Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems

Santi Seguí | 2016-2017

Some tricks (II)

Recursive methods: In order to avoid cold-start we can apply a recursive method for new users or for sparse data sets.

Some tricks (IV)

Clustering applied as a “first step” for **shrinking** the candidate set of users.

Long Tail Problem

- Usually, **popular items** provide **less profit** that non-popular.

- **Difficult** to provide good rating prediction for those item **in the long tail**.

- Popular items does **not** provide the **same information** about tastes than non-popular items

Impact of the long Tail

Some movies are really popular since other very unpopular. Popular ratings can sometimes worsen the quality of the recommendations since they tend to be less discriminative across different users

Each item *j* can be weighted by w_j as follows

$$w_j = log\left(\frac{m}{m_j}\right)$$

where *m* is the total number of users, and m_j is the number of users who have rated the item j

$$Pearson(u,v) = \frac{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \bar{r}_u)(r_{vk} - \bar{r}_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \bar{r}_u)^2} \sqrt{\sum_{k \in I_u \cap I_v} w_k (r_{vk} - \bar{r}_v)^2}}$$

Clustering

- Computing the similarity between users is computationally expensive.
- Clustering is cheaper to than computing the mxm similarity matrix
- Redefine top similar users using only the subset of users in the same cluster
- Problem: *MxN* is an incomplete, really sparse, matrix.

Dimensionality Reduction and Neighborhood Methods

- Dimensionality reduction can **improve** neighborhood methods in terms of **accuracy** and also in terms of **efficiency**.

- Similarities are hard to be computed in huge dimensional sparse rating matrices.

  - Latent factor models

Sparse Linear Models

- Computes the item-item relations, by estimating an item x item sparse aggregation coefficient matrix **S**.

- The recommendation score of an unrated item **i** for a user **u** is:

$$\hat{r}_{ui} = \mathbf{r}_u^T \mathbf{s}_i.$$

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \tfrac{1}{2}\sum_{u,i}(r_{ui} - \hat{r}_{ui})^2 + \tfrac{\beta}{2}||S||_F^2 + \lambda||S||_1, \\ \text{subject to} \quad & S \geq 0, \text{and} \\ & \text{diag}(S) = 0. \end{aligned}$$

SLIM: Sparse linear methods for top-n recommender systems
X Ning, G Karypis
Data Mining (ICDM), 2011 IEEE 11th International Conference on, 497-506

115 2011

Explaining recommendations

- Help on:
  - Transparency
  - Trust

A survey of explanations in recommender systems N Tintarev, J Masthoff Data Engineering Workshop, 2007 IEEE 23rd International Conference on, 801-810	226	2007

What is an Explanation

- Additional data to help users understand a specific recommendation
    - It is totally separate from an explanation if how the system works as a whole
    - Some explanation are confidence values
- Show distributions
- Sometimes tied to ability to edit profile to improve recommendations..

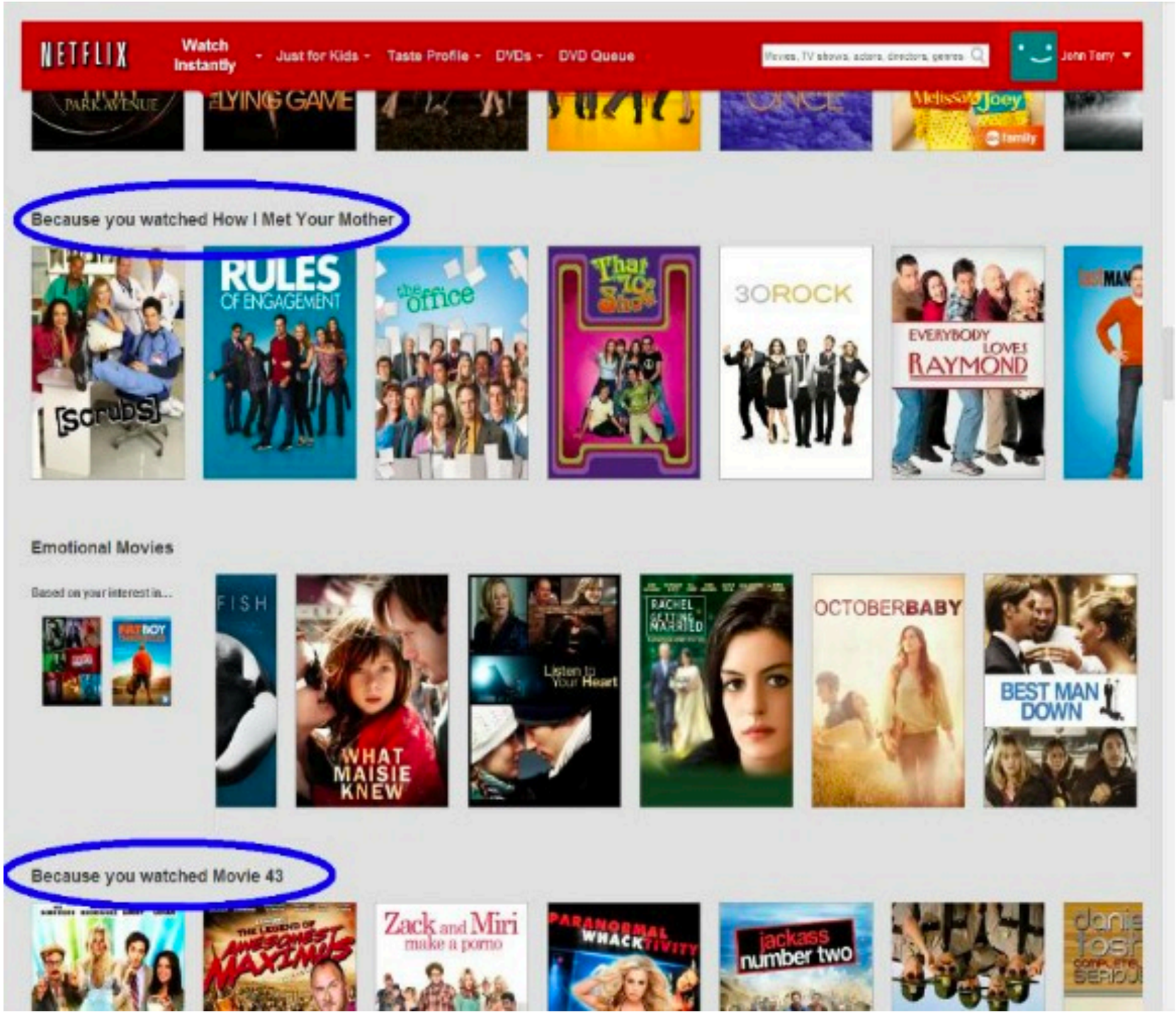
#		N	Mean Response	Std Dev
1	Histogram with grouping	76	5.25	1.29
2	Past performance	77	5.19	1.16
3	Neighbor ratings histogram	78	5.09	1.22
4	Table of neighbors ratings	78	4.97	1.29
5	Similarity to other movies rated	77	4.97	1.50
6	Favorite actor or actress	76	4.92	1.73
7	MovieLens percent confidence in prediction	77	4.71	1.02
8	Won awards	76	4.67	1.49
9	Detailed process description	77	4.64	1.40
10	# neighbors	75	4.60	1.29
11	No extra data – focus on system	75	4.53	1.20
12	No extra data – focus on users	78	4.51	1.35
13	MovieLens confidence in prediction	77	4.51	1.20
14	Good profile	77	4.45	1.53
15	Overall percent rated 4+	75	4.37	1.26
16	Complex graph: count, ratings, similarity	74	4.36	1.47
17	Recommended by movie critics	76	4.21	1.47
18	Rating and %agreement of closest neighbor	77	4.21	1.20
19	# neighbors with std. deviation	78	4.19	1.45
20	# neighbors with avg correlation	76	4.08	1.46
21	Overall average rating	77	3.94	1.22

Table 1. Mean response of users to each explanation interface, based on a scale of one to seven. Explanations 11 and 12 represent the base case of no additional information. Shaded rows indicate explanations with a mean response significantly different from the base cases (two-tailed $\alpha = 0.05$).

Figure 1. One of the twenty-one different explanation interfaces given shown in the user survey. Notice that the title has been encoded, so that it does not influence a user's decision to try a movie.

Explaining collaborative filtering recommendations
JL Herlocker, JA Konstan, J Riedl
Proceedings of the 2000 ACM conference on Computer supported cooperative ... 1264 2000

Example


NETFLIX
Watch Instantly
Just for Kids
Taste Profile
DVDs
DVD Queue
John Terry
Because you watched How I Met Your Mother
RULES OF ENGAGEMENT
the office
That '70s Show
30 ROCK
EVERYBODY LOVES RAYMOND
Emotional Movies
Based on your interest in...
FISH
WHAT MAISIE KNEW
Listen to Your Heart
OCTOBERBABY
BEST MAN DOWN
Because you watched Movie 43
Zack and Miri make a porno
PARANORMAL WHACKTIVITY
jackass number two

Master on Foundations of Data Science

Recommender Systems

Collaborative Recommender Systems - RecChallenge

Santi Seguí | 2016-2017

ACM RecSys Challenge 2017

About ▾ Participate ▾ Timeline Workshop ▾ Challenge 2016 RecSys conference

We are currently finishing the datasets, evaluation metrics and submission system. Please follow us on Twitter to get updates: @recsyschallenge

About

The ACM RecSys Challenge 2017 is focussing on the problem of job recommendations on XING in a cold-start scenario. The challenge will consists of two phases:

1. **offline evaluation:** fixed historic dataset and fixed targets for which recommendations/solutions need to be computed/submitted.
2. **online evaluation:** dynamically changing targets. Recommendations submitted by the teams will actually be rolled out in XING's live system (details about the online evaluation such as the approach for ensuring fair conditions between the teams will follow).

Both phases aim at the following task:

> **Task:** given a new job posting p, the goal is to identify those users that (a) may be interested in receiving the job posting as a push recommendation and (b) that are also appropriate candidates for the given job.

For both offline and online evaluation, the same evaluation metrics and the same types of data sets will be used. The offline evaluation is essentially used as an entry gate to the online evaluation:

http://2017.recsyschallenge.com/

RecSys Challenge

The ACM RecSys Challenge 2017 is focussing on the problem of job recommendations on XING in a cold-start scenario. The challenge will consists of two phases:

1. **offline evaluation:** fixed historic dataset and fixed targets for which recommendations/solutions need to be computed/submitted.
2. **online evaluation:** dynamically changing targets. Recommendations submitted by the teams will actually be rolled out in XING's live system (details about the online evaluation such as the approach for ensuring fair conditions between the teams will follow).

Timeline ▲ top

When?	What?
Beginning of March	**RecSys challenge starts:** • RecSys challenge starts with *offline evaluation* • Submission system will be available via recsys.xing.com (currently offline) • Maximum number of submissions per day: 20
April 16th (23:59 Hawaiian time)	**Offline evaluation ends:** • Based on the overall leaderboard of the *offline challenge*, the top teams (probably top 20) are asked to join the *online challenge* • Top teams get access to API of the online challenge. • The submission of the offline challenge will continue to be open • After this deadline there will be a possibility to join the online challenge for teams who achieve awesome scores
May 1st	**Online challenge starts:** • Every day, new target items will be released for which the teams are supposed to compute recommendations, i.e. identify users that may be interested in these items. • Teams download the new target list via API, compute recommendations and submit their solutions via API. Teams can script their recommender systems to regulalry pull from the API to check for updates.
June 4th (23:59 Hawaiian time)	**Online evaluation ends:** • Last day of online evaluation • Submission system closes
June 12th	• Official results will be announced • Winner of the challenge = winner of the online challenge
June 18th	**Paper submission deadline** for RecSys Challenge workshop
July 3rd	Notifications about paper acceptance
July 17th	Deadline for camera-ready papers
August	Workshop will take place as part of the RecSys conference in

Challenges ▲ top

Some challenges that the participating teams will need to solve:

- **Balancing user interest and recruiter demands:** In contrast to last year's challenge which was solely focusing on estimating how relevant a job is for a given user, this year we will focus on both: Job recommendations should be relevant to the users and at the same time the users who receive those job recommendations also need to be appropriate candidates for the given job (e.g. the fact whether a user received interest from a given recruiter is part of the evaluation measure).
- **Balancing relevance and revenue:** Some of the content is paid and some users pay for subscriptions. Teams will need to balance between relevance of recommendations and monetary aspects (i.e. the money that is earned with the recommendation).
- **Novelty / sparsity:** recommendations need to be computed particularly for newly created job postings (those postings have not received any interaction).
- **Smart targeting of push recommendations:** also, teams will need to estimate how likely it is that a user is actually interested in job recommendations, e.g. users that are not interested in job recommendations may delete recommendations or disable push recommendation notifications in case they receive too many (the latter is primarily relevant for teams that participate in the online challenge).

Evaluation Metrics ▲ top

Given a list of target items `targetItems`, for which the recommender selects those users to whom `item in T`, is pushed as recommendation, we compute the the leaderboard score as follows:

```
score(targetItems) = targetItems.map(item => score(item, recommendations(item))).sum
```

Here, `recommendations(item)` specifies the list of users who will receive the given item as push recommendation. The function `score(item, users)` is defined as follows:

```
score(item, users) =
users.map(u => userSuccess(item,u)).sum + itemSuccess(item, users)

userSucess(item, user) =
  (
      if (clicked) 1 else 0
    + if (bookmarked || replied) 5 else 0
    + if (recruiter interest) 20 else 0
    - if (delete only) 10 else 0
  ) * premiumBoost(user)

premiumBoost(user) = if (user.isPremium) 2 else 1

itemSuccess(item, users) =
  if (users.filter(u => success(item, u) > 0).size > 1) {
    if (item.isPaid) 50
    else 25
  } else 0
```

Dataset released

03 | 03 | 2017

UNIVERSITAT DE BARCELONA

DATA SCIENCE @ UNIVERSITAT DE BARCELONA

```
$  unzip -v
UnZip 5.52 ....

$ unzip data_2017.zip
Archive:  data_2017.zip
  inflating: interactions.csv
  inflating: items.csv
  inflating: targetItems.csv
  inflating: targetUsers.csv
  inflating: users.csv

$ ls -alh
... 1.4G   Mar  3 22:53   data_2017.zip
... 8.4G   Mar  2 15:50   interactions.csv
... 226M   Mar  1 21:29   items.csv
... 341K   Mar  2 18:01   targetItems.csv
... 550K   Mar  3 16:02   targetUsers.csv
... 82M    Mar  1 21:27   users.csv

$ wc -l interactions.csv
322776003 interactions.csv

$ head interactions.csv
recsyschallenge_v2017_interactions_final_anonym_training_export.user_id recsyschallenge_v2017_intera
2082156 80      1       1484299172
1934123 140     1       1486388563
1320213 240     1       1479409825
297303  310     1       1484817366
1635596 310     1       1486370081
857319  340     1       1485121421
324595  350     1       1484591946
510320  350     1       1484841341
499620  390     1       1479387826
```

Users ▲ top

users.csv: Details about those users who appear in the above datasets.
Fields:

- `id` anonymized ID of the user (referenced as `user_id` in the other datasets above)
- `jobroles` comma-separated list of jobrole terms (numeric IDs) that were extracted from the user's current job titles
- `career_level` career level ID (e.g. beginner, experienced, manager):
  - 0 = unknown
  - 1 = Student/Intern
  - 2 = Entry Level (Beginner)
  - 3 = Professional/Experienced
  - 4 = Manager (Manager/Supervisor)
  - 5 = Executive (VP, SVP, etc.)
  - 6 = Senior Executive (CEO, CFO, President)
- `discipline_id` anonymized IDs represent disciplines such as "Consulting", "HR", etc.
- `industry_id` anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
- `country` describes the country in which the user is currently working
  - de = Germany
  - at = Austria
  - ch = Switzerland
  - non dach = non of the above countries
- `region` is specified for some users who have as country `de`. Meaning of the regions see below
- `experience_n_entries_class` identifies the number of CV entries

Items ▲top

items.csv: Details about the job postings that were and should be recommended to the users.

- `id` anonymized ID of the item (referenced as `item_id` in the other datasets above)
- `industry_id` anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
- `discipline_id` anonymized IDs represent disciplines such as "Consulting", "HR", etc.
- `is_paid` indicates that the posting is a paid for by a compnay
- `career_level` career level ID (e.g. beginner, experienced, manager)
  - 0 = unknown
  - 1 = Student/Intern
  - 2 = Entry Level (Beginner)
  - 3 = Professional/Experienced
  - 4 = Manager (Manager/Supervisor)
  - 5 = Executive (VP, SVP, etc.)
  - 6 = Senior Executive (CEO, CFO, President)
- `country` code of the country in which the job is offered
- `latitude` latitude information (rounded to ca. 10km)
- `longitude` longitude information (rounded to ca. 10km)
- `region` is specified for some users who have as country `de`. Meaning of the regions: see below.
- `employment` the type of emploment
  - 0 = unknown
  - 1 = full-time
  - 2 = part-time

Interactions ▲ top

interactions.csv: Interactions are all transactions between a user and an item including recruiter interests as well as impressions. Fields:

- `user_id` ID of the user who performed the interaction (points to `users.id`)
- `item_id` ID of the item on which the interaction was performed (points to `items.id`)
- `created_at` a unix time stamp timestamp representing the time when the interaction got created
- `interaction_type` the type of interaction that was performed on the item:
  - 0 = XING showed this item to a user (= impression)
  - 1 = the user clicked on the item
  - 2 = the user bookmarked the item on XING
  - 3 = the user clicked on the *reply button* or *application form button* that is shown on some job postings
  - 4 = the user deleted a recommendation from his/her list of recommendation (clicking on "x") which has the effect that the recommendation will no longer been shown to the user and that a new recommendation item will be loaded and displayed to the user
  - 5 = a recruiter from the items company showed interest into the user. (e.g. clicked on the profile)

RecSys Challenge

- Training database is huge:

	#
Users	**1.497.020**
Items	**1.306.054**
Interactions	**322.776.002**

- Test Set:
  - Items -> 46.559 (not prior interactions)
  - Users -> 74.840

Training Dataset density

n = 1.306.054

m = 1.497.020

$$\text{Density} = \frac{26.614.333}{1.306.054 \times 1.497.020} = 0.00136\ \%$$

Training Dataset density removing rows/columns with all 0

n = 613.823

m = 1.193.996

$$\text{Density} = \frac{26.614.333}{1.193.996 \times 613.823} = 0.00363\ \%$$

Test Dataset

n = 46.559

m = 74.840

Training Items
1.306.054

Test Items
46.559

Training Users
1.193.996

Test Users
74.840

Mostly Zeros
impressions +
recruiter

some of them without any previous interaction

Interactions ▲ top

interactions.csv: Interactions are all transactions between a user and an item including recruiter interests as well as impressions. Fields:

In [11]:
```
interactions.head()
```

Out[11]:

	user_id	item_id	interaction_type	created_at
0	2082156	80	1	1484299172
1	1934123	140	1	1486388563
2	1320213	240	1	1479409825
3	297303	310	1	1484817366
4	1635596	310	1	1486370081

In [14]:
```
interactions[['user_id','interaction_type']].groupby(by="interaction_type").count()
```

Out[14]:

	user_id
interaction_type	
0	314501101
1	6867579
2	281672
3	117843
4	906836
5	100971

Interactions Table

Interaction Type	#Interactions	#Interactions removing duplicates
0 (impression)	314.501.101	21.675.180
1 (click)	6.867.579	5.091.857
2 (bookmarked)	281.672	281.672
3 (reply)	117.843	92.330
4 (deleted)	906.836	906.836
5 (recruited showed interests)	100.971	2.502
Total	**322.776.002**	**28.050.377**

	items	users	Score
#	1.306.054	1.497.020	0
# in interactions file	613.823	1.193.996	0
# with interactions (1-5)	538.683	689.144	-
# interaction 1	511.822	671.948	1/2
# interaction 2	96.223	89.143	5/10
# interaction 3	52.181	47.828	5/10
# interaction 4	166.142	74.508	-10/-20
# interaction 5	1.763	1.288	25/50

Total Number of interactions
322.776.002

Number of interaction without duplicates
28.050.377

Number of joint interactions user/item
26.614.333

Users in the target_items

- **74.840** Different users
- **24.184.053** interactions from these users
    - Same interaction from a user to at different timestamps
- Removing repeated interactions
    - **2.258.628**

Baseline released

3-03-2017

Baseline

This is the simple baseline that creates the sample_solution.csv file. The baseline system extracts features from interacted user-item pairs and "learns" if a user will interact positively with an item. The underlying learning algorithm is XGboost. Which is a tree ensembeling method that most winning teams used last year.

The features are:

- number of matches in title ids [Int]
- discipline matches [0, 1]
- career level matches [0, 1]
- industry matches [0, 1]
- country match[0, 1]
- region match [0,1]

Files:

- The data model along with the feature extraction can be found in `model.py`.
- Parsing the data is performed in `parser.py`.
- A parallel prediction algorithm is given in `recommendation_worker.py`. The recommendation worker uses the target items and users. If we have a title match at all the score for each user is predicted.
- The main training and testing is performed in `xgb.py`

In order to build the baseline XGboost has to be installed with the python binindgs.

```
users.head()
```

	user_id	jobroles	career_level	discipline_id	industry_id	country	region	experience_n_entries_class	experi
0	30	2551922	3	0	0	de	0	1	3
1	50	4375874,3415336,2152789,1431010	3	4	15	de	7	3	4
2	70	851763,2070276	3	17	4	de	2	2	7
3	90	2139882,2177068,1520218,3113130,399936	0	0	0	de	0	1	0
4	100	233434,3142896,3836967,987884	3	0	16	non_dach	0	3	5

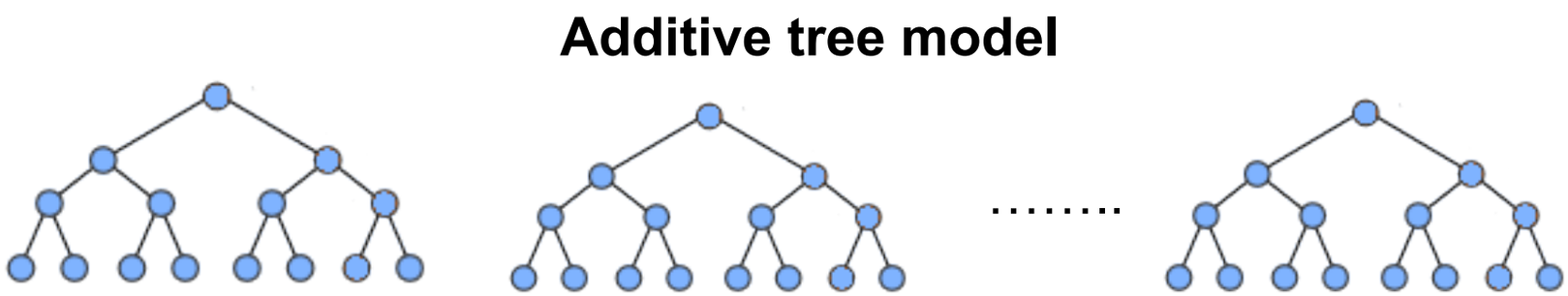
In [30]:
```
items.head()
```

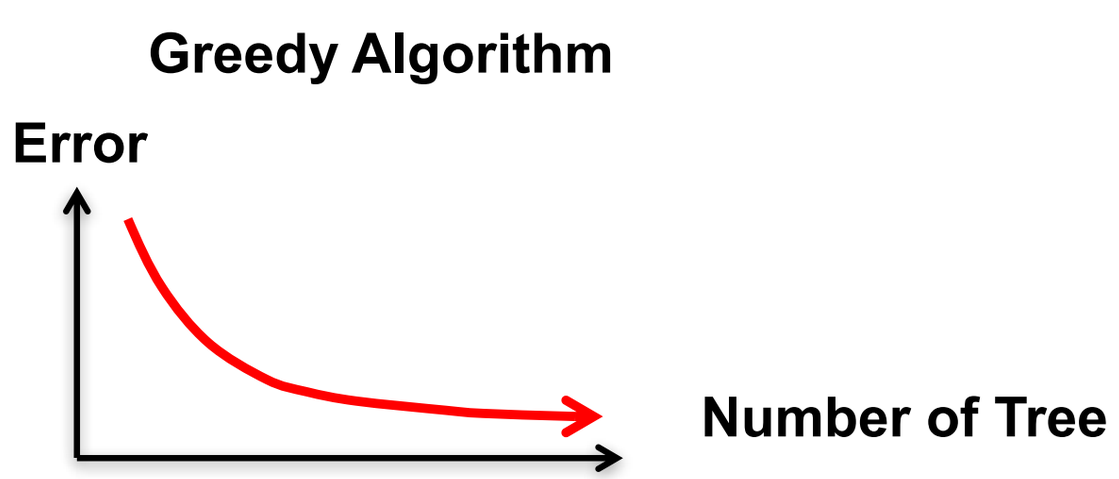
Out[30]:

	item_id	title	career_level	discipline_id	industry_id	country	is_payed	region	latitude	longitude	employm
0	30	NaN	3	15	9	de	0	7	50.8	8.9	1
1	70	2994300,665762,901938,127655,3680343	3	5	18	de	0	1	48.7	9.0	1
2	80	3275701,157228,4179835	1	10	1	de	0	1	48.8	8.1	1
3	90	2343474	3	8	9	de	0	3	52.5	13.4	1
4	100	2003688	4	8	3	at	0	0	48.1	15.1	1

XGBoost

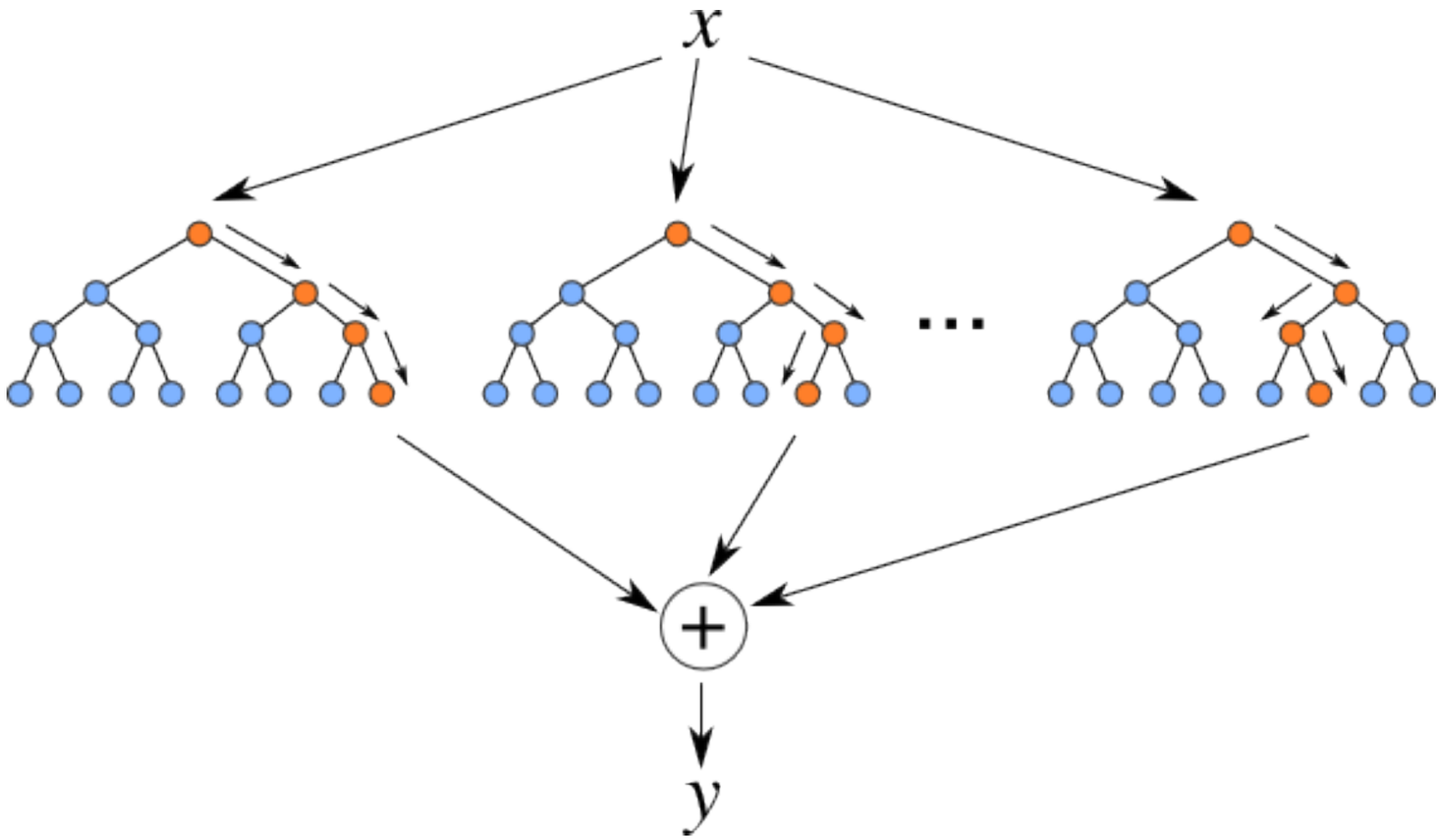
- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy





XGBoost

- Additive tree model: add new trees that complement the already-built ones
- Response is the optimal linear combination of all decision trees
- Popular in Kaggle competitions for efficiency and accuracy



Leaders

Rank	Team	Score
1	passionate17	13337
2	RecSys@DataScienceUB	10457
3	Donauschifffahrtsgesellschaftsdatenwissenschaftler	10240
4	Lunatic Goats	10184
5	Simple baseline by organizers	10004
5	XING Daniel Dan The Daniel Man	10004
5	Avito	10004
5	FuzzyPrint	10004
5	feijoada	10004
5	vha	10004
5	leavingseason	10004

How to win?

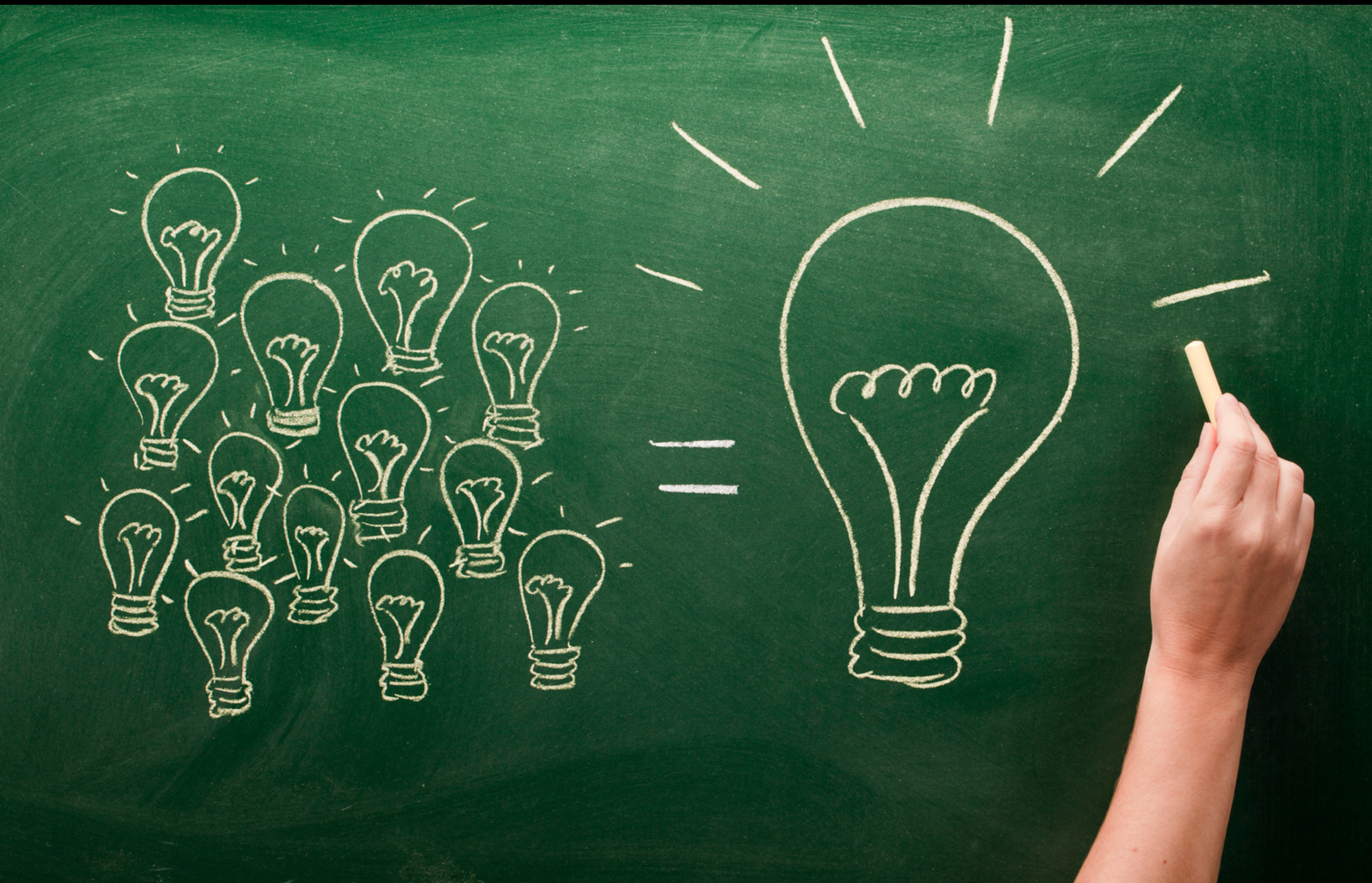


Table 3: Key submissions

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

Job recommendation based on factorization machine and topic modelling

Full Text: PDF

Authors: Vasily Leksin Avito.ru
Andrey Ostapets Avito.ru

2016 Article

Published in:

· Proceeding
RecSys Challenge '16 Proceedings of the Recommender Systems Challenge
Article No. 6

Boston, Massachusetts — September 15 - 15, 2016
ACM New York, NY, USA ©2016
table of contents ISBN: 978-1-4503-4801-0
doi>10.1145/2987538.2987542

Bibliometrics
· Citation Count: 0
· Downloads (cumulative): 35
· Downloads (12 Months): 35
· Downloads (6 Weeks): 22

Alibaba.com

Job Recommendation with Hawkes Process

W. Xiao, X. Xu, K. Liang, J. Mao, and J. Wang
OneSearch Team, Alibaba Group
Boston, MA, USA

RECSYS / BOSTON / 2016

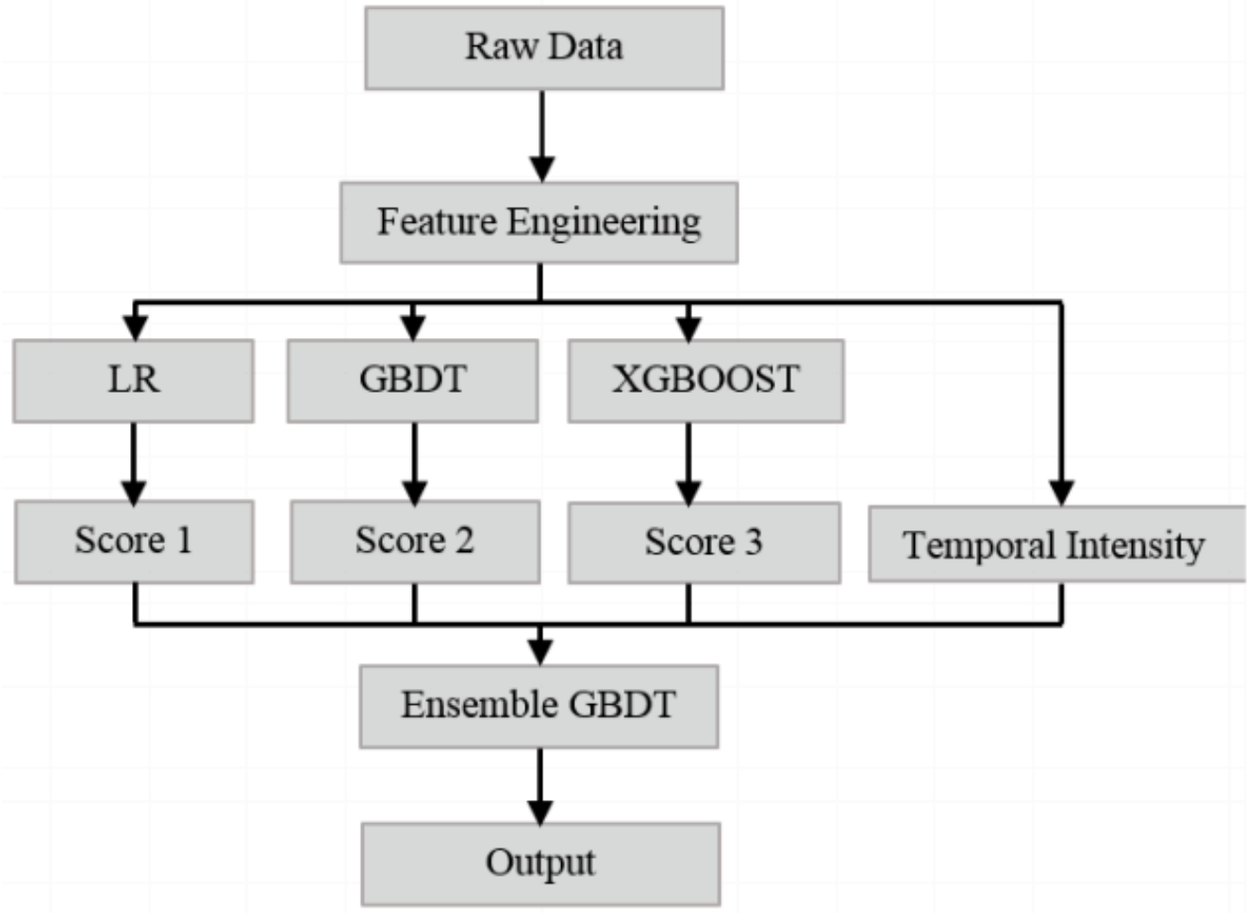
Overall Framework

An Hierarchical LTR framework

- LTR with various models to absorb diverse information
- Ensemble GBDT with the input from initial ranking
- Hawkes Process (self-exciting) to capture temporal patterns

Final Goal

- Generate the right list of job posting to recommend in the right timing



Features from Low to High Level

Alibaba.com

user&item profile: 0/1

user/item/ui interaction statistics: (time window) 0/1, count, distinct count, days……

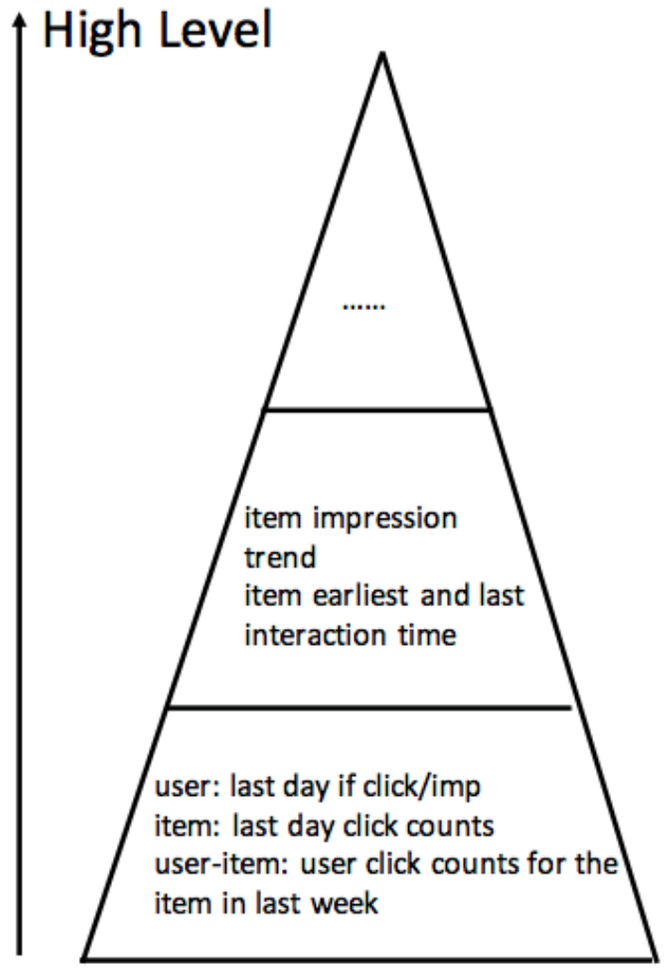
similarity(interaction&profiling) : u2u, i2i, u2i

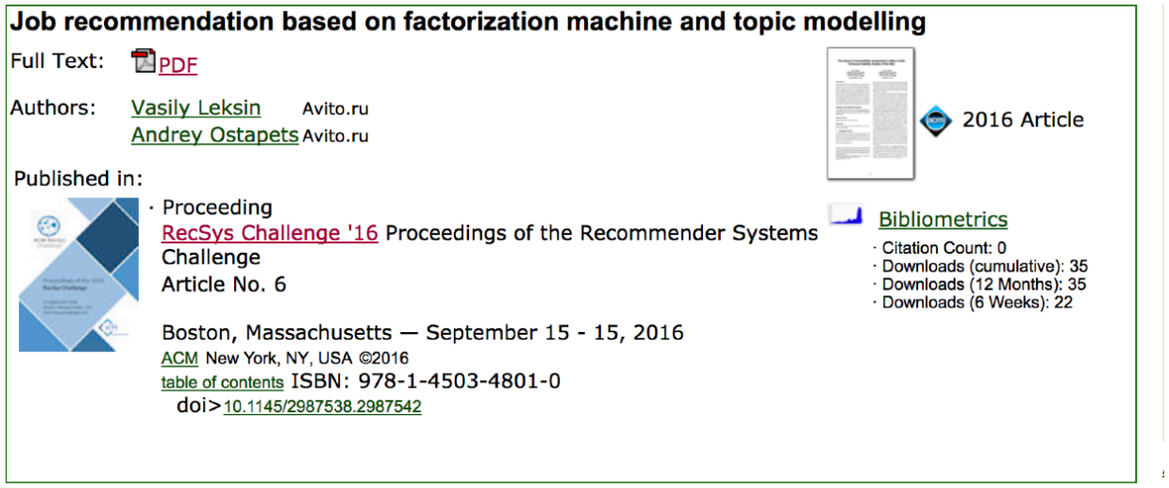
ratio: ui/u, ui/i

trend: item interaction counts

poi: item, user interaction

conditional probability: impression to interaction, interaction to interaction





Job recommendation based on factorization machine and topic modelling

Full Text: PDF

Authors: Vasily Leksin Avito.ru
Andrey Ostapets Avito.ru

Published in:

· Proceeding
RecSys Challenge '16 Proceedings of the Recommender Systems Challenge
Article No. 6

Boston, Massachusetts — September 15 - 15, 2016
ACM New York, NY, USA ©2016
table of contents ISBN: 978-1-4503-4801-0
doi>10.1145/2987538.2987542

2016 Article

Bibliometrics
· Citation Count: 0
· Downloads (cumulative): 35
· Downloads (12 Months): 35
· Downloads (6 Weeks): 22

- "Impressions model"

?

- Item Based Method

$$score(u,j) = \frac{\sum_{i \in I_u} sim(i,j)}{|I_u|}$$

- Factorization Models

We will see soon

- Content Based Approach

Topic Modelling

User-similarity based on interactions

User-similarity based on content

Item-similarity based on content

Prediction function

How to compute the score of the (user,item)

$$\text{Score}(user,item) = w_1(n) * CB + w_2(n) * CF + w_3(n) * Other?$$

How can we validate it?

- Create a validation wet with a subset of the interactions? interactions from the last week

Date	Member	Method	Result	Position
8/03/03	Santi	Baseline strategy no impressions new score accoring the challenge score	11514	3

How to work?

- Private GitHub?
  - + Wiki
- Where can we share the results? GitHub, drive, dropbox,…

Baseline

This is the simple baseline that creates the sample_solution.csv file. The baseline system extracts features from interacted user-item pairs and "learns" if a user will interact positively with an item. The underlying learning algorithm is XGboost. Which is a tree ensembeling method that most winning teams used last year.

The features are:

- number of matches in title ids [Int]
- discipline matches [0, 1]
- career level matches [0, 1]
- industry matches [0, 1]
- country match[0, 1]
- region match [0,1]

Just the intersection?

What about correlated disciplines?

same importance all jobs? all levels? what about higher level?

just match? Barcelona nicer than Madrid

Distance?

Files:

- The data model along with the feature extraction can be found in `model.py`.
- Parsing the data is performed in `parser.py`.
- A parallel prediction algorithm is given in `recommendation_worker.py`. The recommendation worker uses the target items and users. If we have a title match at all the score for each user is predicted.
- The main training and testing is performed in `xgb.py`

In order to build the baseline XGboost has to be installed with the python binindgs.

More questions

- Who is contacted by the recruiters?
- Is there something that makes them special?
- Which are the items that makes recruiters contact users?