Order Fulfillment Cycle Time Estimation for On-Demand Food Delivery

Lin Zhu, Wei Yu, Kairong Zhou, Xing Wang, Wenxing Feng, Pengyu Wang, Ning Chen, and Pei Lee
{linzhu.zl,yuwei.yuwei,zkr178397,wx214561,wenxing.fwx,wpy224927,daniel.cn,lp171042}@alibaba-inc.com
Alibaba Group
Shanghai, China

ABSTRACT

By providing customers with conveniences such as easy access to an extensive variety of restaurants, effortless food ordering and fast delivery, on-demand food delivery (OFD) platforms have achieved explosive growth in recent years. A crucial machine learning task performed at OFD platforms is prediction of the Order Fulfillment Cycle Time (OFCT), which refers to the amount of time elapsed between a customer places an order and he/she receives the meal. The accuracy of predicted OFCT is important for customer satisfaction, as it needs to be communicated to a customer before he/she places the order, and is considered as a service promise that should be fulfilled as well as possible. As a result, the estimated OFCT also heavily influences planning decisions such as dispatching and routing.

In this paper, we present the OFCT prediction model that is currently deployed at Ele.me, which is one of the world's largest OFD platforms and delivers over 10 million meals in more than 200 Chinese cities every day. By dissecting the order fulfillment cycle of a meal order, we identify key factors behind OFCT, and capture them with numerous features constructed using a wide range of data sources. These features are fed into a deep neural network (DNN), which further incorporates representations of couriers, restaurants and delivery destinations to enhance prediction efficacy. Finally, a novel post-processing layer is introduced to improve convergence speed by better accounting for the distributional mismatch between the true OFCT values and those predicted by the model at initialization. Extensive offline and online experiments demonstrate the effectiveness of our approach.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Applied computing** → *Forecasting*; • **Computing methodologies** → Neural networks.

KEYWORDS

on-demand food delivery; neural networks; order fulfillment cycle time estimation

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403307

ACM Reference Format:
Lin Zhu, Wei Yu, Kairong Zhou, Xing Wang, Wenxing Feng, Pengyu Wang, Ning Chen, and Pei Lee. 2020. Order Fulfillment Cycle Time Estimation for On-Demand Food Delivery. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403307

1 INTRODUCTION

Fueled by the wide adoption of technological advances such as cellular network and global positioning system (GPS), the food service industry is undergoing a rapid transformation. Tech-savvy customers, who are accustomed to the convenience of simple-and-immediate purchasing and delivery through online E-commerce platforms, increasingly desire the same experience when it comes to eating. As a result, on-demand food delivery (OFD) platforms such as Uber Eats[1], DoorDash[2], Meituan-Dianping[3] and Ele.me[4] have experienced explosive growth in recent years [4, 8]. OFD platforms connect customers and restaurants via mobile applications, where one can order food with just a few taps of the phone screen. Meanwhile, by establishing their own delivery fleets of vehicles and couriers, the platforms also directly handle the food delivery tasks for restaurants. For restaurants, such platforms open a convenient new channel to drive new sales, without inflicting additional investments in logistics. For customers, these platforms offer miscellaneous conveniences such as access to an extensive variety of restaurants, effortless food ordering, as well as fast delivery.

An important machine learning task performed at OFD performs is Order Fulfillment Cycle Time (OFCT) prediction, namely estimation of the amount of time elapsed between a customer places an order and receipt of the food [32]. The estimated OFCT is presented to a customer before he/she makes the decision on ordering, and its accuracy is important for setting up appropriate expectation and avoiding inconvenience caused by failing to deliver the meal package on time. Meanwhile, the predicted OFCT is also considered as a service promise that needs to be fulfilled by the platform as well as possible, and thus is also an important factor to consider when making planning decisions such as selection of the courier who will receive the order.

In this paper, we present the OFCT prediction model that is currently deployed at Ele.me, which is one of the world's largest OFD platforms, and delivers over 10 million meals in more than 200 Chinese cities every day. We construct a wide range of features intended to capture different aspects of the delivery process.

[1] https://www.ubereats.com
[2] https://www.doordash.com
[3] https://waimai.meituan.com/
[4] https://www.ele.me/

These features are fed into a deep neural network (DNN), which further incorporates latent representations of restaurants, origins and destinations to enhance predictive power. Extensive offline experiments and online A/B testing demonstrate the effectiveness of our approach.

2 RELATED WORK

2.1 OFCT Prediction

Clearly, our work is most closely related to the OFCT prediction systems deployed at other OFD platforms. Unfortunately, publicly available documents about these systems are limited to high-level overviews (e.g., [22, 32]), which only offer sketchy descriptions of implementation details, such as the features used and their individual contributions to the model accuracy, and the strategies taken to handle the unavailability of critical information. For example, similar to Section 4.4 in our paper, [32] also propose to infer the unknown ground-truth cooking time using other signals, however, their specific approach to achieve this goal is not discussed. Meanwhile, these systems are generally based on Gradient Boosting Machine (GBM), while we demonstrate that a properly designed DNN model can achieve substantially better performance. To the best of our knowledge, we are the first to fully examine and disclose detailed solutions to the important issues that need to be addressed for predicting OFCT.

2.2 Estimated Time of Arrival

OFCT prediction could also be viewed as a variant of the Estimated Time of Arrival (ETA) problem [17–19, 31, 33]. In the broadest sense, ETA consists of evaluating the time required to travel between an origin-destination (OD) pair, and there are two typical scenarios: route-based and OD-based. Route-based ones estimate travel time for a specific route, and generally proceed by estimating and then aggregating travel times for road segments contained in that route [17, 18]. These methods do not work well when the actual route is unknown and cannot be accurately predicted [19]. On the other hand, OD-based approaches only require the locations of the OD pair and the starting time as input. For example, Wang et al. propose TEMP+R [31], which estimates the travel time as the weighted average of travel times of similar historical trips. To remedy the problem that important route-related information are unavailable as input for OD-based approaches, Li et al. [19] devise MURAT, a multi-task learning framework to instead encode route information as extra training tasks, thereby learning more meaningful representations that improve the predictive performance.

As will be elaborated in Section 3.1, compared with general ETA problems, several issues make OFCT prediction a unique challenge, as listed below:

(1) **Significantly more influencing factors.** The primary issues considered in general ETA problems are limited to weather/traffic conditions, temporal information, as well as geographical information about the OD-pair and the travel route [18, 33]. In contrast, OFCT is also additionally influenced by the location and characteristics of the restaurant, the food preparation time, the orders assigned to the same courier (which will alter his/her delivery itinerary), etc.

(2) **Unavailability of critical information.** The predicted OFCT must be communicated to the customer before the order is created. However, information about many of the critical factors mentioned in the preceding paragraph is still not known at that early moment, including but not limited to the courier (and thus the origin of the delivery process) and the actual route.

Due to these issues, it is infeasible to directly apply standard ETA methods to our setting. However, the possibilities of borrowing ideas from existing ETA methods to improve our model have been explored (e.g., Section 4.7). Meanwhile, in Section 6.1.4 we will also attempt to adapt TEMP+R and MURAT for OFCT prediction, and evaluate their performance.

2.3 Time Window Assignment Problem

As will be discussed in Section 3.1, the predicted OFCTs are used to generate soft time window (TW) constraints on the combinatorical optimization problem solved by the logistics planning engine. From this perspective, OFCT prediction is also very similar to the TW assignment problem studied in the vehicle routing literature [26]. However, most of existing work on TW assignment only deal with the scenario where all customers that would place orders are known *a priori* (e.g, [24, 26, 29]), and also generally make restrictive assumptions about the generative process of demands. For example, [24, 26] assume that all possible demand scenarios are sampled from a known finite mixture model, while [28] is based on the assumption that the ground-truth total travel time and itinerary for serving an arbitrary set of customer requests can be obtained via simulations. As a result, these approaches are also unsuitable for our problem setting, where such strong assumptions are unlikely to hold, and future requests are not yet known when OFCT needs to be predicted.

3 BACKGROUND

3.1 Basic Concepts

In preparation for description of the proposed OFCT prediction model, in this section we provide necessary background information about the logistics planning system of Ele.me, as well as how our model fits in the overall system.

Ele.me organizes its food delivery network as a two-tiered system, which is typical for urban last-mile delivery companies [6, 12]. Concretely, the entire service area of each city is partitioned into smaller regions called *grids*, each of which is equipped with a *station* to solely handle the delivery service therein. The couriers are allocated to specific stations, and the allocation decisions remain fixed for an extended period of time once they are made. This kind of stableness allows the couriers to get familiar with the geographical details of assigned grids, thereby improving service efficiency and customer satisfaction in the long run [30]. Every grid contains a number of *positions-of-interest (POIs)*. POIs are classified into many categories, such as *residential area*, *factory*, *school*, and *office building*. **The delivery destination of every order must be inside a specific POI**.

On a daily basis, the platform needs to serve dynamic customer requests using available couriers. As illustrated in Figure 1, a complete order fulfillment cycle can be described as follows:

(1) The OFCT estimate is communicated to the customer when he/she makes the the final decision on ordering by paying the fees (shown in Figure 2). After the order is created, the restaurant is notified to prepare the food, the OFCT estimate stays unchanged thereafter, and the following issues will be determined:
   (a) *dispatching*, which refers to selection of the courier who will deliver the order. Note that it is possible that the selected courier is already en route to execute other orders. A set of orders that are delivered by the same courier in a single trip is called a *bundle*, and the usage of bundles can increase delivery efficiency and ameliorate sudden vagaries of supply and demand [22, 27].
   (b) *routing*, namely to determine the sequence of pickups and drop-offs for the courier.

   Greedily solving (a) and (b) for every order immediately after its creation can lead to myopic decisions that do not work well, as the viability of serving future unknown orders is not taken into consideration [21]. Instead, the planning system would postpone the decisions for a while to wait for new orders, and then infer planning decisions for these orders altogether by solving a combinatorial optimization problem. The loss function of this optimization problem consists of two parts: *logistics cost* and *customer inconvenience cost*. The former measures factors such as the total travel distance/time of all couriers, while the latter quantifies how many of these orders would be delivered too late or too early with respect to their OFCT estimates. Computationally, this is essentially a vehicle routing problem with time window constraints (VRPTW), which is well-studied in the literature [1, 21, 23, 27].
(2) The chosen courier goes to the restaurant and fetches the food package when it is ready. Note that the courier may need to pick up multiple orders at the restaurant, and can only leave when all of them are ready for delivery.
(3) The courier drops off the food package at the delivery destination, and completes the order. Note that new orders can still be assigned to the courier up to this moment, and the courier is not restricted to finish all pending deliveries before a new pickup.

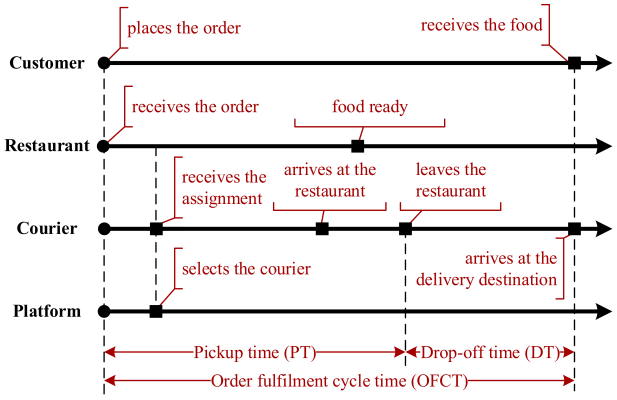

Figure 1: A complete order fulfillment cycle.

Figure 2: Estimated OFCT shown in Ele.me app.

3.2 Problem Formulation

With the above introduced concepts in mind, we next provide a formal definition of the OFCT prediction problem.

We consider a specific grid, and O denotes the set of orders placed therein. Let $o \in O$ be an arbitrary food order, r_o is the restaurant from which the food is requested, d_o is the delivery destination. Critical moments in the order fulfillment cycle of o are denoted by t_o^{creation}, t_o^{arrival}, $t_o^{\text{departure}}$ and t_o^{receival}. Definitions of these time points can be found in Table 1. $O_o^{\mathbb{H}} \subseteq O$ denotes the set of orders that are completed before t_o^{creation}.

Problem. By using $O_o^{\mathbb{H}}$ and other information that is available at t_o^{creation}, produce an accurate prediction of the OFCT of o, defined as:

$$\text{OFCT}_o = t_o^{\text{receival}} - t_o^{\text{creation}}. \tag{1}$$

To facilitate discussions, we further divide OFCT into two parts, namely *pickup time (PT)* and *drop-off time (DT)*, which are respectively defined as:

$$\text{PT}_o = t_o^{\text{departure}} - t_o^{\text{creation}}, \tag{2}$$

$$\text{DT}_o = t_o^{\text{receival}} - t_o^{\text{departure}}. \tag{3}$$

Meanwhile, the time taken by r_o to cook food for o is denoted by CT_o:

$$\text{CT}_o = t_o^{\text{ready}} - t_o^{\text{creation}}. \tag{4}$$

4 FEATURE EXTRACTION

In this section we describe the various classes of features that we derive to predict (1).

4.1 Order Information

These features include:

(1) Spatial features, such as the identifiers (IDs) of d_o, r_o and the city/grid where o is placed at. The coordinates of d_o and r_o are also used.
(2) Temporal features. We adopt hour of the day and workday as the temporal features. As is shown in Figure 3(a), due to demand fluctuations, OFCT is generally lower in workday, and would also rise sharply during meal times.

Table 1: Table of symbols.

Symbol	Definition
O	The set of meal orders
C	The set of active couriers
o	A meal order
r_o	The restaurant associated with o
c_o	The courier who picks up and delivers o
d_o	The delivery destination of o
OFCT_o	The order fulfillment cycle time of o
PT_o	The pickup time of o
DT_o	The drop-off time of o
CT_o	The food preparation time of o
t_o^{creation}	The time point at which o is created
t_o^{ready}	The time point at which o is ready for pickup
t_o^{arrival}	The time point at which c_o arrives at r_o
$t_o^{\text{departure}}$	The time point at which c_o leaves r_o
t_o^{receival}	The time point at which the customer receives the food
$O_o^{\mathbb{H}}$	Historical orders that are completed before t_o^{creation}
$O_o^{\text{uncompleted}}$	The set of uncompleted orders at t_o^{creation}
O_o^{fetched}	Orders that have been fetched from restaurants
O_c	Uncompleted orders that are assigned to c
$\text{OFCT}_o^{\text{predict}}$	OFCT estimate given by the model

(3) Order size features: a bigger meal order could takes longer time for the restaurant to prepare, and is also harder to deliver due to increased weight and volume of the food package. We use the following features to describe size of o:
   (a) sku_num_o: every individual item that could be bought at Ele.me is registered internally as a stock keeping unit (SKU), thus a straightforward way to measure order size would be to count the number of SKUs that a order contains.
   (b) price_o: sku_num_o may underestimate the size of o if it contains a *set meal*, which bundles several dishes and beverages together as a single SKU. Thus we also include the total price of SKUs as an additional feature, since set meals would be priced higher. Note that the original order price before discount is used. Meanwhile, the price feature can also differentiate sizes of orders that contains the same number of SKUs. See Figure 3(b) for comparisons of these two features.

4.2 Aggregation Features

Useful information could be gathered through the sensors built in the mobile phones of couriers. For instance, GPS can record the outdoor trajectories of couriers, while WiFi and Bluetooth can be used to infer their positions indoors [34]. By integrating these signals, ground-truth information about DTs/PTs/OFCTs of orders in $O_o^{\mathbb{H}}$ can be obtained, and we use statistical summaries to aggregate them as features for o. Four different summary statistics are used, including 20th percentile, mean, standard deviation, and 80th percentile. We first select some of the historical orders according to a certain rule, and then compute summary features using these orders only. Multiple variants of these features are constructed by using different rules, such as orders that are placed in the same city/grid/restaurant/hour of day as o, or orders that are completed less than 1 hour/10 minutes before t_o^{creation}.

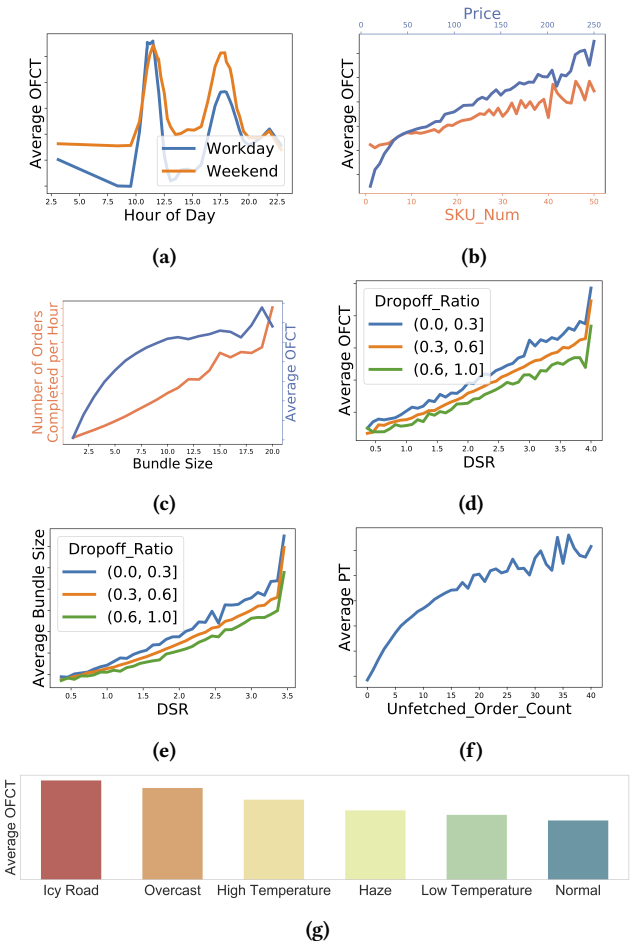

Figure 3: Feature analysis using 5 million randomly sampled orders that were placed at Ele.me between May 2019 and December 2019.

4.3 Dish Features

These features are intended to capture the influence of differences between dishes on OFCT. As each SKU is assigned a unique identifier (UID), we can represent o as a bag of SKU UIDs that it contains. Meanwhile, we classify SKUs into 116 categories, such as *ramen*, *pizza*, *hamburger*, *barbecue* and *hotpot*. The classification model uses the title of a SKU as input, and is obtained by fine-tuning the Chinese-language BERT model [7] using around 20,000 training examples with hand-annotated labels. The set of predicted categories of SKUs included in o is then used as an additional feature.

4.4 Cooking Time Features

The cooking time (4) is an important factor to consider when estimating OFCT_o. Unfortunately, the ground truth of cooking time is unavailable for model training, as the restaurants generally lack both manpower and incentive to record such information for the platform [32].

In order to improve the model by taking the cooking time into account, we attempt to identify historical orders whose cooking times could be more reliably estimated using available signals. Concretely, based on domain expertise, we make the assumption that one of the following three possible scenarios could happen when the courier goes to the restaurant to pick up the food: (i) the courier may arrive too early and has to wait until the food is ready; (ii) the courier may need to pick up other orders from the restaurant, and can only leave after the latest ready time of these orders; and (iii) the courier arrives after the food is ready and could immediately fetch the food package and depart for the customer. Based on this assumption, for $\forall \widetilde{o} \in O_o^{\mathbb{H}}$, let $O_{\widetilde{o}}^{\mathrm{r}}$ denote the set of orders that $c_{\widetilde{o}}$ picks up from $r_{\widetilde{o}}$ along with $\widetilde{o}$, we can approximately re-express $t_{\widetilde{o}}^{\mathrm{departure}}$ as the maximum of the ready times of orders in $O_{\widetilde{o}}^{\mathrm{r}}$ and the courier's arrival time at the restaurant:

$$\left| t_{\widetilde{o}}^{\mathrm{departure}} - \max\left\{ t_{\widetilde{o}}^{\mathrm{arrival}}, \max_{\widehat{o} \in O_{\widetilde{o}}^{\mathrm{r}}} \left\{ t_{\widehat{o}}^{\mathrm{ready}} \right\} \right\} \right| < \alpha, \tag{5}$$

where $\alpha > 0$ is a predefined hyperparameter to account for potential error in such an approximation. Meanwhile, let $O_o^{\mathrm{CT}} \subseteq O_o^{\mathbb{H}}$ be defined as

$$O_o^{\mathrm{CT}} = \left\{ \widetilde{o} \in O_o^{\mathbb{H}} \middle| t_{\widetilde{o}}^{\mathrm{departure}} - t_{\widetilde{o}}^{\mathrm{arrival}} > \alpha, O_{\widetilde{o}}^{\mathrm{r}} = \{\widetilde{o}\} \right\}, \tag{6}$$

then according to (4), (5) and (6), for any $\widetilde{o} \in O_o^{\mathrm{CT}}$ we have

$$\max\left(\max_{\widehat{o} \in O_{\widetilde{o}}^{\mathrm{r}}} \left(t_{\widehat{o}}^{\mathrm{ready}} \right), t_{\widetilde{o}}^{\mathrm{arrival}} \right) > t_{\widetilde{o}}^{\mathrm{departure}} - \alpha > t_{\widetilde{o}}^{\mathrm{arrival}}.$$

It follows that $\max_{\widehat{o} \in O_{\widetilde{o}}^{\mathrm{r}}} \left(t_{\widehat{o}}^{\mathrm{ready}} \right) > t_{\widetilde{o}}^{\mathrm{arrival}}$, and

$$\begin{aligned} \left| \mathrm{CT}_{\widetilde{o}} - \mathrm{PT}_{\widetilde{o}} \right| &= \left| t_{\widetilde{o}}^{\mathrm{ready}} - t_{\widetilde{o}}^{\mathrm{departure}} \right| \\ &= \left| \max\left(\max_{\widehat{o} \in O_{\widetilde{o}}^{\mathrm{r}}} \left(t_{\widehat{o}}^{\mathrm{ready}} \right), t_{\widetilde{o}}^{\mathrm{arrival}} \right) - t_{\widetilde{o}}^{\mathrm{departure}} \right| \\ &< \alpha. \end{aligned}$$

In other words, the departure time from the restaurant could be used as a surrogate of the cooking time for orders in O_o^{CT}. Leveraging this conclusion, we construct cooking time features for restaurants by aggregating departure times of orders in O_o^{CT}, using the same summary statistics as in Section 4.2.

4.5 Supply-and-Demand Features

Due to the variance of customer demand, the number of incoming food orders can vary significantly and abruptly both in spatial and temporal dimensions. Such fluctuations have a strong impact on OFCT.

Firstly, sudden spikes in customer demand can be ameliorated by the bundling strategy described in Section 3.1. However, as shown in Figure 3, while increasing of the bundle size could effectively improve logistics efficiency (measured by the average number of orders completed per unit time), the average OFCT of involved orders also becomes longer. This is because the courier cannot fulfill every order efficiently by taking the fastest route, and needs to detour to pickup and deliver other orders (see Figure 4 for illustrative

Figure 4: Two illustrative examples. In each panel, P1 and P2 denote the pickup locations (i.e., restaurants) of two orders, while D1 and D2 are their delivery destinations. Blue line is the final courier trajectory when delivering these two orders, and red line is the fastest route to travel between P1 and D1, suggested by a map service provider.

examples). We use demand-to-supply ratio (DSR) to describe this phenomenon:

$$\mathrm{DSR}_o = |C|^{-1} \left| O_o^{\mathrm{uncompleted}} \right|, \tag{7}$$

where $O_o^{\mathrm{uncompleted}}$ denotes the set of uncompleted orders at t_o^{creation}, $|\cdot|$ denotes the cardinality of a set. We also define the dropoff_ratio feature as:

$$\mathrm{dropoff\_ratio}_o = \left| O_o^{\mathrm{uncompleted}} \right|^{-1} \left| O_o^{\mathrm{dropoff}} \right|, \tag{8}$$

where $O_o^{\mathrm{dropoff}} \subseteq O_o^{\mathrm{uncompleted}}$ denotes the set of orders that have already been picked up from the restaurants and are in the dropoff stage. Figure 3(d) and (e) depict the relationship between these features and OFCT/bundle size. If DSR has a higher value, average bundle size will increase and thus OFCT would also become longer. On the other hand, high value of dropoff_ratio implies that a large portion of orders are nearing completion, and the corresponding couriers will then be 'freed' to accept other orders. As a result, the bundle size and OFCT both would also decrease accordingly.

Secondly, the customer requests may sometimes be concentrated in a specific popular restaurant and exceed its food preparation

capacity. We quantify how busy the restaurant is via the feature unfetched_order_count, defined as the number of orders that request food from the restaurant and have not yet been fetched by the associated couriers. The pickup time generally increases when unfetched_order_count becomes larger as the courier needs to spend more time waiting at the restaurant (Figure 3(f)).

All of the aforementioned demand-and-supply features are constructed using orders that are already created at the moment when OFCT is estimated. A naturally arising concern is that they may be biased since future unknown customer requests are ignored. This concern can be addressed to an extent by performing demand forecasting using a dedicated sub-model, the output of which is then used as input for the main OFCT estimation model. However, we failed to get any noticeable performance improvement by implementing this idea in various ways. The underlying reason is that customer demand is highly volatile and cannot be predicted with very high precision. Meanwhile, demand forecasting itself relies on information that is already known, and some of the features that we use for OFCT estimation (such as the temporal/spatial/meteorological ones) are exactly the features that are commonly adopted for demand prediction [35]. As a result, the estimated demand does not bring in new useful information for improving OFCT prediction.

4.6 Courier Features

As explained in Section 3.1, the courier that will pickup and deliver o is not determined yet when OFCT_o needs to be predicted. To ameliorate this problem, we propose to use a separate model component to rank couriers based on the probabilities that they will receive o, and use information of top-ranked couriers to enhance model accuracy. For this purpose, we first define a number of features for every $c \in C$, based on the criteria that they should be predictive of (i) the likelihood that c will receive the assignment and (ii) OFCT_o if c is indeed selected. Meanwhile, for efficiency considerations, computationally expensive ones (e.g., those obtained by simulating the VRPTW used to determine couriers) are excluded. The resultant features include:

- **Pickup distance.** This feature is defined as the shortest road network distance from the current location of c to the restaurant r_o. The intuition behind this feature is immediate: recall from Section 3.1 that the selected courier should minimize the combination of logistics cost and user inconvenience cost, thus c is not likely to be chosen if pickup distance has a high value, because in such cases c needs to travel long distance to pick up o, leading to higher logistics cost.
- **Work load.** Denote O_c as the set of uncompleted orders that are currently assigned to c, this feature is defined as the cardinality of O_c. As discussed in Section 4.5, larger work load will increase the total amount of detour and therefore the OFCTs of orders in O_c, resulting in higher logistics cost and user inconvenience cost.
- **Urgency.** These features are defined as the average or minimal remaining times of orders in O_c. Assignment of o to a courier with urgent orders may cause him/her to violate these orders' time window constraints, in which case the customer inconvenience cost will increase.
- **Mutual Distances.** These features are defined as average or minimum of shortest road network distances between the destination/restaurant of o and those of the orders in O_c. High values of these features imply that c may need to take a significant detour to deliver o, which will increase the overall logistics cost.

4.7 ETA-based Drop-off Time Feature

As both the origin (i.e., r_o) and the destination (i.e., d_o) of the drop-off segment of the delivery process are known, we attempt to characterize DT_o by using a modified version of TEMP+R [31], an OD-based ETA method which we briefly reviewed in Section 2.2. Specifically, we first map every order $\widetilde{o} \in O$ to a vector $\mathbf{x}^{\text{eta}}_{\widetilde{o}} \in \mathbb{R}^5$ as:

$$\mathbf{x}^{\text{eta}}_{\widetilde{o}} = \left[\text{lon}\left(r_{\widetilde{o}}\right), \text{lat}\left(r_{\widetilde{o}}\right), \text{lon}\left(p_{\widetilde{o}}\right), \text{lat}\left(p_{\widetilde{o}}\right), \text{hr}\left(a^{\text{creation}}_{\widetilde{o}}\right)\right]^T \odot \mathbf{w},$$

where $\text{hr}(\cdot)$, $\text{lon}(\cdot)$, and $\text{lat}(\cdot)$ are functions that map the input to hour of the day, longitude and latitude, respectively. $\mathbf{w} \in \mathbb{R}^5$ is used to assign different weights to elements in $\mathbf{x}^{\text{eta}}_{\widetilde{o}}$. Then the dissimilarity between $\forall\widetilde{o} \in O$ and $\forall\widehat{o} \in O$ is defined as the Euclidean distance between $\mathbf{x}^{\text{eta}}_{\widetilde{o}}$ and $\mathbf{x}^{\text{eta}}_{\widehat{o}}$:

$$\text{dissim}\left(\widetilde{o}, \widehat{o}\right) = \left\|\mathbf{x}^{\text{eta}}_{\widetilde{o}} - \mathbf{x}^{\text{eta}}_{\widehat{o}}\right\|_2. \tag{9}$$

Based on (9), we retrieve $\{o_i\}_{i=1}^k \subseteq O^{\mathbb{H}}_o$, which consists of k historical orders that are most similar to o, and create the following feature:

$$\left(\sum_{i=1}^{k} w_i\right)^{-1} \sum_{i=1}^{k} \left(w_i \cdot \text{DT}_{o_i}\right), \tag{10}$$

where $w_i = \exp\left(-\frac{\text{dissim}(o_i, o)}{\sigma^2}\right)$, σ is the softmax temperature parameter.

As is the case with TEMP+R, (10) predicts DT_o using weighted average of the DTs of historical trips, and higher weights are assigned to more similar trips. The main difference is that Euclidean distance is directly used here to define (dis)similarity, which allows us to use accelerated similarity search algorithms [13, 14] to efficiently retrieve similar trips.

4.8 Meteorological Features

Available weather data include the temperature, the Air Quality Index (AQI), wind speed and weather condition. The first three are directly adopted as numerical features. On the other hand, weather condition is described using terms such as *icy/flooded road*, *high/normal/low temperature* and *haze*. These terms are not necessarily mutually exclusive and can used in combination. Figure 3(g) depicts the correlation between weather conditions and average values of OFCT.

5 MODEL

Similar to [5, 11, 37], we follow the "encode&predict" paradigm to build the main body of our model: firstly, heterogeneous features of different fields are respectively encoded as fixed-length vectors; these vectors are concatenated into a wide feature vector, which is then fed into a regression module to obtain the final predictive result. The overall model architecture is illustrated in Figure 5.

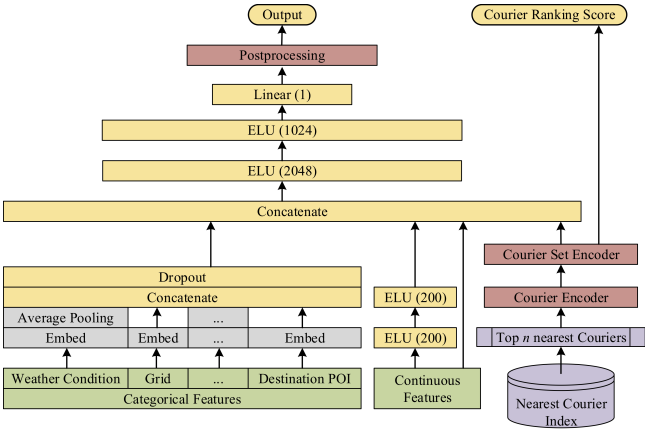

Figure 5: Model architecture.

5.1 Feature Encoder

5.1.1 Encoder for Numerical Features. Given the sensitivity of neural networks to input scaling [5, 11], numerical features are normalized to the interval [0, 1]. Meanwhile, the dimensionality of the orginal numerical features is relatively low (around 100), which may restrict the expressive power of neural networks. We thus pass them through 2 layers of fully connected exponential linear units (ELUs) [3], then concatenate the output with the original features to obtain an enhanced vector representation.

5.1.2 Encoder for Categorical Features. Embedding layers are used to transform categorical features into dense vector representations, and a dropout layer [25] is incorporated to control overfitting. Additionally, some of the adopted features, such as dish features in Section 4.1 and weather condition in Section 4.8, are essentially sets of categorical ids. We follow [11, 37] and obtain their representations by simply averaging the embeddings of contained ids.

5.1.3 Encoder for Couriers Features. As mentioned in Section 4.6, the couriers need to be ranked using constructed features. To accelerate computation, we adopt a two-stage strategy that is commonly used in large-scale ranking applications [5, 36]: a small set of candidate couriers, denoted by $C^{\text{candidate}}$, is first selected from C via a efficient ranker, and then re-ranked by a more sophisticated neural module. Here, we simply use pickup distance defined in Section 4.6 as the first-stage ranker, and thus the resultant $C^{\text{candidate}}$ consists of couriers that are closest to r_o. As for the re-ranking module, we adopt the Query Invariant Listwise Context Modeling (QILCM) method proposed in our previous work [38], as it could achieve state-of-the-art performance for ranking refinement on several benchmark datasets. Mathematical formulations of this module can be found in Appendix A.2.

For $\forall c \in C^{\text{candidate}}$, QILCM generates a vector representation $\widetilde{\mathbf{h}}_c$ as well as a ranking score s_c, the overall courier set representation is then defined as the score-weighted average of courier representations:

$$\sum\nolimits_{\forall c \in C^{\text{candidate}}} s_c \widetilde{\mathbf{h}}_c. \tag{11}$$

5.2 Regression Module

The regression module used in our model is a simple Multilayer Perceptron (MLP) with two hidden layers of fully connected ELUs.

5.2.1 Postprocessing. The components introduced so far already form a fully functional model that can be used for training and inference. However, in our attempts to train this model, we found it converged slowly and performed worse than expected. We traced the issue to the fact that there is a significant distributional mismatch between the true OFCT values and those predicted by the model at initialization (See Figure 6). As a result, in addition to identification of signals that are useful to OFCT prediction, the model also needs to learn from data to correct this distributional bias during the training process, which results in higher learning burden and thus slower convergence speed.

Let the output of the regression module for $\forall o \in O$ be denoted as y_o^{out} , we tackle the aforementioned problem by simply adding a postprocessing layer that rescales and shifts y_o^{out} as:

$$\text{OFCT}_o^{\text{predict}} = \mu^{\text{real}} + \sigma^{\text{real}} \frac{y_o^{\text{out}} - \mu^{\text{ini}}}{\sigma^{\text{ini}}}, \tag{12}$$

where μ^{real} and μ^{ini} are respectively the means of true OFCT values and results given by the regression module at initialization, while σ^{real} and σ^{ini} are respectively their standard deviations. These statistics are computed over the entire training set. It is easy to see that by using (12), the predictive results of the model with initial weights will have the same mean and variance as the ground truth OFCT values, and thus their distributional difference is reduced.

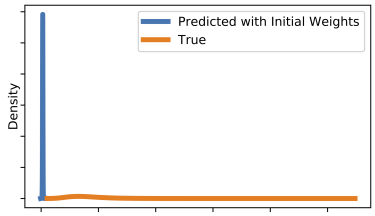

Figure 6: Comparison between distributions of the true OFCT values and those predicted by the model at initialization.

5.3 Objective Function

For each order o in the training set, we jointly minimize the *ranking loss* and the *predictive loss* with a weight parameter λ:

$$\ell_{\text{predict}} + \lambda \ell_{\text{rank}}, \tag{13}$$

where ℓ_{predict} measures the error of predicting OFCT via mean absolute error (MAE):

$$\ell_{\text{predict}} = \left| \text{OFCT}_o^{\text{predict}} - \text{OFCT}_o \right|, \tag{14}$$

and ranking loss simply measures the the error of predicting c_o via the cross entropy loss:

$$\ell_{\text{rank}} = -\sum\nolimits_{\forall c \in C^{\text{candidate}}} (\mathbb{I}(c = c_o) \log(s_c)), \tag{15}$$

where $\mathbb{I}(\cdot)$ is the indicator function which returns 1 if the input argument is true and 0 otherwise.

6 EXPERIMENTS

In this section we evaluate and analyze the performance of our proposed model. How to determine various parameters in our model is discussed in Appendix A.1.

6.1 Offline Experiments

6.1.1 Dataset. The evaluation dataset is constructed using historical orders placed at Ele.me, of which one month's orders are used for training and orders of the following week are used for testing.

6.1.2 Evaluation Metrics. Alongside MAE defined in (14), we also adopt Mean Absolute Percentage Error (MAPE) as an additional metric:

$$\text{MAPE}=\frac{1}{|O_{\text{test}}|}\sum_{\widehat{o}\in O_{\text{test}}}\frac{\left|\text{OFCT}_{\widehat{o}}^{\text{pred}}-\text{OFCT}_{\widehat{o}}\right|}{\text{OFCT}_{\widehat{o}}}, \tag{16}$$

6.1.3 Ablation Analysis of Feature Groups. In this section, we evaluate the contribution of individual feature groups to the performance of the full model. The results are shown in Table 2: for each group of features, we train a new model on the same dataset, using all features except for those in the group. We then report the relative change of the performance metrics of each new model with respect to the original model that uses all features. These results demonstrate all of the feature groups that we use capture non-redundant information about OFCT, and order information and the demand-and-supply features are the most important ones.

Table 2: Relative changes of performance metrics for models trained using all but the ablated feature group, compared to the original model trained with all features.

Ablated Feature Group	MAE	MAPE
Order Information	+11.2%	+15.9%
Demand-and-Supply	+8.0%	+11.9%
ETA-based Drop-off Time	+7.0%	+9.1%
Cooking Time	+6.1%	+10.1%
Courier	+4.5%	+5.2%
Meteorological	+2.9%	+3.3%
Aggregation	+1.8%	+2.5%
Dish	+0.9%	+2.9%

6.1.4 Comparisons between Different Models. Here, we evaluate the performance of different models, which include:

- **GBM** is the previous version of model that we adopt for OFCT prediction. LightGBM [15] is used here for model fitting, as it could efficiently handle the categorical features that we build. Note that courier and bags-of-ids features are not used in GBM model as it cannot handle these features.
- **Deep Factorization machine (DeepFM) [10] and eXtreme Deep Factorization Machine (xDeepFM) [20].** In our model, the vector representations of different fields are simply concatenated and fed into the regression module. On the other hand, DeepFM and xDeepFM are two recently proposed methods that could model complex interactions between features in different fields. It would be interesting to investigate whether the accuracy can be improved by adopting these more sophisticated models. DeepFM and xDeepFM use the same features as our model.
- **TEMP+R [31] and MURAT [19]** are two OD-based ETA methods. As mentioned in Section 2.2, these two methods cannot be directly applied here since the origins of delivery processes are not known. How we modify these methods for predicting OFCT are discussed in Appendix A.3.

The overall comparison results are given in Table 3, from which we can see that:

- ETA-based methods perform much worse than other methods. This is not surprising, as our previous analysis clearly shows that OFCT is influenced by many factors that are not considered in general ETA methods;
- Our model significantly outperforms GBM, which demonstrates the usefulness of neural networks for modeling heterogeneous data;
- DeepFM and xDeepFM do not outperform our model which uses the simple concatenation&MLP strategy. A possible reason is that these methods are primarily designed for applications such as click-through-rate (CTR) prediction and recommendation system, and may not be suitable for predicting OFCT;
- The postprocessing layer slightly improves the model performance. Meanwhile, Figure 7 also shows that this layer substantially accelerates the training progress. These observations are consistent with previous findings that demonstrate the importance of initialization for neural network training [9].

Table 3: Relative performance of various methods compared to the GBM baseline. The lower the better.

Model	MAE	MAPE
GBM	0%	0%
TEMP+R	+39.8%	+46.5%
MURAT	+32.8%	+45.7%
DeepFM	-10.2%	-11.6%
xDeepFM	-10.1%	-12.0%
Our Model without Postprocessing	-10.4%	-12.2%
Our Model	-11.1%	-14.4%

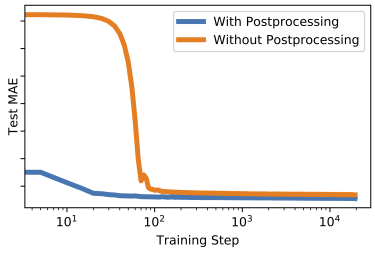

Figure 7: The test accuracy of the our model trained with and without the postprocessing layer, versus the number of training steps.

6.2 Online A/B Test

Motivated by the encouraging offline evaluation results regarding the proposed model, we decide to test the its performance online. A/B test is performed by splitting grids in 50/50 ratio. Compared with those served by the original GBM model, test grids served by our new model have a 19.3% significant decrease in number of customer complaints about the inaccurateness of OFCT estimates and a 9.8% decrease in average predictive error (Table 4). This data indicates that customer satisfaction is significantly improved by our model.

Table 4: A/B Test of the proposed model compared to the GBM baseline.

Metric	Change
MAE	-9.8%
Customer complaints about OFCT estimates	-19.3%

7 CONCLUSION

In this paper, we introduce various aspects of OFCT prediction model that is currently deployed at Ele.me. Extensive offline and online experiments demonstrate the effectiveness of the introduced approach. There are several directions in which we intend to extend this work. Firstly, feature ablation studies in Section 6.1.3 demonstrates that cooking time features are quite helpful for OFCT prediction, however, our current approach for modeling cooking time is still quite simplistic and leaves much room for improvement. Secondly, it would be interesting to investigate whether prediction efficacy can be improved by using better neural model architectures.

REFERENCES

[1] Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research* 218, 1 (2012), 1 – 6.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *NIPS*. 2546–2554.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*.

[4] Daxue Consulting. 2019. The food delivery market in Great China in 2019. Retrieved September 1, 2019 from https://daxueconsulting.com/o2o-food-delivery-market-in-china/

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.

[6] Teodor Gabriel Crainic, Nicoletta Ricciardi, and Giovanni Storchi. 2009. Models for evaluating and planning city logistics systems. *Transportation Science* 43, 4 (2009), 432–454.

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

[8] Uber Eats. 2019. Uber Announces Results for Third Quarter 2019. Retrieved December 20, 2019 from https://investor.uber.com/news-events/news/press-release-details/2019/Uber-Announces-Results-for-Third-Quarter-2019/

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

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

[11] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C Turnbull, Brendan M Collins, et al. 2019. Applying deep learning to Airbnb search. In *SIGKDD*. 1927–1935.

[12] Yixiao Huang, Lei Zhao, Warren B Powell, Yue Tong, and Ilya O Ryzhov. 2019. Optimal Learning for Urban Delivery Fleet Allocation. *Transportation Science* 53, 3 (2019), 623–641.

[13] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *TPAMI* 33, 1 (2010), 117–128.

[14] J. Johnson, M. Douze, and H. JšŠgou. 2020. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* (2020), in press.

[15] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*. 3146–3154.

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

[17] Wang-Chien Lee, Weiping Si, Ling-Jyh Chen, and Meng Chang Chen. 2012. HTTP: a new framework for bus travel time prediction based on historical trajectories. In *SIGSPATIAL*. 279–288.

[18] Xiucheng Li, Gao Cong, Aixin Sun, and Yun Cheng. 2019. Learning Travel Time Distributions with Deep Generative Model. In *WWW*. 1017–1027.

[19] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *SIGKDD*. 1695–1704.

[20] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*. 1754–1763.

[21] Vitória Pureza and Gilbert Laporte. 2008. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. *INFOR* 46, 3 (2008), 165–175.

[22] Raghav Ramesh. 2018. How DoorDash leverages AI in its world-class on-demand logistics engine. Retrieved December 15, 2019 from https://conferences.oreilly.com/artificial-intelligence/ai-ny-2018/public/schedule/detail/65038/

[23] Matteo Salani and Maria Battarra. 2018. The opportunity cost of time window violations. *EURO Journal on Transportation and Logistics* 7, 4 (2018), 343–361.

[24] Remy Spliet and Adriana F. Gabor. 2015. The Time Window Assignment Vehicle Routing Problem. *Transportation Science* 49, 4 (2015), 721–731.

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

[26] Anirudh Subramanyam, Akang Wang, and Chrysanthos E Gounaris. 2018. A scenario decomposition algorithm for strategic time window assignment vehicle routing problems. *Transportation Research Part B: Methodological* 117 (2018), 296–317.

[27] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A unified approach to route planning for shared mobility. *VLDB* 11, 11 (2018), 1633–1646.

[28] Marlin W. Ulmer and Barrett W. Thomas. 2019. Enough Waiting for the Cable Guy–Estimating Arrival Times for Service Vehicle Routing. *Transportation Science* 53, 3 (2019), 897–916.

[29] Anastasios D Vareias, Panagiotis P Repoussis, and Christos D Tarantilis. 2017. Assessing customer service reliability in route planning with self-imposed time windows and stochastic travel times. *Transportation Science* 53, 1 (2017), 256–281.

[30] Thibaut Vidal, Gilbert Laporte, and Piotr Matl. 2020. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research* 286, 2 (2020), 401 – 416.

[31] Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2019. A simple baseline for travel time estimation using large-scale trip data. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 19.

[32] Zi Wang. 2019. Time Predictions in Uber Eats. Retrieved December 15, 2019 from https://www.infoq.com/articles/uber-eats-time-predictions/

[33] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to estimate the travel time. In *SIGKDD*. 858–866.

[34] Zheng Yang, Chenshu Wu, and Yunhao Liu. 2012. Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention. In *MobiCom*. 269–280.

[35] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI*.

[36] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM*. 497–506.

[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 *SIGKDD*. 1059–1068.

[38] Lin Zhu, Yihong Chen, and Bowen He. 2019. A Domain Generalization Perspective on Listwise Context Modeling. In *AAAI*. 5965–5972.

A APPENDIX

A.1 Parameter Selection

In our neural model, the embedding size were fixed as 10, and the model is trained with the Adam algorithm [16] with a learning rate of 0.001, batch size of 4096. Training generally converged after one pass through the entire training dataset.

For hyper-parameters k, $\mathbf{w}$, and σ in Section 4.7, we select the values that minimize the discrepancies between (10) and the ground-truth DTs for a held-out validation set, measured by the mean absolute error (MAE) metric. The resultant cost function, which is neither continuous nor differentiable with respect to k and $\mathbf{w}$, is minimized using Tree of Parzen Estimators (TPE) [2], a derivative-free Bayesian optimization algorithm.

α in Section 4.4 is set as 5 minute.

The size of $C^{\text{candidate}}$ in Section 5.1.3 is fixed as 10.

A.2 Courier Ranking and Pooling

To enhance reproducibility, in this section we provide self-contained description about how to obtain the vector representation $\widetilde{\mathbf{h}}_c$ and the ranking score s_c for $\forall c \in C^{\text{candidate}}$ via QILCM. Detailed explanations of these formulations and experimental evaluations of QILCM can be found in [38].

For $\forall c \in C^{\text{candidate}}$, let the features defined in Section 4.6 be combined into a feature vector as $\mathbf{x}_c$, we first pass it through two layers of ELUs:

$$\begin{aligned} \mathbf{x}_c^{(1)} &= \text{ELU}\left(\mathbf{W}^{(1)}\mathbf{x}_c + \mathbf{b}^{(1)}\right), \\ \mathbf{x}_c^{(2)} &= \text{ELU}\left(\mathbf{W}^{(2)}\mathbf{x}_c^{(1)} + \mathbf{b}^{(2)}\right). \end{aligned} \tag{17}$$

A new feature vector $\mathbf{h}_c$ is then obtained as:

$$\mathbf{h}_c = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_c^{(2)} \end{bmatrix}. \tag{18}$$

The attention distribution is then generated by feeding $\{\mathbf{h}_c\}_{c \in C^{\text{candidate}}}$ into a multilayer perceptron MLP_{att} and then a softmax function:

$$a_c = \frac{\exp\left(\text{MLP}_{\text{att}}\left(\mathbf{h}_c\right)\right)}{\sum_{\widehat{c} \in C^{\text{candidate}}} \exp\left(\text{MLP}_{\text{att}}\left(\mathbf{h}_{\widehat{c}}\right)\right)}, c \in C^{\text{candidate}}. \tag{19}$$

Based on (19), the context embedding $\mathbf{h}_{C^{\text{candidate}}}$ is defined as:

$$\mathbf{h}_{C^{\text{candidate}}} = \sum_{c \in C^{\text{candidate}}} a_c \mathbf{h}_c. \tag{20}$$

(20) is then combined with the original courier representations to form the refined representation vectors as:

$$\widetilde{\mathbf{h}}_c = \begin{bmatrix} \mathbf{h}_{C^{\text{candidate}}} \odot \mathbf{h}_c \\ \mathbf{h}_c \end{bmatrix}, c \in C^{\text{candidate}}, \tag{21}$$

where $\odot$ denotes the Hadamard product. Meanwhile, to generate the ranking score, we use a customized-designed query-normalization layer to process $\widetilde{\mathbf{h}}_c$:

$$\overline{\mathbf{h}}_c = \left(\widetilde{\mathbf{h}}_c - \overline{\mathbf{h}}_{C^{\text{candidate}}}\right) \odot \left(\overline{\boldsymbol{\sigma}}_{C^{\text{candidate}}} + \varepsilon\right)^{-1}, \tag{22}$$

where ε is a small positive constant to avoid division by zero, $\overline{\mathbf{h}}_{C^{\text{candidate}}}$ and $\overline{\boldsymbol{\sigma}}_{C^{\text{candidate}}}$ are defined as:

$$\overline{\mathbf{h}}_{C^{\text{candidate}}} = \sum_{c \in C^{\text{candidate}}} a_c \widetilde{\mathbf{h}}_c, \tag{23}$$

and

$$\overline{\boldsymbol{\sigma}}^2_{C^{\text{candidate}}} = \sum_{c \in C^{\text{candidate}}} a_c \left(\widetilde{\mathbf{h}}_c - \overline{\mathbf{h}}_{C^{\text{candidate}}}\right)^2. \tag{24}$$

The final ranking score is then obtained as

$$s_c = \frac{\exp\left(\text{MLP}_{\text{rank}}\left(\overline{\mathbf{h}}_c\right)\right)}{\sum_{\widehat{c} \in C^{\text{candidate}}} \exp\left(\text{MLP}_{\text{rank}}\left(\overline{\mathbf{h}}_{\widehat{c}}\right)\right)}, c \in C^{\text{candidate}}. \tag{25}$$

A.3 Modifications of ETA Methods for OFCT Prediction

As the starting location of the delivery process is not available as input, we simply use the associated restaurant as the origin for TEMP+R and MURAT. However, the learning target is still kept as OFCT. We generally adopt the parameter settings specified in the original papers. Meanwhile, MURAT requires definitions of a set of auxiliary tasks, and we adopt the following tasks for OFCT prediction:

- The total travel distance of courier for delivering the order;
- The number of road segments contained in the courier's actual itinerary;
- The pickup time;
- The dropoff time;
- The number of traffic lights in the courier's actual itinerary;
- The selected courier.