Introduction

The Scipr system is an online Recommender Algorithm (RA) Evaluation application where developers can integrate their RA and based on user feedback can gauge the RA performance based on a number of metrics. This document will present a conceptual workflow of different activities ,the evaluation metrics planned to be incorporated and certain design decisions for those metrics.

Objective:

As previosuly stated our main aim is to give algorithm developers exposure to a large number of actual users who can objectively assess their RA.

In this context we have to understand that our test subjects are real people and not a pre collected dataset. So an individual needs some motivation to be part of an experiment. The motivation here would be the access to a novel system that helps him in his research -

- Provides a personalised scientific paper catalog: A repository(metadata only) of all scientific papers he has read and liked.
- Scientific paper search and recommendation capabilities together: A system that connects to IEEE, DBLP, citeseer etc and ability to use his choice of RA to provide recommendations

System Functionalities:

We present the major functionalities of ScipR:

1. *Searching for scientific papers and Idea of Personal paper catalog or My Papers*

a. For a new user system cannot provide recommendations (Cold start not within scope of the project). However a user can Search for papers using the ScipR system.

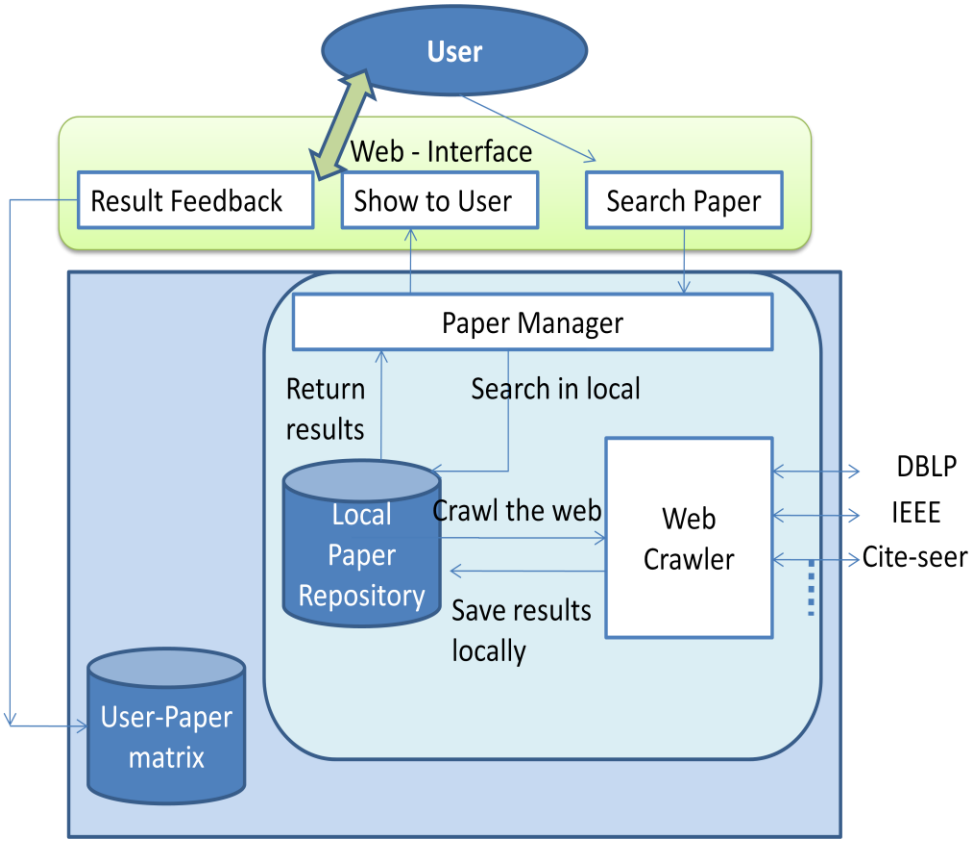
b. If the paper is found locally, it will be shown to the user else the Paper Manager using its API will search in external databases and then return the results to the user.

c. If the returned paper is what user was looking for, he can add it to his My_Papers repository. Addition of a paper to personal repository is type of user feedback

My_Papers is like the user_paper matrix , the input feature dataset for many RAs. Here the user feedback is with respect to the Search results and there are two types of feedback we propose to collect:

1. Relevance: Implicitly collected when user adds a paper to his My_Paper catalog
2. Rating : For non rated inputs (but added to my_catalog), save with a neutral rating(if user hadn't liked it, he wouldn't have added to his catalog. So definitely not a 0 rating).

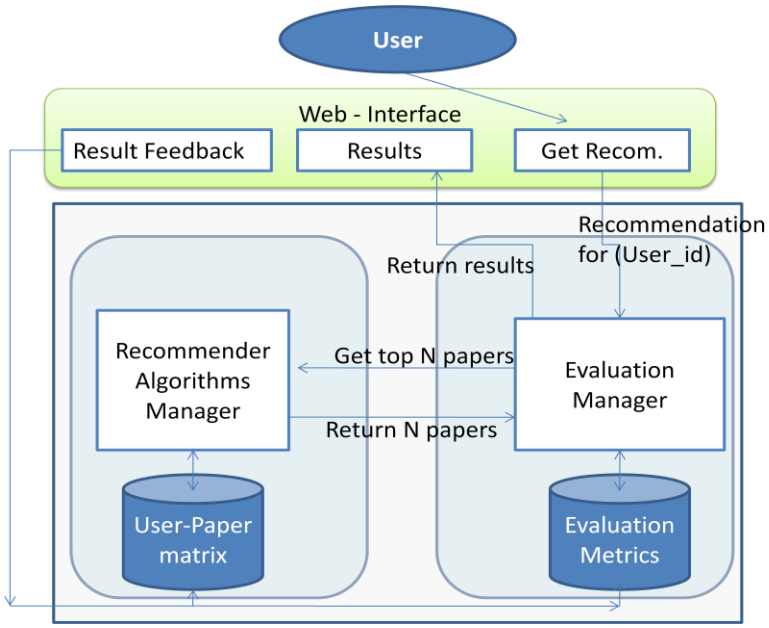
A representation of the above functionality:



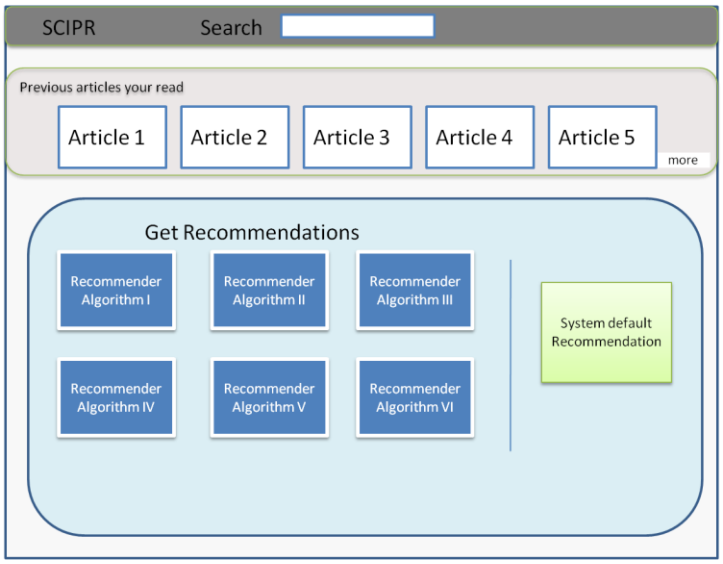
2. *Getting Recommendations for scientific papers:* The user will have the flexibility to choose the RA for getting recommendations. However the request shall be routed through the Evaluation Manager

a. User sends a recommendation request. System forwards it to Evaluation Manager
b. Eval Manager calls the appropriate RA's get_recommendation method exposed by RA Manager.
c. The results returned by RA manager are then logged by Evaluation Manager and then presented to the user.
d. User gives feedback on the results:
    i. Add to personal catalog: This means the paper is relevant
    ii. Give a rating

This feedback will be analysed with respect to different parameters and presented to RA developers.



Recommendation method presentation



<Result presentation>

Relevance

1. If a user adds a paper to his My_catalog only then, the paper is marked as relevant

2. User can provide ratings from this result page. Or user can drill down to a result to view abstract and other details and from that page ,provide ratings or add to his catalog.

3. **Ratings: If a user doesn't add to my_catalog but still rates the paper then if** ratings >=3 we add this paper to user_relevant table

Not relevant documents: We can identify the list of non relevant documents from the above list. If a user views a page with 10 results, but rates only 2-3 results, it means he has seen all the 10 results, but finds only 2-3 relevant. Similarly if a user goes to next page, but does not rates or adds anything, we conclude the next 10 results were not relevant and add them to the non-relevant item list.
Thus for each user we have the list of 1. Relevant items 2. Non-relevant items 3. And unknown Items

If a previously rated article is shown, it will be indicated with a flag. Internally the Evaluation Manager will mark that result as relevant. So we can avoid unnecessary user input.

Although the domain here is Scientific Paper recommendation, the evaluation measures we should provide and the way we gather the corresponding metrics should enable an algorithm developer to predict his RA performance in other areas as well.

In this regard certain design decisions that need to be discussed

1. Many RAs return a predicted rating of a paper. Here an evaluation metric would be the RMSE , MAE etc On the other hand ,many RA return similarity scores. Ex Content based filtering algorithms return Pearson or Cosine similarity scores. So a user's preference rating on a scale of 1to 5 would not match with the predicted similarity score. In such cases RMSE etc doesn't help in evaluating the 2 different types of RA. So we need to know what type of measure is returned by Algorithm manager and depending on that we shall choose evaluation metrics.

Paper_Id	Prediction_Value	Prediction_Type
ASD234	.889	Similarity
ASD333	.886	Similarity

Below table shows the Evaluation metrics applicable for different RA

	Type of Algorithm			
	Rating Prediction	Usage Prediction	Rank Prediction	Hybrid
Rating Metrics like MAE,RMSE etc	X	O	X	?
Usage prediction Precision, Recall	X	X	X	
Ranking Evaluation NDCG,MRR etc	X	X	x	
Coverage	X	X	X	
Novelty, Serendipty etc	X	X	X	X

We cannot compare the RMSE metrics of a rating prediction algorithm with a usage prediction

2. All RAs take certain time to give robust recommendations. Also the type of input these RAs take are different, some require ratings while others require paper metadata. So for an objective evaluation we cannot let the output from one RA , assist in other RA's learning. That is, the learning dataset of

RA ‘x’ should not increase with the invocation of RA ‘y’ (no contribution from user feedback received for other RAs).

~~[Or maybe it is fine. Don’t present any metrics like performance vs Time]~~

a. To achieve this we should maintain separate user-feedback table acting as that RA’s input
b. **But** we will collect the feedback from all RAs to create a single ground truth table for each user. As users try different RAs, the Evaluation Manager will learn the relevance and ratings of different papers to each user.

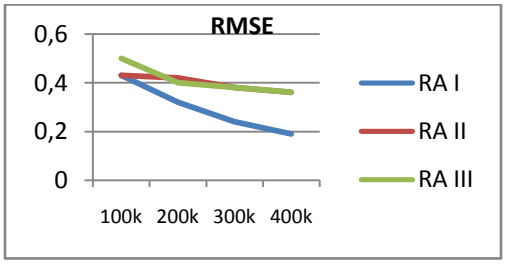
   At the end of each day, the evaluation manager will take recommendations from all RA for each user who used any RA that day. And using the users feedback for the RA he used, evaluate other RA’s performances.(as a kron job Eval manager will run all algorithms) .Thus each RA will get enough feedback to refit its data.

Problem: We may end up ignoring valid results that user would have marked as relevant had he seen the result. We are refitting one algorithm based on another algorithm’s feedback. So better to have a common feedback table which can be referred by all RAs .RAs will then refit their model (daily or hourly) based on this feedback

{ I am not able to convince myself how to go ahead}

RA have certain properties based on which it is evaluated. They have been listed down along with the proposed methods to capture its corresponding metric.

1. Rating Metrics: We shall compute Mean absolute error and RMSE error.
   [Or should we try a distortion measure as we are allowing users to rate on 5 stars. 0-2 stars mean dislike (a reasonable assumption). 3 means neutral and 4-5 mean like. We would then convert stars to ratings as given in [Shani and Gunawardhana]]



   We shall plot Error vs Amount of learning data

   MAE= $\frac{1}{N}\sum_{No.of\ results\ \ viewd\ \ by\ user} |User\ rating - Predicted\ rating|$

   RMSE= $\sqrt[2]{\frac{1}{N}\sum_{No.of\ results\ \ viewd\ \ by\ user} (User\ rating - Predicted\ rating)^2}$

   From wiki

   No of results viewed by user= (results in each page)*(no of pages viewed by user).

2. Prediction Accuracy: We shall present 10 results in each page allowing users to view upto 5 pages. So we shall calculate P@10,P@20...P@50 and similarly R@10...R@50 and avg them out on user

count. Since certain applications require quicker and accurate results and some all relevant results, both curves should be captured.

We would also present a Mean Avg Precision(MAP) vs Amount of learning data. This would be a measure independent of application domain.

AP for each query= P@R1 + P@R2 + P@R3 ..../|R| . MAP will be avg AP for all queries.

Also show Precision-Recall curve

3. Ranking Metrics : We would compute NDCG and MRR . If our rating ground truth table is big enough we can use the rating itself as utility of a result in DCG calculation

   DCG= $rel_1 + \sum_{i=2}^{i=n} \frac{\text{rel i}}{\log(1+i)}$

   where rel_i is the relevance of result at i^{th} rank in the result page. Relevance can be kept as 1 (or the rating if>=3) for relevant documents and 0 otherwise. n is the no. of results viewed by user (i.e 10*pages visited. This may penalize the system which causes the users to view more pages). This will be averaged for all users.

4. Coverage: What proportion of the entire catalog can be recommended by the RA. This is usually based on the input dataset of the algorithm. As a starting point we can find it as

$$\frac{\left|\bigcup_{No\ of\ recommendations\quad each\ ref it\ cycle} relevant\ results\ in\ Recommendation\ SET\right|}{Size\ of\ catalog}$$

   for each algorithm. So if an RA recommends the same Paper many times and its not finding anymore new results its coverage would not increase.

5. Novelty: Those recommendations which are relevant and new.
   a. Here if an RA recommends an article which has been already recommended it will be penalized. One method is:
      We will keep track when a user marked a paper as relevant. Using this info we compute the P@20 (novel results might never be shown in first 10 results). Instead of adding 1 for each true positive (tp), we add 1*k whenever we encounter a result that was marked relevant at a later point in time.
   b. Compute precision over the results, but penalize those results which are very popular (%of users who have marked the paper relevant)

6. Serendipity: This can be calculated using a distance metric(or similarity metric) between rated articles (or the user profile) based on their content. If an RA recommends relevant or highly rated articles which are very dissimilar, we can reward those recommendations. We can again use Precision metric and increase the contribution for those results which are dissimilar but relevant. Another method is based on the assumption “that high accuracy RA’s are giving obvious results and a serendipitous result is something which is deviation from the natural prediction”[287 Recommender Handbook]. We reward those recommendations which the RA would have deemed unlikely (the predicted rating was low but user rated highly, or the RA predicted low probability but it was marked relevant)

Both the above methods require information about the Recommendation Algorithm or calculating similarity scores which might be computationally intensive.

7. Scalability: We can show
    a. Model fitting time vs size of input dataset [and hyperparametrs]
    b. Memory consumption vs size of input dataset [and hyperparametrs]

    Scalability also includes throughput ,but it depends on how we build our Webapp and not on the RA. The Rating metrics like RMSE and MAE also captures certain aspects of scalability.

Other RA properties which need further analysis:
Diversity, Adaptivity,

Properties which might not be relevant:

- Utility :useful for ecommerce websites so might not be relevant here
- Risk :more relevant in stock recommendations
- Robustness: how fake recommendations affect RA. Do not expect any fake data
- Privacy: More relevant for shopping or retail websites.

Methods to be implemented by Recommendation Algorithm Manager

- Get_Recommendation(user_id,n=no of recommendations)
- Set_RelevantPapers(user_id,relevant_paper_array)
- Set_PaperRatings(user_id,[(paper_id:rating), (paper_id:rating),…])