Learning To Rank Diversely At Airbnb

Malay Haldar
Airbnb, Inc.
USA
malay.haldar@airbnb.com

Mustafa Abdool
Airbnb, Inc.
USA
moose.abdool@airbnb.com

Liwei He
Airbnb, Inc.
USA
liwei.he@airbnb.com

Dillon Davis
Airbnb, Inc.
USA
dillon.davis@airbnb.com

Huiji Gao
Airbnb, Inc.
USA
huiji.gao@airbnb.com

Sanjeev Katariya
Airbnb, Inc.
USA
sanjeev.katariya@airbnb.com

1 INTRODUCTION
Production e-commerce search ranking systems have to account for a range of target metrics, and search ranking at Airbnb is no exception. Ranking at Airbnb aims to optimize the guest and host experience end-to-end. Targets include increasing bookings, decreasing negative outcomes like cancellations, as well as increasing final trip ratings. Each of these targets are valuable and different parts of ranking are dedicated towards improving these targets. The core model that forms the foundation of ranking is focused on increasing bookings, and it does so by ordering listings by their booking probability.

For a lot of applications, reaching performance comparable to humans is the gold standard. In our case, the booking probability model has far surpassed this level. As part of an evaluation exercise, a task was given to ranking engineers to identify which listing out of a pair was booked by a searcher. Ranking engineers could identify the booked listing for 70% of the pairs, in comparison to 88% for the ranking model.

In our pursuit to further improve ranking, focus turned to the question: what aspects of the ranking model could not be improved by a straightforward refinement of the probability of booking? One area that came up as a possible answer was, diversity in search ranking – or the lack thereof. A survey of existing literature revealed several techniques for diversification – enforcing fairness in ranking ([12], [7]), setwise ([6], [10], [17]) and listwise([3], [2], [18]) ranking, and supplying ranking with features about the result set ([5], [1]). Each of these techniques had their strengths. But instead of picking a solution and fitting it to our context, we started by asking a few questions.

2 WHY SEARCH RESULTS LACK DIVERSITY?
To understand why typical ranking solutions do not diversify search results out of the box, we dive deeper into the specific case of pairwise learning to rank. A similar reasoning applies to pointwise and listwise learning to rank.

How do listings get ranked?
In pairwise learning to rank, we construct training examples from pairs of listings. Consider a search result in response to query \( q \), issued by a user \( u \). Let \( l_x \) be a listing in the search result that was booked, and \( l_y \) a listing that appeared along with \( l_x \) but was not booked. Then the pair \( \{l_x, l_y\} \) forms a training example. Let \( f_{\theta}(q, u, l) \) represent a model with query, user, and the listing to be
ranked as inputs, and $\theta$ the trainable parameters, or “weights” of the model.

To train the model, we get the estimated logits for both the listings in a training example as $\logit_x = f_0(q, u, l_x)$ and $\logit_y = f_0(q, u, l_y)$. The cross-entropy loss is given by

$$\text{crossEntropy} = - \log \left( \frac{e^{\logit_x}}{e^{\logit_x} + e^{\logit_y}} \right)$$

(1)

Minimizing the cross-entropy loss summed over all the training examples leads to a model where for any given pair of listings, $e^{\logit_x}/(e^{\logit_x} + e^{\logit_y})$ can be interpreted as the pairwise booking probability, written as $P_{\text{booking}}(l_x > l_y \mid q, u)$. We omit the conditional $\{q, u\}$ going forward for brevity, and write it as $P_{\text{booking}}(l_x > l_y)$. If the combined booking count for $l_x$ and $l_y$ is scaled to 1.0, then $P_{\text{booking}}(l_x > l_y)$ represents the estimated fraction of bookings for $l_x$, whereas $(1 - P_{\text{booking}}(l_x > l_y))$ represents the fraction for $l_y$.

If we define $P_{\text{booking}}(l_x)$ as the ordinary pointwise probability of booking $l_x$ given an impression in search results, and similarly define $P_{\text{booking}}(l_y)$, then they can be related to the pairwise booking probability by the Bradley–Terry model [13]:

$$P_{\text{booking}}(l_x > l_y) = \frac{P_{\text{booking}}(l_x)}{P_{\text{booking}}(l_x) + P_{\text{booking}}(l_y)}$$

For example, consider 100 search results where $l_x$ and $l_y$ were shown. If $l_x$ got 2 bookings while $l_y$ got 6, then

$$P_{\text{booking}}(l_x) = \frac{2}{100}$$

$$P_{\text{booking}}(l_y) = \frac{6}{100}$$

$$P_{\text{booking}}(l_x > l_y) = \frac{2}{2 + 6} = 0.25$$

Given the pair $\{l_x, l_y\}$, for every 1 booking of $l_x$, we expect 3 of $l_y$.

In order to rank a given set of listings $\{l_1, l_2, \ldots, l_n\}$, we apply the ranking model $f_0(q, u, l)$ to each listing to get the corresponding logits $\{f_0(q, u, l_1), f_0(q, u, l_2), \ldots, f_0(q, u, l_n)\}$, then sort the listings by their logits in descending order.

When sorting by logits, the condition for ranking listing $l_x$ higher than $l_y$ can be successively restated as:

$$f_0(q, u, l_x) > f_0(q, u, l_y)$$

$$\frac{e^{f_0(q, u, l_x)}}{e^{f_0(q, u, l_x)} + e^{f_0(q, u, l_y)}} > \frac{e^{f_0(q, u, l_y)}}{e^{f_0(q, u, l_x)} + e^{f_0(q, u, l_y)}}$$

(2)

We can therefore claim that pairwise learning to rank orders the listings by their pairwise booking probabilities. Inequality 2 can be further rewritten as:

$$P_{\text{booking}}(l_x > l_y) > P_{\text{booking}}(l_y > l_x)$$

$$P_{\text{booking}}(l_x) + P_{\text{booking}}(l_y) > P_{\text{booking}}(l_y) + P_{\text{booking}}(l_x)$$

This establishes the following:

**Property 1.** Ranking listings by their pairwise booking logits is equivalent to ranking them by their pointwise booking probabilities.

**How does ranking by booking probability affect diversity?**

The inequality $P_{\text{booking}}(l_x > l_y) > P_{\text{booking}}(l_y > l_x)$ can also be expressed as:

$$P_{\text{booking}}(l_x > l_y) > 1 - P_{\text{booking}}(l_x > l_y)$$

$$P_{\text{booking}}(l_x > l_y) > 0.5$$

which establishes:

**Property 2.** When ranking two listings by their pairwise booking logits, the one estimated to get more than 50% of the bookings is ranked higher.

At Airbnb, a query $q$ from user $u$ typically results in 0 or 1 booking. In such a scenario, the listing estimated to get more than 50% of the bookings also represents the listing preferred by more than 50% of the past bookers who issued query $q$, or the majority preference for the segment $\{q, u\}$. It is built into the very foundation of pairwise learning to rank to abide by the majority preference at each ranking position. Note, as in real life elections, majority is defined by those who vote, or in our case, those who book. The majority preference is not defined by the entire population that visits Airbnb, as preference of non-bookers remain hidden. In theory, a heavily-personalized ranking model could customize results for the minority preference appropriately. In practice though, accurately identifying users with minority preferences, and personalizing their search results in a fruitful manner, is an open challenge at Airbnb. The majority preference dominates the model’s decisions for all practical purposes.

But majority preference isn’t necessarily the best way to accommodate the preference of the entire population. Let’s elaborate the idea by an example. An important consideration for bookers at Airbnb is price, and the majority of bookings lean towards economical ones. Learning from this user behavior, the ranking model demotes listings if price increases. Experiments directly measuring price sensitivity confirm the same, that bookings drop sharply in response to price increases. But gravity towards affordability doesn’t tell the whole story, as Airbnb is a very diverse marketplace. The pareto principle [15], or the 80/20 rule, provides a better perspective. In most cities, if we segment the total value of bookings, roughly ~20% of bookings account for 50% of the aggregated booking value (see Figure 1).

Using booking value as an indicator of quality, we can apply the broad classification that the majority ~80% of users are affordability-leaning. The remaining minority ~20% are quality-leaning, with average booking value four times higher compared to the 80% majority. In reality, though, users can’t be boxed into such neat binary classifications. Every guest has preference towards both quality and affordability, and the distribution is continuous. Still, the simplified and binarized 80/20 perspective is useful to see where ranking is falling short.

Let’s consider how the first page of search results, where most of the user attention is spent, is impacted by the two effects:

- **Majority principle:** ranking is driven by the majority preference at each position, as shown in Property 2.
- **Pareto principle:** user preferences are distributed smoothly with a long tail, and can be roughly binarized by a 80/20 split, as shown in Figure 1.
To see how diversification of search results can lead to bookings, the answers to the questions happen to be – No, No, and No. We use the log-normal distribution of booking value. Red line plots the percentage of total bookings in Rome that have booking value less than or equal to the corresponding point on x-axis, and the green line plots the percentage of total booking value for Rome covered by those bookings. Splitting total booking value 50/50 splits bookings into two unequal groups of 80/20.

The combined effect is that 100% of the first page results are dictated by what the 80% majority prefers. It’s the tyranny of the majority [16] in search ranking. Intuitively, this suggests that factoring in the minority preference, and giving them proportionate representation, should improve the overall utility of search results. At the same time, the model based on pairwise learning to rank, friend of the tyrant majority oblivious of optimizing.

Why does NDCG correlate with total bookings?
A lot of effort has gone into refining the pairwise booking optimization method. Does diversity need a radically different evaluation approach? Is improving diversity of search results a distraction from offline, paired with measuring bookings in A/B tests online, is the gold standard for evaluating ranking models. Can some simple changes allow it to tackle diversity “for free”?

Figure 1: A distribution of booking values for 2 guests, 2 nights bookings in Rome. X-axis corresponds to booking values in USD, log-scale. Left y-axis is the number of bookings corresponding to each price point on the x-axis. The orange shape confirms the log-normal distribution of booking value. Red line plots the percentage of total bookings in Rome that have booking value less than or equal to the corresponding point on x-axis, and the green line plots the percentage of total booking value for Rome covered by those bookings. Splitting total booking value 50/50 splits bookings into two unequal groups of 80/20.

At the end of the evening, we can sum up the actual realized wins from each bet to get the total, which is the observed outcome of the stochastic process. The observed total dollars won would converge to the expected total dollars won, provided the number of games played N is large enough [14].

To analyze the total number of bookings from a given set of search results, we can employ a reasoning similar to roulette. Consider N search results, each with K listings. Getting a booking for a listing is equivalent to the winning event in roulette, and it depends on the attributes of the listing, as well as the listing’s position in search results. For the ith search result, let the listing placed at the jth position be \( l_{ij} \). We denote its complete probability of booking as \( p_{booking}(l_{ij}) \). The complete probability of booking the listing \( l_{ij} \) and \( p_{attention}(j) \) is the probability that the guest examines the jth position of the search result. The expected number of wins in each case is one booking, so \( win(i, j) = 1 \). The total number of bookings expected can therefore be written as:

\[
E[bookings] = \sum_{i=0}^{N} \sum_{j=0}^{K} p_{booking}(l_{ij}) * p_{attention}(j) \tag{4}
\]

Under the assumption that user attention drops monotonically as they scan the search results from top to bottom, i.e. \( p_{attention}(a) > p_{attention}(b) \) if \( a < b \), we can show that \( E[bookings] \) is maximized if the listings are sorted by their booking probabilities. This property can be established using a proof by contradiction.

Assume we have maximized \( E[bookings] \), but the listings are not sorted by their booking probabilities. Then there must exist a pair of listings such that \( p_{booking}(l_{ix}) < p_{booking}(l_{iy}) \) and \( p_{attention}(x) > p_{attention}(y) \). Consider swapping the positions of the two listings \( l_{ix} \) and \( l_{iy} \). The difference in expected bookings due to the swap is given by:

\[
\begin{align*}
B_x &= p_{booking}(l_{ix}) \cdot B_y = p_{booking}(l_{iy}) \cdot B_x < B_y \\
A_x &= p_{attention}(x) \cdot A_y = p_{attention}(y) \cdot A_x > A_y \\
\Delta E[bookings] &= (B_x A_y + B_y A_x) - (B_x A_x + B_y A_y) \\
&= (B_y - B_x)(A_x - A_y)
\end{align*}
\tag{5}
\]

Since both terms of the product in Equation 5 are positive, \( \Delta E[bookings] > 0 \). This implies the previous sum could not be the maximum. Hence the listings must be sorted by booking probability to maximize the total expected bookings. For alternate arguments supporting the property, see [11].

The difference in total expected bookings due to an unsorted pair, \( \Delta E[bookings] \), is the product of the difference in booking probability and the difference in attention to the two positions. As we discuss next, NDCG tracks total bookings so well because it follows the same conditions.

Let’s begin by considering how NDCG is computed. We adopt a binary definition of relevance where a booked listing has a relevance of 1, and all other listings 0 relevance. For simplicity, we assume only a single listing is booked from a given search result. NDCG

be the corresponding winning amount in dollars. The total dollars won is expected to be:

\[
E[\text{total dollars won}] = \sum_{i=0}^{N} \sum_{j=0}^{K} p_{win}(i, j) * win(i, j) \tag{3}
\]
Algorithm 1

**Input:** A set of N listings \( \{l_0, l_1, \ldots l_{N-1}\} \)

**Output:** Listing positions \( \{\text{pos}(l_0), \text{pos}(l_1), \ldots \text{pos}(l_{N-1})\} \)

1. \( \mathcal{L} \leftarrow \{l_0, l_1, \ldots l_{N-1}\} \)
2. for \( k \leftarrow 0 \) until \( N \) do
3.   Compute logit\( (l_i) \) as \( f_0(q, u, l_i) \) for each \( l_i \in \mathcal{L} \)
4.   \( l_{\text{max}} \leftarrow \text{argmax}(\text{logit}(l_i), l_i \in \mathcal{L}) \)
5.   \( \text{pos}(l_{\text{max}}) \leftarrow k \)
6.   \( \mathcal{L} \leftarrow \mathcal{L} \setminus l_{\text{max}} \)
7. end for

In Algorithm 1, we rely on Property 1, which allows us to use the pairwise booking logit interchangeably with booking probability.

The formula for NDCG in Equation 6 is computing the observed total value of this stochastic process, a simple sum over the realized individual wins. What about the expected value would be the sum of the winning amounts for each position, weighted by the probability of attaining the win. This can be written as:

\[
E[\text{NDCG}] = \frac{1}{N} \sum_{i=0}^{N} \sum_{j=0}^{K} P(l_i, y) \cdot \frac{\log(2)}{\log(2 + j)}
\]

where we swap the listings not ordered by booking probability to demonstrate a contradiction, continues to work. That’s because the positional discount curve is constructed based on how user attention decays by position, making \( \log(2)/(2 + j) \) a strong indicator of gain in total bookings.

The monotonic decay of \( \log(2)/(2 + j) \) together with Equation 7 implies that sorting the listings by their booking probabilities maximizes NDCG. The proof is by contradiction once again. Further, the drop in NDCG from a pair of listings \( l_{x,y} \) not ordered by booking probabilities is given by:

\[
\Delta E[\text{NDCG}] = E_0[\text{NDCG}] \quad \text{(8)}
\]

where the antecedent listing at position \( j \) depends only on the attributes of the listing being ranked, \( l_i \), besides the query and the user. Since attributes of a listing are invariant all throughout, line 3 in Algorithm 1 can be taken out of the for loop. This reduces Algorithm 1 to computing logit\( (l_i) \) once for each listing, and then using it to sort the listings.

But it is not a binding restriction that the booking probability of a listing should be independent of the other listings, and must depend on attributes of the given listing alone. Specifically, consider iteration \( K+1 \) of the for loop in Algorithm 1. We have already placed listings in position 0 through \( K-1 \), and we need to select a listing for position \( K \) by computing the logits of the remaining \( N-K \) listings. When computing the logit for a given listing \( l_i \), we can factor in the attributes of \( l_i \), as well as all the attributes of the \( K \) listings placed at 0 through \( K-1 \) by including the attributes of listings at 0 through \( K-1 \). We can ensure the calculated logits of the \( N-K \) listings are more accurate. Listings being ranked which are too similar to the listings at 0 through \( K-1 \) can have their logits corrected to lower values.

Maximization of NDCG is preserved through this process of extending the inputs to include attributes of the listings placed at 0 through \( K-1 \). The critical step in the proof for maximal NDCG, where we swap the listings not ordered by booking probability to demonstrate a contradiction, continues to work. That’s because the extended inputs from the listings at 0 through \( K-1 \) are invariant in the swapping process.

This provides a mechanism for diversification that is aligned with maximizing NDCG. We require no new metrics to evaluate diversity. Due to the relation between NDCG and total bookings, we expect this mechanism to directly increase total bookings as well. To summarize, diversity removes redundant choices, thereby improving utilization of positions in search results, which get reflected in improved NDCG and total bookings.

While we don’t need a new metric for evaluating diversity, the situation is different when it comes to implementing diversity. For diversity aware booking probabilities, we require the attributes of the listings that are placed before. This information is not available in the garden variety pointwise or pairwise learning to rank frameworks. Hence we need a new kind of model, which we discuss next.

**4 HOW TO IMPLEMENT DIVERSITY IN RANKING?**

In this section we build a framework to rank listings for \( N \) positions while incorporating diversity. Instead of building a single model, we build \( N \) models, a dedicated model for each of the \( N \) positions.

For position 0, we reuse the regular pairwise booking probability model described in Section 2. Let’s name the model \( f_{0,\theta_0}(q, u, l) \), where the 0 index refers to the position in search result, and \( \theta_0 \) the parameters of the model. To recap from Section 2, \( f_{0,\theta_0}(q, u, l) \) maps each listing to a pairwise logit, where sigmoid of the difference of two pairwise logits gives the pairwise booking probability.

Now let’s construct \( f_{1,\theta_1}(q, u, l_0) \), the model for position 1. This model has an additional input \( l_0 \), which we call the **antecedent listing** from position 0. A user scanning the search results from top to bottom would consider the listing at position 1, only if the antecedent listing at position 0 did not meet their requirements.
See [4] and [8] for an in-depth study of this phenomenon. Thus when ranking for position 1, we have incrementally more information than we did when ranking for position 0. Leveraging this new information, we construct $f_{i, 0}(q, u, l, b_0)$ conditional upon the fact that the user has rejected the listing at position 0.

To construct training examples for this conditional pairwise ranking model, we go through the search logs and discard all searches where the listing at position 0 was booked. For the remaining searches, we set aside the listing at position 0, denoting it the antecedent listing. From the listings below position 0, we create pairs of booked and not booked listings, similar to how pairwise ranking examples are created. For a training example with $l_x$ as the booked listing, $l_y$ as the not booked listing, and $l_0$ as the antecedent listing, we compute

$$\text{logit}_x = f_{i, 0}(q, u, l_x, l_0); \text{logit}_y = f_{i, 0}(q, u, l_y, l_0)$$

and cross-entropy loss the same as Equation 1. By minimizing the cross-entropy loss summed over all training examples, we can infer parameters $\theta_i$ of the model such that $e^{\text{logit}_x} / (e^{\text{logit}_x} + e^{\text{logit}_y})$ represents the pairwise probability a user will book $l_x$ over $l_y$, given the condition they have rejected the antecedent listing $l_0$. We write this conditional probability as $P_{\text{booking}}(l_x > l_y \mid A = \{l_0\})$. The conditional part of this pairwise booking probability arises because of the training data. The model is learning about pairwise preference between $l_x$ vs. $l_y$, but only from those users who have rejected $l_0$. Intuitively, we expect the choice of such users to be different from $l_0$, providing us a notion of diversity that can be learnt from the training data.

For position 2, we follow a similar strategy. We discard all searches where either listing at position 0 or 1 was booked, to arrive at the model $f_{i, 0}(q, u, l_0, l_1)$ which gives $P_{\text{booking}}(l_x > l_y \mid A = \{l_0, l_1\})$. Generalizing for position $k$, we get $f_{i, 0}(q, u, l_0, l_{k-1})$ which predicts the conditional probability $P_{\text{booking}}(l_x > l_y \mid A = \{l_0, \ldots, l_{k-1}\})$. For ranking $N$ positions, we now have $N$ distinct ranking models ($f_{0, 0}(q, u, l), f_{i, 0}(q, u, l, l_0), \ldots, f_{N-1, 0}(q, u, l, l_{N-1})$). These models can be plugged into Algorithm 1. In the $K$th iteration of loop at line 3, we can employ $f_{i, 0}(q, u, l, l_{K-1})$ to evaluate the logits, using listings already placed at 0 through $K-1$ as antecedent listings. The logits are computed afresh in each iteration of the loop, now incorporating diversity.

Though the theory presented in this section is simple, it is not a very practical one. For ranking $N$ positions, the number of models needed to be trained is $O(N)$. On top of it, the computational complexity of Algorithm 1 is $O(N^3)$, since the loop at line 2 is $O(N)$, the iteration over each listing at line 3 is $O(N)$, and the complexity of evaluating $f_{i, 0}(q, u, l, l_{K-1})$ is $O(N)$. In the next section we discuss how to make the framework more practical.

5 HOW TO EFFICIENTLY IMPLEMENT DIVERSITY IN RANKING?

We start with the $N$ distinct models constructed for each of the $N$ positions, and simplify them one by one.

The model for position 0, $f_{0, 0}(q, u, l)$, is our regular pairwise ranking model $f_0(q, u, l)$ from Section 2. We treat this as our core model and write the base case as $f_{0, 0}(q, u, l) = f_0(q, u, l)$. To simplify $f_{i, 0}(q, u, l, b_0)$, the model for position 1, we compare it with $f_0(q, u, l)$. Both models are obtained by minimizing the cross-entropy loss over pairwise training examples consisting of booked and not booked listing pairs. The differences between them are:

- $f_0(q, u, l)$ is trained over pairs constructed from all searches. $f_{i, 0}(q, u, l, b_0)$ is trained on the subset of searches where the booked listing appears below position 0.
- $f_0(q, u, l)$ has the listing being ranked $l$ as the input, whereas $f_{i, 0}(q, u, l, b_0)$ has the listing being ranked $l$, as well as the antecedent listing $b_0$, as inputs.

We expect $f_0(q, u, l)$ and $f_{i, 0}(q, u, l, b_0)$ to be fairly close since a large part of their training examples are shared. But we expect $f_{i, 0}(q, u, l, b_0)$ to outperform $f_0(q, u, l)$ for position 1 since it can downrank listings that are too similar to the antecedent $b_0$. We use this insight to simplify $f_{i, 0}(q, u, l, b_0)$, refactoring it into two models as:

$$f_{i, 0}(q, u, l, b_0) = f_0(q, u, l) - s_\phi(q, u, l, b_0)$$

First part of the refactor is $f_0(q, u, l)$, the regular pairwise booking probability model. The second part is a new model $s_\phi(q, u, l, b_0)$ parameterized by $\phi$. It adds a negative term based on the similarity between $l$ and $b_0$. This similarity is not defined by us, instead it is learnt from the training data that we built for $f_{i, 0}(q, u, l, b_0)$. We train $f_0(q, u, l)$ prior to training $f_{i, 0}(q, u, l, b_0)$. When training $f_{i, 0}(q, u, l, b_0)$, we don’t need to train the $\theta$ parameters again. We can simply substitute $f_0(q, u, l)$ by the unconditional booking logit $u_b l$, where $u_b l = f_0(q, u, l)$ is obtained by evaluating the model for $l$. Thus,

$$f_{i, 0}(q, u, l, b_0) = u_b l - s_\phi(q, u, l, b_0)$$

Given a training example for $f_{i, 0}(q, u, l, b_0)$, with $l_k$ as booked, $l_y$ as not booked, and $l_0$ as antecedent, we have

$$\text{logit}_x = u_b l_k - s_\phi(q, u, l_k, b_0); \text{logit}_y = u_b l_y - s_\phi(q, u, l_y, b_0)$$

cross-entropy loss defined by Equation 1. The parameters $\phi$ obtained by minimizing the cross-entropy loss over all the training examples gives us a simplified construction of $f_{i, 0}(q, u, l, b_0)$ according to Equation 9.

For position 2, we have two antecedent listings, one at position 0 and the other one at position 1. We need to account for the similarity to both these antecedent listings. Though this time around, we do not need to learn the similarity model all over again. We can reuse the similarity model we learnt as part of $f_{i, 0}(q, u, l, b_0)$. The refactored model for position 2 can therefore be written as:

$$f_{2, 0}(q, u, l, l_0, l_1) = u_b l_1 - s_\phi(q, u, l_1)$$

But Equation 10 is valid only if the effect of $l_0$ and $l_1$ are completely independent of each other. On the other hand, if $l_0$ was an exact duplicate of $l_1$, then $l_1$ would not have any incremental effect and we could completely ignore it to rewrite Equation 10 as

$$f_{2, 0}(q, u, l, l_0, l_1) = u_b l_1 - s_\phi(q, u, l_0)$$

In reality, we expect the true effect to be somewhere in between Equation 10 and 11, which we write as

$$f_{2, 0}(q, u, l, l_0, l_1) = \lambda \cdot u_b l_1 - s_\phi(q, u, l_0) - \lambda \cdot s_\phi(q, u, l_1)$$

$$f_{2, 0}(q, u, l, l_0, l_1) = f_{1, 0}(q, u, l, l_0) - \lambda \cdot s_\phi(q, u, l_1)$$

where $0 \leq \lambda \leq 1$.

Moving on to position 3, we have to factor in the effect of similarity to the additional antecedent listing $l_2$ given by $s_\phi(q, u, l, l_2)$. 

4613
But we expect the effect to be reduced under the assumption that $l_2$ isn’t completely independent of $l_0$ and $l_1$. To denote the incremental impact of $l_2$, we scale the contribution from $l_2$ by $\lambda^2$, once for overlap with $l_0$, the second time for overlap with $l_1$. This gets us the refactored equation:

$$f_{3, \theta_3}(q, u, l_0, l_1, l_2) = f_{2, \theta_2}(q, u, l_0, l_1) - \lambda^2 \cdot s_{\phi}(q, u, l)$$

Generalizing for position $(k+1)$, we can write

$$f_{k+1, \theta_{k+1}}(q, u, l_0, \ldots, l_{k-1}) = f_{k, \theta_k}(q, u, l_0, \ldots, l_{k-1}) - \lambda^k \cdot s_{\phi}(q, u, l_k)$$

$$= f_{0}(q, u, l) - \sum_{i=0}^{k} \lambda^i \cdot s_{\phi}(q, u, l_i)$$

We now need to build only two models, $f_0(q, u, l)$ and $s_{\phi}(q, u, l, l_0)$. The models for each of the positions can be expressed in terms of those two.

Using $f_0(q, u, l)$ and $s_{\phi}(q, u, l, l_0)$, we can construct Algorithm 2 which provides an iterative way to construct the search results, while taking diversity into account.

**Algorithm 2 Ranking diversely**

**Input:** A set of $N$ listings $\{l_0, l_1, \ldots, l_{N-1}\}$

**Output:** Listing positions $(\text{pos}(l_0), \text{pos}(l_1), \ldots, \text{pos}(l_{N-1}))$

1. $L \leftarrow \{l_0, l_1, \ldots, l_{N-1}\}$
2. Compute $\logit(l_i) \leftarrow f_0(q, u, l_i)$ for each $l_i \in L$
3. $l_{\text{max}} \leftarrow \text{argmax}(\logit(l_i), l_i \in L)$
4. $\text{pos}(l_{\text{max}}) \leftarrow 0$
5. $l_{\text{atcdnt}} \leftarrow l_{\text{max}}$
6. $L \leftarrow L \setminus l_{\text{max}}$
7. for $k \leftarrow 1$ until $N$
8. \hspace{0.5cm} $\logit(l_i) \leftarrow \logit(l_i) - \lambda^k \cdot s_{\phi}(q, u, l, l_{\text{atcdnt}})$ for each $l_i \in L$
9. \hspace{0.5cm} $l_{\text{max}} \leftarrow \text{argmax}(\logit(l_i), l_i \in L)$
10. \hspace{0.5cm} $\text{pos}(l_{\text{max}}) \leftarrow k$
11. \hspace{0.5cm} $l_{\text{atcdnt}} \leftarrow l_{\text{max}}$
12. \hspace{0.5cm} $L \leftarrow L \setminus l_{\text{max}}$
13. end for

We can treat $\lambda$ as a hyperparameter here and sweep through different values to find the one that maximizes NDCG. In our case, we settled at $\lambda = \frac{1}{2}$.

The simplification in this section reduced the number of models from $O(N)$ to 2, and the computational complexity from $O(N^3)$ to $O(N^2)$. Line 8 is now the bottleneck in Algorithm 2. The number of iterations of the loop is $O(N)$, and iterating over each listing in line 8 is $O(N)$. The main computation in this inner loop is evaluation of $s_{\phi}(q, u, l, l_{\text{atcdnt}})$. We can further optimize the evaluation of $s_{\phi}(q, u, l, l_{\text{atcdnt}})$ by caching the results of the neural network layers that process the listing features.

6 HOW DOES THE THEORY WORK IN PRACTICE?

We developed the theory in Sections 4 and 5 before embarking on actual implementation and experimentation. This allowed us to make certain predictions about the experimental results. We subsequently verified these predicted effects from online A/B tests. NDCG. The first prediction to come out of the theory was a simple one: that NDCG should improve. What made the prediction interesting was the expectation of an NDCG gain, with no additional information added to the training data. In contrast, most NDCG gains over the past years required incrementally new information in the form of features, labels, or bug fixes. On the test set for $f_{1, \theta_1}(q, u, l, l_0)$, we observed an NDCG gain of 0.45% compared to the baseline Algorithm 1. When measured on the test set of $f_0(q, u, l)$, this gain translated to 0.2%. This gain is smaller than 0.45% because the listing at the first position remains invariant between Algorithm 1 and 2, diluting the effect. The impact on NDCG measured in the online A/B test was much stronger, where we observed an increase of 1.5%.

**Bookings & Booking Value.** From the 0.2% NDCG gain, we expected a similar gain in bookings online. Further, we expected these bookings to come from preference groups which were a minority in the training data. The most prominent minority preference being quality-leaning searchers, as discussed in Section 2. This allowed us to make a further prediction: that there would be gains in gross booking value, or the sum of the prices of the booked trips, which would be multiple times over the bookings gain. In the online A/B test, we observed a bookings gain of 0.29%. Along with it, we saw a 0.8% gain in gross booking value. Segmenting the bookings gain by user groups, we found almost the entire gain came from users who were booking a listing on Airbnb for the first time.

**Price & Location.** To directly measure diversification, we compared metrics along some key dimensions. The first measure compared the variance in price among the top 8 results. We observed an increase of 3.4% in treatment, which captures the increased diversity in price, and hence quality by proxy. The second metric compared the number of listings in the top 8 results that were within 0.5 km of each other. We noted a decrease of 0.62%, which shows reduced redundancy in location.

**Trip Quality.** To measure the impact of diversity on the entire user experience, we waited for 90 days after the end of the A/B test. This allowed the majority of the trips booked during the experiment to be realized. Comparing the ratings from guests checking out of their stays, we noted an increase of 0.4% in 5-star ratings. Diversifying search results shifts the balance away from the majority preference of affordability towards the minority preference of quality. This shift towards quality ultimately surfaces in improved trip ratings.

7 CONCLUSION

Our diversification attempts started in 2017 with category-based diversification. Various categories were tried, based on price, location, and amenities. All these efforts resulted in disappointment. The breakthrough came in 2019 with [1]. We revisited the problem in 2022 with a theory-first approach, letting the model learn the notion of diversity from the training data. This lead to one of the most impactful ranking changes of the year. But as discussed in Section 2, this training data itself is biased against diversity. After the launch of the diversity ranker, we expect future training data to have richer examples to learn from, enabling a virtuous cycle of diversification. An extended version of the paper is available at [9].
REFERENCES

[1] Mustafa Abdool, Malay Haldar, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Manaswala, Lynn Yang, Bradley Turnbull, Qing Zhang, and Thomas Legrand. 2020. Managing Diversity in Airbnb Search. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Virtual Event, CA, USA) (KDD ’20). 2952–2960.

[2] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (Ann Arbor, MI, USA) (SIGIR ’18). 135–144.

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In Proceedings of the 24th International Conference on Machine Learning (Corvalis, Oregon, USA) (ICML ’07). 129–136.

[4] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In Proceedings of the 2008 International Conference on Web Search and Data Mining (Palo Alto, California, USA) (WSDM ’08). 87–94.

[5] Weicong Ding, Dinesh Govindaraj, and S V N Vishwanathan. 2019. Whole Page Optimization with Global Constraints. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Anchorage, AK, USA) (KDD ’19). 3153–3161.

[6] Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2018. From Greedy Selection to Exploratory Decision-Making: Diverse Ranking with Policy-Value Networks. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (Ann Arbor, MI, USA) (SIGIR ’18). 125–134.

[7] Ruoyuan Gao and Chirag Shah. 2020. Toward Creating a Fairer Ranking in Search Engine Results. Inf. Process. Manage. 57, 1 (jan 2020), 19 pages.

[8] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. 2009. Click Chain Model in Web Search. In Proceedings of the 18th International Conference on World Wide Web (Madrid, Spain) (WWW ’09). Association for Computing Machinery, New York, NY, USA, 11–20.

[9] Malay Haldar, Mustafa Abdool, Liwei He, Dillon Davis, Huiji Gao, and Sanjeev Katariya. 2023. Learning To Rank Diversely At Airbnb. arXiv:2210.07774 [cs.IR]

[10] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020).

[11] Stephen E. Robertson. 1977. The probability ranking principle in ir. Journal of documentation 33(4) (1977), 294–304.

[12] Ashudeep Singh and Thorsten Joachims. 2019. Policy Learning for Fairness in Ranking.

[13] Wikipedia. 2022. Bradley–Terry model. https://en.wikipedia.org/wiki/Bradley%E2%80%93Terry_model [Online; accessed 14-August-2022].

[14] Wikipedia. 2022. Expected value. https://en.wikipedia.org/wiki/Expected_value [Online; accessed 14-August-2022].

[15] Wikipedia. 2022. Pareto principle. https://en.wikipedia.org/wiki/Pareto_principle [Online; accessed 14-August-2022].

[16] Wikipedia. 2022. Tyranny of the majority. https://en.wikipedia.org/wiki/Tyranny_of_the_majority [Online; accessed 14-August-2022].

[17] Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H. Chi, and Jennifer Gillenwater. 2018. Practical Diversified Recommendations on YouTube with Determinantal Point Processes. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (Torino, Italy) (CIKM ’18). 2165–2173.

[18] Le Yan, Zhen Qin, Rama Pasumarthi, Xuanhui Wang, and Michael Bendersky. 2021. Diversification-Aware Learning to Rank using Distributed Representation. 127–136. https://doi.org/10.1145/3442381.3449831