Test Report

Lei Linfei

Feburary 5, 2021

1 Algorithm Design and Implementation

Base on the idea of synergy decision for platform, I integrate the result of Recommender System (RS) into the component of Order Dispatch and Routing (ODR). And similarly, the RS will consider the result from the ODR, and the final listing that it shows to the user can be re-organized to better coordinate with the ODR. I'll present each component respectively and address the synergy separately. Because of the time limitation, .

1.1 Recommender System

The RS usually include two phases, the first phase is called Recalling, which is to generate candidates from millions of items, in our case, restaurants. With the candidates generated, the second phase, called Ranking, will further rank the candidates to better fit the user's interests and other metrics. The major difference between the two phases is that the Calling involves calculation with millions of restaurants, so the model has to be designed efficiently to make the calculation more tractable. The ranking normally predicts Click-Through Ratio (CTR) or Conversion Ratio (CVR) among the candidates, so the labor of computation is much lighter, which allows a more complex model to incorporate more information to improve its performance.

1.1.1 Implementation

Since we focus on the synergy decision system, and what we need from the RS to make the synergy decision is the ranking results, which is the CVR matrix, so I implement a simple Ranking model to obtain this information.

Specifically, given a user and a restaurant, we need the model to predict the corresponding CVR. To achieve this, I embed the users and the restaurants into the vector space. And for each user and restaurant pair, I concatenate the two embeddings of the pair as one and input it to the two layers networks, then we can use the network to predict the CVR for this pair. The embeddings are trained with the network.

I generate the ground-truth CVR for each pair of user and restaurant by assigning a random number sampled between 0 and 1. Then I normalize the ratio for each user to generate a new matrix, each element in this new matrix is the probability of click for each user and restaurant pair. Once we obtain these two matrixes, we can generate the training test for the RS as the following: for each user, we simulate 1,000 clicks, and for each click, we sample the clicked restaurants and the conversion, which is placing an order in our case, by using the two matrixes we generate before. So one training

data is a triple-element tuple, the first two elements are users' id and restaurants' id respectively, the third element is a binary variable which is 1 when the order is placed, and 0 otherwise.

I simply evaluate the model by calculating the cosine distance between the ground-truth CVR vector and the predicted CVR vector for each user. Figure 1 is the performance of convergence for the model.

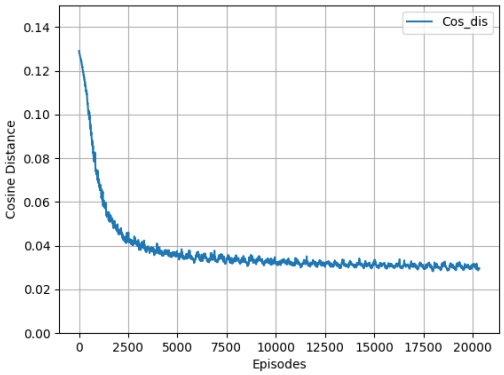


Figure 1: Convergence of ranking model)

1.1.2 Synergy

Since the ODR still on adjustment and the synergy can not be achieved without its results, so below is the description for the idea of the synergy decision in RS.

Except to increase the metrics like CTR or CVR like the normal RS would do, we also want the RS to adjust the recommendation so it can guide the user to choose the restaurants that are not only appealing to the user but also cost-friendly for the whole system, especially for the ODR. For example, we can assume the CVR is the probability of order generated by the user, so by adding those potential orders to the current state of the system, we can derive the future cost by using the value function which we will obtain from the ODR. And by integrating this cost during the Ranking, we can achieve this synergy decision in which we not only care for the CVR but also the potential cost during the delivery.

1.2 Order Dispatch and Routing

During the dispatch and routing for the riders, we aim to optimize the cost in the view of the long term, so it is crucial that we consider the cost for now as well as the potential cost for the future during the decision. One way to solve this problem is to view the future cost as a function, we repeatedly approximate this function by offline training. Once we have the approximate function, we can use the function to help the decision in real-time. We assume that the request for a decision comes with the beginning of every time slice, and once the request arrives, we need to make the decision quickly and wait till the next request arrives.

The ODR is modeled as a Markov decision process, for every time step, the ODR will dispatch the newly generated order and routing for each courier base on the current state.

1.2.1 Implementation for Order dispatch

During the dispatch, we need to assign a batch of orders to a fleet of riders, so the quality of the service and the delivery cost can be optimized. Three requirements need to be fulfilled during the dispatch, (1) a feasible solution must be found in a short time, (2) the quality of the solution should be acceptable, (3) future cost must be considered. Aim to solve those questions, we can adopt heuristics like Adaptive Large Neighborhood Search to generate a feasible and good solution in seconds, and incorporate the approximate function we mentioned before into the evaluation function of the heuristics. Thus, We can achieve foresighted, real-time, and quality dispatch.

For the quick iteration of the algorithm design and implementation, I adopt greedy insertion[2] during the dispatch, by doing this, I can get a fast and feasible dispatch during the training, and it will be much easier for me to evaluate the performance of routing.

1.2.2 Implementation for Routing

For each state, I sequentially decide the next position the courier needs to visit for all couriers. To guarantee security, the destination for the en-route couriers will not be modified, only the position next to the destination will be decided. During the decision, I use value function approximation to obtain the future cost for each decision.

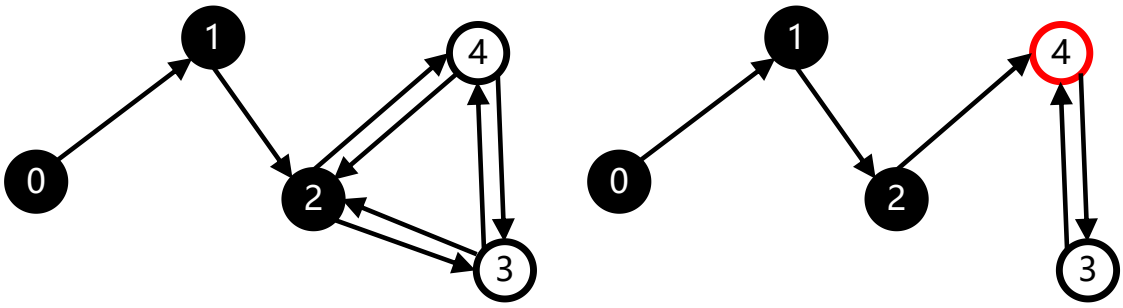
For one decision in the routing, we need to evaluate many sub-decisions for each courier, which is the selection of every accessible position for one courier. To evaluate each accessible position of one courier, I will generate the features for this sub-decision, then input the features into the value function to predict the future cost, for the sub-decision selection, I implement $\varepsilon - greedy$, $softmax$, and greedy selection to select the sub-decision during the evaluation. A decision is obtained once the evaluation for all couriers' sub-decisions is finished.

To obtain valuable features for the sub-decision, I will define a sub-graph which is containing the sub-decision information based on the global graph and the sub-decision, then I use Graph Neural Network(GNN) to embed this sub-graph into vector space. For example, the global graph for the current state is showed in Figure 2(a), the couriers' current position is node 2, and node 3 and 4 is the accessible position, we assume the sub-decision is to assign node 4 as the couriers' next position, and we define the sub-graph for this sub-decision as Figure 2(b). The rule is to make sure the connectivity for the decided route is captured, as for the unassigned positions, we assume the full connectivity for them. By doing this, we can use the sub-graph defined above as input for GNN to generate the graph embedding as information of graph structure for the following.

I also extract some manual features about time information in the current sub-graph and concatenate it with the graph embedding as the input for the value function. By doing this, I can easily generate features for the value function, so the value function can both have the sub-decision information and the global graph information to ensure a more reasonable prediction for the future cost of this sub-decision.

As for the value function approximation, I use a two-layer neural network to predict the future cost. To train the network, I store the post-decision state and reward for each time step, once the simulation is finished, I follow the feature generation procedure described above, to perform the generation on the post-decision state. Also, I use the summation of sampled reward from this post-decision state and the following reward as the label for this training data, which is the future cost for this post-decision state.

The GNN I used here is adopted from[1], it will strictly follow the connectivity of the graph but also pass the message in the global view of the graph, which is a very important feature in our



(a) Graph for current state.

(b) Sub-graph for the sub-decision.

Figure 2: Example of graph definition)

case since we would like the information from early routing can also have an impact for the current routing. I implement a two-layer GNN and simply average all node embeddings outputted by the GNN into one embedding and use it as the graph embedding.

1.2.3 Synergy

To incorporate the result from RS into the ODR, We can sample the orders in the simulations for ODR according to the CVR matrix predicted by the RS. I adopt this kind of synergy in the implementation.

Also, we can view the result of RS as order prediction, by taking a few steps further on the timeline, there will be some new users request to make the order, we use the recommendation result for the new users as their weighted potential orders, where the weights are their CVR respectively. So every time we use GNN to predict the future cost, we use the information of the orders prediction as additional input for the GNN, so the GNN could have more information to generate a more farsighted embedding, and improve the performance of prediction for future cost.

1.3 Experiment Design

Since the ODR cannot perform well and still in the adjustment, so the experiments will not be carried out. Below is preliminary design for the experiments.

1.3.1 Synergy Evaluation

We can evaluate the performance for RS and ODR respectively by running the experiments with and without the synergy. For the ODR, there will be three versions to evaluate, one is synergy with simulation, synergy with order prediction.

1.4 Ablation Experiments for ODR

For ODR, we can test each king of synergy in a fashion of ablation. Also, during the feature generation, we can evaluate the performance brought by the GNN and see how well the prediction will be.

1.4.1 Scalability Evaluation for ODR

To test the scalability of the model, we can define several environment setting for the test, like the number of couriers, the number of orders, etc..

1.5 Discussion

From the research above, I found that for each component, there exist some questions that are needed to be addressed. For the RS, I re-ranking the result by considering the future cost of the immediate assignment for the near riders, this is done by the assumption that the order is immediately generated. If there is a model that can predict the time the users need to make the order, and a model to predict the position of the riders, then the assumption can be eliminated and the future cost can be estimated accurately. There is some work done by the Uber[1], they use the GPS data and motion data to predict the rider's status so they can collect some time information about the restaurants and the riders, with the time information, they can further predict other information like preparation time for the restaurants, etc. Take steps alone these paths, we can also use the trajectory of the riders to predict their position after a certain time from a certain location. As for the hesitating time for the user, by using the historical data about the timing they open the application and the timing they place the order, the model of prediction can be learned. The prediction for riders and the users can also help the ODR since the Routing also make the assumption about the potential orders.

2 Model Analysis

I haven't got the time to model the problem mathematically. So the following are just some rough thoughts about the question listed in the test.

2.1 Analytical Result and Property

The first property is about the RS. The objective of the RS now has two parts, including the user's interest and conversion, and the consideration for the ODR. If we can derive a property that the RS's performance will not be damaged, or the damage is much less than the benefit brought by the synergy decision in terms of the interest of the system. Then the idea of synergy is more promising. As for the ODR, the result of RS only make the offline environment becomes more realistic, so if we can prove that the synergy for the ODR can improve the performance of itself, which I believe there is, then the superiority of the synergy is proved. Also, an increase in the interest of the system brought by the synergy is expected to be derived.

2.2 Model Calibration and Validation

For model calibration and validation, the orders for a certain period, detailed information about the restaurants and the users in the platform are needed. If we can obtain a completed dataset from a food delivery company, we can use those data to directly calibrate the model to achieve good performance like DiDi does[3]. And we can divide the dataset into two parts as the training set and test set, we calibrate the model on the training set and validate the model on the test set. But if we can only get open-source data, which the reliability of the data is much lower than the dataset

[1] https://eng.uber.com/uber-eats-trip-optimization/

aforementioned, we can analyze the open-source dataset to obtain some insight about the data so we can generate the data by ourselves. The extra data we generated can be viewed as noised data, by using these data we can reduce affection for the model caused by the open-source dataset. Or we add the noise directly into the open-source dataset to achieve the reduction of affection.

2.3 Implications for Operation

Companies like Meituan, DiDi, etc. normally test the model by using the A/B test, which can evaluate the model accurately without introducing appreciable influence into the business operation. Once the model is reliable, it can be launched completely on the platform with manual support in some extreme cases. With the launch of the model, we can derive some business metrics to obtain some implications such as that if the human resources are enough, if the order peak for certain restaurants arrives, or if some users care more about the flavor rather than waiting time.

2.4 Discussion

The assumption we made during the modeling may affect the model calibration and validation, and the assumption cannot be the key part of the model and it should be reasonable. So the evaluation of the assumption is important before the modeling. During the calibration, the belief about the data needs to be addressed so that the calibration can be performed at an appropriate level. Design of metrics for the validation is essential too, with the well-designed metrics we will be able to tell whether the performance is achieved through the benefit from the data or the superiority of the model, so the validation is more credible.

References

[1] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

[2] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006.

[3] Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 905–913. ACM, 2018.