# Babel: A Platform for Facilitating Research in Scholarly Article Discovery

Ian Wesley-Smith
Information School
University of Washington
Seattle, WA, 98195-1800
iwsmith@uw.edu

Jevin D. West
Information School
University of Washington
Seattle, WA, 98195-1800
jevinw@uw.edu

## ABSTRACT

The body of scientific literature is growing at an exponential rate. This expansion of scientific knowledge has increased the need for tools to help users find relevant articles. However, researchers developing new scholarly article recommendation algorithms face two substantial hurdles: acquiring high-quality, large-scale scholarly metadata and mechanisms for evaluating their recommendation algorithms. To address these problems we created Babel—an open-source web platform uniting publisher, researchers, and users. Babel includes tens of millions of scholarly articles, several content-based recommendation algorithms, and tools for integrating recommendations into publisher websites and other scholarly platforms.

## Keywords

Recommenders; Scholarly Article Recommendation; Experimentation Platforms; Citation Networks; Information Retrieval

## 1. INTRODUCTION

Thousands of scholarly articles are published every day, making it impossible for researchers to stay apprised of the current literature. There is a strong need to develop algorithms and platforms that help users find relevant papers. Recent developments in the commercial and non-profit space are producing improved academic search engines (e.g., Semantic Scholar, Bioz, Meta, CiteseerX, Microsoft Academic Search, Google Scholar, Web of Science, etc). Though these are exciting improvements in search, not all researchers have access to the data, computational resources, and usage data from these different platforms. Additionally, there is a difference between finding a paper known to exist (search) and *discovering* a paper not known to exist (recommendation). We see a need in the Big Scholarly Data space for a platform providing scholarly article metadata and usage data to enable research in scholarly article recommendation. To fill that void we created Babel[1], a web platform that facilitates scholarly recommender research by removing the barriers to acquiring large-scale scholarly metadata and improves access to usage data for evaluating these novel methods.

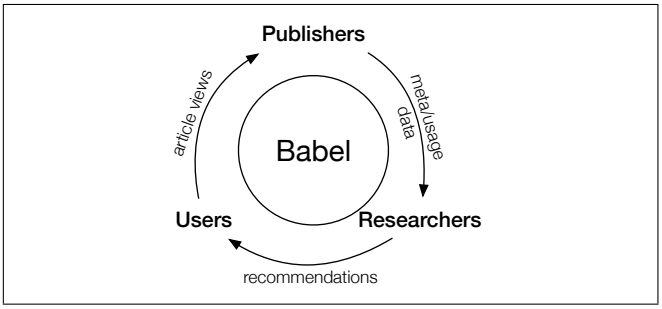


**Figure 1:** Babel is a cooperative model between publishers, researchers, and users where all gain from the venture. Publishers provide high-quality metadata and usage data to Babel. Babel then provides access to this data to researchers in a manner that protects publishers' intellectual property. Researchers benefit from this high-quality data, allowing them to create more effective recommenders. These recommendations are then delivered to users, helping them quickly find the content they need.

Since much of scholarly literature lies behind publisher paywalls, it is difficult to scale and test new recommendation algorithms beyond small samples or discipline-specific corpora (e.g., DBLP). Few publishers make their articles available for data mining, and those that are available are legally encumbered, making it difficult to share results and reproduce the work of others. Some have chosen to crawl publishers to create their own datasets, but this risks legal action over redistribution of copyrighted material, also limiting reproducibility. A few of the more forward thinking publishers have released open datasets to help spur interest in this domain. PLoS, for example, has provided their data for mining and research purposes. DBLP [6] and AMiner [9] are good examples of test collections, but combined they represent a small portion of the full literature, limiting the generalizability of results obtained with them. Conversely, Microsoft Academic Search has released a very large citation graph, but the data tends to be of lower quality since it comes from a web crawler. This leaves researchers[2] in a

[1] http://babel.eigenfactor.org

[2] In this paper, we define "researchers" as researchers doing work specifically on scholarly recommender systems. We refer to "users" as researchers/scholars reading the literature.


*WWW'16 Companion,* April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890517 .

difficult situation: invest incredible effort in getting these datasets from publishers with the knowledge that your experiments can't be replicated, crawl the datasets and risk legal action, or use the small publicly available data sets which are of limited practical use and may not be suitable for recommendation algorithms that require the entire scientific corpus [3, p. 260].

The second issue researchers face is determining the efficacy of their recommendation algorithms. Most methods are written, published and never implemented at scale. Although techniques exist for offline evaluations of recommenders in domains with explicitly rated items (such as movies or products), it is much more difficult to evaluate recommendations of implicitly rated items, requiring either usage data or expert judges [3, p. 273]. Furthermore, offline evaluations of scholarly article recommenders have been shown to be poor predictors of online performance [1]. This has led to a proliferation of ad-hoc techniques and a lack of rigor that has stifled progress in the field, with an inability to determine which avenues of investigation are fruitful. In a comprehensive survey of over 170 articles on research paper recommenders Beel et al. concluded "...that it is currently not possible to determine which recommendation approaches for academic literature are the most promising. However, there is little value in the existence of more than 80 approaches if the best performing approaches are unknown" [2]. We want to help change this.

Our goal with Babel is to build an open-source platform that overcomes the data and usage obstacles noted above. Babel provides up-to-date bibliographic data for more than 37 million scholarly articles and more than 300 million citations from various disciplines (economics, biomedicine, physics, computer science, etc). We want researchers focused on developing new recommenders, not thinking about updating SQL databases, running servers, and establishing usage agreements with publishers.

## 2. RELATED WORK

This idea of a platform to support scholarly literature activities is not a new one. Two well known platforms are the digital libraries, arXiv and CiteSeerX [7]. ArXiv primarily concerns itself with persistent, open hosting of its submissions. CiteSeerX, by contrast, does not serve as a primary archive, but instead aggregates data from many different sources. CiteSeerX also provides additional tooling on top of its collection, including search, summaries and citation statistics. Babel, though, is neither a digital library nor a search engine; it is a service that implements and tests the most current recommendation algorithms on a large-scale data set, spanning many disciplines, and makes these recommendations freely available to the publishing community and other platforms where users come to find articles. Babel was designed to build on digital libraries and leverage their existing collections, allowing these libraries to integrate with Babel to provide recommendations to their users.

Babel is most closely related to theadvisor [5], a project out of The Ohio State University. Theadvisor is a web service providing recommendations for a set of scholarly articles. Much like Babel, theadvisor provides REST access to their recommendations. What makes Babel different is that it generalizes this idea of scholarly article recommendation as a service, providing access to many different recommenders instead of just one. Babel is designed as a platform for experimentation with scholarly article recommenders, providing access to many different recommenders simultaneously, collecting usage data and measuring their relative efficacy.

Our goals for Babel are relatively simple: allow for the evaluation and comparison of recommender algorithms for scholarly literature, decrease the difficulty of developing recommender algorithms for scholarly literature, increase the quality of recommendations available for scholarly literature, and finally to provide enterprise grade reliability and performance on the platform so publishers feel comfortable using it in production.

## 3. AUDIENCE

Babel is built for two groups: consumers of scholarly article recommendations (e.g. publishers and their users) and producers of those recommendations (scholarly article recommender researchers). We anticipate that Babel will be of special interest to smaller publishers who may not have the time or budget to hire a data science team. These publishers would only have to insert a JavaScript widget into their site to gain access to the state-of-the-art in scholarly recommendation, with methods being continually iterated upon by a team of researchers. One such publisher that we are already working with is the Public Library of Science (PLoS). We have integrated their article metadata into Babel and are producing recommendations. PLoS will receive access to these recommendations through our JavaScript widget (available in 2016). We also recently collaborated with JSTOR, providing recommendations for their labs project on sustainability[3].

Although we anticipate most of our usage will come from publishers, we are excited to see what other applications can be developed leveraging this data. For example, services like Authorea and Zotero may benefit from recommendations as authors write their papers and organize content around a particular topic. We have also developed web browser extensions for academic platforms with high usage. For example, we have built browser plugins[4] for Google Chrome and Firefox that provide recommendations as you search Google Scholar.

The second audience for Babel are researchers of recommendation algorithms. Researchers are able to develop against the open datasets in the Babel collection, all of which are provided in a standardized format. Once researchers have successfully integrated against the open datasets, they send us their recommenders and we run them against the publisher datasets and begin providing the recommendations to consumers. Shortly thereafter, we will compare the researchers' algorithm's performance to other recommenders using usage data provided by the publishers. This rapid evaluation will allow researchers to focus on improving the quality of their algorithms, not the details of acquiring datasets and tracking user behavior.

## 4. LICENSING

The source code for the platform, web extensions, JavaScript widgets, and some recommendation algorithms are open source and available on GitHub. Much of the scholarly metadata is

[3] http://labs.jstor.org/sustainability/
[4] https://github.com/kyleestlick/babel

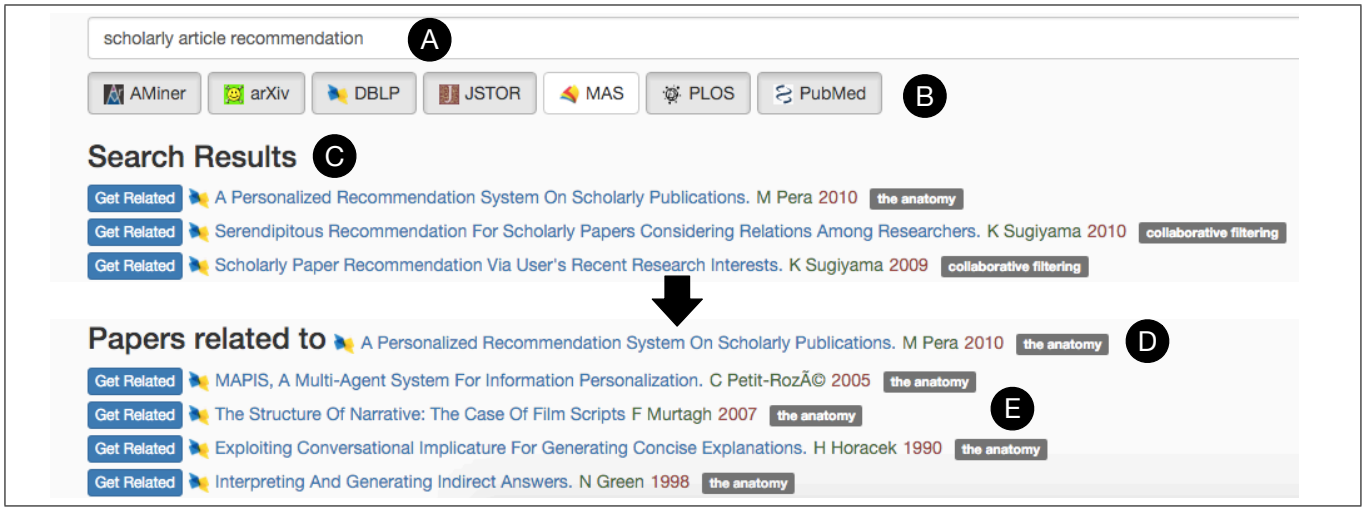


**Figure 2:** Screenshot of the Babel demonstration website. (A) is a standard search box, which will query over all the metadata Babel has, including titles, authors and labels. (B) allows users to select which datasets will be searched. (C) shows a list of results, with each result including a link to get recommended papers, the dataset it was retrieved from, the title of the paper, first author, publication year and a label for the cluster determined by the Eigenfactor Recommends algorithm [10, 11]. On clicking "get related", recommendations (E) are shown for the source paper (D).

also available for research (DBLP, MAS, arXiv, PLoS), but some are restricted by license agreements.

Our goal is to allow other groups to easily setup their own version of Babel, providing API compatibility while keeping their algorithms or data in-house. Authors of recommender algorithms maintain full IP rights to their recommenders, though recommendations on the Babel platform hosted by us will be distributed freely and unencumbered. Recommender authors can, at any time, ask us to remove their recommender and we will comply in a timely fashion.

## 5. EVALUATION CRITERIA

Babel was created to answer one simple question: how can we evaluate the performance of scholarly article recommenders? Broadly there are two approaches to answering this question: offline evaluation and online evaulation. Offline approaches rely on pre-collected data of ratings or user actions, making them much cheaper to use. However, these approaches have limited predictive power and require that these "ground truth" datasets exist; which, for scholarly article recommendation, they don't.

Furthermore, research indicates that scholarly article recommenders are not amenable to offline evaluation. Beel et al. [1] found that offline evaluation could not predict click through rate, and that for citation based methods the predictive capability was especially poor. In fact, for citation based methods the prediction was off by nearly and order of magnitude: "the offline evaluation predicted a disappointing result of 0.96%. In practice, the citation-based approach had a CTR [click-through rate] of 8.27%" [1].

Given this strong critique of offline evaluation it is surprising that a survey of scholarly recommenders [2] found only 5 of the 89 approaches were evaluated with online approaches. The most likely reason for this is cost of performing online evaluations: they require live users and are very expensive to execute. These approaches do provide much stronger validation of recommendations—you are directly influencing user behavior and recording the result. For these reasons Babel uses online evaluation using long-term, observational field studies focused on two outcomes: did a user view a paper's abstract and did a user download a paper? We chose these metrics because they are user centric and they measure the outcomes we are interested in: did a user find a recommendation useful?

Answering these questions requires substantial article-level data and usage data. This is why we are trying to build a tool that is useful to both users and publishers. We define the following metrics which Babel will generate on a per-algorithm basis at regular intervals:

**Click-through rate (CTR)** clicks/impressions * 100%
**Download rate (DR)** downloads/impressions * 100%
**Transition rate (TR)** downloads/clicks * 100%

Impressions are defined as the number of times a recommendation generated by a specific algorithm was shown.

It is important to note that all of these metrics are relative to other recommenders, they don't provide an absolute measure of performance. If a researcher wants to determine the performance of their algorithm they will compare it to benchmark algorithms we have implemented, or, as they iterate on their algorithm, its performance over time. We believe that this can begin to provide a standardized mechanism for online evaluation of scholarly article recommenders.

## 6. ARCHITECTURE

Babel's architecture (figure 4) is designed to provide content based recommendations as a service via REST APIs. Our design goals centered around providing low-latency recommendations suitable for client consumption from a web browser, while scaling horizontally to support a large number of concurrent users. Note that the current version of Babel does not support personalized recommendations. Although these types of recommenders are very important, it was technically infeasible to support them initially, though if Babel is successful we would like to add support for personalized recommendations later.

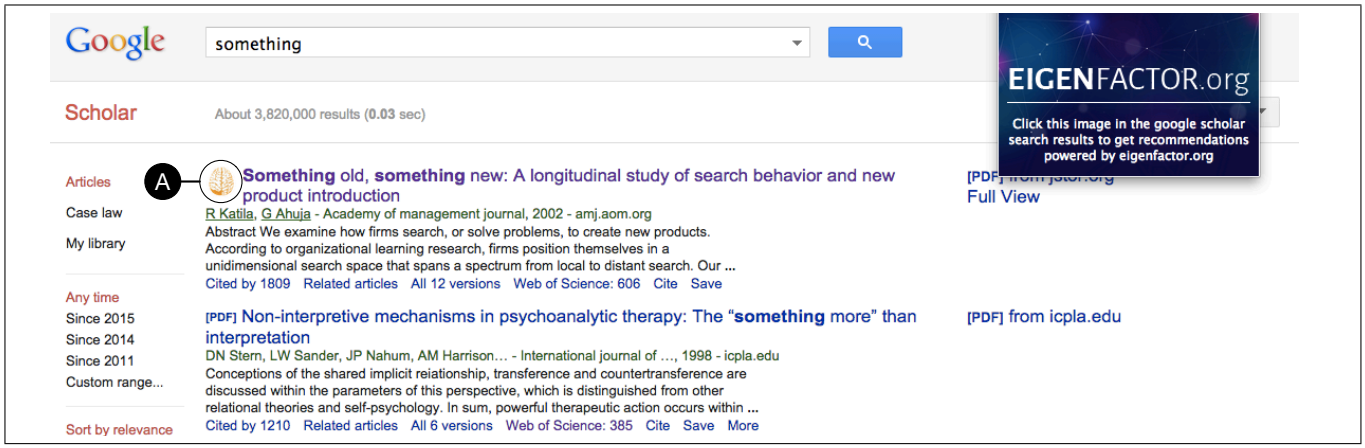


**Figure 3:** Screenshot of our browser plugin (Chrome or Firefox), which provides recommendations directly on the Google Scholar site. The plugin scrapes the titles of a search result and looks them up in Babel. If Babel has recommendations for this title a small brain icon (A) is inserted next to the result. Clicking this icon takes you to the Babel demonstration website (Figure 2), where recommended papers are displayed.

## 6.1 Frontend

The frontend (figure 4.A) is what consumers will interact with. It exposes a REST API, which, given a paper identifier and a publisher, will generate a list of recommendations with a random recommendation algorithm and return them in JSON format. The frontend retrieves recommendations for the recommendation cache, and emits analytic events to allow for evaluation of various recommenders performance. The frontend also provides APIs to allow for querying of the metadata database to find relevant papers, though this is not the primary use-case for Babel. Detailed documentation of the current API exposed by the endpoint is available at `http://babel.eigenfactor.org/api.html`

## 6.2 Backend

The backend (figure 4.B) is responsible for data ingestion, normalization, and recommendation generation. Whenever a publisher uploads a new dataset to Babel, metadata is extracted, normalized, and stored in the metadata database. Simultaneously the publisher's raw data is archived and then transformed into a normalized format suitable for ingestion by recommender algorithms. Once normalization has occurred all available recommenders are run in parallel. This batch processing step allows for expensive content-based recommenders to be run on a regular basis, caching their results. Their output is then pushed to the recommendation cache, which is currently implemented in DynamoDB. At this point queries against the frontend will return new recommendations.

## 6.3 Logging and Analysis

Given that evaluation of recommenders is one of the primary goals of this platform we need to have a mechanism to submit feedback about recommendations. The frontend provides an API for submitting feedback, allowing recommendation consumers (e.g. publishers) to signal what action was taken by a user.

First, anytime a recommendation is requested a `transaction_id` is generated. This value is unique to this recommendation, and it, along with the `paper_id` and `publisher` are logged. Next, if a user acts on a recommendation, the consumer will make a feedback call, including the `transaction_id`, selected `paper_id` and `action` that occurred. There are two possible actions: *click*, which denotes a user clicking on a recommendation and being directed to page with more information about the article, and *download*, which denotes a user actually downloading an article. The API also, optionally, allows for inclusion of a `client_id`—an id uniquely identifying each consumer (e.g. PLoS, Babel demonstration site, Google Scholar plugins). As new requirements are added we will expand the capabilities of the analytics system to record data. Figure 4.C shows where feedback is aggregated and analysis is performed. Currently this is done by a third-party vendor, but once there is enough usage data this will likely be moved in-house.

| Dataset | Papers | Citations | Recommendations |
|---|---|---|---|
| AMiner | 2,092,356 | 8,024,869 | 22,112,496 |
| JSTOR† | 1,787,351 | 8,227,537 | 14,813,224 |
| PLoS | 1,599,712 | 3,232,766 | 8,647,037 |
| PubMed | 5,538,322 | 16,004,596 | 34,026,854 |
| arXiv | 626,441 | 781,108 | 5,624,262 |
| DBLP | 781,108 | 4,191,677 | 2,163,313 |
| MAS | 27,352,532 | 262,554,975 | 245,796,494 |
| Total | 37,894,701 | 303,017,528 | 333,183,680 |

**Table 1: Datasets currently available on Babel. The total paper count is not a measure of unique papers in the corpus; a single paper may appear in several different datasets. † denotes a closed dataset.**

# 7. FUTURE WORK

Currently, Babel's frontend is fully operational, allowing for recommendations to be delivered and feedback on user actions to be submitted and recorded. The publisher widget is still in early stages, but is one of our highest priorities.

Babel currently delivers two types of recommendations: EigenFactor Recommends (Expert and Classic) [11], a citation-based algorithm that relates papers based on their location in citation space. We plan on adding additional algorithms

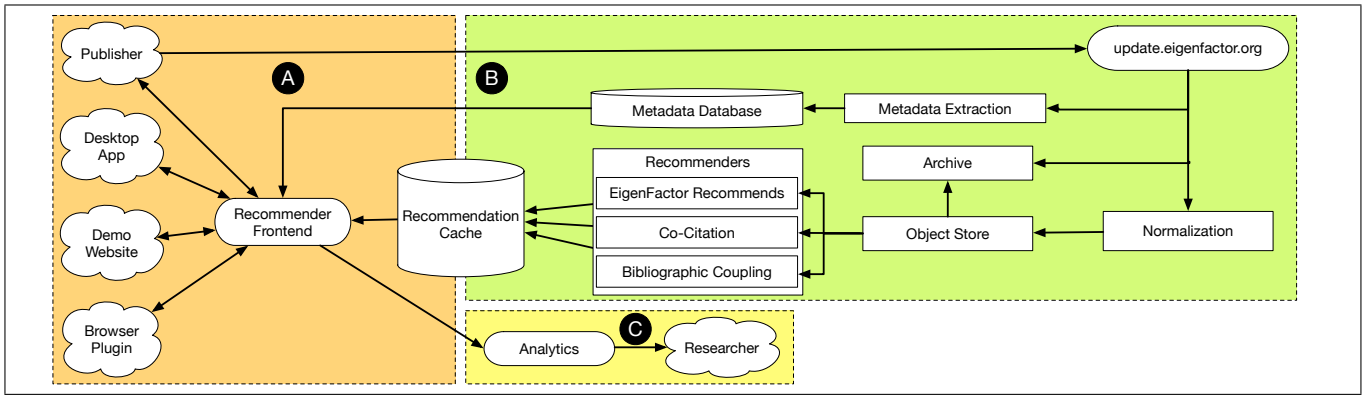


**Figure 4: Babel's Architecture**. The orange area (A) is the frontend that most clients interact with. This is a traditional web service exposing a REST API that provides scholarly article recommendations freely, to anyone in the world. The green area (B) is the backend, where recommendation generation and dataset ingestion and normalization occurs. The yellow area (C) is where logging and analysis occurs. Each recommendation, along with any action taken, is logged and aggregated, allowing researchers to compare relative algorithm performance.

in the near future, including bibliographic coupling [4] and co-citation [8].

Table 1 shows a breakdown of the over 300 million recommendations Babel provides, consisting of seven different datasets and over 39 million papers. Our current goal is to on-board several new publishers of various sizes to validate our endpoint design, specifically around latency, scalability and analytics. Once we have done this we will begin soliciting regular updates from publishers to validate the recommendation generation phase. Finally, we will open the system to researchers who wish to implement new recommendation algorithms. We provide API documentation and aa FAQ at `http://babel.eigenfactor.org`.

Babel is currently a research project, but it will eventually need to find a home. There are several examples of non-profit membership fee based entities, such as CrossRef, ORCID and DOI that provide examples of how "common good" projects like Babel can exist. Since the platform code is open-source, in the worst case scenario another group could fork the code base and start running an improved platform with improved data and tools. The need for finding relevant articles is growing and hopefully additional tools for meeting this need will emerge.

## 8. ACKNOWLEDGMENTS

This work was supported by the Metaknowledge Network funded by the John Templeton Foundation. We would like to thank Kyle Estlick for his development of the Google Chrome and Firefox browser plugins, and Nick Thorpe for feedback on early drafts of this paper.

## 9. REFERENCES

[1] J. Beel, M. Genzmehr, S. Langer, A. Nürnberger, and B. Gipp. A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, pages 7–14, New York, NY, USA, 2013. ACM.

[2] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger, and A. Nürnberger. Research paper recommender system evaluation: A quantitative literature survey. In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, RepSys '13, pages 15–22, New York, NY, USA, 2013. ACM.

[3] P. B. Kantor, F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*. Springer, Dordrecht, 2011.

[4] M. M. Kessler. Bibliographic Coupling Between Scientific Papers. *American Documentation (pre-1986)*, 14(1):10, 1963.

[5] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek. Recommendation on Academic Networks using Direction Aware Citation Analysis. *ArXiv e-prints*, May 2012.

[6] M. Ley. The dblp computer science bibliography: Evolution, research issues, perspectives. In A. Laender and A. Oliveira, editors, *String Processing and Information Retrieval*, volume 2476 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2002.

[7] H. Li, I. Councill, W.-C. Lee, and C. L. Giles. Citeseerx: An architecture and web service design for an academic document search engine. In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 883–884, New York, NY, USA, 2006. ACM.

[8] H. Small. Co-Citation in Scientific Literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973.

[9] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998, 2008.

[10] J. West, M. Rosvall, D. Vilhena, and C. Bergstrom. Ranking and mapping article-level citation networks. *In Prep.*, 2016.

[11] J. West, I. Wesley-Smith, and C. Bergstrom. A recommendation system based on hierarchical clustering of an article-level citation network. *In Prep.*, 2016.