LKSA : un algorithme de sélection de clés de liage dans des données RDF guidée par des paires de classes

Nacira Abbas, Jérôme David, Amedeo Napoli

LKSA : un algorithme de sélection de clés de liage dans des données RDF guidée par des paires de classes

Nacira Abbas*, Jérôme David**
Amedeo Napoli*

* Université de Lorraine, CNRS, Inria, Loria
Nancy, F-54000, France
nacira.abbas@inria.fr,
amedeo.napoli@loria.fr
** Université de Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
Grenoble, F-38000, France
jerome.david@inria.fr

Résumé. Dans cet article, nous nous intéressons à la sélection de clés de liage dans des jeux de données RDF. Une clé de liage permet d'établir des liens d'identité entre deux entités dans le web des données. Le couple d'entités liées détermine à son tour un couple de classes qui peut guider la recherche et la sélection de clés de liage de meilleure qualité. Pour ce faire, nous proposons un algorithme de sélection de clés de liage s'appuyant sur des mesures de qualité adaptées. Une série d'expérimentations sur des jeux de données RDF diversifiés montre le potentiel et l'efficacité de l'approche.

1 Introduction

Nous nous intéressons à la découverte de liens d'identité entre les ressources de deux jeux de données RDF (Resource Description Framework). Cette tâche est cruciale pour le web des données car les liens d'identité autorisent une meilleure interopérabilité entre différentes applications ainsi que l'amélioration de la qualité les données (recherche de "doublons"). Plusieurs approches existent pour trouver les liens d'identité parmi lesquelles l'approche "logique" qui s'appuie sur des axiomes exprimant des conditions suffisantes pour que deux ressources soient identiques (Saïs et al., 2007; Al-Bakri et al., 2015, 2016). Suivant cette ligne, cet article se concentre sur les *clés de liage* qui généralisent la notion de clé sur deux jeux de données (Atencia et al., 2014a). Un exemple de clé de liage est :

$$k = (\{\langle \texttt{designation}, \texttt{titre} \rangle\}, \{\langle \texttt{designation}, \texttt{titre} \rangle, \langle \texttt{author}, \texttt{auteur} \rangle\}, \langle \texttt{Book}, \texttt{Livre} \rangle)$$

qui signifie que si une instance a_1 de la classe `Book` et une instance b_1 de la classe `Livre`, possèdent les mêmes valeurs pour la propriété `designation` et pour la propriété `titre`, et que a_1 et b_1 partagent au moins une valeur pour les propriétés `author` et `auteur`, alors a_1 et b_1 désignent la même entité. En particulier la clé de liage k instancie la paire de classes $\langle \texttt{Book}, \texttt{Livre} \rangle$ et génère un lien d'identité ("same-as") entre a_1 et b_1.

Les clés de liage ne sont généralement pas fournies avec les jeux de données. De fait, un algorithme de découverte de clés de liage dites “candidates” est proposé dans (Atencia et al., 2014a). Le problème de la découverte de clés de liage candidates peut se formuler de façon formelle dans le cadre de l’Analyse Formelle de Concepts (AFC) (Ganter et Wille, 1999) car une clé de liage candidate s’avère être un “fermé” pour un opérateur de fermeture spécifique. Afin d’exploiter le cadre formel bien défini de l’AFC ainsi que les algorithmes d’extraction de fermés offerts par ce formalisme, (Atencia et al., 2014b, 2020) proposent une méthode qui s’appuie sur l’AFC pour extraire des clés de liage candidates. Les algorithmes proposés dans (Atencia et al., 2014a, 2020) peuvent être appliqués avec ou sans alignement des classes et des propriétés issues des jeux de donnés en entrée. En l’absence d’un alignement des classes, les clés de liage candidates découvertes sont dites *clés candidates générales* dans le sens où elles s’appliquent sur tout le jeu de données sans être spécifiques à une paire de classes particulière.

Dans (Abbas et al., 2020), les auteurs partent de l’hypothèse que les clés candidates générales, même les meilleures, peuvent générer des liens d’identité pour une paire de classes particulière et ne générer aucun lien pour les autres paires de classes. Il faut donc distinguer les paires de classes. En s’appuyant sur le formalisme des "pattern structures" (Ganter et Kuznetsov, 2001), une extension de l’AFC pour traiter des données complexes, les auteurs proposent un algorithme appelé `SLKPS` qui extrait des clés de liage candidates instanciant des paires de classes spécifiques. L’algorithme `SLKPS` ne nécessite pas d’alignement de classes en entrée.

Des mesures de qualité ont été proposées dans (Atencia et al., 2014a) pour évaluer les clés de liage candidates et sélectionner celles qui seront utilisées pour générer les liens d’identité. Les meilleures clés de liage candidates selon ces mesures ne permettent pas toujours de générer tous les liens d’identité possibles. De plus elles peuvent générer des liens erronés c.-à-d. des liens entre des instances qui ne sont pas identiques. L’objectif de ce travail de recherche est de proposer un algorithme qui sélectionne un ensemble de clés de liage candidates qui génèrent à la fois le maximum de liens d’identité corrects (rappel) et le minimum de liens erronés (précision). Cet algorithme s’appuie sur une sélection guidée par les paires de classes instanciées par les clés de liage candidates.

Dans suite de cet article, nous commençons par un état de l’art des approches de liage de données (Section 2). Ensuite nous présentons les notations et les définitions nécessaires (Section 3). Puis nous détaillons un algorithme qui sélectionne un ensemble de clés de liage candidates qui maximisent le nombre de liens d’identité corrects et qui minimisent le nombre de liens erronés (Section 4). Enfin, nous évaluons l’efficacité de l’algorithme à l’aide d’expérimentations sur différents jeux de données (Section 5).

2 État de l’art

Le liage de données est la tâche qui consiste à trouver, entre deux jeux de données RDF différents, des couples d’IRI (Internationalized Resource Identifier) représentant la même entité. Le résultat est un ensemble de liens d’identité entre ces IRIs. Pour effectuer cette tâche, des systèmes de liage de données, tels que LIMES (Ngomo et Auer, 2011) et SILK (Volz et al., 2009), s’appuient sur des *spécifications de liens* qui peuvent être numériques ou logiques, indiquant les conditions pour lesquelles deux IRI sont considérées comme identiques. Une spécification logique est un axiome dont les liens d’identité sont des conséquences logiques. Ces spécifications, contrairement aux spécifications numériques, peuvent être combinées avec des éléments

de connaissances provenant d'ontologies ou d'alignements entre ontologies, pour déduire des liens en utilisant un raisonneur.

Les spécifications à base de `clés` entrent dans cette dernière catégorie. Les clés ne sont généralement pas fournies avec les jeux de données, et pour cette raison, plusieurs algorithmes ont été proposés pour les découvrir automatiquement (Assi et al., 2020). Les clés sont utilisées pour trouver les instances identiques entre deux jeux de données, mais cela nécessite un alignement de propriétés et de classes ou un vocabulaire commun aux jeux de données en question. Ces clés sont découvertes pour chaque jeu de données indépendamment, ce qui implique qu'un ensemble de propriétés peut constituer une clé sur un jeu de données mais pas sur l'autre. Dans ce cas, les clés peuvent ne pas générer des liens d'identité. Partant de ce constat, deux solutions ont été étudiées : (i) utiliser des algorithmes et des mesures de sélection de clés adaptées au liage de données (Achichi et al., 2016; Farah et al., 2017), (ii) généraliser la notion clé par celle de *clé de liage* (Atencia et al., 2014a). Cette seconde solution est celle qui nous intéresse directement et c'est dans cette direction que cet article apporte une contribution.

3 Préliminaires et notations

3.1 Clé de liage

Un jeu de données RDF est un ensemble de triplets $\langle$`sujet,propriété,objet`$\rangle \in (U \cup B) \times U \times (U \cup B \cup L)$, où U est un ensemble d'IRI, B un ensemble de noeuds vides représentant des ressources anonymes, L un ensemble de littéraux pouvant être du type chaîne de caractères, entier, etc.

D_1

`Newspaper`

s	designation	pubCity	fYear
a_1	Le Monde	Paris	1944
a_2	Est R	Nancy	
a_3	The NYT	New York	1851
a_4	The WP	Washington	1877
a_5	USA T	McLean	1982
a_6	USA T	McLean	1982

`Person`

s	firstName	name	birthY
a_7	Nancy	Reagan	1921
a_8	Jane	Austen	
a_9	Albert	Camus	
a_{10}	Victor	Hugo	
a_{11}	Emily	Brontë	

D_2

`Journal`

s	désign	villeEdit	dateF
b_1	Le Monde	Paris	1944
b_2	Est R	Nancy	
b_3	The NYT	New York	1851
b_4	The WP	Washington	1877
b_5	USA T	McLean	1982

`Personne`

s	prénom	nom	annéeN
b_7	Nancy	Reagan	1921
b_8	Jane	Austen	
b_9	Albert	Camus	
b_{10}	Paris	Hilton	
b_{11}	Assia	Djebar	

FIG. 1: Exemple de deux jeux de données D_1 et D_2

Soit un jeu de données D, les ensembles $S = \{s | \langle s,p,o \rangle \in D\}$ et $P = \{p | \langle s,p,o \rangle \in D\}$ dénotent respectivement l'ensemble des sujets et l'ensemble des propriétés du jeu de données D. On dit que c est une classe atomique sur D si $\exists s \in S$ tel que $\langle s, rdf:type, c \rangle \in D$. La propriété `rdf:type` est utilisée pour indiquer qu'un sujet appartient à une classe. Par exemple, le triplet $\langle b_1$,`rdf:type,Journal`$\rangle$ déclare que b_1 est une instance de la classe `Journal`. L'ensemble des instances d'une classe atomique c dans D est dénotée $I_c = \{s \mid \langle s,$`rdf:type`$,c \rangle \in D\}$.

Une expression de classes C sur D est une disjonction de conjonction de classes atomiques. Par exemple $c_1 \sqcup (c_2 \sqcap c_3)$ est une expression de classes sur D où c_1, c_2 et c_3 sont des classes atomiques sur D, et où $\sqcup$ et $\sqcap$ sont les opérateurs de disjonction et de conjonction dans les logiques de descriptions (Baader et al., 2003). L'ensemble des instances de l'expression de classes $C \equiv \bigsqcup_{i=1}^{n}(\bigsqcap_{j=1}^{m} c_{ij})$ est $I_C = \bigcup_{i=1}^{n}(\bigcap_{j=1}^{m} I_{c_{ij}})$. L'ensemble des expressions de classes sur D est dénoté par Cl .

Sur la Figure 1, le triplet $\langle b_1, \texttt{villeEdit}, \texttt{"Paris"}\rangle$ établit le fait que le sujet b_1 possède la valeur "Paris" pour la propriété villeEdit. Contrairement au modèle relationnel, un sujet peut avoir zéro ou plusieurs valeurs pour une même propriété. L'ensemble des valeurs du sujet s pour la propriété p est dénoté $p(s)$ tel que $p(s) = \{o | \langle s, p, o\rangle \in D\}$. Par exemple, $\texttt{désign}(b_1) = \{\texttt{"Le Monde"}\}$.

Un lien d'identité est exprimé via la propriété owl:sameAs. Par exemple, le triplet qui exprime l'égalité des ressources a_1 et b_1 est $\langle a_1, \texttt{owl:sameAs}, b_1\rangle$. Cela signifie que a_1 et b_1 représentent la même entité qui est, dans notre exemple, le journal Le Monde. Pour simplifier, nous dénotons ce lien d'identité par la paire $\langle a_1, b_1\rangle$. Les clés de liage sont utilisées pour trouver ces liens d'identité.

Une clé de liage k entre deux jeux de donnés D_1 et D_2 est une expression de la forme :

$$k = (Eq, In, \langle C_1, C_2\rangle)$$

où Eq et In sont des ensembles de paires de propriétés dans $P_1 \times P_2$, $Eq \subseteq In \neq \emptyset$, C_1 (resp. C_2) est une classe atomique ou une expression de classes dans Cl_1 (resp. Cl_2). La clé de liage $k = (Eq, In, \langle C_1, C_2\rangle)$ instancie la paire de classes $\langle C_1, C_2\rangle$. On dit que k est *une clé de liage générale* si elle instancie la paire de classes $\langle \texttt{owl:Thing}, \texttt{owl:Thing}\rangle$. Rappelons que la classe owl:Thing (Smith et al., 2004) contient toutes les instances du jeu de données.

L'ensemble des liens d'identité générés par k est dénoté par $L(k)$ et vérifie :

$$\begin{aligned} L(k) = \{\langle s_1, s_2\rangle | & s_1 \in I_{C_1}, s_2 \in I_{C_2}, \\ & \forall \langle p_1, p_2\rangle \in Eq, p_1(s_1) = p_2(s_2), \\ & \forall \langle p_1, p_2\rangle \in In, p_1(s_1) \cap p_2(s_2) \neq \emptyset\} \end{aligned}$$

Par exemple la clé de liage k,

$$\begin{aligned} k = (&\langle \texttt{designation}, \texttt{désign}\rangle, \langle \texttt{pubCity}, \texttt{villeEdit}\rangle \\ &\langle \texttt{designation}, \texttt{désign}\rangle, \langle \texttt{pubCity}, \texttt{villeEdit}\rangle, \\ &\langle \texttt{Newspaper}, \texttt{Journal}\rangle) \end{aligned}$$

génère l'ensemble de liens $L(k) = \{\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \langle a_3, b_3\rangle, \langle a_4, b_4\rangle, \langle a_5, b_5\rangle, \langle a_6, b_5\rangle\}$.

Dans la suite, pour faciliter la lecture et sans nuire à la généralité de la démarche, nous ne considérons que l'ensemble In dans une clé de liage qui s'écrit alors $k = (In, \langle C_1, C_2\rangle)$, sachant que nous avons toujours $Eq \subseteq In$. De plus, le terme "classe" désigne aussi bien une classe atomique qu'une expression de classes.

Une clé de liage est "candidate" si elle génère au moins un lien et si elle est maximale [1] sur l'ensemble de liens qu'elle génère.

1. La définition de maximalité d'une clé de liage candidate est donnée dans (Atencia et al., 2020; Abbas et al., 2020).

Par exemple, soit l'ensemble de clés de liage $K = \{k_1, k_2, k_3\}$ qui génèrent le même ensemble de liens $L(k_1) = L(k_2) = L(k_3) = \{\langle a_1, b_1\rangle, \langle a_2, b_2\rangle, \langle a_3, b_3\rangle, \langle a_4, b_4\rangle, \langle a_5, b_5\rangle, \langle a_6, b_5\rangle\}$ tels que :

$k_1 = (\{\langle$`designation, désign`$\rangle, \langle$`pubCity, villeEdit`$\rangle\}, \langle$`Newspaper, Journal`$\rangle)$

$k_2 = (\{\langle$`designation, désign`$\rangle\}, \langle$`Newspaper, Journal`$\rangle)$

$k_3 = (\{\langle$`pubCity, villeEdit`$\rangle\}, \langle$`Newspaper, Journal`$\rangle)$

La clé de liage k_1 génère au moins un lien et elle est maximale sur K. Ainsi k_1 est une clé de liage candidate tandis que k_2 et k_3 ne le sont pas.

Une clé de liage peut générer des liens qui peuvent être corrects ou erronés selon l'avis d'un expert du domaine. De fait, les algorithmes de découverte de clés de liage recherchent des clés de liage candidates puis sélectionnent celles qui permettent de maximiser le nombre de liens corrects et de minimiser le nombre de liens erronés. Dans ce qui suit, nous discutons de la qualité des clés de liage retournées.

3.2 Évaluation des clés de liage candidates

La qualité d'une clé de liage dépend du nombre de liens corrects et du nombre de liens erronés qu'elle génère. Ainsi plus une clé de liage candidate génère de liens corrects et moins elle génère de liens erronés, plus elle est considérée de meilleure qualité pour le liage de données.

Cependant, lorsqu'un expert n'est pas présent – et c'est souvent le cas – pour vérifier qu'un lien est correct ou pas, la qualité d'une clé de liage candidate peut être évaluée en utilisant les mesures de couverture et de discriminabilité qui sont introduites dans (Atencia et al., 2014a).

La *couverture* d'une clé de liage $k = (In, \langle C_1, C_2\rangle)$, notée $cov(k)$, mesure le degré de généralité d'une clé de liage candidate. Plus la couverture d'une clé de liage candidate est élevée, plus grande est la probabilité que celle ci génère des liens corrects. La couverture de k se définit comme suit. Soit $L \subseteq (I_{C_1} \times I_{C_2})$ et $\pi_1(L) = \{s_1 \in I_{C_1} | \langle s_1, s_2\rangle \in L\}$, $\pi_2(L) = \{s_2 \in I_{C_2} | \langle s_1, s_2\rangle \in L\}$. $\pi_1(L)$ est l'ensemble des instances de C_1 apparaissant dans L et $\pi_2(L)$ est l'ensemble des instances de C_2 apparaissant dans L.

$$cov(k) = \frac{|\pi_1(L(k)) \cup \pi_2(L(k))|}{|I_{C_1} \cup I_{C_2}|}$$

Lorsque $cov(k)$ est égal à 1, cela signifie que toutes les instances de C_1 et de C_2 sont identifiées par k.

La mesure de *discriminabilité* évalue la capacité d'une clé de liage candidate à discriminer les instances. La discriminabilité fait l'hypothèse que les instances ayant des IRIs différentes au sein d'un même jeu de données sont distinctes (Unique Name Assumption UNA). Ainsi, si cette hypothèse est respectée, il ne devrait pas y avoir plus d'un lien impliquant une instance. Plus une clé de liage est discriminante, plus grande est la probabilité que les liens qu'elle génère soient corrects.

La discriminabilité d'une clé de liage candidate k se définit comme suit :

$$dis(k) = \frac{min(|\pi_1(L(k))|, |\pi_2(L(k))|)}{|L(k)|}$$

La couverture et la discriminabilité peuvent être agrégées par la *moyenne harmonique*, notée $hm_{cd}(k)$, qui est définie ci-après :

candidate	*cov*	*disc*	hm_{cd}
k_1	1	0.83	0.90
k_2	0.81	0.80	0.80
k_3	0.36	1	0.53
k_4	0.20	1	0.33
k_5	0.60	1	0.75
k_6	0.20	1	0.33

TAB. 1: Évaluation des clés de liage candidates de l'exemple

$hm_{cd}(k) = \frac{2 \times cov(k) \times dis(k)}{cov(k)+dis(k)}$

La clé de liage candidate qui possède la meilleure moyenne harmonique hm_{cd} est considérée comme étant celle de meilleure qualité. Cependant une seule clé de liage candidate, même la meilleure, ne permet pas toujours d'identifier tous les liens d'identité. Pour cette raison plusieurs clés de liage candidates peuvent être sélectionnées pour le liage de données. Les questions qui se posent alors sont "comment sélectionner ces clés de liage ?" et "comment détecter les clés qui sont les meilleures ?".

Le problème de sélection d'un sous-ensemble de clés de liage candidates peut s'énoncer comme suit. Soient D_1, D_2 deux jeux de données RDF et $K = \{k | k = (In, \langle C_1, C_2 \rangle)\}$ l'ensemble de clés de liage candidates entre D_1 et D_2. Il faut alors sélectionner un ensemble de clés de liage $KV \subseteq K$ – où V signifie "valide" – qui génèrent le maximum de liens corrects et le minimum de liens erronés entre D_1 et D_2.

Par exemple, soit $K = \{k_1, k_2, k_3, k_4, k_5, k_6\}$ l'ensemble des clés de liage candidates entre les jeux de données D_1 et D_2 de la Figure 1.

k_1 =({⟨designation, désign⟩, ⟨pubCity, villeEdit⟩}, ⟨Newspaper, Journal⟩)
k_2 =({⟨designation, désign⟩, ⟨pubCity, villeEdit⟩, ⟨fYear, dateF⟩},
⟨Newspaper, Journal⟩)
k_3 =({⟨pubCity, prénom⟩}, ⟨Newspaper, Personne⟩)
k_4 =({⟨firstName, villeEdit⟩}, ⟨Person, Journal⟩)
k_5 =({⟨name, nom⟩, ⟨firstName, prénom⟩}, ⟨Person, Personne⟩)
k_6 =({⟨name, nom⟩, ⟨firstName, prénom⟩, ⟨anneeN, birthY⟩}, ⟨Person, Personne⟩)

Comment alors sélectionner un ensemble de clés de liage qui génèrent tous les liens d'identité corrects entre D_1 et D_2 ? Une première proposition est de sélectionner les "Top M" clés de liage de K, c.-à-d. un sous-ensemble de clés de liage candidates composé des M meilleures candidates en termes de hm_{cd}. Néanmoins, ce sous-ensemble ne parvient pas toujours à maximiser le nombre de liens corrects.

Dans l'exemple, nous calculons hm_{cd} pour chaque clé de liage candidate (voir la Table 1). Nous sélectionnons par exemple les Top 2 de K c.-à-d. k_1 et k_2 qui sont les deux meilleures clés de liage selon hm_{cd}. Cependant, ces clés de liage ne génèrent pas tous les liens d'identité entre les jeux de données D_1 et D_2. Cela s'explique par le fait que k_1 et k_2 n'instancient pas la paire de classes ⟨Person, Personne⟩. Par conséquent, les liens d'identité entre ces deux classes ne sont pas générés. Une solution intuitive consiste à sélectionner la meilleure clé de liage pour chaque paire de classes, c.-à-d. k_1, k_3, k_4 et k_5. Néanmoins, ces clés de liage génèrent

des liens erronés. En effet, les instances `Le Monde` et `Paris Hilton` sont jugées identiques, ce qui n'est pas acceptable. Cela s'explique par le fait que k_3 instancie la paire de classes $\langle$`Newspaper,Personne`$\rangle$ dont les classes sont "incompatibles", ce qui signifie qu'aucune entité ne peut être représentée par ces deux classes en même temps.

En nous appuyant sur ces constats, nous proposons ci-après un algorithme de sélection d'un sous-ensemble de clés de liage candidates, où la sélection, guidée par les paires de classes, maximise le nombre de liens corrects et minimise le nombre de liens erronés.

4 Sélection de clés de liage guidée par les classes

Les algorithmes ci-dessous s'appuient sur la méthode `SLKPS` détaillée dans (Abbas et al., 2020) pour extraire les clés de liage candidates. Contrairement aux algorithmes qui retournent des clés de liage candidates générales, la méthode `SLKPS` a la particularité de retourner des clés de liage candidates en spécifiant les paires de classes qui sont instanciées. En outre, `SLKPS` peut générer des clés de liage candidates qui instancient des paires d'expressions de classes.

L'algorithme `LKSA` sélectionne un sous-ensemble de clés de liage KV qui maximise le nombre de liens corrects et minimise le nombre de liens erronés. `LKSA` s'appuie sur deux opérations principales, `TopLkClass` et `RelevantClassForLk`, qui sont détaillées juste après.

Algorithm 1 Sélection des clés de liage candidates

Input
K : ensemble de clés de liage candidates retournées par *SLKPS* pour D_1 et D_2
N : entier, nombre de meilleures clés de liage par paire de classes à sélectionner
Output
KV : ensemble de clés de liage sélectionnées pour le liage de données entre D_1 et D_2.

1: **function** LKSA(K, N)
2: $pcc(K) \leftarrow$ ensemble de paires de classes candidates
3: $KTop \leftarrow$ TOPLKCLASS($K, pcc(K), N$)
4: $KV \leftarrow$ RELEVANTCLASSFORLK($KTop, pcc(K)$)
5: **return** KV
6: **end function**

Algorithm 2 Sélection des N meilleures clés de liage par paire de classes candidate

Input k,$pcc(K)$, N
Output $KTop$: les N meilleures clés de liage par paire de classes en terme de hm_{cd}

1: **function** TOPLKCLASS(K, $pcc(K)$, N)
2: $KTop \leftarrow \emptyset$
3: **for each** $\langle C_1, C_2 \rangle \in pcc(K)$ **do**
4: $KTop[\langle C_1, C_2 \rangle] \leftarrow N$ argmax$_k$ $hm_{cd}(k)$ tel que $k = (In, \langle C_1, C_2 \rangle) \in K$
5: **end for**
6: **return** $KTop$
7: **end function**

`TopLkClass` : cette fonction sélectionne un sous-ensemble de clés de liage candidates $KTop$ qui vise à maximiser le nombre de liens corrects générés.

Algorithm 3 Sélection des clés de liage instanciant les paires de classes pertinentes pour le liage

Input $KTop$, $pcc(K)$
Output KV

1: **function** RELEVANTCLASSFORLK($KTop$,$pcc(K)$)
2: $Class1 \leftarrow \{C_1 | \langle C_1, C_2 \rangle \in pcc(K)\}$
3: $Class2 \leftarrow \emptyset; KV \leftarrow \emptyset$
4: **for each** $C_1 \in Class1$ **do**
5: $bestC2 \leftarrow \text{argmax}_{C_2}\ hm_{cd}(KTop[\langle C_1, C_2 \rangle])$
6: $Class2 \leftarrow Class2 \cup \{bestC2\}$
7: **end for**
8: **for each** $C_2 \in Class2$ **do**
9: $bestC1 \leftarrow \text{argmax}_{C_1}\ hm_{cd}(KTop[\langle C_1, C_2 \rangle])$
10: $KV \leftarrow KV \cup KTop[\langle bestC1, C_2 \rangle]$
11: **end for**
12: **return** KV
13: **end function**

L'ensemble des paires de classes qui sont instanciées par les clés de liage candidates de K est dénoté $pcc(K) = \{\langle C_1, C_2 \rangle | \exists k \in K$ tel que $k = (In, \langle C_1, C_2 \rangle\}$. La fonction `TopLkClass` part d'un ensemble de clés de liage candidates K et sélectionne pour chaque paire de classes candidates $\langle C_1, C_2 \rangle \in pcc(K)$ les N meilleures clés de liage candidates en terme de hm_{cd}, où N est un nombre fixé à l'avance par l'utilisateur.

Sur l'exemple en cours, l'ensemble des paires de classes candidates est :
$pcc(K) = \{\langle$`Newspaper,Journal`$\rangle, \langle$`Newspaper,Personne`$\rangle, \langle$`Person,Journal`$\rangle$, $\langle$`Person,Personne`$\rangle\}$. Nous appliquons `TopLkClass` avec $N = 1$. Pour chaque paire de classes de $pcc(K)$ est sélectionnée la meilleure clé de liage candidate qui l'instancie. Ainsi, la paire $\langle$`Newspaper,Journal`$\rangle$ est instanciée par les clés de liage candidates k_1 et k_2, et k_1 est sélectionnée car $hm_{cd}(k_1) > hm_{cd}(k_2)$. La procédure est itérée pour chacune des paires de classes de $pcc(K)$ et le calcul de $KTop$ retourne l'ensemble de clés de liage $\{k_1, k_3, k_4, k_5\}$. Cet ensemble $KTop$ permet d'identifier tous les liens d'identité corrects entre les jeux de données D_1 et D_2 de l'exemple. Cependant, certaines clés de liage de $KTop$ génèrent aussi des liens erronés. En effet, les instances `Le Monde` et `Paris Hilton` sont toujours jugées identiques par k_3 qui instancie $\langle$`Newspaper,Personne`$\rangle$.

Dans la suite, nous dirons qu'une paire de classes est "pertinente pour le liage (de données)" s'il existe au moins un lien d'identité correct entre ces deux classes. Par exemple, la paire de classes $\langle$`Newspaper,Journal`$\rangle$ est pertinente pour le liage tandis que la paire $\langle$`Newspaper,Personne`$\rangle$ ne l'est pas.

Nous avons vu au dessus que $KTop$ génère des liens erronés parce que certaines clés de liage instancient des paires de classes non pertinentes pour le liage de données. La fonction `RelevantClassForLk` est justement conçue pour sélectionner un ensemble $KV \subseteq KTop$ comportant des clés qui instancient uniquement des paires de classes qui ont plus de chances d'être pertinentes pour le liage de données.

`RelevantClassForLk` : cette fonction vise à minimiser le nombre de liens erronés générés par $KTop$. Lorsque il n'y a pas d'expert pour juger de la validité des liens générés, la

fonction `RelevantClassForLk` retourne un sous-ensemble de paires de classes qui ont le plus de chances d'être pertinentes pour le liage de données.

Nous faisons l'hypothèse que plus le nombre d'entités partagées par deux classes est élevé, plus grande est la probabilité que cette paire de classes soit pertinente pour le liage de données. Pour cela, nous calculons le degré de recouvrement de chaque paire de classes instanciées par les clés de liage de $KTop$.

Le recouvrement d'une paire de classes $\langle C_1, C_2 \rangle$ instanciées par l'ensemble des clés de liage $KTop[\langle C_1, C_2 \rangle] = \{k | k = (In, \langle C_1, C_2 \rangle)\}$ peut se mesurer par $hm_{cd}(KTop[\langle C_1, C_2 \rangle])$.

Lorsque $hm_{cd}(KTop[\langle C_1, C_2 \rangle]) = 1$ cela signifie que $KTop[\langle C_1, C_2 \rangle]$ identifie toutes les entités de C_1 et toutes les entités de C_2, tout en garantissant que chaque entité est représentée par une seule instance dans C_1 et un seule instance dans C_2. Plus le recouvrement d'une paire de classes est élevé, plus grande est la probabilité que les classes soient pertinentes pour le liage de données.

La fonction `RelevantClassForLk` permet de sélectionner dans $KTop$ les clés de liage candidates qui instancient les paires de classes qui ont plus de chances d'être pertinentes pour le liage. Ainsi dans l'algorithme 3, pour chaque classe C_1 de $Class1 = \{C_1 | \langle C_1, C_2 \rangle \in pcc(K)\}$, `RelevantClassForLk` sélectionne la classe C_2 qui donne le meilleur recouvrement pour C_1. Cette classe C_2 est alors ajoutée à l'ensemble $Class2$. Ensuite, pour chaque classe $C_2 \in Class2$, l'algorithme sélectionne la classe C_1 qui donne le meilleur recouvrement pour C_2. L'ensemble des clés de liage $KTop[\langle C_1, C_2 \rangle]$ est alors ajouté à KV.

Dans l'exemple, $Class1 = \{$`Newspaper`, `Person`$\}$ et la classe `Newspaper` possède un meilleur recouvrement avec la classe `Journal` que la classe `Personne` car $hm_{cd}(\{k_1\})$ qui instancie la paire $\langle$`Newspaper`, `Journal`$\rangle$ est supérieure à $hm_{cd}(\{k_3\})$ qui instancie la paire $\langle$`Newspaper`, `Personne`$\rangle$. La procédure est réitérée pour la classe `Person` et c'est la classe `Personne` qui est sélectionnée, et $Class2 = \{$`Journal`, `Personne`$\}$. Le processus est réitéré avec les classes de $Class2$. Pour la classe `Journal`, la classe `Newspaper` qui donne le meilleur recouvrement est sélectionnée et $\{k_1\}$ est ajouté à KV. Pour la classe `Personne`, la classe `Person` est sélectionnée et $\{k_5\}$ est ajouté à KV. Ainsi, la sortie de l'algorithme est l'ensemble $KV = \{k_1, k_5\}$ qui identifie tous les liens d'identité corrects et uniquement ceux-là entre les jeux de données D_1 et D_2, c.-à-d. l'ensemble des liens $\{\langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \langle a_3, b_3 \rangle, \langle a_4, b_4 \rangle, \langle a_5, b_5 \rangle, \langle a_6, b_5 \rangle, \langle a_7, b_7 \rangle, \langle a_8, b_8 \rangle, \langle a_9, b_9 \rangle\}$.

Dans la suite, nous testons les hypothèses avancées, les algorithmes présentés ainsi que leur efficacité.

5 Expérimentations

L'objectif de ces expérimentations est de vérifier empiriquement l'efficacité de l'algorithme `LKSA` pour sélectionner un sous-ensemble de clés de liage candidates qui maximise le nombre de liens corrects et minimise le nombre de liens erronés. Toutes nos expérimentations [2] ont été réalisées en utilisant un MacBook Pro Intel Core i7, 2,6 GHz avec 16 Go de RAM alloués à la machine virtuelle Java.

2. Les détails des expérimentations sont donnés dans https ://gitlab.inria.fr/nabbas/lksa.

5.1 Jeux de données et protocole expérimental

Nous avons considéré les jeux de données décrits dans la Table 2.

— Pour les jeux de données *Restaurants* et *Person1* de l'OAEI 2010 [3], des liens d'identité ont été fournis entre :
  — Les instances de la classe `Restaurant` pour les jeux de données *Restaurant1* et *Restaurant2*.
  — Les instances de la classe `Person` pour les jeux de données *Person11* et *Person12*.
  Comme notre travail porte sur les clés de liage instanciant différentes paires de classes, nous avons également considéré la découverte de liens d'identité entre les instances de la classe `Adress` présentes dans ces jeux de données. De plus nous avons considéré les jeux de données *pr1* qui est la fusion de *Restaurant1* et de *Person1*, et *pr2* qui est la fusion de *Restaurant2* et *Person2*.

— Pour les jeux de données *Abox1* et *Abox2* de *SPIMBench Sandbox* de l'OAEI 2018 [4], nous avons considéré les classes `Programme`, `BlogPost` et `NewsItem`.

— Le jeu de données *Db-Yago* [5], est composé de 10 classes de DBpedia et de 10 classes de Yago qui sont : `Actor`, `Album`, `Book`, `City`, `Film`, `Montain`, `Organisation`, `Scientist`, `Museum` et `University`.

Tâche	Jeux de données	#class.	#inst.	#prop.	#triples	#lk
Restaurants[1]	Restaurant1	3	339	6	1 130	6
	Restaurant2	3	2 256	6	7 520	
Person1[1]	Person11	4	2000	13	9 000	80
	Person12	2	1000	12	7000	
pr	pr1	7	2 339	19	10 130	90
	pr2	5	3 256	18	14 520	
SPIMBench[2]	Abox1	10	1 126	46	10 001	7 401
	Abox2	17	1 130	66	10 022	
Db-Yago[3]	Db	10	442 403	47	8 618 591	16 299
	Yago	10	1 064 548	49	10 988 374	

TAB. 2: Description des jeux de données utilisés dans les expérimentations

Pour chaque jeu de données, nous avons généré les clés de liage candidates avec la méthode `SLKPS`. Nous avons ensuite sélectionné un sous-ensemble de clés de liage candidates *KV* avec l'algorithme `LKSA`. Les résultats de `LKSA` sont comparés avec les *M* meilleures clés de liage générales (`Top` *M*) en termes de hm_{cd} en s'assurant que $M = |KV|$. La performance des deux méthodes est calculée en terme de *précision*, *rappel* et *F-mesure*.

5.2 Résultats et discussion

Les résultats de ces expérimentations sont donnés dans la Table 3. Nous remarquons que dans tous les cas, nous obtenons avec `LKSA` une *F-mesure* supérieure ou égale à celle des *M* meilleures clés de liage candidates. `LKSA` obtient une meilleure *F-mesure* que `Top` *M* pour les jeux de données *Person1*, *pr*, *SPIMBench* et *Db-Yago*. Le rappel est amélioré pour *Person1* et *pr*. La précision est améliorée pour *pr*, *SPIMBench* et *Db-Yago*.

3. http ://oaei.ontologymatching.org/2010/im/index.html
4. http ://oaei.ontologymatching.org/2018/spimbench.html
5. https ://github.com/lgalarra/vickey

Tâche	sélection	#lk	rap.	F-mesure	prec.	temps
Restaurants	Top M	2	0.634	0.719	0.83	
	LKSA, $N = 1$	2	0.634	0.719	0.83	$< 1''$
Person1	Top M	2	0.649	0.787	1	
	LKSA, $N = 1$	2	0.954	**0.866**	0.793	$< 1''$
pr	Top M	4	0.772	0.681	0.61	
	LKSA, $N = 1$	4	0.882	**0.838**	0.799	$< 1''$
SPIMBench	Top M	7	0.772	0.788	0.804	
	LKSA, $N = 1$	7	0.772	**0.792**	0.813	$< 8''$
Db-Yago	Top M	22	0.736	0.089	0.047	$< 17''$
	LKSA, $N = 1$	22	0.630	**0.504**	0.420	$< 4min$

TAB. 3: Les résultats de la sélection

La *F-mesure* n'est pas modifiée pour le jeu données *Restaurants* car les M meilleures clés de liage correspondent exactement aux clés de liage sélectionnées par LKSA.

Pour les jeux de données *Person1* et *pr*, les M meilleures clés de liage n'instancient pas toutes les paires de classes possibles, ce qui explique un rappel inférieur. L'algorithme LKSA quant à lui donne un meilleur rappel car il sélectionne les clés de liage qui instancient plus de paires de classes que les M meilleures clés de liage.

Pour les jeux de données *Db-Yago* et *SPIMBench*, nous améliorons la précision car LKSA sélectionne plus de clés de liage instanciant des paires de classes pertinentes comparant à Top M, d'où la génération de moins de liens erronés.

Nous avons démontré le potentiel de l'algorithme LKSA en expérimentant sur plusieurs jeux de données. Ici, nous constatons l'importance de distinguer les paires de classes qui sont instanciées par les clés de liage, ce qui est justement l'objectif de la méthode SLKPS (Abbas et al., 2020).

6 Conclusion

Dans ce travail, nous avons traité le problème de la sélection de clés de liage candidates en exploitant les paires de classes qu'elles instancient. Nous avons proposé l'algorithme LKSA qui repose à la fois sur la sélection des meilleures clés de liage pour chaque paire de classes ainsi que sur la sélection des paires de classes dites pertinentes. Les résultats expérimentaux montrent que l'algorithme LKSA obtient de meilleurs résultats qu'une sélection uniquement basée sur les mesures de discriminabilité et couverture (Atencia et al., 2014a).

Par la suite, nous comptons adapter l'algorithme LKSA pour sélectionner des clés de liage interdépendantes, comme celles qui sont introduites dans (Atencia et al., 2020).

Références

Abbas, N., J. David, et A. Napoli (2020). Discovery of link keys in RDF data based on pattern structures : Preliminary steps. In *Proceedings of CLA*, pp. 235–246.

Achichi, M., M. B. Ellefi, D. Symeonidou, et K. Todorov (2016). Automatic key selection for data linking. In *EKAW*, pp. 3–18. Springer.

Al-Bakri, M., M. Atencia, J. David, S. Lalande, et M.-C. Rousset (2016). Uncertainty-sensitive reasoning for inferring same as facts in linked data. In *Proceedings of ECAI*, pp. 698–706. IOS press.

Al-Bakri, M., M. Atencia, S. Lalande, et M.-C. Rousset (2015). Inferring same-as facts from linked data : an iterative import-by-query approach. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Assi, A., H. Mcheick, et W. Dhifli (2020). Data linking over rdf knowledge graphs : A survey. *Concurrency and Computation : Practice and Experience*, e5746.

Atencia, M., J. David, et J. Euzenat (2014a). Data interlinking through robust linkkey extraction. In *ECAI*, pp. 15–20.

Atencia, M., J. David, et J. Euzenat (2014b). What can fca do for database linkkey extraction ? In *3rd ECAI workshop on What can FCA do for Artificial Intelligence ?(FCA4AI)*, pp. 85–92.

Atencia, M., J. David, J. Euzenat, A. Napoli, et J. Vizzini (2020). Link key candidate extraction with relational concept analysis. *Discrete applied mathematics 273*, 2–20.

Baader, F., D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, et al. (2003). *The description logic handbook : Theory, implementation and applications*. Cambridge university press.

Farah, H., D. Symeonidou, et K. Todorov (2017). Keyranker : Automatic rdf key ranking for data linking. In *Proceedings of K-CAP*, pp. 1–8.

Ganter, B. et S. O. Kuznetsov (2001). Pattern structures and their projections. In *ICCS*, pp. 129–142. Springer.

Ganter, B. et R. Wille (1999). *Formal concept analysis : mathematical foundations*. Springer.

Ngomo, A.-C. N. et S. Auer (2011). Limes—a time-efficient approach for large-scale link discovery on the web of data. In *IJCAI*.

Saïs, F., N. Pernelle, et M.-C. Rousset (2007). L2r : A logical method for reference reconciliation. In *Proceedings of AAAI*, pp. 329–334.

Smith, M. K., C. Welty, et D. L. McGuinness (2004). OWL web ontology language guide. Recommendation, W3C. `https://www.w3.org/TR/owl-guide/`.

Volz, J., C. Bizer, M. Gaedke, et G. Kobilarov (2009). Silk-a link discovery framework for the web of data. *LDOW 538*.

Summary

In this paper we are interested in link key selection in RDF datasets. A link key is used to find identity links between two entities in the web of data. Such pair of entities determines a pair of classes which in turn can guide the selection link key of better quality. For this purpose, we propose an algorithm for selecting link keys based on adapted quality measures. Finally, experiments on different data sets show the potential and the effectiveness of the algorithm.