Evaluating the Trustworthiness of Contributors in a Collaborative Environment

Cam Tu Phan Le, Frédéric Cuppens, Nora Cuppens, and Patrick Maillé

Institut TELECOM ; TELECOM Bretagne
2, rue de la Châtaigneraie CS 17607
35576 Cesson Sévigné Cedex, France
firstname.lastname@telecom-bretagne.eu

Abstract. We propose a method to evaluate the contributions of each participant to the development of a document in a collaborative environment. The algorithm proceeds ex post, by analyzing the different steps that led to the final (assumed satisfying) version of the document. Such an evaluation might be considered as a trust or reputation note, and therefore can be used as an input for trust mechanisms aimed at incentivizing users to contribute efficiently.

We implemented this evaluation mechanism in Java, when the document has been developed with Subversion, a version control system used to maintain current and former versions of files.

Keywords: Collaborative work, trust.

1 Introduction

There are more and more digital documents that cannot be elaborated by a single person or entity, because of the prohibitive size of the document, or of the numerous knowledge fields and competences that they require. Thus a whole community or organization is often needed to build a complete document of satisfying quality. For example, online encyclopedies such as Wikipedia [www.wikipedia.org] need contributions from a huge number of persons. Likewise, requests to calls for proposals in industry often imply several (possibly competing) companies joining their efforts in order to build an offer that fulfills the client's needs.

In those contexts, a reliability or trustworthiness evaluation of contributors would be of great help to decide the treatment applied to the contribution: if a participant is known to provide high quality contributions, then a minimum checking might be needed, whereas contributions from untrustworthy participants should be carefully checked or simply ignored. Since deeply checking contributions is necessarily costly, trust or reputation scores of their authors would help improve the document building process.

Moreover, such trust scores could also act as an incentive for participants to perform high quality contributions. Indeed, reputation scores can for example be

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 451–460, 2009.

used as inputs for access control or usage control policies [3], or even for resource allocation mechanisms (e.g. to share revenues among contributors).

While the notion of trust in networks has recently received quite a lot of attention for peer-to-peer networks (see [4,6] and references therein) or to build social networks [2,5], research on trust mechanisms in collaborative frameworks is only emerging.

In this paper, we propose an objective evaluation scheme for the contributors of a document, that can be used as a trust or reputation score. For sake of simplicity, we restrict our attention to a text document. We assume that the document has reached a stable (final) version, that has been validated, and propose a method to perform trust score calculations.

The remainder of the paper is organized as follows. Section 2 introduces the model considered in terms of participants roles and document development process. Then Section 3 describes our proposal of trust measure. Finally, Section 4 presents our implementation for trust computation, and conclusions and directions for future work are given in Section 5.

2 Model

We present here the assumptions we make regarding the development of the text document. We define three types of agents involved in the development process, and describe that process.

2.1 Roles

We assume that three roles are defined for the elaboration of the document, as described below.

Writers. Those participants have access to the current version of the document. They can add text but cannot delete text written by the other participants. Moreover, they do not see the text that has been deleted (more precisely, proposed for deletion) by reviewers. We denote by W the set of writers.

Reviewers. They can read the current version of the document, and see the parts that have been proposed for deletion by the other reviewers. The possible actions for them are text addition, and proposition of text deletion for parts of the document that have or not already been proposed for deletion. A reviewer can contradict a previous proposition of another reviewer, therefore modifying the text visible to writers. The set of reviewers is denoted by R.

Validator. He sees all the text, including the one that has been proposed for deletion. He can add and delete text to the document, and agree with or contradict deletion propositions of reviewers. In this paper we assume that only one entity is the validator. He is the participant who stops the development process: his choices are definitive and the corresponding version is the final one, that is assumed to be of good quality. Therefore the validator should be the guarantor

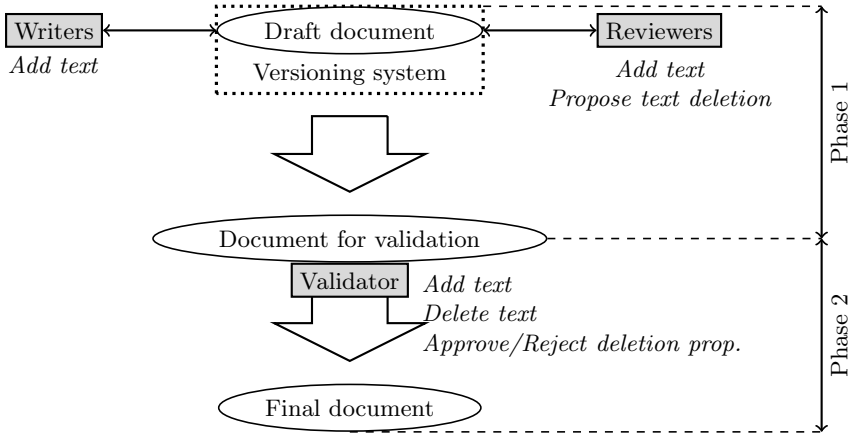

Fig. 1. The document development process, with action rights for each role (in italics)

of the quality of the document, and the reference as regards the trust scores described in the next section.

Notice that we do not enter the rules that determine those roles. The rule to decide whether a participant is a writer or a reviewer may for example take into account the involvement of the contributor in the project (if any), and the trust scores obtained from previous experience and/or recommendation processes [2].

2.2 Document Development

The document development process consists in two phases. First, writers and reviewers work on the document, according to their access rights defined previously. Some collaborative working tools such as versioning softwares can classically be used to manage the evolution of the document. In the second phase, the validator takes the actions that he considers necessary for the document to be of sufficient quality: text deletion, confirmation/cancelling of deletion propositions, and possibly text addition.

The process is illustrated in Figure 1, where actors appear in gray.

If the contributors in the first phase are trustworthy, then the work of the validator should be minimal.

3 Trust Calculation: Algorithm and Implementation

We introduce here a proposal of trust score based on objective measures (still assuming the final validated version is of best quality). We first explain the general principles that we want to apply, and present and justify the mathematical expressions of trust scores for writers and reviewers.

As in most references (e.g. [2,6] and references therein), a trust score will be a real number in the interval $[0, 1]$, the value 0 meaning that the participant has

no positive effect on the document development, and the value 1 corresponding to a perfectly trustable participant.

3.1 Principles for the Trust Score Definition

In this paper, we quantify contributions in function of their number of words. This measure is surely imperfect, since changing very few words can completely modify the meaning of the document. A measure based on the signification of contributions would be much better suited, but would involve the use of semantic analysis tools, which is beyond the scope of this paper. Notice that our trust calculation proposals can easily be modified to include such semantic-based measures. Nevertheless we use here the number of words to fix ideas and give concrete examples.

In all this paper, the validator is the reference for trust scores. He is therefore always given a fixed trust score of 1. Therefore we focus here on the trust score definition for writers and reviewers.

We believe that trust scores should respect the following principles.

P1. The trust score of a participant should be increasing with his contribution to the final version. In other words, if a significant part of the validated document comes from his contributions then his trust score should increase.
P2. The trust score of a participant should be decreasing with the proportion of his contributions that have not been kept in the final version.
P3. The trust score of a reviewer should be increasing with his contribution to the deletion propositions that have been validated.
P4. The trust score of a reviewer should be decreasing with the proportion of his deletion propositions that have not been validated.

We actually compute a numerical measure corresponding to each of those principles, and define the trust score as a weighted sum of those measures.

3.2 Trust Score Components

The score associated to Principle P1 should measure the *quantity* of his work with respect to the overall document content. It should answer the question *"How much did this participant contribute to the final version?"*. We therefore simply define it for each participant $i \in W \cup R$ as

$$t_{i,w\_qt} := \frac{n_{i,final}}{n_{final}}, \tag{1}$$

where $n_{i,final}$ is the number of words validated in the final version that come from participant i, and n_{final} is the total number of words in the final document. (w_qt stands for "writing quantity" of the contributions.)

Principle P2 aims to refer to the *quality* of the contributions of participant i, by answering the question *"Were the contributions of i satisfying?"*. The numerical

measure we take to answer this question is the proportion t_{i,w_ql} of the words written by i that were validated in the final version.

$$t_{i,w\_ql} := \frac{n_{i,final}}{n_i}, \tag{2}$$

where n_i is the total number of words that participant i has introduced to the document.

While Principles P1 and P2 respectively correspond to the quantity and quality of a participant's writing behavior, the two other principles should have the same meaning as concerns the deleting behavior. Since only reviewer have the right to propose text deletions, the corresponding scores only apply to participants $i \in R$.

If we denote by n_{i,del_prop} the number of words that reviewer i proposed for deletion, and by n_{del_val} the total number of words that were actually deleted in the final version, then we define the numerical measure t_{i,d_qt} associated to Principle P3 as

$$t_{i,d\_qt} := \frac{n_{i,del\_prop}}{n_{del\_val}}. \tag{3}$$

This ratio answers the question *"Does reviewer i delete low-quality text?"*, and reflects the *quantity* of his deletion proposition work.

On the other hand, the *quality* of his deletion proposition work can be evaluated by his degree of accordance with the validator as concerns deletions. This corresponds to Principle P4 and the question *"Does reviewer i delete only low-quality text?"*, which we quantify by defining t_{i,d_ql} as

$$t_{i,d\_ql} := \frac{n_{i,del\_prop}}{n_{i,del\_val}}, \tag{4}$$

where n_{i,del_val} is the number of words that have been proposed for deletion by i, and have effectively been deleted in the final version.

3.3 Definition of Trust Scores

We now propose an overall trust value expression for participants in $W \cup R$, as a weighted sum of the different trust score components defined in the previous subsection.

Writers Trust Score. For a writer $i \in W$, the system designer (possibly the validator) should decide which of Principles P1 or P2 are most important. In other words, he should choose whether to prefer to favor writers who contribute a lot, or those whose contributions are of high quality.

We propose to define the overall trust t_i of writer $i \in W$ as

$$t_i := \alpha_{qt} t_{i,w\_qt} + \alpha_{ql} t_{i,w\_ql}, \tag{5}$$

where α_{qt} and α_{ql} are two positive numbers, with $\alpha_{qt} + \alpha_{ql} = 1$, reflecting the system designer's preferences.

Reviewers Trust Score. For reviewers, we propose the same type of formula, but encompassing Principles P1-P4.

We define the trust score t_i of a reviewer $i \in R$ as

$$t_i := \beta_{w\_qt} t_{i,w\_qt} + \beta_{w\_ql} t_{i,w\_ql} + \beta_{d\_qt} t_{i,d\_qt} + \beta_{d\_ql} t_{i,d\_ql}, \tag{6}$$

where the βs are positive real numbers that sum to 1, and which represent the system designer's priorities in terms of quantity and quality (regarding writing and deletion) in reviewer's work. It would be natural that the weights associated to the writing behaviour have the same relative importance as for writers, i.e. that $\frac{\beta_{w\_qt}}{\beta_{w\_qt}+\beta_{w\_ql}} = \alpha_{qt}$, and $\frac{\beta_{w\_ql}}{\beta_{w\_qt}+\beta_{w\_ql}} = \alpha_{ql}$.

4 Algorithm and Implementation

We intend here to automate the computation on trust scores, based on the history of the versions that have been submitted to the versioning server. We have used Subversion [1] as a versioning tool, and implemented the trust score calculation in Java.

The algorithm proceeds backwards, and compares each version with the previous one using the Subversion command `svn diff`.

4.1 The Command `Svn Diff`

This command provides three kinds of results, depending whether text has been added "`a`", deleted "`d`" or cut "`c`". As an example, consider the two successive versions of a file given in Figure 2, where lines are numbered.

Then the command `svn diff` would give the result file of Figure 3.

The results interpret as follows:

- `1d0`: line 1 of the old version has been deleted, the text of the new version begins at line 0.
- `2a2`: some text has been added after line 2 of the old version. In the new version this text is at line 2.
- `4c4,5`: line 4 of the old version has been replaced by lines 4,5 of the new one.

Successive comparisons of the versions in the server can therefore be used to deduce the author of a line, or the reviewers that suggested to delete a given line.

file.txt	
1	Monday
2	Tuesday Wednesday
3	Thursday
4	Friday

file.txt	
1	Tuesday Wednesday
2	added text
3	Thursday
4	Saturday
5	Sunday week-end

Fig. 2. Two successive versions of a document

diff_file.txt

```
1  Index:file.txt
2  ========================
3  1d0
4  <Monday
5  2a2
6  >added text
7  4c4,5
8  <Friday
9  ...
10 >Saturday
11 >Sunday week-end
```

Fig. 3. `svn diff` applied to the example file of Figure 2

del_prop_file.txt

```
1  Index:file.txt
2  ========================
3  0d1,3
4  Monday
5  4d1
6  Friday
7  5d3
8  Sunday week-end
```

Fig. 4. Example of a deletion proposition file

4.2 Management of Deletion Propositions

The management of writers prohibition to delete text is simply made by the versioning server, that refuses to upload a new version if it includes a deletion. Text that has been so far proposed for deletion does not appear in the current version, and therefore is not visible to writers.

To make those (temporary) deletions visible to reviewers, we store them when they are detected at each new version upload (through an `svn diff`), and indicate them to reviewers in a separate file, together with their position in the text. Remark that those positions are updated at each new version upload, so that an appropriate interface could mix the text file with the deletion proposition file, to show reviewers a unique file (for example using different colors to distinguish deletion propositions). Building such a user-friendly interface is left to future work.

An example of deletion proposition file, refering to our example for the `svn diff` command, is given in Figure 4. This example reads as follows: we have assumed that the participant that uploaded the new file version of figure 2 has the identifier 1, and that the validator (with identifier 3) has validated that version, but he decided to keep the text `Friday` that had been proposed for deletion by 1, and to delete the text `Sunday week-end` that 1 had added. We therefore read

- `0d1,3;Monday`: that text would be at line 0 of the document if it were not deleted, and it has been proposed for deletion by participants 1, 3 (it has thus been deleted in the validated version since 3 is the validator here).
- `4d1,Friday`: that text would be at line 4 of the document, it has been proposed for deletion by participant 1, but not by the validator. Therefore the text is reinserted into the final document.
- `5d3,Sunday week-end`: that text was at line 5 of the document, and has been deleted by the validator only.

In the document building process, reviewers can choose to read or not the parts proposed for deletion so far, and to confirm or contradict the deletion propositions. In that latter case, the text becomes visible again to writers (it is reinjected in the current version, but still remains in the deletion proposition file to store the fact that some users have proposed it for deletion).

4.3 Computing Trust Scores

We now describe how the trust scores described in Section 3 can be practically calculated after the validator has brought the last changes and validated the final document version.

The total number of words n_{final} in the final version is obviously the most easy index to obtain. Likewise, the total number of words that have been deleted n_{del_val} is simple to compute, simply by counting the words in the last version of the deletion proposition file.

We describe below how we proceed to compute the other indices needed to calculate the trust score of a participant $i \in W \cup R$, namely n_i, n_{i_final} (for writers and reviewers), and $n_{i,del\_prop}, n_{i,del\_val}$ (for reviewers), with the notations of Subsection 3.2.

Indices $n_{i,del}$ and n_{i,del_val} *(reviewers).* Those indices are respectively the number of words that reviewer i proposed for deletion and the number of words that among those have effectively been deleted. They can be computed quite easily from the deletion proposition file exemplified in Figure 4: $n_{i,del}$ is the number of words of propositions for which identifier i occurs, and n_{i,del_val} is the number of words of propositions for which both identifiers i and V occur, where V is the identifier of the validator.

Indices n_i and $n_{i,final}$ *writers and reviewers.* Our method uses the text deletion proposition file, and the results of `svn diff` applied to the successive versions of the document, starting with the latest version, to calculate n_i and the difference $n_{i,bad} := n_i - n_{i,final}$ for each participant i. We proceed by updating a table containing the values of the indices we are looking for, the table being initialized with all values equal to 0. Then the comparison with the previous version allows to determine, for the author of the current version, the number of words that have been added. We moreover use this exploration of the file history to identify the authors of the deleted parts, and update accordingly their index $n_{i,bad}$.

More precisely, the procedure works as follows:

- at each version compared to the previous one, increment n_k where k is the participant that uploaded the version. The value used to increment is simply obtained by counting the words corresponding to the "`a`"s and "`c`"s in the `svn diff` result (see Figure 3).
- for each line of the final deletion proposition fine where the validator appears, increment by the corresponding number of words the $n_{j,bad}$, where j is the author of that line. The text can be tracked via successive `svn diff` of the latest versions, until its apparition as an addition to the document. The identifier of the uploader of that version is the j we are looking for.

After this scan of the versions, we have the exact values of n_i and $n_{i,final} = n_i - n_{i,bad}$, that can be used to compute the trust scores.

4.4 Why a Reverse Order Processing?

We could also have calculated the indices needed to compute trust scores by comparing the successive document versions in a chronological order. However, we believe that using an anti-chronological method would provide more options.

- We assumed that the versioning system keeps all versions of the document. However we might imagine that very old versions might not be useful and could be deleted. Our procedure could then be easily adapted to that case, simply considering that the text contained in the oldest stored version has no author.
- Following the same ideas, it is possible to imagine that the time component be taken into account in the trust score. For example, early contributions might be preferred to last minute text additions. The trust score formulas we suggest could also be adapted to that case, by adding timing coefficients into the word count of indices $n_i, n_{i,final}, n_{i,del}$, and n_{i,del_val}.

5 Conclusions and Perspectives

In this paper, we have proposed some criteria to evaluate the contributions of each participant in the development of a text document. We have proposed to use a combination of those criteria to compute a trust score for each participant, which can then be used for several purposes (role or revenue distribution, decisions to collaborate or not with that participant in the future...).

We have implemented our proposed procedure to automatically calculate the trust scores, using Java for interfaces and file processing, and Subversion for version management.

Our trust evaluation mechanism stands under quite restrictive assumptions. Relaxing those assumptions gives directions for future work. In particular, we would like to extend our mechanism to the case where there can be more than one validator. Also, it would be useful to allow back-and-forth exchanges between the validator(s) and the writers/reviewers before a version is considered final.

Introducing the time component into the trust scores could also enrich the model, and prevent some problems such as late contributions that are less reviewed. Finally, we aim at using the trust scores (possibly obtained through previous experience) during the document development itself. Indeed, the decision to carefully read or not, to delete or not a text part could rely on the trust score of the author of that part: the efficiency of the document creation process would then be improved, by reducing the reviewing work to the less trustworthy parts.

References

1. Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version control with Subversion. O'Reilly, Sebastopol (2004)
2. Golbeck, J., Hendler, J.: Inferring binary trust relationships in web-based social networks. ACM Trans. on Internet Technology 6(4), 497–529 (2006)
3. Liu, Y.: Trust-based access control for collaborative system. In: ISECS Intl. Colloquium on Computing, Communication, Control, and Management, Guangzhou City, China, pp. 444–448 (August 2008)
4. Marti, S.: Trust and Reputation in Peer-to-Peer Networks. PhD thesis, Stanford University (May 2005)
5. Matthew, R., Agrawal, R., Domingos, P.: Trust management for the semantic web. In: Proc. of 2nd International Semantic Web Conference, Sanibel Island, Florida (2003)
6. Suryanarayana, G., Taylor, R.N.: A survey of trust management and resource discovery technologies in peer-to-peer applications. Technical Report UCI-ISR-04-6, University of California, Irvine (July 2004)