A Combination of Simple Models by Forward Predictor Selection for Job Recommendation

Dávid Zibriczky
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Budapest, Hungary
david.zibriczky@gmail.com

ABSTRACT

The present paper introduces a solution for the RecSys Challenge 2016. The principle of the proposed technique is to define various models capturing the specificity of the dataset and then to subsequently find the optimal combinations of these by considering different user categories. The approach follows a practical way for the fine-tuning of recommender algorithms, highlighting their components, training- and prediction time. Based on forward predictor selection, it can be shown that item-neighbor methods and the recommendation of already shown or interacted items have great potential in improving the offline accuracy. The best composition consists of 11 predictor instances that achieved the third place with 665,592 leaderboard score and 2,005,263 final score.

CCS Concepts

•**Information systems** → **Recommender systems;**

1. INTRODUCTION

In 2016, the RecSys Challenge aimed to compete solutions for a job recommendation in the social network of XING. For the competition, XING provided an anonymized dataset about their users, items, recommendations and corresponding user interactions in their system. The goal of the challenge was to estimate on which jobs the users actually clicked. The accuracy of a solution is measured by a combination of precision, recall, and hit ratio.

2. THE DATASET

The training data consists of four sets: (1) *impressions* that are a list of items that were shown for the users by XING, (2) *interactions* that are positive or negative feedbacks on job posting items, and (3-4) *user/item catalog* that contains the metadata about the users/job postings. The values of the metadata are described on the website of the challenge. The test set contains a list of 150K target user ids. The goal of the challenge was to give top 30 lists of job postings that would likely be clicked by the target users in the following seven days, independently from time and context.

RecSys Challenge '16, September 15 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2987538.2987548

2.1 Preprocessing

The original data is transformed into two types of format:

1. Events: `(time, userid, itemid, type, value)`
   As the exact time of impressions is unknown, the time of the event is approximated with the Unix time of 12 p.m. Thursday of the `year-week`. Each `(time, user_id, item_id)` triple is transformed into a new record, counting its occurrence in `value`, and setting its event type to 5. The interactions set fits into this format.
2. User/item metadata: `(id, key1, key2, ..., keyN)`.
   The user- and item metadata set is provided according to the desired format, therefore no transformation is required for these sets. However, a significant number of duplications are removed from the user catalog and all unknown values are set to empty value.

Based of the analysis of item metadata, the country and region of the items are not consistent with their geo-location. Using `longitude` and `latitude` metadata, all of the job postings are assigned to the closest city with at least 1,000 population[1]. The metadata of items is enriched via cities using continent, country, region, city name, and population.

2.2 Statistics

Table 1 summarizes statistics about the dataset considering the number of events, unique users, and unique items. The results show that 95% of the events are impressions that covers approx. 4 times more users than interaction events. This concludes that three quarters of the users are inactive. The statistics highlight that half of the users who have impressions are missing from the catalog. Another important statistic is that the average number of interactions on an item is around 8, which is considered low.

Data source	#events	#users	#items
E(1): Click	7,183,038	769,396	998,424
E(2): Bookmark	206,191	59,063	142,908
E(3): Reply	422,026	107,463	190,099
E(4): Delete	1,015,423	44,595	215,844
E(5): Impression	201,872,093	2,755,167	846,814
E(1-5): All events	210,698,771	2,792,405	1,257,422
Catalog	-	1,367,057	1,358,098
Catalog and E(5)	-	1,292,662	843,589
Catalog and E(1-4)	-	784,687	1,025,582
Catalog or E(1-5)	-	2,829,563	1,362,890

Table 1: Statistics about data source.

[1]https://www.maxmind.com/en/free-world-cities-database

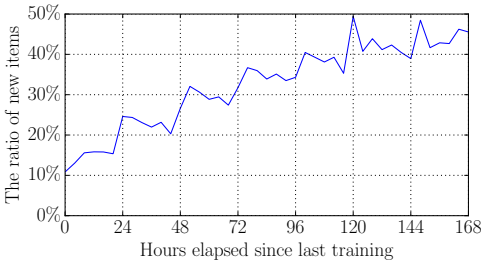
Based on the user statistics, three categories can be distinguished: (1) *new users* who have metadata only, (2) *inactive users* who have impressions only, and (3) *active users* who have interactions. The distribution of these categories among target users is 73% active, 16% inactive, and 12% new. It has been verified, that all of the target users appear in the user catalog. Analogously to users, we can define items without training events as *new items*.

Figure 1a shows the ratio of new items in interactions based on the time elapsed since the last time stamp of model training. One can see that 25% of interactions are made on new items after one day, which increases to almost 50% in one week. Overall, the expected ratio of new items in the next 7 days is 30.2%.

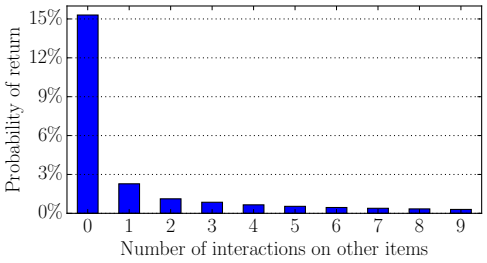
An important finding is that the users tend to return to an already seen item. Figure 1b shows the probability of clicking on an item again after a number of interactions on different items. The results show that the users click on an already seen items a subsequent time with 15% probability.

A significant rate of repeat can be measured among impressions. Figure 1c summarizes the distribution of impressions based on how many weeks have elapsed since the first recommendation of an item for a user. The findings show that approximately 38% of impressions are made on items that have previously been recommended, which allows us to conclude that the original recommender tends to recommend the same items in the following week.

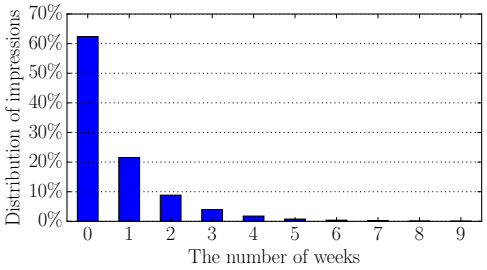
An interesting pattern was discovered about the change of the popularity of an item over time. Figure 1d shows the expected relative change in popularity in the next 7 days based on the age of an item at a given time. It can be shown that the current popularity of an item with age 25-30 days or 55-60 days is likely to drop by 50-80% in the next week, which allows us to assume that there is a 30- and 60-day promotion/availability period in the system.



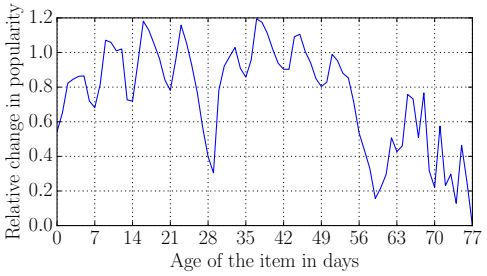
(a) The ratio of new items in interactions over time.



(b) Probability of returning to an already seen item.



(c) Distribution of impressions based on the number of weeks elapsed from the first appearance.



(d) Expected relative change in the popularity of the item in the next 7 days by item age.

Figure 1: Relevant statistics describing the dataset.

3. METHODS

The concept of this work is to define various predictors that exploit the characteristics of the data set and to find an optimal weighting for their combination. Some of the methods are instantiated using different parameters to capture different behaviors in the consumption.

3.1 Common functions

We assign a weight for every event using the following definition: $w(e) = w_y(e)(1 - \delta^{t_{max}-t(e)})$, where $w_y(e)$ is a constant weight assigned to the type of event e, δ is the base of the time decay, t_{max} is the latest time stamp of the training set, and $t(e)$ is the time stamp of event e.[2] Although time decay and event type-based weighting is not highlighted in the following definitions, the application of these functions are assumed implicitly.

Let Y be an arbitrary set of event types. For a given user u and item i, let $w_Y(u,i)$ note the weight of the latest event e, assuming that $y(e) \in Y$, where $y(e)$ is the type of event e. If there is no event for (u,i) whose type is in Y then $w_Y(u,i) = 0$. Using this notation, we define the support of item i as $s_Y(i) = \sum_u w_Y(u,i)$. Furthermore, the input-output-based co-occurrence of item i and item j is written as $s_{Y,Z}(i,j) = \sum_u w_Y(u,i)w_Z(u,j)$, where Y and Z represents the input and output sets, respectively.

[2]Practically, the time difference is calculated in days and the optimal value of δ falls between 0.05 and 0.3.

3.2 IKNN: Item-based K-nearest neighbors

Considering the significant amount of events, the application of collaborative filtering is straightforward. As the user-item matrix is very sparse in this dataset, nearest neighbor methods [3] are preferred over matrix factorization. Their application has two goals. First, based on the interactions, to find the best next items; second, to estimate the items that were originally shown the user by XING.

The similarity between item i and item j is defined as

$$sim_{I,O}(i,j) = \frac{s_{I,O}(i,j)}{(s_I(i)+\lambda)^{\alpha}(s_I(j)+\lambda)^{1-\alpha}}, \qquad (1)$$

where I and O are the sets of input and output event types, respectively; λ is the coefficient of regularization and α is the exponent of normalization that is a penalization of popularity factor. We note that $sim_{I,O}(i,i)$ is zero by design.

The prediction of item i for user u is

$$p(u,i) = \left(\sum_{e \in E(u)} w(e) \right)^{-1} \sum_{e \in E(u)} w(e) sim_{I,O}(i(e), i), \quad (2)$$

where $E(u)$ is the set of events for user u, and $i(e)$ is the item of event e. An instance of this method is denoted by IKNN(I,O).

Although matrix factorization [1] is an efficient collaborative filtering method generally, no significant improvements are achieved in addition to the combination of other models for this task. The same result is measured for user-based nearest neighbor methods. Therefore, matrix factorization and User-kNN are excluded from the final composition.

3.3 RCTR: Recalling recommendations

Based on Figure 1c, the behavior of the original recommender is partially predictable because there is a significant probability on recommending the same items as in the previous week. For this property, a click-through rate-based model is designed with implicit aging function. The prediction of item i for user u is defined as

$$p(u,i) = \sum_{e \in E(u)} \frac{s_C(i(e))}{s_R(i(e))} \mathbb{1}(y(e) = 5), \quad (3)$$

where C and R represent the clicks and impressions, respectively, and $\mathbb{1}$ is the indication function.

3.4 AS: Already seen items

Based on Figure 1b, the probability of returning to an already seen item is significant. To model this phenomenon, these items are recommended based on event types and implicit aging of the events. The prediction is

$$p(u,i) = \sum_{e \in E(u)} w(e) \mathbb{1}(y(e) \in I). \quad (4)$$

This method may perform poorly by itself, but combining just a few of these already seen items with the final composition may have significant impact on ranking-based metrics. An instance of this method is denoted by AS(I), where I refers to the type of input events.

3.5 UPOP: User metadata-based popularity

In Table 1, we have seen that a significant amount of target users are new. For those users, interactions and impressions cannot be used, therefore a popularity method is applied. For every user metadata token, the popularity of items is calculated based on users that contain that token using event aging function. The popularity of item i for user group g is calculated as

$$pop_g(i) = \frac{1}{|U(g)|} \sum_{u \in U(g)} \sum_{e \in E(u)} w(e) \mathbb{1}(i = i(e)), \quad (5)$$

where $U(g)$ is the set of the users in user group g. The prediction of item i for user u is

$$p(u,i) = \frac{1}{|G(u)|} \sum_{g \in G(u)} \frac{pop_g(i)}{pop_g(i) + \lambda}, \quad (6)$$

where $G(u)$ is the user groups that user u belongs to. For the final solution, the following keys are used to define user groups: `jobroles`, `edu_fieldofstudies`.

3.6 MS: Meta cosine similarity

For content-based filtering, a metadata-based cosine similarity is used [2]. The items are modeled in vectors based on their metadata using tf-idf weighting. The similarity $sim(i,j)$ between item i and item j is calculated by the cosine similarity of their vectors. The prediction of the preference of user u for item i is

$$p(u,i) = \left(\sum_{e \in E(u)} w(e) \right)^{-1} \sum_{e \in E(u)} w(e) sim(i(e), i). \quad (7)$$

In this setting, the following keys are used for item modeling: `tags`, `title`, `industry_id`, `geo_country`, `geo_region`, `discipline_id`.

3.7 AP: Age-based popularity change

Figure 1d shows that a significant drop in popularity can be measured among the items with age of 25-30 or 55-60 days. As the exact time stamp is not provided for the recommendation requests, an average rate of drop is calculated for these items for the next 7 days. With this method a subset of items are under-weighted for the next week that will be shown with less probability. The prediction of

$$p(u,i,t) = \frac{\sum_{k=1}^{7} (8-k) pop_a(age(i,t)+k)}{28 * pop_a(age(i,t))}, \quad (8)$$

where $pop_a(d)$ is the expected popularity of an item with age d, and $age(i,t)$ is the rounded age of item i at time t in days. For this challenge, we consider $p(u,i) = p(u,i,t_{max})$.

3.8 Optimization

For the optimization, a time-based train-test split is generated considering the last week as test set. Every predictor instance is optimized by a gradient coordinate descent optimization method. The most accurate ones are nominated as candidate predictors for the composition that is a linear combination of prediction values of sub-predictors. In order to keep the number of sub-predictors low, we propose the following greedy Forward Predictor Selection method:

1. Let $P = \{p_1, p_2, ..., p_k\}$ note a set of predictors, and assign a weight vector $W = (w_1, w_2, ..., w_k)$ to the corresponding indexes. Denote the accuracy of the combination of predictors P using vector W measured in test set T by $r(P, W, T)$.
2. Create n overlapping test folds $F = \{T_1, T_2, ..., T_n\}$, where $T_j \subset T$.
3. Let $P_C = P$, $P_S = \emptyset$ and $W = (0, 0, ..., 0)$.
4. For each $p_i \in P_C$, let $P_S^i = P_S \cup p_i$ and calculate the gain in accuracy by $g_i = r(P_S^i, W_i, T) - r(P_S, W, T)$, where $W_i = \frac{1}{|F|} \sum_{T_j \in F} \left(\arg\max_W r(P_S^i, W, T_j) \right)$ that is found by a gradient descent optimization.
5. Select i, where g_i is maximal, move p_i from P_C to P_S, and set $W = W_i$.
6. Repeat Steps 4-5 until $g_i > 0$ and $P_C \neq \emptyset$.

The method applies cross-validation that helps us to avoid over-fitting during the process. Re-optimizing all weights

in each iteration allows us to find a good combination of selected algorithms for each number of predictors involved.

The weighting can be fine-tuned if the users are grouped and different combination weights are assigned for different groups; therefore, we propose a support-based grouping. We define user support as $s(u) = \sum_{e \in E(u)} w_y(e)$, where $w_y(e) = 0.05$ if $y(e) = 5$, and 1.0, otherwise. 12 user groups are defined that are (1) new users who have no events, (2) inactive users who have impressions only, and (3-12) 10 equal sized groups of active users categorized by the deciles of their support-based distribution. For each group of users, the previously introduced process is run using a corresponding set of test cases.

3.9 OM: The omit method

It can be observed that some items are recommended too often or targeted poorly. In these cases, the expected value of an item may be negative because it may take the first positions from other items. To handle this problem, a method is proposed: (1) The original train set is split into a sub-train and sub-test set. (2) The recommender methods are retrained on the sub-train set. (3) In each case of sub-test set, a top-N recommendation is calculated. (4) For each item in the top-N set, it is measured how the overall evaluation metric would change if the item would be excluded. (5) The value is summed for all items. (6) The items with significant negative expected value are omitted for all recommendations.

The method identifies a number of elements that are positioned poorly for some reasons. By omitting these items completely, weak elements can be avoided, which rather improves than decreases offline ranking metrics. For the final solution, the last 2 days are used as sub-test set and the worst 100 items are omitted.

4. FRAMEWORK AND EVALUATION

For data processing and statistical analysis, Python and relevant packages are used. For experimenting and optimization, a Java-based framework is applied using a number of external packages. A specificity of the architecture of the framework is that the prediction values are calculated for all recommendable items. However, the majority of the algorithms recommend event-supported items only in this setup. Therefore, a reasonable consideration is that the recommendable items are limited to the items that have at least one event, which reduces set to 194,217 items from 327,003, improving the prediction time significantly. Although 30% of the items are new, blending of these into the top-N recommendations remained unsolved in this competition. The training of algorithms and the generation of recommendations were run in a server with Intel Xeon E-1231v3 3.4 GHz CPU and 32 GB memory.

Table 2 summarizes the optimization process of candidate predictors. The best single method is an Item-kNN model using interactions for both input and output, resulting approx. 450K leaderboard score with 2.5 minutes training time and 7.2 milliseconds average prediction time. Results show that a 600K score can be achieved by combining 4 algorithms. 95% of the final score is achieved by combining 6 predictors, where the score becomes more saturated. One can see that the improvement in score does not decrease monotonically. The reason for that is the difference between evaluation sets and a possible over-fitting during the optimization. Significant increase in training time can be observed by (1) introducing impressions-based Item-kNN, due to a more dense user-item matrix of impressions and (2) applying the omit method, due to the complete retraining and recommendation process on the last 2 days. For the final solution, 11 algorithm instances are selected using various weights for 12 distinct user groups. The training time of the final solution took around 3 hours providing a recommendation in approximately 200 milliseconds per user scoring 665,592 and taking the third place in the RecSys Challenge 2016. An interesting observation is that applying only the first 5 models would also have been enough for this place.

	Predictor	t_{TR}(s)	t_{PR}(s)	Score	R
1	IKNN(C,C)	148.1	7.2	450,046	24
2	+RCTR	207.6	14.8	548,339	9
3	+AS(1)	236.5	17.2	590,526	6
4	+UPOP	247.2	50.3	614,674	5
5	+MS	363.9	122.4	623,909	3
6	+IKNN(R,R)	1,149.7	168.2	635,278	3
7	+AS(3)	1,204.5	177.7	636,498	3
8	+IKNN(R,C)	1,556.8	196.8	643,145	3
9	+AS(4)	1,582.0	202.1	644,710	3
10	+AP	1,620.6	207.0	652,802	3
-	SUPP_C(1-10)	1,638.9	193.8	661,359	3
11	+OM	11,790.3	199.3	665,592	3

Notes: The table summarizes the result of forward predictor selection in row 1-10 and row 12. The first row is the best single method, the following rows represent the best additional predictors that result the best possible combinations. SUPP_C(1-10) represents a re-optimization of weights of the first 10 predictors considering support-based user categories. t_{TR} is the total training time in seconds, t_{PR} shows the average prediction time for a user in milliseconds. Score is the leaderboard score of the combination. R indicates the rank in leaderboard that could be achieved by that composition not counting the third place of the final solution. For instances of IKNN, the set of impressions and the set of interactions are denoted by R and C, respectively.

Table 2: Results of forward predictor selection.

5. CONCLUSIONS

An approach of composition have been introduced for the RecSys Challenge 2016. Due to the sparsity of user-item matrix, various item-based nearest neighbor methods are preferred over matrix factorization. A significant improvement is achieved by recommending already shown and clicked items. User metadata is incorporated by user category-based popularity, and item metadata is applied by metadata-based cosine similarity prediction. Optimizing the composition on various user categories, a more sophisticated weighting is calculated. Additional gain is achieved by under-weighting or omitting items that are likely not to be promoted in the future or positioned poorly by the recommender.

6. REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[2] M. J. Pazzani and D. Billsus. *Content-Based Recommendation Systems*, pages 325–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW 2001*, pages 285–295. ACM, 2001.