Log In Join

Back To Module Home

Machine Learning System Design

0% completed

Search Module

Rental Search Ranking System Design

Mark Module as Completed

Rental Search Ranking System Design

Learn about the rental search ranking system design for Airbnb.

We'll cover the following

4. Calculation & estimation#

Assumptions#

- 100 million monthly active users
- On average, users book rental homes 5 times per year. Users see about 30 rentals from the search result before booking.
- There are 5 * 30 * 10^8 or 15 billion observations/samples per year or 1.25 billion samples per month.

Data size#

- Assume there are hundreds of features per sample. For the sake of simplicity, each row takes 500 bytes to store.
- Total data size: 500 * 1.25 * 10^9 = 625 * 10^9 bytes = 625 GB. To save costs, we can keep the last 6 months or 1 year of data in the data lake, and archive old data in cold storage.

Scale#

- Support 150 million users

5. High-level design#

- Feature pipeline: Processes online features and stores features in key-value storage for low latency, down-stream processing.

- Feature store is a features values storage. During inference, we need low latency (<10ms) to access features before scoring. Examples of feature stores include MySQL Cluster, Redis, and DynamoDB.

- Model Store is a distributed storage, like S3, to store models.

Let’s examine the flow of the system:

Created with Fabric.js 3.6.3

User searchs rentals and request Application Server for result

1 of 5

- The user searches for rentals with given queries, i.e., city and time. The Application Server receives the query and sends a request to the Search Service.
- Search Service looks up in the indexing database and retrieves a list of rental candidates. It then sends those candidates list to the Ranking Service.
- Ranking service uses the Machine Learning model to score each candidate. The score represents how likely it is a user will book a specific rental. The Ranking service returns the list of candidates with their score.
- Search Service receives the candidates list with booking probability and uses the probability to sort the candidates. It then returns a list of candidates to the Application server, which, in turn, returns it to the users.

As you can see, we started with a simple design, but this would not scale well for our demands, i.e., 150 million users and 1.25 billion searches per month. We will see how to scale the design in the next section.

6. Scale the design#

- To scale and serve millions of requests per second, we can start by scaling out Application Servers and use Load Balancers to distribute the load accordingly.

- We can also scale out Search Services and Ranking Services to handle millions of requests from the Application Server.

- Finally, we need to log all candidates that we recommended as training data, so the Search Service needs to send logs to a cloud storage or send a message to a Kafka cluster.

7. Follow up questions#

Question	Answer
What are the cons of using ListingID embedding as features?	ListingIDs are limited with only millions of unique IDs, plus each listing has a limited number of bookings per year which might not be sufficient to train embedding.
Assuming there is a strong correlation between how long users spend on listings and the likelihood of booking, how would you redesign the network architecture?	Train multiple output networks for two outputs: view_time and booking
How often do we need to retrain models?	It depends, we need to have infrastructure in place to monitor the online metrics. When online metrics go down, we might want to trigger retraining the models.

8. Summary#

- We learned to formulate search ranking as a Machine Learning problem by using booking likelihood as a target.
- We learned to use `Discounted Cumulative Gain` as one metric to train a model.

- We also learned how to scale our system to handle millions of requests per second.
- We can read more about how companies scale there design here.

Booking Model

Next

Problem Statement and Metrics

Mark as Completed

Report an Issue