Applying and Developing Semantic Web Technologies for Exploiting a Corpus in History of Science: the Case Study of the Henri Poincaré Correspondence

Olivier Bruneau, Nicolas Lasolle, Jean Lieber, Emmanuel Nauer, Siyana Pavlova, Laurent Rollet

▶ **To cite this version:**

Olivier Bruneau, Nicolas Lasolle, Jean Lieber, Emmanuel Nauer, Siyana Pavlova, et al.. Applying and Developing Semantic Web Technologies for Exploiting a Corpus in History of Science: the Case Study of the Henri Poincaré Correspondence. Semantic Web – Interoperability, Usability, Applicability, In press. hal-02879820v1

HAL Id: hal-02879820

https://hal.univ-lorraine.fr/hal-02879820v1

Submitted on 24 Jun 2020 (v1), last revised 22 Sep 2020 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Semantic Web 0 (0) 1
IOS Press

Applying and Developing Semantic Web Technologies for Exploiting a Corpus in History of Science: the Case Study of the Henri Poincaré Correspondence

Olivier Bruneau [a,*], Nicolas Lasolle [a,b], Jean Lieber [b], Emmanuel Nauer [b], Siyana Pavlova [a,b], and Laurent Rollet [a]

[a] *AHP-PReST, Université de Lorraine, CNRS, Université de Strasbourg, France*
E-mails: olivier.bruneau@univ-lorraine.fr, nicolas.lasolle@univ-lorraine.fr, siyana.pavlova@gmail.com, laurent.rollet@univ-lorraine.fr
[b] *LORIA, Université de Lorraine, CNRS, Inria, France*
E-mails: nicolas.lasolle@univ-lorraine.fr, jean.lieber@loria.fr, emmanuel.nauer@loria.fr, siyana.pavlova@gmail.com

Abstract. The Henri Poincaré correspondence is a corpus of letters sent and received by this mathematician. The edition of this correspondence is a long-term project begun during the 1990s. Since 1999, a website is devoted to publish online this correspondence with digitized letters. In 2017, it has been decided to reforge this website using Omeka S. This content management system offers useful services but some user needs have led to the development of an RDFS infrastructure associated to it. Approximate and explained searches are managed thanks to SPARQL query transformations. A prototype for efficient RDF annotation of this corpus (and similar corpora) has been designed and implemented. This article deals with these three research issues and how they are addressed.

Keywords: history of science, Digital Humanities, Henri Poincaré, scientific correspondence, RDF(S), approximate and explained search, SPARQL query transformation, corpus annotation

1. Introduction

Jules Henri Poincaré is a great name of the history of science. Born in Nancy (France) on April 29, 1854, he died in Paris on July 17, 1912.

His name is associated with discoveries or works of primary importance. He is responsible for, among other things, the discovery of Fuchsian functions in mathematics, an essential contribution to the resolution of the Three-Body problem in celestial mechanics, for which he won the Oscar II Prize, King of Sweden's mathematical competition, in 1889, as well as the introduction in topology of the concepts of homotopy and homology. His theoretical research on new mechanics after 1900 prepared the discovery of the special theory of relativity by Albert Einstein in 1905.

Mathematician, physicist, astronomer, research administrator, Henri Poincaré was also very active in the field of philosophy as a regular contributor to the *Revue de métaphysique et de morale*. His philosophical books, such as *La science et l'hypothèse* [*Science and Hypothesis*] [1] contributed to the birth of French philosophy of sciences and gave him a great reputation in France and at international level. Although not very engaged on the political scene, he nevertheless played

*Corresponding author. E-mail: olivier.bruneau@univ-lorraine.fr.

1570-0844/0-1900/$35.00 © 0 – IOS Press and the authors. All rights reserved

a significant role in the Dreyfus Affair through several mathematical expertises.

Henri Poincaré was elected in 1887 at the *Académie des sciences* of Paris, at the age of 33. Throughout his life he was also a member of the *Bureau des longitudes* (1893), of the *Académie Française* (1908) and of numerous foreign learned societies and academies.

In 1992, the laboratory of history of science and philosophy Archives Henri Poincaré was created to study Henri Poincaré's manuscripts and to organize the publication of his scientific and private correspondence. For more than 25 years, this long-term project has produced four volumes of letters. The first one is devoted to the letters exchanged between Henri Poincaré and the Swedish mathematician Gösta Mittag-Leffler [2]. The second one concerns the correspondence with physicists, chemists and engineers [3]. The third volume gathers the correspondence with astronomers and, in particular, geodesists [4]. The fourth one deals with the Henri Poincaré's youth correspondence [5]. Two other volumes are in preparation. The first one will be devoted to the letters from or of mathematicians. The second will contain Henri Poincaré's administrative, academic and private correspondence.

The corpus[1] consists of around 2100 letters, 1126 sent by Henri Poincaré and 956 received by him. Some other letters have been discovered recently. About 50% of these letters form a correspondence with scientists. Original letters come from 63 different archive centers and libraries from 14 countries. All known letters are digitized[2] and around 60% of them are in plain text (in LaTeX and XML versions). Lots of letters contain mathematical and physical formulae. The correspondence is available on Henri Poincaré website.[3] The letters of this website are indexed with Dublin Core extended metadata.[4] This enables to query the corpus by e.g.

$$\mathcal{Q}_1 = \left| \text{"Give me the letters sent by Henri Poincaré in 1885"} \right.$$

$$\mathcal{Q}_2 = \left| \text{"Give me the letters received by Eugénie Launois between 1882 and 1894"} \right.$$

Plain text search engine exists for the letters that are already transcribed[5] but the proposed set of results can be incomplete or incorrect. For instance, if the query is to find letters sent or received by Henri Poincaré with the topic "lacunary functions", the result is only one letter though there are more than 10 letters referring partially to this topic. The problem is that, in these letters, the term "lacunaire" does not explicitly occur, hence the incompleteness of this search from a semantic viewpoint. In other letters, "espaces lacunaires" can be found but it refers to titles of Henri Poincaré's papers, hence its incorrectness.

This article is organized as follows. Section 2 presents the Henri Poincaré correspondence and how it was edited, annotated and published on the web of documents. It has appeared that principles and technologies of the Semantic Web should prove useful for the exploitation of this corpus, in particular, the RDF(S) technology that is briefly recalled in Section 3, with some examples related to the corpus. The remainder of the paper presents three main works related to Semantic Web for the Henri Poincaré correspondence. Section 4 explains how the RDF(S) infrastructure is added on the Henri Poincaré correspondence website, which involves some translation mechanisms. This makes possible the interrogation of this corpus by SPARQL queries. The need for more flexible querying is motivated in Section 5, together with a way of handling this flexibility. Finally, the prototype of a tool for efficient editing of RDF triples for the purpose of indexing the Henri Poincaré correspondence is described in Section 6.

2. Editing and annotating the correspondence of Henri Poincaré

From 1999 to 2015, the Henri Poincaré Papers website which was created and maintained by Scott Walter

[1] The notion of corpus is in accordance with the terminology in history i.e. a collection of documents of a specific subject gathered and stored to be exploitable. In natural language processing, the appropriate term could be *semantic digital library*.

[2] Due to copyright laws, some images of letters are not available online, though transcripts are.

[3] http://henripoincare.fr.

[4] Different projects exist that are devoted to scientific correspondences for example the CKCC project (http://ckcc.huygens.knaw.nl [6]) or Mapping the Republic of Letters project (http://republicofletters.stanford.edu). More generally, several projects in Semantic Web are dedicated to history. A recent paper [7] proposes a state of the art on this issue. For instance, SemanticHPST is a project in which Semantic Web principles are applied to the history and philosophy of science and technology [8]. A newly founded European consortium called Data for History (http://dataforhistory.org/) aims at uniting such projects.

[5] The search-engine Solr is installed in the platform.

to highlight the Henri Poincaré correspondence knew several developments. The last version was a web application associated with the open source full text search engine Sphinx.[6] One could find, when available, the digitized letters, a transcript, critical apparatus and Dublin Core metadata. This website was harvested by OAI-PMH.[7]

In 2017, the Archives Henri Poincaré decided to reforge the website. In order to better structure the site and to benefit from semantic annotation, this new platform has been based on the content management system (CMS) Omeka S.[8] Developed by the Roy Rosenzweig Center for History and New Media, this CMS has been created for publishing and promoting cultural heritage collections. This system has been used to build digital collections from various institutions such as the Metropolitan New York Library Council (METRO) collection [9], the University of Binghamton [10] or the University of São Paulo [11]. It can also be relevant for the management and share of educational resources [12].

Omeka S allows semantic annotations. In the backoffice, several vocabularies are already available: Dublin Core Terms, Friend of a Friend, and Bibliographic Ontology. Adding other vocabularies is possible, like the Archives Henri Poincaré Ontology.[9] A search engine based on the properties is available but it is not very user friendly. All data of the previous website (digitized letters, transcripts, metadata) are available in the platform. An environment has been created in this CMS in which letters can be visualized (as an image or in plain text) with a critical apparatus essentially coming from the printed edition of this correspondence and an indexation (Figure 1).

For all letters, there are two types of indexing. The first one is a physical description (type of letter – telegram, autograph letter, minutes, etc. –, number of pages, location of the letter within the archive, sender, recipient, date and place of expedition, etc.). Some pieces of information are missing. For example, there are letters for which the exact sending date is unknown, but for which the year and the month are known from the context and associated to the letter.

The second type of indexing relates to the content of the letters; all relevant information can be indexed (see Figure 2 for an example). These data are relevant from the viewpoint of historians. For instance, people or publication quoted, mathematical theories or formulae, philosophical concepts are taken into account.

3. Preliminaries on RDFS and SPARQL

This section makes a presentation of RDF, RDFS and SPARQL that is simplified for the needs of this article, and exemplified in the domain of the correspondence of Henri Poincaré.

The atoms of RDF are resources and literals. A resource is either anonymous or named. An *anonymous resource* (aka *blank node*) is an existentially quantified variable; by convention its identifier starts with a question mark (e.g., `?x` or `?firstName`). A *named resource* is identified by a name without a question mark; it is a constant of any type: instance (e.g., `henriPoincaré`), class (e.g., `Mathematician`), etc. A *property* is a resource denoting a binary relation (e.g., `sentBy` is a property relating a letter to the person who sent it). A *literal* is a constant of a predefined datatype, such as `integer`, `float` or `date`. The term "value" is used in this paper for "resource or literal".

An *RDF triple* is a triple $\tau = \langle s \;\; p \;\; o \rangle$ where s is a resource (the *subject* of τ), p is a property (the *predicate* of τ) and o is either a resource or a literal (the *object* of τ). For example, the RDF triple $\langle$`letter22 sender henriPoincaré`$\rangle$ states that `letter22` was sent by Henri Poincaré. An *RDF graph* represents a set of RDF triples: if $\langle s \;\; p \;\; o \rangle$ belongs to an RDF graph $\mathcal{G}$ then s and o are nodes of $\mathcal{G}$, and p labels the edge (s, o) of $\mathcal{G}$. The set of named resources of an RDF graph $\mathcal{G}$ is denoted by $\texttt{Res}(\mathcal{G})$.

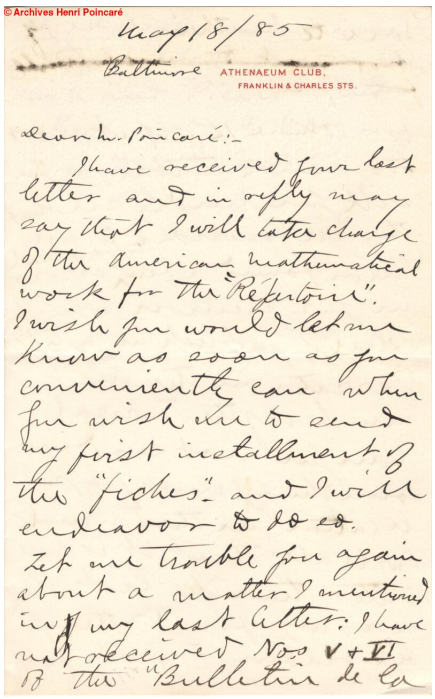
RDFS is a logic whose language is RDF and inference relation $\vdash$ is defined by a set of inference rules, that are based on some given resources: `rdf:type`, `rdfs:subclassof`, `rdfs:subpropertyof`, `rdfs:domain` and `rdfs:range`, respectively abbreviated in `a`, `subc`, `subp`, `domain` and `range`.

[6] http://sphinxsearch.com.

[7] OAI-PHM is a protocol developed by the Open Archives Initiative for harvesting metadata description of archives. It is mainly based on the Dublin Core model. See http://www.openarchives.org/pmh/.

[8] https://omeka.org/s/.

[9] For now, this ontology is rather basic. It still requires an epistemological investigation which will rely on other ontologies in digital humanities such as the CIDOC-CRM ontology (http://www.cidoc-crm.org).



(a) A digitized letter.

Baltimore, May 18/85[1]
Dear M. Poincaré:
I have received your last letter and in reply may say that I will take charge of the American Mathematical work for the "Répertoire".
I wish you would let me know as soon as you conveniently can when you wish me to send my first installment of the "fiches", and I will endeavor to do so.[2]
Let me trouble you again about a matter I mentioned in my last letter: I have not received Nos V+IV of the "Bulletin de la Société Mathématique" for volume 12, if it is not too much trouble for you will you kindly see that they are forwarded to me.[3] I have found a few typographical errors in your memoir which I shall publish in a table of errata in the next number of the Journal. Can you not send me something more for publication in the Journal?[4] I will see that it is done better than the last, which was my fist attempt at reading French proof sheets. Please remember me kindly to Misters Hermite and Picard and believe me to be Faithfully yours

Thomas Craig

(b) Transcription of the letter (a).

Semantic annotation of the letter

```
⟨letterCraig10 sentBy thomasCraig⟩
⟨letterCraig10 sentTo henriPoincaré⟩
⟨letterCraig10 hasDate 1885-05-18⟩
⟨letterCraig10 hasPlaceOfExpedition
                          baltimore⟩
⟨letterCraig10 hasSubject répertoire⟩
⟨letterCraig10 quoteName charlesHermite⟩
⟨letterCraig10 quoteName émilePicard⟩
```

Semantic annotation of the critical apparatus

```
⟨letterCraig10 refersTo letterCraig8⟩
⟨letterCraig10 refersTo letterCraig9⟩
```

(c) Metadata associated with the letter (a) in a RDF syntax.

1. This letter was written with a letterhead paper of the *Atheneum club* – Franklin & Charles Sts.
2. Each referenced article is associated with an index card. When there are enough index cards for the same index entry, a directory card that contains about 10 references can be written. See the previous letter.
3. See letter 8.
4. See letter 9.

(d) Critical apparatus of the letter (a) translated from French.

Fig. 1. Example of transcription and metadata from a letter.

$\langle$letter3 quote orpheeAuxEnfers$\rangle$
$\langle$letter2 quoteTextOf albertEinstein$\rangle$
$\langle$letter4 earliestPossibleWritingDate 1881-02-05$\rangle$

Fig. 2. Examples of indexation by RDF triples.

Let p, q and r be properties, x and y be resources, and C, D, E and R be classes. The inference rules considered in this paper are

$$\frac{\langle x \ \texttt{a} \ C\rangle \ \langle C \ \texttt{subc} \ D\rangle}{\langle x \ \texttt{a} \ D\rangle} r_1, \quad \frac{\langle C \ \texttt{subc} \ D\rangle \ \langle D \ \texttt{subc} \ E\rangle}{\langle C \ \texttt{subc} \ E\rangle} r_2,$$
$$\frac{\langle p \ \texttt{subp} \ q\rangle \ \langle q \ \texttt{subp} \ r\rangle}{\langle p \ \texttt{subp} \ r\rangle} r_3, \quad \frac{\langle x \ p \ y\rangle \ \langle p \ \texttt{subp} \ q\rangle}{\langle x \ q \ y\rangle} r_4,$$
$$\frac{\langle x \ p \ y\rangle \ \langle p \ \texttt{domain} \ D\rangle}{\langle x \ \texttt{a} \ D\rangle} r_5 \quad \text{and} \quad \frac{\langle x \ p \ y\rangle \ \langle p \ \texttt{range} \ R\rangle}{\langle y \ \texttt{a} \ R\rangle} r_6.$$

The (RDFS) deductive closure of an RDF graph $\mathcal{G}$ is

$$\mathcal{G}^{\vdash} = \{\tau : \text{ RDF triple} \mid \mathcal{G} \vdash \tau\}$$

An RDF graph $\mathcal{G}$ can be partitioned into $\mathcal{G} = \mathcal{O} \cup \mathcal{D}$ where $\mathcal{O}$ is an *ontology* (containing all the triples whose predicates are `subc`, `subp`, `domain` or `range`) and $\mathcal{D}$ gathers the *data* (that are about the individuals). The RDFS graph $\mathcal{G}_{ex}$ of Figure 3 is used as an example in the remainder of the paper.

SPARQL is a query language for RDF. In this article, SPARQL querying is assumed to be performed by an engine using RDFS entailment, meaning that querying an RDF graph $\mathcal{G}$ by a SPARQL query `Q` gives the same result as interrogating $\mathcal{G}^{\vdash}$ with `Q`. Moreover, the SPARQL queries considered have the following syntax: SELECT *vars* WHERE $\{body\}$, where *vars* is a sequence of anonymous resources and *body* is a sequence of statements separated by ".", a statement being either an RDF triple written without $\langle$ and $\rangle$ (i.e., $s \ p \ o$ instead of $\langle s \ \ p \ \ o\rangle$) or a filter statement of the form FILTER $(condition)$, where *condition* is a Boolean expression using Boolean operators, equality, and inequality relations on numbers and dates. Moreover, without loss of generality, it is assumed that only one filter statement occurs in the body of a SPARQL query. For example, the following is a query for letters sent by Henri Poincaré to a scientist before 1898:

$$\text{SELECT } ?\ell \text{ WHERE } \left\{\begin{array}{l} ?\ell \ \texttt{sentBy} \ \texttt{henriPoincaré} \ . \\ ?\ell \ \texttt{sentTo} \ \texttt{?x} \ . \\ \texttt{?x} \ \texttt{a} \ \texttt{Scientist} \ . \\ ?\ell \ \texttt{sendingDate} \ \texttt{?d} \\ \text{FILTER } (\texttt{?d} < \texttt{1898-01-01}) \end{array}\right\}$$

More generally, the execution of a SPARQL query Q on an RDF graph $\mathcal{G}$ consists in finding matchings between the body of the query to $\mathcal{G}$ and results as a set of *bindings*, a binding being an assignment of each variable by a resource or a literal, this assignment having to satisfy the constraint given in the filter statement. The result of this execution, denoted by $\texttt{exec}(Q, G)$, is the set of bindings restricted to the variables given after the SELECT keyword. For example:

$$\text{if } \texttt{Q} = \left|\begin{array}{l} \text{SELECT } \texttt{?p} \ \texttt{?o} \\ \text{WHERE } \{\texttt{letter22} \ \texttt{?p} \ \texttt{?o}\} \end{array}\right.$$

then $\texttt{exec}(\texttt{Q}, \mathcal{G}_{ex})$

$=$

?p	?o
a	Letter
sentBy	henriPoincaré
sentTo	felixKlein
correspondent	henriPoincaré
correspondent	felixKlein
topic	fuchsianFunc

4. Adding an RDFS infrastructure on an Omeka S site

The Henri Poincaré website is hosted and managed using a Huma-Num service. Huma-Num is a French infrastructure dedicated to Digital Humanities [13]. A system, branded "Huma-Num Box", has been developed to address several issues related to the treatment of data for Humanities (scalability, volume, accessibility, security, etc.). Using such a service enhances the development of standards when dealing with Digital Humanities. The Omeka S instance (vocabularies, content, website modules, etc.) has been installed on a shared server.

Omeka S comes with a search engine called Solr,[10] that lets users query the database to get information about transcribed letters. A specific configuration has been made by using lemmatization to improve search results. Although it can be very powerful, this search tool is limited in some situations and does not take advantage of the Semantic Web technologies. That is why the need to install a SPARQL endpoint has emerged. Omeka S data is stored using a MySQL dedicated database which cannot be directly interrogated through SPARQL queries. A specific RDFS base had

[10] Solr is an open source search platform built on Apache Lucene.

$\mathcal{G}_{ex} = \mathcal{O}_{ex} \cup \mathcal{D}_{ex}$

$\mathcal{O}_{ex} = \{\langle$`Mathematician subc Scientist`$\rangle, \langle$`Scientist subc Person`$\rangle,$

$\langle$`Person subc Agent`$\rangle, \langle$`Institution subc Agent`$\rangle,$

$\langle$`sentBy subp correspondent`$\rangle, \langle$`sentTo subp correspondent`$\rangle,$

$\langle$`correspondent domain Letter`$\rangle, \langle$`correspondent range Agent`$\rangle,$

$\langle$`topic domain Letter`$\rangle, \langle$`topic range Topic`$\rangle\}$

Meaning that mathematicians are scientists, that scientists are persons, that persons and institutions are agents, that sender and recipient of a letter are correspondents of this letter, that if x has for correspondent y then x is a letter and y is an agent, and that if x has for topic y then x is a letter and y is a topic.

$\mathcal{D}_{ex} = \{\langle$`henriPoincaré a Mathematician`$\rangle, \langle$`felixKlein a Mathematician`$\rangle,$

$\langle$`göstaMittagLeffler a Mathematician`$\rangle, \langle$`letter22 sentBy henriPoincaré`$\rangle,$

$\langle$`letter22 sentTo felixKlein`$\rangle, \langle$`letter22 topic fuchsianFunc`$\rangle,$

$\langle$`letter33 sentBy henriPoincaré`$\rangle, \langle$`letter33 sentTo göstaMittagLeffler`$\rangle,$

$\langle$`letter33 writtenDate 1881-07-26`$\rangle, \langle$`letter33 topic complexAnalysis`$\rangle\}$

Meaning that Henri Poincaré, Felix Klein and Gösta Mittag-Leffler are mathematicians, that `letter22` is sent by Henri Poincaré to Felix Klein, that a topic of `letter22` is Fuchsian functions, that `letter33` is sent by Henri Poincaré to Gösta Mittag-Leffler, and was written the 26^{th} of July, 1881, and that a topic of `letter33` is complex analysis.

$\mathcal{G}_{ex}^{\vdash} \supseteq \{\langle$`Mathematician subc Person`$\rangle, \langle$`henriPoincaré a Scientist`$\rangle,$

$\langle$`felixKlein a Scientist`$\rangle, \langle$`letter22 a Letter`$\rangle, \langle$`fuchsianFunc a Topic`$\rangle\}$

The deduced triples which are given there mean that mathematicians are persons, that Henri Poincaré and Felix Klein are scientists, and that `letter22` is a letter and that Fuchsian functions "is" a topic.

Fig. 3. An RDF graph $\mathcal{G}_{ex}$ and a part of its deductive closure.

to be installed. In addition to the server used to manage Omeka S installation, a virtual machine has been configured in which a dedicated Java application is running to manage an RDF database. This application uses the *Jena* engine [14] to manipulate the RDF documents and to execute SPARQL queries. The textual RDF syntax Turtle [15] was chosen to write the triples because it is easy to read. An associated interface has been developed and is embedded on the Omeka S website (with no impact on user navigation).

An automatic script retrieves data from Omeka S to update this RDFS base on a daily basis.[11] If an ontology is modified, it will also be updated. At the time of writing this paper, the RDFS base is composed of more than $200\,000$ triples. The ontology used contains 17 classes and 60 properties, and the database is gathering more than $6\,000$ intances. For instance, there are around $2\,100$ letters, $1\,700$ instances of the class `Person`, 700 documents of the class `Article` and more than 80 identified `ArchivePlace`.

On the website, a user can choose between using the basic input search and creating more complex queries with SPARQL. Three SPARQL query editing modes are proposed:

[11] However, although Omeka S allows export to different data formats, Turtle is not part of it. The script exports Omeka S data in *JSON-LD* format, and then converts it to Turtle before updating the RDFS base.

Classical mode The user can directly write SPARQL queries, within a text area, to access the RDFS database It allows creating complex queries by taking advantage of the expressiveness of the SPARQL language. But this requires a good understanding of SPARQL syntax which is not well suited for historians of science and people that are not familiar with the Semantic Web technologies.

Form-based mode A form containing a set of inputs is proposed to the user to help him/her building the query. This is a mode suitable for all users, as it does not require any specific knowledge.

Graphical mode A graphical interface is presented to let the user construct a graph corresponding to an inference graph. This mode can be a good compromise because it is not too difficult to apprehend, but also keeps a certain expressiveness when formulating queries. The interface has been created using *D3.js* library, which is adapted for manipulating documents based on data [16].

It is important to mention that the different blocks shaping the website are invisible for users. They can easily retrieve data they are interested in by choosing the appropriate search engine. In practice, for the daily use, historians and experts of the domain tend to use the *form-based* and *graphical* modes. Figure 4 illustrates the website architecture presented above. For the historians, Omeka S grant access to a specific back interface to visualize and update data (collections, vocabularies and content). As the data transfer from Omeka S to the RDFS base is automatic, they do not need to manage this particular task.

Consider a user who wants to express the following informal query using SPARQL:

$$\mathcal{Q} = \left| \begin{array}{l} \text{“Give me the letters sent} \\ \text{ or received by Paul Appell} \\ \text{ between } 1890 \text{ and } 1905\text{”} \end{array} \right.$$

He/she can use any of the three modes to express the SPARQL query and to find the corresponding letters. Figure 5 shows the edition of this query in the three modes.

5. Approximate and explained search in the Henri Poincaré correspondence

To be exploitable, the Henri Poincaré correspondence has to be queried. SPARQL querying, with adequate user interfaces, serves this purpose. However, more flexible searches can be useful, which is explained in Section 5.1. The way flexible searches are managed is based on SPARQL query transformations, as explained in Section 5.2. These transformations are based on rules, and a tool for expressing and managing these rules is briefly presented in Section 5.3. This work is implemented and works well, but can be improved: some future improvements are described in Section 5.4.

5.1. Motivating approximate and explained searches

Two reasons why flexible search is useful in the context of the correspondence of Henri Poincaré are presented below and exemplified.

Taking into account vagueness. Consider the following informal query:

$$\mathcal{Q} = \left| \begin{array}{l} \text{“Give me the letters sent by Henri} \\ \text{ Poincaré to Felix Klein at the} \\ \text{ end of the } 19^{\text{th}} \text{ century”} \end{array} \right. \quad (1)$$

The time period “end of the 19^{th} century” is vague, as many notions used by human beings. For example, the years 1876 and 1903 might be considered to belong to this time period.[12] By contrast, there is a large concensus on the year interval $[1890, 1900]$ to be a part of this time period. Therefore, if two letters ℓ_1 and ℓ_2 have Henri Poincaré as sender and Felix Klein as recipient, ℓ_1 written in 1892 and ℓ_2, in 1876, then ℓ_1 undoubtedly is an answer to $\mathcal{Q}$ but ℓ_2 could be considered also as an answer to $\mathcal{Q}$.

One way to handle these kinds of informal queries, where boundaries of notions are imprecise, consists in using the tools of fuzzy set theory: a fuzzy query with a kernel corresponding to the years $[1890, 1900]$ seems appropriate. However, for the examples presented below, fuzzy set theory seems to be inappropriate or, at least, incomplete.

Finding related results. Now, consider a historian of science querying the corpus about the query `Q` of the letters sent by Felix Klein to Henri Poincaré that are

[12] Some historians of science state that the end of the 19^{th} century is the year 1905 during which Albert Einstein published three of his most important articles, while some historians of geopolitics consider that the 20^{th} century starts in 1914.

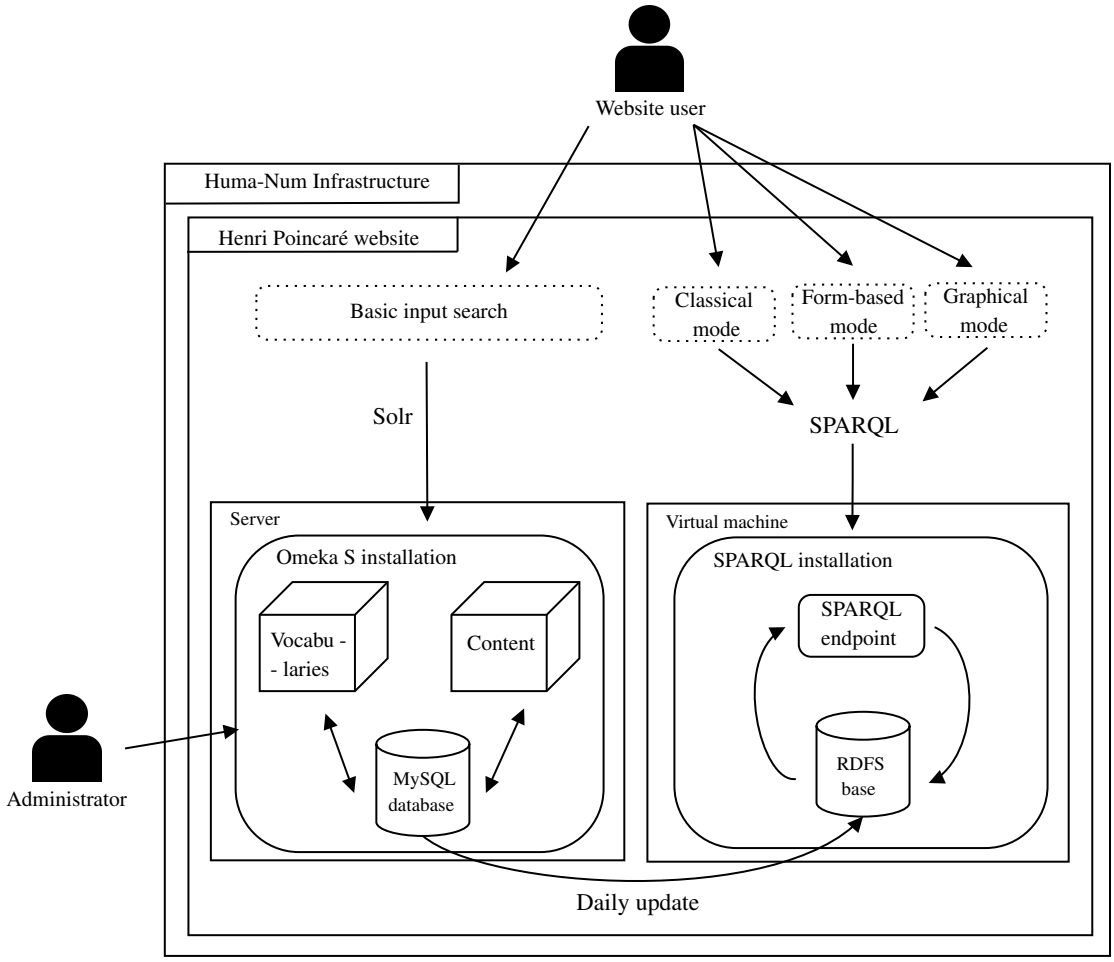


Fig. 4. The architecture of the Henri Poincaré website.

about complex analysis. Such a query can be easily formulated in SPARQL:

$$\texttt{Q} = \begin{array}{r} \text{SELECT } ?\ell \\ \text{WHERE} \end{array} \left\{ \begin{array}{l} ?\ell \texttt{ a Letter .} \\ ?\ell \texttt{ sentBy felixKlein .} \\ ?\ell \texttt{ sentTo henriPoincaré .} \\ ?\ell \texttt{ topic complexAnalysis} \end{array} \right. \quad (2)$$

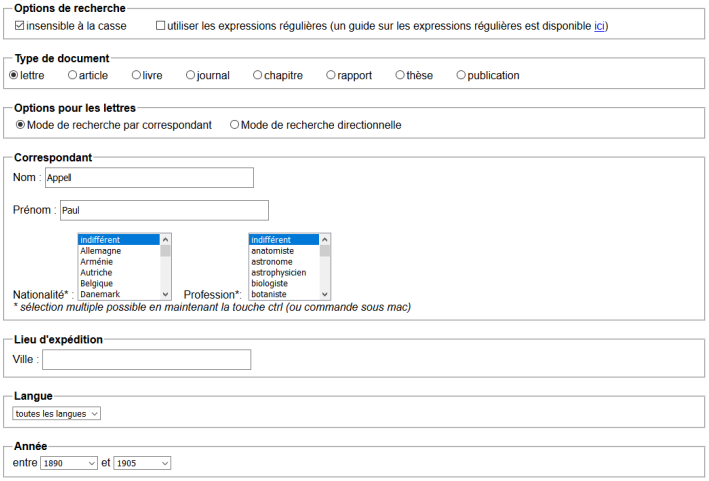
The execution of this query gives a set of letters that match exactly the formulated query Q. Now, imagine that the historian wants to go further, to get more letters related to the query Q. This may occur in particular in the following situations:

- The set of results is empty or, at least, too small (for the historian accessing the corpus): the historian may desire that the system provides more letters associated to explanations pointing out the mismatch between these letters and the query, something like "This is *not* a letter about complex analysis but it is about analysis."
- The corpus is incomplete: there exist letters sent or received by Henri Poincaré that do not exist anymore and also, there probably exist such letters that are not yet discovered. However, there may be a letter ℓ_{reply} of the corpus sent by Henri Poincaré to Felix Klein that is a reply to a letter exactly matching Q but that is not in the corpus. Therefore ℓ_{reply} should be interesting for the historian whose initial query was Q despite the fact that $\ell_{\text{reply}} \notin \texttt{exec}\,(\texttt{Q}, \mathcal{G}_{\text{HP}})$.

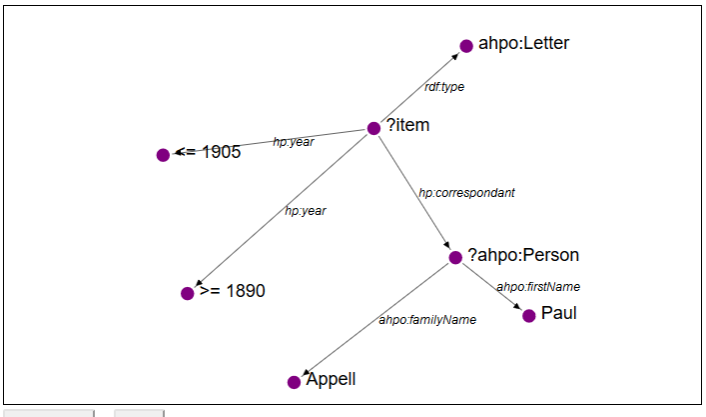
```
prefix ahpo:  <http://e-hp.ahp-numerique.fr/ahpo#>
prefix foaf:  <http://xmlns.com/foaf/0.1/>
prefix locn:  <http://www.w3.org/ns/locn#>
prefix o:     <http://omeka.org/s/vocabs/o#>
prefix cnt:   <http://www.w3.org/2011/content#>

SELECT DISTINCT ?titre ?date ?document ?lien WHERE {
?item dcterms:title ?titre
?item hp:created ?date .
?item hp:correspondant ?person .
?person ahpo:familyName ?test_corres_name .
FILTER(LCASE(?test_corres_name) = LCASE("Appell")) .
?person ahpo:firstName ?test_corres_firstname .
FILTER(LCASE(?test_corres_firstname) = LCASE("Paul")) .
?item hp:year ?date_intervalle .
?item hp:link ?lien .
?item rdf:type ?type .
?type hp:nicename ?document .
FILTER(?type = ahpo:Letter) .
 FILTER( ?date_intervalle >=  1890 && ?date_intervalle <= 1905 ) .
} order by ASC(?date)
```

Exécuter

(a) Classical mode (SPARQL query syntax).



(b) Form-based mode.



Rechercher Reset

(c) Graphical mode.

Fig. 5. SPARQL query editing modes available on Henri Poincaré website (http://henripoincare.fr/s/correspondance/page/sparql).

Different tools and methods have been proposed to deal with the notion of approximation when using SPARQL querying. The *f-SPARQL* engine [17] is a flexible extension of SPARQL which introduces the use of fuzzy set theory. Several new terms and operators (HIGH, RECENT, CLOSE TO, AT MOST, etc.) are proposed. Other approaches try to integrate user preferences within the SPARQL query. The *PrefSPARQL* extension [18] introduces new operators (HIGHEST, LOWEST, AROUND, MORE THAN, etc.) which can be used within SPARQL FILTER clauses. New clauses are also proposed (PREFERRING and PRIOR TO).

Query transformations can be useful for taking into account vagueness and for finding results that are related to the query but do not match it exactly. A method proposes a framework to relax query by analysing the failing causes of the initial query [19].

Another line of work related to this issue is the approach of case retrieval in case-based reasoning based on query transformations that has been applied in various application domains with various query languages (e.g., in machine translation [20], organic chemistry synthesis [21] and cooking [22]).

An approach based on the use of transformations rules is explained in the next section.

5.2. SPARQL query transformations

Consider first the informal query $\mathcal{Q}$ of equation (1). Let a and b be two integers with $a \leqslant b$. Let $\mathtt{Q}_a^b$ be the query for the letters sent by Henri Poincaré to Felix Klein between year a and year b:

$$\mathtt{Q}_a^b = \text{SELECT } ?\ell \text{ WHERE } \left\{ \begin{array}{l} ?\ell \texttt{ a Letter .} \\ ?\ell \texttt{ sentBy felixKlein .} \\ ?\ell \texttt{ sentTo henriPoincaré .} \\ ?\ell \texttt{ sentInYear ?y} \\ \text{FILTER (?y >= } a \text{ AND ?y <= } b\text{)} \end{array} \right\}$$

Let L_a^b be the set of letters resulting from the execution of the query $\mathtt{Q}_a^b$ on the RDFS graph $\mathcal{G}_{\text{HP}}$ of the Henri Poincaré correspondence ($L_a^b = \{\ell \mid (?\ell, \ell) \in \texttt{exec}\left(\mathtt{Q}_a^b, \mathcal{G}_{\text{HP}}\right)\}$). A letter $\ell_0 \in L_{1890}^{1900}$ is, under a strong consensus, an answer to the informal query $\mathcal{Q}$. A letter $\ell_1 \in L_{1885}^{1900} \setminus L_{1890}^{1900}$ can be accepted as an answer to $\mathcal{Q}$, but it is more debatable. For a letter $\ell_1 \in L_{1880}^{1900} \setminus L_{1885}^{1900}$, this is even more debatable. Thus, the idea is to generate sets of letters, starting from L_{1890}^{1900} and enlarging progressively the interval by steps of 5 years to get results that could be considered as answers to $\mathcal{Q}$ but with less and less relevance.

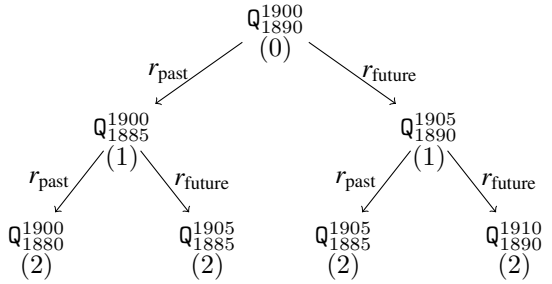


Fig. 6. A search tree (truncated at depth 2) for an informal query related to the end of the 19th century (penalties are below the queries, assuming $\texttt{cost}(r_{\text{past}}) = \texttt{cost}(r_{\text{future}}) = 1$).

To implement this idea, the notion of SPARQL query transformation rules is considered. Let r_{past} and r_{future} be two such rules that are both applicable to a query $\mathtt{Q}_a^b$ and such that their application to this query respectively gives $\mathtt{Q}_{a-5}^b$ and $\mathtt{Q}_a^{b+5}$. For example, $\mathtt{Q}_{1890}^{1900} \xrightarrow{r_{\text{past}}} \mathtt{Q}_{1885}^{1900}$. A search tree can be defined using SPARQL queries as states and these two rules to generate successor states. Figure 6 presents this search tree. To each rule r is associated a transformation cost $\texttt{cost}(r) > 0$ that is assumed to be additive: this cost is used to associate to a query generated by rules a penalty such that if $\mathtt{Q}$ has a penalty of π and $\mathtt{Q} \xrightarrow{r} \mathtt{Q}'$, then a penalty $\pi' = \pi + \texttt{cost}(r)$ is associated to $\mathtt{Q}'$.

Thus, the informal query $\mathcal{Q}$ is modeled by an *elastic query* $\left(\mathtt{Q}_{1885}^{1900}, \{r_{\text{past}}, r_{\text{future}}\}\right)$. More generally, an elastic query is an ordered pair $(\mathtt{Q}, R)$ where $\mathtt{Q}$ is a SPARQL query and R is a finite set of SPARQL query transformation rules. The result of an elastic query is a stream of letters ordered by increasing penalty and resulting from a search in the tree associated to $(\mathtt{Q}, R)$: the states of this tree are SPARQL queries with penalties $(\mathtt{Q}, \pi)$, the root of this search tree is $(\mathtt{Q}, 0)$ a successor of a tree node $(\mathtt{Q}, \pi)$ is a node $(\mathtt{Q}', \pi')$ such that there is a $r \in R$ that can be applied on $\mathtt{Q}$ with $\mathtt{Q} \xrightarrow{r} \mathtt{Q}'$ and $\pi' = \pi + \texttt{cost}(r)$.

For the example of the query $\mathtt{Q}$ of equation (2), the elastic query $(\mathtt{Q}, R)$ can be considered, with R a set of query transformation rules such as the ones defined informally below:

- ($r_{\texttt{GenPred}}$) Substitution of the predicate p of a triple by a superproperty q of p (i.e., $\mathcal{O} \vdash \langle p \texttt{ subp } q \rangle$).
- ($r_{\texttt{GenObjInst}}$) Substitution of the object o of a triple $\langle s \;\; p \;\; o \rangle$ when it is an instance of a class C by

a variable instance of this class (i.e., replace this triple by the triples $\langle s\ p\ \texttt{?x}\rangle$ and $\langle \texttt{?x a } C\rangle$).

($r_{\texttt{exchangeSenderRecipient}}$) If the body of a query Q has two triples of the form $\langle s\ \texttt{sentBy}\ o_1\rangle$ and $\langle s\ \texttt{sentTo}\ o_2\rangle$ then this rule can be applied on it and its application consists in replacing these triples with $\langle s\ \texttt{sentTo}\ o_1\rangle$ and $\langle s\ \texttt{sentBy}\ o_2\rangle$.

($r_{\texttt{substByColleague}}$) If the body of a query Q has a triple $\langle s\ p\ o\rangle$ such that $\mathcal{O} \vdash \langle o\ \texttt{worksWith}\ c\rangle$ then the rule substitutes this triple by $\langle s\ p\ c\rangle$. E.g., this transforms a query from Felix Klein by a query from a colleague of this mathematician.

It is noteworthy that the first two rules are generalization rules that can be applied to other domains.[13] Several such rules can be defined (generalization of classes in subject or object, removal of a triple, etc.) The two other rules are not generalization rules and are rule defined for the application domain of a correspondence corpus.

5.3. SQTRL

SQTRL (SPARQL Query Transformation Rule Language) is a language associated with a tool designed for handling SPARQL query transformations. This tool has been used in various application contexts [23].

An SQTRL transformation rule r is characterized by the following fields:

- $\texttt{name}(r)$: an identifier of the rule by a string;
- $\texttt{context}(r)$: a set of RDFS triples;
- $\texttt{left}(r)$: a set of RDFS triples;
- $\texttt{right}(r)$: a set of RDFS triples;
- $\texttt{cost}(r)$: a positive float;
- $\texttt{explanation}(r)$: a text describing the transformation that may contain blank nodes occurring in the fields `context`, `left` and `right`.

The RDFS triples of such a rule appear in SPARQL syntax.

Given an RDFS graph $\mathcal{G}$, a SPARQL query Q, and an SQTRL rule r, r is applicable on Q (given $\mathcal{G}$) if $\texttt{context}(r)$ can be bound with the graph $\mathcal{G}$ and $\texttt{left}(r)$ can be bound with the body of Q, provided that these bindings are consistent: if ?x occurs in both $\texttt{context}(r)$ and $\texttt{left}(r)$, only the values x such that $(\texttt{?x}, x)$ appears in both bindings are kept. If so, the application of r gives queries Q′ such that $\texttt{left}(r)$ is substituted by $\texttt{right}(r)$, where blank nodes are substituted by their values in the bindings.

For example, consider first the query transformation rule $r_{\texttt{GenPred}}$, presented in Section 5.2 and corresponding to the generalization of a predicate p by a superproperty q of p. This rule can be described as follows ($r = r_{\texttt{GenPred}}$):

$\texttt{name}(r) = \texttt{"Generalize a property "} + \texttt{"in predicate position"}$
$\texttt{context}(r) = \texttt{?p subp ?q}$
$\texttt{left}(r) = \texttt{?s ?p ?o}$
$\texttt{right}(r) = \texttt{?s ?q ?o}$
$\texttt{cost}(r) = \texttt{1.0}$
$\texttt{explanation}(r) = \texttt{"Generalize ?p in ?q"}$

The rule $r_{\texttt{GenPred}}$ is a generalization rule that can be applied to many application contexts. As pointed out in Section 5.2, it is possible to add domain-dependent rules. For example, the rule $r = r_{\texttt{exchangeSenderRecipient}}$ can be represented as follows:

$\texttt{name}(r) = \texttt{"Exchange sender and recipient"}$
$\texttt{context}(r) =$ *(empty context)*
$\texttt{left}(r) = \texttt{?s sentBy ?o1 . ?s sentTo ?o2}$
$\texttt{right}(r) = \texttt{?s sentTo ?o1 . ?s sentBy ?o2}$
$\texttt{cost}(r) = \texttt{1.0}$
$\texttt{explanation}(r) = \texttt{"Exchange sender ?o1 "} + \texttt{"and recipient ?o2"}$

An XML syntax has been chosen to properly define a transformation rule. As an example, Figure 7 illustrates the XML syntax associated with the rule $r_{\texttt{GenObjInst}}$ corresponding to the generalization of an object instance.

The SQTRL tool internally reuses the *Corese* engine [24]. Such an engine enables to query Semantic Web data, stored as RDF(S) files.

5.4. Future work on approximate and explained search

The current version of SQTRL has some limitations that require to be overcome. This section lists some of them and points out future studies for addressing them.

The first limitation is related to the costs associated to a rule: such a cost is a constant, whereas it is sometimes more relevant to have costs that depend on the bindings of anonymous resources occurring in the rule (in the fields `context`, `left` and `right`) at rule application time. For instance, the cost of generalizing a class C into a class D may depend on the "generalization leap" from C to D.

[13] A generalization rule is a rule r such that if $\texttt{Q} \xrightarrow{r} \texttt{Q}'$ then, for any RDFS graph $\mathcal{G}$, $\texttt{exec}(\texttt{Q}, \mathcal{G}) \subseteq \texttt{exec}(\texttt{Q}', \mathcal{G})$.

```
<rule name="Generalize an instance in object position by any instance of one of its classes">
  <context>?o a ?C</context>
  <left>?s ?p ?o</left>
  <right>?s ?p ?x . ?x a ?C</right>
  <cost> 1.0</cost>
  <explanation>Generalize ?o in any instance of ?C </explanation>
</rule>
```

Fig. 7. An example of SQTRL rule in XML syntax.

The second limitation is linked with the unnecessary applications of some rules given the rules already applied in the same branch of the search tree. For example, it is not necessary to apply sequentially twice the rule $r_{\texttt{exchangeSenderRecipient}}$ on a query $\texttt{Q}$, since it leads back to $\texttt{Q}$. Another example is linked with the rules r_{past} and r_{future}: it is unnecessary to apply both in the same branch since the set of results they add are already generated by queries with a lower or equal cost in the search tree. Therefore, in Figure 6, the two nodes labelled by $\texttt{Q}^{1905}_{1885}$ are both useless (the letters answering this query are in the union of the answers to queries $\texttt{Q}^{1900}_{1885}$ and $\texttt{Q}^{1905}_{1890}$ that are generated at a higher level in the tree). The objective of a future work is thus to avoid such unnecessary composition of rules, which would have a positive impact both on the computing time and (more importantly) on the user load, that would have to examine fewer generated results appearing several times with equivalent (but not necessarily equal) explanations.

The third limitation is related to the filter statements of the SPARQL queries. Currently, only filter statements representing intervals (e.g., r_{past} and r_{future}) are handled by SQTRL rules. More complex filter statements (involving, e.g., negations or disjunctions) are not considered currently. Taking them into account is a complex research direction, since complex filter statements are propositionally closed which involves that a purely syntactic handling is not sufficient if query transformation rules are considered at a semantical level (i.e., if the execution of two queries $\texttt{Q}$ and $\texttt{Q}'$ give the same results on any RDFS graph then a SQTRL rule should be applicable in the same manner on $\texttt{Q}$ or $\texttt{Q}'$).

The fourth limitation is more domain-dependent since it is related to the notion of time in history. For example, the rules r_{past} and r_{future} enlarge time intervals by steps of 5 years. By contrast, a historian often considers the time scale according to milestones (e.g., the year 1914 for political history and the year 1905 for history of physics). Therefore, using such milestones in time interval transformation rules, as well as their significance in the context of the search, is a challenging future work. For this purpose, ontologies for history[14] should be useful.

6. A tool for efficient editing indexing triples

Currently, the Henri Poincaré correspondence is indexed in a satisfying way by RDFS files. However, new properties may emerge from research in history that would require additional annotation work and, more importantly, new correspondence corpora exist that are not yet annotated and that are of interest for nearby colleagues in history of science. This justifies the development of an editing tool of RDF triples for the purpose of corpus annotation. The RDF graph $\mathcal{G}$ is partitioned in an ontology $\mathcal{O}$ that is supposed to be given and in a set of triples $\mathcal{D}$ that has to be edited by this tool. The development of this tool is an ongoing work that is described below in four parts: the edition tool, the use of deductive inferences in RDFS for improving the efficiency of this tool, the use of hypothetical inferences on RDFS for the same purpose using case-based reasoning principles, and a first evaluation of this work. Finally, some research directions on this issue are presented.

6.1. *RDFWebEditor4Humanities: an editor for indexing corpora*

The indexing work done by historians of science for the Henri Poincaré correspondence was mainly done using the user interface of Omeka S: the RDF files are generated by translating the information edited via Omeka S (in a SQL server) to RDF, as described before, in Figure 4. It was decided to implement an RDF triple editor prototype for the new annotation tool in order to benefit from the RDF(S) infrastructure.

Several tools and methods have been proposed to assist users in RDF data editing. Protégé [25] is one fre-

[14] In particular data for history, http://ontome.dataforhistory.org/.

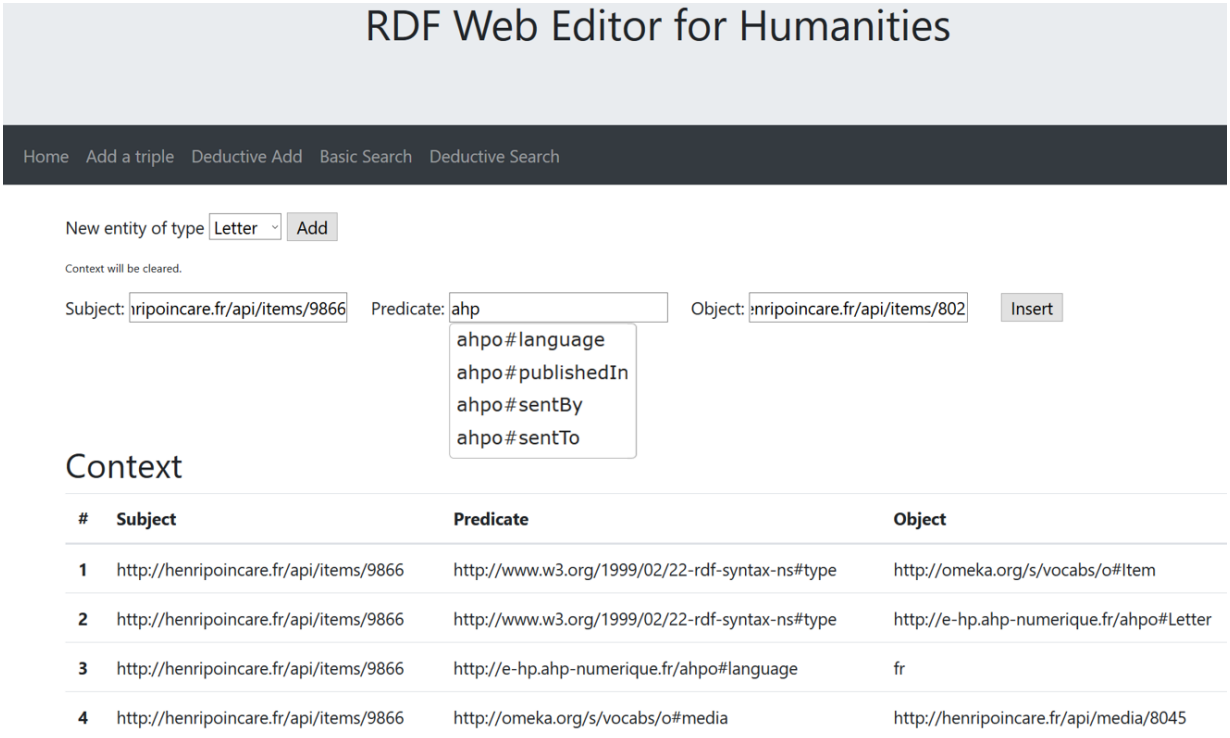


Fig. 8. A screenshot of RDFWebEditor4Humanities.

quently used tool which benefits from an user friendly interface but lacks of real assistance to find resources and to edit new triples. Several methods use language processing to let users edit RDF databases in a simpler way. For example, the GINO editing tool [26] introduces a guided and controlled natural language to define statements. Another example is the CLOnE language [27] which provides a method for editing data without any knowledge about Semantic Web standards.

Figure 8 presents a screenshot of the developed interface. An autocomplete mechanism has been implemented which assists users in writing triples values. Applying such a filter is particularly relevant when working with a base containing a large number of triples. The set of potential values for a field are ranked according to the alphabetical order.

The editor also displays the *context* of the edition. For the edition of a correspondence, the context is a letter and the already edited triples about this letter. If the current letter is `letter44`, the context is the set of triples $\langle$`letter44` p $o\rangle$ where p and o result from the execution of the query SELECT `?p ?o` WHERE $\{$`letter44 ?p ?o`$\}$.

When editing a state for which only one of the three values is missing, the tool avoids giving to the user a value that forms a triple already known. For example, when the subject s and the predicate p are already edited, the values o resulting from the query SELECT `?o` WHERE $\{$`s p ?o`$\}$ are not proposed.

Figure 8 example displays the context of the letter which is currently being edited. This letter is identified as item n°9866 of the Henri Poincaré database and its context contains four triples.

6.2. *Indexing support using RDFS deduction*

Consider an expert indexing the Henri Poincaré correspondence: $\mathcal{D}$ is the set of triples already edited at current time. At a given time, an *editing question* is raised, that is composed of (1) a triple with 1, 2 or 3 missing values, denoted by anonymous resources, and (2) a mark on the field that the editor wants to fill, denoted by framing the *highlighted* anonymous resource. For example, the editing question $\langle$`letter44` `?p` `?o`$\rangle$ corresponds to the state when the editor has already edited the subject of the triple but neither its predicate nor its object, and wants to edit now its predicate. The editing assistance pro-

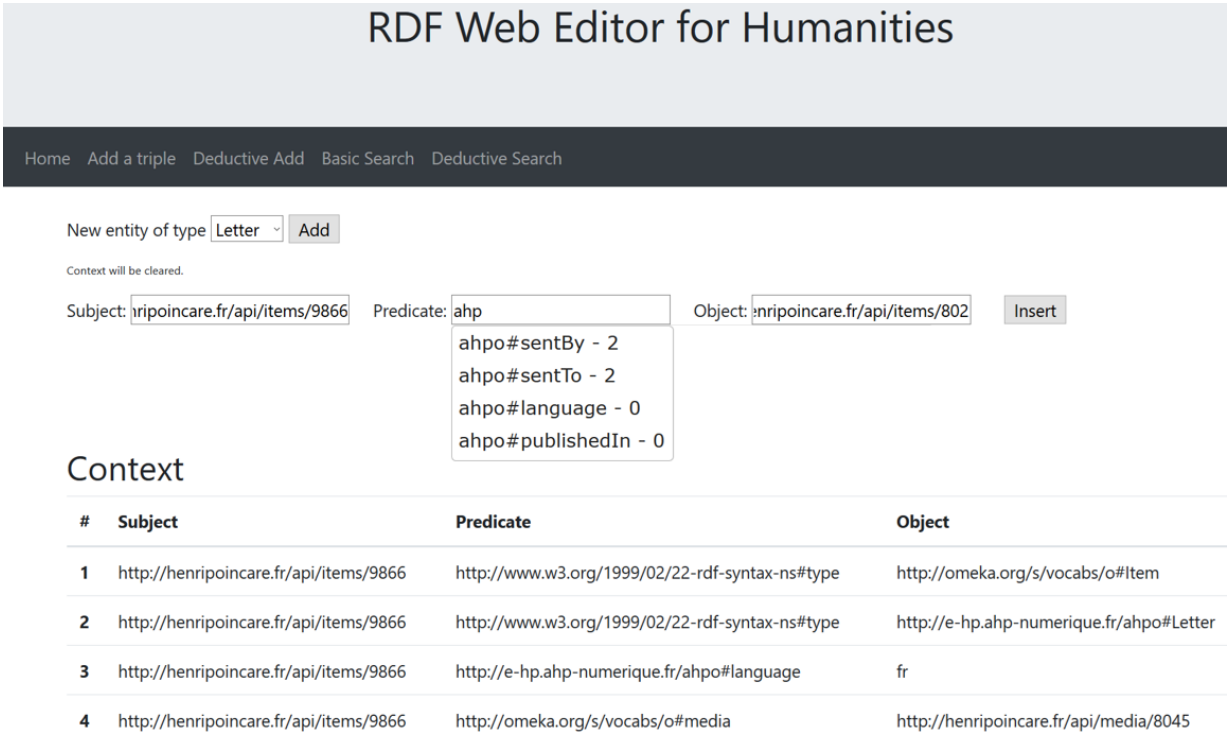


Fig. 9. A screenshot of RDFWebEditor4Humanities using RDFS deduction to rank the set of potential values.

posed here aims essentially at re-ranking the set of potential values for the highlighted field, the objective being that the rank of the value chosen by the expert is, on the average, lower than the rank given by the alphabetical order. For this purpose, some features about the missing values are deduced from $\mathcal{G} = \mathcal{O} \cup \mathcal{D}$ and the more features a potential value has, the higher its rank in the list will be. In the following, given a value c that is in the set of the candidate values for the highlighted field, c`.#f` denotes the number of features of c.

For example, consider the editing question $\langle$`letter44 sentTo` `?o`$\rangle$ and $\mathcal{G} = \mathcal{G}_{ex}$ from Figure 3. Once `?o` is replaced by a value o, it can be deduced that o is an instance of `Agent`:

$$\mathcal{G} \cup \{\langle \texttt{letter44 sentTo } o\rangle\} \vdash \langle o \texttt{ a Agent}\rangle$$

The set of candidate values for `?o` is $\texttt{Res}(\mathcal{G})$. For such a candidate value c, two situations may occur:

- S^+ Either $\mathcal{G} \vdash \langle c \texttt{ a Agent}\rangle$;
- $S^?$ Or $\mathcal{G} \nvdash \langle c \texttt{ a Agent}\rangle$: it cannot be deduced from $\mathcal{G}$ that c is an instance of `Agent`.

Let C^+ (resp., $C^?$) bet the set of $c \in \texttt{Res}(\mathcal{G})$ in situation S^+ (resp., $S^?$). If $c_1 \in C^+$ and $c_2 \in C^?$, all other things being equal, then c_1`.#f` $= c_2$`.#f` $+ 1$.

This editing question is of the type $\langle s \;\; p \;\; \boxed{\texttt{?o}}\rangle$. There are 12 editing question types: cf. first column in Figure 10. For an editing question of the type $\langle s \;\; p \;\; \boxed{\texttt{?o}}\rangle$, the knowledge about the ranges of p can be used (there may be several ranges for a property, each of them providing an additional constraint on the value for `?o`). Let R be such a range (i.e., $\mathcal{G} \vdash \langle p \texttt{ range } R\rangle$). If R is a datatype then the value for `?o` has to be a literal of this datatype. Otherwise, the value for `?o` is either new or has to be selected in $\texttt{Res}(\mathcal{G})$. Therefore, for each $c \in \texttt{Res}(\mathcal{G})$, if $\mathcal{G} \cup \{\langle s \;\; p \;\; c\rangle\} \vdash \langle c \texttt{ a } R\rangle$ then c`.#f` is incremented. This can be computed by executing the query SELECT `?c` WHERE $\{$`?c a` $R\}$ on $\mathcal{G}$: if $(\texttt{?c}, c)$ belongs to the result, c`.#f` is incremented. This is called `rangePred` in Figure 10. For editing questions of the type $\langle$`?s` $\;\; p \;\; o\rangle$, the idea is similar, except that `domain` is used instead of `range` (called `domainPred`).

Consider an editing question of the type $\langle s \;\; \boxed{\texttt{?p}} \;\; o\rangle$. First, the value to be edited for `?p` has to be a property, so a candidate value that is a property (written $\mathcal{G} \vdash \langle c \texttt{ a rdf:Property}\rangle$) has its feature count f`.#f` incremented (this is called `predProperty` in Figure 10). The value of s can also be used: if s is an instance of a class D then a candidate value c hav-

editing question type	domain-Pred	range-Pred	pred-Property	subject-InDomain	object-InRange	subjIm-Rel
⟨?s p o⟩	×					
⟨s ?p o⟩			×	×	×	
⟨s p ?o⟩		×				
⟨?s ?p o⟩						×
⟨?s ?p o⟩			×		×	
⟨?s p ?o⟩	×					
⟨?s p ?o⟩		×				
⟨s ?p ?o⟩				×		
⟨s ?p ?o⟩						×
⟨?s ?p ?o⟩						
⟨?s ?p ?o⟩			×			
⟨?s ?p ?o⟩						

Fig. 10. Editing question types and associated features of candidate values.

ing D as domain (i.e., $\mathcal{G} \vdash \langle c \ \texttt{domain} \ D\rangle$) has its c.#f incremented (called subjectInDomain). This can be computed thanks to the execution of the query SELECT ?c WHERE $\{s$ a ?D . ?c domain ?D$\}$ on $\mathcal{G}$. In a symmetrical way, the value of o can be used (with range instead of domain): this is called objectInRange in Figure 10.

For the editing questions of the type ⟨?s ?p o⟩, the feature count c.#f for the subject is increased each time c is in the domain of a property whose range contains o, which corresponds to the query SELECT ?c WHERE $\left\{\begin{array}{l}\texttt{?c a ?D .}\\ \texttt{?p domain ?D .}\\ \texttt{?p range ?R .}\\ o \ \texttt{a ?R}\end{array}\right\}$. For the editing questions of the type ⟨s ?p ?o⟩, a similar idea applies (with exchange of domain and range). For both types of editing questions, this is denoted by subjImRel(subject-image relationship) in Figure 10.

For editing questions of the type ⟨s ?p ?o⟩ (resp., ⟨?s ?p o⟩), the edition of a value for ?p is similar as for ⟨s ?p o⟩, except that range (resp., domain) is not used. For ⟨?s ?p ?o⟩, this also applies, but without the use of domain and range (the fact that the value of ?p should be a property is used).

For the other possibilities no re-ranking is provided in the current version of the tool.

Figure 9 illustrates the use of RDFS deduction in the edition tool. The editing question is the same as the one presented in Figure 8. This is a question of the type ⟨s ?p o⟩, where s is a Letter (letter9866) and o is a Person (thomasCraig). Imagine that the user wants to define thomasCraig as the sender of the letter9866. The question is ⟨letter9866 ?p thomasCraig⟩. The graph contains the triples:

⟨sentBy domain Letter⟩,
⟨sentTo domain Letter⟩,
⟨sentBy range Person⟩,
⟨sentTo range Person⟩,
⟨letter9866 a Letter⟩, and
⟨thomasCraig a Person⟩

As the class Letter is domain of the properties sentBy and sentTo, the counts associated with the two properties are incremented. Moreover, the class Person is range of the two properties which also increases the counts by one. The property sentBy is now the first proposition among the values proposed by the tool while it was in third position when using the basic editor. This version of the tool provides a better ranking in this situation. It is important to point out that applying RDFS deduction does not add any sensitive delay to the use of the tool.

6.3. Indexing support using case-based reasoning

The work presented in this section is freely inspired from the research on the UTILIS system that assists users in RDFS graph updating [28] and on the case-based reasoning methodology.

Case-based reasoning (CBR [29]) aims at solving problems with the help of a case base, where a case

is a representation of a problem-solving episode. The target problem is the problem currently under resolution. A source case is an element of the case base. The reasoning process is usually composed of several steps: (1) *retrieval* chooses one or several source cases judged similar to the target problem, (2) *adaptation* modifies, if necessary, the retrieved case, (3) *storage* adds the newly formed case, possibly after a correction from the user. In this ongoing work, only retrieval is considered (adaptation is currently a mere copy and storage is quite straightforward).

In this application framework, a problem is an editing problem, defined by an *editing question* (as in the previous section), and the context of the problem (as defined in Section 6.1). For example, consider the following target problem for the running example developed below:

$\texttt{tgt} =$

question: ⟨`letter44 topic` `?o`⟩
ctxt: ⟨`letter44 sentBy felixKlein`⟩, ⟨`letter44 sentTo henriPoincaré`⟩

A solution is a plausible value for the highlighted field in the editing question of the problem. The set of edited data, $\mathcal{D}$, is used as a case base. It can be noticed that $\mathcal{D}$ is not a set of problem-solution pairs, but appears as a whole in which such pairs can be "cut out". Such situations occur for other CBR systems, as depicted already in [29].

The retrieval step aims at extracting cases from $\mathcal{D}$. This gives candidate solutions that are proposed for the edition and ranked according to the similarity to the target problem (the more similar the cases are to the target problem, the higher in the list they are proposed). For example, consider $\mathcal{D} = \mathcal{D}_{ex}$ from Figure 3. Both letter instances `letter22` and `letter33` have topics associated to them:

$$\mathcal{G}_{ex} \vdash \langle \texttt{letter22 topic fuchsianFunc} \rangle$$
$$\mathcal{G}_{ex} \vdash \langle \texttt{letter33 topic complexAnalysis} \rangle$$

Both topics can be proposed as candidate values, but which one should be proposed before the other? The answer is based on the similarity between the letters of $\mathcal{D}$ and the letter of the target problem: if `letter22` is more similar to `letter44` than `letter33` then the value `fuchsianFunc` is proposed before the value `complexAnalysis`.

So, the question is how to assess the similarity between a letter and the target problem. The answer proposed for the UTILIS system cited above is based on query relaxation. For RDFWebEditor4Humanities, this idea is reused, and query relaxation is replaced with SQTRL query transformation (cf. Section 5). More precisely, the target problem `tgt` is transformed into a SPARQL query $\texttt{Q}_{\texttt{tgt}} = \text{SELECT } \textit{var} \text{ WHERE } \{\textit{body}\}$, where *var* is the variable highlighted in the editing question (*var* = `?o` in the running example) and *body* is constituted by the triple of the editing question and the triples of the context, with the substitution of the current letter (`letter44` in the example) by a variable $?\ell$. This gives, for the running example:

$$\texttt{Q}_{\texttt{tgt}} = \text{SELECT ?o WHERE } \left\{ \begin{array}{l} ?\ell \texttt{ topic ?o .} \\ ?\ell \texttt{ sentBy felixKlein .} \\ ?\ell \texttt{ sentTo henriPoincaré} \end{array} \right\}$$

Then retrieval amounts to an approximate search, as described in Section 5, taking $\texttt{Q}_{\texttt{tgt}}$ as an initial query and a set of SQTRL rules. More precisely, the search tree of root $\texttt{Q}_{\texttt{tgt}}$ is developed with a given maximal transformation cost and then, for each `Q` of this tree associated with a transformation cost *tc*, the execution of `Q` on the edited graph gives a (possibly empty) set of values *v* (in the example, values for `?o`). These values *v* are then ranked according to the increasing value of the transformation cost *tc*, which provides the list of potential values for the expert that indexes the letters.

With the running example, consider only the SQTRL rules $r_{\texttt{GenPred}}$ and $r_{\texttt{GenObjInst}}$, defined in Section 5.2, with a cost of 1 for both. In the search tree of root $\texttt{Q}_{\texttt{tgt}}$, the query $\texttt{Q}_1$ is generated at depth 2 (corresponding to 2 applications of $r_{\texttt{GenPred}}$ on $\texttt{Q}_{\texttt{tgt}}$):

$$\texttt{Q}_1 = \text{SELECT ?o WHERE } \left\{ \begin{array}{l} ?\ell \texttt{ topic ?o .} \\ ?\ell \texttt{ correspondent felixKlein .} \\ ?\ell \texttt{ correspondent henriPoincaré} \end{array} \right\}$$

The execution of $\texttt{Q}_1$ on $\mathcal{D}_{ex}$ gives $\{(\texttt{?o}, \texttt{fuchsianFunc})\}$. At depth 3, the query $\texttt{Q}_2$ is generated (corresponding to 1 application of $r_{\texttt{GenObjInst}}$ on $\texttt{Q}_1$):

$\mathtt{Q}_2 =$

$$\text{SELECT } \mathtt{?o} \text{ WHERE } \left\{ \begin{array}{l} \mathtt{?}\ell \text{ topic ?o .} \\ \mathtt{?}\ell \text{ correspondent ?x .} \\ \mathtt{?x} \text{ a Mathematician .} \\ \mathtt{?}\ell \text{ correspondent henriPoincaré} \end{array} \right\}$$

The execution of $\mathtt{Q}_2$ on $\mathcal{D}_{ex}$ gives $\{(\mathtt{?o}, \mathtt{fuchsianFunc}), (\mathtt{?o}, \mathtt{complexAnalysis})\}$.

Finally, the transformation cost for obtaining `fuchsianFunc` is $tc = \min\{2, 3\} = 2$ and the one for `complexAnalysis` is $tc = 3$: the former is proposed before the latter in the list proposed during the editing process.

This work and its combination with the work of Section 6.2 are still in their early stages, so it is not evaluated yet and thus, not considered in the next section.

6.4. A first evaluation

A first evaluation has been carried out that is fully automated, though it is acknowledged that a complete evaluation of an editing tool has to involve human editors.

For this evaluation, the baseline is the editor tool using only alphabetical order to rank the values (cf. Section 6.1) and it is compared to the editing prototype using RDFS entailment (cf. Section 6.2). The autocomplete mechanism has been disabled for the automatic evaluation.

Given an editing question *eq*, the measure used in this evaluation is the rank $\mathtt{rank}(eq)$ of the value chosen by the editor in the list of the values proposed by the tool: the lower $\mathtt{rank}(eq)$ is, the better the tool is.

The test set is built on the basis of the current RDF graph $\mathcal{G}_{\text{HP}} = \mathcal{O}_{\text{HP}} \cup \mathcal{D}_{\text{HP}}$ that indexes the Henri Poincaré correspondence. More precisely, the ontology $\mathcal{O}_{\text{HP}}$ (that imports other ontologies, such as FOAF) is given and the edition of the data in $\mathcal{D}_{\text{HP}}$ is simulated as follows:

- Initially, $\mathcal{D} = \emptyset$, `Ranks` is the empty multiset.
- For each triple $\langle s \ \ p \ \ o \rangle \in \mathcal{D}_{\text{HP}}$ considered in a random order,
  - For each of the 3 fields subject, predicate and object, considered in a random order,
    - Let *eq* be the editing question corresponding to this field;
    - Let $\mathtt{rank}(eq)$ be the rank of the edited value (i.e., *s*, *p* or *o*) in the list of the proposed values given by the evaluated editing tool using $\mathcal{G} = \mathcal{O}_{\text{HP}} \cup \mathcal{D}$;
    - Add $\mathtt{rank}(eq)$ to `Ranks`;
  - Add $\langle s \ \ p \ \ o \rangle$ to $\mathcal{D}$.

	baseline (cf. §6.1)	using deduction (cf. §6.2)
average	12.5	9.3
standard deviation	11.4	8.1

Fig. 11. The average and the standard deviation of the values $\mathtt{rank}(eq)$, for editing questions *eq* simulated from the Henri Poincaré correspondence RDF graph, for two versions of the editor.

At the end $\mathcal{D} = \mathcal{D}_{\text{HP}}$ and `Ranks` is the multiset of the ranks. The average and standard deviation of the elements of `Ranks` are computed.

The results are presented in Figure 11.

Although the evaluation has been carried out on a subset of the Henri Poincaré Correspondence, this was constructed in order to be representative of the whole correspondence. What emerges of these results is that the use of RDFS deduction brings a better general ranking and thus could effectively assist historians during the indexing work.

6.5. Future work on assisted indexing

It is noteworthy that the tool presented here is still a prototype needing some technical and ergonomic improvements that are necessary before carrying out the complete evaluation involving human users.

The CBR approach presented in Section 6.3 is not fully implemented (and not evaluated) and this constitutes the first future work.

The combination of the deductive and CBR approaches for editing also has to be investigated. In particular, the following hypothesis is made: the editing improvements of these methods are largely independent one from the other, so a well-designed combination of them should give interesting results. One promising way to do such a combination is to apply one of these methods and then, to decide between the ties using the other method. More sophisticated combination techniques can also be investigated.

Another future work is related to the deductive approach and can be illustrated by the example of the editing question $\langle \mathtt{letter44} \ \boxed{\mathtt{?p}} \ \mathtt{?o} \rangle$ introduced in Section 6.2. Two types of situations were distinguished for a candidate value *c* for `?o`: S^+ and $S^?$. A third type of situations can be considered, provided that the representation language is extended to class complement (as in OWL DL and many of its fragments):

S^- $\mathcal{G} \vdash \langle c$ `a` $\neg$`Agent`$\rangle$.

where $\neg$`Agent` represents the entities that are *not* agents, $\mathcal{G}$ is the graph in its current state and c is a candidate value for `?o`. For example, if $\mathcal{G}$ entails that the classes `Letter` and `Agent` are disjoint, then any c such that $\mathcal{G} \vdash \langle c$ `a Letter`$\rangle$ is in situation S^-. For such a c, it is certain that it is not a potential value for `?o` and then it can be removed from the list of candidate values proposed to the expert. More generally using a more expressive logic than RDFS that contains some sort of negation would enable to remove elements from the list of candidates. The drawback of this extension is the computation time involved by using this more expressive logic, which is a particularly sensitive issue when dealing with a user interface. Thus, an option to be investigated would be to make offline deductions in this more expressive than RDFS logic and to use these deduced data online in order to point out elements that cannot be correct values for a given editing question.

A last potential future work would be to integrate this work in an existing RDF editing tool such as Neologism [30], Protégé [25] and Vitro [31].

7. Conclusion and Future Work

The Henri Poincaré correspondence is a corpus gathering letters received and sent by this famous scientist. This article has presented the application of Semantic Web technologies on this corpus. Before these technologies were used on this corpus, letters were digitized, transcribed in plain text and associated to metadata, within the content management system Omeka S.

Then, a translation script of these pieces of information into RDFS has been implemented, which has made possible the access to the corpus via an RDF endpoint using the vocabularies of standard ontologies and also the ontology developed for the purpose of this corpus. Therefore, the SPARQL querying became possible, and three user interfaces were developed for this purpose: a classical interface with the whole SPARQL querying, a form-based interface more suited to the habits of the users and an interface using a graphical view.

Now, SPARQL querying is sometimes insufficient, when approximate searches are needed. This is the case when vague notions have to be taken into account in the query (e.g., "the end of the 19^{th} century") or when results that are somehow related to the initial query can be of interest for the user. For this purpose, a mechanism based on the notion of elastic query has been designed. The result of the execution of an elastic query is a stream ordered by increasing cost, each result being associated with explanations of the mismatch between the SPARQL query and this result.

Currently, the indexing of the corpus is done via Omeka S. It is planned to do it directly in RDFS and an editing tool is currently under study. A prototype has been developed in which the user edits each element of a triple by selecting a resource in a list (or creating a new resource or a literal). This prototype has been improved by the use of RDFS entailment in order to propose most promising values first. Another (potential) improvement of this prototype using case-based reasoning principles has been presented and is an ongoing work.

These contributions have different degrees of maturity and have been presented from the more mature one (that is currently in use) to the less mature one (that is an ongoing work). To the best of our knowledge, there exists no other work that uses the methods presented in this article in the context of cultural heritage. Future works have been planned for approximate and explained search and for assisted indexing of triples (cf. Sections 5.4 and 6.5) and are not recalled here. Therefore, the first ongoing work consists in putting all these contributions in practice so that they are all as routine uses.

Another future work will aim at dealing with uncertainty and imprecision of some information about the letters, and, more specifically, about the dates. When the date is written in the preamble of a letter, it is reasonable to consider it as its writing date, but in many letters this information is missing. However, an imprecise date can be given, e.g., because it relates to an event (a historical event or a relation to another letter that is well-dated). In such situations, the precise date is often difficult to know, thus an imprecise date (e.g., giving only the month and the year) is associated to the letter. Sometimes, a precise yet hypothetical date can be inferred by historians, e.g., when the letter refers to something that is interpreted by a historical event. Dealing with imprecise and/or uncertain dates is a challenging issue that requires both a careful ontology modeling and inference mechanisms. Another challenge related to the ontology is the representation within it of mathematical formulae. Historians of science would benefit from the implementation of a search engine for mathematical content. Different methods have already been investigated [32] and

should be considered in order to propose an integration in the framework of this research.

In this paper, the representation formalism for the ontologies and for the data does not go beyond RDFS. However, in order to carry out some of the future studies described in this conclusion as well as in other sections of the paper, the use of larger fragments of OWL is likely to be required. For instance, in Section 6.5, it was shown how the use of negation can improve the editor using entailment. It was also noted in this same section that the issue of computing time increasing potentially involved by a more expressive formalism can be harmful to the practical use of such a system. Some offline deductions may be used to alleviate the online computing burden.

Although this work has been carried out for the Henri Poincaré correspondence, it is planned to be applied to other correspondences of known scientists. More widely, it should be reusable for another history of science corpora.

Acknowledgements

This work was supported partly by the French PIA project "Lorraine Université d'Excellence", reference ANR-15-IDEX-04-LUE. It was also supported by the CPER LCHN (Contrat de Plan État-Région Lorrain "Langues, Connaissances et Humanités Numériques") that financed engineer Ismaël Bada who participated to this project.

References

[1] H. Poincaré, *La Science et l'Hypothèse*, Ernest Flammarion, Paris, 1902.

[2] P. Nabonnand (ed.), *La correspondance entre Henri Poincaré et Gösta Mittag-Leffler*, Birkhäuser, Basel, 1998.

[3] S. Walter, E. Bolmont and A. Coré (eds), *La correspondance entre Henri Poincaré et les physiciens, chimistes et ingénieurs*, Birkhäuser, Basel, 2007.

[4] S. Walter, R. Krömer and M. Schiavon (eds), *La correspondance entre Henri Poincaré avec les astronomes et les géodésiens*, Birkhäuser, Basel, 2014.

[5] L. Rollet (ed.), *La correspondance de jeunesse d'Henri Poincaré: Les années de formation. De l'École polytechnique à l'École des Mines (1873-1878)*, Publications of the Henri Poincaré Archives, Springer International Publishing, Basel, 2017. ISBN 9783319559599.

[6] P. Wittek and W. Ravenek, Supporting the Exploration of a Corpus of 17th-Century Scholarly Correspondences by Topic Modeling, in: *Supporting Digital Humanities 2011: Answering the unaskable*, B. Maegaard, ed., Copenhagen, Denmark, 2011.

[7] A. Meroño-Peñuela, A. Ashkpour, M. van Erp, K. Mandemakers, L. Breure, A. Scharnhorst, S. Schlobach and F. van Harmelen, Semantic technologies for historical research: A survey, *Semantic Web Journal* (2015), 1–27. doi:10.3233/SW-140158. http://iospress.metapress.com/content/A842V11135QK5055.

[8] O. Bruneau, S. Garlatti, M. Guedj, S. Laubé and J. Lieber, SemanticHPST: Applying Semantic Web Principles and Technologies to the History and Philosophy of Science and Technology, in: *The Semantic Web: ESWC 2015 Satellite Events*, F. Gandon, C. Guéret, S. Villata, J. Breslin, C. Faron-Zucker and A. Zimmermann, eds, Lecture Notes in Computer Science, Vol. 9341, Springer International Publishing, 2015, pp. 416–427.

[9] J. Kucsma, K. Reiss and A. Sidman, Using Omeka to build digital collections: The METRO case study, *D-Lib magazine* **16**(3/4) (2010), 1–11.

[10] A.E. Gay, Using a Content Management System for Student Digital Humanities Projects: A Pilot Run, 2019.

[11] F.C. Paletta, M.M. Macambyra, S.L. Ferreira and V.M.A. Lima, Digital Library of the Artistic Production of ECA USP IFLA 20919 (2019).

[12] J.-M. Meunier, S. Szoniecky and D. Berthereau, Utilisation d'Omeka-S pour la conception et le partage de ressources pédagogiques, in: *Zotero & Omeka - des outils pour les humanités numériques*, Poitiers, France, 2019. https://hal-univ-paris8.archives-ouvertes.fr/hal-02018389.

[13] N. Larrousse and J. Marchand, A Techno-Human Mesh for Humanities in France: Dealing with preservation complexity, in: *DH 2019*, Utrecht, Netherlands, 2019. https://hal.archives-ouvertes.fr/hal-02153016.

[14] B. McBride, Jena: A Semantic Web toolkit, *IEEE Internet computing* **6**(6) (2002), 55–59.

[15] G. Carothers and E. Prud'hommeaux, RDF 1.1 Turtle, 2014. http://www.w3.org/TR/2014/REC-turtle-20140225/.

[16] M. Bostock, D3.js - Data-Driven Documents, 2012. http://d3js.org/.

[17] J. Cheng, Z. Ma and L. Yan, f-SPARQL: a flexible extension of SPARQL, in: *International Conference on Database and Expert Systems Applications*, Springer, 2010, pp. 487–494.

[18] M. Gueroussova, A. Polleres and S.A. McIlraith, SPARQL with Qualitative and Quantitative Preferences., in: *OrdRing@ ISWC*, 2013, pp. 2–8.

[19] G. Fokou, S. Jean, A. Hadjali and M. Baron, Handling failing RDF queries: from diagnosis to relaxation, *Knowledge and Information Systems* **50**(1) (2017), 167–195.

[20] H. Shimazu, H. Kitano and A. Shibata, Retrieving Cases from Relational Data-Bases: Another Stride Towards Corporate-Wide Case-Based Systems, in: *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93), Chambéry*, 1993, pp. 909–914.

[21] J. Lieber and A. Napoli, Using Classification in Case-Based Planning, in: *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96), Budapest, Hungary*, W. Wahlster, ed., John Wiley & Sons, Ltd., 1996, pp. 132–136.

[22] A. Cordier, V. Dufour-Lussier, J. Lieber, E. Nauer, F. Badra, J. Cojan, E. Gaillard, L. Infante-Blanco, P. Molli, A. Napoli and H. Skaf-Molli, Taaable: a Case-Based System for personalized Cooking, in: *Successful Case-based Reasoning Applications-2*, S. Montani and L.C. Jain, eds, Studies in Computational Intelligence, Vol. 494, Springer, 2014,

pp. 121–162. doi:10.1007/978-3-642-38736-4_7. https://hal.inria.fr/hal-00912767.

[23] O. Bruneau, E. Gaillard, N. Lasolle, J. Lieber, E. Nauer and J. Reynaud, A SPARQL Query Transformation Rule Language — Application to Retrieval and Adaptation in Case-Based Reasoning, in: *Case-Based Reasoning Research and Development. ICCBR 2017*, D. Aha and J. Lieber, eds, Lecture Notes in Computer Science, Springer, 2017, pp. 76–91.

[24] O. Corby, R. Dieng-Kuntz and C. Faron Zucker, Querying the Semantic Web with Corese Search Engine, in: *European Conference on Artificial Intelligence*, Valence, Spain, 2004. https://hal.inria.fr/hal-01531219.

[25] N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson and M.A. Musen, Creating Semantic Web Contents with Protégé-2000, *IEEE intelligent systems* **16**(2) (2001), 60–71.

[26] A. Bernstein and E. Kaufmann, GINO–a guided input natural language ontology editor, in: *International semantic web conference*, Springer, 2006, pp. 144–157.

[27] A. Funk, V. Tablan, K. Bontcheva, H. Cunningham, B. Davis and S. Handschuh, Clone: Controlled language for ontology editing, in: *The Semantic Web*, Springer, 2007, pp. 142–155.

[28] A. Hermann, S. Ferré and M. Ducassé, An interactive guidance process supporting consistent updates of RDFS graphs, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2012, pp. 185–199.

[29] C.K. Riesbeck and R.C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 1989.

[30] C. Basca, S. Corlosquet, R. Cyganiak, S. Fernández and T. Schandl, Neologism: Easy vocabulary publishing (2008).

[31] B. Lowe, B. Caruso, N. Cappadona, M. Worthington, S. Mitchell and J. Corson-Rikert, The Vitro Integrated Ontology Editor and Semantic Web Application., in: *ICBO*, 2011.

[32] A. Elizarov, A. Kirillovich, E. Lipachev and O. Nevzorova, Semantic formula search in digital mathematical libraries, in: *2017 Second Russia and Pacific Conference on Computer Technology and Applications (RPC)*, IEEE, 2017, pp. 39–43.