Reconstructing Provenance
Preliminary Results - Technical Report

Sara Magliacane
VU University Amsterdam

Abstract. In our work, we target the problem of reconstructing provenance of documents in a shared folder setting, assuming that only standard filesystem metadata are available. In this report, we show some preliminary results of reconstructing dependencies between documents.

1 Preliminary results

The first approach for reconstructing missing provenance we devised was inspired by AI planning techniques and change detection algorithms, as described in [2]. The goal of this approach is to reconstruct the sequence of transformations between entities by using the A* algorithm combined with a heuristic function based on the edit distance. In this case, there are three major drawbacks:

– we need to define the library of possible operations;
– the search space is unbound;
– for each entity, we compute the edit distance for all entities, not only the more promising entities.

Therefore, we developed a complementary approach, which considers the simpler problem of reconstructing provenance intended as dependencies between entities. The rationale is that once we are able to distinguish dependent entities, it becomes possible to refine the dependency relationships into sequences of operations. In the following, we describe a prototype implementation of this approach and some preliminary results.

2 Prototype implementation

There has been some prior work addressing reconstructing provenance as dependencies, in particular for documents (e.g., [1]). Though the idea of using text similarity as measure of dependency between documents is quite naive, it leads to good results. We expand this approach by considering several multi-modal similarity measures, in particular image similarity and metadata similarity, and aggregating them into a single score.

To show the potentiality of combining different similarity measures, we developed a prototype, mostly by taking advantage of existing libraries and frameworks. The setting we consider is reconstructing dependencies among a set of documents of different types (including images, Latex files, PDFs, MS Office documents, etc.) in a shared folder. The prototype performs the following tasks:
We evaluated the prototype in a preliminary experiment, which we now describe.

1 https://www.dropbox.com/developers/reference/sdk
2 http://tika.apache.org/
3 http://lucene.apache.org/
4 http://sourceforge.net/projects/simmetrics/
5 https://github.com/lucmoreau/ProvToolbox
2.1 Experimental setting

The experimental setting consisted of a Dropbox folder containing all data for a workshop paper. The folder contained images, Tex and BibTeX files, PDFs, Doc files and a zip file. Due to the nature of the experimental set, the folder contained several versions and revisions of the paper. The provenance of the files in the folder, including the activities performed on them, was manually annotated in PROV-DM, as shown in Fig. 2. The manually annotated provenance graph was converted into a dependency graph (Fig. 3), in which each node represents a file and each edge a dependency of the origin file from the destination file. The process of converting the provenance graph into a dependency graph consisted in removing the activities and computing a transitive closure of the resulting edges.

Fig. 2. Manual annotation of the folder’s provenance
2.2 Results and evaluation

We ran our prototype on the previously described experimental setting with different sets of similarity measures and we obtained several predictions of dependency graphs. One example can be seen in Fig. ??, in which we show the predicted dependency graph using all the implemented similarity metrics. In order to evaluate the results we obtained, we compared the edges of the original dependency graph and each dependency graph, predicted with our method. In particular, we considered predicting a dependency between two documents as a classification problem, so we could define the following quantities:

- False Positives: number of edges only in the predicted graph (not in the original)
- False Negatives: number of edges only in the original graph (not in the predicted)
- True Positives: number of edges both in original and predicted graph
– True Negatives: number of edges that are neither in the original nor in the predicted graph (considering each possible couple of nodes)

Using these values, we measured the precision and recall of our predictions. The results are shown in Table 1, where the rows represent the evaluation using different similarity measures. The first row represents our baseline, i.e. the approach described in[1].

<table>
<thead>
<tr>
<th>Similarity measures</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>0.577</td>
<td>0.659</td>
<td>0.615</td>
</tr>
<tr>
<td>text, metadata</td>
<td>0.652</td>
<td>0.693</td>
<td>0.656</td>
</tr>
<tr>
<td>text, metadata, images</td>
<td>0.648</td>
<td>0.769</td>
<td>0.703</td>
</tr>
<tr>
<td>text, metadata, images, match title</td>
<td>0.629</td>
<td>0.802</td>
<td>0.705</td>
</tr>
</tbody>
</table>

Table 1. Comparison of results using different similarity measures

References