1 Introduction

Institutional and multidisciplinary open access repositories play a crucial role in knowledge transfer by enabling immediate accessibility to all kinds of research output. One important element still missing from open access repositories, however, is a quantitative assessment of the hosted research items that will facilitate the process of selecting the most relevant and distinguished content. Common currently available metrics, such as number of visits and downloads, do not reflect the quality of a research product, which can only be assessed directly by peers offering their expert opinion together with quantitative ratings based on specific criteria.

To address this issue we develop an open peer review module as an overlay service to existing institutional or other repositories. Digital research works hosted in repositories using our module can be evaluated by an unlimited number of peers that offer not only a qualitative assessment in the form of text, but also quantitative measures to build the work’s reputation. Crucially, our open peer review module also includes a reviewer reputation system based on the assessment of reviews themselves, both by the community of users and by other peer reviewers. This allows for a sophisticated scaling of the importance of each review on the overall assessment of a research work, based on the reputation of the reviewer.

The details of the model to quantify reputation for papers, authors, reviewers, and reviews are presented below.

2 Data and Notation

In order to compute reputation values for papers, authors, reviewers, and reviews we need to extract a Reputation Data Set from each paper repository to be included in the system.

Definition 2.1 (Data). A Reputation data Set is a tuple \( \langle P, R, E, D, a, o, v \rangle \), where

- \( P = \{ p_i \}_{i \in P} \) is a set of papers (e.g. DOIs).
- \( R = \{ r_j \}_{j \in R} \) is a set of researcher names or identifiers (e.g. the ORCHID identifier).
- \( E = \{ e_i \}_{i \in E} \cup \{ \bot \} \) is a totally ordered evaluation space, where \( e_i \in \mathbb{N} \setminus \{0\} \) and \( e_i < e_j \) iff \( i < j \) and \( \bot \) stands for the absence of evaluation. We suggest the range \([0,100]\).
- \( D = \{ d_k \}_{k \in K} \) is a set of evaluation dimensions.
- \( a : P \rightarrow 2^R \) is a function giving the names/identifiers of researchers authoring a paper.
- \( o : R \times P \times D \times Time \rightarrow E \), where \( o(r, p, d, t) \in E \) is a function giving the opinion of a reviewer, as a value in \( E \), on a dimension \( d \) of a paper \( p \) at a given instant of time \( t \).\(^2\)

\(^1\)If there are different versions of papers this set represents the latest versions.
\(^2\)If there are different versions of the reviews this represents the last one.
• $v : R \times R \times P \times \text{Time} \rightarrow E$, where $v(r, r', p, t) = e$ is a function giving the judgment of researcher $r$ over the opinion of researcher $r'$, on paper $p$ as a value $e \in E$. Therefore, a judgment is a reviewer’s opinion about another reviewer’s opinion.

2.1 Limitations of our approach

• We do not consider paper versions (for now). We just use the latest one.

• We do not consider review versions. We just use the latest one.

2.2 Notation

We will not include the dimension (or the criteria being evaluated, such as originality, soundness, etc.) in the equations to simplify the notation. There are no interactions among dimensions so the set of equations apply to each of the dimensions under evaluation.

Also, we will also omit the reference to time in all the equations. Time is essential as all measures are dynamic and thus they evolve along time. We will make the simplifying assumption that all opinions and judgments are maintained in time, that is, they are not modified. Including time would not change the essence of the equations, it will simply make the computation complexity heavier.

3 Reputation of a Paper

We say the reputation of a paper, for a given dimension (omitted here), is a weighted aggregation of its reviews, where the weight is the reputation of the reviewer (Section 5).

$$R_{p}(p) = \begin{cases} \sum_{\forall r \in \text{rev}(p)} R_{R}(r) \cdot o(r, p) & \text{if } |\text{rev}(p)| \geq k \\ \sum_{\forall r \in \text{rev}(p)} R_{R}(r) \perp & \text{otherwise} \end{cases}$$

where $\text{rev}(p) = \{ r \in R | o(r, p) \neq \perp \land R_{R}(r) \neq \perp \}$ denotes the reviewers of a given paper.

Note that when a paper receives less that $k$ reviews, its reputation will not depend on the reviews, but it will inherit the reputation of its best (or most reputable) author.

We currently leave $k$ as a parameter, though we suggest that $k > 1$, so that the reputation of a paper is not dependent on a single review. However, we recommend small numbers for $k$, such as 2 or 3, because we believe it is usually difficult to obtain reviews. As such, new papers can quickly start having a reputation that is not dependent on their authors’ reputation.

4 Reputation of an Author

We consider that a researcher’s author reputation, on a given dimension omitted here, is an aggregation of the reputation of her papers.

The aggregation is based on the concept that the impact of a paper’s reputation on its authors’ reputation is inversely proportional to the total number of its authors. In other words, if one researcher is the sole author of a paper, then this author is the only person responsible for this paper, and any (positive or negative) feedback about this paper is propagated as is to its sole author. However, if the researcher has coauthored the paper with several other researchers, then the impact (whether positive

\[3\] In tools like ConfMaster (www.confmaster.net) this information could be gathered by simply adding a private question to each paper review, answered with elements in $E$, one value in $E$ for the judgment on each fellow reviewer’s review.
or negative) that this paper has on the researcher decreases with the increasing number of coauthors. We argue that collaborating with different researchers usually increases the quality of a research work since the combined expertise of more than one researcher is always better than the expertise of a single researcher. Nevertheless, the gain in a researcher’s reputation decreases as the number of coauthors increase. Hence, our model might cause researchers to be more careful when selecting their collaborators, since they should aim at increasing the quality of the papers they produce in such a way that the gain for each author is still larger than the gain it could have received if it was to work on the same research problem on her own.

We say the similarity between two opinions is the difference between the two:

\[ \text{Sim}(\bar{o}(r_i,p), \bar{o}(r_j,p)) = \begin{cases} v(r_i,r_j,p) & \text{if } v(r_i,r_j,p) \neq \bot \\ \text{Sim}(\bar{o}(r_i,p), \bar{o}(r_j,p)) & \text{if } \bar{o}(r_i,p) \neq \bot \text{ and } \bar{o}(r_j,p) \neq \bot \\ \bot & \text{Otherwise} \end{cases} \]

where \( \bar{o}(r_i,p) \) denotes the extended judgment of \( r_i \) over \( p \), and \( v(r_i,r_j,p) \) denotes the similarity of \( r_i \) and \( r_j \) over \( p \). Note that the previous equation does not restrict judgments to be given by reviewers of the paper. Any one can judge a review without having expressed an opinion/review on the paper. Thus, this is coherent with an open review model.

We say the similarity between two opinions is the difference between the two: \( \text{Sim}(\bar{o}(r_i,p), \bar{o}(r_j,p)) = 100 - |\bar{o}(r_i,p) - \bar{o}(r_j,p)| \).

Given this, we consider that the overall opinion of a researcher on the capacity of another researcher to make good reviews is calculated as follows. Consider the set of judgments of \( r_i \) over reviews made by \( r_j \) as: \( V^*(r_i,r_j) = \{ v^*(r_i,r_j,p) | v(r_i,r_j,p) \neq \bot \text{ and } p \in P \} \). This set might be empty. Then, define the judgment of a reviewer over another one as a simple average:

\[ V^*(r_i,r_j) = \frac{v^*(r_i,r_j)}{|V^*(r_i,r_j)|} \]

5 Reputation of a Reviewer

Similar to the reputation of authors (Section 4), we consider that if a reviewer produces ‘good’ reviews, then the reviewer is considered to be a ‘reputed’ reviewer. Furthermore, we consider that the reputation of a reviewer is essentially an aggregation of the opinions over her reviews.

\[ R_A(r) = \begin{cases} \sum_{p \in \text{pap}(r)} \gamma(p) \times R_P(p) + (1 - \gamma(p)) \times 50 & \text{if } \text{pap}(r) \neq \emptyset \\ \bot & \text{otherwise} \end{cases} \]

where \( \text{pap}(r) = \{ p \in P | r \in a(p) \land R_P(p) \neq \bot \} \) denotes the papers authored by a given researcher \( r \), 50 represents complete ignorance, \( \gamma(p) = \frac{1}{a(p)} \) is the coefficient that takes into consideration the number of authors of a paper (recall that \( a(p) \) denotes the authors of a paper \( p \)), and \( \alpha \) is a tuning factor that controls the rate of decrease of the \( \gamma(p) \) coefficient.
Finally, the reputation of a reviewer, $R_r(r)$, is an aggregation of judgments that her colleagues make about her capability to produce good reviews. We weight this with the reputation of the colleagues as a reviewer:

$$R_R(r) = \begin{cases} \sum_{\forall r_i \in R^*} R_R(r_i) \cdot R_R(r_i, r) & R^* \neq \emptyset \\ 50 & \text{otherwise} \end{cases}$$

(5)

where $R^* = \{r_i \in R | V^*(r_i, r) \neq \emptyset \}$. When no judgments have been made over $r$ we take his/her reputation as author as its default reputation as reviewer.

Note that the reputation of a reviewer depends on the reputation of other reviewers. In other words, every time the reputation of one reviewer will change, it will trigger changing the reputation of other reviewers, which might lead to an infinite loop of modifying the reputation of reviewers. We address this by using an algorithm similar to the EigenTrust algorithm, as illustrated by Algorithm 5. In fact, this algorithm may be considered as a variation of the EigenTrust algorithm, which will require some testing to confirm how fast it converges.

6 Reputation of a Review

The reputation of a review is similar to the one for papers but using judgments instead of opinions. We say the reputation of a review, for a given dimension omitted here, is a weighted aggregation of its judgments, where the weight is the reputation of the reviewer (Section 5).

The reputation of a review makes sense when we allow for reviews to be open. However, we do not want that judgments on reviews made by researchers without a reputation have influence in the final reputation. The following equation accounts for this.

$$R_O(r', p) = \begin{cases} \sum_{\forall r \in \text{jud}(r', p)} R_R(r) \cdot v^*(r, r', p) & |\text{jud}(r', p)| \geq k \\ \sum_{\forall r \in \text{jud}(r', p)} R_R(r) & \text{otherwise} \end{cases}$$

(6)

where $\text{jud}(r', p) = \{r \in R | v^*(r, r', p) \neq \bot \land R_R(r) \neq \bot \}$ denotes the set of judges of a particular review written by $r'$ on a given paper $p$.

Note that when a review receives less than $k$ judgments, its reputation will not depend on the judgments, but it will inherit the reputation of the author of the review.

We currently leave $k$ as a parameter, though we suggest that $k > 1$, so that the reputation of a review is not dependent on a single judge. However, we recommend small numbers for $k$, such as 2 or 3, because we believe it will be difficult to obtain large numbers of judgments.

7 A Note on Dependencies

Figure 1 shows the dependencies between the different measures (reputation measures, opinions, and judgments). The decision of When to re-calculate those measures is then based on those dependencies.
We provide a summary of this below. Note that measures in white are not calculated, but provided by the users. As such, we only discuss those in grey (grey rectangles represent reputation measures, whereas the grey oval represents the extended judgments).

- **Author’s Reputation.** The reputation of the author depends on the reputation of its papers (Equation 2). As such, every time the reputation of one of his papers changes, or every time a new paper is created, the reputation of the author is recalculated (Algorithm 2).

- **Paper’s Reputation.** The reputation of the paper depends on the opinions it receives, and the reputation of the reviewers giving those opinions (Equation 1). As such, every time a paper receives a new opinion, or every time the reputation of one of the reviewers changes, then the reputation of the paper is recalculated (Algorithm 1).

- **Review’s Reputation.** The reputation of a review depends on the extended judgments it receives, and the reputation of the reviewers giving those judgments (Equation 6). As such, every time a review receives a new extended judgments, or every time the reputation of one of the reviewers changes, then the reputation of the review is recalculated (Algorithm 6).

- **Reviewer’s Reputation.** The reputation of a reviewer depends on the extended judgments of other reviewers and their reputation (Equation 5). As such, the reputation of the reviewer should be modified every time there is a new extended judgment or the reputation of one of the reviewers changes. As the reputation of a reviewer depends on the reputation of reviewers, then we suggest to calculate the reputation of all reviewers repeatedly (in a manner similar to EigenTrust) in order to converge (Algorithm 5). If this will be computationally expensive, then this can be computed once a day, as opposed to triggered by extended judgments and the change in reviewers’ reputation.

- **x-judgment.** The extended judgment is calculated either based on judgments (if available) or the similarity between opinions (when judgments are not available) (Equation 3). As such, the extended judgment should be recalculated every time a new (direct) judgment is made, or every time a new opinion is added on a paper which already has opinions by other reviewers (Algorithm 3).

![Figure 1: Dependencies](image)
8 Algorithms

Algorithm 1: Reputation of a paper

Function \texttt{ReputationPaper}(p : P;[0,100]) :=
\begin{itemize}
  \item \textbf{Data:} \texttt{p : P} /* a paper identifier */
  \item \textbf{Data:} \texttt{aut : P \to R list} /* function returning the list of authors of papers */
  \item \textbf{Data:} \texttt{o : (R \times E) list} /* list of evaluations of reviewers over paper \texttt{p} */
  \item \textbf{Data:} \texttt{k : integer} /* minimum number of reviewers to compute non-default reputation \texttt{k} > 1 */
  \item \textbf{Result:} \texttt{RepPaper : [0,100]} /* the reputation value of paper \texttt{p} */
\end{itemize}

/* This function computes the reputation of a paper for a given dimension. It must be called every time a new review is created for this paper, and every time the reputation of one of the paper’s reviewers is modified. */

\begin{verbatim}
rev = \emptyset;
for (r, e) \in o do
    if R_R(r) \neq null then
        rev = rev \cup (r, e);
    end
end
if length(rev) < k then
    RepPaper \leftarrow \texttt{null};
else
    normal \leftarrow 0;
    for (r, e) \in rev do
        normal \leftarrow normal + ReputaitonReviewer(r);
    end
    num \leftarrow 0.0;
    for (r, e) \in rev do
        num \leftarrow num + ReputaitonReviewer(r) \ast e;
    end
    RepPaper \leftarrow num/normal;
end
return RepPaper;
\end{verbatim}
Algorithm 2: Reputation of an author

Function ReputationAuthor(r : R):[0,100] =

Data: r : R /* a researcher identifier */
Data: pap : R → P list /* function returning the list of papers of authors */
Data: aut : P → R list /* function returning the list of authors of papers */
Data: alpha : real /* tuning factor for coefficient gamma */
Result: RepAuthor : [0,100] /* the reputation value of author r */

/* This function computes the reputation of an author. It must be called
every time a new paper is created for this author, and every time the
reputation of one of the author’s papers is modified. */
pap2 = ∅;
for p ∈ pap(r) do
    if RP(p) ≠ null then
        pap2 = pap2 ∪ p;
    end
end
num ← 0.0;
if pap2 ≠ ∅ then
    for p ∈ pap2 do
        gamma ← 1/length(aut(p)) /* length gives the length of a list */
        num ←
        num + exp(gamma, alpha) * ReputationPaper(p) + (1 - exp(gamma, alpha)) * 50
    end
    RepAuthor = num/|pap2|;
else
    RepAuthor = null
end
return RepAuthor
Algorithm 3: Auxiliary functions, used by Algorithms 5 and 6

Function \( v^*(r_i, r_j, p) : [0,100] + \text{null} = \)
| Data: \( r_i, r_j : R \) /* researcher identifiers */ |
| Data: \( p : P \) /* a paper identifier */ |
| Data: \( o_{bar} : (R \times E^k) \) list /* list of vector evaluations of reviewers over \( p \) */ |
| Data: \( v : (R \times R \times E) \) list /* list of judgments over paper \( p \) */ |
| Result: \( \text{extjudge} : [0,100] + \text{null} /* extended judgment of \( r_i \) on \( r_j \)'s opinion of \( p \) */ |

/* This function computes extended judgments. It must be called every time a new judgment is made, and every time a new review is added on a paper which already has reviews by others. It is also called by the \( \text{AverageJudgment} \) function below and the \( \text{ReputationReview} \) function of Algorithm 6. */

if \( \exists e : (r_i, r_j, e) \in v \) then
| \( \text{extjudge} \leftarrow e \)
else
| if \( \exists e_{bar}, e_{bar}' : (r_i, e_{bar}) \in o_{bar} \) and \( (r_j, e_{bar}') \in o_{bar} \) then
| \( \text{extjudge} \leftarrow \text{sim}(e_{bar}, e_{bar}') \)
else
| \( \text{extjudge} \leftarrow \text{null} \)
end
end
return \( \text{extjudge} \)

Function \( \text{sim}(e, e' : E^k) : [0,100] = \)
| Data: \( e, e' : E^k \) /* evaluation vectors */ |
| Result: \( \text{similar} : [0,100] + \text{null} /* difference */ |

/* This function computes the similarity between two vectors. It is only called by the \( v^* \) function above. */

\( \text{num} \leftarrow 0; \)
\( \text{num}' \leftarrow 0; \)
\( \text{den} \leftarrow 0; \)
\( \text{den}' \leftarrow 0; \)
for \( i \in [1, k] \) do
    if \( e[i] \neq \text{null} \) then
        \( \text{num} \leftarrow \text{num} + e[i]; \)
        \( \text{den} \leftarrow \text{den} + 1; \)
    end
    if \( e'[i] \neq \text{null} \) then
        \( \text{num}' \leftarrow \text{num}' + e'[i]; \)
        \( \text{den}' \leftarrow \text{den}' + 1; \)
    end
end
if \( \text{den} \neq 0 \) and \( \text{den}' \neq 0 \) then
    \( x \leftarrow \text{num}/\text{den}; \)
    \( x' \leftarrow \text{num}'/\text{den}'; \)
    \( \text{similar} \leftarrow 100 - |x - x'|; \)
end
\( \text{similar} \leftarrow \text{null}; \)
return \( \text{similar} \)
Algorithm 4: Auxiliary functions, used by Algorithms 5 and 6 (CONTINUED)

Function \textit{AverageJudgment}(r : R, r' : R) : [0,100] + null =

Data: \( r : R, r' : R \) /* two research identifiers */
Result: \( \text{AvgJudge} : [0,100] + \text{null} \) /* the average judgment of \( r \) over \( r' \)'s opinions */

/* This function computes the average judgment of one reviewer over another. It is only called by the \textit{ReputationReviewer} function of Algorithm 5. */

\begin{algorithmic}
\State \( \text{judgements} \leftarrow 0.0; \)
\State \( \text{num} \leftarrow 0.0; \)
\For {\( p \in P \)}
\State \If {\( v^*(r, r', p) \neq \text{null} \)}
\State \( \text{judgements} \leftarrow \text{judgements} + 1; \)
\State \( \text{num} \leftarrow \text{num} + v^*(r, r', p) \)
\EndIf
\EndFor
\If {\( \text{judgements} \neq 0.0 \)}
\State \( \text{AvgJudge} \leftarrow \text{num}/\text{judgements} \)
\Else
\State \( \text{AvgJudge} \leftarrow \text{null} \)
\EndIf
\Return \( \text{AvgJudge} \)
\end{algorithmic}
Algorithm 5: Reputation of a reviewer

**Function** ReputationReviewer($r : R$): [0,100] =

- **Data:** $r : R$ /* a researcher identifier */
- **Data:** $RepReviewer(r) : [0,100]$ /* the reputation value of author $r$ */

- **Result:** $RepReviewer(r) : [0,100]$ /* the new reputation value of author $r$ */

/* This function computes the reputation of a single reviewer. It is only called by the function ReputationReviewers and itself, ReputationReviewer. */

```
den ← 0.0;
num ← 0.0;
for $r' \in R, r' \neq r$ do
    if $AverageJudgment(r', r) \neq \text{null}$ then
        den ← den + $RepReviewer(r')$;
        num ← num + $RepReviewer(r') \times AverageJudgment(r', r)$;
    end
end
if den > 0.0 then
    $RepReviewer(r) ← num/den$;
else
    $RepReviewer(r) ← 50$;
end
return $RepReviewer(r)$;
```

**Function** ReputationReviewers : [0,100] list =

- **Data:** $\epsilon : [0,100]$ /* a threshold specifying when is the difference in reputation value considered negligible */
- **Data:** $r : R$ /* a researcher identifier */
- **Data:** $RepReviewer(r) : [0,100]$ /* the reputation value of an author $r$ in $R$. Initially it is set to $RepReviewer(r) = \text{null}$ */

- **Result:** $RepReviewers : [0,100]$ list /* returns the list of updated reputation value for all authors $r$ in $R$; that is, $RepReviewers = \{RepReviewer(r)\}_{r \in R}$ */

/* This function computes the reputation of all reviewers. It must be called every time an extended judgment over an opinion of $r$ is created or modified (calculated by the function $v^*$ of Algorithm 3). Alternatively, this might be called once a day. */

```
repeat ← true;
$RepReviewers ← \{RepReviewer(r)\}_{r \in R}$;
while repeat \neq false do
    repeat ← false;
    for $r \in R$ do
        $RepReviewer(r)_{OLD} ← RepReviewer(r)$;
        $RepReviewers ← \{RepReviewers − RepReviewer(r)\} \cup ReputationReviewer(r)$;
        if $|RepReviewer(r)_{OLD} − RepReviewer(r)| > \epsilon$ then
            repeat ← true;
        end
    end
end
return $RepReviewers$;
```
Algorithm 6: Reputation of a review

Function ReputationReview(r : R, p : P, k : integer):[0,100] =

Data: r : R /* a researcher identifier */
Data: p : P /* a paper identifier */
Data: k : integer /* minimum number of judgments to compute non-default reputation review value, k > 0 */
Result: RepReview : [0,100] /* the reputation value of the review of r over p */

/* This function computes the reputation of a particular review. It must be called every time an extended judgment over that opinion of r is created of modified (calculated by the function v* of Algorithm 3), and every time the reputation of the author of the review is modified. */

jud = ∅;
for r' ∈ R, r' ≠ r do
  if v*(r', r, p) ≠ null ∧ R_R(r') ≠ null then
    jud = jud ∪ r';
  end
end

den ← 0.0;
um ← 0.0;
if jud ≠ ∅ then
  for r' ∈ jud do
    den ← den + ReputationReviewer(r');
    num ← num + ReputationReviewer(r') * v*(r', r, p)
  end
  RepReview ← num/den;
else
  RepReview ← ReputationReviewer(r)
end
return RepReview;