An Experimental Platform for Scholarly Article Recommendation

Ian Wesley-Smith[1], Ralph Dandrea[2], and Jevin West[1]

[1] Information School, University of Washington iwsmith,jevinw@uw.edu
[2] ITX Corp rdandrea@itx.net

Abstract. We describe the experimental recommendation platform created in collaboration with the Social Science Research Network (SSRN). This system allows for researchers to test recommendation algorithm on SSRN's users and quickly collect feedback on the efficacy of their recommendations. We further describe a test run performed using EigenFactor recommends and compare its performance to SSRN's production recommender.

1 Introduction

The Social Science Research Network (SSRN) is a world wide collaborative of over 272,100 authors and more than 1.7 million users that is devoted to the rapid worldwide dissemination of social science research. Like many publishers of this scale SSRN is contending with an rapid increase in the amount of scholarly literature currently being written. To help their users find relevant articles SSRN has implemented a recommender system based on co-downloads, a collaborative filtering mechanism. Although the co-download system performs well, SSRN sought to further improve recommendations while also driving research in the area of bibliometrics. The authors, in collaboration with SSRN, built an experimental platform to test a novel, citation network based recommendation algorithm: EigenFactor Recommends.

Much of the research into scholarly article recommendation has suffered from poor datasets and difficulty testing with real users. As discussed in depth by Beel et al[1], offline datasets tend to have poor predictive capability. In another work Beel et al[2] surveys 70 different approaches to recommending scholarly articles, finding that only five (7%) of these approaches were validated using online evaluations. These two findings taken together suggest that most researchers don't have access to the tools necessary to effectively test their hypothesis for building better recommenders. The authors believe this is a serious roadblock to improving scholarly article recommendation, and as such set out to build a platform to remedy this problem. The collaboration with SSRN represents our first attempt at addressing this problem.

In addition to improving access to online validation, the authors wanted to validate their algorithm, EigenFactor Recommends. This algorithm is a novel citation based recommender which exploits the hierarchical nature of academic

literature. Some of the earliest work on citation based recommenders for scholarly literature was done in the late 1960s and early 1970s, notably bibliographic coupling[5] and co-citations[12]. The area again received renewed interest in the early 2000s, perhaps spurred on by the success of PageRank[9]. There was substantial activity as the concepts were applied to various networks, resulting in ArticleRank[7], AuthorRank[8] and Y-factor[3]. These methods, however, sought to quantify impact, not provide recommendations. More recent work sought to apply these ideas to the problem of recommendations, with theadvisor[6] being a method very similar to our own.

2 Experimental Platform

In this section we describe the experimental platform we constructed in collaboration with SSRN. This platform is, in the author's opinion, unique in that it allows for easy testing of different recommendation algorithms on live users of a very large publisher of academic literature. To fully understand this platform we will start by describing what a user sees when visiting SSRN, then discuss the *appearance log* and *click log*, and finally detail the *experiment module*.

Figure 2 shows a mock up of an article view on SSRN; this is the page a user is presented when viewing an article, such as http://ssrn.com/abstract=1636719. The current article's title, author list, publication date and abstract are shown in box 1. Box 2 contains various statistics about this article, while box 3 shows the recommendations for this article. The algorithm used to generate these recommendations is selected according to the weights defined in the *experiment module*, and all recommendations listed in box 3 are generated by the same algorithm. Initially only up to three recommendations are shown, but if more are available clicking the *more* button will show any additional recommendations (box 4), up to ten total.

Whenever a user views an article several entries are generated in the *appearance log*, an example of which is shown in figure 2. Each entry in this log file corresponds to a single recommendation being shown on an article view page, including information about when the recommendation was generated, what algorithm generated it, what position this recommendation occupied in the "recommends" box, the article that was viewed (source) and the recommended article (target). A boolean flag is also present denoting if SSRN's fraud system considered this activity fraudulent. Note that since each entry in this log corresponds to a specific recommendation a single article view could generate up to ten entries. Furthermore, recommendations "under the fold" only have an entry in the *appearance log* if a user actually viewed them – that is a user must have clicked the more button.

If a user clicks on a recommendation they are taken to that article's view. This will result in a new set of entries being generated in the *appearance log*, but also in the *click log*, an example of which is shown in figure 3. This log contains the same data as the *appearance log*, but also includes the ID of the user and a flag indicating if the file was downloaded or not. Note that for this experiment

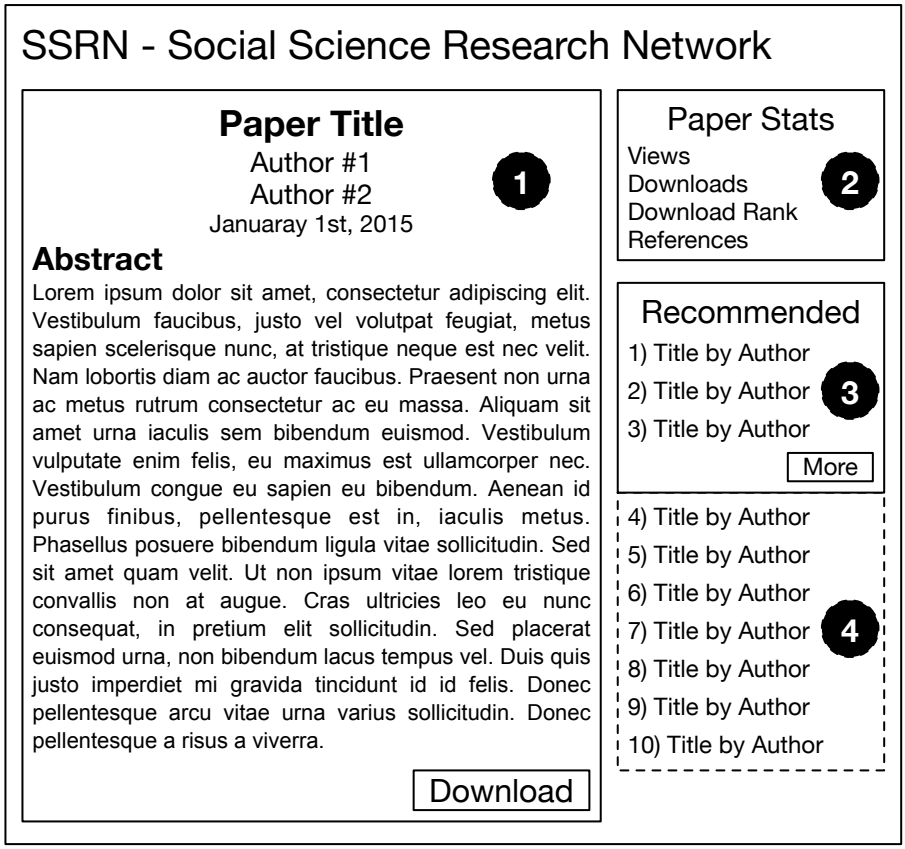

Fig. 1. A mockup of SSRN's article view, including an abstract (1), article statistics (2) and recommendations (3). Initially only up to three recommendations are shown, but if more are available clicking the *more*(4) button will show any additional recommendations, up to ten total.

all user data was stripped from the dataset as it was not used. By correlating the data from the *appearance log* and the *click log* we can reconstruct user actions and through careful analysis determine the efficacy of various recommendation algorithms.

The final part of this platform is the *experiment module*, an administrative tool that allows us to easily add recommendations to SSRN and configure the amount of traffic each algorithm receives. For example, one could direct 85% of traffic to a production algorithm while splitting the remaining 15% of traffic between several experimental algorithms. The module also allows us to download the *appearance log* and *click log*, while providing some very basic analysis of the recommendation algorithm's performance. One current limitation of this system pertains to how recommendations are provided. Recommendations are provided in CSV file with a paper ID followed by a list of recommended paper IDs. This means recommendations are limited to item-to-item recommendations; we are unable to provide customized recommendations on a per-user basis.

```
ID,Timestamp,Algorithm,Position,Source,Target,Fraud
242082,01/27/2015 12:00:00 AM,Control,2,2414016,2422084,No
242079,01/27/2015 12:00:00 AM,Control,1,2414016,2345028,No
242080,01/27/2015 12:00:00 AM,Control,1,2414016,2345028,No
```

Fig. 2. Example of data included in the *appearance log*. Each entry in this log corresponds to a single recommendation being shown on an article view page (Figure 2 box 3). Each entry includes an ID which is unique to this log and not correlated with any other logs, a timestamp for the event, what algorithm generated this recommendation, what position the recommendation occupied in the recommendation list, the article that was viewed (source) and the recommended article (target). A boolean flag is also present denoting if SSRN's fraud system considered this activity fraudulent.

```
ID,Timestamp,User,Algorithm,Position,Source,Target,Fraud,Downloaded
30391,01/27/2015 12:00:51 AM,XXXXXX,Control,1,2212771,1413952,No,No
30392,01/27/2015 12:00:58 AM,XXXXXX,Control,2,2212771,2305863,No,No
30396,01/27/2015 12:13:03 AM,XXXXXX,ef_expert,1,1752911,1641052,No,Yes
```

Fig. 3. Example of data included in the *clicks log*. An entry is generated anytime a user clicks on a recommendations (see Figure 2 box 3). Each entry includes the same data as in the *appearance log*, but also includes the ID of the user performing this action (if logged in) and if the article was ultimately downloaded. Note that ID does not refer to the user but rather the entry. The user identifier is present in the "user" field, and has been anonymized in this example.

3 Methodology

We ran an experiment on SSRN for a one week period, from January 27th 2014 through February 3rd, 2015 (inclusive). During the experiment a user could receive recommendations from one of four different algorithms: control, co-download, EigenFactor expert or EigenFactor serendipity. The experimental platform was configured so that 85% of users were given recommendations from the co-download algorithm, while the remaining algorithms were each given 5% of traffic. The co-download algorithm is the default for SSRN when not running experiments, so it was given a larger portion of traffic to mitigate the risk of giving bad recommendations to users.

The experiment recorded a total of 1416404 recommendations, of which 239974 (16.94%) were considered fraudulent by SSRN's fraud detection system. These events are excluded from subsequent analysis, leaving 1176430 recommendations. A detailed breakdown of these numbers is available in table 1.

Fraud	Appearances	Clicks	Downloads
False	1176430	44002	1989
True	239974	3817	140
Total	1416404	47819	2129

Table 1. Fraudulent and Genuine Events

3.1 Control

The control algorithm consists of recommendations chosen at random from the SSRN corpus. As table 2 shows users view the abstract of these recommendations (clicks) at a slightly lower rate compared to either EigenFactor algorithm (0.65% vs 0.86-0.95%), and they download at a much lower rate of 1.72% vs 4-6% for real recommendations. It is possible that users find the title's of the recommended papers interesting, but after reading the abstract realize they don't relate to the original paper. This behavior is inline with our expectations for a random control.

3.2 Co-Download

The current production recommender used by SSRN is a collaborative filtering algorithm based on co-downloads. The algorithm works by tracking user downloads. To generate a recommendation for a paper the algorithm selects all users who have downloaded the source paper, then gathers a list of all the papers those users have downloaded. These papers are then counted and sorted by count, descending. The papers that have the most downloads are the top recommendations. Note that this algorithm is undirected, it doesn't know if paper 1 was downloaded then paper 2, only that 1 and 2 were both downloaded.

The current implementation also limits the co-downloads to papers downloaded within the last two years, which helps to provide more recent recommendations.

```
#!/usr/bin/env python
from collections import Counter
users = #set of users who downloaded paper i
co_dl = Counter()
for user in users:
    for paper in user.get_downloads()
        co_dl[paper] += 1
return co_dl.most_common(3)
```

3.3 EigenFactor Recommends

EigenFactor (EF) recommends is a variant of the EigenFactor algorithm combined with the MapEquation algorithm [15, 10]. The Eigenfactor Metrics ranks scholarly journals, authors, papers and institutions [13, 14]. These methods are based on eigenvector centrality methods, first developed by sociologist Phillip Bonacich in 1972 [4]. Eigenvector centrality is used for a wide variety of network analysis tasks, including (and perhaps most famously) Brin and Page's PageRank [9].

EigenFactor recommends is different than the co-downloads approach in that it uses the citation network, rather than usage data[3]. The recommendations are based on the hierarchical structure of the SSRN corpus. The multi-level structure is extracted using a variant of InfoMap [11]. The article-level Eigenfactor is then used to identify key papers within specific fields and sub-fields [15].

Two variants of EigenFactor recommends were tested, *expert* and *serendipity*. Expert works be selecting the highest scoring articles in the most local cluster (i.e., the endleafs of the hierarchical tree). Serendipity also operates on the most local cluster, but instead selects a paper at random within this local clusters. Typically, these endleaf nodes consist of hundreds of papers.

To generate both variants of EigenFactor recommends, we first constructed the full citation network from SSRN. This network included 2,414,097 individual citations over 156,570 papers[4]. From this network, we produced 218,825 recommendations for both the expert and serendipity variants. These recommendations were then uploaded into the *experiment module* and made available to users on SSRN.

[3] The method is also different, which is explained in the West et al. paper [15].

[4] SSRN is a pre-print and post-print archive. Therefore, multiple versions of a paper are listed on SSRN. We only count one instance of a paper. If there are multiple versions of the paper, we track the group of papers associated with one piece of work.

4 Results

This experiment was the first use of the SSRN recommendation system. As such, it was not only an experiment on collaborative filtering algorithms vs citation based algorithms, but also a test run of the system itself. Initial analysis uncovered several issues with both data collection and experimental design.

4.1 Data Analysis

Although the metrics below are more thoroughly described in the Experimental Platform section, a brief summary is provided. There are three different metrics we capture: *appearances*, *clicks* and *downloads*. *Appearances* refers to a recommendation being shown to the user when they visit a page on SSRN. In figure 2 this is the box titled "Recommended" on the right hand side. Each recommendation counts as an appearance for that algorithm, so if three recommendations are shown that would count as three appearances for the algorithm that generated those recommendations. Recommendations for a given page view are all generated from the same algorithm, which is selected based on the weights provided in the *experiment module*. *Clicks* tracks when a user clicks on a recommendation. Doing so will take you to the abstract of the recommended paper. The final metric we track is *downloads*, which is a measure of when a document is downloaded from a recommendation. Only downloads that are due to a recommendation are recorded in this metric. For each of these metrics counts represents the count of events while % is the percentage of that event type a specific algorithm accounts for.

We also present three useful statistics for each algorithm: C/A, D/C and D/A. C/A is the number of clicks divided by the number of appearances. This is effectively a click-through rate, and also could be considered the probability that a recommendation will be clicked. D/C is downloads over clicks, which is the percentage of clicks that lead to a download. The final value is D/A, downloads over appearances, which can be thought of as the probability that a recommendation will be downloaded.

Algorithm	Appearances		Clicks		Downloads		C/A	D/C	D/A
	Count	%	Count	%	Count	%			
Control	26805	2.28%	174	0.40%	3	0.15%	0.65%	1.72%	0.01%
Co-Download	1102847	93.75%	43405	98.64%	1960	98.54%	3.94%	4.52%	0.18%
EF Expert	21837	1.86%	208	0.47%	12	0.60%	0.95%	5.77%	0.05%
EF Serendipity	24941	2.12%	215	0.49%	14	0.70%	0.86%	6.51%	0.06%

Table 2. Data collected from January 27th, 2015 through February 3rd, 2015 inclusive.

4.2 Position Problems

One issue we encountered was the number of recommendations shown for each algorithm. Due to an oversight when providing data to the experimentation platform, EigenFactor and the control algorithm only displayed a single recommendation while co-downloaded displayed several, potentially up to 10. This is apparent when one looks at the data that was captured for all positions and compares it to data only at position one (see table 3). Part of this is an overcounting problem, which can be rectified, but a more serious issue arises that is intractable: since co-download gets to show three different recommendations it has a higher chance of a user clicking one (or more) of its recommendations. This gives co-download an advantage of 0.77% for C/A as shown by table 3, though it does result in a worse download rate (D/C).

Position	Appearances	Clicks	Downloads	C/A	D/C	D/A
First	412188	13067	716	3.17%	5.48%	0.17%
All	1102847	43405	1960	3.94%	4.55%	0.18%

Table 3. A comparison of the Co-Download at all recommendation positions vs only the first

4.3 Recommendation Position

As table 4 shows the position of a recommendation has some impact on the click through and download rates. For recommendations this is not unexpected: if recommendations are presented in order of strength and the algorithm is effective one would expect higher ranked recommendations to be selected more often. Furthermore, recommendations below the "cutoff" on the article view page (anything with position greater than three) are clicked and downloaded at a much lower rate, as we would also expect. There is, however, an interesting question of primacy effect: does a recommendation being listed first increase the rate at which it is downloaded, independent of the recommendation quality? Table 4 provides some evidence that this primacy effect is occurring. The clicks % shows a large discrepancy in the number of clicks given to item one vs two and three. Furthermore, positions two and three show a substantially higher download rate, which implies more interest in items in position two and three when they are clicked.

4.4 Recommended Papers Comparison

Table 5 shows a list of the most clicked recommendations by algorithm. Although no in-depth analysis about these titles has been performed, it is clear that finance related articles are very heavily represented.

Position	Appearances		Clicks		Downloads		C/A	D/C	D/A
	Count	%	Count	%	Count	%			
1	412188	37.37	13067	30.10	716	36.53	3.17	5.48	0.17
2	357476	32.41	5674	13.07	463	23.62	1.59	8.16	0.13
3	327801	29.72	5245	12.08	417	21.28	1.60	7.95	0.13

Table 4. Co-Download Recommendations by Position

Algorithm	Paper Title	Count
Co-Download	An Intermarket Approach to Tactical Risk Rotation: Using the Signaling Power of Treasuries to Generate Alpha and Enhance Asset Allocation	451
	Factor Investing	262
	A Five-Factor Asset Pricing Model	249
EF Serendipity	An Economic Evaluation of Empirical Exchange Rate Models	3
	Strategic Hedge Fund of Fund Portfolio Construction	3
	Hyperinflation - It's More than Just a Monetary Phenomenon	3
EF Expert	Games and Information, Third Edition, Preface	6
	Boards of Directors as an Endogenously Determined Institution: A Survey of the Economic Literature	5
	The Tactical and Strategic Value of Commodity Futures	5

Table 5. A comparison of the top clicked recommendations generated by Co-Download and EigenFactor

5 Conclusion

During this experiment several issues with experimental design and the data collection platform were discovered. Although unfortunate, this is unsurprising. Though data quality issues mean few strong conclusions can be drawn, we did see evidence that the co-download algorithm results in a significantly higher click through rate (3.94%) over either of the EigenFactor algorithms (0.95% and 0.86%). It is, however, unclear why co-download performed nearly three times better, and this will be an area for future investigation.

However, once a user is viewing an abstract co-download has a slightly lower download rate compared to the EigenFactor algorithm (4.55% vs 5.77%). It is very interesting that the EigenFactor serendipity algorithm has the highest download rate, though given the small number of downloads overall this could just be noise in the dataset. One could view this as EigenFactor having higher deviation in the recommendations it generates. It may recommend articles with interesting titles less often, but when it does that article is downloaded at a higher rate.

Finally, evidence of a primacy effect was found in the co-download data set. The next experiment will seek to validate this with additional data in the control set.

We are already planning a larger scale experiment to validate these findings, running the algorithms for at least a month. We will also be fixing all issues

that were discovered in this process and more aggressively validating the data collection process.

References

1. Joeran Beel, Marcel Genzmehr, Stefan Langer, Andreas Nürnberger, and Bela Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation - RepSys '13*, pages 7–14, 2013.
2. Joeran Beel and Stefan Langer. Research Paper Recommender System Evaluation : A Quantitative Literature Survey. (April), 2013.
3. Johan Bollen, Marko A. Rodriquez, and Herbert Van de Sompel. Journal status. *Scientometrics*, 69(3):669–687, December 2006.
4. Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1):113–120, 1972.
5. M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, 1963.
6. O Küçüktunç, Erik Saule, Kamer Kaya, and ÜV Çatalyürek. Direction awareness in citation recommendation. i:3–8, 2012.
7. Jiang Li and Peter Willett. ArticleRank: a PageRank?based alternative to numbers of citations for analysing citation networks. *Aslib Proceedings*, 61(6):605–618, November 2009.
8. Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. Co-authorship networks in the digital library research community. *Inf. Process. Manage.*, 41(6):1462–1480, December 2005.
9. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.
10. Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.
11. Martin Rosvall and Carl T Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.
12. Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973.
13. J.D. West, T.C. Bergstrom, and C.T. Bergstrom. The eigenfactor metrics: A network approach to assessing scholarly journals. *College and Research Libraries*, 71(3):236–244, 2010.
14. Jevin D. West, Michael C. Jensen, Ralph J. Dandrea, Gregory J. Gordon, and Carl T. Bergstrom. Author-level Eigenfactor metrics: Evaluating the influence of authors, institutions, and countries within the social science research network community. *Journal of the American Society for Information Science and Technology*, 64(4):787–801, April 2013.
15. Jevin D West, Ian Wesley-Smith, Martin Rosvall, and Carl Bergstrom. A recommendation system based on hierarchical clustering of an article-level citation network. *in prep*, 2015.