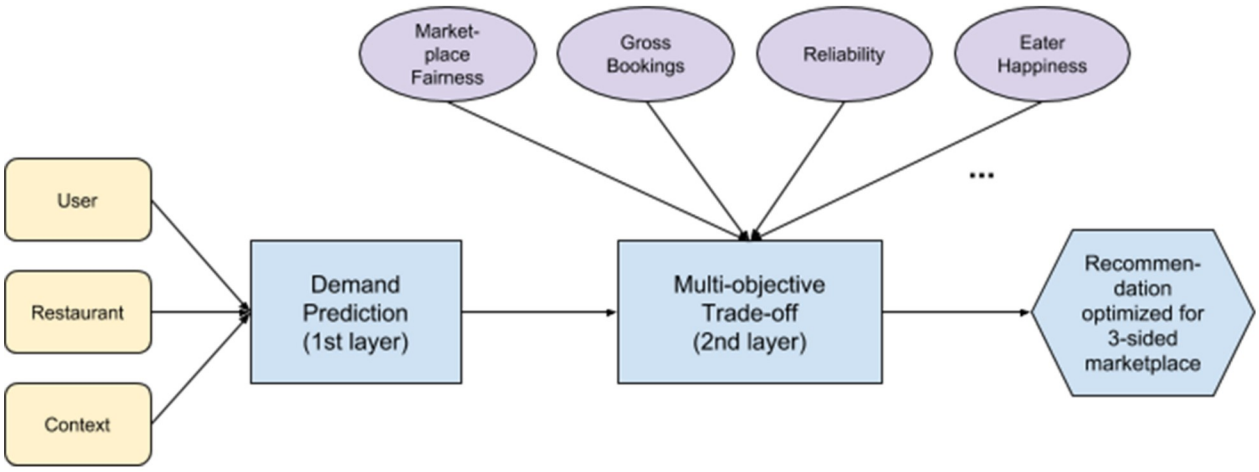
Engineering

Food Discovery with Uber Eats: Recommending for the Marketplace

Yuyan Wang, Yuanchi Ning, Isaac Liu, and Xian Xing Zhang September 10, 2018



Even as we improve Uber Eats to better understand eaters' intentions when they use search, there are times when eaters just don't know what they want to eat. In those situations, the Uber Eats app provides a personalized experience for each individual through restaurant recommendations.

From search results to the list of restaurants and explicit recommendations on the app's homescreen, we tailor the selection and ranking of restaurants to our understanding of what our eaters crave. For example, an eater may notice that if they order a lot of spicy Szechuan food, they will begin to see more Chinese restaurants selling spicy food in the app. At the same time, additional Thai, Japanese, or other Asian recommendations might show up, too.

We generate these recommendations based on both our modeling of each user's taste preference through machine learning and our semantic understanding of food types. These recommendations also serve to support the overall health of the Uber Eats marketplace. Through multi-objective optimization, we can help eaters discover a diverse array of restaurants and ensure that our restaurant-partners receive a fair amount of exposure in the app based on eater interest.

In this two-part series, we look under the hood of the Uber Eats app and walk through the efforts we take to aid eaters in their decision-making process. The first part of the series focused on building a query understanding engine for Uber Eats through an in-house food knowledge graph and representation learning. In this second article, we discuss how we recommend restaurants based not only an eater's

Engineering

Our recommender system's journey

When we launched the first version of the Uber Eats restaurant ranking and recommender system, we were optimizing for a single objective: the eater's probability to order from a restaurant (eater conversion rate). Different types of supervised machine learning models and learning to rank algorithms were built to achieve that purpose. We observed significant business metric lifts when we switched from an offline and non-personalized model to an online and personalized model, validating the importance of real-time features and customization.

However, we soon realized that optimizing only for eater conversion did not provide the best overall experience. As we continued to sign-up restaurant-partners for our platform, we found that new restaurants, even well-known and popular ones, were not getting the same volume of orders as we expected. As restaurant-partners make up one side of our three-sided marketplace, we began exploring how to ensure that all restaurants on our platform get their fair share of exposure, which should result in more orders.

We also needed to optimize for the health and efficiency of the overall marketplace, which also has the side-benefit of exposing eaters to a more diverse selection of restaurants. Since then, we have evolved our system, as we discuss later in the article.

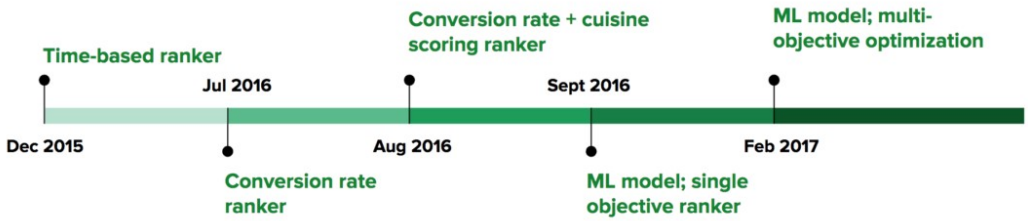


Figure 1: This timeline illustrates our journey and the changes involved in improving our system.

The marketplace

The Uber Eats marketplace consists of three sides: eaters, restaurant-partners, and delivery-partners. Eaters discover and order food through our platform. Restaurant-partners use our platform as a sales channel to find customers. And delivery-partners earn income by picking up food from restaurants and delivering it to eaters.

All sides of this marketplace are equally important to ensure a seamless Uber Eats experience. If there are not enough eaters placing orders, restaurants will not want to participate. If there are not enough restaurants, the selection decreases and fewer eaters will want to order from the platform. If orders

Engineering

On the other hand, if most orders are concentrated to a certain restaurant, for example, due to that restaurant being recommended to all eaters, it will cause problems too. For example, the restaurant may not be able to handle the sudden increase in incoming orders so food preparation may get delayed, or even worse, the quality of the food may be compromised. And even if the restaurant can prepare all orders in time, we may not have enough nearby delivery-partners to pick up the food and deliver it to the eater. So either the order goes unfulfilled or we dispatch a delivery-partner who is too far away to pick up the food in a timely manner, which may cause other cascading problems. As a result, such an unbalanced market demand-supply dynamic degrades the overall Uber Eats experience.

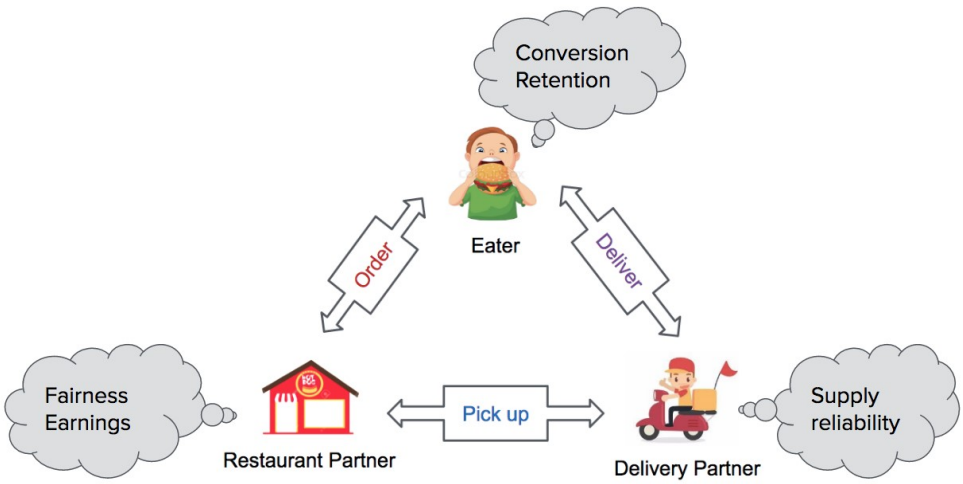


Figure 2 : All parties in Uber Eats' three-sided marketplace contribute to its overall health. (Image credits: Restaurant-Partner: gst/Shutterstock.com, Delivery Partner: Kikuchi/Shutterstock.com, Eater: tigatelu/Depositphotos)

Building a recommender system for a three-sided marketplace introduces a unique set of challenges, as each side of the marketplace has intrinsic and different values that we need to take into consideration. Sometimes the relationship is straightforward. For example, more orders tends to benefit restaurants and delivery-partners since it enables them both to receive more income from orders.

But more often than not, balancing the marketplace entails a subtle trade-off requiring an intricate solution. Our previous example about new restaurants illustrates the issue. By optimizing purely for eater conversion (more eater orders), we might up-rank only restaurants we know our eaters like, leading to newer restaurants to the platform not receiving sufficient orders to keep them on the platform, thereby decreasing restaurant supply. With fewer restaurants, we might also lose potential eaters who do not find sufficient restaurant diversity on our platform.

Moving beyond the goal of eater conversion, we can use restaurant rankings and recommendations to help meet the needs of all sides of our marketplace.

Engineering

Eats is to “make eating well effortless at anytime, for anyone.” How do we provide eaters with an “effortless” experience for finding the right restaurant or dishes for their tastes? How do we ensure that eaters “eat well”? Is the “eating well” criteria the same for different eaters? To tackle these challenges, a lot of effort has been spent on improving the relevance of the Uber Eats app home screen content in a personalized manner.

The current Uber Eats home screen contains heterogeneous types of content: collections of restaurants (portrayed in a carousel format), a plain list of all available restaurants, dishes and cuisines. When an eater opens the Uber Eats app, intelligence on the backend determines how many restaurant carousels should be displayed; what kind of restaurant carousels should be presented to this eater; and how to rank the plain list of all restaurants in order to display restaurants the eater would like to order from at the top.

Many different considerations play a part in what the app displays. If we take a deeper look at how to best rank all the available restaurants for a particular eater, these factors include:

- **How popular is the restaurant itself and what are the best features to determine popularity?** Do we rely on the rating, the historic total orders, or the most recent month’s total orders? The order/impression ratio?
- **What are the most representative attributes of the restaurant?** Does the restaurant prepare and deliver food faster than others? What type of cuisines does it offer?
- **What are the current contextual factors?** Is it breakfast time or dinner time? What are the current traffic conditions along the delivery path? Is it a weekday or a weekend?
- **What attributes best describe the eater?** How many orders has the eater placed so far or in the last month? What kind of cuisines does the eater order most frequently? Is this person a new eater? Did this eater put any dishes into their shopping cart from the restaurant they just clicked into?
- **What factors represents an eater’s preference for a particular restaurant?** Do we look at how often this eater has clicked into/ordered from this restaurant? Did the eater give this restaurant a high rating?
- **What would like-minded eaters order?** How do we represent and find similar eaters? How about similar restaurants and dishes? How do we cold start new eaters in the context of collaborative filtering?

To surface the most relevant restaurants to our eaters in the app, we need to determine how to best select and represent the above factors. Using this data requires a lot experimentation and iterations of different kinds of models, along with various feature engineering techniques. Only through constant effort aimed at answering these difficult questions can we provide the best possible eater experiences.

Diversity

While optimizing restaurant and dish display for relevance to the eater is the obvious approach, it is not

primarily of ramen restaurants, as depicted in Figure 3, below.

Although the eater might really like ramen, such a list gives the impression that Uber Eats only offers deliveries from ramen restaurants, and does not take into account that the eater might not feel like ordering ramen every time they open the app.

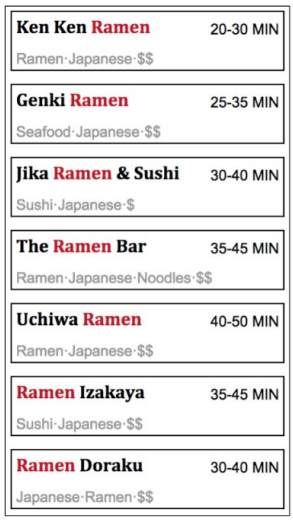


Figure 3: Ranking only by relevance can lead to a lack of variety, such as this list of ramen restaurants.

This problem motivated us to think about another ranking objective: diversity. More specifically, a good recommendation should be both relevant (so that eaters can easily find relevant restaurants) and diverse (so that eaters can explore different types of food as well). To this end, we developed a personalized diversification algorithm that takes the eater's taste profile and restaurant cuisine profile as vector representations of the eater and the restaurant, and optimizes a combined objective of relevance and diversity. Optimization is done in a greedy way in that the restaurant at each position is determined sequentially. The launch of personalized diversification for Uber Eats leads to a significant lift in business metrics.

Keeping eaters on our platform

Our efforts to rank restaurants and dishes by relevance and diversity increase the likelihood of eaters ordering from our platform. But placing an order is only the beginning of their journey. We also want to leverage our recommender system to make sure that eaters have a delightful delivery and dining experience.

This is not a trivial task as the food that an eater orders from a given restaurant may not meet their expectations. For example, an eater may order from a restaurant that looks appealing or has reasonable prices, but if the taste or quantity of the food is not to their liking, or if there is friction in the delivery, the eater will not be satisfied and may not order again from Uber Eats. For this reason, we would like to incorporate signals that correlate with more long-term success into our ranking and recommendation framework. An accurate predictive signal, which preempts the likelihood of an eater ordering from a restaurant they won't like, can help us recommend restaurants that not only attract eaters in the first place, but also help keep them on our platform.

Whether the eater will order again from Uber Eats is a delayed signal, which makes satisfaction more challenging to model than other objectives such as relevance and diversity. To address this challenge, we came up with a machine learning model that predicts this delayed signal, or more specifically, the probability that the eater will order from Uber Eats again if the eater orders from a particular restaurant. The model uses features from both the eater's and restaurant's past order experiences, such as food

Engineering

Restaurant-partners

Maintaining the Uber Eats marketplace also means paying attention to the performance of our restaurant-partners. The popularity of restaurants is, of course, completely up to the eaters on our platform. However, we do want to give restaurants the opportunity to be seen on our platform.

The relevance ranking model for a plain list of restaurants outlined above relies heavily on historic data. Without special treatment, a new restaurant on our platform may have a low ranking since it has no impressions and no historic orders. To give new restaurants a fair opportunity to rank high and gather exposure, we used the multi-armed bandit (MAB) framework.

In this case, we applied the upper confidence bound (UCB, one of the methodologies of MAB) approach to facilitate the exploration of new restaurants or restaurants with low impressions. We calculate a UCB score for each restaurant based on metrics such as its historic impression number, total click number, and boosting factor. The UCB score, along with other objectives discussed above, decides the ranking order among restaurants. A new restaurant will have a relatively high UCB score initially and hence rank highly, increasing its exposure. As the new restaurant gathers more impressions, the UCB score will smoothly decrease and gradually transfer ranking weight back to other objective scores such as relevance.

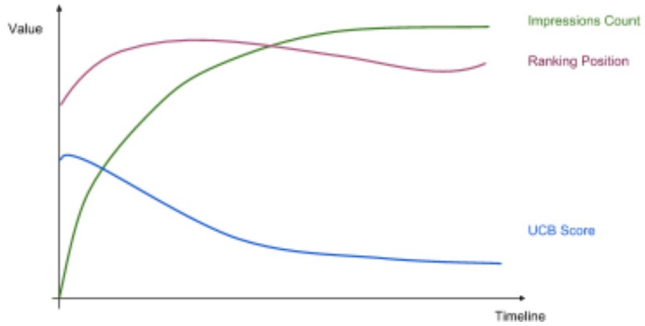


Figure 4: A new restaurant's UCB score decreases over time as its impressions increase.

Delivery-partners

Without delivery-partners, food could not get from our restaurant-partners to eaters, and we would not have a three-sided marketplace. Other teams at Uber have focused on improving the trip experience of our delivery-partners through dispatch optimization. Complementing that work, we have begun to explore how ranking and recommendations can improve the delivery-partner experience.

Earnings are typically the main incentive that motivates delivery-partners, and utilization, the number of

we can't simply use the current location of the delivery-partners because it takes time to prepare the food once the restaurant receives the order, so we would need to use the predicted location at the predicted time when the food will be ready. (Read our article on Michelangelo, our machine learning platform, for a use case on how we use machine learning to estimate Uber Eats orders time of arrival.)

In addition to delivery-partner locations, the pick-up time at restaurants and drop-off times at different locations also impact utilization. In addition to improving the logistics and pick-up experience, we are also exploring ways to effectively leverage this data in ranking and recommendation. For instance, we can potentially incentivize restaurants to improve the delivery-partner experience by down-ranking restaurants with longer pick-up times, which also leads to longer delivery times. These efforts will allow us to discover new ways to improve their experience in our marketplace.

Multi-objective optimization

Although we would love to maximize all of the objectives for all sides the marketplace, in reality, objectives often trade off one another. The most interesting challenge then is how to optimize multiple objectives simultaneously. If two objectives have a strong positive correlation with each other, then optimizing one would automatically optimize the other.

Unfortunately, we find few positive correlations between objectives for the Uber Eats' business. Within the three-sided marketplace, we can easily identify objectives from different sides that simply cannot reach optimality simultaneously. For example, Figure 5, below, shows the trade-off curve between the eater conversion rate objective, getting eaters to place orders, and the marketplace fairness objective, exposing less popular restaurants to more eaters:

If we only optimize for eater conversion, we would probably only surface a few very popular and well-established restaurants to almost all eaters. New or recently onboarded restaurants would not show up on the homescreen for eaters and would get fewer orders, hurting marketplace fairness. On the other hand, if we simply optimize for marketplace fairness, ensuring every restaurant gets equal exposure to every eater, we would probably be recommending some irrelevant restaurants, affecting eater conversation rate.

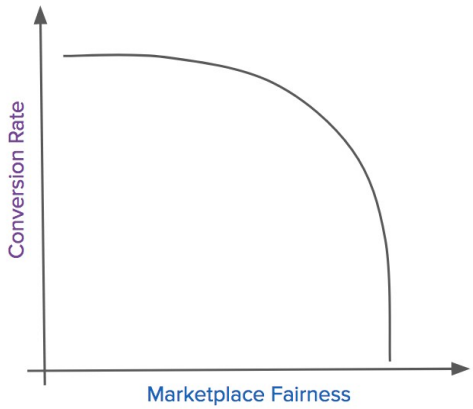


Figure 5: There is a steep trade-off between conversion rate and marketplace fairness.

Multi-objective optimization gives us a mathematically principled solution for the trade-off among competing objectives. More specifically, we developed an optimization framework with quadratic programming. We set constraints on the sacrifice we are willing to

matrix of all instances of x_{ij}. Let I_{ij} be the indicator of restaurant j being recommended to eater i, O_{ij} the indicator of eater i ordering from restaurant j, and G_{ij} the dollar amount of the order between eater i and restaurant j. Then, the expected *total number of orders* is:

$$T(X) = \mathrm{E}[\sum_i \sum_j I_{ij} O_{ij}] = \sum_i \sum_j x_{ij} p_{ij},$$

and the expected *total gross bookings* is:

$$G(X) = \mathrm{E}[\sum_i \sum_j I_{ij} O_{ij} G_{ij}] = \sum_i \sum_j x_{ij} p_{ij} g_{ij}.$$

Here $p_{ij} = E[O_{ij}]$ is the predicted conversion rate between eater i and restaurant j, and $g_{ij} = E[G_{ij}]$ is the expected bookings between eater i and restaurant j. Both p_{ij} and g_{ij} are personalized machine learning models that use features from eaters, restaurants, and real-time contexts. We further define the uniform ranking algorithm $U = (u_{ij})MN$ where $u_{ij} = 1/(MN)$ gives every restaurant equal probability. The optimization is then formalized as

$$\max_X G(X) - \frac{1}{2}\lambda \|X - U\|_F^2$$

$$s.t. \quad T(X) \geq \alpha \cdot T(X^*)$$

$$x_{ij} \geq 0, \forall\, i, j$$

$$\sum_j x_{ij} = 1, \forall\, i$$

where we maximize restaurant gross bookings (with a quadratic penalty) within percent of optimal total orders $T(X^*)$ $(0<\alpha<1)$.

Using Lagrangian duality and KKT conditions, this constrained optimization problem effectively yields the ultimate ranking function for every eater as a combination of different objectives with analytical formula. Moreover, each objective is associated with a coefficient that is a function of the constraint parameters (i.e. in the above example) and can be further tuned.

With tunable coefficients for each objective as free parameters in the multi-objective optimization framework, we then need to efficiently find the optimal weight combination. An intuitive way to accomplish this is through online A/B testing: generate different sets of coefficient values and observe their performance online. However, the exploration space for the coefficients grows exponentially with the number of objectives, and online A/B testing takes too long to iterate and is costly especially given

want to exploit the current best coefficients that have the highest payoff so far, while at the same time explore other coefficients that potentially have a higher return. Bayesian optimization with contextual multi-armed bandit can be applied here. Given the current best set of coefficient values, we also have a holdout of production sessions running with the epsilon-greedy method that generates new sets of coefficients for less biased training data, which is then used for subsequent model training and online optimization iterations.

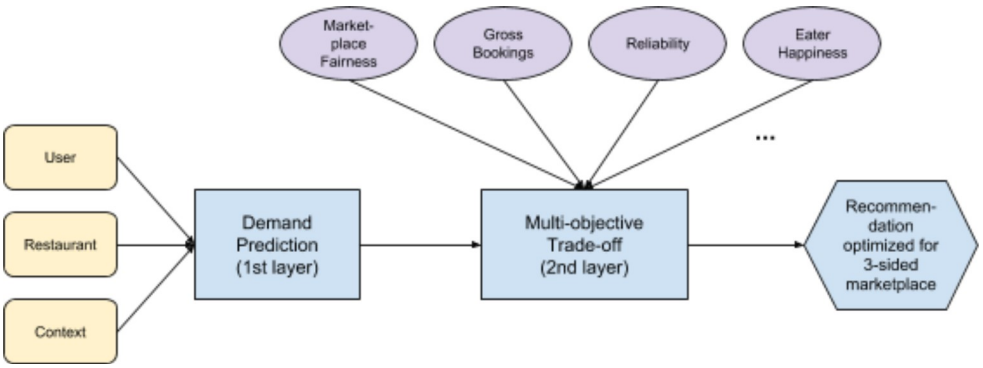


Figure 6: Our two-layer solution combines machine learning with multi-objective optimization.

Figure 6, above, shows a flow chart of our entire ranking framework with multi-objective optimization. In the first layer, we use a machine learning model to calculate demand prediction. The second layer adds other objectives and determines trade-off and optimization. The framework combines personalization and multi-objective trade-off with an elegant mathematical formulation, and the output is a personalized recommendation optimized for the three-sided marketplace.

Moving forward

The Uber Eats Discovery team strives to build the best app experience possible for our customers. In a three-sided marketplace, the customer is not just eaters, but restaurants and delivery-partners as well. In this article, we discussed how our ranking and recommendations optimize for the marketplace as a whole, and as we learned, there are still many challenges ahead, as each side of the marketplace has different needs, sometimes requiring trade-offs.

For eaters, our system offers personalized restaurant recommendations, but ultimately eaters are looking for specific dishes to order. So, we are working on taking our recommendations to the dish level, creating more tailored eater experiences. This is analogous to the music industry's shift from selling albums to selling songs, and we believe it will be a huge leap forward in terms of the experience we can provide. In addition, for new eaters that are checking out the platform, we are working on methods to

of eaters who respond to them.

We are excited to tackle challenges ahead and look forward to further improving the Uber Eats marketplace.

To further this vision, we are looking for talented data scientists and engineers to join our team. If you are interested in helping us solve these types of problems, consider applying for a role on the team!

Subscribe to our newsletter to keep up with the latest innovations from Uber Engineering.

Comments

Yuyan Wang

Yuyan Wang is a senior data scientist at Uber, currently focused on developing personalized ranking and recommendation algorithms within the Uber Eats app. She holds a Ph.D. in Statistics from Princeton University, and a bachelor's degree from Special Class for Gifted Young at University of Science and Technology of China.

Yuanchi Ning

Yuanchi Ning is a senior engineer at Uber, working on optimizing personalization, ranking and recommendation challenges on the Uber Eats app.

Engineering

Uber Eats app. He holds a Ph.D. in Electrical Engineering and Computer Science from UC Berkeley.

Xian Xing Zhang

Xian Xing is a data science manager on the UberEverything team.