Trivago Challenge

Recommenders

Alejandro Castrelo
Gerard Marrugat

Data Preprocessing

- From the item_metadata.csv we extracted the properties that we considered more relevant at the item (hotel) level (we called this part “metadata”).

  ‘2 Star', '3 Star', '4 Star', '5 Star', 'Beach', 'Free WiFi (Rooms)', 'Gym', 'Massage', 'Nightclub', 'Romantic', 'Sauna', 'Spa Hotel', 'Good Rating', 'Very Good Rating'

- The train.csv was cleaned. We keep just the sessions in which the user does a clickout action.

  The features extracted from the interaction of the user during the session are:

  - The price of the hotel shown to the user(impressions). We divide it into three ranges: ***Price less than 100€, Price less btw 100€ and 300€, Price more than 300€.***
  - The position of each clickout in the impressions list, ***position***
  - The number of clickouts per item using the get_popularity provided function, ***n_clicks***.
  - If the user has interacted during the session with the impression items, ***Has interacted with impression items***. This feature is set at the (session_id, impression) level, for those features the user has interacted with the feature is set to 1, otherwise it is set to 0.

Training Process

Finally we have prepared a dataset with 20 features.

Price less than 100€	Price btw 100€ and 300€	Price more than 300€	Has interacted with impression items	2 Star	3 Star	4 Star	5 Star	Beach	Free WiFi (Rooms)	Gym	Massage	Nightclub	Romantic	Sauna	Spa Hotel	Good Rating	Very Good Rating	n_clicks	position
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	15
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	7
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	7
1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	23
1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	23
1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	5	9
1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	5	9

We noticed that adding or removing features from the metadata doesn’t have much impact, so we wanted to focus on the ones we thought that were more important, such as price, position and interactions.

Test and Evaluation

When the test_set is passed to the our model (XGBoost), this returns the probability of each item to be clicked by the user.

This results are grouped by session and listed in order.

	user_id	session_id	timestamp	step	item_recommendations
0	000324D9BBUC	89643988fdbfb	1541593942	10	923407 1729121 1050792 97171 353141 106315 218...
1	0004Q49X39PY	9de47d9a66494	1541641157	1	2213014 3184842 10213134 4504242 4486372 38120...
2	0004Q49X39PY	beea5c27030cb	1541561202	1	3202894 2292254 1984229 2227896 2227026 195366...
3	00071784XQ6B	9617600e1ba7c	1541630328	2	3498754 22721 3067559 16121 22727 22854 22819 ...
4	0008BO33KUQ0	2d0e2102ee0dc	1541636411	6	507861 2176280 1669587 502066 1352530 4342348 ...

Test and Evaluation

The best result achieved when submitting the recommendations is a score of 0.5459.

Time	Status	Score
2019-04-09 21:30:00 (submission_9.csv)	valid	0.5459

Things to improve

- Process pseudo-features from the features extracted from the metadata_set.
- Improve the parameters of the model.
- Try different metrics when training the model.
- Add more features.

Conclusions

- We have seen that the most important part in this type of problems is feature extraction.
- We would’ve liked to add more features but the time is very limiting and our baseline knowledge on feature engineering (pandas for instance) was almost zero.
- We realized that the model is not very crucial, just use the best one available.
- In this kind of problems, the memory management can be critical. We had a lot of issues with memory and we had to optimize the code and limit the number of features (we were using all 157 features from metadata at the beginning).