A Software Architecture for Narratives

Carlo Meghini, Valentina Bartalesi, Daniele Metilli, Filippo Benedetti

Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR Pisa, Italy,
{carlo.meghini,valentina.bartalesi,daniele.metilli,filippo.benedetti}@isti.cnr.it

Abstract. The current Digital Libraries (DLs) usually return as answer of a user's query a ranked list of the resources included in the DLs but no semantic relation among the resources are reported. Using the Semantic Web technologies it is possible to improve these search functionalities introducing *narratives* as new search method. As *narratives* we intend semantic networks of events that are linked to the objects of the DLs and are endowed with a set of semantic relations that connect an event to another. These semantic networks may help the users to obtain a more complete knowledge on the subject of their searches. In this paper, we present a software architecture for building narratives in order to introduce them in DLs. Our architecture is composed of several tools (automatic and semi-automatic tools) for creating, storing and visualizing narratives. When possible, we reused open source components already available on-line, and for the software we developed, we freely distribute it for research aims.

Keywords: Software Architecture, Semantic Web, Semantic Reasoner, Narratives, OWL, Digital Libraries

1 Introduction

The search functionalities of the current Digital Libraries (DLs) are usually basic systems that answer to a user's query, expressed in natural language, with a ranked list of the resources included in the DLs but no semantic relations among the returned objects are reported. The Semantic Web [3], and the Linked Open Data [11] paradigm, can overcome the limitations of these search functionalities. The long-term aim of our research is to develop and integrate in DLs a new search method, using the semantic Web technologies: the *narrative*. We intend narratives as semantic networks composed of events that are linked to the objects of the DL and are endowed with a set of semantic relations connecting these events, i.e. actions or occurrences taking place at a certain time at a specific location. In our vision, instead of list of objects, DLs should provide the narratives as answers of the queries, which could be useful for users in order to obtain a more complete knowledge on the subject of their searches. To reach this aim, we developed a software architecture that allows to create narratives using the Semantic Web technologies. This architecture is composed of a set of tools (automatic and semi-automatic tools) for creating, storing, querying, and

visualizing narratives. The stored knowledge is formally represented following an OWL ontology for representing narratives [1] we developed, encoded in the OWL 2 DL profile [5]. In order to maximize its interoperability, our ontology was developed as an extension of the CIDOC CRM standard ontology [4].
We have created some narratives using this architecture. In particular, two biographical narratives were produced by a Digital Humanities researcher at the Italian National Research Council (CNR): (i) on the life of Dante Alighieri[1], the major Italian poet of the Middle Age; (ii) on the life of the Austrian painter Gustav Klimt[2]. The third narrative was developed by a researcher in Computational Biology at the CNR to narrate the discoveries related to the giant squid[3].
Several components of our architecture are already developed and open source, thus we reused them. For what regards the software we developed, we distribute it freely for research aims.

2 Architecture

This section describes our current architecture for the representation of narratives. Figure 1 shows the architecture, whose main components are the following:

1. *a narrative-building tool.* It is used for creating, modifying or visualizing a narrative, possibly representing knowledge that has been derived by reading some texts. The user operates through the Graphical User Interface (GUI) of the narrative-building tool, by manually inserting the narrative data and, at the same time, importing resources from Wikidata[4]. The created narrative is stored as an intermediate JSON representation[5];
2. *an OWL triplifier.* Once the narrative is complete, the corresponding JSON representation is given as input to the Java Triplifier. The triplifier transforms the JSON file into an OWL (Web Ontology Language) ontology encoded as an RDF graph, using the OWL API library [6]. The organization of the knowledge in the graph follows the structure defined in the ontology for narratives we developed [1].
3. *a semantic reasoner.* It is used by the triplifier to infer new knowledge. The triplifier takes as input also a file with SWRL rules [7] that are used by the reasoner to support the temporal reasoning on the narrative.
4. *a triple store.* The triplifier stores the resulting graph, expanded with inferences produced by both the reasoner and the SWRL rules, into a Blazegraph triple store[6];
5. *a visualization interface.* Finally, the user can access the knowledge stored in the triple store through a Web interface. The knowledge is extracted using SPARQL queries [10] and shown using graphic libraries.

[1] https://dlnarratives.eu/timeline/dante.html
[2] https://dlnarratives.eu/timeline/klimt.html
[3] https://dlnarratives.eu/timeline/squid.html
[4] https://www.wikidata.org
[5] http://json.org/
[6] https://www.blazegraph.com/

A review of approaches relevant to our study is reported in [8].

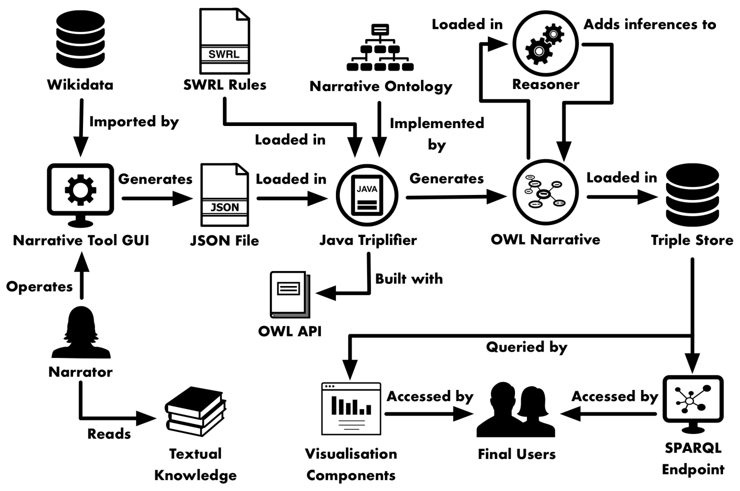


Fig. 1. The schema of our architecture.

2.1 Narrative-building Tool

In order to facilitate the creation of a narrative and its semantic representation by the narrator, we built a web-based *narrative-building tool*. The tool is built with HTML5[7] and JavaScript (ECMAScript6[8]), using the jQuery[9], jQuery UI[10], Bootstrap[11], and Typeahead.js[12] libraries.
The main interface of the tool is based on simple drag-and-drop metaphors, allowing the user to create events and drag the appropriate entities that compose them from a list of entities (e.g. location, person, object) automatically extracted from the Wikidata knowledge base through its SPARQL endpoint. The entities are color-coded according to their class, i.e. person, organization, place, object, concept, or work. These are linked to the corresponding CRM classes by using a specific mapping we defined. The interface also allows the user to link together events by using two different semantic relations: (i) the mereological (part-of) relation, or (ii) the causal dependency relation. A view of the NBVT interface is reported in Figure 2.

[7] https://www.w3.org/TR/html5/
[8] http://www.ecma-international.org/ecma-262/6.0/
[9] https://jquery.com
[10] https://jqueryui.com
[11] https://getbootstrap.com
[12] https://typeahead.js.org

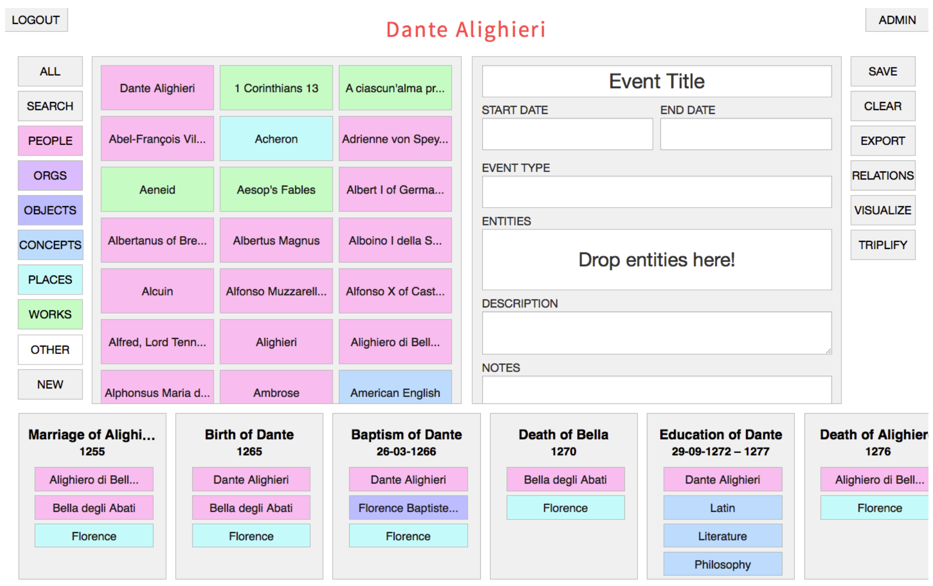


Fig. 2. The interface of the NBVT.

The tool is available online[13], it is open-source and released under the GPLv3 license[14]. As the user inserts knowledge in the interface of the tool, the data is stored into a CouchDB[15] database, using the PouchDB[16] library to interface with it. This allows automatic saving of the data, revisioning, and synchronization between a local and remote database. Finally, the resulting timeline of events is exported in the JSON format.

2.2 OWL Triplifier

The knowledge exported from the tool in JSON format is subsequently imported into a Java-based triplifier. The triplifier makes use of the OWL API library to define an ontology model. Then, it loads the JSON file and converts it to an intermediate Java representation. Then the OWL API library takes as input this representation that is used for populating the model. The triplifier also imports some SWRL rules. The Semantic Web Rule Language (SWRL) is a proposed language for the Semantic Web that can be used to express rules as well as logic, combining OWL DL or OWL Lite with a subset of the Rule Markup Language. We added SWRL rules to overcome the limitations of the OWL 2 DL about: (i) the definition of a relation as simultaneously transitive and irreflexive, as in

[13] https://dlnarratives.eu/tool.html
[14] https://www.gnu.org/licenses/gpl–3.0.en.html
[15] http://couchdb.apache.org
[16] https://pouchdb.com

the case of the part-of and causality relations, (ii) the definition of a relation as simultaneously transitive and disjoint [9], as in the case of the temporal relation, (iii) the implication between part-of and temporal relations and between causality and temporal relations. For this reason, we use SWRL rules and OWL 2 DL axioms simultaneously. The SWRL rules were produced using a software developed by Batsakis et al. [2] for what concerns the temporal relations. The implications between part-of and temporal relations and between causality and temporal relations were defined by writing the SWRL rules manually. The SWRL rules are taken as input by the triplifier as an OWL file. The result of the process of triplifier is an OWL graph that represents the narrative, exportable in RDF/XML[17], Turtle[18], or several other syntaxes[19].

2.3 Reasoner

At this point, a reasoning is performed on the knowledge in order to perform consistency checks and make inferences. The reasoner we adopted is Openllet[20] version 2.6. The main reasons for this choice are the following: (i) Openllet supports all the features of OWL 2 DL; (ii) it fully supports SWRL rules; (iii) it is Java-based and easily integrated with OWL API[21]; (iv) it is an open source software actively maintained. Openllet provides functionality to check consistency of ontologies, compute the classification hierarchy, explain inferences, the graph is stored into a triple store.

2.4 Triple Store

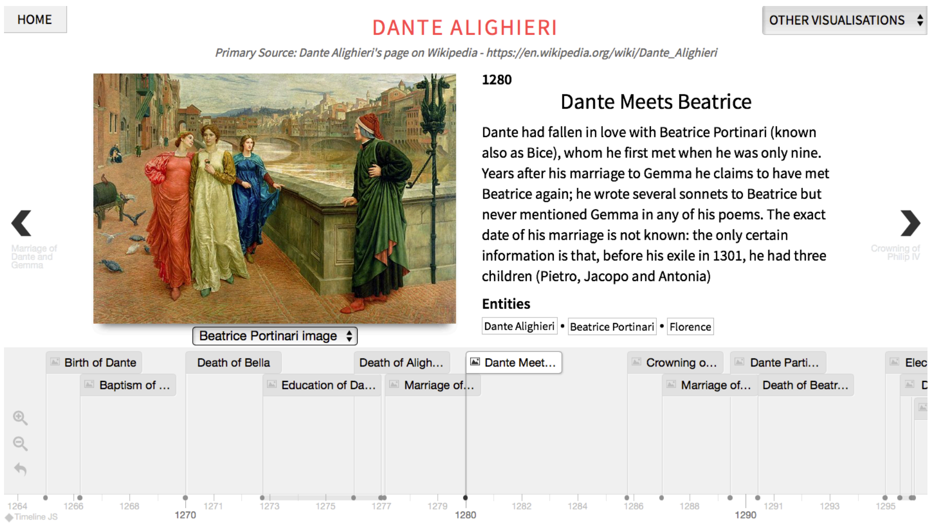
The knowledge is exported to a triple store. The triple store we chose is Blazegraph [22]. Blazegraph is a standards-based, high-performance, scalable, open-source graph database. Written entirely in Java, the platform supports the RDF data model and the SPARQL 1.1 family of specifications, including Query, Update, Basic Federated Query, and Service Description. The knowledge stored in Blazegraph is shown to the user through a visualization interface. We implemented SPARQL queries to retrieve this knowledge from the triple store.

2.5 Visualization Interface

First of all, in order to give a complete overview of the narrative, the events were placed on a timeline. We used TimelineJS library[23] for the implementation. For each event on the timeline, the more meaningful information is reported, i.e. title,

[17] https://jena.apache.org/documentation/io/rdf-output.html
[18] https://www.w3.org/TR/turtle/
[19] https://jena.apache.org/documentation/io/rdf-output.html
[20] https://github.com/Galigator/openllet
[21] https://owlcs.github.io/owlapi/
[22] https://www.blazegraph.com/
[23] https://timeline.knightlab.com/

date, primary sources, related digital objects, related images. Events occurred at the same time are allowed and visualised on a timeline. Figure 3 shows an event of the timeline of Dante Alighieri's life.



1280

Dante Meets Beatrice

Dante had fallen in love with Beatrice Portinari (known also as Bice), whom he first met when he was only nine. Years after his marriage to Gemma he claims to have met Beatrice again; he wrote several sonnets to Beatrice but never mentioned Gemma in any of his poems. The exact date of his marriage is not known: the only certain information is that, before his exile in 1301, he had three children (Pietro, Jacopo and Antonia)

Entities

Dante Alighieri • Beatrice Portinari • Florence

Fig. 3. An event of Dante Alighieri's life on the timeline.

Another requirement for the tool is the visualization of the entities that compose each event. To this aim, a SPARQL query to get this information from the knowledge base was implemented. This query retrieves, for each event title, the names and IRIs of the corresponding entities. The vis.js[24] JavaScript library was used to implement the visualization. One of the most important requirements for a scholar who studies historical events, is the knowledge of their primary sources. For each event of the narrative, the tool allows to visualize the primary sources and in particular the title and the author of a primary source, the textual fragment of the primary source that describes the event, the reference of the textual fragment. This information is visualized in tabular format. Finally, the user has the possibility to visualize all events that occurred in a specified period of time. Upon specifying the desired period, the user can freely insert the dates using a widget to select a full date or the year only. The results of the query are shown in form of table, where for each event its dates are shown.
It is possible to explore the visualization interface on-line[25], browsing the three narratives that are available on our Web site[26].

[24] http://visjs.org
[25] https://dlnarratives.eu/narratives.html
[26] https://dlnarratives.eu

3 Conclusions and Future Work

In this paper we have presented a software architecture for building narratives using the Semantic Web technologies. In this context, we intend narratives as semantic networks of events linked to each other and to digital objects by semantic relations. In order to represent the knowledge we have developed an OWL ontology for narratives as an extension of the CIDOC CRM standard vocabulary. Where possible, to develop this architecture, we have reused software open source already available. For what regards the software we developed, it is freely distributed, released under the GPLv3 license.
The long-term goal of our study is introducing the narrative as new first-class search functionality of digital libraries. As output of a query, this new search functionality should not only return a list of objects but it should also present one or more narratives on the topic of the search. The architecture presented in this paper would be used to create narratives that later could be imported and shown in the DL interfaces.

References

1. Bartalesi, V., Meghini, C., Metilli, D.: Steps towards a formal ontology of narratives based on narratology. In: OASIcs-OpenAccess Series in Informatics. vol. 53. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
2. Batsakis, S., Petrakis, E., Tachmazidis, I., Antoniou, G.: Temporal representation and reasoning in OWL 2. Semantic Web 8(6), 981–1000 (2016)
3. Berners-Lee, T., Hendler, J., Lassila, O., et al.: The semantic web. Scientific american 284(5), 28–37 (2001)
4. Doerr, M.: The cidoc conceptual reference module: an ontological approach to semantic interoperability of metadata. AI magazine 24(3), 75 (2003)
5. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: Owl 2 web ontology language primer. W3C recommendation 27(1), 123 (2009)
6. Horridge, M., Bechhofer, S.: The owl api: A java api for working with owl 2 ontologies. In: Proceedings of the 6th International Conference on OWL: Experiences and Directions – Volume 529. pp. 49–58. OWLED 2009, CEUR-WS.org, Aachen, Germany (2009), http://dl.acm.org/citation.cfm?id=2890046.2890052
7. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al.: Swrl: A semantic web rule language combining owl and ruleml (2004)
8. Meghini, C., Bartalesi, V., Metilli, D.: Using formal narratives in digital libraries. In: Italian Research Conference on Digital Libraries. pp. 83–94. Springer (2017)
9. Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: Owl 2 web ontology language: Structural specification and functional-style syntax. W3C recommendation 27(65), 159 (2009)
10. Prud, E., Seaborne, A., et al.: Sparql query language for rdf (2006)
11. Yu, L.: Linked open data. In: A Developers Guide to the Semantic Web, pp. 409–466. Springer (2011)