Assister l'édition manuelle de données RDF à l'aide du raisonnement à partir de cas

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

Assister l'édition manuelle de données RDF à l'aide du raisonnement à partir de cas

N. Lasolle[1,2], O. Bruneau[1], J. Lieber[2], E. Nauer[2] et S. Pavlova[2]

[1] Université de Lorraine, CNRS, Université de Strasbourg, AHP-PReST, F-54000 Nancy, France
[2] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

[1] prénom.nom@univ-lorraine.fr
[2] prénom.nom@loria.fr

Résumé

Les technologies du Web sémantique fournissent des outils pour structurer, exploiter et enrichir des corpus historiques tels que le corpus de la correspondance d'Henri Poincaré. Cependant, le processus d'édition de données RDF *est un processus qui est souvent manuel et peut sembler fastidieux pour les contributeurs. Cet article introduit un système de suggestions qui s'appuie sur le raisonnement à partir de cas pour assister l'édition de données* RDF*. Ce système se décline en quatre versions qui sont comparées au travers d'une double évaluation.*

Mots-clés

Web sémantique, raisonnement à partir de cas, édition de données RDF, humanités numériques, transformation de requêtes SPARQL, corpus historique.

Abstract

Semantic Web technologies provide a way to structure, to exploit and to enhance historical corpora data such as the Henri Poincaré correspondence corpus. However, RDF *data editing is often manual and may seem tedious for the concerned user. This article introduces a suggestion system which uses case-based reasoning to assist manual* RDF *data editing. This system is available in four versions which are compared to each other through a double evaluation.*

Keywords

Semantic Web, case-based reasoning, RDF data editing, Digital Humanities, SPARQL query transformation, historical corpus.

1 Introduction

Dans le contexte des bases RDF, le processus d'édition des données nécessite fréquemment une intervention humaine. Cette tâche peut rapidement s'avérer fastidieuse pour les personnes concernées avec un risque d'erreur important. Ce constat a notamment été porté lors de l'édition du corpus de la correspondance d'Henri Poincaré. Ce corpus historique rassemble des échanges d'ordre scientifique, administratif et privé qui forment une source d'information importante pour les historiens. Le cœur du corpus est constitué d'un ensemble d'environ 2100 lettres envoyées ou reçues par Henri Poincaré qui sont liées à des documents divers (article, rapport, compte-rendu, etc.), des personnalités (de France et d'ailleurs), des lieux et institutions, etc. Des technologies du Web sémantique (RDF, RDFS et SPARQL) ont été utilisées pour exploiter et enrichir ce corpus historique. Les lettres sont accessibles sur le site `http://henripoincare.fr` [1]. Chaque lettre comporte une numérisation du document original [2], une transcription, un apparat critique ainsi qu'un ensemble de méta-données. Ces dernières permettent à la fois la description physique de la lettre (expéditeur, destinataire, date de rédaction, etc.) et de son contenu (thèmes scientifiques abordés, personnes et institutions citées, etc.). Ce site a été créé grâce au système de gestion de contenus *Omeka S* [4] qui permet à des institutions (musées, archives, etc.) d'éditer, de publier et de rendre accessibles des corpus.

Lors de l'édition du corpus de la correspondance d'Henri Poincaré, plusieurs types d'erreurs ont été identifiés. *L'erreur de duplication* se produit lorsque qu'un utilisateur insère des données qui existent déjà dans la base. *L'erreur d'ambiguïté* se produit lorsque qu'un utilisateur ne dispose pas de suffisamment d'informations pour distinguer des éléments. Par exemple, si une recherche est effectuée sur la base de la chaîne "Henri Poincaré", différents types de ressources peuvent être retournés. En effet, la réponse attendue la plus plausible devrait se référer au célèbre scientifique, mais ce terme se réfère également à différents instituts et écoles et, depuis 1997, un prix de physique mathématique a été créé en sa mémoire. *L'erreur de frappe* se produit lorsqu'un utilisateur souhaite écrire un mot existant pour désigner une ressource spécifique, mais qu'il commet une erreur lors de la saisie de l'identifiant. Si elle n'est pas remarquée, une erreur de ce type peut conduire à la création d'une nouvelle ressource dans la base au lieu de faire référence à une ressource existante. Outre ces possibles erreurs, la charge cognitive associée à l'utilisation d'un système d'annotation ne doit pas être négligée. Selon le vo-

1. Sur ce site, d'autres éléments sont disponibles : les travaux de Poincaré, une iconographie et une bibliographie de ce mathématicien.

2. Certaines numérisations ne sont pas disponibles pour des raisons en lien avec les droits d'auteur.

lume du corpus à annoter, ce processus pourrait être un projet à long terme (plusieurs années). Maintenir la motivation des contributeurs lors de l'exécution des tâches associées est donc primordial.

Cet article présente un outil pour assister l'édition manuelle de données RDF. Quelques notions préliminaires sur le Web sémantique sont données (section 2). Ensuite, le système de suggestions, décliné en quatre versions est présenté (section 3). L'outil s'appuie notamment sur un mécanisme de raisonnement à partir de cas qui s'intéresse aux liens entre la ressource en cours d'édition et des ressources déjà éditées dans la base de connaissances. Une double évaluation a été réalisée pour mesurer la pertinence des suggestions proposées au regard de cas d'éditions réels (section 4). Cet article présente également des travaux liés autour de l'édition de données RDF et des systèmes de recommandation (section 5). Une conclusion et une présentation de perspectives de recherche sont données en section 6. Ce travail de recherche a été préalablement présenté lors de la conférence internationale du Web sémantique (ISWC 2020) [9].

2 Préliminaires sur le Web sémantique

Cette section présente les technologies du Web sémantique utilisées pour exploiter les données de ce corpus : le modèle RDF, le langage de représentation de connaissances RDFS et le langage d'interrogation de graphes SPARQL.

2.1 Le modèle RDF

Resource Description Framework (RDF [10]) est un modèle de représentation de données fondé sur l'utilisation de graphes orientés et étiquetés. C'est un standard développé et maintenu par le *World Wide Web Consortium* (W3C) et qui est fortement utilisé par la communauté du Web sémantique. Un graphe RDF est composé de trois types de nœuds : des *ressources nommées*, des *ressources anonymes* et des *littéraux*. Une *ressource nommée* est identifiée par un *Internationalized Resource Identifier* (IRI) et permet de décrire une classe (p. ex. `Personne`, `Mathématicien`, `Lettre`, etc.), une propriété (p. ex. `expéditeur`, `destinataire`, etc.) ou une instance (p. ex. `henriPoincaré`, `lettre11`, etc.) [3]. Une *ressource anonyme* représente une ressource qui n'est pas explicitement identifiée (*nœud vide*). Une telle ressource est désignée par l'utilisation d'un point d'interrogation préfixant un nom de variable (p. ex. `?x`, `?personne`, etc.). Un littéral correspond à une valeur constante d'un type donné (entier, chaîne de caractères, date, etc.).

Il est possible de définir des relations entre les nœuds du graphe par l'utilisation de propriétés décrivant les ressources les composant. Ces relations sont caractérisées par des triplets de la forme $\langle sujet\ \ prédicat\ \ objet \rangle$. Le sujet représente la ressource (nommée ou anonyme) à décrire. Le prédicat est une propriété qui décrit cette ressource. L'objet est la valeur associée à la propriété et peut être une ressource nommée, une ressource anonyme ou un littéral.

3. Dans un souci de lisibilité, les ressources nommées ne sont pas représentées en utilisant des IRI complets dans ce document.

2.2 RDF Schema

RDF Schema (RDFS [5]) est un langage de représentation de connaissances qui étend le modèle RDF. Plusieurs propriétés sont utilisées pour structurer les ressources : `rdfs:subclassof` (resp. `rdfs:subpropertyof`) permet de créer une hiérarchie entre des classes (resp. propriétés). Par exemple, le triplet ⟨`Lettre rdfs:subclassof Document`⟩ indique que le concept de `Lettre` est plus spécifique que le concept de `Document`. `rdfs:domain` (resp. `rdfs:range`) s'applique à une propriété et permet d'ajouter une contrainte à propos du type de la ressource se trouvant en position de $sujet$ (resp. $objet$) au sein d'un triplet. Un graphe RDF constitue une base de données à partir de laquelle de nouvelles données peuvent être engendrées. La déduction RDFS désigne l'application de règles d'inférence RDFS afin d'inférer de nouvelles données. Ce mécanisme est utilisé pour mener des raisonnements au sein des graphes RDF.

2.3 SPARQL

SPARQL est le langage recommandé par le W3C pour interroger des graphes RDF [12]. Une forme courante de requêtes SPARQL est constituée d'une clause SELECT contenant une ou plusieurs variables suivie par une clause WHERE composée d'un *patron de graphe* divisé en un ou plusieurs *patrons de triplet* et d'une possible clause FILTER qui permet d'ajouter des contraintes pour les valeurs littérales associées aux propriétés. Considérons la requête informelle suivante :

$\mathcal{Q} =$ « Donner les lettres rédigées par Henri Poincaré entre 1885 et 1890 et ayant pour thème la géométrie »

Cette requête peut s'exprimer de la façon suivante en utilisant le langage SPARQL :

$Q =$

```
SELECT ?l
WHERE
   {
     ?l type Lettre .
     ?l expéditeur henriPoincaré .
     ?l thème géométrie .
     ?l dateDeRédaction ?y .
     FILTER(YEAR(?y) >= 1885
        AND YEAR(?y) <= 1890)
   }
```

3 Un système de suggestions pour assister l'édition de données RDF

Comme décrit dans l'introduction de ce chapitre, le processus d'édition est un travail fastidieux qui justifie la nécessité de créer un éditeur dédié pour assister l'utilisateur. Ce système devrait permettre une mise à jour interactive efficace d'une base RDFS, en visualisant les faits déjà édités et en fournissant des suggestions adaptées au contexte d'édition.

Pour répondre à cette problématique, quatre versions d'un moteur de suggestions ont été mises en place :

Le système basique assiste l'utilisateur en proposant un mécanisme d'autocomplétion dans laquelle les suggestions sont classées par ordre alphabétique. Les suggestions proposées ne dépendent ni du problème d'édition courant ni des connaissances définies dans l'ontologie.

Le système déductif bénéficie de l'utilisation des connaissances de l'ontologie pour le classement des suggestions fournies à l'utilisateur.

Le système à base de cas s'appuie sur des problèmes d'édition similaires au problème courant.

Le système combiné combine les deux précédentes méthodes.

3.1 Le système déductif

La notion de *question d'annotation* est ici introduite : cela correspond à un triplet pour lequel 1, 2 ou les 3 éléments sont inconnus, et pour lequel un champ est en cours d'édition. Ce champ est représenté en utilisant un cadre autour d'une variable existentielle (c'est-à-dire $\boxed{\texttt{?p}}$, $\boxed{\texttt{?o}}$). Par exemple, $\langle s \ \boxed{\texttt{?p}} \ \texttt{?o} \rangle$ correspond à un type de question d'annotation pour lequel le sujet est connu, le prédicat est actuellement en cours d'édition et l'objet est inconnu. Il existe douze types de questions d'annotation différents (voir [7] pour plus de détails). Pour chacun d'entre eux, les connaissances liées aux domaines et co-domaines des propriétés peuvent être utilisées pour classer les valeurs candidates pour le champ cible.

Soit la question d'annotation $\langle \texttt{lettre11 expéditeur} \ \boxed{\texttt{?o}} \rangle$ qui est du type $\langle s \ p \ \boxed{\texttt{?o}} \rangle$. L'objectif est ici de fournir des suggestions appropriées en classant des valeurs potentielles pour l'objet. Pour cette version du système de suggestions, les premières suggestions sont les ressources des classes `Personne` et `Institution` car ces classes font partie du co-domaine de la propriété `expéditeur`. Cette connaissance est donc utilisée pour favoriser les instances de ces classes. Cependant, les ressources qui n'appartiennent pas explicitement à ces classes sont toujours proposées parce que RDFS fonctionne selon l'hypothèse du monde ouvert [4].

Il existe différentes règles qui peuvent être utilisées pour classer les valeurs potentielles et dont les applications dépendent du type de la question d'annotation. Pour répondre à une question d'annotation, un score est calculé pour chaque valeur candidate `?v` en fonction du nombre de règles qui ont récupéré cette valeur. La liste finale des suggestions est classée en fonction de ce score par ordre décroissant. Pour les valeurs potentielles ayant le même nombre, l'ordre alphabétique est utilisé.

4. Si un fait n'est pas affirmé, cela ne signifie pas qu'il soit faux. Dans cette situation, il peut exister une ressource r qui est destinée à représenter une personne (resp. une institution) mais est telle que le triplet $\langle r \ \texttt{a Personne} \rangle$ (resp. $\langle r \ \texttt{a Institution} \rangle$) ne peut être inféré par l'état actuel de la base RDFS. Par conséquent, r peut également être suggérée, bien que plus loin dans la liste de suggestions.

3.2 Le système à base de cas

L'utilisation de la déduction RDFS apporte une première amélioration au système de suggestions en utilisant les connaissances liées aux domaines et co-domaines des propriétés définies dans la base. Toutefois, dans certaines situations, cela ne suffit pas pour proposer les ressources les plus appropriées relatives à la question d'annotation courante.

À titre d'exemple, considérons un triplet en cours d'édition pour lequel le sujet est une instance de `Lettre` (`lettre2100`), le prédicat est `destinataire` et pour lequel des suggestions concernant le champ d'objet sont attendues. Étant donné que la classe `Personne` fait partie du co-domaine de la propriété `destinataire`, le système favorisera les instances de cette classe dans la liste de suggestions. Mais le problème est qu'il y a de nombreuses instances de cette classe dans la base [5], et il n'y a aucune garantie que la valeur appropriée sera parmi les premières suggestions de la liste. En effet, pour les valeurs ayant le même score, c'est l'ordre alphabétique qui est utilisé. Une autre façon d'obtenir un classement pertinent de la liste de suggestions est d'utiliser le raisonnement à partir de cas : dans la situation actuelle, des éléments d'information provenant de situations similaires peuvent être réutilisés.

3.2.1 Préliminaires sur le raisonnement à partir de cas

Le raisonnement à partir de cas (RàPC [13]) vise à résoudre des problèmes à l'aide d'une *base de cas* `BC`, c'est-à-dire un ensemble fini de cas, où un cas représente un problème passé avec une solution associée. Un cas est souvent défini comme un couple ordonné $(\texttt{x}, \texttt{y})$ où `x` est un problème et `y` est une solution de `x`. Un cas $(\texttt{x}^s, \texttt{y}^s)$ de la base de cas est appelé *cas source*, avec $\texttt{x}^s$ représentant un problème source et $\texttt{y}^s$ la solution de $\texttt{x}^s$. L'entrée d'un système de raisonnement à partir de cas est un problème appelé le *problème cible* et désigné par $\texttt{x}^{\texttt{cible}}$.

Le processus de RàPC peut être décomposé en quatre étapes [1]. (1) Un cas $(\texttt{x}^s, \texttt{y}^s) \in \texttt{BC}$ jugé similaire à $\texttt{x}^{\texttt{cible}}$ est sélectionné (*remémoration*). (2) Ce cas $(\texttt{x}^s, \texttt{y}^s)$ est utilisé pour résoudre le problème $\texttt{x}^{\texttt{cible}}$ (*réutilisation*). La solution proposée $\texttt{y}^{\texttt{cible}}$ peut être égale à $\texttt{y}^s$ (réutilisée telle que) ou adaptée pour tenir compte de différence entre $\texttt{x}^s$ et $\texttt{x}^{\texttt{cible}}$. (3) Le couple $(\texttt{x}^{\texttt{cible}}, \texttt{y}^{\texttt{cible}})$ est évalué pour vérifier que $\texttt{y}^{\texttt{cible}}$ résout correctement $\texttt{x}^{\texttt{cible}}$ et, dans le cas contraire, $\texttt{y}^{\texttt{cible}}$ peut être modifiée en conséquence (*révision*). (4) Enfin, le cas nouvellement formé $(\texttt{x}^{\texttt{cible}}, \texttt{y}^{\texttt{cible}})$ est ajouté à `BC` si cet ajout est jugé approprié (*mémorisation*).

3.2.2 Explication du mécanisme

Dans le contexte de ce système de suggestions, un problème d'annotation $\texttt{x}^{\texttt{cible}}$ est composé d'une question d'annotation et d'un contexte. Pour les questions d'annotation du type $\langle s \ p \ \boxed{\texttt{?o}} \rangle$, il est défini comme suit :

$$\texttt{x}^{\texttt{cible}} = \boxed{\begin{array}{l}\textbf{question :} \ \langle subj^{\texttt{cible}} \ pred^{\texttt{cible}} \ \boxed{\texttt{?o}} \rangle \\ \textbf{contexte :} \ \text{l'ensemble des triplets liés à } subj^{\texttt{cible}}\end{array}}$$

5. Au moment de la rédaction de cet article, il y a environ 1800 personnes définies dans la base de données.

Pour l'exemple lié à la lettre 2100, cela donne $\mathtt{x}^{\mathtt{cible}} =$

question :	⟨lettre2100 destinataire ?o⟩
contexte :	⟨lettre2100 expéditeur henriPoincaré⟩ ⟨lettre2100 thème écolePolytechnique⟩ ⟨lettre2100 cite paulAppell⟩

La base de cas correspond à la base RDF $\mathcal{D}_{\text{HP}}$. Un cas source est donné par un triplet $\langle subj^{\mathtt{s}}\ pred^{\mathtt{s}}\ obj^{\mathtt{s}}\rangle$ de $\mathcal{D}_{\text{HP}}$, considéré parmi tous les triplets de $\mathcal{D}_{\text{HP}}$, et qui, en lien avec $\mathtt{x}^{\mathtt{cible}}$, peut être décomposé en un problème $\mathtt{x}^s$ et une solution $\mathtt{y}^s$:

$$\mathtt{x}^s = \begin{array}{|ll|}\hline \textbf{question :} & \langle subj^{\mathtt{s}}\ pred^{\mathtt{s}}\ \boxed{?o}\rangle \\ \textbf{contexte :} & \text{la base } \mathcal{D}_{\text{HP}} \\ \hline\end{array}$$

et la solution $\mathtt{y}^s = obj^{\mathtt{s}}$. Pour les besoins de cet exemple, considérons un extrait $\mathcal{D}_{ex}$ de la base du corpus de la correspondance Henri Poincaré $\mathcal{D}_{\text{HP}}$ composé des lettres relatives aux instances suivantes de la classe `Personne` : `göstaMittagLeffler`, `alineBoutroux`, `eugénieLaunois`, `felixKlein` et `henriPoincaré`. Comment ordonner la liste composée de ces 5 ressources ? Pour proposer une solution à ce problème, la méthode consiste à récupérer et utiliser les cas qui correspondent le mieux au problème d'annotation actuel. À chaque cas source $\mathtt{x}^s$ est associée une valeur $\mathtt{y}^s$ qui est utilisée comme solution candidate de $\mathtt{x}^{\mathtt{cible}}$. Un score est calculé pour le classement des solutions candidates. Ce score correspond au nombre de lettres similaires ayant cette valeur associée à la propriété `destinataire`. Une requête initiale SPARQL $\mathcal{Q}$ générée à partir de $\mathtt{x}^{\mathtt{cible}}$ est définie pour calculer ce score. Pour l'exemple courant, cela donne :

$\mathcal{Q} =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `destinataire`
où les lettres ont `écolePolytechnique`
comme thème, citent `paulAppell` et
ont été rédigées par `henriPoincaré`"

L'exécution de $\mathcal{Q}$ sur $\mathcal{D}_{ex}$ ne retourne aucun résultat. En effet, il est rare de trouver deux lettres différentes ayant exactement le même contexte. La question est donc de trouver une méthode pour retrouver les cas les plus similaires. Cette question peut être traitée en utilisant un mécanisme de transformation de requêtes SPARQL. Un système a déjà été conçu pour gérer des règles de transformation et s'est révélé utile dans différents contextes, notamment pour la recherche dans le corpus de la correspondance d'Henri Poincaré et dans le système de cuisine *Taaable* [6].

Les règles sont configurées par l'utilisateur et peuvent être générales ou dépendantes d'une application. À chaque règle est associé un coût, correspondant à un coût de transformation de la requête. Deux règles sont considérées dans l'exemple courant :

- $r_{\texttt{échange}}$: échange de l'expéditeur et du destinataire de la lettre (coût de 2) ;
- $r_{\texttt{genObj}}$: généralise une instance de classe en position d'objet en remplaçant cette instance par un nœud anonyme du type de cette classe (coût de 3).

Un arbre de recherche peut être parcouru par coût croissant à partir de la requête initiale en appliquant successivement une ou plusieurs règles de transformation. Un coût maximum est défini pour limiter la profondeur d'exploration de l'arbre de recherche. Pour cette application, ce coût maximum est fixé à 10. À la profondeur 1, l'application de la règle $r_{\texttt{échange}}$ sur $\mathcal{Q}$ génère la requête $\mathcal{Q}_1$ avec un coût de 2 (la partie modifiée de la requête est soulignée) :

$\mathcal{Q}_1 =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `expéditeur`
où les lettres ont `écolePolytechnique`
comme thème, citent `paulAppell` et
ont été reçues par `henriPoincaré`"

Le résultat de l'exécution de $\mathcal{Q}_1$ sur $\mathcal{D}_{ex}$ est : [{`eugénieLaunois` : 2}, {`alineBoutroux` : 1}]. Trois applications de la règle $r_{\texttt{genObj}}$ existent à la profondeur 1, chacune d'entre elles pour un coût de 3. La première s'applique pour la personne citée, en remplaçant `paulAppell` par toute instance de la classe `Mathématicien` (parce que Paul Appell appartient à cette classe), la deuxième s'applique à l'expéditeur de la lettre, et la dernière s'applique au thème `écolePolytechnique`. Les requêtes générées sont $\mathcal{Q}_2$, $\mathcal{Q}_3$ et $\mathcal{Q}_4$:

$\mathcal{Q}_2 =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `destinataire`
où les lettres ont `écolePolytechnique`
comme thème, citent un `mathématicien` et
ont été rédigées par `henriPoincaré`"

$\mathcal{Q}_3 =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `destinataire`
où les lettres ont `écolePolytechnique`
comme thème, citent `paulAppell` et
ont été rédigées par un `mathématicien`"

$\mathcal{Q}_4 =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `destinataire`
où les lettres ont un thème lié
à `l'éducation`, citent `paulAppell` et
ont été rédigées par `henriPoincaré`"

L'exécution de $\mathcal{Q}_2$ sur $\mathcal{D}_{ex}$ donne : [{`eugénieLaunois` : 138}, {`alineBoutroux` : 4}]. Les exécutions de $\mathcal{Q}_3$ et $\mathcal{Q}_4$ ne retournent aucun résultat. Le coût maximum ayant été fixé à 10, il est possible de continuer l'exploration de l'arbre sur les différentes branches afin de réordonner la liste de suggestions.

À la profondeur 2, l'application de $r_{\texttt{genObj}}$ sur $\mathcal{Q}_2$ (généralisation du thème) génère la requête :

$\mathcal{Q}_{21} =$ "Donner, pour chaque valeur candidate, `?o`,
le nombre de lettres ayant cette valeur associée
à la propriété `destinataire`
où les lettres ont un thème lié
à `l'éducation`, citent un `mathématicien` et
ont été rédigées par `henriPoincaré`"

L'exécution de $\mathcal{Q}_{21}$ sur $\mathcal{D}_{ex}$ retourne : [{`eugénieLaunois` : 280},

{göstaMittagLeffler : 74}, {alineBoutroux : 17}]. À la profondeur 3, l'application de $r_{\texttt{genObj}}$ sur $\mathcal{Q}_{21}$ (pour l'expéditeur) génère la requête :

$\mathcal{Q}_{211} =$ "Donner, pour chaque valeur candidate, ?o, le nombre de lettres ayant cette valeur associée à la propriété destinataire où les lettres ont un thème lié à l'éducation, citent un mathématicien et ont été rédigées par un mathématicien"

L'exécution de $\mathcal{Q}_{211}$ sur $\mathcal{D}_{ex}$ donne : [{eugénieLaunois : 305}, {henriPoincaré : 219}, {göstaMittagLeffler : 141}, {felixKlein : 25}, {alineBoutroux : 21}]. Les autres applications possibles de la règle (en tenant compte du coût maximum) génèrent des requêtes déjà proposées par d'autres combinaisons ou qui donnent les mêmes ressources mais avec un coût plus élevé. La liste finale des suggestions est classée par le coût de transformation minimal requis. Pour les ressources ayant le même coût minimal, le score lié à l'exécution de la requête associé à ce coût est utilisé (par ordre décroissant). Pour l'exemple courant, cela donne, pour les 5 premières suggestions, du numéro 1 au numéro 5 : eugénieLaunois, alineBoutroux, göstaMittagLeffler, henriPoincaré[6] et felixKlein. Le reste des suggestions est composé de toutes les ressources de la base classées par ordre alphabétique. Cette approche constitue l'étape de remémoration du modèle de RàPC. L'étape de réutilisation est une approche de réutilisation en tant que telle : il n'y a pas d'adaptation des ressources proposées. Après cela, l'utilisateur choisit la ressource appropriée, qui peut être considéré comme une étape de *révision*. Ensuite, le triplet édité est inséré dans la base de connaissances (étape de *mémorisation*).

3.3 Le système combiné

La dernière version du système combine l'utilisation de la déduction RDFS avec le RàPC. Elle tire parti à la fois des connaissances sur les ressources similaires à celle en cours d'édition et des domaines et co-domaines des propriétés utilisées lors de l'édition. Les ressources trouvées en utilisant le RàPC sont en tête de la liste de suggestions. Pour les autres ressources, le calcul du score tel que présenté dans la section 3.1 est appliqué. Considérons l'exemple présenté ci-dessus, dans lequel la question d'annotation était ⟨lettre2100 destinataire ?o⟩. En utilisant le système de raisonnement à partir de cas, les cinq premières suggestions sont des ressources qui semblent pertinentes compte tenu du contexte actuel d'édition et en recherchant des objets similaires dans la base de données. Mais pour le reste des suggestions, seul l'ordre alphabétique est utilisé pour le classement. Pour y remédier, il est possible d'utiliser le co-domaine de la propriété destinataire (comme expliqué dans la section 3.1) pour classer la deuxième partie de la liste de suggestions. Sachant que la classe Personne fait partie du co-domaine de la propriété expéditeur, toutes les instances de cette classe seront plus hautes dans la liste de suggestions que celles des autres classes (p. ex. Article, Revue, Adresse).

6. Cette suggestion pourrait être supprimée si le système sait que le destinataire d'une lettre ne peut être son expéditeur.

4 Évaluation

L'objectif de cette évaluation est de comparer l'efficacité des différentes versions du système pour des situations d'annotation concrètes. La première évaluation est humaine, par l'intermédiaire d'un utilisateur qui teste et compare les quatre versions du système au travers d'un outil Web. Une deuxième évaluation est gérée par un programme dédié et fournit des mesures objectives. Différentes classes existent dans la base de connaissances du corpus de la correspondance d'Henri Poincaré (Lettre, Personne, Article, etc.) mais pour cette évaluation, l'accent est mis sur l'édition des lettres. Les deux évaluations se concentrent sur un sous-ensemble de 7 propriétés parmi les plus fréquemment utilisées lors de l'édition de lettres : expéditeur définit l'expéditeur ; destinataire définit le destinataire ; thème donne l'un des thèmes ; archivéeÀ précise le lieu d'archivage ; aPourRéponse donne une lettre de réponse à la lettre actuelle ; répondÀ donne une lettre à laquelle répond la lettre actuelle ; cite fait référence à une personne mentionnée dans la transcription de la lettre.

4.1 Évaluation humaine

4.1.1 Un outil Web pour éditer les données RDF

Une interface Web a été développée pour utiliser et comparer les différentes versions du système de suggestions. Cet outil propose à un utilisateur un mécanisme d'autocomplétion qui utilise le système de suggestions pour fournir des valeurs. L'interface est commune à toutes les versions du système. L'outil permet la visualisation et la mise à jour des bases de données RDF. Trois champs sont disponibles pour définir les valeurs du sujet, du prédicat et de l'objet. L'utilisation de préfixes a été mise en place pour améliorer la lisibilité de l'outil. La liste des espaces de noms existants et des préfixes associés est accessible dans un tableau récapitulatif. Lors de l'édition d'un triplet, l'éditeur affiche le *contexte* associé qui est rafraîchi à chaque fois que la valeur du champ sujet est mise à jour. Lorsqu'un nouveau triplet est créé et inséré dans la base de données, il est ajouté au contexte actuel. L'interface complète associée à plusieurs cas d'utilisation fait l'objet d'une vidéo de démonstration accessible en ligne[7].

4.1.2 Méthodologie

Cette évaluation implique un utilisateur unique qui est l'une des personnes chargées de l'édition du corpus de la correspondance d'Henri Poincaré. Il n'avait aucune expérience préalable avec cet outil au moment où il a conduit l'évaluation. L'ensemble de test est composé de 10 lettres qui ont été choisies au hasard parmi un ensemble de 30 lettres inédites provenant de la correspondance d'Henri Poincaré.

7. https://videos.ahp-numerique.fr/videos/watch/0d544e5b-b4be-423e-9497-216f29ab44f3

Cet ensemble constitue un véritable cas d'annotation par rapport aux lettres déjà éditées dans la base du corpus. Les éléments du corpus d'évaluation ont été édités en utilisant Omeka S avant le début de l'évaluation, de manière à veiller à ce qu'aucune version du système ne souffre d'être la première à être évaluée. Pour chaque version du système, l'utilisateur édite en une fois les 10 mêmes lettres en utilisant l'interface fournie. Les versions sont présentées dans un ordre aléatoire et inconnu. Avant de passer à la version suivante, la base RDF est réinitialisée pour correspondre à l'état initial.

Après avoir édité l'ensemble des lettres pour une version du système, l'utilisateur est invité à compléter un questionnaire pour fournir un retour d'expérience pour cette version. Cette enquête insiste sur l'appréciation de l'efficacité du mécanisme d'autocomplétion – mais un retour d'expérience sur l'interface utilisateur est également attendu. Pour chaque propriété, l'utilisateur est invité à attribuer une note en utilisant une *échelle de Likert* [2], de 1 (pas du tout pertinent) à 7 (très pertinent) pour caractériser la pertinence des suggestions fournies pour les questions d'annotation liées à cette propriété.

4.1.3 Résultats et analyse

Il ressort de cette évaluation que le système combinant déduction RDFS et raisonnement à partir de cas est perçu comme le plus efficace, et ce pour toutes les propriétés de l'évaluation. Les scores moyens de toutes les évaluations pour les différentes propriétés sont indiqués dans le tableau 1. Le système classique est la version qui a obtenu le score le plus bas. Il a été perçu comme « n'aidant pas à l'annotation », mais ne causant pas de problèmes à l'utilisateur. Le système déductif et le système à base de cas ont obtenu des notes moyennes élevées. Toutefois, dans les situations où la recherche de cas sources conduit à un ensemble de cas vide, le moteur de RàPC utilise uniquement l'ordre alphabétique pour le classement de la liste des suggestions, et peut fournir des ressources non pertinentes. Cela a causé de la frustration à l'utilisateur et explique pourquoi le score moyen du système à base de cas est inférieur à celui du système déductif. La combinaison des deux systèmes est une bonne méthode pour éviter ces situations. En outre, l'interface associée à l'outil a permis d'éviter les erreurs décrites dans l'introduction : elle empêche l'insertion de triplets qui existent déjà dans la base du corpus (*erreur de duplication*), le type de la ressource sélectionnée est toujours visible (*erreur d'ambiguïté*), et l'utilisation d'étiquettes simplifie la gestion des ressources pour l'utilisateur (*erreur de frappe*).

4.2 Évaluation automatique

4.2.1 Méthodologie

L'objectif de l'évaluation automatique est de comparer les performances des différentes versions de l'outil par des mesures. Les mesures choisies sont liées au rang de la valeur attendue $\texttt{rang}(qa)$ où qa est la question d'annotation actuelle. $\texttt{rang}(qa) = 1$ signifie que la valeur associée est la première dans la liste de suggestions. En d'autres termes, plus le rang moyen est petit, meilleure est la version.

Le graphe RDF de la correspondance d'Henri Poincaré $\mathcal{G}_{\text{HP}}$ a été utilisé comme un ensemble de tests. Ce graphe est formé par l'union de la base de faits $\mathcal{D}_{\text{HP}}$ et de l'ontologie $\mathcal{O}_{\text{HP}}$: $\mathcal{G}_{\text{HP}} = \mathcal{D}_{\text{HP}} \cup \mathcal{O}_{\text{HP}}$. Au moment de la rédaction de cet article, le graphe est composé d'environ 220 000 triplets. Pour cette évaluation, l'application des règles d'inférence RDFS mentionnées dans la section 2 a été considérée. Un ensemble de 100 lettres est aléatoirement extrait de l'ensemble existant des lettres éditées. Pour chaque lettre de cet ensemble, les triplets formant son contexte sont utilisées pour simuler des questions d'annotation dont la réponse est déjà connue. Pour chaque triplet, l'édition des 3 champs (sujet, prédicat et objet) est considéré dans un ordre aléatoire afin d'inclure différents types de questions d'annotation dans l'évaluation. Pour chaque question d'annotation qa, les quatre systèmes de suggestion sont appelés pour fournir une liste de suggestions ordonnée. Le rang de la valeur attendue $\texttt{rang}(qa)$ dans la liste est enregistré pour chaque version et est ajouté au multi-ensemble $\texttt{Rangs}(\textit{système})$ correspondant. À l'issue de l'évaluation, des mesures relatives aux éléments de $\texttt{Rangs}(\textit{système})$ sont calculées. Ces mesures correspondent au pourcentage de questions annotées dont la valeur escomptée a été donnée parmi les n premières propositions.

4.2.2 Résultats et analyse

Les résultats de cette évaluation sont présentés dans le tableau 2 pour chaque version du système. Différentes valeurs de n ont été choisies (5, 10 et 15) mais l'évolution de l'efficacité des versions reste la même dans toutes les situations. Cela montre que le système combiné fournit les meilleurs résultats pour les différentes questions d'annotation liées à cette évaluation parce qu'il suggère plus souvent la valeur appropriée. Il est donc plus susceptible d'aider l'utilisateur pendant le processus d'édition. Bien qu'il n'ait pas été utilisé comme mesure lors de l'évaluation, le temps de calcul a été pris en considération. Il correspond au temps nécessaire pour fournir la liste de suggestions pour une question d'annotation. En effet, le temps de réaction aux demandes doit être pris en compte dans un système d'interaction humaine d'autant plus que ce système utilise un mécanisme d'autocomplétion pour lequel un utilisateur ne s'attend pas à une latence. Le temps de calcul est plus important lorsqu'on utilise le système combiné mais cela reste suffisamment bas pour ne pas avoir d'impact sur l'utilisateur.

5 Discussion et travaux proches

5.1 Les éditeurs de données RDF

La méthode de RàPC présentée dans la section 3 est inspirée du système UTILIS [8]. Ce système introduit l'idée de rechercher des ressources similaires à celle qui est éditée pour suggérer des valeurs qui pourraient être appropriées au problème d'édition courant. Mais la forme de relaxation de requêtes proposée est différente car elle s'appuie principalement sur des règles de généralisation indépendantes du domaine d'application. Le moteur présenté dans cet article permet aux utilisateurs de définir leurs propres règles pour

TABLEAU 1 – Score moyen (sur une échelle de 1 à 7) associé a la pertinence des suggestions pour les différentes versions du système.

	Système classique	Système déductif	Système à base de cas	Système combiné
Score moyen	3,4	5,7	5,3	7

TABLEAU 2 – Mesures du rang lié aux suggestions pour les quatre versions du système.

	Système classique	Système déductif	Système à base de cas	Système combiné
$rang \leqslant 15$	11,3%	22,11%	49,0%	49,5%
$rang \leqslant 10$	7,1%	21,15%	43,2%	43,2%
$rang \leqslant 5$	2,7%	19,23%	33,6%	34,7%

exploiter des connaissances spécifiques à un domaine d'application. Combiné à l'utilisation des connaissances d'une ontologie RDFS, ce système propose des suggestions adaptées aux différents types de questions d'annotation.
Protégé [11] est l'un des outils les plus fréquemment utilisés pour éditer les données du Web sémantique. Lors de l'édition d'une instance d'une classe spécifique, Protégé s'appuie sur les domaines et co-domaines des propriétés pour faire des suggestions de valeurs de prédicats et d'objets. Mais ces suggestions ne s'appuient pas sur les triplets déjà édités présents dans la base. D'autres approches existent pour faciliter l'édition des bases RDF, plusieurs d'entre elles sont fondées sur le traitement automatique des langues. L'outil d'édition GINO [3] propose l'utilisation d'un langage naturel guidé et contrôlé qui permet à l'utilisateur de préciser des phrases correspondant à des faits RDF. L'idée principale est que les principes du Web sémantique ne sont parfois pas facilement appréhendés par les non spécialistes, et devraient donc être intégrés dans un système plus convivial. La syntaxe de ce langage est proche de la syntaxe de l'anglais (par exemple « There is a mount named Everest », « The height of mount Everest is 29 029 feet », etc.). Un mécanisme de suggestion propose des classes, des instances et des propriétés pour compléter l'annotation en cours. Le principal défi de ce système concerne l'interprétation de la demande de l'utilisateur afin de construire des triplets à partir de phrases.

5.2 Les systèmes de recommandation

Plus généralement, l'outil présenté dans cet article pourrait être défini comme un système de recommandation. Ces systèmes visent à aider l'utilisateur en présentant des informations susceptibles de l'intéresser. Différents systèmes de recommandation, tels que celui présenté ici, s'appuient sur le RàPC [14]. Il existe une grande variété de méthodes, et l'outil présenté dans cet article pourrait bénéficier de plusieurs d'entre elles. À titre d'exemple, l'implication de l'utilisateur dans le mécanisme de proposition de suggestions est envisagée. L'explicabilité de l'outil pourrait être renforcée car il peut être important de comprendre pourquoi certaines ressources ont été favorisées pour un contexte d'édition particulier. Ce système pourrait également bénéficier de l'utilisation d'un système de retour d'utilisation fondé sur des préférences, ce qui pourrait améliorer les résultats de l'outil dans plusieurs situations, donner à l'utilisateur le sentiment d'être inclus et renforcer ainsi la perception positive de l'outil. D'autre part, le mécanisme de transformation de requêtes utilisé ici pourrait être réutilisé dans d'autres systèmes de recommandation.

6 Conclusion et perspectives

Un processus d'annotation manuelle a été choisi pour éditer les données relatives aux ressources du corpus de la correspondance d'Henri Poincaré. Ce processus a été identifié comme étant fastidieux pour les utilisateurs chargés de l'édition. Pour remédier à ce problème, un outil proposant un système de suggestions a été mis en place. Il s'agit d'un outil général pour l'édition des données RDF. Il utilise des déductions s'appuyant sur une ontologie RDFS combinée à du RàPC. Différentes versions du système ont été créées. La première version classe les valeurs potentielles en utilisant l'ordre alphabétique. La deuxième version tire parti des connaissances sur les domaines et co-domaines des propriétés de l'ontologie. La troisième version utilise du RàPC pour exploiter les connaissances à propos des ressources éditées similaires à celle en cours d'édition. La dernière version est une combinaison des deux précédentes. Deux évaluations différentes ont été réalisées. L'évaluation humaine a permis de comparer les différentes versions du système entre elles et avec le système d'annotation actuel (Omeka S). L'évaluation automatique a apporté des mesures en comparant, pour un ensemble sélectionné de questions d'annotation, les suggestions des différentes versions du système. La pertinence des suggestions et le temps de calcul ont été pris en compte. Comme expliqué dans la section 4, alors que les mesures calculées par l'évaluation automatique montrent que l'utilisation du RàPC seul a donné de meilleurs résultats que l'utilisation de la déduction RDFS seule, le système à base de cas est parfois insuffisant et peut fournir des ressources non pertinentes dans certaines situations. Pour les deux évaluations, les résultats montrent que la dernière version du système qui est la combinaison de l'utilisation de la déduction RDFS avec celle du RàPC est la plus efficace.Toutefois, dans certaines situations, ce système présente certaines limites. Par exemple, considérons la question d'annotation présentée dans la section 3. Les troisième et quatrième versions du système proposent `henriPoincaré` comme réponse plausible bien qu'il soit déjà défini comme l'expéditeur de la lettre en

cours d'édition. Une façon de traiter cette question serait d'utiliser certaines connaissances du domaine en tant que contraintes d'intégrité. Un autre point observé lors des évaluations humaine et automatique est que l'ordre d'édition des différentes propriétés affecte grandement l'efficacité du moteur de suggestion. En effet, certaines valeurs de propriétés donnent plus d'informations sur la ressource que d'autres, et donc le fait de remplir ces valeurs en premier devrait améliorer le classement des suggestions. Le principal défi consiste alors à trouver le meilleur ordre d'édition pour les propriétés de la base. Ceci constitue une perspective de recherche. Un autre axe de recherche est lié à l'utilisation d'une logique plus expressive que RDFS telle que OWL-DL. Une logique contenant une forme de négation permettrait de supprimer certaines valeurs de la liste des valeurs potentielles. Toutefois, une telle extension pourrait affecter le temps de calcul et sa mise en œuvre devrait donc être étudiée.

Remerciements

Ce travail a bénéficié d'une aide de l'État, gérée par l'Agence Nationale de la Recherche, au titre du projet Investissements d'Avenir Lorraine Université d'Excellence, portant la référence ANR-15-IDEX-04-LUE.

Références

[1] A. Aamodt et E. Plaza. Case-based Reasoning : Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1) :39-59, 1994.

[2] I. E. Allen et C. A. Seaman. Likert scales and data analyses. *Quality progress*, 40(7) :64-65, 2007.

[3] A. Bernstein et E. Kaufmann. GINO–a guided input natural language ontology editor. *International semantic web conference*, pages 144-157. Springer, 2006.

[4] C. Boulaire et R. Carabelli. Du digital naive au bricoleur numérique : les images et le logiciel Omeka. É. Cavalié, F. Clavert, O. Legendre et D. Martin, éditeurs, *Expérimenter les humanités numériques. Des outils individuels aux projets collectifs*, chapitre 7, pages 81-103. Les Presses de l'Université de Montréal, Montréal, Québec, 2017.

[5] D. Brickley, R. V. Guha et B. McBride. RDF Schema 1.1, 2014. URL : `https://www.w3.org/TR/rdf-schema/`. Dernière consultation : février 2021.

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

[7] O. Bruneau, N. Lasolle, J. Lieber, E. Nauer, S. Pavlova et L. Rollet. Applying and Developing Semantic Web Technologies for Exploiting a Corpus in History of Science : the Case Study of the Henri Poincaré Correspondence. *Semantic Web*, 12(2), 2021.

[8] A. Hermann, S. Ferré et M. Ducassé. An Interactive Guidance Process Supporting Consistent Updates of RDFS Graphs. A. ten Teije, J. Völker, S. Handschuh, H. Stuckenschmidt, M. d'Acquin, A. Nikolov, N. Aussenac-Gilles et N. Hernandez, éditeurs, *Knowledge Engineering and Knowledge Management*, pages 185-199, Berlin, Heidelberg. Springer Berlin Heidelberg, 2012.

[9] N. Lasolle, O. Bruneau, J. Lieber, E. Nauer et S. Pavlova. Assisting the RDF Annotation of a Digital Humanities Corpus Using Case-Based Reasoning. J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne et L. Kagal, éditeurs, *The Semantic Web - ISWC 2020*, pages 617-633, Cham. Springer International Publishing, 2020.

[10] F. Manola, E. Miller, B. McBride et al. RDF Primer, 2004. URL : `https://www.w3.org/TR/rdf-primer`. Dernière consultation : février 2021.

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

[12] E. Prud'hommeaux. SPARQL Query Language for RDF. E. Prud'hommeaux et A. Seaborne, éditeurs, 2008. URL : `http://www.w3.org/TR/rdf-sparql-query/`. Dernière consultation : février 2021.

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

[14] B. Smyth. Case-based recommendation. P. Brusilovsky, A. Kobsa et W. Nejdl, éditeurs, *The adaptive web*, pages 342-376. Springer, Berlin, 2007.