Crowdsourcing Peer Review: As We May Do

Michael Soprano(✉) and Stefano Mizzaro

Department of Mathematics, Computer Science, and Physics, University of Udine, Udine, Italy
michael.soprano@outlook.com, mizzaro@uniud.it

Abstract. This paper describes Readersourcing 2.0, an ecosystem providing an implementation of the Readersourcing approach proposed by Mizzaro [10]. Readersourcing is proposed as an alternative to the standard peer review activity that aims to exploit the otherwise lost opinions of readers. Readersourcing 2.0 implements two different models based on the so-called codetermination algorithms. We describe the requirements, present the overall architecture, and show how the end-user can interact with the system. Readersourcing 2.0 will be used in the future to study also other topics, like the idea of shepherding the users to achieve a better quality of the reviews and the differences between a review activity carried out with a single-blind or a double-blind approach.

Keywords: Scholarly publishing · Peer review · Crowdsourcing

1 Introduction

The main mechanism to spread scientific knowledge is the *scholarly publishing* process, which is based on the *peer review* activity; a scientific article written by some authors is judged and rated by colleagues of the same degree of competence.

Although peer review is a reasonable and well established a priori mechanism to ensure the quality of scientific publications, it is not free from problems, and indeed it is characterized by various issues related to the process itself and the malicious behaviour of some stakeholders. Just to cite an example, in some cases reviewers cannot correctly evaluate a publication, e.g., when the paper reports data from an experiment which is long and complex and, therefore, not replicable by the reviewer itself; thus, an act of faith (that the author is honest) is sometimes required [4].

Also taking into account several of the issues and flaws of peer review that are widely analyzed in the literature, Mizzaro [10] conjectures that reviewers of scientific publications can be seen as a scarce resource which is being exhausted. To support such a thesis Mizzaro describes in detail ten different factors that contribute to it. As a solution, he proposes to take advantage of readers' opinions by outsourcing the peer review activity to their community and calls this approach *Readersourcing*, as a portmanteau for "crowdsourcing" and "readers".

P. Manghi et al. (Eds.): IRCDL 2019, CCIS 988, pp. 259–273, 2019.
https://doi.org/10.1007/978-3-030-11226-4_21

Although this might seem a radical solution, it is important to remark that: (i) similar approaches, suggesting variants and changes to peer review including collaborative reviews and/or a more distributed peer review practice, have already been proposed in the past [2,3,8]; and (ii) the usage of crowdsourcing in scholarly publishing is being proposed and analyzed for even more radical approaches, for example to outsource some steps of writing of scientific publications [15].

A crucial aspect of Readersourcing that must necessarily be addressed consists in providing a mechanism for which being a "good" reviewer is gratifying, in order to encourage readers to express adequate ratings (i.e., ratings which are truthful and unbiased). As a related issue, some stakeholder of the scholarly publishing process can have a malicious behaviour. This is reported from multiple sources and it can be caused by different factors. In this paper we do not provide an exhaustive description of all those behaviours and their causes, but we mention some examples. In particular, there is a recent article published on The Economist [16] according to which there are more and more journals where peer review activity is not performed, in contrast with what stated by their publishers. This leads scholars to inflate the list of their publications with articles that, probably, would not pass peer review. There are several discussions related to this phenomenon. One of the main causes (although it is not the only one) is the change of the business model that allows publishers to get a profit. In recent years those publishers have gone from monetization through the resale of subscriptions to readers to a payment request of a publication fee to the authors of articles. These articles can subsequently be read without any payment according to the open access model. This model, therefore, promotes the dissemination of knowledge but, at the same time, risks corrupting it. The article of The Economist [16] goes on by describing different malicious behaviours adopted by publishers of at least questionable periodicals to appear respectable and trustworthy when in the reality it is not like that at all. All this is also caused by the institutions of the scientific world which seem to worry less about where the financed research is published. We remark that we are not stating the existence of connections between the lack of reviewers and the malicious behaviour of some stakeholders; rather, these issues provide two different motivations to our work.

One of the possible solutions to deal with the above mentioned issues could be to rely on readers. As hypothesized by other researchers, it can be assumed that readers are a resource of which there is no shortage: they are many more than the reviewers, so if their opinions can be gathered they might allow to rate publications quality. This approach is not free from problems itself (e.g., lobbies, lazy readers [9]); although these need to be taken into consideration we do not have the space in this paper to discuss them.

In this paper we present our implementation of a system which is called *Readersourcing 2.0* that has two main goals: (i) to implement different models in order to take advantage of readers' ratings, as well as making easy to add more of them in the future, and (ii) to allow readers to express their ratings in a way that does not require too much effort, with just a few clicks or keystrokes. This paper is structured as it follows: Sect. 2 describes what a generic Readersourcing model should to able to do and compute, and how it should do that; then, two of those models are presented. Section 3 presents Readersourcing 2.0 implementation; we list its requirements, sketch its architecture, and briefly present the technologies actually used for its development. Moreover, we describe it from the point of view of a reader by showing and commenting its user interface and interaction capabilities. Section 4 concludes the paper.

2 Models

To outsource the peer review activity of publications to their readers a model of some kind can be useful. Every publication is characterized by one or more numerical ratings, each one provided by a reader. These ratings are the input data of such a model. From these data the model should define a way to measure the overall quality of a publication as well as the reputation of a reader as an assessor; moreover, from these measures it should be possible to derive the reputation of a scholar as an author. In other terms, the main issue to deal with consists in how the ratings that the assessed entity (i.e., a publication) receives should be aggregated into indexes of quality and, from these indexes, how to compute indexes of reputation for the assessors (i.e., the readers) and, eventually, indexes of how much an author is "skilled" (i.e., a measure of his ability to publish papers which are positively rated by their readers). In the following, we hypothesize to compute a single index for each of these measures.

This aggregation must be carried out by taking into consideration the fact that not all ratings are equal and that each of them has an intrinsic adequacy (i.e., a measure of how much truthful and unbiased they are) that characterizes it; in other words, it has to be possible to distinguish adequate from inadequate ratings and, then, good from bad assessors and, again, skilled from unskilled authors. Models that are able to do this are based on *co-determination algorithms*. In these algorithms the quality of the assessed entities is used to estimate the corresponding reputation/skill of the assessors/authors. Every time a new rating is given, these quantities are updated.

There is a further aspect to consider: what scale of values readers should use to express their ratings? continuous or discrete values? which interval of values should be used? Inevitably, this choice has an impact on the chosen co-determination algorithm. Medo and Rushton Wakeling [7] study the performance of different co-determination algorithms with ratings characterized by continuous or discrete values. From their analysis emerges as the best alternative the use of a scale characterized by an interval of values sufficiently "detailed" (e.g., 0–100) as it leads, in general, to the best performance for the co-determination process. Such a scale, in a real application, can be easily implemented and used by means of a slider component in the user interface.

A key point of the above characterization is that it allows to exploit the Readersourcing approach as a pre-publication replacement or as a post-publication addition to the standard peer review activity.

We now briefly describe two models based on co-determination algorithms.

2.1 The Readersourcing Model

The first model that we describe is the *Readersourcing Model* (RSM) proposed by Mizzaro [10] on the basis of a previous work [9]. In RSM, three different entities are identified: publications, authors, and readers. Each of them is characterized by a rating; articles scores aim to measure their quality and authors/readers scores measure their skill/reputation. A generic user of a system based on this model can assume both the roles of an author and a reader. As a reader of publications, the user is asked to give a numerical rating to those he read. As an author of a publication, a user is characterized by a score computed on the basis of the ratings given by readers. More generally, scores are dynamic and they change depending on user behaviour. For example, if an author with a low score publishes an article positively judged by readers, that score increases. If a reader expresses an inadequate rating (i.e., a rating which is judged as untruthful and/or biased because "distant" from other ratings) about an article, his score decreases, and so on.

Each entity characterized by a score also has an associated steadiness value (of the score itself). For example, publications read and rated by many readers will have high steadiness, while those of new authors and readers will have low steadiness. Steadiness affects the update of the scores because a high (low) value of it leads to faster (slower) changes of the scores themselves. As time passes, authors add new publications while readers read and rate them with numerical ratings. Those actions lead to an update of scores and steadiness values. High values of the scores represent, depending on the entity to which they refer, quality publications or skilled/good authors and readers, while steadiness values provide an estimate of how much that scores are reliable and stable.

2.2 The TrueReview Model

A second model is the *TrueReview Model* (TRM) proposed by De Alfaro and Faella [5]. They identify issues similar to those identified by Mizzaro [10] and they define an incentive system for the readers of publications to ensure that those publications will receive adequate reviews and precise evaluations.

There is a key difference that characterizes TRM with respect to RSM; the former does not computes a quality score for publications, while the latter does that. TRM, indeed, computes only a score for readers and leaves complete freedom about how to aggregate into a single index the ratings received by a single publication. However, the basic reader action does not change; readers are asked to give a single numerical rating to the publications they read.

The incentive scheme proposed by De Alfaro and Faella [5] aims to reward a reader every time that he provides a rating which is *informative* and *accurate*. Therefore, every given rating contributes with a certain "bonus" to reader's reputation which is computed on the basis of those two parameters. The aggregation of all these bonuses gives the reader's reputation itself and the one with the highest reputation is the "best" reviewer. Then, those reviewers should be ranked by their reputation and publicly shown in order to establish a healthy competition by pushing "bad" reviewers to improve themselves.

In order to compute the bonus for a given rating, the *informativeness* provides an incentive to select publications whose current evaluation is most different from what the future consensus will be, so it depends on the *previous* and the *future* ratings of the paper, but not on the one given by the reader under consideration. Thus, once the reader rates a certain publication, informativeness plays no further role and the bonus depends entirely on *accuracy loss*. The accuracy loss is computed by comparing the current rating provided by the reader under consideration with future ratings only. This eliminates any incentive to give a rating similar to those already expressed in the past by other reviewers which, probably, would end up going against the true beliefs of the reader itself.

In other words, this incentive scheme based on the concepts of informativeness and accuracy aims to reward reviewers who provide new information which is then presented to other reviewers in a compelling manner.

3 Readersourcing 2.0

In Sect. 1 we have outlined the motivations and the models which are the basis for the idea of outsourcing the peer review activity of scientific publications to their readers to improve the quality of the scholarly publishing process. We have implemented a system that actually allows to take advantage of this approach which is called *Readersourcing 2.0* and is now described.

3.1 Requirements

There are four main requirements that Readersourcing 2.0 has to satisfy:

- **R1**: provide to the reader a way to rate a publication by expressing a numerical rating in a seamless and effortless way;
- **R2**: allow readers to review publications in a way which is independent from the used device or software;
- **R3**: be able to aggregate the ratings received by a publication according to both RSM and TRM and show the computed scores to the users;
- **R4**: be general, extensible, and easily adaptable to other models besides RSM and TRM.

R1 is imperative: if the rating activity is not seamless and fast, readers will simply not rate the papers they read; the system must not require too much effort to the reader, which has to be able to express the rating with just a few clicks or keystrokes and without the need to open more windows, new browser tabs or even an external software. R1 is related to R2, which depends on how scholarly publishing is carried out. Usually, scientific articles are collected into journals that are made available to the scholars community through some publishing systems. So, the digital library of a publishing company consists in a large collection of files mainly encoded in PDF format. This might not be the best solution, but it is a sort of de facto standard. When a reader wants to read a publication, he will look for one of those PDF files available on such publishing systems. Once he finds the wanted one, it is opened inside a browser (tab) and then the reader itself has the choice to read it there or to download and store it somewhere on his filesystem, or even on an external device. This is the pivotal point; the reader must be able to easily rate the publication directly from the browser or from some sort of a reference stored *inside* the downloaded file.

A rating component located on the user interface of the client browser will suffice to satisfy R1: the user will simply select a rating for the publication and click a button. However, that cannot satisfy all the use cases, and R2 must be taken into consideration. Indeed, a reader could just close the browser and read the publication by using other software (PDF viewers) or even other devices (e.g., tablets). These considerations justify the previously introduced "file-oriented" approach. Our proposed solution consists in the following steps: (i) the publication chosen by the reader is downloaded locally to Readersourcing 2.0 server, (ii) the corresponding PDF file is annotated with a link that, once clicked, will take the reader to an ad-hoc rating page. After that, the system makes the paper available for the download.

3.2 Architecture

Readersourcing 2.0 is an ecosystem composed of more than one application. Indeed, there must be one application that acts as a server to gather all the ratings given by readers and one that acts as a client to allow readers to effectively rate publications. There is one additional component since the task of editing files encoded in PDF format is carried out by an ad hoc software library exploited by the server side application. An overview of Readersourcing 2.0 architecture is shown in Fig. 1; in the following we briefly describe these three components.

RS_Server [13] is the server-side application which has the task to collect and aggregate the ratings given by readers and to use RSM and TRM to compute quality scores for readers and publications. RS_Server must be deployed on a machine along with an instance of RS_PDF, otherwise it can not work properly. Then, there are up to n different browsers, with the corresponding end-users, which communicate with the server: each of them has an instance of *RS_Rate*, which is the true client. Both RS_PDF and RS_Rate are described in the following. This setup means that every interaction between readers and server is carried out through clients installed on readers' browsers and these clients have to handle the registration and authentication of readers, the rating action and the download action of link-annotated publications.

During the design phase of RS_Server some strategies have been adopted to ensure its extensibility and generality, to meet the R4 requirement proposed in Sect. 3.1. This means that: (i) it is straightforward to add new models, (ii) each model shares the same input data format, and (iii) if a model needs to save values locally to the RS_Server (i.e., in its database), there is a standard procedure to allow that.

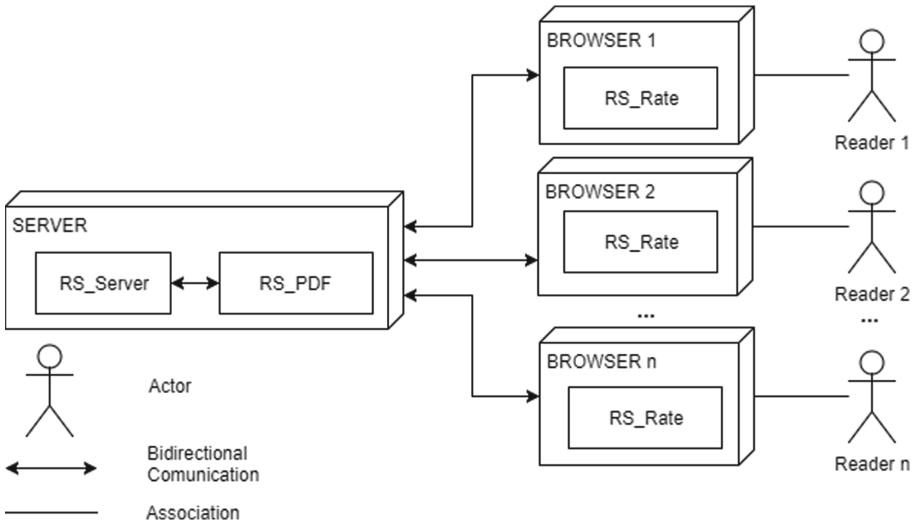


Fig. 1. Architecture of Readersourcing 2.0.

RS_PDF [11] is the software library which is exploited by RS_Server to actually edit the PDF files to add the URL required when a reader requests to save for later the publication that he is reading, as outlined in Sect. 3.1. It is a software characterized by a command line interface and this means that RS_Server can use it directly since they are deployed one along the other, without using complex communication channels and paradigms.

RS_Rate [12] is an extension for *Google Chrome*[1] and the client that readers actually use to rate publications; this means that every interaction with RS_Server is carried out through this client. We intend to generalize RS_Rate by providing an implementation for each of the major browsers (i.e., Firefox and Safari); moreover, we intend to provide an implementation of a fully fledged web application and a mobile application for each of the major operating systems (i.e., iOS and Android).

3.3 Implementation and Technologies

RS_Server is developed in Ruby on Rails,[2] which is a framework that allows to build applications strongly based on the Model-View-Controller (MVC) architectural pattern.

RS_Server is a Web Service (Server API-Only, according to Rails terminology) based on a communication paradigm composed of RESTful (REpresentational

[1] https://www.google.com/chrome/.
[2] https://rubyonrails.org/.

State Transfer) interfaces and on the exchange of messages encoded in JSON format through the transport layer provided by the HTTP protocol.

The technology used to develop RS_PDF is the Kotlin object-oriented programming language, whose main feature is to be fully compatible with the Java Virtual Machine. This feature is of great importance because it allows a developer to exploit code contained in any other software published in jar format and, more generally, to import any Java class, interacting with them through the syntax of Kotlin itself.

We have chosen Kotlin because it has many modern features (it has been created just three years ago) and it is supported rather intensively; furthermore, there are openings to other platforms that have greatly expanded its use possibilities. The most important reason, however, is that the underlying tool used to actually edit files encoded in PDF format is PDFBox,[3] which is a software library developed with Java and proposed as a complete toolkit to edit files in that specific format. So, RS_PDF is a wrapper for PDFBox that adds the needed links inside the PDFs requested by readers.

RS_Rate is an extension for Google Chrome; those extensions are developed using standard web technologies such as HTML, CSS and Javascript. Therefore, they are simple "collections" of files packaged in a CRX archive. This particular format is nothing more than a modified version of a ZIP archive with the addition of some special headers exploited by Google Chrome.

As for the Javascript component, RS_Rate does not actually use the "pure" language but instead uses the jQuery library, to simplify the selection, manipulation, management of events and the animation of DOM elements in HTML pages, as well as the implementation of AJAX features, that are widely used by RS_Rate to improve the user experience during its use.

3.4 User Interface

In this section we describe Readersourcing 2.0 user interface and we provide some details about how a reader could interact with it. A technical documentation on the design details of the components of the system can be seen in [14].

Accordingly to the R1 requirement proposed in Sect. 3.1, which states that a reader should be able to seamlessly rate a publication with a low effort and a few clicks or keystrokes, we have chosen to characterize our client (Google Chrome extension) as a popup which appears in the page that the user is browsing, when he clicks its toolbar button.

Figure 2 shows a section of a Google Chrome instance with our client active as a popup for a publication. This is the typical situation of a reader visiting a publisher's web site to access the PDF of a paper he is interested in. The figure also shows the first page that a reader looks at when he interacts with the client itself, which is used to take him to the login page, which is shown in Fig. 3a, or to the sign up page. From the login page a reader who has forgotten his password can reach the password recovery page (not shown), very similar to the login one.

[3] https://pdfbox.apache.org/.

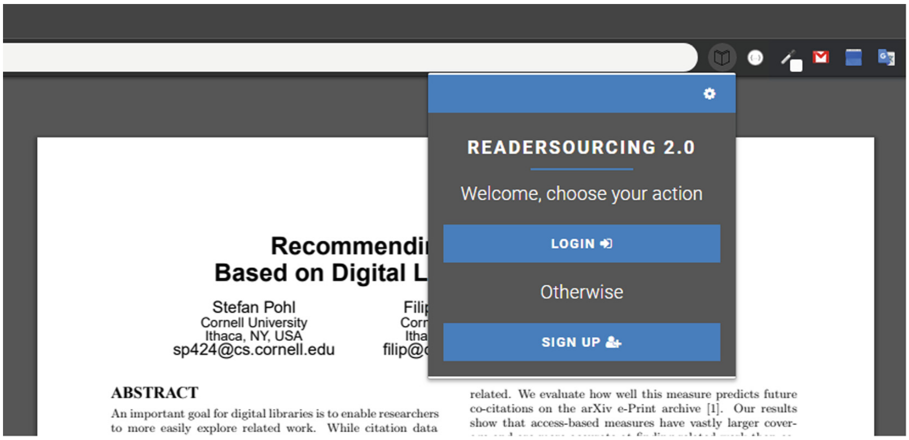
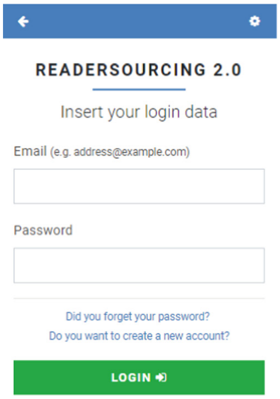

READERSOURCING 2.0
Welcome, choose your action
LOGIN
Otherwise
SIGN UP
Recommendi
Based on Digital L
Stefan Pohl
Cornell University
Ithaca, NY, USA
sp424@cs.cornell.edu
ABSTRACT
An important goal for digital libraries is to enable researchers
to more easily explore related work. While citation data
related. We evaluate how well this measure predicts future
co-citations on the arXiv e-Print archive [1]. Our results
show that access-based measures have vastly larger cover-

Fig. 2. RS_Rate as an Google Chrome extension characterized by a popup action.

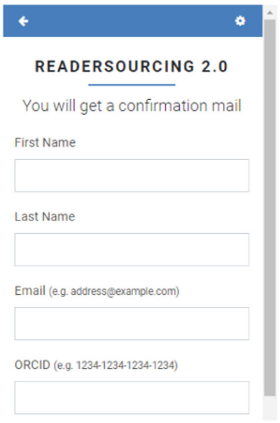
If a reader has still to sign up to Readersourcing 2.0 he can reach, from the page shown in Fig. 2, the one shown in Fig. 3b and fill in the sign up form. Once he completes those standard sign up and login operations, he will finally find himself into the rating page, which is shown in Fig. 3c, where the operations outlined in Sect. 3.1 can take place.

Regarding the requirement R1 outlined in Sect. 3.1, in the central section of the rating page a reader can use the slider to choose a rating value in a 0–100 interval, as suggested by Medo and Rushton Wakeling [7]. Once he selects the desired rating, he only needs to click the green *Rate* button and that will be all; with just three clicks and a slide action he can submit his rating. Furthermore, he can also click the options button and, if preferred, check an option to anonymize the rating he is about to provide. We remark that the reader has to be logged in to express an anonymous rating to prevent spamming, which in this case would be a very dangerous phenomenon. When such a rating is processed, the information regarding its reader will not be exploited (apart from avoiding the reader to rate the same publication multiple times).

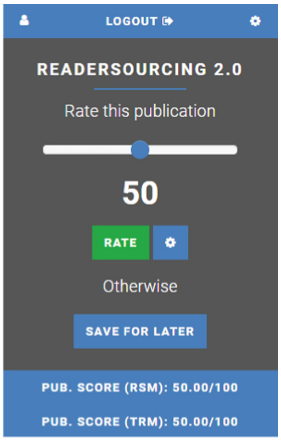
Regarding the requirement R2 outlined in Sect. 3.1, if the reader wants to provide his rating at a later time rather than directly rate the publication, he can instead click the *Save for later* button and take advantage of the editing procedure of publications that stores a reference (an URL link) inside the PDF file that he is looking at. As soon as such an editing procedure is completed (usually just a few seconds), the *Save for later* button becomes a *Download* button, as shown in Fig. 3d. The reader can finally download the link-annotated publication by clicking on it. Furthermore, he can also use the refresh button (on the right of the *Download* button) to, as it says, refresh the link-annotated publication. This means that a new copy of the publication file will be downloaded, annotated, and made available to the reader. This feature is useful since a publication could be updated at a later time by its author.


READERSOURCING 2.0
Insert your login data
Email (e.g. address@example.com)
Password
Did you forget your password?
Do you want to create a new account?
LOGIN

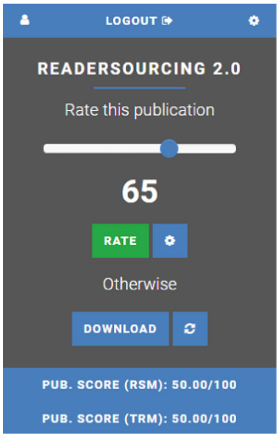
(a) Login page.


READERSOURCING 2.0
You will get a confirmation mail
First Name
Last Name
Email (e.g. address@example.com)
ORCID (e.g. 1234-1234-1234-1234)

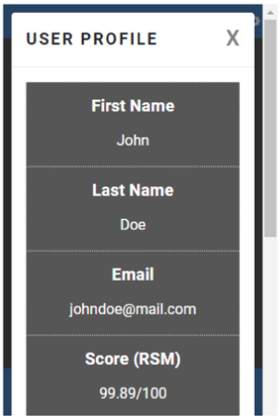
(b) Signup page.



(c) Rating page (before a save for later request).



(d) Rating page (after a save for later request).



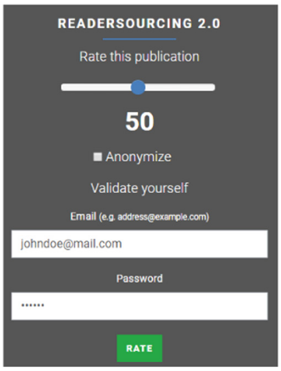
First Name
John

Last Name
Doe

Email
johndoe@mail.com

Score (RSM)
99.89/100

(e) Profile page with reader scores.



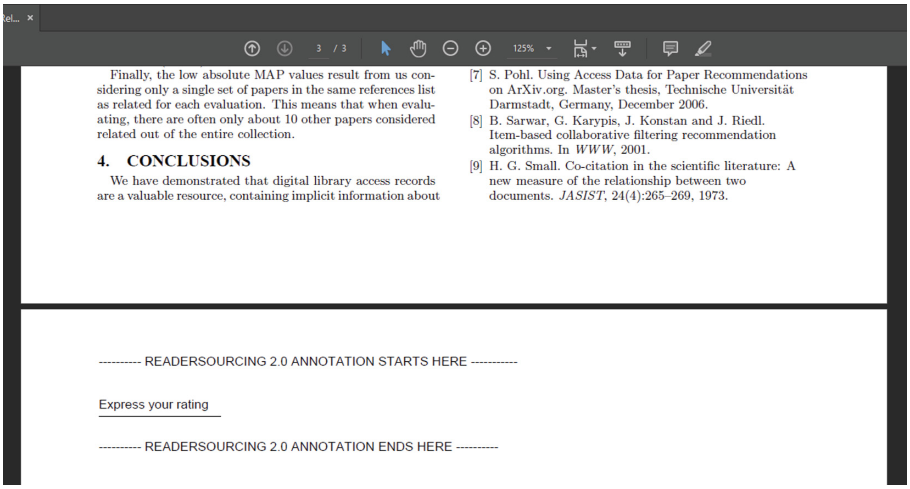
Email (e.g. address@example.com)
johndoe@mail.com
Password
••••••

(f) A server-side interface to rate a publication.

Fig. 3. The user interface of RS_Rate.

As soon as the link-annotated publication is downloaded, the reader will find a PDF containing a new final page with the URL. In Fig. 4 an example of such link-annotated publication can be seen; in that case, the reader has chosen to open it with his favourite PDF reader.

Once the reader clicks on the reference which, as outlined in Sect. 3.1, is a special link to RS_Server, he will be taken to the server-side application itself, which will show to him an interface that allows him to express his rating in a way that is independent from the browser extension used to actually store the reference. Therefore, if he sends his link-annotated publication to a tablet-like device, for example, he will take advantage of the built-in browser to express


Finally, the low absolute MAP values result from us considering only a single set of papers in the same references list as related for each evaluation. This means that when evaluating, there are often only about 10 other papers considered related out of the entire collection.

4. CONCLUSIONS

We have demonstrated that digital library access records are a valuable resource, containing implicit information about

[7] S. Pohl. Using Access Data for Paper Recommendations on ArXiv.org. Master's thesis, Technische Universität Darmstadt, Germany, December 2006.
[8] B. Sarwar, G. Karypis, J. Konstan and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.
[9] H. G. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *JASIST*, 24(4):265–269, 1973.

---------- READERSOURCING 2.0 ANNOTATION STARTS HERE ----------

Express your rating

---------- READERSOURCING 2.0 ANNOTATION ENDS HERE ----------

Fig. 4. A link-annotated publication.

his rating. Figure 3f shows the interface that the reader sees after a click on the stored reference. The reader is required to authenticate himself again as a form of security since, otherwise, the stored reference could be used by anyone who gets a copy of the link-annotated publication.

Regarding the requirement R3, every time a reader rates a publication every score is updated according to both RSM and TRM, and each reader can see the result through RS_Rate. In the bottom section of the rating page the score of the current publication can be seen (one for each model), as shown in Figs. 3c and d. To see his score as a reader (once again, one for each model), a user must click the profile button on the upper right corner. Once he does that, he will see the interface shown in Fig. 3e. From there, he could also edit his password since that interface acts as a profile page.

An overview of Readersourcing 2.0 capabilities is shown in Fig. 5. Let us suppose that there are four readers which are using RS_Rate to rate a publication P1, namely RD1, RD2, RD3 and RD4. Both RD1, RD2 and RD3 exploit the *Save for later* functionality of RS_Rate itself to express their rating at a later time. By doing this, they receive a link-annotated version of P1, namely P1+Link. After some time, RD1 chooses to open P1+Link with his favourite PDF reader. RD2, instead, chooses to send it to his iPad, while RD3 simply opens it with his instance of Google Chrome. When they click on the URL added by RS_Server, they are taken to the special page provided by RS_Server where they provide their rating. On the contrary, RD4 simply chooses to give his rating as soon as he finishes to read P1 directly through the interface of RS_Rate.

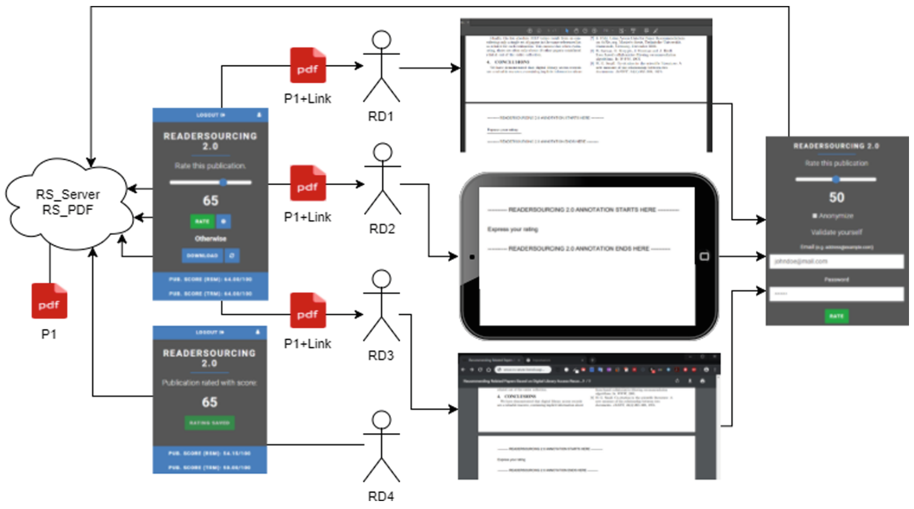


Fig. 5. Readers's interaction modalities with the Readersourcing 2.0 ecosystem

4 Conclusions and Future Work

Readersourcing 2.0 is still an early prototype and it must be considered as work in progress. During the next months we will keep improving it and what we have shown in the previous section about its appearance could still change. However, the overall architecture and the core functionalities will hardly be changed.

As one obvious future development from an architectural standpoint, we plan to take into account the other common browsers besides Google Chrome, and implement browser extensions for them. As a second step we also plan to implement some stand alone app for various architectures, to build a more complete ecosystem that allows the users to choose their preferred interaction modalities.

Once a more stable version is built it will be used to gather fresh data in terms of ratings of publications which will be analyzed and validated and, moreover, it will be used to study some specific topics that we briefly outline in the following. The results of those analysis will be published in a future work.

4.1 Review Quality

One of the biggest criticisms of an approach based on *Crowdsourcing* is that often the so called "workers" performing a tasks tend to do it hastily and approximately, without attention/motivation. Since the Readersourcing approach is nothing more than a particular type of Crowdsourcing, it is reasonable to analyze the question. In particular, to encourage reviewers to express quality ratings, in addition to the co-determination algorithms proposed by Mizzaro [9] and De Alfaro and Faella [5], there are techniques independent from the application domain that can be used.

One of such technique is studied by Dow et al. [6]. They focus on the benefits of user *shepherding*, namely guiding users through self-evaluation practices of the work carried out within the assigned task or through the evaluation of the work itself by external feedback. They compare two scenarios, one with shepherding and one without, and they analyze specific aspects of such practices (when to show feedback, how much detailed it should be, who should provide it, etc.) Once done that, they describe how self-assessment and evaluation carried out through external feedback lead workers to obtain different benefits and, in general, a higher quality outcome for the assigned task.

Finally, they ask themselves if workers are able to become shepherds for the remaining ones, suggesting that the more “expert” ones can assume that role. In the light of these results, it may be useful to add to our Readersourcing 2.0 ecosystem a self-assessment page of the rating that a reader has just expressed to obtain a higher quality of process.

Another well known technique to increase review quality is to establish awards and other forms of public recognition. We plan to evaluate this technique within a community of expert readers to see if it can improve the results obtained by our Readersourcing 2.0 ecosystem.

4.2 Single Blind vs. Double Blind Review

One of the various issues to be addressed in the context of a peer review process (even when it is outsourced to the readers of the publications) consists in showing (*single-blind review*) or not (*double-blind review*) the name of the author and his affiliation during the review phase of the publication. Tomkins et al. [17] analyze this question in two different scenarios.

In the first scenario, the reviewers are divided into two groups, one for the single-blind approach and one for the double-blind approach. Subsequently, each of them is asked to consider a set of publications and to state, one by one, if they intend to carry out the review, if they would consider the possibility or if they are not interested.

In the second scenario, the reviewers are asked to proceed with the review activity by giving a rating to each publication. The results of the experiments lead Tomkins et al. [17] to state that although many of the possible bias described by the detractors of the single-blind approach do not lead to statistically significant effects, the choice to show or not the names of the authors of a publication and their affiliation has effects on the behavior of the reviewers.

In particular, those who have access to such information tend to recommend acceptance of the publications of the most “famous” authors (both personally and in relation to their affiliation) compared to those who perform the review with the double-blind approach. According to Tomkins et al. [17], therefore, this is an aspect that must be taken into account during the definition of a peer review process. Since our process is an outsourced form of the standard peer review process itself, this is an interesting question to study.

4.3 Legal Issues

Regarding any legal issues related to the editing of proprietary PDF content, we hypothesize that it depends on where the PDF content has been published. There are many publishing systems and/or repositories where an author can distribute his papers under a Creative Commons license of his choice. Such a form of licensing allows to "remix, transform and build upon the material for any purpose" [1]; therefore, it should be possible to edit such PDF contents freely. However, there are publishers which have their own publishing and copyright licenses that must be studied on a case-by-case basis; therefore, the right approach within our system could be to allow the use of the "save for later" functionality of Readersourcing 2.0 outlined in Sect. 3.4 only within publishing systems which are safe from a legal viewpoint. This topic needs further investigation.

References

1. CC Attribution 4.0 International Public License (2010). https://creativecommons.org/licenses/by/4.0/legalcode
2. OpenReview (2016). https://openreview.net/
3. Akst, J.: I Hate Your Paper: many say the peer review system is broken. Here's how some journals are trying to fix it. Sci. **24**, 36 (2010). http://www.the-scientist.com/2010/8/1/36/1/
4. Arms, W.Y.: What are the alternatives to peer review? Quality control in scholarly publishing on the web. JEP **8**(1) (2002). https://doi.org/10.3998/3336451.0008.103
5. De Alfaro, L., Faella, M.: TrueReview: a platform for post-publication peer review. CoRR (2016). http://arxiv.org/abs/1608.07878
6. Dow, S., Kulkarni, A., Klemmer, S., Hartmann, B.: Shepherding the crowd yields better work. In: Proceedings of ACM 2012 CSCW, pp. 1013–1022. ACM (2012)
7. Medo, M., Rushton Wakeling, J.: The effect of discrete vs. continuous-valued ratings on reputation and ranking systems. EPL **91**(4), 48004 (2010). http://stacks.iop.org/0295-5075/91/i=4/a=48004
8. Meyer, B.: Fixing the process of computer science refereeing, October 2010. https://cacm.acm.org/blogs/blog-cacm/100030-fixing-the-process-of-computer-science-refereeing/fulltext
9. Mizzaro, S.: Quality control in scholarly publishing: a new proposal. JASIST **54**(11), 989–1005 (2003). https://doi.org/10.1002/asi.22668
10. Mizzaro, S.: Readersourcing - a manifesto. JASIST **63**(8), 1666–1672 (2012). https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.22668
11. Soprano, M., Mizzaro, S.: Readersourcing 2.0: RS_PDF, October 2018. https://doi.org/10.5281/zenodo.1442597
12. Soprano, M., Mizzaro, S.: Readersourcing 2.0: RS_Rate, October 2018. https://doi.org/10.5281/zenodo.1442599
13. Soprano, M., Mizzaro, S.: Readersourcing 2.0: RS_Server, October 2018. https://doi.org/10.5281/zenodo.1442630
14. Soprano, M., Mizzaro, S.: Readersourcing 2.0: technical documentation, October 2018. https://doi.org/10.5281/zenodo.1443371
15. Sun, Y., et al.: Crowdsourcing information extraction for biomedical systematic reviews. CoRR abs/1609.01017 (2016)

16. The Economist: Some science journals that claim to peer review papers do not do so (2018). https://www.economist.com/science-and-technology/2018/06/23/some-science-journals-that-claim-to-peer-review-papers-do-not-do-so
17. Tomkins, A., Zhang, M., Heavlin, W.D.: Reviewer bias in single- versus double-blind peer review. PNAS **114**(48), 12708–12713 (2017). http://www.pnas.org/content/114/48/12708