Infer Implicit Contexts in Real-time Online-to-Offline Recommendation

Xichen Ding[1], Jie Tang[2], Tracy Liu[2], Cheng Xu[1], Yaping Zhang[1], Feng Shi[1], Qixia Jiang[1], Dan Shen[1]
[1] Koubei, Alibaba Group, Beijing, China
[2] Tsinghua University, Beijing, China
{xichen.dxc,haoze.xc,yaping.zyp,sam.sf,qixia.jqx,dan.sd}@alibaba-inc.com,{jietang,liuxiao}@tsinghua.edu.cn

ABSTRACT

Understanding users' context is essential for successful recommendations, especially for Online-to-Offline (O2O) recommendation, such as Yelp, Groupon, and Koubei[1]. Different from traditional recommendation where individual preference is mostly static, O2O recommendation should be dynamic to capture variation of users' purposes across time and location. However, precisely inferring users' *real-time contexts* information, especially those implicit ones, is extremely difficult, and it is a central challenge for O2O recommendation. In this paper, we propose a new approach, called Mixture Attentional Constrained Denoise AutoEncoder (*MACDAE*), to infer implicit contexts and consequently, to improve the quality of real-time O2O recommendation. In *MACDAE*, we first leverage the interaction among users, items, and explicit contexts to infer users' implicit contexts, then combine the learned implicit-context representation into an end-to-end model to make the recommendation. *MACDAE* works quite well in the real system. We conducted both offline and online evaluations of the proposed approach. Experiments on several real-world datasets (Yelp, Dianping, and Koubei) show our approach could achieve significant improvements over state-of-the-arts. Furthermore, online A/B test suggests a 2.9% increase for click-through rate and 5.6% improvement for conversion rate in real-world traffic. Our model has been deployed in the product of "Guess You Like" recommendation in Koubei.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Data mining*; • **Computing methodologies** → *Neural networks*;

KEYWORDS

online-to-offline recommendation, implicit context, attention

ACM Reference Format:
Xichen Ding, Jie Tang, Tracy Liu, Cheng Xu, Yaping Zhang, Feng Shi, Qixia Jiang, Dan Shen. 2019. Infer Implicit Contexts in Real-time Online-to-Offline Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330716

[1]www.koubei.com, Alibaba's local service company.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330716

1 INTRODUCTION

Online-to-Offline (O2O) services, which provide on-demand door-to-door services, have become prevalent in recent years. For example, local business service platforms such as Yelp, Groupon, Dianping, and Koubei, allow users to order products and services online, and receive the products and services offline. Koubei.com belongs to Alibaba's local service company. It serves tens of millions of users everyday by bringing local businesses online and providing various local services to customers, including recommendation of restaurants, local businesses, coupons, etc. Users can also place the order through Koubei mobile app in advance before they get to physical place (e.g., the restaurant) and then complete the transaction later. Compared with other e-commence recommendation, O2O recommendation has several unique characteristics. First, the objective of O2O recommendation is to provide customers real-time recommendation to satisfy their dynamic needs. This requires us to precisely capture customers' dynamic contexts, e.g. their location, time, and status (alone or with friends). Though certain contexts such as weekday, time, location, are relatively easy to obtain, other contexts, which are usually implicit, are difficult to infer. For example, users' purposes (e.g., whether they are looking for foods or activities), and status (alone or with friends). Figure 1 illustrates one example extracted from our data. At a weekday 8:00 am, users may be interested in coupons for breakfast close to their company; around 1:00 pm, they may want to search for deals in a dine-in restaurant; at 4:00 pm, users may want to find a place for afternoon tea. After going back home in the evening, they may switch to search for dinner options and for recreational activities after dinner such as spa and salon. The implicit contexts behind each scenario have multiple dimensions. For example, in the morning the implicit context is to have quick breakfast alone in a cheap restaurant; and for lunch it is to find a place in medium-high price range for group of colleagues. In the afternoon, it changes to find a quite place for discussion,e.g., one-on-one discussion with the manager. In the evening, it is to find restaurants good for family with kids.

Moreover, users' online behaviors at time t may be strongly influenced by other offline contexts. As the example in Figure 1 shows, when using Koubei's local business recommendation, users may be interested in items in both current context, representing their real-time dynamic needs, and future contexts, signaling periodical or personal preference. For instance, at 1:00 pm, a customer at work may be interested in both coupons for lunch now, and those for future activities (e.g. dinner for anniversary two days

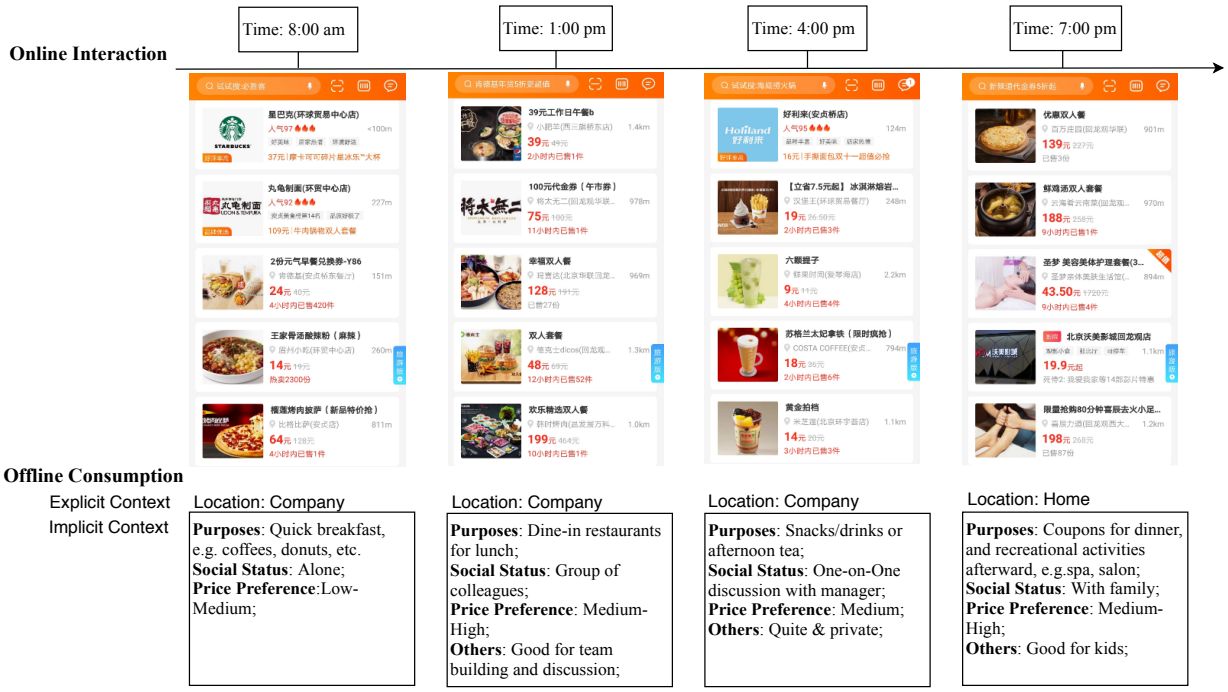

Figure 1: The illustration of implicit contexts in Online-to-Offline recommendation

later). In Koubei's "Guess You Like" recommendation, our analysis shows that 20.8% users click on variety of different service providers, including restaurants, gym, spa, etc.

In summary, the fundamental challenge for O2O recommendation, is how to infer users' contexts and how to make accurate recommendations based on the context. To tackle the problem, we first infer users' implicit multi-contexts from observational data, including the interactions among users, items, and the explicit contexts. This step is also called pre-training. Let latent variables $[X_1, X_2, X_3, ..., X_n]$ denote one real-time context, where each variable represents an attribute and their combination describes the context. For example, X_1 denotes users' purposes, X_2 denotes the social status, X_3 denotes price preference, etc. The implicit multi-contexts can be represented as C_i. We propose a novel approach, called Mixture Attentional Constrained Denoise AutoEncoder (*MACDAE*), to learn the implicit multi-contexts by using generative models. Assuming that there are K multiple implicit contexts, we use C_{ik} to denote the k_{th} contextual component we want to infer from the observational data. We further combine the learned implicit context representation with original input and feed them to downstream supervised learning models, which learn score on the $< U, I, C_e, C_i >$ (User, Item, Explicit Contexts, Implicit Contexts) tuple. We compare several generative models in the pre-training stage to infer implicit context, including Denoise AutoEncoder (DAE) [19], Variational AutoEncoder(VAE) [12], and find our proposed model *MACDAE* achieves the best performance for inferring implicit contexts. We adopt multi-head structure to represent multiple contextual components, in which each head represents one latent contextual component. Furthermore, the importance of each component is learned by attention mechanism. To avoid the problem that different heads learn identical contextual representation, we further apply constraints on the objective function of our generative models.

To summarize, our contributions include:

- **Implicit Context Modeling**: We take the first attempt to infer users' implicit context from observational data and model implicit multi-contexts in the Online-to-Offline (O2O) recommendation.
- **Context-based Recommendation**: Based on the learned implicit context representations, we present an effective recommendation model using multi-head attentions.
- **High Performance in both Offline and Online Evaluations.** We conduct both offline and online evaluations to the proposed approach. Experiments on several real-world datasets show our approach achieves significant improvements over state-of-the-arts. Online A/B test shows 2.9% lift for click-through rate and 5.6% lift for conversion rate in real-world traffic. Our model has been deployed in the product of "Guess You Like" recommendation in Koubei.

The rest of the paper is organized as follows: Section 2 provides an overview of our context-based recommendation system. Section 3 describes details of our proposed model and Section 4 contains results from offline experiments, online A/B test and analysis. We discuss related work in Section 5 and conclude in Section 6. Additional information for reproducibility is provided in supplements.

2 SYSTEM OVERVIEW

Figure 2 presents the overview of our context-based recommendation system. Users submit a page view (PV) request to the recommendation system. The system first query a set of candidates from

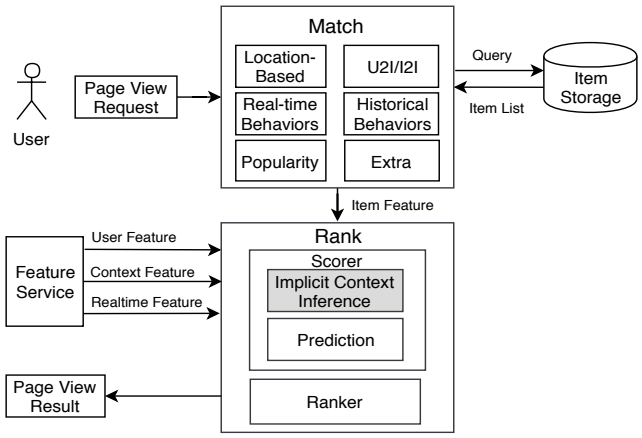

Figure 2: System Overview of Context-Based Recommendation in Koubei

storage. There are multiple match strategies, including location-based (retrieve all the items nearby users' current location or objective location), U2I/I2I (user-to-item and item-to-item and other relation based), users' real-time behaviors and historical behaviors, popularity based and others. Second, in the rank step, feature service generates features of users, explicit contexts, and real-time features. These features and item features are fed to the scorer. Simultaneously, users' implicit contexts are inferred from the interaction among users, items and explicit contexts. And in the prediction step, our model predicts a score for each candidate. Finally, candidates are sorted by prediction scores in the ranker. In the following section, we will focus our discussion on the scorer part of the system, especially the implicit context inference step.

3 PROPOSED MODEL

3.1 Model Intuition

There are several approaches to model the implicit multi-contexts in Online-to-Offline recommendation. One straightforward approach is to enumerate all the possible combination of factors that influence users' behaviors, such as weekday, time interval, location, social status and purposes. This approach works when the number of combinations is small, such as finding restaurants for dinner (purpose) in the evening (time interval) near home (location) which is good for family members including kids (social status). However, in practice, there are often too many scenarios that the number of combinations will explode with the growing number of factors.

This leads us to another approach, i.e., inferring users' implicit context as hidden state representation from the observational data, including user, item and explicit context. In addition, the hidden state representation is not a single static one and should reflect the multi-contexts characteristic in the complex Online-to-Offline recommendation. We will discuss several instances of this approach in the model description section.

3.2 Problem Formulation

We formulate our implicit multi-contexts modeling problem for recommendation as follows: Let C_i denote the implicit multi-contexts representation. Basically, we want to infer implicit contexts from the three-way interaction of tuple $< U, I, C_e >$ (User, Item, Explicit Context). Assuming that there are K different contextual components in the current implicit context. We use C_{ik} to denote the k_{th} contextual component we want to infer, which is a d_k dimension vector. The final implicit contextual representation is:

$$C_i = \sum_K \mu_k C_{ik} \tag{1}$$

$$C_{ik} = g_k(U, I, C_e) \tag{2}$$

$$y_{ui} = f(U, I, C_e, C_i) \tag{3}$$

The μ_k denotes the weight importance of the k_{th} component, with $\sum_K \mu_k = 1$. The function $g_k(.)$ denotes the latent representation function. And the complete recommendation model predicts a score y_{ui} for user and each item in the candidates.

We apply different generative models to instantiate $g_k(.)$, including Denoise AutoEncoder (DAE) [19], Variational AutoEncoder (VAE) [12], and our proposed model Mixture Attentional Constrained Denoise AutoEncoder (MACDAE). In Section 4, we will compare these models in details.

3.3 Model Description

For recommendation with implicit feedback, given a set of users U and a set of items I, we aim to predict a score y_{ui} for each pair of user and item in the list of candidates and then recommend the ranked list of items I_j to user U_i. To model the implicit multi-contexts in recommendation, we propose a unified framework, which consists of two steps: First, we infer implicit context representation C_i from interaction data of $< U, I, C_e >$ (User, Item, Explicit Context) in the pre-training task. Second, we include the extracted implicit context representation C_i as additional features and combine with original input and predict the score on the tuple $< U, I, C_e, C_i >$ (User, Item, Explicit Context, Implicit Context).

Implicit Context Inference in Pre-training. In the pre-training stage, the pre-training dataset comes from positive user-item interaction data, such as clicks and buys. We want to find some latent patterns of implicit context from these observational data. Pre-training method is widely used in many deep-learning tasks such as NLP and Computer Vision. In NLP, language models are pre-trained over large corpus to learn good embedding representation of words and sentences [7]. In computer vision, researchers find it's beneficial to pre-train image with large dataset and fine-tune the last few layers in application specific tasks [6]. There are two main approaches to apply the pre-trained representation to the following supervised tasks: feature-based approach and fine-tuning approach [7]. Feature-based approach will include the representation as additional features and fine-tuning approach tunes the same model architecture in the downstream supervised learning tasks.

Inspired by previous work, we also use pre-training to learn implicit multi-contexts representation from positive user item interaction data. In our model, the implicit context is inferred from (User, Item and Explicit Context) tuple. We denote x as the input to pre-training model architecture, $g(x)$ as learned implicit context representation. The input x consists of user embedding e_u, item embedding e_i, their two-way interaction $e_u \circ e_i$ represented by

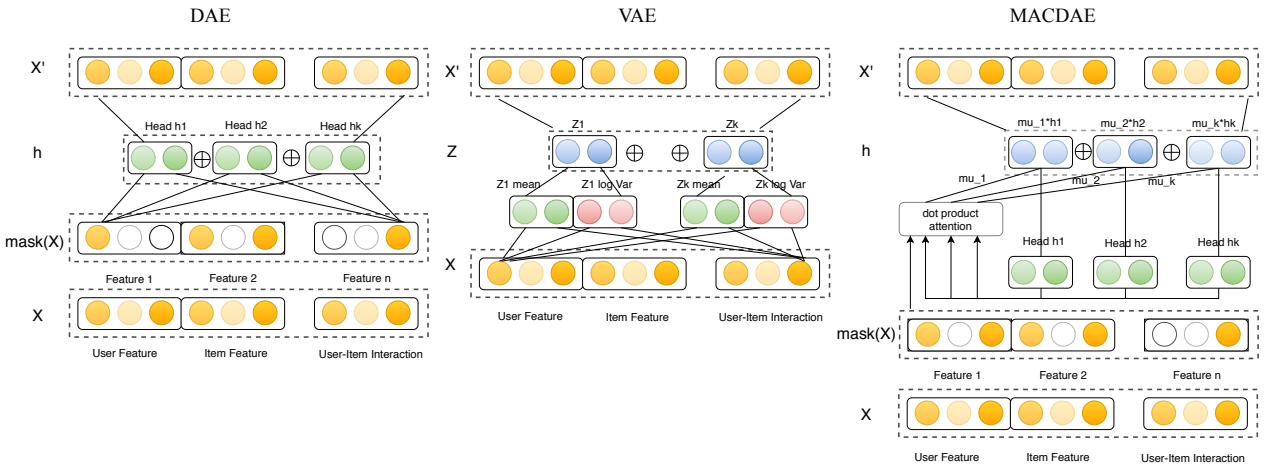

Figure 3: The model architecture of DAE, VAE and MACDAE.
The DAE, VAE, MACDAE models are implemented as multi-head version

their element-wise product, and other side information of users and items e_{side}.

$$x = concat(e_u, e_i, e_u \circ e_i, e_{side}) \tag{4}$$

In the downstream supervised learning tasks, we adopt different strategies for different datasets, including feature-based approach and fine-tuning approach. For feature-based approach, we include the pre-trained representation $g(x)$ in the deep model architecture, initialize $g(x)$ with parameters from pre-training task and freeze them during supervised training. For fine-tuning approach we allow the fine tuning of pre-trained parameters of $g(x)$.

Multi-Head DAE/VAE. Usually, the implicit contexts have multiple components, which cannot be described by a single static representation. We propose to use a multi-head structure of representation $h = g(x)$ to represent the multiple implicit contextual components, where x is the input and $g(x)$ is the learning function. The implicit contextual representation h is the combination of multiple contextual components. As illustrated in Figure 3, the hidden layer h is the concatenation of K multiple heads h_k, each representing one contextual component C_{ik}. Each head (hidden state) h_k only captures partial contextual information from subspaces of the original input x. We denote K as the assumed number of contextual components, and use h_k to represent the k_{th} latent hidden contextual component C_{ik}. For the generative model, Denoise AutoEncoder(DAE) [19] and Variational AutoEncoder(VAE) [12], we simply concatenate multiple heads as the final implicit context representation.

The classic Auto-Encoder[3] model is a one-hidden layer neural network, which first maps an input vector x to a latent representation h and then reconstruct the input as $x^{'}$. The objective is to minimize the squared loss between the original input x and the reconstruction $x^{'}$. The Denoise AutoEncoder[19] further masks out partial information of input with probability p and gets the corrupted version of input as $\tilde{x}$. $\tilde{x} = mask(x)$. Then the network aims to encode the corrupted input $\tilde{x}$ as hidden state and to reconstruct the original input x from the hidden state. As illustrated in Figure 3, for the multi-head version of DAE, the hidden state h is concatenation of K components. N denotes the total number of examples.

$$h = h_1 \oplus h_2 \oplus ... \oplus h_K, \ k \in K \tag{5}$$

$$h_k = \sigma(W_k \tilde{x} + b_k), \ k \in K \tag{6}$$

$$x^{'} = \sigma(W^{'} h + b^{'}) \tag{7}$$

$$L_{reconstruct} = \frac{1}{N} \sum_{N} ||x - x^{'}||^2 \tag{8}$$

Variational AutoEncoder (VAE) [12] is another very important generative deep learning model, which assumes that the data distribution of input x is controlled by a set of latent random variables z.

$$p(x) = \int p_\theta(x|z)p(z)dz \tag{9}$$

To optimize $p(x)$, VAE assumes that the latent variable comes from a simple distribution, usually $N(0, I)$ and the conditional probability $p_\theta(x|z)$ is also a Gaussian distribution $N(f(z;\theta), \sigma^2 \times I)$ with mean $f(z;\theta)$ and covariance $\sigma^2 \times I$. An encoder (MLP) learns the mean and variance of data distribution and then a decoder reconstructs the input. In the encoder part, since the posterior of $p_\theta(z|x)$ is intractable, VAE use $q_\phi(z|x)$ to approximate the original $p_\theta(z|x)$. Consequently, the overall structure of VAE becomes an encoder $q_\phi(z|x)$ and a decoder $p_\theta(x|z)$. The objective function also consists of two parts: the reconstruction part and the Kullback-Leibler divergence D_{KL} between $q_\phi(z|x)$ and $p_\theta(z)$. As shown in Figure 3, we also concatenate the state representation of each head z_k to get the hidden representation $z = z_1 \oplus z_2 \oplus ... \oplus z_K$, and reconstructs the original input from z. For the following supervised training step, we feed the concatenation of mean state vector of each head $\bar{z} = \bar{z_1} \oplus \bar{z_2} \oplus ... \oplus \bar{z_K}$ to the prediction model.

$$z = z_1 \oplus z_2 \oplus ... \oplus z_K, \ k \in K \tag{10}$$

$$z_k \sim N(\mu_k(x), \Sigma_k(x)), \ k \in K \tag{11}$$

$$x^{'} = \sigma(W^{'}z + b^{'}) \tag{12}$$

Mixture Attentional Constrained Denoise AutoEncoder. We propose a new model: Mixture Attentional Constrained Denoise AutoEncoder (MACDAE) (Cf. Figure 3) to infer the multiple implicit contexts. Different from the standard Denoise AutoEncoder(DAE) model, in MACDAE, the hidden layer h is the concatenation of K weighted multiple heads h_k, with each representing one contextual component C_{ik}. The basic idea is that different implicit contextual components contribute differently to the final representation. The multi-head DAE model can be considered as a special case of the weighted concatenation where different components contribute equal weights. The weight of each component in MACDAE can be learned using the attention mechanism.

$$h = \mu_1 h_1 \oplus \mu_2 h_2 \oplus ... \oplus \mu_K h_K,\ k \in K \tag{13}$$

The final implicit multi-contexts representation h is the concatenation of weighted heads h_k. The weight μ_k is learned by an attentional function, which maps the uncorrupted version of input x to the k_{th} hidden representation h_k. In the implementation, we use the dot-product (multiplicative) attention. The (Q, K, V) tuple of the attention function is that: the query Q is represented by $(W_a x)^T$ and the keys and values are multiple hidden components h_k. The original input x has the dimension of d_m and after multiplication with matrix W_a(shape $[d_k, d_m]$) the dimension of $W_a x$ becomes d_k, which has the same dimension as that of the hidden component h_k. The dimension is also equal to total hidden state dimension d_h divided by number of heads K, $d_k = d_h/K$. The multiple hidden states h_k are packed into matrix H(shape $[K, d_k]$) and the attention function becomes as below. The reconstruction layer is the same as the standard DAE model.

$$Q = (W_a x)^T \tag{14}$$

$$[\mu_1, \mu_2, ..., \mu_K] = softmax(QH^T), k \in K \tag{15}$$

Constraint Definition. The loss function of the proposed mixture attentional constrained model uses the squared loss between the original input x and the reconstruction $\tilde{x}^{'}$, which is similar to the standard DAE. As we stated previously, one downside of multi-head attention training is the homogeneity between each pair of heads, which suggests that they tend to learn identical representations given the same input x and learning steps. Empirically, the average cosine similarity between multiple heads can be as high as 0.788, as illustrated in Figure 7, from experiments on the Koubei dataset.

Another idea is to apply constraint on the cosine similarity between each pair of heads of multiple contextual components. Denote h_i and h_j as the i_{th} and j_{th} heads (hidden representation), and the constraint is formulated as:

$$\cos(h_i, h_j) \leq \varepsilon, \forall i, j \in K \tag{16}$$

where ε is a hyperparameter, which is within the range $\varepsilon < 1$. Since two identical vectors with 0 degree angel has cosine similarity as $\cos 0 = 1$. ε denotes the maximum cosine similarity we set on each pair of K components, e.g. $\varepsilon = 0.75$. We also apply other distance measure, e.g. the euclidean distance between h_i and h_j, and find that using cosine distance outperforms the euclidean distance in our experiments.

Now the problem becomes a constrained optimization problem. We want to minimize the reconstruction loss $L_{reconstruct}$ subject to the total C_K^2 constraints on multiple heads.

$$min\ L = L_{reconstruct} \tag{17}$$

$$s.t.\ cos(h_i, h_j) - \varepsilon \leq 0, \forall i, j \in K \tag{18}$$

To apply the constraint on the parameters of neural networks, we transform the formulation by adding penalty term to the original objective function and get the final objective function L_{new}. λ is a hyper-parameter of the penalty cost we set on the constraint term of similarity.

$$min\ L_{new} = L_{reconstruct} + \sum_{i,j \in K} \lambda(cos(h_i, h_j) - \varepsilon) \tag{19}$$

User Sequential Behavior Modeling. For real-world recommendation, there are plenty of user behavior information in logs, such as query, click, purchase, and etc. In Koubei's recommendation, we use a RNN network to model users' sequential behaviors data, with attention mechanism applied to the hidden state of each time step h_j. We use J to denote the total number of steps of users' sequential behaviors. C_e denotes the explicit contextual information when user's certain behavior actually happens. Let's denote a_{ij} as attentional weights that candidate item I_i, denoted as x_{item_i}, places on j_{th} users' sequential behavior h_j, which is conditional on the context information C_e. The final representation of user's sequential behavior given current item I_i, denoted by x_{attn_i}, is the weighted sum of the hidden state of RNN as $x_{attn_i} = \sum_J a_{ij} \times h_j$. The fixed-length encoding vector x_{attn_i} of users' sequential behavior data under current context C_e is also combined with the original input x, implicit context modeling $g(x)$, and fed to the first layer input of the deep part in supervised learning model $x_0 = concat(x, g(x), x_{attn_i})$.

$$a_{ij} = f_{attn}(x_{item_i}, h_j, C_e) \tag{20}$$

Learning and Updating of the Two-stage Framework. Our proposed framework consists of pre-training stage and downstream supervised learning stage. We denote N as the duration of pre-training dataset, e.g. N consecutive days of positive user-item interaction data. M denotes the duration of training dataset, including both impressions and clicks/buys data. The advantage of unsupervised pre-training is that it allows much larger pre-training dataset than the supervised learning dataset, which means $N >> M$. The pre-training model is updated on a weekly basis. It does not need to be updated frequently or by online learning, which saves abundant of computing resources. As for the downstream supervised learning task, since fine-tuning and feature-based approaches are usually faster than learning everything from scratch, we are updating the prediction model once a day after midnight.

4 EXPERIMENT

We conduct both offline and online evaluations to the proposed approach. In this section, we first introduce the offline evaluation on three different datasets, and then use the online A/B test to evaluate our proposed model.

Table 1: Statistics of Yelp, Dianping, Koubei Datasets

Dataset	User	Item	Interaction
Yelp	24.8K	130.7K	1.2M
Dianping	8.8K	14.1K	121.5K
Koubei*	4.0M	1.1M	43.4M

*Koubei offline dataset is randomly sampled from larger production dataset.

4.1 Experiments Setup

Dataset. We choose three datasets to evaluate our models: Yelp Business dataset[2], Dianping dataset[3][15] and Koubei dataset. Table 1 presents the statistics for each dataset. M denotes million and K denotes thousand.

- **Yelp Dataset** The Yelp dataset (Yelp Dataset Challenge) contains users' review data. It is publicly available and widely used in top-N recommendation evaluation. User profile and business attribute information are also provided. Following [10, 14], we convert the explicit review data to implicit feedback data. Reviews with four or five stars are rated as 1 and the others are rated as 0. Detailed train/test split methods are provided in supplements.
- **Dianping Dataset** Dianping.com is one of the largest business review websites. The dataset we use contains customer review data (similar to Yelp Dataset). We also convert review to implicit feedback and prepare the dataset the same way as Yelp dataset.
- **Koubei Dataset** Koubei.com belongs to the local service company of Alibaba. The goal of the recommendation is to predict user's click-through rate (CTR) and conversion rate (CVR) of candidate items. To evaluate our implicit contextual representation models on this large real-world commercial dataset, we collect 30 consecutive days of positive instance (user click) for pre-training and use 14 days of customer logs data, both positive (user click) and negative (impression without click), for training, and use the data in the next day for testing. For offline evaluation, we randomly sample a subset of a larger production dataset of CTR.

Evaluation Metrics. For the first two datasets, i.e., Yelp and Dianping, we use Normalized Discounted Cumulative Gain at rank K ($NDCG@K$) as metrics for model evaluation, which is widely used in top-N recommendation with implicit feedback [10]. For negative sampling of Yelp and Dianping, we randomly sample 50 negative samples for 1 positive sample and then rank the list. The ranked list is evaluated at $K = 5, 10$ and the final results are averaged across all users. For the Koubei dataset, we have more information, so we use the Area Under the ROC curve(AUC) of the user's CTR as the metric [14], which measures the probability that the model ranks randomly sampled positive instance higher than negative instance.

4.2 Comparison Methods

We conduct pre-training and compare three models, including DAE, VAE and our proposed *MACDAE*. After we get the representation of implicit context, we adopt different strategies for different datasets. For Koubei dataset, we combine the latent features $h = g(x)$ (hidden state) with the original input x as additional features, and feed $(x + g(x))$ to the baseline model. Then the model is trained with the parameters of $g(x)$ frozen. As for Yelp and Dianping dataset, we just use the latent features $g(x)$ directly as input and feed to the baseline model. Since the latent contextual features of Dianping and Yelp are concatenation of embeddings, the final model is fine-tuned and the parameters of the generative model $g(x)$ is not frozen. We choose the Wide&Deep[4] model as the baseline model, and compare other models below.

- **Wide&Deep (BaseModel)** : The baseline model Wide&Deep[4] combines the wide linear model with deep neural networks. The outputs of the wide part and the deep part are combined using weighted sum and then the model applies a sigmoid function to the final output.
- **DeepFM**: DeepFM[8] model combines two components, Deep Neural Network component and Factorization Machine (FM) component. The FM component models the second-order interaction, compared to the linear part in Wide&Deep.
- **NFM**: Neural Factorization Machines (NFM)[9] model combines the linearity of Factorization Machine (FM) to model second-order interaction with the non-linearity of neural network to model higher-order feature interactions. A bi-interaction pooling component, which models the interaction among sparse features, is used as hidden layer in the deep structure.
- **BaseModel+DAE**: The pre-training model uses a multi-head version of Denoise AutoEncoder(DAE)[19]. And the hidden state h is a concatenation of K different heads (hidden state) as illustrated in Figure 3. Finally, we feed the hidden state learned by DAE to the baseline (Wide&Deep) model.
- **BaseModel+VAE**: The pre-training model uses a multi-head version of Variational AutoEncoder(VAE)[12] as shown in Figure 3. For the downstream training step, we feed the concatenation of mean state vector of each head $\bar{z_k}$ to the baseline model.
- **BaseModel+MACDAE**: The pre-training model uses our proposed Mixture Attentional Constrained Denoise AutoEncoder(MACDAE) as the objective. We use the weighted concatenate of multiple heads as hidden state $h = \mu_1 h_1 \oplus \mu_2 h_2 \oplus \ldots \oplus \mu_K h_K$ and feed it to the baseline model. For comparison purpose, we implement MACDAE model in three configurations with different multi-head number K, $K = [4, 8, 16]$.

4.3 Results from Pre-training Dataset

We have compared DAE, VAE and MACDAE models on three pre-training datasets. The illustration of latent representations learned from the pre-training datasets are presented in Figure 4. The illustration is shown based on 8K samples from Yelp dataset, 8K samples from Dianping dataset and 25.6K samples from Koubei dataset. We use t-SNE[17] to map the latent representations h into 2D space. We also cluster the latent representations and set the total cluster number of Yelp and Dianping to 8 and Koubei to 15. Our illustration

[2]https://www.kaggle.com/yelp-dataset/yelp-dataset
[3]http://shichuan.org/HIN_dataset.html

Table 2: Results on evaluation of three datasets."*" indicates best performing model and "#" indicates best baseline model.

Model	Yelp		Dianping		Koubei
	NDCG@5	NDCG@10	NDCG@5	NDCG@10	AUC
Base(Wide&Deep)	0.3774	0.4272	0.4301	0.4691	0.6754
DeepFM	#0.3893	#0.4414	#0.4564	0.4854	0.6660
NFM	0.3813	0.4242	0.4527	#0.4996	#0.6881
Base+DAE	0.4159	0.4639	0.4507	0.4954	0.6810
Base+VAE	0.4192	0.4661	0.4517	0.4985	0.6790
Base+MACDAE(K=4)	*0.4410	*0.4886	0.4569	0.5024	0.6902
Base+MACDAE(K=8)	0.4136	0.4650	*0.4614	*0.5050	0.6936
Base+MACDAE(K=16)	0.4058	0.4521	0.4470	0.4960	*0.6954

shows that the VAE extracts the latent representations as high-variance compared to DAE and MACDAE. The detailed analysis of data distribution will be discussed in the following section.

4.4 Offline Model Evaluation Results

We have conducted extensive experiments on the Yelp, Dianping and Koubei datasets. The Yelp and Dianping datasets are evaluated by NDCG@5 and NDCG@10. The Koubei Dataset of click through rate(CTR) is evaluated by AUC performance. Table 2 reports results of model comparison. First, by comparing three fine-tuned baseline models with features from pre-trained models (Base+DAE, Base+VAE, Base+MACDAE) to baseline models without any pre-training, we find that pre-training step and implicit context representations are very effective. For Yelp dataset, compared to Wide&Deep model without pretraining, the pre-training on DAE, VAE and our proposed MACDAE models gain 3.9%, 4.2%, 6.4% absolute improvement on NDCG@5 and 3.7%, 3.9%, 6.1% on NDCG@10 respectively. For Koubei CTR dataset, the pre-training step also achieves 0.6%, 0.4%, 2.0% absolute improvement on AUC score compared to the baseline model without pretraining. The experiment results are consistent with our hypothesis that adding implicit multi-contexts representations would be helpful for improving the overall performance. Secondly, by comparing different pre-training objectives, our proposed model MACDAE outperforms others, multi-head versions of DAE and VAE. Since multi-head DAE simply uses concatenation of multiple heads, in which all heads contribute equal weights, it is a special case of the general model MACDAE. The attentional weight of multiple components are also effective compared to simple concatenation with equal weights. Moreover, the hyperparameter of multi-head number K also influences the performance, and we will discuss the effect of multi-head number K in the following section. Thirdly, the results show the effectiveness of adding two-way interaction of user and item features to the input x of pre-training model MACDAE. Comparing our proposed model Base+MACDAE with DeepFM/NFM, even if the baseline model (Wide&Deep) alone performs worse than the DeepFM/NFM due to the loss of high-order feature interaction, our proposed Base+MACDAE still beats these two models. The reason is that the input to the pre-training model MACDAE also contains the features of the two-way interaction between user and item embedding $e_u \circ e_i$, which is also fed to the baseline model.

Table 3: Online A/B Test Results of CTR/CVR in Koubei Recommendation.

Days	CTR (%Lift)	CVR (%Lift)
D1	+3.5%	+7.0%
D2	+1.8%	+6.4%
D3	+2.6%	+7.0%
D4	+2.5%	+6.3%
D5	+3.6%	+2.6%
D6	+2.9%	+4.7%
D7	+3.4%	+5.3%
Average	+2.9%	+5.6%

4.5 Online A/B Test Results

After comparing the models on Koubei offline dataset, we conduct online A/B test in real-world traffic and compare our proposed model (Wide&Deep model with MACDAE pre-training) with the baseline model (Wide&Deep model without pre-training). The daily online results report an average 2.9% lift on click-through rate(CTR) and 5.6% lift on conversion-rate(CVR) in Koubei recommendation. The detailed results are reported in table 3. Online testing indicates that model Base+MACDAE achieves great improvements over the existing online models. More importantly, we have already deployed our new model in production and served tens of millions of customers in Koubei's "Guess You Like" recommendation section everyday.

4.6 Results from Data Mining Perspective of Context in Recommendation

4.6.1 ***Distribution of Latent Representations Learned by Generative Models.*** We conduct experiments to answer the question: What is the data distribution of original input and the latent representations learned by generative models like in recommendation dataset? We compare examples of Koubei dataset with examples of Yelp dataset. Koubei dataset consists of dense features and one-hot encoded sparse features of users and items, and Yelp dataset consists of embedding features learned by user and item id lookup. The results are presented in Figure 5. We calculate the mean $\bar{x}$ and variance σ^2 of feature values within each vector of example and the results are then averaged across different samples. We discuss several interesting findings here. First, for Koubei

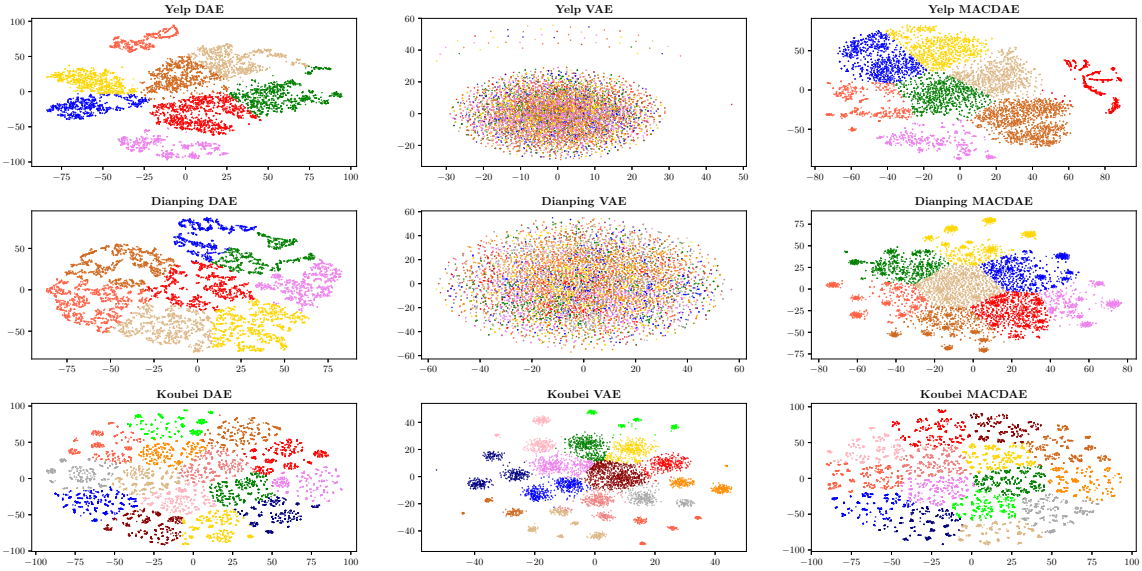

Figure 4: The illustration of the latent hidden states of implicit contexts extracted by models

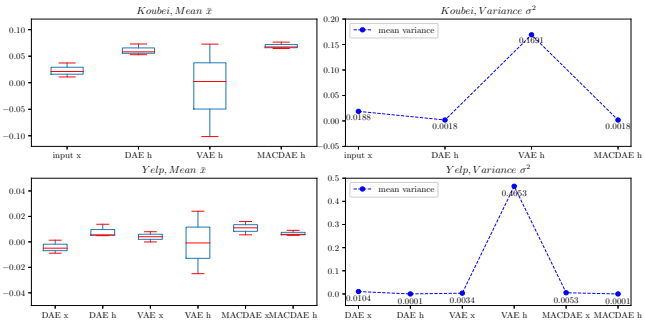

Figure 5: Mean and Variance Distribution of Original Input x and Hidden State h Learned by DAE, VAE, MACDAE

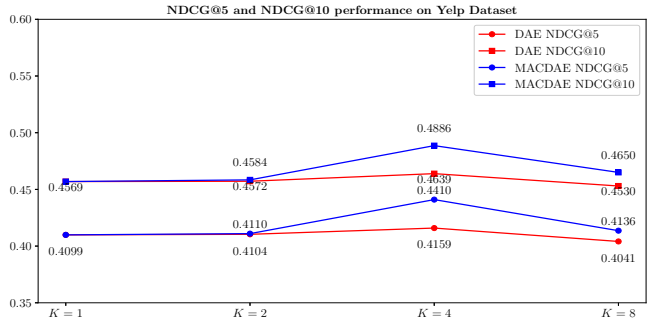

Figure 6: The performace of NDCG@5 and NDCG@10 on Yelp dataset with different multi-head number K

dataset, comparing the same input vector x with hidden state h learned by different models, DAE model transforms the original input x to hidden state h by increasing the mean of the features and reducing the variance. In contrast, VAE model transforms the input x by reducing the mean and increasing the variance. This is aligned with the illustration of a round-shaped embedding of VAE model trained on Yelp dataset in Figure 4. MACDAE is similar to DAE and further reduces the variance in hidden state representation. Second, by comparing VAE model on two different datasets, we find that for Koubei dataset, the hidden state h learned by VAE has $avg(\bar{h}) = 0.0023, min(\bar{h}) = -0.1015, max(\bar{h}) = 0.0728, \sigma^2 = 0.1691$, which shows that the hidden state is not centered around $\bar{h} = 0.0$. For the Yelp dataset which is learned from randomly initialized embedding, the hidden state h has $avg(\bar{h}) = -0.0008, min(\bar{h}) = -0.0250, max(\bar{h}) = 0.0241, \sigma^2 = 0.4653$, which is centered around $\bar{h} = 0.0$. Our conjecture is that the original input vector of Koubei dataset consist of one-hot encoded features and dense features, which is very sparse and not normally distributed.

4.6.2 ***Effect of Multi-Head Number K.*** We empirically evaluate the impact of parameter multi-head number K, which represents the assumed hidden state number of the implicit multi-contexts. To highlight the influence of parameter K, we choose the Yelp dataset and compare multi-head DAE model with our proposed Mixture Attentional Constrained Denoise AutoEncoder (MACDAE) model on different levels of K, $K = [1, 2, 4, 8]$. The performance of evaluation metrics NDCG@5 and NDCG@10 are shown in Figure 6. The hidden state dimension in Yelp dataset is [256], and K=4 achieves the best performance of both DAE and MACDAE. For K=1, both DAE and MACDAE become the same vanilla DAE model with single hidden state. Furthermore, increasing head number K does not always improve the metrics.

5 RELATED WORK

Traditional context-aware recommendation systems use context information as pre-filtering, post-filtering conditions and features in the contextual modeling [1]. Methods such as Wide&Deep[4],

DeepFM[8] and NFM [9] only predict score on the $< U, I, C_e >$ tuple(User, Item, Explicit Context) and neglect the implicit context. Latent factor models including matrix factorization [13] and tensor factorization [5], learn hidden factors of user, item, context and their interaction, but only model single latent representation and neglect the characteristic of multi-contexts in O2O recommendation.

Most of recent research on context-aware recommendation systems (CARS) focus on extension of Collaborative Filtering methods [21], Matrix Factorization or Tensor Factorization methods [5], and Latent Factor models. For example, HyPLSA[2] is high order factorization method that learns interaction among user, item and context. Traditional latent factor models like LDA learns latent subspace from data, but it is time-consuming to train and can't be easily integrated with downstream tasks, such as deep neural networks. Recently pre-training is widely used in NLP, e.g., ELMo[16] and BERT[7] models are pre-trained on large corpus and achieve great success in many tasks, e.g., question answering, etc. The building block Transformer [18] uses multi-head attention to jointly attend to information from different representation subspaces, which has advantages over single head. We adopt similar multi-head structure as Transformer to represent multiple implicit contextual components while Transformer attend on tokens in sentences.

Attentive Contextual Denoising Autoencoder(ACDA) [11] model extends idea of CDAE [20] and uses a single attentional layer to apply a weighted arithmetic mean on features of the hidden layer representation, which also considers the explicit contextual features. The key difference between our proposed MACDAE and methods mentioned above is that: CDAE and ACDA aim to learn latent features that reflect user's preferences over all N candidate items at the same time and only consider the explicit contextual features. MACDAE infers the implicit contextual representation from the interaction between each pair of user, item and explicit context.

6 CONCLUSIONS

In this paper, we propose a unified framework to model users' implicit multi-contexts in Online-to-Offline (O2O) recommendation. To infer the implicit contexts from observational data, we compare multiple generative models with our proposed Mixture Attentional Constrained Denoise AutoEncoder (MACDAE) model. Experiments show that *MACDAE* significantly outperforms several baseline models. Additionally, we conduct extensive analysis on the actual data distribution of latent representations and gain insights on properties of different generative models. Online A/B test reports great improvements on click-through rate (CTR) and conversion rate (CVR), and we have deployed our model online in Koubei's "Guess You Like" recommendation and served tens of millions of users everyday.

7 ACKNOWLEDGEMENTS

We thank algorithm-engineering teams of Koubei and Ant Financial for their support on recommendation platform, feature service, distributed machine learning platform and online model serving system, which enable the successful deployment in production.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2008. Context-aware Recommender Systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)*. ACM, New York, NY, USA, 335–336. https://doi.org/10.1145/1454008.1454068

[2] Robert W. Alan Said. 2009. A hybrid PLSA approach for warmer cold start in folksonomy recommendation. *Proceedings of the International Conference on Recommender Systems* (2009), 87–90.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2006. Greedy Layer-wise Training of Deep Networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems (NIPS'06)*. MIT Press, Cambridge, MA, USA, 153–160. http://dl.acm.org/citation.cfm?id=2976456.2976476

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* abs/1606.07792 (2016). arXiv:1606.07792 http://arxiv.org/abs/1606.07792

[5] Tiago Cunha, Carlos Soares, and André C.P.L.F. Carvalho. 2017. Metalearning for Context-aware Filtering: Selection of Tensor Factorization Algorithms. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 14–22. https://doi.org/10.1145/3109859.3109899

[6] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. 2009. Imagenet: A large-scale hierarchical image database. In *In CVPR*.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *CoRR* abs/1703.04247 (2017). arXiv:1703.04247 http://arxiv.org/abs/1703.04247

[9] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 355–364. https://doi.org/10.1145/3077136.3080777

[10] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path Based Context for Top- N Recommendation with A Neural Co-Attention Model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM, New York, NY, USA, 1531–1540. https://doi.org/10.1145/3219819.3219965

[11] Yogesh Jhamb, Travis Ebesu, and Yi Fang. 2018. Attentive Contextual Denoising Autoencoder for Recommendation. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR '18)*. ACM, New York, NY, USA, 27–34. https://doi.org/10.1145/3234944.3234956

[12] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *ICLR* abs/1312.6114v2 (2014). arXiv:1312.6114v2 https://arxiv.org/abs/1312.6114v2

[13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS.

[14] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *CoRR* abs/1803.05170 (2018). arXiv:1803.05170 http://arxiv.org/abs/1803.05170

[15] Jian Liu, Chuan Shi, Binbin Hu, Shenghua Liu, and Philip S. Yu. 2017. Personalized Ranking Recommendation via Integrating Multiple Feedbacks. In *Advances in Knowledge Discovery and Data Mining*, Jinho Kim, Kyuseok Shim, Longbing Cao, Jae-Gil Lee, Xuemin Lin, and Yang-Sae Moon (Eds.). Springer International Publishing, Cham, 131–143.

[16] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

[17] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. https://arxiv.org/pdf/1706.03762.pdf

[19] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 1096–1103. https://doi.org/10.1145/1390156.1390294

[20] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, New York, NY, USA, 153–162. https://doi.org/10.1145/2835776.2835837

[21] Yong Zheng, Robin Burke, and Bamshad Mobasher. 2012. Optimal feature selection for context-aware recommendation using differential relaxation. In *In ACM RecSys' 12, Proceedings of the 4th International Workshop on Context-Aware Recommender Systems (CARS 2012). ACM.*

A SUPPLEMENT

In this section, we provide details for reproducibility of our experiments and results.

A.1 Implementation Notes

Dataset Processing

We have compared multiple generative models on Yelp[4], Dianping[5][15] and Koubei datasets as described in the experiments section. And we want to add more details of the methods we use for dataset processing.

For Yelp public dataset, we convert the explicit Yelp review data (stars) to implicit feedback data (0/1). We follow [10][14] to treat reviews with 4 and 5 stars as positive ($y = 1$) and the others as negative ($y = 0$). We only keep active users with sufficient interactions and filter the subset of users with minimum reviews number as 20. The train/test split is 80%/20%. For each user, we randomly sample 80% of the interactions for training and use the other 20% for testing. Since evaluating NDCG@K on all candidates is time consuming, we adopt the common negative sampling strategy and set the rate $NS = 50$. This means that for each positive user-item interaction data, we randomly sample 50 items which user does not interact with as negative samples. For fair comparison, we prepare the pre-training dataset using only the positive interaction data from the training dataset.

The Dianping dataset is prepared in the same way as Yelp dataset, except we are using "leave-one-out" strategy for train/test split. For each user, we randomly hold one positive item for testing and the remaining positive items are used for training. Users' minimum reviews number is set to 4 and the negative sampling rate is set to $NS = 50$.

For Koubei dataset, we collect 30 consecutive days of positive instance (user click) for pre-training and use 14 days of user logs data, both positive (user click) and negative (impression without click), for training, and use the data in the next day for testing. For offline evaluation, we randomly sample subset from the production CTR dataset and the statistics of pre-train/train/test instances of the offline dataset is presented in table 4.

Model Configurations and Hyperparameters

In our experiments, we implement and compare three different generative models during pre-training stage. We implement multi-head DAE and VAE with the same head number K as MACDAE for comparison. To compare the effect of different head number K, we evaluate the MACDAE model on three different configurations, $K = [4, 8, 16]$. For the Koubei dataset, the dimension of the original input vector x is 532, so we set the hidden layer size d_h to 256 and the number of heads K to 8, and each head has dimension of 32. As for Dianping dataset, the dimension of input x to encoder is 320, the hidden layer dimension is set to 128 and the number of heads K is set to 4 with each head as dimension 32. For Yelp dataset, the input x has dimension 403, and we set hidden layer size to 256 with the number of heads K to 4. For hyper-parameters of DAE and MACDAE, dropout probability p is set to 0.95. For MACDAE, we set the penalty cost λ to 0.05 for Koubei dataset and 0.005 for Yelp/Dianping dataset. And constraint on cosine similarity ϵ is set to 0.75. We also use Adam as the default optimizer with learning rate set to 0.001. The pre-training for all datasets lasts 5 epochs. And the training for downstream supervised learning model lasts 10 epochs. In the supervised learning task, the Wide&Deep base model is implemented as follows. For Koubei dataset, the Wide&Deep model has deep hidden layers as [512, 256, 256]. For Yelp dataset, we set the embedding dimension of user and item to 64, deep hidden layers to [256]. For Dianping dataset, we also set embedding dimension of user and item to 64 and deep hidden layers to [128].

Running Environment

In terms of running environment, the pre-training/training/testing of Yelp and Dianping datasets are performed on the MACBOOK Pro with 2.2 GHz Intel Core i7 8 cores CPU and 16 GB memory, which lasts around 10 hours for pre-training and 9 hours for training. The evaluation of Koubei offline dataset is performed on large-scale distributed systems with 30 workers. For online production model, the pre-training of Koubei production dataset lasts around one day. The training of supervised learning models of Koubei dataset last 9-10 hours everyday.

Table 4: Statistics of Koubei Dataset

Dataset	Pretrain Instance	Train Instance	Test Instance
Koubei	17.7M	40.5M	2.9M

A.2 Algorithm and Code Implementation

We implement all the algorithms described above using tensorflow[6] in python. And we will describe the code implementation of the models we compared.

- **MACDAE**: We implement our proposed Mixture Attentional Constrained Denoise AutoEncoder (MACDAE) model by tensorflow API in python. The experimental code is available on GitHub repo(https://github.com/rockingdingo/context_recommendation).
- **Multi-Head DAE/VAE**: For baseline comparison models multi-head Denoise AutoEncoder(DAE), and multi-head Variational AutoEncoder(VAE), we slightly modify the original research code of tensorflow (https://github.com/tensorflow/models/tree/master/research/autoencoder) to the multi-head version.
- **Wide&Deep**: We follow the official release of Wide&Deep implementation of tensorflow models (https://github.com/tensorflow/models/tree/master/official/wide_deep). Our implementation of Wide&Deep model with pre-training slightly modifies the original code. It restores the pre-training model parameters in current session of tensorflow first and the parameters of the prediction model are updated by either feature-based approach or fine-tuning approach.
- **DeepFM/NFM**: For the baseline comparison model, we refer the DeepFM and NFM model implementation in this GitHub repo (https://github.com/princewen/tensorflow_practice/tree/master/recommendation). We make several modifications to the original implementation, such as feature extractor to fit the input of our datasets.

[4]https://www.kaggle.com/yelp-dataset/yelp-dataset
[5]http://shichuan.org/HIN_dataset.html
[6]https://www.tensorflow.org/

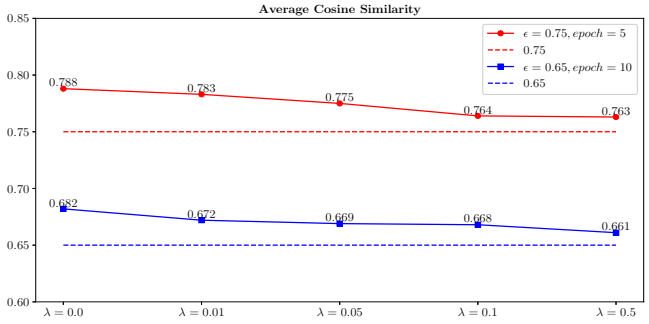

Figure 7: Average cosine similarity of multi-heads in MACDAE model pre-trained on Koubei dataset

A.3 Discussions

Effect of Constraint Threshold and Penalty

To avoid the problem that multiple contextual components converge to similar latent subspaces, we have applied constraint on the cosine similarity between each heads and add penalty cost to the overall cost function. We set the hyperparameter ϵ as the threshold of maximal cosine similarity between two heads, and λ as the penalty cost of breaking the constraint in the objective function. Figure 7 shows the empirical analysis result of the MACDAE model pre-trained on the Koubei dataset with different hyper-parameters. The multi-head number K is set to 8. The penalty cost λ is set to $[0.0, 0.01, 0.05, 0.1, 0.5]$. The ϵ is set to 0.75 for 5 pre-training epochs and 0.65 for 10 pre-training epochs. The results show that without applying the constraint, the average cosine similarity among multiple heads can be as high as 0.788, which means that the angle is close to 0 degree. The average cosine similarity will gradually decrease as we increase the penalty cost λ and the epoch number.

Contextual Feature Importance of Real-world Dataset

We will present our findings of contextual feature importance of real-world recommendation dataset collected from Koubei's online "Guess You Like" recommendation in Table 5. To evaluate the importance of contextual features, we adopt the popular "leave-one-out" strategy, which leaves one contextual feature out and repeat the training process, then evaluate the actual AUC change compared to the baseline model which all the contextual features are available. The dataset is collected on click-through rate(CTR) and conversion-rate(CVR) respectively. CTR measures the percentage of users have impression on the candidates that actually click, and CVR measures the percentage of users click the candidates that actually purchase.

The analyses suggest that time-related contextual features c.time (time intervals of the day, e.g. morning, afternoon, evening, ...), c.weekday (weekdays or weekends) and distance-related features u.s.dist (real-time distance between users' location and recommended shop) rank much higher in AUC contribution on the conversion rate(CVR) dataset than the click-through rate (CTR) dataset. For the click-through rate (CTR) dataset, users' behavioral features, u.s.30d.buy.id (shops that users purchased in the last 30 days) and u.s.30d.clk.id (shops that users clicked in the last 30 days), contribute more than the contextual features and price related features. While for conversion rate (CVR) dataset, the situation is different, where s.price (average transaction price of shop) ranks on the top

Table 5: Contextual Feature Importance(AUC change) of CTR and CVR datasets in Decreasing Order

CTR Dataset		CVR Dataset	
Features	AUC Change	Features	AUC Change
u.s.30d.buy.id	0.0458	s.price	0.0745
u.age,u.gender	0.0347	c.time,c.weekday	0.0477
u.s.30d.clk.id	0.0278	u.s.dist	0.0463
c.time,c.weekday	0.0265	u.s.30d.buy.id	0.0449
s.price	0.0263	u.s.30d.clk.id	0.0360
u.s.dist	0.0244	u.age,u.gender	0.0231

in AUC contribution. This is aligned with our expectation that users are heavily influenced by their past behaviors of clicks and purchases in the first impression of click-through rate (CTR). Price related features will influence users' decision more in the second conversion stage (CVR), in which clicks convert to actual purchases.