RecSys Challenge 2016: job recommendations based on preselection of offers and gradient boosting

Andrzej Pacuk
Institute of Informatics,
University of Warsaw, Poland
apacuk@mimuw.edu.pl

Piotr Sankowski
Institute of Informatics,
University of Warsaw, Poland
sank@mimuw.edu.pl

Karol Węgrzycki
Institute of Informatics,
University of Warsaw, Poland
k.wegrzycki@mimuw.edu.pl

Adam Witkowski
Institute of Informatics,
University of Warsaw, Poland
a.witkowski@mimuw.edu.pl

Piotr Wygocki
Institute of Informatics,
University of Warsaw, Poland
wygos@mimuw.edu.pl

ABSTRACT

We present the Mim-Solution's approach to the RecSys Challenge 2016, which ranked 2nd. The goal of the competition was to prepare job recommendations for the users of the website Xing.com.

Our two phase algorithm consists of candidate selection followed by the candidate ranking. We ranked the candidates by the predicted probability that the user will positively interact with the job offer. We have used Gradient Boosting Decision Trees as the regression tool.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommender systems, RecSys Challenge 2016, Gradient Boosting.

1. INTRODUCTION

The Recsys Challenge is an annual competition of recommender systems. The 2016th edition[1] was based on data provided by xing.com – a platform for business networking. On Xing, users search for job offers that could fit them. Each user has a profile containing information such as: place of living, industry branch and experience (in years). Job offers are described by a related set of properties. There are various ways in which a user can interact with a job offer (called item): by clicking it, as well as bookmarking interesting ones, replying to offers and finally by deleting a recommendation.

---

[1] http://2016.recsyschallenge.com/

The task in the challenge was to predict for a given XING user 30 items that this user will positively interact with (click, bookmark or reply to).

Dataset.

The dataset consisted of properties of users and items, interactions of the users (which items user clicked, bookmarked, replied to or deleted) and impressions (items shown to users by XING recommender system). The interactions and impressions were gathered from a 3 month period.

All data was anonymized by changing all properties to numerical values and adding an unknown number of artificial users, items, interactions and impressions.

We were also given a set of both user and item properties. Common attributes of users and items were: career level, discipline id, industry id, country and region. Besides that users had following attributes: jobroles, experience n entries class, experience years experience, experience years in current, edu degree, edu fieldofstudies. Items had attributes: title, latitude, longitude, employment, created at and active during test[2]. Values of those attributes we will denote by, e.g., $\mathsf{career\_level}(i)$ for a given item i.

Each impression is a tuple containing: user ID, item ID, number of the week in which impression occurred. Each interaction contains: user ID, item ID, interaction type (click, bookmark, reply, delete) and timestamp of the interaction. We will denote the positive interactions of a user u by Int_u, the negative interactions by Del_u and impressions by Imp_u.

10% of users were target users: users which we needed to compute the predictions for. The predicted items had to come from a subset (24%) of all items (these were the job offers open during the period). Exact datasets sizes are presented in Table 1.

Table 1: Statistics of the Recsys Challenge 2016 training dataset.

dataset	size
impressions	1 078 627 301
interactions	8 826 678
users (target users)	1 500 000 (150 000)
items (items active during test)	1 358 098 (327 003)

---

[2] A more detailed specification of the dataset can be found on https://github.com/recsyschallenge/2016/blob/master/TrainingDataset.md

Evaluation measure.

The *ground truth* is a mapping that assigns to each user, the set of items he interacted positively with during the test week. Let T be the set of target users, I be the set of all items, and $G : T \to 2^I$ the ground truth. We will also denote the sequence of items predicted for a user by $\mathsf{pred}\,(u)$.

The quality of recommendations was measured by a function:

$$\mathsf{score}(\mathsf{pred}, G) = \sum_{u \in T} \mathsf{userScore}(\mathsf{pred}(u), G(u)), \qquad (1)$$

where:

$$\mathsf{userScore}(\mathsf{pred}(u), G(u)) =$$
$$20 \cdot \big[\mathsf{p}(\mathsf{pred}(u), G(u), 2) + \mathsf{p}(\mathsf{pred}(u), G(u), 4) + $$
$$\mathsf{us}(\mathsf{pred}(u), G(u)) + \mathsf{r}(\mathsf{pred}(u), G(u)) \big] + $$
$$10 \cdot \big[\mathsf{p}(\mathsf{pred}(u), G(u), 6) + \mathsf{p}(\mathsf{pred}(u), G(u), 20) \big]$$

and for a sequence $\bar{a} = a_1, a_2, \ldots, a_{30}$, a set B and a natural number k:

$$\mathsf{p}(\bar{a}, B, k) = |\{a_1, a_2, \ldots, a_k\} \cap B|/k$$
$$\mathsf{r}(\bar{a}, B) = |\{a_1, a_2, \ldots, a_{30}\} \cap B|/\min(1, |B|)$$
$$\mathsf{us}(\bar{a}, B) = \min(1, |\{a_1, a_2, \ldots, a_{30}\} \cap B|).$$

Note that: $\mathsf{p}(\bar{a}, B, k)$ is the precision for the first k elements of a, $\mathsf{r}(\bar{a}, B)$ is the recall and $\mathsf{us}(\bar{a}, B)$ is user success (1 if we predicted at least one item for this user correctly, 0 otherwise).

Solutions were evaluated by the submission system and contestants received instant feedback with score value calculated on a sample of $\frac{1}{3}$ of target users.

2. OUR SOLUTION

Our solution consists of two parts:

1. for each user we calculate a set of candidate items, much smaller than the whole set I (only those candidates were considered when creating a submission),

2. learn for each (user, candidate) pair (u, i) the probability $\mathbb{P}\,[i \in G(u)]$ that the user will interact with this item.

We submitted, for each user, 30 items with the highest predicted probability of interaction, excluding items that the user deleted. Limiting the number of considered items per user allowed us to substantially decrease the time required to train a model and prepare a submission. Examining all possible user-item pairs (150.000×327.000 pairs) was infeasible considering our resources. To learn the probabilities $\mathbb{P}\,[i \in G(u)]$, we have used Gradient Boosting Decision Trees (GBDT) [1], optimizing the logloss measure. We learned the probabilities instead of directly optimizing the score function since it does not give results for a single user-item pair. The schema of our solution is presented in Figure 1.

Our solution was ranked 2nd in the competition, scoring 675985.03 and 2035964.16 points on the public and private leaderboard respectively. To put this into perspective, submitting for each user u, sorted from most recent Int_u but without items from Del_u (adding Imp_u if there were less than 30 unique interactions) achieved a score of 495k on the public leaderboard. All of the computation were performed on 12 cores (24 threads), 64 GB RAM Linux server.
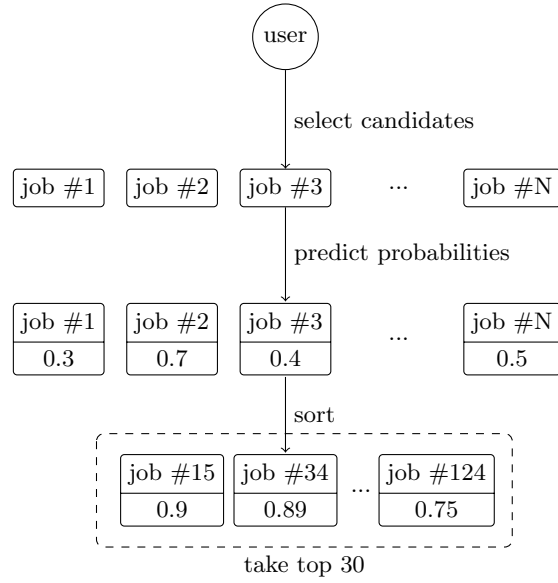

Figure 1: Schema of our solution.

2.1 Training set

In this problem, there was no clearly defined training set and the first challenge was to create it. Since our task was to predict users' interactions in the week following the end of the available data, we trained our model on all the data except the last available week and then used this last week data to compute the training ground truth G to evaluate the model. This way, we could calculate the score and determine if we are making progress without sending an official submission (every team was allowed max. 5 submissions per day).

There was an overlap between the training data and the test data (the full dataset). Both candidates and features were computed separately for the training set and the full dataset.

2.2 Candidate items selection

Since there were 150k target users and more than 300k items, making a prediction for each user-item pair would take too long. To address this issue, we chose for each user u a set of promising items, which we called candidates.

To define candidates, we will need a few notions of similarity. The Jaccard coefficient between two sets A and B is $J(A, B) \equiv \frac{|A \cap B|}{|A \cup B|}$. Interactions (impressions) similarity between two users u, u', denoted $\mathsf{Int\text{-}sim}(u, u')$ ($\mathsf{Imp\text{-}sim}(u, u')$), is the Jaccard coefficient between the sets of items from their positive interactions (impressions). For example, $\mathsf{Int\text{-}sim}(u, u') = J(\mathsf{Int}_u, \mathsf{Int}_{u'})$.

For items i, i', we will denote $\mathsf{common\text{-}tags}(i, i') \equiv |\mathsf{tags}(i) \cap \mathsf{tags}(i')|$ and $\mathsf{common\text{-}title}(i, i') \equiv |\mathsf{title}(i) \cap \mathsf{title}(i')|$.

For an user u, the candidates were:

1. Int_u sorted by the week of occurrence (most recent first) and the number of interactions,

2. Imp_u sorted the same way as 1,

3. $\mathsf{Int}_{u'}$ for users u' sorted by $\mathsf{Int\text{-}sim}(u, u')$,

4. $\mathsf{Imp}_{u'}$ of users u' sorted by $\mathsf{Imp\text{-}sim}(u, u')$,

5. items i sorted by $\max_{i' \in \mathsf{Int}_u}$ common-tags$(\mathsf{i}, \mathsf{i}')$, similarly for $\max_{i' \in \mathsf{Int}_u}$ common-title$(\mathsf{i}, \mathsf{i}')$, $\max |\mathsf{title}(i') \cap \mathsf{tags}(i)|$ and $\max |\mathsf{title}(i) \cap \mathsf{tags}(i')|$,

6. same as 5, just with max taken over $i' \in \mathsf{Imp}_u$,

7. items i sorted by $|\mathsf{jobroles}(u) \cap \mathsf{tags}(i)|$,

8. items i sorted by $|\mathsf{jobroles}(u) \cap \mathsf{title}(i)|$,

9. the most popular items (globally, this list was the same for all users).

The popularity of an item was measured by the number of interactions of all users with this item. From each category, we took 60 best candidates for each user. Using more candidates per category did not change the score substantially, but increased the training time. This approach gave us 43M (user,item) pairs, on average just short of 300 candidates per user. On the training set, candidates chosen this way covered 37% of the training ground truth.

Of course, we could have chosen a different notions of similarity between users/items. In particular we considered similarity based on user and item properties such as region, industry, etc. Candidates based on those measures of similarity did not improve the score sufficiently, probably due to anonymization of the data.

2.3 Learning the probabilities

We wanted to construct a model that given a (u, i) pair and values of features for this pair will compute the probability $\mathbb{P}[i \in G(u)]$. In order to estimate the probabilities, we have used XGBoost[3], a machine learning library implementing GBDT.

For training the ranking model, we split all the users with at least one item in the training ground truth into two sets of equal size. The users from the first set were used for training XGBoost model, and the users from the second set were used for validation of the model. The training file (and validation file) contained for each user:

- all the training candidate items, which occurred in the training ground truth (positive candidates)

- and up to 5 training candidate items, which did not occur in the training ground truth (negative candidates).

Just after deadline for submitting solutions we observed, that training a model on all users with all positive and $1/4$ of negative candidates improves our score by 6.5k points on unofficial public leaderboard[4].

Evaluation.

We used maximum likelihood as the objective function optimized by GBDT. Additionally we verified models by computing the score function based on the validation part of the training ground truth. Major improvements in this validation score translated to comparable improvements on the score achieved via the submission system. We measured on training ground truth that our way of ordering previously selected candidate items achieved score which was 77.5% of best possible result based only on those candidates.

[3] https://github.com/dmlc/xgboost
[4] Note that difference between 1st and 2nd place was 5.7k, so extending training file earlier could result in winning a competition by our team.

Parameters tuning.

We found that the optimal XGBoost parameters for our task were:

- max_depth, the maximum depth of a tree, in range $[4, 6]$,

- min_child_weight, the minimum weight of a node, in range $[4, 6]$,

- eta, the learning rate, equal to 0.1,

- gamma, the minimum loss reduction to make a node partition, equal to 1.0,

- num_round, the number of rounds, equal to 1000 (validation logloss did not improve after 1000 rounds).

2.4 Features

Each feature is a function that for user u and item i maps the pair (u, i) to some real number. We ended up with 273 features. Many of these features were highly correlated, however, we observed that redundant features do not reduce the quality of the model. Many features differ only in:

- used different events source: impressions instead of interactions, only positive/negative interactions, only interactions of one type,

- instead of Jaccard coefficient we used size of sets intersection $|A \cap B|$ and vice-versa,

- used various aggregate functions: maximum, minimum, sum, average, count or unique count (count without duplicate entries).

Because of this we will only describe the important groups of features. There are 12 such groups. Table 2 summarizes the importance of the features we used. Ideas for features were inspired by papers of previous RecSys Challenge [2, 3] and similar competitions hosted on Kaggle platform[5].

Event based features are percentages of items from Int_u that had some property (e.g., item's career level) equal to item's i corresponding property. Dually we also used the percentages of users from $\mathsf{Int\text{-}users}(i)$ (i.e., users who interact a given item i) that had some attribute equal to user u attribute. From this group of features, the best were those based on item tags and item title fields.

Item global popularity is the number of times item i was clicked by any user. Additionally we computed the trend of popularity: clicks count in the last week divided by the clicks count in the previous last week. Another way to observe week trend was to compute trend based on events from last and previous last Mondays, Tuesdays, . . . , Sundays.

Colaborative filtering most similar are features that measure similarity between the item i and the items from Int_u using I-sim, and dually between the user u and users that interacted with i, using Int-sim. Formally, these are given by formulas:

- $\max_{i' \in \mathsf{Int}_u} I\text{-sim}(i, i')$,

- $\max_{u' \in \mathsf{Int\text{-}users}(i)} \mathsf{Int\text{-}sim}(u, u')$.

User total events is just $|\mathsf{Int}_u|$ both with repetitions of items and without. We also used this feature limited to the user's last week of activity.

Seconds from last user activity are the differences, in seconds, between:

[5] www.kaggle.com

Table 2: Feature groups importance: sum of how many times each feature was used to split the data (based on XGBoost's model fscore metric). The main features groups are in bold, additionally we list important subgroups.

feature group	importance
event based	**5499**
tags + title	946
item global popularity	**2913**
trend	1392
weekday	599
cf most similar	**1410**
item clicked by user	790
user who clicked item	620
user total events	**1066**
in last week	494
seconds from last user activity	**882**
max common tags with clicked item	**527**
position on the candidates list	**456**
user-item events count in last week	**288**
item properties	**375**
created at	166
longitude	125
latitude	84
content based user-item similarity	**83**
career level difference	47
common user job roles and item titles	15
distance to the closest clicked item	**55**
items cluster	**7**

- the last time u clicked i and the maximal timestamp in data,
- the last time u clicked i and the last time u clicked any item,
- the last time u clicked any item and the maximal timestamp in data.

Analogous features based on impressions were timed in weeks.
Max common tags with clicked item are features that return the maximum number of common tags between item i and items from Int_u: $\max_{i' \in \mathsf{Int}_u} \mathsf{common\text{-}tags}(\mathsf{i}, \mathsf{i}')$. We used those features with impressions instead of interactions and with $\mathsf{common\text{-}title}(\mathsf{i}, \mathsf{i}')$ instead of $\mathsf{common\text{-}tags}(\mathsf{i}, \mathsf{i}')$.
Position on the candidates list is position number of item i on user's u candidates list. There was separate feature for each candidate algorithm (see Subsection 2.2).
User-item events count in last week is the count of how many times user u clicked item i in the last week of this users activity, or in the last week from the dataset.
Item properties are values of item's i attributes: created at, latitude and longitude.
Content based user-item similarity is a group of features based only on properties of u and i:

- $\mathsf{career\_level}(i) - \mathsf{career\_level}(u)$,
- $|\mathsf{jobroles}(u) \cap \mathsf{title}(i)|$,
- $|\mathsf{jobroles}(u) \cap \mathsf{tags}(i)|$.
- 1 if $\mathsf{attr}(u) = \mathsf{attr}(i)$ else 0, for the rest of matching attributes.

Distance to the closest clicked item is the Euclidean distance between the location of item i and the location of the closest item from Int_u.

Item cluster is a boolean feature, true if the item i is in a cluster of some item from Int_u, where $i \in \mathsf{cluster}(i')$ if $i \neq i'$ and there exists a user u that clicked both i and i' within 10 minutes. This feature is another variation of the similarity between items. This time, we say that two items are similar if a user clicked both of them in a similar time. The motivation is simple: if both those items were interesting for some user, then they probably have something in common that appeals to this user.

Some of the features (especially time-related) had to be calculated separately and with care for the training dataset and the full dataset. For example, we had a feature "timestamp of the last interaction". This feature on training instance should be shifted in order to cover the same range of values as on the full dataset.

2.5 Blending

In the final step in the construction of our solution we merged two models (max_depth 4, min_child_weight 4 for the first model and max_depth 6, min_child_weight 6 for the second) with the best scores. Since both models were similar and XGBoost based, we took for each (u, i) pair, the arithmetic mean of probabilities $\mathbb{P}[i \in G(u)]$ calculated by those models. Finally, for each target user, we sorted candidate items by these averaged probabilities and selected the top 30.

3. CONCLUSIONS

We have presented Mim-Solution's approach to the 2016th RecSys Challenge. We used XGBoost to predict probabilities that a user will be interested in a job offer, but only for preselected offers. This allowed to efficiently train and evaluate the model as well as use complex and robust features.

Even tough our score was good (28.5 % of best possible score measured on our train data), there was definitely a lot of room to improve. One easy improvement, mentioned already in the paper, is increasing the size of the training set. There was also some place for improvements in ordering of candidate items, but we suspect that twice bigger room for improvement was in expanding the set of candidates (we achieved 77.5% and 37% of best possible results in the layer of sorting and selecting candidates respectively).

4. REFERENCES

[1] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.
[2] P. Romov and E. Sokolov. Recsys challenge 2015: Ensemble learning with categorical features. In *Proceedings of the 2015 International ACM Recommender Systems Challenge*, RecSys '15 Challenge, pages 1:1–1:4, New York, NY, USA, 2015. ACM.
[3] M. Volkovs. Two-stage approach to item recommendation from user sessions. In *Proceedings of the 2015 International ACM Recommender Systems Challenge*, RecSys '15 Challenge, pages 3:1–3:4, New York, NY, USA, 2015. ACM.