A preliminary study on a recommender system for the Job Recommendation Challenge

Mirko Polato
University of Padova
Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy
mpolato@math.unipd.it

Fabio Aiolli
University of Padova
Department of Mathematics
Via Trieste, 63, 35121 Padova - Italy
aiolli@math.unipd.it

ABSTRACT

In this paper we present our method used in the RecSys '16 Challenge.

In particular, we propose a general collaborative filtering framework where many predictors can be cast. The framework is able to incorporate information about the content but in a collaborative fashion. Using this framework we instantiate a set of different predictors that consider different aspects of the dataset provided for the challenge. In order to merge all these aspects together, we also provide a method able to linearly combine the predictors. This method learns the weights of the predictors by solving a quadratic optimization problem.

In the experimental section we show the performance using different predictors combinations. Results highlight the fact that the combination always outperforms the single predictor.

CCS Concepts

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

Keywords

Collaborative Filtering, Top-N recommendation, Job Recommendation Challenge

1. INTRODUCTION

Collaborative filtering (CF) methods recommend to a user those items that other users similar to him/her liked in the past. Generally, the recommendation task can be divided into two categories: rating prediction and top-N recommendation. In the former category, the aim of the recommender is to predict how much a user will like an item, while in the latter the goal is to provide a ranking over the items.

Typically, the choice between these two tasks is driven by the dataset. Top-N recommendation is done over datasets that are characterized by implicit feedback, while rating prediction over datasets which have explicit ratings. The setting in which there are only implicit feedback is usually called one-class collaborative filtering [5]. Despite the dataset of the challenge contains four different kind of feedbacks, we divide them into positives and negatives in order to see the problem as a one-class CF.

There are two principal recognized approaches to collaborative filtering [1]: model-based CF, a.k.a. Matrix Factorization methods and memory-based CF which do not need the construction of a model. Even though there exist different state-of-the-art matrix factorization algorithms [4, 6] they usually struggle with huge datasets like the one we have in the RecSys '16 challenge [7].

For this reason we face the challenge using a framework which follows the guidelines of the memory-based approaches. In particular, we propose a general CF framework where many different predictors can be cast and we also provide a learning algorithm in order to linearly combine many predictors built upon our framework. An important contribution of our framework is the fact that it is able to take advantage of both the typical collaborative information and the content of users and items. In this way it is possible to define a set of representations, based on different sources of information, for both users and items, and then to combine them using different predictors.

2. THE CHALLENGE

The RecSys Challenge 2016 is co-organized by XING[1], CrowdRec[2] and MTA SZTAKI[3].

In the challenge, the task of the participants is the following: given a XING user, the recommender should predict those job postings (items) that the user will (positively) interact with in the next week.

The dataset is provided by XING and it is a sample of the whole XING data. The dataset has been partially synthesized with the addition of some artificial users whose presence contributed to the anonymization.

2.1 The dataset

The dataset consists of four main entities:

Users ($\mathcal{U}$, $|\mathcal{U}| = n$) : the users of the system;

Items ($\mathcal{I}$, $|\mathcal{I}| = m$) : the job postings that should be recommended to the user;

[1]XING: a social network for business.
[2]CrowdRec: a project funded from the European Union's Seventh Framework Programme for research
[3]Hungarian Academy of Science - Institute for Computer Science and Control

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

DOI: http://dx.doi.org/10.1145/2987538.2987549

Impressions ($\mathcal{E}$) : which items were shown by the existing XING job recommender;

Interactions ($\mathcal{R}$) : interactions that users performed on the job posting items. There are four different types of interactions: click, bookmarking, deletion and application/response. All interactions but the deletion are considered positive interactions.

Inside our framework, we consider the positive interactions as implicit positive feedback and hence they define the well known rating matrix $\mathbf{R} \in \{0,1\}^{m \times n}$.

We will call $\mathcal{I}_u$ the set of items with which the user u has been at least a positive interaction. Similarly, $\mathcal{U}_i$ is the set of users who positively interact with i.

Table 1 shows the number of records for each of the above mentioned entities. Users are characterized by a set of attributes that can be considered as a structured summary of the *curriculum vitae*.

Similarly, items are defined by a set of attributes which provide the description of the job positions, the required skills and the geographical locations.

Entity	# of records
Users	1367057
Items	1358098
Impressions	10130410
Interactions	8826678

Table 1: Number of entries for each entity.

Impressions and Interactions, instead, are connections between users and items which indicate, respectively, that a particular job has been shown to the user or had an interaction by the user.

However, Interactions have also a type, in particular, the distribution is peaked on the interaction of type 1, i.e., click, with over 7M entries, while types 2 and 3 (i.e., bookmarking and reply/application) together are less than 800K. The remaining interactions, i.e., type 4 (Deletion), are roughly 1M. Interaction types from 1 to 3 are considered positives.

For a comprehensive description of the dataset please visit [3].

2.2 Top-N recommendation

Even though there exists different kinds of interactions, we can essentially divide them into two groups: positives (i.e., from type 1 to 3) and negatives (i.e., type 4). This kind of setting is known as implicit feedback setting in which the recommendation task is to compute, for each user, a ranking of the items. In particular, in the challenge the task is to get the top-30 recommendations for each of the 150K target users. The evaluation is done using an *ad-hoc* metric which merges together precision and recall.

Formally, let $A : \mathcal{U} \rightarrow \mathit{Perm}(\mathcal{I})$ be a recommendation function, where $\mathit{Perm}(\mathcal{I})$ is a permutation of the items, and let $\mathcal{T}_u \subseteq \mathcal{I}$ be the set of relevant items for a target user u in the test set. Then, given the set of target users $\mathcal{U}_{Ts}$, the score for A is given by (for space reasons in the functions we omit the parameters $(A, \mathcal{T}_u)$):

$$s(A, \mathcal{T}_u) = \sum_{u \in \mathcal{U}_{Ts}} [20 \cdot (P_2 + P_4 + R + X) + 10 \cdot (P_6 + P_{20})],$$

where $P_k(A(u), \mathcal{T}_u)$ is the precision at k, $R(A(u), \mathcal{T}_u)$ is the recall, and X is the user success function defined as:

$$X(A(u), \mathcal{T}_u) = \mathbb{I}(|A(u)_{:30} \cap \mathcal{T}_u| > 0)$$

where $A(u)_{:30}$ takes the first 30 items of the list and $\mathbb{I} : \mathit{Bool} \rightarrow \{0,1\}$ is the indicator function (i.e., $\mathbb{I}(\mathit{True}) = 1$ else 0).

3. OUR FRAMEWORK

We consider a general framework where many different approaches can be cast. Let the matrix $\mathbf{W} \in \mathbb{R}^{n \times k}$ be the embedding of users in a factor space of dimension k with users representations (i.e., $\mathbf{w}_u$) as rows. Similarly, let the matrix $\mathbf{X} \in \mathbb{R}^{k \times m}$ be the embedding of items in the same factor space with items representations (i.e., $\mathbf{x}_i$) as columns. A recommendation is performed on the basis of the ranking induced by the following factorization:

$$\hat{\mathbf{R}} \sim \mathbf{WX}.$$

In our framework the representations pair $\mathbf{WX}$ is formed by an evidence-based and a similarity-based representation. The former is the simpler because it represents (as is) an evidence about users and/or items. The latter, instead, encodes a concept of similarity between objects and it can be seen as the "complex" representation. We will refer to them as *evidence* and *similarity* representation, respectively.

Whenever the similarity representation is performed between users we will have a user-based method. Conversely, when the similarity-based representation is calculated between items we will have an item-based method.

3.1 Predictors

In this section we present all the combined predictors and how they can be cast into our general framework.

As said in the previous section, we can divide predictors, which belong to our framework, into two categories: item-based and user-based. In both settings, the normalized score of the predictor k for a user-item pair (u, i) is calculated by:

$$\hat{r}_{ui}^{(k)} = \frac{\mathbf{w}_u \mathbf{x}_i}{\|\mathbf{w}_u\|_2 \|\mathbf{x}_i\|_2}.$$

Inside our predictors we often used as similarity function the well known cosine similarity. Given two vectors $\mathbf{a}, \mathbf{b} \in \{0,1\}^q$ the cosine similarity is calculated by:

$$cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}.$$

In the reminder we are going to denote with $\boldsymbol{\phi}_x$ the vectorial representation of the object x onto some feature space.

3.1.1 Item-based predictors

In this section we present the item-based predictors where we have an evidence-based representation for the users and a similarity-based representation for the items.

Interaction-based (IB) This is a personalized predictor which suggests to a user u items that are similar to the ones he/she interacted with in the past. In our framework:

$$\mathbf{W} = \mathbf{R}, \quad \boldsymbol{\phi}_i = \mathbf{r}_i, \quad \mathbf{X} : x_{ij} = cos(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j).$$

Evidence-Impression-based (EIB) This predictor differs from the previous one only on the users evidence which is represented by the impressions. Given $\mathbf{E} \in \{0,1\}^{n \times m}$ the matrix in which there is a 1 in the entry (u,i) if the item i has been impressed to the user u, we can cast this method in our framework as:

$$\mathbf{W} = \mathbf{E}, \quad \boldsymbol{\phi}_i = \mathbf{r}_i, \quad \mathbf{X} : x_{ij} = cos(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j).$$

Impression-based (IEB) It is possible to create a new predictor by switching the representation of the EIB predictor. In particular, we use as evidence for the users the interactions and as the items representation the impression, obtaining the following instantiation of our framework:

$$\mathbf{W} = \mathbf{R}, \quad \boldsymbol{\phi}_i = \mathbf{e}_i, \quad \mathbf{X} : x_{ij} = cos(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j).$$

Tag-based (TGB) Here the item representation is characterized by the tags that describe the job posting.

Let $\mathcal{G}$ be the set of all possible tag and let $\mathbf{g}_i \in \{0,1\}^{|\mathcal{G}|}$ be the vector representation of the item i in which are set to 1 all the entries which correspond to a tag that describe the item. Then:

$$\mathbf{W} = \mathbf{R}, \quad \boldsymbol{\phi}_i = \mathbf{g}_i, \quad \mathbf{X} : x_{ij} = cos(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j).$$

Title-based (TTB) Similarly to the tag-based, in this predictor the item representation is characterized by the terms that describe the job title.

Let $\mathcal{L}$ be the set of all possible terms in the title and let $\mathbf{l}_i \in \{0,1\}^{|\mathcal{L}|}$ be the vector representation of the item i in which are set to 1 all the entries which correspond to a term in the title. Then:

$$\mathbf{W} = \mathbf{R}, \quad \boldsymbol{\phi}_i = \mathbf{l}_i, \quad \mathbf{X} : x_{ij} = cos(\boldsymbol{\phi}_i, \boldsymbol{\phi}_j).$$

3.1.2 User-based predictors

In this section we present the user-based predictors where we have an evidence-based representation for the items and a similarity-based representation for the users.

Popularity-based (PB) This predictor is not really personalized, it simply proposes to every user the ranking induced by the popularity of the items, that is, items are sorted by number of ratings (i.e., positive interactions). Inside the framework the popularity-based predictor belong to the user-based family, where the user and item representations are the following:

$$\mathbf{W} = \mathbf{1}, \qquad \mathbf{X} = \mathbf{R}.$$

Interaction-based (UB) This predictor suggests to a user u items that have been rated by similar users in the past. In our framework:

$$\boldsymbol{\phi}_u = \mathbf{r}_u, \quad \mathbf{W} : w_{uv} = cos(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v), \quad \mathbf{X} = \mathbf{R}.$$

Evidence-Impression-based (EUB) As for the item-based case, the only difference from the previous predictor is the evidence for the items:

$$\boldsymbol{\phi}_u = \mathbf{r}_u, \quad \mathbf{W} : w_{uv} = cos(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v), \quad \mathbf{X} = \mathbf{E}.$$

Impression-based (UEB) Similarly to IEB, swapping the item evidence and the user representations we obtain a new predictor:

$$\boldsymbol{\phi}_u = \mathbf{e}_u, \quad \mathbf{W} : w_{uv} = cos(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v), \quad \mathbf{X} = \mathbf{R}.$$

FOS-based (FB) In this predictor the user representation is identified by his/her field of studies.

Let $\mathcal{F}$ be the set of all possible field of studies and let $\mathbf{f}_u \in \{0,1\}^{|\mathcal{F}|}$ be the vector representation of the user u in which are set to 1 all the entries which correspond to a field studied by u. Then:

$$\boldsymbol{\phi}_u = \mathbf{f}_u, \quad \mathbf{W} : w_{uv} = cos(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v), \quad \mathbf{X} = \mathbf{R}.$$

Job role-based (JB) In this predictor the user representation is identified by the his/her job role terms.

Let $\mathcal{J}$ be the set of all possible job role terms and let $\mathbf{j}_u \in \{0,1\}^{|\mathcal{J}|}$ be the vector representation of the user u in which are set to 1 all the entries which correspond to a job role of u. Then:

$$\boldsymbol{\phi}_u = \mathbf{j}_u, \quad \mathbf{W} : w_{uv} = cos(\boldsymbol{\phi}_u, \boldsymbol{\phi}_v), \quad \mathbf{X} = \mathbf{R}.$$

3.2 Learning

In order to get the final score we combined the scores obtained from all the predictors using a linear combination where the weights are learned using a method inspired by [4, 2].

Given a set of predictors $\{\Psi_1, \Psi_2, \dots \Psi_K\}$ such that $\forall i, \Psi_i : \mathcal{U} \times \mathcal{I} \to \mathcal{S}$, where $\mathcal{S} \equiv [0,1]$ is the set of possible normalized scores, we define the final score for a user-item pair as:

$$\hat{r}_{ui} = \sum_{k=1}^{K} \eta_k \cdot \Psi_k(u,i) = \sum_{k=1}^{K} \eta_k \cdot \hat{r}_{ui}^{(k)},$$

where $\eta_k \geq 0$ and $\sum_{k=1}^{K} \eta_k = 1$.

Given a user u and an item i we can define the scoring vector for the pair (u,i) as

$$\mathbf{s}_{ui} = \left[\hat{r}_{ui}^{(1)}, \hat{r}_{ui}^{(2)}, \dots \hat{r}_{ui}^{(K)}\right]^\top \in \mathbb{R}^{K \times 1}.$$

In order to learn $\boldsymbol{\eta}$, we define the following regression problem, in which we want to minimize the squared difference between the score predicted by the combination of the predictors and the real score (i.e., r_{ui}):

$$\min_{\boldsymbol{\eta}} \sum_{u,i} c_{ui} (\boldsymbol{\eta}^\top \mathbf{s}_{ui} - r_{ui})^2. \tag{1}$$

which can be easily reduced to:

$$\min_{\boldsymbol{\eta}} \frac{1}{2} \boldsymbol{\eta}^\top \left(\sum_{u,i} c_{ui} (\mathbf{s}_{ui} \mathbf{s}_{ui}^\top) \right) \boldsymbol{\eta} - \boldsymbol{\eta}^\top \left(\sum_{\substack{u,i \\ r_{ui}=1}} c_{ui} \mathbf{s}_{ui} \right) \tag{2}$$

subject to

$$\|\boldsymbol{\eta}\|_1 = 1, \ \eta_k \geq 0.$$

where c_{ui} are non-negative constants which represent the confidence for the pairs (u,i). In general, $c_{ui} > c_{uj}$ for each $i \in \mathcal{I}_u$ and $j \notin \mathcal{I}_u$, while if i and j are both positives or negatives they share the same weight.

The idea is to give more importance to interactions that are known to be positives, while negative ones, that are intrinsically ambiguous, should have lower confidence.

The optimization problem (2) is known as a *quadratic programming* one, which can be solved efficiently by software libraries like, for example, CVXOPT[4].

[4] http://cvxopt.org

UB	IB	PB	EIB	EUB	IEB	UEB	TGB	TTB	FB	JB
11533.32	10831.53	2429.54	10663.91	9113.55	9457.75	7795.68	7369.27	6161.02	2424.23	6848.59

Table 2: Score obtained by the single predictors over 5000 randomly selected target users.

UB	IB	PB	EIB	EUB	IEB	UEB	TGB	TTB	FB	JB	Score
0.5	0.5										10962.31*
0.5	0.4	0.1									11853.15*
0.515	0.474	0.011									11806.53
0.39	0.37	0.008	0.042	0.19							13531.86
			0.03	0.12	0.09	0.23	0.13	0.14	0.14	0.12	13583.49
0.09	0.07	0.033	0.023	0.101	0.062	0.191	0.10	0.11	0.12	0.1	**14158.14**

Table 3: Score obtained by the combination of the predictors over 5000 randomly selected target users.

Since the number of pairs (u, i) with the challenge's dataset is huge, in order to solve the problem (2) efficiently we did a sampling in which we took randomly 3000 positive and 3000 negative pairs. The learning phase has been performed splitting the dataset into a training and a test set. The splitting has been done chronologically, roughly the first 70% as training set and the last 30% as test set.

3.3 Pre-processing filtering

In order to reduce the computational complexity of the optimization problems (2), for each user we have pre-selected a set of 1000 most promising items with respect to the UB predictor which is the one with better performance in the single predictor setting. Moreover, we also filtered out from these sets all the items that have been a negative feedback by the user. So, the sum in the first part of (2) spans over at most 1000 items.

4. EXPERIMENTS

To speed up the experimental process we randomly took from the set of target users, that are 150K, a subset of 5000 users. The best performing parameters were then submitted to the system in order to get the score over all the target users. Scores are calculated using the metric presented in Section 2.2.

Table 2 shows the results achieved by the single predictors. As said in Section 3.3 the predictor with the best performance is the UB, followed by IB and EIB.

In Table 3 instead it is possible to see the scores of some combinations of the predictors. Scores marked with the (*) are combination made by hand, while all the others are learned by our algorithm. We fixed the weighting parameter $c_{ui} = 40$ if $i \in \mathcal{I}_u$ and $c_{ui} = 1$ otherwise, like the best configuration in [4].

Results show how combined predictors achieve better results than the single. In particular all the combinations obtained through the learning algorithm outperform the single UB which is the best in the single setting. It is worth to notice that predictors like UB and IB which have good scores in the single setting, have quite small weights in the combination. This means that part of the information they have is redundant. Another interesting fact is that the contribution of the impression-based predictors, especially user-based ones, got the highest weights. The best performing combination achieved 431249.98 points on the entire test set with a 27th position in the leaderboard.

5. CONCLUSIONS

We proposed a general collaborative filtering framework able to incorporate, despite the usual information about user-item interactions, content information. We have also provided different instantiations of our framework and an algorithm able to learn a good linear combination of those predictors. Experiments, performed over the dataset of the challenge, show how the combination of different predictors always outperfoms the single.

We think that exploiting job-specific peculiarities in order to build better user-item representations could improve the performance of our method.

6. REFERENCES

[1] F. Aiolli. Efficient top-N recommendation for very large scale binary rated datasets. In *ACM Recommender Systems Conference*, pages 273–280, Hong Kong, China, 2013.

[2] F. Aiolli. Convex AUC optimization for top-N recommendation with implicit feedback. In *ACM Recommender Systems Conference*, pages 293–296, New York, USA, 2014.

[3] R. P. F. Abel, D. Kohlsdorf. Acm recsys challenge 2016: Training data. https://recsys.xing.com/data, 2016.

[4] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[5] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[7] S. Sedhain, A. Menon, S. Sanner, and D. Braziunas. On the effectiveness of linear models for one-class collaborative filtering, 2016.