Timelining Knowledge Graphs in the Browser

Damien Graux(✉)[1,2], Fabrizio Orlandi(✉)[2],
Tanmay Kaushik[2], David Kavanagh[2], Hailing Jiang[2], Brian Bredican[2],
Matthew Grouse[2], and Dáithí Geary[2]

[1] Inria, Université Côte d'Azur, CNRS, I3S, France
[2] ADAPT SFI Research Centre & Trinity College Dublin, Ireland
{grauxd,orlandif}@tcd.ie

Abstract. Knowledge graphs, available on the Web via SPARQL endpoints, provide practitioners with various kinds of information from general considerations to more specific ones such as temporal data. In this article, we propose a light-weight solution to visually grasp, navigate and compare, in a Web browser, temporal information available from SPARQL endpoints. Furthermore, we use Wikidata's public SPARQL endpoint to demonstrate our solution and allow users to navigate Wikidata's temporal information.

1 Introduction

Over the past two decades many data sources have been published on the Web. Most of the time, they follow the recommendations and standards promoted by the World Wide Web Consortium (W3C) within the Semantic Web movement, driven by the desire to create a "Web of data" from the conventional "Web of documents". These datasets, generally represented thanks to the RDF format [9] and accessible via the SPARQL language [12], deal with subjects ranging from generalist knowledge such as DBpedia [8], YAGO [10] or Wikidata [11] to specific knowledge such as legal court cases [6], source codes [7] or medical information [13]. Thus, the amount of semantic data now (publicly) accessible makes it possible to create new applications combining for instance several datasets at once. In addition, among the nowadays available datasets, several ones are open online and offer public endpoints on which users may send queries.

To help users navigate this large amount of RDF information, data architects rely on the use of ontologies to structure their datasets. Typically, they declare how entities may be related to each other, for instance a person is of type *human* and should have *parents* who are also of type *human*[1]. More specifically, any kind of type and or relation could be represented in RDF and thus serves to structure knowledge. Among the various facets which might be represented, one is often present in knowledge graphs: temporal information. These can take

[1] See for example the friend-of-a-friend structure devoted to linking people and information using the Web: http://xmlns.com/foaf/spec/

various forms, from the representation of a specific point in time (*i.e.* a date) to a time span. Practically, in RDF, dated information can correspond to a typed date represented by a literal like so: `"25/10/2021"^^xsd:date` which means that objects are the date-*carriers*[2] and predicates express is the date is related to a point in time, or to a starting date, *etc.* As a consequence, there is available temporal information in most of the RDF knowledge graphs, however, having access to it is not a trivial task as the RDF triple structure splits the information across several statements.

In this article, we present an approach to gather, in a Web browser, temporal information coming from SPARQL endpoints. We describe how our system relies on SPARQL queries in order to retrieve the necessary information and how the pipeline can be modified to adapt to other data sources. Visually, our solution allows to present temporal information using timelines which are enriched with additional features such as "click-&-follow" or "compare" nodes. Finally, to validate our approach, we deploy our strategy on the Wikidata [11] public endpoint and host it on a Github page at https://wikitimeline.github.io/.

2 Collection of temporal information

As introduced in the previous section, knowledge graphs are generally structured using the RDF standard [9]. Technically it organises data in *triple* statements composed of a *subject*, a *predicate* and an *object*. In addition to that, each element is entitled to have a specific semantic role: the predicate can only be a Uniform Resource Identifier (*e.g.* <http://purl.org/dc/terms/title>[3]) when an object can also be a literal value such as `"some description"@en` tagged by the English language. Therefore, if there are some temporally typed data existing in an RDF dataset, it should be as a literal in an object field.

Associated with the RDF standard comes SPARQL [12]: its *de facto* query language, standardised by the W3C too. SPARQL adopts an imperative SQL-like syntax for fetching information together with graph-specific features and a large set of functions specific to dealing with RDF data; for example, `isIRI` checks whether the mapped value behind a variable is a URI or not, and it can be used to test a variable having an object role in the graph...

As a consequence, using exclusively SPARQL (standard) queries, we can filter the graphs in order to retrieve temporally typed/tagged information together with their related context (close-triples). For example in:

```
SELECT * WHERE {
 ?s ?p ?o . FILTER ( isLiteral(?o) )
 FILTER ( DATATYPE(?o) = <http://www.w3.org/2001/XMLSchema#date> )
}
```

The query will run through the entire RDF default graph (pattern `?s ?p ?o`) and returns the triple if the object `?o` is a literal **and** if it is a date `<..Schema#date>`.

[2] As only objects can be literal in RDF [9].
[3] URIs are *dereferenceable* and could thereby be mapped to a resource.

In the context of more complex / richer knowledge graphs, we apply a similar strategy following the dataset ontology in order to know how (and where) the dates or the temporal information are stored.

3 Timelines in the Web browser

In order to be able to visualise temporal information from triple endpoints, we devised a twofold architecture. First, we design a way to render such information visually and thus decide to represent time spans using timelines. In particular, we choose to rely on the timelines-chart library[4] to draw and navigate through temporal information. Practically, the solution is implemented using basic JavaScript and does not require the use of heavy external libraries. Second, to fetch the necessary data needed by the visualiser, we write SPARQL queries to be sent to the chosen endpoint where the RDF triples are stored. These queries are in charge of filtering the graphs to extract only the "interesting" parts together with the temporal information.

3.1 Adaptability with SPARQL

For adaptability purposes, we design our approach so that changing the accessed SPARQL endpoint (and thus the RDF graphs queried) does not imply a complete redevelopment of our tool. Indeed, to change the considered knowledge graph, one should modify (1) the address of the endpoint and (2) the queries used. If the first step is straightforward, the second may still lead to some simple query rewriting. To limit this effort, we compact the queries so that all the necessary fields are retrieved with two queries. More generally, these two queries should return the following elements to guarantee that the visualiser can properly process the data:

Q_1: `SELECT ?pred ?obj ?pName` – to find all dates (`?obj`) and their relationships (`?pred & ?pName`) related to a specified subject;
Q_2: `SELECT ?pred ?pName ?obj ?oName ?start ?end` – to find start & end dates metadata about statements (`?sub ?pred ?obj`) on a specified subject.

Such a uniform query output structure allows therefore the practitioners to change the content of the query as much as they want. For instance, on Wikidata [11], and considering the "*qualifiers*"[5] *i.e.* the way Wikidata represents statements of statements, Q_2 would be as follows:

```
SELECT ?pred ?pName ?obj ?oName ?start ?end WHERE {
 ?subj ?pred ?statement. ?statement ?predPS ?obj.
 ?statement pq:P580 ?start.
 OPTIONAL { ?statement pq:P582 ?end. FILTER(?end >= ?start).}
 FILTER(STRSTARTS(STR(?predPS), "http://www.wikidata.org/prop/statement/"))
 FILTER(STRSTARTS(STR(?pred), "http://www.wikidata.org/prop/"))
 FILTER(STRSTARTS(STR(?statement), "http://www.wikidata.org/entity/statement/"))
 ?x wikibase:claim ?pred. ?x rdfs:label ?pName.
```

[4] https://github.com/vasturiano/timelines-chart (built on D3.js)
[5] https://www.wikidata.org/wiki/Help:Qualifiers

```
 ?obj rdfs:label ?oName.
 FILTER((LANG(?pName)) = "en"). FILTER((LANG(?oName)) = "en").
}
```

However, on YAGO4 [10], considering that RDF-star [5] is used to reify triples, Q_2 would be:

```
SELECT ?pred ?pName ?obj ?oName ?start ?end WHERE {
 << ?subj ?pred ?obj >> <http://schema.org/startDate> ?start .
 OPTIONAL{ << ?subj ?pred ?obj >> <http://schema.org/endDate> ?end .}
 ?pred rdfs:label ?pName .
 ?obj rdfs:label ?oName .  FILTER(LANG(?oName)='en').
}
```

Hence, our solution can be deployed on any endpoint as long as the involved queries are answering the same structures.

3.2 Visual Features

From the graphical point-of-view, we develop several features to enrich the users' experience. Practically, timelines are representing temporal information related to one single entity (usually a subject in RDF). As the number of elements describing a specific entity can be large and since time can span over decades for some entity (such as states for instance), the interface enables users to zoom on a particular period of time in order to "restrict" the window-view, using a *click-and-drag* movement. Similarly, to help users distinguishing similar kinds of information, we group the same predicates by color. Additionally, more details can be read when the users *hover* above a time-bar displaying the exact starting and ending dates of it. Furthermore, in order to "navigate within" the endpoint's RDF graph, we set up a *click-and-follow* mechanism on the time-bars. Actually, clicking on time-bar redirects the users on the timeline corresponding to the entity the initial bar was about. Such a mechanism allows a seamless experience for the user who can explore the available temporal elements from one entity to another. Finally, we implemented a specific *compare* feature to allow users comparing two entities at once by grouping the two timelines together.

3.3 Practical use-case: Wikidata timelines

In order to validate our approach, we tested the system using the available Wikidata SPARQL endpoint. To do so, we designed the necessary SPARQL queries and hosted our interface online. Typically, a comparison between two entities is shown in Figure 1 where the timelines of Presidents George W. Bush and Barack Obama are displayed. In addition, for this online interface, we added an auto-completion feature to fasten the search of any entity as Wikidata uses exotic names *e.g.* B. Obama is "Q76". Finally, following on the dereferenceable principle, we also enable URL access to let user share timelines, for example Tim Berners-Lee's timeline is available from: https://wikitimeline.github.io/search.html?subj=http://www.wikidata.org/entity/Q80.

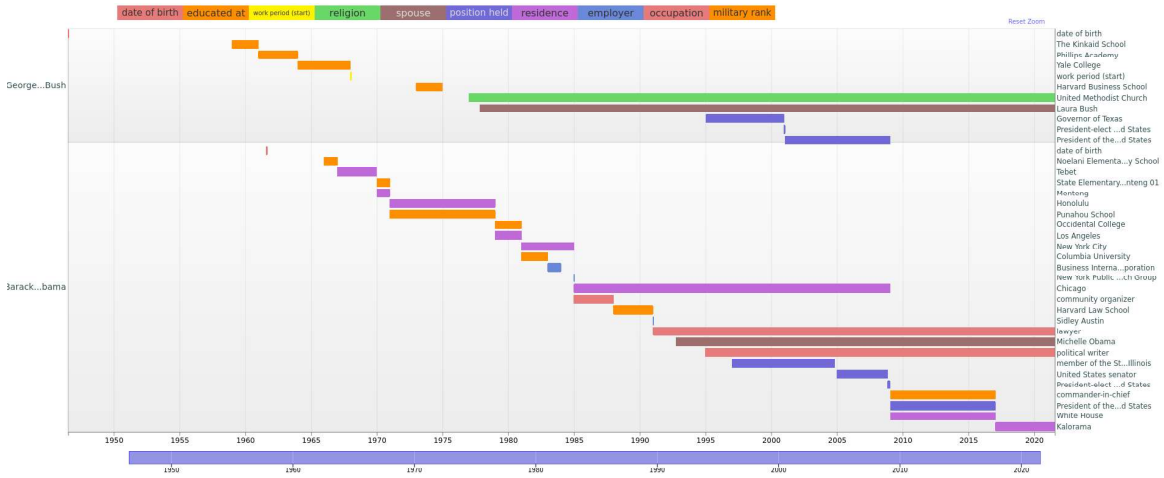

Fig. 1. Comparing George W. Bush and Barack Obama. See these timelines

4 Related Work

To the best of our knowledge, the literature does not contain solutions focusing exclusively on "timelining" temporal information contained in KGs. Nevertheless, timeline representations have already been used within larger platforms as a side-tool such as Metaphactory [4]. Similarly, `rdf:SynopsViz` [1] has a dedicated tab to draw timeline from date properties available in RDF datasets; however, the platform needs to prior ingest the dataset, whereas our solution relies on SPARQL endpoints. Recently, Gottschalk et al. [2,3] formalised the notion of temporal knowledge graph and instantiated it using several bases such as YAGO or Wikidata. Regarding the visualisation, the difference with our approach is that they created their own KG, based on their own schema, therefore the visualisations need to be updated when original datasets change.

5 Conclusion

The proposed web-app visualises and compares Wikidata entities according to their temporal information. A demonstrator is hosted on:

https://wikitimeline.github.io/

under an MIT license[6], providing users a live example of what the application could be locally, would someone be interested in deploying the interfaces at their premises. Further, the presented architecture can be easily deployed on alternative SPARQL endpoints by only changing the queries (which retrieve temporal data) as long as their results are structured similarly.

We presented in this article the first version of our interface focused on representing Wikidat's temporal information using timelines. Practically, we are cur-

[6] Project's code base: https://github.com/wikitimeline/wikitimeline.github.io

rently setting up a user validation experiment in order to improve the timeline *navigation* and experience. On a different note, we are also planning to improve the WebApp-side with additional features such as: allowing timeline exports as image or improving the coloring to for instance group predicate together. Finally, it is worth noticing that our architecture is not bound to querying a single endpoint at once; indeed, it is possible to extend the *compare* feature presented above to query two or more SPARQL endpoints, allowing a comparison of the temporal information available in different databases at a glance.

Acknowledgments This research was conducted with the financial support of the EU Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 713567 at the ADAPT SFI Research Centre at Trinity College Dublin. The ADAPT SFI Centre is funded by Science Foundation Ireland through the SFI Research Centres Programme and co-funded under the European Regional Development Fund Grant #13/RC/2106.

References

1. Bikakis, N., Skourla, M., Papastefanatos, G.: rdf:synopsviz – A framework for hierarchical linked data visual exploration and analysis. In: European Semantic Web Conference. pp. 292–297. Springer (2014)
2. Gottschalk, S., Demidova, E.: EventKG: A multilingual event-centric temporal knowledge graph. In: ESWC. pp. 272–287. Springer (2018)
3. Gottschalk, S., Demidova, E.: EventKG–the hub of event knowledge on the web–and biographical timeline generation. Semantic Web **10**(6), 1039–1070 (2019)
4. Haase, P., Herzig, D.M., Kozlov, A., Nikolov, A., Trame, J.: metaphactory: A platform for knowledge graph management. Semantic Web **10**(6), 1109–1125 (2019)
5. Hartig, O.: Foundations of RDF* and SPARQL*:(An alternative approach to statement-level metadata in RDF). In: AMW 2017 11th Int. Workshop on Foundations of Data Management and the Web. vol. 1912 (2017)
6. Junior, A.C., Orlandi, F., Graux, D., Hossari, M., O'Sullivan, D., Hartz, C., Dirschl, C.: Knowledge graph-based legal search over german court cases. In: ESWC (2020)
7. Kubitza, D.O., Böckmann, M., Graux, D.: Semangit: A linked dataset from git. In: International Semantic Web Conference. pp. 215–228. Springer (2019)
8. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal **6**(2), 167–195 (2015)
9. Manola, F., Miller, E., McBride, B., et al.: RDF primer. W3C recommendation **10**(1-107), 6 (2004)
10. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge. In: WWW. pp. 697–706. ACM, New York, NY, USA (2007)
11. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
12. W3C SPARQL Working Group, et al.: SPARQL 1.1 overview (2013), http://www.w3.org/TR/sparql11-overview/
13. Wishart, D.S., Knox, C., Guo, A.C., Cheng, D., Shrivastava, S., Tzur, D., Gautam, B., Hassanali, M.: Drugbank: a knowledgebase for drugs, drug actions and drug targets. Nucleic acids research **36**(suppl_1), D901–D906 (2008)