Real-World Recommender Systems for Academia: The Pain and Gain in Building, Operating, and Researching them

Joeran Beel[1,2] and Siddharth Dinesh[3]

[1]Trinity College Dublin, Department of Computer Science, ADAPT Centre, Ireland
joeran.beel@adaptcentre.ie
[2]National Institute of Informatics, Digital Content and Media Sciences Research Division, Tokyo, Japan
beel@nii.ac.jp
[3]Birla Institute of Technology and Science, Pilani, India
f2012519@goa.bits-pilani.ac.in

Abstract. Research on recommender systems is a challenging task, as is building and operating such systems. Major challenges include non-reproducible research results, dealing with noisy data, and answering many questions such as how many recommendations to display, how often, and, of course, how to generate recommendations most effectively. In the past six years, we built three research-article recommender systems for digital libraries and reference managers, and conducted research on these systems. In this paper, we share some experiences we made during that time. Among others, we discuss the required skills to build recommender systems, and why the literature provides little help in identifying promising recommendation approaches. We explain the challenge in creating a randomization engine to run A/B tests, and how low data quality impacts the calculation of bibliometrics. We further discuss why several of our experiments delivered disappointing results, and provide statistics on how many researchers showed interest in our recommendation dataset.

Keywords: recommender system, digital library, reference management

1 Introduction

Recommender systems is a fascinating topic for both researchers and industry. Researchers find in recommender systems a topic that is relevant for many disciplines: machine learning, text mining, artificial intelligence, network analysis, bibliometrics, databases, cloud computing, scalability, data science, visualization, human computer interaction, and many more. That makes research in recommender systems interesting and creates many opportunities to cooperate with other researchers. For industry, recommender systems offer an opportunity to provide additional value to customers by helping them finding relevant items. Recommender systems may also provide a justification to store user-related data, which may be used for generating additional revenue.

A long version of this article is available online https://arxiv.org/abs/1704.00156

In addition, recommender systems may even become a major part of the business model, as companies such as Netflix, Spotify, and Amazon demonstrate.

Over the past six years, we built, operated, and researched three research-article recommender systems in the context of digital libraries and reference management. The work was often rewarding, but also challenging and occasionally even painful. We share some of our experiences in this article. This article is not a research article but a mixture of a project report, lessons learned, text-book, and summary of our previous research, enriched with some novel research results.[1]

The primary audience of this article are researchers and developers who think about developing a real-world recommender system for research purposes, or for integrating the recommender system on-top of a real product. While our focus lies on recommender system in the context of digital libraries and reference managers, researchers and developers from other disciplines may also find some relevant information in our article. As this paper is an invited paper for the "5th International Workshop on Bibliometric-enhanced Information Retrieval", we particularly discuss our work in the context of bibliometric-enhanced recommender systems and information retrieval.

2 Our Recommender Systems

2.1 SciPlore MindMapping

In 2009, we introduced SciPlore MindMapping (Beel, Gipp, & Mueller, 2009). The software enabled researchers to manage their references, annotations, and PDF files in mind-maps. In these mind-maps, users could create categories that reflect their research interests or that represent sections of a new manuscript. Users could then sort their PDF files, annotations, and references in these categories. In 2011, we integrated a recommender system in SciPlore MindMapping (Beel, 2011). The system was rather simple. Whenever users selected a node in a mind-map, the text of the node was sent as search query to Google Scholar, and the first three results of Google Scholar were shown as recommendations.

2.2 Docear

Docear[2] is the successor of SciPlore MindMapping, pursuing the same goal, i.e. enabling researchers to manage their references, annotations, and PDF files in mind maps (Beel, Gipp, Langer, & Genzmehr, 2011).[3] In contrast to SciPlore MindMapping, Docear has more features, a neater interface, and a more sophisticated recommender system (Beel, Langer, Genzmehr, & Nürnberger, 2013; Beel, Langer, Gipp, & Nürnberger, 2014). The recommender system features a comprehensive user modeling

[1] The data and scripts we used for the novel analyses is available at http://data.mr-dlib.org

[2] http://docear.org

[3] Currently, Docear's recommender system is offline because we focus on the development of Mr. DLib.

engine, and uses Apache Lucene/Solr for content-based filtering as well as some proprietary implementations of other recommendation approaches. Docear is a desktop software but transfers users' mind maps to Docear's servers. On the servers, Docear's recommender system calculates user specific recommendations. Recommendations are shown every couple of days to users, and users may also request recommendations explicitly. Docear has a corpus of around 2 million full-text documents freely available on the Web.

2.3 Mr. DLib

Our latest recommender system is Mr. DLib[4], a machine-readable digital library that provides recommendations as-a-service to third parties (Beel, Gipp, & Aizawa, 2017). This means, third parties such as such as digital libraries and reference managers can easily integrate a recommender system into their product via Mr. DLib. The recommender system is hosted and operated by Mr. DLib and partners only need to request recommendations for a specific item via a REST API (Figure 1).

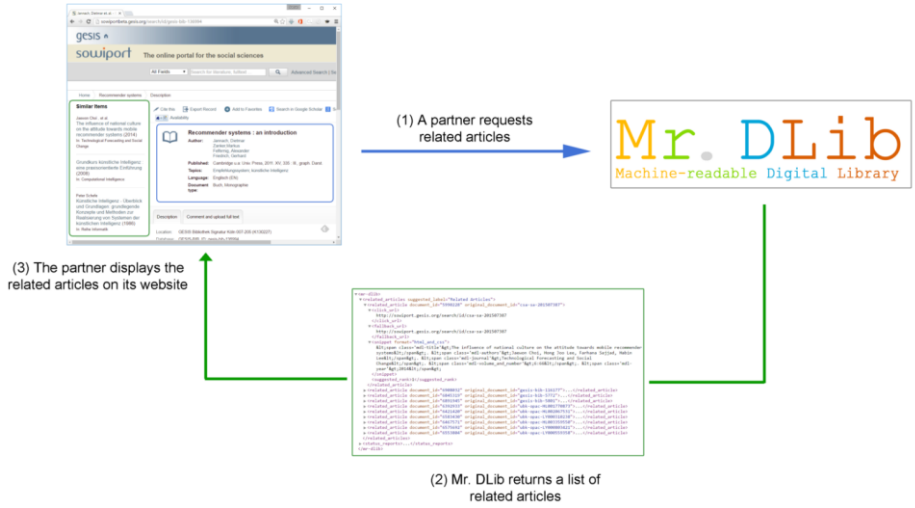


Figure 1: Illustration of Mr. DLib's recommendation process

Our first pilot partner is Sowiport (Hienert, Sawitzki, & Mayr, 2015). Sowiport is operated by 'GESIS – Leibniz-Institute for the Social Sciences', which is the largest infrastructure institution for the Social Sciences in Germany. Sowiport contains about 9.6 million literature references and 50,000 research projects from 18 different databases, mostly relating to the social and political sciences. Literature references usually cover keywords, classifications, author(s) and journal or conference information and if available: citations, references and links to full texts. We additionally integrated Mr. DLib into the reference manager JabRef (Feyer, Siebert, Gipp, Aizawa, & Beel, 2017), and currently discuss the integration with two more organizations.

[4] http://mr-dlib.org

Recommendations in Mr. DLib are primarily calculated with Apache Lucene / Solr, i.e. based on the metadata of the documents in Sowiports' corpus. We further experiment with stereotype and most-popular recommendations (Beel, Dinesh, Mayr, Carevic, & Raghvendra, 2017), as well as with enhanced content-based filtering based on key-phrase extraction and re-ranking with Mendeley readership statistics (Siebert, Dinesh, & Feyer, 2017). In the future, we plan to add further recommendation approaches; especially link/citation-based approaches seem promising (Schwarzer et al., 2016).

3 Recommender-System Development

3.1 Required skills

To build and operate a recommender system, more than just knowledge about recommender systems and related disciplines such as text mining and machine learning is required. Server administration, databases, web technologies, data formats (e.g. XML or JSON), and data processing (crawling, parsing, transforming) are probably the most important ones, but also knowledge about software engineering in general (e.g. agile development, unit testing, etc.), scalability, data privacy laws, and project management is helpful. Niche knowledge that does not directly relate to recommender systems may also be beneficial and lead to novel recommendation approaches. For instance, knowledge in bibliometrics could help to develop novel re-ranking algorithms for content-based research-paper recommender systems (Cabanac, 2011). In such systems, a list of recommendation candidates, would be re-ranked based on e.g. how many citations the candidate papers have, or based on the h-index of the candidate papers' authors, which, however, then might strengthen the Mathew effect (Cabanac & Preuss, 2013). We have experimented with such algorithms but were only partly successful – maybe because we lack the expert-knowledge in bibliometrics (Siebert et al., 2017).

3.2 (No) help from the literature

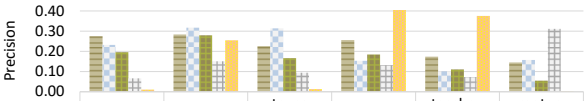
There are hundreds of research articles about recommender systems in Academia (Beel, Gipp, Langer, & Breitinger, 2016), and probably thousands of articles about recommender systems in other domains. One might expect that such a large corpus of literature would provide advice on how to build a recommender system, and which recommendation approaches to use. Unfortunately, this is not the case, at least in the domain of research-paper recommender systems. The reasons are manifold. Many recommendation approaches were not evaluated at all, compared against too simple baselines, evaluated with too few users, or evaluated with highly tweaked datasets (Beel, Gipp, et al., 2016). Consequently, the meaningfulness of the results is questionable.

Even if evaluations were sound, recommendation effectiveness may vary a lot. In other words, only because a recommendation approach performed well in one scenario, does not mean it will perform well in another scenario. For instance, the TechLens team proposed and evaluated several content-based filtering (CBF) and collaborative filtering (CF) approaches for research-paper recommendations. In one experiment, CF and

CBF performed similarly well (McNee et al., 2002). In other experiments, CBF outperformed CF (Dong, Tokarchuk, & Ma, 2009; McNee et al., 2002; Torres, McNee, Abel, Konstan, & Riedl, 2004), and in some more experiments CF outperformed CBF (Ekstrand et al., 2010; McNee, Kapoor, & Konstan, 2006; Torres et al., 2004). In other words: it remains speculative how CBF and CF would perform in a scenario that differs from one of those used in the existing evaluations.

Some authors used bibliometrics to enhance recommender systems in digital libraries. Popular metrics were PageRank (Bethard & Jurafsky, 2010), HITS (He, Pei, Kifer, Mitra, & Giles, 2010), Katz (He et al., 2010), citation counts (Bethard & Jurafsky, 2010; He et al., 2010; Rokach, Mitra, Kataria, Huang, & Giles, 2013), venues' citation counts (Bethard & Jurafsky, 2010; Rokach et al., 2013), citation counts of the authors' affiliations (Rokach et al., 2013), authors' citation count (Bethard & Jurafsky, 2010; Rokach et al., 2013), h-index (Bethard & Jurafsky, 2010), recency of articles (Bethard & Jurafsky, 2010), title length (Rokach et al., 2013), number of co-authors (Rokach et al., 2013), number of affiliations (Rokach et al., 2013), and venue type (Rokach et al., 2013). Again, results are not always coherent. For instance, Bethard and Jurafsky (2010) reported that using citation counts in the recommendation process *strongly* increased the effectiveness of their recommendation approach, while He et al. (2010) reported that citation counts *slightly* increased the effectiveness of their approach.

Our own research confirms that recommendation approaches perform very differently in different scenarios. We recently applied five recommendation approaches on six news websites (Beel, Breitinger, Langer, Lommatzsch, & Gipp, 2016). The results showed that recommendation approaches performing well on one news website performed poorly on others (Figure 2). For instance, the most-popular approach performed worst on tagesspiegel.de but best on cio.de.



	ksta.de	sport1.de	tages spiegel.de	cio.de	tecchan nel.de	motor-talk.de
User-based CF	0.28	0.28	0.23	0.26	0.17	0.15
Mst. ppl. sequence	0.23	0.32	0.31	0.15	0.10	0.16
Item-based CF	0.20	0.28	0.17	0.18	0.11	0.05
Content-based	0.07	0.15	0.09	0.13	0.07	0.31
Most popular	0.01	0.25	0.01	0.56	0.38	0.00

Website

Figure 2: Recommendation Effectiveness on Different News Websites

There are several potential reasons for the unpredictability. In some cases, different evaluation methods were used. In other cases variations in algorithms or user populations might have had an impact (Beel, Breitinger, et al., 2016; Beel, Langer, Nürnberger, & Genzmehr, 2013; Langer & Beel, 2014). However, it seems that, for instance, the operator of a news website cannot estimate how effective the most-popular recommendation approach would be until the operator has implemented and evaluated the approach on that particular website. Therefore, our advice is to read a recommender-system text book to get a general idea of recommender systems (Jannach, 2014;

Konstan & Ekstrand, 2015; Ricci, Rokach, & Shapira, 2015). Then, choose a few recommendation frameworks and try to find the best recommendation approach for one's recommender system.

3.3 The randomization engine

To build an effective recommender system, A/B testing is essential. This means, the recommender system needs a pool of recommendation algorithms to choose from, and a logging mechanism to record, which algorithm was used and what the user feedback was (e.g. how many recommendations were clicked or downloaded). Unfortunately, in most situations, a simple A/B test with only two alternative algorithms will not be enough. In Mr. DLib, we implemented a "randomization engine" that first picks from several recommendation classes randomly, and then varies the parameters of the recommendation algorithms. The recommendation classes are content-based filtering (90% chance), stereotyping (4.9% chance), most-popular recommendations (4.9% chance), and random recommendations as baseline (0.2% chance). For each of the recommendation classes, the randomization engine chooses some parameters randomly. For instance, when content-based filtering is chosen, the system randomly selects whether to use "normal" terms, or key-phrases[5]. When key-phrases are chosen, the system randomly selects if key-phrases from the abstract, title, or title *and* abstract are used. Then the system randomly selects if unigrams, bigrams, or trigrams are used. Then the system randomly selects if one, two, three, … or twenty key-phrases are used to calculate document similarity.

Once the recommendations are calculated, the randomization engine chooses randomly if the recommendation candidates should be re-ranked based on bibliometrics. Re-ranking means that from the top *x* recommendations those ten documents are eventually recommended that have, for instance, the highest bibliometric score. Again, there are many parameters that are randomly chosen by the randomization engine. The engine selects the bibliometric (plain readership count, readership count normalized by age of the document, readership count normalized by the number of authors, etc.), the number of recommendation candidates to re-rank (10 to 100), and how the bibliometric and text relevance scores should be combined (bibliometric only, multiplication of scores, and some more variations). While we currently only work with readership data from Mendeley, we plan to obtain additional data from sources such as Google Scholar, Scopus, or Microsoft Academic (Sinha et al., 2015).

Developing such a randomization engine is a non-trivial task and we are not perfectly satisfied with our solution. It occurs too often that a fallback algorithm must be used because the randomly assembled algorithm cannot be applied because, for instance, a document does not have 20 key-phrases in its title.

[5] Keyphrases are the most meaningful phrases describing a document. Keyphrases are extracted with stemming, part-of-speech tagging and other mechanisms that we describe in an upcoming paper.

3.4 Data quality

Crucial for content-based recommendations is the quality of metadata. When using publicly available datasets, the quality may often be good because datasets were designed and maybe even curated to be published. In the real-world, however, data quality is often low. For instance, some of the most productive "authors" in Mr. DLib's database are "et al.", "and others", and "AnoN". Obviously, these are no real authors. When this data is used e.g. to recommend papers of co-authors, or to re-rank recommendations based on h-index, the resulting recommendations will be of suboptimal quality.

Cleaning data from third parties is a labor-intensive and usually boring task (it is much easier to motivate a colleague to implement a novel recommendation algorithm than convincing a colleague to spend some weeks cleaning data in a database). Data cleaning becomes particularly challenging, when the data comes from a third party and the data are updated occasionally by the partner. In that case, one would need a process to decide how to deal with the updated data and judge if the new data from the partner is better than the manually changed data in our system. From our experience in other projects, we know that manually cleaning data usually causes many problems. Therefore, we decided to do no manual data cleaning in Mr. DLib, and only apply a few heuristics such as ignoring "et al." when calculating bibliometrics.

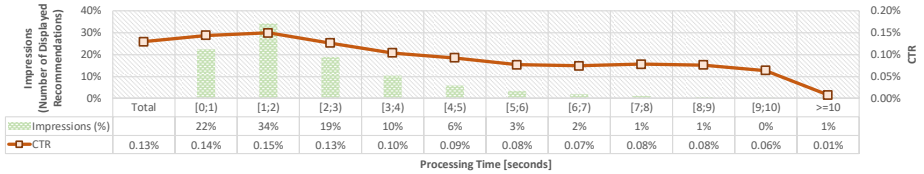


Figure 3: Click-Through Rate (CTR) based on Processing Time

4 Recommender-System Operation

4.1 Ensure timely delivery of recommendations

On the Internet, users tend to be impatient: the longer they wait for content, the less satisfied they become (Guse, Schuck, Hohlfeld, Raake, & Möller, 2015; Kim, Xiong, & Liang, 2017; Selvidge, Chaparro, & Bender, 2002). This holds true for recommender systems, too. We observed that the longer users had to wait for recommendations, the less likely they were to click a recommendation. Figure 3 shows that processing time[6] for most recommendations (34%) was between 1 and 2 seconds. These recommendations also had the highest click-through rate (CTR) of 0.15% on average. In contrast, recommendations that needed 7 to 8 seconds for calculation had a CTR of 0.08%.

[6] "Processing Time" is the time from receiving a request until delivering the response on side of Mr. DLib. It may be that a user who must wait too long, leaves the web page and does not even see the recommendations.

The re-ranking of recommendations based on bibliometrics requires rather a lot of calculation (primarily due to the randomization engine and because we store many statistics when calculating the bibliometrics). Consequently, when evaluating the effectiveness of bibliometric re-ranking, one need to additionally consider if the additional effectiveness if worth the additional time users need to wait.

4.2 The need to deliver only good recommendations

In the recommender-system community, it is often reported that a recommender system should try to avoid making 'bad' recommendations as this hurts the users' trust. In other words, it is better to recommend 5 good and 5 mediocre items than recommending 9 excellent but 1 bad item. To avoid bad recommendations, some relevance score is needed that indicates how good a recommendation is. Ideally, only recommendations above a certain threshold would then be recommended. However, at least Lucene has no such threshold that would allow a prediction of how relevant a recommendation is[7]. The Lucene text relevance score only allows to rank recommendations for one given query and compare the relevance of the results returned for that one query. In a recent analysis, we found that Lucene relevance scores and CTR correlate, but still it is not possible to avoid "bad" recommendations (Langer & Beel, 2017).

Even if Lucene had an "absolute" text relevance score, this score would only be able to prevent bad recommendations to some extent. We see a high potential in bibliometrics to support recommender systems in not recommending bad items.

4.3 Number of recommendations

Another question that may seem simple to answer is how many recommendations to display? One? Two? ...Ten? We experimented in Mr. DLib with varying numbers of recommendations between 1 and 15 (Beierle, Aizawa, & Beel, 2017). We observed that the more recommendations were displayed, the lower click-through rate became (Figure 4). From these results, one cannot conclude how many recommendations to display, and more research is necessary.

5 Recommender-System Research

5.1 No tweaking of data

In offline evaluations, it is common to tweak datasets. For instance, Caragea et al. removed papers with fewer than ten and more than 100 citations from the evaluation corpus, as well as papers citing fewer than 15 and more than 50 papers (Caragea, Silvescu, Mitra, & Giles, 2013). From originally 1.3 million papers in the corpus, around 16,000 remained (1.2%). Pennock et al. removed documents with fewer than 15 implicit ratings from the corpus (Pennock, Horvitz, Lawrence, & Giles, 2000). From

[7] Using the standard More-Like-This or search function https://wiki.apache.org/lucene-java/ScoresAsPercentages

originally 270,000 documents, 1,575 remained (0.58%). Such tweaking and pruning of datasets may be convenient for the research and lead potentially to high precision. However, applying a recommendation approach to only 0.58% of the documents in a corpus, will lead to a very poor recall, i.e. the results that have little relevance for running a real-world recommender system. In other words, in a real-world recommender system such a tweaking would be difficult, unless one would accept that recommendations can be delivered only for a fraction of documents in the corpus.

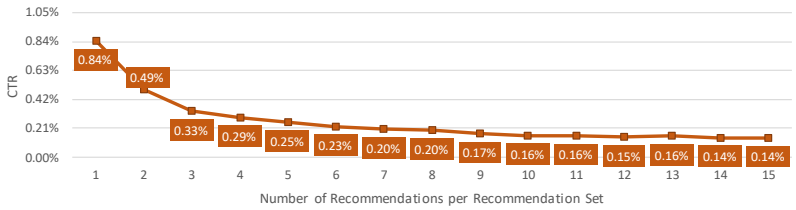


Figure 4: Number of displayed recommendations and CTR (Beierle et al., 2017)

5.2 Accept failure

When we reviewed over 200 research articles about recommender systems, every article that introduced a new recommendation approach reported to outperform the state-of-the art (Beel, Gipp, et al., 2016). We were not that lucky.

We re-ranked content-based recommendation with readership data from Mendeley (Siebert et al., 2017), but results were not as good as expected. We used a key-phrase-extraction approach to improve our content-based filtering approach, and experimented with a variation of parameters: we varied the text-fields from which key-phrases were extracted, we varied the number of key-phrases being used, and we varied the type of key-phrases (unigrams, bigrams, trigrams, or a mix). None of these variations performed better than an out-of-the-box Apache Lucene baseline.[8] We experimented with stereotype and most-popular recommendations, two approaches that are effective in domains such as movie recommendations and hotel search (Beel, Dinesh, et al., 2017). Again, the approaches performed not better than Apache Lucene.

There are a few potential reasons why our experiments delivered disappointing results (besides the possibility that the recommendation approaches are just not effective). For instance, in case of the bibliometric re-ranking, we assume that click-through rate might be not appropriate to measure the re-ranking effectiveness. However, even while there might be plausible reasons for the results, the fact remains that many of our experiments delivered rather disappointing results, but this is probably rather the rule than the exception in a real-world scenario.

[8] The analysis is still in process, and the final results will be published soon.

5.3 Other researchers' interest in datasets

One advantage of working on a real-world recommender system is the possibility to release datasets, which then can be used (and cited) by other researchers. Some datasets such as MovieLens are very popular in the recommender system community. The MovieLens dataset was downloaded 140,000 times in 2014 (Harper & Konstan, 2016), and Google Scholar lists 10,600 papers that mention the MovieLens dataset[9].

Not all datasets become that popular. In 2014, we released a dataset of Docear's recommender system (Beel et al., 2014). The datasets contained metadata of 9.4 million articles, including 1.8 million articles publicly available on the Web; the articles' citation network; anonymized information on 8,059 Docear users; information about the users' 52,202 mind-maps and personal libraries; and details on the 308,146 recommendations that the recommender system delivered. In the 2.5 years since publication, 31 researchers requested to download the dataset. To the best of our knowledge, none of these researchers has eventually analyzed the dataset and published their results.

6 Summary & Conclusion

Building and operating real-world recommender systems is a time-consuming and challenging task, as is research on such systems.

To develop recommender systems, knowledge from various disciplines is required such as machine learning, server administration, databases, web technologies, and data formats. When building our own recommender systems, we could find no guidance in the literature. Most published research results had questionable evaluations and even if evaluations were sound, recommender systems seem to perform just too differently in different scenarios. Consequently, we used Apache Lucene as recommendation framework and began from scratch. Evaluating different recommendation approaches requires a randomization engine that selects and assembles recommendation algorithms automatically. In addition, a detailed logging mechanism is required that records which algorithm created which recommendation and how the user reaction was to the different algorithms. Another challenge lies in dealing with sub-optimal data quality from partners. Due to time constraints, we decided to not manually improve the data but just work with what we got.

Running a recommender system requires fast generation and delivery of recommendations. Otherwise, users become dissatisfied and click-through rates decrease. Although it is widely known that recommender systems should avoid making bad recommendations, this is not easily accomplished in practice, at least if Lucene is used as recommendation framework. Lucene has no absolute relevance score and hence no mechanism to recommend only items above a certain relevance threshold.

Research on real-world recommender systems is probably more frustrating than research in a lab-environment, mostly because data cannot be tweaked that easily. Consequently, we had to accept that many experiments with novel recommendation approaches failed. Similarly, while some recommendation datasets such as MovieLens

[9] https://scholar.google.com/scholar?hl=en&q="movielens"

are widely used in the community, we could not yet manage to establish our datasets as interesting source for other researchers.

Despite all these challenges, research on real-world recommender systems is a rewarding and worthwhile effort. We feel that working on real systems provides much more relevant research results. In addition, offering a real-world open-source project attracts many volunteers, students, and project partners. This, in turn, enabled us to conduct research in many areas, and be quite productive in terms of publication output.

7 Acknowledgements

This work was supported by a fellowship within the FITweltweit programme of the German Academic Exchange Service (DAAD). In addition, this publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/RC/2106. We are also grateful for the support received by Sophie Siebert, Stefan Feyer, Felix Beierle, Sara Mahmoud, Gabor Neusch, and Mr. DLib's partners.

References

Beel, J. (2011). SciPlore MindMapping now provides literature recommendations (Beta 15). *http://sciplore.org/2011/sciplore-mindmapping-now-provides-literature-recommendations-beta-15/*.

Beel, J., Breitinger, C., Langer, S., Lommatzsch, A., & Gipp, B. (2016). Towards Reproducibility in Recommender-Systems Research. *User Modeling and User-Adapted Interaction (UMUAI)*, *26*(1), 69–101. doi:10.1007/s11257-016-9174-x

Beel, J., Dinesh, S., Mayr, P., Carevic, Z., & Raghvendra, J. (2017). Stereotype and Most-Popular Recommendations in the Digital Library Sowiport. *Proceedings of the 15th International Symposium of Information Science (ISI)*.

Beel, J., Gipp, B., & Aizawa, A. (2017). Mr. DLib: Recommendations-as-a-Service (RaaS) for Academia. *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research Paper Recommender Systems: A Literature Survey. *International Journal on Digital Libraries*, (4), 305–338. doi:10.1007/s00799-015-0156-0

Beel, J., Gipp, B., Langer, S., & Genzmehr, M. (2011). Docear: An Academic Literature Suite for Searching, Organizing and Creating Academic Literature. *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, JCDL '11 (pp. 465–466). doi:10.1145/1998076.1998188

Beel, J., Gipp, B., & Mueller, C. (2009). SciPlore MindMapping' - A Tool for Creating Mind Maps Combined with PDF and Reference Management. *D-Lib Magazine*, *15*(11). doi:10.1045/november2009-inbrief

Beel, J., Langer, S., Genzmehr, M., & Nürnberger, A. (2013). Introducing Docear's Research Paper Recommender System. *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'13)* (pp. 459–460). ACM. doi:10.1145/2467696.2467786

Beel, J., Langer, S., Gipp, B., & Nürnberger, A. (2014). The Architecture and Datasets of Docear's Research Paper Recommender System. *D-Lib Magazine*, *20*(11/12). doi:10.1045/november14-beel

Beel, J., Langer, S., Nürnberger, A., & Genzmehr, M. (2013). The Impact of Demographics (Age and Gender) and Other User Characteristics on Evaluating Recommender Systems. In *Proceedings of the 17th International Conference on Theory and Practice of Digital Libraries (TPDL)* (pp. 400–404). Springer.

Beierle, F., Aizawa, A., & Beel, J. (2017). Exploring Choice Overload in Related-Article Recommendations in Digital Libraries. *5th International Workshop on Bibliometric-enhanced Information Retrieval (BIR) at the 39th European Conference on Information Retrieval (ECIR)*.

Bethard, S., & Jurafsky, D. (2010). Who should I cite: learning literature search models from citation behavior. *Proceedings of the 19th ACM international conference on Information and knowledge management* (pp. 609–618). ACM.

Cabanac, G. (2011). Accuracy of inter-researcher similarity measures based on topical and social clues. *Scientometrics*, *87*(3), 597–620.

Cabanac, G., & Preuss, T. (2013). Capitalizing on order effects in the bids of peer-reviewed conferences to secure reviews by expert referees. *Journal of the American Society for Information Science and Technology*, *64*(2), 405–415.

Caragea, C., Silvescu, A., Mitra, P., & Giles, C. L. (2013). Can't See the Forest for the Trees? A Citation Recommendation System. *iConference 2013 Proceedings* (pp. 849–851). doi:10.9776/13434

Dong, R., Tokarchuk, L., & Ma, A. (2009). Digging Friendship: Paper Recommendation in Social Network. *Proceedings of Networking & Electronic Commerce Research Conference (NAEC 2009)* (pp. 21–28).

Ekstrand, M. D., Kannan, P., Stemper, J. A., Butler, J. T., Konstan, J. A., & Riedl, J. T. (2010). Automatically building research reading lists. *Proceedings of the fourth ACM conference on Recommender systems* (pp. 159–166). ACM.

Feyer, S., Siebert, S., Gipp, B., Aizawa, A., & Beel, J. (2017). Integration of the Scientific Recommender System Mr. DLib into the Reference Manager JabRef. *Proceedings of the 39th European Conference on Information Retrieval (ECIR)*.

Guse, D., Schuck, S., Hohlfeld, O., Raake, A., & Möller, S. (2015). Subjective quality of webpage loading: The impact of delayed and missing elements on quality ratings and task completion time. *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on* (pp. 1–6). IEEE.

Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *5*(4), 19.

He, Q., Pei, J., Kifer, D., Mitra, P., & Giles, L. (2010). Context-aware citation recommendation. *Proceedings of the 19th international conference on World wide web* (pp. 421–430). ACM.

Hienert, D., Sawitzki, F., & Mayr, P. (2015). Digital Library Research in Action – Supporting Information Retrieval in Sowiport. *D-Lib Magazine*, *21*(3/4). doi:10.1045/march2015-hienert

Jannach, D. (2014). Recommender systems: an introduction. *Lecture Slides (PhD School 2014)*.

Kim, W., Xiong, S., & Liang, Z. (2017). Effect of Loading Symbol of Online Video on Perception of Waiting Time. *International Journal of Human–Computer Interaction*, (just-accepted).

Konstan, J., & Ekstrand, M. D. (2015). Introduction to Recommender Systems. *Coursera Lecture Slides*.

Langer, S., & Beel, J. (2014). The Comparability of Recommender System Evaluations and Characteristics of Docear's Users. *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design (REDD) at the 2014 ACM Conference Series on Recommender Systems (RecSys)* (pp. 1–6).

Langer, S., & Beel, J. (2017). Apache Lucene as Content-Based-Filtering Recommender System: 3 Lessons Learned. *5th International Workshop on Bibliometric-enhanced Information Retrieval (BIR) at the 39th European Conference on Information Retrieval (ECIR)*.

McNee, S. M., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S. K., Rashid, A. M., Konstan, J. A., et al. (2002). On the Recommending of Citations for Research Papers. *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. doi:http://doi.acm.org/10.1145/587078.587096

McNee, S. M., Kapoor, N., & Konstan, J. A. (2006). Don't look stupid: avoiding pitfalls when recommending research papers. *Proceedings of the 20th Conference on Computer supported cooperative work*.

Pennock, D. M., Horvitz, E., Lawrence, S., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence* (pp. 473–480). Morgan Kaufmann Publishers Inc.

Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender Systems Handbook (2nd edt.)*. Springer.

Rokach, L., Mitra, P., Kataria, S., Huang, W., & Giles, L. (2013). A Supervised Learning Method for Context-Aware Citation Recommendation in a Large Corpus. *Proceedings of the Large-Scale and Distributed Systems for Information Retrieval Workshop (LSDS-IR)* (pp. 17–22).

Schwarzer, M., Schubotz, M., Meuschke, N., Breitinger, C., Markl, V., & Gipp, B. (2016). Evaluating Link-based Recommendations for Wikipedia. *Proceedings of the 16th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*, JCDL '16 (pp. 191–200). doi:10.1145/2910896.2910908

Selvidge, P. R., Chaparro, B. S., & Bender, G. T. (2002). The world wide wait: effects of delays on user performance. *International Journal of Industrial Ergonomics*, *29*(1), 15–20.

Siebert, S., Dinesh, S., & Feyer, S. (2017). Extending a Research Paper Recommendation System with Bibliometric Measures. *5th International Workshop on Bibliometric-enhanced Information Retrieval (BIR) at the 39th European Conference on Information Retrieval (ECIR)*.

Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B. P., & Wang, K. (2015). An overview of Microsoft Academic Service (MAS) and applications. *Proceedings of the 24th WWW conference* (pp. 243–246).

Torres, R., McNee, S. M., Abel, M., Konstan, J. A., & Riedl, J. (2004). Enhancing digital libraries with TechLens+. *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries* (pp. 228–236).