Job Recommendation Based on Factorization Machine and Topic Modelling

Vasily Leksin
Avito.ru
vleksin@avito.ru

Andrey Ostapets
Avito.ru
aostapets@avito.ru

ABSTRACT

This paper describes our solution for the RecSys Challenge 2016. In the challenge, several datasets were provided from a social network for business XING. The goal of the competition was to use these data to predict job postings that a user will interact positively with (click, bookmark or reply). Our solution to this problem includes three different types of models: Factorization Machine, item-based collaborative filtering, and content-based topic model on tags. Thus, we combined collaborative and content-based approaches in our solution. Our best submission, which was a blend of ten models, achieved 7th place in the challenge's final leaderboard with a score of 1 677 898.52. The approaches presented in this paper are general and scalable. Therefore they can be applied to another problem of this type.

CCS Concepts

•**Information systems** → **Recommender systems;** Information retrieval; •**Computing methodologies** → *Topic modeling;*

Keywords

Recommender System; Collaborative Filtering; Factorization Machine; Topic Modelling

1. INTRODUCTION

In contrast to the classical problems of machine learning, the problem of personal recommendations is usually characterized by a huge number of possible pairs (user, item) which we want to rank and select for each user the best top recommendations. Such large data sizes require us to select innovative and parallel algorithms that are going to scale to these massive data sets.

The RecSys Challenge 2016 considered job recommendations for a business social network. The data contained information about millions of users, hundreds of thousands of items (job postings) and hundreds of millions events.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

© 2016 ACM. ISBN 978-1-4503-4801-0/16/09…$15.00

DOI: http://dx.doi.org/10.1145/2987538.2987542

Collaborative filtering has proven to be useful approach to solving problems of this type. However, as will be shown later, almost half of the users, for which we need to build recommendations have not made a sufficient number of actions for using collaborative filtering for them. For those users, we are forced to use a content-based approach.

2. DATA DESCRIPTION

In the challenge, four major datasets were provided from a social network for business XING:

- **Impressions** — details about which items (job postings) were shown to which user by the existing recommender in which week of the year (events from 19 August 2015 to 9 November 2015).
- **Interactions** — interactions that the user performed on the job posting items (clicked, bookmarked, replied or deleted).
- **Users** — details about those users who appear in the above datasets: job roles, career level, discipline, industry, location, experience, and education.
- **Items** — job postings details that were and should be recommended to the users: title, career level, discipline, industry, location, employment type, tags, created time and flag if item was active during the test.

Impressions dataset contains 201 million of unique user-item-week records, 2.7 million of unique users and 846 thousand of unique items.

Interactions dataset contains 8.8 millions of events (clicked — 7.2 million, deleted — 1.0 million, replied — 422 thousand, bookmarked — 206 thousand), 785 thousand of unique users, 1.03 million of unique items. It is important that only 2.8 million from 6.9 million (40%) of user-item pairs contained in impressions (the user can perform actions not only on recommended items but also through the search, for example).

3. PROBLEM STATEMENT

Target users dataset for offline evaluation consists of 150 thousand of users. User details dataset contains all those target users, but 39.7 thousand (26.5%) of them do not have events in the interactions, 59.5 thousand (39.6%) have less than two events, 70.6 thousand (47.1%) have less than three events. It means that we need to use a hybrid approach

that takes into account not only collaborative filtering but the content data of items and users.

Item details dataset contains 327 thousand items marked as active during the test. 129 thousand (39.55%) of them do not have events in the interactions, 164 thousand (50.1%) have less than two events, 188 thousand (57.6%) have less than three events.

The task of the challenge was to predict 30 (or less) items that a user will positively interact with (click, bookmark or reply) within the next week after 9 November 2015 for each of the $N = 150\,000$ target users.

Let $U = \{0, \ldots, N-1\}$ be the list of target users' indexes. Let $R = \{r_u\}_{u \in U}$ be the ordered lists of relevant items for every target user. Relevant items are those items on which the user performed a positive interaction in the test. Let $\hat{R} = \{\hat{r}_u\}_{u \in U}$ be the solution (ordered lists of items) that generated by the algorithm for each of the target users.

The evaluation measure calculated as follows:

$$score(R, \hat{R}) = \sum_{u \in U} 20(P_2(r_u, \hat{r}_u) + P_4(r_u, \hat{r}_u) + R_{30}(r_u, \hat{r}_u) + \\ + S_{30}(r_u, \hat{r}_u)) + 10(P_6(r_u, \hat{r}_u) + P_{20}(r_u, \hat{r}_u)),$$

where $P_k(r_u, \hat{r}_u)$ – precision at top k recommended items for user with index u, $R_{30}(r_u, \hat{r}_u)$ – recall at top 30 items, $S_{30}(r_u, \hat{r}_u)$ – user success (user positive interaction with one of the top-30 is counted as a success).

Our insight was that impressions slowly change over time. As mentioned above 40% of the user's actions are made of impressions, that is, the presence of a pair of user-item in impressions is a useful feature, and we use it as the separate model.

We tried to add geographical features (the distance from the average coordinates of user's search and item, entering the geographic clusters based on coordinates), but none of these improve score. After looking closely at the user sessions, we saw that there is not much correlation between the region specified by the user in profile and the coordinates of viewed items. We include none of the features based on geographical coordinates in the final model.

4. MODELS

In this section, we describe the three main models we used in our solution: item-based collaborative filtering, Factorization Machine and content-based topic model based on tags.

4.1 Item-based collaborative filtering

An item-based model [4] ranks items according to its similarity to other items observed for this user.

This model firstly calculates the similarity between the items using the user observations that have interacted with the both items. Given the similarity $S(i, j)$ between item i and j, the model predicts a score of item j for user u using a weighted average of previous observations I_u of the user.

There are three similarity metrics to use for this model: Jaccard, cosine, and Pearson.

Jaccard similarity between two items is calculated as

$$S_J(i, j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

where U_i is the set of users who interacted the item i.

Cosine similarity between pair of items is calculated as

$$S_C(i, j) = \frac{\sum_{u \in U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in U_i} r_{ui}^2} \sqrt{\sum_{u \in U_j} r_{uj}^2}},$$

where U_i – a set of users who interacted with the item i, and U_{ij} is the set of users who interacted both items i and j.

Pearson Correlation similarity:

$$S_P(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

In our case rating r_{ij} is binary and means positive transaction.

Final prediction for item j is calculated as:

$$y_{uj} = \frac{\sum_{i \in I_u} SIM(i, j)}{|I_u|}.$$

4.2 Factorization Machine

A Factorization Machine (FM) [3] predicts the probability of any (user, item) pair interaction by learning latent factors for each user and item.

Numeric or categorical side features may be provided on the training stage. User IDs and item IDs are treated as categorical variables.

The model trains on known interactions between users and items. Recommendations are then based on these interactions.

Generative model for user i on item j is given by

$$p(i, j) = \mu + w_i + w_j + a^T x_i + b^T y_j + u_i^T v_j,$$

where μ – a global bias term, w_i and w_j are weight terms for user i and item j respectively, x_i and y_j are the user and item side feature vectors, and a and b are the weight vectors for those side features. The latent factors, which are vectors of fixed length (number of factors is a parameter), are given by u_i and v_j.

In the binary case (training on interactions), the logistic loss is used to fit a model. To train this model using Stochastic Gradient Descent (SGD), for each interaction pair (user, item) we choose several negative items that the user in the given interaction has not interacted with. Number of negative samples is a model parameter.

The model learns latent factors for all variables, including the side features, and also provides interactions between all pairs of variables. Thus complex relationships in the data can be modeled. We used GraphLab implementation for FM [2].

4.3 Content-based topic model

For content-based analysis, we decided to use topic model originally oriented at document corpus analysis but suitable for a variety of tasks.

Let document associated with each user be all title and tags tokens of items, which the user interacted with and job roles tokens from user description.

Latent Semantic Indexing (LSI) [1] considers the collection of documents as a whole, to see what other documents contain some of the same words. LSI considers documents that have many words jointly to be semantically close and

have few words in common to be semantically apart. This method correlates well with how a human, looking at content, would classify a collection of documents.

First, we convert each document into a token occurrences vector. The number of dimensions of this vector is equal to the number of unique tokens in the document corpus. Next, we transform values in each vector to TF-IDF statistics and combine all vectors into a large token-document matrix. Rows represent tokens; columns represent documents.

Then we apply Singular Value Decomposition (SVD) technique on the token-document matrix [5].

Let document related with each item be concatenation of title and tags of this item. The similarity between user and item will be the similarity between corresponding latent vectors.

5. EXPERIMENTS

In this section, we describe our experimental set-up and results. For all experiments, we used the last full week of interactions for validation. We selected 10 000 users from those who committed any positive transactions during this week. Then we removed old items (created more than a month ago) on which nobody had ever interacted. Reducing the number of users for validation and the number of items for recommendations made it possible to obtain the validation score for a short time. Obtained score through this validation was highly correlated with the result on the Public Leaderboard. It was critical for tuning model parameters.

5.1 Models training

For item-based collaborative filtering, we used three types of similarity measures between items: Jaccard, cosine, and Pearson. Cosine similarity measure had shown very low quality, and we did not include it in the final model.

We trained FM on both (user, item) pairs and side features for users and items. For side features, we encoded all used features using a one-hot scheme. For users, we used the following features: job roles, career level, discipline, industry, country, region, experience: N-entries class, years experience, years in current job, education degree, education field of studies. For items, we used title, tags, career level, discipline, industry, country, region, employment type.

Major model parameters which affected on score were the number of sampled negative examples and the number of latent factors. Increasing the number of negative examples gave a better performance at the cost of speed, in particular when the number of items is high. Increasing the number of latent factors also led to better results, but it slowed training speed.

We used different event types of transactions for both types of models item-based and FM. In the final model, we used the following combinations: all positive transactions, only click events, only impressions. The number of iterations parameter also affects the final score.

In the LSI topic model, the documents corpus was 2.7 million, tokens corpus – 80 thousand. We set the number of latent factors equal to 100 in our model.

5.2 Blending and results

At the first stage we combined three base models:

- SIM_0 – item-based Jaccard similarity model trained on positive interactions;
- FM_0 – Factorization Machine model with 400 latent factors and 20 negative examples trained on positive interactions;
- Impressed – simplest binary model determined for (user, item) pair whether this item has been at least once impressed to the user before the week for which items are predicted.

Table 1: Initial set of models

FM_0	SIM_0	Impressed	Local score
1	0	0	76 995
0	1	0	69 622
0	0	1	104 495
1	1	1	132 505

Results of basic models for local validation are presented in Table 1. It should be highlighted that linear combination of models can significantly improve the obtained results.

The final set of models includes nine models (Table 2) and the tenth "Impressed" model.

Table 2: Final set of models

SIM_jac	Item-based Jaccard similarity
SIM_click	Item-based Jaccard similarity on clicks
SIM_pearson	Item-based Pearson similarity
SIM_imp	Item-based Jaccard similarity on impressions
FM_f100_i25	FM, n_factors=100, n_iter=25
FM_f400_i70	FM, n_factors=400, n_iter=70
FM_f400_i50_no_side	FM, no side data
FM_imp	FM on impressions
TM	LSI topic model

The last version of blending, reaching 146 569 on local validation is as follows:

$$1.0 * FM_4 + 15.0 * (FM_4^{8.0} * SIM_4) + 13.0 * SIM_4 + \\ +1.0 * Impressed - 0.4 * SIM\_pearson - \\ 0.3 * FM\_f400\_i50\_no\_side + \\ 0.5 * (FM\_imp^{2.0} * SIM\_imp) + 0.2 * TM,$$

where $FM_4 = 0.5 * FM\_f100\_i25 + 0.5 * FM\_f400\_i70$ and $SIM_4 = 0.4 * SIM\_jac + 0.6 * SIM\_click$.

Our final model reached 554 655 on the Public Leaderboard. All the major steps of our solution are presented in Table 3. These steps are iterative; each new step includes the results from previous steps.

6. CONCLUSION

As shown in Table 3 considerable improvements were gained after adding the "Impressed" feature, adding FM model with side data, blending of different types of models and adding LSI topic model.

In our solution, we paid attention to collaborative filtering approach, and to content-based approach, and then combine them. This hybrid scheme, in our opinion, allowed us to get a high score in the RecSys Challenge 2016 and climb to the 7 line in Private Leaderboard. Also, two out of three of our

Table 3: Key submissions

Score	Public rank	Description	Date
554 655	9	Increasing topic model factors from 50 to 100	06/27/16
548 366	8	Increasing number of candidates from every model from 30 to 150	06/25/16
543 284	8	10 models: 4xFM + 4xSIM + TM + Impressed	06/24/16
537 157	9	Topic model added	06/23/16
530 599	10	FM + SIM + Impressed + tuned coefficients	06/23/16
497 136	15	FM + SIM + Impressed	06/22/16
496 241	1	FM with side data + Impressed	03/20/16
397 604	1	"Impressed" model added	03/11/16
132 790	1	Simple item-based recommender	03/10/16

basic models use dimensionality reduction technic, which is scalable. Moreover, all three models effectively paralleled during training and generation of recommendations. This scalability allows someone to use approaches from our solution for other similar recommendation problems.

7. REFERENCES

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[2] Y. Low, J. E. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. *CoRR*, abs/1408.2041, 2014.

[3] S. Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[4] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[5] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997.