Provenance as data mining: Combining file system metadata with content analysis

Vinay Deolalikar Hernan Laffitte
Storage and Information Management Platforms Lab
Hewlett Packard Labs
Palo Alto, CA 95014
{vinay.deolalikar, hernan.laffitte}@hp.com

February 12, 2009

Abstract

Provenance describes how an object came to be in its present state. Thus, it describes the evolution of the object over time. Prior work on provenance has focussed on databases and the file system. The database or file system is enhanced or augmented in order to capture additional information about the historical evolution of document collections, and thus answer the provenance question. We address the question of provenance for unstructured information (i.e., document corpii from file systems) but without any enhancements to the file system. To provide a solution in this setting, we model the provenance problem in such a setting as a problem of data mining. We show that data mining can provide provenance information for repositories of unstructured information, including chains of historical evolution. Thus, we do not require any additions to the file system, and we can operate on legacy documents. Experimental results indicate a strong performance of our approach.

1 The Provenance problem

The explosion of data of all forms in recent times has given rise to many research investigations, one of which is Provenance. Roughly speaking, Provenance is the study of (a) the origins and (b) the history of evolution of data. In other words, the Provenance question for a piece of data asks "How did this data originate, and how did it evolve into its current form?"

The provenance question can be asked in unstructured (filesystem) environments, and in structured (database) environments. We will be concerned only with unstructured provenance in this paper. Concretely, given an evolving universe $\mathscr{U}_t$ of documents at time t, and a finite subset of documents $\{D_i\}_{1\leq i\leq M} \subset \mathscr{U}_T$ from the universe of documents at a time T, the Provenance question is a specific question about the relationship of $\{D_i\}$ to $\mathscr{U}_t$ for $t \leq T$. It asks:

(i) Which documents in $\mathscr{U}_t$ for $t \leq T$ contributed to the evolution of documents $\{D_i\} \subset \mathscr{U}_T$?

(ii) What are the first document(s) in the chain of ideas that culminated in the documents $\{D_i\}$?

(iii) When did the chain of documents that led to $\{D_i\} \subset \mathscr{U}_t$ begin? In other words, what is the lowest value of t for which $\mathscr{U}_t$ contains a document which has contributed to the evolution of a document in $\{D_i\} \subset \mathscr{U}_T$?

We shall refer to the first two questions as *Where Provenance* and the third question as *When Provenance.*

A little thought indicates that the answer to a Provenance query forms a directed acyclic graph (DAG). A node in the DAG represents a document (or a process) and a directed edge from a to b indicates a causal relationship which should loosely be translated as "a resulted, under certain processes, in b". Note that such causal predecessors are by no means unique. Concretely, not just a, but in general $\{a_i\}$ result in b. This means the graph is not a tree. Since relationships capture information flow in the direction of time, the graph must be acyclic.

Why is provenance valuable, and when might one need to query the provenance of a document? Provenance is valuable because it allows us a track an entity to its original sources, and gives us the paths the intermediate document took in order to arrive at the final form that we possess.

2 Previous work in Provenance

Prior work in this area has been deterministic and syntactic. In particular, research has focussed around modifying the file or the filesystem (in unstructured environments) and the database (in structured environments). We rely on the excellent discussion in [14] for more information on these, described below.

The obvious approach to provenance maintenance in files is to include provenance inside the file. Astronomys Flexible Image Transport (FITS) format [3] and the Spatial Data Transfer Standard (SDTS) are examples of this approach. A FITS file header consists of a collection of tagged attribute/value pairs, some of which are provenance. Whenever a file is transformed, additional provenance is added to this header. This approach addresses the challenge of making the provenance and data inseparable, but it introduces other disadvantages. It is expensive to search the attribute space to find objects meeting some criteria. Tools that operate on such files must read and write the headers and be provenance-aware. The validity and completeness of the provenance is entirely dependent upon the tools that process the data. Worse yet, there is no way to determine if provenance is complete or accurate. As shown in Section 2, complete system-level provenance provides functionality unavailable in other systems. A second, and perhaps more important, difference is that LinFS delays provenance collection, performing it at user-level by writing it to an external database. In contrast, PASS manages its provenance database directly in the kernel, providing greater synchronicity between data and provenance; PASTA, our provenance-aware file system, manages both the data and provenance producing a tighter coupling than provided by a separate user-level database.

As mentioned earlier, prior work in provenance has focussed on database provenance, and making storage provenance-aware by enhancing the file system. An example of a system that enhances the file system to obtain provenance is PASS (Provenance aware storage systems) [14]. PASS is a modified Linux kernel that automatically and transparently captures provenance by intercepting system calls in real time. It tracks what files a process read and wrote and records this information together with the data in the same file system. The Lineage File System (LinFS) [12] is most similar to PASS. LinFS is a file system that automatically tracks provenance at the file system level, focusing on executables, command lines and input files as the only source of provenance, ignoring the hardware and software environment in which such processes run.

Trio [16] is to databases what a PASS is to file systems. Trio is a database system that incorporates uncertainty, managing both data and its provenance. It extends SQL to support lineage and accuracy information when requested by a user or application. Trio and PASS are complimentary. Trio focuses on the formalism to describe uncertainty via lineage and operates on tuples within a database framework; PASS focuses on a less structured environment and operates on files.

An architecture for provenance management in databases is also described by [1]. This is for curated databases which are used for archival purposes.

3 Preliminaries for provenance as data mining

From this section onwards, we describe our solution to the provenance problem for universes of unstructured documents using content analysis. Our methods come mostly from the field of text mining, which is the branch of data mining that deals with text documents.

3.1 The vector space model

The first task is to represent each document in the universe as a vector in a suitably high-dimensional vector space of terms that occur in the universe. This representation is called the vector space model [5]. There are many such representations possible. If one were to simply represent each document D by its vector of its term-frequencies (tf), one would obtain the representation

$$\mathbf{v}_D^{tf} := (tf_1, tf_2, \ldots, tf_T),$$

where tf_t refers to the frequency with which the t^{th} term in the universe occurs in document D. The total number of terms in the universe is denoted by T. Notice that tf_t could well be zero.

The major drawback of the above model is that there is no "global" information since each document's vector is computed based only on information obtained from the vector itself. This is easily rectified by weighting all the term frequencies by a global quantity - the inverse document frequency (idf) which is the inverse of the fraction of documents in the universe that contain a given term. The resulting model represents a document D by its $tf - idf$ vector given by

$$\mathbf{v}_D^{tf-idf} := (tf_1 \log \frac{|\mathscr{U}|}{df_1}, tf_2 \log \frac{|\mathscr{U}|}{df_2}, \ldots, tf_T \log \frac{|\mathscr{U}|}{df_T}),$$

where df_t is the number of documents that contain the t^{th} term.

There is still one issue, namely, that different documents have different lengths, leading to D_{tf-idf} vectors of different magnitudes. This is also easily fixed by normalizing all the D_{tf-idf} vectors to have unit magnitude so that they lie on the hypersphere. We will assume all our documents are represented in this normalized fashion.

3.2 Similarity metrics

All clustering algorithms rely on some underlying notion of similarity on the document universe. Formally, similarity is a function $s(\_,\_) : \mathscr{U} \times \mathscr{U} \rightarrow \mathbb{R}^+$ on pairs of documents that satisfies our intuitive idea of what it means for two documents to be "similar". It will, in general, not be a metric.

Various notions of similarity have been proposed and studied in literature. By far, the most commonly used for the purposes of clustering of textual documents is Cosine similarity given by

$$s(D_i, D_j) := \cos(\mathbf{v}_{D_i}, \mathbf{v}_{D_j}) = \frac{\mathbf{v}_{D_i} . \mathbf{v}_{D_j}}{\|\mathbf{v}_{D_i}\| \|\mathbf{v}_{D_j}\|}.$$

We will use this notion of similarity for our clustering algorithms.

3.3 Clustering algorithms

Clustering is the staple of data miners. It is usually the first algorithm run on any data set since it unearths the natural groupings inherent in the data set.

The general problem of of clustering is defined as follows. Given a universe $\mathscr{U}$ of documents, we would like to partition them into a pre-determined number of k subsets (known as clusters) $\{C_1, C_2, \ldots, C_k\}$ such that the documents within a cluster are more similar to each other than they are to documents that lie in other clusters.

There are various approaches to clustering documents, and the clusters produced by different approaches produce different clusters, based on the notion of cohesiveness of document classes used by the approach. Clustering algorithms can be categorized based either on the underlying methodology of the algorithm, or on the structure of the clusters that are output by the algorithm. The first approach results in a division into agglomerative or partitional approaches, while the second leads to a division into hierarchical or nonhierarchical solutions.

Agglomerative algorithms work "bottom-up". They find clusters by engaging in a while loop that initially assigns each object to its own cluster and then repeatedly merges pairs of clusters until a certain stopping criterion is met. These algorithms then differ mainly in how they select the next pair of clusters that are to be merged. Various methods to do this have been proposed, such as group-average [8], single-link [15], complete-link [11] and others.

On the other hand, partitional algorithms work from "top-down". They either find the k clusters directly, or through a sequence of repeated bisections where they create finer clusters at each step. Classical partitional algorithms include k-means [13] and K-medoids [8], among many others.

It is generally accepted [4] that when clustering large document collections, partitional clustering algorithms are preferable due to their relatively low computational requirements. We used a version of k-means clustering.

4 Provenance as data mining

4.1 Intuition

Clearly, a precise mathematical formulation of the questions above requires a quantifiable measure for the contribution of one document to another. Since we insist on not adding any syntax to track provenance, we would like this measure to be computable from the unstructured data itself. The measure we propose is document similarity. Again, there are many measures of document similarity, and they will give different meanings to these questions. But the key idea is that a document that a user accesses before he creates another document is likely to have influenced the creation of the newer document if it bears similarity to it. This is a statement about posteriors - namely, if the newer document is similar to an older one that the user accessed recently, then it is likely that the older document played a role in the thought process that led to the creation of the newer document. Working with such settings is precisely what data mining excels at. This appears to be a reasonable idea, and since most work is incremental, i.e., builds slowly upon previously existing work, it would seem reasonable that at each step of the knowledge creation process, there is a high degree of similarity between what existed and what has just been created based upon that. As one gets farther and farther away from a particular document, the documents that have evolved from it will bear lesser and lesser resemblance to it. This is typical of any evolutionary process.

Thus, when documents undergo typical processes such as edits, merges, derivations (where an author is influenced by an earlier document, but does not necessary use any text from it directly) etc. they do "leave a trace" of the original document. This trace can be captured using any notion of similarity between documents. Clearly, the more intensive a process, the lesser the similarity between the document(s) that existed at the start of the process and those at the finish. But there is always some residual similarity. It follows that if we could trace back document chains along similarity metrics, we would have unearthed its provenance. This is the basic intuition behind our approach.

This leads to a conceptually simple (and as we shall see in the experimental results section, very effective) algorithm. The idea is to extract all the documents that could potentially be in the provenance of the given document by "drawing" a sphere of a certain similarity radius around the document. Since we are using the vector space model, this simply means identifying all documents that lie within a certain angle of the vector representing our document. In practise, this just translates to clustering the document universe at the required granularity.

Once we have unearthed the documents in the provenance of the original document, it only remains to order them by time. For this, we use the *ctime* (creation time stamp) that is available on both UNIX and Windows file systems. We could also use *mtime* (time of last modification).

4.2 Algorithm

We can now describe our algorithm. The first issue we must tackle is the lack of scale-up of clustering algorithms. We get around this by employing the technique of two-stage clustering. In the first stage, we cluster our document universe coarsely so that the average cluster size is N. The number of clusters are this stage depends only on the size of the document universe. For our experimental study, the document universe had size roughly 15K, and our first stage clustering was into 1000 clusters. This can be done once and for all. Now, a provenance query comes in for a document D. The first thing we must do is to estimate a tight upper bound n on the size of the result to the provenance query. For instance, if the query is regarding a research paper, it is unlikely that there have been more than 50 initial drafts, merged earlier papers etc. that occurred in the development cycle of the research paper. Thus, we are looking for clusters that have roughly $n \sim 50$ documents. Based on this desideratum, we cluster the relevant coarse cluster into finer sub-clusters so that the finer sub-clusters have the size we would expect from the provenance result. We look up which of the coarse clusters D falls in, and now *only* re-cluster this particular cluster more finely so that the resulting second stage clusters will have 50 documents on the average.

One must note here that clustering takes into account natural groupings. So if indeed the provenance of D has 75 documents and these are cohesive, it is quite likely that there will be a cluster of size at least 75 that will emerge, even though the average cluster size is 50. Clustering aims to mimic the spatial distribution of document vectors in the high-dimensional space of representation, and so one has only to estimate a rough mean cluster size - the algorithm will take care of variances around this mean as long as the data reflects this grouping.

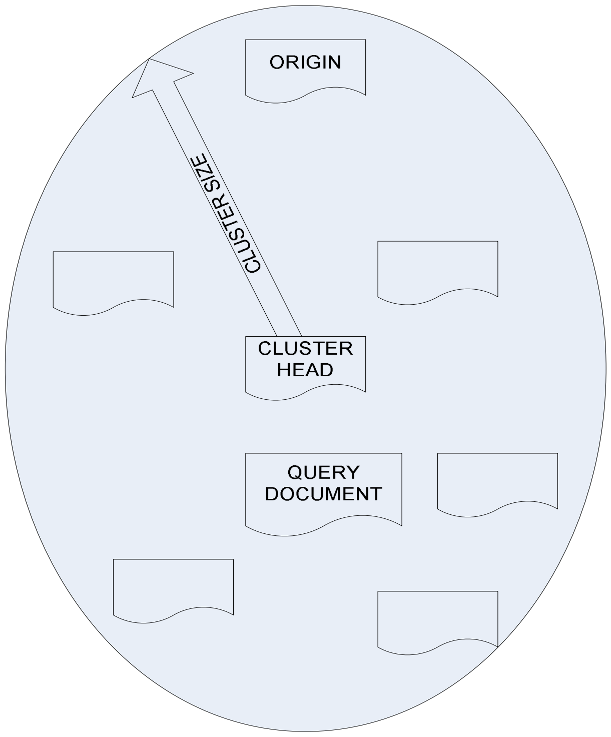

Figure 1: The choice of final cluster size after second stage clustering of the relevant first stage cluster. Note that we cannot guarantee that the query document is the cluster head. We must estimate what cluster size will result in all the provenance documents being inside the cluster.

4.3 Pseudo-code

The following is the pseudo-code for our algorithm. Here N is the estimate for cluster size following coarse clustering, while n is the desired cluster size after finer clustering. Note that N is just a system imposed constraint because clustering the entire document corpus finely can be prohibitive given the issues clustering algorithms have with scaling up.

```
1: Input: Document $D$ and $\mathscr{U}$
2: Determine $N$ based on $|\mathscr{U}|$
3: Do coarse clustering
Ensure: Cluster size $\sim N$
4: Input: Document $D$ and $\mathscr{U}$
Ensure: $D \in \mathscr{U}_t$
5: Determine $n$ based on $D$
6: Identify coarse cluster containing $D$
7: Do fine clustering
```

```
Ensure: Cluster size for cluster $C_D$ containing $D \sim n$
8: Sort documents in $C_D$ by ctime
9: Output sorted list
```

5 Experimental setup and results

The first task in testing the algorithm described above is to gather a "universe" of documents. We were helped in this regard by a storage system that had been built for research purposes at HP Labs, called Jumbo store [2]. The Jumbo storage system provided us with a corpus of research documents from HP Labs whose creation times began in 2003, and it is still operational (so that creation times of files vary over a period of 5 years).

We created a corpus of around 15K documents which did not include source code files, in order to test our algorithm. The statistics of this corpus are given in Tables 1 and 2. From these 15K files, we isolated files that had filenames ending in "_final" or "_submitted" in order to obtain files that were the end nodes of their development. Mostly, these files were the final versions of submitted research papers written by groups of 5-10 researchers. The papers had varying provenances. Some were essentially solo works, where one author began with an initial draft and refined it over time to produce her final submission. Others had more complex provenances with multiple merges and edits, forming fairly complex HDAGs.

We then queried the document corpus using the files with names ending in "_final" or "_submitted" and recorded the resulting provenance results. We compared these to the known ground truth regarding these paper submissions to obtain our precision and recall numbers.

We should stress that we envision this algorithm being used in a closed loop fashion. In other words, we would start with an initial query document, which would result in a provenance result. We would then perhaps wish to query one of the resulting documents to obtain its provenance, and so on.

Table 1: Breakup of documents by type in our document corpus.

Document Type	Number
MS Word (doc)	2349
MS Powerpoint (ppt)	1496
Rich text format (rtf)	468
ASCII (txt)	9124
MS Excel (xls)	1176
Total	14613

There are inherent problems in measuring the performance of such an algorithm. Precision is not hard to compute since we could easily go through the list of provenance results and verify whether they indeed had been in the provenance of the document. Recall is a different matter since it is not easy to rule out that some document that does not appear in the list had something to do with the development of the queried document. We might be able to "verify" with a certain degree of confidence that no *major* intermediate version was left out by the algorithm.

We ran the algorithm on multiple queries. In each case, we were able to obtain very high precision and recall. While we can say that the precision was indeed almost 100%, we cannot make such a statement about recall for the reasons described above. We provide two sample results in the appendix. In both these cases, the precision and recall, as far as we could tell by examining various sources that might reveal ground truth, was nearly 100%.

Acknowledgements

We thank Mark Lillibridge and Brad Morrey for providing us with the research corpus of documents from the Jumbo store for our study.

References

[1] BUNEMAN, P., CHAPMAN, A., and CHENEY, J. 2006. Provenance management in cu-

Table 2: Table of various statistics of size of documents in the corpus. Size is measured in number of words.

Statistic on word counts	Value
Mean word count	11516
Variance	65947586333
Standard deviation	256802
Trimmed Mean(.25)	217
Max	20431991
Min	0
Percentile(25)	22
Percentile(50)	137
Percentile(75)	752
Percentile(90)	2570

rated databases. In Proceedings of the 2006 ACM SIGMOD international Conference on Management of Data (Chicago, IL, USA, June 27 - 29, 2006). SIGMOD '06. ACM, New York, NY, 539-550.

[2] ESHGHI, K., LILLIBRIDGE, M., WILCOCK, L., BELROSE, G., HAWKES, R. Jumbo Store: Providing Efficient Incremental Upload and Versioning for a Utility Rendering Service. HP Labs technical report HPL-2006-144R1. Available online at http://www.hpl.hp.com/techreports/2006/HPL-2006-144R1.html

[3] NOST. Definition of the flexible image transport system (FITS), 1999.

[4] CUTTING, D.R., PEDERSEN, J.O., KARGER, D.R., and TUKEY, J.W. Scatter/gather: A cluster-based approach to browsing large document collections. In Proceedings of the ACM SIGIR, pages pages 318329, Copenhagen, 1992.

[5] SALTON, G., WONG, A., and YANG, C. S. "A Vector Space Model for Automatic Indexing," Communications of the ACM, vol. 18(11), 613620, (1975).

[6] GOLBECK, J. and HENDLER, J. A Semantic Web approach to the provenance challenge. Concurr. Comput. : Pract. Exper. 20, 5 (Apr. 2008), 431-439.

[7] HAN, J., KAMBER, M., and TUNG, A.K.H. Spatial clustering methods in data mining: A survey. In H. Miller and J. Han, editors, Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001.

[8] JAIN, A.K., and DUBES, R.C . Algorithms for Clustering Data. Prentice Hall, 1988.

[9] JAIN, A.K., MURTY, M.N., and FLYNN, P.J. Data clustering: A review. ACM Computing Surveys, 31(3):264323, 1999.

[10] GROTH, P., JIANG, S., MILES, S., MUNROE, S., TAN, V., TSASAKOU, S., AND MOREAU, L. D3.1.1: An architecture for provenance systems. Tech. rep., University of Southampton, Feb. 2006.

[11] KING, B. Step-wise clustering procedures. Journal of the American Statistical Association, 69:86101, 1967.

[12] Lineage File System. http://crypto.stanford.edu/Xcao/lineage.html

[13] MacQUEEN, J. Some methods for classification and analysis of multivariate observations. In Proc. 5th Symp. Math. Statist, Prob., pages 281297, 1967.

[14] MUNISWAMY-REDDY, K.-K., HOLLAND, D. A., BRAUN, U., AND SELTZER, M. Provenance-aware storage systems. In Proceedings of the 2006 USENIX Annual Technical Conference (June 2006).

[15] SNEATH, P. H., and SOKAL, R. R. Numerical Taxonomy. Freeman, London, UK, 1973.

[16] WIDOM, J. Trio: A System for Integrated Management of Data, Accuracy, and Lineage. In Conference on Innovative Data Systems Research (Asilomar, CA, January 2005).

Table 3: The output of the Provenance algorithm. It gives a list of documents sorted by the time of creation. The date and time of creation, obtained from the *ctime* metadata that is available from the filesystem, both in UNIX and Windows. The actual provenance for the paper is given in Fig. 2. The precision and recall of this particular result were almost 100%. Notice that the provenance spanned a period of around 3 years, and all of it was recovered by the algorithm. Both *where* and *when* provenance are answered. The user "mdl" was lead author on the paper, which is also reflected in the provenance results.

Date created	Time created	File name and path
2005-09-08	16:03:29	root_C/Documents/Double Hash/Project_0.doc
2006-08-31	10:47:57	mdl/Project_2/seed/paper/pre-submission versions/urs10.doc
2006-09-01	14:03:06	mdl/Project_2/seed/paper/pre-submission versions/urs16.doc
2006-09-01	14:09:20	mdl/Project_2/seed/paper/pre-submission versions/urs17.doc
2006-09-01	14:16:16	mdl/Project_2/seed/paper/pre-submission versions/urs18.doc
2006-09-01	16:16:14	mdl/Project_2/seed/paper/pre-submission versions/urs19.doc
2006-09-03	18:08:27	root_C/Documents/Project_1/urs2.doc
2006-09-03	18:08:51	root_C/Documents/Project_1/urs3.doc
2006-09-03	18:09:07	root_C/Documents/Project_1/urs4.doc
2006-09-04	15:09:35	mdl/Project_2/seed/paper/pre-submission versions/merged10.doc
2006-09-04	18:19:48	root_C/Documents/Project_1/merged15.doc
2006-09-04	18:20:10	root_C/Documents/Project_1/merged16.doc
2006-09-04	18:36:25	mdl/Project_2/seed/paper/pre-submission versions/post1.doc
2006-09-04	20:46:57	mdl/Project_2/seed/paper/pre-submission versions/post5.doc
2006-09-04	23:47:50	mdl/Project_2/seed/paper/pre-submission versions/post8.doc
2006-09-04	23:49:11	mdl/Project_2/seed/paper/pre-submission versions/post9.doc
2006-09-05	08:12:01	mdl/Project_2/seed/paper/pre-submission versions/post10.doc
2006-09-05	09:20:01	mdl/Project_2/seed/paper/pre-submission versions/post11.doc
2006-09-05	13:52:07	root_C/Documents/Project_1/merged10.doc
2006-09-05	13:58:20	root_C/Documents/Project_1/related.doc
2006-09-05	14:03:21	root_C/Documents/Project_1/Related work.doc
2006-09-05	19:34:16	root_C/Documents/Project_1/References.doc
2006-09-06	11:00:01	root_C/Documents/Project_1/Conclusion.doc
2006-09-06	16:42:57	mdl/Project_2/seed/paper/submission/Conf_Name submission.doc
2006-09-12	16:25:59	mdl/Project_2/seed/paper/submission/Conf_Name submission before blinding2.doc
2006-09-12	16:38:04	mdl/Project_2/seed/paper/submission/Conf_Name submission before blinding3.doc
2006-10-10	10:46:28	mdl/Project_2/seed/paper/abstract.rtf
2006-10-10	11:04:28	mdl/Project_2/seed/paper/short-abstract.txt
2006-12-16	00:12:43	mdl/Project_2/seed/paper/Conf_Name final copy.doc
2006-12-16	14:26:03	root_C/Documents/Project_1/Conf_Name final draft-final.doc
2007-11-27	09:02:54	mdl/Published/Internal/Jumbo/abstract.txt

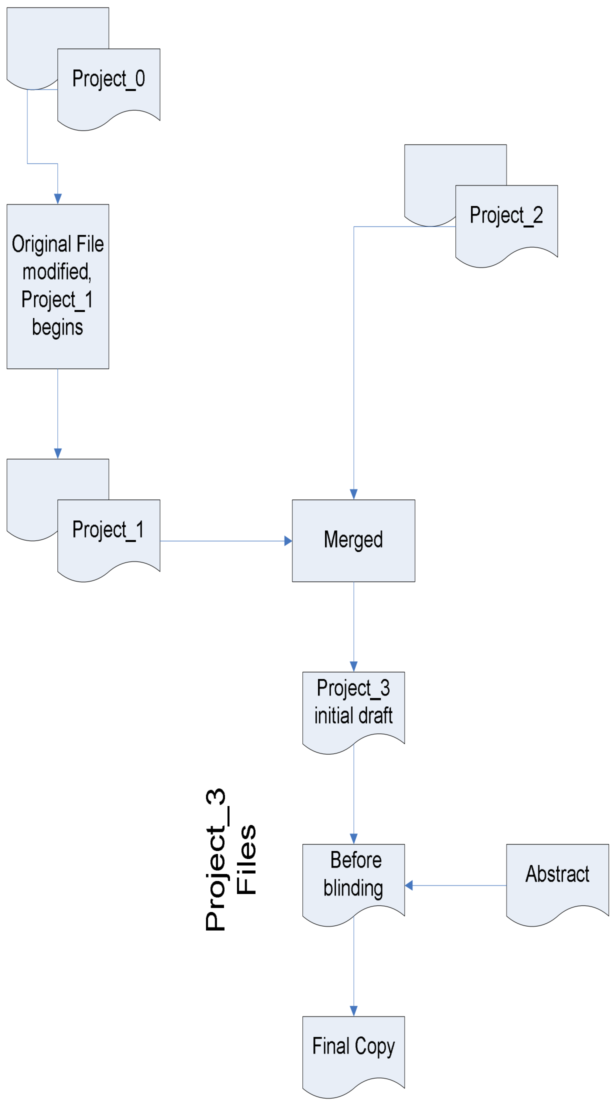

Figure 2: The provenance DAG (directed acyclic graph) captured by Table 3. The provenance of the query document, which was the final version of a conference paper, included edits, derivations, and merges. The project had remote roots in Project_0, from which a process of derivation resulted in the first files for Project_1. These were edited, and later merged with files from Project_2 to give the initial files of Project_3. These were modified and sections merged, to produce the final document. These flows were successfully captured by the provenance algorithm.

Table 4: The output of the Provenance algorithm. Names have been obfuscated. There were 3 users who contributed to this paper, and this is reflected in the provenance results. The date and time of creation, obtained from the *ctime* metadata that is available from the filesystem, both in UNIX and Windows. The precision and recall of this particular result were almost 100%. The provenance here spanned roughly a year, and was recovered by the algorithm. Both *where* and *when* provenance are answered.

Date created	Time created	File name and path
2003-10-16	16:41:06	usrname_1/p5/talks/usrname_2-final-talk.ppt
2003-10-16	16:41:07	usrname_1/p5/talks/Conf_Name2003/Conf_Name-draft-1.ppt
2003-10-16	16:41:07	usrname_1/p5/talks/Conf_Name2003/Conf_Name-draft-2.ppt
2003-10-16	16:41:08	usrname_1/p5/talks/Conf_Name2003/Conf_Name-draft-4.ppt
2003-10-16	17:00:28	usrname_1/p5/talks/Conf_Name2003/Conf_Name-draft-5.ppt
2003-11-10	18:11:06	usrname_1/p5/talks/Conf_Name2003/Conf_Name-draft-6.ppt
2003-11-10	18:11:07	usrname_1/p5/talks/Conf_Name2003/usrname_3-Conf_Name-final.ppt
2003-11-10	22:21:50	usrname_1/p5/talks/usrname_1-ietl-2003.ppt
2004-01-30	15:46:42	usrname_1/p5/talks/future-12-03.ppt
2004-05-24	10:39:46	usrname_1/p5/talks/Conf_Name2003/usrname_1-Conf_Name.ppt
2004-11-09	17:20:14	usrname_1/p5/presentation/wisp/wisp.ppt
2004-11-15	16:59:34	usrname_1/p5/presentation/wisp/wisp-posted.ppt