

[General System Design interview Tips](#)

[ML Design Interview Template](#)

[Stages of a Ranking System](#)

[Case Studies](#)

[Doordash Recommendation System](#)

[Instacart - Item substitution](#)

[Youtube Recommendation System](#)

General System Design interview Tips

1. Start with documenting your summary/overview in Google docs/Excalidraw or Zoom whiteboard. Even if the company hasn't provided a link and interviewer insists on the conversation to be purely verbal - Document key bullet points.
2. Present your interview systematically; lead the conversation and don't wait for the interviewer to ask questions. At the beginning of the interview, present the discussion's structure and ask the interviewer about their main areas of interest.
3. Show your understanding of the business implications by sharing insights on metrics. Understand what the product truly expects from you.
4. Actively listen to the interviewer. At the start, ask: "What are you primarily looking for?" . Address the whole process, from collecting and labeling data to defining metrics.
5. Assess the importance of the modeling process.
6. Familiarize yourself with the nuances of ML-Ops, such as: At the start of the interview, get a feel for if the interviewer seems interested in ML-Ops. You'll mostly get a clear signal on whether or not they are interested.
 - a. Managing model versions
 - b. Training models
 - c. Using model execution engines
7. Keep your resume at hand and review it before starting the interview.


ML Design Interview Template

1. Problems setup and Label Collection
 - a. Clarifying questions
 - b. Definition of success
 - c. Positive and Negative Labels
 - i. Different Options to define labels
 1. Join a group

- 2. Retention after a week
 - 3. Interaction with other folks
 - 4. Meaningful interaction: time spent or friends in the group
 - ii. Fairness: Makes sure we have enough data for under-represented group
 - d. Label Generation
 - i. Engagement as Proxy
 - 1. User click as yes, no click as no
 - ii. Use Labelers
 - 1. Use Semi-Supervised or Unsupervised (clustering) to increase the efficiency of labelers
 - 2. Visits in a session (pinterest or Doordash) are considered to be similar pins or restaurants
 - e. Downsampling the Dominant class using Negative Sampling
 - i. Only downsample the training data, Validation and Test keep the same distribution
 - f. Bias in Training
 - i. Limit the number of labels per user or per video or restaurant to prevent active users or popular restaurant dominate the labels
2. Feature Engineering
- a. User Features
 - b. Group Features
 - c. Cross Features between Group and users

User	Candidate	Action	Context	Aggregation
uid gender age location cohort ...	id publisher category picture quality ...	like share click long click ...	time of day day of week mobile device wifi vs 4g ...	count rate

- i.
 - d. Contextual Features (time of day, holiday, mobile device, wifi v.s. 4g)
 - e. Start with basic counters/ratio => gbdt
- 3. Modeling
 - a. Two tower model
 - b. How to create embeddings (Graph Embeddings)

- c. Retrieval (optimize for recall)
 - i. Collaborative Filtering
 - d. Diversification of sources
 - e. Ranking
 - i. Two tower model
 - ii. Optimize for precision
- 4. Measurement
 - a. Online <-> offline
 - b. NDCG
 - c. Precision @top5
 - i. How many relevant items are present in the top-k recommendations of your system
 - d. Mean Average Precision
 - i. The mean of the AP@K for all the users.
 - ii. For example, to calculate MAP@3: *sum AP@3 for all the users and divide that value by the amount of users*
 - e. Good explanation on which one are you using in each stage
 - f. Online: We care about business metrics the most
 - g. A/B test or Multi-Armed bandit
- 5. Debugging
 - a. Have a structured way of approaching all the issues
 - b. Write them down as you speak
 - c.  Online Offline Model Debugging
- 6. Feature Logging
 - a. Training
 - b. Debugging
- 7. Preparing the training pipeline
- 8. Deployment
 - a. Novelty effects - where the treatment arm shows positive engagement for the early days and slowly trends down to neutral.
 - b. Sometimes when we refresh a model - even subtle changes can temporarily drive engagement - We believe it stems from the fact that long running models tend to “exploit” too much and those tests bring some new areas of exploration.

Stages of a Ranking System

Funnel ai has a good series:

- Part 1: <https://blog.fennel.ai/p/real-world-recommendation-system?s=w>
- Part 2: <https://blog.fennel.ai/p/real-world-recommendation-systems?s=r>
- Part 3: <https://blog.fennel.ai/p/real-world-recommendation-systems-21e>

- Feature Engineering: <https://blog.fennel.ai/p/feature-engineering-for-recommendation>

1. Retrieval

- a. From Millions to 5K
- b. Simple features
- c. Generators:
 - i. Social Graph: Friends' Video, Friend of a Friends engagement
 - ii. Popular and Trending
 - iii. Candidates in the Same Language
 - iv. Same Location
 - v. Users or posts with the same Embeddings
 - vi. Posts with high co-occurrence: Use a liked post as a seed
- d. **Collaborative Filtering** to suggest candidate base on user-ads interaction
 - i. Generate an embedding for users and ads in the same space (d dimension)
 - ii. Matrix Factorization is a way to generate embeddings for Collaborative Filtering
- e. Two Tower System to choose the candidate for each generator
- f. Metric: Focused to improve **Recall**

2. Filtering

- a. Based on user preference (don't like washington post)
- b. A post is 2 months old
- c. A product is out of stock

3. Feature Extraction

- a. Limited time
- b. Feature store: Make sure there is no discrepancy between train and test data due to feature generation logic
- c. **Normalization or Standardization of dense features**
- d. **Information leakage:**
<https://www.fennel.ai/blog/two-types-of-information-leakage/>
- e. **Feature logging:**
<https://fennel.ai/blog/feature-engineering-for-recommendation-031/>

4. Ranking

- a. From 5K to 100
- b. More Complex Features and Models to capture Context
- c. Model Selection
 - i. Collaborative Filtering
 1. Easy but cold start problem and cannot generalize to all user features
 - ii. Pointwise Models
 1. Two Tower Model

- a. Cross Features are not included (it is the first step)
 - 2. Deep and Wide (Google)
 - a. Deep & Cross v2
 - 3. Deep Learning Recommendation Model (DLRM: Facebook)
- iii. Pairwise Model
 - 1. RankNet
 - 2. LambdaRank
- d. Several Models for different criteria (MTML is another option)
 - i. Time Spent
 - ii. Increase Interaction
 - iii. User Retention
 - iv. Meaningful Interaction (based on friends or time spent in community)
 - v. Click prediction, $P(\text{Lead submission} | \text{Click})$
- e. Value Model
 - i. Combination of different model: $a * p(\text{view}) + b * p(\text{click} | \text{view})$
 - ii. Provide flexibility in for different product using the same mode by adjusting a, b
- f. Metric: Focused on the **Precision**
- g. Multi Stage:
 - i. GBDT in the first stage
 - ii. NN in the second stage
- h. Two Tower Sparse Network: TTSN
 - i. Cache During Serving: We could precalculate the result of ads (or store) tower and cache it
 - ii. At run time we only calculate the embedding for the user tower
 - iii. Multi-modal features
 - 1. Video modality:
 - a. Sample multiple frames across the video and use classic CNN for each of them, then aggregate into a single vector
 - i. Use transfer learning or train from scratch
 - b. Recurrent NN (LSTM): seq2vec
 - c. output : video vector (k)
 - 2. Image Modality:
 - a. A CNN based model (VGG-16) to provide an embedding for an image
 - 3. Text modality:
 - a. Description, comments, title, etc.
 - i. Use similar approaches to the above (transfer learning)
 - ii. Or train from scratch
 - b. Output: text vector of size k
 - 4. Combine them into one embedding vector
 - a. Concatenate and use FC layer to map it to the same size as query embedding

- b. In case several images or comments: Max Pooling on top of them to pick the max elements for each dimension
- c. Final output: size of k

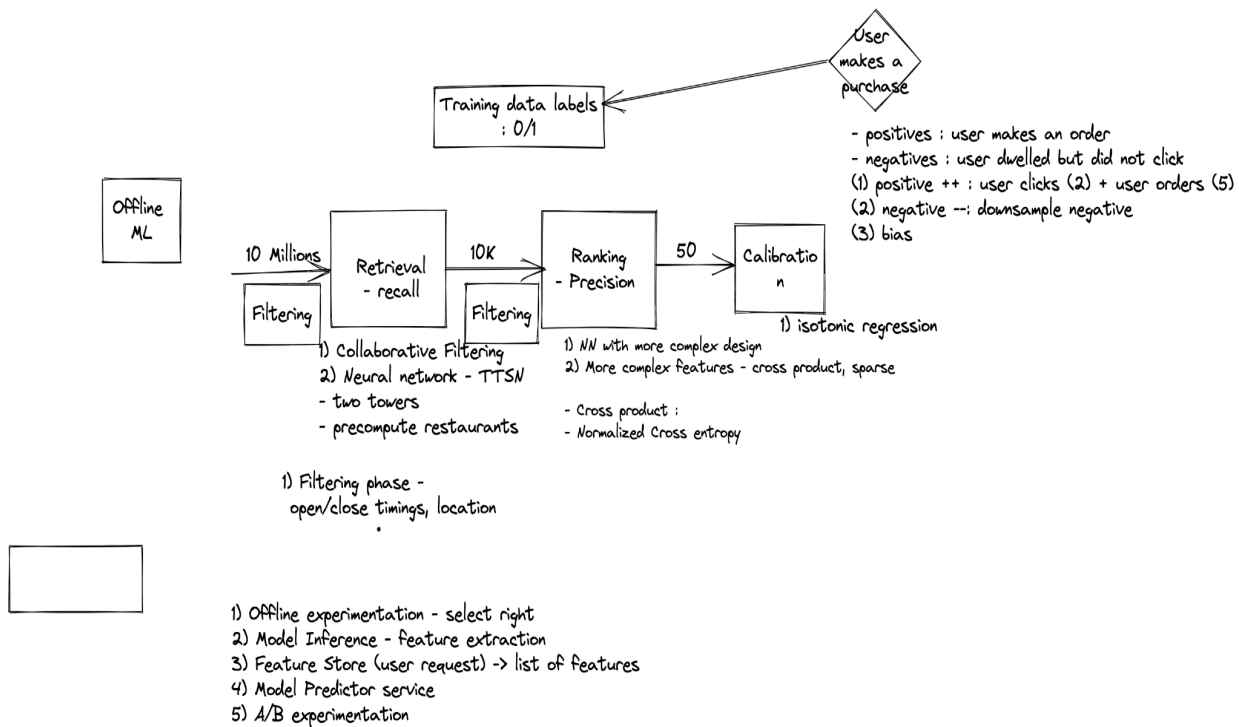
5. Re-Ranking

- a. Combine Several Ranking Models by some Weights
 - b. Diversity of the Sources
 - i. Different generators
 - ii. Different topics
 - iii. Different products
 - 1. New Product: Upranking Trending Posts to compensate for the lack of data for the trending posts
 - c. Freshness of the items (boost new items)
 - d. Integrity: Downrank borderline content (anti-Vaxx)
 - e. Fairness:
 - i. Different model for underserved group
 - ii. Track online/offline metrics on each demography to watch
 - f. Position Bias
 - g. Cool Off: Impression Discount
 - i. If a place has been shown recently and user has not clicked on it or ignored it then don't show it for a while (discount its score by d)
 - ii. Discount factor D is a hyper-parameter that the model can learn
6. Calibration
7. Measurement
- a. Offline
 - i. ROC AUC
 - 1. ROC AUC: Focus on both classes
 - 2. Precision Recall AUC: Useful for **imbalanced classification problems** by focusing on minority class : Ref
 - ii. Precision, Recall and F1
 - 1. Are False Negatives More Important? (focus on recall)
 - a. Use F2-Measure
 - 2. Are False Positives More Important? (focus on precision)
 - a. Use F0.5-Measure
 - iii. Precision@5: If user only sees the top 5 results
 - 1. Precision@25: If user decided to look deeply into more candidates
 - iv. mAP: Mean Average Precision
 - 1. The positive samples get to the top of the list then it is equal to 1 Ref, ref
 - v. NDCG: Normalized Discounted Cumulative Gain (@top 5 locations)
 - 1. Each sample has a relevance and it gets top max value if samples are sorted based on their relevance descendingly (high relevance first)
 - b. Online
 - i. Business Metrics:

1. Revenue
2. CTR
3. Time Spent (in app, on the page, in groups)
4. Engagement (comments, like, **meaningful comments**)
5. Stickiness: Retention
 - a. Users who revisited in the 7 days or 14 days or 30 days
6. Satisfaction Metrics
 - a. Survey
 - b. Number of report or x-outs
7. We can also log the online data to measure the same offline metrics: NDCG or Precision@5
 - a. But it could be tricky: If user only clicks on the first restaurant, does it mean they don't care about the rest or it means the first one was good enough and they did not need to check the second one
8. Feature Logging
 - a. Log all the features and result for sanity check and future debugging
 - b. Features change during time, so we have to dump them for training
 - i. Another solution is feature generation which could be very complicated but is an option if it takes a long time to log enough features for training. The feature generation is dangerous since the training and serving may have different logic to generate features and it could cause mismatch between train and test data.
9. Training Data
 - a. Generate ID for each training data point so we could join the label with the features logged during prediction
 - b. Tricky: What if there were no impressions of the post after position 5? Only generate labels for the first 5 candidate posts and ignore the rest of the posts
 - c. Attribution Window: it could take several days for the labels to get back to us. We may not get all of them back

Case Studies

Doordash Recommendation System



Instacart - Item substitution

Problem : Design an item recommendation system : Item in cart is missing

Qns :

- Item missing- Added to cart and came back
- product behaviour - place the order and the prompt of
- 3 items available -> recommendation of 5 time

- Generic missing item recommendation -
- Store specific item recommendation
- Total number of items in inventory -~10000 per store

Overview of the system

- Metrics :
- success would like if
 - a) selects the recommended item - "click" event /positive feedback
 - b) user-reports on recommended - sparse event/click based

- item is recommended
- item unavailable / open to see if relevant & approve / add it to cart

* Training Data

- + label - clicks (item1, item2, positive label)
- label - randomly sampled, 2nd and 3rd page items are negatives

Caveat - 2nd/3rd page is relevant

* Filtering

- rules : consider items from store, consider only available stocks

* Modeling

--> Candidate generation (recall)

- * a) Collaborative filtering
- * b). Text description
- * c) Image embedding based
- * + Filtering -> category based , store

--> Ranking:

- GBDT model :
- Logistic regression
- * classification : loss : Cross entropy
(100 -> 10)

* Feature Engineering

- item feature [past history of out of stock, name , description, historical substitution rate, success rate, \$\$]
- user features -[categories of items that user past purchased, user historical satisfaction]
- cross item features [
- store feature [store name, history of out of stock, \$\$ total sale value]
- context features - [time of day, weather]

-- Evaluation metrics

- Offline metrics: Cross entropy ,
- NDCG, Precision @ K,
- Online metrics :
- * total accepted substitution
- * overall increase in purchase

-- Deployment/ Model debugging

- A/B Experimentation

Youtube Recommendation System

Assumption

- Few billion videos
- QPS of 10K /s
- User is logged in - personalized
- Recommendation on home page

High level components

- 1) Business metrics
 - a) What does success like
- 2) Overview of our system
 - a) Model training
 - b) Model serving
- 3) Model training
 - a) Training data generation
 - b) Feature engineering
 - c) Modeling choices
 - d) Offline metrics
- 4) Model serving
 - a) ...
- 5) Other niche considerations
 - a) Fairness.

- Business metrics
 - Revenue gains,
 - Time spent on the site -> daily time spent/ time spent
 - - demographic weighted timespent, region specific timespent
- Offline
 - Video views / session
 - Precision @ k
 - Number of clicks
- Model training
 - Training data generation
 - Label = 0/1
 - User logs
 - (video id, label, timestamp, userid)
 - Label distribution - ~<2% of training samples to be positives
 - Downsampling
 - Importance sampling
 - Positive -> high quality positives

- By filtering out samples where video watch ration < 10%
 - Hard negatives
- user/channel
- [0/1]
- Feature engineering
 - User features -
 - Profile features - age/attributes
 - Engagement - time spent on site, 7d, 14d, categories user watches,
 - User Embedding -
 - Video features
 - Content understanding - video embeddings
 - Engagements - likes, clicks, views over [past hour, 1 day, 3 hours]
 - Negative engagement - [xout, dislikes, takedown -> [time frames]
 - Channel author
 - User X videos
 - Video id's that user engaged over past day
 - Author id's that user liked over past..
 - Considerations
 - When are features logged?
 - Feature leakage
- Modeling
 - Inference less than 50ms
 - Multi stage rankings system s
 - Retrieval - focus on recall
 - Less complex model
 - 1B -> 10000
 - Ranking - focus on precision
 - More complex model with more complex user-vide features
 - 10000 -> 100
 - 10000 -> 500 -> 100
 - Point wise ranking - classifier model
 - Retrieval
 - Collaborative filtering ->
 - Logistic regression
 - Two tower sparse nn
 - User feature
 - Video features
 - FAISS - given a user embedding - find near similar duplicates
 - Filtering layer
 - Videos with views < 100
 - Topics that user doesn't like
 - User reports
 - Ranking

- 10000 -> 500 -> 10
- 2 stage ranking
- DCN / Sparse NN /
- Cross entropy loss
- Deep and wide NN
- DLRM -
- Considerations
 - Diversity
 - Repetition
 - Freshness
 - Cold start effect
 - Embedding , feature cold start
 - Trending videos
 - Novelty effect

Hope this was useful.

Credit to Hesam Salehian for providing the initial version of the document. I've added more pointers and case studies.