Chunyi Liu *, Hao Feng, Zang Li, and Jieping Ye. 2020. Masked-field Pre-training for User Intent Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412726

* Jiang Xu and Chunyi Liu have equal contributions.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412726

1 INTRODUCTION

Predicting the user's intent is very useful for many applications. For example, e-commerce applications predict which items users are interested in, to improve sales. Advertising systems predict click-through-rate (CTR) to measure which ads match users' interests. In ride-hailing firms like Uber and DiDi, customer service needs to anticipate which problems drivers and passengers may have. An automatic service can display relevant <Question, Answer> pairs to users even before they begin to describe their problems. In this case, user-intent prediction boosts the user experience and saves human customer service resources.

We treat intent prediction as a multi-label classification problem, as each user may have multiple questions. Given the user profile and behaviors, the system predicts the probability of each intent and recommends the TOP-K corresponding <Question, Answer> pairs. User intent prediction is closely related to the CTR prediction problems. Recently, CTR prediction models achieve great progresses with the development of deep learning. These models generally follow an embedding & DNN paradigm [5]: an embedding layer maps the high-dimension features into low-dimensional embedding vectors, then the embedding vectors are fed into a deep neural network. The DNN part learns the feature interactions and produces the output. Simple DNN model just stacks fully connected layers together (also known as a multilayer perception, MLP). The recently emerged complex architectures such as attention mechanisms or recurrent layers are introduced to model the complex feature interactions [26, 29, 36, 37].

Even with this progress, user intent prediction still poses great challenges. First, there is a limited amount of labeled data, especially for sparse intents. The prediction models tend to overfit the noise in the data instead of learning general patterns. Second, the previous methods compress all the information into a single vector, then use it to predict all the intents. Generally, each intent is only related to some features. For example, to predict whether the user has questions about the bill, the model only needs to consider features such as bill details, payment time, payment method etc. When predicting whether the user encounters detour related issues, the model only needs to consider features such as estimated trip distance, actual trip distance, etc. As a result, the previous models fail to predict the intent well, especially for sparse intents.

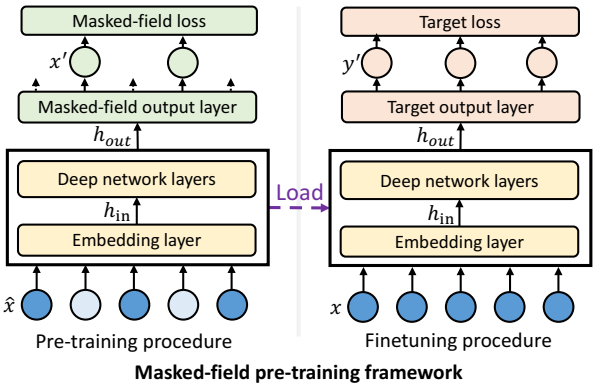


Figure 1: Illustration of the masked-field pre-training framework. In the pre-training procedure, we learn to predict the masked field features with unlabeled data. Then in the finetuning procedure, we reuse the embedding layer and DNN layers, and finetune the model for the target task with labeled data.

To solve the first challenge, we propose a masked-field pre-training framework. The framework is motivated by the great success of pre-trained models in NLP tasks [6, 20, 31]. For example, BERT [6], which is pre-trained with massive unlabeled texts, greatly boosts the performance on a set of NLP tasks. The pre-trained model learns the feature interaction patterns with the unlabeled data, and reuses these patterns for the target task. Here is an intuitive example to illustrate how it works. Assume that the intent A is closely related to a sparse feature B. With limited labeled data, the model may fail to learn their relations, so that feature may get ignored. However, if the pre-trained model learns how to infer feature B with other features with massive unlabeled data, then the model can infer the similar representation of feature B even if feature B is missing. As a result, we will predict intent A more precisely.

In the masked-field pre-training framework, we use both unlabeled and labeled data for training the intent prediction model. In the pre-training procedure, we mask small percentage of the input features at random, and learn to predict the masked features with a multi-task model. In the process, the pre-trained model learns the feature interactions and how to represent sparse features with other features. In the finetuning procedure, we use the pre-training model as the starting point for the target task, so as to reuse the patterns instead of learning them from scratch. The masked-field pre-trained model helps improve the target tasks in three ways. First, as there are abundant unlabeled data, the pre-training model can learn the feature embeddings better, as the sparse features can be absent or rarely seen in the limited labeled data. Second, the feature interaction patterns can be useful for both feature prediction and target task. But with limited labeled data, the task model may fail to figure out the patterns. Third, some intents are closely related to certain features. The knowledge for predicting the features can be directly reused for predicting the intents. This is especially useful for sparse intents. Note that the framework can be used to train various deep models.

To tackle the second challenge which raised by multi-label classification, we propose a Field-Independent Transformer network (FI-Trans in short). The network is composed of an embedding layer, field-independent Transformer layers which aims to learn the feature interactions and produce contextual representations of features, and an output layer with an attention mechanism. Instead of using a single representation to predict the CTR for all intents like aforementioned CTR models [5, 37], the field-independent Transformer layers give separate representation for each field, then in the output layer, each intent only attends to partial features to generate final results. In this manner, FI-Trans model tends to achieve better performance in the multi-label task.

The contributions of this paper are as follows:

- We propose a masked-field pre-training framework which utilizes unlabeled data for learning feature interactions and sparse feature representations. The pre-training model can boost the performance of target tasks.
- We propose a field-independent Transformer network which well fits the multi-label classification problem.
- We test the masked-field pre-training framework on various network architectures as well as the proposed Fi-Trans model, and find that it significantly improves the prediction performance on our user intent prediction task as well as several public datasets. It indicates that masked-field pre-training framework can be applied in all prediction tasks with abundant unlabeled data but limited labeled data.
- We experimentally find that Fi-Trans model trained with masked-field pre-training framework outperforms other state-of-the-art methods in the user intent prediction task. It significantly improves the CTR of user intent recommendation for online traffic.

2 RELATED WORKS

Our proposed methods are closely related to recommendation, pre-training, and sparse auto-encoder.

2.1 Recommendation

Our work shares similarity with recommendation in that recommendation tries to predict users' interested items before the users give explicit queries; while user intent prediction tries to predict intents before they ask questions. The early stages of recommendation models directly adopt linear models (e.g., Logistic Regression) or nonlinear models (e.g., XGBoost[2] and FM[22]) to predict the CTR of <user, item> pairs and recommend the items with highest CTR. Recently, deep models are widely adopted as they can better capture the complex interactions between features. The Wide&Deep model [3] jointly learns a wide linear model (for memorization) and deep model (for generalization) to combine the benefits of both parts. Similarly, the DeepFM [9] combines the power of factorization machines and deep learning in a new neural network architecture. Attention FM [29] believes that the feature interactions should not share the same weights, so it uses the attention mechanism to model the importance of feature interactions. The recent progress of CTR models argues that sequential behavior and interest evolution are

important for prediction. The representative work includes Deep Interest Network (DIN) [37], Deep Interest Evolution Network (DIEN) [36], and Bert4Rec [26].

Our proposed FI-Trans model can also learn the complex feature interaction patterns. And if we have features that represent a user's behavior at different time slots, our model can also handle sequential behaviors. These aforementioned models produce a single feature representation for all prediction tasks which works well for single task learning. While the user intent prediction is a multi-label classification problem, our model produces a representation for each field. For each intent, it uses an attention mechanism to aggregate partial relevant field representations for prediction.

2.2 Pre-training models.

Pre-training is a machine learning method where we reuse a model trained to solve one problem and apply it to a different but related problem. Pre-training works if the learned knowledge from the source task are general, instead of specific to the source task [32]. It achieves great success in natural language processing [4, 6, 7, 31], computer vision [12, 33], and recommendation [30, 35].

In recent years in natural language processing, the pre-trained models, such as BERT [6] and XLNet [31] trained on unlabeled data, has greatly improved the performance of the state-of-the-art methods. It motivates us to explore unsupervised pre-training for our user intent prediction task. The aforementioned models can only handle sequential data; while our model needs to handle unordered features with various data types.

2.3 Sparse Auto-encoder.

The sparse auto-encoder [17] aims to compress the sparse high-dimensional features into lower dimensions. The dense representation reserve the key information of items and users and have better generalization ability. Recommendation systems can use auto-encoders to learn representations of items and users so as to improve the performance of collaborative filtering [14–16, 24, 28, 34].

In the training procedure of sparse auto-encoder, all the features are given as input and act as labels. The model mainly learns how to compress the features. If the hidden state is large enough, the auto-encoder only learns how to copy the input to the output, and does not necessarily learn useful feature interactions. In our masked-field pre-training procedure, the pre-training model learns feature interactions as the target features are masked and inferred with other features. This knowledge can be generalized to other tasks.

3 PRELIMINARIES

In this section, we introduce our problem settings. We have an unlabeled dataset $D^u = \{x_1, x_2, \cdots, x_N\}$ with N records. We collect the unlabeled data from the traffic logs where the users do not explicitly express their intents. We also have a labeled dataset $D^l = \{(x_1, y_1), (x_2, y_2), \cdots, (x_M, y_M)\}$ with M records. D^l is from the traffic logs where the users click the recommended intents or express their intents which belong to our intents set. In our settings, we have $M \ll N$.

For each record, there are K fields, i.e., $x_i = \left\{x_i^1, x_i^2, \cdots, x_i^K\right\}$. We briefly clarify the relations between fields and features. A field is composed of a set of closely related features. For example, *Sunday* is a feature, which corresponds to a unique one-hot vector; while *weekday* is a field, which corresponds to a set of one-hot vectors with the same length. For data records, a feature can be sparse, i.e., the value rarely appears; while all the records share an identical set of fields. For real-world data, fields have different data types, e.g., numeric, category, text, etc. We turn all types of data into one-hot or multi-hot vectors. Specifically, we turn numerical features into the one-hot vector with quantile statistics, and turn categorical features into one-hot vectors (e.g., day of the week) or multi-hot vectors (e.g., historical intents); we turn text into multi-hot vectors with a bag-of-words model. Note that all vectors for fields have the lengths equal to the number of possible features plus one. The extra position is used to indicate that the field is masked in the masked-field pre-training framework. For example, the field *weekday* is a 8-dimension vector (7+1), and vector $[0, 0, \cdots, 0, 1]$ indicates that the field is masked. The record x_i is a vector of $\{0, 1\}$:

$$\underbrace{[0, 1, \cdots, 0]}_{\text{weekday=Monday}} \quad \underbrace{[0, 1, \cdots, 0]}_{\text{fee=18}} \quad \underbrace{[0, 1, 0, 1, \cdots 0]}_{\text{comments=kind and warmhearted}}$$

For labeled record (x_i, y_i), y_i is the label vector. We have $y_i = \left\{y_i^1, y_i^2, \cdots, y_i^T\right\}$, where T is the label size. $y_i^t \in \{0, 1\}$, where 0 indicates the absence of t^{th} intent, while 1 indicates the existence.

In this paper, we study how to learn the prediction function $f(x_i) \longrightarrow y_i$, with both D_u and D_l.

4 MASKED-FIELD PRE-TRAINING FRAMEWORK

In this framework, the training process is composed of a masked-field pre-training procedure and a finetuning procedure (as illustrated in Figure 1). The masked-field pre-training procedure learns useful feature interaction patterns with unlabeled data, and the finetuning procedure reuses this knowledge for the target task. We go through the details of the framework.

4.1 The Pre-training Procedure

The basic idea of masked-field pre-training is to learn feature interaction patterns with unlabeled data. For each record, we randomly mask partial fields and treat them as unknowns, and we learn to predict their values with the other fields. We denote the masked fields for record i as F_i, the learning target as $x_i^{mask} = \left\{x_i^k | k \in F_i\right\}$. The number of masked fields is set to $\lceil \gamma K \rceil$, where γ is the mask ratio. For input records, the masked fields are set to special values (as described in Section 3). We annotate the transformed input as $\widehat{x_i}$. The pre-training model aims to predict the masked features in a multi-task manner (as illustrated in Figure 1), i.e., $f_{pre}(\widehat{x_i}) \longrightarrow x_i^{mask}$.

The pre-training model is composed of an embedding layer, deep network layers, an output layer, and a loss layer. The embedding layer turns the one-hot and multi-hot high-dimensional vectors $\widehat{x}$ into low-dimensional dense vectors $h_{in} = \left\{h^1, h^2, \cdots, h^K\right\}$. The deep network layers take h_{in} as input and output the crossed representations of fields h_{out}. The output layer gives the predicted fields x'. Note that there are two types of fields. For one-hot fields, the output is given by a softmax function, $x'^k = softmax(h_{out} W^k)$, where $W^k \in R^{d \times L^k}$ is a learnable parameter and L^k is the length of field k, and d is the size of h_{out}. While for multi-hot fields, we predict the

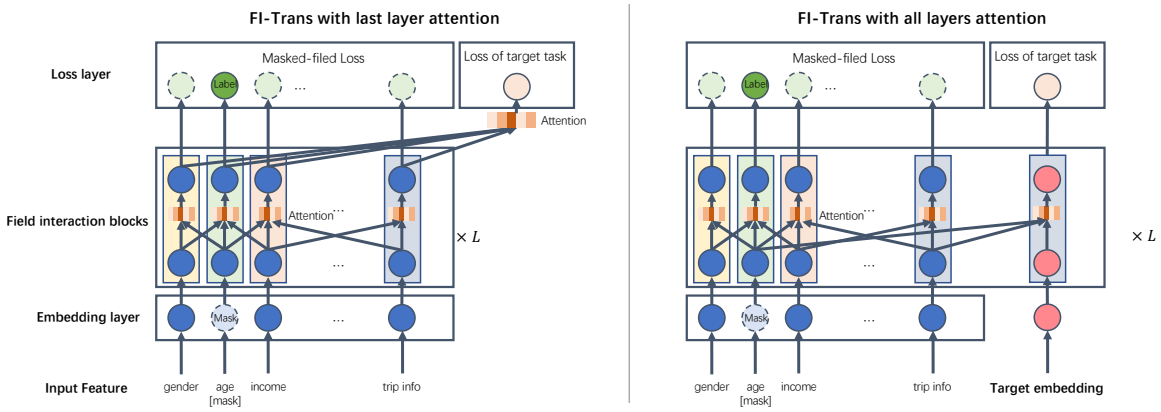


Figure 2: Illustration of FI-Trans model. The model is composed of an embedding layer, a set of field interaction blocks, and output layers for both pre-training and target tasks. The left part illustrates the FI-Trans with last layer attention pooling; the right part illustrates FI-Trans with all layers attention pooling. We omit some lines for attention pooling to keep it clean.

probability of each feature in the field, i.e., $x'^k = sigmoid(h_{out}V^k)$, where V^k is also a learnable parameter as W^k. The loss layer gives the training loss. For a single record, the loss function is

$$Loss(x_i^{mask}, x_i') = \sum_{k \in F_i^o} L_o(x_i^k, x'^k_i) + \sum_{k \in F_i^m} \frac{1}{L^k} \sum_{l=1}^{L^k} L_m(x_i^{k,l}, x'^{k,l}_i) \tag{1}$$

where F^m is the masked multi-hot field set, while F^o is the masked one-hot field set. L_o and L_m are the losses for one-hot and multi-hot fields respectively.

Note that, in the above section, we do not specify the deep network architecture, as we argue that the pre-training framework works well on a wide range of deep network architectures.

4.2 Finetuning Procedure

The finetuning procedure is to train $f_{task}(\cdot)$ with the labeled data D^l. In the procedure, we reuse the embedding layer and the deep network layers in the pre-training model, and only need to append a task-specific output layer and a loss layer.

For our user intent prediction task, we use T binary classifiers for prediction, one for each intent. The outcome is $y' = \{y'_1, y'_2, \cdots, y'_T\}$. The loss can be calculated as:

$$Loss(y_i, y_i') = \sum_{t=1}^{T} y_i^t \log y'^t_i + (1 - y_i^t) \log \left(1 - y'^t_i\right) \tag{2}$$

Both the common layers and the task-specific layers are optimized for the target task.

5 FIELD INDEPENDENT TRANSFORMER

In the previous section, we propose a pre-training framework which can be used to train various deep models. In this section, we propose a network architecture, i.e., Field-Independent Transformer (FI-Trans), which well fits the user intent prediction task.

The basic idea is that for each intent, the FI-Trans attends to relevant fields instead of using a universal representation to predict all intents. The FI-Trans is composed of an embedding layer, field interaction layers, and an output layer. The embedding layer maps the input x into dense vectors $h_{in} = \{h^1, h^2, \cdots, h^K\}$ as described in the Section 4.1, so we skip this part.

5.1 Field Interaction Layer

The field interaction layer takes context-free field representations h_{in} as input, and produce the *contextual* representation for each field as output. Here, *contextual* means that the representation already incorporates the information of other fields.

A field interaction layer is composed of K parallel stacked field interaction blocks (FIB for short), one for each field. The structure of FIB is similar to the Transformer block [27]. But our FIBs do not share parameters across fields, while the Transformer model shares the parameters across different positions (similar to the fields in our paper). To emphasize the differences, we name our model as Field-Independent Transformer. For the FIB of field k in the n-th layer, it uses attention mechanism to aggregate information from other fields into a new representation, i.e.,

$$FIB_k\left(h_{n-1}^k, \left\{h_{n-1}^1, h_{n-1}^2, \cdots, h_{n-1}^K\right\}\right) \longrightarrow h_n^k \tag{3}$$

FIB is composed of a multi-head self-attention sub-layer, a point-wise feed forward sub-layer. To be clear, we omit the layer subscript n in the notations. In the multi-head self-attention sub-layer, self attentions [27] are parallel applied with r heads. For each head i, we map the input representations of all fields as follows:

$$Q_{(i)}^k = W_{Q_{(i)}}^k h^k,\ M_{(i)}^k = W_{M_{(i)}}^k h^k,\ V_{(i)}^k = W_{V_{(i)}}^k h^k, \tag{4}$$

where $W_{Q_{(i)}}^k \in \mathbb{R}^{(d/r)\times d}$, $W_{M_{(i)}}^k \in \mathbb{R}^{(d/r)\times d}$ and $W_{V_{(i)}}^k \in \mathbb{R}^{(d/r)\times d}$ for $k = 1, 2, \cdots, K$ and $i = 1, 2, \cdots, r$ are field-independent and head-independent trainable parameters, and d is the size of h^k. We concatenate the results of all the heads as output h_{mha}^k. The multi-head self-attention sub-layer works as follows:

$$\begin{aligned} h_{mha}^k &= [head_1^k; head_2^k; \cdots; head_r^k] W_O^k \\ head_i^k &= Att\left(Q_{(i)}^k, \left\{M_{(i)}^1, \cdots, M_{(i)}^K\right\}, \left\{V_{(i)}^1, \cdots, V_{(i)}^K\right\}\right) \\ Att(Q, M, V) &= softmax\left(QM^T/\sqrt{d/r}\right)V \end{aligned} \tag{5}$$

where $W_O^k \in \mathbb{R}^{d \times d}$ is field-independent trainable parameters.

Then, we apply the filed-independent feed-forward sub-layer on the h_{mha}^k to produce the output of FIB as follows:

$$h_{pwf}^k = W_{F_2}^k \text{Gelu}\left(W_{F_1}^k h_{mha}^k + b_{F_1}^k\right) + b_{F_2}^k \tag{6}$$

where $W_{F_1}^k \in \mathbb{R}^{d' \times d}, W_{F_2}^k \in \mathbb{R}^{d \times d'}, b_{F_1}^k \in \mathbb{R}^{d'}$ and $b_{F_2}^k \in \mathbb{R}^d$ are trainable parameters for the k-the field, Gelu$(\cdot)$ [11] is the active function, and d' is the intermediate dimension. Moreover, we adopt the residual connection [10] and layer normalization [1] among the inputs and outputs of the multi-head self-attention sub-layer and point-wise feed-forward sub-layer to get h_n^k.

For each field, we stack L layers of FIBs. The final output of field interaction layers h_{out} is a set of contextual field representations, i.e., $h_{out} = \{h_{out}^1, h_{out}^2, \cdots, h_{out}^K\}$.

5.2 Output Layers

The output layers in the pre-training task and finetuning task are different. For the pre-training task, for each masked field k, we use its contextual representation, i.e., h_{out}^k, to produce the final output. Because h_{out}^k contains the information most relevant to field k. The details of the output and the loss layer can refer to Section 4.1. For the finetuning task, we propose two methods for generating the final outputs.

5.2.1 Output with last layer attention pooling. For target task t, we use attention pooling to aggregate relevant information from h_{out} into a single vector h^t,

$$\begin{aligned} h^t &= softmax\left([z_1, z_2, \cdots, z_K]\right) h_{out} \\ z_k &= v_t^T \tanh\left(W_t h_{out}^k + b_t\right) \end{aligned} \tag{7}$$

where $v_t \in \mathbb{R}^d, W_t \in \mathbb{R}^{d \times d}$ and $b_t \in \mathbb{R}^d$ are trainable parameters. Thus, the output logit is calculated with $s_t = w_t^T h^t + b_t$, where $w_t \in \mathbb{R}^d$ and $b_t \in \mathbb{R}$ are trainable parameters. The output layer for each intent has independent parameters.

5.2.2 Output with all layers attention pooling . Instead of only using the output of field interaction layers h_{out} for generating the final output, we can also use the internal states of field interaction layers, i.e., $\{h_1, h_2, \cdots, h_n\}$. We achieve this by treating the target intents as [CLS] fields, which is motivated by BERT model [6].

In the finetuning procedure, we treat the [CLS] fields as masked fields and try to predict their values. In the field interaction layers, for each layer, the [CLS] field can aggregate information from all the feature fields. The output of field interaction layers becomes $h_{out} = \left\{h_{out}^1, \cdots, h_{out}^K, h_{out}^{[CLS_1]}, \cdots, h_{out}^{[CLS_T]}\right\}$, where T is the total number of intents. Thus, for intent t, the logit is $s_t = w_t^T h_{out}^{[CLS_t]} + b_t$, where $w_t \in \mathbb{R}^d$ and $b_t \in \mathbb{R}$ are trainable parameters.

5.3 Discussion

We briefly explain why the field interaction layers have such designs. With FIBs, each field has its own contextual representation h_{out}^k. The capacity h_{out} is growing with the number of fields. For the target tasks, we use attention pooling mechanism to aggregate information from relevant fields. In contrast, h_{out} given by previous models is a single vector. It compresses all the information for target tasks. The capacity of a single vector is very limited, and optimizing the representations for one task has a direct impact on other tasks.

Table 1: Statistics of the datasets

Datasets	Unlabeled	Train	Dev	Test	Fields
Driver UIP	4.6M	100k	100k	100k	210
Passenger UIP	4.4M	100k	100k	100k	250
Criteo	30M	3M	3M	3M	39
Avazu	40M	4M	4M	4M	21

Table 2: The hyper parameters and model size.

Datasets	Mask ratio	Emb	Hidden	Head	Layer	Total
Driver UIP	0.1	8	32	4	2	5.0M
Passenger UIP	0.1	8	32	4	2	5.9M
Criteo	0.1	8	32	4	4	2.3M
Avazu	0.1	16	128	2	4	15M

6 EXPERIMENTAL STUDY

6.1 Data

We conduct experiments on the following datasets.

6.1.1 User intent prediction (UIP) dataset. We collect the user intent prediction datasets from the logs in DiDi customer service center. The intents and features of passengers and drivers are quite different, so we divide the data into a *Passenger* dataset and a *Driver* dataset. For both datasets, the 200 intents with the highest frequency account for 99% of data records. For these tasks, we evaluate the models with the average *AUC* over all intents.

6.1.2 Public datasets. We introduce two public datasets to verify whether our methods work on other fields. The tasks of the Criteo dataset [1] and Avazu dataset [2] are to predict the click-through-rate on display advertisements. These datasets are collected from the click through logs. The labels are whether the users click the ads. For these tasks, we evaluate the model performance with *AUC*.

In these datasets, all records are labeled. We randomly select a portion of records as the unlabeled data for pre-training, and use the remaining as the labeled data. All labeled data are split into 3 partitions for training, development and testing. We list the data statistics in Table 1.

6.2 Comparison Methods

We implement several prediction models as comparisons.

- The factorization machine (**FM**) [22] considers the second order feature interactions.
- The deep factorization machine (**DeepFM**) [9] deepens the FM with neural architecture to learn high-order interactions.
- The attention factorization machine (**FM-Att**) [29] introduces the attention mechanism to learn the importance of second order features in feature interactions.
- The deep neural network (**DNN**) is a widely used for various prediction tasks. In our settings, the DNN is composed of

[1]The dataset is available in https://www.kaggle.com/c/criteo-display-ad-challenge
[2]The dataset is available in https://www.kaggle.com/c/avazu-ctr-prediction/data

Table 3: AUC results for UIP and Criteo datasets (higher is better). We compare different (training mode, model) combinations, and highlight the best combinations on each dataset. Rel. diff is the relative differences over the scratch mode.

Datasets	Training Mode	Models								
		FM	DeepFM	FM-Att	DNN	W&D	IPNN	Trans	FI-Trans+LL	FI-Trans+AL
Passenger UIP	Scratch	0.6985	0.7409	0.6989	0.7199	0.7232	0.7334	0.7144	0.7564	0.7600
	MF pre-training	0.6850	0.7494	0.6855	0.7573	0.7473	0.7699	0.7664	**0.7788**	0.7780
	Rel. diff	-1.93%	1.15%	-1.92%	5.20%	3.33%	4.98%	7.28%	2.96%	2.37%
Driver UIP	Scratch	0.6725	0.6449	0.6534	0.6455	0.6585	0.6582	0.6680	0.6837	0.6896
	MF pre-training	0.6566	0.6775	0.6387	0.6799	0.6834	0.6865	0.6846	0.7010	**0.7064**
	Rel. diff	-2.36%	5.06%	-2.25%	5.33%	3.78%	4.30%	2.49%	2.53%	2.44%
Criteo	Scratch	0.7836	0.7873	0.7865	0.7894	0.7897	0.7892	0.7838	0.7896	0.7890
	MF pre-training	0.7762	0.7846	0.7845	0.7904	0.7902	0.7910	0.7892	**0.7952**	0.7934
	Rel. diff	-0.94%	-0.34%	-0.25%	0.13%	0.06%	0.23%	0.69%	0.71%	0.56%
Avazu	Scratch	0.7373	0.7351	0.7370	0.7350	0.7353	0.7351	0.7333	0.7318	0.7297
	MF pre-training	0.7307	0.7364	0.7301	0.7371	0.7387	0.7387	0.7399	**0.7407**	0.7394
	Rel. diff	-0.90%	0.18%	-0.94%	0.29%	0.46%	0.49%	0.90%	1.22%	1.33%

an embedding layer, MLP layers, and an output layer. The embeddings of all field are concatenated as the input of the feed-forward layers.

- The Wide&Deep (**W&D**) [3] model is useful for generic large-scale regression and classification problems with sparse inputs. It jointly trains a wide linear model which captures the shadow connections between the labels and features alongside a deep neural network which captures the deep and crossed connections between the features.
- The product-based neural network (**PNN**) [19] is basically a DNN but with product layers to capture interactive patterns between fields. There are two types of PNNs, with inner or outer product operations in the product layers. As the PNN with inner product (**IPNN**) is more computationally efficient than that with outer product, and their performances are close [19]. We only implement the IPNN model.
- The **Transformer** [27] model is proposed to solve the machine translation problem. In our settings, we treat each field as a word, and a record as sequence of words. As the fields are unordered, we drop the position embedding layer.
- The Field-Independent Transformer (**FI-Trans**) model is the proposed model. In result tables, **FI-Trans+LL** denotes the FI-Trans with last layer attention pooling (in Section 5.2.1), while **FI-Trans+AL** denotes FI-Trans with all layers attention pooling (in Section 5.2.2).

6.3 Training Settings

The hyper-parameters in masked-filed pre-training and FI-Trans include: the mask ratio (γ), the embedding size (emb), hidden size (d), head numbers (r), and stacked layer number (L). We test different combinations of hyper-parameters with $\gamma \in \{0.05, 0.1, 0.15, 0.2\}$, emb in $\{8, 16, 32, 64\}$, $d \in \{16, 32, 64, 128\}$, $r \in \{2, 4, 8\}$, and $L \in \{2, 3, 4, 5, 6\}$. We list the best settings for each dataset in Table 2. For comparison models, we use similar way to find the optimal hyper-parameters.

In pre-training procedure, we use the Adam optimizer [13] with learning rate of 1e-3, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The batch size is 256, the training epoch is 5. In the finetuning procedure, most parameters are the same with the pre-training procedure. We only present the differences. The learning rate is 5e-4, and the training epoch is 20. We add dropout [25] to alleviate overfitting, and the drop probability is 0.1. Moreover, we apply early stopping strategy, if the loss on development dataset is not decreasing, the training procedure is terminated.

In experiments, we compare two training modes: the training from *scratch* with only labeled data, and the masked-field pre-training (*MF pre-training*) in Section 4.

6.4 Results

In this section, we experimentally answer the following questions.

- Whether masked-field pre-training framework can help improve model performance on target tasks;
- What's the performance of FI-Trans model compared with the state-of-the-art methods;
- Whether the random features harm the performances of the proposed methods;

6.4.1 Impact of masked-field pre-training framework. Table 3 shows the experiment results trained with *scratch* model and *MF pre-training* mode. From the relative difference over scratch (Rel. diff rows in table), we observe that, for deep models, the masked field pre-training framework signifcantly improves the results. For example, in both driver and passenger UIP tasks, the MF pre-training gives around 5% improvement for DNN and 2.5% improvement for FI-Trans. In CTR prediction tasks, i.e., Avazu and Criteo datasets, the pre-training framework also shows modest and steady improvements. The improvement is because with only labeled data, the model tends to overfit the noise of sparse features and labels in the data. The pre-trained models learn general feature interaction patterns with unlabeled data. With abundant unlabeled data, the patterns are less likely to be affected by noise in the data. In the finetuning procedure, the model reuses the learned patterns instead of learning from scratch. As a result, the performance is significantly improved. For shadow models (e.g., the FM), the MF pre-training does not show positive effects. The difference is caused by the capacity of the feature interaction modules in models. The masked-field

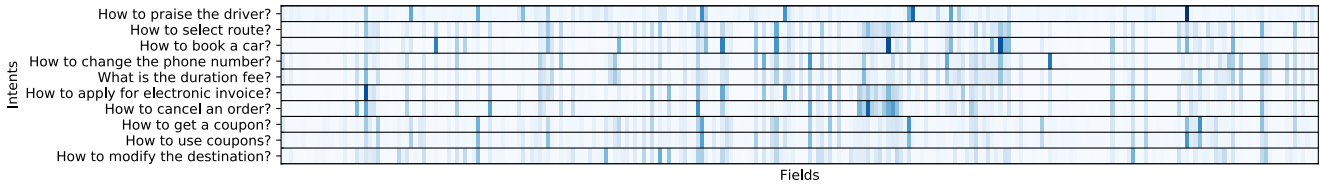
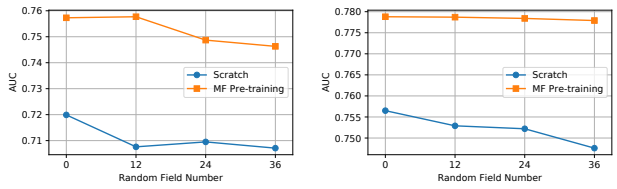


Figure 3: The attention probabilities over the inputs fields in the passenger UIP dataset. One line demonstrate the attention distribution over the 250 fields.



(a) DNN (b) FI-Trans

Figure 4: The change of AUC performance for the DNN and FI-Trans with respect to different random field numbers on passenger UIP dataset.

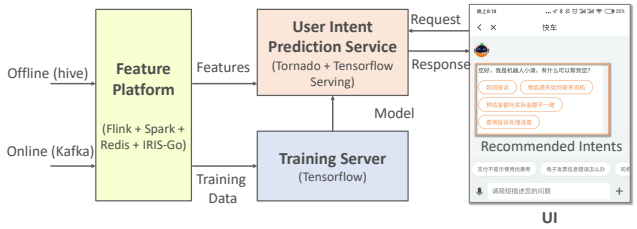


Figure 5: Illustration of the user intent prediction system.

pre-training method can learn feature interaction patterns from the unlabeled data. If the network have few layers, then its ability to learn feature interactions is limited. In summary, the proposed pre-training framework can be used to improve wide range of deep models for various prediction tasks.

6.4.2 Performance of different models. In this section, we compare the performance of models listed in Section 6.2. We can see from the Table 3 that in both UIP datasets, the proposed FI-Trans model significantly outperforms other methods. For example, for driver intent predictions, the AUC of FI-Trans model with all layers' attention pooling achieves 0.706, while the best AUC of the state-of-the-art methods is 0.687. But for the Criteo and Avazu datasets, the advantages of FI-Trans models over the compared methods are modest. The main reason is that the user intent prediction is a multi-label classification task, and each intent only related with partial features. In FI-Trans model, the output layer uses attention pooling to aggregate its relevant field representations. Figure 3 visualizes the attention probabilities of 10 intents over the 250 feature fields in the passenger UIP dataset. We observe that different intents attend to different field sets, which is consistent to our intuition. Thus, FI-Trans model is suitable to solve the multi-label classification problem. The Criteo and Avazu datasets only have one prediction target, so the advantage of FI-Trans over other methods is trivial.

6.4.3 Impact of random feature fields. In this experiment, we test how adding random feature fields affects the model performances. Here, *random* means the feature fields are randomly generated and have no correlation with other fields and labels. We conduct the experiment by adding $\{12, 24, 36\}$ random fields into the passenger UIP dataset, and test how the performance change with more noise fields added. We test four combinations of model and training mode, i.e., <DNN, scratch>, <DNN, MF pre-training>, <FI-Trans+LL, scratch> and <FI-Trans+LL, MF pre-training>. From Figure 4, we observe that the performance of most combinations significantly decline with more random fields added. However, the combination of our proposed methods <FI-Trans+LL, MF pre-training> is very robust. There are two reasons for that. First, with limited labeled data, the models may overfit the random noise. Pre-training helps relieve the problem. Second, FI-Trans+LL uses attention pooling to aggregate field representations, so the representation of random fields cannot affect the target tasks.

7 ONLINE PERFORMANCE

The proposed methods are tested in the user intent prediction system in DiDi. The architecture of the system is illustrated in Figure 5. The feature platform (left part) has two functions. First, the feature platform collects and mines features from both online data streams and historical data. Second, it provides a high concurrency API for querying features in the platform. The user intent prediction service (top middle) which handles requests from the APP, queries the API for features, predicts the users' intents with the proposed model, and return the results. The training server (bottom middle) trains the model, packs the model into a Tensorflow Serving docker, and deploys the model. The models are periodically trained and updated. When a user launches the chatting page of customer service, the client sends a request to the system and the system responses four intents with highest predicted scores for the user.

We compare the proposed methods with several baselines by conducting A/B tests with the online traffic. We use the CTR as evaluation metrics. As we have limited traffic and experiment time, we only compared the eight <model, training mode> combinations as listed in Table 4. In the A/B test, we equally divide the users into several buckets, and ensure each user belongs to the same bucket throughout the experiment. We conduct the A/B tests for successive 3 weeks during April 2020. We observe that the proposed <FI-Trans+LL, MF pre-training> significantly improves the CTR of

Table 4: CTR performance for UIP task with online A/B test

<Model, Training mode>	Passenger UIP	Driver UIP
<DNN, Scratch>	41.7%	36.8%
<DNN, MF Pre-training>	42.4%	38.0%
<Inner-PNN, Scratch>	42.8%	37.0%
<Inner-PNN, MF Pre-training>	42.7%	37.8%
<Transformer, Scratch>	40.1%	36.0%
<Transformer, MF Pre-training>	43.9%	37.7%
<FI-Trans+LL, Scratch>	44.5%	37.5%
<FI-Trans+LL, MF Pre-training>	**45.6%**	**38.2%**

the recommended intents. The higher CTR means that users can more efficiently get the answers to their questions and save users the trouble of asking for human customer service agents. Although the complex model increases the system latency by around 20 ms, it is within an acceptable range. We deployed the proposed methods in our system since April 2020.

8 CONCLUSIONS

In this paper, we study the problem of how to utilize the unlabeled data to tackle the data sparsity problem on the user intent prediction task. We propose a novel masked-field pre-training framework which learns feature interaction patterns from unlabeled data, then reuses these patterns for the target tasks. The framework can be used to improve various deep models. Along with the framework, we also propose a Field-Independent Transformer (FI-Trans) model. FI-Trans well fits the multi-label classification problem, as it aggregates the relevant field representations with the attention mechanism for each intent. In experiments, we demonstrate that the masked-field pre-training framework improves the performance of many deep models on user intent prediction datasets and several public datasets. Furthermore, the FI-trans model with pre-training significantly outperforms the state-of-the-art methods in the user intent prediction task. These comprehensive experiments also demonstrate that the proposed methods are effective and can be widely applied to various prediction problems.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proc. SIGKDD*. ACM, 785–794.

[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 *Proc. DLRS*. ACM, 7–10.

[4] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research* 12, Aug (2011), 2493–2537.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proc. RecSys* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, 191–198.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL*. 4171–4186.

[7] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc. ICML*. 513–520.

[8] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. 2011. Domain adaptation for object recognition: An unsupervised approach. In *Proc. ICCV*. IEEE, 999–1006.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proc. AAAI*. AAAI Press, 1725–1731.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proc. CVPR*. 770–778.

[11] Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint 1606.08415* (2016).

[12] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614* (2016).

[13] Diederik P Kingma and Jimmy Ba. 2014. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[14] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proc. CIKM*. 811–820.

[15] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *Proc. SIGKDD*. 305–314.

[16] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proc. WWW*. 689–698.

[17] Andrew Ng et al. 2011. Sparse autoencoder. *CS294A Lecture notes* 72, 2011 (2011), 1–19.

[18] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proc. EMNLP-IJCNLP*. 188–197.

[19] 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 *Proc. ICDM*. IEEE, 1149–1154.

[20] Alec Radford. 2018. Improving Language Understanding by Generative Pre-Training.

[21] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *Proc. ICML*. ACM, 759–766.

[22] Steffen Rendle. 2010. Factorization machines. In *Proc. ICDM*. IEEE, 995–1000.

[23] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting visual category models to new domains. In *Proc. ECCV*. Springer, 213–226.

[24] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proc. WWW*. 111–112.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proc. CIKM*. 1441–1450.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*. 5998–6008.

[28] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proc. WSDM*. 153–162.

[29] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: learning the weight of feature interactions via attention networks. In *Proc. AAAI*. AAAI Press, 3119–3125.

[30] Zhixian Yan, Lai Wei, Yunshan Lu, Zhongqiang Wu, and Bo Tao. 2017. You are what apps you use: Transfer Learning for Personalized Content and Ad Recommendation. In *Proc. RecSys*. ACM, 350–350.

[31] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).

[32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Proc. NeurIPS*. 3320–3328.

[33] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. 2018. Taskonomy: Disentangling task transfer learning. In *Proc. CVPR*. 3712–3722.

[34] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++ An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *Proc. SIGIR*. 957–960.

[35] Lili Zhao, Sinno Jialin Pan, Evan Wei Xiang, Erheng Zhong, Zhongqi Lu, and Qiang Yang. 2013. Active transfer learning for cross-system recommendation. In *Proc. AAAI*.

[36] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proc. AAAI*, Vol. 33. 5941–5948.

[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proc. SIGKDD*. 1059–1068.