UNIVERSITY OF Southampton

University of Southampton Research Repository
ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Provenance Enriched Data Rating Assessment for Crowdsourcing

by

Amir Sezavar Keshavarz

Thesis for the degree of Doctor of Philosophy

September 2015

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Doctor of Philosophy

PROVENANCE ENRICHED DATA RATING ASSESSMENT FOR CROWDSOURCING

by Amir Sezavar Keshavarz

Data collection is a vast area that involves the process of gathering information on variables of interest that enables one to answer research questions, validate hypotheses, and evaluate a model. For example, banks collect data about finance and debt; environmental agencies may collect data for varying purposes such as forecasting weather or analysing pollution; universities may collect data about students and their satisfaction.

Stanton (1998) highlights the viability of World Wide Web data collection. One of the increasingly popular approaches to collect data is the idea of outsourcing data collection tasks to users of the Internet. This practice is called crowdsourcing (Howe, 2006a). The immediate advantage of crowdsourcing is that the task is undertaken quickly and cheaply. However, due to varying knowledge, heterogeneity, and expertise of users, it is fundamental for the task requester to assess the quality of data. The ultimate goal is to accept data of high quality and discard those with low quality.

In order to support data quality assessment, Bernstein et al. (2010) introduce a workflow for crowdsourcing applications, called Find-Fix-Verify (FFV). Simply put, an FFV-based crowdsourcing application breaks tasks into smaller tasks or micro-tasks, each of which is undertaken by a user and verified by other users. Not only are micro-tasks easier and of shorter durations, but they can also be distributed among different users, facilitating cross-verification by peer users and minimizing individual biases.

The quality assessment in FFV-based crowdsourcing applications can be done based on the users' verifications and how good a user has been performing in the application (i.e. the reliability of user). We assume that a user's past performance is an indicator of their reliability and how they may perform in the future. Teacy et al. (2006) consider the interaction history of users to construct a general trust model and Miles and Griffiths (2015) claim the context of previous interactions contains information that could be valuable for reputation assessment. Similarly, we consider the full interaction history of users to assess their reliability in a crowdsourcing application.

In order to record users' past interactions and the verification activities in a crowdsourcing application, we propose to record the provenance of data and exploit it to assess the quality of data and the reliability of users. Provenance is a description of how an artifact (any resource on the Web and beyond) was created, who created it, and what else was involved. This dissertation presents a data quality assessment approach for any FFV-based crowdsourcing application that exploits the provenance of data to assess the quality of data and the reliability of users. In so doing, we form the following contributions.

Our first contribution (Chapter 3) is a set of provenance patterns that shape the provenance of data for different activities that are undertaken in a FFV-based crowdsourcing application. The provenance shaped by these patterns is then exploited so that the quality of data and reliability of users are assessed.

By exploiting the provenance of data, it is now possible to rate data and users. Data is rated by a quality measure that describes the validity of data. A user is rated by another quality measure that describes the reliability of the user. These quality measures are utilised by the crowdsourcing application to make a set of quality-based decisions; for example to accept data of high quality and discard low quality ones. Before being able to compute these quality measures, we require a generic approach to be able to traverse the provenance graph encompassing the provenance that is shaped by the provenance patterns. Our second contribution (Chapter 4) is a framework, called "Annotation Computation Framework" (*ACF*) that provides a generic mechanism for an application to traverse a provenance graph and enables the application to compute extra information over the graph.

Being able to traverse the provenance graph and compute extra information over the graph, allows the exploitation of provenance that is shaped by the provenance patterns and computation of quality measures. Our third contribution (Chapter 5) lies in proposing an instantiation of *ACF* called "Provenance Enriched Data Rating Assessment" (*PEDRA*) that exploits the provenance of data generated in any FFV-based crowdsourcing application to assess the quality of data and the reliability of users. *PEDRA* computes a set of quality measures that assist the crowdsourcing application to either accept or discard data after the crowdsourcing application finishes its execution and the data collection is finished.

However, we also envision situations where *PEDRA* is required to be utilised while the crowdsourcing application is executing live. The online application of *PEDRA* allows the crowdsourcing application to make online quality-based decisions such as dynamic task termination based on the reliability of contributors. Furthermore, online decision making allows the application to discard data of low quality and not ask for further users' contribution or deactivate adversarial users. The ultimate goal of the online application of *PEDRA* is to improve the performance of the crowdsourcing application. Our fourth contribution (Chapter 6) is the proposal of a data quality assessment service in which any FFV-based crowdsourcing application submits the provenance of data, shaped by the provenance patterns, for data quality assessment. In order to regulate the interaction between *PEDRA* and the crowdsourcing applications, we envision an online architecture by proposing an online contract, known as *PEDRA-O*[1], specifying how and when the crowdsourcing application is expected to submit the provenance of data. In return, the contract states how the quality measures computed by the service are returned to the crowdsourcing application.

Furthermore, we evaluate both *PEDRA* and *PEDRA-O* by retrofitting them into CollabMap, an exemplar FFV-based crowdsourcing application. The accuracy of *PEDRA* outperforms the current non provenance-based mechanism in CollabMap by 6.5%. Furthermore, the utility of provenance in crowdsourcing is measured in terms of uncertainty.

[1]The O in the name refers to the online application of *PEDRA*.

We show that as more provenance is exploited by *PEDRA*, the uncertainty over computed quality measures reduces. We also establish that both results on accuracy and utility of provenance are statistically significant.

The evaluation also shows that the online application of *PEDRA* improves the performance of the crowdsourcing application by providing online feedback to the application. By utilising the computed quality measures, the crowdsourcing application is able to make quality-based decisions such as when to terminate a task, resulting in a better use of available resources (reducing workers fee by at least 20%) while maintaining the same or higher level of accuracy. We further establish that the decrease in workers' fee is statistically significant. Furthermore, the evaluation shows that the online application introduces only negligible, predictable, and manageable costs to the crowdsourcing application.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Nomenclature

d	A data product
u	A worker
v	A version
d^v	A version of a data product
u^v	A version of a worker
$score$	A vote or a rating, all subsumed under the category of a score
S^+	Total number of positive scores
S^-	Total number of negative scores
Q_V	An estimate of the validity
L_{Q_V}	The validity label
S_C^+	Total number of valid data product creation
S_C^-	Total number of invalid data product creation
Q_C	An estimate of the consistency of a creator in creating valid data products
$\mathbb{D}(u^v)$	The set of all data products created by all versions of worker u up until version v
S_S^+	Total number of aligned scores
S_S^-	Total number of not aligned scores
Q_S	An estimate of the regularity of a scorer to agree with the consensus of the other workers
CWS	The summation of the reliability of all scorers given the value of the $score$ they provide
T	Specifies if a data product should be accepted or discarded, or further collaboration is required
γ_{Q_V}	The uncertainty over Q_V in terms of S^+ and S^-
γ_{Q_C}	The uncertainty over Q_C in terms of S_C^+ and S_C^-
D_{Q_V}	The amount of provenance that was exploited to compute validity estimate measure for d_v
D_{Q_C}	The amount of provenance that was exploited to compute creator reliability measure for u^v
$P(d^v)$	Provenance lifespan of a version of a data product

$\mathrm{Var}[X]$	The variance of a beta distribution for a random variable X with parameter α and β
$Measure(e^v)$	The measure of a version (v) of a provenance element (e)
PDF	Probability density function
CDF	Cumulative distribution function

Acknowledgements

In memory of my brother, Ali, who always had my back no matter how complicated it got. His heart yearned to stay, but the strength I always loved in him, finally gave way.

I am heartily grateful to ...

... my mom, for always being there for me, dealing with me being worlds away, and listening to my non-stop complains. Thank you for showing me how to live. Thank you for tying up so many loose ends in my absence.

... my dad, for his endless and extensive support through my life. I could not have asked for a better role-model. Thank you for taking us to all those amazing trips even though we were not as enthusiastic as we should have been but then enjoying every day of the trip!

... my sis, for encouraging me and expressing confidence in my abilities when I could only do the opposite!

... Luc Moreau, for being an amazing supervisor from master dissertation to PhD thesis, for providing such an incredible opportunity to undertake this research under his guidance. Thank you Luc for the precious help, patience, support, and for having a keen eye for detail.

... Dong, my co-author, for all his advice and help in writing papers. I thank him for his contribution and his good-natured support.

... Aris-Athanasios (Thanos), Pavlos, and Lampros, for all the thoughtful, long, and rather philosophical talks! Thank you for motivating me by playing Football Table!

... Faranak, Michael (Mike), and Danius (Dan) for reading chapters of this thesis, providing feedback, and improving the quality (and readability!) of this work. A special thank you goes to Zoltán (Zoli) for helping me with the mathematics and making (better) plots for my thesis!

Finally, I would like to thank all my colleagues, companions, and the lab managers in "Building 32" who made working in the lab such a pleasant and an incredible experience. Special thanks to the "Electronics and Computer Science (ECS)", ORCHID project, and EPSRC for funding my studies.

To my mother for always being there for me.
To my father for his endless support.
To my sister for her kindness.
In memory of my brother.

Chapter 1

Introduction

Crowdsourcing (Howe, 2006a) is an approach to distribute tasks among human contributors and collect data. Crowdsourcing leverages the skills and expertise of people via the Web to perform simple or complex activities, as such, a task can be undertaken in a fast and cost-effective way (Snow et al., 2008). These tasks include activities such as classifying images, rating objects, or transcription.

The crowdsourcing model is based on a web-based interaction between two types of bodies: (1) *task requester* is an individual or an organisation who are looking to collect data and (2) *worker* is a person who wishes to undertake a task. In order to facilitate this interaction, there are Internet-based marketplaces for crowdsourcing such as Amazon Mechanical Turk (AMT)[1] or CrowdFlower[2]. These marketplaces allow a task requester to post tasks to the Internet. A worker finds these tasks in the marketplace and attempts to undertake them in exchange of monetary or non-monetary (volunteering or reputation) rewards (Ipeirotis, 2010). It is not required to find workers through these marketplaces and the crowdsourcing application may find and recruit workers through its own system.

Unfortunately, however, there is often a great deal of uncertainty over the performance of people and the quality of their work, especially in crowdsourcing. As we cannot read minds and we may not have enough information about people, we cannot be certain about their intentions nor their competency. Hence, it is difficult to trust their work. As such, questions such as "is this particular person reliable?", "is their work trustworthy and acceptable?", or similar about the quality of data and reliability of workers may arise.

Answering these quality-based questions is a challenging problem and one that, in part, we aim to address in this dissertation. To do so, this dissertation elaborates on a data quality assessment approach that is designed specifically for crowdsourcing applications

[1] Amazon Mechanical Turk: https://www.mturk.com/mturk/welcome
[2] CrowdFlower: http://www.crowdflower.com/

to assess the quality of data. But before delving into this approach, we need to explain what we mean by quality and the precise set of problems that we aim to address.

To this end, the rest of this chapter is divided into the following eight sections:

- Section 1.1 discusses crowdsourcing and provides some successful examples of crowdsourcing. Then, some examples where the management of crowdsourcing went wrong are presented.
- Section 1.2 focuses on quality assessment in crowdsourcing by introducing a crowdsourcing workflow.
- Section 1.3 defines provenance and presents it as a generic mechanism through which the quality of data can be assessed.
- Section 1.4 justifies the use of provenance in crowdsourcing.
- Section 1.5 introduces a generic provenance enriched quality assessment approach for crowdsourcing applications.
- Sections 1.6 details the research objectives of this dissertation.
- Section 1.7 presents the research contributions of this dissertation.
- Section 1.8 describes the structure of the rest of the dissertation.

1.1 Crowdsourcing

Kittur et al. (2008) highlight that engaging users in data collection may become expensive and as such there might be a trade off between sample size, time requirements, and monetary costs. They state the utility of crowdsourcing through which a large number of users are employed for low time and monetary costs. Simply put, crowdsourcing (Howe, 2006a) is an approach to distribute tasks among human contributors for variety of activities (Schenk and Guittard, 2009).

Regardless of the purpose, a task requester offers tasks to the workers through either crowdsourcing marketplaces or its own website. A worker accepts and works on the tasks and submits the results to the requester. It is noteworthy to mention that the data to be collected may be a side effect of the task and not necessarily the task output. For example, the reCAPTCHA apparatus explores whether the output of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) can be utilised for a useful purpose: helping to digitize old printed material by asking users to decipher scanned words from books that computerized optical character recognition (OCR) failed to recognize (von Ahn et al., 2008). Another example is the ESP game where users implicitly collaborate to label images as a side effect while playing the game (von Ahn and Dabbish, 2004).

Successful Applications of Crowdsourcing

There is an incredible story to be told about human participation and what it could bring to society. Here, some notable and successful scenarios that involve crowdsourcing are mentioned. The goal is to show that crowdsourcing results in obtaining functional and useful services.

Wikipedia: One of the most well known examples of utilising the crowd is Wikipedia[3]. Wikipedia is an international project that uses Wiki software to collaboratively create an encyclopaedia (Bruns, 2008). By July 2015, it contained more than 4,900,000 articles[4].

Galaxy Zoo: Galaxy Zoo[5] provides visual morphological classifications for nearly one million galaxies. This is only possible by inviting the general public to visually inspect and classify these galaxies. The project claims to have obtained morphological classifications of nearly 900,000 galaxies, contributed by hundreds of thousands of volunteers (Goecks et al., 2010).

Ushahidi: Ushahidi[6] is a non-profit organisation that focuses on information collection, visualisation, and interactive mapping through the use of crowdsourcing (Okolloh, 2009). The service was first deployed in the aftermath of Kenya's disputed 2007 presidential election that collected eyewitness reports of violence reported by email and text message and placed them on Google Maps. The creators have an ambition to use crowdsourcing for social activism and citizen journalism. The website allows everyone (local observers) to produce content using a variety of mediums such as mobile phones or the Internet. This service has been employed in many crisis such as the earthquake in Haiti (Morrow et al., 2011), the earthquake in Chile[7], the Louisiana oil spill crisis[8,9,10], and many more.

Potential Problems in Crowdsourcing

While crowdsourcing can be a key contributor to successfully creating content or solving problems, the inefficient management of the crowdsourcing environment can result in counterproductive outcomes that can potentially invalidate the utility of crowdsourcing.

[3]Wikipedia: http://www.wikipedia.org/
[4]Size of Wikipedia: http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia
[5]Galaxy Zoo: http://www.galaxyzoo.org/
[6]Ushahidi: http://www.ushahidi.com/
[7]Ushahidi-Chile platform: http://chile.ushahidi.com/
[8]Ushahidi used to create oil spill crisis map: http://www.ushahidi.com/2010/05/08/labb/
[9]Louisiana Bucket Brigade oil spill crisis map: http://www.rivernetwork.org/resource-library/louisiana-bucket-brigade-oil-spill-crisis-map
[10]Oil spill crisis map initial analysis and review: http://labucketbrigade.org/sites/default/files/Oil%20Spill%20Crisis%20Map%20Initial%20Analysis%20and%20Review.pdf

Ushahidi: One of the issues that the Ushahidi project encounters is in regards to the quality of reports submitted by the users of the system. The initial solution by Ushahidi to such an issue was that all reports had to be manually checked and approved by Ushahidi staff before they went live. The Ushahidi staff reported that many people misreported the true needs and priorities of their emergency (Okolloh, 2009).

Boston Marathon event: Another scenario where the the management of crowdsourcing environment went wrong, was the Boston Marathon bombings[11] incident in April 2013. In order to gather evidence, the FBI asked the public to provide any images or videos they might have. The data for the Boston marathon investigation was crowdsourced. Out of this data, some people were identified by anonymous commenters as suspects based upon their clothes and appearance. In one case, someone's face was broadcast publicly (cover of the NY Post[12,13]).

DARPA Red Balloon challenge: In this challenge, 10 red weather balloons were placed at undisclosed locations throughout the United States. The first team to correctly identify the locations of all balloons would win a $40,000 prize (Tang et al., 2011). A team from the Massachusetts Institute of Technology (MIT) won the competition in less than nine hours by employing a recursive incentive mechanism (Pickard et al., 2011). They recruited almost 4400 individuals and used the prize money as a financial incentive for people who correctly located balloons and those who connect the finder to the MIT team (Pickard et al., 2011). However, the majority of submissions of balloon sightings to the winning MIT team turned out to be false (Naroditskiy et al., 2012).

These scenarios are evidence that the quality of data submitted by unknown workers and workers' reliability are required to be continuously assessed. The ultimate goal in such data quality assessment is to accept data of high quality and discard those with low quality. The next section focuses more on data quality assessment in crowdsourcing.

1.2 Crowdsourcing Workflow and Quality Assessment

The immediate advantage of crowdsourcing is that the task is undertaken quickly and cheaply.

[11] Boston Marathon bombings on Wikipedia: http://en.wikipedia.org/wiki/Boston_Marathon_bombings

[12] Forbes: Lessons from crowdsourcing the Boston Bombing Investigation: http://www.forbes.com/sites/tarunwadhwa/2013/04/22/lessons-from-crowdsourcing-the-boston-marathon-bombings-investigation/

[13] The Washington Post: Backpack brothers an example of the drawbacks to Internet sleuthing: http://www.washingtonpost.com/local/dc-politics/backpack-brothers-an-example-of-the-drawbacks-to-internet-sleuthing/2013/04/18/8c0ea9fa-a852-11e2-b8ad-87b8baf4531b_story.html

However, the task requester does not know the workers, their competence and expertise, nor their culture and language. This diverse community could potentially provide contradictory data. Furthermore, workers could have different goals while undertaking a task. There might be workers' biases or adversarial workers who are trying to maximise their profit by undertaking as many tasks as possible without paying attention to the quality of their work (the data produced by them is called noisy data). As such it is fundamental for the task requester to assess the quality of data. The ultimate goal is to accept data of high quality and discard those with low quality.

In order to make quality assessment more manageable, Bernstein et al. (2010) introduced the Find-Fix-Verify (FFV) workflow for crowdsourcing applications, which has been adopted by a growing number of applications like BudgetFix (Tran-Thanh et al., 2015, 2014), linked data quality assessment (Acosta et al., 2013), CollabMap (Ramchurn et al., 2013), CrowdMap (Sarasua et al., 2012), PlateMate (Noronha et al., 2011), and many more. The FFV workflow breaks complex tasks, such as text editing, into much simpler *micro-tasks*, such as *find*ing a mistake in a sentence, *fix*ing it, and *verify*ing the correction (hence the name of the workflow). Not only are micro-tasks easier and of shorter durations, but they can also be distributed among different workers, facilitating cross-verification by peer workers and minimizing individual biases.

The general procedure in a FFV-based crowdsourcing application is as follows: there is a problem that the task requester wants to be solved (which is called a task in crowdsourcing). The problem is presented to the workers by the task requester. A worker solves the problem; for example, a worker may provide an answer, or fix an issue, or similar. The crowdsourcing application, and according to the FFV workflow, recruits other workers to verify the solutions. Hence, at this stage, there is one problem, one or more solutions, and the verifications. More specifically, data generated in FFV-based crowdsourcing applications follow a common pattern: a problem, one or more solutions to the problem, and their votes (worker's verification); this pattern is repeated until the quality of a solution is deemed satisfactory.

It is required to merge (i.e. fuse) all the verifications on a solution, so that an output is generated. This output instructs the crowdsourcing application to either accept or discard the solution proposed by a worker. Furthermore, based on workers' activities in the crowdsourcing application (solutions they provide and their verifications), it is required to assess how well and consistently a worker has been performing. This allows the crowdsourcing application to, for example, schedule task.

The quality assessment in FFV-based crowdsourcing applications can be done based on the users' verifications and how good a worker has been performing in the application (i.e. the reliability of worker). We assume that a worker's past performance is a good indicator of their reliability and how they may perform in the future (as also discussed by Teacy et al. (2006) and Miles and Griffiths (2015)). In order to record users' past

interactions and the verification activities in a crowdsourcing application, we propose to record the provenance and exploit it to assess the quality of data and reliability of users.

1.3 Provenance and Quality Assessment

Moreau and Groth (2013b) highlight the change in the users' role in the World Wide Web from passive consumers to active publishers. Due to this change, Moreau and Groth (2013b) comment on the importance of provenance from "science to food manufacturing, from data journalism to personal well-being, from social media to art". Simply put, provenance is "a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing" (Moreau et al., 2013a).

In the same book, Moreau and Groth (2013b) make the case for provenance by bringing many examples from various domains, for example, "food provenance" in regards to the horsemeat scandal in Europe[14]. In the horsemeat case, it was required to determine at which point mislabelling took place. As such, it was necessary to trace back through the chain of suppliers. Provenance could have been employed in this case to keep track of the traces and audit processes. Seneviratne (2012); Ko et al. (2011); Hasan et al. (2009) discuss the role of provenance in showing whether a system is accountable and auditable.

Provenance can also be employed to help people to make a judgement as to whether something can be trusted by offering the means to verify data products and infer their quality. Provenance is identified as one of the many salient factors that affect how people determine trust in content available in the Internet (Moreau, 2010; Gil and Artz, 2007; Golbeck, 2006b). In practice, some approaches are developed to infer a notion of trust for users (Stamatogiannakis et al., 2014; Li et al., 2010; Golbeck, 2006a).

More recently, provenance has also been incorporated into crowdsourcing applications. For example, Ainy et al. (2014) discuss the potential of provenance in offering a description on how information was derived which can be employed for credibility assessment in crowdsourcing applications. In this short vision paper, they identify a set of challenges in leveraging provenance for presentation and maintenance. They state that maintaining the full and exact provenance information may be infeasible and offer some initial directions towards addressing this challenge. Furthermore, Huynh et al. (2013a) present an application-independent methodology to analyse provenance graphs to extract some network metrics that correlate to trust.

Even though it has been mentioned in the literature that provenance can be employed to infer trust and help people in quality assessment and even though provenance has

[14] http://www.bbc.co.uk/news/uk-21393180

been incorporated in crowdsourcing with data quality assessment in mind, there is no approach that exploit provenance of data and assess the quality of data practically. In this dissertation, we offer a data quality assessment approach for crowdsourcing application that exploits provenance of data to assess the quality of data in a principle and systematic way. First, we justify the use of provenance in crowdsourcing and then offer the data quality assessment approach.

1.4 Justification of Provenance in Crowdsourcing

As stated previously, a crowdsourcing application requires an approach that assists in quality assessment. The ultimate goal of such assessment is to choose data of high quality and discard the low quality data.

We claim that the quality assessment can be done by considering the verification activities of workers and history of their interactions with the crowdsourcing application. This quality assessment requires a medium that not only can capture what data was generated, but also who was responsible in the data creation and verification activities.

The first case of utilising provenance in crowdsourcing is that provenance of data captures all the necessary information for quality assessment (information such as what data was generated, who generated it, and who verified it). Furthermore, provenance of data captures the information in regards to the interaction history of a worker with the crowdsourcing application which is also deemed necessary for quality assessment.

The second justification of utilising provenance in crowdsourcing is that the quality assessment approach is required to be generic and applicable to any FFV-based crowdsourcing application and does not rely on a specific data model, nor a specific application domain. Although every crowdsourcing application can generate provenance of its data according to a specific data model, in Chapter 3, we propose a set of patterns that shape the provenance of data generated by a crowdsourcing application. The shaped provenance can be exploited so that the quality of data is assessed.

The third justification is to utilise provenance to have a quality assessment approach that is auditable. The quality assessment approach assists the crowdsourcing application to make quality-based decisions. For example, the application can deactivate a worker if they performed poorly. By utilising provenance, it is possible to justify why a decision was made. By providing a standardised account of actions, the provenance traces make the process auditable (Curcin et al., 2014).

In the next section, we discuss the quality assessment approach known as "Provenance Enriched Data Rating Assessment" or *PEDRA* for short.

1.5 Provenance Enriched Data Rating Assessment

Having recorded the provenance of data generated in a FFV-based crowdsourcing application and shaped it according to the patterns (to be introduced in Chapter 3), now it is possible to exploit the provenance to assess the quality of data and reliability of workers.

More specifically, we present a quality assessment approach called "Provenance Enriched Data Rating Assessment" (*PEDRA*) that exploits provenance of data to compute a set of quality measures. For example, *PEDRA* rates generated data with a quality measure that describes the validity of data (whether data is valid or not); it also rates workers with another quality measure that describes the reliability of a worker.

These quality measures are utilised by the crowdsourcing application to make quality-based decisions such as:

- Decisions on the validity of data: The crowdsourcing application can utilise the computed quality measures by *PEDRA* to either accept or discard data.
- Decisions on the reliability of workers: The crowdsourcing application is able to assess the reliability of workers by utilising the quality measures computed by *PEDRA*. Through this assessment, the crowdsourcing application is able to distinguish reliable workers from adversarial or unreliable workers. The application can deactivate those adversarial workers or ask unreliable workers to contribute better.
- Decisions on the termination of tasks: A crowdsourcing application is required to decide how many workers it requires to terminate a task. *PEDRA* computes another measure that can be utilised by the crowdsourcing application for task termination decision. This measure instructs the application on whether more workers are required so that the task can be terminated.

Table 1.1 lists three sample quality measures computed by *PEDRA* and their uses in a crowdsourcing application.

Figure 1.1 demonstrates the interaction between *PEDRA* and a crowdsourcing application. The crowdsourcing application recruited workers and persisted the data in an storage. The task requester is required to assess the quality of data and accept only those of high quality. As such, the application generates provenance of its data and submits it to *PEDRA* where *PEDRA* computes a set of quality measures. The application utilises the computed measures to choose data of high quality (good data) and discard those with low quality (bad data).

Measure Computed by *PEDRA*	Sample Use in a Crowdsourcing Application
Validity estimate	To decide whether to accept or discard data
Worker reliability	To assess the reliability of a worker who created the data, unreliable ones can be banned from participating any more and reliable ones can be awarded
Termination	To decide on the termination of the task and how many workers it requires to dedicate to the task

Table 1.1: Examples of quality measures computed by PEDRA and their sample uses in a crowdsourcing application.

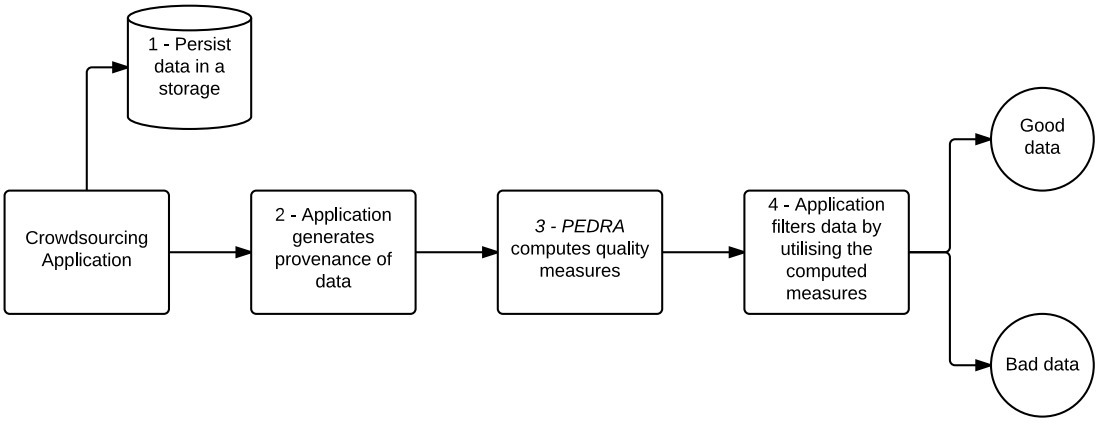


Figure 1.1: Interaction between PEDRA and a crowdsourcing application after the execution of the crowdsourcing application is finished (Offline Environment).

Figure 1.1 demonstrates the application of *PEDRA* after the crowdsourcing application finishes its execution and the data collection phase is finished. From now on, we call this setting "offline environment" where the application of *PEDRA* takes place in an offline environment.

Furthermore, the quality measures computed by *PEDRA* can be fed back to the crowdsourcing application as it is executing so that the application improves its behaviour. For example, if the crowdsourcing application knows a worker is performing poorly, the application can deactivate the worker and does not allocate any more task to the worker. This is a better behaviour as opposed to the case where the application does not do anything and allows the worker to continue collaborating (offline environment).

Figure 1.2 is the modified version of Figure 1.1. According to this figure, a crowdsourcing application allocates a task to a worker. After the task is undertaken, the application generates provenance of data for that task and submits it to *PEDRA* where *PEDRA* computes quality measures by exploiting the provenance shaped by the patterns. The quality measures are then passed back to the application where the crowdsourcing application can make quality-based decisions.

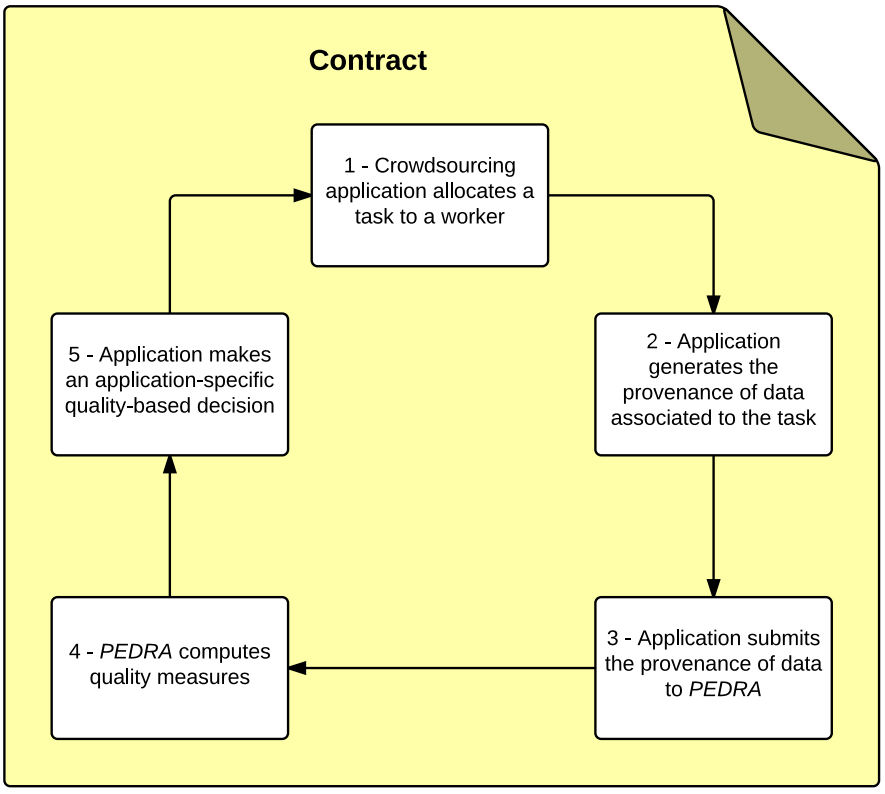


Figure 1.2: Interaction between PEDRA and a crowdsourcing application while the crowdsourcing application is executing live (Online Environment).

From now on, we call this setting "online environment" where the application of *PEDRA* takes place while the crowdsourcing application is executing live. We envision *PEDRA* as a quality assessment service to multiple, different FFV-based crowdsourcing applications. In order to regulate the interactions between *PEDRA* and the crowdsourcing applications, we envision an online architecture by proposing an online contract, known as *PEDRA-O*[15], specifying how and when the crowdsourcing application is expected to submit the provenance of data. In return, the contract states how the quality measures computed by the service are returned to the crowdsourcing application.

The contract mandates that the provenance of data should be shaped based on the patterns and when it should be submitted to the service. As such, considering a software engineering perspective, the contract specifies the input/output interface; i.e. how provenance of data is submitted by the crowdsourcing application and how results are passed back by the service.

In the next section, the research objectives of this dissertation are outlined.

[15]The O in the name refers to the online application of *PEDRA*.

1.6 Research Objectives

So far, we have presented a few successful examples of crowdsourcing where data was collected quickly and cheaply. We have also introduced some examples where the management of the crowdsourcing environment went wrong. One of the factors that could fail a crowdsourcing application is data verification. The data is generated by workers recruited from a typically unknown workforce with varying level of background (e.g. country, language) and expertise (e.g. drawing or mathematical skills). As such, it is important for the crowdsourcing application to assess the quality of data. In order to assist the application in quality assessment, in Section 1.5, we introduced *PEDRA*, as a data quality assessment approach, that assesses the quality of data generated in any FFV-based crowdsourcing application.

We form the following PhD dissertation statement:

> *The provenance of data generated in a FFV-based crowdsourcing application can be exploited so that the quality of data and reliability of workers are assessed with the following two goals: (1) to choose data of high quality and discard data with low quality, and (2) to improve the performance of the crowdsourcing application by providing online feedback as the application is executing live.*

Bellow, we outline the research aims that we intend to fulfil in this dissertation.

1. Decision Facilitation: We aim to develop a quality assessment approach that facilitates quality-based decision making in FFV-based crowdsourcing applications (we enumerate some examples of quality-based decision making in Section 1.5).
2. Offline and Online Applicability: We aim to put forth a quality assessment approach that can be utilised when the execution of the crowdsourcing application is finished with the ultimate goal of filtering data based on quality (in an offline environment) and when the crowdsourcing application is executing live with the goal of improving the behaviour of the application (in an online environment).
3. Assessment Based on Reliability: We aim to base the quality assessment on the verification activities and full interaction history of workers.
4. Efficient Response Time: We aim to present a quality assessment approach whose performance in terms of response time does not deteriorate. We define response time as the time it takes for the quality assessment approach to assess the quality of data in a crowdsourcing task.

5. Auditable Quality Assessment Approach: We aim to develop an auditable quality assessment approach. This allows a thorough and accurate assessment of the approach (for example to check the computation of past measures against rules).

Having outlined this set of aims, next section outlines the contributions of our work.

1.7 Research Contributions

Given the research objectives outlined in the Section 1.6, we provide the following four contributions.

1. Provenance patterns for crowdsourcing applications (Chapter 3): We specify a set of generic patterns that shape the provenance of data for different activities that are undertaken in a FFV-based crowdsourcing application. More specifically, *create pattern* shapes the provenance of data when data is created; *score pattern* shapes the provenance of data when data is verified or scored; *revision pattern* and *agent pattern* shape the provenance of data as a given data product or worker evolves through the execution of a crowdsourcing application.

2. Annotation Computation Framework (*ACF*) (Chapter 4): We put forth a framework, called "Annotation Computation Framework", that provides a generic mechanism to traverse a provenance graph, extract information from the provenance graph, and finally enable the computation of extra information over the provenance graph.

3. Provenance Enriched Data Rating Assessment (*PEDRA*) (Chapter 5): We present a quality assessment approach called *PEDRA*, an instantiation of *ACF*, that exploits the provenance of data shaped by the provenance patterns to compute a set of quality measures that assist the crowdsourcing application to choose data of high quality and discard those with low quality. Furthermore, using *ACF* and *PEDRA*, we are the first to show how a provenance enriched quality assessment approach can be employed as part of a crowdsourcing application, to assess the quality of generated data.

4. Online decision making with *PEDRA* (Chapter 6): We present an architecture for any FFV-based crowdsourcing application with the purpose of assisting the application to improve its behaviour and assess the quality of data while it is executing live. Furthermore, we perform a detailed evaluation to assess the benefits and costs of such architecture on the crowdsourcing application. The evaluation is the first of its kind to integrate a crowdsourcing application and a quality assessment approach in a real world setting where not only the accuracy of the quality assessment approach but also the performance of the approach is measured.

The work described in this dissertation has led to the following peer-reviewed publications:

- Sezavar Keshavarz, Amir, Huynh, Trung Dong and Moreau, Luc (2014) "Provenance for online decision making". International Provenance and Annotation Workshop, Cologne, DE, 09 - 13 June 2014. Springer International Publishing 12pp, 44-55 (doi:10.1007/978-3-319-16462-5_4).

  In this peer-reviewed paper, we presented three of our contributions with an initial evaluation of them. More specifically, we introduced the Annotation Computation Framework (*ACF*) and instantiated it to compute some quality measures for crowdsourcing applications (an earlier version of *PEDRA*). We also presented an online contract as a preliminary to *PEDRA-O*.

- Sezavar Keshavarz, Amir, Huynh, Trung Dong and Moreau, Luc (2014) "Provenance for Online Decision Making". In, Provenance Analytics 2014, Cologne, DE, 09 June 2014. 4pp.

  In this paper, that was accepted in Provenance Analytics, we offered a tutorial on the methodologies employed in this work, mostly the Annotation Computation Framework (*ACF*) and the quality assessment approach (*PEDRA*). With the aid of a scenario, we showed how they can be employed and how a crowdsourcing application can change its behaviour.

- Sezavar Keshavarz, Amir, Huynh, Trung Dong and Moreau, Luc "Provenance Enriched Data Rating Assessment for Crowdsourcing". Under review ACM Transactions on Internet Technology (TOIT).

  This under-review submitted journal paper presents the main research contributions discussed in this dissertation. More specifically, this paper discusses provenance patterns, *PEDRA*, and *PEDRA-O*; with a detailed empirical evaluation of them.

1.8 Thesis Structure

In the rest of the dissertation, we survey the existing literature on crowdsourcing and different approaches to assess the quality of data. We also discuss provenance, how it is used in literature and how trust (quality) can be inferred from provenance data; and describe in detail the work through which we have made the contributions outlined in the Section 1.7. This is achieved through the course of the remaining chapters, which are structured as follows:

- Chapter 2 provides a review of existing mechanisms for assessing quality of data in crowdsourcing applications, concentrating in particular on work that goes some

way towards fulfilling the objectives described in Section 1.6. Furthermore, this chapter presents a discussion on provenance, how it has been used in the literature, and how trust (quality) can be derived from provenance data. From the crowdsourcing literature, we identify the most relevant models with regard to our objectives, and their prominent characteristics are described. In light of these characteristics, we identify several limitations of existing quality models for crowdsourcing applications, and use these to motivate the work described in the proceeding chapters.

- Chapter 3 introduces the provenance patterns that shape the provenance of data generated by a FFV-based crowdsourcing applications. We then bring an example of their instantiations in CollabMap, an exemplar FFV-based crowdsourcing applications.

- Chapter 4 defines a generic framework that allows a system to exploit provenance of data with a purpose different than the one at the generation time. More specifically, the framework allows the system to exploit provenance of data, extract information, and compute new information over the provenance graph.

- Chapter 5 presents *PEDRA* and describes how it instantiates the framework from Chapter 4 to compute quality measures for FFV-based crowdsourcing applications that generated provenance of data based on provenance patterns introduced in Chapter 3. Furthermore, we show how *PEDRA* can be used in practice, by describing how it operates within the CollabMap, and by giving an empirical evaluation of its performance. In particular, this evaluation demonstrates how the quality assessment in *PEDRA* outperforms the current quality assessment employed in CollabMap and how provenance reduces uncertainty over computed quality measures by *PEDRA*.

- Chapter 6 introduces *PEDRA-O* and describes an online architecture that realises the framework from Chapter 4 by building on the characteristics of *PEDRA* from Chapter 5. In particular, we introduce Online Contract which is a contract regulating the communication between a crowdsourcing application and *PEDRA*. We describe how it can be used while a crowdsourcing application is executing live. Finally, we give an empirical evaluation of *PEDRA-O*, demonstrating how it performs under our assumption.

- Finally, Chapter 7 summarises the conclusions drawn throughout the dissertation, and in particular the contributions and reflective on the techniques we have developed. In addition, we outline directions for future work.

Chapter 2

Background

One of the increasingly popular approaches to collect data is the idea of outsourcing data collection tasks to users of the Internet. This practice is called crowdsourcing. The immediate advantage of crowdsourcing is that the task is undertaken quickly and cheaply. However, due to varying knowledge, heterogeneity, and expertise of users, it is fundamental to assess the quality of data. In our work, we are concerned with crowdsourcing and the quality of data, and so in this section, we focus our attention on related work that is relevant to crowdsourcing and quality assessment in crowdsourcing. Specifically, Section 2.1.1 offers a definition on crowdsourcing, Section 2.1.2 presents a short history of crowdsourcing, Section 2.1.3 provides applications of crowdsourcing, Section 2.1.4 specifies how crowdsourcing applications can find workers, and section 2.1.5 outlines some concerns for crowdsourcing.

One of the concerns for the crowdsourcing is the data quality assessment. More specifically, it is required to assess the quality of data so that data of high quality is accepted and data of low quality is discarded. The quality assessment in a crowdsourcing application can be done based on the users' verifications and how good a user has been performing in the application (i.e. the reliability of a user). We assume that a user's past performance is an indicator of their reliability and how they may perform in the future. In order to record users' past interactions and the verification activities in a crowdsourcing application, we propose to record the provenance of data and exploit it to assess the quality of data and reliability of users. To this end, we also pay attention to the related work that is relevant to provenance. Specifically, Section 2.2.1 offers the standard definition of provenance, Section 2.2.2 presents a short history of provenance, Section 2.2.3 provides some examples that records the provenance of data, and Section 2.2.4 brings an overview of the PROV Data Model (PROV-DM), a specification by the World Wide Web Consortium, that we use to record the provenance of data in a crowdsourcing application.

After introducing crowdsourcing and provenance, we focus on quality assessment. Section 2.3.1 offers general trust models that compute a notion of trust. Section 2.3.2 presents some trust models for crowdsourcing application. We look at the pros and cons of each approach and see how they can help us to develop a better quality assessment approach. Section 2.3.3 provides a review of some approaches that exploit provenance to compute a notion of quality.

Finally, we conclude the chapter in Section 2.4.

2.1 Crowdsourcing

2.1.1 Definition of Crowdsourcing

Jeff Howe is the first to coin the term "crowdsourcing" (Howe, 2006b) and offered the following definition (Howe, 2006a):

> Simply defined, crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively), but is also often undertaken by sole individuals. The crucial prerequisite is the use of the open call format and the large network of potential laborers.

Crowdsourcing can apply to a wide range of activities. Estellés-Arolas and González-Ladrón-De-Guevara (2012) studied more than 40 definitions of the term and developed a new integrating definition:

> Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that which the user has brought to the venture, whose form will depend on the type of activity undertaken.

For the rest of this dissertation, we shall adopt the above definitions for crowdsourcing applications. Furthermore, we complement them with two definitions.

Crowdsourcing is based on an interaction between two types of bodies. The first type is *task requester* as defined in the following definition:

Definition 2.1. A task requester is an individual, an institution, a non-profit organization, or a company who wishes to recruit people to undertake a set of tasks.

The second type is *worker* as defined in the following definition:

Definition 2.2. A worker is an individual who undertakes the task and provides the requested output.

The task requester may be willing to pay the worker for their contributions. The payment could be either monetary (usually a few cents) or non-monetary (such as reputation or personal reasons), as agreed.

2.1.2 History of Crowdsourcing

While the term crowdsourcing is mostly applied to Internet-based activities (Brabham, 2008), the concept is not new.

Dawson and Bynghall (2012) name The Longitude Prize[1] as one of the earliest application of crowdsourcing (1714). It was a reward offered by the British government to find a way to measure a ship's longitude.

Dawson and Bynghall (2012) bring another example from 1783, when King Louis XVI offered an award to the person who could "make the alkali"[2] by decomposing sea salt.

We encourage the reader to follow the history as outlined by Dawson and Bynghall (2012): from 1714 and The Longitude Prize to 2006 when Howe coined the term crowdsourcing.

2.1.3 Application of Crowdsourcing

Ushahidi is a collaborative reporting environment that allows large group of people to report an incident (Okolloh, 2009). It was originally developed during violent political unrest in Kenya in 2008 and was used during the Haiti crisis. Ushahidi allows people to anonymously report on a story online or via mobile phone text messages; it then maps the reported stories so it would be visually accessible to people. In the Kenyan

[1]The Longitude Prize: http://en.wikipedia.org/wiki/Longitude_rewards
[2]Alkali: http://en.wikipedia.org/wiki/Alkali

experience of Ushahidi, the main intention was to get information out as quickly as possible; however, all stories had to be manually approved by Ushahidi staff. Stories were counter-checked by comparing with other sources and those seemed to be false, inflammatory, or inaccurate was removed (Okolloh, 2009).

The Defence Advanced Research Projects Agency (DARPA) launched the DARPA Network Challenge in 2009 in which participants were asked to find the location of 10 red weather balloons deployed at undisclosed locations across the continental United States. The first team to correctly identify the locations of all balloons would win a 40,000 dollar prize (Tang et al., 2011). A team from the Massachusetts Institute of Technology (MIT) won the competition in less than nine hours by employing an incentive mechanism (Pickard et al., 2011). They recruited almost 4400 individuals and used the prize money as a financial incentive for people who correctly located balloons and those who connect the finder to the MIT team (Pickard et al., 2011).

Crowdsourcing has been also applied in human optical character recognition (OCR) tasks. Narula et al. (2011) presented a mobile phone-based crowdsourcing platform that divides documents into many small pieces and sends each piece to a different worker.

Another example of using crowd is labelling or classification of large datasets. These hand labelled datasets are required in different applications such as modern machine learning-based approaches to computer vision; for example, Omron face detector requires millions of images for training (Whitehill et al., 2009). People are presented with a set of images and they are required to label that image.

One of these classification applications is a citizen science project called Galaxy Zoo. This project asks the general public to look at images of galaxies from the Sloan Digital Sky Survey (SDSS) and sort galaxies according to their morphology and then divide galaxies they identified as spiral into three subcategories according to the direction of their spiral arms by clicking one of six on screen buttons. The project claims they have obtained morphological classifications of nearly 900,000 galaxies, contributed by hundreds of thousands of volunteers (Goecks et al., 2010). Galaxy Zoo website first provides a brief tutorial on different classes of galaxy to visitors. Then they are tested to ensure that they have enough skills to produce reliable classifications.

In the computer vision domain, Russell et al. (2008) compiled a large collection of images with ground truth labels to be used for object detection research. In order to compile such a dataset, they developed a crowdsourcing application, LabelMe, that recruits users to annotate images. In the LabelMe application, users are presented with an image. Users can label a new object in the image by drawing along the boundary of the desired object and providing a name for it. Users are also presented with all the existing labels in an image and they can edit those labels. Comparison between LabelMe dataset and other existing datasets for object detection shows more object categories in the LabelMe

dataset; furthermore, the LabelMe dataset contains a large number of examples in each object category (Russell et al., 2008).

Labelling datasets can also be entertaining. von Ahn and Dabbish (2004) suggested the collection of data in the form of computer games called The ESP Game. ESP randomly matches two users and presents the same photo to both of them. Users can gain points if they type the same textual label (von Ahn et al., 2006). In this way, users are helping to determine the contents of images by providing meaningful labels for them. They claim that most images on the Web can be labelled in a few months (von Ahn and Dabbish, 2004).

James Surowiecki investigated several cases of crowd wisdom and found that "under the right circumstances, groups are remarkably intelligent, and are often smarter than the smartest people in them" (Surowiecki, 2005). He mentioned that this wisdom of crowds is derived from aggregating all the results obtained from the crowds.

2.1.4 Marketplaces

In order to help finding the crowd, some crowdsourcing marketplace have been introduced. Amazon Mechanical Turk (AMT) is one of the most popular crowdsourcing marketplaces which was introduced by Amazon in 2005 and allows one to recruit workers to do small tasks for small payments (Ipeirotis, 2010). In this online labour market, employees are recruited by employers for the execution of tasks (Human Intelligence Tasks or HITs) in exchange for a wage (Paolacci et al., 2010).

AMT has been used extensively in research. Mason and Suri (2012) used AMT for conducting behavioural research. In another paper, Mason investigated the effect of compensation on performance (Mason and Watts, 2010). They found that increased financial incentives increase the quantity, but not the quality, of work performed by participants.

Marge et al. (2010) investigated the use of AMT as a reliable method for transcription of spoken language data.

Sprouse (2011) recruited users in AMT for psychological experiments such as the acceptability judgement task and compared the results with experimenter-controlled environment (as opposed to user-controlled environment of AMT). The results suggest that AMT data are almost indistinguishable from laboratory data.

Paolacci et al. (2010) recommend the use of AMT for data collection because of various advantages it offers such as reduction in cost and time.

It is not required to find workers through these marketplaces and the crowdsourcing application may find and recruit workers through its own system, for example,

Ushahidi (Okolloh, 2009), CollabMap (Ramchurn et al., 2013), and Galaxy Zoo (Goecks et al., 2010).

2.1.5 Concerns for Crowdsourcing

Data or services can be collected through crowdsourcing in a cheap and quick manner. However, there are a couple of concerns that require further attention. One is about the quality of the work undertaken by unknown workers recruited from online labour marketplaces such as AMT. Another is in regards to ethical concerns.

Quality concerns

It is required to assess the quality of data generated by workers so that data of high quality is accepted. The data is generated by workers recruited from a typically unknown workforce with varying level of background (e.g. country, language) and expertise (e.g. drawing or mathematical skills). Furthermore, due to financial incentives, a worker may complete tasks quickly rather than well.

As such, quality is an issue that needs to be considered. We shall discuss more about this concern in Section 2.3 where we review general trust models and different quality assessment approaches for crowdsourcing applications.

Ethical concerns

There are some ethical concerns about the minimum wage as workers are considered independent contractors rather than employees. As such, workers using AMT may earn less than the minimum wage (Ross et al., 2010). For this reason, Mason and Suri (2012) argues that the wage conditions might be unethical. On the other hand, Busarovs (2013) notes that workers on AMT do not feel that they are exploited and are ready to participate in crowdsourcing activities in the future.

2.1.6 Crowdsourcing Workflow

In order to support data quality assessment, Bernstein et al. (2010) introduce a workflow for crowdsourcing applications, called Find-Fix-Verify (FFV). An FFV-based crowdsourcing application breaks tasks into smaller tasks or micro-tasks, each of which is undertaken by a user and verified by other users. Not only are micro-tasks easier and of shorter durations, but they can also be distributed among different users, facilitating cross-verification by peer users and minimizing individual biases.

The general procedure in a FFV-based crowdsourcing application is as follows: there is a problem that the task requester wants to be solved (which is called a task in crowdsourcing). The problem is presented to the workers by the task requester. A worker solves the problem; for example, a worker may provide an answer, or fix an issue, or similar. The crowdsourcing application, and according to the FFV workflow, recruits other workers to verify the solutions. Hence, at this stage, there is one problem, one or more solutions, and the verifications.

There are alternatives in the literature that rely on organising and visualising crowd workflows. For example, both Turkomatic (Kulkarni et al., 2012) and CrowdWeaver (Kittur et al., 2012) employ graph visualizations to show the organization of crowd tasks, allowing users to better understand their workflow and design for higher quality (Rzeszotarski and Kittur, 2012). CrowdForge (Kittur et al., 2011) and Jabberwocky (Ahmad et al., 2011) use programmatic paradigms to similarly allow for more optimal task designs. These tools are targeted toward organising and managing complex workflows. They are not suitable for most crowdsourcing applications and not fit for all tasks. As such, they are not popular in crowdsourcing.

FFV workflow has been adopted by a growing number of applications like BudgetFix (Tran-Thanh et al., 2015, 2014), linked data quality assessment (Acosta et al., 2013), CollabMap (Ramchurn et al., 2013), CrowdMap (Sarasua et al., 2012), PlateMate (Noronha et al., 2011), and many more. As such, this thesis targets those crowdsourcing applications that are built on top of FFV workflow.

2.1.7 CollabMap: An Exemplar FFV-based Crowdsourcing Application

CollabMap (Ramchurn et al., 2013) is an example of an FFV-based crowdsourcing application. It recruits people to augment existing maps (GoogleMaps or Ordnance Survey) by identifying buildings and drawing their evacuation routes to nearby roads. Participants are required to verify work undertaken by others by providing positive or negative votes on buildings and evacuation routes, helping CollabMap to determine their validity.

CollabMap offers two different views of an area to users: satellite imagery provided by Google Maps and panoramic views provided by Google Streetview. Figure 2.1 shows the shape of a building in yellow and two evacuation routes in green colour.

CollabMap was built upon the FFV workflow; in which first a user draws outlines of a building and then other set of users verify the outline; and similarly for evacuation routes. CollabMap achieves its basic functionalities in two discrete phases (as depicted in Figure 2.2):

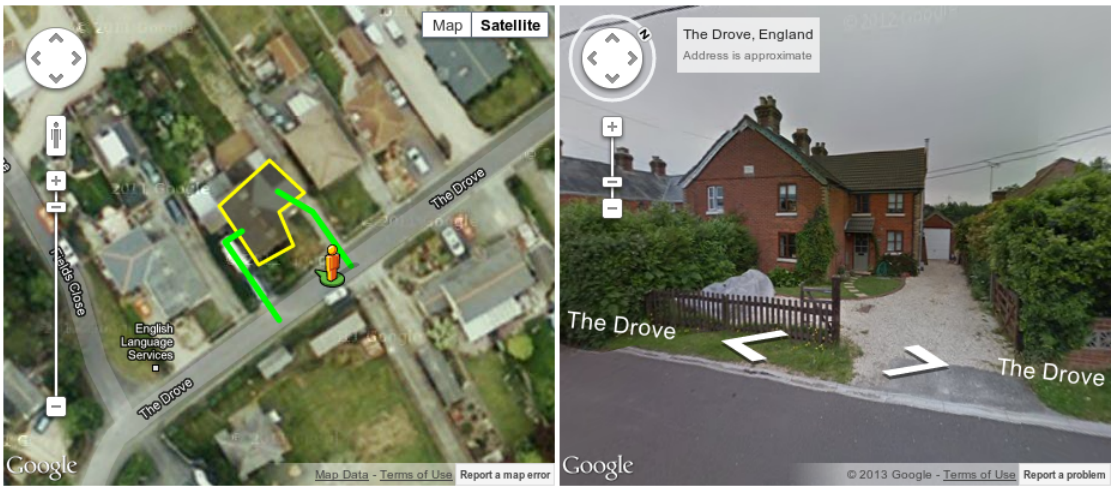


Figure 2.1: Satellite imagery and panoramic views in CollabMap.

- In the building phase, a building is identified by a user. Then other users verify the building by providing either positive or negative votes. If the total score of the building (total sum of positive and negative votes) is above +3, then the building is marked as a valid building. If the score reaches -2, the building is marked as invalid. After marking the building, building phase is finished and route phase is started.
- In the route phase, a user is asked to draw an evacuation route. This evacuation route connects an exit point of the building to a nearby road. This evacuation route is then added to the current "RouteSet". RouteSet is collections of routes belonging to a building. As in the building phase, other users verify all the evacuation routes of the building. Then, they are asked whether all necessary routes have been identified. If not, they can draw a new route. All the evacuation routes that were marked as valid by the user and the newly drawn evacuation routes are added to a new RouteSet.

CollabMap organises these phases around a task. A CollabMap task contains one building and all its evacuation routes. CollabMap follows the FFV workflow by breaking tasks into smaller and more manageable mini-tasks. Each mini-task is associated with a CollabMap activity. For example, a building identification is a mini-task.

Two roll-outs of CollabMap, separate from our work, have taken place and provenance of data is kept for each task. The first roll-out recruited local users from around the University of Southampton campus or the Fawley Area. The second roll-out was deployed on Amazon Mechanical Turk and relied on financially incentivized users from all over the world.

Figure 2.3 displays the provenance graph of one task in CollabMap[3]. According to this

[3]Note that the choice of colour in the provenance graph was made by the CollabMap application and it is not the Provenance Working Group's conventions

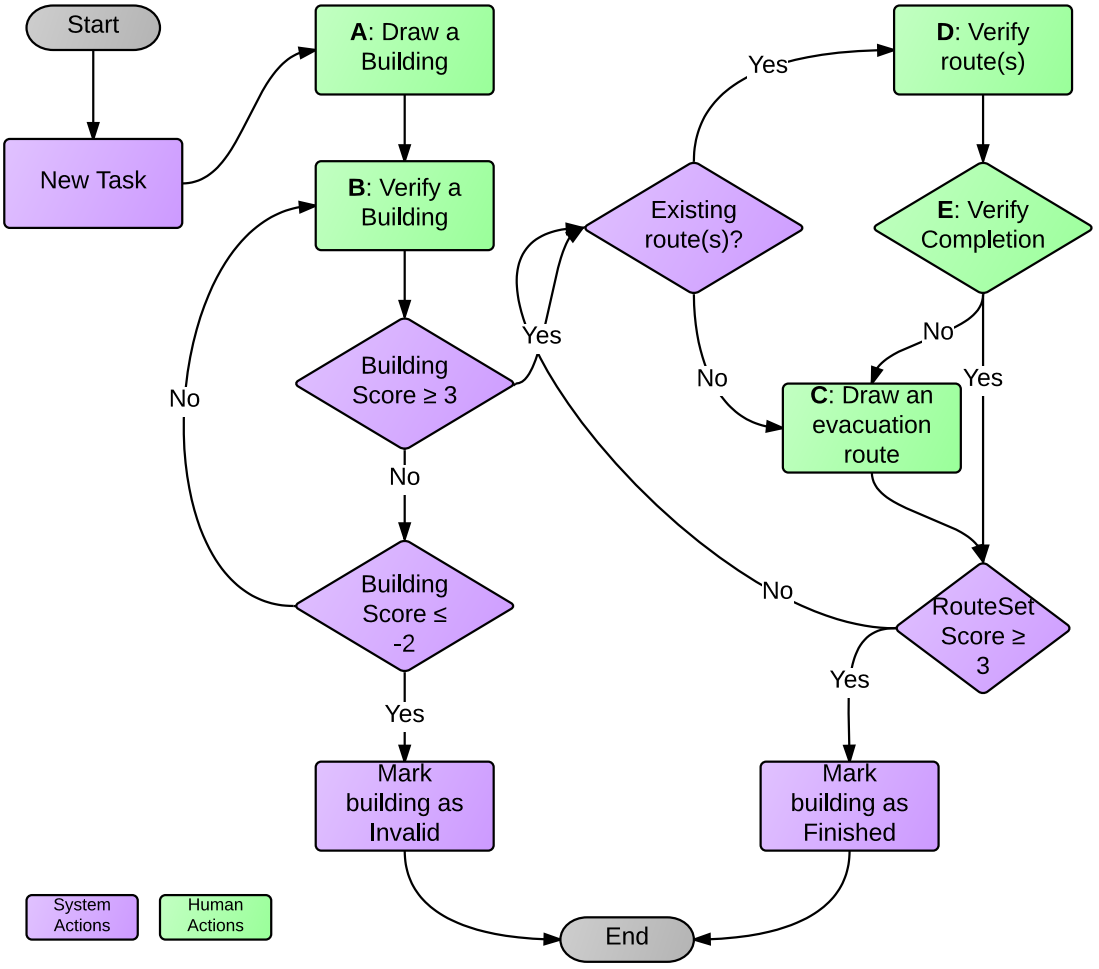


Figure 2.2: The CollabMap application workflow for identifying buildings and their evacuation routes (Ramchurn et al., 2013).

figure, Building832.0 was identified by user7 and two verifiers provided positive votes for the building (user6 and user1). Hence the building phase is finished and the building is marked as valid.

In the route phase, one evacuation route, Route1693.0 was identified by user5, and verified by two verifiers: user4 and user2.

The provenance graph shows how data were generated (building, routes, votes), how there were used by other activities (building verification used building), how they were derived from each other (votes were derived from a building), and how they were associated with the contributors (building identification was associated with a user).

2.2 Provenance

Moreau and Groth (2013b) highlight the change in the users' role in the World Wide

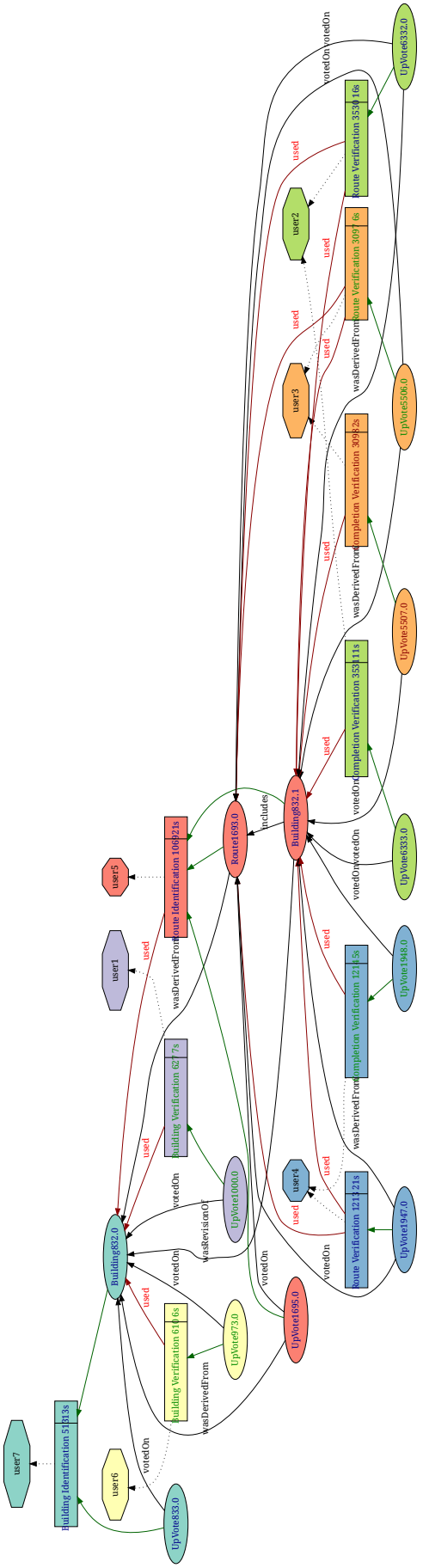


Figure 2.3: Provenance graph of one CollabMap task obtained directly from CollabMap Website[a].

[a]CollabMap: http://collabmap.org/

Web from passive consumers to active publishers and as such "from science to food manufacturing, from data journalism to personal well-being, from social media to art, there is a strong interest in provenance". Provenance of data is very useful, for example it can assist users to make a judgement as to whether something can be trusted. In this section, we shall discuss more about provenance, what it is, how it can be used, why it is useful, and an overview of provenance that is related to our work. We start by bringing the standard definition of provenance.

2.2.1 Definition of Provenance

Provenance is the origin or source of something (Moreau, 2010). However, due to the broad use of provenance, it is hard to have a universal definition. In this dissertation, we adopt the W3C (World Wide Web Consortium) Provenance Working Group's definition of provenance[4]:

> Provenance is defined as a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing.

In order to represent provenance of a system, we use the W3C PROV specifications: PROV Data Model (PROV-DM) and PROV Notation (PROV-N). We shall also provide an overview of PROV-DM in subsequent sections.

2.2.2 History of Provenance

Perhaps one of the earliest applications of provenance (from the French provenir, "to come from") is in the area of art. The primary purpose of recording and maintaining the provenance of an artifact is to provide contextual and circumstantial evidence for the production, discovery, and maintenance of that artifact (e.g. any form of art such as a portrait). The record showed information such as the sequences of artifact's former ownership, custody, and storage. For example, the aim to tracking the provenance of portraits is to authenticate them. A good provenance increases the value of a painting. If the provenance is lost or missing, the portrait is likely to be auctioned for a lower price (Moreau and Groth, 2013b). To the extend that, some galleries are trying to determine the provenance of the portraits for which the provenance is missing[5,6]. The

[4]Definition of Provenance: http://www.w3.org/TR/2013/REC-prov-dm-20130430/#dfn-provenance

[5]National Museum Directors' Conference (1998): http://www.nationalmuseums.org.uk/what-we-do/contributing-sector/spoliation/

[6]Museums across the UK have undertaken detailed research of their collections to identify objects with uncertain provenance between 1933-1945. Details of these objects are published in: http://www.culturalpropertyadvice.gov.uk/search_spoliations

provenance of a portrait establishes that the portrait has not been altered or it is not a forgery, and as such, provenance of the artifact allows the viewer or buyer to ascertain the authenticity of the artifact.

Provenance of a bottle of wine shows detailed information on the contents of the bottle, shedding light in both quality and the risk of wine fraud. With the hope that wine improves with age, provenance of the wine cellar is valuable in estimating the quality of an old vintage wine. This is an important issue to extend that third party companies have accepted to verify the authenticity and quality of an old wine by assessing the temperature and humidity history of the wine[7]. Given the provenance of a wine, the buyer of an old vintage wine can trust the seller by either looking at the verification certificate or going through the provenance of the wine.

2.2.3 Application of Provenance

The term provenance has been used in a wide range of fields, including archaeology, archives, science, and computing. In the following sections, we introduce some of these areas, mostly related to computing. These examples would make the case for the provenance, why it is important, and how it is being utilised.

Provenance and The Web

Nowadays, people take an active role in creating, editing, and rating information. For example, once, the responsibility of reporting news and analysing them was delivered by journalists. However, now it is easier to carry out this activity. There are many tools such as Social Networking Sites (Flipboard[8] and Facebook[9]) and blog-publishing services (such as Blogger[10]) that allow their users to easily write an article and make it available to millions of people. In particular, users of these services read other news and content on the Web, in various formats such as video, photos, audio, and text, and then write their own interpretation and analysis.

People participation in curating data has resulted in many successful services such as Wikipedia. Wikipedia is a free-access encyclopedia that allows people to curate the content available on the website. Wikipedia is the sixth-most popular website and constitutes the Internet's largest and most popular general reference work[11].

Perhaps, before the advent of the Web and Social Web, one of the main questions was "*how can I find an information on something?*". But now, one of the main questions

[7] Provenance of wines on Wikipedia: https://en.wikipedia.org/?title=Provenance#Wines
[8] Flipboard: https://flipboard.com/
[9] Facebook: https://www.facebook.com/
[10] Blogger: https://www.blogger.com/
[11] Wikipedia in Wikipedia: http://en.wikipedia.org/wiki/Wikipedia

arising from this gigantic amount of information is "*how can I trust this information I found?*".

The main issue is we do not know how information was produced and curated. We do not have enough information (meta-data) on the information we found, revealing how they were produced, who was involved, and if there is any evidence of other documents that can back up the information.

This is exactly where provenance is becoming very important. Jeff Jarvis[12] discusses the importance of provenance in data curation and states that:

> "Provenance is becoming more important in many fields. News, like art, requires provenance. Provenance is no longer merely the nicety of artists, academics, and wine makers. It is an ethic we expect."

Hence, in order to be able to answer "*how can I trust this information I found?*", one should not take the word of its creator, but should be able to see the source of that information, to track it, or be able to have access to the provenance of that information.

Provenance indicates who contributed to information, helps consumers to check where information came from, why it was selected, and how it was edited (Moreau and Groth, 2013b).

Provenance and Databases

Provenance is a well-known topic in the database community and several surveys exist on provenance in databases (James Cheney and Tan, 2007; Tan et al., 2007; Glavic and Dittrich, 2007).

Vast amounts of data are stored in databases. These databases are then queried to retrieve a subset of data and make it available to people. The query could be for various purposes such as query for e-science, query for company needs, or query based on people's requirements. However one question remains the same: *Which parts of the database contribute to the input according to the query?* Cui et al. (2000) offer an approach for computing data provenance in the relational frameworks that is applied to answer this question.

James Cheney and Tan (2007) review why, how, and where provenance. Why-provenance is based on the idea of providing information about the witness to a query and is formalized by Buneman et al. (2001). Why-provenance refers to the source data that had some influence on the existence of the data (Buneman et al., 2001).

[12]The importance of provenance: http://buzzmachine.com/2010/06/27/the-importance-of-provenance/

How-provenance, more detailed explanation as compared to the why-provenance, describes how the input tuples lead to the existence of the output tuple (Tan et al., 2007).

Where-provenance refers to the location(s) in the source databases from which the data was extracted (Buneman et al., 2001).

There are databases that integrate provenance in their data model such as DBNotes (Chiticariu et al., 2005), SPIDER (Alexe et al., 2006), or Trio System (Benjelloun et al., 2006). In the Trio System, provenance was utilised to manage the uncertainty over tuples. Benjelloun et al. (2006) states that the confidence of a value in a table can be computed from the confidence of the data in its provenance. More specifically, the ULDB model (Uncertainty Lineage Database) underlying Trio combines not only data, but also the accuracy and provenance of the data, in a database. The provenance is utilised to resolve uncertainty.

This is an important assertion on provenance that we also utilise in this dissertation. Later, we exploit the provenance of data in a crowdsourcing application to assess the quality of data.

In summary, provenance of data in databases can assure accuracy and currency of data. Also it can bring analysis of erroneous data by identifying the exact set of base data items that produced the data.

Given all these works, and more, on the integration of provenance in databases, we can envision a database management system that stores the data and the provenance of the data. Once the database is queried and the result is available, the provenance of the data can be attached to the result. In this way, the user understands where the result is coming from and how it was stored and curated in the database.

Provenance and Reproducibility

Researchers in the scientific community usually collect data from various resources, such as sensors, questionnaires, and etc, and then run their experiment to attain an end result. In science, the results of an experiment should be reproducible, under the model's assumption, by anyone. In this sense, provenance is regarded as the equivalent of a logbook by Moreau and Groth (2013b) that captures "all the steps that were involved in the actual derivation of a result, and which could be used to replay the execution that led to that result, so as to validate it". Miles et al. (2007) suggest that provenance can be utilised in as diverse domains as biology, physics, chemistry, and computer science.

Provenance and Accountability

People give their information to organizations in order to receive some services. This exchange of information takes place under a contract or policy that explains how this information is to be processed, stored, and maintained. Furthermore, users' information could be processed by many services in between, and as such, sometimes it is not clear where information is being passed. Hence, if irregularities happen, it would be difficult to understand who was accountable. The notion of accountability becomes important. Weitzner et al. (2008) state that "the use of information should be transparent so it is possible to determine whether a particular use is appropriate under a given set of rules, and that the system enables individuals and institutions to be held accountable for misuse". In this sense, Moreau (2011) claims a system is accountable if "it can provide explanations for its actions, if its past actions are accountable, and if it can be demonstrated that its processes and decisions are compatible with rules, policies, or broadly regulations".

With this in mind, provenance can be utilised as an evidence to show a system is accountable. For example, Aldeco-Pérez and Moreau (2008) propose a provenance-based architecture for an accountable system. The system utilised provenance to audit the processing of private data.

Various domains utilise provenance to see where the data is coming from, how it was altered, and who was responsible. In the next section, we offer an overview on a generic data model to represent provenance.

2.2.4 PROV-DM: PROV Data Model

In the previous sections, we provide an overview of provenance that is related to our work. In this section, we offer an overview of PROV Data Model (PROV-DM) which is a generic data model to represent provenance on the web. The W3C Provenance Working Group has published 13 documents related to provenance (Moreau and Groth, 2013a). This dissertation makes extensive use of two specifications: PROV-DM: PROV Data Model (Moreau et al., 2013a) and PROV-N: PROV Notation (Moreau et al., 2013b).

Instead of explaining PROV-DM and its types and relations, we introduce a scenario related to crowdsourcing and use it to show those PROV-DM types and relations that this dissertation uses. For a full explanation on PROV-DM, we refer the reader to Moreau et al. (2013a) and Moreau and Groth (2013b).

A News Generation Scenario

This scenario is involved with the creation of news by users and further the verification of each news item by other users. We selected this scenario for a number of reasons: (1) with this scenario, we are able to explain those parts of PROV-DM that this dissertation relies on, (2) the scenario is an example of crowdsourcing, and (3) provenance of data shows how news items are created and verified by users.

CrowdNews[13] is a service that allows people to report news. In order to make sure a news item is correct and genuine, the service asks other users to read the news item and vote on it. A user can provide a positive vote meaning the user thinks the news is correct or provide a negative vote meaning the user thinks the news is incorrect.

There are two types of characters in this scenario: (1) CrowdNews organisation and (2) people. Within CrowdNews, we have following sample people who created or verified a news item: Bob who created a news item and Alice and John who verified the news item.

The first part of this scenario is about Bob creating a news item. Bob found out that there was a flood incident from Twitter and Facebook messages. Later he got a call from a friend, giving him more information about the flood. Bob went to the CrowdNews website and submitted a report about this news.

In this news creation, there is a "Flood_News" artifact that we try to record the provenance of it. This artifact is called an *entity* in PROV terminology. An entity is defined as "a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary" (Moreau et al., 2013a). Other entities that can be identified are "Tweet", "Facebook_Message", and "Phone_Call".

"Flood_News" artifact was created based on a creation activity. There is a PROV terminology: *activity* which is defined as "something that occurs over a period of time and acts upon or with entities" (Moreau et al., 2013a).

The creation activity was associated with "Bob"; "Bob" was the person who created "Flood_News". In PROV terminology, "Bob" is an *agent*, defined as "something that bears some form of responsibility for an activity taking place" (Moreau et al., 2013a).

The "Flood_News" entity was created based on "Tweet", "Facebook_Message", and "Phone_Call". In PROV terminology, the relationship between these entities are called *derivation*; for example, "Flood_News" was derived from "Tweet".

[13]The name "CrowdNews" is not real.

Key	PROV-DM relation type
assoc	wasAssociatedWith
att	wasAttributedTo
gen	wasGeneratedBy
der	wasDerivedFrom
use	used
spe	specializationOf

Table 2.1: Mapping of edge key in PROV figures to PROV relation types.

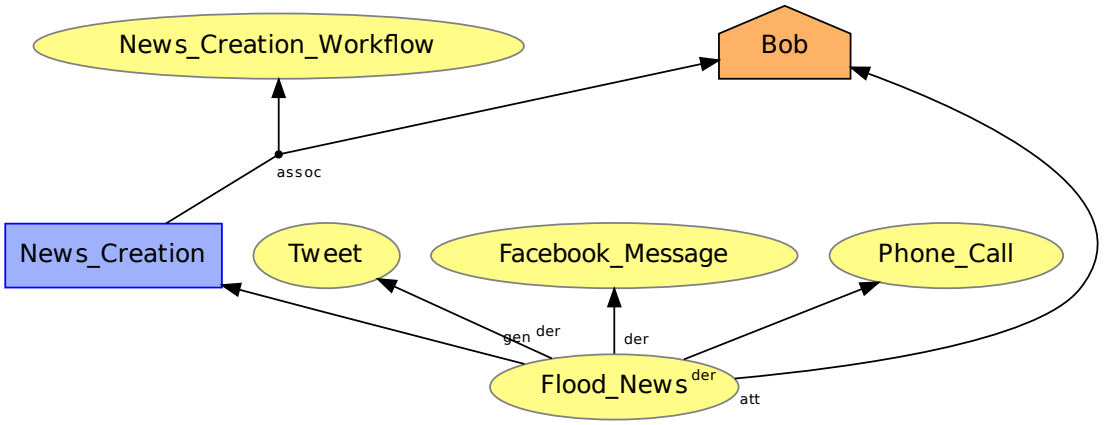


Figure 2.4: Provenance graph representing the provenance of news creation by Bob.

Figure 2.4 represents the provenance of data for the creation of "Flood_News" by "Bob". In order to show a graphical representation of provenance, we adopt here the Provenance Working Group's conventions[14] for presenting PROV graphically: an entity is represented by an ellipsis (yellow), an activity by a rectangle (blue), and an agent by a pentagon (orange).

Note that each edge in Figure 2.4 is associated with a key according to Table 2.1. Furthermore, it is noteworthy to mention that PROV-DM views provenance as a directed graph. In this view, provenance is expressed by a directed graph to show how data came to be produced. A directed graph is a set of nodes connected by directed edges. In this view, nodes are used to represent data items and edges are used to represent data derivations and relationships between data nodes.

Section A.1 presents the description of the news creation case written in PROV-N.

After this news item was created and submitted by "Bob", CrowdNews asked other people to read the news and verify it. "Alice" and "John" read the news and verified it. In the verification process, "vote1" entity was created by "News_verification_1" activity which was associated to "Alice". "vote1" entity was attributed to "Alice" as she provided "vote1". Furthermore, according to CrowdNews data model, "vote1" entity was voted on

[14]See http://www.w3.org/2011/prov/wiki/Diagrams for more information.

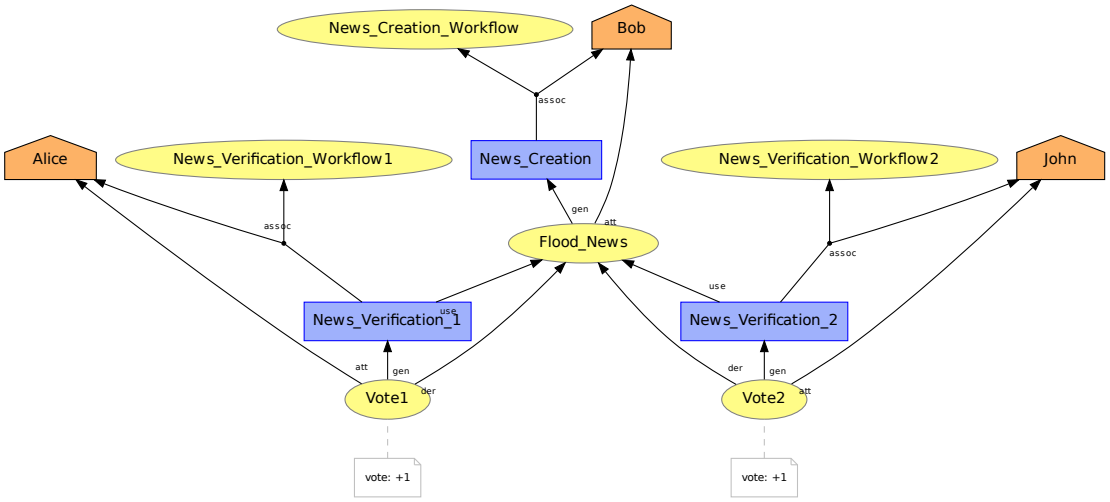


Figure 2.5: Provenance graph representing the provenance of news verification by Alice and John.

"Flood_News", and as such there is a derivation relation between these two entities. The process is the same for "John" and anyone else who might verify the news. Figure 2.5 represents the provenance of data for the verification of "Flood_News" by "Alice" and "John".

Section A.2 presents the description of the news verification case written in PROV-N.

Note that in Figure 2.5, both "vote1" and "vote2" entities are annotated with extra information. In PROV terminology, this extra information is called *attribute*.

2.3 Discussion on Trust and Quality

Trust is a topic of major interest in various fields such as Artificial Intelligence, Multi-Agent Systems, Information and Communication Technologies, and the World Wide Web. However, in computer science, trust is a term whose definition differs among researchers. Artz and Gil (2007) collect three general definitions:

- "The firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context" (Grandison and Sloman, 2000).
- "A subjective expectation an agent has about another's future behaviour based on the history of their encounters" (Mui et al., 2002).
- "Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X)" (Olmedilla et al., 2006).

In the subsequent sections, first, we shall look at the general trust models in the literature. These models allow us to have a better understanding of what has been done so far. However, our main objectives in this dissertation, is to develop a mechanism to assess the quality of data in crowdsourcing applications. Thus, after discussing some general trust models, we focus on some trust models or quality assessment approaches for crowdsourcing applications.

2.3.1 General Trust Models

One of the first works that offered a formal treatment of trust was the trust model proposed by Marsh (1994). The model is based on social properties of trust and integrates aspects of trust taken from sociology and psychology; one continuous value of trust in the range [-1, 1] can be computed. The model is complex and cannot be easily implemented, and, moreover, the model emphasises agents' own experiences such that an agent cannot build a network of trust (Aberer and Despotovic, 2001).

Trust is usually based on an agent's own experience with respect to past interactions with other agents, whereas reputation utilises information collected from third-parties (Moreau, 2010).

Reputation-based trust uses personal experience to make a trust decision about an entity. In lack of personal experience, when there was no interaction between truster and trustee, experience of others are sought, sometimes these two are combined, to make a trust judgement. Empowering users to make trust decisions, rather than relying on consultation with a central trusted third party, was discussed in Abdul-Rahman and Hailes (1997). They appreciate that selfish or malicious entities coexist with innocent ones and as such they developed a distributed trust model and a recommendation system. Some reputation-based trust models follow their ideas (such as Teacy et al. (2006); Yu and Singh (2003); Sabater and Sierra (2002)).

One of the main applications of trust and reputation-based evaluation is in Multi-Agent systems. Sabater and Sierra (2002) propose a system to combine reputation information from the individual and from others enabling users to define who can be trusted and for what they can be trusted.

An often used methodology for trust computation is Bayesian theory (for example the works done in Momani et al. (2008); Teacy et al. (2008); Nielsen et al. (2007); Quercia et al. (2006); Sun et al. (2006); Qi et al. (2005); Lahno (2004); Jøsang (2001)).

Some Bayesian systems take binary ratings as input. They are based on computing reputation scores by statistical updating of beta probability density functions (PDF) (Jsang et al., 2007). Beta distributions are used in Bayesian inference as they provide a family

of conjugate prior distributions for binomial distributions. Jsang and Ismail (2002) apply beta distribution to compute trust value by presenting an approach to take binary ratings as input and compute reputation scores by using beta PDF.

Based on this work, Teacy et al. (2006) look at the Grid, as a Multi-Agent system, to develop a model of trust and reputation to ensure good interactions among software agents given agents may be self-interested, provide false accounts of experiences, and need to interact with each other even though they have little or no past experience. In their presented system, TRAVOS (Trust and Reputation model for Agent-based Virtual OrganisationS), probability theory is applied to compute a trust value for an agent. Based on subjective experiences, a contract between two agents, truster and trustee, is either fulfilled, denoted by 1, or not, denoted by 0. Then the number of successful and unsuccessful interactions are used to compute the tendency of trustee to fulfil its obligations. An agent is considered to be trustworthy if it has a high probability of performing a particular action. They claim that given complete information is not accessible, and according to the Bayesian view, the best we can do is to use the expected value of success probability of a contract to be satisfied. In order to determine this expected value, and in Bayesian analysis, the beta family of probability density function is commonly used as a prior distribution for random variables that take on continuous values in the interval $[0, 1]$. The final trust value is computed by applying the standard equation for the expected value of a beta distribution (Teacy et al., 2006).

Guha et al. (2004) evaluate and rank several methods for propagating trust and distrust. They address the problem of distrust transitivity in which if A distrusts B and B distrusts C, we cannot say whether A trusts C. They assume a universe of n users each of which may assign positive (trust) and negative (distrust) ratings to each other. From these ratings, two real-valued matrices are constructed, one for trust and the other for distrust. The main goal is to predict an unknown trust/distrust value in these two matrices between any two users using the entries which are available.

Gray et al. (2003) consider trust formation and trust propagation in the context of security applications. They study how trust and recommendations can be propagated through relatively short paths connecting entities. In this trust-based security framework, the trust value among two nodes connected by a path is the average of the weighted sum of trust values of all nodes in the path.

Quercia et al. (2007) design a mechanism for trust propagation in order to form opinions about content producers with whom there might be no past interactions. They define trust propagation as the process of forming opinions. Their model works on a graph-based semi-supervised learning scheme. In this graph, nodes are either rated or unrated and are connected to each other if they are related. Two nodes are related if their ratings are similar. The model is able to get the knowledge which is already available in this graph to construct a function that is capable of predicting unrated nodes.

2.3.2 Crowdsourcing and Quality

The advantages of crowdsourcing are mainly in saving cost and increasing speed in data collection, however, the main challenge is to obtain data of high quality. To improve reliability with noisy data (due to individual biases or adversarial workers), one solution is to assign the same task to multiple workers (Bernstein et al., 2010); their responses are then fused together to estimate the correct response.

The *Majority Voting* method is the simplest consensus method for combining workers' responses (Sheng et al., 2008). This approach assumes that all workers are equal and no distinction is made between experts and novices. Furthermore, the *Majority Voting* approach looks at each task in isolation.

In order to solve these limitations, some are looking at the application of expectation maximisation (EM), naïve Bayes (Russell et al., 1995), or an adaptation of them to simultaneously estimate the final output and reliability of workers.

Dawid and Skene (1979) looked at the consistency between labellers for multivalued annotations. They modelled user's accuracy by a confusion matrix. Their solution is based on an EM algorithm which includes two steps: (1) In the E-step, they estimate the correct answer for each task based on all the labels that have been provided by multiple users. In this step, they also consider the expertise of each user. (2) In the M-step, they estimate the expertise (quality) of each user by comparing the submitted answers to the inferred correct answers. The final output of this system is all the estimated correct answers for each task and a confusion matrix for each user, showing expertise and error probabilities for each user.

Demartini et al. (2012) utilised EM in their system known as ZenCrowd. In this case, the reliability of workers enables the detection of adversarial workers. ZenCrowd combines the results of answers generated automatically with the answers by workers in order to link entities recognised in a text with entities in the Linked Open Data (LOD) cloud. Their approach is application-specific to LOD domain. Furthermore, they evaluate the system in an offline environment. Their approach is different from ours in yet another perspective in that they envision a learning phase to label workers and decide how to weight the worker's answers. This might not be always feasible as it requires a list of ground truths.

To this end, Snow et al. (2008) report on their system where an EM-based algorithm and a naïve Bayes classifier are employed to compute confusion matrices for each worker, which capture workers' error behaviour. Given the matrices, it is possible to detect and model workers who produce perfect response as one class and adversarial workers (opposite ones) as other class. More specifically, their technique can be employed for bias correction. They note that the reliability of users varies in which some users are accurate but some make mistakes. They put forth three ways to improve quality: (1)

more users can be recruited, (2) monetary rewards can be paid to highly reliable users and deny payments to unreliable ones, and (3) reliability and biases of users can be modelled and can be corrected. This work and many more in the literature support the assumptions being made in this dissertation:

Redundancy.
A task in the crowdsourcing application should be allocated to more than one worker.

Fusion.
The responses to a task coming from multiple workers should be fused together so that a final response (or result) is estimated.

Reliability.
Reliability of workers should be computed and should be considered as an important factor in quality assessment.

Snow et al. (2008) used AMT to collect labels on a variety of natural language processing tasks and then investigate the accuracy of collected labels. They compared the results obtained from AMT against labels that had been previously created by expert annotators. They reported a high agreement between AMT non-expert annotations and existing expert annotators.

This result is confirmed by the studies done by Sorokin and Forsyth (2008). They showed crowdsourcing data annotation task to AMT is not only cheap and quick but also the quality is good and can be controlled. They recommend three strategies to assess quality of data: The first strategy is to collect multiple annotations for each image. Although this strategy helps to reduce the possible errors and to find malicious users, cost of annotation is increased. Furthermore all these annotations should be combined so a true annotation can be estimated. The second strategy involves performing a grading task. In this strategy, a user is presented with several annotated images and is required to score every annotation. The third strategy is to build a gold standard; that is to build a collection of images with trusted annotations. The gold standard helps to evaluate users and catch adversarial users. Users are presented with images from gold standard from time to time, without knowing whether the image is coming from gold standard. If the annotations provided by the user vary significantly with the gold standard, the user is not performing what has been asked from him. Sorokin and Forsyth (2008) showed that users tried to accomplish the task they have been asked for and most of the disagreements come from difficult cases (when the question was difficult to answer).

Whitehill et al. (2009) identified three main challenges in using crowdsourcing infrastructures like AMT: (1) Users have wide ranging levels of expertise; some users may perform well and some may be adversarial; (2) tasks may vary in their level of difficulty;

and (3) multiple labels for the same task must be combined to provide an estimate of the actual label. To remedy these challenges, they employ EM to estimate the final output, expertise of workers, and difficulty of the task, in their system called GLAD. However, this algorithm requires an initial starting point which is typically randomly guessed. The algorithm is highly sensitive to this initialization, making it difficult to predict the quality of the resulting estimate (Karger et al., 2011). Furthermore this approach does not take different activities in crowdsourcing applications into account (for example creating and scoring of data products).

Similarly, Welinder and Perona (2010) propose a model based on the EM algorithm that infers the final output and computes a multi-dimensional measure of the workers' ability. From this model, they derive an online algorithm that finds and prioritizes experts when requesting labels. This model is applicable in labelling applications where a set of images are required to be labelled. Furthermore, they assume that a pool of workers are available. This assumption does not hold true in crowdsourcing platforms such as Amazon Mechanical Turk where workers are not always online; they look for tasks to undertake and then leave the platform. Moreover, the online algorithm was evaluated in a simulated environment without evaluating the performance of the algorithm itself. Unlike the quality model proposed in that paper, our system does not assume a specific application domain and by using provenance of data, our approach does not rely on the data model deployed in the crowdsourcing application. Furthermore, the quality model does not rely on the availability of a pool of workers. The online algorithm (*PEDRA-O*) was evaluated in a real online environment and its performance, in terms of response time, communication, and storage, was evaluated.

Sheshadri and Lease (2013) classify the above methods into: (1) those that model workers' noise and expertise (GLAD, ZenCrowd), (2) those that model worker biases (EM algorithm, naïve Bayes, and ZenCrowd), and (3) those that additionally consider task difficulty (GLAD). Welinder et al. (2010) incorporated all of these features into a same model. Hence, it is not only able to estimate the underlying true output, but also it considers each task to have different characteristics, and each worker to have different variables such as competence, expertise, and bias.

None of the quality models introduced above fits to our objectives introduced in Section 1.6, mainly because, either tasks are being considered in isolation and not thorough the application, reliability of workers are not being considered, different activities (creation and scoring) are not considered, the model is not generic and rely on application data model or is appropriate for certain application domain, or the quality model has not been evaluated in an online environment. More specifically:

- The *Majority Voting* method assumes that all workers are equal and no distinction is made between experts and novices.

- The approach offered by Demartini et al. (2012) is domain-specific to Linked Open Data domain. Furthermore, this approach requires a learning phase to label workers and decide how to weight the worker's answers.
- Snow et al. (2008) offers a quality assessment approach for crowdsourcing applications, but their approach is domain-specific to natural language processing tasks.
- Whitehill et al. (2009) does not take different activities in crowdsourcing applications into account (for example creating and scoring of data products). Furthermore, their approach is specific to image labelling.
- Both Welinder and Perona (2010) and Welinder et al. (2010) offer quality assessment techniques for crowdsourcing applications but both techniques are domain-specific to labelling applications and they assume a pool of workers are available. Furthermore, they have not considered different kind of activities in a crowdsourcing application (creation and scoring of data products).

Finally, Simpson et al. (2011) present a Bayesian-based classifier that can be utilised to group workers according to their expertise. It is beneficial to classify workers (e.g. experts, novices, or adversarial) so that tasks can be allocated according to their expertise. Simpson et al. (2011) estimate the end result of a task using IBBC (Ghahramani and Kim, 2003) and then identify clusters of users. Therefore, the estimation of the ground truth is done without considering the clustering structure of the workers (Moreno et al., 2014). Furthermore, Simpson et al. (2011) do not differentiate between cases when a data product is created and when a data product is scored, assuming that reliability of a worker in data product creation is different than scoring (i.e. verifying). Simpson et al. (2011) assume individual tasks are independent from each other; as such, their solution cannot be applied to problems involving inter-dependent micro-tasks (Tran-Thanh et al., 2015). For example, Bernstein et al. (2010) utilise Find-Fix-Verify workflow to design a crowdsourcing application where a sentence is presented to workers to find the mistakes in a sentence. Then, these mistakes are presented to other workers so that they can fix the mistakes. At the end, another set of workers verify the fixes (similar workflow is employed in Lin et al. (2012), Ramchurn et al. (2013), and other).

In Section 2.4, we will identify important factors when quality of data is required to be assessed in a crowdsourcing application. Then, we will identify a set of shortcomings in the literature that motivate the work presented in this dissertation.

2.3.3 Provenance and Quality

In this paper, we put forth a quality assessment approach that exploits provenance of data to assess the quality of data. Provenance has been previously used in a number

of contexts for assessing the credibility of data and helping users to make trust judgements (Ainy et al., 2014; Moreau et al., 2013a; Moreau, 2010; Golbeck, 2006a).

Huang (2008) considers the issue of determining the validity and origin of knowledge on the Web. They introduce knowledge provenance consisting standards and processes of how to model and maintain the evolution and validity of knowledge. This model works on truth value of propositions in the Web. They provide an ontology and semantics to give a formal and explicit specification of their model.

Golbeck (2006a) considers trust inference in the Semantic Web by utilising provenance. They suggest a two level approach to integrate trust, provenance, and annotations in Semantic Web systems. First, provenance of existing trust annotations in social networks are used to compute trust recommendations. Then these values are combined with the provenance of other annotations to personalise content.

Dai et al. (2008) propose a data provenance trust model that takes into account various factors and, based on them, assigns trust scores to both data and data providers.

Prat and Madnick (2008) argue that believability is an important aspect of data quality and then provide a precise approach to compute it using provenance.

Willett et al. (2013) propose an approach to make the provenance of data available to the task executor in order to help them understand if the generated data by the worker is reliable or not.

More recently, provenance has been incorporated into crowdsourcing applications. For example, Huynh et al. (2013a) present an application-independent methodology to analyse provenance graphs to extract some network metrics that correlate to trust. They evaluated their approach using CollabMap (Ramchurn et al., 2013) after CollabMap execution was finished and the dataset was ready. Their approach seems to be applicable after the execution of a crowdsourcing application is finished, and as such, cannot be employed while the crowdsourcing application is executing live and provide online feedback to the application so that it can improve its behaviour.

Stamatogiannakis et al. (2014) highlight that in order to produce trusted results, we require both a trustworthy execution environment and trustworthy data. Therefore, they mention provenance plays an important role in ascertaining trustworthiness of data. They offer an approach to capture and reconstruct provenance, thus building trust on both newly generated and existing data.

Baillie et al. (2015) present a model for quality assessment over linked data known as QUAL. This provenance-aware quality model allows an agent to reason about data provenance when performing quality assessment.

Miles and Griffiths (2015) state that the context of previous interactions contains information that could be valuable for reputation assessment. This statement is the foundation of our approach as well where we assume that a worker's past performance is an indicator of their reliability and how they may perform in the future (as backed by Teacy et al. (2006)). However, Miles and Griffiths (2015) claim that existing methods do not fully consider the circumstances in which agents have previously acted. As such, they propose a provenance-based approach to reputation assessment, using a set of patterns that describe the circumstances to determine the relevance of past interactions. Basically, they look into patterns in the provenance that indicate situations relevant to the current client's needs and mitigating circumstances affecting the providers.

2.4 Summary

In this chapter, we divided the discussion into three main areas, each of which are key points related to this dissertation.

First, in Section 2.1, we introduce crowdsourcing as a process of obtaining needed services by recruiting contributions from a large group of people. After a brief overview on the history of crowdsourcing, we focused on application of crowdsourcing. Although we cannot bring an example from each domain, this section shows the wide range of domains in which crowdsourcing has been utilised. At the end, we introduced some concerns over the use of crowdsourcing and we mentioned that this dissertation, in part, would offer a generic mechanism to resolve the concern on quality of data.

Second, in Section 2.2, we introduced provenance as the origin or source of something that can be used to form assessments about its quality, reliability or trustworthiness. After a brief overview on the history of provenance, we focused on the application and definition of provenance. At the end, we introduced PROV-DM as a generic and domain-agnostic data model to model and capture provenance of a system.

Third, in Section 2.3, we presented a discussion on trust and quality. After introducing some general trust models in the literature, we focused on quality issues in crowdsourcing applications and what has been done so far. We further surveyed the literature on works that exploit provenance of data to compute a notion of quality.

Based on the literature review, we identify the following as important factors when quality of data is required to be assessed in a crowdsourcing application:

1. A task should be allocated to more than one worker for verification purposes (Bernstein et al., 2010; Sheng et al., 2008; Sorokin and Forsyth, 2008).
2. The reliability of each worker is required to be assessed and considered in the data quality assessment (Demartini et al., 2012; Snow et al., 2008)

3. A quality assessment approach is required to consider that there might be different kind of activities in a crowdsourcing application and as such is required to consider them when computing a reliability measure for workers (Welinder and Perona, 2010).

4. The verification data supplied by each worker on a task is required to be merged (or fused), so that a final output can be computed, and for this purpose, a probabilistic model can be applied (Teacy et al., 2006; Whitehill et al., 2009; Welinder and Perona, 2010)

Following our discussion on the literature in crowdsourcing, provenance, and quality assessment and based on the above four factors, we identify the following shortcomings that motivate the work presented in the proceeding chapters:

1. Workers' reliability: Some of the quality assessment approaches for crowdsourcing applications evaluate the quality of the data without taking workers' reliability into account (e.g. popular *Majority Voting* approach, refer to Section 2.3.2). As such, it is required to develop a quality assessment approach that considers the reliability of workers as well. In this way, the approach makes a distinction between verification of reliable and unreliable workers. Furthermore, the approach is able to identify adversarial workers.

2. Interoperability: In the literature, there is a lack of interoperability with tools due to application-specific data structures. For example, the trust models suggested in the literature (refer to Section 2.3.2) are not generic and are designed for a specific crowdsourcing domain. As such, it is required to develop a quality assessment approach that is generic (i.e. it does not rely on the design of the crowdsourcing application, neither on its data model). In this way, there is no need to change the quality assessment approach for every crowdsourcing application.

3. Online feedback: There are some quality assessment approaches in the literature that evaluate the quality of data generated in a crowdsourcing application after the execution of the application is finished, i.e. offline environment, (refer to Section 2.3.2). However, the performance of a quality assessment approach in providing online feedback to the crowdsourcing application while it is executing live has not been evaluated in a real-world setting (i.e. online environment). The online deployment of a quality assessment approach is in the pursuit of improving the application's performance.

We should note that we employ probability theory and a heuristic technique in our quality assessment approach. We assume the outcome of an interaction (e.g. in the verification activity of a crowdsourcing application) is represented as a binary variable[15].

In the next chapter, we focus on how provenance can be utilised to capture what happened in the system, what was generated, and who was responsible. Provenance is also employed to capture the full history of workers' interactions that later is exploited so that the quality of data is assessed.

[15]Representing the outcome of an interaction as a binary variable is a simplification made for the purpose of our work that covers variety of crowdsourcing applications. In certain circumstances, a more expressive representation may be appropriate which is part of our future work.

Chapter 3

Provenance Patterns for Crowdsourcing Applications

According to the objectives in Chapter 1, we aim to develop a provenance-enriched quality assessment approach for crowdsourcing applications that are built on the Find-Fix-Verify (FFV) workflow.

Earlier, we mentioned that the quality assessment in FFV-based crowdsourcing applications can be done based on the users' verifications and how good a user has been performing in the application (i.e. the reliability of a user). We assume that a user's past performance is an indicator of their reliability and how they may perform in the future. In order to record users' past interactions and the verification activities in a crowdsourcing application, we propose to record the provenance of data and exploit it to assess the quality of data and reliability of users.

In order to exploit the provenance of data for any FFV-based crowdsourcing application, in this chapter, we propose a set of provenance patterns that shape the provenance of data. More specifically, and recalling from Chapter 1, the contribution of this chapter is as follows:

Contribution We specify a set of generic patterns that shape the provenance of data for different activities that are undertaken in a FFV-based crowdsourcing application. More specifically, *create pattern* shapes the provenance of data when data is created; *score pattern* shapes the provenance of data when data is verified or scored; *revision pattern* and *agent pattern* shape the provenance of data given a data product or a worker evolve through the execution of a crowdsourcing application.

This chapter consists of six parts:

- Section 3.1 introduces the generic provenance patterns that shape the provenance of data in any FFV-based crowdsourcing application.
- Section 3.2 discusses the requirements of provenance patterns.
- Section 3.3 presents the provenance patterns.
- Section 3.4 realises the provenance patterns in CollabMap (refer to Section 2.1.7) to shape the provenance of data generated by this crowdsourcing application.
- Section 3.5 offers a discussion on the provenance patterns.
- Section 3.6 summarises this chapter.

3.1 An Overview on Provenance Patterns

Bernstein et al. (2010) introduced the FFV workflow as one method of utilising crowds to complete open-ended tasks. It consists of three phases in which workers can make different types of contributions. In the *find* phase, participants are requested to identify patches of work that require more attention. In the *fix* phase, workers focus on an identified area, and perform manipulations to "fix" it. Finally, the *verify* phase is concerned with quality control, and relies on workers' voting on the fixes produced in the previous phase.

While the distinction between the find and fix phases is significant in crowdsourcing applications to classify workers' contributions and to determine the kind of worker enrolment and voting that is required, from a provenance perspective, such phases simply result in new data products. On the other hand, quality control relies on votes or ratings, both of which are subsumed under the category of scores. Data products can undergo revisions, whereas votes relate to specific versions of data products. Workers themselves get successively allocated tasks, and their expertise and performance may vary over time: hence, from a provenance perspective workers can also be seen as versioned; having multiple versions at different points in time.

Based on this background, we design four templates, which we call provenance patterns, inspired from PROV-TEMPLATE (Michaelides et al., 2014). We recognised that the provenance generated in a crowdsourcing application follows patterns that are repeated during the lifetime of an application. Thus, we propose the provenance patterns in Section 3.3. Similar to Michaelides et al. (2014), each provenance pattern specifies some variables acting as placeholder for values to be specified.

The provenance patterns are required to cover any FFV-based crowdsourcing application where a data product is created by a worker and scored by multiple workers. Furthermore, they are required to be generic. In each pattern, there should be some identifiers as variables (or placeholders) to be instantiated by values.

3.2 Provenance Patterns Requirements

This section lists all the requirements of the provenance patterns.

Requirement 1 The provenance patterns are required to cover any FFV-based crowdsourcing application.

> Given the research objectives introduced in Section 1.6, the provenance patterns introduced in this section should cover any FFV-based crowdsourcing application where a data product is created by a worker and verified (or scored) by other workers.

Requirement 2 The provenance patterns are required to be generic.

> The provenance patterns should not rely on the data model a crowdsourcing application chooses, neither on the domain the crowdsourcing application is operating.

Requirement 3 The provenance patterns are required to shape the provenance of data when data was created by a worker for any FFV-based crowdsourcing application.

> According to the FFV workflow, a crowdsourcing application assigns a task to a worker, requiring the worker to create a data product. As such, a provenance template should provide placeholders for the crowdsourcing application to fill, which are in regards to the data product creation. In this way, the provenance of data is captured and it is possible to exploit the provenance to retrieve what was created and who created it.

Requirement 4 The provenance patterns are required to shape the provenance of data when data was verified (i.e. scored) by a worker for any FFV-based crowdsourcing application.

> According to the FFV workflow, a crowdsourcing application assigns a task to a worker, requiring the worker to verify (i.e. score) a data product. As such, a provenance template should provide placeholders for the crowdsourcing application to fill, which are in regards to the data product scoring. In this way, the provenance of data is captured and it is possible to exploit the provenance to retrieve what was scored, who scored it, and what score was provided.

Requirement 5 The provenance patterns are required to shape the provenance of data when a data product or a worker is undergoing revision (and more information regarding them is becoming available) for any FFV-based crowdsourcing application.

As workers verify a data product, more information regarding that data product is becoming available; i.e. the data product is undergoing revision. Same holds true for workers: as a worker collaborates with the crowdsourcing application, more information about worker's performance is becoming available; i.e. the worker is undergoing revision. Hence, from a provenance perspective, data products and workers can be seen as versioned; having multiple versions at different points in time. As such, a provenance template should provide placeholders for the crowdsourcing application to fill, which are in regards to tracking different versions of a data product or a worker.

3.3 Provenance Patterns

3.3.1 Create Pattern

In this section, we introduce *create pattern*.

Intent

Shape the provenance of data when something is created by a worker.

Motivation

Consider a crowdsourcing application that allocates a task to a worker. The worker is required to create something (based on the FFV workflow). For example, the worker creates a news entry (refer to the "A News Generation Scenario" introduced in Section 2.2.4), or identifies a mistake in a sentence. The question is how to generate the provenance of this data generation for any FFV-based crowdsourcing application.

We solve this problem by offering a template (a provenance pattern) and specifying some variables acting as placeholder for values to be specified. The pattern is called *create pattern* which is concerned with a data product (dp.v) being created and attributed to an agent (agent.v), on the basis of some input (input.v), in the course of an activity executed according to some workflow or plan (plan.v).

This provenance pattern requires the crowdsourcing application to fill the values for the following placeholders:

1. dp.v: what was created. The provenance element associated to this item is of type DataProduct which specifies this provenance element is a data product.

2. input.v: whether there was additional input. The provenance element associated to this item is of type CreationInput which specifies this provenance element is an input of the data product creation.

3. plan.v: under what plan the creation happened. The provenance element associated to this item is of type CreationPlan which specifies this provenance element is a plan for the creation of the data product.

4. agent.v: who was involved in the creation process. The provenance element associated to this item is of type Creator which specifies this agent created the data product.

5. activity: the activity that happened which resulted into the creation of the data product. The provenance element associated to this item is of type CreationActivity which specifies this provenance element is a creation activity.

Note that there is a "v" in the name of some artifacts, such as a data product (dp.v) or a worker (agent.v). This means that specific artifact is versioned. Refer to *revision pattern* (Section 3.3.3) and *agent pattern* (Section 3.3.4) for further information.

Provenance Graph

Figure 3.1 shows the provenance graph for *create pattern*. It can be seen in this figure that each provenance element (entities, activities, and agents) is of type as described earlier.

PROV Notation

Listing 3.1 is the PROV-N representation of *create pattern*.

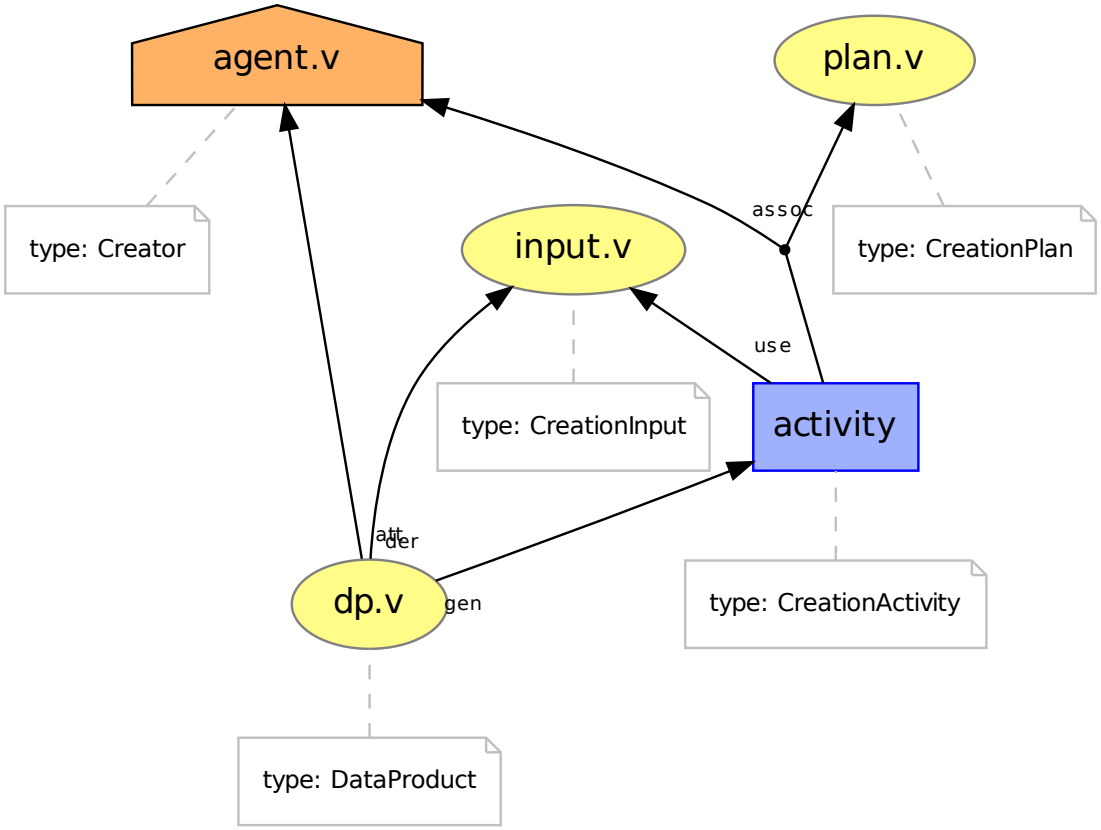


Figure 3.1: Provenance graph shaped by create pattern.

```
document
  prefix var <https://provenance.ecs.soton.ac.uk/var#>
  prefix rating <https://provenance.ecs.soton.ac.uk/ns/rating#>

  activity(var:activity,-,-,[prov:type = 'rating:CreationActivity'])
  entity(var:plan.v,[prov:type = 'rating:CreationPlan'])
  entity(var:input.v,[prov:type = 'rating:CreationInput'])
  entity(var:dp.v,[prov:type = 'rating:DataProduct'])
  agent(var:agent.v,[prov:type = 'rating:Creator'])

  wasAssociatedWith(var:waw1;var:activity,var:agent.v,var:plan.v)
  used(var:use1;var:activity,var:input.v,-)
  wasGeneratedBy(var:wgb1;var:dp.v,var:activity,-)
  wasAttributedTo(var:wat1;var:dp.v, var:agent.v)
  wasDerivedFrom(var:wdf1;var:dp.v, var:input.v)
endDocument
```

Listing 3.1: Provenance of data shaped by create pattern in PROV-N representation

3.3.2 Score Pattern

In this section, we introduce *score pattern*.

Intent

Shape the provenance of data when something is scored or verified by a worker.

Motivation

Consider when a crowdsourcing application allocates a task to a worker in which something is required to be verified or scored. For example, after a news item is created by a worker, another worker is required to read it and verify the correctness of the item (refer to the "A News Generation Scenario" introduced in Section 2.2.4). The question is how to generate provenance of this data verification for any FFV-based crowdsourcing application.

We solve this problem by offering a template (a provenance pattern) and specifying some variables acting as placeholder for values to be specified. The pattern is called *score pattern* which is about the scoring of a data product (dp.v) by an agent (agent.v); the scoring is performed according to a plan (plan.v). With the term score (score), we include notions such as "votes" or "ratings". A new score would result in a new version of the data product.

This provenance pattern requires the crowdsourcing application to fill the values for the following placeholders:

1. agent.v: who was involved in the scoring process. The provenance element associated to this item is of type Scorer which specifies this agent scored the data product.
2. plan.v: under what plan the scoring happened. The provenance element associated to this item is of type ScoringPlan which specifies this provenance element is a plan for the scoring of the data product.
3. dp.v: what was scored. The provenance element associated to this item is of type DataProduct which specifies this provenance element is the data product which was scored.
4. score: what was the score provided by the scorer. The provenance element associated to this item is of type Score which specifies this provenance element is a score. Note that the *prov:value* of this provenance element must be set to the value of the score which is either $+1$ or -1.
5. activity: the activity that happened which resulted into the creation of score. The provenance element associated to this item is of type ScoringActivity which specifies this provenance element is a scoring activity.

Note that there is a "v" in the name of some artifacts, such as a data product (dp.v) or a worker (agent.v). This means that specific artifact is versioned. Refer to *revision pattern* (Section 3.3.3) and *agent pattern* (Section 3.3.4) for further information.

It is important to mention that each time there is a score (i.e. each time a worker verifies a data product), there is a new version of the data product. Therefore, both the score provided by the worker and that version of the data product are attributed to the version of the worker (note that each data product or worker has different versions; refer to Section 3.3.3 and Section 3.3.4 for a discussion on versions).

Provenance Graph

Figure 3.2 shows the provenance graph for *score pattern*. It can be seen in this figure that each provenance element (entities, activities, and agents) is of type as described earlier. Note that the *prov:value* of score is set to X which is a placeholder for the value of the score, either $+1$ or -1.

PROV Notation

Listing 3.2 is the PROV-N representation of *score pattern*.

```
document
  prefix var <https://provenance.ecs.soton.ac.uk/var#>
  prefix rating <https://provenance.ecs.soton.ac.uk/ns/rating#>

  activity(var:activity,-,-,[prov:type = 'rating:ScoringActivity'])
  entity(var:plan.v,[prov:type = 'rating:ScoringPlan'])
  entity(var:score,[prov:type = 'rating:Score', prov:value = 'X'])
  entity(var:dp.v,[prov:type = 'rating:DataProduct'])
  agent(var:agent.v,[prov:type = 'rating:Scorer'])

  wasAssociatedWith(var:waw1;var:activity,var:agent.v,var:plan.v)
  used(var:use1;var:activity,var:dp.v,-)
  wasGeneratedBy(var:wgb1;var:score,var:activity,-)
  wasAttributedTo(var:wat1;var:score, var:agent.v)
  wasAttributedTo(var:wat2;var:dp.v, var:agent.v)
  wasDerivedFrom(var:wdf1;var:score, var:dp.v,[prov:type = 'rating:Scoring'])
endDocument
```

Listing 3.2: Provenance of data shaped by score pattern in PROV-N representation

Note that the derivation relation from score to dataProduct.v is of type "Scoring".

3.3.3 Revision Pattern

In this section, we introduce *revision pattern*.

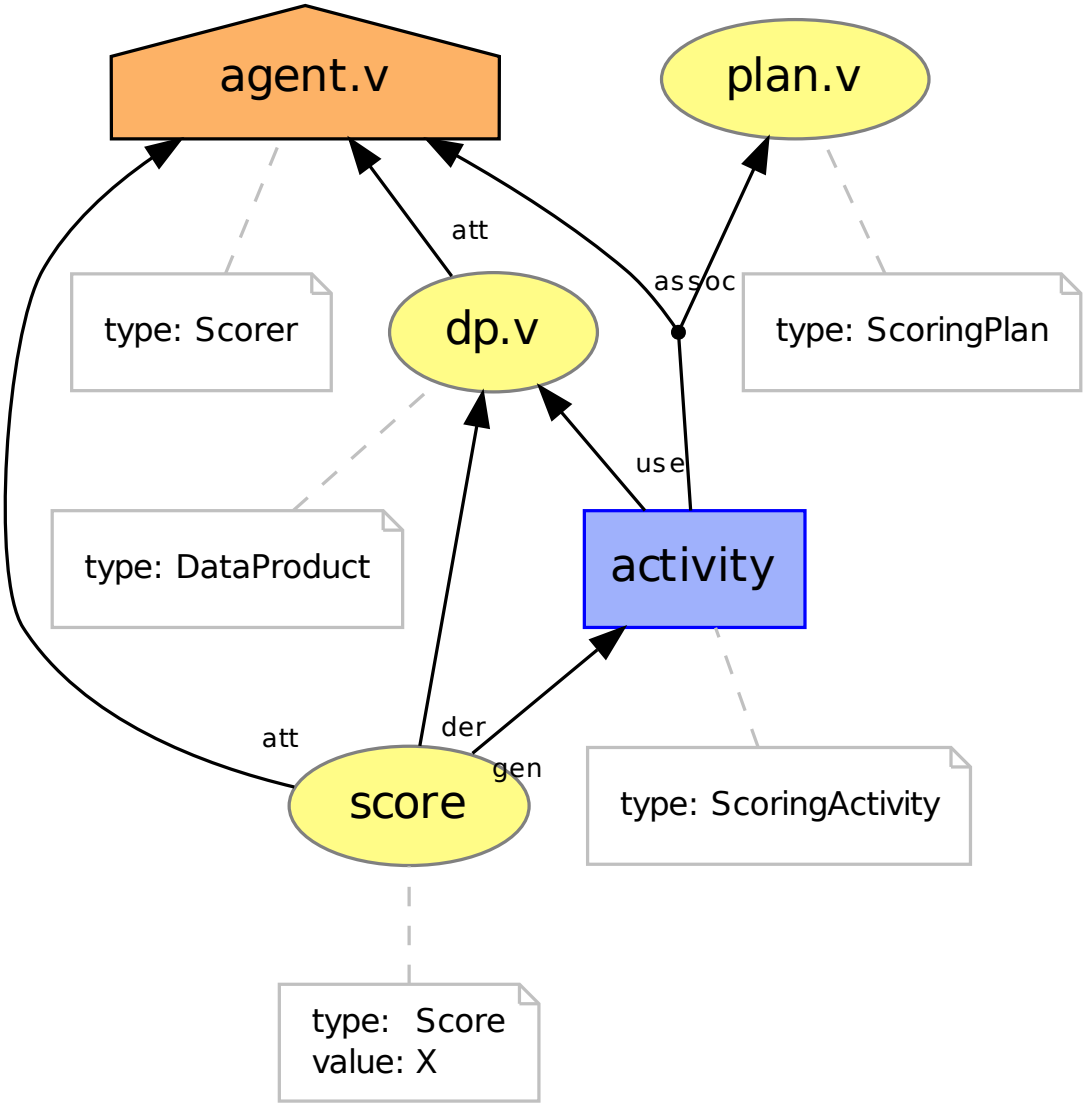


Figure 3.2: Provenance graph shaped by score pattern.

Intent

Shape the provenance of data when something evolves over time.

Motivation

Consider a data product that was created by a worker. This data product is required to be verified by other workers. At each verification, there are more information in regards to the data product that can be used for variety of purposes, such as quality assessment. The question is how to shape the provenance of data when data evolve over time.

We solve this problem by offering a template (a provenance pattern) and specifying some variables acting as placeholder for value to be specified. The pattern is called *revision*

pattern, which shows that all artifacts, such as a data product, evolve over time, and therefore, are versioned. Thus, the *create pattern* results in a new specific version of a data product, based on an input, (which can also be its previous version). Likewise, the *score pattern* allows for the scoring of a given version of a data product.

This provenance pattern requires the crowdsourcing application to fill the values for the following placeholders:

1. dp: which is the main version-less data product. The provenance element associated to this item is of type DataProduct which specifies this provenance element is a data product.
2. dp.v1: which is a version of a data product. The provenance element associated to this item is of type DataProductVersioned which specifies this provenance element is a version of a data product.
3. dp.v2: which is a version of a data product. The provenance element associated to this item is of type DataProductVersioned which specifies this provenance element is a version of a data product.

Being able to distinguish the different versions of artifacts, such as a data product, allows provenance-enabled crowdsourcing applications to define quality measures over these, which in turn can be used for data quality assessment. The provenance of data also contains a data product which may be referred to irrespectively of its actual versions. A specific version is said to be a specialization of the general/non-versioned artifact.

The *revision pattern* is known as "PLAN FOR REVISIONS" (Moreau and Groth, 2013b) where they offered a recipe to answer "How does one express revisions to a resource or document using PROV?". This recipe allows the provenance to be described at multiple levels of abstraction.

We define a data product as some data plus some ratings (i.e. measures), which altogether, we consider as an entity, for the purpose of provenance. Each version of a data product contains a set of fixed attributes that describe more information about that data product at a certain period of time (such as the validity of the data product).

It is noteworthy to mention that the data is not changing (i.e. the worker does not provide new information on the data product), but because scores are added and therefore, ratings (i.e. measures) are changing, we look at them as different entities.

Provenance Graph

Figure 3.3 shows the provenance graph for *revision pattern*. It can be seen in this figure that each provenance entity is of type as described earlier.

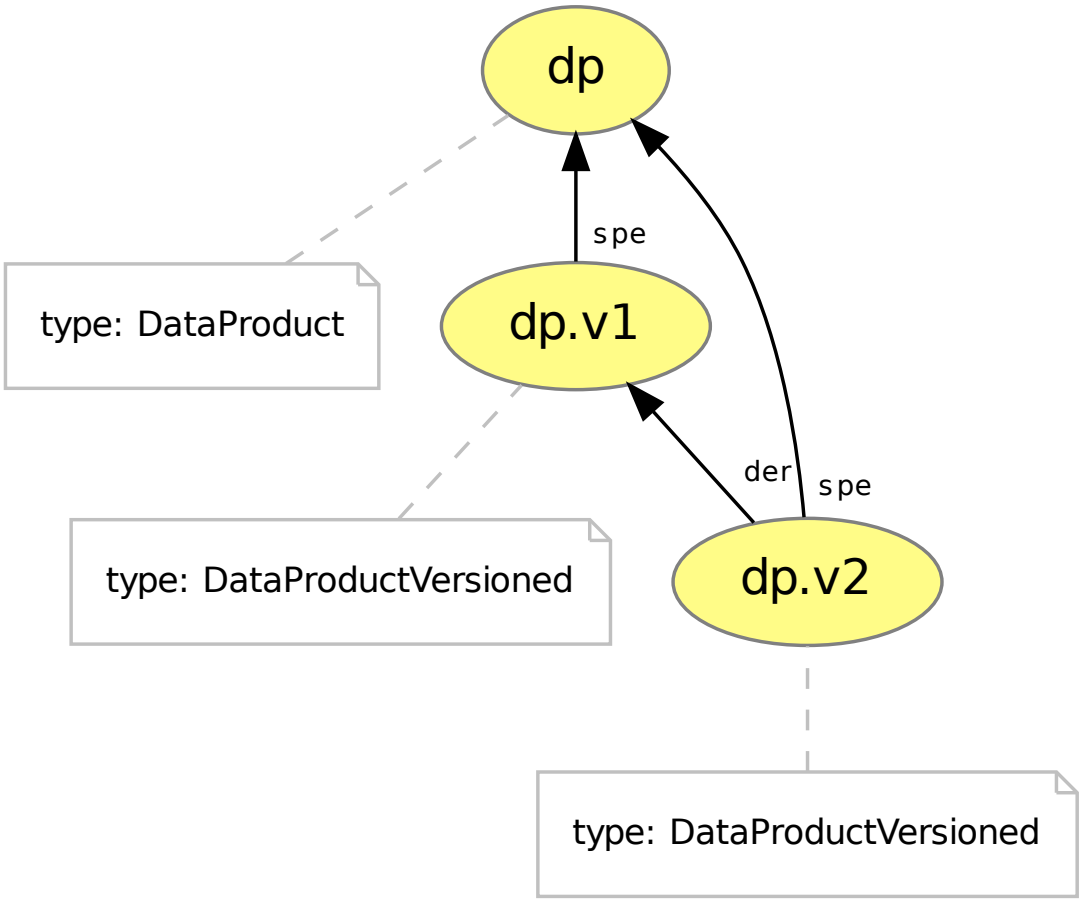


Figure 3.3: Provenance graph shaped by revision pattern.

PROV Notation

Listing 3.3 is the PROV-N representation of *revision pattern*.

```
document
  prefix var <https://provenance.ecs.soton.ac.uk/var#>
  prefix rating <https://provenance.ecs.soton.ac.uk/ns/rating#>

  entity(var:dp,[prov:type = 'rating:DataProduct'])
  entity(var:dp.v1,[prov:type = 'rating:DataProductVersioned'])
  entity(var:dp.v2,[prov:type = 'rating:DataProductVersioned'])

  specializationOf(var:dp.v1,var:dp)
  specializationOf(var:dp.v2,var:dp)
  wasDerivedFrom(var:rev1;var:dp.v2, var:dp.v1,[prov:type = 'prov:Revision'])
endDocument
```

Listing 3.3: Provenance of data shaped by revision pattern in PROV-N representation

3.3.4 Agent Pattern

In this section, we introduce *agent pattern*.

Intent

Shape the provenance of data when an agent (a worker) evolves over time.

Motivation

Consider a worker in the crowdsourcing application who creates and scores data products. As time progresses, there are more information on the performance and reliability of this worker and it can be said that the worker evolves over time. The question is how to shape the provenance of data when a worker evolves over time.

We solve this problem by offering a template (a provenance pattern) and specifying some variables acting as placeholder for value to be specified. The pattern is called *agent pattern*, which shows that all agents (workers), evolve over time, and therefore, are versioned. Thus, both *create pattern* and *score pattern* result in a new specific version of a worker.

This provenance pattern requires the crowdsourcing application to fill the values for the following placeholders:

1. agent: which is the main version-less worker (creator or scorer). The provenance element associated to this item is of type either Creator or Scorer according to the role of the worker which specifies this agent is either a creator or a scorer.
2. agent.v1: which is a version of a worker. The provenance element associated to this item is of type either CreatorVersion or ScorerVersion according to the role of the worker which specifies this agent is either a creator or a scorer.
3. agent.v2: which is a version of a worker. The provenance element associated to this item is of type either CreatorVersion or ScorerVersion according to the role of the worker which specifies this agent is either a creator or a scorer.

Being able to distinguish the different versions of agents allows provenance-enabled crowdsourcing applications to define quality measures over these, which in turn can be used to schedule the work. The provenance of data also contains a worker which may be referred to irrespectively of its actual versions. A specific version is said to be a specialization of the general/non-versioned artifact.

The *agent pattern* is inspired from "PLAN FOR REVISIONS" (Moreau and Groth, 2013b).

Same as *revision pattern*, we define a worker as some agent plus some ratings (i.e. measures), which altogether, we consider as an agent, for the purpose of provenance. Each version of a worker contains a set of fixed attributes that describe more information about that worker at a certain period of time (such as the reliability of the worker).

It is noteworthy that the worker is not changing, but because ratings of a worker (i.e. measures) are changing, we look at them as different agents.

Provenance Graph

Figure 3.4 shows the provenance graph for *agent pattern*. It can be seen in this figure that each provenance agent is of type as described earlier. Note that the type of the general agent is set to X (which is a placeholder for the role of the worker, either creator or scorer) and the type of each version is set to Y (which is a placeholder according to the role of the worker, either CreatorVersioned or ScorerVersioned).

PROV Notation

Listing 3.4 is the PROV-N representation of *agent pattern*.

```
document
  prefix var <https://provenance.ecs.soton.ac.uk/var#>
  prefix rating <https://provenance.ecs.soton.ac.uk/ns/rating#>

  agent(var:agent,[prov:type = 'rating:X'])
  agent(var:agent.v1,[prov:type = 'rating:Y'])
  agent(var:agent.v2,[prov:type = 'rating:Y'])

  specializationOf(var:agent.v1,var:agent)
  specializationOf(var:agent.v2,var:agent)
  wasDerivedFrom(var:rev1;var:agent.v2, var:agent.v1,[prov:type = 'prov:Revision'])
endDocument
```

Listing 3.4: Provenance of data shaped by agent pattern in PROV-N representation

3.3.5 Summary of Types

In this dissertation, we rely on various namespaces identified by the following URIs and denoted by corresponding prefixes (Table 3.1).

Table 3.2 summarises all the types that were used in the provenance patterns in Section 3.3. All of these types are in *rating* prefix shown in Table 3.1.

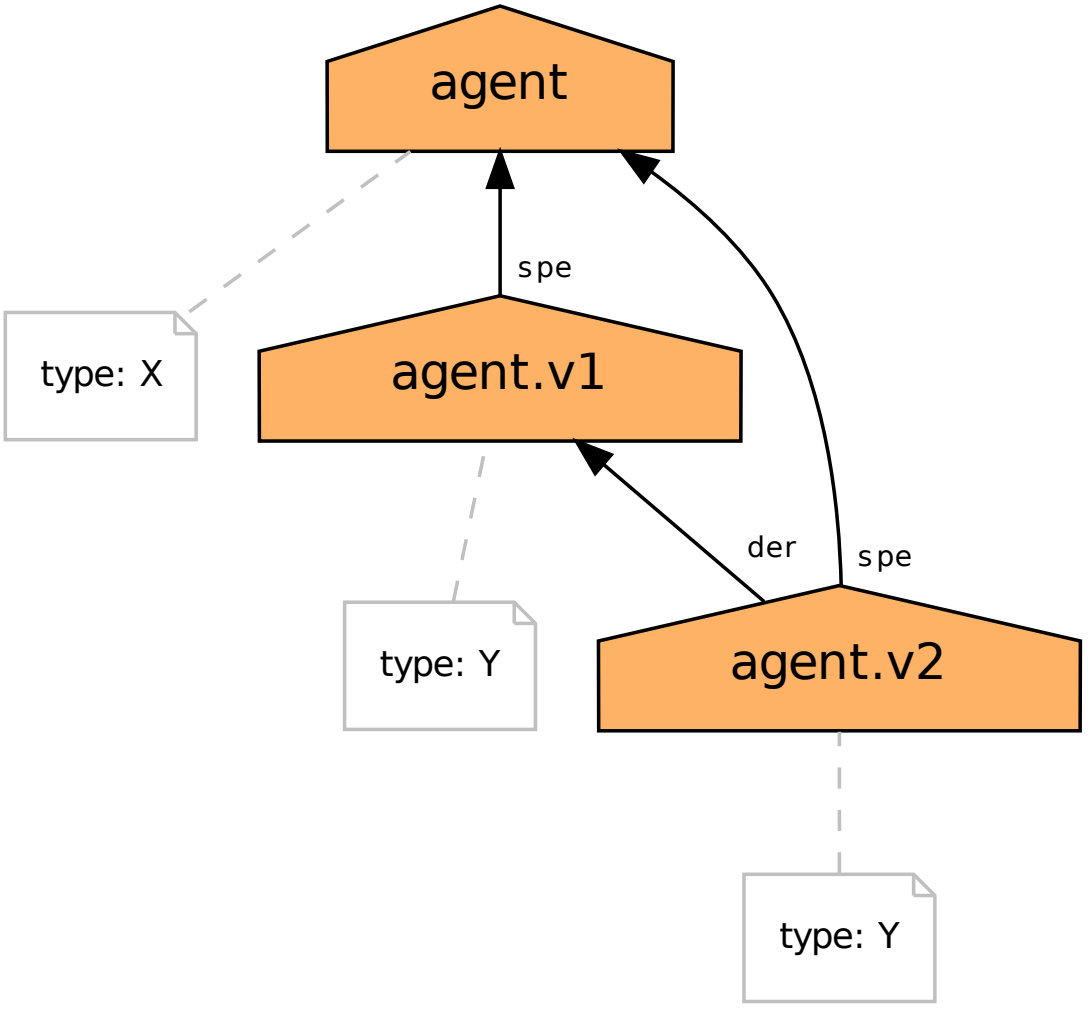


Figure 3.4: Provenance graph shaped by agent pattern.

Prefix	Namespace URI	Description
rating	https://provenance.ecs.soton.ac.uk/ns/rating#	The namespace for ratings
var	https://provenance.ecs.soton.ac.uk/var#	The namespace for examples
prov	http://www.w3.org/ns/prov#	The prov namespace

Table 3.1: Summary of all namespaces in provenance patterns.

3.4 Provenance Patterns in CollabMap

We use the provenance patterns introduced in Section 3.3 to shape the provenance of data generated by the CollabMap application (introduced in Section 2.1.7) by filling the placeholders introduced in each template.

Figure 3.5 shows a sample provenance graph where building10 was created by user7 and

Type	PROV Concept	Description
CreationPlan	Entity	A plan to create a data product
ScoringPlan	Entity	A plan to score a data product
CreationInput	Entity	An input to the creation of a data product
Score	Entity	A score (i.e. a rating or a vote) provided by a worker
DataProduct	Entity	An entity that is a data product
DataProductVersioned	Entity	An entity that is a version of a data product
CreationActivity	Activity	An activity that is concerned with the creation of a data product
ScoringActivity	Activity	An activity that is concerned with the scoring of a data product
Creator	Agent	An agent who created a data product
CreatorVersioned	Agent	An agent that is a version of a creator
Scorer	Agent	An agent who scored a data product
ScorerVersioned	Agent	An agent that is a version of a scorer
Scoring	WasDerivedFrom	A wasDerivedFrom relation that is concerned with scoring of a data product

Table 3.2: Summary of all types in provenance patterns.

verified by user3[1]. As observed in the figure, the provenance of data in this example is shaped according to the provenance patterns.

Note that, for clarity, we have not made the types explicit in this figure.

3.5 Discussion on the Provenance Patterns

In Section 2.1.7, we introduced CollabMap and presented a sample provenance graph of a task generated by CollabMap. In Section 3.4, we showed how provenance of data in CollabMap can be shaped according to the provenance patterns.

Figure 3.6 presents a provenance graph of creation and verification of a building[2]. This figure gives us the chance to compare a provenance graph generated by CollabMap with (Figure 3.5) and without (Figure 3.6) applying the provenance patterns.

Table 3.3 compares the number of PROV concepts, such as entity, activity, generation, and etc, in a provenance graph generated by CollabMap with and without the consideration of provenance patterns. Those rows that have different number of PROV concepts are highlighted with a light-red colour.

[1]Listing A.3 offers the PROV-N representation of this example.
[2]Listing A.4 offers the PROV-N representation of this example.

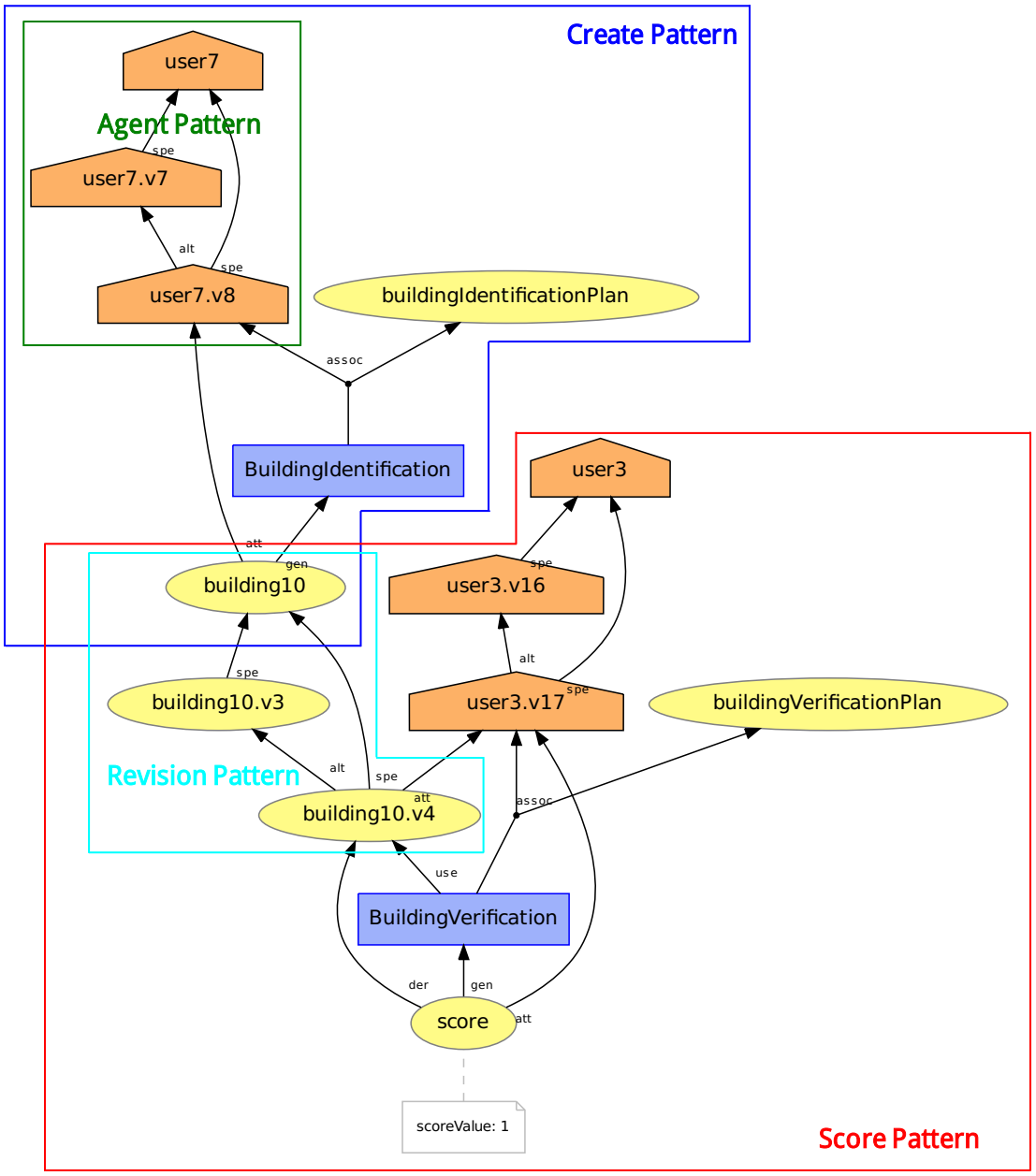


Figure 3.5: A sample provenance graph generated by CollabMap and shaped by the provenance patterns.

According to the Table 3.3, the total number of activities are the same, whereas both number of entities and agents are different. This is due to applying *revision pattern* and *agent pattern*. Furthermore, number of relations are the same except Attribution, Specialization, and Derivation (Revision). CollabMap does not create wasAttributedTo relation between an entity and an agent when the agent created that entity. But given that our quality assessment approach requires to know what data was created by whom, we added this relation when CollabMap creates a provenance graph. Furthermore, due to *revision pattern* and *agent pattern*, we have specializationOf and wasDerivedFrom (with the PROV type Revision) relations.

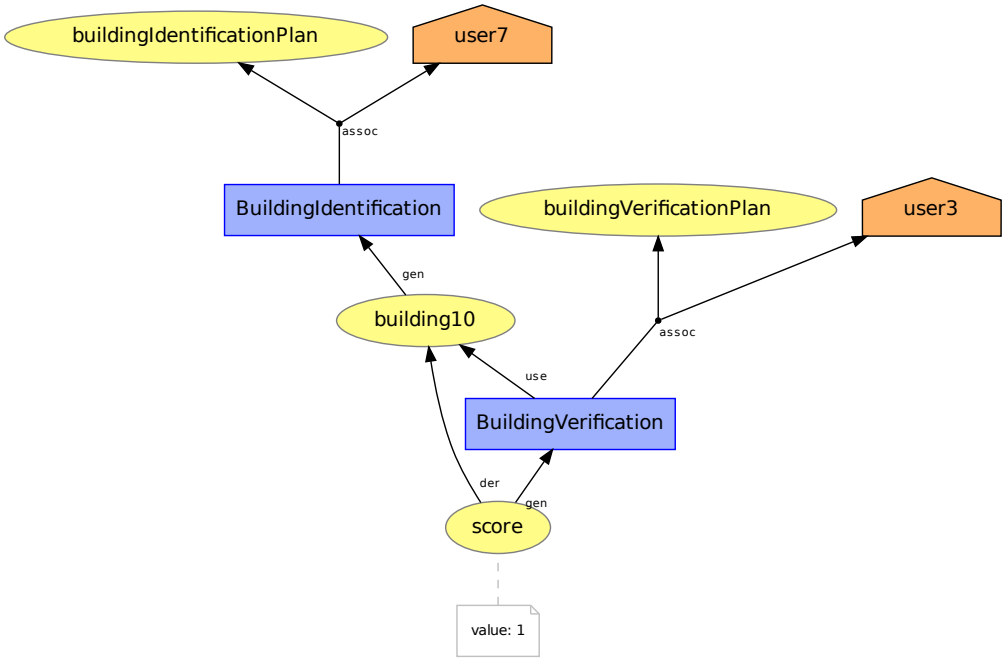


Figure 3.6: A sample provenance graph generated by CollabMap which is not shaped by the provenance patterns.

PROV concepts	With provenance patterns	Without provenance patterns
Entity	6	4
Activity	2	2
Agent	6	2
Derivation	1	1
Usage	1	1
Generation	2	2
Attribution	2	0
Association	2	2
Specialization	6	0
Derivation (Revision)	3	0

Table 3.3: The difference in the number of PROV concepts in a provenance graph shaped with and without provenance patterns.

The changes incurred on a FFV-based crowdsourcing application to be able to shape its provenance according to the provenance patterns are rather small (refer to Table 3.3). The crowdsourcing application is not required to change its internal data model to be compliant with the provenance patterns. Furthermore, all the changes that are required by the application is the way the provenance of data is generated and that should be shaped according to the provenance patterns, so that the quality of data can be assessed.

We should also note that the provenance patterns are applicable beyond FFV workflow. Most crowdsourcing applications are about creation and verification of data. As such, the provenance of data generated by such applications can be shaped by the provenance patterns.

3.6 Summary

This chapter presents a set of generic patterns that shape the provenance of data for different activities that are undertaken in a FFV-based crowdsourcing application. More specifically, *create pattern* shapes the provenance of data when a data product is created by a worker, *score pattern* shapes the provenance of data when a data product is scored by a worker, *revision pattern* and *agent pattern* shape the provenance of data when data products and workers are undergone revisions and evolve over time.

After that, we introduce CollabMap, an exemplar FFV-based crowdsourcing application. Then, we present a sample provenance graph of a task in CollabMap shaped according to the provenance patterns.

The provenance of data shaped by the provenance patterns are exploited by a quality assessment approach so that the quality of data and reliability of workers are assessed. However, before doing that, we require a generic method to be able to traverse a provenance graph which encompasses the provenance shaped by the provenance patterns. The provenance graph traversal allows us to access the information in the provenance graph and compute extra information, such as quality measures. In the next chapter, we focus more on this generic approach.

Chapter 4

Annotation Computation Framework

Provenance of data tells us how data is constructed and is commonly regarded as a medium through which quality of data can be assessed (Moreau, 2010). In order to assess the quality of data in a crowdsourcing application, we propose to record the provenance of data and exploit it to compute a set of quality measures; in turn, these measures are utilised by the crowdsourcing application to make quality-based decisions, such as accepting or discarding data.

In Chapter 3, we offered a set of generic patterns that shape the provenance of data generated by a crowdsourcing application. Before exploiting the shaped provenance in order to compute quality measures, we require an approach that allows us to traverse the provenance graph, extract information from the graph, and finally compute new information.

This is part of a bigger problem that arises when provenance of data is required to be exploited with a purpose different than the one that motivated its design. In order to solve this problem, in this chapter, we offer a framework that allows us to exploit provenance of data systematically and independently of the application that generated the provenance. This framework allows us to extract information and compute new information over the provenance graph. More specifically, and recalling from Chapter 1, the contribution of this chapter is as follows:

Contribution We put forth a framework, called "Annotation Computation Framework", that provides a generic mechanism to traverse a provenance graph, extract information from the provenance graph, and finally enable the computation of extra information over the provenance graph.

This chapter consists of nine parts:

- Section 4.1 gives the intuition behind Annotation Computation Framework, known as *ACF*, and why it is required. We then mention that *ACF* consists of two levels: Annotation Level and Computation Level.
- Section 4.2 presents requirements and non-requirements of *ACF*.
- Section 4.3 introduces the first level of *ACF*, the Annotation Level.
- Section 4.4 focuses on the second level, the Computational Level, by presenting a graph traversal algorithm for provenance graphs, explaining how information are propagated through the provenance graph, and how new information is computed.
- Section 4.5 introduces the interface to the framework and how instantiations of the framework should implement the interface.
- Section 4.6 describes how *ACF* is configured and can be employed.
- Section 4.7 discusses the termination of *ACF*.
- Section 4.8 presents a sample instantiation of *ACF*.
- Section 4.9 summarises this chapter.

4.1 Annotation Lifecycle

Provenance of data allows users to understand and verify how data was derived and who were responsible, and thus, gives an insight on the quality of data and whether data is trustworthy. Provenance has been also used in the literature to show data is reproducible or a system is auditable. Given that provenance of data is recorded by a system, it can be exploited by other systems for other purposes. For instance, provenance of data can be imported by a system to reason over the provenance so that the quality of data is assessed; provenance of data can be imported by a system to check if the original system that generated the provenance was compliant to a set of rules, and similar examples.

What these examples and others have in common is that the provenance of data generated by an application is to be used for a purpose that was not known when this application was designed. Furthermore, systems that import provenance of data may want to process the provenance with other information they have in their own system and generate new information.

Therefore, in order to allow a system to exploit provenance of data, extract information from provenance, use existing information, and compute new information, we present a generic framework (Annotation Computation Framework, or *ACF* for short) which has two levels. In the first level, annotation is utilised as a generic mechanism to enable users to attach any information to elements of a provenance graph. Annotations are meta-data

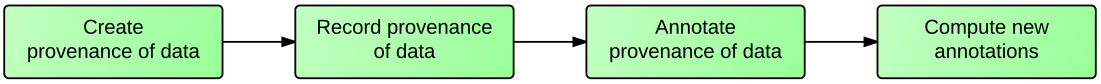


Figure 4.1: The Annotation Lifecycle.

that can provide extra information about the data. In the second level, computation level, annotations are propagated through the provenance graph and new annotations are computed from them.

We revised the Provenance Lifecycle (Moreau et al., 2008) to include the proposed two levels and call it the Annotation Lifecycle (Figure 4.1). Similar to the Provenance Lifecycle, a system creates and records provenance of data (in the *create provenance of data* and *record provenance of data* phases). This provenance of data is required to be exploited with a purpose different than the one at the creation time. The system which wishes to exploit the provenance, imports the data to its own system and annotates the provenance of data if necessary (*annotate provenance of data* phase). Finally, *ACF* propagates the annotations over the provenance graph so that new annotations are computed (*compute new annotations* phase).

In the next section, we discuss the requirements of *ACF*.

4.2 Annotation Computation Framework Requirements

The requirements of *ACF* are as follows:

Requirement 1 The framework should accept any kind of provenance graph encoded with PROV-DM.

> Given that PROV-DM (Moreau et al., 2013a) is a W3C Recommendation to record provenance on the Web, *ACF* should accept any kind of provenance graph encoded with PROV-DM. Furthermore, *ACF* is required to accept a provenance graph with a cycle.

Requirement 2 The framework should be able to propagate and compute annotation in the form of key-value pair.

> The attached information to a provenance record (i.e. annotation) should be in the form of key-value pair, so that, information can be propagated through the provenance graph by its type and its value to be updated.

Requirement 3 The framework should support two kinds of annotation propagation: forward propagation (from source to target) and backward propagation (from target to source).

Provenance describes what happened in the past. As such, it has a notion of time, for example from t_1 to t_2, given $t_1 < t_2$ or t_1 happened earlier than t_2. Two kind of propagations can be envisioned given the time: (1) forward propagation which is propagation from t_1 to t_2 in the direction of the time, and (2) backward propagation which is propagation from t_2 to t_1 in the opposite direction of the time. Hence, annotations are propagated over a provenance graph in both possible directions.

Requirement 4 If a node obtains more than one annotations as a result of annotation propagation and computation, the framework should provide a way to aggregate all the annotations.

More than one annotation could be computed after annotation propagation and computation for a provenance element. As a result of this, *ACF* should support a way to enable its instantiation to aggregate all annotations and compute a new annotation.

The non-requirement requirements of *ACF* are as follows:

Non-Requirement 1 The framework is not required to guarantee termination.

ACF is not required to guarantee termination because otherwise it should impose rigid structure and ordering on rules and annotation computation, which would not make the framework lightweight. As such, the responsibility of termination is left to the instantiation of *ACF*.

Non-Requirement 2 The framework should not change the structure of the provenance graph by inferring new nodes or relations.

ACF is about propagating annotations so that new annotations are computed. As such, the provenance graph should not be changed and stay intact. The framework should not add, change, nor remove a provenance record. Annotation propagation can be applied before or after inference.

Non-Requirement 3 The framework does not assume or support an order by which annotations are propagated and computed over a provenance graph.

The goal of the provenance graph traversal in *ACF* is to find all nodes which have annotations and propagate the annotations over the graph. In this sense, traversal is an operation on provenance graphs. The order is not a property of the graph. It is a property of the business logic (i.e. an instantiation of *ACF*). As such, not only the order has an impact on the efficiency with which we traverse a

provenance graph, but also implementing such feature would not make *ACF* as lightweight as we expected.

It is noteworthy to mention that the scalability of the framework depends on its instantiation. More specifically, if the rules implemented by the instantiation are computationally complex and dependent on each other, the framework might not be scalable.

4.3 Annotation Level

It is required to have a mechanism through which it is possible to attach extra information to any elements of a provenance graph. This extra information could be either existing data that the system which wants to exploit the provenance owns, or the newly computed information, or any other form of extra information.

In order to be able to include this extra information in a provenance graph, we propose to use PROV Attribute. We use attributes to encode annotations. An attribute is a key-value pair and is declared in a specific namespace. PROV attributes are meant to be fixed characteristics for entities (Moreau et al., 2013a). However, we use the mechanism of PROV Attribute to express annotations. As such, the framework accepts any kind of data that can be encoded via Attribute; given the key is a *qualified name* [1] and the value is *Value* [2].

For example, the provenance in Listing 4.1 is annotated with two attributes; as illustrated in Figure 4.2: both pp:goodInteraction and pp:badInteraction are defined in *pp* namespace, providing more information regarding the performance of user1.

```
1 document
2   prefix PP <http://users.ecs.soton.ac.uk/pp/>
3   prefix PROV <http://www.w3.org/ns/prov#>
4
5   agent(PP:user1,[
6     PP:goodInteraction = "10" %% xsd:int,
7     PP:badInteraction = "1" %% xsd:int
8   ])
9 endDocument
```

Listing 4.1: A sample provenance element annotated with two attributes in PROV-N representation

Furthermore, from now on, we assume that there is only one attribute with the same key; for example, in above example, it is illegal to have two attributes with the key goodInteraction.

[1] concept qualified name: http://www.w3.org/TR/prov-dm/#concept-qualifiedName
[2] concept value: http://www.w3.org/TR/prov-dm/#concept-value

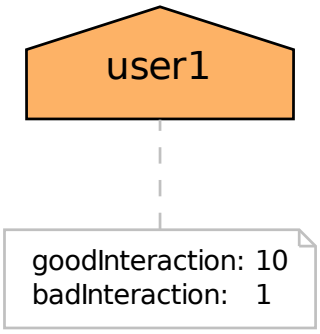


Figure 4.2: Provenance graph containing an agent annotated with two attributes.

4.4 Computational Level

The second part of *ACF* focuses on the creation of new annotations from existing annotations by introducing three fixed computational rules and a provenance graph traversal algorithm. This algorithm propagates the existing annotations from one provenance element to another over the provenance graph, so that new annotations are computed. In this sense, two kinds of propagation can be envisioned: forward and backward propagation. Forward propagation is the propagation of annotations by following nodes and relations between them along the direction of time. In terms of time, annotations of earlier nodes are propagated to more recent nodes. Backward propagation is the propagation of annotations in the opposite direction of forward propagation: against the direction of time. It involves propagating annotations by following nodes and relations between them from more recent nodes to earlier nodes.

The next section focuses on the computational rules.

4.4.1 Computational Rules

ACF supports three fixed computational rules for the creation of new annotations from the existing ones in a provenance graph.

Forward Computational Rule

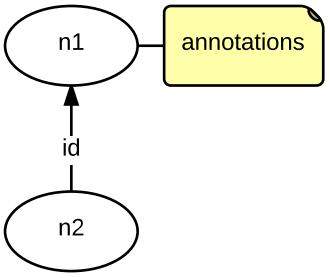
The first computational rule covers forward propagation and is known as "Forward Computational Rule" (Rule 4.1). In this rule, given a provenance graph (G) which contains a directed relation from a target ($n2$) to an source ($n1$), and source ($n1$) has a set of annotations ($annotations$), a new set of annotations for the target is computed based on $annotations$ and defined by $F_{forward}$. In order to allow the computation of new annotations based on the existing ones, a function, $F_{forward}$, is included in the forward

	Source	Target
influence relations	influencer	influencee
hadMember	collection	member
Specialization	general	specific
Alternate	alternate1	alternate2

Table 4.1: Mapping between source/target to PROV relations.

computational rule. Note that, both *annotations* and *newAnnotations* contain one or more annotations.

The *mainUpdate* function (to be introduced later, Algorithm 4.2) simply checks if an annotation is new. If it is, n_2 is updated with the new annotations. If not, n_2 is not updated.

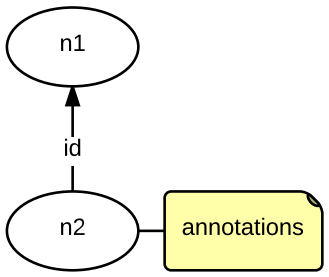


IF $\mathbf{G} \vdash$
$node(n1, annotations)$
$node(n2)$
$edge(id; n2, n1)$
THEN $\exists$ $newAnnotations$ such that
$newAnnotations = F_{forward}(G, id, n1, n2, annotations)$
$mainUpdate(G, n2, newAnnotations)$ (4.1)

Table 4.1 is a mapping from source/target to PROV relations.

Backward Computational Rule

The second computational rule covers backward propagation and is known as "Backward Computational Rule" (Rule 4.2). In this rule, given a provenance graph (G) which contains a directed relation from a target ($n2$) to an source ($n1$), and target ($n2$) has a set of annotations (*annotations*), a new set of annotations for the source is computed based on *annotations* and defined by $F_{backward}$. To allow the computation of new annotations based on the existing ones, a function, $F_{backward}$, is included in the backward computation rule.

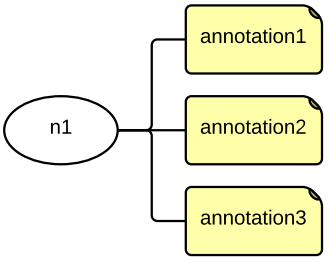


IF $\mathbf{G} \vdash$
$node(n1)$
$node(n2, annotations)$
$edge(id; n2, n1)$
THEN $\exists$ $newAnnotations$ such that
$newAnnotations = F_{backward}(G, id, n1, n2, annotations)$
$mainUpdate(G, n1, newAnnotations)$

(4.2)

Aggregation Computational Rule

The third computational rule is known as "Aggregation Computational Rule" and concentrates on a situation where a provenance element (entity, agent, activity) has more than one annotation and a new set of annotations is computed based on those annotations. In this case, there is only one provenance element and no relation. As such, there is no propagation involved, but just computation (Rule 4.3). In this rule, given a provenance graph (G) which contains a node ($n1$), and the node has a set of annotations ($annotation1$, $annotation2$, $annotation3$), a new set of annotations, ($newAnnotations$), is computed based on all its existing annotations and defined by $F_{aggregate}$. To allow the computation of new annotations based on the existing ones, a function, $F_{aggregate}$, is included in the aggregation computation rule.



IF $\mathbf{G} \vdash$
$node(n1, annotation1)$
$node(n1, annotation2)$
$node(n1, annotation3)$
$\ldots$
THEN $\exists$ $newAnnotations$ such that
$annotations \cup \{annotation1, annotation2, annotation3\}$
$newAnnotations = F_{aggregate}(G, n1, annotations)$
$mainUpdate(G, n1, newAnnotations)$

(4.3)

4.4.2 Provenance Graph Traversal Algorithm

In Section 4.4.1, we explained how annotations are propagated from one provenance element to another. This section presents the provenance graph traversal algorithm that first retrieves provenance elements with annotations, and then propagates annotations over the provenance graph, from one provenance element to another given the direction of propagation.

Algorithm 4.1 demonstrates that given a provenance graph, denoted as G, a set of all elements with a set of annotations, denoted as S, are retrieved. As stated previously, we use PROV Attribute to encode an annotation. As such, Algorithm 4.1 checks if an element has an attribute (Line 4). If it has, the element is added to the set (Line 5 issues the *add* command where e is *added* to S); note that if the set already contains the element, the element would not be added to the set.

Algorithm 4.1 findAllElementsWithAnnotation(G)

Input: G: a provenance graph
Output: set of provenance elements that have annotations

```
1: S ← emptyset
2: E ←  all the provenance elements in G
3: for e ∈ E do
4:     if e has an attribute then
5:         S.add(e)
6:     end if
7: end for
8: return S
```

Listing 4.2 shown an entity which has an annotation:

```
1 document
2   prefix PP <http://users.ecs.soton.ac.uk/pp/>
3   prefix PROV <http://www.w3.org/ns/prov#>
4
5   entity(PP:user1,[
6     PROV:value = "abcd" %% xsd:string
7   ])
8 endDocument
```

Listing 4.2: A PROV entity with an annotation

Listing 4.3 shown an agent which has two user-attached annotations:

```
1 document
2   prefix PP <http://users.ecs.soton.ac.uk/pp/>
3   prefix PROV <http://www.w3.org/ns/prov#>
4
5   agent(PP:user1,[
6     PP:goodInteraction = "10" %% xsd:int,
7     PP:badInteraction = "1" %% xsd:int
8   ])
9 endDocument
```

Listing 4.3: A PROV agent with two annotations

The difference between the annotations in Listing 4.2 and Listing 4.3 is the namespace of annotations. PROV:value is a standard attribute (Moreau et al., 2013a) that provides a value for an entity. However, in Listing 4.3, the annotations are in *PP* namespace and as such interpreted as user-attached annotations. Nevertheless, *ACF* propagates both kind of annotations.

Each element of provenance graph can be assigned with a set of annotations. We use PROV Attribute to encode an annotation. Attribute is a key-value pair. We also assume that a provenance element should not have two attributes with the same key. When an annotation is computed for an element, *ACF* adds the annotation to the set if and only if the annotation is new. An annotation is new if the type of annotation is not available in the set or it has a new value in which the older annotation is replaced with the newer annotation.

Let us consider that two annotations are computed for the provenance element in Listing 4.3: goodInteraction with value 11 and badInteraction with the value 1. The annotation set is updated to hold the new annotations as illustrated in Listing 4.4:

```
1 document
2   prefix PP <http://users.ecs.soton.ac.uk/pp/>
3   prefix PROV <http://www.w3.org/ns/prov#>
4
5   agent(PP:user1,[
6     PP:goodInteraction = "11" %% xsd:int,
7     PP:badInteraction = "1" %% xsd:int
8   ])
9 endDocument
```

Listing 4.4: An updated PROV agent with two newly computed annotations

Algorithm 4.2 is called by Rule 4.1, Rule 4.2, and Rule 4.3. For each newly computed annotation, this algorithm calls Algorithm 4.3. The *individualUpdate* in Line 4 (Algorithm 4.3) returns 1 if and only if the annotation is new. Given that there might be more than one annotation computed, Line 4 is a "logical or" to a temporary variable defined in Line 2. At the end, Line 7 updates S by adding the provenance element n to the set if $T = 1$ (i.e. a new annotation was computed).

Algorithm 4.2 mainUpdate($G, n, setAnn$).

Input: G: a provenance graph
Input: n: provenance element
Input: $setAnn$: a set of key-value annotations
Output: S: updated set

1: $S \leftarrow findAllElementsWithAnnotation(G)$
2: $T = 0$
3: **for** $ann \in setAnn$ **do**
4: $\quad T \leftarrow T \vee individualUpdate(G, n, ann)$
5: **end for**
6: **if** T=1 **then**
7: $\quad S.add(n)$
8: **end if**
9: **return** Q

Algorithm 4.3 presents the *individualUpdate* algorithm being called from Algorithm 4.2. Line 1 states that an annotation is represented as a key-value pair. Line 2 extracts all

the annotations of the provenance element and assign them to a set. Line 3 checks if the newly computed annotation is in the set and assigns the result (0 or 1) to a temporary variable. Line 4 to Line 9 update the set of annotations of the provenance element with the newly computed annotations. Simply put, if the key of the annotation is new, the set of annotation is updated with the newly computed annotations. If the key of the newly computed annotation exists in the set of annotation, the set of annotation is updated with the value of the newly computed annotation if and only if the value is different. At the end, Line 10 returns T specifying if the annotation is new.

Algorithm 4.3 individualUpdate(G, n, ann).

Input: G: a provenance graph
Input: e: a provenance element
Input: ann: key-value annotation pair
Output: T: 0/1 value whether the annotation is new

1: $\langle k, v\rangle \leftarrow ann$
2: $A \leftarrow$ the set of all annotations of e
3: $T \leftarrow (ann \notin A)$
4: **if** $\langle k, v'\rangle \in A$ for some v' **then**
5: $\quad A \leftarrow A \setminus \langle k, v'\rangle \cup \langle k, v\rangle$ ▷ has a side effect on G
6: **end if**
7: **if** $\langle k, v'\rangle \notin A$ for any v' **then**
8: $\quad A \leftarrow A \cup \langle k, v\rangle$ ▷ has a side effect on G
9: **end if**
10: **return** T

In summary, Algorithm 4.2 accepts a provenance graph, the node that annotations are computed for, and a set of computed annotations (which might be more than one annotations). In order to check if an annotation is new, this algorithm has a loop over all computed annotations and for each annotation, it calls Algorithm 4.3. Algorithm 4.3 checks if that annotation is new, given all the annotation the element has. Note that, Algorithm 4.3 accepts only one key-value annotation pair as opposed to Algorithm 4.2. If that annotations is new, then: (1) the element is attached with the new annotation and (2) Algorithm 4.3 returns back 1 to Algorithm 4.2 which states there was a new annotation. As such, that element is added to the set.

Algorithm 4.4 retrieves all the relations in the provenance graph based on the direction of propagation which could be forward, backward, or both and add them to a list. More specifically, for a given provenance element, denoted as e, and the propagation direction, denoted as D, all relations from or to that element are retrieved, denoted as R. If the propagation direction is set to forward, then all the relations from the provenance element are retrieved (Line 2). In this case, the provenance element is said to be the cause element in the provenance graph. If the propagation direction is set to backward, then all the relations to the provenance element are retrieved; the provenance element is the effect (Line 4). The propagation direction can be set to both, in this case all the relations from and to the provenance element are retrieved (Line 6).

Algorithm 4.4 findAllRelationsForElement(e, D).

Input: e: a provenance element
Input: D: propagation direction
Output: a set of relations

```
1: if D = forward then
2:     return all relations where e is cause
3: else if D = backward then
4:     return all relations where e is effect
5: else if D = both then
6:     return all relations where e is cause or effect
7: end if
```

Algorithm 4.5 is the main provenance graph traversal algorithm. Given the propagation direction, denoted as D, and the provenance graph, denoted as G, S is a set of all provenance elements with a set of annotations (Line 1, See Algorithm 4.1). Line 2 introduces a loop over all the provenance elements available in S. Then, Line 3 retrieves the first element in the set by issuing *first* command on S which results the first element to be retrieved and stored in e and removed from S.

For each provenance element in S, R holds a list of all relations from or to that element based on the direction (Line 4, See Algorithm 4.4). For each relation, lines 5 to 17 reflect one of the computational rules based on propagation direction (denoted as D), in which, n_1, in Line 6, refers to the source element of r (refer to Table 4.1 for source/target definition) and n_2, in Line 7, refers to the target element of r. Based on the propagation direction, one of the $f_{forward}$, or $f_{backward}$, or f_{both} function is called (according to Lines 9 to Line 16).

Algorithm 4.6 presents the forward annotation propagation algorithm that is utilised in the provenance graph traversal algorithm in Line 10 and Line 14.

Line 1 calls $F_{forward}$ implementation on the instantiation side, providing the provenance graph, the relation, source, target, and a set of annotations to be propagated. The output of $F_{forward}$ function is a set of computed annotations by the instantiation. Line 2 calls the *mainUpdate* function (refer to Algorithm 4.2).

Note that there is a directed relation, r, from n_2 to n_1. Given that this is the forward propagation, annotations of n_1 are propagated forward to n_2. As such, the *mainUpdate* function in Line 2 is called with n_2 as an input parameter (i.e. the newly computed annotations in Line 1 are for n_2 and n_2 should be updated with them).

Afterwards, $F_{aggregate}$ function is called to aggregate all the annotations of n_2 (Line 4). Having aggregated all the annotations, Line 5 calls the *mainUpdate* once again to update all the annotations of n_2. Note that, *mainUpdate* has n_2 as the input parameter.

Algorithm 4.7 presents the backward annotation propagation algorithm that is utilised in the provenance graph traversal algorithm in Line 12 and Line 15.

Algorithm 4.5 provenanceGraphTraversalAlgorithm(G, D).

Input: G: a provenance graph
Input: D: propagation direction
1: $S \leftarrow$ findAllElementsWithAnnotations(G)
2: **while** $S \neq \varnothing$ **do** ▷ terminates with G updated with propagated annotations
3: $e \leftarrow S.first()$
4: $R \leftarrow$ findAllRelationsForElement(e, D)
5: **for** $r \in R$ **do**
6: $n_1 \leftarrow r.source$
7: $n_2 \leftarrow r.target$
8: $setAnn \leftarrow$ set of all annotations of e
9: **if** $D =$ forward **then**
10: $forwardPropagation(G, r, n_1, n_2, setAnn)$ ▷ has a side effect on G
11: **else if** $D =$ backward **then**
12: $backwardPropagation(G, r, n_1, n_2, setAnn)$ ▷ has a side effect on G
13: **else if** $D =$ both **then**
14: $forwardPropagation(G, r, n_1, n_2, setAnn)$ ▷ has a side effect on G
15: $backwardPropagation(G, r, n_1, n_2, setAnn)$ ▷ has a side effect on G
16: **end if**
17: **end for**
18: **end while**

Algorithm 4.6 forwardPropagation($G, r, n_1, n_2, setAnn$).

Input: G: a provenance graph
Input: r: a relation
Input: n_1: a provenance element (source)
Input: n_2: a provenance element (effect)
Input: $setAnn$: a set of annotations to be propagated
1: $newAnn \leftarrow F_{forward}(G, r, n_1, n_2, setAnn)$
2: $S \leftarrow mainUpdate(G, n_2, newAnn)$
3: $allAnn \leftarrow$ set of all annotations of n_2
4: $aggAnn \leftarrow F_{aggregate}(G, n_2, allAnn)$
5: $S \leftarrow mainUpdate(G, n_2, aggAnn)$

Line 1 calls $F_{backward}$ implementation on the instantiation side, providing the provenance graph, the relation, source, target, and a set of annotations to be propagated. The output of $F_{backward}$ function is a set of computed annotations by the instantiation. Line 2 calls the *mainUpdate* function (refer to Algorithm 4.2).

Note that there is a directed relation, r, from n_2 to n_1. Given that this is the backward propagation, annotations of n_2 are propagated backward to n_1. As such, the *mainUpdate* function in Line 2 is called with n_1 as an input parameter (i.e. the newly computed annotations in Line 1 are for n_1 and n_1 should be updated with them).

Afterwards, $F_{aggregate}$ function is called to aggregate all the annotations of n_1 (Line 4). Having aggregated all the annotations, Line 5 calls the *mainUpdate* once again to update all the annotations of n_1. Note that, *mainUpdate* has n_1 as the input parameter.

Algorithm 4.7 backwardPropagation$(G, r, n_1, n_2, setAnn)$.

Input: G: a provenance graph
Input: r: a relation
Input: n_1: a provenance element (source)
Input: n_2: a provenance element (effect)
Input: $setAnn$: a set of annotations to be propagated

1: $newAnn \leftarrow F_{backward}(G, r, n_1, n_2, setAnn)$
2: $S \leftarrow mainUpdate(G, n_1, newAnn)$
3: $allAnn \leftarrow$ set of all annotations of n_1
4: $aggAnn \leftarrow F_{aggregate}(G, n_1, allAnn)$
5: $S \leftarrow mainUpdate(G, n_1, aggAnn)$

4.5 Framework Interface

In this section, we detail how an instantiation of ACF can implement the three computational rules: $F_{forward}$, $F_{backward}$, and $F_{aggregate}$.

We did not define a language to express these functions. Instead, we decided to use Java programming language. As such, an instantiation of the framework must implement a Java interface, as illustrated in Listing 4.5.

```
1  public interface IPropagation {
2    public void backward(
3      Document graph,
4      Relation0 relation,
5      Element cause,
6      Element effect,
7      Hashtable<QName, Object> inputAnnotation,
8      Hashtable<QName, Object> outputAnnotation);
9
10   public void forward(
11     Document graph,
12     Relation0 relation,
13     Element cause,
14     Element effect,
15     Hashtable<QName, Object> inputAnnotation,
16     Hashtable<QName, Object> outputAnnotation);
17
18   public void aggregate(
19     Document graph,
20     Element annotatedElement,
21     Hashtable<QName, Object> inputAnnotation,
22     Hashtable<QName, Object> outputAnnotation);
23 }
```

Listing 4.5: Java interface to ACF.

In this thesis, we utilise ProvToolbox[3], a Java library to create Java representations of PROV-DM. The Document class in Listing 4.5 represents a provenance graph, Element

[3]ProvToolbox: http://lucmoreau.github.io/ProvToolbox/

represents either PROV entity, activity, or agent, and Relation0 represents any of PROV relations such as wasDerivedFrom, used, or etc.

An instantiation of *ACF* is the implementation of the interface presented in Listing 4.5: backward method should be implemented for $F_{backward}$ function, forward method should be implemented for $F_{forward}$ function, and aggregate method should be implemented for $F_{aggregate}$ function. For each method, user needs to provide the following:

- graph: provenance graph
- relation: relation between two nodes (cause and effect)
- cause: the node which is the cause of the relation
- effect: the node which is the effect of the relation
- inputAnnotation: inputAnnotation which is the annotation of the node
- outputAnnotation: outputAnnotation which is used to hold the computed annotation

4.6 Framework Configuration

ACF is a generic framework and as such it needs to be instantiated and configured. In order to configure the framework, it is required to provide the following:

- Provenance graph: The provenance graph is required to be provided in any standard format as specified in the PROV specification such as XML, JSON, RDF, or etc.
- Propagation direction: The direction of propagation should be also provided. The framework supports all possible directions: forward, backward, and both directions.
- Instantiation: An instantiation of the framework is the user implementation of the interface presented in Listing 4.5 which instructs how annotations are computed. More specifically, the instantiation is required to implement at least one of $F_{forward}$, $F_{backward}$, or $F_{aggregate}$.

4.7 Termination

As mentioned in Section 4.2, *ACF* offers no support to manage non-termination and rule ordering (Non-Requirement 1 and Non- Requirement 3), mainly because instantiations of *ACF* can provide a better control and be free to optimise as they see fit.

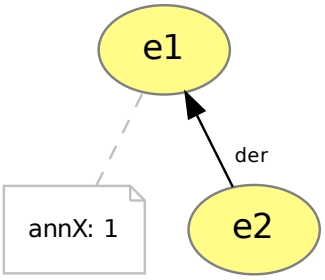


Figure 4.3: A sample provenance graph associated with the 1 of termination.

In this section, we look at three cases in regards to termination. More specifically, Case 1 and Case 2 are about non-termination (i.e. propagation does not terminate) and Case 3 looks at a sufficient condition under which termination occurs. The purpose is to provide illustrations to help the reader to understand termination.

Case 1

This section presents a case where the rules implemented by an instantiation introduce circularity.

Let us consider the provenance graph of Figure 4.3 where e1 is annotated with annX with the value 1.

Given this provenance graph, let us consider that an instantiation of *ACF* computed a new annotation, "annY" for e2 by propagating annX annotation of e1 and adding one to its value as specified in Equation 4.4.

After this step (i.e. computation of "annY" in terms of annX), if the instantiation updates the value of annX in terms of the value of "annY" (by propagating "annY" backward to e1) as per Equation 4.5, the propagation and computation never terminates because the computational rules implemented by the instantiation introduce circularity.

$$annY := annX + 1 \tag{4.4}$$

$$annX := annY + 1 \tag{4.5}$$

Appendix B.1 provides the full source code of this example.

Case 2

This section presents a case where circularity is introduced in the provenance graph. It is noteworthy to mention that this example is invalid (as per PROV- Constraints (Nies,

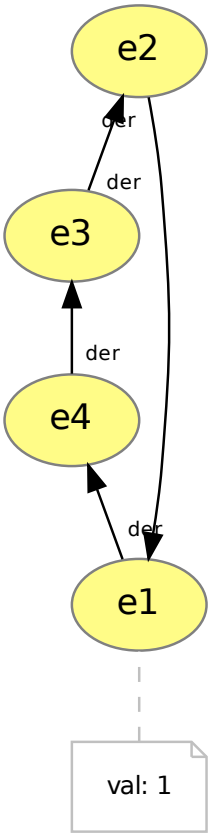


Figure 4.4: A sample provenance graph associated with Case 2 of termination.

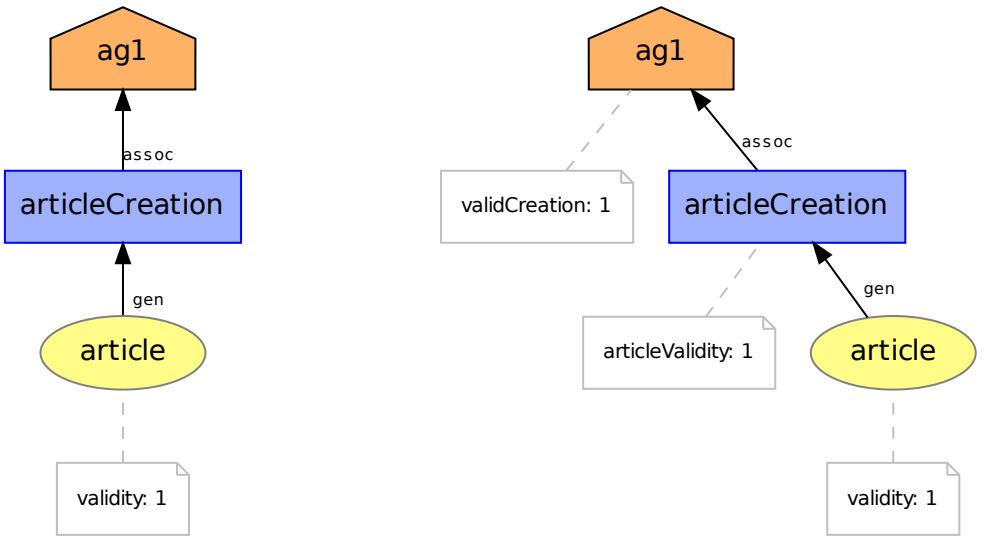
2013)), but it it also possible to create valid examples that are circular. We want the annotation propagation to be applicable to both valid and invalid provenance.

Let us consider the provenance graph of Figure 4.4 where e1 is annotated with val with the value 1.

Given this provenance graph, let us consider that an instantiation of ACF implements the computational rule as per Equation 4.6; stating if there is a wasDerivedFrom relation between two provenance element (e.g. between e2 and e1 in Figure 4.4), then compute a new annotation for e2 in terms of the annotation of e1.

$$\begin{aligned} &F_{forward}: \\ &\text{If } \text{wasDerivedFrom}(el_2, el_1) \\ &\quad el_2.val \leftarrow el_1.val + 1 \end{aligned} \tag{4.6}$$

Given Equation 4.6, the propagation and computation never terminates because circularity is introduced in the provenance graph (there is a wasDerivedFrom relation from e1 to e4).



(a) (b)

Figure 4.5: A sample provenance graph recording the creation of an article: (a) original provenance graph and (b) annotated provenance graph.

Case 3

This section looks at a sufficient condition under which termination occurs. This condition is sufficient but not necessary. There may be other cases of termination where this condition does not hold.

The condition requires that the annotation computation over a provenance graph form a strict order.

Let us consider the provenance graph of Figure 4.5(a) that records the provenance of data when an article was created. More specifically, article was generated by an activity, articleCreation, which was associated with an agent identified as ag1. The article element is annotated with validity with the value 1 which means the article is valid.

Given this provenance graph, let us consider that an instantiation of ACF wants to compute if an agent created a valid or invalid article. As such, first, validity annotation of article is propagated backward through the wasGeneratedBy relation to articleCreation and a new annotation, articleValidity, is computed as per Equation 4.7.

$$\begin{aligned} &F_{backward}: \\ &\text{If } \text{wasGeneratedBy}(e, ac) \\ &\quad ac.articleCreation \leftarrow e.validity \end{aligned} \tag{4.7}$$

Then, articleValidity annotation of articleCreation activity is propagated backward through the wasAssociatedWith relation to ag1 and a new annotation, validCreation, is computed as per Equation 4.8.

$$\begin{aligned} &F_{backward}: \\ &\text{If } \text{wasAssociatedWith}(ac, ag) \\ &\quad ag.validCreation \leftarrow ac.articleValidity \end{aligned} \tag{4.8}$$

In this sample example, the computational of new annotations form a strict order as follows: first Equation 4.7 is executed, then Equation 4.8.

Precisely and given Figure 4.5, the order is as follows:

$$(article, validity) < (articleCreation, articleValidity) < (ag1, validCreation)$$

In general, the instantiation should be able to order pairs, for example $(n1, a) < (n2, b) < (n3, c)$, where $n1$, $n2$, and $n3$ are provenance elements and a, b, and c are annotations. If the order is strict, we cannot end up with $(n1, a) < (n2, b) < (n3, c) < (n1, a)$ in which the pair $(n1, a)$ precedes it self.

Further, as a practical solution to avoid non terminating executions, awe have also introduced a counter that can be utilised to be the maximum number of annotations to be propagated. This counter is a function of total number of PROV elements (entities, activities, and agents) and PROV relations. This counter is decremented every time an annotation is propagated over the provenance graph.

4.8 Simulated Experiments

This section presents a sample use case to show how *ACF* is configured and utilised. The sample use case is about a news agency that records the provenance of an article it publishes. Although part of this dissertation is about crowdsourcing, the scenario introduced in this section is not in crowdsourcing domain. The reason being *ACF* is a standalone contribution to the literature, and given it is a generic framework, it can be instantiated in any domain. Later in Chapter 5, *ACF* is instantiated in crowdsourcing domain.

An article is generated from a set of media: textual content (e.g. tweets or other news) and non-textual content (e.g. photos and videos). It is required to compute the number of dependencies of each article and for that, we decided to use the provenance of each

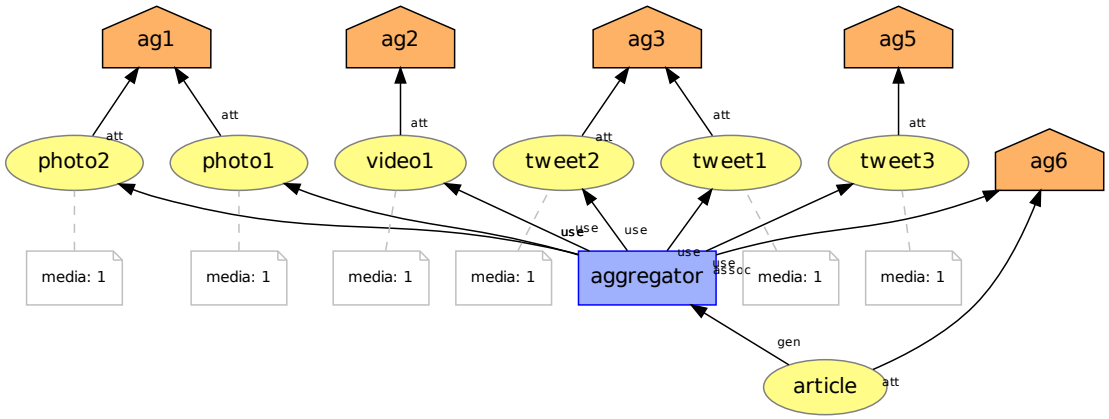


Figure 4.6: Original provenance graph generated by the news agency.

article. We define the number of dependencies of an article as the total number of media and textual sources an article depends on (or was derived from).

Figure 4.6 demonstrates the original provenance graph generated by the news agency. It shows that various resources such as photo1, tweet1, and others were collected by different agents. Then an activity (aggregator) aggregated all sources and generated the article.

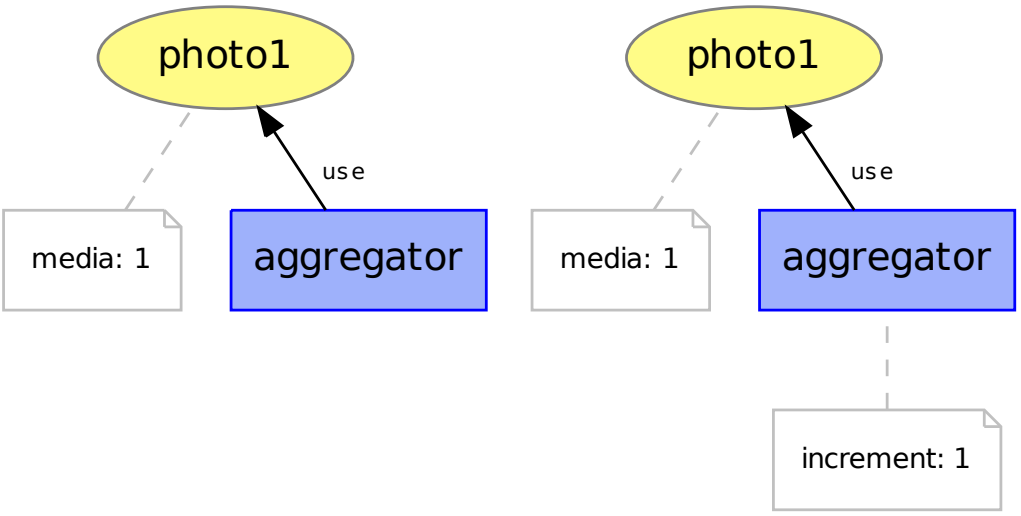
We mentioned that each article is generated from a set of media. As such, each provenance element which is a media (such as photo1, video1, and ...) is annotated with a media annotation.

In order to compute the number of dependencies for the article, we should count how many provenance elements are annotated with media annotation. As such, we describe the behaviour of $F_{forward}$, as a function that propagates the contents associated with key "media" of each provenance element with this annotation (e.g. photo1) to the key "increment" of aggregator activity. The annotation key "increment" states that the article was derived from a new media. Equation 4.9 states the forward computational rule where X in the equation refers to a provenance element which has media annotation.

$$\begin{aligned} &F_{forward}: \\ &\text{If } \mathrm{wasDerivedFrom}(X, aggregator) \\ &\quad aggregator.increment \leftarrow X.media \end{aligned} \tag{4.9}$$

Figure 4.7(a) demonstrates a sample portion of the original provenance graph where Equation 4.9 would be applied. Figure 4.7(b) demonstrates the same provenance graph after applying Equation 4.9. It can be seen that aggregator is annotated with a new annotation, increment.

After the forward computational rule and according to Algorithm 4.5, *ACF* calls the $F_{aggregate}$ function. As such, we describe the behaviour of $F_{aggregate}$, as a function that



(a) (b)

Figure 4.7: A portion of the original provenance graph describing how the news aggregator used a resource and how a new annotation is computed by implementing $F_{forward}$: (a): original provenance graph, and (b) annotated provenance graph

aggregates all the annotations of aggregator activity in the provenance graph so that a new annotation key, dependency is computed. Annotation dependency states the number of dependencies of an article (how many media the article depends). Equation 4.10 states the aggregate computational rule.

$$
\begin{aligned}
&F_{aggregate}: \\
&\quad d \leftarrow aggregator.dependency \\
&\quad i \leftarrow aggregator.increment \\
&\quad d \leftarrow d + i \\
&\quad aggregator.dependency \leftarrow d
\end{aligned}
\tag{4.10}
$$

Figure 4.8(a) demonstrates a sample portion of the original provenance graph after Equation 4.9 was applied. Figure 4.8(b) demonstrates the same provenance graph after applying Equation 4.10 (the $f_{aggregate}$ function). It can be seen that aggregator is annotated with a new annotation, dependency.

After all forward propagation and aggregate rules are applied to the provenance graph (associated with photo1, video1, and ...), the annotation key *dependency* of aggregator is required to propagate forward to the article provenance element. As such, we describe the behaviour of $F_{forward}$, as a function that propagates the contents associated with key "dependency" of aggregator to the key "dependency" of article provenance element. Equation 4.11 states the forward computational rule.

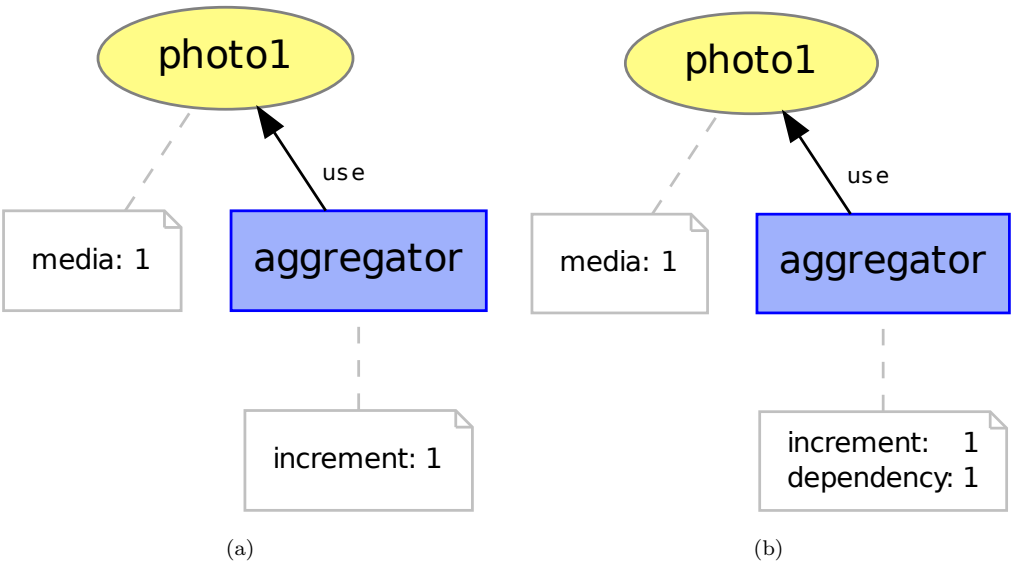


Figure 4.8: A portion of the original provenance graph describing how the news aggregator used a resource and how a new annotation is computed by implementing $F_{aggregate}$: (a): original provenance graph, and (b) annotated provenance graph

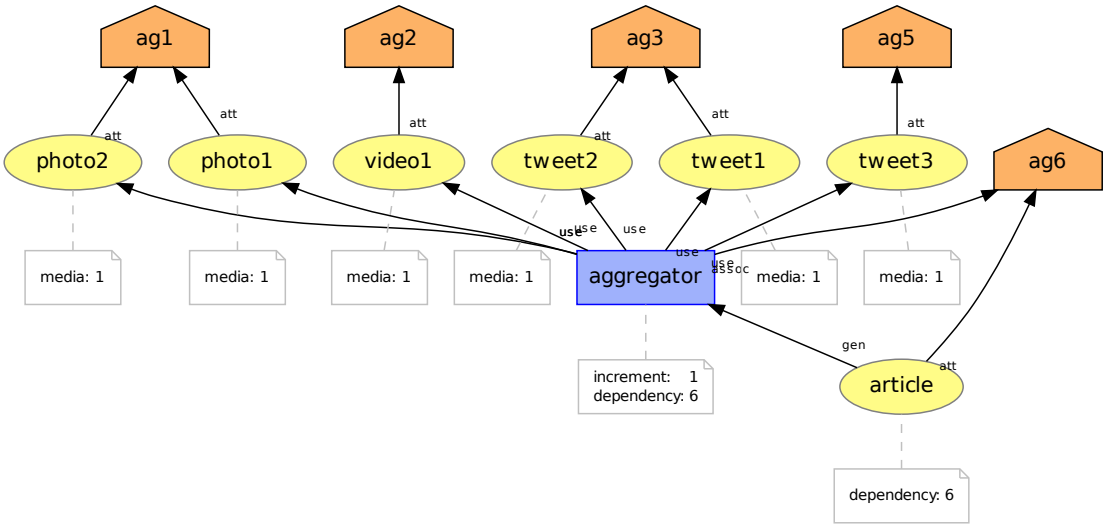


Figure 4.9: Annotated provenance graph returned by the Dependency-Graph instantiation.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasGeneratedBy}(aggregator, article)\\
&\qquad article.dependency \leftarrow aggregator.dependency
\end{aligned}
\tag{4.11}
$$

Figure 4.9 illustrates the resulting annotated provenance graph.

Appendix B.2 provides a Java source code for Dependency-Graph instantiation.

4.9 Summary

Provenance of data generated by a system can be imported by another system and reasoned over for a variety of purposes such as quality assessment or audit. In this case, the provenance of data is required to be exploited with a purpose different than the one at the generation time.

In order to facilitate exploiting provenance with a different purpose, in this chapter, we put forth a framework (known as *ACF*) that allows a system to exploit provenance of data systematically and independently of the application that generated the provenance. The framework allows the system to extract information from the provenance and compute new information based on existing ones.

At the outset, in Section 4.2, we specified 4 requirements for *ACF*. Based on the requirements: (1) *ACF* accepts any kind of provenance graph as long as it is encoded with PROV-DM (satisfying Requirement 1), (2) *ACF* propagates annotations over the provenance graph where each annotation is encoded as a PROV attribute; an attribute is a key-value pair (satisfying Requirement 2), (3) *ACF* supports both forward and backward annotation propagation by offering forward computational rule and backward computational rule (satisfying Requirement 3), and finally, (4) *ACF* supports aggregation of multiple annotations by offering aggregation computational rule.

The provenance graph traversal implemented in the framework propagates existing information over the provenance graph, so that new information can be computed. We assume that the annotation computation is ordered and it is not circular, hence, the propagation and computation of annotations terminate at some point.

This framework facilitates our goal of developing a data quality assessment approach for crowdsourcing applications. In the next chapter, we put forth a quality assessment approach that implements $F_{forward}$, $F_{backward}$, and $F_{aggregate}$ in the computational rules to assess the quality of data generated in a crowdsourcing application. More specifically, provenance of data that was generated by a crowdsourcing application is exploited by the data quality assessment with a different purpose (quality assessment).

Chapter 5

PEDRA: Provenance Enriched Data Rating Assessment

An increasing number of crowdsourcing applications follow the Find-Fix-Verify (FFV) workflow (Bernstein et al., 2010) which divides complex tasks into simpler micro-tasks and thus makes quality assessment more manageable. However, due to the open nature of crowdsourcing, varying background (e.g. country, language) and expertise (e.g. drawing or mathematical skills), it is required for the task requesters to assess the quality of data obtained from workers. There are three kind of issues prevalent in a crowdsourcing application:

- Issue 1: to assess the quality of the data products generated by unknown workers
- Issue 2: to assess the reliability of workers
- Issue 3: to determine when to stop recruiting more workers to score a data product

In order to solve these issues, we propose to record workers' verifications and their past performance which is an indicator of their reliability and how they may perform in the future. In order to record this information, we record the provenance of data. Therefore, for a crowdsourcing application to assess the quality of its data, first, the application recruits workers to create and verify data. The application also records and maintains the provenance of data which is shaped by the provenance patterns (See Chapter 3). The shaped provenance is exploited by a data quality assessment approach in order to assess the quality of data. In order to do so, in this chapter, we put forth a quality assessment approach, known as "Provenance Enriched Data Rating Assessment" or *PEDRA* for short, that computes a set of quality measures for data and workers. These quality measures are utilised by the crowdsourcing application to make quality-based decisions, such as accepting or discarding a data product.

In summary, the following two research statements are made in this chapter and then a set of evidences as part of the evaluation are presented to support them:

- Statement 1: The provenance of a data product and a worker, which is shaped by the provenance patterns, can be exploited by *PEDRA* so that the quality of data and reliability of the worker is assessed.
- Statement 2: As more provenance is exploited by *PEDRA*, the uncertainty over the quality of a data product and reliability of a worker is reduced.

More specifically, and recalling from Chapter 1, the contribution of this chapter is as follows:

Contribution We present a quality assessment approach called *PEDRA*, an instantiation of *ACF*, that exploits the provenance of data shaped by the provenance patterns to compute a set of quality measures that assist the crowdsourcing application to choose data of high quality and discard those with low quality. Furthermore, using *ACF* and *PEDRA*, we are the first to show how a provenance enriched quality assessment approach can be employed as part of a crowdsourcing application, to asses the quality of generated data.

The rest of this chapter is structured as follows:

- Section 5.1.1 explains how provenance of data is exploited by *PEDRA* so that quality of generated data by unknown workers is assessed (Issue 1, Statement 1).
- Section 5.1.2 shows how provenance of data regarding workers' verifications and their past interactions with the crowdsourcing application is exploited by *PEDRA* so that reliability of workers is assessed (Issue 2, Statement 1).
- Section 5.1.3 expands the discussion on how provenance of data is exploited by *PEDRA* to determine how many workers are required to score a data product and when a task is deemed to be terminated (Issue 3).
- Section 5.2 discusses the mapping between *PEDRA* and *ACF* and how *PEDRA* implements the interface to *ACF*.
- Section 5.3 discusses how annotations (quality measures) propagation and computation terminate at some point in *PEDRA* instantiation.
- Section 5.4 details how *PEDRA* can be applied in practice and gives an empirical evaluation of the model.
- Section 5.5 summarises this chapter.

5.1 Quality Measures

PEDRA is an instantiation of *ACF* that implements all three computational rules ($F_{forward}$, $F_{backward}$, $F_{aggregate}$,) to traverse the provenance graph submitted by a crowd-sourcing application, extract information, and compute new information (i.e. quality measures). This section presents how these quality measures are computed.

5.1.1 Validity Estimate

As mentioned above, the quality of each data product needs to be assessed; valid data products are to be kept, and invalid ones ignored or discarded.

We define VALIDITY ESTIMATE as per Definition 5.1.

Definition 5.1. VALIDITY ESTIMATE is an estimate of being acceptable.

The VALIDITY ESTIMATE represents how valid a specific version v of a data product d is (denoted as d^v, with $v \in \mathbb{N}$), as a function of its current score (denoted as $score(d^v)$, representing a vote or rating) and those of its previous versions (i.e. $score(d^k)$ for $k = 0 \dots v-1$).

Let us assume that a score is a binary value, either positive ($+1$) or negative (-1) (e.g. Soylent, CollabMap, BudgetFix). Equation 5.1 and 5.2 below define the total number of POSITIVE SCORES and NEGATIVE SCORES, denoted as S^+ and S^-, for a version of a data product.

$$S^+(d^v) = \left|\left\{d^k : score(d^k) = 1, k \in \mathbb{N}, k \leq v\right\}\right| \tag{5.1}$$

$$S^-(d^v) = \left|\left\{d^k : score(d^k) = -1, k \in \mathbb{N}, k \leq v\right\}\right| \tag{5.2}$$

where d^k with $k \in \mathbb{N}, k \leq v$ is a version of the data product d up until its version v. Those versions can be identified from the provenance graph of d^v by following the *revision pattern* (Refer to Section 3.3.3).

We utilise a heuristic technique inspired by TRAVOS (Teacy et al., 2006), a trust model built on the beta family of probability distribution functions, to compute VALIDITY ESTIMATE of a version of a data product, denoted as $Q_V(d^v)$, in terms of $S^+(d^v)$ and $S^-(d^v)$ as follows:

$$Q_V(d^v) = \frac{S^+(d^v) + 1}{S^+(d^v) + S^-(d^v) + 2} \tag{5.3}$$

Measure	Description	Symbol
SCORE	Score provided for d^v	$score(d^v)$
POSITIVE SCORES	Total number of positive scores for d^v	$S^+(d^v)$
NEGATIVE SCORES	Total number of negative scores for d^v	$S^-(d^v)$
VALIDITY ESTIMATE	Validity estimate for d^v	$Q_V(d^v)$
VALIDITY LABEL	Validity label for d^v	$L_{Q_V}(d^v)$

Table 5.1: Quality measures associated with the validity estimate for a version of a data product denoted as d^v.

Equation 5.3 is applicable for cases where users' responses is modelled as a binary event. This is indeed the case in many FFV-based crowdsourcing applications such as Soylent, CollabMap, BudgetFix, and more.

Given that $Q_V(d^v) \in [0, 1]$, the crowdsourcing application is required to set two thresholds, TV_{high} and TV_{low}; if $Q_V(d^v)$ is equal to or greater than TV_{high}, the data product is accepted as a valid data product and if $Q_V(d^v)$ is equal to or smaller than TV_{low}, the data product is accepted as an invalid data product as per Equation 5.4.

$$L_{Q_V}(d^v) = \begin{cases} 1 & \text{if } Q_V(d^v) \geq TV_{high} \\ -1 & \text{if } Q_V(d^v) \leq TV_{low} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

Table 5.1 summarises the quality measures introduced in this section.

5.1.2 Worker Reliability

There are two types of workers in FFV-based crowdsourcing applications: (1) creators who create data products (as represented by the *create pattern*: Section 3.3.1) and (2) scorers who score data products (as represented by the *score pattern*: Section 3.3.2).

The reliability measure for creators, CREATOR RELIABILITY (Q_C), rates how reliable a creator is in creating data products and is defined as per Definition 5.2.

Definition 5.2. CREATOR RELIABILITY is an estimate of the regularity of a creator in creating valid data products.

Similarly, the reliability measure for scorers, SCORER RELIABILITY (Q_S), rates how reliable a scorer is in verifying data products and is defined as per Definition 5.3.

Definition 5.3. SCORER RELIABILITY is an estimate of the regularity of a scorer to agree with the consensus[1] of the other workers.

[1]Cambridge English Dictionary defines consensus as a "generally accepted opinion or decision among a group of people"

Measure	Description	Symbol
VALID CREATION	Total number of valid data product creation for u^v	$S_C^+(u^v)$
INVALID CREATION	Total number of invalid data product creation for u^v	$S_C^-(u^v)$
CREATOR RELIABILITY	Creator reliability of u^v	$Q_C(u^v)$

Table 5.2: Quality measures associated with the creator reliability for a version of a creator denoted as u^v.

The first step to compute CREATOR RELIABILITY, Q_C, is to determine the numbers of VALID CREATION (i.e. total number of valid data product creation) and INVALID CREATION (i.e. total number of invalid data product creation) by the worker concerned, denoted as S_C^+ and S_C^-, respectively (Equations 5.6 and 5.7).

$$\mathbb{D}(u^v) = \left\{d^k : \text{wasAttributedTo}(d^k, u^i), i \in \mathbb{N}, i \leq v\right\} \tag{5.5}$$

$$S_C^+(u^v) = \left|\left\{d^k : L_{Q_V}(d^k) = 1, d^k \in \mathbb{D}(u^v)\right\}\right| \tag{5.6}$$

$$S_C^-(u^v) = \left|\left\{d^k : L_{Q_V}(d^k) = -1, d^k \in \mathbb{D}(u^v)\right\}\right| \tag{5.7}$$

$\mathbb{D}(u^v)$ (in Equation 5.5) is the set of all data products created by all versions of worker u up until version v (following the *create pattern* in Figure 3.1).

Having computed the VALID CREATION (S_C^+) and INVALID CREATION (S_C^-) by u^v from the set $\mathbb{D}(u^v)$ as per the above equations, it is now possible to calculate CREATOR RELIABILITY ($Q_C(u^v)$) in the similar fashion as in Equation 5.3:

$$Q_C(u^v) = \frac{S_C^+(u^v) + 1}{S_C^+(u^v) + S_C^-(u^v) + 2} \tag{5.8}$$

Table 5.2 summarises the quality measures introduced in this section.

Similarly, the same worker can be assessed with respect to the quality of their verification tasks, known as SCORER RELIABILITY and denoted as $Q_S(u^v)$. In this case, however, we need to check if a score given by the worker to a data product is "aligned" with its VALIDITY LABEL. This can only be determined and taken into account if the validity of d is already determined; i.e. $L_{Q_V}(d^m) \neq 0$, where d^m is the latest version of d ($m \geq k$). If so, a $score(d^k)$ is said to be aligned if $score(d^k) = L_{Q_V}(d^m)$.

Let $\mathbb{V}(u^v) = \left\{score(d^k) : \text{wasAttributedTo}(score(d^k), u^i), i \in \mathbb{N}, i \leq v\right\}$ be the set of all verification scores contributed by worker u up to its version v; Let $L_{Q_V}(d)$ be the VALIDITY LABEL of the latest version of d for any d^k ($\text{specializationOf}(d^k, d)$). The total

Measure	Description	Symbol
ALIGNED SCORE	Total number of aligned scores for u^v	$S_S^+(u^v)$
NOT-ALIGNED SCORE	Total number of not aligned scores for u^v	$S_S^-(u^v)$
SCORER RELIABILITY	Scorer reliability measure of u^v	$Q_S(u^v)$

Table 5.3: Quality measures associated with the scorer reliability for a version of a scorer denoted as u^v.

numbers of aligned and non-aligned scores given by a worker u^v, ALIGNED SCORE denoted as $S_S^+(u^v)$ and NOT-ALIGNED SCORE denoted as $S_S^-(u^v)$, are defined, as follows:

$$\mathbb{V}(u^v) = \Big\{score(d^k) : \text{wasAttributedTo}(score(d^k), u^i), i \in \mathbb{N}, i \leq v\Big\} \tag{5.9}$$

$$S_S^+(u^v) = \Big|\Big\{score(d^k) \in \mathbb{V}(u^v) : L_{Q_V}(d) \neq 0 \wedge score(d^k) = L_{Q_V}(d)\Big\}\Big| \tag{5.10}$$

$$S_S^-(u^v) = \Big|\Big\{score(d^k) \in \mathbb{V}(u^v) : L_{Q_V}(d) \neq 0 \wedge score(d^k) \neq L_{Q_V}(d)\Big\}\Big| \tag{5.11}$$

The SCORER RELIABILITY for a version v of worker u, denoted as $Q_S(u^v)$, is defined based on $S_S^+(u^v)$ and $S_S^-(u^v)$ as follows:

$$Q_S(u^v) = \frac{S_S^+(u^v) + 1}{S_S^+(u^v) + S_S^-(u^v) + 2} \tag{5.12}$$

Table 5.3 summarises the quality measures introduced in this section.

5.1.3 Validity Rating

As introduced in Section 1.2, a task in FFV-based crowdsourcing applications typically requires contributions from multiple workers, mainly to minimise individual biases and to allow for cross verifications. For each data product, such an application routinely needs to decide whether to *accept* or to *discard* the data product; or in case it is uncertain, to *continue* allocate the data product to more workers for additional verification.

In order to assist the application with deciding task termination, *PEDRA* employs a heuristic technique to estimate how many scorers a data product requires. *PEDRA* computes the cumulative weighted score for a version of a data product, denoted as $\text{CWS}(d^v)$, that takes into account the scores of the data product *and* the reliability of the scorers. This measure is called VALIDITY RATING.

$$\text{CWS}(d^v) = \sum_{k,m=0}^{v} score(d^k) \times Q_S(u^m) \tag{5.13}$$

$$\text{wasAttributedTo}(score(d^k), u^m) \tag{5.14}$$

where u^m is the worker who submitted $score(d^k)$, or its author.

The VALIDITY RATING of a version of a data product, $\text{CWS}(d^v)$, assists the crowdsourcing application to make a decision on when it is deemed to be sufficiently certain to accept or reject d^v. For this purpose, the crowdsourcing application is required to set two thresholds, TT_{high} and TT_{low}; if $\text{CWS}(d^v)$ is equal to or greater than TT_{high} or it is equal to or smaller than TT_{low}, no more verification vote is required and a TERMINATION measure, denoted as T, for a version of a data product can be computed as per Equation 5.15.

$$T(d^v) = \begin{cases} \text{Accept} & \text{if } \text{CWS}(d^v) \geq TT_{high} \\ \text{Discard} & \text{if } \text{CWS}(d^v) \leq -TT_{high} \\ \text{Continue} & \text{otherwise} \end{cases} \tag{5.15}$$

For instance, if two scorers with high SCORER RELIABILITY (Q_S) both give positive scores for d^v, the TERMINATION above will accept the data product and the task is terminated (because the positive scores are given by scorers that are deemed reliable). However, if d^v only has scorers with low Q_S, we might require more than two scorers before d^v is either accepted or rejected.

Justification of the heuristic function Equation 5.17 defines how SR_{sum} is computed which is the sum over the probability of a score to be aligned with the consensus and the probability of the score not to be aligned with the consensus.

$$SR_{sum} = \sum_{k,m=0}^{v} \Big[score(d^k) \times Q_S(u^m) \Big] + \Big[(-score(d^k)) \times (1 - Q_S(u^m)) \Big] \tag{5.16}$$

$$SR_{sum} = \sum_{k,m=0}^{v} score(d^k) \times \Big[[2 \times Q_S(u^m)] - 1 \Big] \tag{5.17}$$

Equation 5.18 states how MV_{sum} is computed which simply is the majority voting over the value of all verification votes.

Measure	Description	Symbol
VALIDITY RATING	The cumulative weighted score for d^v	CWS(d^v)
TERMINATION	Termination measure for d^v	$T(d^v)$

Table 5.4: Quality measures associated with the validity rating for a version of a data product denoted as d^v.

$$MV_{sum} = \sum_{k=0}^{v} score(d^k) \tag{5.18}$$

At the end, $CWS(d)$ of a data product is computed as per Equation 5.19 which is the average over SR_{sum} and MV_{sum}.

$$CWS(d) = \frac{SR_{sum} + MV_{sum}}{2} \tag{5.19}$$

Equation 5.13 is equivalent to Equation 5.19.

Table 5.4 summarises the quality measures introduced in this section.

So far, in this chapter, we have introduced two measures for a version of a data product (d^v) that may seem similar but have a very very different purpose: (1) VALIDITY LABEL and (2) TERMINATION. These two are different in essence. Although VALIDITY LABEL states if a data product is valid or not, it does not consider the reliability of workers. There might be enough SCORE on a data product to compute VALIDITY LABEL, but not be able to compute TERMINATION because either the reliability of scorers are not yet known (the scorers have not been participating in the system enough) or scorers are unreliable and as such we require more scorers. Therefore, in order to terminate a task, we look at the TERMINATION measure.

5.2 Mapping Between PEDRA and ACF

This section provides the mapping between *PEDRA* and *ACF*. It shows how the computational rules provided by *ACF* ($F_{forward}$, $F_{backward}$, and $F_{aggregate}$) are described by the functions in *PEDRA*, so that the quality measures introduced in Section 5.1 are computed.

Let us first start with the annotations computed in Section 5.1.1 for VALIDITY ESTIMATE.

5.2.1 Mapping to Compute Validity Estimate

The following sections demonstrate how *PEDRA* instantiates *ACF* by implementing the three computational rules ($F_{forward}$, $F_{backward}$, $F_{aggregate}$) in order to compute quality measures listed in Table 5.1.

Rules 1 - 3

With Rule 5.20, we describe the behaviour of $F_{backward}$, as a function that propagates the contents associated with key "scoreValue" of *score* to the key "verificationValue" of d^v. Note that $score.scoreValue$ means *score* entity has a key-value pair (annotation) with the key *scoreValue*, its value could be either $+1$ or -1. In this case, one annotation is computed for d^v: "verificationValue" that specifies VERIFICATION VOTE of d^v.

$$\begin{aligned} &F_{backward}\text{:} \\ &\text{If wasDerivedFrom}(score, d^v) \\ &\quad d^v.verificationValue \leftarrow score.scoreValue \end{aligned} \tag{5.20}$$

Rule 5.21 describes a function that propagates forward the contents associated with key "positiveScores" and "negativeScores" of d^v to the key "oldPositiveScores" and "oldNegativeScores" of d^{v+1}, respectively. In this case, two annotations are computed for d^{v+1}: (1) "oldPositiveScores" that specifies total number of OLD POSITIVE SCORES of d^{v+1} and (2) "oldNegativeScores" that specifies total number of OLD NEGATIVE SCORES of d^{v+1}.

$$\begin{aligned} &F_{forward}\text{:} \\ &\text{If wasDerivedFrom}(d^{v+1}, d^v, \text{[prov:type=prov:Revision]}) \\ &\quad d^{v+1}.oldPositiveScores \leftarrow d^v.positiveScores \\ &\quad d^{v+1}.oldNegativeScores \leftarrow d^v.negativeScores \end{aligned} \tag{5.21}$$

Rule 5.22 describes a function that aggregates all the annotations of d^v to compute new annotation for d^v. In this case, four annotations are computed: (1) "positiveScores" that specifies the total number of POSITIVE SCORES of d^v, (2) "negativeScores" that specifies the total number of NEGATIVE SCORES of d^v, (3) "validityEstimate" that specifies the VALIDITY ESTIMATE of d^v, and (4) "validityLabel" that specifies the VALIDITY LABEL of d^v.

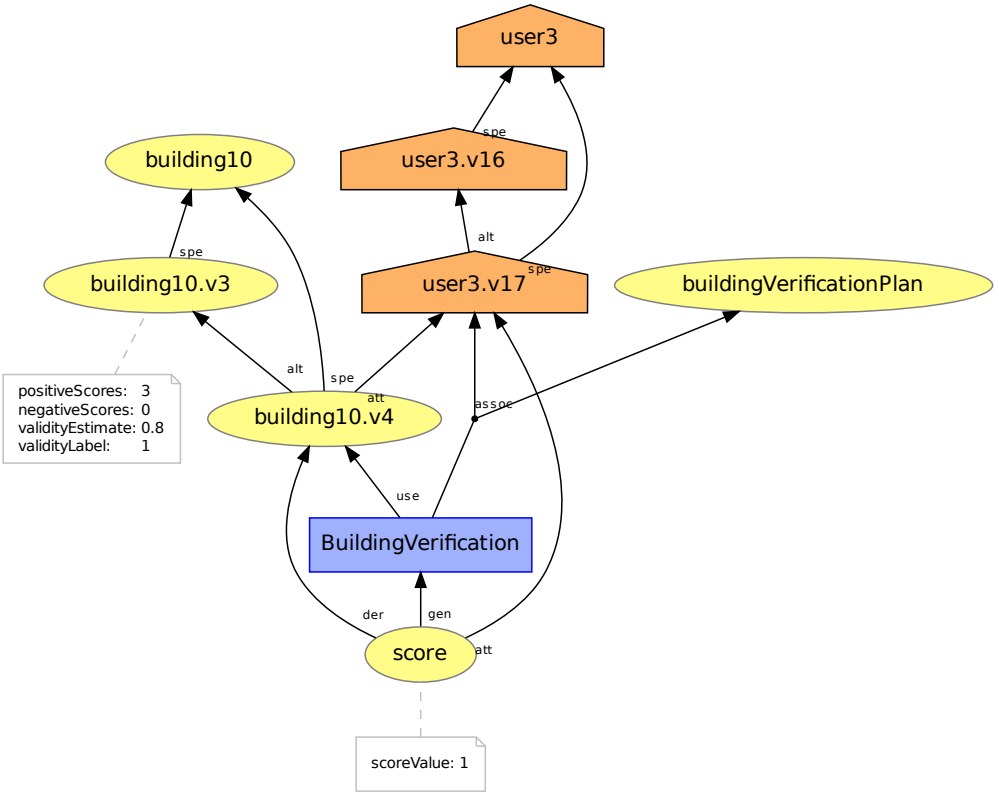


Figure 5.1: Sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product).

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad d^v.positiveScores \leftarrow d^v.oldPositiveScores + d^v.verificationValue\\
&\qquad\quad \text{if } d^v.verificationValue > 0\\
&\quad d^v.negativeScores \leftarrow d^v.oldNegativeScores + d^v.verificationValue\\
&\qquad\quad \text{if } d^v.verificationValue < 0\\
&\quad d^v.validityEstimate \leftarrow \text{Apply Equation 5.3 with}\\
&\qquad\quad S^+(d^v) = d^v.positiveScores\\
&\qquad\quad S^-(d^v) = d^v.negativeScores\\
&\quad d^v.validityLabel \leftarrow \text{Apply Equation 5.4 with}\\
&\qquad\quad Q_V(d^v) = d^v.validityEstimate
\end{aligned}
\tag{5.22}
$$

Figure 5.1 represents a sample provenance graph generated by a crowdsourcing application[2]. The provenance of scoring of a data product (Building10) is shaped by the provenance patterns.

[2]Appendix A.5 provides the PROV-N representation of this provenance graph

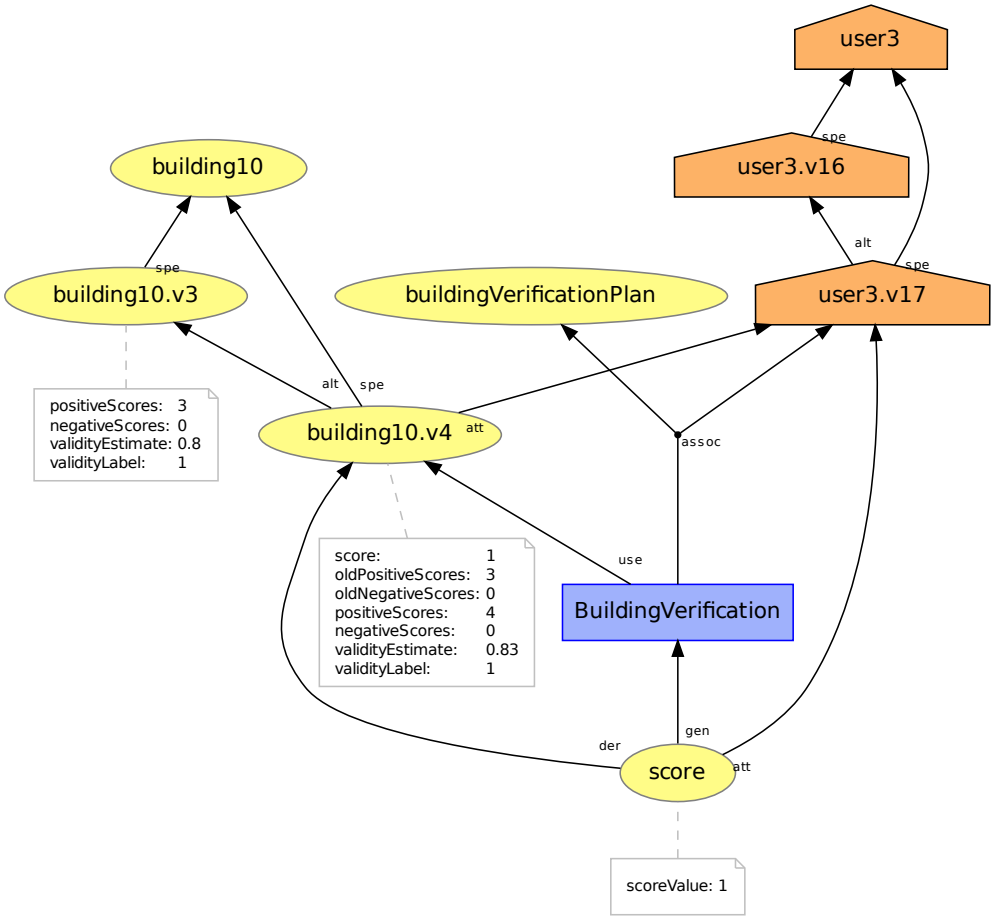


Figure 5.2: Annotated provenance graph with validity estimate measures.

Figure 5.2 shows the annotated provenance graph of Figure 5.1 with the quality measures introduced in Section 5.1.1 and according to the rules introduced in Section 5.2.1.

5.2.2 Mapping to Compute Creator Reliability

The following sections demonstrate how *PEDRA* instantiates *ACF* by implementing the three computational rules ($F_{forward}$, $F_{backward}$, $F_{aggregate}$) in order to compute quality measures listed in Table 5.2.

Rules 4 - 10

In order to compute CREATOR RELIABILITY for a version of a worker, we first need to propagate backward the $L_{Q_V}(d^k)$ where k is the last version of the data product. Hence, first, we need to find the last version of the data product. Figure 5.3 presents a sample

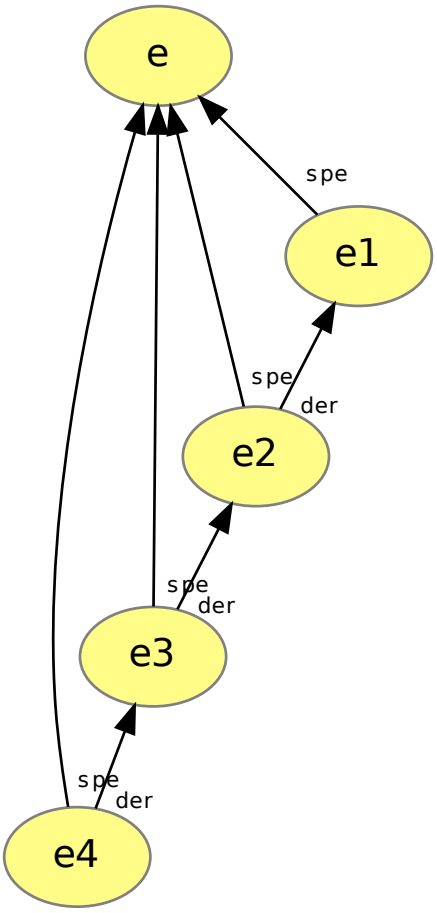


Figure 5.3: Sample provenance graph where a data product has 4 versions.

provenance graph where a data product has 4 versions. In this example, the $L_{Q_V}(e4)$ is required to propagate backward to d.

Rule 5.23 describes a $f_{backward}$ for each wasDerivedFrom relation with the type prov:Revision that computes an annotation with the key "hasRevision" for any node which has a revision. A version of a data product has a type "DataProductVersioned" according to the *revision pattern* (Section 3.3.3). In this case, one annotation is computed for d^v: "hasRevision" which specifies that d^v has a revision (i.e. that specific version of the data product is not the last version).

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If wasDerivedFrom}(d^{v+1}, d^v, \text{[prov:type=prov:Revision]})\\
&\quad \text{and } d^v.type = \text{DataProductVersioned and}\\
&\quad \text{and } d^{v+1}.type = \text{DataProductVersioned}\\
&\qquad d^v.hasRevision \leftarrow 1
\end{aligned}
\tag{5.23}
$$

According to Rule 5.23, the last version of a data product is not annotated with the annotation with the key hasRevision.

Rule 5.24 describes a function that propagates backward the contents associated with key "validityLabel" of d^{v+1} (i.e. VALIDITY LABEL of d^{v+1}) to the key "mainValidityLabel" of d^v. In this case, one annotation is computed for d^v: "mainValidityLabel" which specifies the VALIDITY LABEL of d^v. This rule propagates the annotation if a condition holds: the condition is if d^{v+1} is not annotated with "hasRevision" but d^v is annotated with "hasRevision"; i.e. d^{v+1} is the last version.

$$\begin{aligned} &F_{backward}: \\ &\text{If } \text{wasDerivedFrom}(d^{v+1}, d^v, [\text{prov:type=prov:Revision}]) \\ &\quad \text{and } d^{v+1}.hasRevision \neq 1 \text{ and } d^v.hasRevision = 1 \\ &\quad\quad d^v.mainValidityLabel \leftarrow d^{v+1}.validityLabel \end{aligned} \tag{5.24}$$

Rule 5.25 describes a function that propagates backward the contents associated with key "mainValidityLabel" of d^v to the key "mainValidityLabelFromVersion" of d. In this case, one annotation is computed for d: "mainValidityLabelFromVersion" which specifies the VALIDITY LABEL of d if a condition holds: the condition is if d^{v+1} is not annotated with "hasRevision" but d^v is annotated with "hasRevision"; i.e. d^{v+1} is the last version. The "mainValidityLabel" of d^v is given the "validityLabel" of d^{v+1}, if d^{v+1} is established to be the last version, i.e. it does not have a revision, but itself is a revision of d^v.

$$\begin{aligned} &F_{backward}: \\ &\text{If } \text{specializationOf}(d^v, d) \\ &\quad d.mainValidityLabelFromVersion \leftarrow d^v.mainValidityLabel \end{aligned} \tag{5.25}$$

Rule 5.26 describes a function that propagates backward the contents associated with key "mainValidityLabelFromVersion" of d to the key "mainValidityLabelFromOriginal" of the creation activity, denoted as dc, (refer to Section 3.3.1 for *create pattern*). In this case, one annotation is computed for dc: "mainValidityLabelFromOriginal" which specifies the VALIDITY LABEL of dc.

$$\begin{aligned} &F_{backward}: \\ &\text{If } \text{wasGeneratedBy}(d, dc) \\ &\quad dc.mainValidityLabelFromOriginal \leftarrow d.mainValidityLabelFromVersion \end{aligned} \tag{5.26}$$

Rule 5.27 describes a function that propagates backward the contents associated with key "mainValidityLabelFromOriginal" of dc to the key "validCreation" or "invalidCreation" of u^v, depending on the value of "mainValidityLabelFromOriginal". In this case, two annotations are computed for u^v: (1) "validCreation" which specifies VALID CREATIONS of u^v and (2) "invalidCreation" which specifies INVALID CREATIONS of u^v.

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \mathrm{wasAssociatedWith}(dc, u^v)\\
&\quad u^v.validCreation \leftarrow dc.mainValidityLabelFromOriginal\\
&\qquad \text{if } dc.mainValidityLabelFromOriginal = 1\\
&\quad u^v.invalidCreation \leftarrow dc.mainValidityLabelFromOriginal\\
&\qquad \text{if } dc.mainValidityLabelFromOriginal = -1
\end{aligned}
\tag{5.27}
$$

Rule 5.28 describes a function that propagates forward the contents associated with key "validCreations" and "invalidCreations" of u^v to the key "oldValidCreations" and "oldInvalidCreations" of u^{v+1}, respectively. In this case, two annotations are computed for u^{v+1}: (1) "oldValidCreations" which specifies OLD VALID CREATIONS of u^{v+1} and (2) "oldInvalidCreations" which specifies OLD INVALID CREATIONS of u^{v+1}.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasDerivedFrom}(u^{v+1}, u^v, [\text{prov:type=prov:Revision}])\\
&\quad u^{v+1}.oldValidCreations \leftarrow u^v.validCreations\\
&\quad u^{v+1}.oldInvalidCreations \leftarrow u^v.invalidCreations
\end{aligned}
\tag{5.28}
$$

Rule 5.29 describes a function that aggregates all the annotations of u^v to compute a new annotation for u^v. In this case, three annotations are computed for u^v: (1) "validCreations" which specifies VALID CREATIONS of u^v, (2) "invalidCreations" which specifies INVALID CREATIONS of u^v, (1) "creatorReliability" which specifies CREATOR RELIABILITY of u^v,

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad u^v.validCreations \leftarrow u^v.oldValidCreations + u^v.validCreation\\
&\quad u^v.invalidCreations \leftarrow u^v.oldInvalidCreations + u^v.invalidCreation\\
&\quad u^v.creatorReliability \leftarrow \text{Apply Equation 5.8 with}\\
&\qquad S_C^+(u^v) = u^v.validCreations\\
&\qquad S_C^-(u^v) = u^v.invalidCreations
\end{aligned}
\tag{5.29}
$$

Figure 5.4 represents a sample provenance graph generated by a crowdsourcing application[3]. The provenance of creation of a data product (Building10) is shaped by the provenance patterns.

Figure 5.5 shows the annotated provenance graph of Figure 5.4 with the quality measures introduced in Section 5.1.2 and according to the rules introduced in Section 5.2.2.

[3]Appendix A.6 provides the PROV-N representation of this provenance graph

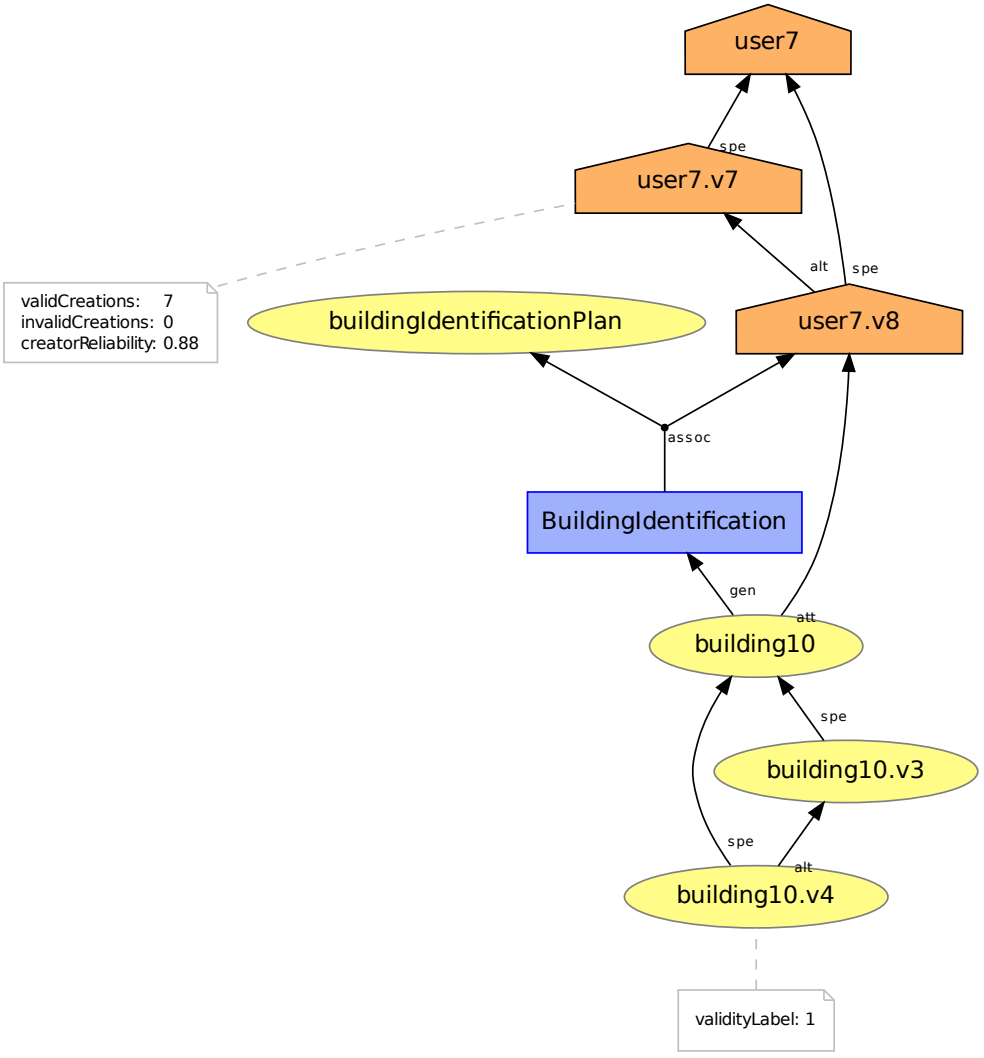


Figure 5.4: Sample provenance graph generated by a crowdsourcing application shaped by create pattern (provenance of creation of a data product).

5.2.3 Mapping to Compute Scorer Reliability

The following sections demonstrate how *PEDRA* instantiates *ACF* by implementing the three computational rules ($F_{forward}$, $F_{backward}$, $F_{aggregate}$) in order to compute quality measures listed in Table 5.3.

Rules 11 - 14

Rule 5.30 describes a function that propagates backward the contents associated with key "scoreValue" of *score* to the key "verificationValue" of d^v, respectively. In this case,

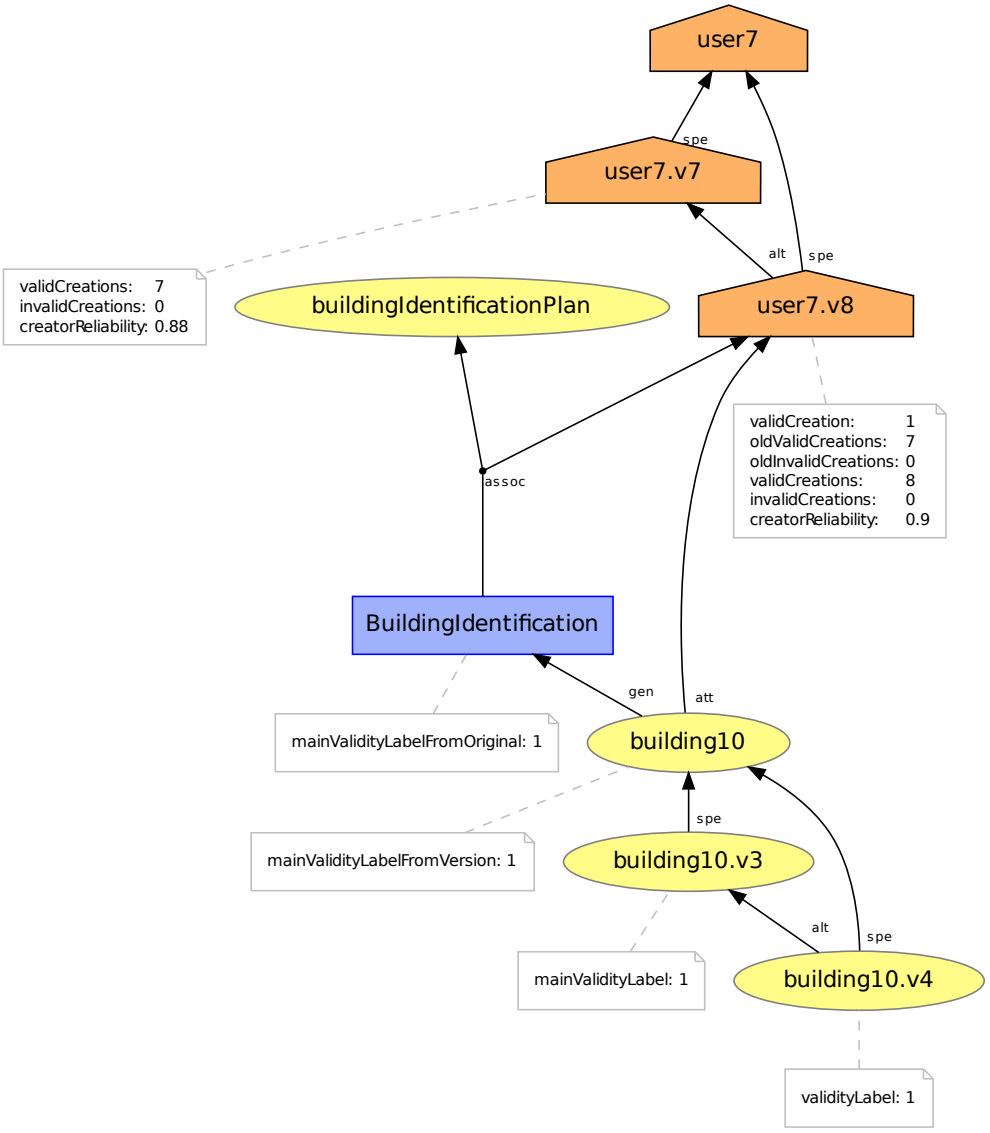


Figure 5.5: Annotated provenance graph with creator reliability measures.

one annotations is computed for d^v: "verificationValue" which specifies VERIFICATION VOTE of d^v.

$$
\begin{aligned}
&F_{backward}: \\
&\text{If } \text{wasDerivedFrom}(score, d^v) \\
&\quad d^v..verificationValue \leftarrow score.scoreValue
\end{aligned}
\tag{5.30}
$$

Rule 5.31 describes a function that checks if the contents associated with key "verificationValue" and "validityLabel" of d^v are the same or not. If they are the same, a new

annotation with the key "alignedScore" is computed for u^v. Otherwise, a new annotation with the key *notAlignedScore* is computed for u^v. In this case, two annotations are computed for u^v: (1) "alignedScore" which specifies ALIGNED SCORES of u^v and (2) "notAlignedScore" which specifies NOT-ALIGNED SCORES of u^v.

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \mathrm{wasAttributedTo}(d^v, u^v)\\
&\quad u^v.alignedScore \leftarrow 1\\
&\qquad\qquad \text{if } d^v.verificationValue = d^v.validityLabel\\
&\quad u^v.notAlignedScore \leftarrow 1\\
&\qquad\qquad \text{if } d^v.verificationValue \neq d^v.validityLabel
\end{aligned}
\tag{5.31}
$$

Rule 5.32 describes a function that propagates forward the contents associated with key "alignedScores" and "notAlignedScores" of u^v to the key "oldAlignedScores" and "oldNotAlignedScores" of u^{v+1}, respectively. In this case, two annotations are computed for u^{v+1}: (1) "oldAlignedScores" which specifies OLD ALIGNED SCORES of u^{v+1} and (2) "oldNotAlignedScores" which specifies OLD NOT-ALIGNED SCORES of u^{v+1}.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasDerivedFrom}(u^{v+1}, u^v, [\text{prov:type=prov:Revision}])\\
&\quad u^{v+1}.oldAlignedScores \leftarrow u^v.alignedScores\\
&\quad u^{v+1}.oldNotAlignedScores \leftarrow u^v.notAlignedScores
\end{aligned}
\tag{5.32}
$$

Rule 5.33 describes a function that aggregates all the annotations of u^v to compute new annotation for u^v. In this case, three annotations are computed for u^v: (1) "alignedScores" which specifies ALIGNED SCORES for u^v, (2) "notAlignedScores" which specifies NOT-ALIGNED SCORES for u^v, and (3) "scorerReliability" which specifies SCORER RELIABILITY for u^v.

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad u^v.alignedScores \leftarrow u^v.oldAlignedScores + u^v.alignedScore\\
&\quad u^v.notAlignedScores \leftarrow u^v.oldNotAlignedScores + u^v.notAlignedScore\\
&\quad u^v.scorerReliability \leftarrow \text{Apply Equation 5.12 with}\\
&\qquad S_S^+(u^v) = u^v.alignedScores\\
&\qquad S_S^-(u^v) = u^v.notAlignedScores
\end{aligned}
\tag{5.33}
$$

Figure 5.6 represents a sample provenance graph generated by a crowdsourcing application[4]. The provenance of scoring of a data product (Building10) is shaped by the provenance patterns.

[4]Appendix A.7 provides the PROV-N representation of this provenance graph

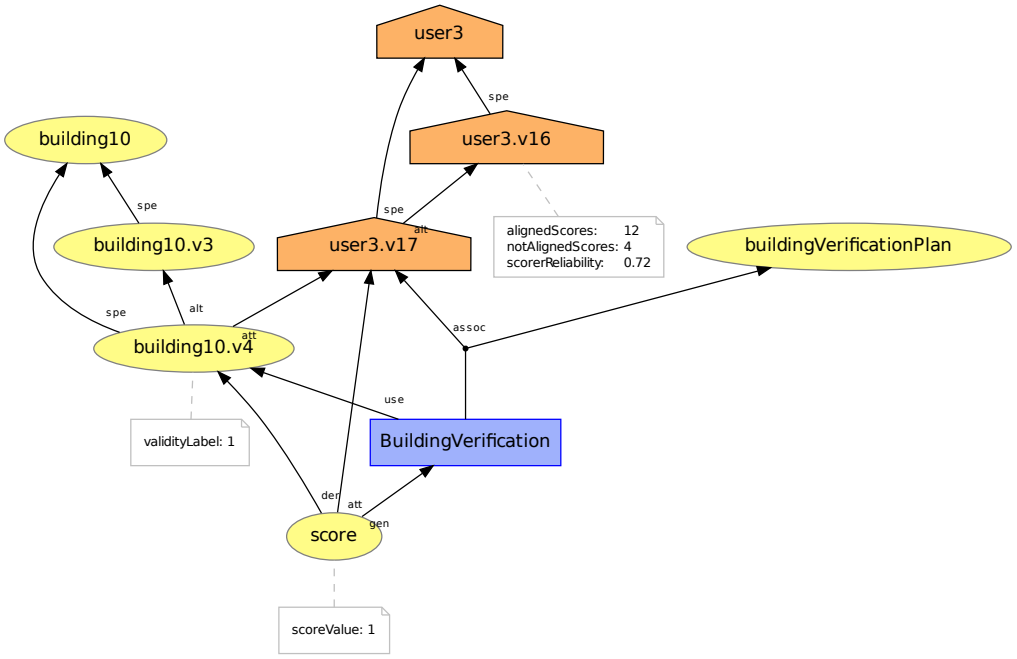


Figure 5.6: Sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product) to compute scorer reliability.

Figure 5.7 shows the annotated provenance graph of Figure 5.6 with the quality measures introduced in Section 5.1.2 and according to the rules introduced in Section 5.2.3.

5.2.4 Mapping to Compute Validity Rating

The following sections demonstrate how *PEDRA* instantiates *ACF* by implementing the three computational rules ($F_{forward}$, $F_{backward}$, $F_{aggregate}$) in order to compute quality measures listed in Table 5.4.

Rules 15 - 19

Rule 5.34 describes a function that propagates forward the contents associated with key "scorerReliability" of u^v to the key "scorerReliabilityFromVersion" of the verification activity, denoted as bv (refer to Section 3.3.2 for *score pattern*). In this case, one annotations is computed for bv: "scorerReliabilityFromVersion" which specifies SCORER RELIABILITY FROM VERSION of bv, given the SCORER RELIABILITY of u^v.

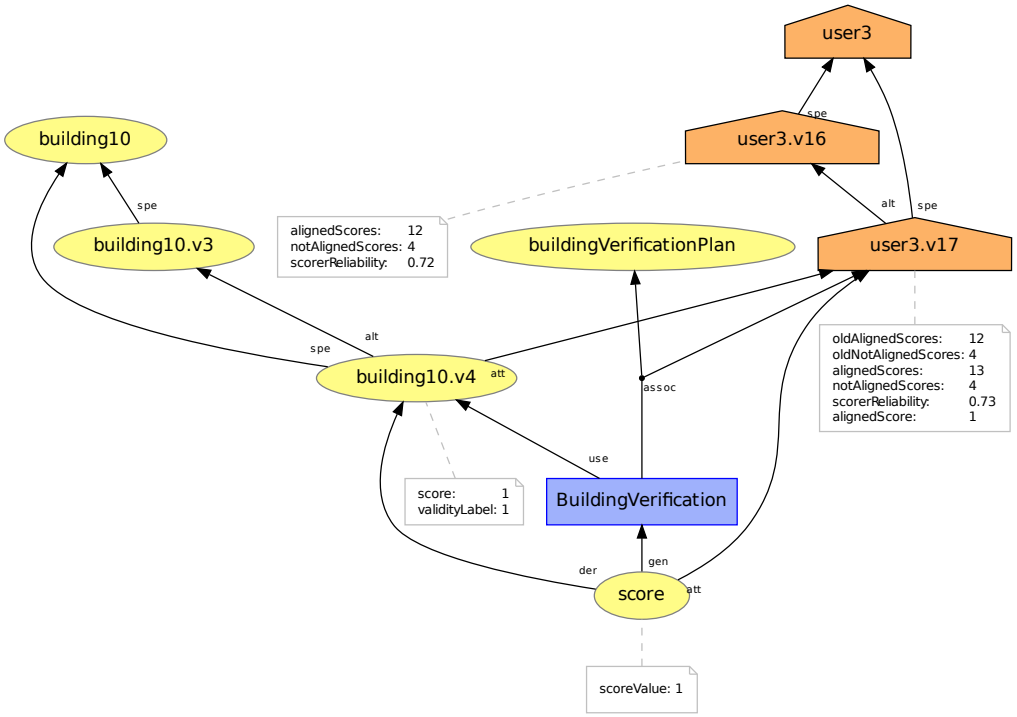


Figure 5.7: Annotated provenance graph with scorer reliability annotations.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasAssociatedWith}(bv, u^v) \\
&\qquad bv.scorerReliabilityFromVersion \leftarrow u^v.scorerReliability
\end{aligned}
\tag{5.34}
$$

Rule 5.35 describes a function that propagates forward the contents associated with key "scorerReliabilityFromVersion" of *bv* to the key "scorerReliabilityFromActivity" of *score*. In this case, one annotation is computed for *score*: "scorerReliabilityFromActivity" that specifies SCORER RELIABILITY FROM ACTIVITY of *score* given SCORER RELIABILITY FROM VERSION of *bv*.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasGeneratedBy}(score, bv)\\
&\qquad score.scorerReliabilityFromActivity \leftarrow bv.scorerReliabilityFromVersion
\end{aligned}
\tag{5.35}
$$

Rule 5.36 describes a function that propagates backward the contents associated with key "scorerReliabilityFromActivity" of *score* to the key "scorerReliabilityFromScore" of d^v. In this case, one annotation is computed for d^v: "scorerReliabilityFromScore" that

specifies SCORER RELIABILITY FROM SCORE of d^v given SCORER RELIABILITY FROM ACTIVITY of *score*.

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \mathrm{wasDerivedFrom}(score, d^v)\\
&\quad d^v.scorerReliabilityFromScore \leftarrow +score.scorerReliabilityFromActivity\\
&\qquad \text{if } score.value > 0\\
&\quad d^v.scorerReliabilityFromScore \leftarrow -score.scorerReliabilityFromActivity\\
&\qquad \text{if } score.value < 0
\end{aligned}
\tag{5.36}
$$

Rule 5.37 describes a function that propagates forward the contents associated with key "cws" of d^v to the key "oldCws" of d^{v+1}. In this case, one annotation is computed for d^{v+1}: "oldCws" that specifies OLD VALIDITY RATING of d^{v+1}.

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \mathrm{wasDerivedFrom}(d^{v+1}, d^v, [\text{prov:type=prov:Revision}])\\
&\quad d^{v+1}.oldCws \leftarrow d^v.cws
\end{aligned}
\tag{5.37}
$$

Rule 5.38 describes a function that aggregates all the annotations of d^v to compute new annotation for d^v. In this case, two annotations are computed for d^v: (1) "cws" that specifies VALIDITY RATING of d^v and (2) "termination" that specifies TERMINATION of d^v.

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad d^v.cws \leftarrow d^v.oldCws + d^v.scorerReliabilityFromScore\\
&\quad d^v.termination \leftarrow \text{Apply Equation 5.15 with}\\
&\qquad \mathrm{CWS}(d^v) = d^v.cws
\end{aligned}
\tag{5.38}
$$

Figure 5.8 represents a sample provenance graph generated by a crowdsourcing application[5]. The provenance of scoring of a data product (Building10) is shaped by the provenance patterns.

Figure 5.9 shows the annotated provenance graph of Figure 5.8 with the quality measures introduced in Section 5.1.3 and according to the rules introduced in Section 5.2.4.

[5]Appendix A.8 provides the PROV-N representation of this provenance graph

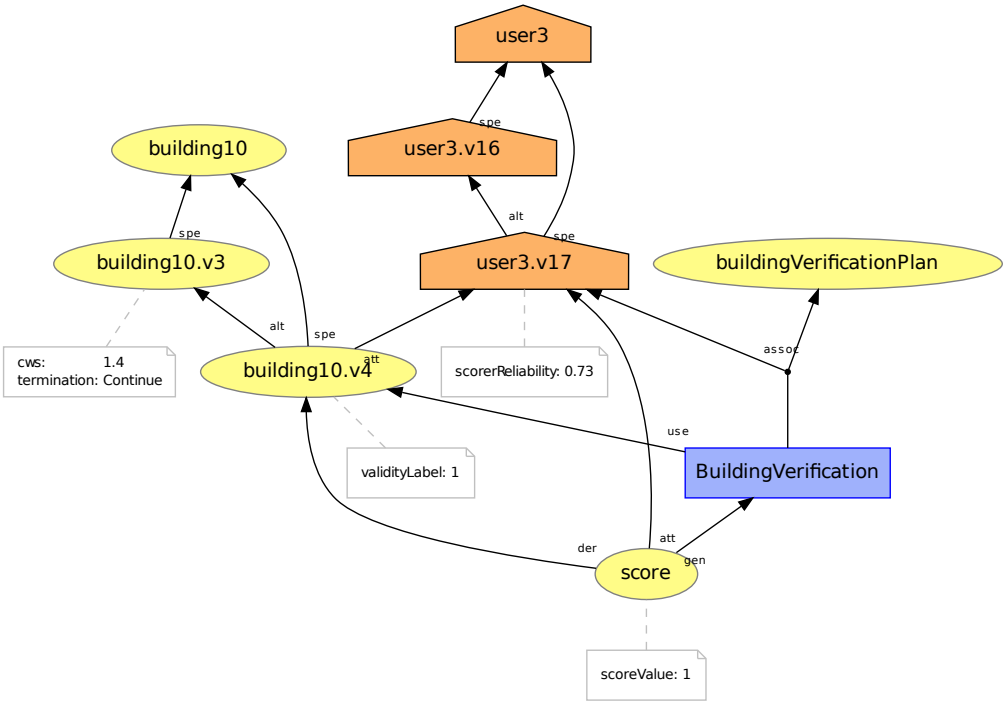


Figure 5.8: Sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product) to compute validity rating.

5.3 Termination of PEDRA

In Section 5.1, we introduced the quality measures computed by *PEDRA* and in Section 5.2, we showed how this computation is mapped to *ACF*.

In Section 4.7, we stated that propagation and computation terminate at some point given the following conditions:

- The instantiation must not propagate rules across edges that form cycles.
- Annotations updated by rules can form a strict order.

In this section, we show how computation of quality measures by *PEDRA* terminates at some point. Note that, we assume the provenance graph submitted by a crowdsourcing application that is shaped by the provenance patterns (Chapter 3) does not have a cycle.

In order to compute VALIDITY ESTIMATE for a version of a data product (d^v), *PEDRA* follows the below items in a strict order.

1. compute "verificationValue" attribute for d^v based on Rule 5.20

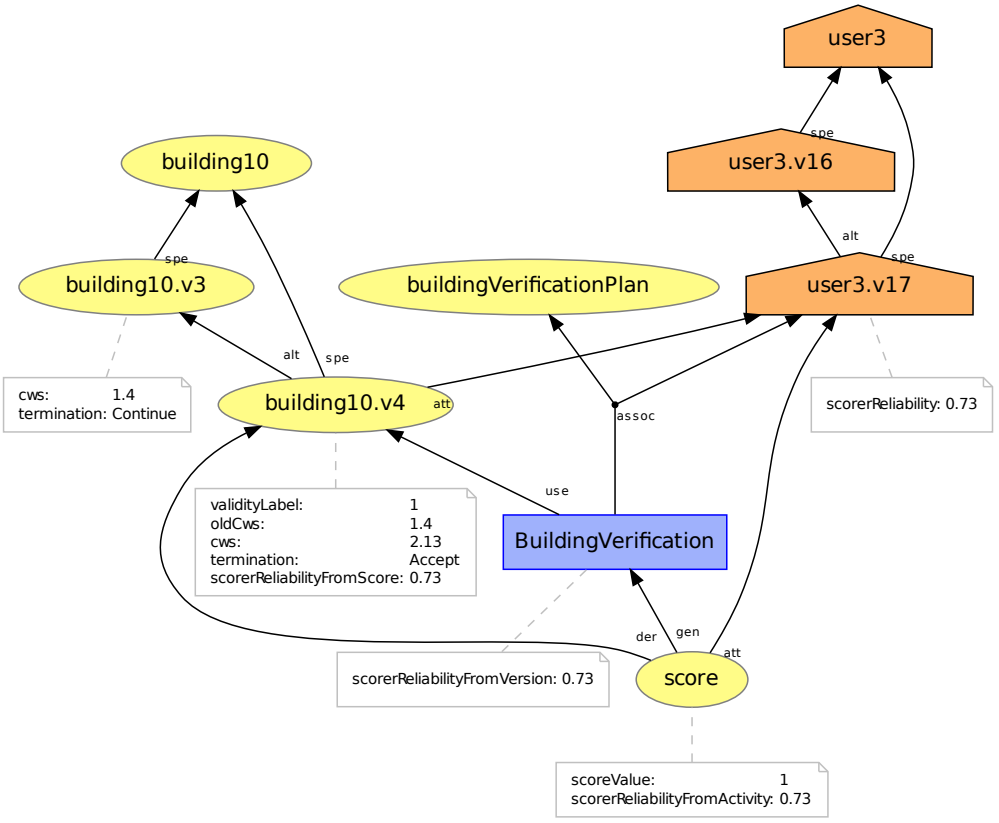


Figure 5.9: Annotated provenance graph with validity rating annotations.

2. compute "oldPositiveScores" and "oldNegativeScores" for d^v based on Rule 5.21
3. compute "positiveScores", "negativeScores", "validityEstimate", and "validityLabel" for d^v based on Rule 5.22

In order to compute CREATOR RELIABILITY for a version of a worker (u^v), *PEDRA* follows the below items in a strict order.

1. compute "mainValidityLabel" for a version of a data product (d^v) that u^v created based on Rule 5.23
2. compute "mainValidityLabelFromVersion" for the general/version-less data product based on Rule 5.24
3. compute "mainValidityLabelFromOriginal" for the creation activity based on Rule 5.26
4. compute "validCreation" and "invalidCreation" for u^v based on Rule 5.27
5. compute "oldValidCreations" and "oldInvalidCreations" for u^v based on Rule 5.28

6. compute "validCreations", "invalidCreations", and "creatorReliability" for u^v based on Rule 5.29

In order to compute SCORER RELIABILITY for a version of a worker (u^v), *PEDRA* follows the below items in a strict order.

1. compute "verificationValue" for a version of a data product (d^v) that u^v scored based on Rule 5.30
2. compute "alignedScore" and "notAlignedScore" for u^v based on Rule 5.31
3. compute "oldAlignedScores" and "oldNotAlignedScores" for u^v based on Rule 5.32
4. compute "alignedScores", "notAlignedScores", and "scorerReliability" for u^v based on Rule 5.33

In order to compute VALIDITY RATING for a version of a data product (d^v), *PEDRA* follows the below items in a strict order.

1. compute "scorerReliabilityFromVersion" for the scoring activity done by a scorer based on Rule 5.34
2. compute "scorerReliabilityFromActivity" for the score the scorer provided based on Rule 5.35
3. compute "scorerReliabilityFromScore" for d^v based on Rule 5.36
4. compute "oldCws" for d^v based on Rule 5.37
5. compute "cws" and "termination" for d^v based on Rule 5.38

As it can be seen, the propagation and computation of annotations (quality measures) in *PEDRA* form a strict order. Furthermore, in the antecedent of a rule, an annotation of a version of a data product or a worker is computed only based on the previous version (and according to the rule order). The consequent rule creates an annotation that did not exist before. As such, we conclude that the annotation propagation and computation in *PEDRA* terminates at some point.

5.4 Empirical Study

This section presents the advantages that *PEDRA* offers through empirical evaluation. First, the environment and the methodology used to perform the experiment are described (Section 5.4.1). CollabMap is chosen as the scenario since it already records the

provenance of its data. With minor modifications to ensure its provenance conforms with our provenance patterns, it is possible to retrofit *PEDRA* into CollabMap. This facilitates the comparison between the accuracy of the quality model originally employed in CollabMap with *PEDRA* (Section 5.4.2). Besides comparing the accuracy, the utility of provenance in terms of reducing uncertainty over computed quality measures for a data product or a worker is investigated in Section 5.4.3.

5.4.1 Experiment Methodology

We ran the original CollabMap application on Amazon Mechanical Turk (AMT), posting more than 2,700 HITs (Human Intelligence Tasks). We applied for ethics approval through the University of Southampton, under the reference number ERGO/FPSE/11509. The complete dataset, containing the provenance data in JSON representation, can be accessed from: `http://eprints.soton.ac.uk/id/eprint/378382`.

For each completed HIT, a worker was paid 3 cents. This version of CollabMap (Ramchurn et al., 2013) employs a majority voting approach for accepting/rejecting tasks, hence, the result from this version is referred as *CollabMap-MajorityVoting*.

After the data was collected by CollabMap, *PEDRA* was employed to analyse the provenance generated by CollabMap and assess again the quality of all its data as per TERMINATION measure (Equation 5.15). The result produced by *PEDRA* is referred to as *CollabMap-PEDRA*. Note that *CollabMap-PEDRA* relied solely on provenance as the means of communication between *PEDRA* and CollabMap to ensure that the results reported here apply beyond this specific application.

Following, the accuracy of both approaches in terms of correctly classifying data are compared.

It was mentioned in Section 2.1.7 that two roll-outs of CollabMap, separate from our work, took place. Inspecting the data sets that were collected from these roll-outs, Table 5.5 introduces the thresholds for choosing data of sufficient quality and discarding those of low quality (refer to Equation 5.4 and Equation 5.15).

Quality Measure	Threshold	Value
$L_{Q_V}(d^v)$	TV_{high}	0.8
	TV_{low}	0.3
$T(d^v)$	TT_{high}	1.5
	TT_{low}	-1.5

Table 5.5: The threshold set on the validity label and termination of a version of a data product.

5.4.2 Accuracy

Method We compile a list of ground truths that contains the validity label of each data product by asking an expert to check all the data products generated in the experiment and to either accept or reject them. Using the ground truths, the number of times each approach agrees with the expert, denoted by X, is measured. Let K be the total number of data products ($K > 0$), the *accuracy* of an approach is defined as $\frac{X}{K}$.

Hypothesis 1 The quality assessment in *CollabMap-PEDRA* has the same or higher accuracy than that of *CollabMap-MajorityVoting*.

If X denotes the number of successes in K independent trials with a binary outcome, each of which may result in a success with probability p, then X can be modelled as: $X \sim Binomial(K, p)$. Let $X_1 \sim Binomial(K, p_1)$ represents the binomial distribution for *CollabMap-PEDRA* and $X_2 \sim Binomial(K, p_2)$ represents the binomial distribution for *CollabMap-MajorityVoting*. Given that K is the total number of data products, X_1 and X_2 are the number of times the result of *CollabMap-PEDRA* or *CollabMap-MajorityVoting* were equal to the ground truth (number of successes), U_1 and U_2 are the number of times a decision on the validity of the data product was not made by *CollabMap-PEDRA* or *CollabMap-MajorityVoting*, and p_1 and p_2 are the success probabilities of the two approaches. Hypothesis 1 is then equivalent to $p_1 \geq p_2$. Therefore, the null hypothesis is formed as follows:

$$H_0 : p_1 < p_2 \tag{5.39}$$

Using the Bayesian approach, the distributions p_1 and p_2 can be determined using the actual value of X_1 and X_2 (MacKay, 2003) as follows.

$$p_1 \sim BetaDistribution(1 + X_1, 1 + K - X_1) \tag{5.40}$$
$$p_2 \sim BetaDistribution(1 + X_2, 1 + K - X_2) \tag{5.41}$$

The *p-value* is computed as the integral of probability of $[p_1 < p_2]$ over all values of p_1:

$$p\text{-}value = P(p_1 < p_2) \tag{5.42}$$
$$\mathbb{E}_{p_1}[P(p_1 < p_2)] = \int_0^1 P(p_2 > x) * \mathrm{PDF}_{p_1}(x)\,\mathrm{d}x \tag{5.43}$$
$$= \int_0^1 \left[1 - \mathrm{CDF}_{p_2}(x)\right] * \mathrm{PDF}_{p_1}(x)\,\mathrm{d}x \tag{5.44}$$

where CDF_{p_2} denotes the cumulative distribution function of p_2 and PDF_{p_1} denotes the probability density function of p_1.

Equation 5.44 is computed by using Wolfram Mathematica[6] as per Equation 5.47 where U_1 and U_2 are the total number of data products that *CollabMap-PEDRA* and *CollabMap-MajorityVoting* are uncertain about the validity of the data product, respectively:

$$pedra = BetaDistribution[1 + X_1, 1 + (K - U_1) - X_1]; \quad (5.45)$$

$$mv = BetaDistribution[1 + X_2, 1 + (K - U_2) - X_2]; \quad (5.46)$$

$$Integrate[(1 - CDF[mv, a1]) * PDF[pedra, a1], a1, 0, 1] \quad (5.47)$$

Analysis In this experiment, 612 data products were created and scored by around 200 workers. The accuracy of *CollabMap-PEDRA* is $\sim 95.5\%$ and accuracy of *CollabMap-MajorityVoting* is $\sim 89.1\%$.

Equation 5.44 gives *p-value* < 0.00002. Figure 5.10 shows the beta distribution of the result of *CollabMap-PEDRA* and *CollabMap-MajorityVoting*; the y-axis represents PDF. It can be observed from the plot that they overlap only slightly as shown by Equation 5.44.

Based on the computed accuracy and *p-value*, we conclude that there is a strong evidence to reject the null hypothesis (H_0) with over 99% confidence. In other words, Hypothesis 1 is validated: the accuracy of *CollabMap-PEDRA* is higher than *CollabMap-MajorityVoting* and it is statistically significant.

5.4.3 Provenance and Uncertainty

While the quality measures introduced in this chapter rely on application data, we argue that the availability of provenance information was critical in retrieving such application data. Here, we do not seek to demonstrate the benefits of a given provenance model; instead, we show that it is the provenance relation that allows the accessing of application data relevant to quality assessment. Intuitively, the more provenance information are exploited and the more historical application data is accessed, the lower the uncertainty in the quality measures computed over such data.

Hypothesis 2 There is an inverse correlation between the amount of provenance information exploited to compute the quality measures (i.e. VALIDITY ESTIMATE and CREATOR RELIABILITY) and the uncertainty over them.

[6] http://www.wolfram.com/mathematica/

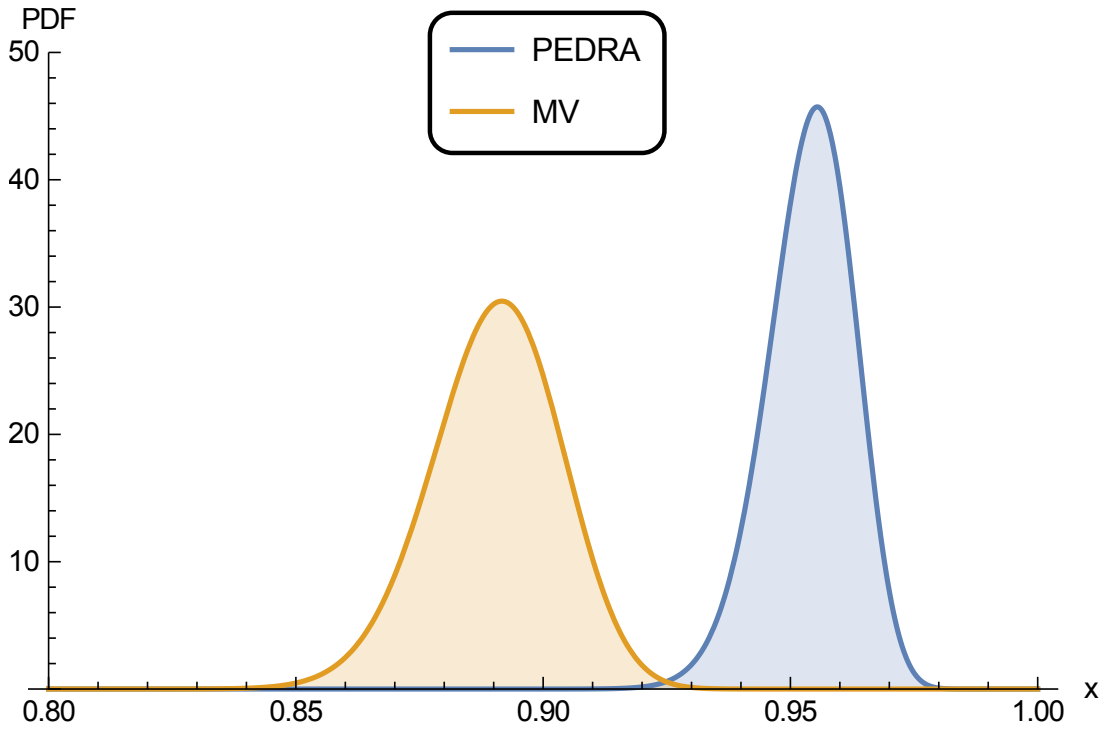


Figure 5.10: The beta distribution of CollabMap-PEDRA and CollabMap-MajorityVoting.

Method We analyse the utility of provenance in terms of uncertainty. More specifically, we consider VALIDITY ESTIMATE of a version of a data product and CREATOR RELIABILITY of a version of a creator, and compute (1) how much provenance was exploited to compute the quality measure and (2) the uncertainty over the quality measure.

First, we focus the analysis on VALIDITY ESTIMATE and data products. The analysis is based over the lifespan of a data product which starts when a scorer provides a score to a version of the data product (*score pattern* See Section 3.3.2) and ends when a decision of accepting or rejecting that data product is made (see the TERMINATION in Section 5.1.3).

For each version of a data product, d^v, we compute how much provenance was exploited to compute its VALIDITY ESTIMATE (denoted as $D_{Q_V}(d^v)$) by looking at the *revision pattern* (See Section 3.3.3) and computing the total number of versions up to d^v as per Equation 5.48.

$$D_{Q_V}(d^v) = |\{d_k | k \leq v\}| \tag{5.48}$$

After quantifying how much provenance was exploited to compute VALIDITY ESTIMATE for a version of a data product, then, we compute provenance lifespan of a version of a data product, $P(d^v)$, by dividing $D_{Q_V}(d^v)$ (Equation 5.48) by the maximum number of $D_{Q_V}(d)$ of the data product which is the $D_{Q_V}(d^v)$ of the last version of the data

product. This allows us to base our analysis in terms of the data product provenance lifespan (0% to 100% completion).

$$P(d^v) = \frac{D_{Q_V}(d^v)}{\max D_{Q_V}(d)} \tag{5.49}$$

We quantify the uncertainty of VALIDITY ESTIMATE ($Q_V(d^v)$), denoted as $\gamma_{Q_V}(d^v)$, by its variance. For a random variable X beta-distributed with parameters α and β, its variance is given by Johnson and Kotz (1970) as follows:

$$\text{Var}\left[X\right] = \frac{\alpha \times \beta}{(\alpha+\beta)^2 \times (\alpha+\beta+1)} \tag{5.50}$$

Since $Q_V(d^v)$ is calculated from the beta distribution with parameters $\alpha = S^+(d^v) + 1$ and $\beta = S^-(d^v) + 1$, we can calculate its variance as follows:

$$\gamma_{Q_V}(d^v) = \text{Var}\left[Q_V(d^v)\right] \tag{5.51}$$

For each version of a creator, u^v, we compute how much provenance was exploited to compute its CREATOR RELIABILITY (denoted as $D_{Q_C}(u^v)$) by looking at the *agent pattern* (See Section 3.3.4) and computing the total number of versions up to u^v as per Equation 5.52.

$$D_{Q_C}(u^v) \quad = \quad |\{u_k | k \leq v\}| \tag{5.52}$$

Similar to computing uncertainty over VALIDITY ESTIMATE for a version of a data product, we quantify the uncertainty of CREATOR RELIABILITY, denoted as $\gamma_{Q_C}(u^v)$, by its variance, as per Equation 5.53.

$$\gamma_{Q_C}(u^v) \quad = \quad \text{Var}\left[Q_C(u^v)\right] \tag{5.53}$$

in which, $\alpha = S_C^+(u^v) + 1$ and $\beta = S_C^-(u^v) + 1$.

Analysis Figure 5.11 demonstrates the average uncertainty over VALIDITY ESTIMATE against how much provenance was exploited using a logarithmic scale. The x-axis represents the lifespan of data products and the y-axis represents the average uncertainty over VALIDITY ESTIMATE for all data products that have the same lifespan.

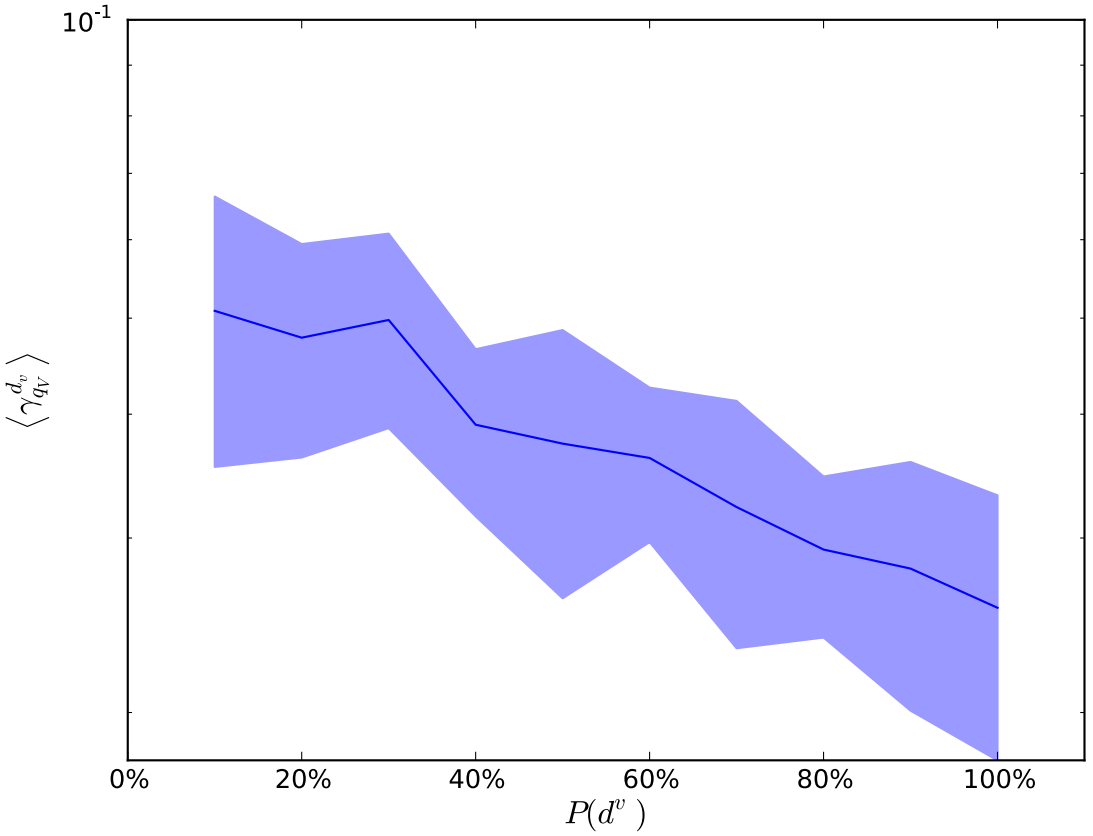


Figure 5.11: The average uncertainty over validity estimate against the amount of exploited provenance

Similarly, Figure 5.12 demonstrates the average uncertainty over CREATOR RELIABILITY against how much provenance was exploited using a logarithmic scale. The x-axis represents $D_{Q_C}(u^v)$ and the y-axis represents the average uncertainty over CREATOR RELIABILITY.

The shaded region around the lines in both Figure 5.11 and Figure 5.12 represents the standard deviation when computing the average uncertainty.

Both figures demonstrate that as more provenance is exploited, the uncertainty over computed quality measures decreases.

In order to quantify the correlation between the uncertainty over VALIDITY ESTIMATE and CREATOR RELIABILITY and the amount of provenance exploited to compute them, we calculate the Pearson correlation coefficients for the data plotted in both Figure 5.11 and Figure 5.12 and the corresponding p-value (as presented in Table 5.6). In all cases, the Pearson correlation coefficients show very strong negative correlation with very small p-values. Hence, we can confirm Hypothesis 2, i.e. There is an inverse correlation between the amount of provenance information exploited to compute the quality measures (i.e. VALIDITY ESTIMATE and CREATOR RELIABILITY) and the uncertainty over them.

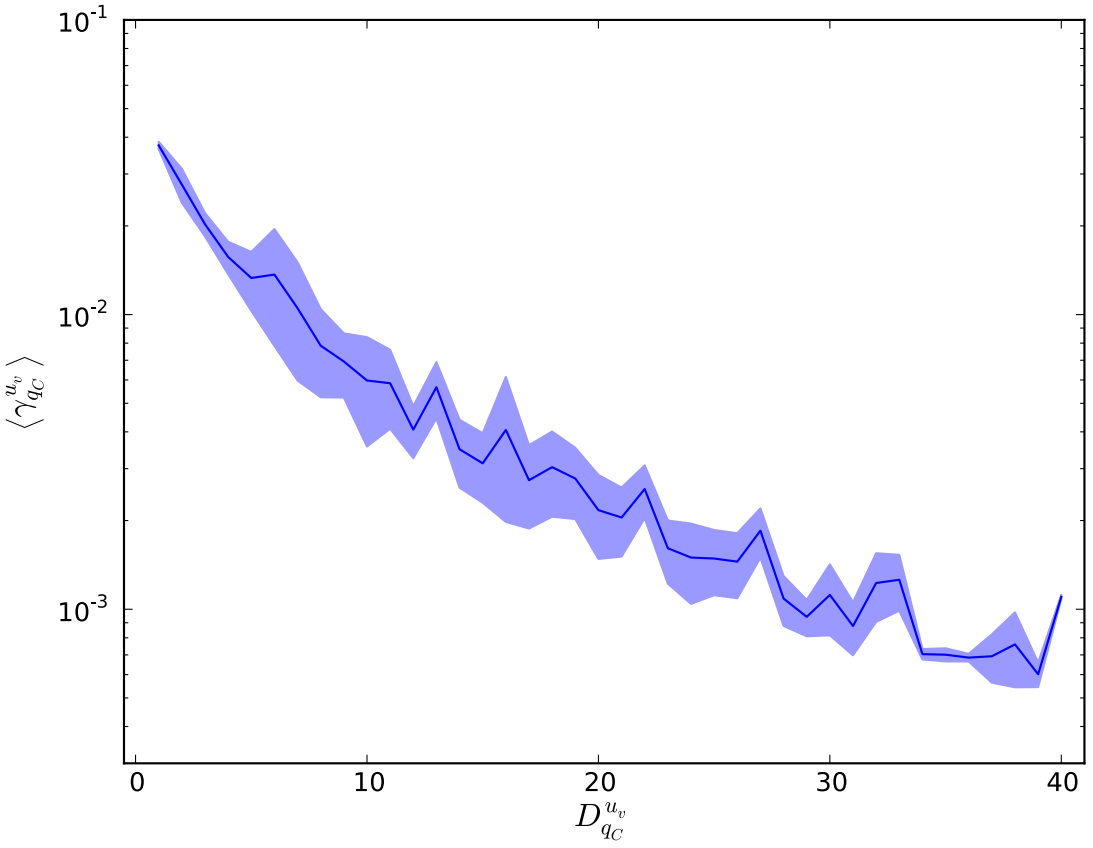


Figure 5.12: The average uncertainty over creator reliability against the amount of exploited provenance

Measure	Pearson Correlation	p-value
$\gamma_{Q_V}(d^v)$ and $D_{Q_V}(d^v)$	$\sim -98\%$	$\sim 5.47 \times 10^{-07}$
$\gamma_{Q_C}(u^v)$ and $D_{Q_C}(u^v)$	$\sim -97\%$	$\sim 5.63 \times 10^{-23}$

Table 5.6: Pearson correlation coefficients and p-value for the correlation between the uncertainty over computed validity estimate and creator reliability and the amount of provenance exploited to compute them.

5.4.4 Evolution of Sample Quality Measures

In this section, we show how some of the quality measures introduced in Section 5.1 evolve through the execution of CollabMap.

We first start with TERMINATION measure for a data product. Figure 5.13 illustrates the evolution of TERMINATION measure over the period of CollabMap execution. The x-axis represents the execution day and the y-axis represents the total number of data product that a decision on their termination was made at the end of a specific execution day. The blue solid line represents those data products that TERMINATION measure instructs CollabMap to terminate and no further verification is required. The red dashed

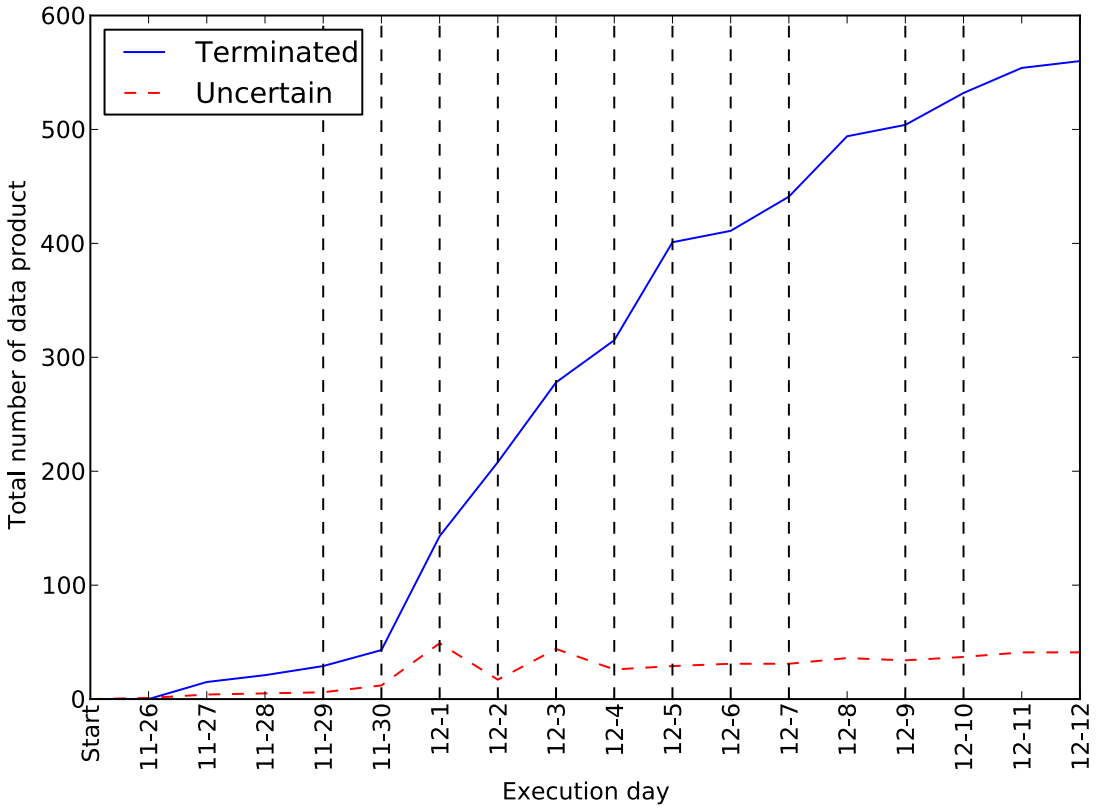


Figure 5.13: Evolution of termination measure over the period of CollabMap execution.

line represents those data products that TERMINATION measure instructs CollabMap to continue recruiting workers.

Note that in Section 5.4.1, we mention that we released 2,700 HITs to AMT. However, we decided to release HITs in batches rather than releasing all HITs at the start of CollabMap execution. The dashed vertical lines on Figure 5.13 show the time at which a new batch of HITs was released. As such, through the execution of CollabMap, there were periodically new tasks (i.e. new data products to be created and scored by workers). Had we released all 2,700 HITs at the start, we would have a different plot than Figure 5.13 in which the red dashed line (representing the uncertain data products) would have been higher than the current one and decreased the slope as CollabMap continued its execution. Nevertheless, this figure shows the evolution of TERMINATION through the execution of CollabMap.

Following, we look at the worker's reliability measure and how it evolves through the execution of CollabMap. Figure 5.14 shows the evolution of this measure for a sample worker identified as worker 442. The x-axis represents the evolution of the worker (different versions of a worker according to *agent pattern*) and the y-axis represents worker's reliability as the worker continued collaborating with the system. We can classify this worker as a reliable worker. The figure shows how this worker's reliability

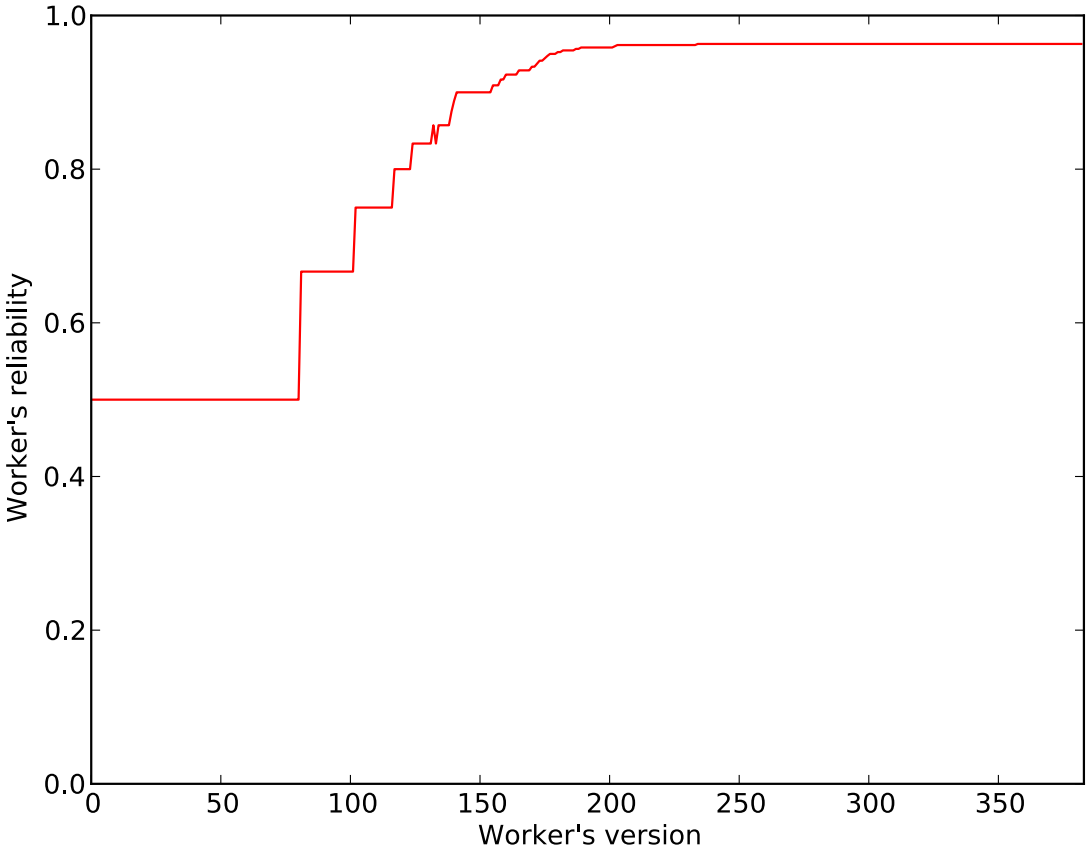


Figure 5.14: Evolution of reliability measure for a sample worker identified as worker 442 over the period of CollabMap execution.

starts from 0.5 (*PEDRA* does not know the reliability of the worker) and evolve to the higher values.

Same as Figure 5.14, Figure 5.15 illustrates the evolution of worker's reliability for a sample worker that we classify the worker as an unreliable worker. The figure shows how this worker's reliability starts from 0.5 and decreases over the course of CollabMap execution.

Figure 5.16 demonstrates the evolution of worker's reliability for a sample worker, known as worker 647. This figure shows an example where the worker cannot be classified as either reliable nor unreliable. As Figure 5.16 shows at starts the worker's reliability decreases, however, after sometime the reliability started to increase and then decreases again.

5.5 Summary

We started this chapter by identifying three common issues in crowdsourcing applications. In order to solve these issues, we put forth a quality assessment approach

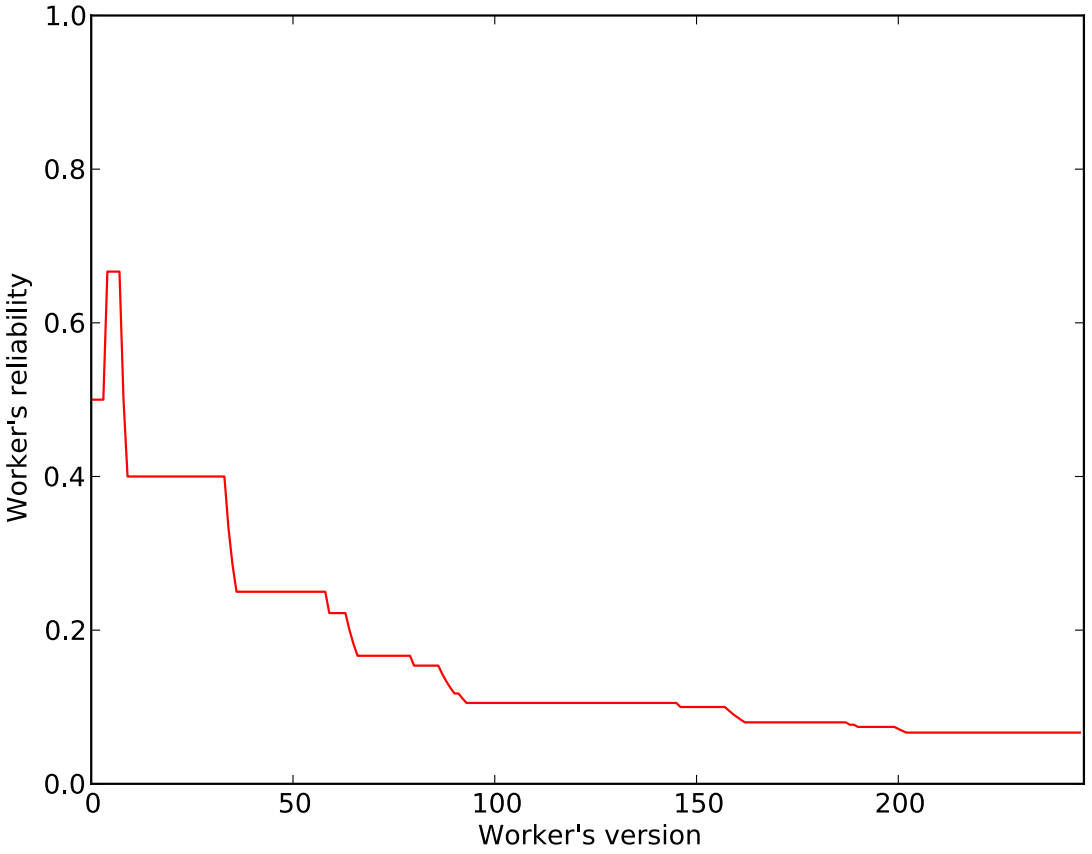


Figure 5.15: Evolution of reliability measure for a sample worker identified as worker 425 over the period of CollabMap execution.

(*PEDRA*) that exploits the provenance of data shaped by a crowdsourcing application to compute a set of quality measures. These measures assist the application to make quality-based decisions such as accepting data of high quality and discarding those with low quality.

More specifically, the first identified issue is about assessing the quality of data products generated in a crowdsourcing application. This is an important issue because knowing about the quality of data products, the application can filter out data products with high quality and discard those of low quality. In order to address this issue, *PEDRA* exploits the provenance of data shaped by the provenance patterns to compute a set of quality measures (such as VALIDITY ESTIMATE) for each data product that assist the crowdsourcing application to either accept or discard a data product.

The second issue is about assessing the reliability of workers. This is an important issue because knowing about the reliability of workers, the application can choose the most reliable workers, schedule its works and etc. In order to address this issue, *PEDRA* first distinguishes between two types of workers: creators who create data products, and scorers who score (verify) the generated data products. Then, *PEDRA* exploits the provenance of data shaped by the provenance patterns to compute a set of quality

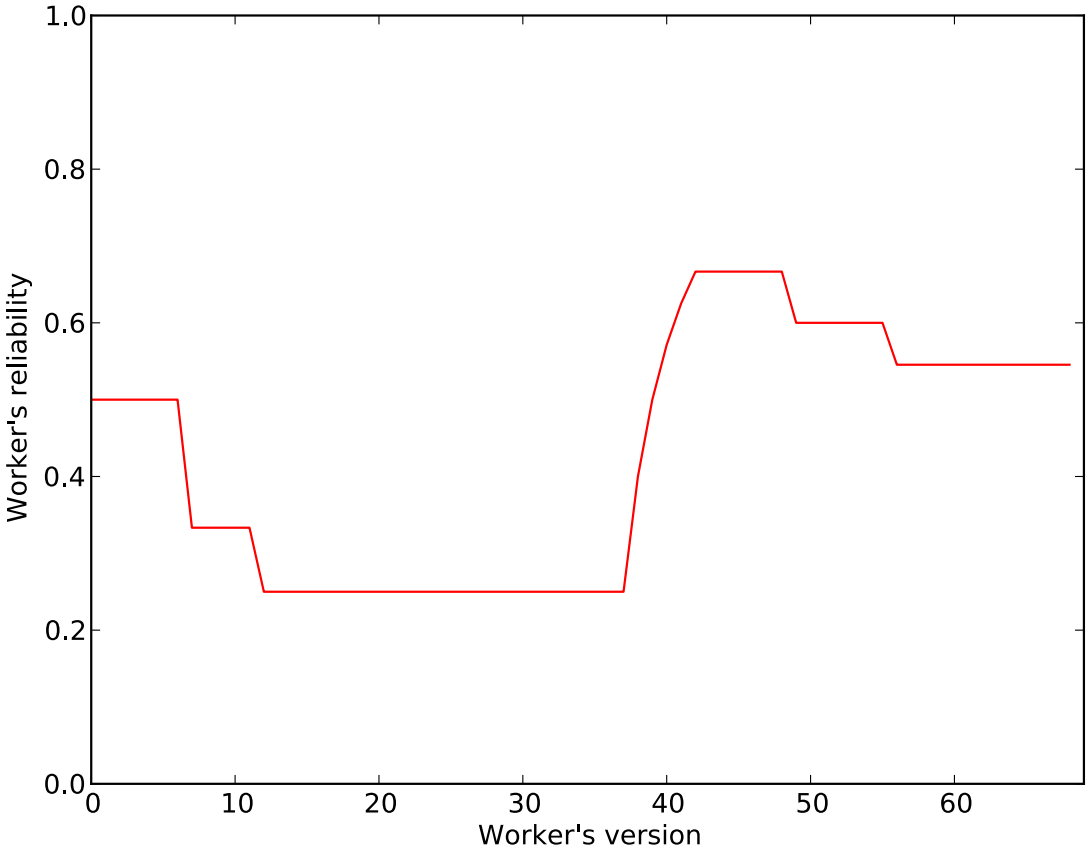


Figure 5.16: Evolution of reliability measure for a sample worker identified as worker 647 over the period of CollabMap execution.

measures for both creators and scorers (such as CREATOR RELIABILITY and SCORER RELIABILITY).

The third issue is about terminating a task with certainty. This is also an important issue because knowing when to terminate a task allows the application to manage its resources (such as workers, money, and time) efficiently. In order to address this issue, *PEDRA* exploits the provenance of data of scorers and data products to compute VALIDITY RATING, and derived from that TERMINATION that instructs the application if further collaboration is required.

Following on from this, we then demonstrated how *PEDRA* works in practice in three ways. First, we showed how it can be used alongside CollabMap, an exemplar FFV-based crowdsourcing application, to assess the quality of data.

Second, we compared the accuracy of *PEDRA* with the current quality model employed in CollabMap (and many other crowdsourcing applications). We showed that *PEDRA* outperforms the current quality assessment approach employed in CollabMap, by at least 6.5%.

And third, we showed that there is an inverse correlation between the amount of exploited provenance to compute a quality measure and the uncertainty over the computed measure; in that, the more provenance exploited, the less uncertainty over the computed measure. Based on this, we conclude that provenance has two main utilities in crowdsourcing:

1. *PEDRA* does not rely on the data model chosen by the crowdsourcing application. As long as the crowdsourcing application generates its provenance shaped by the provenance patterns, *PEDRA* exploits such shaped provenance to compute the described quality measures.

2. The amount of provenance exploited by *PEDRA* to compute quality measures (such as VALIDITY ESTIMATE and CREATOR RELIABILITY) can be quantified by looking at the *revision pattern* (refer to Section 3.3.3) and *agent pattern* (refer to Section 3.3.3) and counting the total number of versions for a data product or a worker. We further showed that uncertainty over a computed measure decreases as more provenance is exploited by *PEDRA*.

Comparing the accuracy of *PEDRA* to the quality model employed in CollabMap (which is a popular approach in crowdsourcing), the accuracy of *PEDRA* outperforms by at least 6.5% and we showed the result is statistically significant. We extrapolate this is because of two reasons. First, the quality model employed in CollabMap is known as *Majority Voting*. CollabMap utilises this approach without exploiting provenance whereas *PEDRA* exploits the provenance of data shaped by provenance patterns. Second, *Majority Voting* does not consider the reliability of workers in contrast to *PEDRA* that distinguishes between two types of workers; and computes different quality measures for them and considers those measures in quality assessment.

It is noteworthy to mention that the proposed quality assessment approach in this chapter (*PEDRA*) is applicable beyond CollabMap, as *PEDRA* does not rely on the data model of the crowdsourcing application. The approach works for any FFV-based crowdsourcing application, as outlined in our PhD objectives in Section 1.7, provided that the application records provenance of data and shape this provenance according to the provenance patterns presented in Chapter 3.

In the next chapter, we envision situations where *PEDRA* is required to be utilised while the crowdsourcing application is executing live. The online application of *PEDRA* allows the application to make online quality-based decisions such as decisions on the validity of data or decisions on the reliability of workers.

Chapter 6

PEDRA-O: Provenance-based online decision making with *PEDRA*

Chapter 5 presented a quality assessment approach (*PEDRA*) that exploits the provenance of data generated in a crowdsourcing application to assess the quality of data and reliability of workers. Chapter 5 also demonstrated the application of *PEDRA* after the crowdsourcing application finishes its execution and the data collection phase is finished.

Chapter 2 introduces crowdsourcing as a medium through which data can be collected fast and cheap, and it is useful provided that the quality of generated data is assessed. A crowdsourcing application can make a set of quality-based decisions to improve its performance or change its normal behaviour while it is executing live. For example, if a worker keeps creating invalid data products, the application can be notified of this action and does not allocate more tasks to the worker (the application changes its normal behaviour given the normal behaviour of a crowdsourcing application is to allocate tasks to all workers). The crowdsourcing application can also be instructed on when to terminate a task. This is an important decision because it has a direct influence on worker's fee, given the crowdsourcing application is required to pay workers (the application improves its performance by making dynamic task termination decision).

The quality measures computed by *PEDRA* in Chapter 5 facilitate such online quality-based decisions by the crowdsourcing application. In this chapter, we envision *PEDRA* as a data quality assessment service to multiple, different FFV-based crowdsourcing applications. In order to regulate the interaction between *PEDRA* and the crowdsourcing applications, we envision an online architecture by proposing an online contract, *PEDRA-O*[1], specifying how and when the crowdsourcing application is expected to

[1]The O in the name refers to the online application of *PEDRA*.

submit the provenance of data. In return, the contract states how the quality measures computed by the service are returned to the crowdsourcing application.

More specifically, and recalling from Chapter 1, the contribution of this chapter is as follows:

Contribution We present an architecture for any FFV-based crowdsourcing application with the purpose of assisting the application to improve its behaviour and assess the quality of data while it is executing live. Furthermore, we perform a detailed evaluation to assess the benefits and costs of such architecture on the crowdsourcing application. The evaluation is the first of its kind to integrate a crowdsourcing application and a quality assessment approach in a real world setting where not only the accuracy of the quality assessment approach but also the performance of the approach is measured.

In the following sections, we elaborate more on the online use of provenance in crowdsourcing applications by putting forth an online contract that regulates the interaction between the application and *PEDRA*. More specifically:

- Section 6.1 introduces an architecture and a contract that regulates the interaction between a crowdsourcing application and *PEDRA*.
- Section 6.2 discusses the implementation of the quality assessment service.
- Section 6.3 gives an empirical evaluation of *PEDRA-O*.
- Section 6.4 summarises the chapter.

6.1 Online Contract

In Chapter 5 we devised an architecture that has two components: (1) a FFV-based crowdsourcing engine that recruits workers to create and verify data and (2) a data quality assessment approach known as *PEDRA* that assesses the quality of data and reliability of workers.

The crowdsourcing engine submits provenance graphs for analysis to *PEDRA* which computes quality measures by exploiting the submitted provenance. Once measures are computed, they are passed back to the crowdsourcing engine which utilises the measures to make online decisions and improve its performance while executing.

By so doing, we decouple the data quality assessment approach (*PEDRA*) from the crowdsourcing engine. It can now be deployed as a data quality assessment service to multiple, different FFV-based crowdsourcing engines. The interactions between the two components should adhere to the workflow in Figure 6.1, which is detailed as follows.

1. When a task is completed by a worker, the crowdsourcing engine generates provenance for the task according to the provenance patterns (Chapter 3).
2. The provenance graph is submitted to the service.
3. The service computes quality measures over the provenance graph (such as those introduced in Chapter 5) and annotates the original graph's elements with the computed measures (see Section 5.1 on how quality measures are computed and how PROV elements are annotated with the quality measures).
4. The annotated provenance graph and computed quality measures are persisted on the *PEDRA* side.
5. The computed quality measures are passed to the crowdsourcing engine.
6. The crowdsourcing engine uses the computed measures to make application-specific decisions such as whether to accept or reject the completed task.
7. At this point, if the crowdsourcing engine has more tasks to submit to PEDRA, the workflow repeats (by going back to the first item); otherwise, the crowdsourcing engine finishes its execution.

The decoupling of the quality assessment approach allows reusing the approach for more than one FFV-based crowdsourcing application without the need to re-implement it. However, it also inevitably introduces communication and storage overheads between the two components. In order to minimise its impacts, we introduce the following requirements for the crowdsourcing engine:

Requirement 1

The chosen quality measures (e.g. VALIDITY ESTIMATE, CREATOR RELIABILITY, and SCORER RELIABILITY) must be incrementally computable. Specifically, the measures of a version of an element (e^v) in the provenance graph (i.e. entities, activities, and agents, such as a data product or a worker) should be computable from the measures of its previous version (e^{v-1}). In other words, there must exist a function f such that

$$\text{Measure}(e^v) = \begin{cases} f(G) & \text{if } v = 0 \\ f\left(G, \text{Measure}(e^{v-1})\right) & \text{if } v \neq 0 \end{cases} \tag{6.1}$$

According to Equation 6.1, if an element does not have a previous version (i.e. e^v where $v = 0$), the crowdsourcing engine is required to send the whole provenance graph. In the submitted provenance graph, there is no previously computed measure as it is the first submission.

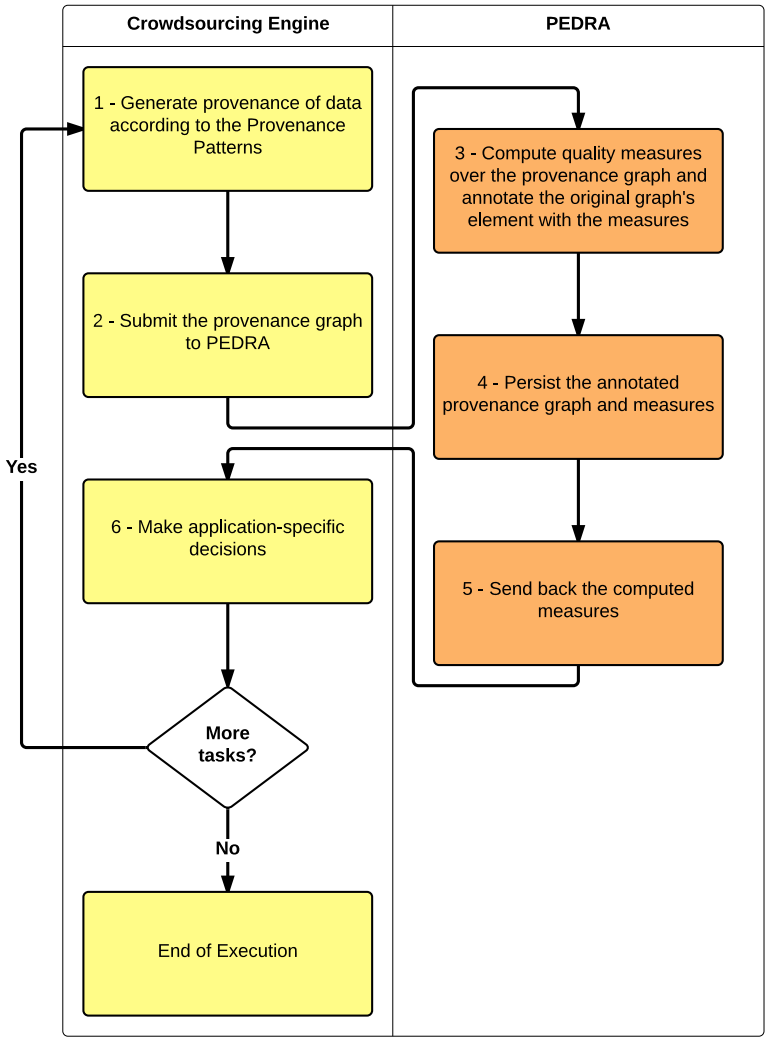


Figure 6.1: The workflow for PEDRA-O detailing the communications between the crowdsourcing engine and PEDRA. If the crowdsourcing engine has more tasks to submit to PEDRA, the workflow repeats. Otherwise, the crowdsourcing engine finishes its execution.

However, if the provenance element has a previous version (i.e. e^v where $v \neq 0$), the crowdsourcing engine is required to send the provenance of data of what happened between e^{v-1} and e^v. For instance, if $v = 2$ for a provenance element (i.e. e^{v_2}), the crowdsourcing engine is required to send the provenance of data of what happened between e^{v_1} and e^{v_2} and not the whole provenance graph, i.e. from e^{v_0} to e^{v_2}.

This requirement makes it sufficient for the crowdsourcing engine to send only the provenance of activities that have happened since the previous submission, instead of sending the whole provenance graph since its initialization to *PEDRA*. G in Equation 6.1 denotes a provenance graph. Note that, the provenance graph does not contain all the provenance information from the initialization of the task, rather it contains all the new information that was not sent from the previous submission. By including the previously

computed measures, the crowdsourcing application is not required to submit the whole provenance graph of a task as such we require less information in G.

It is noteworthy to mention that the quality metrics that were introduced in Chapter 5 are all compatible with this requirement. For instance, in Section 5.2.1, Rule 5.21 defines how POSITIVE SCORES of a version of a data product is computed in terms of its previous version. According to this rule, Equation 6.2 defines how POSITIVE SCORES of a version of a data product is a function of the POSITIVE SCORES of the previous version.

$$\mathrm{S}^{+}(d^{v}) = \begin{cases} f\left(G\right)) & \text{if } v = 0 \\ f\left(\mathrm{G}, \mathrm{S}^{+}(d^{v-1})\right) & \text{if } v \neq 0 \end{cases} \tag{6.2}$$

Requirement 2

The crowdsourcing engine is permitted to send multiple provenance graphs in parallel to *PEDRA* provided that they all document different tasks.

In addition, when the crowdsourcing engine submits a provenance graph of a task, it is required not to change the current state associated with anything referred to by the graph. It must wait for the computed measures to be returned by *PEDRA* before carrying on with the same task. For example, after a contributor scores a data product and the corresponding provenance is submitted, the application must not recruit more contributors for the same data product or assign another task to the same contributor until all the measures are computed and returned for that data product.

It should be noted, however, that a FFV-based crowdsourcing application typically has numerous tasks and workers. Hence, it can still recruit other workers for other tasks without delay and send the associated provenance to *PEDRA* as soon as those tasks finish.

We call the workflow and the two requirements above the *Online Contract* for *PEDRA* (OCP). FFV-based crowdsourcing applications, adhering to OCP, will be able to make live decisions based on the quality measures produced by *PEDRA*.

PEDRA is required to persist all the computed measures and any extra data that was required in the computation. In this way, we support: (1) audit, (2) re-computation of the quality measures, and (3) showing why a decision was made. This allows a thorough and accurate assessment of the approach (for example to check the computation of past measures against rules).

In the next section, we analyse benefits and the communication overheads introduced by OCP.

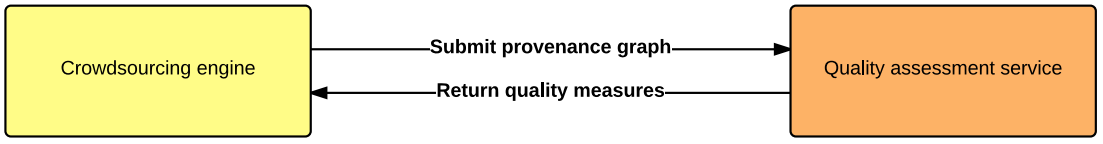


Figure 6.2: Interaction between a crowdsourcing engine and the quality assessment service.

Title	Compute quality measures
Description	This is the entry point to the quality assessment service, the API accepts the provenance graph submitted by the client (i.e. a crowdsourcing engine)
URL	https://waisvm-ask2g10.ecs.soton.ac.uk/rest/OAPS/entry
Method	POST
Data Params	application/json (PROV-JSON) (a provenance graph as specified in Section 6.1)
Return	application/json (PROV-JSON) (all the computed quality measures)

Table 6.1: RESTful API endpoint for OCP.

6.2 Implementation of the Quality Assessment Service

As stated in Section 6.1, the crowdsourcing engine submits the provenance of data to the quality assessment service. The service computes the quality measures and passes back the measures to the crowdsourcing application, as depicted in Figure 6.2.

In order to facilitate the interaction between a crowdsourcing engine and the quality assessment service, the service offers a RESTful API. Table 6.1 outlines the API endpoint.

Table 6.1 specifies the RESTful API endpoint for OCP. It uses the HTTP method POST. It accepts a provenance graph as specified in Section 6.1 in PROV-JSON representation (as the data params) and returns back all the computed measures in PROV-JSON representation. The PROV-JSON representation (Huynh et al., 2013b) is a W3C member submission that specifies a JSON representation for the PROV-DM (Moreau et al., 2013a). It supports fast data look-up and is particularly suitable for interchanging PROV documents between web services and clients (Huynh et al., 2013b).

Figure 6.3 demonstrates a sample provenance graph submitted to the quality assessment service. In this provenance graph, a data product, Building14379.4 was scored by User442.261. The provenance graph is shaped according to the *score pattern* by filling the placeholders such as data product and scorer. This is a sample provenance graph submitted to the quality assessment service by a crowdsourcing application in PROV-JSON data format.

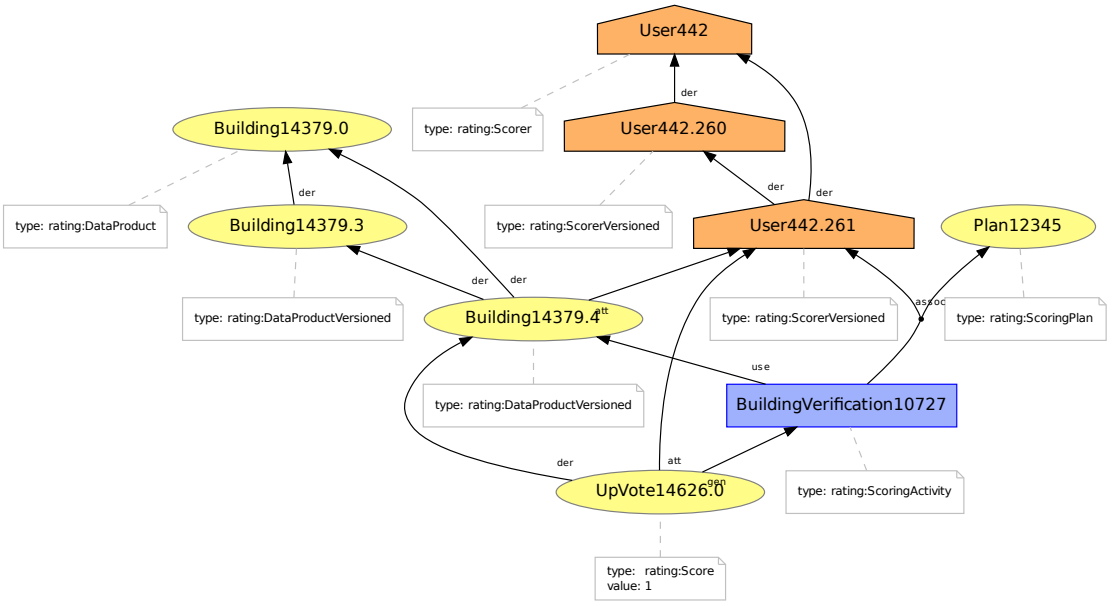


Figure 6.3: A sample provenance graph submitted to the quality assessment service.

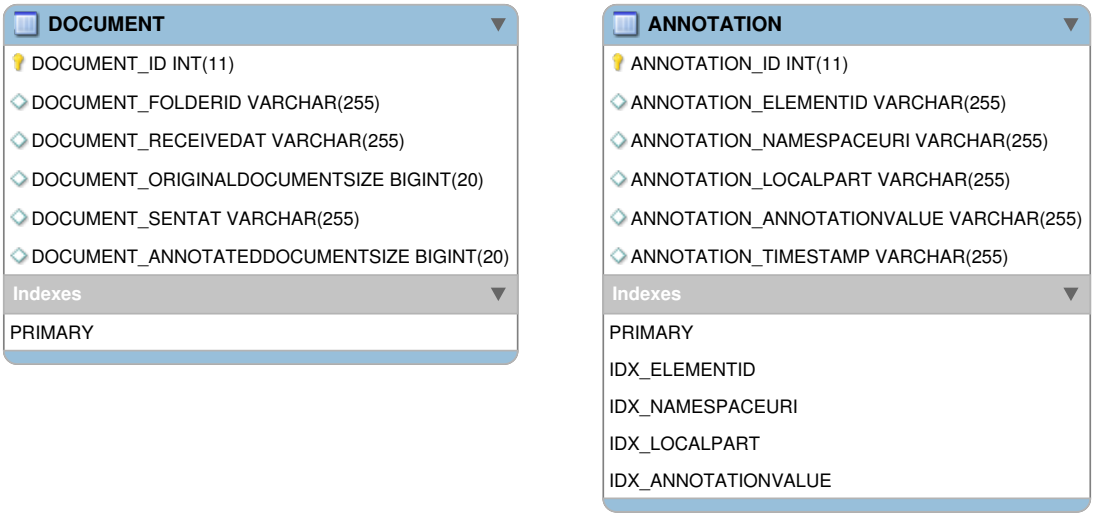


Figure 6.4: SQL schema view.

After annotation propagation and computation, a set of quality measures are computed by *PEDRA* which should returned back to the crowdsourcing application in PROV-JSON representation.

After annotations (i.e. quality measures) are computed by *PEDRA*, they should be maintained in a storage for future use. We decided to use a SQL database with two tables: DOCUMENT and ANNOTATION; as depicted in Figure 6.4.

The DOCUMENT table maintains information such as when a provenance graph was received by the service and when the computed measures passed back to the crowdsourcing engine. It also stores a reference to where the submitted provenance graph

is maintained on the disk (note that every provenance graph received by the service is maintained in disk).

The ANNOTATION table maintains information about the computed measures. It stores the namespace, key and value of the computed attributes, and the time an attribute was stored.

6.3 Empirical Study

In this section, we present the results from the *online* application of *PEDRA* for quality assessment and task termination in CollabMap. We call this deployment *CollabMap-PEDRA-O*. First, the methodology used to perform the experiment is described in Section 6.3.1. Then, we measure the workers' fees saved in *CollabMap-PEDRA-O* as compared to that incurred in the *offline* application of *PEDRA*, called *CollabMap-PEDRA* (described in Section 5.4) in Section 6.3.2. We also measure the overheads introduced by *PEDRA-O* in terms of communication, storage, and delay of task allocation and analyse them in Section 6.3.3.

6.3.1 Experiment Methodology

For the purpose of this experiment, we decided to run both *Majority Voting* and *PEDRA* in parallel by retrofitting them into CollabMap. In this case, one shared dataset is analysed by *PEDRA* and *Majority Voting*. Then, it is possible to evaluate the performance of each quality assessment approach with the same dataset. The alternative, that is running CollabMap to produce two different datasets, is not acceptable as one could argue, for example, data products of one dataset are more difficult than the other dataset and this is why the performance of each approach differs. Furthermore, there is no guarantee that the same worker would collaborate in both roll-outs of CollabMap.

As such, in this setting, both approaches execute in parallel. Earlier in Section 5.4, we demonstrated that the accuracy of *CollabMap-PEDRA* outperforms the accuracy of *CollabMap-MajorityVoting*. In this section, and working upon our findings, we analyse the benefits and costs of the online application of *PEDRA* compared to its offline application.

6.3.2 The Benefit of OCP

As all CollabMap HITs have the same cost, comparing the cost of a task incurred by the online and offline applications of *PEDRA* is equivalent to comparing the number of HITs required by the two approaches for completing the task. Hence, in this analysis, we

count the average number of HITs that *CollabMap-PEDRA-O* and *CollabMap-PEDRA* required before a task was terminated in our experiment. Our hypothesis is as follows:

Hypothesis 3 On average, *CollabMap-PEDRA-O* requires fewer number of HITs to complete a task than *CollabMap-PEDRA*.

Method We recorded the total number of HITs each approach requires to complete a task. Then, at the end of the experiment, we carry out an analysis that contains three parts: (1) we analyse the average number of HITs each approach (*CollabMap-PEDRA-O* and *CollabMap-PEDRA*) requires per task to terminate a data product (Refer to Section 5.1.3 for TERMINATION measure), (2) we analyse the total number of terminated data products by each approach through the execution time, and (3) we analyse the cumulative number of HITs over the course of execution. These three analyses assist us to study the benefit of OCP.

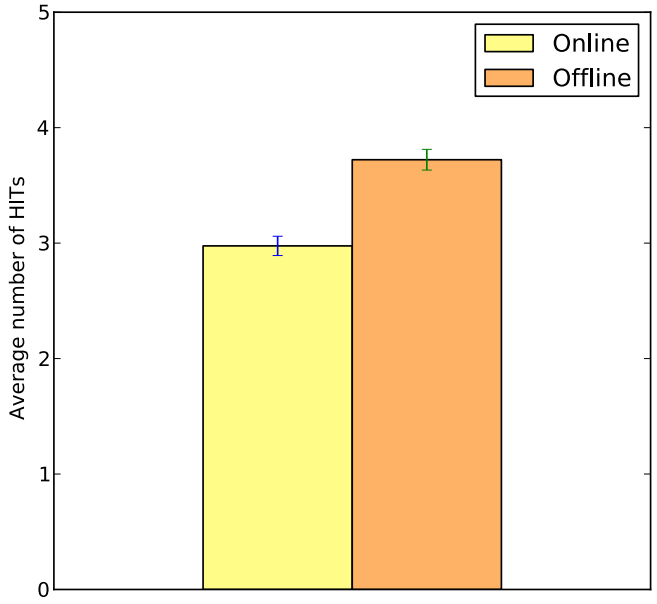
Furthermore, in order to ensure the results are statistically significant, we use the standard paired t-test with the null hypothesis that the difference between number of required HITs for *CollabMap-PEDRA-O* and *CollabMap-PEDRA* comes from a normal distribution with mean equal to zero (i.e. the difference between them is not statistically significant).

Analysis Figure 6.5 demonstrates average number of HITs for *CollabMap-PEDRA-O* and *CollabMap-PEDRA*. The first yellow bar represents *CollabMap-PEDRA-O* (online) and the second orange bar represents *CollabMap-PEDRA* (offline). The y-axis represents average number of HITs per task. On each bar, standard error is shown. This figure shows that, on average, *CollabMap-PEDRA-O* requires fewer number of HITs per task to terminate a data product compared to *CollabMap-PEDRA*.

On average, we found that *CollabMap-PEDRA-O* requires 2.9 HITs where *CollabMap-PEDRA* requires 3.7 HITs over 300 tasks. Hence, *CollabMap-PEDRA-O* required 20% less HITs per task than its offline counterpart.

Figure 6.6 demonstrates the total number of terminated data products by both *CollabMap-PEDRA-O* and *CollabMap-PEDRA* through the execution period of CollabMap. CollabMap was executing from 2014-11-25 till 2014-12-11. We counted the total number of terminated data products at the end of each day; the x-axis represents each execution day and the y-axis represents the cumulative number of terminated data products. The red solid line in Figure 6.6 represents *CollabMap-PEDRA-O* (online) and the blue dashed line represents *CollabMap-PEDRA* (offline).

In Figure 6.5, we showed that *CollabMap-PEDRA-O* requires 20% less HITs than *CollabMap-PEDRA*. This means that we can expect *CollabMap-PEDRA-O* to terminate data products faster than *CollabMap-PEDRA*. This can be approved by Figure 6.6



	Average number of HITs
Online	2.9
Offline	3.7

Figure 6.5: On average, CollabMap-PEDRA-O requires 20% less HITs per task compared to CollabMap-PEDRA.

where the online line (red) stays above the offline line (blue), which means *CollabMap-PEDRA-O* terminates more data products (i.e. *CollabMap-PEDRA-O* is faster in termination). Given that both approaches have access to the same resources, we conclude that *CollabMap-PEDRA-O* requires fewer number of HITs to terminate a data product.

Figure 6.7 demonstrates the cumulative number of HITs that *CollabMap-PEDRA-O* and *CollabMap-PEDRA* require to terminate data products (i.e. no more scorer is required to score the data product). The x-axis represents the cumulative number of data products sorted by their termination time and the y-axis represents the cumulative number of HITs. The red solid line represents *CollabMap-PEDRA-O* (online) and the blue dashed line represents *CollabMap-PEDRA* (offline).

Figure 6.7 shows that the online line (red) stays bellow the offline line (blue), which means given the whole execution time of CollabMap, *CollabMap-PEDRA-O* requires less HITs. More specifically, the ratio between total number of HITs required by *CollabMap-PEDRA-O* and *CollabMap-PEDRA* starts with $\sim$ 0.33 at the start of the execution, continuously moving towards $\sim$ 0.50, and at the end, the ratio was $\sim$ 0.80, which confirms that *CollabMap-PEDRA-O* requires 20% less HITs compared to *CollabMap-PEDRA*.

Figure 6.8 represents the cumulative number of *extra HITs* that *CollabMap-PEDRA-O* and *CollabMap-PEDRA* require as compared to each other the total number of data products. The x-axis represents cumulative number of data products and the y-axis represents cumulative number of extra HITs an approach require to terminate a data product. The red solid line represents *CollabMap-PEDRA-O* (online) and the blue dashed line represents *CollabMap-PEDRA* (offline).

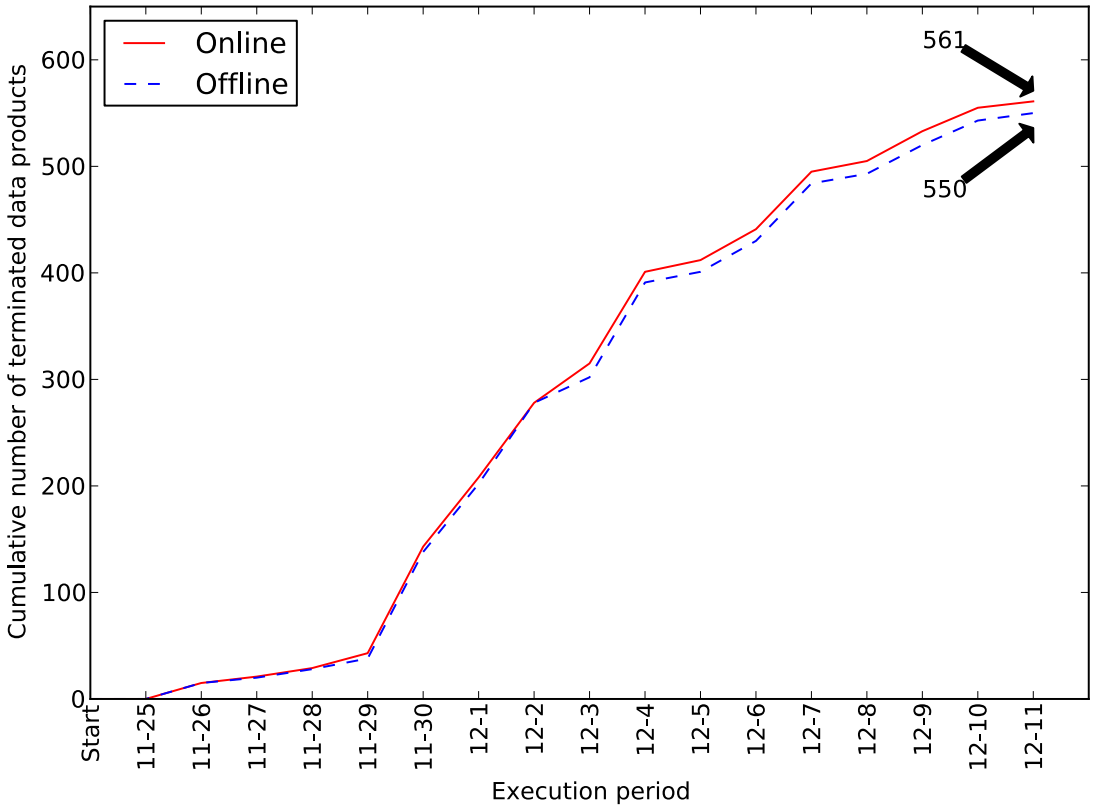


Figure 6.6: Number of terminated data products by *CollabMap-PEDRA-O* and *CollabMap-PEDRA* through the execution period of CollabMap.

This figure shows that the offline line (blue) stays well above the online line (red), which means *CollabMap-PEDRA* requires extra HITs compared to *CollabMap-PEDRA-O*. More specifically, and as shown earlier in Figure 6.5, *CollabMap-PEDRA* requires 20% extra HITs compared to *CollabMap-PEDRA-O* at the end of the execution.

The paired t-test confirms the result provided above with a very small p-value, 2.37×10^{-14}, allowing us to reject the null hypothesis with over 99% confidence. Given this, we can conclude that *PEDRA-O* helped CollabMap save 20% in worker fees while maintaining the same or higher level of accuracy as the offline application (shown earlier in Section 5.4.2).

The analysis offered in this section shows the benefit of OCP in saving workers' fee in a crowdsourcing application. PEDRA evaluates the quality of data after the execution of the crowdsourcing application (e.g. CollabMap) is finished. As such, the decision on terminating a task is not made by PEDRA as PEDRA is operating offline (after the execution of CollabMap). The decision is made by Majority Voting in CollabMap. So, for the purpose of termination of tasks, CollabMap is the same as CollabMap-PEDRA. What CollabMap-PEDRA gives extra is the rating of buildings/routes.

On the other hand, PEDRA-O, is an algorithm that allows a crowdsourcing application (e.g. CollabMap) to evaluate the quality of data while it is executing live. Furthermore,

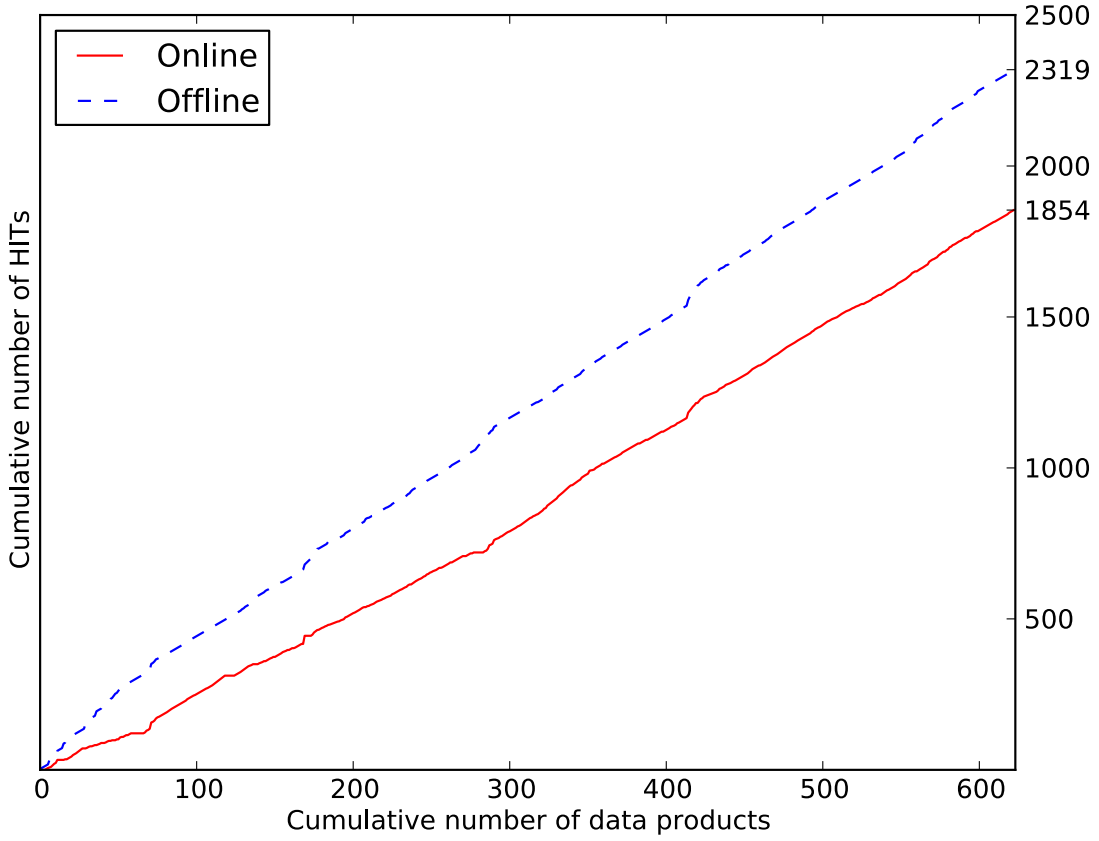


Figure 6.7: Cumulative number of HITs that are required to terminate a data product by CollabMap-PEDRA-O and CollabMap-PEDRA.

the computed ratings instruct CollabMap on when to terminate a task. In the evaluation section, we showed that using PEDRA-O, we require 20% less jobs.

6.3.3 Overheads of OCP

We now study the overheads introduced due to the decoupling of *PEDRA* from the crowdsourcing engine in terms of the extra communication and storage costs, and the delays during which the crowdsourcing engine waits for responses from *PEDRA*. First, we start with looking at how much provenance the crowdsourcing engine is required to submit to *PEDRA* (communication cost analysis).

6.3.3.1 Communication Cost

Method We measure the communication cost in terms of the number of provenance records in each submitted provenance document as time progresses, which include PROV entities, activities, agents, and the relations between them. Based on the provenance patterns in Section 3.3, we compute the upper-bounds for the total number of provenance records generated when a data product is created and when it is scored.

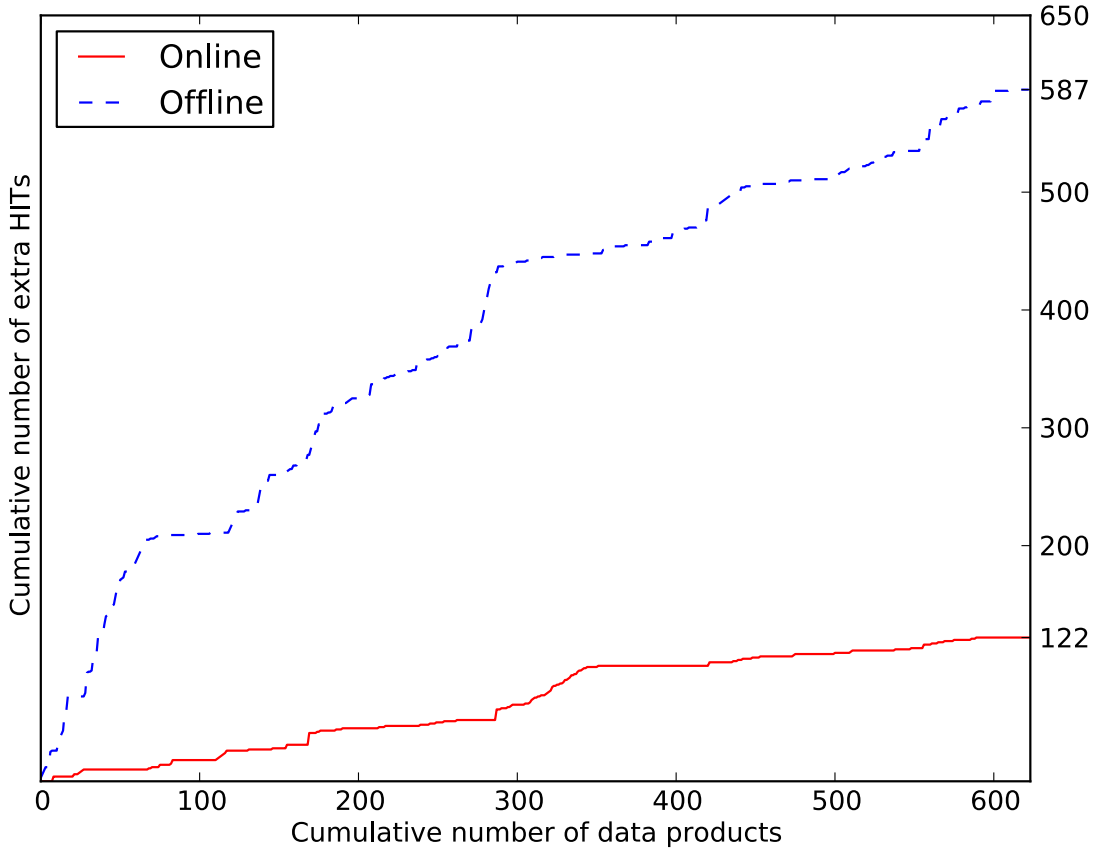


Figure 6.8: Number of extra HITs that *CollabMap-PEDRA-O* and *CollabMap-PEDRA* required as compared to each other.

Analysis Figure 6.9 shows the communication cost between a FFV-based crowdsourcing engine (CollabMap) and a quality assessment approach (*PEDRA*). The x-axis represents the HIT number and the y-axis represents the number of provenance records for a given HIT.

According to Figure 6.9, there was a provenance graph submitted by CollabMap that contained 14 provenance records. On the other hand, there was a provenance graph submitted by CollabMap that contained 115 provenance records. Looking at the Figure 6.9, we can see that, the number of provenance records in submitted provenance graphs has a wide range of values (from 14 provenance records up to 115 provenance records). Given that all submitted provenance graphs are shaped according to either *create pattern* or *score pattern*, we expected the number of provenance records to be close to each other.

This highly varied communication cost affects the performance of the online application of *PEDRA*; the more provenance records, the more quality measures propagated by *ACF* and computed by *PEDRA* which subsequently means it takes more time for *PEDRA* to return the computed measures to CollabMap. As such, following dives into the details on why the communication cost is so varied through the course of execution of CollabMap in particular, and any FFV-based crowdsourcing application in general.

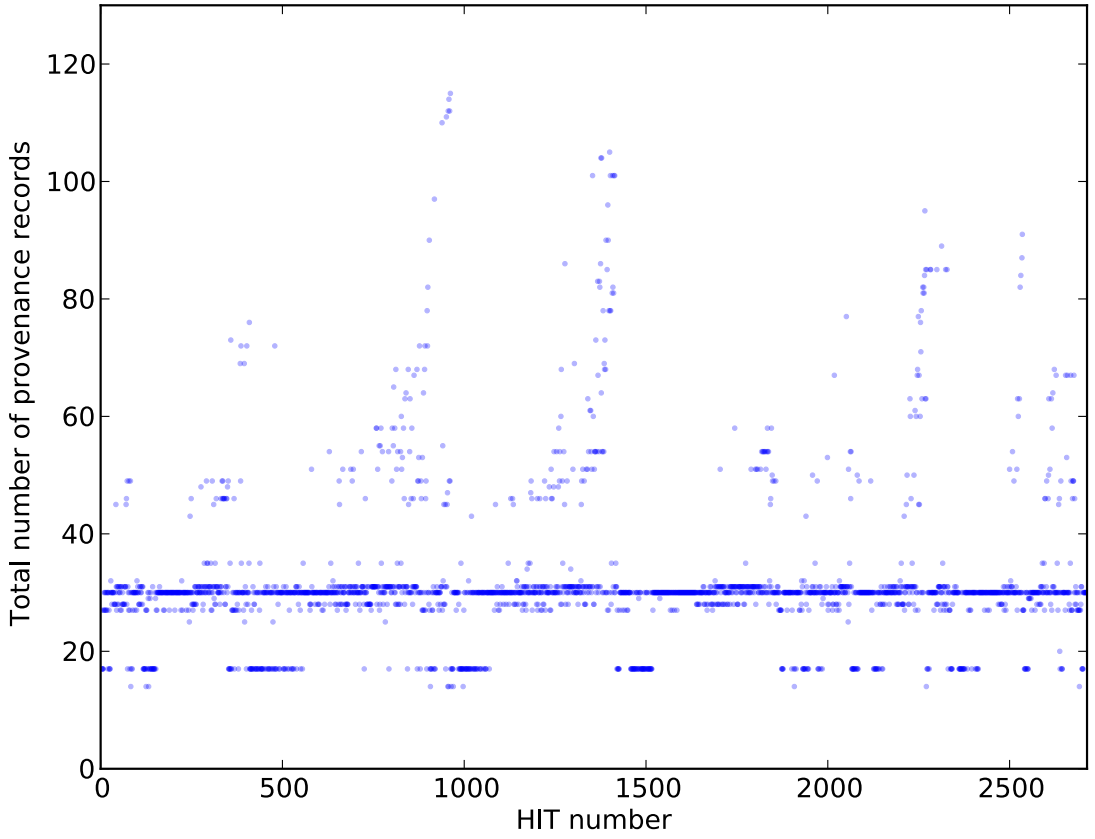


Figure 6.9: The communication cost between a FFV-based crowdsourcing engine and PEDRA.

In order to understand why communication cost varies significantly among the submitted provenance graphs, we analyse communication cost in two cases: (1) when a data product was created and the provenance graph was shaped according to the *create pattern* and (2) when a data product was scored and the provenance graph was shaped according to the *score pattern*.

Figure 6.10 represents the communication cost between CollabMap and *PEDRA* for cases when a data product was created. The provenance graph submitted by CollabMap was shaped according to *create pattern*. In this figure, the x-axis represents the HIT number and the y-axis represents the number of provenance records for a given HIT. According to Figure 6.10, there was a provenance graph submitted by CollabMap that contained 14 provenance records. On the other hand, there was a provenance graph submitted by CollabMap that contained 29 provenance records.

Figure 6.11 represents the communication cost between CollabMap and *PEDRA* for cases when a data product was scored. The provenance graph submitted by CollabMap was shaped according to *score pattern*. In this figure, the x-axis represents the HIT number and the y-axis represents the number of provenance records for a given HIT. According to Figure 6.11, there was a provenance graph submitted by CollabMap that

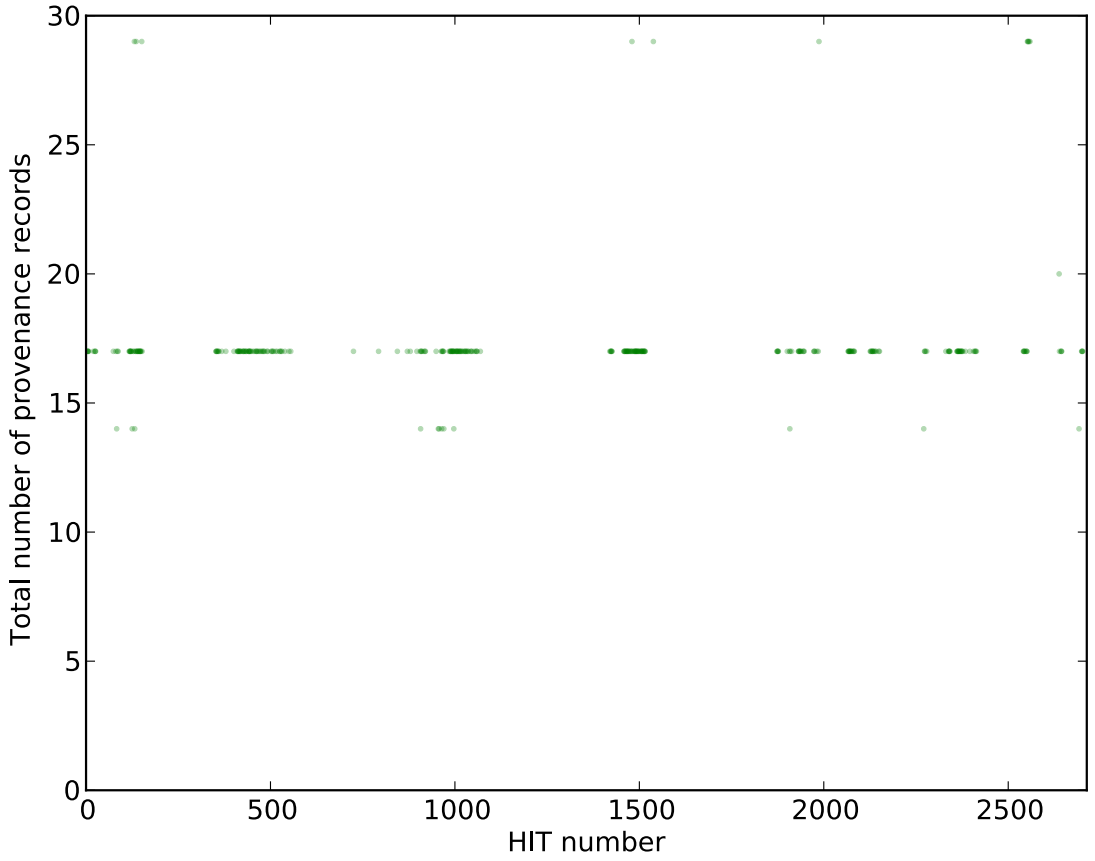


Figure 6.10: The communication cost between a FFV-based crowdsourcing application and *PEDRA* when a data product was created (i.e. the provenance of data was shaped according to *create pattern*).

contained 25 provenance records. On the other hand, there was a provenance graph submitted by CollabMap that contained 115 provenance records.

Looking at *score pattern* (Refer to Section 3.3), a provenance graph shaped by this pattern cannot contain 115 provenance records, as happened in the case of CollabMap. The reasons behind this highly varied number of provenance records in a provenance graph are as follows:

1. The provenance graph submitted by a crowdsourcing engine is about both creating and scoring of a data product; in this case, all four provenance patterns were utilised to shape the provenance data; i.e. when a data product was scored by a scorer, the crowdsourcing engine includes the provenance of the creation of the data product (shaped according to *create pattern*) and the provenance of the scoring of the data product (shaped according to *score pattern*) and then submits the aggregated provenance graph to *PEDRA*.

2. More than one data product may be scored by a scorer, and as such, the crowdsourcing engine may aggregate the provenance of all verifications in one submission.

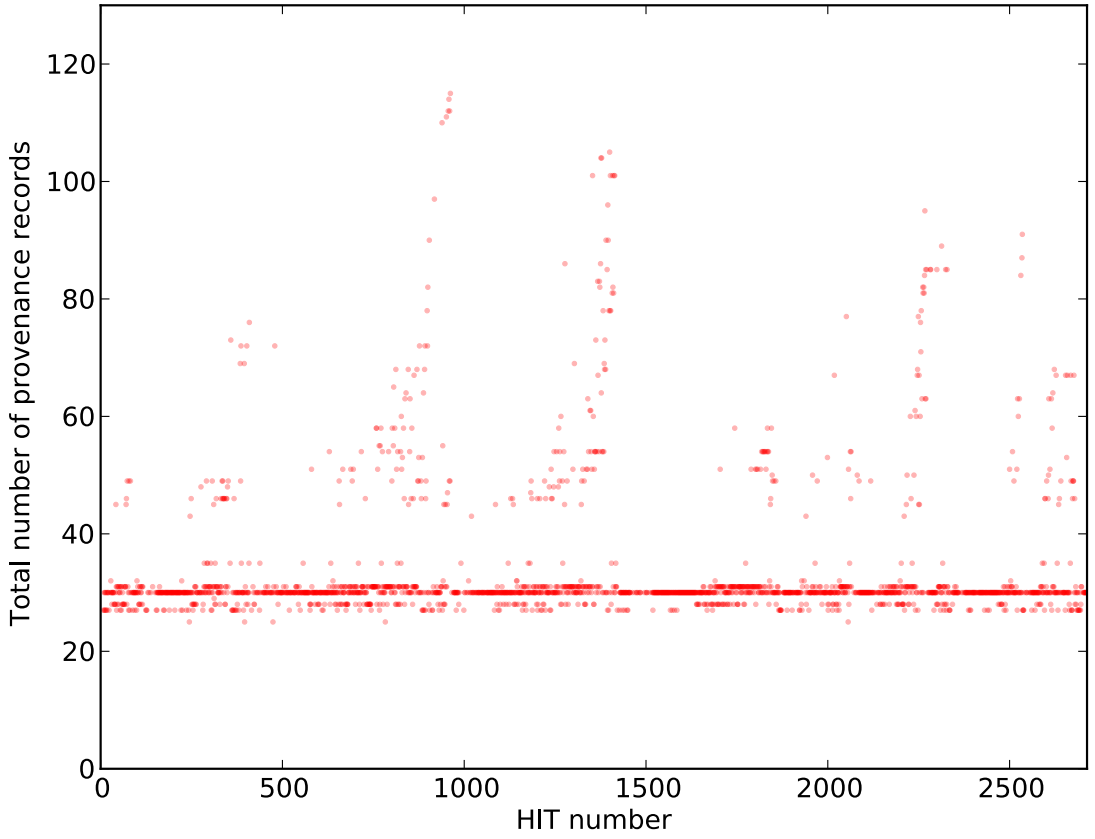


Figure 6.11: The communication cost between a FFV-based crowdsourcing application and *PEDRA* when a data product was scored (i.e. the provenance of data was shaped according to *score pattern*).

In this case, more than one provenance pattern (depends on the number of verified data products) are employed to shape the provenance data. For instance, in the case of CollabMap, the provenance graph that contained 115 provenance records, contained the provenance of creation of 4 data products and the provenance of scoring of all those 4 data products, yielding a high number of provenance records.

3. Even if the provenance of data is shaped according to only one provenance patterns (either *create pattern* or *score pattern*), it may also be shaped according to *revision pattern* (whether the data product was scored before or not) or *agent pattern* (whether the scorer participated before or not). As such, depending on the total number of provenance patterns employed to shape the provenance of data, the total number of provenance records in one submission could differ.

Figure 6.12 shows the total number of provenance patterns that were employed to shape the provenance of data for each HIT. In this figure, the x-axis represents the HIT number, and the y-axis represents the number of employed provenance patterns. This figure shows that in majority of HITs, more than one provenance patterns were employed to shape the provenance of data; in most cases, two or three provenance patterns were

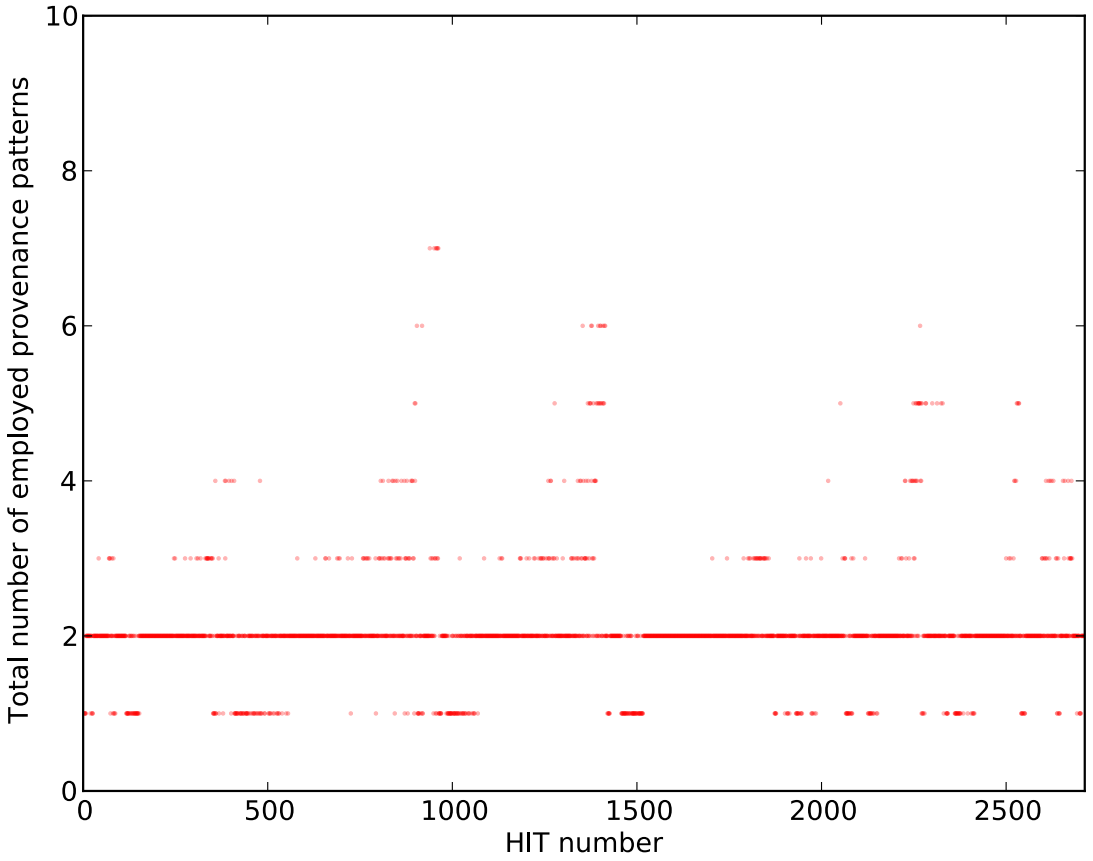


Figure 6.12: Total number of employed provenance patterns to shape provenance of data submitted to PEDRA by CollabMap

employed. This figure confirms our analysis on why total number of provenance records varies highly in submitted provenance graphs.

In summary, according to the OCP, *CollabMap-PEDRA-O* submits provenance records documenting the activities that occurred in a HIT after it finishes to *PEDRA*. Since a HIT may involve more than one provenance patterns, the number of provenance records covering a HIT varies. Even if only one data product or one score was generated in a HIT, the provenance records may also be shaped by *revision pattern* and/or an *agent pattern*, in addition to either a *create pattern* or a *score pattern*. Moreover, the crowdsourcing engine may allow/request multiple data products or scores to be generated in a single HIT. For example, a worker may score more than one data product at the same time, as in the case of CollabMap. Therefore, the number of provenance records covering a HIT can vary significantly depending on the actual workflow implemented by the crowdsourcing engine.

In order to simplify the analysis of communication cost, we compute here the upper-bounds of the number of provenance records generated when a **single** data product (*create pattern*) or score (*score pattern*) is created in a HIT. In order to do that, we look

at the provenance patterns in isolation and not when they are employed in a specific application. Doing that empowers us to compute the upper-bounds in a generic way. After computing the upper-bound, we continue the analysis on the communication cost and normalise the plots presented earlier based on the total number of employed provenance patterns in a submission.

According to the Equation 6.1 of OCP, measures of a version of a provenance element (such as a data product or a worker) should be computable based on the previous version. Thus, the crowdsourcing engine is required to submit the most recent version of a provenance element and its previous version. Having this in mind, there are maximum 3 PROV entities and maximum 3 PROV relations in *revision pattern* (See Figure 3.3). Similarly, there are maximum 3 PROV agents and maximum 3 PROV relations in *agent pattern* (See Figure 3.4). We conclude that there are maximum 6 provenance records in either of *revision pattern* or *agent pattern*.

The *create pattern* is concerned with creation of a data product which is attributed to a worker. The creation activity is on the basis of some input and according to some plan (See Figure 3.1). In this pattern, *data product*, *input*, and *plan* follow the *revision pattern* and *agent* follows *agent pattern*. As such, and including the activity, there are 13 PROV elements (entities, activities, and agents) and 17 PROV relations. There are maximum 30 provenance records in *create pattern*.

The *score pattern* is about the scoring of a data product by an agent; the scoring is performed according to some plan (See Figure 3.2). In this pattern, *data product* and *plan* follow *revision pattern* and *agent* follows *agent pattern*. As such, and including the activity, there are 11 PROV elements and 15 PROV relations. There are maximum 26 provenance records in *score pattern*.

We conclude that if the provenance of data submitted by a crowdsourcing engine to *PEDRA* is about either the creation of one data product or the scoring of one data product (i.e. the provenance of data is shaped according to *create pattern* or *score pattern*), then the total number of provenance records will not exceed 30 provenance records.

However, in reality, this may not happen, as in the case of CollabMap. In these cases, *PEDRA* is able to compute quality measures and return them back to the crowdsourcing engine for online quality-based decision making (as demonstrated in the case of CollabMap). However, the number of provenance records could potentially have wide range of values. As such, the crowdsourcing engine should take necessary steps to mitigate this. Given that a crowdsourcing engine has numerous tasks to be done and many workers are available to participate, the varied communication cost will not affect the performance of the crowdsourcing engine (as will be analysed in Section 6.3.3.2).

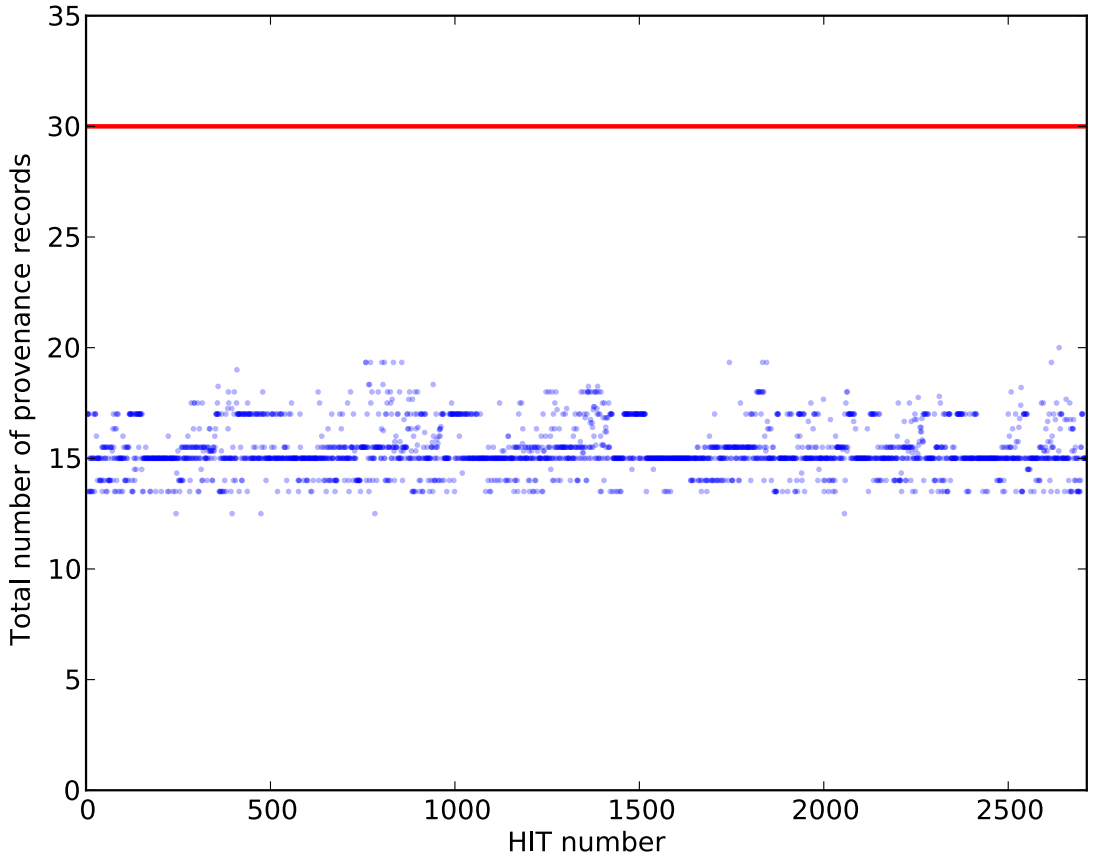


Figure 6.13: The communication cost is bounded and as such stays constant. The top red line shows the upper-bounds for the communication cost and the blue scatter plot shows the normalised number of provenance records.

Having shown that there exist upper-bounds for the number of provenance records sent to *PEDRA* in the cases when one data product and one score is created, we can conclude that the communication cost between the crowdsourcing engine and *PEDRA* is bounded (i.e. the total number of provenance records will not exceed 30) for any individual communication because either *create pattern* or *score pattern* can be sent at a time.

Indeed, this is demonstrated in Figure 6.13, in which we plot the number of provenance records sent to *PEDRA* after each HIT finished in our experiment. In order to avoid biases due to varied numbers of data products or scores generated in a HIT, we normalised the number of provenance records by the sum of number of *create pattern* and *score pattern* in a HIT. In this figure, the x-axis represents all the HITs completed in the order of time and the y-axis represents the normalised number of provenance records sent for a HIT; the upper-bound computed above shown as the red line in the plot. It can be seen here that the normalised number of provenance records sent were in fact lower than the predefined bound for all the HITs in our experiment.

We further analyse the communication cost in terms of the total number of submitted

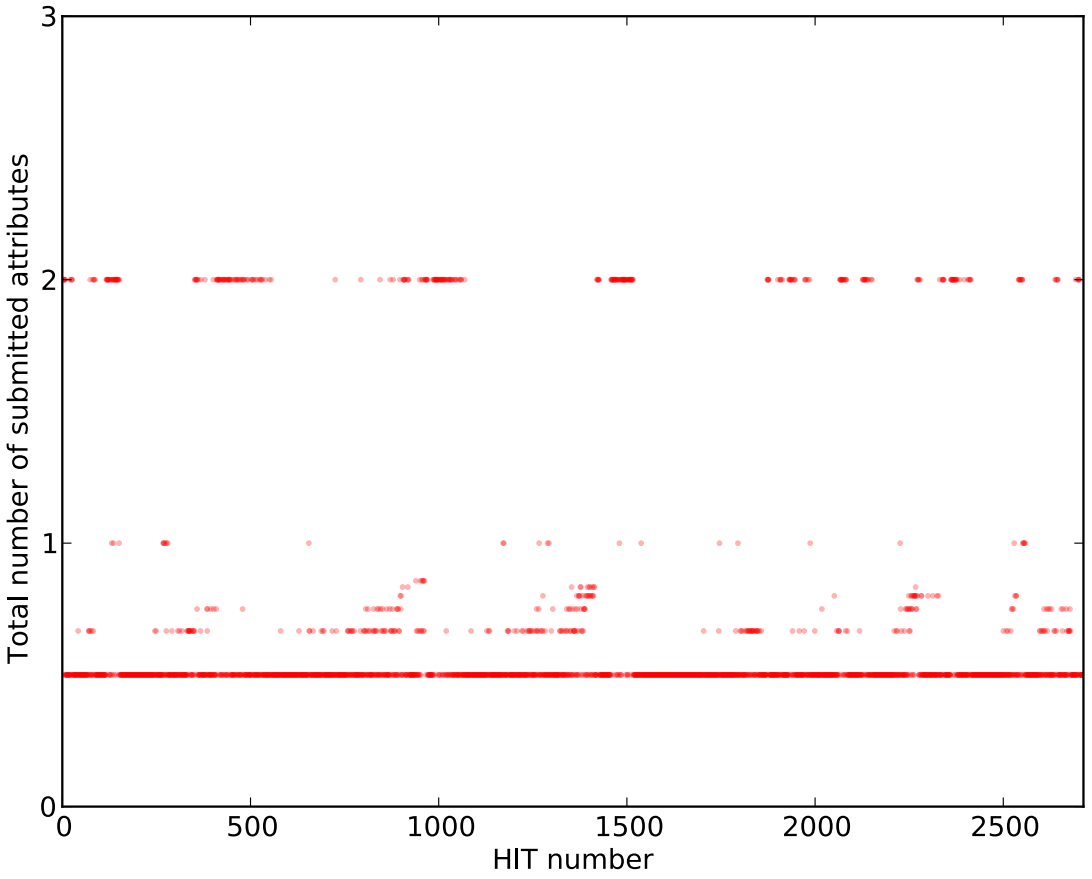


Figure 6.14: The communication cost in terms of the total number of submitted attributes by CollabMap to PEDRA.

attributes (i.e. annotations) by a crowdsourcing engine (CollabMap) to *PEDRA*. Figure 6.14 shows the total number of submitted attributes by CollabMap normalised by the sum of number of *create pattern* and *score pattern* in a HIT.

Following, we compute the upper-bounds on the total number of submitted attributes for each provenance graph, given that each provenance graph is shaped according to either *create pattern* or *score pattern*. The only attribute that *PEDRA* requires is the value of the score a scorer provided. CollabMap annotates the score entity in the provenance graph with the value of the score (either positive or negative). As such, we conclude that the total number of submitted attributes by the crowdsourcing engine is bounded (i.e. it will not exceed one attribute) for any individual submission because either *create pattern* or *score pattern* can be sent at a time.

Looking at Figure 6.14, there are some provenance graphs that contain two attributes (as opposed to the computed one attribute for the upper-bounds value). Same as the analysis on communication cost, in reality, the crowdsourcing engine may annotate more than one PROV elements in the submitted provenance graph with application-specific attributes that, in theory, *PEDRA* does not require such extra information. This happened in

the case of CollabMap where the application annotates some provenance graphs with two attributes: (1) one attribute was the value for score that *PEDRA* requires and (2) another attribute was application-specific for a type of data product (that attribute specifies total number of edges for a building) that *PEDRA* does not require.

6.3.3.2 Response Time Cost

Method While *CollabMap-PEDRA-O* was running, we recorded the time at which the provenance documents were received and sent by *PEDRA*, the difference of the two is the response time of a quality assessment request.

We want to analyse the response time as the crowdsourcing engine continues its execution. In order to do so, similar to the previous analysis, all the HITs were sorted by the time their provenance data were received by *PEDRA*. Then, we record the response time for each provenance graph. To reveal the trend obscured by noisy data points, we apply median smoothing (Cohen, 1996, p.55). This technique replaces each data point i with the median of $2k + 1$ data points around i from $i - k$ to $i + k$ including i, with $2k + 1$ is called the window size. We chose $k = 10$.

Analysis Figure 6.15 shows the scatter plot of response time of *PEDRA* over all HITs (ordered by time). The x-axis represents all the HITs and the y-axis represents the response time measured in seconds. Over a total of 2,714 HITs, the response time varied between 0.03 seconds and 0.23 seconds, with the average around 0.06 seconds.

As we ran the experiment over 17 days, with an average delay of about 2 minutes between the time a HIT was submitted to *PEDRA* and the following HIT was accepted by CollabMap, the delay above is negligible. In fact, we observed that CollabMap was not blocked by *PEDRA* during the course of the experiment. It should also be noted that, our implementation was not optimised in any way and further improvement to reduce the response time of *PEDRA* is possible.

6.3.3.3 Storage Cost

Finally, due to Requirement 1 of the OCP, *PEDRA* needs to store all the previous measures it computes. Recall from Requirement 1 that the chosen quality measures (e.g. VALIDITY ESTIMATE, CREATOR RELIABILITY, and SCORER RELIABILITY) must be incrementally computable which means the quality measures of a version of a data product or a worker are computed based on its previous version. As such, in order to compute the quality measures for a version, *PEDRA* requires to have persisted the previously compute quality measures.

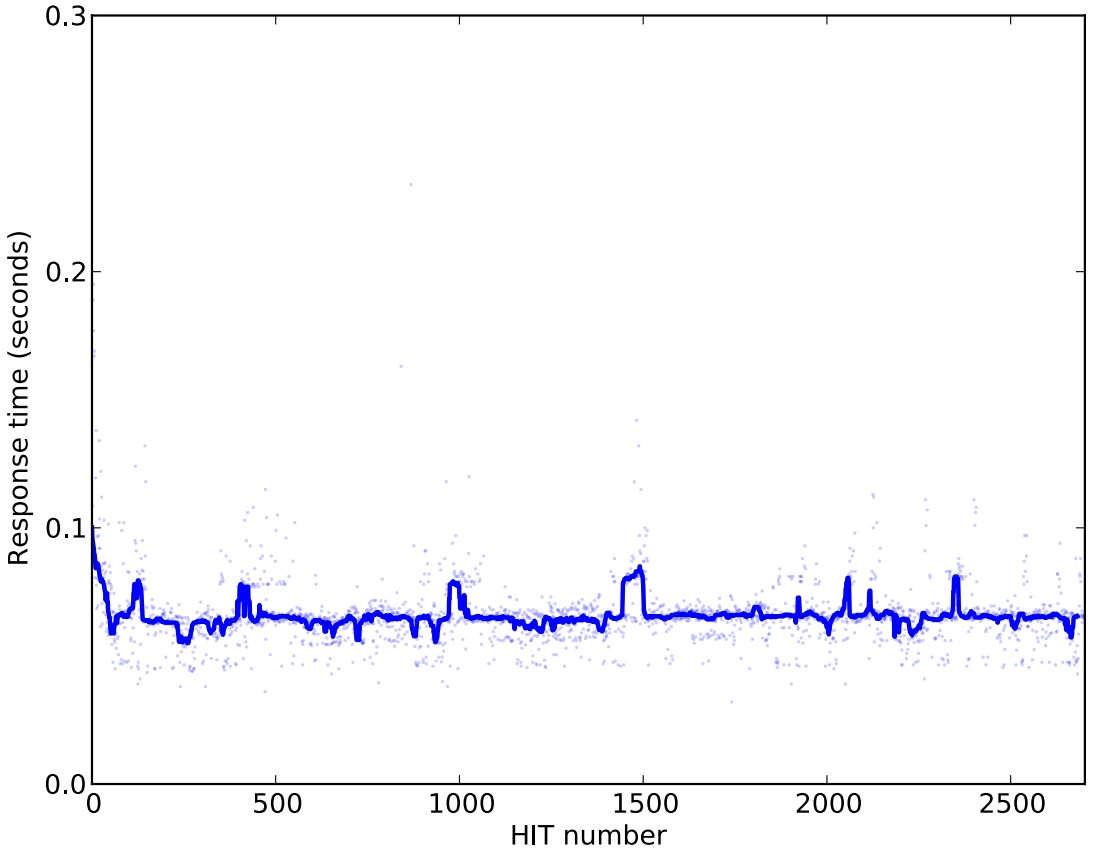


Figure 6.15: Response time of PEDRA over all HITs. OCP does not block CollabMap and the response time is acceptable to the crowdsourcing engine.

In the next analysis, we measure the *storage cost* incurred by *PEDRA* in terms of the number of measures it stores as the crowdsourcing engine progresses.

Method During the execution of *CollabMap-PEDRA-O*, we recorded the total number of measures stored by *PEDRA* after each quality assessment request (by the crowdsourcing engine when a HIT was completed).

Analysis Figure 6.16 shows the total number of measures stored by *PEDRA* grows linearly to the number of HITs at the rate of about 59 measures per HITs. Given this, the additional storage cost required by *PEDRA-O* is dictated by the number of HITs (or tasks) required by the crowdsourcing engine. This cost is linear to the number of HITs and, thus, also predictable and controllable.

Figure 6.17 illustrates the average number of measures for a version of a data product stored by *PEDRA* and used by CollabMap. The x-axis represents a version of a data product, and the y-axis represents average number of stored measures. The standard error is also shown on the plot. The red top line represents *PEDRA* and the blue bottom line represents CollabMap.

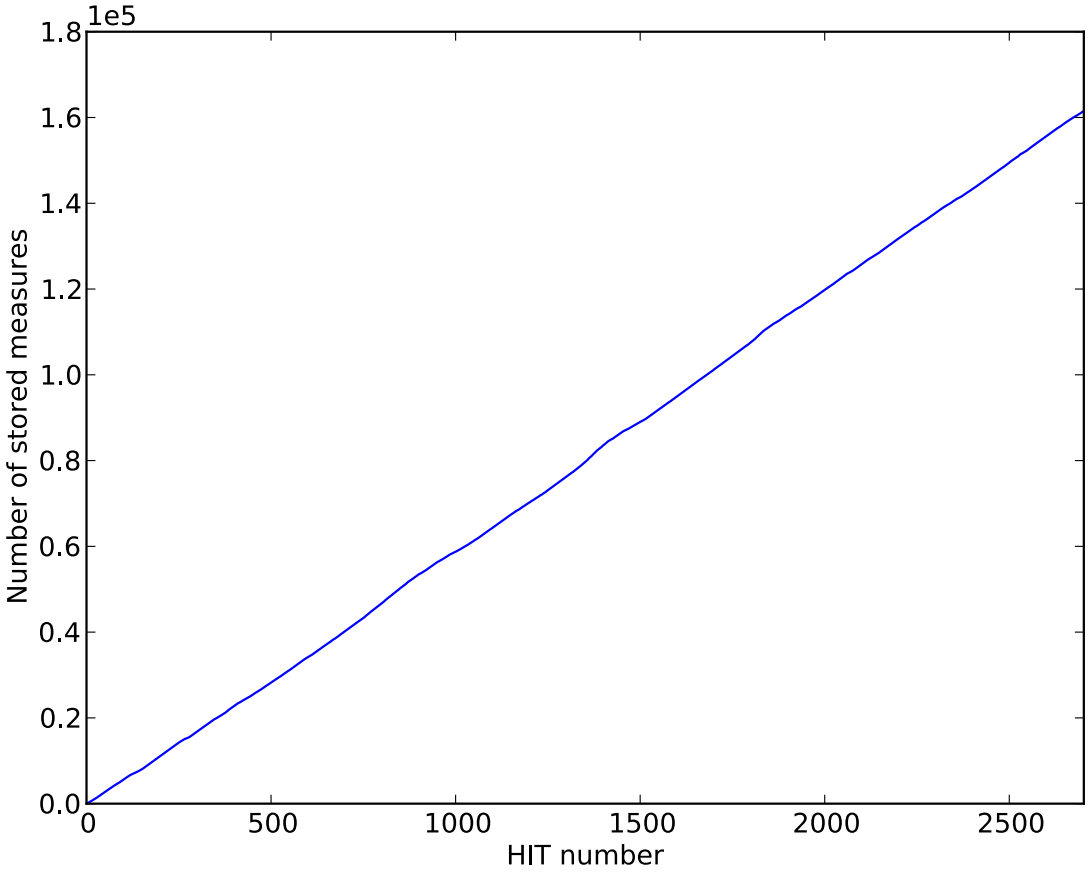


Figure 6.16: The total number of measures stored by PEDRA. The storage cost is linear to the number of HITs and, thus, predictable and controllable.

Figure 6.17 shows that CollabMap uses two measures (VALIDITY ESTIMATE and TERMINATION). It can also be seen that the average number of stored measures on *PEDRA* stays almost constant.

Figure 6.18 illustrates the average number of measures for a version of a worker stored by *PEDRA* and used by CollabMap. The x-axis represents a version of a worker, and the y-axis represents average number of stored measures. The standard error is also shown on the plot. The red top line represents *PEDRA* and the blue bottom line represents CollabMap.

Figure 6.18 shows that CollabMap uses some measures computed for a worker such as CREATOR RELIABILITY or SCORER RELIABILITY. It can also be seen that the average number of stored measures on *PEDRA* stays almost constant.

6.4 Summary

In Chapter 5, we offered a quality assessment approach called *PEDRA* that exploits the provenance of data generated in a crowdsourcing application to assess the quality of data.

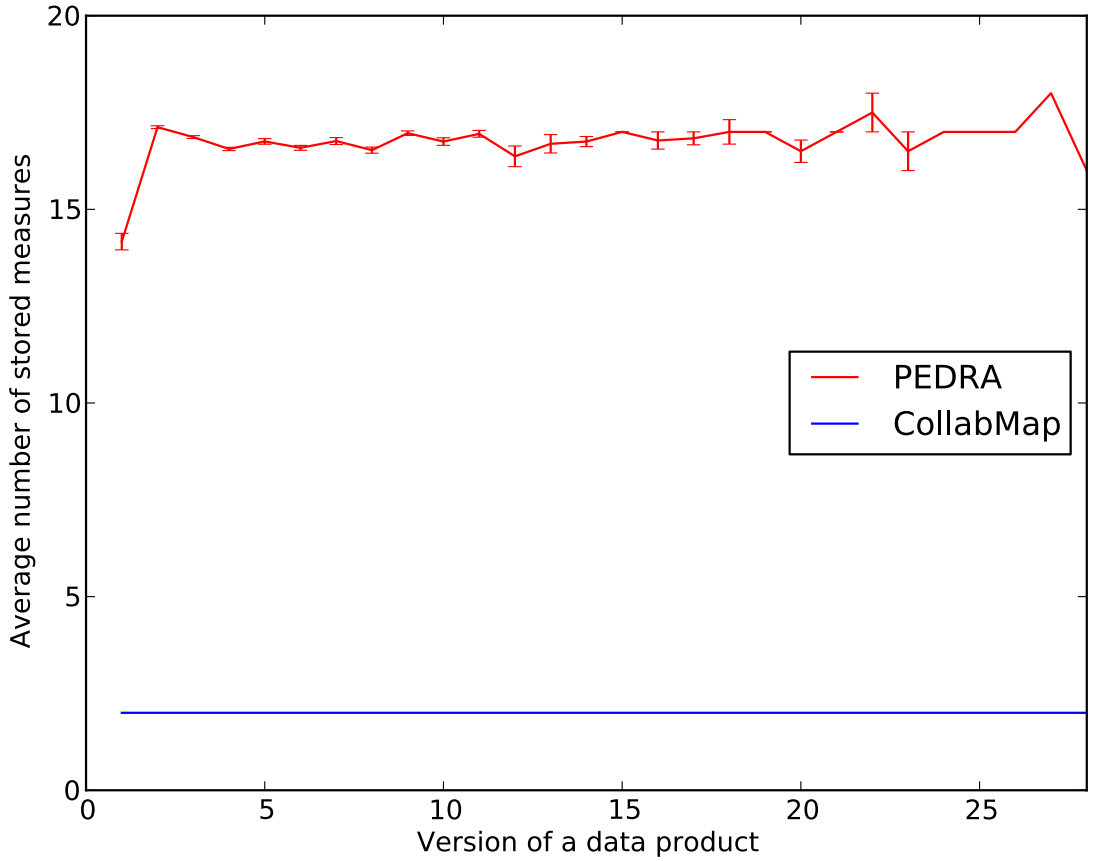


Figure 6.17: Average number of stored measures for a version of a data product.

The ultimate goal of this approach is to assist a crowdsourcing application to choose data of high quality and discard those with low quality. The crowdsourcing application recruits workers to create and verify data. After the execution of the application is completed and the dataset is ready, *PEDRA* exploits the provenance of data and assesses the quality of data. We call this the offline application of *PEDRA* as it is deployed after the execution of the application is finished; i.e. offline.

In this chapter, we identified an important advantage of online application of *PEDRA*: to assess the quality of data and reliability of workers as the crowdsourcing application is executing live. The advantage is that the quality measures computed by *PEDRA* are passed back to the crowdsourcing application which it can utilise to change its behaviour and improve its performance. One of the most important implication of this online interaction between a crowdsourcing application and *PEDRA* is dynamic task termination which directly affects workers' fee.

In order to facilitate online quality-based decision making, we envision an online architecture by offering an *Online Contract* for *PEDRA* (OCP) that regulates the interactions between the crowdsourcing engine and *PEDRA*. Furthermore, we introduced two requirements for the online use of *PEDRA*. The first requirement states that the quality measures must be incrementally computable. The second requirement states how and

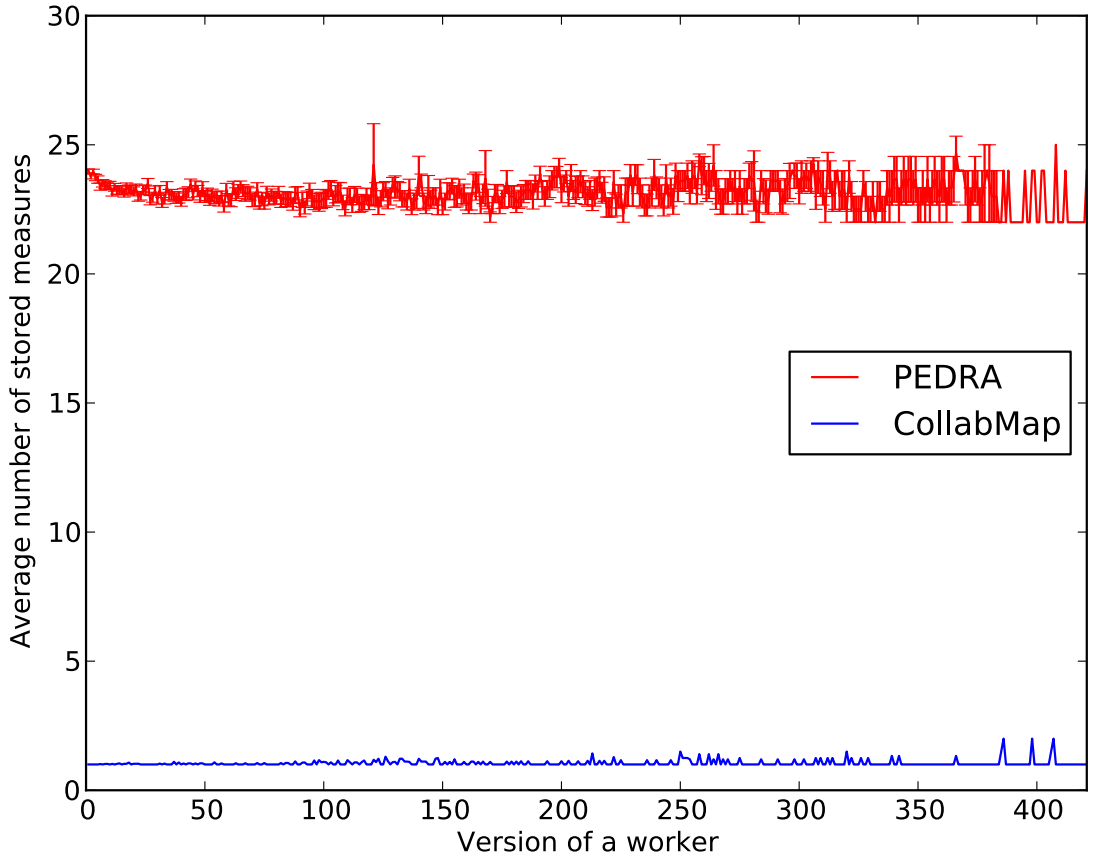


Figure 6.18: Average number of stored measures for a version of a worker.

when the crowdsourcing application is expected to send provenance of data and how computed quality measures are passed back to the crowdsourcing application.

We then compared the online application of *PEDRA* with the offline application and measured the cost that the *PEDRA-O* would incur on CollabMap, an exemplar FFV based crowdsourcing application. We showed that on average, *CollabMap-PEDRA-O* requires 20% less HITs per task compared to *CollabMap-PEDRA*. Furthermore, the communication cost is bounded and as such stays constant, OCP does not block CollabMap and the response time is acceptable to the crowdsourcing engine, and the storage cost is linear to the number of HITs and, thus, also predictable and controllable.

Chapter 7

Conclusions and Future Work

In this dissertation, we developed a novel and generic provenance enriched quality assessment approach for those crowdsourcing applications that are built on top of Find-Fix-Verify (FFV) workflow; helping them to make quality-based decisions over the crowdsourced information. Specifically, we focused on the following challenges: (1) to assess the validity of data products generated by unknown crowds, (2) to assess the reliability of workers, and (3) to estimate the number of workers required for a crowdsourcing task. The motivation behind this research is that the quality of data obtained through crowdsourcing applications is required to be assessed as the data are generated by unknown workers with varying background and expertise.

To facilitate the discussion, Chapter 2 reviews the relevant literature in the field of crowdsourcing, provenance, and trust (quality). Emphasis was given to the class of quality assessment approaches that compute workers' reliability and consider it as a factor in quality assessment. Through this chapter, we identify two main limitations of the existing models:

- Some quality models are designed for specific domains or specific applications; making the quality model not generic.
- Some quality assessment approaches assess the quality of data while the crowdsourcing application is executing live; for example, Ramchurn et al. (2013) employ *Majority Voting* quality assessment approach in CollabMap but did not present a through assessment on accuracy and quality of the work and Welinder and Perona (2010) devise an online algorithm but evaluate it in a simulated environment and do not evaluate the performance of the online algorithm in terms of the potential costs incurred by the algorithm on the crowdsourcing system. As such, first, the utility of the quality model while the crowdsourcing application is executing is not measured; and second, the performance of the quality model, in terms of the possible costs incurred on the crowdsourcing application, stays unknown.

Following the work that has already been done in the literature and considering the current limitations, we formed the following PhD thesis statement:

> *The provenance of data generated in a FFV-based crowdsourcing application can be exploited so that the quality of data and reliability of workers are assessed with the following two goals: (1) to choose data of high quality and discard data with low quality, and (2) to improve the performance of the crowdsourcing application by providing online feedback as the application is executing live.*

In order to facilitate quality assessment in FFV-based crowdsourcing applications, we assume a user's past performance is an indicator of their reliability and how they may perform in the future, thus, we propose to look at the user's verifications and how a user has been performing in the application (i.e. the reliability of a user).

In order to record a user's past interactions and the verification activities in a crowdsourcing application, we propose to record the provenance of data and exploit it to assess the quality of data and reliability of users.

The quality assessment approach is required to be generic (i.e. do not rely on the data model of the application or design specifically for a domain). As such, and given that data generated in a FFV-based crowdsourcing applications follow a common pattern, we devise a set of provenance patterns that shape the provenance of data generated from FFV-based applications (Chapter 3). More specifically, the *create pattern* is concerned with the creation of a data product and its attribution to a worker (known as creator). The *score pattern* focuses on the scoring of a data product; the scoring activity is associated with the worker (known as scorer) and the score is attributed to the scorer. Furthermore, we put forth two more provenance patterns that shape the provenance of data when a data product or a worker is undergoing revision and evolve through the application (*revision pattern* and *agent pattern* respectively). Being able to distinguish the different versions of data products and workers allows the provenance-enabled crowdsourcing applications to define quality measures over them.

Our next contribution is the proposal of a generic framework, known as *ACF*, that allows any system to traverse the provenance of data, extract information from provenance, and compute new information (Chapter 4). This framework can be utilised when provenance of data is required to be exploited with a purpose different than the one at the generation time. We utilised this framework in the context of crowdsourcing and quality assessment. A crowdsourcing application generates and maintains provenance of data. Another system is required to exploit the provenance of data with another purpose that is quality assessment. In this case, *ACF* allows the system to exploit the provenance of data, extract necessary information, and compute quality measures over the provenance graph.

The provenance of data shaped by the patterns and *ACF* facilitate our goal of designing a generic quality assessment approach. Our third contribution is the proposal of a quality assessment approach, *PEDRA*, that by instantiating *ACF*, exploits the shaped provenance of data and computes a set of quality measures that assists a crowdsourcing application to either accept or discard data (Chapter 5). Therefore, after the application finishes its execution and the dataset is ready, *PEDRA* exploits the provenance of data and annotates data with their validity; valid data to be accepted by the application and invalid ones to be discarded.

PEDRA can also be utilised to improve the performance of the crowdsourcing application by providing an online feedback as the application is executing live, which we call online application of *PEDRA*. One the advantages of the online application of *PEDRA* is dynamic task termination; *PEDRA* computes a quality measure that can be utilised by the crowdsourcing application to make an online decision on when to terminate a task. Naturally, the decision on task termination affects workers' fee and as such it should be taken seriously. In order to facilitate such online quality-based decision making, we envision an online architecture by putting forward an online contract for *PEDRA*, known as *PEDRA-O* (Chapter 6). The online contract regulates the interaction between the crowdsourcing application and *PEDRA*, stating how and when provenance of data should be submitted to *PEDRA* and how computed quality measures are passed back by *PEDRA*.

7.1 Summary of Results

The results of the work presented in this dissertation were detailed in four chapters that are associated to the research contributions listed in Chapter 1. Bellow, we outline a summary of the results:

Provenance patterns for crowdsourcing applications

Chapter 3 presents the first set of patterns that shape the provenance of data generated in FFV-based applications. The key feature of the provenance patterns is that they are generic and as such shape the provenance of data generated in any FFV-based crowdsourcing application. This chapter, also, shows how these provenance patterns can be utilised in a sample FFV-based crowdsourcing application.

Annotation Computation Framework (*ACF*)

Chapter 4 offers a generic framework, known as *ACF*, that allows a system to exploit the provenance of data with a purpose that could be potentially different than the one at the generation time. It does so by enabling the system to traverse the provenance graph, extract information, and compute new information. As *ACF* is generic, an instantiation of the framework is required to provide instructions on how new information is supposed to be computed.

Provenance Enriched Data Rating Assessment (*PEDRA*)

Chapter 5 presents the application of *PEDRA* in CollabMap, an exemplar FFV-based crowdsourcing application. We are the first to show how a provenance enriched quality assessment approach can be employed as part of a crowdsourcing application and assess the quality of generated data. In so doing, this chapter first shows the accuracy of *PEDRA* outperforms the accuracy of the current quality model employed in CollabMap by at least 6.5%. The current quality model (*Majority Voting*) is employed in many other crowdsourcing applications. Furthermore, this chapter establishes the result to be statistically significant. After this, we studied the utility of provenance in terms of uncertainty over computed quality measures. This chapter confirms that the more provenance data is exploited, the less uncertainty over the computed measures. In particular, we showed that there is an inverse correlation between the amount of provenance exploited and the uncertainty over computed quality measures and it is statistically significant.

Online decision making with *PEDRA*

Chapter 6 details an evaluation on the performance of *PEDRA* once it was deployed as a quality assessment service which is undertaken while the crowdsourcing application is executing live. We are the first to present a provenance enriched quality assessment service that accepts the quality assessment requests from FFV-based crowdsourcing applications and compute quality measures online. To best of our knowledge, we are the first to evaluate the performance of such online service for crowdsourcing applications in terms of the cost it incurred on the applications. In so doing, we demonstrate that it can be deployed for quality-based decisions, such as deciding when a crowdsourcing task is deemed to be completed and, thus, can be terminated, while the application is running live. More importantly, we show that doing so reduces at least 20% of workers' fee while achieving the same performance as the original version of CollabMap without *PEDRA*. Last but not least, the overheads introduced by *PEDRA* in terms of computational delay and communication are negligible.

7.2 Impact of our Results

The first impact of our work is on the crowdsourcing domain. One of the issues that most crowdsourcing applications face is the quality assessment of the crowdsourced information. It is required by the application to either accept or discard data generated by the unknown crowd, and for this, the application requires quality assessment.

The work described in this dissertation tackles the issue of quality assessment by proposing a generic quality assessment approach. However, what makes our approach distinct to others, is the applicability of the approach in providing online quality feedback to

the crowdsourcing application as it is executing live. The impact of this feature is (1) a better use of available resources and (2) an online task allocation, alongside with a reduction on workers' fee.

The second impact of our work is on the provenance domain. Through this work, not only we provided an approach that exploits provenance of data to assess the quality of data in a generic, systematic, and independent way, but also we showed the utility of provenance in crowdsourcing by proposing a provenance enriched quality assessment approach:

- Using provenance, we put forth a set of provenance patterns that shape the provenance of a broad number of crowdsourcing applications which facilitates the development of our generic quality assessment approach.
- The quality assessment approach exploits the shaped provenance data to compute quality measures. We showed that the more provenance is exploited, the less uncertainty over computed quality measures. In particular, we showed an inverse correlation between the use of provenance and uncertainty.

7.3 Future Work

There are still some open issues for which we do not provide a solution. Based on them, we identify the following three areas in which further research is warranted.

7.3.1 Worker's Response Value

Through *PEDRA*, we provide a quality assessment approach for situations where the worker's response is represented as a binary value. The response is normally in the form of positive/negative rating or yes/no answer. They are interpreted as +1/-1 and then the probabilistic model employed in *PEDRA* computes a VALIDITY ESTIMATE based on the value.

What we have not offered is a solution to other representations, for example when a worker's response should be represented as a real-value scalars or non-binary discrete action-spaces. For instance a crowdsourcing application may ask workers to provide their level of agreement or satisfaction in the range 1 to 10; 1 means complete dissatisfaction and 10 means complete satisfaction.

Thus, to deal with these cases, would require another instantiation of *ACF* (See Chapter 4).

Teacy et al. (2006, 2008) offer a probabilistic trust model for continuous action spaces. The proposed trust model is a general solution to make trust judgements. A line of investigation would be to adapt this trust model to crowdsourcing applications and extracting the necessary information from the provenance of data which is shaped by the provenance patterns. Given the architecture is decoupled, changing the quality model behind *PEDRA* is not only possible but also easy to do, without modifying the crowdsourcing application.

Welinder and Perona (2010) propose a model for those crowdsourcing applications that are concerned with labelling processes. The model includes label uncertainty, as well a multidimensional measure of the worker's ability. They claim the model can handle binary, multi-valued, and continuous annotations. This quality model can also be adapted to our work. However more work is necessary to make the model generic to any FFV-based crowdsourcing application (and not just labelling applications). Furthermore, the model is required to be evaluated in an online environment so its performance is evaluated.

There is no requirement to change/update the provenance patterns. The *create pattern* describes the provenance of the creation of data products and *score pattern* describes the provenance of scoring of a data product. The score entity would represent other action spaces.

7.3.2 Worker's Reliability

PEDRA computes the reliability of workers based on the full history of workers' interactions with the crowdsourcing application. There are two alternatives for computing the reliability measure for a worker.

The first alternative considers the recent interactions to be more important than older interactions. In order to do so, there are two approaches. One approach specifies a window size and considers the interactions of a worker over that specified window size (e.g. it computes reliability measure based on the last 10 interactions). Another approach utilises a decay function that defines how to lessen the effect of older interactions. This means that the influence of past interactions on reliability of a worker is considered based on recency. However, Griffiths and Miles (2013) argue that some older interactions may be significant while recent interactions may be less applicable.

The second alternative involves the consideration of mitigating circumstances once computing the reliability measure for a worker. As discussed by Miles and Griffiths (2015), mitigating circumstances may have affected previous service provision. In the crowdsourcing domain, the mitigating circumstances are unfamiliarity of workers with the system, difficulty of a task, or other environmental factors on the workers' side. As

such, future work can focus on how to incorporate these circumstances when computing the reliability measure for a worker.

Furthermore, we can envision a situation where a crowdsourcing application does not have prior knowledge over the reliability of a worker. In this case, the application may ask other crowdsourcing applications for information regarding that specific worker. This is called reputation assessment. Teacy et al. (2006) offers a reputation assessment mechanism for multi-agent systems through which if the confidence on the computed trust value based on personal experience is bellow a predetermined minimum confidence level, then the agent may seek the opinions of other agents. Further research is required to apply this reputation mechanism to our work.

7.3.3 Beyond FFV Workflow

Chapter 3 offers a set of provenance patterns that shape the provenance of data generated in a FFV-based crowdsourcing application (Refer to Section 2.1.6 for a discussion on different crowdsourcing workflows). *PEDRA* exploits the shaped provenance to assess the quality of data and reliability of workers. *PEDRA* computes a set of quality measures and rates data and workers with the computed quality measures. The crowdsourcing application can then utilise the measures to choose data of high quality.

From here, there are two broad lines of research that warrant further work:

1. The provenance patterns shape the provenance of data when data is created by workers, when data is verified/scored by the workers, and when both data and workers evolve through the time. A line of research could investigate how these provenance patterns, and subsequently *PEDRA*, can be adapted for any application (beyond FFV workflow) that has the purpose of compiling a high quality data set.

2. A line of research could focus on how the mechanisms offered in this dissertation (such as propagation, computation, and rating) could be applied to non-crowdsourcing applications. Consider a financial institute that captures the provenance of their transactions. An instantiation of *ACF* can be developed so that by propagating checks and compliance flags, the instantiation rates transactions according to their level of believed compliance.

Appendix A

PROV Notation

This appendix presents the PROV-N representation of some provenance graph that were discussed in this dissertation. The PROV-N of each example can be imported into, for example, a visualisation tool[1], for further inspection.

A.1 News Creation

```
document
  prefix CROWD <http://users.ecs.soton.ac.uk/crowd/>
  prefix PROV <http://www.w3.org/ns/prov#>

  activity(CROWD:News_Creation,2015-07-18T19:20:10.000+01:00,2015-07-18T19:20:45.000+01:00)
  entity(CROWD:Flood_News)
  entity(CROWD:News_Creation_Workflow)
  entity(CROWD:Tweet)
  entity(CROWD:Facebook_Message)
  entity(CROWD:Phone_Call)
  agent(CROWD:Bob)

  wasGeneratedBy(CROWD:wgb1;CROWD:Flood_News,CROWD:News_Creation,2015-07-18T19:20:45.000+01:00)
  wasAttributedTo(CROWD:wat1;CROWD:Flood_News, CROWD:Bob)
  wasAssociatedWith(CROWD:waw1;CROWD:News_Creation,CROWD:Bob,CROWD:News_Creation_Workflow)
  wasDerivedFrom(CROWD:wdf1;CROWD:Flood_News, CROWD:Tweet)
  wasDerivedFrom(CROWD:wdf2;CROWD:Flood_News, CROWD:Facebook_Message)
  wasDerivedFrom(CROWD:wdf3;CROWD:Flood_News, CROWD:Phone_Call)
endDocument
```

Listing A.1: Provenance of news creation scenario in PROV-N representation

A.2 News Verification

[1]ProvTranslator: https://provenance.ecs.soton.ac.uk/validator/view/translator.html

```
document
  prefix CROWD <http://users.ecs.soton.ac.uk/crowd/>
  prefix PROV <http://www.w3.org/ns/prov#>

  activity(CROWD:News_Creation,2015-07-18T19:20:10.000+01:00,2015-07-18T19:20:45.000+01:00)
  activity(CROWD:News_Verification_1,2015-07-19T10:35:10.000+01:00,2015-07-19T10
   :35:15.000+01:00)
  activity(CROWD:News_Verification_2,2015-07-19T12:20:00.000+01:00,2015-07-19T12
   :20:10.000+01:00)
  entity(CROWD:Flood_News)
  entity(CROWD:News_Creation_Workflow)
  entity(CROWD:vote1,[CROWD:vote = "+1" %% xsd:string])
  entity(CROWD:vote2,[CROWD:vote = "+1" %% xsd:string])
  entity(CROWD:News_Verification_Workflow_1)
  entity(CROWD:News_Verification_Workflow_2)
  agent(CROWD:Bob)
  agent(CROWD:Alice)
  agent(CROWD:John)

  wasGeneratedBy(CROWD:wgb1;CROWD:Flood_News,CROWD:News_Creation,2015-07-18T19:20:45.000+01:00)
  wasGeneratedBy(CROWD:wgb2;CROWD:vote1,CROWD:News_Verification_1,2015-07-19T10:35:15.000+01:00)
  wasGeneratedBy(CROWD:wgb3;CROWD:vote2,CROWD:News_Verification_2,2015-07-19T12:20:10.000+01:00)
  wasAttributedTo(CROWD:wat1;CROWD:Flood_News, CROWD:Bob)
  wasAttributedTo(CROWD:wat2;CROWD:vote1, CROWD:Alice)
  wasAttributedTo(CROWD:wat3;CROWD:vote2, CROWD:John)
  wasAssociatedWith(CROWD:waw1;CROWD:News_Creation,CROWD:Bob,CROWD:News_Creation_Workflow)
  wasAssociatedWith(CROWD:waw2;CROWD:News_Verification_1,CROWD:Alice,CROWD:
   News_Verification_Workflow_1)
  wasAssociatedWith(CROWD:waw3;CROWD:News_Verification_2,CROWD:John,CROWD:
   News_Verification_Workflow_2)
  wasDerivedFrom(CROWD:wdf4;CROWD:vote1, CROWD:Flood_News)
  wasDerivedFrom(CROWD:wdf5;CROWD:vote2, CROWD:Flood_News)
  used(CROWD:use1;CROWD:News_Verification_1,CROWD:Flood_News,2015-07-19T10:35:15.000+01:00)
  used(CROWD:use1;CROWD:News_Verification_2,CROWD:Flood_News,2015-07-19T12:20:10.000+01:00)
endDocument
```

Listing A.2: Provenance of news verification scenario in PROV-N representation

A.3 A Sample Provenance Graph Generated by CollabMap and Shaped by Provenance Patterns

```
document
  prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
  prefix pp <http://users.ecs.soton.ac.uk/pp/>
  prefix prov <http://www.w3.org/ns/prov#>

  activity(collabmap:BuildingIdentification,2015-07-18T20:20:10.000+01:00,2015-07-18T20
   :20:45.000+01:00)
  activity(collabmap:BuildingVerification,2015-07-19T10:15:20.000+01:00,2015-07-19T10
   :16:45.000+01:00)
  entity(collabmap:building10)
  entity(collabmap:building10.v3)
  entity(collabmap:building10.v4)
  entity(pp:score,[prov:scoreValue = "1" %% xsd:integer])
```

```
12    entity(collabmap:buildingVerificationPlan)
13    entity(collabmap:buildingIdentificationPlan)
14    agent(collabmap:user3)
15    agent(collabmap:user3.v16)
16    agent(collabmap:user3.v17)
17    agent(collabmap:user7)
18    agent(collabmap:user7.v7)
19    agent(collabmap:user7.v8)
20
21    specializationOf(collabmap:building10.v3,collabmap:building10)
22    specializationOf(collabmap:building10.v4,collabmap:building10)
23    specializationOf(collabmap:user3.v16,collabmap:user3)
24    specializationOf(collabmap:user3.v17,collabmap:user3)
25    specializationOf(collabmap:user7.v7,collabmap:user7)
26    specializationOf(collabmap:user7.v8,collabmap:user7)
27    alternateOf(collabmap:building10.v4,collabmap:building10.v3)
28    alternateOf(collabmap:user3.v17,collabmap:user3.v16)
29    alternateOf(collabmap:user7.v8,collabmap:user7.v7)
30    wasDerivedFrom(pp:score, collabmap:building10.v4)
31    used(collabmap:BuildingVerification,collabmap:building10.v4,-,[prov:role = "" %% xsd:string])
32    wasAssociatedWith(collabmap:buildingVerificationPlan;collabmap:BuildingVerification,collabmap:
       user3.v17,collabmap:buildingVerificationPlan)
33    wasAttributedTo(prov:attributionScore;pp:score, collabmap:user3.v17)
34    wasGeneratedBy(pp:score,collabmap:BuildingVerification,-,[prov:role = "scoreGeneration" %% xsd
       :string])
35    wasGeneratedBy(pp:wgb2;collabmap:building10,collabmap:BuildingIdentification,2015-07-18T20
       :20:45.000+01:00)
36    wasAttributedTo(pp:wat1;collabmap:building10, collabmap:user7.v8)
37    wasAttributedTo(pp:wat2;collabmap:building10.v4, collabmap:user3.v17)
38    wasAssociatedWith(pp:waw1;collabmap:BuildingIdentification,collabmap:user7.v8,collabmap:
       buildingIdentificationPlan)
39  endDocument
```

Listing A.3: A sample provenance graph generated by CollabMap and shaped by provenance patterns in PROV-N representation

A.4 A Sample Provenance Graph Generated by CollabMap About Creation and Verification of a Data Product

```
1   document
2     prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
3     prefix pp <http://users.ecs.soton.ac.uk/pp/>
4     prefix prov <http://www.w3.org/ns/prov#>
5
6     activity(collabmap:BuildingIdentification,2015-07-18T20:20:10.000+01:00,2015-07-18T20
       :20:45.000+01:00)
7     activity(collabmap:BuildingVerification,2015-07-19T10:15:20.000+01:00,2015-07-19T10
       :16:45.000+01:00)
8     entity(collabmap:building10)
9     entity(pp:score,[prov:value = "1" %% xsd:integer])
10    entity(collabmap:buildingVerificationPlan)
11    entity(collabmap:buildingIdentificationPlan)
12    agent(collabmap:user3)
13    agent(collabmap:user7)
14
```

```
15    wasDerivedFrom(pp:score, collabmap:building10)
16    used(collabmap:BuildingVerification,collabmap:building10,-,[prov:role = "" %% xsd:string])
17    wasAssociatedWith(collabmap:buildingVerificationPlan;collabmap:BuildingVerification,collabmap:
       user3,collabmap:buildingVerificationPlan)
18    wasGeneratedBy(pp:score,collabmap:BuildingVerification,-,[prov:role = "scoreGeneration" %% xsd
       :string])
19    wasGeneratedBy(pp:wgb2;collabmap:building10,collabmap:BuildingIdentification,2015-07-18T20
       :20:45.000+01:00)
20    wasAssociatedWith(pp:waw1;collabmap:BuildingIdentification,collabmap:user7,collabmap:
       buildingIdentificationPlan)
21  endDocument
```

Listing A.4: A sample provenance graph generated by CollabMap about creation and verification of a data product in PROV-N representation

A.5 A Sample Provenance Graph Generated by a Crowdsourcing Application Shaped by Score Pattern

```
1   document
2     prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
3     prefix pp <http://users.ecs.soton.ac.uk/pp/>
4     prefix prov <http://www.w3.org/ns/prov#>
5
6     activity(collabmap:BuildingVerification,2015-07-19T10:15:20.000+01:00,2015-07-19T10
       :16:45.000+01:00)
7     entity(collabmap:building10)
8     entity(collabmap:building10.v3,[pp:positiveScores = "3" %% xsd:integer, pp:negativeScores =
       "0" %% xsd:integer, pp:validityEstimate = "0.8" %% xsd:double, pp:validityLabel = "1" %% xsd:
       integer])
9     entity(collabmap:building10.v4)
10    entity(pp:score,[prov:scoreValue = "1" %% xsd:integer])
11    entity(collabmap:buildingVerificationPlan)
12    agent(collabmap:user3)
13    agent(collabmap:user3.v16)
14    agent(collabmap:user3.v17)
15    specializationOf(collabmap:building10.v3,collabmap:building10)
16    specializationOf(collabmap:building10.v4,collabmap:building10)
17    specializationOf(collabmap:user3.v16,collabmap:user3)
18    specializationOf(collabmap:user3.v17,collabmap:user3)
19    alternateOf(collabmap:building10.v4,collabmap:building10.v3)
20    alternateOf(collabmap:user3.v17,collabmap:user3.v16)
21    wasDerivedFrom(pp:score, collabmap:building10.v4)
22    used(collabmap:BuildingVerification,collabmap:building10.v4,-,[prov:role = "" %% xsd:string])
23    wasAssociatedWith(collabmap:buildingVerificationPlan;collabmap:BuildingVerification,collabmap:
       user3.v17,collabmap:buildingVerificationPlan)
24    wasAttributedTo(prov:attributionScore;pp:score, collabmap:user3.v17)
25    wasAttributedTo(prov:attributionRevision;collabmap:building10.v4, collabmap:user3.v17)
26    wasGeneratedBy(pp:score,collabmap:BuildingVerification,-,[prov:role = "scoreGeneration" %% xsd
       :string])
27  endDocument
```

Listing A.5: A sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product) in PROV-N representation

A.6 A Sample Provenance Graph Generated by a Crowdsourcing Application Shaped by Create Pattern

```
document
  prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
  prefix pp <http://users.ecs.soton.ac.uk/pp/>
  prefix prov <http://www.w3.org/ns/prov#>

  activity(collabmap:BuildingIdentification,2015-07-18T20:20:10.000+01:00,2015-07-18T20
   :20:45.000+01:00)
  entity(collabmap:building10)
  entity(collabmap:building10.v3)
  entity(collabmap:building10.v4,[pp:validityLabel = "1" %% xsd:integer])
  entity(collabmap:buildingIdentificationPlan)
  agent(collabmap:user7)
  agent(collabmap:user7.v7,[pp:validCreations = "7" %% xsd:integer, pp:invalidCreations = "0" %%
    xsd:integer, pp:creatorReliability = "0.88" %% xsd:double])
  agent(collabmap:user7.v8)

  specializationOf(collabmap:building10.v3,collabmap:building10)
  specializationOf(collabmap:building10.v4,collabmap:building10)
  specializationOf(collabmap:user7.v7,collabmap:user7)
  specializationOf(collabmap:user7.v8,collabmap:user7)
  alternateOf(collabmap:building10.v4,collabmap:building10.v3)
  alternateOf(collabmap:user7.v8,collabmap:user7.v7)
  wasGeneratedBy(pp:wgb2;collabmap:building10,collabmap:BuildingIdentification,2015-07-18T20
   :20:45.000+01:00)
  wasAttributedTo(pp:wat1;collabmap:building10, collabmap:user7.v8)
  wasAssociatedWith(pp:waw1;collabmap:BuildingIdentification,collabmap:user7.v8,collabmap:
   buildingIdentificationPlan)
endDocument
```

Listing A.6: A sample provenance graph generated by a crowdsourcing application shaped by create pattern (provenance of creation of a data product) in PROV-N representation

A.7 A Sample Provenance Graph Generated by a Crowdsourcing Application Shaped by Score Pattern Including Scorer Reliability Measures

```
document
  prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
  prefix pp <http://users.ecs.soton.ac.uk/pp/>
  prefix prov <http://www.w3.org/ns/prov#>

  activity(collabmap:BuildingVerification,2015-07-19T10:15:20.000+01:00,2015-07-19T10
   :16:45.000+01:00)
  entity(collabmap:building10)
  entity(collabmap:building10.v3)
  entity(collabmap:building10.v4,[pp:validityLabel = "1" %% xsd:integer])
  entity(pp:score,[prov:scoreValue = "1" %% xsd:integer])
  entity(collabmap:buildingVerificationPlan)
```

```
12    agent(collabmap:user3)
13    agent(collabmap:user3.v16,[pp:alignedScores = "12" %% xsd:integer, pp:notAlignedScores = "4"
       %% xsd:integer, pp:scorerReliability = "0.72" %% xsd:double])
14    agent(collabmap:user3.v17)
15    specializationOf(collabmap:building10.v3,collabmap:building10)
16    specializationOf(collabmap:building10.v4,collabmap:building10)
17    specializationOf(collabmap:user3.v16,collabmap:user3)
18    specializationOf(collabmap:user3.v17,collabmap:user3)
19    alternateOf(collabmap:building10.v4,collabmap:building10.v3)
20    alternateOf(collabmap:user3.v17,collabmap:user3.v16)
21    wasDerivedFrom(pp:score, collabmap:building10.v4)
22    used(collabmap:BuildingVerification,collabmap:building10.v4,-,[prov:role = "" %% xsd:string])
23    wasAssociatedWith(collabmap:buildingVerificationPlan;collabmap:BuildingVerification,collabmap:
       user3.v17,collabmap:buildingVerificationPlan)
24    wasAttributedTo(prov:attributionScore;pp:score, collabmap:user3.v17)
25    wasAttributedTo(prov:attributionRevision;collabmap:building10.v4, collabmap:user3.v17)
26    wasGeneratedBy(pp:score,collabmap:BuildingVerification,-,[prov:role = "scoreGeneration" %% xsd
       :string])
27  endDocument
```

Listing A.7: A sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product) to compute scorer reliability in PROV-N representation

A.8 A Sample Provenance Graph Generated by a Crowdsourcing Application Shaped by Score Pattern Including Validity Rating Measures

```
1   document
2     prefix collabmap <http://www.orchid.ac.uk/ontologies/collabmap.owl#>
3     prefix pp <http://users.ecs.soton.ac.uk/pp/>
4     prefix prov <http://www.w3.org/ns/prov#>
5     activity(collabmap:BuildingVerification,2015-07-19T10:15:20.000+01:00,2015-07-19T10
       :16:45.000+01:00)
6     entity(collabmap:building10)
7     entity(collabmap:building10.v3,[pp:cws = "1.4" %% xsd:double, pp:termination = "Continue" %%
       xsd:string])
8     entity(collabmap:building10.v4,[pp:validityLabel = "1" %% xsd:integer])
9     entity(pp:score,[prov:scoreValue = "1" %% xsd:integer])
10    entity(collabmap:buildingVerificationPlan)
11    agent(collabmap:user3)
12    agent(collabmap:user3.v16)
13    agent(collabmap:user3.v17,[pp:scorerReliability = "0.73" %% xsd:double])
14    specializationOf(collabmap:building10.v3,collabmap:building10)
15    specializationOf(collabmap:building10.v4,collabmap:building10)
16    specializationOf(collabmap:user3.v16,collabmap:user3)
17    specializationOf(collabmap:user3.v17,collabmap:user3)
18    alternateOf(collabmap:building10.v4,collabmap:building10.v3)
19    alternateOf(collabmap:user3.v17,collabmap:user3.v16)
20    wasDerivedFrom(pp:score, collabmap:building10.v4)
21    used(collabmap:BuildingVerification,collabmap:building10.v4,-,[prov:role = "" %% xsd:string])
22    wasAssociatedWith(collabmap:buildingVerificationPlan;collabmap:BuildingVerification,collabmap:
       user3.v17,collabmap:buildingVerificationPlan)
```

```
23    wasAttributedTo(prov:attributionScore;pp:score, collabmap:user3.v17)
24    wasAttributedTo(prov:attributionRevision;collabmap:building10.v4, collabmap:user3.v17)
25    wasGeneratedBy(pp:score,collabmap:BuildingVerification,-,[prov:role = "scoreGeneration" %% xsd
       :string])
26  endDocument
```

Listing A.8: A sample provenance graph generated by a crowdsourcing application shaped by score pattern (provenance of scoring of a data product) to compute validity rating in PROV-N representation

Appendix B

Java Source Code

B.1 No Termination Instantiation

```
public class noTerminationInstantiation implements IPropagation {
  public void backward(Document graph, Relation0 relation, Element cause,
    Element effect, Hashtable<QName, Object> inputAnnotation,
    Hashtable<QName, Object> outputAnnotation) {
    c = 0
    if(inputAnnotation.get(annY) != null) {
      c = inputAnnotation.get(annY);
    }
    c = c + 1
    outputAnnotation.put(annX, c)
  }

  public void forward(Document graph, Relation0 relation, Element cause,
    Element effect, Hashtable<QName, Object> inputAnnotation,
    Hashtable<QName, Object> outputAnnotation) {
    c = 0
    if(inputAnnotation.get(annX) != null) {
      c = inputAnnotation.get(annX);
    }
    c = c + 1
    outputAnnotation.put(annY, c)
  }

  public void aggregation(Document graph, Element annotatedElement,
    Hashtable<QName, Object> inputAnnotation,
    Hashtable<QName, Object> outputAnnotation) {
    // Java code
  }
}
```

Listing B.1: Full Java source code for no termination instantiation.

B.2 Dependency-Graph Instantiation

```
1  public class SampleInstantiation implements IPropagation {
2    public void backward(
3      Document graph,
4      Relation0 relation,
5      Element cause,
6      Element effect,
7      Hashtable<QName, Object> inputAnnotation,
8      Hashtable<QName, Object> outputAnnotation) {
9      // Empty method as it is not needed
10   }
11
12   public void forward(
13     Document graph,
14     Relation0 relation,
15     Element cause,
16     Element effect,
17     Hashtable<QName, Object> inputAnnotation,
18     Hashtable<QName, Object> outputAnnotation) {
19     if(relation instanceof Used) {
20       forwardUsage(graph, (Used)relation, cause, effect, inputAnnotation, outputAnnotation);
21     }
22     else if(relation instanceof WasGeneratedBy) {
23       forwardGeneration(graph, (WasGeneratedBy)relation, cause, effect, inputAnnotation,
     outputAnnotation);
24     }
25   }
26
27   public void aggregation(
28     Document graph, Element annotatedElement,
29     Hashtable<QName, Object> inputAnnotation,
30     Hashtable<QName, Object> outputAnnotation) {
31     if (inputAnnotation.get(INCREMENT) != null) {
32       int increment = (Integer) inputAnnotation.get(INCREMENT);
33       int dependency = (inputAnnotation.get(DEPENDENCY) == null) ? 0
34         : (Integer) inputAnnotation.get(DEPENDENCY);
35       dependency = dependency + increment;
36       outputAnnotation.put(DEPENDENCY, dependency);
37     }
38   }
39
40   public void forwardUsage(
41     Document graph,
42     Used relation,
43     Entity cause,
44     Activity effect,
45     Hashtable<QName, Object> inputAnnotation,
46     Hashtable<QName, Object> outputAnnotation) {
47     outputAnnotation.put(INCREMENT, inputAnnotation.get(MEDIA) != null);
48   }
49
50   public void forwardGeneration(
51     Document graph,
52     WasGeneratedBy relation,
53     Activity cause,
54     Entity effect,
55     Hashtable<QName, Object> inputAnnotation,
56     Hashtable<QName, Object> outputAnnotation) {
57       outputAnnotation.put(DEPENDENCY, inputAnnotation.get(DEPENDENCY) != null);
```

```
58 }
```

Listing B.2: Full Java source code for the Dependency-Graph instantiation.

Appendix C

Mapping Between PEDRA and ACF

This appendix summarises all the propagation and computational rules employed in *PEDRA* as the mapping between *PEDRA* and *ACF*.

C.1 Rule 1 - 3

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If wasDerivedFrom}(score, d^v)\\
&\quad d^v.verificationValue \leftarrow score.scoreValue
\end{aligned}
\tag{C.1}
$$

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If wasDerivedFrom}(d^{v+1}, d^v, \text{[prov:type=prov:Revision]})\\
&\quad d^{v+1}.oldPositiveScores \leftarrow d^v.positiveScores\\
&\quad d^{v+1}.oldNegativeScores \leftarrow d^v.negativeScores
\end{aligned}
\tag{C.2}
$$

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad d^v.positiveScores \leftarrow d^v.oldPositiveScores + d^v.verificationValue\\
&\qquad \text{if } d^v.verificationValue > 0\\
&\quad d^v.negativeScores \leftarrow d^v.oldNegativeScores + d^v.verificationValue\\
&\qquad \text{if } d^v.verificationValue < 0\\
&\quad d^v.validityEstimate \leftarrow \text{Apply Equation 5.3 with}\\
&\qquad S^+(d^v) = d^v.positiveScores\\
&\qquad S^-(d^v) = d^v.negativeScores\\
&\quad d^v.validityLabel \leftarrow \text{Apply Equation 5.4 with}\\
&\qquad Q_V(d^v) = d^v.validityEstimate
\end{aligned}
\tag{C.3}
$$

C.2 Rule 4 - 10

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \text{wasDerivedFrom}(d^{v+1}, d^{v}, \text{[prov:type=prov:Revision]})\\
&\quad \text{and } d^{v}.type = \text{dataProductVersion and}\\
&\quad \text{and } d^{v+1}.type = \text{dataProductVersion}\\
&\qquad d^{v}.hasRevision \leftarrow 1
\end{aligned}
\tag{C.4}
$$

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \text{wasDerivedFrom}(d^{v+1}, d^{v}, \text{[prov:type=prov:Revision]})\\
&\quad \text{and } d^{v+1}.hasRevision \neq 1 \text{ and } d^{v}.hasRevision = 1\\
&\qquad d_{v}.mainValidityLabel \leftarrow d^{v+1}.validityLabel
\end{aligned}
\tag{C.5}
$$

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \text{specializationOf}(d^{v}, d)\\
&\quad d.mainValidityLabelFromVersion \leftarrow d_{v}.mainValidityLabel
\end{aligned}
\tag{C.6}
$$

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \text{wasGeneratedBy}(d, dc)\\
&\quad dc.mainValidityLabelFromOriginal \leftarrow d.mainValidityLabelFromVersion
\end{aligned}
\tag{C.7}
$$

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If } \text{wasAssociatedWith}(dc, u^{v})\\
&\quad u^{v}.validCreation \leftarrow dc.mainValidityLabelFromOriginal\\
&\qquad \text{if } dc.mainValidityLabelFromOriginal = 1\\
&\quad u^{v}.invalidCreation \leftarrow dc.mainValidityLabelFromOriginal\\
&\qquad \text{if } dc.mainValidityLabelFromOriginal = -1
\end{aligned}
\tag{C.8}
$$

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If } \text{wasDerivedFrom}(u^{v+1}, u^{v}, \text{[prov:type=prov:Revision]})\\
&\quad u^{v+1}.oldValidCreations \leftarrow u^{v}.validCreations\\
&\quad u^{v+1}.oldInvalidCreations \leftarrow u^{v}.invalidCreations
\end{aligned}
\tag{C.9}
$$

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad u^v.validCreations \leftarrow u^v.oldValidCreations + u^v.validCreation\\
&\quad u^v.invalidCreations \leftarrow u^v.oldInvalidCreations + u^v.invalidCreation\\
&\quad u^v.creatorReliability \leftarrow \text{Apply Equation 5.8 with}\\
&\qquad S_C^+(u^v) = u^v.validCreations\\
&\qquad S_C^-(u^v) = u^v.invalidCreations
\end{aligned}
\tag{C.10}
$$

C.3 Rule 11 - 14

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If wasDerivedFrom}(score, d^v)\\
&\quad d^v.verificationValue \leftarrow score.scoreValue
\end{aligned}
\tag{C.11}
$$

$$
\begin{aligned}
&F_{backward}\text{:}\\
&\text{If wasAttributedTo}(d^v, u^v)\\
&\quad u^v.alignedScore \leftarrow 1\\
&\qquad \text{if } d^v.verificationValue = d^v.validityLabel\\
&\quad u^v.notAlignedScore \leftarrow 1\\
&\qquad \text{if } d^v.verificationValue \neq d^v.validityLabel
\end{aligned}
\tag{C.12}
$$

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If wasDerivedFrom}(u^{v+1}, u^v, \text{[prov:type=prov:Revision]})\\
&\quad u^{v+1}.oldAlignedScores \leftarrow u^v.alignedScores\\
&\quad u^{v+1}.oldNotAlignedScores \leftarrow u^v.notAlignedScores
\end{aligned}
\tag{C.13}
$$

$$
\begin{aligned}
&F_{aggregate}\text{:}\\
&\quad u^v.alignedScores \leftarrow u^v.oldAlignedScores + u^v.alignedScore\\
&\quad u^v.notAlignedScores \leftarrow u^v.oldNotAlignedScores + u^v.notAlignedScore\\
&\quad u^v.scorerReliability \leftarrow \text{Apply Equation 5.12 with}\\
&\qquad S_S^+(u^v) = u^v.alignedScores\\
&\qquad S_S^-(u^v) = u^v.notAlignedScores
\end{aligned}
\tag{C.14}
$$

C.4 Rule 15 - 19

$$
\begin{aligned}
&F_{forward}\text{:}\\
&\text{If wasAssociatedWith}(bv, u^v)\\
&\quad bv.scorerReliabilityFromVersion \leftarrow u^v.scorerReliability
\end{aligned}
\tag{C.15}
$$

$F_{forward}$:
If wasGeneratedBy$(score, bv)$
$score.scorerReliabilityFromActivity \leftarrow bv.scorerReliabilityFromVersion$
(C.16)

$F_{backward}$:
If wasDerivedFrom$(score, d^v)$
$d^v.scorerReliabilityFromScore \leftarrow +score.scorerReliabilityFromActivity$
if $score.value > 0$
$d^v.scorerReliabilityFromScore \leftarrow -score.scorerReliabilityFromActivity$
if $score.value < 0$
(C.17)

$F_{forward}$:
If wasDerivedFrom$(d^{v+1}, d^v,$ [prov:type=prov:Revision]) (C.18)
$d^{v+1}.oldCws \leftarrow d^v.cws$

$F_{aggregate}$:
$d^v.cws \leftarrow d^v.oldCws + d^v.scorerReliabilityFromScore$
$d^v.termination \leftarrow$ Apply Equation 5.15 with
CWS$(d^v) = d^v.cws$ (C.19)

References

Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 Workshop on New Security Paradigms*, NSPW '97, pages 48–60, New York, NY, USA, 1997. ACM. ISBN 0-89791-986-6.

Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 310–317, New York, NY, USA, 2001. ACM. ISBN 1-58113-436-3.

Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sren Auer, and Jens Lehmann. Crowdsourcing linked data quality assessment. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web ISWC 2013*, volume 8219 of *Lecture Notes in Computer Science*, pages 260–276. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-41337-7.

Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander Kamvar. The jabberwocky programming environment for structured social computing. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 53–64, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1.

Eleanor Ainy, Susan B Davidson, Daniel Deutch, and Tova Milo. Approximated provenance for complex applications. In *6th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2014)*, Cologne, 2014. USENIX Association.

Rocío Aldeco-Pérez and Luc Moreau. Provenance-based auditing of private data use. In *Proceedings of the 2008 International Conference on Visions of Computer Science: BCS International Academic Conference*, VoCS'08, pages 141–152, Swinton, UK, August 2008. British Computer Society.

Bogdan Alexe, Laura Chiticariu, and Wang-Chiew Tan. Spider: A schema mapping debugger. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB '06, pages 1179–1182. VLDB Endowment, 2006.

Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58 – 71, June 2007. ISSN 1570-8268.

Chris Baillie, Peter Edwards, and Edoardo Pignotti. Qual: A provenance-aware quality model. *J. Data and Information Quality*, 5(3):12:1–12:22, March 2015. ISSN 1936-1955.

Omar Benjelloun, Anish Das Sarma, Chris Hayworth, and Jennifer Widom. An introduction to uldbs and the trio system. Technical Report 2006-7, Stanford InfoLab, 2006.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A word processor with a crowd inside. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5.

Daren C Brabham. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: the international journal of research into new media technologies*, 14(1):75–90, 2008.

Axel Bruns. *Blogs, Wikipedia, Second Life, and beyond: From production to produsage*, volume 45. Peter Lang Publishing Inc, 2008. ISBN 0820488666.

Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-41456-8.

Aleksejs Busarovs. Ethical aspects of crowdsourcing, or is it a modern form of exploitation. *International Journal of Economics & Business Administration (IJEBA)*, I(1): 3–14, 2013.

Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. Dbnotes: A post-it system for relational databases based on provenance. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 942–944, New York, NY, USA, 2005. ACM. ISBN 1-59593-060-4.

Paul R Cohen. *Empirical methods for artificial intelligence*. MIT Press, 1996. ISBN 9780262032254.

Yingwei Cui, Jennifer Widom, and Janet L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, June 2000. ISSN 0362-5915.

Vasa Curcin, Simon Miles, R Danger, Y Chen, Richard Bache, and Adel Taweel. Implementing interoperable provenance in biomedical research. *Future Generation Computer Systems*, 34:1–16, 2014.

Chenyun Dai, Dan Lin, Elisa Bertino, and Murat Kantarcioglu. An approach to evaluate data trustworthiness based on data provenance. In Willem Jonker and Milan Petkovi, editors, *Secure Data Management*, volume 5159 of *Lecture Notes in Computer Science*, pages 82–98. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-85258-2.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

Ross Dawson and Steve Bynghall. *Getting results from crowds*. Advanced Human Technologies San Francisco, 2012. ISBN 0984783822.

Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 469–478, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1229-5.

Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. Towards an integrated crowdsourcing definition. *J. Inf. Sci.*, 38(2):189–200, April 2012. ISSN 0165-5515.

Zoubin Ghahramani and Hyun-Chul Kim. Bayesian classifier combination. 2003.

Yolanda Gil and Donovan Artz. Towards content trust of web resources. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(4):227 – 239, 2007. ISSN 1570-8268. World Wide Web Conference 2006Semantic Web Track.

Boris Glavic and Klaus R Dittrich. Data provenance: A categorization of existing approaches. In *Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 7.-9. März 2007, Aachen, Germany*, volume 7, pages 227–241. Citeseer, 2007.

Jeremy Goecks, Anton Nekrutenko, and James Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8), 2010.

Jennifer Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In Luc Moreau and Ian Foster, editors, *Provenance and Annotation of Data*, volume 4145 of *Lecture Notes in Computer Science*, pages 101–108. Springer Berlin Heidelberg, 2006a. ISBN 978-3-540-46302-3.

Jennifer Golbeck. Trust on the world wide web: A survey. *Found. Trends Web Sci.*, 1 (2):131–197, January 2006b. ISSN 1555-077X.

Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *Communications Surveys & Tutorials, IEEE*, 3(4):2–16, 2000. ISSN 1553-877X.

Elizabeth Gray, Jean-Marc Seigneur, Yong Chen, and Christian Jensen. Trust propagation in small worlds. In Paddy Nixon and Sotirios Terzis, editors, *Trust Management*, volume 2692 of *Lecture Notes in Computer Science*, pages 239–254. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40224-4.

Nathan Griffiths and Simon Miles. An architecture for justified assessments of service provider reputation. In *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*, pages 345–352, September 2013.

R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X.

Ragib Hasan, Radu Sion, and Marianne Winslett. The case of the fake picasso: Preventing history forgery with secure provenance. In *Proccedings of the 7th Conference on File and Storage Technologies*, FAST '09, pages 1–14, Berkeley, CA, USA, 2009. USENIX Association.

Jeff Howe. Crowdsourcing: A definition. *Crowdsourcing: Tracking the rise of the amateur*, 2006a.

Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006b.

Jingwei Huang. *Knowledge provenance: An approach to modeling and maintaining the evolution and validity of knowledge*. PhD thesis, Department of Mechanical and Industrial Engineering, University of Toronto, July 2008.

Trung Dong Huynh, Mark Ebden, Matteo Venanzi, Sarvapali Ramchurn, Stephen Roberts, and Luc Moreau. Interpretation of crowdsourced activities using provenance network analysis. In *The First AAAI Conference on Human Computation and Crowdsourcing*. Association for the Advancement of Artificial Intelligence, November 2013a.

Trung Dong Huynh, Michael O. Jewell, Amir Sezavar Keshavarz, Danius T. Michaelides, Huanjia Yang, and Luc Moreau. The prov-json serialization. W3c member submission, 2013b.

Panagiotis G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, 17 (2):16–21, December 2010. ISSN 1528-4972.

Laura Chiticariu James Cheney and Wang-Chiew Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2007. ISSN 1931-7883.

Norman L Johnson and Samuel Kotz. *Distributions in Statistics: Continuous Univariate Distributions: Vol.: 2*. Houghton Mifflin, 1970.

Audun Jøsang. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3):279–311, June 2001. ISSN 0218-4885.

Audun Jsang and Roslan Ismail. The beta reputation system. In *Proceedings of the 15th bled electronic commerce conference*, pages 41–55, 2002.

Audun Jsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618 – 644, 2007. ISSN 0167-9236. Emerging Issues in Collaborative Commerce.

David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1953–1961. Curran Associates, Inc., 2011.

Amir Sezavar Keshavarz, Trung Dong Huynh, and Luc Moreau. Provenance for online decision making. In Bertram Ludäscher and Beth Plale, editors, *Provenance and Annotation of Data and Processes*, volume 8628 of *Lecture Notes in Computer Science*, pages 44–55. Springer International Publishing, 2015. ISBN 978-3-319-16461-8.

Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 453–456, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1.

Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. Crowdweaver: Visually managing complex crowd work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1033–1036, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1086-4.

Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. Crowdforge: Crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 43–52, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1.

Ryan K. L. Ko, Peter Jagadpramana, Miranda Mowbray, Siani Pearson, Markus Kirchberg, Qianhui Liang, and Bu Sung Lee. Trustcloud: A framework for accountability and trust in cloud computing. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 584–588, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4461-8.

Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1003–1012, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1086-4.

Bernd Lahno. Is trust the result of bayesian learning? In Joachim Behnke, Thomas Plmper, and Hans-Peter Burth, editors, *Jahrbuch fr Handlungs- und Entscheidungstheorie*, Jahrbuch fr Handlungs- und Entscheidungstheorie, pages 47–68. VS Verlag fr Sozialwissenschaften, 2004. ISBN 978-3-531-14339-2.

Xian Li, Timothy Lebo, and DeborahL. McGuinness. Provenance-based strategies to develop trust in semantic web applications. In DeborahL. McGuinness, JamesR. Michaelis, and Luc Moreau, editors, *Provenance and Annotation of Data and Processes*, volume 6378 of *Lecture Notes in Computer Science*, pages 182–197. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17818-4.

Christopher H Lin, Mausam Daniel, and S Weld. Dynamically switching between synergistic workflows for crowdsourcing. In *In Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI12*, 2012.

David JC MacKay. *Information theory, inference, and learning algorithms*, volume 7. Cambridge university press, 2003. ISBN 0521642981.

M. Marge, S. Banerjee, and A.I. Rudnicky. Using the amazon mechanical turk for transcription of spoken language. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5270–5273, March 2010.

Stephen Paul Marsh. Formalising trust as a computational concept. *PhD dissertation. University of Stirling, scotland*, 1994.

Winter Mason and Siddharth Suri. Conducting behavioral research on amazons mechanical turk. *Behavior Research Methods*, 44(1):1–23, 2012.

Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". *SIGKDD Explor. Newsl.*, 11(2):100–108, May 2010. ISSN 1931-0145.

Danius Michaelides, Trung Dong Huynh, and Luc Moreau. PROV-Template: A template system for prov documents. Public working draft of a potential specification, 2014.

Simon Miles and Nathan Griffiths. Incorporating mitigating circumstances into reputation assessment. May 2015.

Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of using provenance in e-science experiments. *Journal of Grid Computing*, 5(1):1–25, 2007. ISSN 1570-7873.

M. Momani, S. Challa, and R. Alhmouz. Can we trust trusted nodes in wireless sensor networks? In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 1227–1232, May 2008.

Luc Moreau. The foundations for provenance on the web. *Found. Trends Web Sci.*, 2 (2–3):99–241, February 2010. ISSN 1555-077X.

Luc Moreau. Provenance-based reproducibility in the semantic web. *Web Semantics: Science Services and Agents on the World Wide Web*, 9(2):202–221, July 2011. ISSN 1570-8268.

Luc Moreau and Paul Groth. PROV-Overview: An overview of the PROV family of documents. World wide web consortium note, 2013a.

Luc Moreau and Paul Groth. Provenance: An introduction to prov. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 3(4):1–129, 2013b.

Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, and Laszlo Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, April 2008. ISSN 0001-0782.

Luc Moreau, Paolo Missier, Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. PROV-DM: The PROV data model. World wide web consortium recommendation, 2013a.

Luc Moreau, Paolo Missier, James Cheney, and Stian Soiland-Reyes. PROV-N: The provenance notation. World wide web consortium recommendation, 2013b.

Pablo G Moreno, Yee Whye Teh, Fernando Perez-Cruz, and Antonio Artés-Rodríguez. Bayesian nonparametric crowdsourcing. *arXiv preprint arXiv:1407.5017*, 2014.

Nathan Morrow, Nancy Mock, Adam Papendieck, and Nicholas Kocmich. Independent evaluation of the ushahidi haiti project. Technical report, April 2011.

L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 2431–2439, January 2002.

Victor Naroditskiy, Iyad Rahwan, Manuel Cebrian, and Nicholas R. Jennings. Verification in referral-based crowdsourcing. *PLoS ONE*, 7(10):e45924, October 2012.

Prayag Narula, Philipp Gutheim, David Rolnitzky, Anand Kulkarni, and Bjoern Hartmann. Mobileworks: A mobile crowdsourcing platform for workers at the bottom of the pyramid. *Human Computation*, 11:11, 2011.

Mogens Nielsen, Karl Krukow, and Vladimiro Sassone. A bayesian model for event-based trust. *Electronic Notes in Theoretical Computer Science*, 172:499–521, 2007.

Tom De Nies. Constraints of the prov data model. World wide web consortium recommendation, 2013.

Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z. Gajos. Platemate: Crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 1–12, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1.

Ory Okolloh. Ushahidi, or'testimony': Web 2.0 tools for crowdsourcing crisis information. *Participatory learning and action*, 59(1):65–70, 2009.

Daniel Olmedilla, Omer F Rana, Brian Matthews, and Wolfgang Nejdl. Security and trust issues in semantic grids. In *Semantic Grid: The Convergence of Technologies*, number 05271 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, June 2010.

Galen Pickard, Wei Pan, Iyad Rahwan, Manuel Cebrian, Riley Crane, Anmol Madan, and Alex Pentland. Time-critical social mobilization. *Science*, 334(6055):509–512, 2011.

N. Prat and S. Madnick. Measuring data believability: A provenance approach. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 393–393, January 2008.

Jian-Jun Qi, Zeng-Zhi Li, and Ling Wei. A trust model based on bayesian approach. In PiotrS. Szczepaniak, Janusz Kacprzyk, and Adam Niewiadomski, editors, *Advances in Web Intelligence*, volume 3528 of *Lecture Notes in Computer Science*, pages 374–379. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-26219-0.

D. Quercia, S. Hailes, and L. Capra. Lightweight distributed trust propagation. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 282–291, October 2007.

Daniele Quercia, Stephen Hailes, and Licia Capra. B-trust: Bayesian trust framework for pervasive computing. In Ketil Stlen, WilliamH. Winsborough, Fabio Martinelli, and Fabio Massacci, editors, *Trust Management*, volume 3986 of *Lecture Notes in Computer Science*, pages 298–312. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-34295-3.

Sarvapali D. Ramchurn, Trung Dong Huynh, Matteo Venanzi, and Bing Shi. Collabmap: Crowdsourcing maps for emergency planning. In *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13, pages 326–335, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1889-1.

Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers?: Shifting demographics in mechanical turk. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 2863–2872, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-930-5.

BryanC. Russell, Antonio Torralba, KevinP. Murphy, and WilliamT. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008. ISSN 0920-5691.

Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25, 1995.

Jeffrey Rzeszotarski and Aniket Kittur. Crowdscape: Interactively visualizing user behavior and output. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 55–62, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7.

Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, AAMAS '02, pages 475–482, New York, NY, USA, 2002. ACM. ISBN 1-58113-480-0.

Cristina Sarasua, Elena Simperl, and NatalyaF. Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In Philippe Cudr-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jrme Euzenat, Manfred Hauswirth, JosianeXavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web ISWC 2012*, volume 7649 of *Lecture Notes in Computer Science*, pages 525–541. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35175-4.

Eric Schenk and Claude Guittard. Crowdsourcing: What can be outsourced to the crowd, and why. Technical report, 2009.

Oshani Wasana Seneviratne. Augmenting the web with accountability. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 185–190, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and*

Data Mining, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4.

Aashish Sheshadri and Matthew Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proceedings of the 1st AAAI Conference on Human Computation (HCOMP)*, pages 156–164, 2013.

Edwin Simpson, Stephen J Roberts, Arfon Smith, and Chris Lintott. Bayesian combination of multiple, imperfect classifiers. 2011.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshop*, pages 1–8. IEEE, June 2008.

Jon Sprouse. A validation of amazon mechanical turk for the collection of acceptability judgments in linguistic theory. *Behavior Research Methods*, 43(1):155–167, 2011.

Manolis Stamatogiannakis, Paul Groth, and Herbert Bos. Facilitating trust on data through provenance. In Thorsten Holz and Sotiris Ioannidis, editors, *Trust and Trustworthy Computing*, volume 8564 of *Lecture Notes in Computer Science*, pages 220–221. Springer International Publishing, 2014. ISBN 978-3-319-08592-0.

Jeffrey M Stanton. An empirical assessment of data collection using the internet. *Personnel Psychology*, 51(3):709–725, 1998. ISSN 1744-6570.

Y.L. Sun, Zhu Han, Wei Yu, and K.J.R. Liu. A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, April 2006.

James Surowiecki. *The wisdom of crowds*. Abacus, March 2005. ISBN 0349116059.

Wang Chiew Tan et al. Provenance in databases: Past, current, and future. *IEEE Data Eng. Bull.*, 30(4):3–12, 2007.

John C. Tang, Manuel Cebrian, Nicklaus A. Giacobe, Hyun-Woo Kim, Taemie Kim, and Douglas "Beaker" Wickert. Reflecting on the darpa red balloon challenge. *Commun. ACM*, 54(4):78–85, April 2011. ISSN 0001-0782.

W. T. Luke Teacy, Nicholas R. Jennings, Alex Rogers, and Michael Luck. A hierarchical bayesian trust model based on reputation and group behaviour. December 2008.

W.T.Luke Teacy, Jigar Patel, NicholasR. Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006. ISSN 1387-2532.

Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali Ramchurn, and Nicholas Jennings. Crowdsourcing complex workflows under budget constraints. 2015.

Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D. Ramchurn, and Nicholas R. Jennings. Budgetfix: Budget limited crowdsourcing for interdependent task allocation with quality guarantees. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 477–484, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-2738-1.

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8.

Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A game for locating objects in images. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 55–64, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James Hendler, and Gerald Jay Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, June 2008. ISSN 0001-0782.

P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 25–32, June 2010.

Peter Welinder, Steve Branson, Pietro Perona, and Serge J Belongie. The multidimensional wisdom of crowds. In *Advances in neural information processing systems*, pages 2424–2432, 2010.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in neural information processing systems*, pages 2035–2043, 2009.

W. Willett, S. Ginosar, A. Steinitz, B. Hartmann, and M. Agrawala. Identifying redundancy and exposing provenance in crowdsourced data analysis. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2198–2206, December 2013. ISSN 1077-2626.

Bin Yu and Munindar P. Singh. Detecting deception in reputation management. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 73–80, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8.