The Probabilistic Provenance Graph

Nwokedi Idika, Mayank Varia, and Harry Phan
MIT Lincoln Laboratory
{*nwokedi.idika, mayank.varia, harry.phan*}*@ll.mit.edu*

Abstract

Previous provenance models have assumed that there is complete certainty in the provenance relationships. But what if this assumption does not hold? In this work, we propose a probabilistic provenance graph (PPG) model to characterize scenarios where provenance relationships are uncertain. We describe two motivating examples. The first example demonstrates the uncertainty associated with the provenance of an email. The second example demonstrates and characterizes the uncertainty associated with the provenance of statements in documents.

Keywords

provenance; graphs; trust; linguistics

1. Introduction

Although researchers have noted that the entities associated with a provenance system may have some uncertainty [1], previous work on provenance collection and storage has assumed that the provenance information itself is certain [2]–[5]. This certainty about provenance helps reduce the uncertainty associated with the data that the provenance describes [6]. However, there are plausible scenarios for which this assumption does not hold, and, in this work, we present two such scenarios.

In the first scenario, we model the provenance of an email in an employee's inbox. We demonstrate how the different core W3C PROV relationships are actually uncertain in this context (we use PROV and PROV-DM interchangeably as DM stands for "Data Model"). In the second scenario, we examine in detail how the provenance associated with the content of documents may be uncertain, and we show how this uncertainty may be modeled.

In this work, we introduce the probabilistic provenance graph (PPG) to accommodate scenarios like the aforementioned, where we are uncertain about the provenance relationships.

There are two major benefits to our approach. First, it enables users to reason about the behavior of systems under inspection more precisely. Provenance collection mechanisms do not always possess enough information to have sufficient certainty of events; in these circumstances, the PPG can help capture those uncertainties. Second, in a networked environment, one no longer requires total control or willing participants in order to reason about the provenance of entities (see Section 5).

Our contributions include the following:

- Probabilistic extensions to the W3C PROV provenance model,
- The PPG model,
- A meta-algorithm for identifying relationships within a "linguistic provenance graph," which is a type of PPG, and

In the next section, we cover relevant provenance models and focus on the most recently proposed provenance model PROV. In Section 3, we motivate the need for the PPG and the corresponding extensions to the PROV model. In Section 4, we detail the PPG model. Then, in Section 5, we discuss what linguistic provenance is and the prototype system we use to obtain it. In the final section, we give concluding remarks.

2. Related Work

The "Open Provenance Model" (OPM) [7] was the first widely adopted specification for storing and

DOI 10.1109/SPW.2013.27

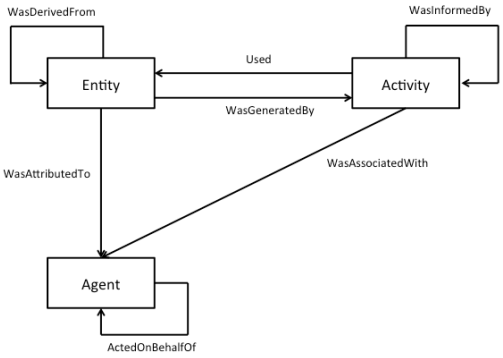

Figure 1. Summary of the "core" node and relationship types in the PROV model; image replicated from [8]

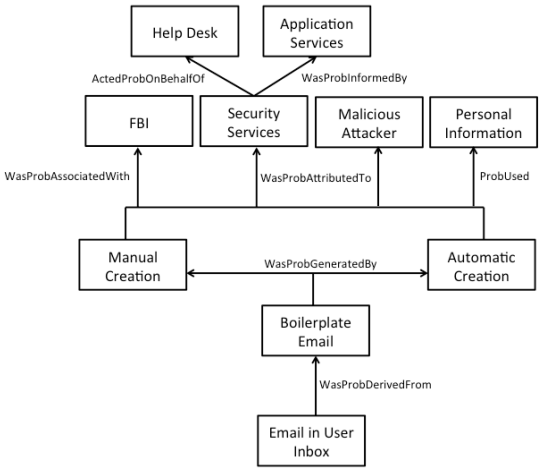

Figure 2. A user's perspective on a situation regarding an suspicious email received via probabilistic PROV relationships

modeling provenance. Building on this effort, the W3C standard body formed a provenance working group [8] that established a standard data model for provenance information called PROV. PROV represents provenance as a directed acyclic graph, or DAG. Vertices in the PROV model may correspond to entities, activities, or agents. The edges connecting these items are annotated with the possible relationships between the entity, activity, and/or agent. For a summary of the core relationships and entities in the PROV model, see Figure 1. Note that the term "entity" is broad and may therefore also refer to an agent or activity.

The working group has created an RDF description for the PROV model, and has described a syntax for encoding provenance in XML and other formats. The working group has also developed algorithms for searching and querying provenance that follow the PROV model. As a result, PROV appears to be the most mature and actively developed standard for representing provenance, so we use it as the base model for our extensions. Given that PROV leverages OPM, our extensions could also be applied to the OPM model, if one preferred that standard. In Section 4 we describe how the core PROV relationships should be extended to represent the PPG model.

In [1], Widom proposes the Trio Data Model (TDM). Within the context of a database, TDM specifies ways for capturing uncertainty about data and its related provenance. Although it was created for databases, TDM is general enough to represent uncertainty in provenance beyond databases. We show this in Section 4. However, provenance (referred to as *lineage* in [1]) is assumed to have a confidence, or a belief probability, of 1 because no "clear use for < 1 confidences" could be seen for the application of interest. In subsequent work on Uncertain-Lineage Databases (ULDBs) [6] that utilizes TDM, lineage is assumed to be deterministic and certain.

Our work differs from that of ULDBs. While ULDBs are concerned with the uncertainty of *the data* in the database, we are concerned with the uncertainty of *the provenance* of that data.

3. The Provenance of an Email

In this section, we motivate the use of a probabilistic provenance graph (PPG) with an example. The core node and relationship types for PROV are depicted in Figure 1. Each core relationship assumes complete certainty; however, in practice, there can be uncertainty for each relationship. We demonstrate this fact in the example that follows.

A user, at his place of work, has noticed that he is unable to sign into some software applications provided by his organization. Complicating things further, he has been unable to get help from his organization's help desk department. Shortly thereafter, he received an email claiming that he has committed a serious

security violation. Moreover, the email mentions that the FBI provided the Security Services department with the tip that ultimately led to the identification of the user's security violation. The email states that its claims are substantiated by personal information about the user and if the user wishes to refute these allegations, the user must provide additional personal information.

Having never received such an email before, the user is uncertain of the legitimacy of this email. The user's uncertainty is depicted by the PPG in Figure 2. Note that the figure is a single representation of uncertainty. There could be other equally valid alternative representations. We have chosen this particular representation to demonstrate the uncertainty that could be associated with each of the core PROV relationships.

3.1. The Semantics of the PPG in Context

Now that we have established the situation, we are now ready to begin interpreting the graph in Figure 2 from the end user's perspective.

- The end user has incomplete information about whether the email message received "WasDerivedFrom" some boilerplate email that is typically sent out to end users.
- The end user is unsure whether the email "WasGeneratedBy" a human ("Manual Creation") or a machine ("Automatic Creation").
- Because of the style, tone, and requests made in the email, the end user has uncertainty about whether the email "WasAttributedTo" his "Security Services" department, or some "Malicious Attacker" performing some type of social engineering attack.
- Due to the email's claim of personal information and request for more personal information, the end user wonders whether or not his "Personal Information" was "Used" in constructing the email in question.
- The end user is uncertain about whether his inability to log into some applications "WasInformedBy" information from "Security Services."
- When the help desk refused to help the end user, he began to wonder whether the "Help Desk" had "ActedOnBehalfOf" itself or the Security Services department.

Certain PROV-DM Relations	Probabilisitic PROV-DM Relations
WasGeneratedBy	WasProbGeneratedBy
Used	ProbUsed
WasInformedBy	WasProbInformedBy
WasDerivedFrom	WasProbDerivedFrom
WasAttributedTo	WasProbAttributedTo
WasAssociatedWith	WasProbAssociatedWith
ActedOnBehalf	ActedProbOnBehalf

Table 1. Core PROV-DM certain and probabilistic relations

- Because the FBI was mentioned in the email, the user wonders if the FBI "WasAssociatedWith" the security incident.

3.2. Extensions to the PROV Model

With this small example, we have demonstrated that there can be uncertainty associated with each of the seven core PROV relationships that are depicted in Figure 1. To accommodate uncertainty in provenance relationships, we augment the PROV model by adding a probabilistic variant of each core PROV relationship. These duals are given in Table 1. (The non-core relationships can be similarly extended.) Each relationship is given a probability attribute to capture the degree of confidence in the relationship. How the probabilities are obtained will vary with the application domain. Anything from precise equations, to expert knowledge, to some combination thereof is possible.

4. The Probabilistic Provenance Graph

A probabilistic provenance graph (PPG) is a directed acyclic graph $G_t = (V, E)$ where V corresponds to the set of vertices and E corresponds to the set of edges at time t. A node $v \in V$ corresponds to state in the PPG, which may be an artifact, an agent, or an activity. An edge $e \in E$ corresponds to an event in the PPG. There is a "relationship function" $r : E \rightarrow T$ that identifies the type of event that is represented by each edge; that is, $r(e)$ denotes a probabilistic PROV-DM relation in Table 1. Only edges with the same parent and the same type from r may belong to the same sample space.

Have "time" as part of the model enables the capturing of possible changes in provenance over time due to changes in the user's knowledge. We expect that, in general, these changes will be enhancements

to the original provenance graph resulting in additions or removals of nodes/edges. When the provenance of a process is well understood and its provenance graph satisfactorily represents the process, t can be ignored. We will use t only if time is important to the discussion.

Additionally, we associate a family of (potentially independent) partial functions $\{f_1, f_2, f_3, ..., f_n\}$ to a PPG, where $1 \leq n \leq |E|$. These functions are used to model the beliefs in the assignments of the relationship function r. Specifically, each $f_i : E_i \rightarrow [0, 1]$, for some subset of edges $E_i \subseteq E$. The function output $f_i(e)$ corresponds to the belief probability that edge e belongs to $r(e)$. Typically, each function f_i will be applicable to a proper subset of edges in the graph.

For example, in Figure 2, the "WasProbGeneratedBy" edges may be associated with a function f_1 that is represented by a Logistic Regression Authorship Identification classifier, with its outputs representing probabilities. However, this Authorship Identification classifier representation of f_1 would not be useful for representing, say, function f_2 that is associated with the "ProbUsed" relationship that exists between the "Automatic Creation" and "Personal Information" nodes. On the other hand, if f_1 was represented as the frequency of each event ("Manual Creation" vs. "Automatic Creation"), then it is possible that f_1 could also be used for the "ProbUsed" relationship.

This relationship representation in Figure 2 demonstrates another aspect of the PPG—*implicit edges*. Because "ProbUsed" has no siblings that are also the same relation explicitly represented in the graph, under the PPG model, there is an implicit sibling to "ProbUsed" with the same relation that captures the complement of the destination node, which in this case is "Not Personal Information." Implicit edges exist for economy of representation. A user can choose to explicitly represent the implicit edges in the graph. For relationships that have siblings of the same relationship, any implicit nodes should be listed explicitly to enhance clarity.

We have previously stated that the partial functions associated with a PPG *may* be independent because some functions may be dependent. For instance, if several functions refer to the same sample space, and there are $k > 0$ possible events in this sample space, then there are $k-1$ degrees of freedom. Thus, one function may be completely determined by the other functions that specify the probabilities for the other events in the sample space. More generally, dependency may exist because a user finds that one or more functions in the family can be represented in terms of other functions in the family.

The PPG model is only well-defined if a probability assignment is given to every edge in the graph; that is, $\cup_i E_i = E$. Practically, however, this may not be feasible or desirable. Regarding feasibility, a user may have insufficient knowledge or data to fully and accurately represent the PPG of interest. Regarding desirability, a user may have queries that deal only with a subgraph of G and therefore would only need to represent that subgraph. Thus, in practice what will generally be used is a PPG $G' = (V'' \cup V', E'' \cup E')$ such that $V' \subseteq V$, $E' \subseteq E$, $V'' \cap V = \emptyset$ and $E'' \cap E = \emptyset$. The introduction of V'' and E'' suggests that nodes and edges that are not part of G may be part of G' used in practice. Returning to our email example, if a user was interested in emails that were attributed to individuals outside of his department, he might be interested in a graph that differs from that given in Figure 2. If we take G to be the PPG in Figure 2, then the user may create a new PPG G' by making $V'' = \{$"External to Department"$\}$ and $V' = \{$"Email in User Inbox"$\}$. Continuing with the definition of G', the user would also define the following edge set: $E'' = \{($"External to Department", "Email in User Inbox"$)\}$ and $E' = \emptyset$. To complete the definition, the user would define r for $e \in E''$ to map to "WasProbAttributedTo."

Although f_i produces belief probabilities, G is not required to be a bayesian network. The primary benefit of bayesian networks is that they are well understood and have standard algorithms for inference. However, the bayesian graphical depiction may not be what we want to display to the end user as it could be more difficult for an end user to visually interpret relative to the PPG. So one approach would be to utilize a bayesian representation for performing inference, and utilize the PPG representation to end users. In the next section, we discuss how one might obtain a bayesian network from an alternative PPG representation. Then, we demonstrate how to represent the PPG under the well-known Trio Data Model (TDM).

4.1. Formulating the PPG as a Bayesian Network

A PPG is likely to conform to how the user of a provenance system wants to capture metadata. That is, a PPG derived from an audit-based process will probably follow some workflow trace. However, graphically

this causal flow may not be maintained if the user desires conformance to the PROV model. In bayesian networks, it is preferable to represent probabilistic graphs as causal relationships where the causal node is at the tail of the edge and the effect node is at the head of the edge [9]. Moreover, there may be nodes in the original PPG that may not appear in the bayesian network version of the PPG because they correspond to outcomes of one or more random variables (vertices) in the graph. We demonstrate the transformation of an original PROV-compliant PPG (Figure 2) to a Bayesian PPG (Figure 3).

The two nodes, "Manual Creation" and "Automatic Creation," are merged into the "Machine vs. Human" node given the two nodes' existence in the same sample space and their mutual exclusivity. For the same reason, the "Security Services" node and the "Malicious Attacker" nodes are merged into a single node "Catalyst" in the bayesian network. Any edges that were associated with the old nodes are now associated with the newly created node. In our example, the following relationships put causes at the tail of an edge and effects at its head: "ActedProbOnBehalfOf" and "WasProbInformedBy." All other relationships reverse the position of the causal and effect nodes. So for these relationships, we reverse their edge direction. The output of this process is Figure 3.

In general, the process for transforming the PROV-compliant PPG into a bayesian PPG is given by the following steps:

1) If two or more nodes have the same parent and relationship, for each such parent, merge its children creating a new single node x.
2) For each relationship that has the causal node at the edge head and an effect node at the edge tail, reverse the edge direction.

Step 1 defines a sample space for x by restricting the nodes that comprise it to those that have the same parents and relationships. Step 2 ensures that the graph has a causal flow direction. By using this general process, we can leverage the standard inference methods for bayesian networks and learn more from the PPG. Note that this bayesian graph is not unique– there are other legitimate representations.

4.2. Using the Trio Data Model (TDM) to Represent the PPG

In TDM [1], lineage (i.e., provenance) is given by the following tuple (tupleID, derivation-type, time,

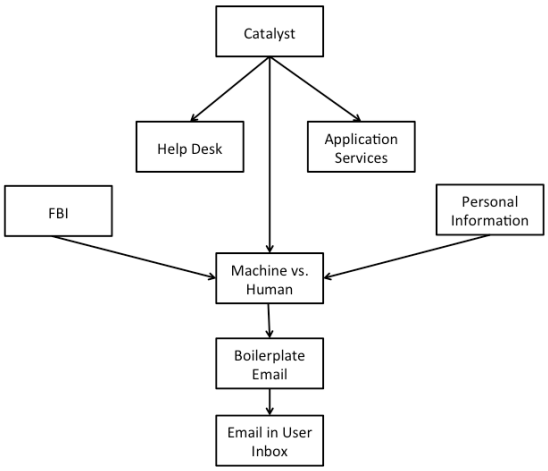

Figure 3. A bayesian network representation of the PPG in Figure 2

how-derived, lineage-data). The tupleID is a unique identifier for the tuple. The derivation type includes the following: query-based, program-based, update-based, load-based, and import-based. Each derivation type may have different values for how-derived and the lineage-data attributes [1].

We do the following to represent a PPG in TDM:

1) Define a data type for each vertex type in a PPG: entity, activity, and an agent.
2) For each probabilistic PROV-DM relation, create a lineage relation of the same name.
3) Extend the lineage schema with an additional "confidence" attribute.

A tuple in the lineage relation then corresponds to an edge in the PPG. The confidence attribute would correspond to the belief probability associated with the provenance information represented in the rest of the tuple. Although [1] notes some distinctions between approximations and confidence, we do not make such distinctions. We use confidence, belief probabilities, in a sufficiently general manner to represent confidence and approximations.

5. The Provenance of Statements in Documents: Lincoln Laboratory Plagiarism for Provenance System (LLPlā)

When assessing the trustworthiness of a document, an analyst may wish to know source of the statements

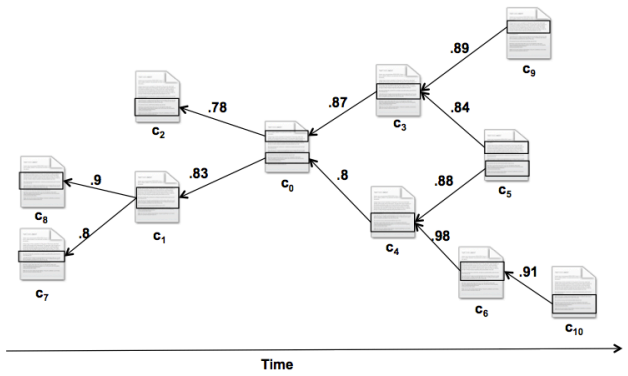

Figure 4. Example probabilistic provenance graph

in that document. This is of particular importance for documents that have no citations. For documents containing citations, an analyst would like some confidence that the citations are relevant, correct, and comprehensive.

What the analyst really desires is what we term the *linguistic provenance* of the statements in the document. Linguistic provenance provides the provenance for words, phrases and statements in a given document. Assuming there is a corpus of documents C and a document of interest d, a linguistic provenance graph will reveal all relationships between documents that contain sufficiently similar sections of information.

The linguistic provenance graph is an instance of a PPG. Under linguistic provenance, nodes map to document sections of text, whereas edges have the same representation and semantics as edges in the more general PPG. Of the core probabilistic PROV relationships, "WasProbDerivedFrom" is the one needed for linguistic provenance graphs. An example PPG is given in Figure 4 and is superimposed on the related documents for context. The rectangles in the documents correspond to the nodes of the PPG. The position of the node with respect to the timeline indicates when the document content was created. The numbers on the edges correspond to the probabilities of sections of text being derived from other sections of text. So, in Figure 4, the node in c_9 is derived from the node in c_3 with an 89% probability. We now depict the Lincoln Laboratory Plagiarism for Provenance System (LLPlā), which is an approach we have developed to realize linguistic provenance graphs.

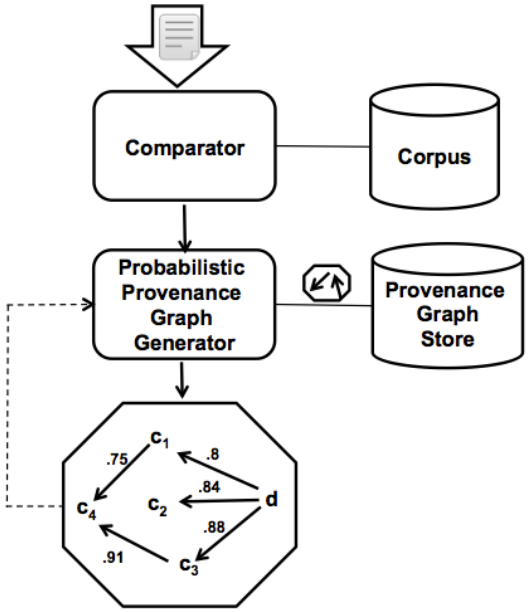

Figure 5. Evaluation architecture

5.1. Prototype Architecture

The prototype architecture depicted in Figure 5 contains four components: the comparator, the document corpus, the PPG generator, and the PPG store. We discuss each component below.

Comparator. The Comparator contains one or more plagiarism detection algorithms implemented as python scripts. When a document d comes into the Comparator, it is compared against the documents in the "Document Corpus." As a plagiarism detection algorithm produces plagiarism detections, the detections are sent to the PPG Generator. When the plagiarism detection algorithms are done with d, the Comparator stores the document in the Document Corpus. At the conclusion of this process, d is part of the corpus.

Document Corpus. The Document Corpus is a store of all of the documents of interest in the system. Our prototype stores documents in the file system and Neo4j.

Probabilistic Provenance Graph Generator. The PPG Generator receives the detections sent from the Comparator. The PPG generator can also produce a

PPG as its output. In our prototype, this effect is achieved by using Ext JS and JSPlumb [10].

Probabilistic Provenance Graph Store. The PPG Store maintains the PPGs. Because users can modify a PPG (i.e., create a new PPG with modifications), thereby creating a new PPG local to that user, users can have different derivative views of the same global PPG. Via this mechanism, users are able to modify probabilities based on their own knowledge or assumptions and obtain PPGs that do not affect the PPGs that everyone else uses. In our prototype, PPGs are stored in Neo4j [11].

5.2. Similar Text Detection with Pipeline Approach

A key aspect to the performance of LLPlā is the plagiarism detection algorithms used. These detections form that relationships that ultimately connect the nodes of the linguistic provenance graph. So now, we focus on a meta-algorithm for detecting more types of plagiarism cases.

Different plagiarism detection algorithms can exhibit different properties. Some of these properties may be desirable under different circumstances. For instance, some plagiarism detection algorithms may run faster than others. Some plagiarism detection algorithms may exhibit high precision scores but lack in recall scores for particular types of plagiarism cases. In fact, this variation can even be observed by a single algorithm when run with different parameters. When working with a labeled dataset, "ground truth" is known, so one can tune parameters for that dataset. However, when operating on data where ground truth is unknown, a user may be interested in "casting a wide net."

For instance, state-of-the-art plagiarism detection algorithms have heuristic-based lower bounds on the type of plagiarism cases they will be able to detect. Because it is generally impossible to know how long the plagiarism cases will be in the document, it may be desirable to search for different sized plagiarism cases in the documents being compared. This allows for detection of plagiarism cases that would have been missed if only one algorithm were used.

We are interested in detecting the various forms of plagiarism cases that might appear in an unlabeled corpus. To accomplish this aim, we propose a pipeline approach where a series of algorithms with different strengths are sequenced and run against documents of interest. This approach is depicted in the algorithm below.

```
function RUNPIPELINE(algos, d_1, d_2)
    acc_d ← ∅
    for a ∈ algos do
        detections ← a.detailed_compare(d_1, d_2)
        if |detections| > 0 then
            acc_d ← acc_d ∪ detections
            d_1, d_2 ← remove_sections(detections, d_1, d_2)
        end if
    end for
    return acc_d
end function
```

The list "algos" contains the algorithms to be applied to the documents d_1 and d_2. The variable acc_d accumulates the plagiarism detections found between d_1 and d_2. The method "detailed_compare(...)" performs the detailed comparison of d_1 and d_2 based on the implementation provided by algorithm a. The function "remove_sections(...)" removes the sections of text from both documents that are involved in a plagiarism detection. This removal ensures that the next algorithm in the pipeline does not detect sections of text that have been detected by preceding algorithms.

The sequence of the algorithms in "algos" should be chosen carefully as the order of the algorithms affects detection performance. There are some heuristics that should be adhered to if this pipeline approach is used. First, the most precise algorithms should be placed toward the front of the pipeline and the least precise algorithms should be placed toward the back of the pipeline. Also, algorithms that require more time to process documents should be placed toward the end of the pipeline. The detections for these algorithms can be weighted to decrease the confidence in them when included in the PPG. Algorithms requiring more running time are generally more sophisticated algorithms. That is, such algorithms will be able to detect more complex forms of plagiarisms. In this scenario, as detections are found, they would need to be sent to the PPG generator. In this way, the provenance associated with a document can be updated as new relationships are discovered. To accomplish this with the current algorithm, we would simply replace the acc_d assignment with a "send(detections)" function and there would be no need to return a value.

Although the algorithm suggests a sequential ap-

proach, nothing prevents the realization of a parallel approach. In this scenario, the documents of interest would be given to at least two different algorithms at once. To minimize duplication of effort, we would want to ensure that the algorithms have very little overlap in the type of plagiarism cases they detected.

6. Conclusions

In this work, we have explained the PPG model and motivated its use. We have discussed how to convert a PROV-compliant PPG into a bayesian PPG. We have proposed extensions to the PROV model to accommodate uncertainty. We have detailed a meta-algorithm for utilizing plagiarism detection algorithms to generate linguistic provenance graphs. While we use text similarity and document creation date to determine edge probabilities in linguistic provenance graphs, these items are just two features that should feed into a more elaborate classifier. Features like the author of the documents, the author's affiliation, the topic of the document, the style of writing could all be additional features to produce more accurate edge probabilities.

Acknowledgments

This work is sponsored by the Assistant Secretary of Defense for Research & Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

The authors would like to thank the following people for their assistance in this work: Suresh Damodaran, Yaron Rachlin, Dan Van Hook, Matthew Daggett, Douglas Marquis, Jeffrey Gottschalk, Joshua Haines, Arkady Yerukhimovich, Nabil Schear, Joseph Cooley (all of MIT LL), Jonathan Kurz (formerly of MIT LL), representatives of In-Q-Tel, and the reviewers for their comments.

References

[1] J. Widom, "Trio: a system for integrated management of data, accuracy, and lineage," 2005.

[2] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance techniques," Tech. Rep., 2005.

[3] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware storage systems," in *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267359.1267363

[4] A. Chapman, B. T. Blaustein, L. Seligman, and M. D. Allen, "Plus: A provenance manager for integrated information," in *IRI*. IEEE Systems, Man, and Cybernetics Society, 2011, pp. 269–275.

[5] A. Chapman, M. D. Allen, and B. Blaustein, "It's about the data: Provenance as a tool for assessing data fitness," in *Proceedings of the 4th USENIX Workshop on the Theory and Practice of Provenance (TAPP'12)*, Jun. 2012.

[6] O. Benjelloun, A. D. Sarma, and J. Widom, "Uldbs: Databases with uncertainty and lineage," in *In VLDB*, 2006, pp. 953–964.

[7] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, and P. Paulson, "The open provenance model: An overview," in *IPAW*, ser. Lecture Notes in Computer Science, vol. 5272. Springer, 2008, pp. 323–326.

[8] W. W. W. Consortium, "Provenance working group," http://www.w3.org/2011/prov/wiki/Main_Page.

[9] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (3rd Edition)*. Prentice Hall, December 2009.

[10] jsPlumb, "jsplumb," http://jsplumb.org/, [Online; Accessed 11-October-2012].

[11] N. Technology, "Neo4j," http://neo4j.org/, [Online; Accessed 11-October-2012].