Multi-Stack Ensemble for Job Recommendation

Tommaso Carpi[1], Marco Edemanti[1], Ervin Kamberoski[1], Elena Sacchi[1],
Paolo Cremonesi[2], Roberto Pagano[2], and Massimo Quadrana[2]

DEIB, Politecnico Di Milano, Milan, Italy – [1]<name.surname>@mail.polimi.it – [2]<name.surname>@polimi.it

ABSTRACT

This paper describes the approach that team PumpkinPie adopted in the 2016 Recsys Challenge. The task of the competition, organized by XING, is to predict which job postings the user has interacted with. The team's approach mainly consists in generating a set of models using different techniques, and then combining them in a multi-stack ensemble. This strategy granted the fourth position in the final leaderboard to the team, with an overall score of 1.86M.

1. INTRODUCTION

Job recommendation domain presents some unique challenges for recommender systems. First of all, it makes sense to recommend already seen items, since users can browse them many times. Second, jobs have a trendiness and freshness window: it is very likely that old job postings are not available anymore [1]. Third, it is key to this domain to match the appropriate job to the appropriate user as this is not only a matter of personal taste. Each job posting requires a particular set of skills that have to be matched by the user personal skills [3].

Our approach consists of an ensemble of many different recommendation algorithms. Ensemble theory states that the more the algorithms are diverse and can learn different relations in the data, the better is the performance of the ensemble [4, 5]. Figure 1 presents an outlook of the layers of our ensemble solution. Algorithms are represented by solid-line rectangles, while ensembles by dashed-line rectangles. The arrows shows the ensemble flow. The rest of the paper is organized as follows: Section 2 presents the dataset and the competition organization, including the evaluation methodology. The algorithms will be presented in Section 3, while the ensembling methods in Section 4. Finally, Section 5 present the results and draws some insight for the job recommendation domain.

2. PROBLEM DEFINITION

The dataset provided is a subset of the interactions of the

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.

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

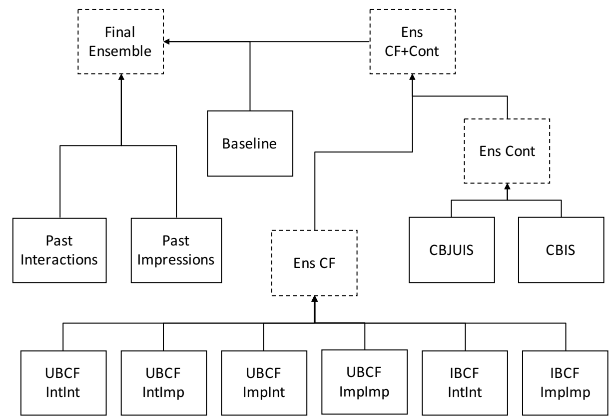

Figure 1: Ensemble Stack

Xing users in a time span of 12 weeks. The (hidden) test set is relative to the positive interactions (click, bookmark, reply) of the test users during the week immediately following the training set. The data provided for the competition consisted in:
$\mathcal{U}$: 1.5M records of users and their features.
$\mathcal{I}$: 1.3M records on items and their features.
$\mathcal{A}$: 10.1M records on users and the job offerings that the XING recommender system showed the user during the training set period, grouped by week. Xing does not ensure that these were actually seen by the user.
$\mathcal{B}$: 8.8M records on users and their interactions with job offerings over the training period. For each user-job pair also the type of interaction (click, bookmark, reply to offer, hide from view) and their timestamps were present.
$\mathcal{C} = \mathcal{C}_U \cup \mathcal{C}_I$: The set of the concepts ($\tilde{1}$00k) relative either to the users ($\mathcal{C}_U$) or to the items ($\mathcal{C}_I$).

Some of the records in set $\mathcal{U}$ and $\mathcal{I}$ have missing features. The test set consists of 150k users that were all present in $\mathcal{U}$, but not all of them were also present in $\mathcal{A}$ or in $\mathcal{B}$ or in both: 107.4K of them were in both $\mathcal{A}$ and $\mathcal{B}$, 25.2K of them were in $\mathcal{A}$ and not in $\mathcal{B}$, 2.7K of them were in $\mathcal{B}$ and not in $\mathcal{A}$, and 14.7k users were the cold start users, i.e. they did not have any interaction or impression. Among the 1.3M job offerings, only about 300k were recommendable (active) during the test period.
Both $\mathcal{U}$ and $\mathcal{I}$ share some features, such as the the industry id, the discipline id, the region id, and the career level. The users have also the feature relative to the concepts of their

career, while the items have the same concepts extracted from the title and the description of the job posting.

The task of the competition is to provide, for each user u, a ordered list of at most 30 recommended items ($reclist_u$). The evaluation metric for the challenge combines different metrics like *precision at k* ($p@k$), *recall* and *user success*.

$$p_u@k = \frac{reclist_u@k \cap groundTruth_u}{k} \tag{1}$$

$$\text{recall}_u = \frac{reclist_u \cap groundTruth_u}{|groundTruth_u|} \tag{2}$$

$$\text{userSuccess}_u = \begin{cases} 1 & \text{if } reclist_u \cap groundTruth_u \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $reclist@k_u$ is the recommendation list for user u of length k (with no subscript it is the entire recommendation list), $groundTruth_u$ are the items in the test set for user u, $p_u@k$ is the precision at k for user u, recall_u is the recall for user u, and userSuccess_u measure if there is at least an item in both $reclist_u$ and $groundTruth_u$.
The score for user u is then calculated as follows:

$$\begin{aligned} s_u &= 20(p_u@2 + p_u@4 + \text{recall}_u + \text{userSuccess}_u) \\ &\quad + 10(p_u@6 + p_u@20) \end{aligned} \tag{4}$$

and the final score is the sum of the scores for all test users.

3. ALGORITHMS

We used CF algorithms, Content-based algorithms, past interactions, past impressions and the baseline.

3.1 Collaborative Filtering

We created a total of 6 collaborative filtering models: 4 User Based (UBCF) Collaborative Filtering , and 2 Item Based Collaborative Filtering (IBCF) . All of the methods consider every positive interaction as an implicit rating (*signal*) of the user's interest in a particular job. However we converted this implicit signal to an explicit rating as detailed below:

$$r_{ui} = \begin{cases} 0 \text{ if user } u \text{ did not interact with item } i \\ \log\left(\frac{\text{total \# of interactions}}{\text{\# of interactions with job } i}\right) \quad \text{otherwise} \end{cases} \tag{5}$$

3.1.1 User Based Collaborative Filtering

Once the rating function is established, we can calculate the user-based similarity. In order to reduce the computational effort and to remove some noise from the datasets, we compute the similarity only between users that have at least a number of rated items above a fixed threshold z. We define the set $\mathcal{U}_z$ as the set of all users with at least z ratings. Given two users u and w in $\mathcal{U}_z$ we compute the user similarity as:

$$sim\_user(u, w) = \frac{\sum_i r_{ui} r_{wi}}{\sqrt{\sum_i r_{ui}^2}\sqrt{\sum_i r_{wi}^2} + \beta} \tag{6}$$

where the summations over i are calculated over the set of all Jobs that belong to the samples (impressions or interaction) on which we were computing the similarity. The shrink term β is used in order to give more importance to users that have an high number of co-rated items.

Table 1: User-Based and Item-based Collaborative Filtering model parameters

Name	Similarity	Target	z	β	K
UBCF IntInt	Interaction	Interaction	6	7	500
UBCF IntImp	Interaction	Impression	6	7	500
UBCF ImpInt	Impression	Interaction	15	9	500
UBCF ImpImp	Impression	Impression	15	9	500
IBCF IntInt	Interaction	Interaction	15	7	500
IBCF ImpImp	Impression	Impression	10	7	500

Having both impressions and interactions data we are able to create two different similarity measures between users, one interaction-based and one impression-based. The final recommendation for user u and item i is then calculated as the weighted average of the ratings given to item i of the top-K similar users to user u.

With two similarity measures and two different options for creating the user profile, we are able to obtain four sufficiently different predictions for each user that could be mixed together to obtain a better recommendation. Table 1 reports the best parameters for each model.

3.1.2 Item Based Collaborative Filtering

As for the user-based approach, we define the set $\mathcal{I}_z$ the set of items having at least z ratings. The similarity between two item i and j in $\mathcal{I}_z$ is calculated as:

$$sim\_item(i, j) = \frac{\sum_{u \in \mathcal{U}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in \mathcal{U}} r_{ui}^2}\sqrt{\sum_{u \in \mathcal{U}} r_{uj}^2} + \beta} \tag{7}$$

where β is the shrink factor. The predicted rating for an user u over an item i can then be estimated as the average of the ratings given by user u over the top-K similar items to i weighted by their similarity.

Differently from the user-based approach, the cross combinations IntImp and ImpInt did not yielded good results, so we decided to keep only the IntInt and the ImpImp variants. Table 1 reports the best parameters of the used models.

3.2 Content Based algorithms

We have implemented two different content based algorithms, exploiting the similarity the concepts of the items and the similarity of the concepts of users and items.

3.2.1 Concept-Based Item similarity

An item is described by a vector in a vector space model composed by its concepts present in the title and the description of the job posting. Each element of the vector represent the importance of the feature that we calculated as the IDF [2] thus:

$$w_{ci} = \begin{cases} 0 \text{ if item } i \text{ does not have concept } c \\ \log\left(\frac{|\mathcal{C}_I|}{\text{\# of items having concept } c}\right) \quad \text{otherwise} \end{cases} \tag{8}$$

The similarity between two items i and j is then simply defined as the cosine similarity between the two items vector:

$$sim\_item(i, j) = \frac{\sum_{c \in \mathcal{C}} w_{ci} w_{cj}}{\sqrt{\sum_{c \in \mathcal{C}} w_{ci}^2}\sqrt{\sum_{c \in \mathcal{C}} w_{cj}^2} + \beta} \tag{9}$$

where β is the shrink factor. Differently from the CF, for the prediction we used the non weighted average of the similarities of the top 30 similar items to the target item.

3.2.2 Concept-Based Joint User-Item similarity

We use the concept-based vector space model to represent both users and items in terms of their concepts. A user (document), will be represented by a vector UF of concepts that appeared in the jobs they interacted with in the past. Each element of the vector corresponds to the weight (or importance) of that specific concept for the current user. To calculate these vectors we used TF-IDF normalized weights, calculated as :

$$\mathrm{UF}(c,u) = \mathrm{TF}_U(c,u) \cdot \mathrm{IDF}_U(c) \tag{10}$$

$$\mathrm{TF}_U(c,u) = \frac{\sum_i b_{uic}}{\sum_i b_{ui}} \tag{11}$$

$$\mathrm{IDF}_U(c) = \log\left(\frac{\#\text{ users} \in \mathcal{B}}{\#\text{ users} \in \mathcal{B} \text{ having concept } c}\right) \tag{12}$$

where b_{uic} is 1 if the user u interacted with item i having concept c and 0 otherwise and b_{ui} is 1 if the user u interacted with item i and 0 otherwise.

A similar formula was used for the item-feature vector representation. In this case the item represents the document, and the feature represents the term. The difference is that the term-frequency part is binary.

$$\mathrm{IF}(c,i) = \mathrm{TF}_I(c,i) \cdot \mathrm{IDF}_I(c) \tag{13}$$

$$\mathrm{TF}_I(c,i) = \begin{cases} 1 & \text{if item } i \text{ has concept } c \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$IDF_I(c) = \log\left(\frac{|\mathcal{I}|}{\#\text{ of items having concept } c}\right) \tag{15}$$

Once we have built the vectors for both users and items, we can proceed with calculating the similarity between each pair of user-item. In order to calculate the similarity between user u and item i we used the cosine between the angles formed by their vectors UF_u and IF_i. Once the similarities for each user-item pair are computed, we sort these similarities for each user u in decreasing order and recommend the first 30 items.

3.3 Past Interactions

Differently from other domains, for job recommendation it makes sense to recommend already seen items, since users are likely to re-interact with past items for example to review the job description.

In order to take into account this information, we processed the past interaction with a filtering and an ordering step. We filtered out all the items whose latest interaction did not happened during the last 2 weeks of activity of each users. Moreover we also did not take into account the "remove" interactions, since they do not constitute neither a positive nor a negative feedback. We then sorted all the items per descending interaction count, following the principle that the more times an item had been interacted with the more interesting it is for the user, since he had continuously got back to that job posting.

3.4 Past Impressions

Impressions, i.e. the output of the Xing recommender, cannot be ignored, since they heavily bias the available choices of each user. This simple algorithm takes the last week of each user for which the data is available. We noticed that simply reordering these items in the list of recommendation led to very different results with a difference up to 50k points. We tried many reordering methods, but the best performing one was to sort the items for each user using the Concept based joint user-item similarity (See Section 3.2.2).

3.5 Baseline

This algorithm was provided by XING and slightly tweaked by us. For each item we compute a score, giving points for each item feature matching a user feature (like region, discipline, industry). This algorithm lead to poor results, it was used mainly to fill those recommendations that our algorithms were not able to provide.

4. ENSEMBLING METHODS

We chose to create an ensemble of the previous algorithms using a multi-stack ensemble technique [6], where we created a hierarchy of hybrid models in order to exploit the best out of each single learner we had. Each ensemble in the stack uses a voting-based method combined with a reduce function in order to build the hybrid recommendation, as detailed below:

$$s_{aui} = f_E(\mathrm{rank}_{aui}, \Theta_E) \tag{16}$$

$$s_{Eui} = \sum_{a \in E} s_{aui} \tag{17}$$

$$\mathrm{rank}_{Eui} = \mathrm{sort}(s_{Eui}) \tag{18}$$

where $a \in E$ is an algorithm in the ensemble E, s_{aui} is the score assigned to item i of user u for algorithm a by the scoring function f_E dependent on the parameters of the ensemble Θ_E. s_{Eui} is the final score for each item i of the user u for the ensemble E. rank_{Eui} is the final rank of each item i for the user u calculated by the sort function that sorts the items in descending s_{Eui} order and takes the top 30 items.

Our idea is that if an item is recommended by more than one different technique it has higher probability to be a good recommendation.

Hereafter, we will describe the different voting techniques that we implemented in order to assign a score to each element of the algorithms inside the stack. Table 2 reports the name of the algorithm produced by the ensemble together with the used parameters.

4.1 Linear Ensemble

In the linear ensemble we used two per-algorithm parameters: the weight w_a of the algorithm a and the decay d_a of algorithm a. The score s_{aui} is calculated as:

$$s_{aui} = w_a - \mathrm{rank}_{aui} \cdot d_a \tag{19}$$

We used integers in order to establish a priority over the algorithms, being d_a a very low value in the order of magnitude of 10^{-3}. We then used d_a as an *interleaving factor* that allows to interleave the recommendations of algorithms having the same priority (same weight w_a). We set d_a proportionally to the public leaderboard score of algorithm a. This ensemble method is used for Ens Cont, Ens CF+Cont and for the Final Ensemble.

Table 2: Parameters and public leaderboard scores l_a for each algorithm and ensemble

Ens. Name	Algorithm	Ens. Type	w_a	d_a	l_a
CF	UBCF IntInt	Eval	.0339	NA	108k
	UBCF IntImp	Eval	.0322	NA	103k
	UBCF ImpInt	Eval	.0238	NA	91k
	UBCF ImpImp	Eval	.0408	NA	156k
	IBCF ImpImp	Eval	.0314	NA	121k
	IBCF IntInt	Eval	.0300	NA	95k
Cont	CBJUIS	Lin	1	.001	128k
	CBIS	Lin	1	.0015	102k
CF+ Cont	Ens CF	Lin	2	.001	180k
	Ens Cont	Lin	2	.0015	140k
Final	Ens CF+Cont	Lin	3.98	.001	250k
	P. Interactions	Lin	4	.001	230k
	P. Impressions	Lin	4	.001	400k
	Baseline	Lin	.1	.0001	46k
	Final				622k

4.2 Evaluation Score Ensemble

The Evaluation Score ensemble assigns to each item in the recommendation list a score that reflects the maximum score that can be obtained using the provided evaluation metric by recommending a relevant item in that particular position.

$$s_{aui} = w_a \cdot e(\text{rank}_{aui}) \tag{20}$$

where $e(\text{rank}_{aui})$ is defined as:

$$e(\text{rank}_{aui}) = \begin{cases} 37.83, & \text{rank}_{aui} \in [1,2] \\ 27.83, & \text{rank}_{aui} \in [3,4] \\ 22.83, & \text{rank}_{aui} \in [5,6] \\ 21.17, & \text{rank}_{aui} \in [7,20] \\ 20.67, & \text{rank}_{aui} \in [21,N] \end{cases} \tag{21}$$

For determining the weights w_a we defined the *score density ratio* as the ratio between the leaderboard score and the total number of recommended items:

$$w_a = \frac{l_a}{n_a} \tag{22}$$

where n_a is the number of items recommended by algorithm a and l_a is the public leaderboard score for algorithm a. We use this ensemble method for the Ens CF.

5. RESULTS AND CONCLUSIONS

Table 2 shows the public leaderboard score for every single algorithm and ensemble of our method. The ordering of past impressions and interactions already provides good results, therefore from every other algorithm we excluded the items that were present either in the past interaction or in the past impressions for each user. In this way the recommendations would have been different in each of our methods, thus giving more diverse information to the ensembling algorithm. This is the reason for the relatively low performance of collaborative and content algorithms: they do not contain the items that the user is more likely to interact with. Without any filtering their scores are much higher.

We can observe that any ensembling method is effectively exploiting the differences in the algorithms, always delivering better performance. Taking Ens CF as an example, it scores 180k compared to the best performance of the best CF algorithm, i.e. 156k. The Eval ensembling method was used only in the Ens CF: this can be explained by the fact that CF algorithms' confidence in recommendations degrades pretty fast and a linear ensemble is not able to weight them in the proper way. The linear ensembling, instead, has been proven the best for the other ensembles: by controlling the weight and the decay it basically produces an interleaving with priority: an higher weighted algorithm is recommended before a lower priority one, while maintaining its internal ordering. Two equally weighted algorithms, instead, are interleaved proportionally to their decay ratio: this is the case of the Final Ensemble, where interactions and impressions have the same decay and weight, so the final ensemble will interleave an item from the past impressions and an item from the past interactions. Of course, if an item is present in more than one algorithm contributing to the ensemble, this is pushed higher in the recommendation list.

Xing claims that its algorithm can score much better on the same data of the competition. However, the evaluation introduces a bias that underestimates the proposed algorithms. The introduced bias is relative to the impressions: of course it is very likely that a user clicks on items that are recommended by the Xing recommender, as this is the main way for users to interact with the system. This fact imposes a bias on the ground truth, because it is very likely to be a subset of Xing recommendations: we believe that evaluating the algorithms proposed by different teams in an online setting would improve the reported performances, since the users are presented with the actual recommendations. The way in which the evaluation has been carried out, instead, promotes the algorithms that learn which are the best items among the ones recommended by the Xing platform. Thus any comparison with the Xing recommender is penalized.

6. REFERENCES

[1] D. Agarwal, B.-C. Chen, R. Gupta, J. Hartman, Q. He, A. Iyer, S. Kolar, Y. Ma, P. Shivaswamy, A. Singh, et al. Activity ranking in linkedin feed. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1603–1612. ACM, 2014.

[2] A. Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[3] M. Bastian, M. Hayes, W. Vaughan, S. Shah, P. Skomoroch, H. Kim, S. Uryasev, and C. Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8. ACM, 2014.

[4] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[5] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[6] J. Sill, G. Takács, L. Mackey, and D. Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.