PROV provenance traces from Wikipedia history pages

Paolo Missier
School of Computing Science
Newcastle University, UK
Paolo.Missier@ncl.ac.uk

Ziyu Chen
School of Computing Science
Newcastle University, UK
z.chen3@newcastle.ac.uk

ABSTRACT

Wikipedia History pages contain provenance metadata that describes the history of revisions of each Wikipedia article. We have developed a simple extractor which, starting from a user-specified article page, crawls through the graph of its associated history pages, and encodes the essential elements of those pages according to the PROV data model.[1] The crawling is performed on the live pages using the Wikipedia REST interface. The resulting PROV provenance graphs are stored in a graph database (Neo4J), where they can be queried using the Cypher graph query language (proprietary to Neo4J), or traversed programmatically using the Neo4J Java Traversal API.

1. INTRODUCTION

This short paper describes provenance traces that contain "history" metadata for selected Wikipedia pages. The traces are encoded using the PROV provenance model from the W3C,[2] and are available as Neo4J data files[3] through the ProvBench repository[4]. In the rest of this report we present the structure and content of a Wikipedia PROV provenance graph, describe the crawling model and the capabilities of the tool used for the extraction, and show the general query style of the Cypher graph query language, by presenting typical queries on the traces.

2. ENCODING WIKIPEDIA HISTORY USING PROV

Wikipedia revision history pages describe the timeline of changes that occurred to an underlying page. The main revision history page (Fig.1) consists of a time series of records, containing much of the same information found in any version control system: the date of the edit and the user responsible for it along with notes and links to the pages before/after the edit, and a content diff page. Additional useful information includes the size of the page before and after the edit, and the size of the modified text.

The revision history itself can be captured very simply by creating a sequence of revision relations, between pairs of subsequent versions of an article. In PROV, this is achieved using the built-in `wasDerivedFrom` relation together with a type qualifier, namely `prov:type= 'prov:revision'`. Thus, the basic PROV pattern for revision, expressed here using the PROV-N notation[5], is the following:

```
entity (oldVersion,
    [ prov:type='article', pageid='X',
      revid='old' , title='Y' ])
entity (newVersion,
    [ prov:type='article' , pageid='X',
      revid='new', title='Y' ])
wasDerivedFrom(newVersion, oldVersion,
    [ prov:type='prov:Revision' ])
```

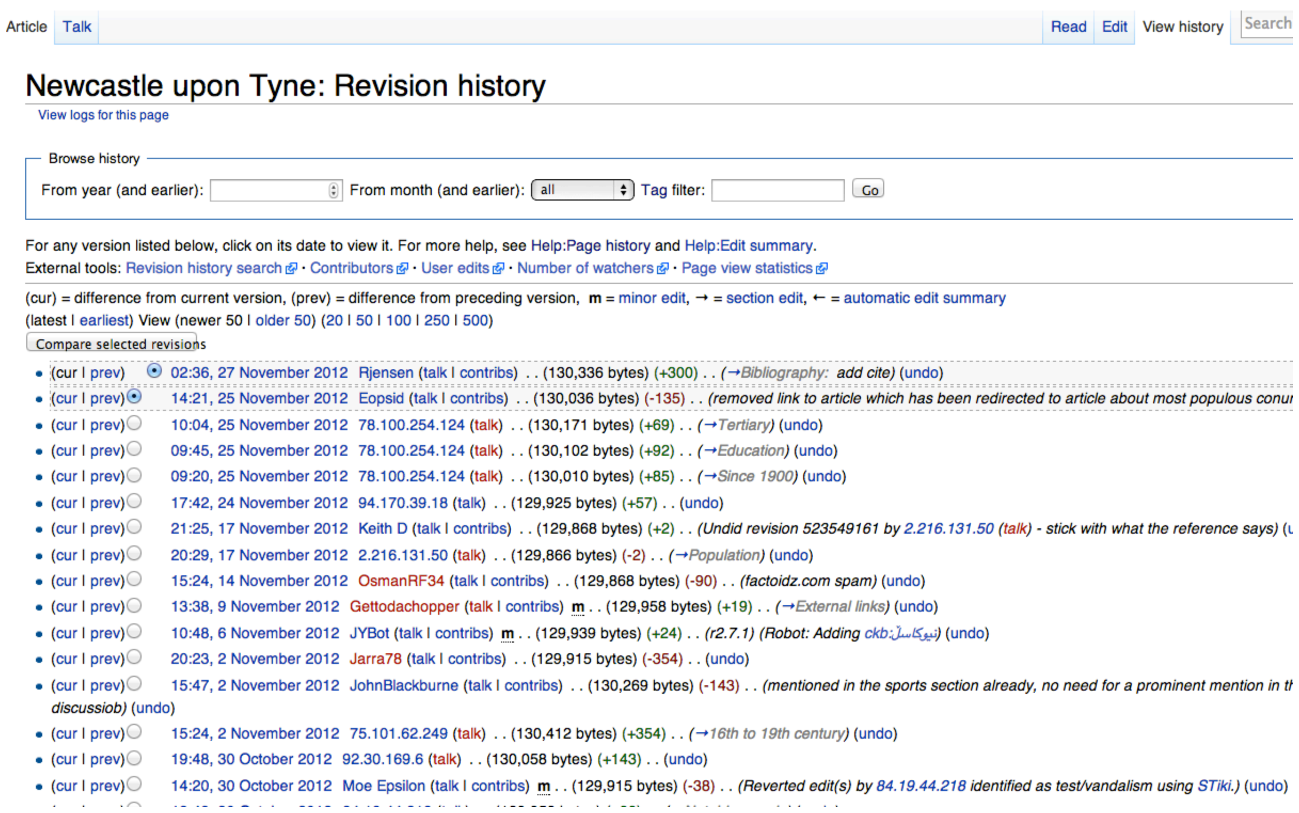
where the two entities that represent the old and new version share the same pageid and title, and have different revision ids.

The complete pattern also includes the editing activity, as well as the agent who was responsible for the editing. Using PROV-N, this is expressed as follows:

```
agent(aEditingAgent,
    [ prov:type='editor', username='...', id ='...'])
activity (aEditing, startTime, endTime,
    [ prov:type='edit'])
wasAssociatewdWith(aEditing, aEditingAgent,
    [ comment='commentId'])
used(aEditing, oldVersion, time)
wasGeneratedBy(newVersion, aEditing, time)
```

We have implemented a simple crawler to harvest wikipedia history pages and encode them using PROV. The crawler's exploration can be controlled using several parameters. Starting from a user-specified Wikipedia topic, for instance "Newcastle Upon Tyne"[6], the crawler begins by following its re-

[1] http://www.w3.org/TR/prov-dm/
[2] http://www.w3.org/TR/prov-dm/
[3] The community edition of the Neo4J graph database is available at neo4j.org.
[4] The Wikipedia area of the ProvBench repository is available at: https://github.com/provbench/Wikipedia-PROV
[5] http://www.w3.org/TR/prov-n/
[6] The article itself is recommended reading, as the city's history dates back from the late Roman empire and was once one of the end points of Hadrian's wall.


Article Talk Read Edit View history Search

Newcastle upon Tyne: Revision history

View logs for this page

Browse history

From year (and earlier): From month (and earlier): all Tag filter: Go

For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary.

External tools: Revision history search · Contributors · User edits · Number of watchers · Page view statistics

(cur) = difference from current version, (prev) = difference from preceding version, **m** = minor edit, → = section edit, ← = automatic edit summary

(latest | earliest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 02:36, 27 November 2012 Rjensen (talk | contribs) . . (130,336 bytes) (+300) . . *(→Bibliography: add cite)* (undo)
- (cur | prev) 14:21, 25 November 2012 Eopsid (talk | contribs) . . (130,036 bytes) (-135) . . *(removed link to article which has been redirected to article about most populous conu*
- (cur | prev) 10:04, 25 November 2012 78.100.254.124 (talk) . . (130,171 bytes) (+69) . . *(→Tertiary)* (undo)
- (cur | prev) 09:45, 25 November 2012 78.100.254.124 (talk) . . (130,102 bytes) (+92) . . *(→Education)* (undo)
- (cur | prev) 09:20, 25 November 2012 78.100.254.124 (talk) . . (130,010 bytes) (+85) . . *(→Since 1900)* (undo)
- (cur | prev) 17:42, 24 November 2012 94.170.39.18 (talk) . . (129,925 bytes) (+57) . . (undo)
- (cur | prev) 21:25, 17 November 2012 Keith D (talk | contribs) . . (129,868 bytes) (+2) . . *(Undid revision 523549161 by 2.216.131.50 (talk) - stick with what the reference says)* (u
- (cur | prev) 20:29, 17 November 2012 2.216.131.50 (talk) . . (129,866 bytes) (-2) . . *(→Population)* (undo)
- (cur | prev) 15:24, 14 November 2012 OsmanRF34 (talk | contribs) . . (129,868 bytes) (-90) . . *(factoidz.com spam)* (undo)
- (cur | prev) 13:38, 9 November 2012 Gettodachopper (talk | contribs) **m** . . (129,958 bytes) (+19) . . *(→External links)* (undo)
- (cur | prev) 10:48, 6 November 2012 JYBot (talk | contribs) **m** . . (129,939 bytes) (+24) . . *(r2.7.1) (Robot: Adding ckb:نیوکاسڵ)* (undo)
- (cur | prev) 20:23, 2 November 2012 Jarra78 (talk | contribs) . . (129,915 bytes) (-354) . . (undo)
- (cur | prev) 15:47, 2 November 2012 JohnBlackburne (talk | contribs) . . (130,269 bytes) (-143) . . *(mentioned in the sports section already, no need for a prominent mention in tl discussiob)* (undo)
- (cur | prev) 15:24, 2 November 2012 75.101.62.249 (talk) . . (130,412 bytes) (+354) . . *(→16th to 19th century)* (undo)
- (cur | prev) 19:48, 30 October 2012 92.30.169.6 (talk) . . (130,058 bytes) (+143) . . (undo)
- (cur | prev) 14:20, 30 October 2012 Moe Epsilon (talk | contribs) **m** . . (129,915 bytes) (-38) . . *(Reverted edit(s) by 84.19.44.218 identified as test/vandalism using STiki.)* (undo)

Figure 1: Screenshot of a Wikipedia revision history page

vision history back to a user-specified max number of revisions. An example of how the basic pattern described above unfolds appears in the graph of Fig. 2 (the label `wasRevisionOf` is used here as shorthand for the qualified derivation shown above. Also, the ids for the specific edit activities are not shown). One can see that three editors were responsible for the history fragment portrayed in the graph. At this point, the crawler may explore the user dimension of the revision graph, by visiting the pages associated to each of those editors, where the history of the edits made by the user on articles are found. From there, the crawler may then follow one of the articles and once again trace the article's revisions. By repeatedly switching between the article space and the user space, the crawler progressively uncovers a connected component of the Wikipedia provenance graph. The process results in very large graphs that exhibit a regular structure. A small fragment with the entire pattern is shown in Fig. 3.

This graph portrays two agents. The one on the left has been involved in the Newcastle article as well as in at least one other article, while the one on the right is known to have carried out 19 other revisions (collapsed in one circle node in the figure). Note that these articles may be completely unrelated, and thus there is no expectation that the crawler will actually stay focused on related topics. In many cases, these agents can be recognized as being one of the many *bots* that continuously crawl Wikipedia trying to fix formatting problems, to revert vandalised pages, or, as in this case, disambiguating the names of people who appear in the articles. The crawler filters out bot activities that it detects using a combination of regular expressions applied to user comments (which sometimes clarify that those are bot-made edits), as well as a user-provided list of bot usernames. This is the only attempt that the crawler makes at staying within a topical area of Wikipedia, however. Full-fledged focused crawling of provenance is beyond the scope of this implementation.

Three parameters are used to control the extent of the user/article spaced visited by the crawler. Firstly, the *revision length* determines the max. number of *wasRevisionOf* relations traversed, towards the past, from a landing revision page. Secondly, the *max users* parameter determines the max number of *wasAssociatedWith* relations, i.e., the max number of contributions explored per user. Thirdly, the *depth* parameter determines how many times the switchover between article space and user space may occur. For example, setting $depth = 3$ results in the exploration of revisions for articles that are connected to the original seed article through at most 2 intermediate users: base article → user1 → article2 → user2 → article3.

3. STORAGE AND QUERYING OF PROV GRAPHS

As mentioned, the crawler stores the PROV traces in a Neo4J graph database[7]. The Neo4J DBMS comes with a proprietary graph query language, called Cypher, and a Java Traversal API (Cypher queries can be embedded within

[7]Details on how to obtain and install a free license of Neo4J community edition are provided in separate documentation provided with the traces themselves.

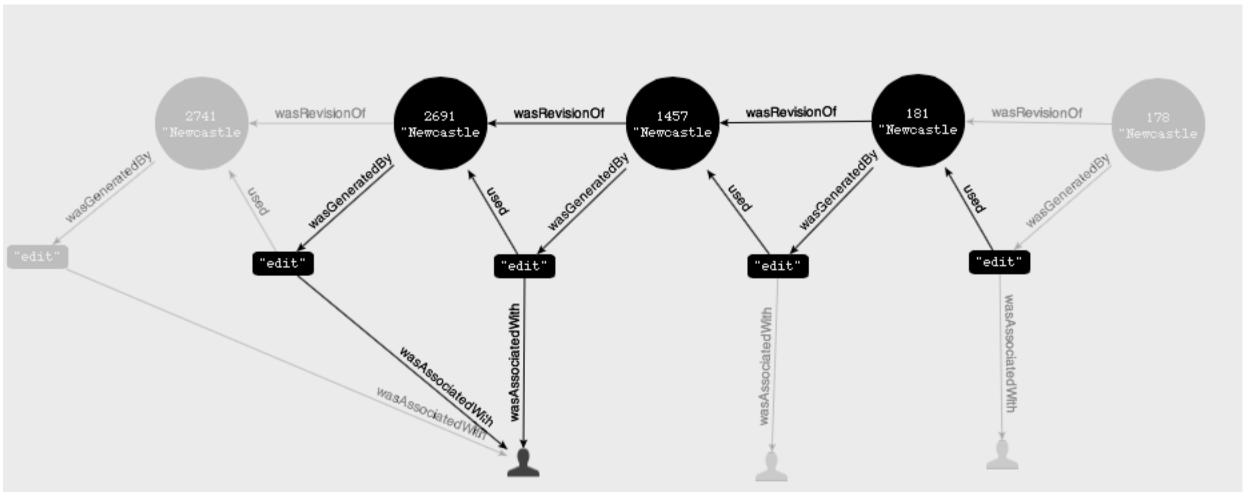


Figure 2: Basic graph pattern for a revision history

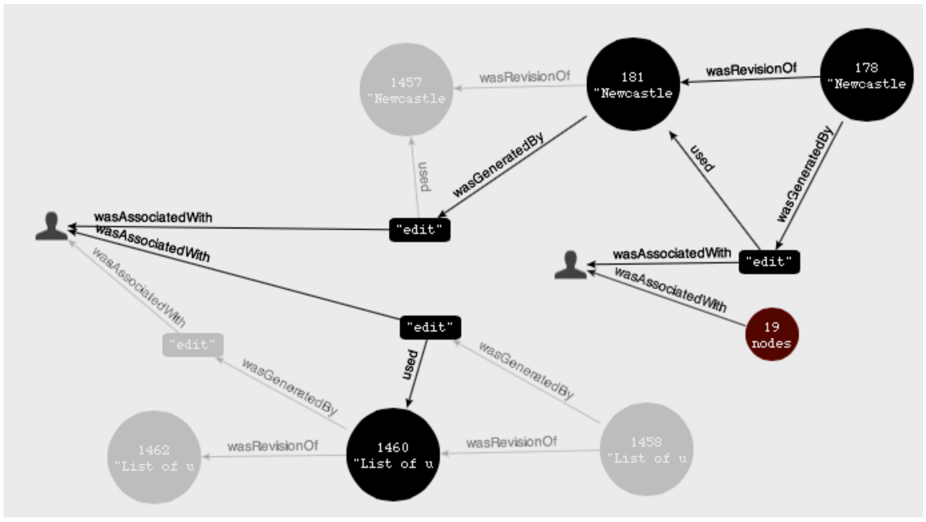


Figure 3: Exploring the article space and the user space

Java, as well. Third party graph traversal languages such as Gremlin[8] are also supported). This facilitates the implementation of several types of graph analysis, including for instance discovering connections amongst users (by clustering them according to the sets of pages they co-edit), measuring the heterogeneity of their interests and the stability of pages over time, the frequency of acts of vandalism over certain pages, and so forth. While the crawler incorporates some of these functions, those are more generally best supported by means of external applications over the PROV database.

The following sample Cypher queries demonstrate the style of declarative specification of graph traversal, as well as the potential of this DBMS for provenance analysis. The first query is designed to traverse revision histories, i.e., paths consisting entirely of `wasRevisionOf` relations (up to 100), starting from the home node, and counting the number of revisions along the way:

[8]`https://github.com/tinkerpop/gremlin/wiki`

```
START n=node(1)
MATCH revisionChain=n−[r?:wasRevisionOf*1..100]−>n2
RETURN n2.title, n2.time?, n2.revid, length(revisionChain), r
ORDER BY n2.revid
```

The query returns a table, listing the names of the revision nodes, optionally the time, the revision ID and the length of the revision chain from the start node (1). The structure of a query generally includes a starting point (START clause), consisting of a selection of nodes (either selected by node id, as in this example, or by index lookup), followed by graph patterns that include a combination of nodes and relationhip types (MATCH clause), and a RETURN clause that is roughly equivalent to a SQL SELECT statement.

The next query lists all articles edited by each agent, by finding all graph patterns where an article is connected to an agent node by way of one editing activity:

```
START agentNode=node(*)
```

```
MATCH article −[:wasGeneratedBy] −>
          edit  −[:wasAssociatedWith]−>agentNode
return distinct  article . title ,  agentNode.user_name
order by agentNode.user_name
```

The result include for instance the following pairs:

JohnBlackburne	The Fabulous Thunderbirds
JohnBlackburne	Distance from a point to a line
JohnBlackburne	Macau
...	
Rjensen	Newcastle upon Tyne
Rjensen	Harlan F. Stone
Rjensen	John Marshall Harlan II
Rjensen	Thomas Johnson (jurist)

Finally, the next query, a variation of the previous, illustrates the aggregation capabilities of Cypher, by counting the number of revisions for each agent and for each page title:

```
START agentNode=node(*)
MATCH article −[:wasGeneratedBy] −>
             edit  −[:wasAssociatedWith]−>agentNode
return  article . title ,  agentNode.user_name, count(edit)
order by agentNode.user_name
```

4. CONCLUSIONS

We have presented a simple crawler for Wikipedia history pages, which produces PROV-compliant provenance graphs and stores them into a Neo4J database. Some sample traces in the form of Neo4J data files have been contributed to the *ProvBench* traces repository[9]. The crawler is at preliminary stages of development, however the code is available for download[10].

[9] `https://github.com/provbench/Wikipedia-PROV`
[10] `https://github.com/PaoloMissier/wikipedia-provenance`