Hotel Ranking
Learning to rank framework

Narasimha Medeme,
Director,
DataScience, MakeMyTrip

Arpit Katiyar - Lead Data Scientist
Pulkit Bansal - Lead Data Scientist
Rakhi Agrawal - Principal Data Scientist

Importance of Hotel Ranking

- About **80%** of the our hotel bookings are **mobile** transactions. One of the factors that will increase further.

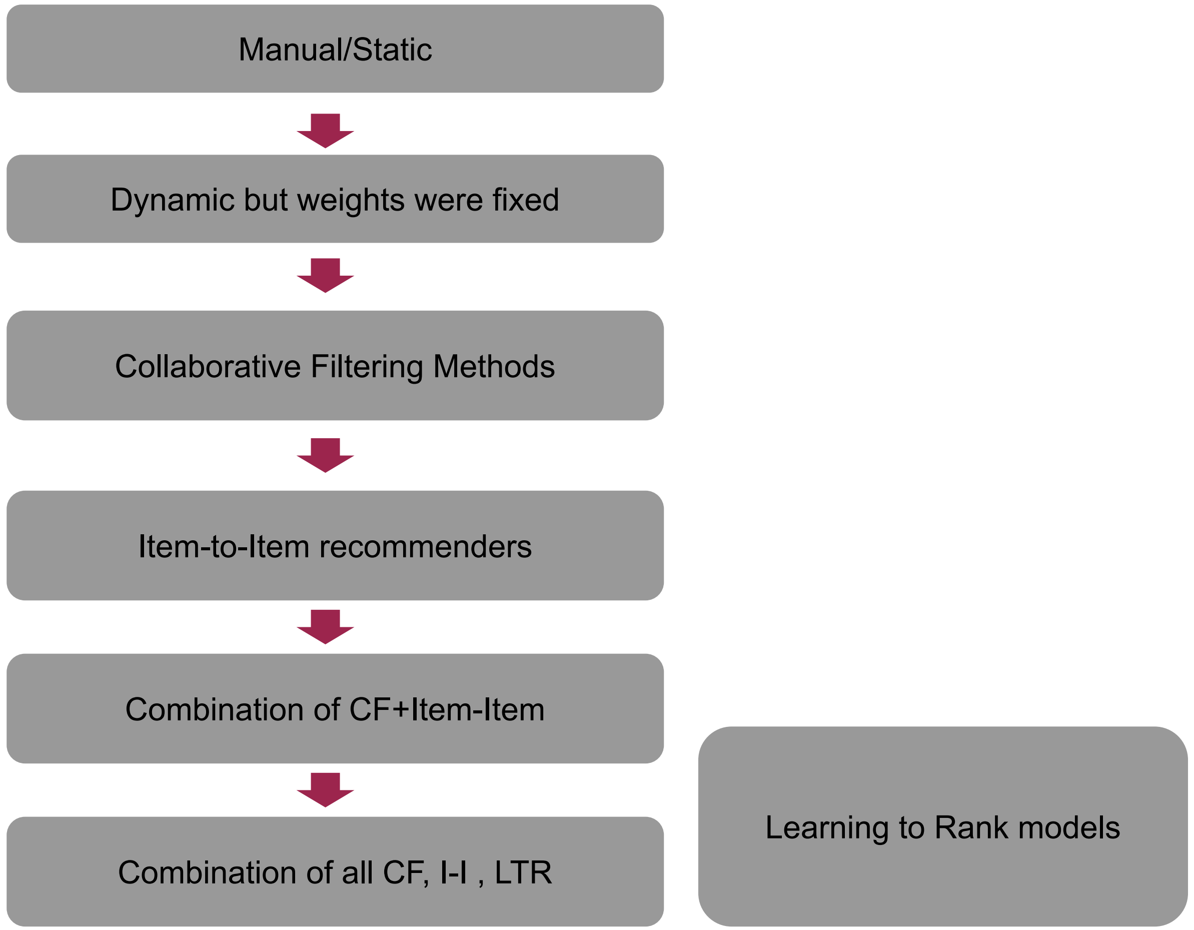
- Every bit of the **app landscape is important,** so one has to make best use of landing page, i.e. show exactly what consumers would need.

- Also consider **long tail of product/partners** we serve. An efficient market place should be able to give platform for all types of hotels **especially the less popular but great-value hotels**.

- Said differently, we serve both customers and hotels**,** we believe matching the right customer to the right hotel will **increase customer and hotel partner satisfaction**, thereby **leading to repeat purchase behavior of customers**.

- Improve biz metrics - spread, funnel conversion rate from listint to detail and listing to book.

Journey


Manual/Static
Dynamic but weights were fixed
Collaborative Filtering Methods
Item-to-Item recommenders
Combination of CF+Item-Item
Combination of all CF, I-I , LTR
Learning to Rank models

Learning to Rank - Introduction

Learning to rank model begins with a dataset of queries and items, often shortlisted using another simpler/rule based algorithm if there are way too many items in the candidate pool to show.

Learn the interactions between query features, item features, user features and query-item interaction features which drives/predicts “relevance” of the items to the user-query.

- It is not unlike supervised classification /regression models.
- Data is grouped into sets, typically group all items within unique User x Search contexts into a “group”
    - Assume independence (IID error) across groups.
    - Not unlike a “choice set”, in discrete choice econometric models. Although relevance enforces ordinality across items.
- Custom loss functions.
- Relevance is the outcome score/label defining the importance of item to the user-query.
- Relevance could be implicit or explicit feedback.
- Relevance could be binary or multi-level. Multiple levels/values could correspond to different levels of importance/feedback types. They should be monotonically increasing although need not have smoother distribution.

Ranking metrics

- **Precision** = TP / (TP + FP) - fraction of the hotels shown to the user that are relevant
- Recall = TP / (TP + FN) - fraction of relevant hotels shown to the user
- **MAP (Mean Average Precision)**
  - Considers the rank of each relevant hotel
  - Case 1: [NR, NR, NR, R, R, R]
  - Precision at 4: ¼
  - Precision at 5: ⅖
  - Precision at 6: 3/6
  - Average precision: (¼+⅖+3/6) / 6 = 0.192
  - Case 2: [R, R, R, NR, NR, NR]
  - Precision at 1: 1
  - Precision at 2: 1
  - Precision at 3: 1
  - Average precision: 3/6 = 0.5
  - If all 6 results were relevant, then AP = 1
  - MAP = mean (average precision for all queries)
- **Expected Reciprocal Rank** (factoring cascading blocks of relationship).
- **MRR (Mean Reciprocal Rank)**
  - Reciprocal Rank: Let k be the rank of the first relevant hotel. Then reciprocal rank = 1/k
  - Case 1: ¼, Case 2: 1
  - =MRR = mean (reciprocal ranks for all queries)
- **NDCG (Normalized Discounted Cumulative Gain)**
  - Discounted Gain: The lower the rank of a relevant document, the less useful it is to the user since it is less likely to be seen
  - Case 1: [NR, NR, NR, R, R, R]
  - DG @ 4 = 1/log2(4+1)
  - DG @ 5 = 1/log2(5+1)
  - DG @ 6 = 1/log2(6+1)
  - DCG = 1.17 (sum of the 3 scores above)
  - Case 2: [R, R, R, NR, NR, NR]
  - DG @ 1 = 1/log2(1+1)
  - DG @ 2 = 1/log2(2+1)
  - DG @ 3 = 1/log2(3+1)
  - DCG = 2.13
  - **NDCG:** Normalize scores by **dividing by an "ideal"** DCG.

Description of the Methods

LTR model types/loss functions - a few!

- Pointwise methods
  - All standard regression/classification models

- Pair wise evaluations
  - Classify/predict relative order of each pair
    - Ranknet
    - Rankboost
    - Ranking SVM

- List wise evaluations
  - Optimize across entire list of items in query
    - LambdaRank
    - LambdaMART
    - ListNet/Plackett–Luce permutation model/rank order discrete choice model
    - ListMLE
    - StructRank - Cumulative Distribution Network based model
    - Multi-item groupwise ranking (TFranking).
    - Many more......

https://drive.google.com/open?id=1I_jVHZFaT0dZzQ-0MFaOMYE6oEH2fkkD

Ranknet

- Can use any model/function approximator for which the output of the model is a differentiable function of the model parameters
- For a given query, each pair of items U_i and U_j with information (differing labels) is chosen
- Each such pair (with feature vectors x_i and x_j) is presented to the model (typically a neural network model, akin to siamese-network)
- Compute the scores $s_i = f(x_i)$ and $s_j = f(x_j)$, using a NN function approximator.
- Map the output of function approximator to a learned probability of preference (item i preferred over item j)

$$\begin{aligned} l(\mathbf{y}, \mathbf{s}) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{I}_{y_i > y_j} \log_2(1 + e^{-\sigma(s_i - s_j)}) \\ &= \sum_{y_i > y_j} \log_2(1 + e^{-\sigma(s_i - s_j)}) \end{aligned}$$

- Apply cross entropy or logistic loss function to penalize deviation of probabilities from actual preference labels, across all ordered pairs in the dataset.
- Minimize cost/loss.

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/1e34e05e5e4bf2d12f41eb9ff29ac3da9fdb4de3.pdf
https://www.cs.cmu.edu/~pinard/Papers/sigirfp092-donmez.pdf

LambdaRank

- Since overall list level metrics such as NDCG, MRR are not differentiable, identify proxy gradients to act on.
- In other words, use different weights for each pair of items' loss to roll up. And dynamically adjust the loss during the training based on ranking metrics

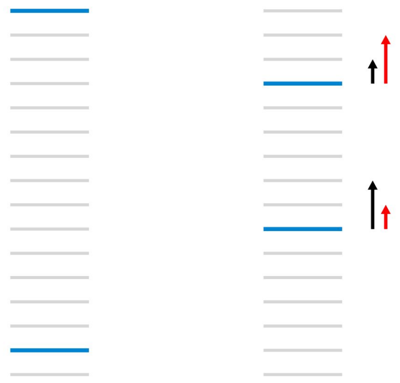


Fig. 1 A set of urls ordered for a given query using a binary relevance measure. The light gray bars represent urls that are not relevant to the query, while the dark blue bars represent urls that are relevant to the query. Left: the total number of pairwise errors is thirteen. Right: by moving the top url down three rank levels, and the bottom relevant url up five, the total number of pairwise errors has been reduced to eleven. However for IR measures like NDCG and ERR that emphasize the top few results, this is not what we want. The (black) arrows on the left denote the RankNet gradients (which increase with the number of pairwise errors), whereas what we'd really like are the (red) arrows on the right.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634&rep=rep1&type=pdf

LambdaRank - NDCG

- Compute approximate lambda gradients for NDCG i.e. appropriate weights for pairwise losses, such that NDCG improves when weighted loss, i.e. total list level loss decreases.

.

$$\mathrm{NDCG} = \frac{1}{\mathrm{maxDCG}} \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log_2(1+i)} = \sum_{i=1}^{n} \frac{G_i}{D_i}$$

where

$$G_i = \frac{2^{y_i} - 1}{\mathrm{maxDCG}}, \; D_i = \log_2(1+i)$$

ΔNDCG is defined as the absolute difference between the NDCG values when two documents i and j are swapped

$$\Delta\mathrm{NDCG}(i,j) = |G_i - G_j|\left|\frac{1}{D_i} - \frac{1}{D_j}\right|$$

$$l(\mathbf{y}, \mathbf{s}) = \sum_{y_i > y_j} \Delta\mathrm{NDCG}(i,j) \log_2(1 + e^{-\sigma(s_i - s_j)})$$

*notation can be confusing while switching between weights vs lambda-gradients.
https://www.cs.cmu.edu/~pinard/Papers/sigirfp092-donmez.pdf
https://storage.googleapis.com/pub-tools-public-publication-data/pdf/1e34e05e5e4bf2d12f41eb9ff29ac3da9fdb4de3.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634&rep=rep1&type=pdf

LambdaRank - MRR

- Lambda-gradients or weights, w.r.t. Mean Reciprocal Rate. Instead of delta ΔNDCG, use ΔRR.

Assume document i is relevant and document j is non-relevant, thus $l_i > l_j$ then ΔRR calculates the difference in the reciprocal rank of the top relevant document as a result of the swap. The r is the rank of the top relevant document in the ordered list.

$$\Delta \mathrm{RR}(r_i, r_j) = \begin{cases} \frac{1}{r_j} - \frac{1}{r} & \text{if } r_j < r \leq r_i \\ 0 & \text{otherwise} \end{cases}$$

ONly if rank of top document shifts, MRR changes. Else no change. Yes, far less smoother function than NDCG

$$l(\mathbf{y}, \mathbf{s}) = \sum_{y_i > y_j} \Delta \mathrm{RR}(r_i, r_j) \log_2(1 + e^{-\sigma(s_i - s_j)})$$

https://www.cs.cmu.edu/~pinard/Papers/sigirfp092-donmez.pdf

ListNET Model

- Defines a loss function using the probability distribution on permutations between items.

  Assume a model outputs ranking scores for items s_j , j =1 to m items. The probability for each possible permutation π of the documents, based on the chain rule, as follows:

$$P(\pi|\mathbf{s}) = \prod_{j=1}^{m} \frac{\varphi(s_{\pi^{-1}(j)})}{\sum_{u=1}^{m} \varphi(s_{\pi^{-1}(u)})},$$

  where $\pi^{-1}(j)$ denotes the document ranked at the j th position of permutation π, and φ is transformation function- linear, exponential, or sigmoid.

  Define model score based permutation probabilities,
  Define $P_y(\pi)$ based on the ground truth label - substitute relevance scores in place of model scores.
  Compute K-L divergence between ground truth permutation and respective permutations' probability distribution.

$$L(f; \mathbf{x}, \Omega_y) = D\big(P_y(\pi) \big\| P\big(\pi|\big(f(w, \mathbf{x})\big)\big).$$

https://drive.google.com/open?id=1I_jVHZFaT0dZzQ-0MFaOMYE6oEH2fkkD

ListMLE - simplied ListNet

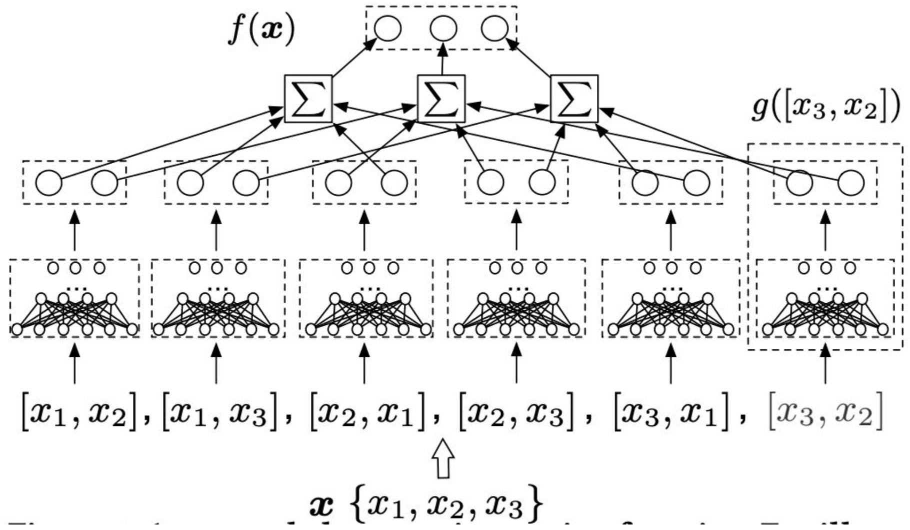
- Do not use all permutations. No K-L divergence.
- Minimize log likelihood of ground truth permutation.

$$L(f; \mathbf{x}, \pi_y) = -\log P\left(\pi_y | f(w, \mathbf{x})\right).$$

https://drive.google.com/open?id=1I_jVHZFaT0dZzQ-0MFaOMYE6oEH2fkkD

TFRanking - Multi-item flexible groupwise scoring functions

- Compare groups of items first and pool together (cross entropy or logistic or sigmoid losses) across all groups.
- Group sizes can be a hyper parameter. Supports multiple loss functions.
- Groupwise scoring method has permutations similar to ListNET but loss function is not akin ListNet, which uses K-L divergence loss.



$f(\boldsymbol{x})$
$g([x_3, x_2])$
$[x_1, x_2], [x_1, x_3], [x_2, x_1], [x_2, x_3], [x_3, x_1], [x_3, x_2]$
$\boldsymbol{x}\ \{x_1, x_2, x_3\}$

https://arxiv.org/pdf/1811.04415.

LambdaMart

- It is a combination of LambdaRank and Multiple Additive Regression Trees (MART).

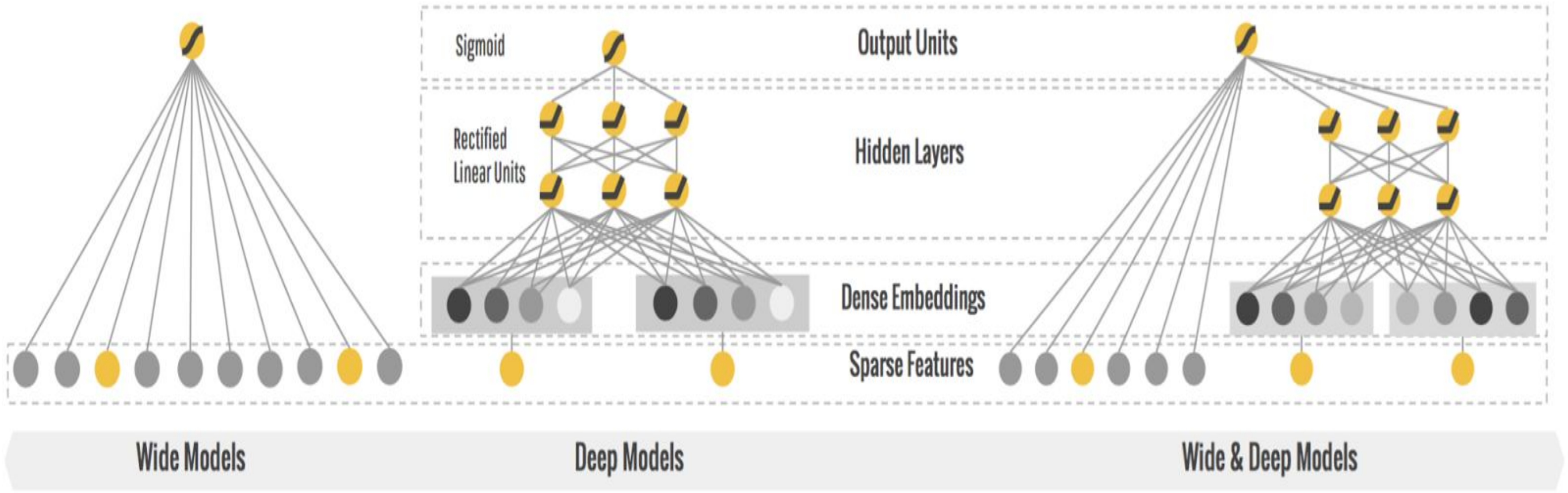
- MART (not unlike gradient boosted trees), are family of models. Typically regression models.

- In LambdaMART, each tree models the λ_i weights/gradients across entire dataset (with bagging tricks), where λ_i is sum over all pairs w.r.t. item i. Since each tree in such additive models fit the gradient of the negative of loss function the gradients are replaced with lambda/weights which are approximate proxies for gradient of in-differentiable Z_{ij}. The ΔZ_{ij} could be Δ $NDCG_{ij}$

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.180.634&rep=rep1&type=pdf

Sample NN architectures from other methodologies

- Most of the NN architectures can be used, by replacing the **final layer, loss function, data structure and appropriate negative sampling**.

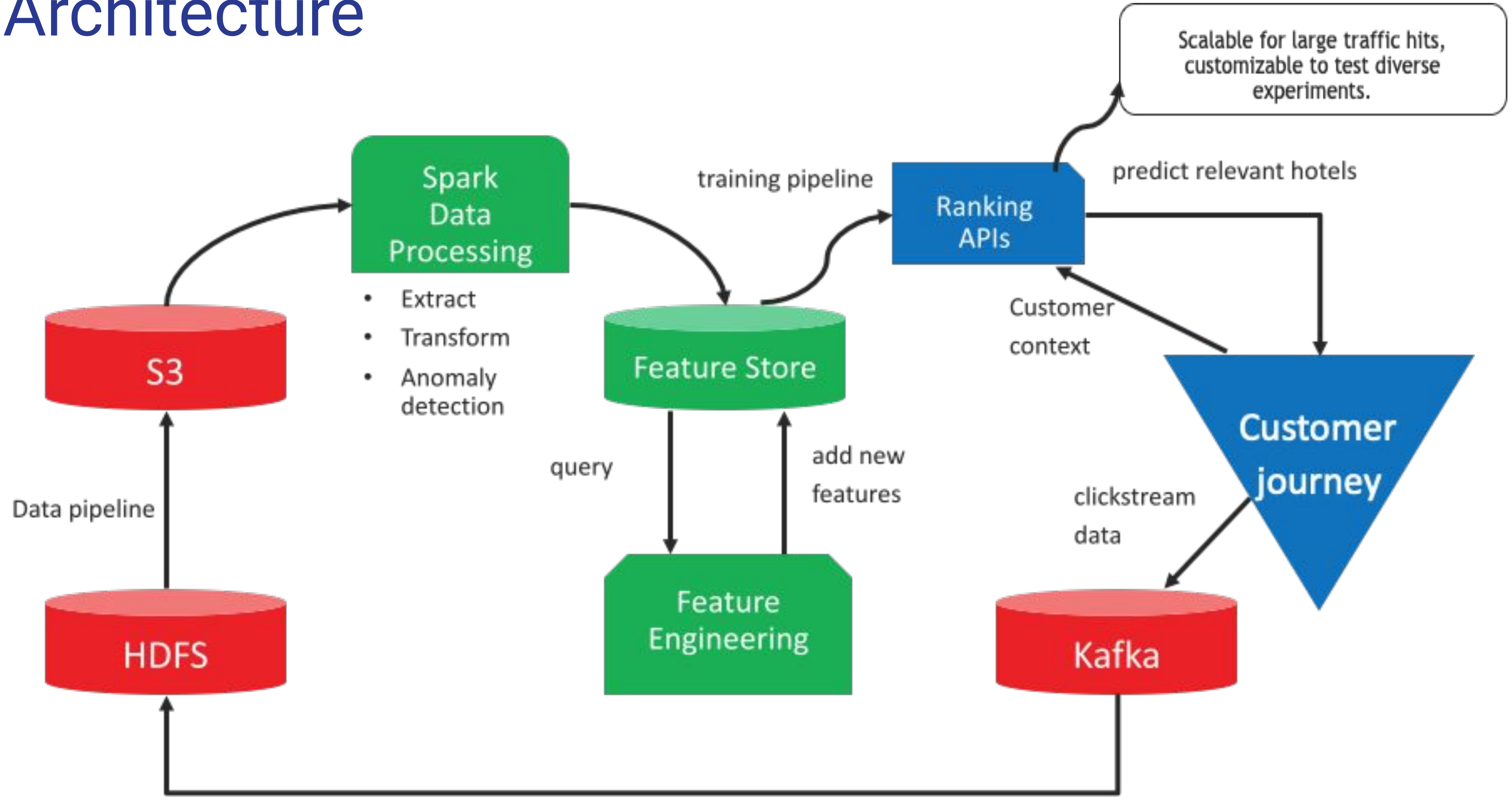


Wide & Deep Learning: Better Together with TensorFlow

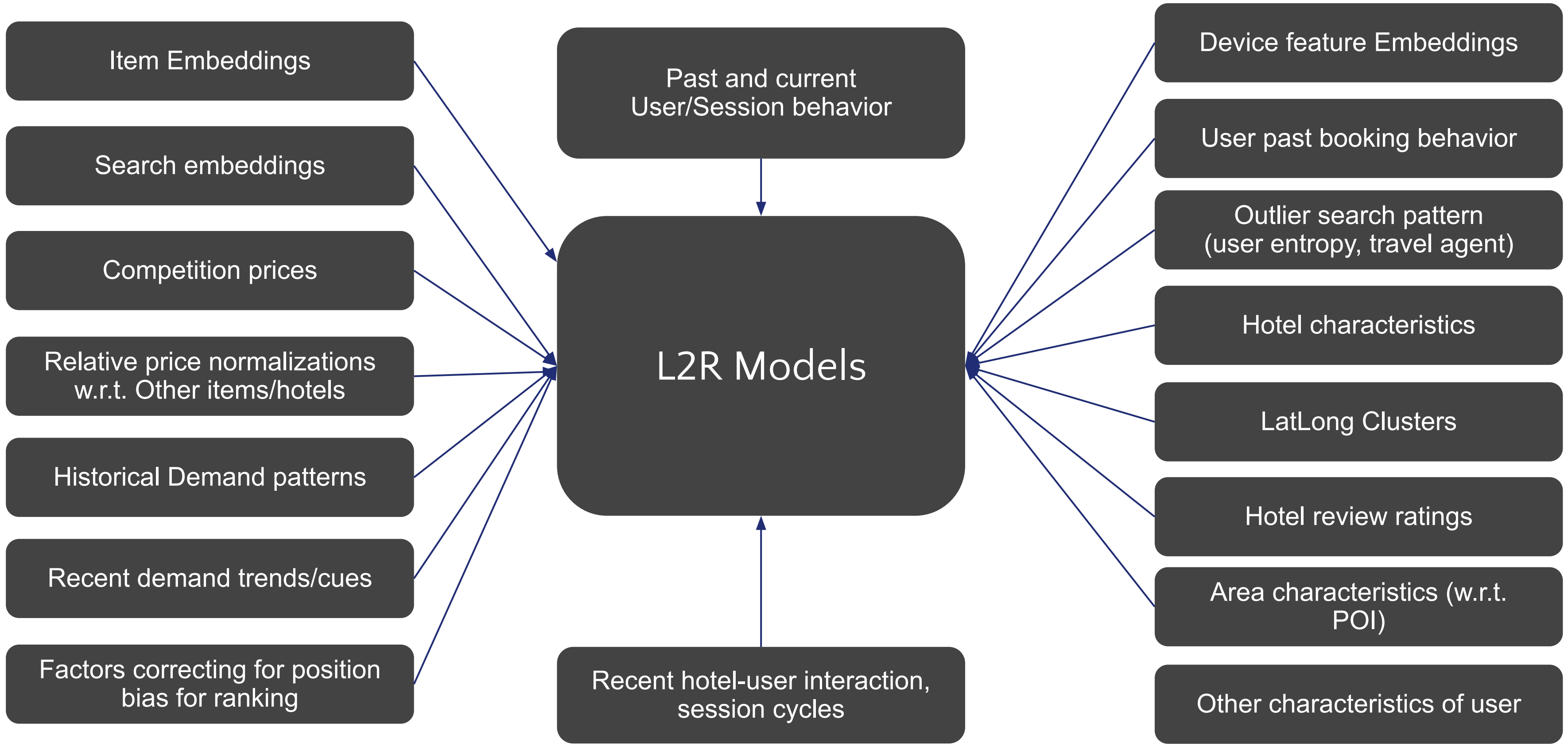
https://arxiv.org/pdf/1606.07792.pdf

Integrating features, platform

Architecture



Submodels /features


Item Embeddings
Search embeddings
Competition prices
Relative price normalizations w.r.t. Other items/hotels
Historical Demand patterns
Recent demand trends/cues
Factors correcting for position bias for ranking
Past and current User/Session behavior
L2R Models
Recent hotel-user interaction, session cycles
Device feature Embeddings
User past booking behavior
Outlier search pattern (user entropy, travel agent)
Hotel characteristics
LatLong Clusters
Hotel review ratings
Area characteristics (w.r.t. POI)
Other characteristics of user

Insights beyond model form

Challenges

1. Poor **quality data**, lack of standardization across groups, inconsistency across multiple components in product and business and market/consumer browsing behavior.

2. The **selection bias** of the users in training data. Consumers see only a subset of hotels in hotel listing page and make decisions. These hotels are used in the model, to understand and predict which hotels are clicked/booked vs not. In production, entire hotel list in the city must be ranked. Patterns learned among subset of selected hotels (biased by nature of hotels) need not generalize for all other unseen hotels. Handle with **negative sampling** strategies.

3. Model **performance vs explainability** trade-offs. Production system has to balance both explainable patterns as well as better overall prediction performance.

4. Since entire hotel set in the city has to be ranked for each user search hit, computations have to be very robust and in some sense as minimalistic as possible in order to lower latency. **Low latency** and model/feature complexity compete against one another.

5. **Sparse features** i.e. the patterns of association are subtle. There is **high noise to signal ratio**.

6. **Drifting data patterns** over time. Nature of queries changes regularly, and also covariates. So online and offline model performance should be monitored

7. **Aesthetic cues/presentation** of items matters. If there are business rules driving such aesthetics, DS should factor them in. **Assess from a consumer** POV.

Experiments change consumer behavior

Sometimes models deployed in production would change the consumer behavior.

Models are often oriented towards predicting what user is likely to book, i.e. taking the final chosen hotel as his/her true interest. However, in ecommerce funnel consumers often may want to browse/”window-shop” around for good looking hotels/experiences, though they may end up clicking/booking only those which they really want.

If we do not show elements which users “window-shop” that too could affect overall likelihood to click/book other hotels.

In addition to the model scores, a few simple “least intrusive” business rules might have to be applied to engage users interest by showing hotels with good ratings/content among the top mix. And, to control for egregious cases.

Importance of multiple evaluation metrics

- Baseline rule based recommenders might push popular and hotels with good content up. Whereas relevance based models would push items/hotels which users, eventually, book.

- Model could learn from local/seasonable attractive patterns and push items up, but some of them may not have good content/reviews/ratings.

This warrants review of multiple metrics.

- NDCG
- MRR
- Average rank of key business metrics (top 30 rankings)
- Average user rating of the recommended hotels (top 30 rankings)
- Average “content score” of the recommended hotels (top 30 rankings)

Current/Future work, within Hotel Ranking

1. Enhance item embeddings
2. Enhance customer embeddings
3. Embedding learners to tap behavior across LOBs.
4. Greatvalue hotel discovery
5. Explainable recommendation systems

Questions