# Approximated Provenance for Complex Applications

Eleanor Ainy
Tel Aviv University
eleanora@cs.tau.ac.il

Susan B. Davidson
University of Pennsylvania
susan@cis.upenn.edu

Daniel Deutch
Tel Aviv University
danielde@post.tau.ac.il

Tova Milo
Tel Aviv University
milo@post.tau.ac.il

## Abstract

Many applications nowadays involve the collection of mass data from multiple users, aggregating and manipulating it in intricate ways. The complexity of such applications, combined with the typically large size of collected data, makes it difficult to understand how information was derived, and consequently to asses its credibility, to optimize and debug its derivation, etc. Provenance has been helpful in achieving such goals in different contexts, and we illustrate its potential for novel complex applications such as those performing *crowd-sourcing*. We note that maintaining (and presenting) the full and exact provenance information may be infeasible for such applications, due to the provenance large size and complex structure. We consequently propose initial directions towards addressing this challenge, through a notion of approximated provenance.

## 1. Introduction

Applications that involve the collection, aggregation and manipulation of mass data, interacting with multiple users in intricate ways, have become increasingly popular. A notable example is the flourish of *crowd-sourcing applications* such as Wikipedia, social tagging systems for images, traffic information aggregators like Waze, hotel and movie ratings like TripAdvisor and IMDb. and platforms for performing complex tasks like protein folding via the online game FoldIt.

Several questions arise relating to *how data was derived*: as a consumer, what is the basis for trusting the information? If the information seems wrong, how can the provider debug why it is wrong? And if some data is found to be wrong, e.g. created by a spammer, how can it be "undone", affecting some aggregated result? At its core, the answer to these questions is based on the *provenance* of the collected data and resulting information, that is *who* provided the information in *what* context, and *how* the information was manipulated.


*CONF 'yy*, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

We start by providing two concrete examples of provenance for complex applications, focusing on the crowd-sourcing domain.

**TripAdvisor** aggregates crowd-sourced travel-related reviews (of restaurants, hotels, etc.), and presents average ratings of reviewed destinations. The individual reviews thus stand as the *provenance* of the average rating. TripAdvisor also provides users with several different views of this provenance: viewing ratings and individual reviews, or aggregating them separately for different trip types (e.g. family, couple, or solo). Through such views, we may understand better how the aggregated rating was computed, who provided the information, when they traveled, etc.

**Wikipedia** keeps extensive information about provenance of edited pages. This includes the ID of the user who generated the page and information regarding changes to pages, recording when the change was made, who made the change, a (natural language) summary of the change, etc. Wikipedia also provides several views on this provenance information, e.g. view by page (through clicking on the "View History" tab) or by editor (which gives information about them and all their contributions to Wikipedia pages).

As exemplified above, current systems use a simple solution for provenance management, namely recording provenance and presenting some given views of it upon request. While provenance is indeed useful for *presentation and explanation*, we will discuss below how it is also useful for *maintenance and optimization* of the application. We claim that simply recording all provenance is an inadequate approach for these purposes in this settings, due to the *complexity* of processes and the typically *large size* of provenance information. For instance, in a crowd-sourcing setting, the process is complex since data is collected through a variety of means, ranging from simple questions (e.g. "In which picture is this person the oldest?"or "What is the capital of France?"), to Datalog-style reasoning [7], to dynamic processes that evolve as answers are received from the crowd (e.g. mining association rules from the crowd [4]). The complexity of the process, together with the number of user inputs involved in the derivation of even a single value, lead to an especially large provenance size, which leads to further difficulties in viewing and understanding the information.

We next identify particular challenges in leveraging provenance for presentation and maintenance, while accounting for its size and complexity.

**Summarization.** A consequence of the large size of provenance is the need for ways of summarizing or abstracting provenance information. For example, in heavily edited Wikipedia pages we would wish to view a higher level summary of the changes, such as "entries $x_1, x_2, x_3$ are formatting changes, entries $y_1, y_2, y_3, y_4$ add content, entries $z_1, z_2$ represent divergent viewpoints", or

"entries $u_1, u_2, u_3$ represent edits by robots, entries $v_1, v_2$ represent edits by Wikipedia administrators." As another example, in a movie-rating application we may wish to summarize the provenance of the average rating for "Blue Jasmine" by saying that Europeans aged 20-30 gave high ratings (8-10) whereas American teenagers aged 15-18 gave low ratings (3-5).

**Support of Multidimensional Views.** The environment in complex applications such as those performing crowdsourcing consists of multiple components, each of which provides a perspective through which provenance can be viewed or mined. For example, Wikipedia allows you to see the edit history of a user, as well as of an entry. In TripAdvisor, if there is an "outlier" review (i.e. very negative), it would be nice to be able to see what other reviews the person has provided to be able to calibrate their response. In applications where questions are posed to the crowd (e.g. in FoldIt), a *question* perspective could be used to determine which questions were bad or unclear, i.e. those where many users did not answer the question or where the spread of answers was large, or we may wish to determine why a particular question was asked of a particular user.

**Mining.** Conventional management of provenance typically addresses questions such as "What data or processes contributed to this result (and in what ways)". For instance, in database provenance, provenance may be used to identify whether a piece of data is a component of an input tuple; in workflow provenance it may be used to identify whether the data is a parameter or input data. We aspire to use provenance in additional ways, and in particular to mine it and potentially combine it with contextual information. Returning to the rating application, we may have (contextual) information about users, e.g. age, nationality, and sex; we may also have information about hotels, e.g. chain name, location, etc. By mining the provenance, we may be able to identify demographics of the users (e.g. Europeans aged 20-30, or Americans aged 15-18) that correlate with certain answers (e.g. ranges of scores) for classes of hotels, providing a more in-depth analysis of the application behavior and results.

**Compact Representation for Maintenance and Cleaning.** The ability to mine provenance is useful not only for presentation, but also for data maintenance and cleaning. For example, mining could be used to identify spammers, whose "bad" answers should then be removed from an aggregate calculation. Provenance management techniques should therefore enable hypothetical reasoning, and updated propagation, with respect to the effect of the removal or deletion of certain users, questions and/or answers on the computation. Known techniques for trust assessment, spammers detection etc. may also be employed, taking (some form of) the provenance as input, for such goals. This is of particular importance since the mining of provenance may lag behind the aggregate calculation; for example, detecting a spammer may only be possible when they have answered enough questions, or when enough answers have been obtained from other users. Note that the aggregate calculation may in turn have already been used in a downstream calculation, or have been used to direct the process itself.

***Perspective and Scope*** Provenance models have been extensively studied in multiple lines of research such as provenance for database transformations (see [5] for an overview), for workflows (see [6] for an overview), for the web [1], for data mining applications (initial ideas can be found in [8]), and many others. The basic provenance model that we will rely on in Section 2 is borrowed from these foundations; the challenges listed above, however, were not addressed. As explained above, these challenges are particularly crucial to address in pursuit of a provenance framework for complex applications.

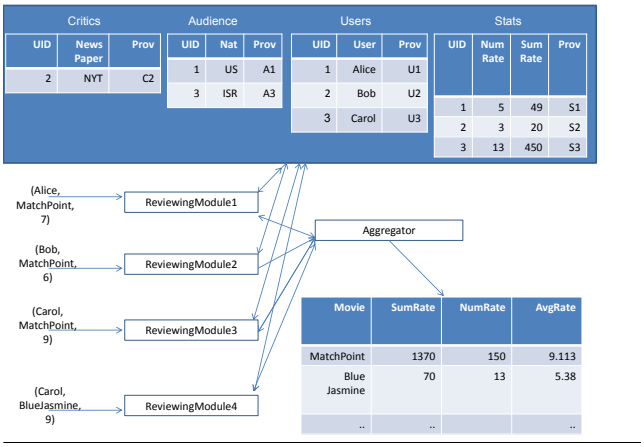


**Figure 1.** Crowdsourcing Application Workflow

## 2. Approximated Provenance

We propose initial directions towards a solution, based on a novel notion of *approximated provenance*. Our starting point is the (exact) provenance model presented in [2], which "marries" database style and workflow-style provenance. We first explain this model. We then explain the need for approximations, identify considerations in defining good approximations, and provide initial computational results.

### 2.1 Workflow and Provenance Model

We start by presenting the notion of workflows that query an underlying database via an example from the crowd-sourcing domain.

EXAMPLE 2.1. *Consider a movie reviews aggregator platform, whose logic is captured by the workflow in Figure 1. Inputs for the platform are reviews (scores) from users, whose identifiers and names are stored in the* Users *table. Users have different roles (e.g. movie critics, directors, audience, etc.); information about two such roles,* Critics *and* Audience*, is shown in the corresponding relations. Reviews are fed through different reviewing modules, which "crawl" different reviewing platforms such as IMDB, newspaper web-sites etc. Each such module updates statistics in the* Stats *table, e.g. how many reviews the user has submitted (*NumRate*), what their average score is (computed as* SumRate *divided by* NumRate*), etc. A reviewing module also consults* Stats *to output a* "sanitized review"*, by implementing some logic. The sanitized reviews are then fed to an aggregator, which computes an aggregate movies scores. There are many plausible logics for the reviewing modules; we exemplify one in which each module "sanitizes" the reviews by joining the users, reviews and audience/critic relation (depending on the module), keeping only reviews of users listed under the corresponding role (audience/critic), and are "active", i.e. submitted more than 2 reviews. The aggregator combines the reviews obtained from all modules to compute overall movie ratings (sum, num, avg).*

An *execution* of such a workflow is a repeated application of the modules, updating the database accordingly.

***Provenance*** By looking only at the database obtained after workflow executions, we obtain only partial information about what happened. For instance, in our example it is hard to understand details behind the overall rating of a given movie, i.e. how was it computed, what kinds of users liked it more than others, etc. Similarly, if we conclude (e.g. using dedicated tools) that a particular user was a "spammer", it may be difficult to estimate the effect of the

spammer on the overall movie rating, hence the rating may need to be re-calculated.

However, using the provenance semirings approach of [2, 3, 9] we may track this information. We start by assigning *"provenance tokens"* to input tuples in the workflow database. These can be thought of as abstract variables identifying the tuples. Provenance is then propagated through manipulations of the data, creating provenance-aware output tuples. There is an intuitive correspondence between algebraic operations in the *semi-module structure* [3] and data manipulation. In particular, $+$ corresponds to the *alternative use* of data (as in union and projection), $\cdot$ to the *joint use* of data (as in join), $\otimes$ is used to "pair" annotations and values for aggregation queries, and $\oplus$ is used to capture the aggregation function. Additional correspondences include the use of 1 for annotation of data that is always available (we do not track its provenance), and 0 for data that is absent. We further introduce special "condition tokens". Intuitively such token $[(d_1 \cdot d_2) \otimes m > 2]$ is kept abstract and can be used in conjunction with other tokens. Given concrete values for $d_1, d_2$ and $m$ one may then test the truth value of the equality.

EXAMPLE 2.2. *Returning to Figure 1, note now the* Prov *column storing provenance of the different tuples. Tuples in the* Users *relation are associated with a token $U_i$, those in the* Audience *relation are associated with $A_i$ etc. The provenance-aware value stored as* SumRate *for the "MatchPoint" tuple would be:*

$$(U_1 \cdot A_1) \cdot [S_1 \cdot U_1 \otimes 5 > 2] \otimes 7 \oplus (U_2 \cdot C_2) \cdot [S_2 \cdot U_2 \otimes 3 > 2] \otimes 6 \oplus (U_3 \cdot A_3) \cdot [S_3 \cdot U_3 \otimes 13 > 2] \otimes 9 \oplus ...$$

*Intuitively, each numeric rating is associated with the provenance of the tuple obtained as the output of the reviewing module, namely the "$U_i$" annotation identifying the user, multiplied by either an $A$ or $C$ annotation, representing the join query applied by the module to check whether the user's role is "critic" or "audience". Each such sub-expression is multiplied by an inequality term serving as a conditional "guard", indicating that the number of reviews recorded for the user is above the threshold (2). Applying aggregation then results in coupling values (numeric reviews) with annotations to form the expression above.*

***Using Provenance*** Provenance may serve as the basis for addressing several of the issues listed in the introduction. In particular, maintaining provenance expressions such as those shown above enable users/administrators of the workflow to *understand* how specific movie ratings were computed. We may also use the provenance expression for efficient *data maintenance and cleaning*: for instance if we realize that user $U_2$ is a spammer, we may "map" its provenance annotation to false (0). By the algebraic laws of the structure, this will have the effect of mapping the entire expression $(U_2 \cdot C_2) \cdot [S_2 \cdot U_2 \otimes 3 > 2] \otimes 6$ to 0, thus disabling its effect on the overall aggregate ratings.

However, storing, or showing to users, the exact and full provenance expression may be infeasible. For example, there may be many reviewers for a movie, thus the provenance expression may be very large. We propose to address this challenge using *approximated provenance*, i.e. we will allow some loss of information to allow compact representation.

## 2.2 Approximations via Mappings

We propose to pursue provenance approximation based on *mappings*. Let $Ann$ be a domain of annotations and let $Ann'$ be a domain of annotation "summaries". Typically, we expect that $|Ann'| << |Ann|$. To model approximation, we define a mapping $h : Ann \mapsto Ann'$ which maps each annotation to a corresponding "summary". This extends naturally to expressions over $Ann$, via the definition of a homomorphism. Essentially, to apply $h$ to a provenance expression $p$ (we denote the result by $h(p)$), each occurrence of $a \in Ann$ in $p$ is replaced by $h(a)$.

EXAMPLE 2.3. *Recall Example 2.2, and let $Ann = \{U_1, ...\} \cup \{A_1, ..\} \cup \{C_1, ...\} \cup \{S_1, ...\}$). Consider a homomorphism that maps all $U_i$ and $S_i$ annotations to 1, all $A_i$ annotations to $A$ and all $C_i$ annotations to $C$ (so $Ann' = \{C, A\}$). Note that the mapping is applied only to annotations and not to numeric values. By applying the homomorphism to the provenance-aware values stored as movie ratings, and applying the congruence rules of the semimodule, we obtain for each movie expressions of the sort:*

$$C \otimes V_C \oplus A \otimes V_A$$

*where $V_C$ ($V_A$) is some concrete value, which is the sum of scores of critics (audience) for which the "threshold" condition holds. The condition itself was mapped to 1 for such users (thus disappeared), and to 0 for others (causing the summand corresponding to their review to disappear).*

*To understand how the above simplification was obtained note that, for instance, for the $SumRate$ value corresponding to MatchPoint we will get*

$$(1 \cdot A) \cdot [1 \otimes 5 > 2] \otimes 7 \oplus (1 \cdot C) \cdot [1 \otimes 3 > 2] \otimes 6 \oplus (1 \cdot A) \cdot [1 \otimes 13 > 2] \otimes 9 \oplus ...$$

*which simplifies to*

$A \otimes 7 \oplus C \otimes 6 \oplus A \otimes 9 \equiv A \otimes 16 \oplus C \otimes 6 \oplus ....$

*Note that we are interested in the* average *score. To this end, a similar transformation will be applied to the values in the $NumRate$ column, resulting in an expression of the same flavor, where $V_C$ would be the* number *of critics reviews on the movie and $V_A$ would be the number of audience reviews (details omitted). These quantities may then be used to compute averages by critics and by audience.*

In the example, we used one possible mapping $h$, that clusters together reviews based on role. In general, however, there may be many possible mappings and the challenge is, given a provenance expression $p$ to (a) define what a "good" mapping $h$ is (and consequently what is a "good" approximation $h(p)$), and (b) find such "good" $h$.

***Quantifying Approximation Quality*** Several, possibly competing, considerations need to be combined.

**Provenance size.** Since the goal of approximation is to reduce the provenance size, it is natural to use the size of the obtained expression as a measure of its quality.

**Semantic Constraints.** The obtained provenance expression may be of little use if it is constructed by identifying multiple unrelated annotations; it is thus natural to impose constraints on which annotations may be grouped together. Example for a simple such constraint is to allow two annotations $a, a' \in Ann$ to be mapped to the same annotation in $Ann'$ (with the exception of 0 and 1) only if they annotate tuples in the *same input table*, intuitively meaning that they belong to the same domain. E.g. in the above example it allows mapping annotations $R_i, R_j$ to the same token, but disallows mapping $R_i, S_j$ to the same token. Other more complicated constraints may be based, e.g., on clustering w.r.t similar same values in possibly multiple attributes, etc.

Further constraints may be employed to further reduce the size of search space.

**Distance.** Depending on the *intended use* of the provenance expression, we may *quantify the distance* between the original and approximated expression. As an example, consider a distance function designed with the intention to use provenance for *recomputation in the presence of spammers*. Recall that provenance

expressions enable this using *truth valuations* for the tokens. Intuitively, specifying that $u_1$ is a spammer corresponds to mapping it to $false$ (and that she is reliable corresponds to mapping the token to $true$), and recomputing the derived value w.r.t this valuation. Such valuation can again be extended in the standard way to a valuation $V : N[Ann] \mapsto \{true, false\}$. Now let $\mathcal{V}_{Ann}$ be the set of all such valuations for a set $Ann$ of annotations. A central issue is how we "transform" a valuation in $\mathcal{V}_{Ann}$ to one in $\mathcal{V}_{Ann'}$. We propose that this will be given by a "combiner" function $\phi$. We can then define the distance between a provenance expression $p$ and its approximated version $h(p)$ as an average over all truth valuations, of some property of $p$, $h(p)$, and the valuation. This property is based on yet another function, whose choice depends on the intended provenance use. For *maintenance*, we may e.g. use a function that returns the absolute difference between the two expressions values under the valuation or, alternatively, a function whose value is 0 if the two expressions agree under the valuations, and 1 otherwise (so the overall distance is the fraction of disagreeing valuations).

**Putting it all together.** Several computational problems are then of interest. Given a provenance expression $p$ (and a fixed $\phi$) we may wish to find a mapping $h$, s.t. the approximated expression $p'$ defined by $p, h, \phi$ (a) satisfies the semantic constraints and (b) either (1) minimizes the distance from $p$ out of all such $p'$ of bounded size, or (2) minimizes the expression size, out of all expressions (obtained through some $h, \phi$) within a bounded distance from $p$. Variants of interest include treating the constraints as "soft" ones, assigning weights to the different factors (size, constraint violations, distance) etc.

***Computing Approximations*** A natural question is how difficult it is to compute such approximations. Our initial results show that even computing the goodness of an approximation defined by a *given* $h$ is already $\sharp P$*-hard*. This is even if the combiner $\phi = +$ and if $p$ includes no tensor elements (no aggregation in the workflow), and even if there are no semantic constraints. On the other hand, we can show an *absolute approximation* algorithm for computing distance between two such provenance expressions. This allows greedy heuristics that, starting from the original set of variables, constructs the homomorphism $h$ gradually, by choosing at each step to identify a pair of variables that leads to minimal increase in distance.

***Incorporating Additional Knowledge*** The search space defined by possible mappings may be quite large in practice. One may add additional constraints that guide the search using additional knowledge. For instance, an estimation of how much we trust different users (derived via dedicated techniques) can be used as an additional constraints over mapping / provenance approximations, disqualifying ones that discard "too many" contributions of trusted users. As another example, additional semantic knowledge such as ontologies can further be used to limit the scope of considered mappings, intuitively restricting the attention to mappings that group together "semantically related" annotations.

## 3. Conclusion

In this short vision paper, we discussed the challenges associated with provenance for complex applications, and proposed the use of a provenance model which "marries" database and workflow-style provenance, with a novel notion of approximated provenance as the basis for addressing these challenges. We note that the way in which the approximated provenance expression is computed essentially involves clustering of annotations – the clusters being defined by the choice of homomorphism – but, unlike standard clustering techniques, here the choice of clusters is guided by the particular provenance expression which is in turn effected by the tracked computation (e.g. which aggregation function was used). The approximation-via-homomorphism approach also has the advantage of compatibility with the underlying semiring-based provenance model, thereby allowing for robust foundations. Adaptation to specific contexts can be achieved through the use of application-depended distance functions.

However, this is just a first step and much remains to be done. In particular: (1) How should we generate the mapping function $h$ for a particular approximation usage? In the examples we have given, this could be done by mining the combined contextual and provenance information to find patterns in the ratings, e.g. demographics of the users that correlate with certain answers. (2) How does the mining process interact with the distance function? (3) How can approximated provenance be used for repair? Deletion propagation and hypothetical reasoning are employed here with only partial information. Efficient implementation of these ideas is also an open challenge.

## References


[1] Provenance working group. `http://www.w3.org/2011/prov/`.

[2] Y. Amsterdamer, S. B. Davidson, D. Deutch, T. Milo, J. Stoyanovich, and V. Tannen. Putting Lipstick on Pig: Enabling Database-style Workflow Provenance. *PVLDB*, 2012.

[3] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *PODS*, pages 153–164, 2011.

[4] Y. Amsterdamer, Y. Grossman, T. Milo, and P. Senellart. Crowd mining. In *SIGMOD Conference*, 2013.

[5] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[6] S. B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD Conference*, pages 1345–1350, 2008.

[7] D. Deutch, O. Greenshpan, B. Kostenko, and T. Milo. Declarative platform for data sourcing games. In *WWW*, pages 779–788, 2012.

[8] B. Glavic, J. Siddique, P. Andritsos, and R. J. Miller. Provenance for Data Mining. In *Theory and Practice of Provenance (TAPP)*, 2013.

[9] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.