CRSRL: Customer Routing System Using Reinforcement Learning

Chong Long[1] , **Zining Liu**[2] , **Xiaolu Lu**[3] , **Zehong Hu**[4] and **Yafang Wang**[1*]
[1]Ant Financial Services Group, Z Space No. 556 Xixi Road Hangzhou, 310099, Zhejiang, China
[2]School of Software, Shandong University, Jinan 250101, Shandong, China
[3]RMIT University, GPO Box 2476, Melbourne VIC 3001, Australia
[4]Alibaba Group, 969 West Wen Yi Road, Hangzhou 311121, Zhejiang, China
{huangxuan.lc, yafang.wyf}@antfin.com, liuzining@mail.sdu.edu.cn, xiaolu.lu@rmit.edu.au,
zehong.hzh@alibaba-inc.com

Abstract

Allocating resources to customers in the customer service is a difficult problem, because designing an optimal strategy to achieve an optimal trade-off between available resources and customers' satisfaction is non-trivial. In this paper, we formalize the customer routing problem, and propose a novel framework based on deep reinforcement learning (RL) to address this problem. To make it more practical, a demo is provided to show and compare different models, which visualizes all decision process, and in particular, the system shows how the optimal strategy is reached. Besides, our demo system also ships with a variety of models that users can choose based on their needs.

1 Introduction

The customer requests routing (or dispatching) problem is crucial but difficult to a company, because the satisfaction of customers may be affected by many aspects. Take customer service call center for example, customer's satisfaction is measured beyond problem-solving quality: the customers' queuing time also needs to be considered. For the purpose of reducing congestion, major companies often provided multiple communication channels for customers to choose, for example, mobile App, web-based self-service message and the traditional call center (or hotline service). Different communication channels have their own limited quota for responding to customers' requests. While the hotline channel is preferred by most customers, it is usually very crowded at peak time. Also, some customers are willing to be directed to another communication channel to avoid a long waiting time.

This paper focus on two aspects: firstly, we propose a deep-reinforcement-learning [Sutton and Barto, 2018] based framework to perform the customer service requests routing. The proposed framework is in sharply contrast to the rule-based system, as our framework directly captures customers' preference and each channel's future traffic. Second, we build a demo[1] where users are free to switch and compare various supervised and RL models.

*Corresponding author

[1]See https://youtu.be/aLAjuTW8MWQ for the demo video.

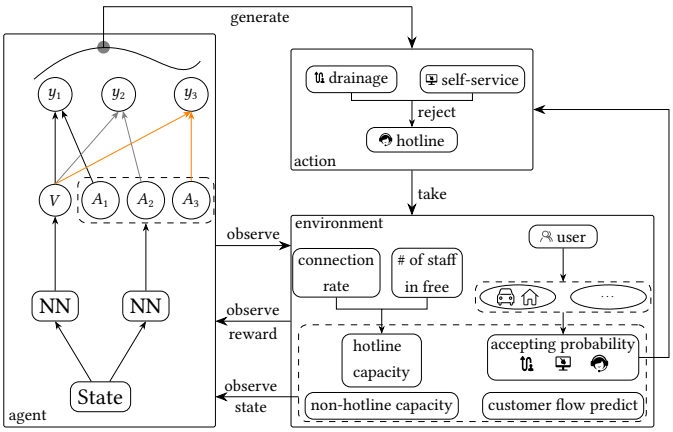

Figure 1: Overview of the proposed routing framework.

2 The CRSRL Framework

In this paper, we consider the multi-channel customer service setting and formulate the customer requests routing problem as follows. Let n be the total number of communication channels, each of which has a capacity of $c_i (1 \leq i \leq n)$. Assume all customers will use the customer service platforms. Each customer has a user profile $\mathbf{u} = \langle f_0, f_1, \ldots, f_k \rangle$, where f_i represents a value of i-th attribute. We consider at time t, each channel will also have a request flow, referred to as e_t. The utilization rate v_i to represent the percentage of the free resources of the current channel, relative to the channel capacity. Formally, the routing problem is: Given a user and the request, and the current capacity of each channel c_i, the problem is to recommend user a communication channel i, such that: (i) the overall users' satisfaction rate is maximized; (ii) the overall utilization rates of communication channels are maximized.

An overview of our proposed customer service routing framework is shown in Figure 1, which has two major components: the environment and the agent.

Environment. The environment is shown in the right bottom pane of Figure 1, which consists of a channel model (the left-hand side in the pane) and a user model (the right-hand side in the pane). We use the channel model to estimate the request flow volume, which is a part of the states that

the agent can observe: we should avoid allocating more customers to a channel which is expected to have a large number of incoming requests in the next time window. Since the customer service request show a strong temporal correlation, we formulate the problem of channel requests predication as a time-series prediction problem. Moreover, as in peak hours, the data stream may come in with a high volume and velocity, so pre-estimating the number of the incoming requests helps the system to make a better routing decision. Another important element of the environment is the user model, describing customers' preference over different communication channels. Intuitively, customers preferences vary on all possible communication channels. If the recommended channel is not preferred by a customer, then it is highly likely the customer will reject the channel routing, which leads to a longer waiting queue of a popular channel, for example, the hotline channel. We build the user model using a feed-forward neural network: for each user, we input their attributes into the user model and output their acceptance probability of each communication channel.

Agent. The agent which takes actions based on observed states is shown in the leftmost pane in Figure 1. It takes the state observed from the environment as the input and then a neural network is used to recommend a channel to the customer. If a customer accepts our recommendation, then a successful routing is performed, which indicates we may reduce the workload of some bottleneck channels to some extent and also the user is willing to accept a channel redirection.

Besides, the overall routing model is based on deep reinforcement learning, and more specifically, the DQN[Mnih *et al.*, 2015] variants

Action. The system aims to learn a policy that can determine which channel should be recommended to users, so the action here is to select a channel among n candidates, where n is determined by the total number of channels in the real application. We represent the action as a, $(a \in \{1, 2, 3, \ldots, n\})$.

State. A state is expected to express the customer's channel preference, and channel feasibility of handling furthermore requests. So the state includes: the customers' preferences over different channels $\mathbf{u}$, the channel capacity $\mathbf{c}$ and the channel's future request flow traffic $\hat{\mathbf{e}}_t$. The entire state is represented as $\mathbf{s} = \langle \mathbf{u}, \hat{\mathbf{e}}_t, \mathbf{c} \rangle$.

Reward Function. There are duel reward aspects we need to consider: from the customers' perspective and from the channel capacity. We use $g_{a,t}$ be the reward that is given to the user's current action at time t, indicating if we make a good recommendation from a customer's perspective, based on which the reward is:

$$R = g_{a,t} - \sum_{i=1}^{2} \lambda_i \cdot (\textit{Relu}\left(\min(\mathbf{c}_t - \lambda_3 \cdot \hat{\mathbf{e}}_{t+1})\right))^i \qquad (1)$$

3 The Demo System

Our demo consists of three parts. Users can firstly input the models, parameters, and initial states on "input page", then monitor the RL routing process on "show page", and finally compare the results of different models on "result page".

Input page. This page is to initialize three types of states: channel capabilities, learning models and scheduled data. We have three common customer service channels: Hotline, Self-Service and Drainage. Learning models are pre-trained offline for several epoches, and submitted to our page. They can either be RL model or general supervised deep learning model. Scheduled data contain a list of "upcoming" customers and their information, including their arrival time, departure time, and profiles u. All the scheduled data are predefined and uploaded so different algorithms can be compared under the same states. Now all the uploaded data in our demo are simulated according to the distribution of our customer service.

Show page. This page is show the whole process of our customer routing system. Customers come one by one according to their arrival time. Then RL system takes the profile as a part of the state. The other parts of the states are the vacancies c (which are calculated according to the maximum capability and the existing online customers), and the estimated customer flow e, respectively. After getting the states, RL system takes actions to recommend one of the channels. The customer decided to accept or reject the recommendation. The decision is simulated according to the value of profile u. As our scenario is under hotline process, we set all the users' acceptance rate of hotline channel to 100%. If a recommended channel is accepted by the customer, its capability will be updated; but if rejected, it means that the costumer sticks to the hotline channel, so the hotline channel's capability is updated instead. Finally, the reward is calculated and presented.

Result page. After finishing all the scheduled customers, the system will calculate the metrics: the acceptance rate, which reflects the recommendation accuracy, and the congestion rate, which are closely related to customer satisfactory. Also historical experiment results are all listed and compared in one result page. The listed history results (models) include Deep Q-Network (DQN)[Mnih *et al.*, 2015], DoDQN[van Hasselt *et al.*, 2016], DDQN[Wang *et al.*, 2016], and those with Experience Replay[Hosu and Rebedea, 2016] and Prioritized Experience Replay[Schaul *et al.*, 2015], etc.

4 Conclusion and Acknowledgements

We formulate the classic customer request routing problem into an optimization problem by considering both channel resources and customers' satisfaction, and then proposed a deep-reinforcement-learning based framework to address the problem. Moreover, we provide a demo system that visualizes the customer service routing process using both traditional and deep RL algorithms. We also notice that there is still a large space for us to improve our routing algorithm, for example: (1) we can get more accurate user preferences through employing multiple dialog communications; (2) our proposed demo system can be generalized to a platform where the bad cases can be detected and fixed.

Finally, we would like to extend our thanks to our contractors Chenfei Yue, Yanxin Song and Desheng Li, for their hard work on building our demo system.

References

[Hosu and Rebedea, 2016] Ionel-Alexandru Hosu and Traian Rebedea. Playing atari games with deep reinforcement learning and human checkpoint replay. *CoRR*, abs/1607.05077, 2016.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Schaul *et al.*, 2015] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[van Hasselt *et al.*, 2016] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proc. AAAI*, pages 2094–2100, 2016.

[Wang *et al.*, 2016] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In *Proc. ICML*, pages 1995–2003, 2016.