Personalize Expedia Hotel Searches –2nd Place

Jun Wang

Alexandros Kalousis

Outline

- Problem Setting
- Modelling
- Feature Engineering
- Conclusion and Future Work

Learning Problem

- Learning to rank hotels such that purchased and clicked hotels are ranked at top among all hotels associated with a search query.

Formally, let $\boldsymbol{x}_i \in \mathcal{X}$ be the ith search query,

$\mathcal{H}_i = \{h_{i1}, \ldots, h_{im_i}\} \subset \mathcal{H}$ be the set of hotels associated with $\boldsymbol{x}_i$

$\boldsymbol{s}_i = (s_{i1}, \ldots, s_{im_i})$ be their corresponding relevance scores.

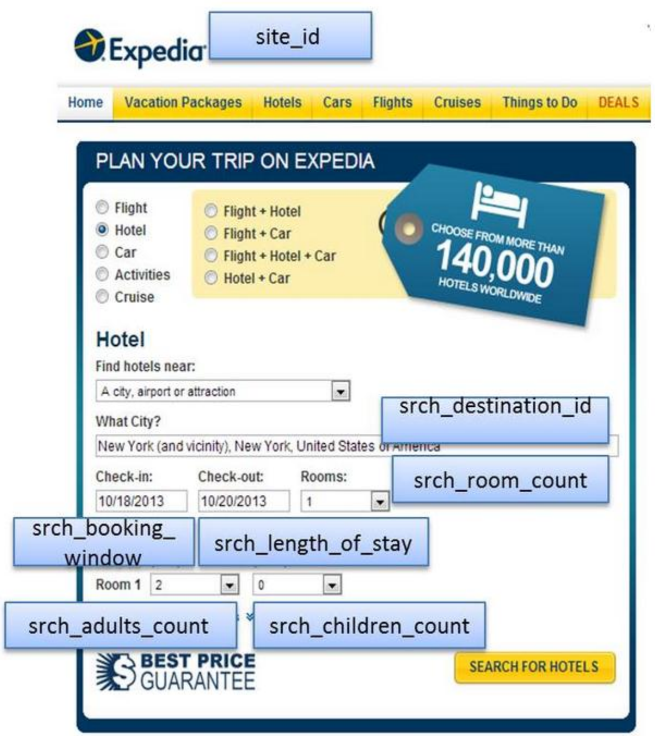
We want to learn a ranking model

$$f : \boldsymbol{x}_i \times \mathcal{H}_i \to f(\boldsymbol{x}_i, \mathcal{H}_i) := \hat{\boldsymbol{s}}_i = (\hat{s}_{i1}, \ldots, \hat{s}_{im_i})$$

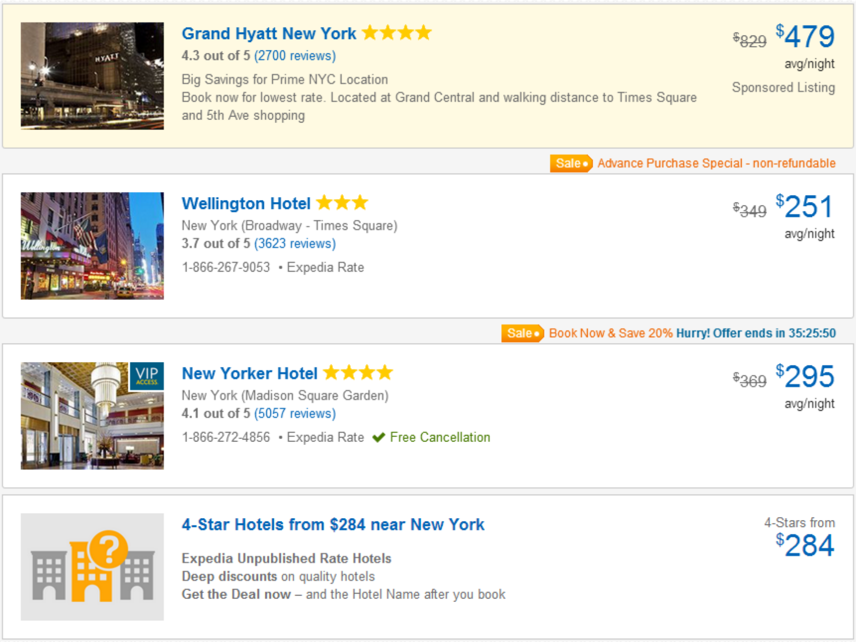
such that the ranking order of predicted scores $\hat{\boldsymbol{s}}_{\boldsymbol{i}}$ are exactly equal to the ranking order of $\boldsymbol{s}_i$.

$$\mathcal{H}_i = \{h_{i1}, \ldots, h_{im_i}\}$$

$$x_i \in \mathcal{X}$$



Ranking Model



Assumption

- The relevance score reflects user's rational behaviour based on the quality of hotels provided by Expedia.

$$\begin{cases} s_{ij} > s_{ik} & h_j \text{ is better than } h_k \text{ for user } i \\ s_{ij} = s_{ik} & h_j \text{ is equally good as } h_k \text{ for user } i \\ s_{ij} < s_{ik} & h_j \text{ is worse than } h_k \text{ for user } i \end{cases}$$

- Any kind of irrational behaviour of user is treated as random noise in the data

Outline

- Problem Setting
- Modelling
- Feature Engineering
- Conclusion and Future Work

Modelling Methodology

- Linear – difficult to handle categorical (discrete) features
  - Linear Regression
  - linear learning-to-rank model (RankSVM and SGD approach)
- Nonlinear
  - Gradient Boosted Trees
  - RandomForest
  - LambdaMART

LambdaMART

- LambdaMART is a learning to rank algorithm based on Multiple Additive Regression Tree (MART).
- It is nonlinear and computationally efficient.

$$\min_{\hat{s}_{ij},\hat{s}_{ik}} \sum_{i} \sum_{jk \in \mathcal{H}_i} |\Delta Z_i^{jk}| \log(1 + e^{\sigma y_{ijk}(\hat{s}_{ij} - \hat{s}_{ik})})$$

$$\text{where} \quad |\Delta Z_i^{jk}| = |NDCG(\hat{\boldsymbol{s}}_i, \boldsymbol{s}_i) - NDCG(\hat{\boldsymbol{s}}_i^{jk}, \boldsymbol{s}_i)|$$

$$y_{ijk} = \begin{cases} -1 & s_{ij} > s_{ik} \\ 0 & s_{ij} = s_{ik} \\ 1 & s_{ij} < s_{ik} \end{cases}$$

NDCG score of $\hat{\boldsymbol{s}}_i$

NDCG score of $\hat{\boldsymbol{s}}_i$, with j, k elements swapped

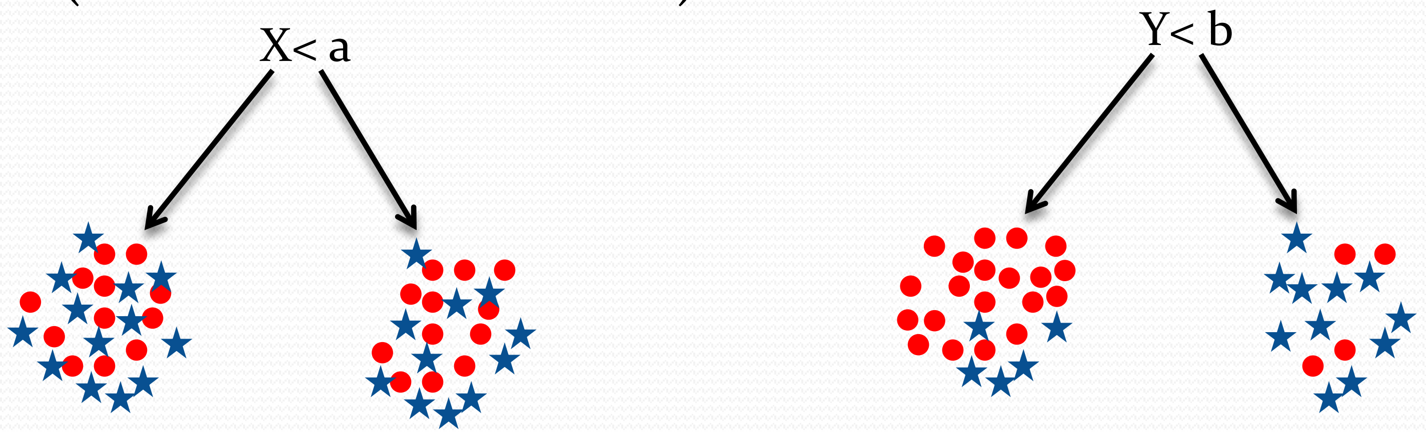
This objective is minimized by learning relevance score through MART.

Outline

- Problem Setting
- Modelling
- Feature Engineering
- Conclusion and Future Work

Feature Engineering (1)

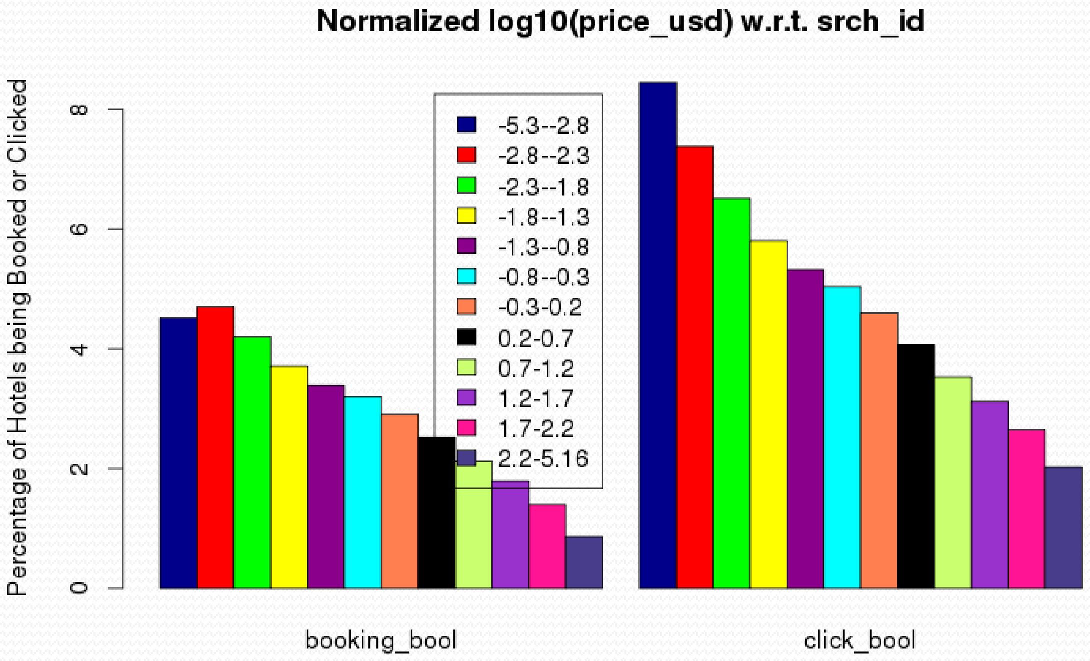
- What is a good representation for LambdaMART (tree-based classifier) ?



Feature Y is more discriminative than feature X for tree classifier

Feature Engineering (2)

- We want feature with monotonic utility w.r.t. target variable!



Feature Engineering Tasks

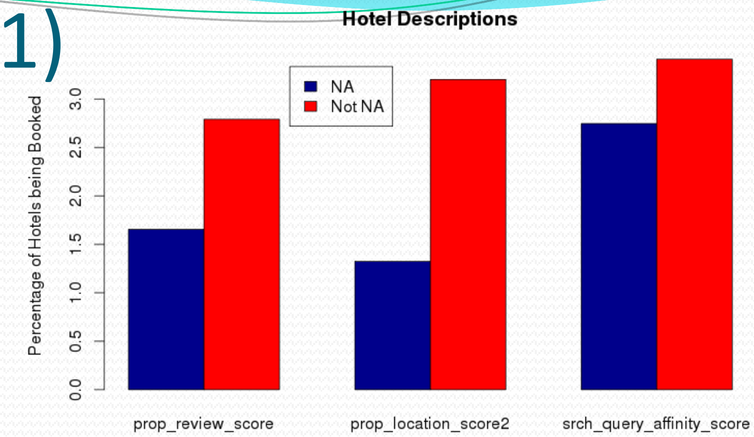
- Missing value estimation
  - On hotel descriptions
  - On user's historical data
  - On competitor descriptions
- Feature extraction
- Feature normalization

Missing Value Estimation (1)

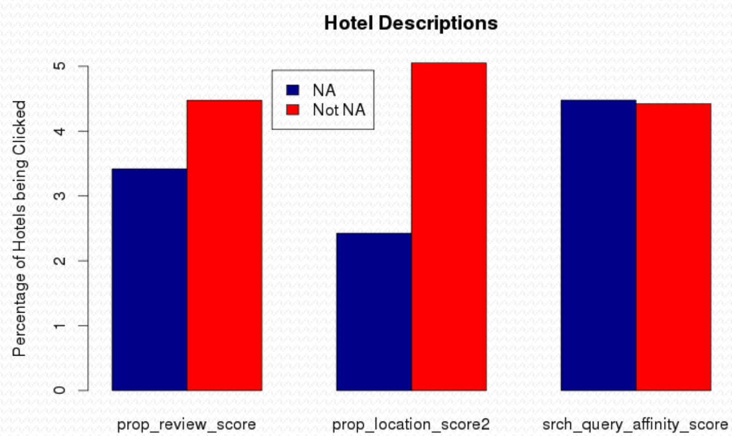
Hotel Descriptions

- User do not like to book and click hotels with missing values.

- Our solution: fill missing value of hotel description with worse case scenario.



booking_bool

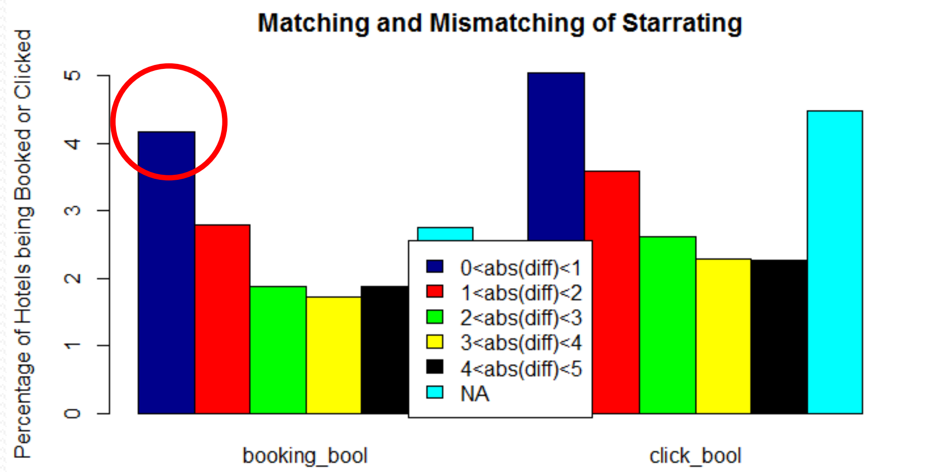


click_bool

Missing Value Estimation (2)

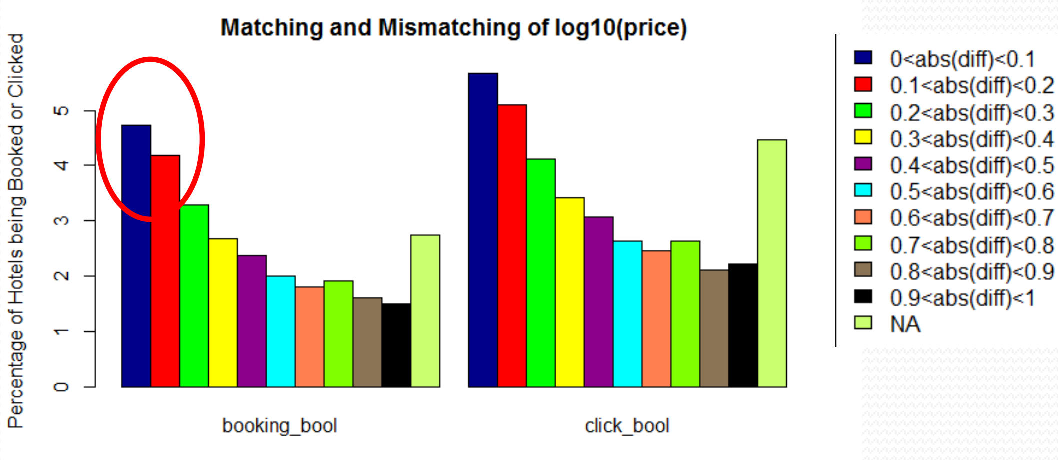
User's Historical Data

- Approximately 95% of user's historical data are missing!!!
- No enough information to model user's historical data



$$starrating\_diff = |visitor\_hist\_starrating - prop\_starrating|$$

- Our solution: highlight the matching and (or) mismatching between historical data and the given hotel data



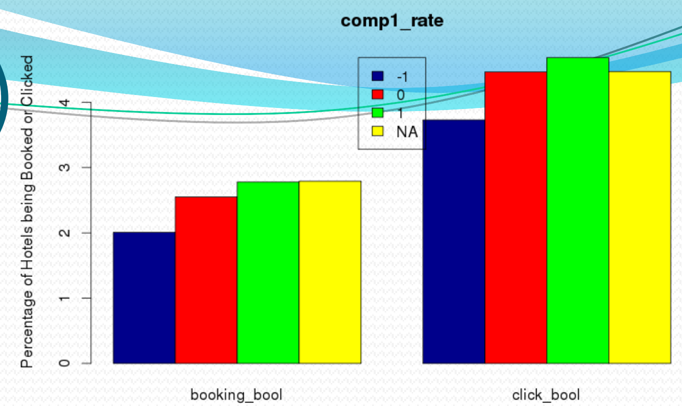
$$usd\_diff = |visitor\_hist\_adr\_usd - price\_usd|$$

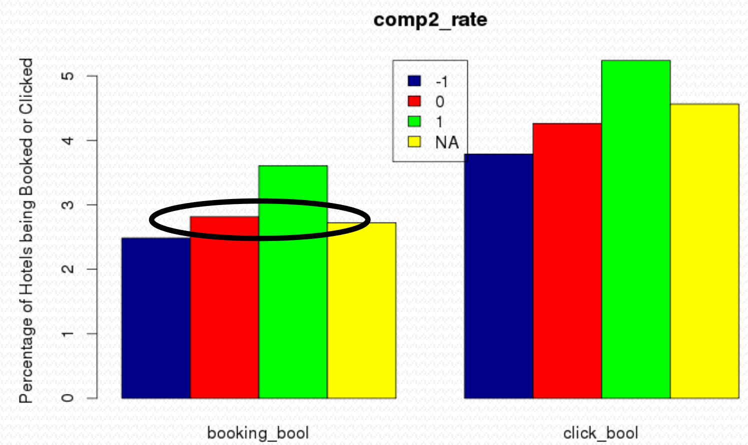
Missing Value Estimation (3)

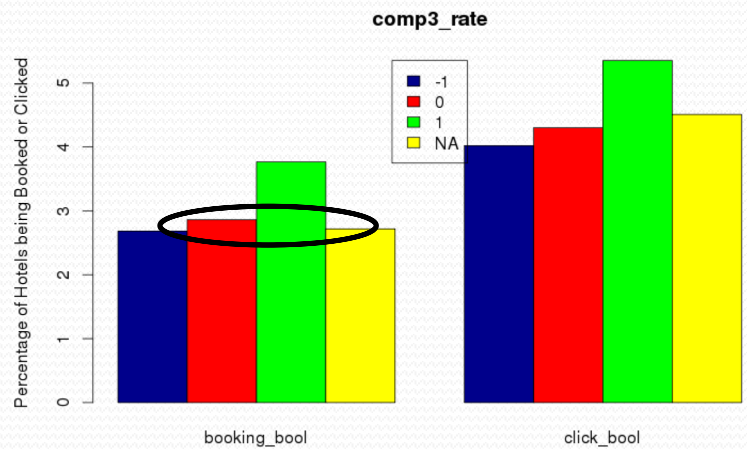
Competitor Descriptions

- Missing value of competitor descriptions are all set to zero

- In total, there is no significant hotel price difference between Expedia and other competitors.







Feature Engineering Tasks

- Missing value estimation
- Feature extraction
  - Hotel quality estimation
  - Non-Monotonicity of Feature Utility
- Feature normalization

Feature Extraction (1)

- On user's historical data

$$starrating\_diff = |visitor\_hist\_starrating - prop\_starrating|$$

$$usd\_diff = |visitor\_hist\_adr\_usd - price\_usd|$$

- On hotel quality
  - The probability of each hotel being booked or clicked varies very much.
  - This probability is estimated by

$$\frac{booking(prop\_id)}{counting(prop\_id)} \quad \text{and} \quad \frac{click(prop\_id)}{counting(prop\_id)}$$

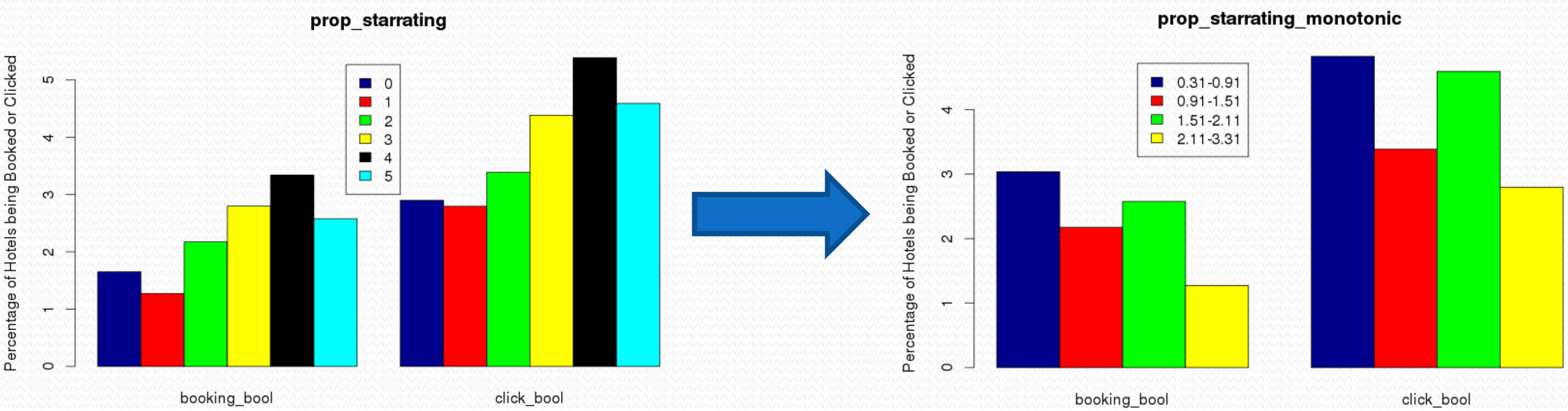
The number of times that prop_id was booked

The number of times that prop_id appeared in the data

Feature Extraction (2)

- On the Non-Monotonicity of Feature Utility
  - Some features have non-monotonicity utility functions.
  - E.g. prop_starrating

$$prop\_starrating\_monotonic = |prop\_starrating - mean(prop\_starrating[booking\_bool])|$$



Feature Engineering Tasks

- Missing value estimation
- Feature extraction
- Feature normalization

Feature Normalization (1)

- Market, e.g. hotel price, varies in different cities, at different times.

- To build a good ranking model for all hotels worldwide, corresponding features should be normalized in an appropriate manner.
  - Removing scaling factors

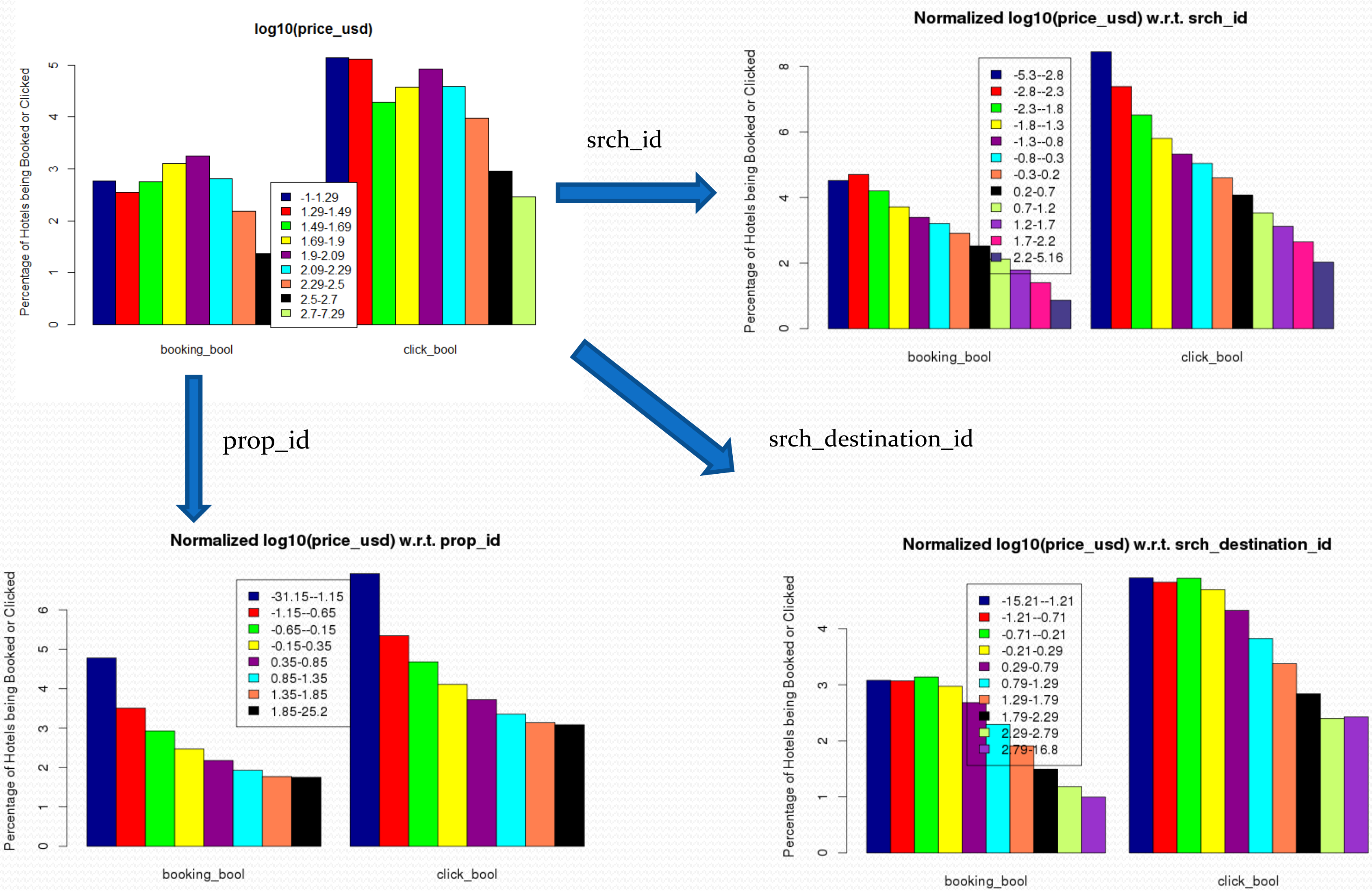
Feature Normalization (2)

- Our solution: normalize hotel and competitor descriptions with respect to different indicators

srch_id, prop_id, month, srch_booking_window, srch_destination_id, prop_country_id

  - srch_id: compare the quality of hotels in the impression list
  - prop_id: compare the quality of hotels in their own history
  - srch_des_id: compare the quality of hotels in a given region
  - month: compare the quality of hotels in a given time
  - srch_book_window: compare the quality of hotels for a given booking window
  - ...

Example -- Hotel Price

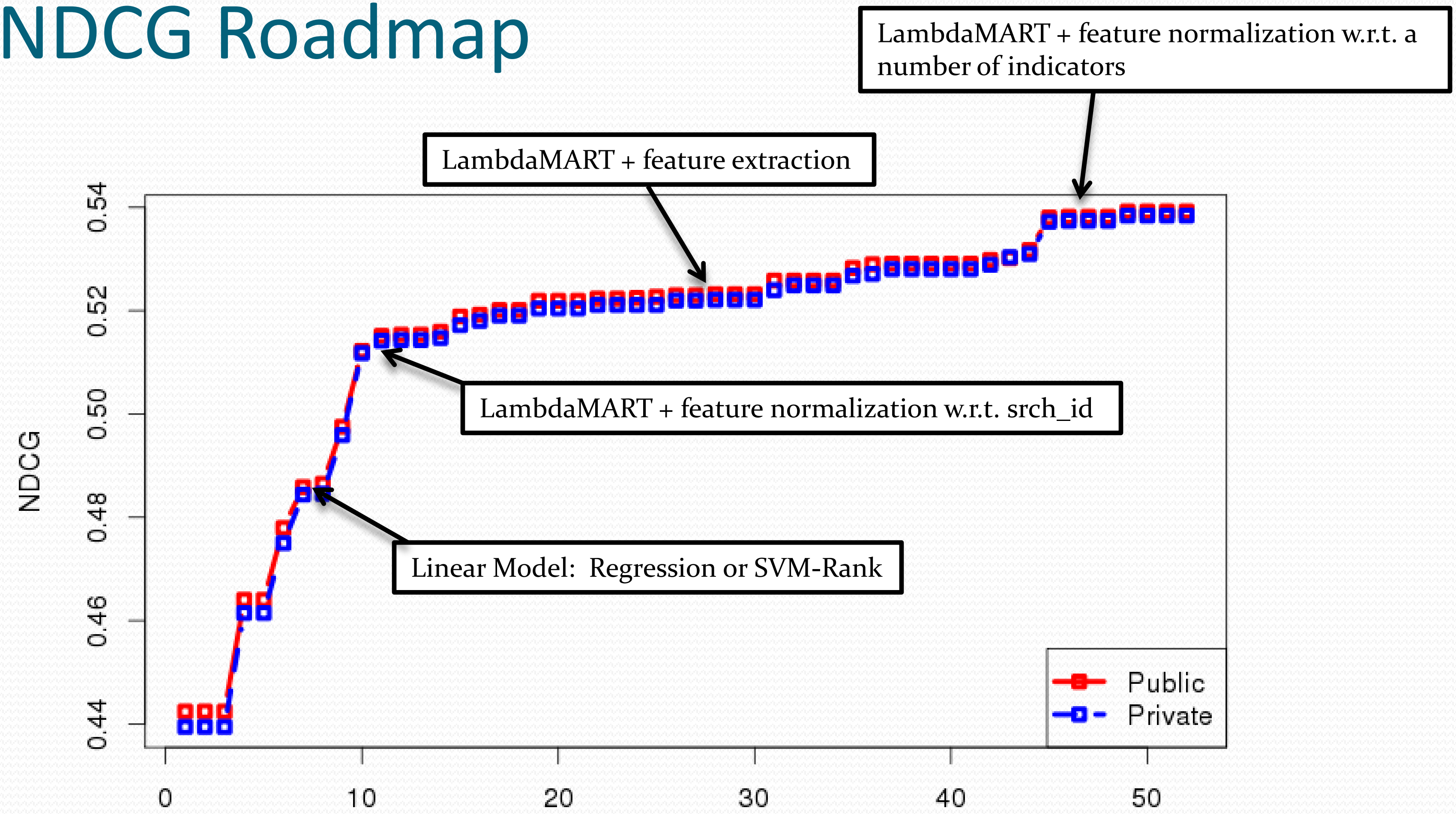


Learning

- Training Data
  - Num. of instance: 9M
  - Training data 80% and Validation data 20%
  - Num. of feature: 300 (after feature engineering)

- Methodology
  - Linear model: regression or SVM-Rank
  - Nonlinear model: LambdaMART

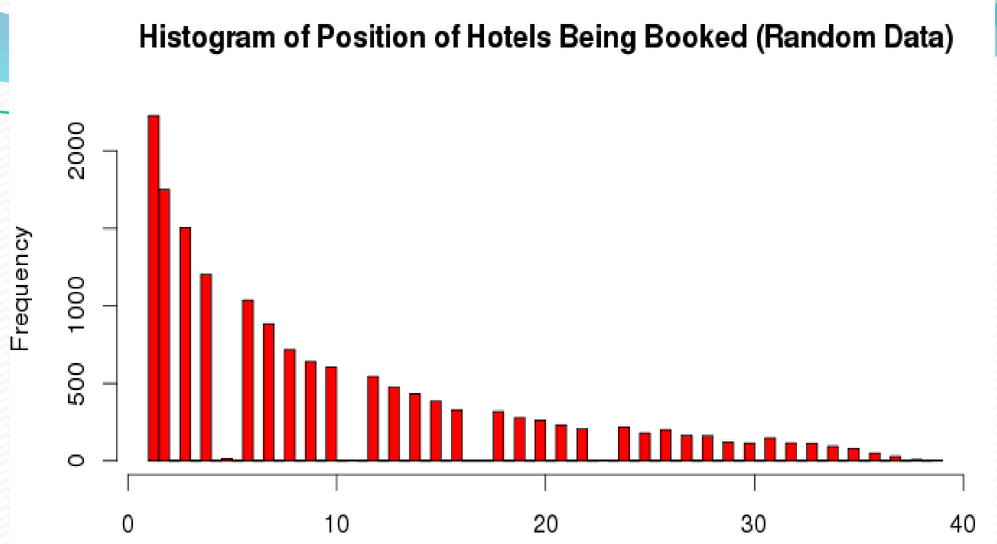
NDCG Roadmap



What matters?

- Feature Engineering
- Linear vs. Nonlinear
- Number of learning instances

Future works



- User's behaviour is not rational
  - Modelling the position bias
- Feature engineering
  - Missing value estimation in a more principle way
  - Normalizing features with multiple indicators
  - Remove redundant features
- Modelling Methodology
  - Improving LambdaMART
  - New modelling formulation based on MART