A HYBRID APPROACH FOR NEWS RECOMMENDER SYSTEM USING OPTIMIZATION METHODS

A PREPRINT

Alexander Smirnov
Department of Information and Analytical Systems
Saint Petersburg State University
Russia, Saint Petersburg
ru.alexander.smirnov@gmail.com

Elena Mikhailova
Department of Information and Analytical Systems
Saint Petersburg State University
Russia, Saint Petersburg
e.mikhaylova@spbu.ru

February 25, 2021

ABSTRACT

Recommender system is an essential part of any social media application. Most recommender systems now use a hybrid approach, combining collaborative filtering, content-based filtering, and other approaches. Most common problems in the field of hybrid recommenders are cold start and data sparsity [Çano, 2017]. In this paper we address the abovementioned problems by proposing a hybrid weighted news recommender system which combines different approaches.

Keywords hybrid recommender systems · content-based recommender · collaborative recommender · optimizations

1 Introduction

Recommender system is a crucial part of every application that operates with content and user activity. Enormous amount of information leads to the problem that user is not able to find relevant content.

Common approaches, such as collaborative filtering, has its own problems: cold start, scalability and data sparsity. Content-based approaches suffer from the fact that we have to somehow represent recommended item in feature space.

To be consistent during the paper we list some domain specific vocabulary with their meanings:

- Rating: expression or preference
  - explicit (direct from user, e.g. user rated film)
  - implicit (inferred from user activity, e.g. user stopped watching movie after 5 minutes)
- Prediction: estimate of preference
- Recommendation: selected items for user
- Content: attributes, text, etc; everything about item

The remainder of this paper is organized as follows:

- Section 2 describes the relevant related work
- Section 3 describes input data
- Section 4 explains our modular design and architecture
- Section 5 describes the implementation of the algorithms in a real system
- Section 6 provides tests and experiments validating our systems results
- Section 7 explains future work
- Section 8 presents conclusions

2 Related work

According to the study [Dacrema et al., 2019], deep learning techniques are not supposed to beat conceptually and computationally simpler algorithms, so we won't touch them.

Our goal is to choose optimal algorithm for each of the following tasks:

- **Collaborative filtering**: generating predictions about the interests of a user by collecting preferences or taste information from other users. It is based on the assumption that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person
- **Content-based filtering**:
- **Session filtering**:
- **Popularity filtering**:
- **Demographic filtering**:
- **Time-based filtering**:

We face the problem that millions of news are theoretically suitable for being recommended, so it is not correct to use abovementioned methods on such large corpus of data. Instead, as stated in [Covington et al., 2016] paper, we want to implement `candidate generation` $\rightarrow$ `ranking` pipeline to reduce number of candidates.

2.1 Collaborative filtering

There were many studies on this topic, but paper [Rendle et al., 2019] proves that well-tuned basic SVD++ approach beats newely presented algorithms.

3 Input data

As we solving domain specific task, we have domain specific data.

metadata

item_id	*date*	*source_id*	*category*
1	2021-01-08 22:08:39	9	politics, conflicts
2	2021-01-09 10:28:58	5	IT, social media
3	2021-01-09 14:20:34	12	accident
⋮	⋮	⋮	⋮

Table 1: news metadata

- *source_id*: source of news item

content

item_id	*news title*	*news content*
1	Azerbaijan denies reports on construction of Turkish air bases in the country	Information that Turkey will create air bases ...
2	Durov announced the massive transition of WhatsApp users to Telegram	Telegram developer Pavel Durov said in his channel ...
3	Passenger plane that disappeared from radar crashed	Passenger plane taking off from Jakarta, disappeared ...
⋮	⋮	⋮

Table 2: news item

shows & views

- *shows*: if *item_id* was shown to the *user_id*

user_id	*item_id*
10	1
10	2
23	1
23	3
23	2
38	3
38	1
⋮	⋮

Table 3: news shows

user_id	*item_id*
10	1
10	2
23	1
38	3
⋮	⋮

Table 4: news views

- *views*: if *item_id* was clicked by the *user_id*

emotions & comments

There is an option to react on item via leaving emotion and/or writing a comment.

user_id	*item_id*	*emotion_id*
10	1	1
10	2	3
23	1	3
38	3	2
⋮	⋮	⋮

Table 5: users' emotions

user_id	*item_id*	*comment*
10	1	that's great
10	1	wish it will continue
23	2	whatsapp is not competetive anymore
⋮	⋮	⋮

Table 6: users' comments

- *emotion_id*: one of {😄, 😮, ☹, 😣, ♥ }

users' subscriptions

If *user_id* subscribed to the *source_id*.

user_id	*source_id*
10	9
23	5
⋮	⋮

Table 7: users' subscriptions

4 Overview of our approach

Out goal is to combine state-of-the-art approaches in recommender systems.

Solution consists of 2 parts:

- **Candidate generation**: lowering number of items to recommend. These candidates are intended to be generally relevant to the user with high precision. The candidate generation part only provides broad personalization
- **Ranking**: applying state-of-the-art algorithms to rank candidates generated on previous step

Architecture provided below on fig. 1:

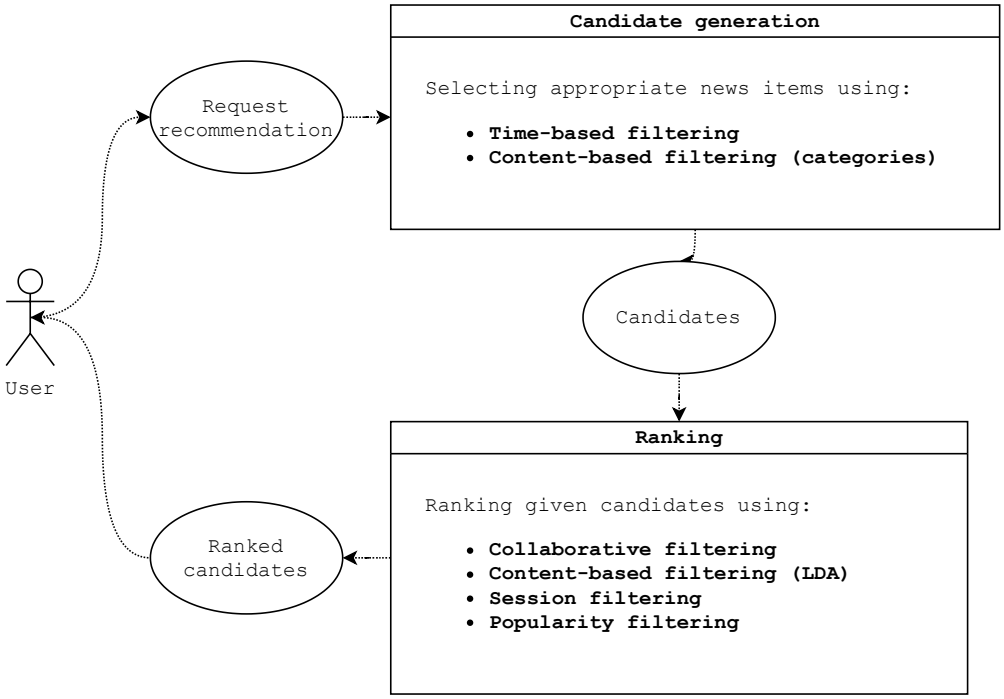


Figure 1: architecture

Candidate generation stage not only filters items, but also attaches weight to items.

4.1 Candidate generation components overview

Goal of candidate generation step is to remove generally irrelevant items so furher algorithms won't suffer from the amount of data. Both time-based filtering and content-based filtering have their own weight at start. As user more interacting with the system, content-based filtering increasing its weight.

Using recommenders described below, we attach score to each item and pick top n (e.g. 50000) items.

4.1.1 Time-based filtering

As we operating with news data, first filter is the time filter. This part consists of 2 steps:

- **Filtering**: remove all items which are older than 3 days
- **Ranking**: attach weights to all items left from filtering

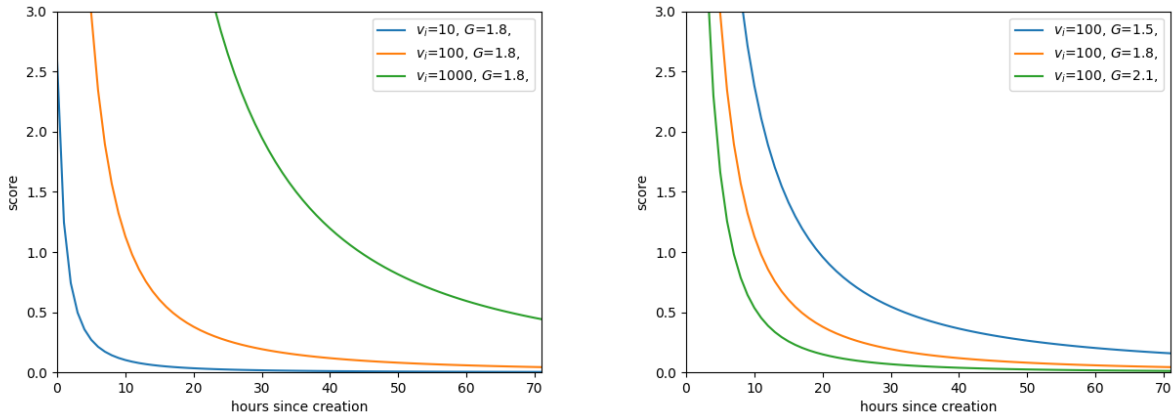
For ranking we will use following formula:

$$r_i = \frac{(v_i - 1)}{(t - t_i + 2)^G}$$

- r_i – score for $item_i$
- v_i – number of views of $item_i$
- t – time right now, t_i – time of creation of $item_i$, $t - t_i$ – hours passed since item created
- G – gravity factor

The score decreases as $t - t_i$ increases, meaning that older items will get lower and lower scores. v_i gets substracted by -1 to negate submitter's view. $t - t_i$ increased by 2 so even if $t - t_i = 0$ gravity factor G will take effect.

Below are show dependencies between score and hours since creation:



(a) different views number

(b) different gravity factor

Figure 2: time-based ranking scoring

At the end scores are normalized by `min-max` normalization.

4.1.2 Content-based filtering (categories)

As was told before, content-based filtering have its own impact weight, which is small at start (we don't want to restrict user from content just because he made some random clicks), but it inscreases as user interacts with the system and we may make predictions about his categorial preferences.

4.2 Ranking components overview

4.2.1 Collaborative filtering

We are using SVD++ algorithm.

For collaborative filtering recommendation we should have something known as user-item matrix which may be formed from user activity from tables (cite tables)

4.2.2 Popularity filtering

For measuring news item popularity following data can be aggregated: shows, views, emotions, comments.

item_id	*shows_num*	*views_num*	*emotions_num*	*comments_num*
1	1043	231	52	7
2	828	478	78	11
3	163	25	5	0
⋮	⋮	⋮	⋮	⋮

Table 8: aggregated popularity data

4.2.3 Session filtering

4.2.4 Content-based filtering

5 Implementation

6 Evaluation

For evaluation we have information about what recommendation list was given to each user, and what is the source of the recommendation:

user_id	*recommendation_list*	*content_based_filtering*	*collaborative_filtering*
2	{(2, 0.91), (1, 0.74), (3, 0.23)}	{(2, 0.45), (1, 0.54), (3, 0.08)}	{(2, 0.92), (1, 0.4), (3, 0.3)}
1	{(3, 0.73), (1, 0.69), (2, 0.15)}	{(2, 0.6), (1, 0.44), (3, 0.04)}	{(2, 0.58), (1, 0.58), (3, 0.14)}
⋮	⋮	⋮	⋮

Table 9: recommendations' logs

- *recommendation_list*: list of recommendations that consists of pairs (*item_id*, *score*)
- *content_based_filtering & collaborative_filtering*: sources of recommendation

As we use weighted sum of our recommmenders, we have unique boost values for every user:

user_id	*content_based_filtering*	*collaborative_filtering*
1	0.25	1
2	1	0.5
⋮	⋮	⋮

Table 10: boost values

To illustrate score calculation of recommendations, take a look at 1st row of table recommendations' logs. We see recommendation $r = \{(2, 0.91), (1, 0.74), (3, 0.23)\}$ for user $u = 2$, so we should find boost values for this particular user in table boost values: content-based filtering boost b_{cbf} for u is 1, collaborative filtering boost b_{cf} is 0.5. So final score is calculated in the following way: $0.91 = 0.45 * b_{cbf} + 0.92 * b_{cf} = 0.45 * 1 + 0.92 * 0.5 = 0.45 + 0.46$.

Also we have information about shows and views, so we are able to track what item in recommendation list was clicked and what item was skipped, so depending on this information we can track if user liked item or not.

6.1 Online

6.2 Offline

7 Further research

8 Summary

Results show that combining different approaches leads to rise of users' involvement.

References

Erion Çano. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21:1487–1524, 11 2017. doi:10.3233/IDA-163209.

Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 101109, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi:10.1145/3298689.3347058.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016.

Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems. 05 2019.