Thibaut SOULARD, Fatiha SAÏS, Joe RAAD

Étude de transférabilité de clés pour un liage explicable de données entre graphes de connaissances

Volume 6, nº 1-2 (2025), p. 9-33.

https://doi.org/10.5802/roia.91

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Volume 6, n° 1-2, 2025, 9-33

Étude de transférabilité de clés pour un liage explicable de données entre graphes de connaissances

Thibaut SOULARD[a], Fatiha SAÏS[a], Joe RAAD[a]

[a] LISN, CNRS (UMR 9015), Université Paris Saclay (France)
E-mail : soulard@lisn.fr, fatiha.sais@lisn.fr, joe.raad@universite-paris-saclay.fr
URL *:* https://perso.lisn.upsaclay.fr/sais/
URL *:* http://www.joe-raad.com/.

Résumé. — Le liage de données dans les graphes de connaissances est un problème crucial et de longue date ; il consiste à déterminer des liens d'identité entre les descriptions des entités de deux graphes désignant une même entité du monde réel (*e.g.*, même personne, même livre, même protéine). Les clés, qui sont des sous-ensembles de propriétés permettant d'identifier chaque instance dans un graphe, sont des éléments importants pour la découverte de ces liens d'identité. L'approche classique de liage de données fondée sur les clés consiste à découvrir un ensemble de clés dans chaque graphe, et d'appliquer ensuite une procédure de fusion des ensembles de clés obtenues. Cependant, cette approche peut conduire à la réduction du nombre de clés valides dans les deux graphes et peut parfois être très coûteuse en temps de calcul. Nous proposons dans ce travail une nouvelle approche de liage de données fondée sur le transfert de clés découvertes sur un graphe source vers un graphe cible impliqué dans la tâche de liage de données. Ce transfert s'appuie sur un alignement de propriétés connues a priori et sur le calcul de métriques permettant de valider la qualité des clés transférées vers le graphe cible. Nous avons conduit des expérimentations sur plusieurs jeux de données extraits du web de données (DBpedia, Wikidata et YAGO) afin d'évaluer la qualité du liage de données et le gain en temps de calcul obtenu grâce au transfert de clés.

Mots-clés. — Liage de données, découverte de clés, graphes de connaissances, transfert de clés, Explicabilité.

1. Introduction

De nombreuses sources de données structurées sont aujourd'hui disponibles sous la forme de données ouvertes liées. Ces données contenant des millions voire des milliards de triplets RDF (Resource Description Framework)[(1)] forment un espace de connaissances structurés en graphes de connaissances liés. Ces graphes contiennent des connaissances qui sont généralement exprimées en RDF, comme des faits de la forme

[(1)] `https://www.w3.org/RDF/`

<sujet, propriété, objet> tels que <Macron, presidentDe, France>. En proposant RDF comme standard, les chercheurs de la communauté du Web sémantique ont promu la représentation des connaissances et des données sous la forme de graphes. Dans de tels graphes, les nœuds représentent des entités (par exemple, Paris) qui peuvent avoir des types représentés par des classes (par exemple, Paris est une ville), les arcs représentent des relations entre les entités (par exemple, estMaireDe). Parfois, les différents types et relations sont représentés dans une ontologie OWL 2 (Web Ontology Language)[(2)], qui définit leurs interrelations et des axiomes tels que la subsomption, la disjonction et la fonctionnalité des propriétés. Nous pouvons citer les graphes de connaissances DBpedia, YAGO et Wikidata dévelopés du côté académique, et le *Google Knowledge Graph* ou *eBay Knowledge Graph* du côté commercial. Ces graphes de connaissances contiennent des entités (telles que des personnes, des protéines ou des livres) et des faits décrivant ces entités. Ces graphes sont soit multi-domaines, comme Yago, DBpedia ou Wikidata, ou spécifiques à un domaine d'application, comme la géographie avec Geonames[(3)] ou la biologie avec Bioportal[(4)].

Pour pouvoir exploiter toute la richesse des données et des connaissances contenues dans ces graphes, il est important d'établir des liens sémantiques entre leurs entités. Les liens d'identité sont très importants pour l'enrichissement des graphes de connaissances, puisqu'ils permettent de relier des descriptions qui réfèrent la même entité du monde réel, ce qui permet de propager les valeurs des propriétés d'une description d'entité à une autre. Différents types d'approches ont été développées pour la détection des liens d'identité dans les graphes de connaissances (voir [3, 6, 18] pour une revue récente de la littérature).

En effet, détecter ces liens entre les entités de deux graphes G_1 et G_2 est un problème complexe, d'une part du fait de la nécessité de comparer les entités deux-à-deux ce qui donne un espace de comparaison de taille $n_1 \times n_2$ (avec n_1 et n_2 le nombre de descriptions d'entités dans G_1 et G_2, respectivement).

Dans cet article, nous nous appuyons sur les travaux de liage de données utilisant des clés [2, 12] pour inférer de nouveaux liens d'identité. Les clés sont des sous-ensembles de propriétés permettant d'identifier chaque entité dans un graphe (*e.g.*, l'ensemble {nom, prénom, e-mail} peut être une clé pour identifier les personnes). L'intérêt d'utiliser les clés pour le liage de données est double : (i) il permet de distinguer des sous-ensembles de propriétés ayant un fort impact sur la décision d'établir ou de ne pas établir un lien d'identité et qui pourront donc potentiellement conduire à des résultats de liage avec un bon taux de précision dans le cas de données hétérogènes et incomplètes, et (ii) il permet également de réduire le nombre de couples de propriétés-valeurs considérés lors de la comparaison des descriptions d'entités. Ce dernier point est d'autant plus important lorsque les graphes contiennent des entités qui sont décrites par un grand nombre de propriétés. En revanche, l'acquisition des clés est une tâche complexe, puisque qu'elles sont soit spécifiées par un expert du domaine ou bien

[(2)] https://www.w3.org/OWL/
[(3)] https://en.wikipedia.org/wiki/GeoNames
[(4)] https://bioportal.bioontology.org/

découvertes automatiquement par des outils dédiés comme [5, 13, 16, 17]. Lorsque les graphes sont volumineux ou contiennent un grand nombre de propriétés, le coût en temps de calcul de la découverte de clés peut atteindre quelques jours, ce qui est important, bien que la tâche puisse être exécutée hors-ligne.

Aussi, nous proposons dans ce travail une nouvelle approche de liage de données fondée sur le transfert de clés découvertes sur un graphe source vers un graphe cible impliqué dans la tâche de liage de données. Ce transfert s'appuie sur un alignement de propriétés connues a priori et sur le calcul de métriques permettant de valider la qualité des clés transférées vers le graphe cible. Nous avons conduit des expérimentations sur plusieurs jeux de données extraits du web de données (DBpedia, Wikidata et YAGO) afin d'évaluer la qualité du liage de l'approche et le gain en temps de calcul obtenu grâce au transfert de clés.

L'article est organisé comme suit : en section 2, nous présentons les travaux connexes. Ensuite, en section 3 nous donnons les définitions préliminaires des notions manipulées dans ce travail. En section 4, nous décrivons la méthodologie que nous avons mise en place pour étudier la transférabilité des clés d'un graphe source à un graphe cible. Ensuite, la section 5, présente l'évaluation expérimentale, une analyse et une discussion des résultats. Enfin, en section 6 nous concluons l'article et donnons quelques perspectives.

2. État de l'Art

Lorsque nous cherchons à combiner ou à exploiter des données issues de différentes sources, il est essentiel de détecter si des descriptions distinctes font référence à une même entité du monde réel (par exemple, une même personne, un même livre ou un même gène). Cette tâche est connue sous le nom du problème de *liage de données* et il est également désigné dans la littérature par de nombreux autres termes tels que le *record linkage*, la résolution d'entités, la déduplication ou encore la réconciliation de références.

De nombreuses approches ont été proposées dans la littérature pour résoudre ce problème (voir les revues présentées dans [3, 6, 18]). Certaines reposent sur des références vers d'autres sources, d'autres exploitent des identifiants uniques (par exemple, ISBN, SIRET, NSS), tandis que d'autres encore s'appuient sur les descriptions des entités pour calculer un score de similarité ou une probabilité, permettant ainsi d'établir un lien d'identité ou de similarité entre ces entités. Pour ces dernières approches, une variété de techniques a été développée. Les premières méthodes considèrent les descriptions des entités comme des sacs de mots, sur lesquels des mesures de similarité adaptées sont appliquées. D'autres travaux, comme ceux présentés dans [10], exploitent des configurations exprimant des conditions de liage. Ces configurations précisent les sources ou parties de sources à connecter, les paires de propriétés à comparer, les mesures de similarité utilisées (par exemple, Jaccard, Jaro-Winkler) ainsi que la fonction d'agrégation des différents scores de similarité. Dans le même esprit, d'autres recherches [2, 12] ont exploré l'utilisation de règles exprimées en logique du premier ordre pour inférer des liens d'identité à l'aide de raisonnements logiques.

Ces règles traduisent des contraintes de clés qui peuvent être représentées dans une ontologie OWL 2 à l'aide du constructeur `owl:hasKey`.

Une autre famille d'approches [8, 9, 18] repose sur l'apprentissage de représentations vectorielles des entités, permettant de calculer des distances dans un espace latent. Ces dernières méthodes se distinguent par leur efficacité et leur robustesse face au bruit et à l'incomplétude des données. Cependant, elles présentent une limite importante : les modèles utilisés pour l'apprentissage des représentations sont souvent de type boîte noire, ce qui rend difficile l'explication des décisions de liage obtenues. Dans cet article, nous nous inscrivons dans le cadre des approches basées sur les clés, en raison de leur capacité à produire des décisions de liage certaines et explicables sans nécessité d'entraînement de modèles sur des quantité importantes de données.

Dans un contexte où les sources de données sont volumineuses, hétérogènes et caractérisées par des schémas complexes (par exemple, un grand nombre de classes et de propriétés), la définition manuelle des clés devient impraticable, en particulier à l'échelle des données du Web (des millions voire des milliards de triplets). Pour répondre à cette problématique, plusieurs approches de découverte de clés ont été proposées, chacune reposant sur des hypothèses variées concernant l'hétérogénéité des schémas, la complétude et la qualité des données. Certaines approches, comme celles présentées dans [16, 17], partent du principe que les schémas sont homogènes (c'est-à-dire qu'ils partagent les mêmes propriétés et classes) et prennent en compte des données incomplètes ainsi que des exceptions éventuelles, souvent dues à des erreurs dans les données. D'autres travaux, tels que [13] supposent que les données sont complètes pour les propriétés multi-valuées et d'autres tels que [1, 5], supposent des schémas hétérogènes.

Pour lier les entités représentées dans deux graphes G_1 et G_2, une approche idéale consisterait à découvrir les clés indépendamment dans G_1 et G_2, puis à calculer leur intersection ou à appliquer une procédure de fusion (voir [11] pour un exemple). Cependant, cette méthode peut être très coûteuse en temps de calcul et aboutir à un nombre limité de clés communes entre les deux graphes. L'approche proposée dans [1, 5] contourne ce problème en exploitant des schémas hétérogènes et en considérant simultanément les deux problèmes d'alignement de propriétés et de découverte de clés valides dans les deux graphes. Comme les approches classiques de découverte de clés, cette méthode nécessite toutefois d'exploiter les données des deux graphes pour identifier les clés pertinentes.

Dans ce travail, nous proposons une nouvelle approche de liage explicable de données reposant sur la découverte des clés dans un seul graphe, suivie de leur transfert vers le second graphe impliqué dans le processus de liage. Nous explorons en particulier la question de la *transférabilité* des clés, c'est-à-dire la possibilité de transférer les clés découvertes dans un graphe G_1 vers un autre graphe G_2. Plus précisément, nous cherchons à identifier l'ensemble des clés découvertes dans G_1 qui restent valides dans G_2 en s'appuyant sur des mesures de qualité des clés transférées. Cette approche permet de s'affranchir de la phase de découverte des clés dans G_2 et de l'étape de fusion des clés entre les deux graphes. À notre connaissance, le problème de la transférabilité des clés n'a pas été abordé dans la littérature.

3. Préliminaires

Dans cette section, nous présentons et définissons les concepts fondamentaux qui sous-tendent notre approche de transfert de clés. Ces notions sont essentielles pour comprendre les principes et les mécanismes mis en œuvre afin de faciliter le transfert efficace de clés entre différents graphes de connaissances.

3.1. Graphes de connaissances

Nous donnons ci-dessous la définition formelle d'un graphe de connaissances et des alignements de propriétés que nous illustrons à travers un exemple de deux graphes montrés en Figures 3.1 et 3.2.

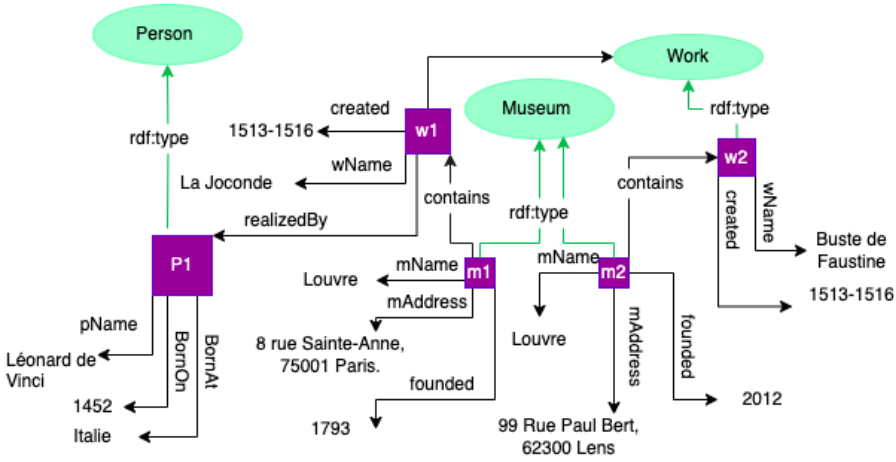

Figure 3.1. G_1- Exemple de graphe de connaissances RDF dans le domaine culturel

Définition 3.1 (Graphe de connaissances RDF). — *Nous considérons un graphe de connaissances RDF défini par un couple* $(\mathcal{O}, \mathcal{G})$*, où :*

- $\mathcal{O} = (\mathcal{C}, \mathcal{P})$ *est une ontologie représentée en OWL 2 et composée d'un ensemble de classes* $\mathcal{C}$ *et de propriétés* $\mathcal{P}$ *pouvant être soit de type* `owl:ObjectProperty`*, dont le domaine et le co-domaine sont des classes, ou de type* `owl:DataTypeProperty`*, dont le domaine est une classe et le co-domaine est un type de données atomique (e.g, date, string, integer).*
- $\mathcal{G}$ *est un ensemble de faits représenté par des triplets de la forme {(sujet, propriété, objet) | sujet* $\in \mathcal{I}$*, propriété* $\in \mathcal{P}$*, objet* $\in \mathcal{I} \cup \mathcal{L}$*}, où* $\mathcal{I}$ *est l'ensemble d'instances de classes* $c \in \mathcal{C}$ *désignées par des IRI (Internationalized Resource Identifier),* $\mathcal{P}$ *est l'ensemble des propriétés, et* $\mathcal{L}$ *est l'ensemble des littéraux (tels que les nombres et les chaînes de caractères). On notera* $I_C \subseteq \mathcal{I}$ *l'ensemble d'instances de la classe* $c \in \mathcal{C}$*.*

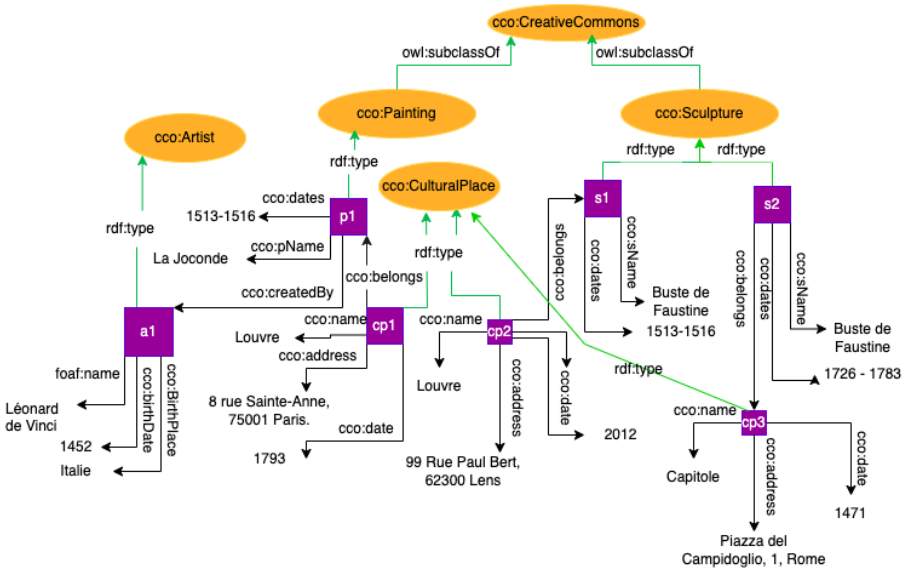

FIGURE 3.2. G_2 - Autre exemple de graphe de connaissances RDF dans le domaine culturel

DÉFINITION 3.2 (Alignement de propriétés). — *Un alignement entre deux propriétés p_1 et p_2 de O_1 et O_2 (resp.) est une relation de mise en correspondance exprimant une relation d'équivalence sémantique entre p_1 et p_2 que nous notons par : $(p_1 \equiv p_2)$.*

Exemple 3.3. — Un alignement de propriétés entre les deux graphes RDF G_1 et G_2 des Figures 3.1 et 3.2 peut contenir les relations d'équivalence suivantes :

$\{(\texttt{wName} \equiv \texttt{cco:pName}), (\texttt{wName} \equiv \texttt{cco:sName}), (\texttt{mName} \equiv \texttt{cco:name}),$

$(\texttt{pName} \equiv \texttt{cco:foaf:name}), (\texttt{created} \equiv \texttt{cco:dates}), \ldots\}$.

Nous pouvons remarquer que des relations $n - m$ peuvent être rencontrées dans un alignement de propriétés comme dans le cas de la propriété `wName` alignée avec les deux propriétés `cco:pName` et `cco:sName`.

DÉFINITION 3.4 (Alignement de classes). — *Un alignement entre deux classes c_1 et c_2 de O_1 et O_2 (resp.) est une relation de mise en correspondance exprimant une relation d'équivalence sémantique entre c_1 et c_2 que nous notons par : $(c_1 \equiv c_2)$.*

3.2. CLÉS

Une clé est un ensemble de propriétés qui permet d'identifier de manière unique chaque instance de classe dans un graphe. Dans le langage OWL 2, il est possible de déclarer des clés pour des expressions de classes (*i.e.*, CE) en utilisant le constructeur `owl:hasKey`. Ce constructeur distingue parmi les propriétés qui composent la clé les propriétés de type `owl:DataTypeProperty` (*i.e.*, DPE) de celles qui sont de type `owl:ObjectProperty` (*i.e.* OPE).

```
HasKey( CE ( OPE1 ... OPEm ) ( DPE1 ... DPEn ) )
    ∀ x , y , z1 , ... , zm , w1 , ... , wn~:
    if x ∈ (CE)C and x ∈ NAMED and
       y ∈ (CE)C and y ∈ NAMED and
       ( x , zi ) ∈ (OPEi)OP and ( y , zi ) ∈ (OPEi)OP and
       zi ∈ NAMED for each 1 <= i <= m and
       ( x , wj ) ∈ (DPEj)DP and ( y , wj ) ∈ (DPEj)DP
                                                  for each 1 <= j <= n
    then x = y
```

Comme exprimé par la sémantique OWL 2 ci-dessus, dans les graphes de connaissances, les clés peuvent être utilisées pour détecter des liens d'identité `owl:sameAs` entre les descriptions d'entités dans les graphes de données RDF. Dans ce travail, nous ne considérons que les classes simples et utilisons seulement les propriétés de type `owl:DataTypeProperty` qui sont supportées par tous les outils de liage de données.

De manière générale, lorsqu'un ensemble de propriétés est défini comme une clé pour une classe, son invalidité dans un graphe de données peut résulter d'erreurs dans les valeurs des propriétés, de la présence de doublons non identifiés ou d'exceptions.

Exemple 3.5. — Dans le graphe G_1 de la figure 3.1 la propriété `wName` est une clé valide pour la classe `Work`. Dans le graphe G_2 de la figure 3.2 la propriété `cci:sName` qui est l'équivalent de `wName` ne peut être une clé que si on tolère deux exceptions, puisque les instances $s1$ et $s2$ partagent la même valeur `"Buste de Faustine"`.

Si des paires d'instances issues de différents graphes RDF ne satisfont pas une clé, elles peuvent être considérées comme des candidates potentielles à l'établissement d'un lien d'identité. En effet, toute paire d'instances partageant les mêmes valeurs pour l'ensemble des propriétés d'une clé peut être envisagée comme candidate au liage d'instances.

Exemple 3.6. — Soient la paire de classes déclarées comme équivalentes `Museum` et `cco:CulturalPlace` des graphes G_1 et G_2. Supposons également l'alignement de propriétés donné dans l'exemple 3.3. Si on considère la clé composée des deux propriétés `cco:name, cco:address` comme valide dans les deux graphes G_1 et G_2, son exploitation pour le liage de données peut conduire à l'inférence d'un lien d'identité pour deux paires d'instances `(m1, cp1)` et `(m2, cp2)` puisqu'elles partagent les mêmes valeurs pour ces propriétés.

Différentes sémantiques de clés ont été proposées dans le domaine du web sémantique (voir [4] pour une comparaison théorique et expérimentale). Elles diffèrent selon la stratégie appliquée pour gérer les propriétés multi-valuées et les valeurs non renseignées des propriétés. Dans cet article nous considérons la sémantique *S*-clé, correspondant à celle du constructeur `owl:hasKey`[(5)] qui n'impose pour chaque propriété de la clé que l'existence d'une unique valeur commune aux deux instances comparées. La définition suivante formalise cette sémantique.

[(5)] `https://w3.org/TR/owl2-direct-semantics/`

Définition 3.7 (Sémantique d'une *S*-clé [4]). — *La sémantique d'une* S*-clé* $\{p_1, \ldots, p_n\}$ *pour une classe*[6] C *est donnée dans la règle en logique du premier ordre suivante :*

$$\forall x \forall y \forall z_1 \ldots z_n \left(C(x) \wedge C(y) \wedge \bigwedge_{i=1}^{n} \Big(p_i(x, z_i) \wedge p_i(y, z_i) \Big) \rightarrow x = y \right)$$

avec x et y sont des variables instanciant des instances nommées de classes.

Déclarer que l'ensemble $\{p_1, \ldots, p_n\}$ est une *S*-clé pour une classe C est noté par *S*-clé$(C, (p_1, \ldots, p_n))$. Nous notons également prop(k) l'ensemble de propriétés formant la clé k.

Remarque 3.8. — Nous notons que cette sémantique de clés est compatible avec celle du constructeur `owl:hasKey`[7] qui s'applique seulement aux instances nommées et pas aux nœuds blancs. Ces derniers sont donc ignorés par notre approche.

Définition 3.9 (Instanciation d'une clé). — *Soit k une* S*-clé*$(C, (p_1, \ldots, p_n))$ *pour une classe C dans* $\mathcal{G}$*. L'instanciation de la clé k pour une instance i de la classe C dans* $\mathcal{G}$ *est un n-uplet de valeurs de propriétés* $\pi(k, i)$ *défini comme suit :*

$$\pi(k, i) = \{(v_1, \ldots, v_n) \mid \{(i, p_1, v_1), \ldots, (i, p_n, v_n)\} \subseteq G\}$$

Exemple 3.10. — Soit k = `(Museum, (mName, mAddress))` une clé de la classe `Museum` dans le graphe G_1. Son instantiation pour l'instance $m1$ dans G_1 est l'ensemble $\pi(k, m1)$ = `{("Louvre", "8 rue Sainte-Anne, 75001 Paris")}`.

4. Approche de transfert de clés

Pour établir des liens entre les instances de deux graphes de connaissances G_1 et G_2, l'approche classique de liage de données basée sur les clés suit généralement deux étapes. Tout d'abord, elle identifie un ensemble de clés K_1 dans G_1 et un autre ensemble K_2 dans G_2. Ensuite, une procédure de fusion de ces clés est appliquée afin de déterminer un ensemble de clés valides pour les deux graphes et de générer des règles permettant d'inférer des liens d'identité entre leurs instances. Dans la littérature, deux méthodes de fusion de clés ont été proposées : la *fusion par intersection*, qui consiste simplement à calculer l'intersection des clés découvertes dans les deux graphes, et la *fusion par produit cartésien*, introduite dans [11], qui consiste à effectuer le produit cartésien des ensembles de propriétés apparaissant dans K_1 et K_2, tout en conservant uniquement les clés minimales (c'est-à-dire celles pour lesquelles aucune autre clé n'a un ensemble de propriétés strictement inclus).

Lorsque les graphes sont de grande taille (contenant des millions, voire des milliards de triplets) ou comportent un grand nombre de propriétés, la découverte de clés peut s'avérer coûteuse et affecter leur couverture. En effet, privilégier des clés minimales

[6] Cela pourrait être également une expression de classe définie en OWL 2 `https://www.w3.org/TR/owl2-direct-semantics/#Class_Expressions`

[7] `https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Keys`

(fusion par intersection) ou des clés spécifiques (fusion par produit cartésien) entraîne une réduction de la couverture des clés obtenues et, par conséquent, du nombre de liens d'identité potentiels.

Dans cet article, nous proposons une approche de transfert de clés qui permet d'éliminer la nécessité de découvrir des clés dans les deux graphes ainsi que l'étape de fusion des ensembles de clés obtenus. Plus précisément, notre approche consiste à transférer des clés découvertes dans un graphe G_1 vers un graphe G_2 tout en préservant leur qualité.

4.1. Mesures de qualité des clés

Nous avons proposé plusieurs mesures de qualité, certaines permettent de sélectionner les clés candidates au transfert du graphe G_1 vers le graphe G_2 et d'autres permettent d'évaluer la qualité des clés transférées dans le graphe G_2.

Nous définissons ci-dessous le support, le nombre d'exceptions et le taux de discriminabilité d'une clé.

Le support d'une clé mesure la proportion d'instances pour lesquelles cette clé est valide. Une clé avec un support élevé est plus représentative du graphe et donc plus fiable pour des tâches comme le liage de données.

Définition 4.1 (Support d'une clé). — *Soit k une* S-*clé*$(C, (p_1, \ldots, p_n))$ *pour une classe C dans $\mathcal{G}$. Le support de k dans $\mathcal{G}$ est le nombre d'instances de C ayant au moins une valeur pour chacune des propriétés* $\text{prop}(k)$. *Plus formellement :*

$$\text{support}(k) = |\{x \mid \forall p \in \text{prop}(k), \exists y, (x, p, y) \in \mathcal{G}\}|$$

Définition 4.2 (Support relatif d'une clé). — *Le support relatif d'une clé est le support ramené au nombre d'instances de la classe. Sa valeur est normalisée dans* $[0..1]$. *Il est formellement défini comme suit :*

$$\text{support}^R(k) = \frac{\text{support}(k)}{|\{x \mid C(x) \in \mathcal{G}\}|}$$

En exploitant le support relatif il sera plus facile de fixer un seuil adapté au nombre d'instances d'une classe.

Un autre critère de qualité d'une clé est son degré de discriminabilité qui mesure à quel à point la clé permet de distinguer une instance parmi toutes les autres instances du graphe. Comme dans [5, 13], pour mesurer ce degré de discriminabilité on s'appuie sur le nombre de partitions d'instances partageant les mêmes valeurs pour les propriétés de la clé et qui sont réduites à une seule instance.

Définition 4.3 (Partition relâchée d'instances d'une clé). — *Étant donnés une clé k =*S-*clé*$(C, (p_1, \ldots, p_n))$ *dans un graphe de données RDF $\mathcal{G}$ et $I_k \subseteq I_C$ l'ensemble des instances de la clé k, la partition relâchée de l'ensemble d'instances de C pouvant être formée par k est définie par l'ensemble :* $\Delta(k, \mathcal{G}) = \{\delta_1, \delta_2, \ldots, \delta_m\}$ *avec* $\delta_i \in \Delta(k, \mathcal{G})$ *l'ensemble d'instances de I_k partageant au moins une valeur pour chaque propriété de la clé k.*

Définition 4.4 (Taux de discriminabilité d'une clé). — *Le taux de discriminabilité d'une clé* k =S-*clé*$(C, (p_1, \ldots, p_n))$ *est le nombre d'ensembles d'instances* $\delta_i \in \Delta(k, G)$ *réduits à une seule instance relativement au nombre total d'ensembles de* $\Delta(k, G)$. *Plus formellement,*

$$\text{discr}(k, G) = \frac{|\{\delta_i \mid \delta_i \in \Delta(k, G), \mid \delta_i \mid = 1\}|}{|\Delta(k, G)|}$$

Lorsque les graphes de connaissances contiennent des entités dupliquées (*i.e.* l'hypothèse du nom unique non vérifiée) ou contiennent des propriétés dont les valeurs sont erronées, la découverte de clés strictes devient impossible. C'est alors pour cela que des approches comme [16] ont introduit la notion de clés tolérant quelques exceptions dont le nombre est fixé par un seuil.

Ci-dessous nous donnons la définition du nombre d'exceptions d'une clé inspirée de [16].

Définition 4.5 (Nombre d'exceptions d'une clé). — *Le nombre d'exceptions d'une clé* k, *de la forme* S-*clé* $(C, (p_1, \ldots, p_n))$, *correspond au nombre d'instances de* C *qui ne respectent pas cette clé. Formellement, il est défini comme suit :*

$$\text{ex}(k) = |\{x \mid \forall p \in \text{prop}(k), \exists y \neq x \text{ tel que } V(x, p) \cap V(y, p) \neq \emptyset\}|,$$

où $V(x, p)$ *désigne l'ensemble des valeurs associées à l'instance* x *pour la propriété* p *(de même pour* $V(y, p)$*).*

La définition 4.5 permet de définir des clés avec des exceptions dont le nombre est calculé par rapport au nombre d'instances de la classe dans le graphe. Cependant, cette définition n'est plus pertinente dans le cas où les valeurs des propriétés ne sont pas toutes renseignées, ce qui est souvent le cas dans les graphes de connaissances (*e.g.*, la propriété `fax-number (P2900)` de la classe `human (Q5)` n'est renseignée que pour 31 instances dans le graphe Wikidata). Dans cet article nous proposons une nouvelle définition du nombre d'exceptions qui tient compte du support des propriétés de la clé.

Définition 4.6 (Taux d'exceptions d'une clé). — *Le taux d'exceptions d'une clé* k *de la forme* S-*clé*$(C, (p_1, \ldots, p_n))$ *relativement au nombre d'instances supportant* k *est formellement défini comme suit :*

$$\text{ex}^R(k) = \frac{\text{ex}(k)}{\text{support}(k)}$$

4.2. Alignement de propriétés

Lorsque les graphes de connaissances sont décrits à l'aide d'ontologies distinctes, il est essentiel d'aligner leurs propriétés afin de traduire les clés d'un graphe en exploitant les propriétés correspondantes dans l'autre graphe. Dans la section 4.3.3 nous détaillons la procédure d'alignement de propriétés utilisée et celle concernant la traduction des clés qui en découle.

Pour calculer les alignements des propriétés décrivant les instances dans deux graphes RDF, nous considérons les ontologies auxquelles sont conformes ces graphes et avons utilisé une nouvelle approche présentée dans [14]. Cette méthode exploite les informations terminologiques et conceptuelles (*i.e.*, définition des domaines et co-domaines des propriétés) directement disponibles dans les ontologies comme les labels, les descriptions, les types, les domaines de valeurs des propriétés ainsi que les labels et les descriptions des classes. Ensuite, à l'aide de techniques de type *transformer* telles que BERT [7] issues du traitement automatique de la langue, nous calculons des scores de similarité entre les propriétés des ontologies décrivant les deux graphes.

Cette méthode permet d'avoir un alignement $n - m$ de propriétés qui permet de capturer, notamment, des cas où les ontologies contiennent des propriétés alternatives implicites et lorsqu'elles diffèrent au niveau de la structure. Un exemple d'un alignement $1 - n$ judicieux pourrait être l'alignement de la propriété `wk:P625` (coordinate location) de Wikidata avec les propriétés `geo:lat` (latitude) & `geo:long` (longitude) ou bien l'alignement de `wk:P35` (head of state) avec les relations `db:monarch` & `db:leaderName` de DBpedia.

4.3. Méthodologie de transfert de clés

L'approche se déroule en plusieurs étapes. Tout d'abord la découverte de clés dans le premier graphe dont le choix peut être arbitraire ou dépendant des caractéristiques sur son volume et la richesse des descriptions de ses entités. Ensuite, un sous ensemble de clés est sélectionné en fonction du taux d'exceptions relatif (voir définition 4.6) en exploitant les alignements de propriétés des deux graphes, les clés découvertes sont traduites en utilisant les propriétés alignées du deuxième graphe. Enfin, la discriminabilité (voir définition 4.4) des clés obtenues est calculée dans le deuxième graphe. Seules celles qui respectent un seuil de discriminabilité sont gardées dans le résultat. Nous détaillons ci-dessous chacune de ces étapes.

4.3.1. *Découverte de clés dans le graphe d'origine*

Pour chaque classe d'instances du graphe G_1, pour lesquelles il existe un alignement avec une classe du graphe G_2, appliquer un outil de découverte de clés pour générer un ensemble de clés $\mathcal{K}_1$. Pour cette première étape, nous avons utilisé l'outil SAKey [16] qui découvre des *S*-clés en considérant un nombre d'exceptions passé en paramètre. Il est à noter, comme indiqué dans la section 2, que SAKey, étant basé sur le calcul des non-clés en premier lieu, ne permet pas de fournir des mesures quantitatives des clés telles que le support et la discriminabilité.

4.3.2. *Sélection des clés candidates au transfert*

Pour chaque clé découverte, nous évaluons sa qualité en termes de nombre d'exceptions générées et son support. Ces deux mesures nous permettent ensuite d'obtenir le taux d'exceptions relatif $\mathrm{ex}^R(k)$ et de sélectionner seulement les clés qui respectent le taux d'exceptions relatif maximum $\mathrm{ex}^R_{\max}$ passé en paramètres.

4.3.3. *Traduction des clés sélectionnées en exploitant l'alignement de propriétés*

Cette étape consiste à identifier les clés alignées, *i.e.*, le sous-ensemble de clés $K_1 \subseteq \mathcal{K}_1$ découvertes sur G_1 pour lesquelles des alignements de propriétés existent avec le graphe G_2.

Définition 4.7 (Traduction d'une clé). — *Soient deux graphes $\mathcal{G}$ et $\mathcal{G}'$ et $\mathcal{M} = \{(p_1 \equiv p'_1), \ldots, (p_m \equiv p'_m)\}$ l'ensemble de m alignements de propriétés de $\mathcal{G}$ et $\mathcal{G}'$. Soit k une* S*-clé$(C, (p_1, \ldots, p_n))$ pour une classe C dans $\mathcal{G}$. Une traduction $\rho(k, \mathcal{M})$ de la clé k pour le graphe $\mathcal{G}'$, selon l'ensemble des alignements de propriétés $\mathcal{M}$, est un ensemble de* S*-clés de la forme* S*-clé$(C', (p'_1, \ldots, p'_n))$ tel que il existe un alignement de propriétés $m = \{(p_1 \equiv p'_1), \ldots (p_n \equiv p'_n)\} \subseteq \mathcal{M}$ et que nous avons $C \equiv C'$.*

4.3.4. *Évaluation de la qualité des clés traduites dans le graphe cible.*

Enfin, pour chaque traduction de clé retenue nous évaluons sur le graphe cible G_2 sa qualité en termes de taux d'exceptions relatif $\text{ex}^R(k)$ en considérant le même seuil qu'en étape (2). Cette évaluation nous permet de détecter les clés transférées qui auraient dégénéré vers une non-clé, *i.e.* un ensemble de propriétés ayant un taux d'exceptions $\text{ex}^R(k) > \text{ex}^R_{\max}$, et de les écarter ensuite pour les étapes de liage d'entités.

5. Évaluation expérimentale

Dans cette section, nous présentons l'évaluation expérimentale de notre framework sur une série de jeux de données. Toutes les expériences sont réalisées sur un processeur « Intel® Xeon® E5-2630 v4 » avec 10 cœurs et 128 Go de RAM. À des fins de reproductibilité, le code, les données et les résultats sont disponibles sur le web[(8)]. Pour l'évaluation expérimentale, nous avons considéré deux types de jeux de données : (i) deux graphes avec deux vocabulaires hétérogènes où nous devons premièrement appliquer un outil d'alignement des propriétés qui peut conduire à un alignement $n - m$ et (ii) deux graphes avec un vocabulaire homogénéisé, ce qui signifie que les deux graphes utilisent le même ensemble de propriétés.

5.1. Graphes avec des vocabulaires hétérogènes

Nous avons mené cette première expérience sur deux graphes hétérogènes provenant de DBpedia[(9)] et Wikidata[(10)] qui utilisent des vocabulaires différents pour représenter les concepts et les entités (c'est-à-dire qu'ils utilisent des classes et des propriétés différentes). Pour gérer l'hétérogénéité des vocabulaires, nous avons utilisé un outil d'alignement des classes et des propriétés présenté dans [14]. En outre, nous supposons que ces graphes peuvent contenir des erreurs et des descriptions redondantes d'entités. Pour découvrir les clés dans le cas de graphes imparfaits, nous admettons un taux d'exception maximal relatif $\text{ex}^R_{\max}$ dont la valeur est choisie dans l'intervalle [0 % ; 10 %].

[(8)] Gitlab : `https://gitlab.lisn.upsaclay.fr/soulard/KeyTransfer`
[(9)] `https://www.dbpedia.org/`
[(10)] `https://www.wikidata.org/`

Le but de cette évaluation expérimentale est : (i) d'évaluer le gain de performance d'une approche de liage de données basée sur les clés lorsqu'elle applique une procédure de transfert de clés du graphe source vers un graphe cible ; (ii) d'évaluer la qualité des clés obtenues en évaluant leur impact sur la discriminabilité des entités dans les deux graphes et (iii) d'évaluer la nécessité d'une vérification de la mise en correspondance pour éviter toute dégénérescence des clés. Dans le reste de cette section, nous présentons le workflow que nous avons suivi pour cette expérience et nous décrivons comment les jeux de données ont été préparés. Enfin, nous donnons les résultats en termes de nombre de clés obtenues avant et après le transfert, en termes de discriminabilité des clés sur les deux graphes et en termes de temps d'exécution pour montrer si le transfert de clés se traduit par un gain de performances.

5.1.1. *Méthodologie expérimentale*

Nous considérons un graphe source G_S, un graphe cible G_T, un ensemble $\mathcal{M}$ de correspondances d'équivalence des propriétés des deux graphes et $\text{ex}^R_{\max}$ le taux d'exceptions maximal relatif et nous procédons comme suit :

(1) Préparation du jeux de données et alignement des propriétés
(2) Découverte de l'ensemble des clés SK dans G_S, chaque clé satisfaisant un taux d'exception $< \text{ex}^R_{\max}$.
(3) Utilisation de $\mathcal{M}$ pour traduire les clés découvertes et sélectionner un sous-ensemble de clés $RK \subset SK$ dans lequel chaque clé a toutes ses propriétés alignées sur au moins une propriété dans G_T.
(4) Calcul du support, le taux d'exception relatif et la discriminabilité de chaque clé de RK dans G_S
(5) Calcul du support, le taux d'exception relatif et la discriminabilité de chaque clé de RK dans G_T.

5.1.2. *Jeux de données*

Nous avons examiné deux jeux de données extraits de DBpedia et de Wikidata.

Pour DBpedia, nous avons extrait la partie du graphe RDF représentant les instances de la classe `Person` (c'est-à-dire ayant comme `rdf:type Person`) extraites en décembre 2022. Comme le montre le tableau 5.1, ce premier ensemble de données contient 1 863 013 entités et 18 960 propriétés. Nous constatons qu'un nombre important de ces propriétés correspond à un IRI mal encodé ou erroné (IRI qui ne correspond pas à un IRI d'une des ontologies décrivant ce graphe), comme la présence de `dbo:award21n,2` au lieu de `dbo:award`. Ces propriétés sont alors supprimées du graphe.

Pour ce qui est de Wikidata, nous avons extrait du graphe Wikidata du 3 mars 2021[(11)] toutes les instances de la classe Human, représentées par le triplet `<?IRI, wdt:P31, wd:Q5>`. Le deuxième ensemble de données obtenu se compose de 3 020 916 entités et de 3 506 propriétés.

[(11)] Disponible sur le service RDF HDT : `https://www.rdfhdt.org/datasets/`

TABLE 5.1. Classes Person/Human dans DBpedia/Wikidata

	# Entités	# Propriétés	# Triplets
DB:Person	1 863 013	18 960 (239)	12 474 844
WK:Q5	3 020 916	3 506 (135)	7 503 002

5.1.3. *Alignement des propriétés*

Pour ces deux ensembles de données, nous avons également pris en compte les ontologies décrivant des instances de personnes dans DBpedia et dans Wikidata, qui contiennent respectivement 4 604 et 10 472 propriétés. Nous avons appliqué un algorithme d'alignement des propriétés décrit dans [14] ; l'algorithme applique une mesure de similarité basée sur le plongement, dans un espace vectoriel de faible dimension, de la description textuelle associée aux propriétés et aux classes des deux ontologies.

Après l'extraction des données, une étape de prétraitement a été appliquée pour réduire le nombre de propriétés inutilisables. Cette étape est motivée par le temps d'exécution de l'algorithme de découverte de clés, qui dépend du nombre de propriétés dans un graphe. Pour ce faire, nous ne conservons que les propriétés utilisées dans les deux graphes pour lesquels nous disposons d'au moins un alignement de propriétés. Nous ne conservons que les propriétés de type `owl:DatatypeProperty`, c'est-à-dire dont les valeurs sont des littéraux. Nous avons appliqué cette stratégie parce qu'une propriété non transférable ou commune aux deux graphes ne peut pas être utilisée pour la réécriture des clés. En outre, une clé composée de `owl:ObjectProperty` ne peut être utilisée que par des outils qui exploitent la propagation des scores de similarité entre les paires d'instances, tels que [2, 12]. Après ce prétraitement, nous obtenons 239 et 135 de propriétés ainsi que 12 474 844 et 7 503 002 triplets pour DBpedia et Wikidata respectivement, comme présenté dans le tableau 5.1.

5.1.4. *Résultats*

Dans cette section nous présentons les résultats de notre approche. Dans le tableau 5.2 nous présentons tout d'abord les résultats d'un point de vue quantitatif exprimés en termes du nombres de clés découvertes dans chaque graphe ainsi en nombre de clés traduites obtenues. Ensuite nous commentons les résultats qualitatifs des clés en utilisant la mesure de discriminabilté donnés dans le tableau 5.3. Enfin, nous présentons dans le tableau 5.4 l'évolution du temps de calcul et nous commentons les situations favorables au transfert de clés d'un graphe à un autre.

NOMBRE DE CLÉS DÉCOUVERTES. — Dans le tableau 5.2, nous présentons le nombre de clés-sources (SK) pour chaque graphe de connaissances, ainsi que le nombre de clés traduites (RK) lors de la vérification des clés dans le graphe cible. Cette vérification est divisée en deux étapes. Dans la première étape, nous filtrons les clés traduites sans instanciation (voir définition 3.9), c'est-à-dire avec un support égal à 0. Dans la

TABLE 5.2. Résultats de la découverte de clés dans l'ensemble de données Person/Human (SK → Source Key, RK → Rewritten Key)

ex^R_{max}		0 %	0,5 %	2 %	5 %	10 %
DB	SK	835	642	642	643	648
	RK brut	1 572	1 199	1 199	1 200	1 205
	RK (sup≠ 0)	69	64	64	65	68
	RK vérifié	49	52	54	56	60
WK	SK	357	237	238	240	241
	RK brut	475	327	328	330	331
	RK (sup≠ 0)	82	82	82	83	83
	RK vérifié	41	48	49	55	57

deuxième étape, nous vérifions que les clés traduites respectent le taux d'exception relatif maximal autorisé.

La raison principale de ce filtrage est directement liée à l'absence d'instanciation des réécritures dans le graphe cible. Ce manque d'instanciation peut s'expliquer par le fait que différents points de vue ou contextes peuvent être utilisés pour décrire les entités, de sorte qu'un graphe peut représenter les entités personnes dans un cadre social et un autre d'un point de vue professionnel. Cela peut également s'expliquer par un alignement $n - m$ de certaines propriétés entraînant une réécriture inutile, ces propriétés n'étant pas utilisées pour décrire les entités `Human`.

Nous pouvons observer une diminution drastique du nombre de clés entre un taux d'exception relatif de 0 % et 0,5 %. Ceci est dû à la suppression de la nécessité d'une clé précise lorsque le taux d'exception est plus faible. En effet, lorsque nous avons une clé possible composée de $\{P_1\}$ et que le taux d'exception relatif est de 0,1, le premier seuil n'autorisera pas cette clé et obligera l'algorithme à ajouter au moins une autre propriété à la clé pour réduire le taux d'exception, ce qui peut donner lieu à de nouvelles clés telles que $\{P_1, P_2\}$ et $\{P_1, P_3\}$.

Enfin, si l'on considère le nombre de clés sources et de clés traduites, on constate que ces étapes sont nécessaires. En effet, lorsque nous partons de DBpedia et pour tout taux d'exception, nous perdons plus de 90 % des clés et pour Wikidata, nous perdons plus de 80 %.

DISCRIMINABILITÉ DES CLÉS. Dans le tableau 5.3, nous présentons la discriminabilité des clés dans les graphes source et cible lorsque DBpedia est considéré comme un graphe source et inversement. Les résultats sont présentés pour un ex^R_{max} variant dans [0 % ; 10 %] d'exceptions.

Nous pouvons observer l'évolution de la discriminabilité de l'ensemble de clés en fonction du taux d'exception relatif maximum. À partir de 0,5 %, nous atteignons un plateau (80,7 % et 28,9 % en partant de DBpedia et 0,9 % et 80,7 % pour Wikidata)

TABLE 5.3. Évolution de la discriminabilité des clés transférés en fonction du taux d'exception relatif maximal.

ex^R_{max}		0 %	0,5 %	2 %	5 %	10 %
Départ DB	DB	0,3 %	80,7 %	80,7 %	80,7 %	80,7 %
	WK	0,9 %	28,9 %	28,9 %	28,9 %	28,9 %
Départ WK	WK	0,3 %	0,9 %	0,9 %	1,0 %	1,0 %
	DB	0,1 %	80,7 %	80,7 %	80,7 %	80,7 %

qui ne nous permet pas de discriminer un plus grand nombre d'entités. Ce plateau est d'autant plus intéressant que pour des taux d'exception maximaux plus élevés, nous introduisons également plus de bruit (c'est-à-dire des exceptions), ce qui rend la tâche finale de mise en relation des entités plus difficile. En outre, ces résultats montrent également une limitation de la méthode de découverte de clés, avec un pourcentage de discriminabilité très faible pour les clés découvertes sur Wikidata et appliquées sur le même graphe, avec au maximum seulement 1,0 % de discriminabilité possible.

TEMPS D'EXÉCUTION. Dans le tableau 5.4, nous présentons les différentes valeurs de temps d'exécution en fonction de la stratégie et du point de départ suivis. Le point le plus critique est le temps d'exécution pour la découverte des clés source de DBpedia (279 minutes). Cette étape est le goulot d'étranglement de la stratégie classique de fusion des clés découvertes dans les deux graphes. Par conséquent, en commençant par le graphe ayant le temps d'exécution le plus rapide pour l'étape *découverte SK*, c'est-à-dire celui

TABLE 5.4. Temps d'exécution en minutes pour chaque scénario et chaque étape. (SK → Source Key, RK → Rewritten Key)

ex^R_{max}		0 %	0,5 %	2 %	5 %	10 %
Classique DB ∩ WK	Découverte SK DB	279,3	276,0	270,8	278,0	282
	Découverte SK WK	5,1	5,0	5,1	5,0	5,1
	Évaluation SK DB	33,7	25,3	25,4	25,4	25,5
	Évaluation SK WK	3,4	1,5	1,5	1,5	1,6
	Total	**321,7**	**308,0**	**302,9**	**310,1**	**313,9**
Transfert Départ DB	Découverte SK	279,3	276,0	270,8	278,0	282
	Évaluation SK	33,7	25,3	25,4	25,4	25,5
	Évaluation RK	11,7	11,6	11,7	11,9	11,7
	Total	**324,8**	**313,0**	**308,0**	**315,4**	**319,2**
Transfert Départ WK	Découverte SK	5,1	5,0	5,1	5,0	5,1
	Évaluation SK	3,4	1,5	1,5	1,5	1,6
	Évaluation RK	5,9	5,8	5,9	6,0	5,9
	Total	**14,5**	**12,4**	**12,6**	**12,6**	**12,6**

ayant le moins de propriétés, (ici le graphe source serait Wikidata) nous pouvons réduire drastiquement le temps d'exécution total, seulement 14,5 minutes au lieu de 321,7 minutes pour un taux d'exception de 0 %. Cependant, si nous commençons par DBpedia, nous perdons les avantages de cette nouvelle stratégie, tout en restant dans une fourchette de temps d'exécution proche de celle des méthodes classiques basées sur la fusion de clés.

5.1.5. *Discussion*

Grâce à cette première évaluation expérimentale, nous avons pu analyser le comportement des clés traduites sur les graphes source et cible. Malgré la nécessité de vérifier les clés traduites dans le graphe cible, nous avons montré que ces clés traduites peuvent encore discriminer un pourcentage significatif des entités dans le graphe cible (Tableau 5.3), même lorsque plus de 80 % des clés ont été perdues lors de la vérification (Tableau 5.2). Néanmoins, la définition des exceptions relatives appliquées aux clés trouvées par SAKey [16] a une incidence sur la monotonicité des clés et, par conséquent, sur la complétude de l'ensemble des clés transférées. En effet, comme nous le démontrons dans l'exemple du tableau 5.5, la clé K_1 composée de $\{P_1, P_2\}$ respecte le taux maximal d'exceptions de 50 % (puisqu'elle comporte 40 % d'exceptions), mais la clé K_2 composée de $\{P_1, P_2, P_3\}$ ne respecte plus ce seuil avec un support plus faible de 2 tout en ayant le même nombre d'exceptions, ce qui donne un taux d'exceptions de 100 % alors que la clé $K_3 = \{P_1, P_2, P_3, P_4\}$ respecte le taux maximum autorisé avec un taux d'exceptions de 0 %.

Table 5.5. Exemple de clés non monotones.

	P_1	P_2	P_3	P_4
e_1	A1	A2	-	-
e_2	B1	B2	A3	B4
e_3	B1	B2	A3	C4
e_4	D1	D2	-	-
e_5	E1	E2	-	-

La complétude est d'autant plus importante que des clés plus précises (c'est-à-dire avec un support plus faible) peuvent différencier des entités que des clés plus générales (c'est-à-dire avec un support plus élevé) ne peuvent pas différencier. L'exemple du tableau 5.5 illustre ce phénomène, la clé K_1 étant incapable de différencier e_2 de e_3 alors que K_3 le peut. On peut donc espérer augmenter le taux de discriminabilité en ajoutant des clés traduites plus précises différenciant des cas particuliers. Cependant, cet ajout n'est pas trivial, car la monotonicité des clés se perd avec le taux d'exception relatif. Mais, bien sûr, l'ajout de clés augmentera le temps d'exécution pour la découverte de clés et la liage d'entités, puisque davantage de valeurs devront être comparées. Ce problème est néanmoins commun à l'approche classique et à celle qui applique le transfert de clés, de sorte qu'il n'a pas d'incidence directe sur la présente étude, mais il pourrait améliorer les résultats des approches fondées sur les clés.

5.2. Graphes avec un vocabulaire homogénéisé

Dans cette seconde évaluation expérimentale, nous utilisons plusieurs graphes mis à disposition par Symeonidou et al. [17]. Ils sont extraits de DBpedia et de YAGO et leurs noms de propriétés sont homogénéisés (c'est-à-dire avec les mêmes IRI) à l'aide d'une procédure semi-automatique d'alignement d'ontologies. Cette évaluation se concentre principalement sur la comparaison qualitative de notre approche de liage de données basée sur le transfert de clés avec l'approche classique qui repose sur l'intersection entre les clés découvertes dans les deux graphes. Nous fournissons également des résultats quantitatifs en termes de caractéristiques des clés et de temps d'exécution. Pour ce faire, nous nous appuierons sur des ensembles de données mis à disposition par [17] et nous comparerons la qualité de l'alignement final des entités entre les deux approches. Comme dans l'expérience précédente, nous avons fait varier le taux d'exception relatif entre quatre valeurs (c'est-à-dire $[0\ \%, 0{,}5\ \%, 2\ \%, 5\ \%]$) comme dans l'exemple précédent, nous avons vu que nous atteignons un plateau dès que le taux d'exception est supérieur à un petit pourcentage. Le déroulement de cette expérience suit pratiquement les mêmes étapes que la première expérience (section 5.1) à l'exception des étapes 1 et 2 qui ne sont plus nécessaires.

Table 5.6. Caractéristiques des 8 classes dans DBpedia et YAGO.

		# Entités	# Propriétés	# Triplets
Actor	DB	5 807	16	47 303
	YAGO	108 424	16	514 790
Album	DB	85 069	5	297 072
	YAGO	137 167	5	381 119
Book	DB	29 849	7	123 686
	YAGO	41 855	7	92 516
City	DB	19 229	17	652 042
	YAGO	83 560	17	1 100 896
Film	DB	82 124	9	684 800
	YAGO	123 912	9	533 542
Moutain	DB	16 401	5	32 981
	YAGO	67 721	6	116 781
Museum	DB	1 826	7	7 970
	YAGO	21 152	7	81 671
University	DB	10 352	9	120 919
	YAGO	23 351	9	131 812

Après avoir trouvé les clés validées dans les deux graphes, nous pouvons utiliser ces clés pour trouver les entités qui partagent au moins une valeur pour chaque propriété de la clé. Cependant, nous pouvons trouver plusieurs liens d'identité pour la même

entité en utilisant plusieurs clés. Comme nous savons que dans les ensembles de données DBpedia et YAGO, l'hypothèse du nom unique est presque vérifiée, nous avons défini une stratégie qui évite les résultats de liens $n-m$ qui réduisent la précision de l'approche. Cette stratégie consiste à trier les clés en fonction de leur support (plus le support est élevé, meilleure est la clé) et de leur taux d'exception relatif (plus le taux d'exception relatif est faible, meilleure est la clé) et à appliquer un couplage de données basé sur la clé en suivant l'ordre obtenu. Un exemple de cette stratégie serait l'ordre $[k_1, k_2]$. Par conséquent, nous utiliserons k_1 pour aligner e avec e', et si k_2 propose un alignement entre e et e'', il ne sera pas conservé car e a déjà été aligné par une meilleure clé.

5.2.1. *Jeux de données*

Nous avons utilisé le jeu de données `dbpedia-yago` fourni dans [17] qui est mis à la disposition du public[(12)]. Cet ensemble de données utilise un vocabulaire homogénéisé et est associé à un gold standard de tous les liens d'identité à trouver. Dans ce jeu de données, nous avons utilisé 8 classes qui diffèrent par leurs caractéristiques pouvant être plus ou moins volumineuses, atteignant plus d'un million de triplets pour la classe `City`. Les caractéristiques de chaque classe sont décrites dans le tableau 5.6, avec le nombre d'entités, de propriétés et de triplets.

Il est surprenant de constater que le nombre de triplets n'est pas corrélé au nombre d'entités, ni au nombre de propriétés (par exemple, `Film` vs `City`).

5.2.2. *Temps d'exécution*

Dans le tableau 5.7, nous comparons la durée d'exécution d'une approche de transfert de clés obtenue en considérant deux scénarios (lorsque DBpedia est la source et YAGO est la cible, et vice versa) avec la durée d'exécution d'une approche classique de découverte de clés où les clés sont extraites dans les deux graphes et ensuite combinées en conservant l'intersection (ligne INTER dans le tableau 5.7). Nous pouvons observer que le temps d'exécution de l'approche par transfert de clés est plus court que celui de l'approche classique pour les 8 classes différentes. Par exemple, nous pouvons constater une diminution de 73 % du temps d'exécution pour la classe `Film`, lorsque nous considérons DBpedia comme le graphe source.

Comme dans l'expérience précédente, nous voulons savoir comment choisir le graphe source en fonction des caractéristiques du graphe. Cependant, comme les graphes ont le même nombre de propriétés, notre goulot d'étranglement (c'est-à-dire la découverte de la clé) n'est plus pertinent et nous devons essayer de trouver de nouvelles idées. En examinant les tableaux 5.6 et 5.7, nous ne pouvons pas déterminer directement quel point de départ permettrait d'optimiser le temps d'exécution total. Par exemple, pour `Actor`, notre approche est plus rapide lorsqu'elle part d'un graphe de connaissances plus petit (c'est-à-dire moins de triplets et moins d'entités), mais

[(12)] à `https://github.com/lgalarra/vickey`

TABLE 5.7. Temps d'exécution pour la découverte de la clé source, la vérification de la source et la vérification de la clé transférée pour chaque scénario (c'est-à-dire à partir de DBpedia, de Yago et de l'intersection).

Étape		S-K Découverte	S-K Vérification	T-K Vérification	Total
Actor	DB	3 s	58 s	2 m 08 s	**3 m 10 s**
	YAGO	4 m 12 s	4 m 02 s	1 m 54 s	9 m 50 s
	INTER	4 m 15 s	4 m 41 s	-	8 m 57 s
Album	DB	1 m 56 s	13 s	14 s	2 m 23 s
	YAGO	12 s	25 s	29 s	**1 m 06 s**
	INTER	2 m 8 s	38 s	-	2 m 47 s
Book	DB	4 s	22 s	17 s	**43 s**
	YAGO	6 s	36 s	23 s	1 m 6 s
	INTER	10 s	58 s	-	1 m 8 s
City	DB	23 s	15 m 15 s	6 m 20 s	22 m 8 s
	YAGO	2 m 4 s	3 m 34 s	3 m 16 s	**8 m 55 s**
	INTER	2 m 27 s	18 m 49 s	-	21 m 17 s
Film	DB	6 m 25 s	1 m 21 s	1 m 04 s	**8 m 51 s**
	YAGO	3 m 05 s	20 m 03 s	49 s	23 m 57 s
	INTER	9 m 30 s	21 m 24 s	-	30 m 54 s
Moun.	DB	1 s	15 s	16 s	**32 s**
	YAGO	4 s	21 s	22 s	47 s
	INTER	5 s	36 s	-	41 s
Mus.	DB	1 s	23 s	35 s	59 s
	YAGO	2 s	26 s	20 s	**48 s**
	INTER	3 s	50 s	-	52 s
Univ.	DB	9 s	27 s	33 s	**1 m 09 s**
	YAGO	12 s	37 s	29 s	1 m 18 s
	INTER	21 s	1 m 04 s	-	1 m 25 s

pour d'autres classes telles que `Album` ou `City`, c'est l'inverse. Nous présentons donc dans le tableau 5.8 une comparaison entre le nombre de clés trouvées par l'algorithme de découverte de clés et les clés transférées (T-K). Moins les clés sont nombreuses et plus elles sont précises (c'est-à-dire plus longues), plus l'étape est rapide. Cependant, avant de lancer le framework sur un graphe spécifique, nous n'avons aucune idée du nombre de clés et de leur précision qui seront trouvées par l'algorithme. Ainsi, outre le nombre de propriétés et le nombre d'instances dans chaque graphe, nous n'avons aucune idée du point de départ qui offrira le meilleur temps d'exécution total.

TABLE 5.8. Nombre et taille moyenne des clés sources et des clés transférées par ensembles de données et scénarios (c'est-à-dire à partir de DBpedia, de YAGO et d'Intersection).

Étape		#Clés	Taille Clés	#T-K	Taille T-K
Actor	DB	93	2,24	50,7	2,3
	YAGO	183	3,08	156,7	3,0
	INTER	25	2,08	25,0	2,1
Album	DB	1	2,0	1,0	2,0
	YAGO	3	2,0	2,0	2,0
	INTER	1	2,0	1,0	2,0
Book	DB	5	2,80	5,0	2,8
	YAGO	8	2,375	3,0	2,2
	INTER	1	2,0	1,0	2,0
City	DB	178	2,88	159,0	2,9
	YAGO	114	2,5	81,3	2,6
	INTER	40	2,5	40,0	2,5
Film	DB	14	3,86	12,3	3,8
	YAGO	9	2,89	5,7	3,7
	INTER	0	-	0	-
Moun.	DB	14	2,0	3,0	2,0
	YAGO	10	2,0	4,3	2,0
	INTER	3	2,0	3	2,0
Mus.	DB	14	2,14	11,3	2,1
	YAGO	10	2,10	9,3	2,1
	INTER	8	2,0	8,0	2,0
Univ.	DB	17	2,18	14,3	2,1
	YAGO	19	2,26	16,3	2,3
	INTER	12	2,08	12,0	2,1

5.2.3. *Résultats de la mise en relation des données*

Dans le tableau 5.9, nous présentons les résultats du liage de données lorsque les clés sont utilisées pour relier les instances de DBpedia et de YAGO pour chaque ensemble de données. Ces résultats sont exprimés en termes de rappel, de précision et de score F1. En comparant les résultats obtenus avec des clés obtenues par l'approche basée sur le transfert de clés et ceux obtenus par l'approche classique de découverte de clés, il est clair que l'approche classique de l'intersection est toujours pire ou équivalente au côté le plus mauvais de l'approche basée sur le transfert de clés. En outre, le gain de performance peut être particulièrement important : pour l'ensemble

de données `Album`, nous passons de 0, 03 % de F1 à 57, 5 %. En effet, l'approche par intersection ne peut tout au plus capturer qu'un sous-ensemble des clés données dans les autres graphes DBpedia et YAGO. Nous pouvons également observer que lorsque un minimum d'exceptions sont tolérées (c'est-à-dire 0, 5 %, 2 % ou 5 %), les meilleurs résultats sont toujours obtenus lorsque YAGO est le graphe source et DBpedia est le graphe cible.

TABLE 5.9. La qualité du liage des données en fonction du pourcentage d'exception maximum variant avec les valeurs {0 %, 0,5 %, 2 %, 5 %}, *D* et *Y* se référant respectivement à DBpedia et YAGO comme graphique de départ, et *I* se référant à l'approche classique de l'intersection).

ex^R_{max}		0			0,5 %			2 %			5 %		
Metric		R	P	F1	R	P	F1	R	P	F1	R	P	F1
Actor	D	23,5	99,8	38,0	25,4	99,6	40,5	27,5	98,0	42,9	27,5	97,8	42,8
	Y	15,8	99,6	27,3	52,6	99,8	68,9	52,6	99,8	**68,9**	52,6	99,8	**68,9**
	I	14,4	99,6	25,2	14,4	99,6	25,2	14,4	99,6	25,2	14,4	99,6	25,2
Album	D	0,01	75,0	0,03	0,01	75,0	0,03	0,01	75,0	0,03	0,01	75,0	0,03
	Y	0,01	75,0	0,03	24,0	96,7	38,5	41,2	95,1	**57,5**	41,2	95,1	**57,5**
	I	0,01	75,0	0,03	0,01	75,0	0,03	0,01	75,0	0,03	0,01	75,0	0,03
Book	D	2,2	84,6	4,3	3,5	94,9	6,7	3,5	94,9	6,7	3,5	94,9	6,7
	Y	2,9	95,4	5,7	31,4	96,9	47,4	44,2	96,3	**60,6**	44,2	96,3	**60,6**
	I	75,0	95,4	5,7	75,0	95,4	5,7	75,0	95,4	5,7	75,0	95,4	5,7
City	D	60,5	98,5	74,7	60,9	98,5	75,3	60,9	98,5	75,3	61,0	98,4	75,3
	Y	31,3	99,6	47,6	65,7	96,5	78,2	65,9	96,3	**78,3**	66,2	95,8	**78,3**
	I	0	100	0	0	100	0	0	100	0	0	100	0
Film	D	3,2	98,5	6,2	4,2	98,1	8,1	4,2	98,1	8,1	4,2	98,1	8,1
	Y	0,72	97,9	1,42	16,9	98,4	28,9	30,5	97,3	**46,5**	30,5	97,3	**46,5**
	I	0	100	0	0	100	0	0	100	0	0	100	0
Moun.	D	0,22	100	0,44	0,22	100	0,44	0,22	100	0,44	0,22	100	0,44
	Y	0,22	100	0,44	1,57	99,5	3,1	1,57	99,5	**3,1**	1,57	99,5	**3,1**
	I	0,22	100	0,44	0,22	100	0,44	0,22	100	0,44	0,22	100	0,44
Mus.	D	3,5	70,2	6,7	3,9	67,9	7,3	3,9	67,9	7,3	3,9	67,9	7,3
	Y	1,3	71,0	2,6	1,3	71,0	2,6	4,1	66,5	**7,7**	4,1	66,5	**7,7**
	I	1,3	71,0	2,6	1,3	71,0	2,6	1,3	71,0	2,6	1,3	71,0	2,6
Univ.	D	8,9	99,1	16,3	9,0	99,0	16,4	9,0	98,9	16,4	9,0	98,9	16,4
	Y	9,0	99,1	16,3	10,1	99,1	18,4	12,4	98,9	**22,1**	12,4	98,9	**22,1**
	I	8,9	99,1	16,3	8,9	99,1	16,3	8,9	99,1	16,3	8,9	99,1	16,3

6. Conclusion et perspectives

Dans cet article, nous avons proposé une stratégie de transfert de clés qui évite la nécessité d'une découverte systématique des clés dans tous les graphes considérés lors de la mise en relation des données. Nous avons montré que cette approche peut réduire le temps d'exécution jusqu'à 22 fois lorsque le bon graphe est sélectionné comme point de départ. Dans le cas d'un vocabulaire hétérogène, nous avons montré dans nos expériences sur DBpedia et Wikidata que le graphe avec le plus petit nombre de relations représente un meilleur point de départ. Cependant, le choix du graphe à utiliser dans le cas d'un vocabulaire homogène n'est pas clair. Nous avons également montré l'importance d'évaluer les clés réécrites dans le graphe cible afin d'éviter l'utilisation de clés non réécrites. Enfin, nous avons observé que lorsque peu d'exceptions sont tolérées (environ 0,5 %), la qualité de l'interconnexion des données est considérablement améliorée par rapport aux approches classiques de découverte de clés.

Dans des travaux futurs, nous avons l'intention de travailler sur l'évaluation comparative des différentes approches de découverte de clés en cas de transfert de clés et de les comparer en termes de performances et d'analyser si les clés obtenues conservent leurs caractéristiques (par exemple, la monotonicité, la minimalité). Nous prévoyons également d'étudier la création d'un outil de découverte de clés qui prenne en compte les paramètres d'exception relatifs et nous permette d'avoir toutes les clés respectant le taux d'exception défini. Enfin, nous voulons étudier si ces approches basées sur le transfert de clés peuvent être exploitées pour développer une méthode hybride de liage de données en explorant la possibilité d'utiliser des clés pour générer automatiquement un premier ensemble réduit de liens d'identité. Ces liens peuvent ensuite être utilisés pour initialiser les techniques de mise en relation des données à l'aide de graph embeddings [15], qui requièrent un ensemble prédéfini de liens entre les deux graphes.

7. Remerciements

L'auteur a bénéficié d'un soutien important de la communauté.

Bibliographie

[1] N. Abbas, A. Bazin, J. David & A. Napoli, « Discovery of link keys in resource description framework datasets based on pattern structures », *Int. J. Approx. Reason.* **161** (2023), article no. 108978.

[2] M. Al-Bakri, M. Atencia, S. Lalande & M. Rousset, « Inferring Same-As Facts from Linked Data: An Iterative Import-by-Query Approach », in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, 2015, p. 9-15.

[3] A. Assi, H. Mcheick & W. Dhifli, « Data linking over RDF knowledge graphs: A survey », *Concurrency and Computation: Practice and Experience* **32** (2020), n° 19, article no. e5746.

[4] M. Atencia, M. Chein, M. Croitoru, J. David, M. Leclère, N. Pernelle, F. Saïs, F. Scharffe & D. Symeonidou, « Defining Key Semantics for the RDF Datasets: Experiments and Evaluations », in *Graph-Based Representation and Reasoning – 21st International Conference on Conceptual Structures, ICCS 2014, Iaşi, Romania, July 27-30, 2014, Proceedings*, 2014, p. 65-78.

[5] M. Atencia, J. David & J. Euzenat, « On the relation between keys and link keys for data interlinking », *Semantic Web* **12** (2021), n° 4, p. 547-567.

[6] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis & K. Stefanidis, « An Overview of End-to-End Entity Resolution for Big Data », *ACM Comput. Surv.* **53** (2020), n° 6, article no. 127 (42 pages).

[7] J. Devlin, M.-W. Chang, K. Lee & K. N. Toutanova, « BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding », `https://arxiv.org/abs/1810.04805`, 2018.

[8] N. Fanourakis, V. Efthymiou, D. Kotzinos & V. Christophides, « Knowledge Graph Embedding Methods for Entity Alignment: An Experimental Review », `https://arxiv.org/abs/2203.09280`, 2022.

[9] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute & V. Raghavendra, « Deep Learning for Entity Matching: A Design Space Exploration », in *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, 2018, p. 19–34.

[10] A.-C. N. Ngomo & S. Auer, « LIMES: a time-efficient approach for large-scale link discovery on the web of data », in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, AAAI Press, 2011, p. 2312–2317.

[11] N. Pernelle, F. Saïs & D. Symeonidou, « An automatic key discovery approach for data linking », *Journal of Web Semantics* **23** (2013), p. 16-30.

[12] F. Saïs, N. Pernelle & M.-C. Rousset, « Combining a logical and a numerical method for data reconciliation », in *Journal on Data Semantics XII*, Springer, 2009, p. 66-94.

[13] T. Soru, E. Marx & A.-C. Ngonga Ngomo, « ROCKER: A Refinement Operator for Key Discovery », in *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, International World Wide Web Conferences Steering Committee, 2015, p. 1025–1033.

[14] T. Soulard, « Knowledge-based Entity Linking in Heterogeneous Knowledge Graphs at Web-Scale », Tech. report, Université Paris Saclay, 2022.

[15] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami & C. Li, « A benchmarking study of embedding-based entity alignment for knowledge graphs », *Proceedings of the VLDB Endowment* **13** (2020), n° 12, p. 2326-2340.

[16] D. Symeonidou, V. Armant, N. Pernelle & F. Saïs, « Sakey: Scalable almost key discovery in RDF data », in *The Semantic Web – ISWC 2014*, Springer, 2014, p. 33-49.

[17] D. Symeonidou, L. Galárraga, N. Pernelle, F. Saïs & F. Suchanek, « Vickey: Mining conditional keys on knowledge bases », in *The Semantic Web — ISWC 2017*, Springer, 2017, p. 661-677.

[18] B. Zhu, R. Wang, J. Wang, F. Shao & K. Wang, « A survey: knowledge graph entity alignment research based on graph embedding », *Artif. Intell. Rev.* **57** (2024), n° 9, article no. 229.

Abstract. — Data linking in knowledge graphs is a crucial and long-standing problem; it involves determining identity links between the descriptions of entities in two graphs designating the same real-world entity (*e.g.* the same person, the same book, the same protein). Keys, which are subsets of properties that identify each instance in a graph, are important elements in the discovery of these identity links. The traditional approach of key-based data linking is to discover a set of keys in each graph, and then apply a fusion procedure to the sets of keys obtained. However, this approach can lead to a reduction in the number of valid keys in both graphs and can sometimes be very computationally expensive. In this work, we propose a new data linking approach based on the transfer of keys discovered on a source graph to a target graph involved in the data linking task. This transfer is based on an alignment of properties known a priori and on the computation of metrics to validate the quality of the keys transferred to the target graph. We have carried out experiments on several datasets extracted from the web of data (DBpedia, Wikidata and YAGO) in order to evaluate the quality of the data linking and the gain in computing time obtained thanks to the transfer of keys.

Keywords. — Data linking, key discovery, knowledge graphs, key transfer, Explainability.